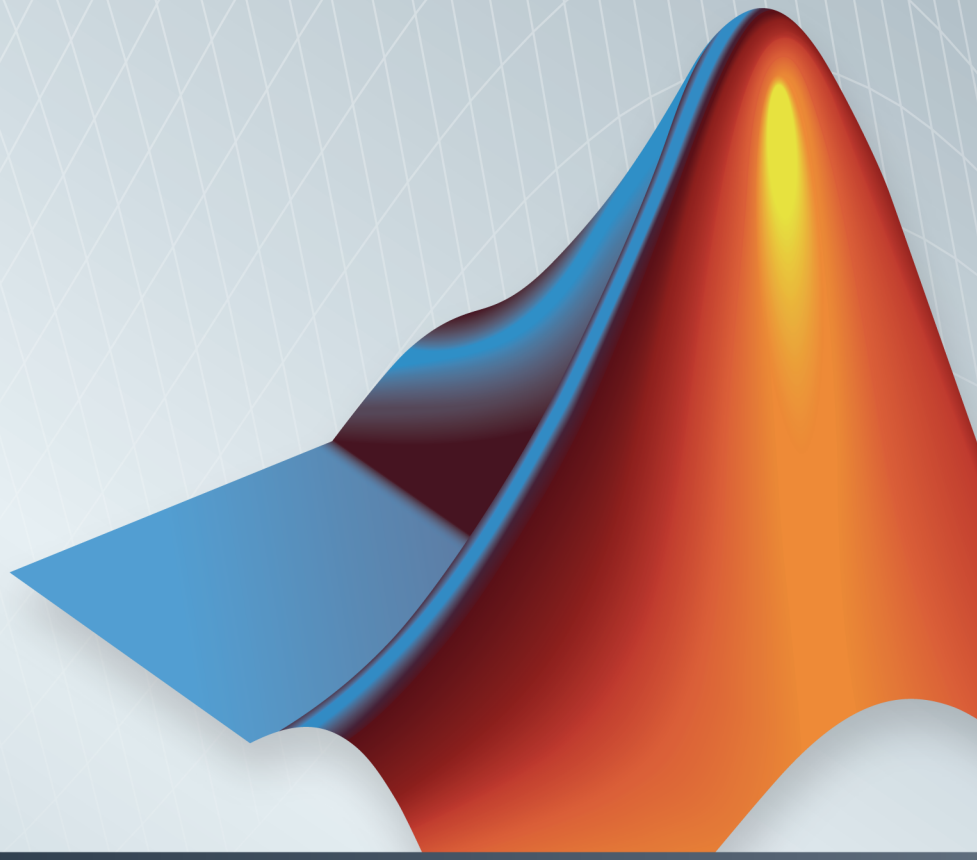
LTE System Toolbox™

Reference

R2014b



MATLAB®

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

LTE System Toolbox™ Reference

Revision History

September 2013	Online only	Revised for Version 1.0 (Release 2013b)
March 2014	Online only	Revised for Version 1.1 (Release 2014a)
October 2014	Online only	Revised for Version 1.2 (Release 2014b)

Contents

A

Functions — Alphabetical List

addlteobsolete

Add obsolete LTE Toolbox interface to search path

Syntax

```
addlteobsolete
```

Description

`addlteobsolete` prefixes the directory `matlabroot\toolbox\lte\lteobsolete` to the MATLAB® path, which enables the obsolete LTE Toolbox interface. This interface is provided for backwards compatibility and will be removed in a later release.

Note: You can undo your changes by calling the `rmlteobsolete` function.

Examples

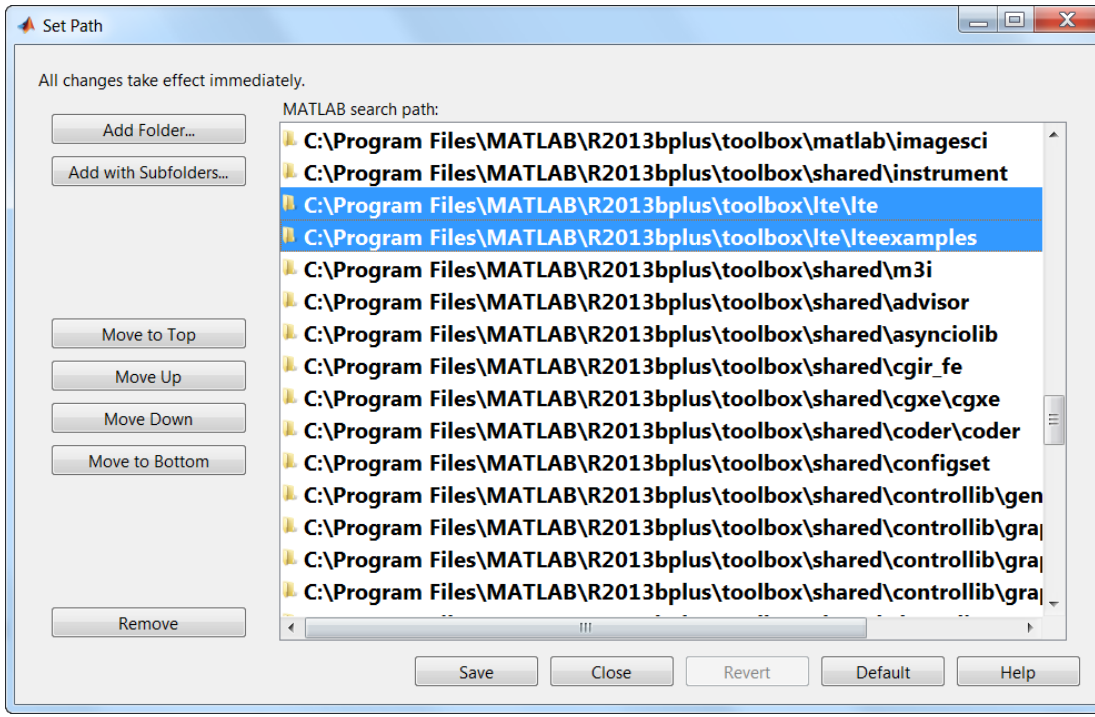
Add Obsolete LTE Toolbox Interface to Search Path

Add the directory associated with the obsolete LTE Toolbox interface to the MATLAB path.

View the current MATLAB path by calling the `pathtool` function.

```
pathtool
```

The Set Path dialog box appears. Scroll down through the listings to find the LTE System Toolbox™ product directories, as shown in the following figure.



If you see the listing `matlabroot\toolbox\lte\lteobsolete`, you do not need to run the `addlteobsolete` function. Exit this example. Otherwise, click **Close** to close the Set Path dialog box.

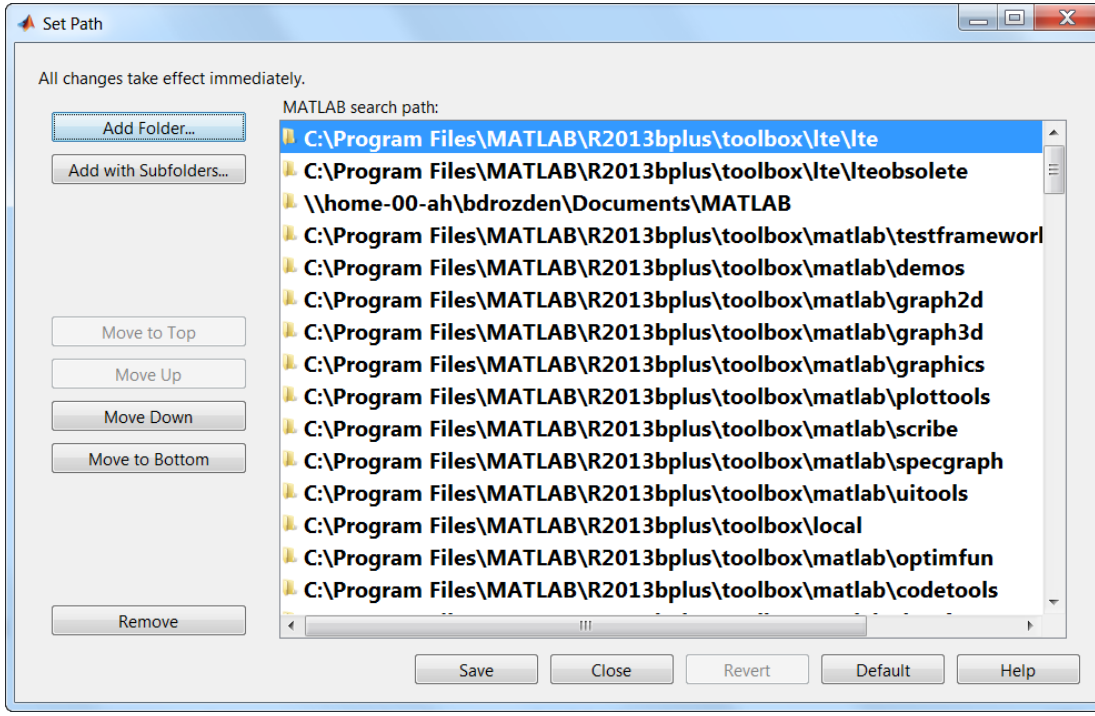
Add the obsolete LTE Toolbox interface to the path.

```
addlteobsolete
```

To confirm the changes, call the `pathtool` function again.

```
pathtool
```

The Set Path dialog box appears. Find the LTE System Toolbox product directories at the top of the list, as shown in the following figure.



The 2nd listing, `matlabroot\toolbox\lte\lteobsolete`, shows that you have added the obsolete LTE Toolbox interface to the search path. Click **Close** again to close the Set Path dialog box.

See Also

`rmlteobsolete`

lteACKDecode

HARQ-ACK channel decoding

Syntax

```
out = lteACKDecode(chs,in)
```

Description

`out = lteACKDecode(chs,in)` performs the block decoding on soft input data, in, assumed to be encoded using the procedure defined for HARQ-ACK in section 5.2.2.6 of [1] for given PUSCH channel transmission configuration chs. The decoded output, out, is a vector of length `OACK`, the number of uncoded HARQ-ACK bits transmitted.

Note: If `NBundled` is 0, TDD ACK-NACK descrambling is disabled.

Multiple codewords can be parameterized by two different forms of the chs structure. Each codeword can be defined by separate elements of a 1-by-2 structure array, or the codeword parameters can be combined together in the fields of a single scalar, or 1-by-1, structure. Any scalar field values apply to both codewords and a scalar `NLayers` is the total number. See “UL-SCH Parameterization” for further details.

The block decoding is performed separately on each soft input data using a maximum likelihood (ML) approach, assuming that in has been demodulated and equalized to best restore the originally transmitted values.

The HARQ-ACK decoder performs different type of block decoding depending upon the number of uncoded HARQ-ACK bits to be recovered (`OACK`). For `OACK` less than 3 bits, the decoder assumes the bits are encoded using the procedure defined in section 5.2.2.6 of [1].

For decoding between 3 and 11 HARQ-ACK bits, the decoder assumes the bits are block encoded using the procedure defined in section 5.2.2.6.4 of [1]. For greater than 11 bits, the decoder performs the inverse procedure described in section 5.2.2.6.5 of [1].

Examples

Decode HARQ-ACK Channel

Decode a block of 3 coded HARQ-ACK information bits.

```
in = [1;0;1];
chs = struct('Modulation','16QAM','QdACK',2,'OACK',length(in));
encodedBits = lteACKEncode(chs,in);
encodedBits(encodedBits == 0) = -1;
lteACKDecode(chs,encodedBits)
```

```
     1
     0
     1
```

Input Arguments

chs — PUSCH-specific channel transmission configuration

scalar structure | structure array

PUSCH-specific channel transmission configuration, specified as a structure or a structure array, which contains the following parameter fields.

Parameter Field	Required or Optional	Values	Description
Modulation	Required	'QPSK', '16QAM', '64QAM', cell array of strings	Modulation type, specified as a string or cell array of strings. If 2 blocks, each cell is associated with a transport block.
OACK	Optional	nonnegative scalar integer, 0 (default)	Number of uncoded HARQ-ACK bits. The HARQ-ACK decoder performs different type of block decoding depending upon the number of uncoded HARQ-ACK bits to be recovered (OACK). For OACK less than 3 bits, the decoder assumes the bits are encoded using the procedure defined in section 5.2.2.6 of [1]. For decoding between 3 and 11 HARQ-ACK bits, the decoder

Parameter Field	Required or Optional	Values	Description
			assumes the bits are block encoded using the procedure defined in section 5.2.2.6.4 of [1]. For greater than 11 bits, the decoder performs the inverse procedure described in section 5.2.2.6.5 of [1].
`NLayers`	Optional	1 (default), 2, 3, 4	Number of transmission layers
`NBundled`	Optional	0 (default), 1, …, 9	TDD HARQ-ACK bundling scrambling sequence index. When set to 0, the function disables the TDD HARQ-ACK bundling scrambling. Therefore, it is off by default.

`in` — Soft input data

numeric vector

Soft input data, specified as a numeric vector. The input data is assumed to be encoded using the procedure defined for HARQ-ACK in section 5.2.2.6 of [1].

Output Arguments

`out` — Decoded HARQ-ACK channel

numeric column vector

Decoded HARQ-ACK channel output, returned as an `OACK`-by-1 column vector.

Data Types: `double`

References

[1] 3GPP TS 36.212. “Multiplexing and channel coding.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteACKEncode` | `lteCQIDecode` | `lteRIDecode` | `lteUCIDecode` | `lteULSCHDecode` | `lteULSCHDeinterleave`

lteACKEncode

HARQ-ACK channel encoding

Syntax

```
out = lteACKEncode(chs,in)
```

Description

`out = lteACKEncode(chs,in)` returns the coded HARQ-ACK information bits after performing block coding defined for HARQ-ACK in section 5.2.2.6 of [1]. The input argument, in, is a vector or cell array containing up to 20 HARQ-ACK information bits. The output argument, out, is the encoded bits in the same form. out can be a cell array if the PUSCH-specific parameter structure, chs, defines multiple codewords.

Multiple codewords can be parameterized by two different forms of the chs structure. Each codeword can be defined by separate elements of a 1-by-2 structure array, or the codeword parameters can be combined together in the fields of a single scalar, or 1-by-1, structure. Any scalar field values apply to both codewords and a scalar `NLayers` is the total number. See “UL-SCH Parameterization” for further details.

Since the HARQ-ACK bits are carried on all defined codewords, a single input results in a cell array of encoded outputs if multiple codewords are parameterized. This allows for easy integration with the other toolbox functions.

The HARQ-ACK coder performs different types of block coding depending upon the number of HARQ-ACK bits in vector in. If in consists of one element, it uses table 5.2.2.6-1 of [1]. If in consists of two elements, it uses table 5.2.2.6-2 of [1] for encoding. The placeholder bits, x and y in the referenced tables, are represented by –1 and –2, respectively.

Similarly, for between 3 and 11 bits, the HARQ-ACK encoding is performed as described in section 5.2.2.6.4. For bits greater than 11, the encoding is performed as described in section 5.2.2.6.5 of [1].

Examples

Encode HARQ-ACK Channel

Encode a HARQ-ACK information bit for one codeword with 16QAM modulation.

```
ackbit = 1;
out1 = lteACKEncode(struct('Modulation','16QAM','QdACK',1),ackbit)
```

```
     1
    -2
    -1
    -1
```

Encode a HARQ-ACK information bit for two codewords with differing modulation schemes.

```
ackbit = 1;
chs = struct('Modulation',{{'16QAM' '64QAM'}},'QdACK',1,'NLayers',2);
out2 = lteACKEncode(chs,ackbit)
```

```
    [4x1 int8]    [6x1 int8]
```

Input Arguments

`chs` — PUSCH-specific channel transmission configuration

scalar structure | structure array

PUSCH-specific channel transmission configuration, specified as a structure or a structure array, which contains the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`QdACK`	Required	nonnegative scalar integer	Number of coded HARQ-ACK symbols for ACK or NACK (*Q'_ACK*)
`Modulation`	Required	`'QPSK'`, `'16QAM'`, `'64QAM'`, cell array of strings	Modulation type, specified as a string or cell array of strings. If 2 blocks, each cell is associated with a transport block.
`NLayers`	Optional	1 (default), 2, 3, 4	Number of transmission layers, total or per codeword

Parameter Field	Required or Optional	Values	Description
`NBundled`	Optional	0 (default), 1, …, 9	TDD HARQ-ACK bundling scrambling sequence index. When set to 0, the function disables the TDD HARQ-ACK bundling scrambling. Therefore, it is off by default.

`in` — HARQ-ACK information bits

logical vector of length 1 to 20 | cell array of logical vectors

HARQ-ACK information bits, specified as a logical vector or a cell array of logical vectors. Each vector can have a length of up to 20 information bits.

Data Types: `logical` | `double` | `cell`

Output Arguments

`out` — Encoded HARQ-ACK information bits

integer column vector | cell array of integer column vectors

Encoded HARQ-ACK information bits, returned as either an integer column vector or a cell array of integer column vectors. The encoded bits are in the same form as the input bits. Therefore, if the PUSCH-specific parameter structure, chs, defines multiple codewords, `out` is a cell array.

Data Types: `int8` | `cell`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteACKDecode` | `lteCQIEncode` | `lteRIEncode` | `lteUCIEncode` | `lteULSCH` | `lteULSCHInterleave`

lteBCH

Broadcast channel

Syntax

```
codeblk = lteBCH(enb,trblk)
```

Description

`codeblk = lteBCH(enb,trblk)` returns a vector of BCH transport channel coded bits.

The input argument, trblk, is a vector of 24 bits. It represents the information bits delivered to the broadcast channel (BCH) every 40 ms. The coded output vector, codeblk, contains 1920 bits for normal cyclic prefix or 1728 bits for extended cyclic prefix.

Given the small input size, which is 24, compared with the large output size, which is 1920 or 1728, the rate matching internal to the coding results in a large number of repetitions of the coded block. This repetition is deliberate so that part of a received block can be successfully decoded in isolation. For example, the receiver can recover the BCH bits from the reception of just one frame, ¼ of the transmitted block, rather than waiting 40 ms, or 4 frames, for the full block to be received.

Examples

Encode BCH Information Bits

Generate a BCH-coded block of length 1920.

```
enb = struct('CellRefP',1,'CyclicPrefix','Normal');
bchCoded = lteBCH(enb,ones(24,1));
size(bchCoded)
```

1920 1

Input Arguments

enb — eNodeB cell-wide settings

scalar structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
CellRefP	Required	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
CyclicPrefix	Optional	'Normal' (default), 'Extended'	Cyclic prefix length

trblk — Transport block

numeric vector

Transport block, specified as a numeric vector of length 24 bits. This argument represents the transport block delivered to the BCH every 40 ms.

Output Arguments

codeblk — BCH transport channel coded bits

numeric column vector

BCH transport channel coded bits, returned as an integer column vector.

Data Types: int8

See Also

lteBCHDecode | lteMIB | ltePBCH

lteBCHDecode

Broadcast channel decoding

Syntax

```
[trblk,cellrefp] = lteBCHDecode(enb,softbits)
```

Description

`[trblk,cellrefp] = lteBCHDecode(enb,softbits)` returns a vector, trblk, of the decoded information bits (24 bits). cellrefp is the number of cell-specific reference signal antenna ports detected (1,2,4) in the CRC mask for given input, softbits, and the structure, enb.

If the cellrefp value is 0, a CRC error has been detected.

The transport block size, 24, is relatively small when compared to the number of coded bits sent in the BCH transmission, 1920 or 1728. For this reason, the rate matching internal to the BCH coding results in a large number of repetitions of the coded block. Therefore, the decoder can successfully decode blocks whose lengths are much shorter than the length of the full coded block. Consequently, this decoder allows the input argument softbits to be of any length.

Examples

Decode BCH-Encoded Block

Perform BCH encoding of one transport block. Then, perform BCH decoding of part of the encoded block, one quarter the length.

```
enb = lteRMCDL('R.4');
bchCoded = lteBCH(enb,ones(24,1));
out = bchCoded(1:length(bchCoded)/4);
[bchDecoded,cellRefP] = lteBCHDecode(enb,out);
bchDecoded(1:10)
```

```
     1
     1
```

```
     1
     1
     1
     1
     1
     1
     1
     1
```

In a practical system, this approach would be used to attempt BCH decoding on the one quarter part of the encoded block that is transmitted in the first subframe of each frame.

Input Arguments

`enb` — eNodeB cell-wide settings

scalar structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length

`softbits` — Soft bits to decode

numeric vector

Soft bits to decode, specified as a numeric vector. This vector can have any length.

The transport block size, 24, is relatively small when compared to the number of coded bits sent in the BCH transmission, 1920 or 1728. For this reason, the rate matching internal to the BCH coding results in a large number of repetitions of the coded block. Therefore, the decoder can successfully decode blocks whose lengths are much shorter than the length of the full coded block.

Output Arguments

`trblk` — Decoded information bits

integer column vector

Decoded information bits, returned as a 24-by-1 integer column vector.

Data Types: `int8`

`cellrefp` — Number of cell-specific reference signal (CRS) antenna ports

0 | 1 | 2 | 4

Number of cell-specific reference signal (CRS) antenna ports detected, returned as a nonnegative scalar integer. Possible values are 0, 1, 2, and 4. If the value is 0, a CRC error has been detected.

Data Types: `uint32`

See Also

`lteBCH` | `ltePBCHDecode`

lteCFI

Control format indicator block encoding

Syntax

```
cw = lteCFI(enb)
```

Description

`cw = lteCFI(enb)` returns a 32-element vector, cw, that represents the rate 1/16 block encoding of the control format indicator (CFI) value defined in the `CFI` field of the enb structure.

The value for CFI can be 1, 2, or 3. This value indicates the time span, in OFDM symbols, of the DCI PDCCH transmission (the control region) in that downlink subframe. For bandwidths in which `NDLRB` is greater than 10 RB, the span of the DCI in OFDM symbols is the same as the actual CFI value. If `NDLRB` is less than or equal to 10 RB, the span is *CFI*+1 symbols.

Examples

Encode CFI Value

Generate the 32-element vector that represents block encoding of a CFI value of 2.

```
cw = lteCFI(struct('CFI',2));
cw(1:10)
```

```
     1
     0
     1
     1
     0
     1
     1
     0
```

```
1
1
```

Input Arguments

enb — eNodeB cell-wide settings

scalar structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`CFI`	Required	1, 2, 3	Control format indicator (CFI) value The value for CFI can be 1, 2, or 3. This value indicates the time span, in OFDM symbols, of the DCI PDCCH transmission (the control region) in that downlink subframe. For bandwidths in which `NDLRB` is greater than 10 RB, the span of the DCI in OFDM symbols is the same as the actual CFI value. If `NDLRB` is less than or equal to 10 RB, the span is *CFI*+1 symbols.

Output Arguments

cw — CFI codeword

integer column vector

CFI codeword, returned as an integer column vector of length 32. This vector represents the 1/16 block encoding of the CFI value defined in structure enb.

Data Types: `int8`

See Also

`lteCFIDecode` | `ltePCFICH`

lteCFIDecode

Control format indicator block decoding

Syntax

```
cfi = lteCFIDecode(ibits)
```

Description

`cfi = lteCFIDecode(ibits)` performs the block decoding on soft input data ibits, assumed to be encoded using procedure defined in section 5.3.4.1 of [1]. The output, cfi, is a scalar representing the control format indicator (CFI) value resulted after performing block decoding on input data. Strictly speaking, ibits should be a vector 32 bits long, as per encoded cfi. See the `lteCFI` function reference for details. However, this function can take any size segment of encoded data to perform decoding.

The value for CFI can be 1, 2, or 3. This value indicates the time span, in OFDM symbols, of the DCI PDCCH transmission (the control region) in that downlink subframe. For bandwidths in which `NDLRB` is greater than 10 RB, the span of the DCI in OFDM symbols is the same as the actual CFI value. If `NDLRB` is less than or equal to 10 RB, the span is *CFI*+1 symbols.

Examples

Decode CFI Block

Decode a noisy 32-element vector that represents the block encoding of the control format indicator (CFI) value.

```
cw = double(lteCFI(struct('CFI',2)));
noisycw = cw + 0.4*randn(length(cw),1);
cfi = lteCFIDecode(noisycw)
```

2

Input Arguments

`ibits` — Soft input data

numeric vector

Soft input data, specified as a numeric vector of length 32. This input data is assumed to be encoded using the procedure defined in section 5.3.4.1 of [1].

Output Arguments

`cfi` — Control format indicator value

1 | 2 | 3

Control format indicator value, returned as a positive scalar integer. This integer represents the CFI value resulting from performing block decoding on a vector of soft input data, ibits.

The value for CFI can be 1, 2, or 3. This value indicates the time span, in OFDM symbols, of the DCI PDCCH transmission (the control region) in that downlink subframe. For bandwidths in which `NDLRB` is greater than 10 RB, the span of the DCI in OFDM symbols is the same as the actual CFI value. If `NDLRB` is less than or equal to 10 RB, the span is *CFI*+1 symbols.

Data Types: `int32`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteCFI` | `ltePCFICHDecode`

lteCQIDecode

Channel quality information channel decoding

Syntax

```
out = lteCQIDecode(chs,in)
```

Description

`out = lteCQIDecode(chs,in)` performs the decoding on soft input data, in, assumed to be encoded using the procedure defined for channel quality information (CQI) in sections 5.2.2.6 and 5.2.2.6.4 of [1] for given channel transmission configuration, chs. The decoded output, out, is a vector of length `OCQI`, the number of uncoded CQI bits transmitted.

Multiple codewords can be parameterized by two different forms of the chs structure. Each codeword can be defined by separate elements of a 1-by-2 structure array, or the codeword parameters can be combined together in the fields of a single scalar, or 1-by-1, structure. Any scalar field values apply to both codewords and a scalar `NLayers` is the total number. See “UL-SCH Parameterization” for further details.

The block decoding is performed separately on each soft input data using a maximum likelihood (ML) approach, which assumes that in has been demodulated and equalized to best restore the original transmitted values. The length of CQI bits defines the decoding process.

If the number of CQI bits, `OCQI`, is less than or equal to 11, a block decoding is performed to invert the coding procedure defined in Section 5.2.2.6.4 of [1]. If `OCQI` is greater than 11, the CQI bits are recovered by performing rate matching to `OCQI`, tail-biting Viterbi decoding, and 8-bit CRC decoding.

Examples

Decode CQI Channel

Decode encoded CQI bits.

```
cqi = [0;1;0;1;0;1];
enc = lteCQIEncode(struct('Modulation','QPSK','QdCQI',16),cqi);
enc(enc == 0) = -1;
rxCqi = lteCQIDecode(struct('OCQI',6),enc)

     0
     1
     0
     1
     0
     1
```

Input Arguments

`chs` — Channel-specific transmission configuration

scalar structure | structure array

Channel-specific transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`OCQI`	Optional	nonnegative scalar integer, 0 (default)	Number of uncoded channel quality information (CQI) bits
`NLayers`	Optional	1 (default), 2, 3, 4	Number of transmission layers

`in` — Encoded soft input data

numeric vector

Encoded soft input data, specified as a numeric vector.

Output Arguments

`out` — Decoded output

logical column vector

Decoded output, returned as a logical column vector of length `OCQI`.

Data Types: `logical`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`lteACKDecode` | `lteCQIEncode` | `lteRIDecode` | `lteUCIDecode` | `lteULSCHDecode`

lteCQIEncode

Channel quality information channel encoding

Syntax

```
out = lteCQIEncode(chs,in)
```

Description

`out = lteCQIEncode(chs,in)` returns the encoded channel quality information (CQI) bits after performing channel coding defined for CQI in sections 5.2.2.6 and 5.2.2.6.4 of [1]. in should be a vector or cell array containing the CQI bits and out is the encoded bits in the same form. out is also cell array if the PUSCH-specific parameter structure, chs, defines multiple codewords.

Multiple codewords can be parameterized by two different forms of the chs structure. Each codeword can be defined by separate elements of a 1-by-2 structure array, or the codeword parameters can be combined together in the fields of a single scalar, or 1-by-1, structure. Any scalar field values apply to both codewords and a scalar `NLayers` is the total number. See “UL-SCH Parameterization” for further details.

While the CQI information bits are carried on one codeword only, a single input still results in a cell array of encoded outputs if multiple codewords are parameterized. In this case, the `QdCQI` field should contain a 0 in the position of the unused codeword. This allows for easy integration with the other toolbox functions.

The CQI coder uses two different coding schemes depending upon the number of CQI bits to be coded. If the number of CQI bits are less than or equal to 11, the channel coding of the CQI bits is performed according to section 5.2.2.6.4 of [1]. For CQI bits greater than 11, the coding process includes 8-bit CRC attachment, tail-biting convolutional coding and rate matching to the output length deduced from parameters `QdCQI` and `Modulation`.

Examples

Encode CQI Channel

Generate the coded CQI bits for a single codeword.

```
in = [0;1;0;1;0;1];
chs1 = struct('Modulation','16QAM','QdCQI',4);
codedCqi1 = lteCQIEncode(chs1,in)
```

```
    1
    1
    1
    1
    0
    1
    0
    1
    0
    1
    1
    0
    0
    1
    1
    0
```

Generate the coded CQI bits for two codewords with CQI on the second codeword.

```
in = [0;1;0;1;0;1];
chs2 = struct('Modulation',{{'16QAM' '16QAM'}},'QdCQI',[0,4],'NLayers',2);
codedCqi2 = lteCQIEncode(chs2,in)
```

```
    [0x1 int8]    [16x1 int8]
```

Input Arguments

chs — Channel-specific transmission configuration

scalar structure | structure array

Channel-specific transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`QdCQI`	Required	nonnegative scalar integer	Number of coded channel quality information (CQI) symbols (*Q'_CQI*)
`Modulation`	Required	`'QPSK'`, `'16QAM'`, `'64QAM'`, cell array of strings	Modulation type, specified as a string or cell array of strings. If 2 blocks, each cell is associated with a transport block.
`NLayers`	Optional	1 (default), 2, 3, 4	Number of transmission layers

`in` — CQI input bits

numeric vector | cell array of numeric vectors

CQI input bits, specified as a numeric vector or a cell array of numeric vectors.

Output Arguments

`out` — Encoded CQI output bits

integer vector | cell array of integer vectors

Encoded CQI output bits, returned as an integer vector or a cell array of integer vectors. This argument contains the coded CQI bits after performing channel coding.

Data Types: `int8` | `cell`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`lteACKEncode` | `lteCQIDecode` | `lteRIEncode` | `lteUCIEncode` | `lteULSCH`

lteCQISelect

PDSCH channel quality indication calculation

Syntax

```
[cqi,sinrs] = lteCQISelect(enb,chs,hest,noiseest)
```

Description

`[cqi,sinrs] = lteCQISelect(enb,chs,hest,noiseest)` calculates PDSCH CQI (Channel Quality Indication) for a given cell wide `enb`, channel configuration structure `chs`, channel estimate resource array `hest` and receiver noise variance `noiseest`.

The function performs the CQI selection by first obtaining SINR (Signal to Interference and Noise Ratio) estimates for a given configuration from `ltePMISelect`. Then the function performs a lookup between those SINR estimates and the CQI index. The lookup tables are precomputed and stored in this function. CQI selection is conditioned on the rank indicated by `CHS.NLayers`, except for the `'TxDiversity'` transmission scheme which has a rank of 1. On PUCCH, CQI selection corresponds to Report Type 2 (for reporting Mode1-1) or Report Type 4 (for reporting mode 1-0). On PUSCH, the reporting will be Mode 1-2, Mode 3-0, or Mode 3-1.

A CQI Index is a scalar (0, ..., 15), indicating the selected value of the CQI index. The CQI index is defined as per [1]. The highest CQI index is when a single PDSCH transport block with a modulation scheme and transport block size of CQI index, and occupying a group of downlink physical resource blocks termed the CSI reference resource, can be received with a transport block error probability not exceeding 0.1. If a CQI index of 1 does not satisfy this condition then, the returned CQI index is 0. The CQI reference resource is defined in [1], Section 7.2.3. The relationship between CQI indices, modulation scheme, and code rate (from which transport block size is derived) is described in[1], Table 7.2.3-1.

A subband differential CQI offset level is the difference between a subband CQI index and the corresponding wideband CQI index.

A spatial differential CQI offset level is the difference between the wideband CQI index for codeword 0 and the wideband CQI index for codeword 1.

Within the 3GPP standard, CQI offsets are reported as *CQI values*. These values are nonnegative integers corresponding to single CQI offset level or range of CQI offset levels (see [1], Tables 7.2-2 and 7.2.1-2). The CQI offset levels reported here are either the single CQI offset level or the boundary value of the CQI offset level range corresponding to the CQI value reported. For example, for a spatial differential CQI offset level of –6, reported as a spatial differential CQI value of 4, this function report as a spatial differential offset level of –4 by this function (see [1], Table 7.2-2).

For transmission schemes using UE-specific beamforming (`'Port 5'`, `'Port 7-8'`, `'Port 8'`, `'Port7-14'`), the performance depends on the beamforming used. For this the appropriate value of CHS.SINRs90pc field is provided. If this field is not provided, for single antenna ports, the function uses default SINRs90pc values.

Examples

Calculate CQI

Populate an empty resource grid for RMC R.13 with cell-specific reference signal symbols. Filter the signal through a channel and demodulate the signal using OFDM. Use estimates of the channel and noise power spectral density for CQI calculation.

Open an empty resource grid RMC R.13. Populate the resource grid with cell specific reference signal symbols.

```
enb = lteRMCDL('R.13');
reGrid = lteResourceGrid(enb);
reGrid(lteCellRSIndices(enb)) = lteCellRS(enb);
txWaveform = lteOFDMModulate(enb,reGrid);
chcfg.SamplingRate = 15360000;
chcfg.DelayProfile = 'EPA';
chcfg.NRxAnts = 4;
chcfg.DopplerFreq = 5;
chcfg.MIMOCorrelation = 'Low';
chcfg.InitTime = 0;
chcfg.Seed = 1;
```

Filter the signal through a channel and demodulate it.

```
rxWaveform = lteFadingChannel(chcfg,txWaveform);
rxSubframe = lteOFDMDemodulate(enb,rxWaveform);
cec.FreqWindow = 1;
cec.TimeWindow = 31;
```

```
cec.InterpType = 'cubic';
cec.PilotAverage = 'UserDefined';
cec.InterpWinSize = 1;
cec.InterpWindow = 'Centered';
```

```
Warning: Using default value for parameter field InitPhase (Random)
Warning: Using default value for parameter field ModelType (GMEDS)
Warning: Using default value for parameter field NTerms (16)
Warning: Using default value for parameter field NormalizeTxAnts (On)
Warning: Using default value for parameter field NormalizePathGains (On)
```

Estimate corresponding channel including noise spectral density and reference signal subcarriers.

```
[hest, noiseEst] = lteDLChannelEstimate(enb,cec,rxSubframe);
```

Calculate the CQI.

```
cqi = lteCQISelect(enb,enb.PDSCH,hest,noiseEst)
```

```
cqi =

    15
```

Input Arguments

enb — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure. The structure contains the following parameter fields.

Parameter Field	Required or Optional	Values	Description
NDLRB	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)

Parameter Field	Required or Optional	Values	Description
NCellID	Required	Nonnegative scalar integer (0, …, 503)	Physical layer cell identity
CellRefP	Required	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
CyclicPrefi	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
DuplexMode	Optional	`'FDD'` (default), `'TDD'`	Duplex mode
The following parameters apply when `DuplexMode` is set to `'TDD'`.			
TDDConf	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
NSubfra	Required	Nonnegative scalar integer	Subframe number
SSC	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)
The following parameters are apply when chs.Txscheme is set to `'Port7–14'`			
CSIRefP	Required	1, 2, 4, 8	Number of CSI-RS antenna ports
CSIRSCo	Required	scalar integer	CSI-RS configuration index. See table 6.10.5.2-1 in TS 36.211.
CSIRSpe	Optional	`'On'` (default), `'Off'`, `Icsi-rs` (0, ..., 154), `[Tcsi-rs Dcsi-rs]`	CSI-RS subframe configuration
NFrame	Optional	0 (default), Nonnegative scalar integer	Frame number

chs — Channel-specific transmission configuration

structure | structure array

Channel-specific transmission configuration, specified as a structure or structure array. The structure contains the following parameter fields:

Parameter Field	Required or Optional	Values	Description
NLayers	Required	1, 2, 3, 4, 5, 6, 7, 8	Number of transmission layers
CSIMode	Required	'PUCCH 1-0', 'PUCCH 1-1', 'PUSCH 1-2', 'PUSCH 3-0', 'PUSCH 3-1'	CSI reporting mode
TxScheme	Required	'SpatialMux' (default), 'Port0', 'TxDiversity', 'CDD', 'MultiUser', 'Port5', 'Port7-8', 'Port8', 'Port7-14'	Transmission scheme, specified as one of the following options. • 'SpatialMux' — Closed-loop spatial multiplexing. • 'Port0' — Single-antenna port, port 0. • 'TxDiversity' — Transmit diversity scheme. • 'CDD' — Large delay CDD scheme. • 'MultiUser' — Multiuser MIMO scheme. • 'Port5' — Single-antenna port, port 5. • 'Port7-8' — Single-antenna port, port 7 (**NLayers** = 1). Dual layer transmission, ports 7 and 8 (**NLayers** = 2). • 'Port8' — Single-antenna port, port 8. • 'Port7-14' — Up to 8-layer transmission, ports 7–14.
Rho	Optional	0 (default), Scalar	PDSCH resource element power allocation, in dB

Parameter Field	Required or Optional	Values	Description
SINRs90pc	Optional	vector double of 15 values, function handle	A vector of 15 SINR values or a function handle to a function of the form f(enb, chs) which returns a vector of 15 SINR values, one for each CQI index 1, ..., 15. These correspond to the lowest SINR for which the throughput of the PDSCH in the CQI/CSI reference resource, for the given configuration and CQI index, is at least 90%. Default is to internally select SINRs based on configuration given in enb and chs, assuming perfect channel estimation and either MMSE equalization or transmit diversity decoding (as appropriate for the transmission scheme) at the receiver.
The following parameters are required only for `'SpatialMux'`, `'MultiUser'`, `'Port5'`, `'Port7-8'`, `'Port8'`, `'Port7–14'` transmission schemes.			
PMISet	Required	Integer vector (0, ..., 15)	Precoder matrix indication (PMI) set. It can contain either a single value, corresponding to single PMI mode, or multiple values, corresponding to multiple or subband PMI mode. The number of values depends on the CellRefP, transmission layers and TxScheme.
Additionally, one of the following fields must be included			
Ncodewo	Required	1, 2	Number of codewords
Modulat	Required	`'QPSK'`, `'16QAM'`, `'64QAM'`, cell array of strings	Modulation type, specified as a string or cell array of strings. If 2 blocks, each cell is associated with a transport block.

You can specify the number of codewords directly in the `NCodewords` field. Alternatively, if the `Modulation` field is provided, the number of codewords is established from the number of modulation formats. This value lets you establish the correct number of codewords using the channel transmission configuration structure chs as provided to

`ltePDSCH` function on the transmit side. The `NCodewords` field takes precedence if present.

`hest` — Channel estimate

multidimensional array

Channel estimate, specified as a multidimensional array. It has size *K*-by-*L*-by-*NRxAnts*-by-*P* where:

- *K* is the number of sub-carriers.
- *L* is the number of OFDM symbols.
- *NRxAnts* is the number of receive antennas.
- *P* is the number of transmit antennas.

Data Types: `double`
Complex Number Support: Yes

`noiseest` — Receiver noise variance

numeric scalar

Receiver noise variance, specified as a numeric scalar. `noiseest` is an estimate of the received noise power spectral density.

Data Types: `double`

Output Arguments

`cqi` — Channel quality information

column vector

Channel quality information, returned as a column vector containing a channel quality information report. Report contents depend on the CSI reporting mode.

Report Mode	Reporting Contents
Single codeword:	
`'PUCCH 1-0'`	A single wideband CQI index
`'PUSCH 3-0'`	A single wideband CQI index, followed by a subband differential CQI offset level for each subband.

Report Mode	Reporting Contents
Two codewords:	
'PUCCH 1-1'	A single wideband CQI index for codeword 0, followed by a spatial differential CQI offset level for codeword 1.
'PUSCH 1-2'	A single wideband CQI index for codeword 0, followed by a single wideband CQI index for codeword 1.
'PUSCH 3-1'	A single wideband CQI index for codeword 0, followed by a subband differential CQI offset level for each subband for codeword 0, followed by a single wideband CQI index for codeword 1, followed by a subband differential CQI offset level for each subband for codeword 1.

Note: CSI reporting modes, are separated into those that support one or two codewords, as described by the standard. The CQI select function derives these code words from `NCodewords` field or `Modulation` field.

`sinrs` — signal-to-interference plus noise ratios

matrix

Signal-to-interference plus noise ratios, in dB, returned as a matrix. Each column of the matrix represents a single codeword. If subband CQI reporting is configured, the SINR for the wideband CQI is in the first row, followed by the sinrs for the subband CQIs in subsequent rows. This is an optional output.

References

[1] 3GPP TS 36.213. “Physical layer procedures.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`ltePMISelect` | `lteRISelect`

lteCRCDecode

Cyclic redundancy check decoding and removal

Syntax

```
[blk,err] = lteCRCDecode(blkcrc,poly)
[blk,err] = lteCRCDecode(blkcrc,poly,mask)
```

Description

`[blk,err] = lteCRCDecode(blkcrc,poly)` checks the input data vector for a CRC error assuming the vector comprises a block of data with the associated CRC bits attached. The data part of the input is returned in vector blk. The logical difference (XOR) between the attached CRC and the CRC recalculated across the data part of the input is returned in `uint32` scalar err. If err is not equal to 0, either an error has occurred or the input CRC has been masked. A logical mask can also be applied directly to err. `lteCRCDecode` returns blk, the data-only part of the combined data and CRC input vector, blkcrc, and the `uint32` err, the logical (XOR) CRC difference. The CRC polynomial is defined by a string, which can be either `'8'`, `'16'`, `'24A'`, or `'24B'`. See section 5.1.1 of [1] for the associated polynomials.

`[blk,err] = lteCRCDecode(blkcrc,poly,mask)` behaves as above except the CRC difference is also XOR-ed with the scalar mask parameter before it is returned in err. The mask value is applied to the CRC bits with the most significant bit (MSB) first and the least significant bit (LSB) last.

Examples

Check Data Vector for CRC Error

Check a data vector for CRC error, by decoding a vector with CRC attached and specifying the optional RNTI value.

First, attach a masked `'24A'`-type CRC to an all-ones vector of length 100.

```
rnti = 8;
```

```
blkcrc = lteCRCEncode(ones(100,1),'24A',rnti);
```

Call `lteCRCDecode` with the data block with appended CRC and the CRC polynomial.

```
[blk1,e1] = lteCRCDecode(blkcrc,'24A');
e1
```

Since the CRC has been masked, the returned output, `e1`, is 8, the value of `rnti`. The reason is that the logical difference between the original CRC and recalculated CRC equals the CRC mask.

Next, call `lteCRCDecode` with three input arguments, specifying the `rnti`, or mask, value.

```
[blk2,e2] = lteCRCDecode(blkcrc,'24A',rnti);
e2
```

The returned output, `e2`, is 0 because the original RNTI mask is XORed with itself.

Input Arguments

`blkcrc` — CRC input data bit vector

numeric column vector

CRC input data bit vector, specified as a numeric column vector. The function checks the input bit vector for a CRC error assuming that the data consists of a block of data with CRC bits attached.

`poly` — CRC polynomial

`'8' | '16' | '24A' | '24B'`

CRC polynomial, specified as a string. See section 5.1.1 of [1] for the associated polynomials.

`mask` — XOR mask

scalar integer

XOR mask, specified as a scalar integer. The CRC difference is XOR-ed with mask before err is returned.

Data Types: `double`

Output Arguments

`blk` — Data bit vector

column vector

Data bit vector, returned as a column vector. blk is the data-only part of the input blkcrc.

Data Types: `int8`

`err` — Logical difference

integer

Logical difference, returned as an integer. err is the logical difference between the CRC and CRC recalculated across the data part of the input.

Data Types: `uint32`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteCodeBlockDesegment` | `lteConvolutionalDecode` | `lteCRCEncode`

lteCRCEncode

Cyclic redundancy check calculation and appending

Syntax

```
blkcrc = lteCRCEncode(blk,poly)
blkcrc = lteCRCEncode(blk,poly,mask)
```

Description

`blkcrc = lteCRCEncode(blk,poly)` calculates a cyclic redundancy check (CRC) for the input data vector and returns a copy of the vector with the CRC attached. To support the correct processing of filler bits, negative input bit values are interpreted as logical 0 for the purposes of the CRC calculation. A value of –1 is used to represent filler bits. `lteCRCEncode` calculates the CRC defined by poly for the input bit vector blk and returns a copy of the input with the CRC appended in vector blkcrc. Valid options for the CRC polynomial are `'8'`, `'16'`, `'24A'`, or `'24B'`. See section 5.1.1 of [1] for the associated polynomials.

`blkcrc = lteCRCEncode(blk,poly,mask)` XOR masks the appended CRC bits with the integral value of mask. The mask value is applied to the CRC bits with the most significant bit (MSB) first and the least significant bit (LSB) last.

Examples

Calculate and Append CRC

Calculate and append the CRC associated with an all zero vector, which is also zero.

```
crc1 = lteCRCEncode(zeros(100,1),'24A');
crc1(1:10)
```

```
     0
     0
     0
     0
     0
```

```
     0
     0
     0
     0
     0
```

The result is an all-zeros vector of length 124.

Calculate and Append CRC with MSB First

Mask the CRC bits in an MSB-first order.

Set the XOR mask to 1 to make the appended CRC bits XOR masked from the most significant to least significant bit.

```
crc2 = lteCRCEncode(zeros(100,1),'24A',1);
crc2(end-10:end)
```

```
     0
     0
     0
     0
     0
     0
     0
     0
     0
     1
```

The result is all zeros, except for a single one in last element position.

Input Arguments

`blk` — Data bit vector

numeric column vector

Data bit vector, specified as a numeric column vector.

`poly` — CRC polynomial

`'8' | '16' | '24A' | '24B'`

CRC polynomial, specified as a string. See Section 5.1.1 of [1] for the associated polynomials.

`mask` — XOR mask

integer

XOR mask, specified as an integer. The appended CRC bits are XOR masked from the most significant to least significant bit.

Output Arguments

`blkcrc` — Bit vector with CRC

column vector

Bit vector with CRC, returned as a column vector.

Data Types: `int8`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`lteCodeBlockSegment` | `lteConvolutionalEncode` | `lteCRCDecode`

lteCSICodebook

Codebook for channel state information reporting

Syntax

```
out = lteCSICodebook(nu,p,n)
out = lteCSICodebook(nu,p,i1,i2)
```

Description

`out = lteCSICodebook(nu,p,n)` returns the precoding matrix, out, associated with channel state information (CSI) reporting as defined in section 7.2.4 of [1]. out is of size p-by-nu, where p is the number of transmission antennas and nu is the number of transmission layers. Using this syntax, the function accepts a number of antennas, p, of 2 or 4, a number of layers, nu, of 1, 2, 3, or 4, and a codebook index, n. In this syntax, p corresponds to the number of cell-specific reference signal antenna ports, `CellRefP`, for the `'Port7-8'` transmission scheme and the number of CSI reference signal antenna ports, `CSIRefP`, for the `'Port7-14'` transmission scheme. The codebook entries are the same in either case.

`out = lteCSICodebook(nu,p,i1,i2)` returns the precoding matrix out for CSI reporting with p=8 antennas, where nu (1...8) specifies the number of layers and i1 and i2 specify the first and second codebook indices respectively. In this syntax, p corresponds to the number of CSI reference signal antenna ports, `CSIRefP`, and the `'Port7-14'` transmission scheme is implied.

Examples

Create Codebook Entry for CSI Reporting

Create the codebook entry for CSI reporting with 4 antennas, 2 layers, and a codebook index of 3.

```
out = lteCSICodebook(2,4,3)
```

```
   0.3536 + 0.0000i   0.0000 + 0.3536i
```

```
   0.0000 - 0.3536i   0.3536 + 0.0000i
  -0.3536 + 0.0000i   0.0000 + 0.3536i
   0.0000 + 0.3536i   0.3536 + 0.0000i
```

Input Arguments

`nu` — Number of transmission layers

1,...,8

Number of transmission layers, specified as a positive scalar integer between 1 and 8.

`p` — Number of transmission antennas

1,...,8

Number of transmission antennas, specified as a positive scalar integer between 1 and 8.

`n` — Codebook index

0,...,15

Codebook index, specified as a nonnegative scalar integer between 0 and 15.

`i1` — First codebook index

0,...,15

First codebook index, specified as a nonnegative scalar integer between 0 and 15.

`i2` — Second codebook index

0,...,15

Second codebook index, specified as a nonnegative scalar integer between 0 and 15.

Output Arguments

`out` — Precoding matrix associated with CSI reporting

complex-valued numeric matrix

Precoding matrix associated with CSI reporting, returned as a complex-valued numeric matrix. out is a matrix of size p-by-nu, where p is the number of transmission antennas and nu is the number of transmission layers.

Data Types: `double`
Complex Number Support: Yes

References

[1] 3GPP TS 36.213. "Physical layer procedures." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`lteDLPrecode` | `ltePDSCH` | `ltePDSCHDecode` | `ltePMIInfo` | `ltePMISelect`

lteCSIRS

Channel state information reference signal

Syntax

```
sym = lteCSIRS(enb)
sym = lteCSIRS(enb,opts)
```

Description

`sym = lteCSIRS(enb)` returns the complex-valued channel state information reference signal (CSI-RS) symbols for transmission in a single subframe on up to eight antenna ports (p = 15,...,22). By default, the symbols are returned as a column vector. The order of the symbols is the same as how they should be mapped into the resource elements along with `lteCSIRSIndices`. If, according to the CSI-RS subframe configuration and duplex mode, there are no CSI-RS scheduled in the subframe, the output is empty. Optionally, the returned symbols can also include zeros representing the resource elements, which should be unused because they are reserved for CSI-RS symbols in one or more of the other ports. When assigned into a populated subframe grid, these zeros create empty resource elements for both Release 8 and Release 10 compatibility. When both nonzero-power and zero-power CSI-RS are output, the zero-power CSI-RS come first in the concatenated output.

sym is a column vector containing the concatenated CSI-RS symbol sequences for each of the enb`.CSIRefP` ports based on the cell-wide parameter settings, enb. The length of sym is the number of resource elements (NRE).

The optional enb`.CSIRSPeriod` and enb`.ZeroPowerCSIRSPeriod` parameters control the downlink subframes in which CSI-RS is present. For more information, see `lteCSIRSIndices`.

`sym = lteCSIRS(enb,opts)` enables control of the contents and format of the returned symbols through a cell array of option strings, opts.

Examples

Create CSI-RS Symbols and Combine with Resource Grid

Generate the CSI-RS symbols and combine them with a 10 MHz, release 8, port 0 PDSCH subframe resource grid.

Create a 10 MHz, release 8, port 0 PDSCH configuration parameter structure. Set the subframe number to 1, the number of antenna ports to 8, the CSI-RS configuration to 0, and the `CSIRSPeriod`, or *Icsi-rs*, value to 6.

```
rmc = lteRMCDL('R.2','FDD',1);
rmc.NSubframe = 1;
rmc.CSIRefP = 8;
rmc.CSIRSConfig = 0;
rmc.CSIRSPeriod = 6;
```

The 8 antenna ports are ports 15 to 22. The setting for `CSIRSPeriod` is *Icsi-rs*, which equals [*Tcsi-rs Dcsi-rs*]=[10 1].

Create a 3-D resource grid to contain the subframes for all eight ports.

```
rgrid = lteResourceGrid(rmc,rmc.CSIRefP);
```

Write the release 8 port 0 transmission into the first plane of the resource grid.

```
[wave,rgrid(:,:,1)] = lteRMCDLTool(rmc,[1,0,0,1]);
```

Create the CSI-RS symbols for ports 15 to 22. Overwrite all ports included in the port 0 transmission with the actual CSI-RS and unused RE.

```
rgrid(lteCSIRSIndices(rmc,'rs+unused')) = lteCSIRS(rmc,'rs+unused');
```

Input Arguments

`enb` — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`NCellID`	Required	Nonnegative scalar integer (0, ..., 503)	Physical layer cell identity
`NSubframe`	Required	Nonnegative scalar integer	Subframe number
`NFrame`	Optional	0 (default), Nonnegative scalar integer	Frame number
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplexing mode, specified as one of the following: • `'FDD'` — Frequency division duplex (default) • `'TDD'` — Time division duplex
The following parameters apply when `DuplexMode` is set to `'TDD'`.			
`TDDConfig`	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
`SSC`	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)
`CSIRSPeriod`	Optional	`'On'` (default), `'Off'`, `Icsi-rs` (0, ..., 154), `[Tcsi-rs Dcsi-rs]`	CSI-RS subframe configuration
The following parameters apply when `CSIRSPeriod` is set to any value but `'Off'`.			
`CSIRSConfig`	Required	Nonnegative scalar integer	CSI-RS configuration index. See table 6.10.5.2-1 in TS 36.211.
`CSIRefP`	Required	1 (default), 2, 4, 8	Number of CSI-RS antenna ports

Parameter Field	Required or Optional	Values	Description
ZeroPowerCSIRSPer	Optional	'Off' (default), 'On', Icsi-rs, (0,...,154), [Tcsi-rs Dcsi-rs]	Zero-power CSI-RS subframe configuration
The following parameters apply when ZeroPowerCSIRSPeriod is set to any value but 'Off'.			
ZeroPowerCSIR	Required	16-bit bitmap string (truncated if not 16 bits or '0' MSB extended), numerical list of CSI-RS configuration indices. See table 6.10.5.2-1 (4 CSI reference signal column) in TS 36.211.	Zero-power CSI-RS configuration index list. See table 6.10.5.2 in TS 36.211.

The optional CSIRSPeriod and ZeroPowerCSIRSPeriod parameters control the downlink subframes in which CSI-RS is present, either always 'On' or always 'Off'. You can define these parameters using the scalar subframe configuration index *Icsi-rs* between 0 and 154. Alternatively, you can define these parameters using the explicit subframe periodicity and offset pair [*Tcsi-rs Dcsi-rs*]. For more information, see section 6.10.5.3 of TS 36.211. The CSI-RS containing subframes are located in conjunction with NSubframe and the optional NFrame parameters. NSubframe can be greater than 10; thus NSubframe = 11 is equivalent to setting NSubframe to 1 and NFrame to 1.

opts — Symbol generation option strings

string | cell array of strings

Options to control the content and format of the returned symbols, specified as a string or a cell array of strings. opts can contain the following option strings.

Option	Values	Description
Symbol style	'ind' (default), 'mat'	Style for returning CSI-RS symbols, specified as one of the following options. • 'ind' — returns the CSI-RS symbols as a column vector (default) • 'mat' — returns the CSI-RS symbols as a matrix, where each column contains symbols for an individual port and CSI-RS configuration. To form a matrix, a column can contain duplicate entries.

Option	Values	Description
Symbol format	`'rsonly'` (default), `'rs+unused'`	Format of the returned symbols, specified as one of the following options. • `'rsonly'` — returns only active CSI-RS symbols (default) • `'rs+unused'` — also returns zeros for the RE locations, which should be unused because of CSI-RS transmissions on other ports

Data Types: `char` | `cell`

Output Arguments

`sym` — Complex CSI-RS symbols

column vector (default) | matrix

Complex CSI-RS symbols for transmission in a single subframe on up to 8 antenna ports, returned as a column vector or matrix. By default, the symbols are returned as a column vector. The order of the symbols is the same as how they should be mapped into the resource elements along with `lteCSIRSIndices`. If, according to the CSI-RS subframe configuration and duplex mode, there are no CSI-RS scheduled in the subframe, the output is empty. Optionally, the returned symbols can also include zeros representing the resource elements, which should be unused because they are reserved for CSI-RS symbols in one or more of the other ports. When assigned into a populated subframe grid, these zeros create empty resource elements for both Release 8 and Release 10 compatibility.

When both nonzero-power and zero-power CSI-RS are output, the zero-power CSI-RS come first in the concatenated output.

Data Types: `double`
Complex Number Support: Yes

References

[1] 3GPP TS 36.211. “Physical Channels and Modulation.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteCellRS` | `lteCSIRSIndices` | `lteDMRS` | `ltePRS`

lteCSIRSIndices

CSI-RS resource element indices

Syntax

```
ind = lteCSIRSIndices(enb)
ind = lteCSIRSIndices(enb,opts)
```

Description

`ind = lteCSIRSIndices(enb)` returns the indices of the channel state information reference signal (CSI-RS) resource elements (RE) in a subframe. This syntax supports the creation of nonzero-power and zero-power CSI-RS. By default, ind is a column vector of indices in 1-based linear indexing form that can directly index elements in an *N*-by-*M*-by-`CSIRefP` array representing the subframe grid across `CSIRefP` antenna ports (p = 15...22). If the output includes the RE that should be empty in a specific port because of CSI-RS transmissions in another port, you can create an alternative index representation. Other index generation options are also available. The order of the indices is the same as how the complex CSI-RS symbols should be mapped. The indices do not include any elements allocated to PBCH, PSS, and SSS. You can define a CSI-RS subframe configuration schedule as required. If the subframe contains no CSI-RS, the function returns an empty vector.

When both nonzero-power and zero-power CSI-RS are output, the indices for the zero-power CSI-RS come first in the concatenated output.

ind is a column vector of 1-based linear indices for the CSI-RS elements in the subframe, given the cell-wide settings parameter structure, enb. The length of ind is the number of resource elements (NRE).

`ind = lteCSIRSIndices(enb,opts)` enables control of the contents and format of the returned indices through option strings, opts.

Examples

Generate Column Vector of CSI-RS RE Indices

Generate a column vector of CSI-RS RE linear indices for ports 15 to 22 of a 10 MHz downlink subframe 0 resource grid.

Create a 10 MHz, downlink, subframe 0 configuration parameter structure. Set the number of antenna ports to 8, the CSI-RS configuration to 0, and the *Icsi-rs* value to 5.

```
rmc = lteRMCDL('R.2');
rmc.CSIRefP = 8;
rmc.CSIRSConfig = 0;
rmc.CSIRSPeriod = 5;
```

The 8 antenna ports are ports 15 to 22. The variable *Icsi-rs* is equivalent to a [*Tcsi-rs Dcsi-rs*] setting of `[10 0]`.

Generate a column of linear indices for all eight ports.

```
csirs1 = lteCSIRSIndices(rmc);
csirs1(1:5)
```

```
        3010
        3022
        3034
        3046
        3058
```

Generate Matrix of CSI-RS RE Indices

Generate a matrix of CSI-RS RE linear indices for ports 15 to 22 of a 10 MHz downlink subframe 0 resource grid.

Create a 10 MHz, downlink, subframe 0 configuration parameter structure. Set the number of antenna ports to 8, the CSI-RS configuration to 0, and the *Icsi-rs* value to 5.

```
rmc = lteRMCDL('R.2');
rmc.CSIRefP = 8;
rmc.CSIRSConfig = 0;
rmc.CSIRSPeriod = 5;
```

Generate a matrix of linear indices with eight columns.

```
csirs2 = lteCSIRSIndices(rmc,'mat');
size(csirs2)
```

```
    88     8
```

Generate Used and Unused CSI-RS RE Indices

Generate both used and unused CSI-RS RE linear indices for ports 15 to 22 of a 10 MHz downlink subframe 0 resource grid.

Create a 10 MHz, downlink, subframe 0 configuration parameter structure. Set the number of antenna ports to 8, the CSI-RS configuration to 0, and the *Icsi-rs* value to 5.

```
rmc = lteRMCDL('R.2');
rmc.CSIRefP = 8;
rmc.CSIRSConfig = 0;
rmc.CSIRSPeriod = 5;
```

Generate both used and unused CSI-RS RE in all ports.

```
csirs3 = lteCSIRSIndices(rmc,'rs+unused');
csirs3(1:5)
```

```
        3010
        3022
        3034
        3046
        3058
```

Input Arguments

enb — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
NDLRB	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
NSubframe	Required	Nonnegative scalar integer	Subframe number
NFrame	Optional	0 (default), Nonnegative scalar integer	Frame number

Parameter Field	Required or Optional	Values	Description
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplexing mode, specified as one of the following: • `'FDD'` — Frequency division duplex (default) • `'TDD'` — Time division duplex
The following apply when `DuplexMode` is set to `'TDD'`.			
`TDDConfig`	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
`SSC`	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)
`CSIRSPeriod`	Optional	`'On'` (default), `'Off'`, `Icsi-rs` (0, ..., 154), `[Tcsi-rs Dcsi-rs]`	CSI-RS subframe configuration
The following parameters apply when `CSIRSPeriod` is set to any value but `'Off'`.			
`CSIRSConfig`	Required	Nonnegative scalar integer	CSI-RS configuration index. See table 6.10.5.2-1 in TS 36.211.
`CSIRefP`	Required	1 (default), 2, 4, 8	Number of CSI-RS antenna ports
`ZeroPowerCSIRSPer`	Optional	`'Off'` (default), `'On'`, `Icsi-rs`, (0,...,154), `[Tcsi-rs Dcsi-rs]`	Zero-power CSI-RS subframe configuration
The following parameters apply when `ZeroPowerCSIRSPeriod` is set to any value but `'Off'`.			

Parameter Field	Required or Optional	Values	Description
`ZeroPowerCSIR`	Required	16-bit bitmap string (truncated if not 16 bits or `'0'` MSB extended), numerical list of CSI-RS configuration indices. See table 6.10.5.2-1 (4 CSI reference signal column) in TS 36.211.	Zero-power CSI-RS configuration index list. See table 6.10.5.2 in TS 36.211.

The optional `CSIRSPeriod` and `ZeroPowerCSIRSPeriod` parameters control the downlink subframes in which CSI-RS is present, either always `'On'` or always `'Off'`. You can define these parameters using the scalar subframe configuration index *Icsi-rs* between 0 and 154. Alternatively, you can define these parameters using the explicit subframe periodicity and offset pair [*Tcsi-rs Dcsi-rs*]. For more information, see section 6.10.5.3 of TS 36.211. The CSI-RS containing subframes are located in conjunction with `NSubframe` and the optional `NFrame` parameters. `NSubframe` can be greater than 10; thus `NSubframe` = 11 is equivalent to setting `NSubframe` to 1 and `NFrame` to 1.

`opts` — Index generation options

string | cell array of strings

Options to control the content and format of the returned indices, specified as a string or a cell array of strings. opts can contain the following option strings.

Option	Values	Description
Indexing style	`'ind'` (default), `'mat'`, `'sub'`	Style for the returned indices, specified as one of the following options. • `'ind'` — returns the indices as a column vector (default) • `'mat'` — returns the indices as a matrix. Each column contains symbols for an individual port. To form a matrix, a column can contain duplicate entries. • `'sub'` — returns the indices in `[subcarrier, symbol, antenna]` subscript row style. The number of rows in the output, ind, is the number of resource elements (NRE). Thus, ind is an NRE-by-3 matrix.
Index base	`'1based'` (default), `'0based'`	Base value of the returned indices. Specify `'1based'` to generate indices where the first value is one. Specify `'0based'` to generate indices where the first value is zero.

Option	Values	Description
Indexing format	`'rsonly'` (default), `'rs+unused'`	Format for the returned indices, specified as one of the following options. • `'rsonly'` — returns only active CSI-RS symbols (default) • `'rs+unused'` — also includes zeros for the resource element (RE) locations that should be unused because of CSI-RS transmission on another port.

Data Types: `char` | `cell`

Output Arguments

`ind` — Channel state information reference signal (CSI-RS) indices

numeric column vector | numeric matrix

Channel state information reference signal (CSI-RS) indices, returned as a vector or matrix. By default, ind is a column vector of indices in 1-based linear indexing form that can directly index elements in an *N*-by-*M*-by-`CSIRefP` array representing the subframe grid across `CSIRefP` antenna ports (p = 15...22). If the output includes the RE that should be empty in a specific port because of CSI-RS transmissions in another port, you can create an alternative index representation. Other index generation options are also available. The order of the indices is the same as how the complex CSI-RS symbols should be mapped. The indices do not include any elements allocated to PBCH, PSS, and SSS. You can define a CSI-RS subframe configuration schedule as required. If the subframe contains no CSI-RS, the function returns an empty vector.

When both nonzero-power and zero-power CSI-RS are output, the zero-power CSI-RS come first in the concatenated output.

Data Types: `uint32`

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteCellRSIndices` | `lteCSIRS` | `lteDMRSIndices` | `ltePRSIndices`

lteCellRS

Cell-specific reference signal

Syntax

```
sym = lteCellRS(enb)
sym = lteCellRS(enb,ports)
```

Description

`sym = lteCellRS(enb)` returns cell-specific reference signal symbols for cell-wide settings in the enb structure. sym is a complex-valued column vector containing cell-specific reference signal symbols. Unlike other physical channels and signals, the symbols for multiple antennas are concatenated into a single column rather than returned in a matrix with a column for each antenna. The reason for this behavior is that the number of symbols varies across the antenna ports.

`sym = lteCellRS(enb,ports)` returns cell-specific reference signal symbols for antenna ports specified in the vector, ports (0,1,2,3), and cell-wide settings structure, enb. In this case, `CellRefP` is ignored if present in enb and ports is used instead.

Examples

Find Length of Cell-Specific Reference Signals

Find the lengths of cell-specific reference signal symbols, transmitted at antenna ports 0 and 2.

Create a configuration for downlink reference measurement channel configuration number six with a 5 MHz bandwidth. Find the number of elements in the cell-specific reference signal for antenna port 0.

```
enb = lteRMCDL('R.6');
cellRefPort0 = length(lteCellRS(enb,0))
```

```
    200
```

There are 200 elements for antenna port 0.

Next, find the number of elements in the cell-specific reference signal for antenna port 2.

```
enb = lteRMCDL('R.6');
cellRefPort2 = length(lteCellRS(enb,2))
```

```
    100
```

In contrast, there are 100 elements for antenna port 2.

Input Arguments

`enb` — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`NCellID`	Required	Nonnegative scalar integer (0, …, 503)	Physical layer cell identity
`NSubframe`	Required	Nonnegative scalar integer	Subframe number
`CellRefP`	Required	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplex mode
The following parameters are dependent upon the condition that `DuplexMode` is set to `'TDD'`.			
`TDDConfig`	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration

Parameter Field	Required or Optional	Values	Description
SSC	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)

`ports` — Antenna ports
numeric vector

Antenna ports, specified as a numeric vector.

Output Arguments

`sym` — Cell-specific reference signal symbols
complex-valued numeric column vector

Cell-specific reference signal symbols, returned as a complex-valued numeric column vector. This argument contains cell-specific reference signal symbols for the specified cell-wide settings, enb, and optional number of antenna ports, ports.

Data Types: `double`
Complex Number Support: Yes

See Also

`lteCellRSIndices` | `lteCSIRS` | `lteDMRS` | `ltePRS`

lteCellRSIndices

CRS resource element indices

Syntax

```
ind = lteCellRSIndices(enb)
ind = lteCellRSIndices(enb,ports)
ind = lteCellRSIndices(enb,ports,opts)
```

Description

`ind = lteCellRSIndices(enb)` returns a column vector of resource element (RE) indices for the cell-specific reference signal (RS), given the cell-wide settings in the enb structure. By default, the indices are returned in 1-based linear indexing form that can directly index elements of a 3-D array representing the subframe resource grid for all antenna ports. These indices are ordered as the reference signal modulation symbols should be mapped. Unlike other physical channels and signals, the indices for multiple antennas are concatenated into a single column rather than returned in a matrix with a column for each antenna. This is because the number of indices varies across the antenna ports.

`ind = lteCellRSIndices(enb,ports)` returns a column vector of RE indices for antenna ports specified in the vector, ports (0,1,2,3), and cell-wide settings structure, enb. In this case, `CellRefP` is ignored if present in enb and ports is used instead.

`ind = lteCellRSIndices(enb,ports,opts)` allows control of the format of the returned indices through a cell array of option strings, opts. These option strings allow for the generation of alternative indexing formats.

Examples

Generate Cell-Specific Reference Signal RE Indices

Generate 0-based cell-specific reference signal (CRS) resource element (RE) indices in subscript form for antenna port 2.

```
enb = lteRMCDL('R.0');
enb.NCellID = 10;
ind = lteCellRSIndices(enb,2,{'0based','sub'});
ind(1:4,:)
```

```
     4     1     2
    10     1     2
    16     1     2
    22     1     2
```

In this case, each row of the generated matrix has three columns, which represent subcarrier, symbol, and antenna port, respectively.

Input Arguments

`enb` — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`NCellID`	Required	Nonnegative scalar integer (0, ..., 503)	Physical layer cell identity
`CellRefP`	Required	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplex mode
The following parameters are dependent upon the condition that `DuplexMode` is set to `'TDD'`.			
`TDDConfig`	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
`SSC`	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)

`Nsubframe`	Optional	Nonnegative scalar integer	Subframe number

`ports` — Antenna ports

numeric vector

Antenna ports, specified as a numeric vector whose elements must be (0, 1, 2, 3).

`opts` — Index generation options

string | cell array of strings

Index generation options, specified as a string or a cell array of strings that can contain the following values.

Option	Values	Description
Indexing style	`'ind'` (default), `'sub'`	Style for the returned indices, specified as one of the following options. • `'ind'` — returns the indices in linear index form as a column vector (default) • `'sub'` — returns the indices in `[subcarrier, symbol, antenna]` subscript row style. The number of rows in the output, ind, is the number of resource elements (NRE). Thus, ind is an NRE-by-3 matrix.
Index base	`'1based'` (default), `'0based'`	Base value of the returned indices. Specify `'1based'` to generate indices where the first value is one. Specify `'0based'` to generate indices where the first value is zero.

Data Types: `char` | `cell`

Output Arguments

`ind` — Cell-specific reference signal RE indices

column vector | numeric matrix

Cell-specific reference signal RE indices, returned as a column vector. Optionally, can be returned as an NRE-by-3 matrix.

Data Types: `uint32`

See Also

`lteCellRS` | `lteCSIRSIndices` | `lteDMRSIndices` | `ltePRSIndices`

lteCellSearch

Cell identity search using PSS and SSS

Syntax

```
[cellid,offset] = lteCellSearch(enb,waveform)
[cellid,offset] = lteCellSearch(enb,waveform,cellids)
```

Description

`[cellid,offset] = lteCellSearch(enb,waveform)` returns the cell identity, cellid, carried by the PSS and SSS signals in the input waveform, and the timing offset offset of the first frame head in that waveform, given the cell-wide settings structure, enb.

The input, waveform, must be a T-by-P matrix, where T is the number of time-domain samples and P is the number of receive antennas. The sampling rate of the time-domain waveform, waveform, must be the same as used in the `lteOFDMModulate` function for the specified number of resource blocks, `NDLRB`. The number of time domain samples provided, T, must be sufficient to provide at least one subframe for FDD, or two subframes for TDD since PSS and SSS lie in adjacent subframes. For the cell search to be successful, the waveform provided must contain the PSS and SSS signals.

`[cellid,offset] = lteCellSearch(enb,waveform,cellids)` returns cellid and offset as above, but constrains the search to the list of cell identities specified in the vector cellids.

Examples

Find Cell Identity

Find the cell identity used in an RMC R.12 waveform.

```
rmc = lteRMCDL('R.12');
rmc.TotSubframes = 1;
cellId = lteCellSearch(rmc,lteRMCDLTool(rmc,{}))
```

0

Input Arguments

`enb` — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length

`waveform` — Time-domain waveform

numeric matrix

Time-domain waveform, specified as a numeric matrix of size T-by-P, where T is the number of time-domain samples and P is the number of receive antennas.

Data Types: `double`
Complex Number Support: Yes

`cellids` — Cell identities

numeric vector

Cell identities, specified as a numeric vector.

Data Types: `double`

Output Arguments

`cellid` — Cell identity

numeric scalar

Cell identity, returned as a numeric scalar.

Data Types: `double`

`offset` — Timing offset

numeric scalar

Timing offset, returned as a numeric scalar.

Data Types: `double`

See Also

`lteDLFrameOffset` | `lteFrequencyCorrect` | `lteFrequencyOffset` | `lteOFDMDemodulate`

lteCodeBlockDesegment

Code block desegmentation and CRC decoding

Syntax

```
[blk,err] = lteCodeBlockDesegment(cbs,blklen)
[blk,err] = lteCodeBlockDesegment(cbs)
```

Description

`[blk,err] = lteCodeBlockDesegment(cbs,blklen)` concatenates the input code block vectors contained in cbs into an output vector, blk, of length blklen. blklen is also used to validate the dimensions of the data in cbs and to calculate the amount of filler to be removed. If cbs is a cell array containing more than one vector, each vector is assumed to have a type-24B CRC attached. This CRC is decoded and stripped from each code block prior to output concatenation and the CRC error result is placed in the associated element of vector err. The length of err is the number of code blocks. If cbs is a single vector or a cell array containing a single vector, no CRC decoding or stripping is performed and err is empty. In all cases, the number of filler bits stripped from the beginning of the (first) code block is calculated from blklen. `lteCodeBlockDesegment` performs the inverse of the code block segmentation and CRC appending (see `lteCodeBlockSegment`). `lteCodeBlockDesegment` concatenates the input code block segments into a single output data block blk, removing any filler and type-24B CRC bits that may be present in the process. The results of code block CRC decoding, if applicable, are available in the err vector.

`[blk,err] = lteCodeBlockDesegment(cbs)` is similar to the above except that no leading filler bits are stripped from the output. The detection and processing of type-24B CRC is carried out as above.

Examples

Desegment Code Block

Perform code block desegmentation and discover when segmentation occurs.

Code block segmentation occurs if the input length is greater than 6144. The input vector of length 6145 is segmented by `lteCodeBlockSegment` into two vectors of length 3072 and 3136. The output of `lteCodeBlockSegment` is a cell array, `cbs`, containing two vectors of lengths 3072 and 3136.

```
cbs = lteCodeBlockSegment(ones(6145,1));
```

Next, perform desegmentation and CRC removal.

```
[blk,err] = lteCodeBlockDesegment(cbs);
size(blk)
```

```
        6160           1
```

The first output, `blk`, is a column vector of length 6160. The second output, `err`, is a column vector of zero values.

Input Arguments

`cbs` — Code block segments

column vector | cell array

Code block segments, specified as a column vector or cell array of column vectors. If cbs is a cell array containing more than one vector, each vector is assumed to have a type-24B CRC attached. This CRC is decoded and stripped from each code block prior to output concatenation and the CRC error result is placed in the associated element of vector err. The length of err is the number of code blocks. If cbs is a single vector or a cell array containing a single vector, no CRC decoding or stripping is performed and err is empty. In all cases, the number of filler bits stripped from the beginning of the (first) code block is calculated from blklen.

`blklen` — Block length

nonnegative integer

Block length, specified as a nonnegative integer.

Output Arguments

`blk` — Output data block

column vector

Output data block, returned as a column vector. The input code blocks are segmented into a single output data block, blk, removing any filler and type-24B CRC bits.

Data Types: `int8`

`err` — Code block CRC decoding errors

column vector | nonnegative integer

Code block CRC decoding errors, returned as a nonnegative integer. The length of err is equal to the number of code blocks. If cbs is a cell array containing multiple vector elements, `lteCodeBlockDesegment` assumes that each vector has a type-24B CRC attached. The CRC is decoded and stripped from each code block prior to output concatenation and the CRC error result is placed in the associated element of err. If cbs is a single vector or a cell array containing a single vector, no CRC decoding or stripping is performed and err is empty.

Data Types: `int8`

See Also

`lteCodeBlockSegment` | `lteCRCDecode` | `lteDLSCHDecode` | `lteTurboDecode` | `lteULSCHDecode`

lteCodeBlockSegment

Code block segmentation and CRC attachment

Syntax

```
cbs = lteCodeBlockSegment(blk)
```

Description

`cbs = lteCodeBlockSegment(blk)` splits the input data bit vector blk into a cell array cbs of code block segments, with filler bits and type-24B CRC appended as appropriate, according to the rules of section 5.1.2 of [1]. Code block segmentation occurs in transport blocks, after initial type-24A CRC appending, for turbo encoded transport channels, including DL-SCH, UL-SCH, PCH, and MCH.

The segmentation and padding operation ensures that code blocks entering the turbo coder are no larger than 6144 in length and are all legal turbo code blocks sizes (only a finite set of code block sizes are supported by the LTE turbo coder). If the input block length is greater than 6144, the input block is split into a cell array of smaller code blocks where each individual block also has a type-24B CRC appended to it. The NULL filler bits, represented by –1 at the output, are prepended to the first code block so that all blocks in the set have acceptable lengths. If the input block length is less than or equal to 6144, no segmentation occurs and no CRC is appended, but the single output code block may have NULL filler bits prepended. The latter case still results in a cell array output containing a single vector.

Examples

Segment Code Block

Perform code block segmentation, providing two vectors with different lengths.

Code block segmentation occurs if the input length is greater than 6144. For example, provide a vector of length 6144.

```
cbs1 = lteCodeBlockSegment(ones(6144,1))
```

```
    [6144x1 int8]
```

No segmentation occurs.

Next, provide a vector of length 6145.

```
cbs2 = lteCodeBlockSegment(ones(6145,1))
```

```
    [3072x1 int8]    [3136x1 int8]
```

Input Arguments

`blk` — Data bit vector

column vector

Data bit vector, specified as a column vector.

Output Arguments

`cbs` — Code block segments

cell array of integer column vectors

Code block segments, returned as a cell array with `int8` column vector elements. If the input block length is less than or equal to 6144, `cbs` is a cell array containing a single column vector. If the input block length is greater than 6144, `cbs` is a cell array of multiple column vectors.

Data Types: `cell`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteCodeBlockDesegment` | `lteCRCEncode` | `lteDLSCH` | `lteDLSCHInfo` | `lteTurboEncode`

lteConvolutionalDecode

Convolutional decoding

Syntax

```
output = lteConvolutionalDecode(input)
```

Description

`output = lteConvolutionalDecode(input)` performs convolutional decoding of the input data vector, input. The input data is assumed to be soft bit data that has been encoded by a tail-biting convolutional code with constraint length 7, coding rate 1/3, and octal polynomials *G0* =133, *G1* =171 and *G2* =165. Since the code is tail-biting, output will be 1/3 of the length of the input. The input data vector is assumed to be structured as three encoded parity streams concatenated block-wise. For example, input is `[D0 D1 D2]`, where *D0*, *D1*, and *D2* are the separate parity streams resulting from the original encoding with individual polynomials *G0*, *G1* and *G2*. The decoder uses a soft input *Viterbi* algorithm without any quantization.

Examples

Perform Convolutional Decoding

This example performs convolutional encoding and decoding and compare the lengths of the input and output vectors.

Perform convolutional encoding and decoding of a vector of length 300. Viterbi decoding of a tail-biting convolutional encoded data of length 300 returns a decoded vector of length 100.

```
codedData = lteConvolutionalEncode(ones(100,1));
decodedData = lteConvolutionalDecode(codedData);
size(decodedData)
```

```
ans =
```

```
    100     1
```

The resulting output is a decoded vector of length 100.

Input Arguments

`input` — Input data

column vector

Input data, specified as a column vector. The data is assumed to be soft bit data encoded by a tail-biting convolutional code with constraint length 7, coding rate 1/3 and octal polynomials *G0*=133, *G1*=171 and *G2*=165.

Output Arguments

`output` — Convolutionally decoded data

column vector

Convolutionally decoded data, specified as a column vector. The decoded data is 1/3 the length of the input input.

Data Types: `int8`

See Also

`lteBCHDecode` | `lteConvolutionalEncode` | `lteCQIDecode` | `lteCRCDecode` | `lteDCIDecode` | `lteRateRecoverConvolutional` | `lteTurboDecode`

lteConvolutionalEncode

Convolutional encoding

Syntax

```
output = lteConvolutionalEncode(input)
```

Description

`output = lteConvolutionalEncode(input)` returns the result of convolutionally encoding the input data vector input. The convolutional code has constraint length 7 and is tail biting with coding rate 1/3 and octal polynomials *G0*=133, *G1*=171 and *G2*=165. Because the code is tail-biting, output is 3 times the length of the input. The three encoded parity streams are concatenated block-wise to form the encoded output i.e. `out = [D0 D1 D2]` where `D0`, `D1`, and `D2` are the separate vectors resulting from encoding the input input with the individual polynomials G0, G1, and G2.

Examples

Perform Convolutional Encoding

Perform convolutional encoding and compare the length of the input vector to the length of the output vector.

Perform convolutional encoding of a vector of length 100.

```
coded = lteConvolutionalEncode(ones(100,1));
size(coded)
```

```
   300     1
```

The resulting output is a coded vector of length 300, which is three times the length of the input vector, as expected.

Input Arguments

`input` — Input data vector

column vector

Input data vector, specified as a column vector.

Output Arguments

`output` — Convolutionally encoded data

column vector

Convolutionally encoded data, returned as a column vector. Because the code is tail biting, output is 3 times the length of the input. The three encoded parity streams are concatenated block-wise to form the encoded output i.e. `out = [D0 D1 D2]` where `D0`,`D1`, and `D2` are the separate vectors resulting from encoding the input input with the individual octal polynomials *G0*=133, *G1*=171, and *G2*=165.

Data Types: `int8`

See Also

`lteBCH` | `lteConvolutionalDecode` | `lteCRCEncode` | `lteDCIEncode` | `lteRateMatchConvolutional` | `lteTurboEncode`

lteDCI

Downlink control information format structures and bit payloads

Syntax

```
[dcistr,dcibits] = lteDCI(enb,istr)
[dcistr,dcibits] = lteDCI(istr)
[dcistr,dcibits] = lteDCI(istr,ibits)
```

Description

`[dcistr,dcibits] = lteDCI(enb,istr)` creates and manipulates downlink control information (DCI) messages of the formats defined in section 5.3.3 of [2]. `lteDCI` creates a DCI message in structure and bit payload forms for the cell-wide settings enb and the input structure istr.This function returns a DCI message in two forms, a structure dcistr whose fields match those of the associated DCI format, and a vector dcibits representing the set of message fields mapped to the information bit payload (including any zero padding). `lteDCI` can be used to create default DCI or blindly decode DCI format types.

The field names associated with the dcistr output structure are dependent on the DCI format field in istr and are described in “Output Arguments” on page 1-78.

By default, all values are set to zero. However, if any of the DCI fields are already present in the input istr, their values are carried forward into dcistr. This allows for easy initialization of DCI field values, particularly the resource allocation type.

Currently, `lteDCI` looks for enb`.NULRB` in case of DCI `Format0`. However, if this parameter is not provided, `lteDCI` uses enb`.NDLRB`, assuming the uplink and downlink system bandwidths are same.

`[dcistr,dcibits] = lteDCI(istr)` creates a DCI message as above, except the fields described in the structure enb above must be present as part of the input structure istr. The dcistr, in this case, also carries forward the enb`.NDLRB` and istr`.DCIFormat` fields. This syntax is deprecated and may be removed in a future release.

`[dcistr,dcibits] = lteDCI(istr,ibits)` creates a DCI where all the message fields are initialized from the input bit vector, ibits. In this case, ibits is treated as the DCI information bit payload; dcibits is equal to ibits. The length of ibits must be one of

the valid sizes for the format type and NULRB/NDLRB (see `lteDCIInfo` for details). The fields described in the enb structure must be present as part of the istr input structure. This syntax differs from the previous one in that the istr input argument does not require the `DCIFormat` field. If this field is not present, the function attempts to decode the format blindly from the length of the payload vector, ibits.

Examples

Create DCI

Create a DCI with Format 1A and distributed VRB allocation type. Set `AllocationType` to 1.

```
istr = struct('DCIFormat','Format1A','AllocationType',1);
dcistr = lteDCI(struct('NDLRB',50),istr)
```

```
         DCIFormat: 'Format1A'
               CIF: 0
    AllocationType: 1
        Allocation: [1x1 struct]
         ModCoding: 0
            HARQNo: 0
           NewData: 0
                RV: 0
          TPCPUCCH: 0
          TDDIndex: 0
```

Input Arguments

enb — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
NDLRB	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)

Parameter Field	Required or Optional	Values	Description
`NULRB`	Required	Positive scalar integer	Number of uplink (UL) resource blocks (RBs)
`CellRefP`	Optional	1 (default), 2, 4	Number of cell-specific reference signal (CRS) antenna ports
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplex mode

`istr` — Input structure

structure

Input structure, specified as a structure that can contain the following fields.

Parameter Field	Required or Optional	Values	Description
`DCIFormat`	Required, but optional if ibits is input	`'Format0'`, `'Format1'`, `'Format1A'`, `'Format1B'`, `'Format1C'`, `'Format1D'`, `'Format2'`, `'Format2A'`, `'Format2B'`, `'Format2C'`, `'Format3'`, `'Format3A'`, `'Format4'`	Downlink control information (DCI) format type string

`ibits` — Input bits

vector

Input bits, specified as a column vector. ibits is treated as the DCI information bit payload i.e. dcibits == ibits. The length of ibits must be one of the valid sizes for the format type and `NULRB`/`NDLRB` (see `lteDCIInfo` for details). When you specify ibits, the structure istr does not require the `DCIFormat` field. If the `DCIFormat` field is not present, `lteDCI` attempts to decode the format from the length of the payload vector ibits.

Data Types: `double`

Output Arguments

`dcistr` — DCI message structure
structure

DCI message structure, returned as a structure whose fields match those of the associated DCI format.

The field names associated with dcistr are dependent on the DCI format field in istr. By default, all values are set to zero. However, if any of the DCI fields are already present in the input istr, their values are carried forward into dcistr. This allows for easy initialization of DCI field values, in particular the resource allocation type.

In case of DCI `Format0`, `lteDCI` looks for `NULRB`; however, if this parameter is not provided, `lteDCI` uses `NDLRB` because it assumes the uplink and downlink system bandwidths are same.

The following table presents the fields associated with each DCI format.

DCI Formats	DCISTR Fields	Size	Description
`'Format0'`	`DCIFormat`	-	`'Format0'`
	`FreqHopping`	1-bit	PUSCH frequency hopping flag
	`Allocation`	variable	Resource block assignment/ allocation
	`ModCoding`	5-bits	Modulation, coding scheme and redundancy version
	`NewData`	1-bit	New data indicator
	`TPC`	2-bits	PUSCH TPC command
	`CShiftDMRS`	3-bits	Cyclic shift for DM RS
	`CQIReq`	1-bit	CQI request
	`TDDIndex`	2-bits	For TDD config 0, this field is the Uplink Index. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.

DCI Formats	DCISTR Fields	Size	Description
'Format1'	DCIFormat	-	'Format1'
	AllocationType	1-bit	Resource allocation header: type 0, type 1 (only if downlink bandwidth is >10 PRBs)
	Allocation	variable	Resource block assignment/ allocation
	ModCoding	5-bits	Modulation and coding scheme
	HARQNo	3-bits (FDD) 4-bits (TDD)	HARQ process number
	NewData	1-bit	New data indicator
	RV	2-bits	Redundancy version
	TPCPUCCH	2-bits	PUCCH TPC command
	TDDIndex	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
'Format1A'	DCIFormat	-	'Format1A'
	AllocationType	1-bit	VRB assignment flag: 0 (localized), 1 (distributed)
	Allocation	variable	Resource block assignment/ allocation
	ModCoding	5-bits	Modulation and coding scheme
	HARQNo	3-bits (FDD) 4-bits (TDD)	HARQ process number
	NewData	1-bit	New data indicator
	RV	2-bits	Redundancy version

DCI Formats	DCISTR Fields	Size	Description
	TPCPUCCH	2-bits	PUCCH TPC command
	TDDIndex	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
'Format1B'	DCIFormat	-	'Format1B'
	AllocationType	1-bit	VRB assignment flag: 0 (localized), 1 (distributed)
	Allocation	variable	Resource block assignment/ allocation
	ModCoding	5-bits	Modulation and coding scheme
	HARQNo	3-bits (FDD) 4-bits (TDD)	HARQ process number
	NewData	1-bit	New data indicator
	RV	2-bits	Redundancy version
	TPCPUCCH	2-bits	PUCCH TPC command
	TPMI	2-bits (2 antennas) 4-bits (4 antennas)	PMI information
	PMI	1-bit	PMI confirmation
	TDDIndex	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
'Format1C'	DCIFormat	-	'Format1C'

DCI Formats	DCISTR Fields	Size	Description
	`Allocation`	variable	Resource block assignment/ allocation
	`ModCoding`	5-bits	Modulation and coding scheme
`'Format1D'`	`DCIFormat`	-	`'Format1D'`
	`AllocationType`	1-bit	VRB assignment flag: 0 (localized), 1 (distributed)
	`Allocation`	variable	Resource block assignment/ allocation
	`ModCoding`	5-bits	Modulation and coding scheme
	`HARQNo`	3-bits (FDD) 4-bits (TDD)	HARQ process number
	`NewData`	1-bit	New data indicator
	`RV`	2-bits	Redundancy version
	`TPCPUCCH`	2-bits	PUCCH TPC command
	`TPMI`	2-bits (2 antennas) 4-bits (4 antennas)	Precoding TPMI information
	`DlPowerOffset`	1-bit	Downlink power offset
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format2'`	`DCIFormat`	-	`'Format2'`

DCI Formats	DCISTR Fields	Size	Description
	`AllocationType`	1-bit	Resource allocation header: type 0, type 1 (only if downlink bandwidth is >10 PRBs)
	`Allocation`	variable	Resource block assignment/ allocation
	`TPCPUCCH`	2-bits	PUCCH TPC command
	`HARQNo`	3-bits (FDD) 4-bits (TDD)	HARQ process number
	`SwapFlag`	1-bit	Transport block to codeword swap flag
	`ModCoding1`	5-bits	Modulation and coding scheme for transport block 1
	`NewData1`	1-bit	New data indicator for transport block 1
	`RV1`	2-bits	Redundancy version for transport block 1
	`ModCoding2`	5-bits	Modulation and coding scheme for transport block 2
	`NewData2`	1-bit	New data indicator for transport block 2
	`RV2`	2-bits	Redundancy version for transport block 2
	`PrecodingInfo`	3-bits (2-antennas) 6-bits (4-antennas)	Precoding information

DCI Formats	DCISTR Fields	Size	Description
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format2A'`	`DCIFormat`	-	`'Format2A'`
	`AllocationType`	1-bit	Resource allocation header: type 0, type 1 (only if downlink bandwidth is >10 PRBs)
	`Allocation`	variable	Resource block assignment/ allocation
	`TPCPUCCH`	2-bits	PUCCH TPC command
	`HARQNo`	3-bits (FDD) 4-bits (TDD)	HARQ process number
	`SwapFlag`	1-bit	Transport block to codeword swap flag
	`ModCoding1`	5-bits	Modulation and coding scheme for transport block 1
	`NewData1`	1-bit	New data indicator for transport block 1
	`RV1`	2-bits	Redundancy version for transport block 1
	`ModCoding2`	5-bits	Modulation and coding scheme for transport block 2
	`NewData2`	1-bit	New data indicator for transport block 2
	`RV2`	2-bits	Redundancy version for transport block 2

DCI Formats	DCISTR Fields	Size	Description
	`PrecodingInfo`	0-bits (2 antennas) 2-bits (4 antennas)	Precoding information
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format2B'`	`DCIFormat`	-	`'Format2B'`
	`AllocationType`	1-bit	Resource allocation header: type 0, type 1 (only if downlink bandwidth is >10 PRBs)
	`Allocation`	variable	Resource block assignment/ allocation
	`TPCPUCCH`	2-bits	PUCCH TPC command
	`HARQNo`	3-bits (FDD) 4-bits (TDD)	HARQ process number
	`ScramblingId`	1-bit	Scrambling identity
	`ModCoding1`	5-bits	Modulation and coding scheme for transport block 1
	`NewData1`	1-bit	New data indicator for transport block 1
	`RV1`	2-bits	Redundancy version for transport block 1
	`ModCoding2`	5-bits	Modulation and coding scheme for transport block 2

DCI Formats	DCISTR Fields	Size	Description
	`NewData2`	1-bit	New data indicator for transport block 2
	`RV2`	2-bits	Redundancy version for transport block 2
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format2C'`	`DCIFormat`	-	`'Format2C'`
	`CIF`	variable	Carrier indicator
	`AllocationType`	1-bit	Resource allocation header: type 0, type 1 (only if downlink bandwidth is >10 PRBs)
	`Allocation`	variable	Resource block assignment/ allocation
	`TPCPUCCH`	2-bits	PUCCH TPC command
	`HARQNo`	3-bits (FDD) 4-bits (TDD)	HARQ process number
	`TxIndication`	3-bits	Antenna port(s), scrambling identity, and number of layers indicator
	`SRSRequest`	variable	SRS request. Only present for TDD.
	`ModCoding1`	5-bits	Modulation and coding scheme for transport block 1
	`NewData1`	1-bit	New data indicator for transport block 1
	`RV1`	2-bits	Redundancy version for transport block 1

DCI Formats	DCISTR Fields	Size	Description
	`ModCoding2`	5-bits	Modulation and coding scheme for transport block 2
	`NewData2`	1-bit	New data indicator for transport block 2
	`RV2`	2-bits	Redundancy version for transport block 2
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format3'`	`DCIFormat`	-	`'Format3'`
	`TPCCommands`	variable	TPC commands for PUCCH and PUSCH
`'Format3A'`	`DCIFormat`	-	`'Format3A'`
	`TPCCommands`	variable	TPC commands for PUCCH and PUSCH
`'Format4'`	`DCIFormat`	-	`'Format4'`
	`CIF`	variable	Carier indicator
	`Allocation`	variable	Resource block assignment/ allocation
	`TPC`	2-bits	PUSCH TPC command
	`CShiftDMRS`	3-bits	Cyclic shift for DM RS
	`TDDIndex`	2-bits	For TDD config 0, this field is Uplink Index. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
	`CQIReq`	variable	CQI request

DCI Formats	DCISTR Fields	Size	Description
	`SRSRequest`	2-bits	SRS request
	`AllocationType`	1-bits	Resource allocation header: non-hopping PUSCH resource allocation type 0, type 1
	`ModCoding`	5-bits	Modulation, coding scheme and redundancy version
	`NewData`	1-bits	New data indicator
	`ModCoding1`	5-bits	Modulation and coding scheme for transport block 1
	`NewData1`	1-bits	New data indicator for transport block 1
	`ModCoding2`	5-bits	Modulation and coding scheme for transport block 2
	`NewData2`	1-bits	New data indicator for transport block 2
	`PrecodingInfo`	3-bits (2 antennas) 6-bits (4 antennas)	Precoding information

The `DCIFormat` fields in the table above take a string indicating the DCI format. All other fields take a string of 0s and 1s of the right size.

The `ModCoding` fields in the table above corresponds the variable I_{MCS} defined in section 7.1.7, table 7.1.7.1-1 of [3]. This field should be assigned a decimal number and the call to `lteDCI` will serialize this into a 5-bit field value. For example, `ModCoding` field for 64QAM modulation (Q_m) and transport block index (I_{TBS}) 15 should be assigned 17 (a decimal number).

By default, all values are set to zero. However if any of the DCI fields are already present in input istr, their values are carried forward intodcistr. This allows for easy initialization of DCI field values, in particular the resource allocation type. The dcistr also carries forward the `NDLRB` and `DCIFormat` fields supplied in istr.

The `Allocation` field is a structure. The field names of the`Allocation` structure are dependent on the format type and are described below. All fields take a string of zeroes and ones with the bit values.

Resource Allocation type 0			
DCI Formats	**`Allocation` Fields**	**Size (bits)**	**Description**

`'Format1'`
`'Format2'`
`'Format2A'`
 	
`'Format2B'`
	`Bitmap`	variable	Bitmap value in terms of RBG, type string

Resource Allocation type 1			
DCI Formats	**`Allocation` Fields**	**Size (bits)**	**Description**

`'Format1'`
`'Format2'`
`'Format2A'`
 	
`'Format2B'`
	`Bitmap`	variable	Bitmap value in terms of RBG, type string

	`RBSubset`	2 bits	Selected resource blocks subset indicator
	`Shift`	1 bit	Shift of the resource allocation span indicator

Resource Allocation type 2 (localized)			
DCI Formats	**`Allocation` Fields**	**Size (bits)**	**Description**

`'Format1A'`
`'Format1B'`
`'Format1C'`
`'Format1D'`
	`RIV`	variable	Resource indication value

Resource Allocation type 2 (distributed)			
DCI Formats	**`Allocation` Fields**	**Size (bits)**	**Description**

`'Format1A'`
`'Format1B'`
`'Format1C'`
`'Format1D'`
	`RIV`	variable	Resource indication value

	`Gap`	1 bit	Gap value: 0 (gap1), 1 (gap2)

Uplink Non-Hopping allocation			
DCI Formats	**`Allocation` Fields**	**Size (bits)**	**Description**
`'Format0'`	`RIV`	variable	Resource indication value

Uplink Hopping allocation			
DCI Formats	**`Allocation` Fields**	**Size (bits)**	**Description**
`'Format0'`	`RIV`	variable	Resource indication value
	`HoppingBits`	variable	Gap value: 0 (gap1), 1 (gap2)

Data Types: `struct`

`dcibits` — DCI message in bit payload form

vector

DCI message in bit payload form, returned as a column vector. dcibits represents the set of message fields mapped to the information bit payload (including any zero-padding).

Data Types: `int8`

More About

Algorithms

Resource Allocation type 0

In type 0 resource allocation, a bit map represents a resource block group (RBG) allocated to a UE. The size of RBG is given by P, which can be deduced from Table 7.1.6.1-1 of [3] for given system bandwidth. The numbers of bits in `Bitmap` field are equal to $\lceil NDLRB / P \rceil$. Each bit in the `Bitmap` will select a small contiguous group whose size depends on the bandwidth (RBG: 1…4). The maximum resource block (RB) coverage of any type 0 allocation is the entire bandwidth i.e. a type 0 allocation with all the bits in bitmap set to '1' is equivalent to the entire bandwidth.

Example — 50 RB Bandwidth, the number of bits in `Bitmap` are 17. Each bit in the 17 bit bitmap selects a group of 3 RB (apart from the last group which will only contains 2 RB for this BW) i.e. each bit is associated with a group of RE with the same color.

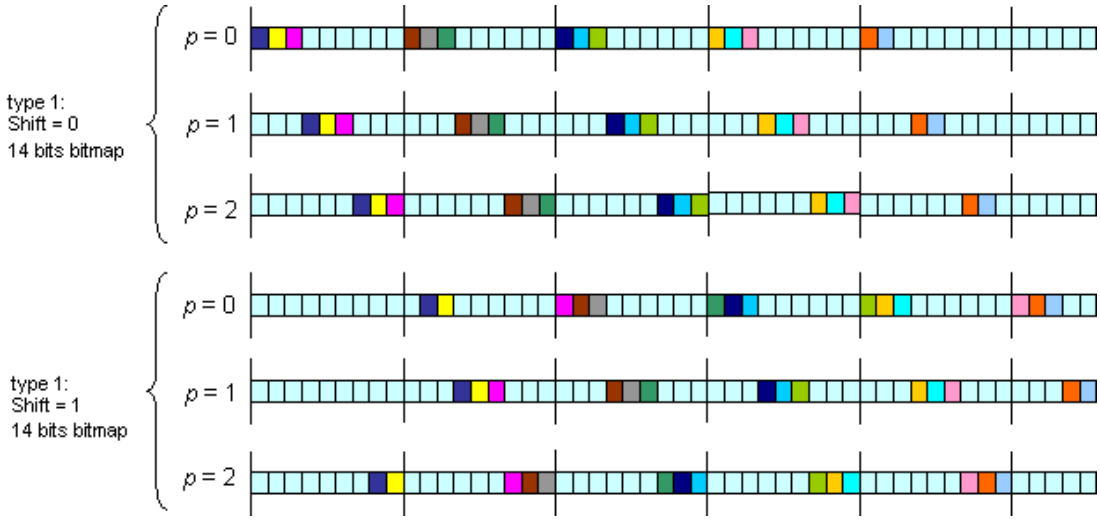
Resource Allocation type 1

In type 1 resource allocation, a bit map indicates physical resource blocks inside a selected resource block group subset p, where $0 \le p < P$. The maximum resource block (RB) coverage of any type 1 allocation is a subset of entire bandwidth i.e. a type 1 allocation, even with all the bits in the `Bitmap` set to '1' does not span the entire bandwidth. Each bit in the bitmap will selects a single RB from 'islands' of small contiguous groups whose size (RBG) and separation depend on the total bandwidth. This provides the provision of selecting a single RB without turning on any other RB.

In type 1, the resource block assignment signaling is split into 3-parts: one part represents the selected resource block group subset (by field `RBSubset`), second part indicates whether to apply offset when interpreting the bitmap (by field `Shift`) and the third part contains the bitmap that indicates to the UE specific physical resource block inside the resource block group subset (by `Bitmap`).

In comparison to type 0, the bitmap size for type 1 is always short by "$\lceil \log_2(P) \rceil + 1$ " number of bits, where P is the same as defined above.

Example – 50 RB Bandwidth, the number of bits in `Bitmap` are 14 (3-bits short as compare to type 0 due to `RBSubset` and `Shift` parameters). Each bit in the 14 bit bitmap will select an individual RB inside a selected subset. Below shows the case of setting all the bits in `Bitmap` field to '1' for different subsets and offset values.



Resource Allocation type 2

In type 2 resource allocation, physical resource blocks are not directly allocated. Instead, virtual resource blocks are allocated which are then mapped onto physical resource blocks. type 2 allocation supports both localized and distributed virtual resource block allocation differentiated by one bit-flag. The information regarding the starting point of virtual resource block and the length in terms of contiguously allocated virtual resource block can be derived from Resource Indication Value (RIV) signaled within the DCI.

Example – 50 RB Bandwidth, a UE shall be assigned an allocation of 25 resource blocks (L_{CRBs}=25), starting from resource block 10 (RB_{start}=10) in the frequency domain. To calculate the RIV value, refer to the formula given in Section 7.1.6.3 of [3], which yields RIV = 1210. This RIV is signaled in DCI and the UE could unambiguously derive the starting resource block and the number of allocated resource blocks from RIV again.

Uplink Non-Hopping Resource Allocation

For uplink non-hopping resource allocation, the rules for type 2 localized resource allocation apply for deriving the resource allocation from the RIV value.

Uplink Hopping Resource Allocation

For uplink hopping resource allocation, two types of hopping are used, Type 1 PUSCH Hopping and Type 2 PUSCH Hopping. These types are not to be confused with downlink resource allocation type 1 and type 2 described earlier. Type 1 PUSCH Hopping is calculated using the RIV value and a number of parameters signalled by higher layers. Type 2 PUSCH Hopping is calculated using a predefined pattern, a function of the subframe and frame number, which is defined in Section 5.3.4 of [1]. The fundamental set of resource blocks used as part of the hopping is calculated via the rules for type 2 localized resource allocation from the RIV value, except either 1 or 2 (depending on system bandwidth) hopping bits have been deducted from the resource allocation bitmap. These hopping bits specify whether Type 1 or Type 2 PUSCH Hopping is to be used, and for the case of 2 bits, variations of the position of the Type 1 hopping in the frequency domain. The definition of the hopping bits can be found in Table 8.4-2 of [3].

References

[1] 3GPP TS 36.211. “Physical Channels and Modulation.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.212. “Multiplexing and channel coding.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[3] 3GPP TS 36.213. “Physical layer procedures.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDCIDecode` | `lteDCIEncode` | `lteDCIInfo` | `lteDCIResourceAllocation`

lteDCIDecode

Downlink control information decoding

Syntax

```
[dcibits,crc_rnti] = lteDCIDecode(dcilen,softbits)
[dcibits,crc_rnti] = lteDCIDecode(ue,softbits)
```

Description

`[dcibits,crc_rnti] = lteDCIDecode(dcilen,softbits)` recovers a downlink control information (DCI) message vector of length dcilen from the input vector of floating-point soft bits, softbits. `lteDCIDecode` performs the inverse DCI processing operation as specified in section 5.3.3 of [1], rate recovery and Viterbi and CRC decoding. `lteDCIDecode` recovers the DCI message bit vector dcibits from an input vector of received soft bits that were previously coded by the DCI processing. `lteDCIDecode` also returns crc_rnti, the 16-bit integer result of the CRC decoder which is equivalent to the RNTI value that would need to mask (XOR) the CRC for no CRC error. This fact is used by the system to identify a DCI message with a specific ue. The function has two different ways of specifying the length of the DCI information payload to be recovered.

`[dcibits,crc_rnti] = lteDCIDecode(ue,softbits)` recovers a DCI message vector from the input vector of floating-point soft bits softbits. In this case, the length of the recovered message is defined through the ue structure in terms of DCI message format and the bandwidth.

Examples

Decode DCI Message

Perform DCI decoding on a sample codeword.

```
dciInfo = lteDCIInfo(struct('NDLRB',50));
dciBits = zeros(dciInfo.Format1,1);
cw = lteDCIEncode(struct('PDCCHFormat',1,'RNTI',10,'NDLRB',50),dciBits);
[dciBits,crcRnti] = lteDCIDecode(dciInfo.Format1,cw);
```

```
crcRnti

          10
```

The first returned output is `dcibits`, the decoded vector of length 45. The second output, `crc_rnti`, has a value of 10, corresponding to the RNTI values used in CRC masking.

Input Arguments

`dcilen` — Length of recovered DCI message vector

integer

Length of recovered DCI message vector, specified as a positive integer.

Data Types: `double`

`softbits` — Floating-point soft bits

vector

Floating-point soft bits, specified as a column vector.

Data Types: `double | int8`

`ue` — DCI message format and bandwidth

structure

DCI message format and bandwidth, specified as a structure that can contain the following fields.

Parameter Field	Required or Optional	Values	Description
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`DCIFormat`	Required	`'Format0'`, `'Format1'`, `'Format1A'`, `'Format1B'`, `'Format1C'`, `'Format1D'`, `'Format2'`, `'Format2A'`, `'Format2B'`, `'Format2C'`, `'Format3'`, `'Format3A'`, `'Format4'`	Downlink control information (DCI) format type string

Parameter Field	Required or Optional	Values	Description
`CellRefP`	Optional	1 (default), 2, 4	Number of cell-specific reference signal (CRS) antenna ports
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplex mode

Output Arguments

`dcibits` — Recovered DCI message bit vector

vector

Recovered DCI message bit vector, returned as a column vector. The length of dcibits is defined though structure ue in terms of DCI message format and the bandwidth.

Data Types: `int8`

`crc_rnti` — 16-bit integer result of the CRC decoder

vector

16-bit integer result of the CRC decoder, returned as a column vector. crc_rnti is equivalent to the RNTI value that would need to mask (XOR) the CRC for no CRC error.

Data Types: `uint32`

References

[1] 3GPP TS 36.212. “Multiplexing and channel coding.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDCI` | `lteDCIEncode` | `lteDCIInfo` | `lteDCIResourceAllocation` | `ltePDCCHDecode`

lteDCIInfo

Downlink control information message information

Syntax

```
info = lteDCIInfo(enb)
```

Description

`info = lteDCIInfo(enb)` returns a structure info containing the payload sizes for all downlink control information (DCI) message formats for the given cell-wide settings, enb.

Examples

Get DCI Message Information

Find the payload size for all DCI message formats when the `NDLRB` parameter is set to 50.

```
info = lteDCIInfo(struct('NDLRB',50))
```

```
     Format0: 27
     Format1: 31
    Format1A: 27
    Format1B: 28
    Format1C: 13
    Format1D: 28
     Format2: 43
    Format2A: 41
    Format2B: 41
    Format2C: 42
     Format3: 27
    Format3A: 27
```

```
Format4: 39
```

Input Arguments

`enb` — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`NULRB`	Required	Positive scalar integer	Number of uplink (UL) resource blocks (RBs)
`CellRefP`	Optional	1 (default), 2, 4	Number of cell-specific reference signal (CRS) antenna ports
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplex mode

Data Types: `struct`

Output Arguments

`info` — Payload sizes for all DCI message formats

structure

Payload sizes for all DCI message formats, returned as a structure with the following parameter fields.

Parameter Field	Description	Values	Data Type
`Format0`	`Format0` payload size. `Format0` is the DCI format used for the scheduling of PUSCH.	Integer	`uint64`
`Format1`	`Format1` payload size. `Format1` is the DCI format used for the scheduling of one PDSCH codeword.	Integer	`uint64`

Parameter Field	Description	Values	Data Type
`Format1A`	`Format1A` payload size. `Format1A` is the DCI format used for the compact scheduling of one PDSCH codeword and random access procedure.	Integer	`uint64`
`Format1B`	`Format1B` payload size. `Format1B` is the DCI format used for the compact scheduling of one PDSCH codeword with precoding information.	Integer	`uint64`
`Format1C`	`Format1C` payload size. `Format1C` is the DCI format used for very compact scheduling of one PDSCH codeword.	Integer	`uint64`
`Format1D`	`Format1D` payload size. `Format1D` is the DCI format used for the compact scheduling of one PDSCH codeword with precoding and power offset information.	Integer	`uint64`
`Format2`	`Format2` payload size. `Format2` is the DCI format used for the scheduling of two PDSCH codewords with precoding information for closed-loop spatial multiplexing.	Integer	`uint64`
`Format2A`	`Format2A` payload size. `Format2A` is the DCI format used for the scheduling of two PDSCH codewords with precoding information for open-loop spatial multiplexing.	Integer	`uint64`
`Format2B`	`Format2B` payload size. `Format2B` is the DCI format used for the scheduling of dual-layer transmission, for antenna ports 7 and 8.	Integer	`uint64`
`Format2C`	`Format2C` payload size. `Format2C` is the DCI format used for the scheduling of up to eight layer transmission, for antenna ports 7 to 14.	Integer	`uint64`
`Format3`	`Format3` payload size. `Format3` is the DCI format used for the transmission of transmit power control (TPC) commands for PUCCH and PUSCH with **2-bit** power adjustments.	Integer	`uint64`

Parameter Field	Description	Values	Data Type
`Format3A`	`Format3A` payload size. `Format3A` is the DCI format used for the transmission of transmit power control (TPC) commands for PUCCH and PUSCH with **1-bit** power adjustments.	Integer	`uint64`
`Format4`	`Format4` payload size. `Format4A` is the DCI format used for the scheduling of PUSCH with multi-antenna port transmission mode.	Integer	`uint64`

According to the rules defined in section 5.3.3 of [1], the payload size of DCI `Format0` and `Format1A` should always be the same and either format should be appended with padding bits, if necessary, to fulfill this condition.

None of the DCI format payload sizes should equal the ambiguous sizes defined in table 5.3.3.1.2-1 of [1] and should be appended with padding bits, if necessary. These ambiguous sizes are 12, 14, 16, 20, 24, 26, 32, 40, 44, and 56.

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDCI` | `lteDCIDecode` | `lteDCIEncode` | `lteDCIResourceAllocation`

lteDCIEncode

Downlink control information encoding

Syntax

```
cw = lteDCIEncode(ue,dcibits)
```

Description

`cw = lteDCIEncode(ue,dcibits)` returns the vector resulting from downlink control information (DCI) processing the input bit vector, dcibits. This operation is parameterized by the structure ue.

DCI processing, as described in section 5.3.3 of [1], involves CRC attachment with `RNTI` masking of the CRC, convolutional coding, and rate matching to the capacity of the PDCCH format. The output vector cw will always be `72*(2^ue.PDCCHFormat)` elements in length, which coincides with the 1, 2, 4, or 8 Control Channel Elements (CCE) associated with PDCCH Formats 0, 1, 2, or 3, respectively. One CCE is 72 bits. The input to DCI processing is the DCI format message bits to be transmitted on a single PDCCH.

Examples

Encode DCI with Zero RNTI

Perform DCI processing on an all-zero input. This processing results in an all-zero output when you set `RNTI` to 0.

Generate a dcibits input vector with zeros for a Format1 DCI message. `enb` is defined with 50 downlink RBs, 1 cell specific reference signal antenna port and FDD duplex mode.

```
enb = struct('NDLRB',50,'CellRefP',1,'DuplexMode','FDD');
dciInfo = lteDCIInfo(enb);
dcibits = zeros(dciInfo.Format1,1);
```

Define `ue` parameter structure with PDCCH Format1 and RNTI set to 0.

```
ue = struct('PDCCHFormat',1,'RNTI',0);
```

Encode the DCI bits.

```
cw = lteDCIEncode(ue,dcibits);
cw(1:10)
```

```
ans =

     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
```

For PDCCH format 1, the output vector length is 144. For this example the output is an all-zero vector since DCI bits were 0 and `RNTI` was set to 0.

Encode DCI with Unity RNTI

Perform DCI processing on an all-zero input with `RNTI` set to 1. This processing results in an non-zero output when you set `RNTI` to 1.

Generate a dcibits input vector with zeros for a Format1 DCI message. `enb` is defined with 50 downlink RBs, 1 cell specific reference signal antenna port and FDD duplex mode.

```
enb = struct('NDLRB',50,'CellRefP',1,'DuplexMode','FDD');
dciInfo = lteDCIInfo(enb);
dcibits = zeros(dciInfo.Format1,1);
```

Define `ue` parameter structure with PDCCH Format1 and RNTI set to 1.

```
ue = struct('PDCCHFormat',1,'RNTI',1);
```

Encode the DCI bits.

```
cw = lteDCIEncode(ue,dcibits);
cw(1:10)

ans =

     0
     0
     0
     0
     0
     0
     0
     1
     0
     0
```

For PDCCH format 1, the output vector length is 144. With `RNTI` set to 1, for this example the output vector is not all-zeros.

Input Arguments

`ue` — Parameter structure for DCI processing

structure

Parameter structure for DCI processing, specified as a structure that must the following fields.

Parameter Field	Required or Optional	Values	Description
`PDCCHFormat`	Required	0, 1, 2, 3	PDCCH format
`RNTI`	Required	Scalar integer	Radio network temporary identifier (RNTI) value (16 bits)

`dcibits` — DCI message bit vector

vector

DCI message bit vector, specified as a column vector.

Data Types: `double | int8`

Output Arguments

cw — Output vector

vector

Output vector resulting from DCI processing, returned as a column vector. cw is the result of DCI processing the input vector, dcibits. cw is `72*(2^ue.PDCCHFormat)` elements in length.

Data Types: `int8`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDCI` | `lteDCIDecode` | `lteDCIInfo` | `lteDCIResourceAllocation` | `ltePDCCH`

lteDCIResourceAllocation

DCI message physical resource blocks allocation

Syntax

```
[prbset, nrbg, rbgsize] = lteDCIResourceAllocation(enbue,dcistr)
[prbset, nrbg, rbgsize] = lteDCIResourceAllocation(dcistr)
```

Description

`[prbset, nrbg, rbgsize] = lteDCIResourceAllocation(enbue,dcistr)` returns the 0-based allocated physical resource block indices for the given input DCI message structure. `lteDCIResourceAllocation` returns a matrix prbset containing the 0-based physical resource block (PRB) indices, number of resource block groups nrbg, and resource block group size rbgsize, defined by the resource allocation part of DCI message structure dcistr and settings from the structure enbue.

The structure dcistr is expected to be one of the set of formats generated by the `lteDCI` function. The returned prbset will be a single column vector or a two-column matrix depending on whether the allocation type defines a different set of PRB in the first and second slots of the subframe. The allocation type may also define a minimum unit of resource block allocation defined by the resource block group size rbgsize. This specifies the number of resource blocks in a group. There are nrbg resource block groups in the allocation. The returned prbset contains 0-based PRBs.

`[prbset, nrbg, rbgsize] = lteDCIResourceAllocation(dcistr)` returns outputs prbset, nrbg, and rbgsize as above, except the fields described in structure enbue must be present as part of dcistr.

The LTE standard specifies three downlink resource allocation types: type 0, 1, and 2. In terms of the DCI formats, formats 1, 2, 2A, and 2B can use either resource allocation type 0 or type 1, with the choice signalled by `dcistr.AllocationType`=0 and `dcistr.AllocationType`=1 respectively. DCI formats 1A, 1B, 1C, and 1D use resource allocation type 2, which can be configured to be localized or distributed across resource blocks, signalled by `dcistr.AllocationType`=0 and `dcistr.AllocationType`=1 respectively.

For uplink allocations (specified in DCI format 0 messages), the allocation type is either hopping or non-hopping, signalled by `dcistr.FreqHopping`=1 and `dcistr.FreqHopping`=0, respectively. For hopping allocations, there are two types of hopping, Type 1 and Type 2. The hopping type is signalled by `dcistr.Allocation.HoppingBits` as described in Table 8.4-2 of [1]. All allocations define a single set of PRB for both slots in a subframe (prbset is a column vector) except for the distributed type 2 and uplink hopping allocations where different PRB sets are used across the slot pair.

Similarly, for non-hopping there are two types of resource allocation Type 0 and Type 1, signalled by `dcistr.AllocationType`=0 and `dcistr.AllocationType`=1 respectively. In case of DCI format 0 and Type 1 resource allocation the concatenation of the frequency hopping flag field (`dcistr.FreHopping`) and the resource block assignment and hopping resource allocation field provides the resource allocation field (`dcistr.Allocation`). The DCI format 4 messages can only signal non-hopping resource allocation Type 0 and Type 1.

Examples

Get Allocated PRB Indices for DCI Message

Create a DCI message structure and find the allocated physical resource block indices, the number of resource block groups, and the resource block group size.

```
enb = struct('NDLRB',50);
dciStr = lteDCI(enb,struct('DCIFormat','Format1A','AllocationType',1));
[prbSet,nrBg,rbgSize] = lteDCIResourceAllocation(enb,dciStr)
```

```
                    0                   27

          17

           3
```

Input Arguments

enbue — DCI message settings

structure

DCI message settings, specified as a structure. enbue can contain the following fields. The NULRB parameter field is only required if dcistr is set to `'Format0'`. The NCellID, NSubframe, NFrame, PUSCHHopping, MacTxNumber, NSubbands, and PUSCHHoppingOffset parameter fields are only required for frequency hopping, if dcistr`.FreqHopping` is set to 1.

`NDLRB` — Number of downlink resource blocks
positive scalar integer (6...110)

Number of downlink resource blocks, specified as a positive scalar integer between 6 and 110.

Data Types: `double`

`CellRefP` — Number of cell-specific reference signal antenna ports
1 (default) | Optional | 2 | 4

Number of cell-specific reference signal antenna ports, specified as 1, 2, or 4. Optional.

Data Types: `double`

`DuplexMode` — Duplex mode
`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, specified as `'FDD'` or `'TDD'`. Optional.

Data Types: `char`

`NULRB` — Number of uplink resource blocks
integer

Number of uplink resource blocks, specified as an integer. Only required if dcistr`.DCIFormat` is set to `'Format0'`.

Data Types: `double`

`NCellID` — Physical layer cell identity
integer

Physical layer cell identity, specified as an integer. Only required for frequency hopping, if dcistr`.FreqHopping` is set to 1.

Data Types: `double`

`NSubframe` — Subframe number
integer

Subframe number, specified as an integer. Only required for frequency hopping, if dcistr.FreqHopping is set to 1.

Data Types: double

NFrame — Frame number
integer

Frame number, specified as an integer. Only required for frequency hopping, if dcistr.FreqHopping is set to 1.

Data Types: double

PUSCHHopping — Uplink subframe hopping mode
'Inter' (default) | Optional | 'InterAndIntra'

Uplink subframe hopping mode, specified as 'Inter' or 'InterAndIntra'. Optional. Only required for frequency hopping, if dcistr.FreqHopping is set to 1.

Data Types: char

MacTxNumber — Number of the current MAC transmission or retransmission
Optional | 0...27

Number of the current MAC transmission or retransmission, specified as an integer from 0 to 27. Optional. Only required for frequency hopping, if dcistr.FreqHopping is set to 1.

Data Types: double

NSubbands — Number of subbands for PMI reporting
1 (default) | Optional | 1...4

Number of subbands for PMI reporting, specified as an integer from 1 to 4. Optional. Only required for frequency hopping, if dcistr.FreqHopping is set to 1.

Data Types: double

PUSCHHoppingOffset — PUSCH hopping offset
0 (default) | Optional | 0...98

PUSCH hopping offset, specifed as an integer from 0 to 98. Optional. Only required for frequency hopping, if dcistr.FreqHopping is set to 1.

Data Types: double

Data Types: struct

`dcistr` — DCI message structure
structure

DCI message structure, returned as a structure whose fields match those of the associated DCI format.

The field names associated with dcistr are dependent on the DCI format. The format is expected to be one of the formats generated by `lteDCI`.

The following table details the DCI formats and accompanying dcistr parameter fields.

DCI Formats	DCISTRFields	Size	Description
`'Format0'`	`DCIFormat`	—	`'Format0'`
	`FreqHopping`	1-bit	PUSCH frequency hopping flag
	`Allocation`	variable	Resource block assignment/ allocation
	`ModCoding`	5-bits	Modulation, coding scheme and redundancy version
	`NewData`	1-bit	New data indicator
	`TPC`	2-bits	PUSCH TPC command
	`CShiftDMRS`	3-bits	Cyclic shift for DM RS
	`CQIReq`	1-bit	CQI request
	`TDDIndex`	2-bits	For TDD config 0, this field is the Uplink Index. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format1'`	`DCIFormat`	—	`'Format1'`
	`AllocationType`	1-bit	Resource allocation header: type 0, type 1 (only if downlink bandwidth is >10 PRBs)
	`Allocation`	variable	Resource block assignment/ allocation
	`ModCoding`	5-bits	Modulation and coding scheme

DCI Formats	DCISTRFields	Size	Description
	`HARQNo`	3-bits (FDD), 4-bits (TDD)	HARQ process number
	`NewData`	1-bit	New data indicator
	`RV`	2-bits	Redundancy version
	`TPCPUCCH`	2-bits	PUCCH TPC command
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format1A'`	`DCIFormat`	—	`'Format1A'`
	`AllocationType`	1-bit	VRB assignment flag: 0 (localized), 1 (distributed)
	`Allocation`	variable	Resource block assignment/ allocation
	`ModCoding`	5-bits	Modulation and coding scheme
	`HARQNo`	3-bits (FDD), 4-bits (TDD)	HARQ process number
	`NewData`	1-bit	New data indicator
	`RV`	2-bits	Redundancy version
	`TPCPUCCH`	2-bits	PUCCH TPC command
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format1B'`	`DCIFormat`	—	`'Format1B'`
	`AllocationType`	1-bit	VRB assignment flag: 0 (localized), 1 (distributed)
	`Allocation`	variable	Resource block assignment/ allocation
	`ModCoding`	5-bits	Modulation and coding scheme

DCI Formats	DCISTRFields	Size	Description
	HARQNo	3-bits (FDD), 4-bits (TDD)	HARQ process number
	NewData	1-bit	New data indicator
	RV	2-bits	Redundancy version
	TPCPUCCH	2-bits	PUCCH TPC command
	TPMI	2-bits (2 antennas), 4-bits (4 antennas)	PMI information
	PMI	1-bit	PMI confirmation
	TDDIndex	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
'Format1C'	DCIFormat	—	'Format1C'
	Allocation	variable	Resource block assignment/ allocation
	ModCoding	5-bits	Modulation and coding scheme
'Format1D'	DCIFormat	—	'Format1D'
	AllocationType	1-bit	VRB assignment flag: 0 (localized), 1 (distributed)
	Allocation	variable	Resource block assignment/ allocation
	ModCoding	5-bits	Modulation and coding scheme
	HARQNo	3-bits (FDD), 4-bits (TDD)	HARQ process number
	NewData	1-bit	New data indicator
	RV	2-bits	Redundancy version
	TPCPUCCH	2-bits	PUCCH TPC command
	TPMI	2-bits (2 antennas), 4-bits (4 antennas)	Precoding TPMI information
	DlPowerOffset	1-bit	Downlink power offset

DCI Formats	DCISTRFields	Size	Description
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format2'`	`DCIFormat`	—	`'Format2'`
	`AllocationType`	1-bit	Resource allocation header: type 0, type 1 (only if downlink bandwidth is >10 PRBs)
	`Allocation`	variable	Resource block assignment/ allocation
	`TPCPUCCH`	2-bits	PUCCH TPC command
	`HARQNo`	3-bits (FDD), 4-bits (TDD)	HARQ process number
	`SwapFlag`	1-bit	Transport block to codeword swap flag
	`ModCoding1`	5-bits	Modulation and coding scheme for transport block 1
	`NewData1`	1-bit	New data indicator for transport block 1
	`RV1`	2-bits	Redundancy version for transport block 1
	`ModCoding2`	5-bits	Modulation and coding scheme for transport block 2
	`NewData2`	1-bit	New data indicator for transport block 2
	`RV2`	2-bits	Redundancy version for transport block 2
	`PrecodingInfo`	3-bits (2-antennas), 6-bits (4-antennas)	Precoding information
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.

DCI Formats	DCISTRFields	Size	Description
'Format2A'	DCIFormat	—	'Format2A'
	AllocationType	1-bit	Resource allocation header: type 0, type 1 (only if downlink bandwidth is >10 PRBs)
	Allocation	variable	Resource block assignment/ allocation
	TPCPUCCH	2-bits	PUCCH TPC command
	HARQNo	3-bits (FDD), 4-bits (TDD)	HARQ process number
	SwapFlag	1-bit	Transport block to codeword swap flag
	ModCoding1	5-bits	Modulation and coding scheme for transport block 1
	NewData1	1-bit	New data indicator for transport block 1
	RV1	2-bits	Redundancy version for transport block 1
	ModCoding2	5-bits	Modulation and coding scheme for transport block 2
	NewData2	1-bit	New data indicator for transport block 2
	RV2	2-bits	Redundancy version for transport block 2
	PrecodingInfo	0-bits (2 antennas), 2-bits (4 antennas)	Precoding information
	TDDIndex	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
'Format2B'	DCIFormat	—	'Format2B'

DCI Formats	DCISTRFields	Size	Description
	`AllocationType`	1-bit	Resource allocation header: type 0, type 1 (only if downlink bandwidth is >10 PRBs)
	`Allocation`	variable	Resource block assignment/ allocation
	`TPCPUCCH`	2-bits	PUCCH TPC command
	`HARQNo`	3-bits (FDD), 4-bits (TDD)	HARQ process number
	`ScramblingId`	1-bit	Scrambling identity
	`ModCoding1`	5-bits	Modulation and coding scheme for transport block 1
	`NewData1`	1-bit	New data indicator for transport block 1
	`RV1`	2-bits	Redundancy version for transport block 1
	`ModCoding2`	5-bits	Modulation and coding scheme for transport block 2
	`NewData2`	1-bit	New data indicator for transport block 2
	`RV2`	2-bits	Redundancy version for transport block 2
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format2C'`	`DCIFormat`	-	`'Format2C'`
	`CIF`	variable	Carrier indicator
	`AllocationType`	1-bit	Resource allocation header: type 0, type 1 (only if downlink bandwidth is >10 PRBs)

DCI Formats	DCISTRFields	Size	Description
	`Allocation`	variable	Resource block assignment/ allocation
	`TPCPUCCH`	2-bits	PUCCH TPC command
	`HARQNo`	3-bits (FDD) 4-bits (TDD)	HARQ process number
	`TxIndication`	3-bits	Antenna port(s), scrambling identity, and number of layers indicator
	`SRSRequest`	variable	SRS request. Only present for TDD.
	`ModCoding1`	5-bits	Modulation and coding scheme for transport block 1
	`NewData1`	1-bit	New data indicator for transport block 1
	`RV1`	2-bits	Redundancy version for transport block 1
	`ModCoding2`	5-bits	Modulation and coding scheme for transport block 2
	`NewData2`	1-bit	New data indicator for transport block 2
	`RV2`	2-bits	Redundancy version for transport block 2
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format3'`	`DCIFormat`	—	`'Format3'`
	`TPCCommands`	variable	TPC commands for PUCCH and PUSCH
`'Format3A'`	`DCIFormat`	—	`'Format3A'`

DCI Formats	DCISTRFields	Size	Description
	`TPCCommands`	variable	TPC commands for PUCCH and PUSCH
`'Format4'`	`DCIFormat`	—	`'Format4'`
	`CIF`	variable	Carrier indicator
	`Allocation`	variable	Resource block assignment/ allocation
	`TPC`	2-bits	PUSCH TPC command
	`CShiftDMRS`	3-bits	Cyclic shift for DMRS
	`TDDIndex`	2-bits	For TDD config 0, this field is Uplink Index. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
	`CQIReq`	variable	CQI request
	`SRSRequest`	2-bits	SRS request
	`AllocationType`	1-bits	Resource allocation header: non-hopping PUSCH resource allocation type 0, type 1
	`ModCoding`	5-bits	Modulation, coding scheme and redundancy version
	`NewData`	1-bits	New data indicator
	`ModCoding1`	5-bits	Modulation and coding scheme for transport block 1
	`NewData1`	1-bits	New data indicator for transport block 1
	`ModCoding2`	5-bits	Modulation and coding scheme for transport block 2
	`NewData2`	1-bits	New data indicator for transport block 2

DCI Formats	DCISTRFields	Size	Description
	`PrecodingInfo`	3-bits (2 antennas) 6-bits (4 antennas)	Precoding information

Data Types: `struct`

Output Arguments

`prbset` — Physical resource block indices

nonnegative integer column vector | nonnegative integer matrix

Physical resource block indices, returned as a nonnegative integer column vector or a nonnegative integer matrix of size *N*-by-2. These indices are 0-based. The number of columns depends on whether the allocation type defines a different set of physical resource block indices in the first and second slots of the subframe.

Data Types: `uint64`

`nrbg` — Number of resource block groups in the allocation

integer

Number of resource block groups in the allocation, returned as an integer.

Data Types: `int32`

`rbgsize` — Resource block group size

integer

Number of resource blocks in a group, returned as an integer.

Data Types: `int32`

References

[1] 3GPP TS 36.213. “Physical layer procedures.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDCI` | `lteDLSCH` | `ltePDCCH`

lteDLChannelEstimate

Downlink channel estimation

Syntax

```
[hest noisest] = lteDLChannelEstimate(enb,rxgrid)
[hest noisest] = lteDLChannelEstimate(enb,cec,rxgrid)
[hest noisest] = lteDLChannelEstimate(enb,chs,cec,rxgrid)
```

Description

`[hest noisest] = lteDLChannelEstimate(enb,rxgrid)` returns the estimated channel coefficients using the method described in annex E and F of [1] and [2], for the purposes of transmitter error vector magnitude (EVM) testing. `lteDLChannelEstimate` returns hest, the estimated channel between each transmit and receive antenna and noisest, an estimate of the noise power spectral density on the reference signal subcarriers.

hest is an *M*-by-*N*-by-`NRxAnts`-by-`CellRefP` (optionally *M*-by-*N*-by-`NRxAnts`-by-`NLayers` for UE-specific beamforming transmission schemes) array where *M* is the number of subcarriers, *N* is the number of OFDM symbols, `NRxAnts` is the number of receive antennas, `CellRefP` is the number of cell-specific reference signal antenna ports, and `NLayers` is the number of transmission layers. Using the reference signals, an estimate of the power spectral density of the noise present on the estimated channel response coefficients is returned.

rxgrid is a 3-D *M*-by-*N*-by-`NRxAnts` array of resource elements. The second dimension of rxgrid can contain any whole number of subframes worth of OFDM symbols. For example, for a normal cyclic prefix, each subframe contains 14 OFDM symbols. Therefore, *N* is a multiple of 14.

Note: To adhere to the estimation method defined in [1] and [2], rxgrid must contain 10 subframes.

`[hest noisest] = lteDLChannelEstimate(enb,cec,rxgrid)` returns the estimated channel using the method and parameters defined by the configuration structure cec.

cec is a structure which defines the type of channel estimation performed.

`[hest noisest] = lteDLChannelEstimate(enb,chs,cec,rxgrid)` returns the estimated channel given the cell-wide settings structure, enb, PDSCH transmission configuration, chs, and channel estimator configuration, cec.

Examples

Estimate Downlink Channel Characteristics

Transmit RMC R.12, 4-antenna transmit diversity.

```
enb = lteRMCDL('R.12');
cec = struct('FreqWindow',1,'TimeWindow',1,'InterpType','cubic');
cec.PilotAverage = 'UserDefined';
cec.InterpWinSize = 3;
cec.InterpWindow = 'Causal';
txWaveform = lteRMCDLTool(enb,[1;0;0;1]);
```

Model the propagation channel by combining all transmit antennas onto one receive antenna.

```
rxWaveform = sum(txWaveform,2);
```

Perform OFDM demodulation.

```
rxGrid = lteOFDMDemodulate(enb,rxWaveform);
```

Finally, estimate channel characteristics for the received resource grid.

```
hest = lteDLChannelEstimate(enb,cec,rxGrid);
```

Input Arguments

enb — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`CellRefP`	Required	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
`NCellID`	Required	Nonnegative scalar integer (0, …, 503)	Physical layer cell identity
`NSubframe`	Required	Nonnegative scalar integer	Subframe number
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplex mode
The following parameters are dependent upon the condition that `DuplexMode` is set to `'TDD'`.			
`TDDConfig`	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
`SSC`	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)
The following parameters are dependent upon the condition that cec`.Reference` is set to `'CSIRS'`.			
`CSIRefP`	Required	1 (default), 2, 4, 8	Number of CSI-RS antenna ports
`CSIRSConfig`	Required	scalar integer	CSI-RS configuration index. See table 6.10.5.2-1 in TS 36.211.
`CSIRSPeriod`	Optional	`'On'` (default), `'Off'`, `Icsi-rs` (0, ..., 154), `[Tcsi-rs Dcsi-rs]`	CSI-RS subframe configuration

If `Reference` is set to `'CSIRS'`, the cell-wide settings structure, enb, can contain the additional fields `CSIRefP`, `CSIRSConfig`, and `CSIRSPeriod`.

CSI-RS-based channel estimation, activated when `Reference` is set to `'CSIRS'`, is strictly only valid within the standard for the `'Port7-14'` transmission scheme. The optional `CSIRSPeriod` parameter controls the downlink subframes in which CSI-RS will be present, either always `'On'` or `'Off'`, or defined by the scalar subframe configuration index, *Icsi-rs* (0...154), or the explicit subframe periodicity and offset pair, [*Tcsi-rs Dcsi-rs*]. For more information, see section 6.10.5.3 in [3].

`rxgrid` — Resource element array

3-D numeric matrix

Resource element array, specified as a 3-D numeric matrix of size *M*-by-*N*-by-`NRxAnts`. The second dimension of rxgrid can contain any whole number of subframes worth of OFDM symbols. For a normal cyclic prefix, each subframe contains 14 OFDM symbols; therefore, *N* is a multiple of 14.

Data Types: `double`
Complex Number Support: Yes

`cec` — Channel estimator configuration

structure

Channel estimator configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`PilotAverage`	Required	`'TestEVM'`, `'UserDefined'`	Type of pilot averaging The `'TestEVM'` pilot averaging will ignore other structure fields in `cec` and the method follows that described in TS36.104/TS36.141 Annex E/F for the purposes of transmitter EVM testing. The `'UserDefined'` pilot averaging uses a rectangular kernel of size `cec.FreqWindow`-by-`cec.TimeWindow` and performs a 2-D filtering operation on the pilots. Pilots near the edge of the

<table>
<tr><th>Parameter Field</th><th>Required or Optional</th><th>Values</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>resource grid are averaged less because they have no neighbors outside of the grid, or a limited number of neighbors outside of the grid obtained by the creation of virtual pilots.</td></tr>
<tr><td><code>FreqWindow</code></td><td>Required</td><td>Nonnegative scalar integer</td><td>Size of window in resource elements used to average over frequency during channel estimation</td></tr>
<tr><td><code>TimeWindow</code></td><td>Required</td><td>Nonnegative scalar integer</td><td>Size of window in resource elements used to average over time during channel estimation</td></tr>
<tr><td><code>InterpType</code></td><td>Required</td><td><code>'nearest'</code>, <code>'linear'</code>, <code>'natural'</code>, <code>'cubic'</code>, <code>'v4'</code></td><td>Type of 2-D interpolation used during interpolation. For details, see <code>griddata</code>. Supported choices are shown in the following table.
<table>
<tr><th>String</th><th>Description</th></tr>
<tr><td><code>'nearest'</code></td><td>Nearest neighbor interpolation</td></tr>
<tr><td><code>'linear'</code></td><td>Linear interpolation</td></tr>
<tr><td><code>'natural'</code></td><td>Natural neighbor interpolation</td></tr>
<tr><td><code>'cubic'</code></td><td>Cubic interpolation</td></tr>
<tr><td><code>'v4'</code></td><td>MATLAB 4 <code>griddata</code> method</td></tr>
</table></td></tr>
<tr><td><code>InterpWindow</code></td><td>Required</td><td><code>'Causal'</code>, <code>'Non-causal'</code>, <code>'Centred'</code>, <code>'Centered'</code></td><td>Interpolation window type used during channel estimation. Options <code>'Centred'</code> and <code>'Centered'</code> are equivalent.</td></tr>
</table>

Parameter Field	Required or Optional	Values	Description
`InterpWinSize`	Required	Positive scalar number. If `InterpWindow` is set to `'Causal'` or `'Non-causal'`, all numbers ≥1. If `InterpWindow` is set to `'Centred'` or `'Centered'`, only odd integers ≥1.	Window size across which to interpolate. The interpolation window size is specified in number of subframes.
The following parameters are dependent upon the condition that chs`.TxScheme` is set to `'Port5'`, `'Port7-8'`, `'Port8'`, or `'Port7-14'`.			
`Reference`	Optional	`'DMRS'` (default), `'CSIRS'`	Specifies point of reference (signals to internally generate) for channel estimation

The `'TestEVM'` pilot averaging will ignore other structure fields in cec. The method for the purpose of transmitter EVM testing follows that described in annex E and F of [1] and [2].

The `'UserDefined'` pilot averaging uses a rectangular kernel of size cec`.FreqWindow`-by-cec`.TimeWindow` and performs a 2-D filtering operation on the pilots. Pilots near the edge of the resource grid are averaged less because they have no neighbors outside of the grid or a limited number of neighbors outside of the grid obtained by the creation of virtual pilots.

`chs` — PDSCH-specific channel transmission configuration

structure

PDSCH-specific channel transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`TxScheme`	Required	`'SpatialMux'` (default),	Transmission scheme, specified as one of the following options.

Parameter Field	Required or Optional	Values	Description
		'Port0', 'TxDiversity' 'CDD', 'MultiUser', 'Port5', 'Port7-8', 'Port8', 'Port7-14'	• 'SpatialMux' — Closed-loop spatial multiplexing. • 'Port0' — Single-antenna port, port 0. • 'TxDiversity' — Transmit diversity scheme. • 'CDD' — Large delay CDD scheme. • 'MultiUser' — Multiuser MIMO scheme. • 'Port5' — Single-antenna port, port 5. • 'Port7-8' — Single-antenna port, port 7 (**NLayers** = 1). Dual layer transmission, ports 7 and 8 (**NLayers** = 2). • 'Port8' — Single-antenna port, port 8. • 'Port7-14' — Up to 8-layer transmission, ports 7–14.
PRBSet	Required	1- or 2-column integer matrix	0-based physical resource block (PRB) indices corresponding to the resource allocations for this PDSCH. As a column vector, the resource allocation is the same in both slots of the subframe. As a 2-column matrix, this parameter specifies different PRBs for each slot in a subframe.
RNTI	Required	Scalar integer	Radio network temporary identifier (RNTI) value (16 bits)
NLayers	Required	1,2,3,4	Number of transmission layers

When TxScheme is set to 'Port5', 'Port7-8', 'Port8' or 'Port7-14' and cec.Reference is set to 'DMRS', the channel estimation is performed using UE-specific reference signals and the returned channel estimate will be of size *M*-by-*N*-by-NRxAnts-by-NLayers. Alternatively, when cec.Reference is set to 'CSIRS', the channel estimation is performed using the CSI reference signals (CSI) and the returned channel estimate will be of size *M*-by-*N*-by-NRxAnts-by-CSIRefP. For other transmission schemes, the channel estimation is performed using cell-specific reference signals. The returned channel estimate will be of size *M*-by-*N*-by-NRxAnts-by-CellRefP.

When TxScheme is set to 'Port7-8' or 'Port7-14' and cec.PilotAverage is set to 'UserDefined', if cec.TimeWindow is 2 or 4 and cec.FreqWindow is 1, the

estimator enters a special case where an averaging window of 2 or 4 pilots in time is used to average the pilot estimates. The averaging is always applied across 2 or 4 pilots, regardless of their separation in OFDM symbols. This operation ensures that averaging is always done on 2 or 4 pilots. This provides the appropriate "despreading" operation required for the case of UE-RS ports / CSI-RS ports which occupy the same time/ frequency locations but use different orthogonal covers to allow them to be differentiated at the receiver. For the CSI-RS and any number of configured CSI-RS ports, given by enb`.CSIRefP`, the pilot REs occur in pairs, one pair per subframe, that require averaging with cec`.TimeWindow`=2 and will result in a single estimate per subframe. For the UE-RS with between 1 and 4 layers (given by chs`.NLayers`), the pilot REs occur in pairs, repeated in each slot, that require averaging with cec`.TimeWindow`=2 and results in two estimates per subframe, one for each slot. For between 5 and 8 layers, the pairs are distinct between the slots of the subframe and the required averaging is cec`.TimeWindow`=4, resulting in one estimate per subframe.

Output Arguments

`hest` — Estimated channel between transmit and receive antennas

4-D numeric array

Estimated channel between transmit and receive antennas, returned as a 4-D numeric array of size *M*-by-*N*-by-`NRxAnts`-by-`CellRefP`. Optionally, for UE-specific beamforming transmission schemes, the array is of size *M*-by-*N*-by-`NRxAnts`-by-`NLayers` . The variable *M* is the number of subcarriers, *N* is the number of OFDM symbols, `NRxAnts` is the number of receive antennas, `CellRefP` is the number of cell-specific reference signal antenna ports, and `NLayers` is the number of transmission layers.

Data Types: `double`
Complex Number Support: Yes

`noisest` — Power spectral density estimate on reference signal subcarriers

numeric scalar

Power spectral density estimate on reference signal subcarriers, specified as a real-valued numeric scalar.

Data Types: `double`

More About

Algorithms

The channel estimation algorithm functions as described in the following steps:

1. Extract the reference signals, or pilot symbols, for a transmit-receive antenna pair from the received grid. Use the reference signals to calculate the least-squares estimates of the channel response at the pilot symbol positions within a received grid.
2. Average the least-squares estimates to reduce any unwanted noise from the pilot symbols.
3. Using the cleaned pilot symbol estimates, interpolate to obtain an estimate of the channel for the entire number of subframes passed into the function.

Least-Squares Estimation

The least-squares estimates of the reference signals are obtained by dividing the received pilot symbols by their expected value. The least-squares estimates are affected by any system noise. This noise needs to be removed or reduced to achieve a reasonable estimation of the channel at pilot symbol locations.

Noise Reduction and Interpolation

To minimize the effects of noise on the pilot symbol estimates, the least-squares estimates are averaged using an averaging window. This simple method produces a substantial reduction in the level of noise found on the pilot symbols. The two pilot symbol averaging methods, which also define the interpolation method performed to obtain the channel estimate, are `'TestEVM'` and `'UserDefined'`.

- `'TestEVM'` — follows the method described in annex F.3.4 of [2]. Time averaging is performed across each pilot symbol carrying subcarrier, resulting in a column vector containing the time averaged estimates of the channel. Frequency averaging is then performed using a moving window, maximum size 19. Linear interpolation is used to estimate the values between the pilot symbols. The estimated vector is then replicated and used as the entire channel estimate.

Note: For `'TestEVM'`, there are no user-defined parameters. Estimation behaves as described in [2].

The algorithm differs from that described in [2] due to the number of subframes across which time-averaging is performed. In [2], the method requires 10 subframes to be used. The function `lteDLChannelEstimate` performs time averaging across the total number of subframes contained in rxgrid.

- `'UserDefined'` — uses an averaging window defined by the user. The averaging window size is in resource elements and any pilot symbols located within the window are used to average the value of the pilot symbol found at the center of the window. The averaged pilot symbol estimates are then used to perform a 2-D interpolation across a window of subframes. The location of pilot symbols within the subframe is not ideally suited to interpolation. To account for this virtual pilots are created and placed out with the area of the current subframe. This allows complete and accurate interpolation to be performed. The subframe window can be specified to be causal, non-causal, or centered, depending on the data available.

- "Channel Estimation"

References

[1] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.141. "Base Station (BS) conformance testing." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[3] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`griddata` | `lteDLPerfectChannelEstimate` | `lteEqualizeMIMO` | `lteEqualizeMMSE` | `lteEqualizeZF` | `lteOFDMDemodulate`

lteDLConformanceTestTool

Downlink PDSCH demodulation conformance test tool

Syntax

```
lteDLConformanceTestTool
```

Description

`lteDLConformanceTestTool` launches a graphical user interface (GUI) to carry out the simulation for a given parameterization. The tool performs the PDSCH demodulation performance test as defined in [1]. The various test setups can easily be configured via the user interface. This tool supports both frame structures—frame structure type-1 (FDD) and type-2 (TDD).

The throughput performance graphs updates dynamically during the simulation run and provides help in an early understanding to the behavior of system for given setup. The graphical throughput visualization is divided in two parts: one updates the cumulative throughput per given signal-to-noise (SNR) ratio for the number of simulated frames and the other holds the trend of throughput curve for each SNR distinguished by a distinct color.

The tool is able to utilize user-defined configuration structure. This is done by selecting **User defined** from the reference channel pop-up menu, which prompts you for a configuration structure name. The tool expects this variable to be present in the MATLAB base workspace. You can create the basic configuration structure with the function `lteRMCDL` by choosing the closely matched RMC and modifying to meet your requirements.

The simulation result is stored in a user-defined variable specified via the GUI in the base workspace.

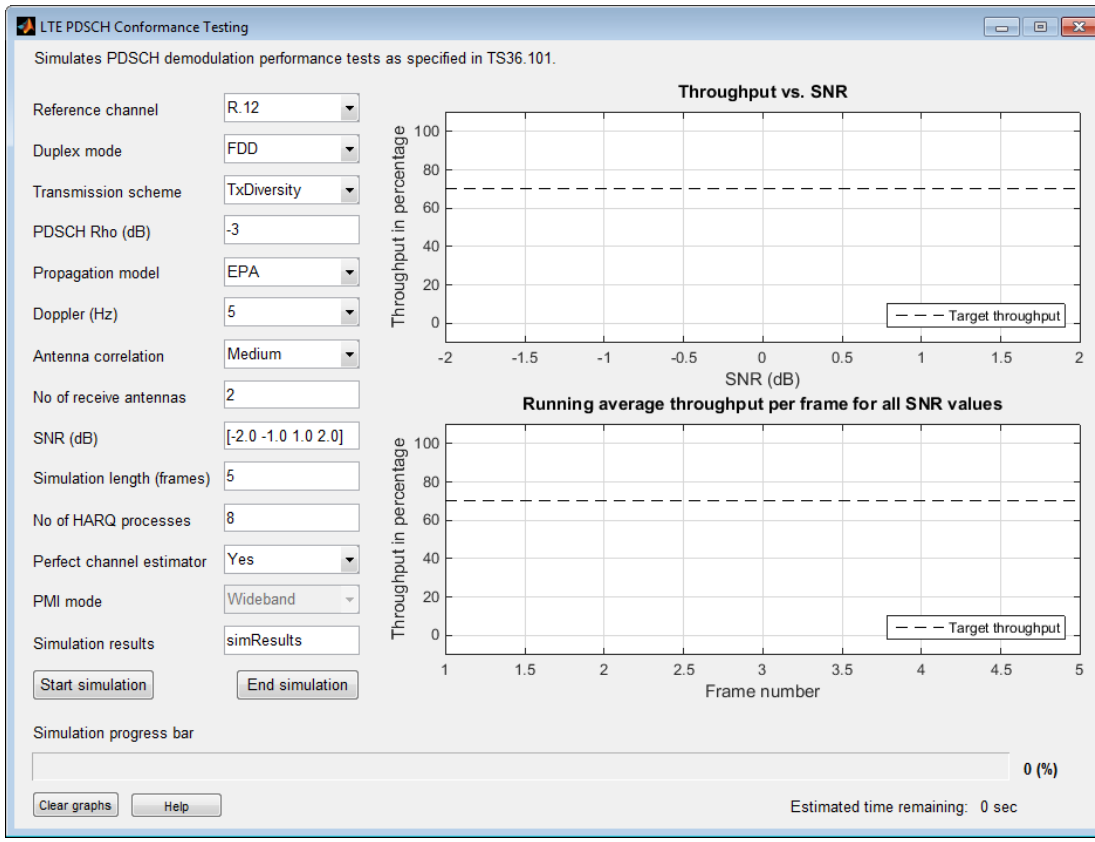
Examples

Launch PDSCH Demodulation Conformance Test Tool

Launch the tool to perform the downlink PDSCH demodulation conformance test.

```
lteDLConformanceTestTool;
```

The LTE PDSCH Conformance Testing dialog box appears.



- “Analyze Throughput for PDSCH Demodulation Performance Test”

References

[1] 3GPP TS 36.101. “User Equipment (UE) Radio Transmission and Reception.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteRMCDLTool` | `lteRMCULTool` | `lteTestModel`

lteDLDeprecode

Downlink deprecoding onto transmission layers

Syntax

```
out = lteDLDeprecode(in,nu,txscheme,codebook)
out = lteDLDeprecode(enb,chs,in)
```

Description

`out = lteDLDeprecode(in,nu,txscheme,codebook)` performs deprecoding using matrix pseudo-inversion to undo processing described in [1], section 6.3.4. The function returns an *M*-by-*NU* matrix `out` containing `nu` layers, with *M* symbols in each layer. The overall operation of the deprecoder is to transpose what is defined in the specification. The symbols for layers and antennas lie in columns rather than in rows.

`out = lteDLDeprecode(enb,chs,in)` performs deprecoding of the precoded symbol matrix, `in`, according to cell-wide settings `enb` and `chs` (channel transmission configurations).

Examples

Perform Deprecoding on Identity Matrix

Deprecode a precoded identity matrix having codebook index 1 for three layers and four antennas.

```
in  = lteDLPrecode(eye(3),4,'SpatialMux',1);
out = lteDLDeprecode(in,3,'SpatialMux',1)
```

```
out =

   1.0000 + 0.0000i   0.0000 - 0.0000i  -0.0000 + 0.0000i
   0.0000 - 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0000 + 0.0000i   0.0000 - 0.0000i   1.0000 + 0.0000i
```

Input Arguments

`in` — Precoded input symbols

numeric matrix

Precoded input symbols, specified as numeric matrix. The size of the matrix is *N*-by-*P*, where *P* is the number of transmission antennas and *N* is the number of symbols per antenna. Generate matrix by extracting a PDSCH using `ltePDSCHIndices` function on a received resource array. You can perform a similar extraction using the index generator for any other downlink channel that utilizes precoding.

`nu` — Number of layers

positive scalar integer (1, …, 8)

Number of layers, specified as a positive scalar integer between 1 and 8. The maximum number of layers depends on the string specified for the transmission scheme, `txscheme`.

`txscheme` — Transmission scheme

`string`

Transmission scheme, specified as a string.

String	Description
`'Port0'`	Single-antenna port, port 0
`'TxDiversity'`	Transmit diversity scheme
`'CDD'`	Large delay CDD scheme
`'SpatialMux'`	Closed-loop spatial multiplexing scheme
`'MultiUser'`	Multiuser MIMO (multiple-input multiple-output) scheme
`'Port5'`	Single-antenna port, port 5
`'Port7-8'`	Single-antenna port, port 7 (when `NLayers` = 1); dual-layer transmission, ports 7 and 8 (when `NLayers` = 2)
`'Port8'`	Single-antenna port, Port 8
`'Port7-14'`	Up to 8 layer transmission, ports 7-14

`codebook` — Codebook index

scalar integer (0, ..., 15)

Codebook index to use during deprecoding, specified as a scalar integer between 0 and 15. This input is ignored for the `'Port0'`, `'TxDiversity'`, and `'CDD'` transmission schemes. You can find codebook matrix corresponding to a particular index in [1], section 6.3.4.

Data Types: `double`

`enb` — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields:

Parameter Field	Required or Optional	Values	Description
The following parameters apply when `chs.TxScheme` is set to `'SpatialMux'` or `'MultiUser'`.			
`NCellID`	Required	Nonnegative scalar integer (0, …, 503)	Physical layer cell identity
`NSubframe`	Required	Nonnegative scalar integer	Subframe number
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`CFI`	Required	1, 2, 3	Control format indicator (CFI) value
`CellRefP`	Required	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplex mode
The following parameters apply when `DuplexMode` is set to `'TDD'`.			
`TDDConfig`	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
`SSC`	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)

Data Types: `struct`

`chs` — Channel-specific transmission configuration

structure

Channel-specific transmission configuration, specified as a structure that can contain the following parameter fields:

Parameter Field	Required or Optional	Values	Description
`TxScheme`	Required	`'SpatialMux'` (default), `'Port0'`, `'TxDiversity'`, `'CDD'`, `'MultiUser'`, `'Port5'`, `'Port7-8'`, `'Port8'`, `'Port7-14'`	Transmission scheme, specified as one of the following options. • `'SpatialMux'` — Closed-loop spatial multiplexing. • `'Port0'` — Single-antenna port, port 0. • `'TxDiversity'` — Transmit diversity scheme. • `'CDD'` — Large delay CDD scheme. • `'MultiUser'` — Multiuser MIMO scheme. • `'Port5'` — Single-antenna port, port 5. • `'Port7-8'` — Single-antenna port, port 7 (**`NLayers`** = 1). Dual layer transmission, ports 7 and 8 (**`NLayers`** = 2). • `'Port8'` — Single-antenna port, port 8.

Parameter Field	Required or Optional	Values	Description
			• `'Port7-14'` — Up to 8-layer transmission, ports 7–14.
`NLayers`	Required	1, …, 8	Number of transmission layers
The following parameters are apply when `TxScheme` is set to `'SpatialMux'` or `'MultiUser'`. You must include either `CodebookIdx` field or both `PMISet` and `PRBSet` fields. For more information, see Algorithms.			
`CodebookIdx`	Required	Nonnegative scalar integer (0, ..., 15)	Codebook index used during precoding
`PMISet`	Required	Integer vector (0, ..., 15)	Precoder matrix indication (PMI) set. It can contain either a single value, corresponding to single PMI mode, or multiple values, corresponding to multiple or subband PMI mode. The number of values depends on the CellRefP, transmission layers and TxScheme. For more information, see `ltePMIInfo`.

Parameter Field	Required or Optional	Values	Description
`PRBSet`	Required	1- or 2-column integer matrix	0-based physical resource block (PRB) indices corresponding to the resource allocations for this PDSCH. As a column vector, the resource allocation is the same in both slots of the subframe. As a 2-column matrix, this parameter specifies different PRBs for each slot in a subframe.

The fields `PMISet` and `PRBSet` are used to determine the frequency-domain position that each precoded symbol in out occupies. This step is performed to apply the correct subband precoder when multiple PMI mode is used. Alternatively, you can provide the `CodebookIdx` parameter field. `CodebookIdx` is a scalar specifying the codebook index to use across the entire bandwidth. Therefore, the `CodebookIdx` field does not support subband precoding. The relationship between PMI values and codebook index is given in [2],Section 7.2.4.

Output Arguments

`out` — Deprecoded downlink output

numeric matrix

Deprecoded downlink output, returned as *M*-by-*NU* numeric matrix, containing *NU* layers, with *M* symbols in each layer.

More About

Algorithms

For transmission schemes `'CDD'`, `'SpatialMux'`, and `'MultiUser'`, and degenerately `'Port0'`, precoding involves multiplying a $P \times v$ precoding matrix, F, by a $v \times M$ matrix,

representing M symbols on each of v transmission layers. This multiplication yields a $P \times M$ matrix, representing M precoded symbols on each of P antenna ports. Depending on the transmission scheme, the precoding matrix can be composed of multiple matrices multiplied together. But the size of the product, F, is always $P \times v$.

For the `'TxDiversity'` transmission scheme, a $P^2 \times 2v$ precoding matrix, F, is multiplied by a $2v \times M$ matrix, formed by splitting the real and imaginary components of a $v \times M$ matrix of symbols on layers. This multiplication yields a $P^2 \times M$ matrix of precoded symbols, which is then reshaped into a $P \times PM$ matrix for transmission. Since v is P for the `'TxDiversity'` transmission scheme, F is of size $P^2 \times 2P$, rather than $P^2 \times 2v$.

When v is P in `'CDD'`, `'SpatialMux'`, and `'MultiUser'` transmission schemes, and when P and v are 2 in the `'TxDiversity'` transmission scheme, the precoding matrix, F, is square. Its size is $2P \times 2P$ for the transmit diversity scheme and $P \times P$ otherwise. In this case, the deprecoder takes the matrix inversion of the precoding matrix to yield the deprecoding matrix F^{-1}. The matrix inversion is computed using LU decomposition with partial pivoting (row exchange):

1. Perform LU decomposition $P_x F = LU$.
2. Solve $LY = I$ using forward substitution.
3. Solve $UX = Y$ using back substitution.
4. $F^{-1} = XP_x$.

The degenerate case of the `'Port0'` transmission scheme falls into this category, with $P = v = 1$.

For the `'CDD'`, `'SpatialMux'`, and `'MultiUser'` transmission schemes, the deprecoding is then performed by multiplying F^{-1} by the transpose of the input `symbols` (`symbols` is size $M \times P$, so its transpose is size $P \times M$). This multiplication recovers the $v \times M$ (equals $P \times M$) matrix of transmission layers. For the `'TxDiversity'` transmission scheme, the deprecoding is performed, multiplying F^{-1} by the transpose of the input `symbols` (`symbols` is size $PM \times P$, so its transpose is size $P \times PM$), having first been reshaped into a $2P \times M$ matrix. This multiplication yields a $2v \times M$, matrix which is then split into two $v \times M$ matrices. The second matrix is multiplied by j and then the two matrices are added together (thus recombining real and imaginary parts) to recover the $v \times M$ matrix of transmission layers.

For the other cases that is `'CDD'`, `'SpatialMux'`, and `'MultiUser'` transmission schemes with $v \neq P$ and the `'TxDiversity'` transmission scheme with $P = 4$, the precoding matrix F is not square. Instead, the matrix is rectangular with size $P \times v$, except in the case of `'TxDiversity'` transmission scheme with $P = 4$, where it is of size $P^2 \times 2P = 16 \times 8$. The number of rows is always greater than the number of columns that is the matrix F is size $m \times n$ with $m > n$.

In this case, the deprecoder takes the matrix pseudo-inversion of the precoding matrix to yield the deprecoding matrix F^+. The matrix pseudo-inversion is computed as follows.

1. Perform LU decomposition $P_x F = LU$.
2. Remove the last $m - n$ rows of U to give $\bar{U}$.
3. Remove the last $m - n$ columns of L to give $\bar{L}$.
4. $X = \bar{U}^H \left(\bar{U}\bar{U}^H\right)^{-1} \left(\bar{L}^H \bar{L}\right)^{-1} \bar{L}^H$ (the matrix inversions are carried out as in the previous steps).
5. $F^+ = XP_x$

The application of the deprecoding matrix F^+ is the same process as described for deprecoding the square matrix case with F^+ in place of F^{-1}.

This method of pseudo-inversion is based on [3], chapter 3.4, equation (56),.

References

[1] 3GPP TS 36.211. “Physical Channels and Modulation.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.213. “Physical layer procedures.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[3] Strang, Gilbert. *Linear Algebra and Its Application*. Academic Press, 1980. 2nd Edition.

See Also

`lteDLPrecode` | `lteLayerDemap`

lteDLFrameOffset

Downlink frame timing estimate

Syntax

```
offset=lteDLFrameOffset(enb,waveform)
[offset,corr]=lteDLFrameOffset(enb,waveform)
[offset,corr]=lteDLFrameOffset(enb,waveform,corrcfg)
[offset,corr]=lteDLFrameOffset(enb,waveform,'TestEVM')
```

Description

`offset=lteDLFrameOffset(enb,waveform)` performs synchronization using the reference signals defined in the LTE standard, returning the measured timing offset from the start of the input waveform to the start of the first frame in offset.

It performs synchronization using the PSS and SSS for the time-domain waveform, waveform, given cell-wide settings structure, enb.

waveform must be a T-by-P matrix, where T is the number of time-domain samples and P is the number of receive antennas. This a matrix can be generated by OFDM modulation of a resource array using the `lteOFDMModulate` function, or by using one of the channel model functions—`lteFadingChannel`, `lteHSTChannel`, or `lteMovingChannel`).

The returned value, offset, indicates the number of samples from the start of the waveform, waveform to the position in that waveform where the first frame begins. offset is computed by extracting the timing of the peak of the correlation between waveform and internally generated time-domain reference waveforms containing PSS and SSS signals. The correlation is performed separately for each antenna; the antenna with the earliest correlation peak is used to compute offset.

This component does not perform PSS/SSS cell identity search; the cell identity must be provided in the cell-wide settings, enb.

`[offset,corr]=lteDLFrameOffset(enb,waveform)` also returns a complex matrix, corr, of the same s as waveform, the signal used to extract the timing offset.

`[offset,corr]=lteDLFrameOffset(enb,waveform,corrcfg)` provides control over which reference signals are used for timing estimation, as specified in the input structure, corrcfg.

`[offset,corr]=lteDLFrameOffset(enb,waveform,'TestEVM')` allows the correlation configuration to be easily set up as required for EVM testing. For more information, see annex E of[1]. Specify the string `'TestEVM'` to set the PSS correlation to `'on'`, the SSS correlation to `'off'`, and the CellRS correlation to the `'OmitEdgeRBs'` mode, as specified by annex E of [1].

Examples

Synchronize and Demodulate Test Model Output

Synchronize and demodulate E-UTRA Test Model (E-TM) output that has been delayed by 5 samples.

```
enb = lteTestModel('1.1','5MHz');
tx = [0;0;0;0;0;lteTestModelTool(enb)];
offset = lteDLFrameOffset(enb,tx)
rxGrid = lteOFDMDemodulate(enb,tx(1+offset:end));
```

```
     5
```

Input Arguments

`enb` — Cell-wide settings

scalar structure

Cell-wide settings, specified as a structure with the following fields.

`NDLRB` — Number of downlink resource blocks

nonnegative scalar integer

Number of downlink resource blocks, specified as a nonnegative scalar integer.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`NCellID` — Physical layer cell identity

nonnegative numeric scalar

Physical layer cell identity, specified as nonnegative numeric scalar.

Data Types: `double`

`DuplexMode` — Duplex mode

`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, specified as a string. Optional.

Data Types: `char`

Data Types: `struct`

`waveform` — Time-domain waveform

numeric matrix

Time-domain waveform, specified as a T-by-P numeric matrix, where T is the number of time-domain samples and P is the number of receive antennas.

Data Types: `double` | `single`

`corrcfg` — Control reference signals used for timing estimation

scalar structure

Control reference signals used for timing estimation, specified as a structure with the following fields.

For the following fields, if the correlation mode is configured as `'Off'`, the reference signal, `PSS`, `SSS` or `CellRS`) will not be used for timing estimation. Conversely, if the correlation mode is configured as `'On'`, the particular reference signal is used. For `CellRS`, the additional mode, `'OmitEdgeRBs'`, removes the uppermost and lowermost resource block of reference signals from the correlation. This method is specified for EVM testing in annex E of [1]. The motivation is that the phase response of any transmit filtering may be nonlinear at the band edge, resulting in a different group delay to that throughout the rest of the band.

`PSS` — PSS correlation mode

`'On'` (default) | Optional | `'Off'`

PSS correlation mode, specified as a string. Optional.

Data Types: `char`

`SSS` — SSS correlation mode

`'On'` (default) | Optional | `'Off'`

SSS correlation mode, specified as a string. Optional.

Data Types: `char`

`CellRS` — CellRS correlation mode

`'Off'` (default) | Optional | `'OmitEdgeRBs'` | `'On'`

CellRS correlation mode, specified as a string. Optional.

Data Types: `char`

`'TestEVM'` — Test EVM setting

`'TestEVM'`

Test EVM setting, specified as `'TestEVM'`. This string specifies setting PSS correlation to `'On'`, `SSS` correlation to `'Off'`, and `CellRS` to `'On'`, as specified by annex E in [1].

Data Types: `char`

Data Types: `struct`

Output Arguments

`offset` — Timing offset from the start of the input waveform to the start of the first frame

numeric scalar

Timing offset from the start of the input waveform to the start of the first frame, returned as a numeric scalar. It indicates the number of samples from the start of the waveform, waveform, to the position in that waveform where the first frame begins.

Data Types: `double`

`corr` — Signal used to extract timing offset

complex numeric matrix

Signal used to extract the timing offset, returned as a complex numeric matrix of the same s as waveform.

Data Types: `double`
Complex Number Support: Yes

References

[1] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteCellSearch` | `lteFadingChannel` | `lteFrequencyCorrect` | `lteFrequencyOffset` | `lteHSTChannel` | `lteMovingChannel` | `lteOFDMDemodulate`

lteDLPerfectChannelEstimate

Downlink perfect channel estimation

Syntax

```
h = lteDLPerfectChannelEstimate(enb,chs)
h = lteDLPerfectChannelEstimate(enb,chs,toffset)
```

Description

`h = lteDLPerfectChannelEstimate(enb,chs)` performs perfect channel estimation for a system configuration given by cell-wide settings, enb, and channel configuration structure, chs. It produces a perfect channel estimate, h, which is an *M*-by-*N*-by-`NRxAnts`-by-`CellRefP` array, where *M* is the number of subcarriers, *N* is the number of OFDM symbols, `NRxAnts` is the number of receive antennas, and `CellRefP` is the number of cell-specific reference signal antenna ports. The perfect channel estimates will only be produced for the LTE System Toolbox product's fading channel model created using the `lteFadingChannel` function. This function provides a perfect MIMO channel estimate after OFDM modulation. This perfection is achieved by setting the channel with the desired configuration and sending a set of known symbols through it for each transmit antenna in turn.

chs is a structure for configuring the channel and must also include sufficient fields for `lteFadingChannel` to be configured. Prior to execution of the channel itself, chs has the field `SamplingRate` set to the sampling rate of the time-domain waveform that will be passed to it for channel filtering.

`h = lteDLPerfectChannelEstimate(enb,chs,toffset)` takes the additional parameter, toffset, which specifies the timing offset from the start of the output of the channel to where OFDM demodulation should be performed. This parameter allows h to be the precise channel that results in the case where the receiver is precisely synchronized, as is done by using the `lteDLFrameOffset` function.

Examples

Perform Perfect Channel Estimation

Perform perfect channel estimation using the `lteFadingChannel` function.

```
enb = struct('NDLRB',6,'CyclicPrefix','Normal','CellRefP',4);
chs = struct('Seed',1,'DelayProfile','EPA','NRxAnts',2,'DopplerFreq',5.0);
chs.MIMOCorrelation = 'Low';
chs.InitPhase = 'Random';
chs.InitTime = 0.0;
enb.TotSubframes = 1;
h = lteDLPerfectChannelEstimate(enb,chs);
size(h)
```

```
    72    14     2     4
```

Input Arguments

`enb` — Cell-wide settings

scalar structure

Cell-wide settings, specified as a structure with the following fields.

`NDLRB` — Number of downlink resource blocks

nonnegative numeric scalar

Number of downlink resource blocks, specified as an integer.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`CellRefP` — Cell-specific reference signal antenna ports

1 | 2 | 4

Number of cell-specific reference signal antenna ports, specified as 1, 2, or 4.

Data Types: `double`

`TotSubframes` — Total number of subframes to be generated

nonnegative numeric scalar

Total number of subframes to be generated, specified by a numeric scalar.

Data Types: `double`

Data Types: `struct`

`chs` — Channel configuration

structure

Channel configuration, specified as a structure having the following fields.

`NRxAnts` — Number of receive antennas

positive scalar integer

Number of receive antennas, specified as a positive scalar integer of 1 or more.

Data Types: `double`

`MIMOCorrelation` — Correlation between UE and eNodeB antennas

`'Low'` | `'Medium'` | `'UplinkMedium'` | `'High'` | `'Custom'`

Correlation between UE and eNodeB antennas, specified as a string. A `'Low'` correlation is equivalent to no correlation between antennas. The `'Medium'` correlation level is applicable to tests defined in [1]. The `'UplinkMedium'` correlation level is applicable to tests defined in [2].

Data Types: `char`

`NormalizeTxAnts` — Transmit antenna number normalization

`'On'` (default) | optional | `'Off'`

Transmit antenna number normalization, specified as a string. Optional. If the value is `'On'`, the model output is normalized by `1/sqrt(P)`, where P is the number of transmit antennas. This ensures that the output power per receive antenna is unaffected by the number of transmit antennas. If the value is `'Off'`, the output is not normalized by this factor.

Data Types: `char`

`DelayProfile` — Delay profile model

`'EPA'` | `'EVA'` | `'ETU'` | `'Custom'` | `'Off'`

Delay profile model, specified as a string. `DelayProfile` set to `'Off'` switches off fading completely and implements a static MIMO channel model. In that case, the antenna geometry is given by the number of transmit antennas, the number of receive antennas, `chs.NRxAnts`, and the MIMO correlation, `chs.MIMOCorrelation`. The temporal part of the model for each link between transmit and receive antennas consists of a single path with zero delay and constant unit gain.

Data Types: `char`

`DopplerFreq` — Doppler frequency

scalar value

Doppler frequency, specified as a scalar value expressed in hertz. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`.

Data Types: `double`

`InitTime` — Fading process time offset

numeric scalar

Fading process time offset, in seconds, specified as a numeric scalar. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`.

Data Types: `double`

`NTerms` — Number of oscillators used in fading path modeling

16 (default) | optional | scalar power of 2

Number of oscillators used in fading path modeling, specified as a scalar power of 2. Optional. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`.

Data Types: `double`

`ModelType` — Rayleigh fading model type

`'GMEDS'` (default) | optional | `'Dent'`

Rayleigh fading model type, specified as `'Dent'` or `'GMEDS'`. Optional. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`.

When `ModelType` is set to `'GMEDS'`, the Rayleigh fading is modeled using the Generalized Method of Exact Doppler Spread (GMEDS), as described in [4]. When `ModelType` is set to `'Dent'`, the Rayleigh fading is modeled using the modified Jakes fading model described in [3].

Data Types: `char`

`NormalizePathGains` — Model output normalization

`'On'` (default) | optional | `'Off'`

Model output normalization, specified as `'On'` or `'Off'`. Optional. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`. If `NormalizePathGains` is set to `'On'`, the model output is normalized such that the average power is unity. If it is set to `'Off'`, the average output power is the sum of the powers of the taps of the delay profile.

Data Types: `char`

`InitPhase` — Phase initialization for the sinusoidal components of the model

`'Random'` (default) | optional | scalar value | *N*-D array

Phase initialization for the sinusoidal components of the model, specified as `'Random'`, a scalar value, or an *N*-D array. Optional. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`. If `InitPhase` is set to `'Random'`, the phases are randomly initialized according to the seed value, `Seed`. If it is set to a scalar, that value, assumed to be in radians, is used to initialize the phases of all components. Otherwise, it is necessary to provide an array of size *N*-by-*L*-by-*P*-by-`NRxAnts` to initialize explicitly the phase in radians of each component. In that case, *N* is the number of phase initialization values per path, *L* is the number of paths, *P* is the number of transmit antennas, and `NRxAnts` is the number of receive antennas. $N = 2 \times$ `NTerms` when `ModelType` is set to `'GMEDS'` and $N =$ `NTerms` when `ModelType` is set to `'Dent'`.

Data Types: `char`

`Seed` — Random number generator seed

scalar value

Random number generator seed, specified as a scalar value. Set `Seed` to zero if you want a random seed. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`. It is not required if `InitPhase` is not set to `'Random'`.

Data Types: `double`

`AveragePathGaindB` — Average gains of the discrete paths

vector

Average gains of the discrete paths, specified as a vector and expressed in dB. This parameter is required when `DelayProfile` is set to `'Custom'`.

Data Types: `double`

`PathDelays` — Delays of the discrete paths
vector

Delays of the discrete paths, specified as a vector and expressed in seconds. This parameter is required when `DelayProfile` is set to `'Custom'`. It must have the same size as `AveragePathGaindB`.

Data Types: `double`

`TxCorrelationMatrix` — Correlation between transmit antennas
matrix

Correlation between transmit antennas, specified as a P-by-P complex matrix. This parameter is required when `MIMOCorrelation` is set to `'Custom'`.

Data Types: `double | single`
Complex Number Support: Yes

`RxCorrelationMatrix` — Correlation between receive antennas
matrix

Correlation between receive antennas, specified as a complex matrix of size `NRxAnts`-by-`NRxAnts`. This parameter is required when `MIMOCorrelation` is set to `'Custom'`.

Data Types: `double | single`

Data Types: `struct`

`toffset` — Timing offset from the start of the output of the channel to where OFDM demodulation should be performed
nonnegative numeric scalar

Timing offset from the start of the output of the channel to where OFDM demodulation should be performed, specified as a nonnegative numeric scalar.

Data Types: `double`

Output Arguments

`h` — Perfect channel estimate
numeric matrix

Perfect channel estimate, returned as a numeric matrix of size *M*-by-*N*-by-`NRxAnts`-by-`CellRefP`, where *M* is the number of subcarriers, *N* is the number of OFDM symbols, `NRxAnts` is the number of receive antennas, and `CellRefP` is the number of cell-specific reference signal antenna ports.

Data Types: `double`

References

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[3] Dent, P., G. E. Bottomley, and T. Croft. "Jakes Fading Model Revisited." *Electronics Letters*. Vol. 29, 1993, Number 13, pp. 1162–1163.

[4] Pätzold, Matthias, Cheng-Xiang Wang, and Bjørn Olav Hogstad. "Two New Sum-of-Sinusoids-Based Methods for the Efficient Generation of Multiple Uncorrelated Rayleigh Fading Waveforms." *IEEE Transactions on Wireless Communications*. Vol. 8, 2009, Number 6, pp. 3122–3131.

See Also

`lteDLChannelEstimate` | `lteEqualizeMMSE` | `lteEqualizeZF` | `lteFadingChannel` | `lteOFDMDemodulate`

lteDLPrecode

Downlink precoding of transmission layers

Syntax

```
out = lteDLPrecode(in,p,txscheme,codebook)
out = lteDLPrecode(enb,chs,in)
```

Description

`out = lteDLPrecode(in,p,txscheme,codebook)` performs precoding according to [1], section 6.3.4. This function returns a *M*-by-*P* matrix, where *P* is the number of transmission antennas and *M* is the number of symbols per antenna. The matrix returned is identical to the matrix returned by `ltePDSCH` for the same set of parameters. The overall operation of the precoder is the transpose of that defined in the specification. The symbols for layers and antennas lie in columns rather than rows.

This function performs precoding of the matrix of layers, in, onto *P* antennas, using the transmission scheme specified by string txscheme.

in is an *N*-by-*NU* matrix, consisting of the *N* modulation symbols for transmission on *NU* layers. This matrix is generated using `lteLayerMap` function.

codebook is a scalar integer specifying the codebook index to be used during precoding. This input argument is ignored for the `'Port0'`, `'TxDiversity'`, and `'CDD'` transmission schemes. You can find codebook matrix corresponding to a particular index can be found in [1], section 6.3.4. Since codebook is a scalar, this syntax does not support subband precoding or multiple PMI mode.

`out = lteDLPrecode(enb,chs,in)` performs precoding of the matrix of layers, in, according to cell-wide settings enb and channel transmission configurations chs.

Examples

Perform Downlink Precoding on Identity Matrix

Perform downlink precoding using an identity matrix as input.

By precoding an identity matrix, you can gain access to the precoding matrices. Obtain the precoding matrix having codebook index 1 for three layers and four antennas.

```
out = lteDLPrecode(eye(3),4,'SpatialMux',1).'
```

```
out =

   0.2887 + 0.0000i   0.0000 - 0.2887i  -0.2887 + 0.0000i
   0.0000 + 0.2887i   0.2887 + 0.0000i   0.0000 + 0.2887i
  -0.2887 + 0.0000i   0.0000 - 0.2887i   0.2887 + 0.0000i
   0.0000 - 0.2887i   0.2887 + 0.0000i   0.0000 - 0.2887i
```

Input Arguments

`in` — Input layers

numeric matrix

Input layers, specified as a *N*-by-*NU* numeric matrix, consisting of the *N* modulation symbols for transmission on *NU* layers. Generate this matrix using `lteLayerMap`.

Data Types: `double`
Complex Number Support: Yes

`p` — Number of antennas

positive numeric scalar

Number of antennas, specified as a positive numeric scalar.

Data Types: `double`

`txscheme` — Transmission scheme

`'Port0' | 'TxDiversity' | 'CDD' | 'SpatialMux' | 'MultiUser' | 'Port5' | 'Port7-8' | 'Port8' | 'Port7-14'`

Transmission scheme, specified as a string.

String	Description
`'Port0'`	Single-antenna port, port 0
`'TxDiversity'`	Transmit diversity scheme
`'CDD'`	Large delay CDD scheme

String	Description
`'SpatialMux'`	Closed-loop spatial multiplexing scheme
`'MultiUser'`	Multiuser MIMO scheme
`'Port5'`	Single-antenna port, port 5
`'Port7-8'`	Single-antenna port, port 7 (when `NLayers` = 1); Dual layer transmission, ports 7 and 8 (when `NLayers` = 2)
`'Port8'`	Single-antenna port, port 8
`'Port7-14'`	Up to 8 layer transmission, ports 7-14

Data Types: `char`

`codebook` — Codebook matrix

nonnegative scalar integer (0, ..., 15)

Codebook matrix to use during precoding, specified as a nonnegative scalar integer between 0 and 15. This input is ignored for the `'Port0'`, `'TxDiversity'`, and `'CDD'` transmission schemes. You can find the codebook matrix corresponding to a particular index can be found in the [1], section 6.3.4. Since codebook is a scalar, the syntax that includes this parameter does not support subband precoding or multiple PMI mode.

Data Types: `double`

`enb` — eNodeB cell-wide settings

structure

eNodeB cell wide settings, specified as a scalar structure. The structure contains the following parameter fields:

Parameter Field	Required or Optional	Values	Description
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`NCellID`	Required	Nonnegative scalar integer (0, …, 503)	Physical layer cell identity

Parameter Field	Required or Optional	Values	Description
CellRefP	Required	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
CyclicPrefi	Optional	'Normal' (default), 'Extended'	Cyclic prefix length
DuplexMode	Optional	'FDD' (default), 'TDD'	Duplex mode
CFI	Required	1, 2, 3	Control format indicator (CFI) value
NSubframe	Required	Nonnegative scalar integer	Subframe number
The following parameters apply when DuplexMode is set to TDD.			
TDDConf	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
SSC	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)

chs — Channel-specific transmission configuration

structure | structure array

Channel specific transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
TxScheme	Optional	'SpatialMux' (default), 'Port0', 'TxDiversity', 'CDD', 'MultiUser', 'Port5', 'Port7-8', 'Port8', 'Port7-14'	Transmission scheme, specified as one of the following options. • 'SpatialMux' — Closed-loop spatial multiplexing. • 'Port0' — Single-antenna port, port 0. • 'TxDiversity' — Transmit diversity scheme.

Parameter Field	Required or Optional	Values	Description
			• `'CDD'` — Large delay CDD scheme. • `'MultiUser'` — Multiuser MIMO scheme. • `'Port5'` — Single-antenna port, port 5. • `'Port7-8'` — Single-antenna port, port 7 (**`NLayers`** = 1). Dual layer transmission, ports 7 and 8 (**`NLayers`** = 2). • `'Port8'` — Single-antenna port, port 8. • `'Port7-14'` — Up to 8-layer transmission, ports 7–14.
The following parameters are dependant upon the condition that TXscheme is set to `'SpatialMux'` or `'MultiUser'`. You must include either Codebook filed or both PMISet and PRBSet fields.			
`Codebookidx`	Optional	Nonnegative scalar integer (0, ..., 15)	Codebook index used during precoding
`PMISet`	Optional	Integer vector (0, ..., 15)	Precoder matrix indication (PMI) set. It can contain either a single value, corresponding to single PMI mode, or multiple values, corresponding to multiple or subband PMI mode. The number of values depends on the CellRefP, transmission layers and TxScheme.
`PRBSet`	Optional	1- or 2-column integer matrix	0-based physical resource block (PRB) indices corresponding to the resource allocations for this PDSCH. As a column vector, the resource allocation is the same in both slots of the subframe. As a 2-column matrix, this parameter specifies different PRBs for each slot in a subframe.

The fields `PMISet` and `PRBSet` determine the frequency-domain position that each precoded symbol in `out` occupies in order to apply the correct subband precoder when multiple PMI mode is being used. Alternatively, you can provide `CodebookIdx` field. `CodebookIdx` is a scalar specifying the codebook index to use across the entire bandwidth. Therefore, the `CodebookIdx` field does not support subband precoding. The relationship between PMI values and codebook indices is given by [2], Section 7.2.4.

Output Arguments

`out` — Precoded downlink output

numeric matrix

Precoded downlink output, returned as an M-by-P numeric matrix, where P is the number of transmission antennas and M is the number of symbols per antenna.

Data Types: `double`

More About

Algorithms

For transmission schemes `'CDD'`, `'SpatialMux'`, and `'MultiUser'`, and degenerately `'Port0'`, precoding involves multiplying a $P \times v$ precoding matrix, denoted as F, by a $v \times M$ matrix, representing M symbols on each of v transmission layers, to yield a $P \times M$ matrix, consisting of M precoded symbols on each of P antenna ports. Depending on the transmission scheme, the precoding matrix can be composed of multiple matrices multiplied together, but the size of the product, F, is always $P \times v$.

For the `'TxDiversity'` transmission scheme, a $P^2 \times 2v$ precoding matrix F is multiplied by a $2v \times M$ matrix, formed by splitting the real and imaginary components of a $v \times M$ matrix of symbols on layers, to yield a $P^2 \times M$ matrix of precoded symbols, which is then reshaped into a $P \times PM$ matrix for transmission. As $v = P$ for the `'TxDiversity'` transmission scheme, we can consider F be of size $P^2 \times 2P$ rather than $P^2 \times 2v$.

For the other cases i.e. `'CDD'`, `'SpatialMux'`, and `'MultiUser'` transmission schemes with $v \neq P$ and the `'TxDiversity'` transmission scheme with $P = 4$, the precoding matrix F is not square; it is rectangular with size $P \times v$ except for the `'TxDiversity'`

transmission scheme with $P = 4$ where it is of size $P^2 \times 2P = 16 \times 8$. The number of rows is always greater than the number of columns i.e. the matrix F is size $m \times n$ with $m > n$.

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

[2] 3GPP TS 36.213. "Physical layer procedures." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`lteDLDeprecode` | `lteLayerMap`

lteDLResourceGrid

Downlink subframe resource array

Syntax

```
grid = lteDLResourceGrid(enb)
grid = lteDLResourceGrid(enb,p)
```

Description

`grid = lteDLResourceGrid(enb)` returns an empty 3-D array used to represent the resource elements for one subframe across all configured antenna ports, as described in "Data Structures". It returns the resource array generated from the cell-wide settings structure, enb.

The size of grid is *N*-by-*M*-by-`CellRefP`, where *N* is the number of subcarriers (12×`NDLRB`), *M* is the number of OFDM symbols in a subframe (14 for normal cyclic prefix and 12 for extended cyclic prefix), and `CellRefP` is the number of transmit antenna ports.

`grid = lteDLResourceGrid(enb,p)` returns a resource array where the number of antenna planes in the array is specified directly by the parameter p. In this syntax, `CellRefP` is not required as a structure field of enb.

Examples

Create Empty Resource Array

Create an empty resource array representing the resource elements for 10MHz bandwidth, one subframe, and two antennas.

```
rgrid = lteDLResourceGrid(struct('NDLRB',50,'CellRefP',2));
size(rgrid)
```

```
    600    14     2
```

Input Arguments

`enb` — Cell-wide settings
structure

Cell-wide settings, specified as a structure having the following fields.

`NDLRB` — Number of downlink resource blocks
positive scalar integer

Number of downlink resource blocks, specified as a positive scalar integer.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`CellRefP` — Number of cell-specific reference signal antenna ports
1 | 2 | 4

Number of cell-specific reference signal antenna ports, specified as 1, 2, or 4.

Data Types: `double`

Data Types: `struct`

`p` — Number of antenna planes
positive scalar integer

Number of antenna planes, specified as a positive scalar integer.

Data Types: `double`

Output Arguments

`grid` — Empty downlink resource grid
3-D numeric array

Empty downlink resource grid, returned as a 3-D numeric array. This array is used to represent the resource elements for one subframe across all configured antenna ports. It has dimensions of:

- *N*-by-*M*-by-`CellRefP`, where *N* is the number of subcarriers (12×`NDLRB`), *M* is the number of OFDM symbols in a subframe (14 for normal cyclic prefix and 12 for extended cyclic prefix), and `CellRefP` is the number of transmit antenna ports. This applies when the function has a single input argument, enb.
- *N*-by-*M*-by-`p`, where `p` is the number of antenna planes. This applies when the function has two input arguments, enb and p.

Data Types: `double`

See Also

`lteDLResourceGridSize` | `lteOFDMModulate` | `lteResourceGrid` | `lteResourceGridSize` | `lteULResourceGrid` | `lteULResourceGridSize`

lteDLResourceGridSize

Size of downlink subframe resource array

Syntax

```
d = lteDLResourceGridSize(enb)
d = lteDLResourceGridSize(enb,p)
```

Description

`d = lteDLResourceGridSize(enb)` returns a three-element row vector of dimension lengths for the multidimensional array used to represent the resource elements for one subframe across all configured antenna ports, as described in "Data Structures". It returns a 3-element row vector of dimension lengths for the resource array generated from the cell-wide settings structure, enb.

The vector d is `[N M CellRefP]`, where `N` is the number of subcarriers (12×`NDLRB`), `M` is the number of OFDM or SC-FDMA symbols in a subframe, 14 for normal cyclic prefix and 12 for extended cyclic prefix, and `CellRefP` is the number of transmit antenna ports.

`d = lteDLResourceGridSize(enb,p)` returns a three-element row vector as above, except the number of antenna planes in the array is specified directly by parameter p. In this syntax, `CellRefP` is not required as a structure field of enb.

Examples

Determine Downlink Subframe Resource Array Size

Determine the size of a downlink subframe resource array.

Determine the dimensions of a downlink subframe resource array, using cell-wide settings, `enb`. Then, use the returned vector directly to create a resource grid as a multidimensional array.

```
enb = struct('NDLRB',50,'CellRefP',2,'CyclicPrefix','Normal');
```

```
rgrid = zeros(lteDLResourceGridSize(enb));
size(rgrid)

ans =

   600    14     2
```

The same result can be obtained by calling the `lteDLResourceGrid` function.

Input Arguments

`enb` — Cell-wide settings

structure

Cell-wide settings, specified as a structure having the following fields.

`NDLRB` — Number of downlink resource blocks

positive scalar integer (6,...,110)

Number of downlink resource blocks, specified as a positive scalar integer between 6 and 110. Standard bandwidth values are 6, 15, 25, 50, 75, and 100.

Data Types: `double`

`CellRefP` — Number of cell-specific reference signal antenna ports

1 | 2 | 4

Number of cell-specific reference signal antenna ports, specified as 1, 2, or 4.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

Data Types: `struct`

`p` — Number of antenna planes

positive scalar integer

Number of antenna planes, specified as a positive scalar integer. This argument directly specifies the number of antenna planes in the array.

Data Types: `double`

Output Arguments

d — Downlink resource grid dimensions

numeric 1-by-3 row vector

Downlink resource grid dimensions, returned as a numeric 1-by-3 row vector. When the function has a single argument, d is `[N M CellRefP]`, where `N` is the number of subcarriers (12×`NDLRB`), `M` is the number of OFDM or SC-FDMA symbols in a subframe, and `CellRefP` is the number of transmit antenna ports. When the number of antenna planes, p, is specified as the second input argument, then d is `[N M p]` and the input field `CellRefP` of enb is not required.

Data Types: `double`

See Also

`lteDLResourceGrid` | `lteResourceGridSize` | `lteULResourceGridSize`

lteDLSCH

Downlink shared channel

Syntax

```
[cwout,chinfo] = lteDLSCH(enb,chs,outlen,trblkin)
```

Description

`[cwout,chinfo] = lteDLSCH(enb,chs,outlen,trblkin)` applies the complete DL-SCH transport channel coding chain to the input data, trblkin, and returns the codewords in cwout. The encoding process includes type-24A CRC calculation, code block segmentation and type-24B CRC attachment, if any, turbo encoding, rate matching with RV, and code block concatenation. Additional information about the encoding process is returned in the fields of structure chinfo. The function is capable of processing both a single transport block or pairs of blocks, contained in a cell array, for the case of spatial multiplexing schemes transmitting two codewords. The type of the return variable, cwout, is the same as that of the input, trblkin. Thus, if trblkin is a cell array containing one or two transport blocks, cwout is a cell array of one or two codewords. If trblkin is a vector of information bits, cwout is a vector also. If encoding a pair of transport blocks, you must define pairs of modulation schemes and RV indicators in the appropriate parameter fields.

Examples

Generate DL-SCH Codewords

Generate the DL-SCH codeword for FDD duplexing mode as defined by RMC R.7 in [1].

```
rmc = lteRMCDL('R.7');
trBlkIn = randi([0,1],rmc.PDSCH.TrBlkSizes(1),1);
codeWord = lteDLSCH(rmc,rmc.PDSCH,rmc.PDSCH.CodedTrBlkSizes(1),trBlkIn);
codeWord(1:10)
```

```
     1
```

```
     0
     0
     1
     1
     1
     0
     0
     0
     0
```

Input Arguments

`enb` — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplex mode
The following parameters are dependent upon the condition that `DuplexMode` is set to `'TDD'`.			
`TDDConfig`	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration

Since the duplex mode, `DuplexMode`, defaults to `'FDD'`, if this field is absent, enb can be an empty structure.

`chs` — Channel configuration

structure

Channel configuration, specified as a structure. It defines aspects of the PDSCH onto which the codewords are mapped. It also defines the DL-SCH soft buffer size and redundancy versions of the generated codewords.

chs can contain the following fields.

Parameter Field	Required or Optional	Values	Description
`Modulation`	Required	`'QPSK'`, `'16QAM'`, `'64QAM'`, cell array of strings	Modulation type, specified as a string or cell array of strings. If 2 blocks, each cell is associated with a transport block.
`NLayers`	Required	1, ..., 8	Total number of transmission layers associated with the transport block or blocks.
`TxScheme`	Required	`'SpatialMux'` (default), `'Port0'`, `'TxDiversity'`, `'CDD'`, `'MultiUser'`, `'Port5'`, `'Port7-8'`, `'Port8'`, `'Port7-14'`	Transmission scheme, specified as one of the following options. • `'SpatialMux'` — Closed-loop spatial multiplexing. • `'Port0'` — Single-antenna port, port 0. • `'TxDiversity'` — Transmit diversity scheme. • `'CDD'` — Large delay CDD scheme. • `'MultiUser'` — Multiuser MIMO scheme. • `'Port5'` — Single-antenna port, port 5. • `'Port7-8'` — Single-antenna port, port 7 (**`NLayers`** = 1). Dual layer transmission, ports 7 and 8 (**`NLayers`** = 2). • `'Port8'` — Single-antenna port, port 8. • `'Port7-14'` — Up to 8-layer transmission, ports 7–14.
`RV`	Required	0, 1, 2, 3, 2-element integer vector	Redundancy version indicators, specified as a vector of 1 or 2 values. Each value can be an integer between 0 and 3.
`NSoftbits`	Optional	Nonnegative scalar integer (default 0)	Total number of soft buffer bits. The default setting of 0 signifies that there is no buffer limit.

`outlen` — Codeword length

numeric vector of one or two elements

Codeword length, specified as a numeric vector of one or two elements. This vector defines the codeword lengths to which the input transport blocks should be rate matched.

It represents the PDSCH capacity for the associated codeword. Therefore, it also represents the lengths of the vectors in cwout.

`trblkin` — Transport block information bits to be encoded

numeric vector | cell array of one or two numeric vectors

Transport block information bits to be encoded, specified as a numeric vector or a cell array of numeric vectors. trblkin is an input parameter containing the transport block information bits to be encoded. If it is a cell array, all rate matching calculations assume that the pair will be transmitting on a single PDSCH, distributed across the total number of layers defined in chs, as per [2]. The lowest order information bit of trblkin maps to the most significant bit of the transport block, as defined in section 6.1.1 of [3].

Output Arguments

`cwout` — DL-SCH encoded codewords

numeric column vector | cell array of one or two numeric column vectors

DL-SCH encoded codewords, returned as a numeric column vector or a cell array of one or two numeric column vectors. It reflects the data type and size of the input data, trblkin.

Data Types: `int8` | `cell`

`chinfo` — Additional information about encoding process

structure array | optional

Additional information about encoding process, returned as a structure array. It contains parameter fields related to code block segmentation and rate matching. If two transport blocks are encoded, chinfo is a structure array of two elements, with one element for each block. The code block segmentation fields in this structure can also be created independently using the `lteDLSCHInfo` function.

chinfo contains the following fields.

Parameter Field	Description	Values	Data Type
`C`	Total number of code blocks	Nonnegative scalar integer	`int32`

Parameter Field	Description	Values	Data Type
Km	Lower code block size (K–)	Nonnegative scalar integer	int32
Cm	Number of code blocks of size Km (C–)	Nonnegative scalar integer	int32
Kp	Upper code block size (K+)	Nonnegative scalar integer	int32
Cp	Number of code blocks of size Kp (C+)	Nonnegative scalar integer	int32
F	Number of filler bits in first block	Nonnegative scalar integer	int32
L	Number of segment cyclic redundancy check (CRC) bits	Nonnegative scalar integer	int32
Bout	Total number of bits in all segments	Nonnegative scalar integer	int32
Qm	Bits per symbol variable used in rate matching calculation	Nonnegative scalar integer	int32
NL	Number of layers used in rate matching calculation	Nonnegative scalar integer	int32
NLayers	Number of transmission layers	Nonnegative scalar integer	int32
NIR	Number of soft bits associated with transport block. Soft buffer size for entire input transport block	Nonnegative scalar integer	int32
RV	Redundancy version indicators, specified as a vector of 1 or 2 values. Each value can be an integer between 0 and 3.	0, 1, 2, 3, 2-element integer vector	int32

References

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.211. “Physical Channels and Modulation.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[3] 3GPP TS 36.321. “Medium Access Control (MAC) Protocol Specification.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDLSCHDecode` | `lteDLSCHInfo` | `ltePDSCH`

lteDLSCHDecode

Downlink shared channel decoding

Syntax

```
[trblkout,blkcrc,stateout] = lteDLSCHDecode(enb,chs,trblklen,cwin,
statein)
```

Description

`[trblkout,blkcrc,stateout] = lteDLSCHDecode(enb,chs,trblklen,cwin, statein)` returns the information bits, trblkout, decoded from the input soft LLR codeword data, cwin. The DL-SCH decoder includes rate recovery, turbo decoding, block concatenation, and CRC calculations. The function also returns the type-24A transport block CRC decoding result in blkcrc and the HARQ process decoding state in stateout. The initial HARQ process state can be provided as the optional statein parameter. The function is capable of processing both a single codeword or pairs of codewords, contained in a cell array, for the case of spatial multiplexing schemes transmitting two codewords. The type of the return variable, trblkout, is the same as the input, cwin. If cwin is a cell array containing one or two codewords, trblkout is a cell array of one or two transport blocks. If cwin is a vector of soft data, trblkout is a vector also. If you are decoding a pair of codewords, you must provide pairs of modulation schemes and RV indicators in the appropriate parameter fields.

enb is an input parameter structure that may include optional fields defining the duplex mode. Since the duplex mode defaults to `'FDD'`, if the `'DuplexMode'` field is absent, enb can be an empty structure.

chs is an input parameter structure defining aspects of the PDSCH onto which the codewords are mapped and the DL-SCH soft buffer size and redundancy versions of the received codewords.

trblklen is an input vector, one or two elements in length, defining the transport block lengths to which the input code blocks are rate recovered and decoded.

cwin is an input parameter containing the floating point soft LLR data of the codewords to be decoded. It is either a single vector or a cell array containing one or two vectors. If

it is a cell array, all rate matching calculations assume that the pair is transmitting on a single PDSCH, distributed across the total number of layers defined in chs, as per [1].

statein is an optional input structure array, empty or one or two elements, which can input the current decoder buffer state for each transport block in an active HARQ process. If statein is not an empty array and it contains a non-empty field, `CBSBuffers`, this field should contain a cell array of vectors representing the LLR soft buffer states for the set of code blocks at the input to the turbo decoder, after explicit rate recovery. The updated buffer states after decoding are returned in the `CBSBuffers` field in the output parameter, stateout. The statein array would normally be generated and recycled from the stateout of previous calls to `lteDLSCHDecode` as part of a sequence of HARQ transmissions.

trblkout is the output parameter containing the decoded information bits. It is either a single vector or a cell array containing one or two vectors, depending on the class and dimensionality of cwin.

blkcrc is an output array, one or two elements, containing the result of the type-24A transport block CRC decoding for the transport blocks.

stateout, the final output parameter, is a one- or two-element structure array containing the internal state of each transport block decoder. The stateout array is normally reapplied via the statein variable of subsequent `lteDLSCHDecode` function calls as part of a sequence of HARQ retransmissions.

Examples

Generate and Decode DL-SCH Transmissions

This example generates and decodes 2 transmissions, one with RV set to 0 and one with RV set to 1, as part of a single codeword HARQ process for RMC R.7.

First, get the definition of RMC R.7.

```
nsf = 1;
rmc = lteRMCDL('R.7');
```

Create a codeword with RV set to 0.

```
trBlkLen = rmc.PDSCH.TrBlkSizes(nsf);
```

```
trBlkData = randi([0,1],trBlkLen,1);
rmc.PDSCH.RV = 0;
cw = lteDLSCH(rmc,rmc.PDSCH,trBlkLen,trBlkData);
```

Turn all logical bits into log-likelihood ratio (LLR) data.

```
cw(cw==0) = -1;
```

Initialize the decoder states for the first HARQ transmission.

```
decState = [];
```

Decode the first DL-SCH transmission.

```
[rxTrBlk,~,decState] = lteDLSCHDecode(rmc, ...
    rmc.PDSCH,trBlkLen,cw,decState);
```

Next, create a second, retransmitted codeword with RV set to 1. Again, turn all logical bits into LLR data

```
rmc.PDSCH.RV = 1;
cw = lteDLSCH(rmc,rmc.PDSCH,trBlkLen,trBlkData);
cw(cw == 0) = -1;
```

Decode the second DL-SCH transmission.

```
rxTrBlk = lteDLSCHDecode(rmc,rmc.PDSCH, ...
    trBlkLen,cw,decState);
size(rxTrBlk);
rxTrBlk(1:10)
```

```
ans =

     1
     1
     0
     1
     1
     0
     0
     1
     1
     1
```

Input Arguments

`enb` — Cell-wide settings
scalar structure

Cell-wide settings, specified as a structure with the following fields.

`DuplexMode` — Duplex mode
`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, specified as a string. Optional. Since the duplex mode defaults to `'FDD'`, if this field is absent, enb can be an empty structure.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration
0 (default) | Optional | 0...6

Uplink or downlink configuration, specified as a nonnegative scalar integer between 0 and 6. Optional. Only required for `'TDD'` duplex mode.

Data Types: `double`

Data Types: `struct`

`chs` — Channel configuration
structure

Channel configuration, specified as a structure having the following fields.

`Modulation` — Modulation type associated with each transport block
`'QPSK'` | `'16QAM'` | `'64QAM'` | cell array of strings

Modulation type associated with each transport block, specified as a string or a cell array of strings if there are 2 blocks.

Data Types: `char` | `cell`

`NLayers` — Total number of layers in transport block
1 | 2 | 3 | 4

Total number of layers associated with the transport block, specified as a positive integer.

Data Types: `double`

`RV` — Redundancy version indicator

0 | 1 | 2 | 3 | 2-element numeric vector

Redundancy version indicator, specified as a numeric vector of 1 or 2 values. Possible values are 0, 1, 2, or 3.

Data Types: `double`

`TxScheme` — Transmission scheme

`'Port0'` | `'TxDiversity'` | `'CDD'` | `'SpatialMux'` | `'MultiUser'` | `'Port5'` | `'Port7-8'` | `'Port8'` | `'Port7-14'`

Transmission scheme, specified as a string. Its possible values and their descriptions are shown in the following table.

String	Description
`'Port0'`	Single-antenna port, Port 0
`'TxDiversity'`	Transmit diversity scheme
`'CDD'`	Large delay CDD scheme
`'SpatialMux'`	Closed-loop spatial multiplexing scheme
`'MultiUser'`	Multi-user MIMO scheme
`'Port5'`	Single-antenna port, Port 5
`'Port7-8'`	Single-antenna port, port 7 (when `NLayers`=1); Dual layer transmission, port 7 and 8 (when `NLayers`=2)
`'Port8'`	Single-antenna port, Port 8
`'Port7-14'`	Up to 8 layer transmission, ports 7-14

Data Types: `char`

`NSoftbits` — Total number of soft buffer bits

0 (default) | Optional | nonnegative scalar integer

Total number of soft buffer bits, specified as a nonnegative scalar integer. Optional. The default is 0 and signifies that there is no buffer limit. If `NSoftbits` is absent, no limit is placed on the number of soft bits.

Data Types: `double`

`NTurboDecIts` — Number of turbo decoder iteration cycles

5 (default) | Optional | positive scalar integer

Number of turbo decoder iteration cycles, specified as a positive scalar integer. Optional.

Data Types: `double`

Data Types: `struct`

`trblklen` — Transport block lengths

one- or two-element numeric vector

Transport block lengths, specified as a one- or two-element numeric vector. It defines the transport block lengths to which the input code blocks should be rate-recovered and decoded.

Data Types: `double` | `single` | `uint8` | `uint16` | `uint32` | `uint64` | `int8` | `int16` | `int32` | `int64`

`cwin` — Soft LLR codeword data

numeric vector | cell array of one or two numeric vectors

Soft LLR data of the codewords to be decoded, specified as either a numeric vector or a cell array containing one or two vectors.

Data Types: `double` | `single` | `uint8` | `uint16` | `uint32` | `uint64` | `int8` | `int16` | `int32` | `int64` | `cell`

`statein` — Initial HARQ process state

Optional | structure array

Initial HARQ process state, specified as a structure array. Optional. This structure array, which can be empty or contain one or two elements, can input the current decoder buffer state for each transport block in an active HARQ process.

Data Types: `struct`

Output Arguments

`trblkout` — Decoded information bits

numeric vector | cell array of one or two numeric vectors

Decoded information bits, returned as a numeric vector or a cell array of one or two numeric vectors. trblkout reflects the data type and size of cwin.

Data Types: `int8 | cell`

`blkcrc` — Type-24A transport block CRC decoding result

logical vector of one or two elements

Type-24A transport block CRC decoding result, returned as a logical vector of one or two elements.

Data Types: `logical`

`stateout` — HARQ process decoding state

structure array of one or two elements

HARQ process decoding state, returned as a structure array of one or two elements. It contains the internal state of each transport block in the following fields.

`CBSBuffers` — LLR soft buffer states

cell array of vectors

LLR soft buffer states, returned as a cell array of vectors representing the LLR soft buffer states for the set of code blocks associated with a single transport block. The buffers are positioned at the input to the turbo decoder, after explicit rate recovery.

Data Types: `cell`

`CBSCRC` — Code block set CRC

logical vector

Code block set CRC, returned as a logical vector. This argument is an array of type-24B code block set CRC decoding results.

Data Types: `logical`

`BLKCRC` — Type-24A transport block CRC decoding error

logical scalar

Type-24A transport block CRC decoding error, returned as a logical scalar.

Data Types: `logical`

Data Types: `struct`

References

[1] 3GPP TS 36.211. “Physical Channels and Modulation.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`lteDLSCH` | `lteDLSCHInfo` | `ltePDSCHDecode`

lteDLSCHInfo

DL-SCH segmentation information

Syntax

```
info = lteDLSCHInfo(blklen)
```

Description

`info = lteDLSCHInfo(blklen)` returns a structure, info, containing the Downlink Shared Channel (DL-SCH) code block segmentation information for given transport block length, blklen.

Examples

Display DL-SCH Segmentation Information

Find the segmentation information before coding for an input transport block of length 132.

```
info = lteDLSCHInfo(132)
```

```
info =

       C: 1
      Km: 0
      Cm: 0
      Kp: 160
      Cp: 1
       F: 4
       L: 0
    Bout: 160
```

The fields of the information structure, `info`, show that there are 4 filler bits and that the total size of one segment after CRC addition is 160.

Input Arguments

`blklen` — Transport block length

positive scalar integer

Transport block length, specified as a positive integer.

Data Types: `double`

Output Arguments

`info` — DL-SCH code block segmentation information

structure

DL-SCH code block segmentation information, returned as a structure including the following fields.

Parameter Field	Description	Values	Data Type
`C`	Total number of code blocks	Nonnegative scalar integer	`int32`
`Km`	Lower code block size (K–)	Nonnegative scalar integer	`int32`
`Cm`	Number of code blocks of size `Km` (C–)	Nonnegative scalar integer	`int32`
`Kp`	Upper code block size (K+)	Nonnegative scalar integer	`int32`
`Cp`	Number of code blocks of size `Kp` (C+)	Nonnegative scalar integer	`int32`
`F`	Number of filler bits in first block	Nonnegative scalar integer	`int32`
`L`	Number of segment cyclic redundancy check (CRC) bits	Nonnegative scalar integer	`int32`

Parameter Field	Description	Values	Data Type
`Bout`	Total number of bits in all segments	Nonnegative scalar integer	`int32`

See Also

`lteDLSCH` | `lteDLSCHDecode`

lteDMRS

UE-specific demodulation reference signals

Syntax

```
sym = lteDMRS(enb,chs)
sym = lteDMRS(enb,chs,opts)
```

Description

`sym = lteDMRS(enb,chs)` returns the downlink UE-specific reference signal (DM-RS) symbols for transmission in a single subframe and supports precoded and non-precoded DM-RS transmitted on DM-RS antenna ports p=5,7,8,and 7,...,(`NLayers`+6). These DM-RS are for use with Release 8, 9, and 10 non-codebook-based PDSCH transmission schemes. By default, the symbols are returned as a column vector and are always ordered as they should be mapped using `lteDMRSIndices` into a N-by-M-by-P array representing the subframe grid across either the non-precoded PDSCH layers or precoded transmit antennas. Other output representations can also be generated.

`lteDMRS` returns a column vector of length `NRE` containing the non-precoded or precoded DM-RS symbol sequences concatenated for all layers/ports or transmit antennas for the transmission scheme. Input parameters are the cell-wide settings structure, enb, and the PDSCH parameter structure, chs.

If the `TxScheme` parameter is not set to one of the schemes related to DM-RS, such as `'SpatialMux'`, sym is empty. If the scheme is single port, `NLayers` is 1 implicitly. If the W precoding matrix field is not present or is empty, as by default,sym contains only the concatenated non-precoded DM-RS symbols for the `NLayers` ports. Otherwise, sym contains all DM-RS symbol values after they are precoded using the `NLayers`-by-`NTxAnts` beamforming matrix, W, onto `NTxAnts` transmit antennas. They are ordered by the concatenation of DM-RS symbols per layer/port if not precoded or projected layers per transmit antenna is precoded.

`sym = lteDMRS(enb,chs,opts)` allows additional control of the contents and format of the symbols through a cell array, opts.

Examples

Assign Non-Precoded DM-RS Symbols to Grid

Generate and assign the non-precoded DM-RS symbols into an 8-layer subframe resource grid.

Generate a reference measurement channel R.1, with 10MHz, 1RB allocation.

```
enb = lteRMCDL('R.1');
```

Change cell-wide settings to support Release 10 transmission.

```
enb.PDSCH.TxScheme = 'Port7-14';
```

Generate and assign DM-RS. Clear the resource elements (RE) that should not be used because of the DM-RS on other ports.

```
enb.PDSCH.NLayers = 8;
subframe = ones(lteResourceGridSize(enb,enb.PDSCH.NLayers));
dmrsInd = lteDMRSIndices(enb,enb.PDSCH,'rs+unused');
dmrs = lteDMRS(enb,enb.PDSCH,'rs+unused');
subframe(dmrsInd) = dmrs;
```

Input Arguments

enb — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
NDLRB	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
NCellID	Required	Nonnegative scalar integer (0, …, 503)	Physical layer cell identity
NSubframe	Required	Nonnegative scalar integer	Subframe number

Parameter Field	Required or Optional	Values	Description
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplex mode
The following parameters are dependent upon the condition that `DuplexMode` is set to `'TDD'`.			
`TDDConfig`	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
`SSC`	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)

`chs` — PDSCH-specific channel transmission configuration

structure

PDSCH-specific channel transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`PRBSet`	Required	1- or 2-column integer matrix	0-based physical resource block (PRB) indices corresponding to the resource allocations for this PDSCH. As a column vector, the resource allocation is the same in both slots of the subframe. As a 2-column matrix, this parameter specifies different PRBs for each slot in a subframe.
`TxScheme`	Optional	`'Port5'` (default), `'Port7-8'`, `'Port8'`, `'Port7-14'`	DM-RS-specific transmission scheme, specified as one of the following options. • `'Port5'` — Rel-8 single-antenna port, port 5 (default) • `'Port7-8'` — Rel-9 single-antenna port, port 7 if `NLayers` is 1. Rel-9 dual-layer

Parameter Field	Required or Optional	Values	Description
			transmission, ports 7 and 8 if `NLayers` is 2. • `'Port8'` — Rel-9 single-antenna port, port 8 • `'Port7-14'` — Rel-10 up to 8 layer transmission, ports 7–14 if `NLayers` is 1,...,8.
`NLayers`	Optional	1 (default), 2, 3, 4, 5, 6, 7, 8	Number of transmission layers
`W`	Optional	Numeric matrix, `[]` (default)	Precoding matrix for the UE-specific beamforming of the DM-RS, of size `NLayers`-by-`NTxAnts`. An empty matrix, `[]`, signifies no precoding.
The following parameter is dependent upon the condition that `TxScheme` is set to `'Port7-8'`, `'Port8'`, or `'Port7-14'`.			
`NSCID`	Optional	0 (default), 1	Scrambling code identity (ID)
The following parameter is dependent upon the condition that `TxScheme` is set to `'Port5'`.			
`RNTI`	Required	Scalar integer	Radio network temporary identifier (RNTI) value (16 bits)

`opts` — Symbol generation option strings

string | cell array of strings

Symbol generation option strings, specified as a string or a cell array of strings that can contain the following values.

Option	Values	Description
Symbol style	`'ind'` (default), `'mat'`	Style for returning DM-RS symbols, specified as one of the following options. • `'ind'` — returns the DM-RS symbols as a column vector (default) • `'mat'` — returns the DM-RS symbols as a matrix. If not precoded, each column contains symbols for an individual port or layer. To form a matrix, a column may contain duplicate entries.

Option	Values	Description
Symbol format	`'rsonly'` (default), `'rs+unused'`	Format for returning DM-RS symbols, specified as one of the following options. • `'rsonly'` — returns only active DM-RS symbols (default) • `'rs+unused'` — returns include zeros for the RE locations that should be unused because of DM-RS transmission on another port or layer. This format is equivalent to precoding with `W` set to `eye(NLayers)`.

Data Types: `char` | `cell`

Output Arguments

`sym` — DM-RS symbol sequences

numeric column vector | numeric matrix

DM-RS symbol sequences, returned as a numeric column vector or a numeric matrix. Its length is the number of resource elements (NRE). It contains the non-precoded or precoded DM-RS symbol sequences concatenated for all layers or ports or transmit antennas for the transmission scheme.

Data Types: `double`
Complex Number Support: Yes

See Also

`lteCellRS` | `lteCSIRS` | `lteDMRSIndices` | `ltePDSCH` | `ltePRS` | `lteSRS`

lteDMRSIndices

UE-specific DM-RS resource element indices

Syntax

```
ind = lteDMRSIndices(enb,chs)
ind = lteDMRSIndices(enb,chs,opts)
```

Description

`ind = lteDMRSIndices(enb,chs)` returns the indices of the downlink UE-specific reference signal (DM-RS) resource elements (RE) in a subframe and supports precoded and non-precoded DM-RS transmitted on antenna ports, p=5,7,8, and 7,...,(`NLayers`+6). By default, ind is a column vector of indices in 1-based linear indexing form that can directly index the DM-RS elements in a N-by-M-by-P array, representing the subframe grid across either the non-precoded PDSCH layers or precoded transmit antennas. Alternative index representations can also be generated. The order of the indices is the same as how the complex DM-RS symbols should be mapped. The DM-RS symbols are generated by `lteDMRS` and do not include any elements allocated to PBCH, PSS, and SSS. If the subframe contains no DM-RS, an empty vector is returned.

The output, ind, is a column vector of 1-based linear indices for the precoded or non-precoded DM-RS resource elements in the subframe, given the cell-wide settings parameter structure, enb and the PDSCH parameter structure, chs. Its length is the number of resource elements (NRE).

If the `TxScheme` parameter is not set to one of the DM-RS related schemes, such as `'Port0'`, ind is empty. If the `NTxAnts` parameter field is not present or is set to 0, as by default, ind contains only the indices of the actual DM-RS REs in the non-precoded N-by-M-by-`NLayers` resource grid. Otherwise, ind contains the locations of all precoded DM-RS symbol values after they are precoded into a N-by-M-by-`NTxAnts` transmit antenna grid. Since precoding projects the DM-RS in each PDSCH layer onto all `NTxAnts` transmit antennas, ind contains the concatenation of all DM-RS locations across all layers, which are then duplicated in all `NTxAnts` planes of the 3-D grid.

`ind = lteDMRSIndices(enb,chs,opts)` allows additional control of the contents and format of the indices through a cell array of option strings, opts.

Examples

Generate Zero-Based DM-RS Indices

Get 0-based DM-RS resource element indices in subscript form.

```
enb = lteRMCDL('R.26');
ind = lteDMRSIndices(enb,enb.PDSCH,{'0based','sub'});
ind(1:4,:)
```

```
ans =

           0           3           0
           4           3           0
           8           3           0
          12           3           0
```

The output, `ind`, is a matrix in which each row has three columns. The first, second, and third columns represent subcarrier, symbol, and antenna port, respectively.

Input Arguments

`enb` — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`NSubframe`	Required	Nonnegative scalar integer	Subframe number
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplex mode

Parameter Field	Required or Optional	Values	Description
The following parameters are dependent upon the condition that `DuplexMode` is set to `'TDD'`.			
`TDDConfig`	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
`SSC`	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)
The following parameter is only required when `TxScheme` is set to `'Port5'`.			
`NCellID`	Required	Nonnegative scalar integer (0, ..., 503)	Physical layer cell identity

`chs` — PDSCH-specific channel transmission configuration

structure

PDSCH-specific channel transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`PRBSet`	Required	1- or 2-column integer matrix	0-based physical resource block (PRB) indices corresponding to the resource allocations for this PDSCH. As a column vector, the resource allocation is the same in both slots of the subframe. As a 2-column matrix, this parameter specifies different PRBs for each slot in a subframe.
`TxScheme`	Optional	`'Port5'` (default), `'Port7-8'`, `'Port8'`, `'Port7-14'`	DM-RS-specific transmission scheme, specified as one of the following options. • `'Port5'` — Rel-8 single-antenna port, port 5 (default) • `'Port7-8'` — Rel-9 single-antenna port, port 7 if `NLayers` is

Parameter Field	Required or Optional	Values	Description
			1. Rel-9 dual-layer transmission, ports 7 and 8 if `NLayers` is 2. • `'Port8'` — Rel-9 single-antenna port, port 8 • `'Port7-14'` — Rel-10 up to 8 layer transmission, ports 7–14 if `NLayers` is 1,...,8.
`NLayers`	Optional	1 (default), 2, 3, 4, 5, 6, 7, 8	Number of transmission layers
`NTxAnts`	Optional	0 (default), 1, 2, ...	Number of transmission antenna ports. This argument is only present for UE-specific demodulation reference symbols.

`opts` — Index generation options

string | cell array of strings

Index generation options, specified as a string or a cell array of strings that can contain the following values.

Option	Values	Description
Indexing style	`'ind'` (default), `'mat'`, `'sub'`	Style for the returned indices, specified as one of the following options. • `'ind'` — returns the indices as a column vector (default) • `'mat'` — returns the indices as a matrix. Each column contains symbols for an individual port. To form a matrix, a column can contain duplicate entries. • `'sub'` — returns the indices in `[subcarrier, symbol, antenna]` subscript row style. The number of rows in the output, ind, is the number of resource elements (NRE). Thus, ind is an NRE-by-3 matrix.
Index base	`'1based'` (default), `'0based'`	Base value of the returned indices. Specify `'1based'` to generate indices where the first value is one. Specify `'0based'` to generate indices where the first value is zero.

Option	Values	Description
Indexing format	`'rsonly'` (default), `'rs+unused'`	Format for the returned indices, specified as one of the following options. • `'rsonly'` — returns only active DM-RS symbols (default) • `'rs+unused'` — also includes zeros for the resource element (RE) locations that should be unused because of DM-RS transmission on another port or layer. This format is equivalent to precoding with `NTxAnts` set to `NLayers`.

Data Types: `char` | `cell`

Output Arguments

`ind` — DM-RS resource element indices

numeric column vector | numeric 3-column matrix

DM-RS resource element indices, returned as a numeric column vector or a numeric 3-column matrix. By default, it is a column vector whose length is the number of resource elements (NRE), and the indices are in 1-based linear indexing form. It can directly index the DM-RS elements in a *N*-by-*M*-by-*P* array, representing the subframe grid across either the non-precoded PDSCH layers or precoded transmit antennas.

Data Types: `uint32`

See Also

`lteCellRSIndices` | `lteCSIRSIndices` | `lteDMRS` | `ltePRSIndices` | `lteSRSIndices`

lteDuplexingInfo

Duplexing information

Syntax

```
info = lteDuplexingInfo(enb)
```

Description

`info = lteDuplexingInfo(enb)` returns a structure, info, providing information on the duplexing arrangement.

Examples

Get Duplexing Information

Get duplexing information to determine if a subframe is used for downlink transmission.

Determine if a subframe is used for downlink transmission by observing whether it has a nonzero number of symbols available for transmission.

```
enb = lteRMCDL('R.0');
duplexingInfo = lteDuplexingInfo(enb);
duplexingInfo.NSymbolsDL
```

```
          14
```

The number of symbols used for transmission in the downlink, `info.NSymbolsDL` is nonzero for all subframes in `FDD` and for downlink or special subframes in `TDD`. Thus, this subframe is used for downlink transmission.

Input Arguments

`enb` — Cell-wide settings

structure

Cell-wide settings, specified as a structure with these fields. The TDDConfig, SSC, and NSubframe parameter fields are only required if DuplexMode is set to `'TDD'`.

`CyclicPrefix` — Cyclic prefix length in downlink
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length in downlink, specified as a string. Optional.

Data Types: `char`

`CyclicPrefixUL` — Cyclic prefix length in uplink
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length in uplink, specified as a string. Optional.

Data Types: `char`

`DuplexMode` — Duplexing mode
`'FDD'` (default) | Optional | `'TDD'`

Duplexing mode, specified as a string. Optional.

Data Types: `double`

`TDDConfig` — Uplink or downlink configuration
0 (default) | Optional | nonnegative scalar integer (0...6)

Uplink or downlink configuration, specified as a nonnegative scalar integer between 0 and 6. Optional. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

`SSC` — Special subframe configuration
0 (default) | Optional | nonnegative scalar integer (0...9)

Special subframe configuration, specified as a nonnegative scalar integer between 0 and 9. Optional. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

`NSubframe` — Subframe number
nonnegative scalar integer

Subframe number, specified as a nonnegative scalar integer. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

Data Types: `struct`

Output Arguments

`info` — Duplexing information

structure

Duplexing information, returned as a structure having the following fields.

`NSymbols` — Total number of symbols in subframe

nonnegative scalar integer

Total number of symbols in subframe, returned as a nonnegative scalar integer.

Data Types: `int32`

`SubframeType` — Type of subframe

`'Downlink' | 'Uplink' | 'Special'`

Type of subframe, returned as a string.

Data Types: `char`

`NSymbolsDL` — Number of symbols used for transmission in downlink

nonnegative scalar integer

Number of symbols used for transmission in downlink (DL), returned as a nonnegative scalar integer.

Data Types: `int32`

`NSymbolsGuard` — Number of symbols in the guard period

nonnegative scalar integer

Number of symbols in the guard period, returned as a nonnegative scalar integer.

Data Types: `int32`

`NSymbolsUL` — Number of symbols used for transmission in uplink

nonnegative scalar integer

Number of symbols used for transmission in uplink (UL), returned as a nonnegative scalar integer.

Data Types: `int32`

See Also

`lteDLResourceGrid` | `lteResourceGrid` | `lteULResourceGrid`

lteEPDCCH

Enhanced physical downlink control channel

Syntax

```
sym = lteEPDCCH(enb,chs,cw)
```

Description

`sym = lteEPDCCH(enb,chs,cw)` returns a vector sym of complex modulation symbols associated with a single Enhanced Physical Downlink Control Channel (EPDCCH) transmission in a subframe. The channel processing includes the stages of scrambling and QPSK modulation. The function is initialized according to the cell-wide settings enb and the channel transmission configuration chs. For a given input bit vector cw, the column output sym contains the QPSK symbols ready to be mapped into the resource elements indicated by `lteEPDCCHIndices`.

This function does not perform any precoding. If required, apply precoding externally.

You can obtain the EPDCCH transmission capacity from the info structure produced by `lteEPDCCHIndices`.

Examples

Generate Complex Modulated EPDCCH Symbols

Specify the cell-wide settings and channel transmission configuration in parameter structures `enb` and `chs`.

```
enb.NSubframe = 4;
chs.EPDCCHNID = 7;
```

Generate EPDCCH symbols by encoding the input `cw` into QPSK symbols.

```
cw = randi([0 1],100,1);
sym = lteEPDCCH(enb,chs,cw);
```

Display the size and the first 10 indices of `sym`. Because these are QPSK symbols, `sym` contains half as many symbols as the number of bits that can be transmitted on the EPDCCH.

```
size(sym)
sym(1:10)
```

```
ans =

    50     1


ans =

  -0.7071 + 0.7071i
   0.7071 + 0.7071i
  -0.7071 + 0.7071i
  -0.7071 - 0.7071i
  -0.7071 + 0.7071i
   0.7071 - 0.7071i
  -0.7071 - 0.7071i
  -0.7071 - 0.7071i
  -0.7071 - 0.7071i
   0.7071 - 0.7071i
```

Input Arguments

`enb` — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure. This argument must contain the following parameter field.

`NSubframe` — Subframe number

Nonnegative scalar integer

Subframe number, specified as a nonnegative scalar integer.

Data Types: `double`

Data Types: `struct`

`chs` — Channel-specific transmission configuration

structure

Channel-specific transmission configuration, specified as a structure. This argument must contain the following parameter field.

`EPDCCHNID` — EPDCCH scrambling sequence initialization

nonnegative scalar integer

EPDCCH nID parameter for scrambling sequence initialization, specified as a nonnegative scalar integer.

Data Types: `double`

Data Types: `struct`

`cw` — Input bit vector

vector

Input bit vector containing the bit values of the EPDCCH codeword for modulation.

Output Arguments

`sym` — EPDCCH modulation symbols

Complex-vector

EPDCCH modulation symbols, given the input bit vector cw, returned as a vector of complex modulation symbols associated with a single EPDCCH transmission in a subframe. sym contains the QPSK symbols ready to be mapped into the resource elements indicated by `lteEPDCCHIndices`.

See Also

`lteDCIEncode` | `lteEPDCCHIndices` | `lteEPDCCHPRBS` | `ltePDCCH`

lteEPDCCHDMRS

EPDCCH demodulation reference signals

Syntax

```
sym = lteEPDCCHDMRS(enb,chs)
sym = lteEPDCCHDMRS(enb,chs,opts)
```

Description

`sym = lteEPDCCHDMRS(enb,chs)` returns the Enhanced Physical Downlink Control Channel Demodulation Reference Signal (EPDCCH DM-RS) symbols for transmission in a single subframe. By default the symbols are returned as a column vector. The order of the symbols is the same as the order that results when you use `lteEPDCCHDMRSIndices` to map them into an *N*-by-*M*-by-4 array. This array represents the resource element subframe grid across the 4 possible EPDCCH antenna ports (p = 107...110).

The symbols are parameterized in terms of a configured PRB pair set which defines:

- The overall set of possible EPDCCH candidates.
- The aggregation of one or more consecutive enhanced control channel elements (ECCE). This aggregation identifies the specific EPDCCH instance that the DM-RS is associated with.

The DM-RS symbols are created only for the specific PRB pairs and antenna ports that the corresponding EPDCCH is mapped to.

For a localized EPDCCH transmission, the EPDCCH are associated with a single antenna port from p = 107...110, dependent on the `RNTI` and ECCEs selected. Thus, the DM-RS antenna port symbols are output only for that single port.

For a distributed transmission, the EPDCCH is mapped to two antenna ports in an alternating fashion. Therefore, the DM-RS symbols are generated for the PRBs in both ports: p = 107,109 for normal cyclic prefix and p = 107,108 for extended cyclic prefix. The output is ordered so that the symbols for the lowest antenna ports index come first. This order matches that of the DM-RS RE indices produced by `lteEPDCCHDMRSIndices`.

`sym = lteEPDCCHDMRS(enb,chs,opts)` enables additional control over the contents and format of the symbols through a cell array of option strings, opts. You can use this syntax to return the symbols as a numeric matrix, where each column contains symbols for an active antenna port.

This function does not perform any precoding. If required, apply precoding externally.

Examples

Generate EPDCCH DM-RS Symbols

Specify the cell-wide settings and channel transmission configuration in parameter structures enb and chs.

```
enb.NDLRB = 6;
enb.NSubframe = 0;
chs.EPDCCHECCE = [0 7];
chs.EPDCCHType = 'Localized';
chs.EPDCCHPRBSet = 2:3;
chs.EPDCCHNID = 0;
chs.RNTI = 1;
```

Generate EPDCCH demodulation reference signal symbols.

```
sym = lteEPDCCHDMRS(enb,chs)
```

```
sym

  0.7071 - 0.7071i
  0.7071 + 0.7071i
  0.7071 + 0.7071i
 -0.7071 + 0.7071i
  0.7071 - 0.7071i
  0.7071 - 0.7071i
  0.7071 + 0.7071i
  0.7071 - 0.7071i
  0.7071 + 0.7071i
 -0.7071 - 0.7071i
  0.7071 - 0.7071i
  0.7071 - 0.7071i
```

Generate DM-RS Symbols for EPDCCH Having a Distributed Transmission

Specify the cell-wide settings and channel transmission configuration in parameter structures enb and chs.

```
enb.NDLRB = 6;
enb.NSubframe = 0;
chs.EPDCCHECCE = [0,7];
chs.EPDCCHType = 'Distributed';
chs.EPDCCHPRBSet = 2:3;
chs.EPDCCHNID = 0;
chs.RNTI = 1;
```

Generate DM-RS symbols for an EPDCCH having a distributed transmission. Return the symbols as a matrix, where each column contains symbols for an active antenna.

```
sym = lteEPDCCHDMRS(enb,chs,'mat')
```

```
sym =

   0.7071 - 0.7071i   0.7071 - 0.7071i
  -0.7071 - 0.7071i  -0.7071 - 0.7071i
   0.7071 + 0.7071i   0.7071 + 0.7071i
   0.7071 - 0.7071i   0.7071 - 0.7071i
   0.7071 - 0.7071i   0.7071 - 0.7071i
  -0.7071 + 0.7071i  -0.7071 + 0.7071i
  -0.7071 - 0.7071i  -0.7071 - 0.7071i
   0.7071 - 0.7071i   0.7071 - 0.7071i
  -0.7071 - 0.7071i  -0.7071 - 0.7071i
  -0.7071 - 0.7071i  -0.7071 - 0.7071i
  -0.7071 + 0.7071i  -0.7071 + 0.7071i
   0.7071 - 0.7071i   0.7071 - 0.7071i
```

Input Arguments

enb — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
`NSubframe`	Required	Nonnegative scalar integer	Subframe number
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplexing mode, specified as one of the following: • `'FDD'` — Frequency division duplex (default) • `'TDD'` — Time division duplex
The following parameters apply when `DuplexMode` is set to `'TDD'`.			
`TDDConfig`	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
`SSC`	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)

`chs` — Channel-specific channel transmission configuration

structure

Channel-specific transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`EPDCCHECCE`	Required	1- or 2-element vector specifying the 0-based ECCE index or inclusive [`begin`, `end`] ECCE index range according to the aggregation level `L` (`L = end – begin + 1`).	The set of one or several consecutive ECCEs defining the EPDCCH transmission candidate in the overall EPDCCH set.

Parameter Field	Required or Optional	Values	Description
		The number of ECCEs in the candidate must be a power of 2. If no transmission is required, leave this parameter empty.	
`EPDCCHType`	Required	`'Localized'`, `'Distributed'`	EPDCCH transmission type
`EPDCCHPRBSet`	Required	Vector of 0-based indices for the PRB pairs corresponding to the EPDCCH PRB set. The number of PRB pair indices must be a power of 2. If no transmission is required, leave this parameter empty.	EPDCCH PRB pair indices
`EPDCCHNID`	Required	Nonnegative scalar integer	EPDCCH nID parameter for scrambling sequence initialization
The following parameters apply when `EPDCCHType` is set to `'Localized'`.			
`RNTI`	Required	Scalar integer	Radio network temporary identifier (RNTI) value (16 bits)

`opts` — Symbol generation option strings

string | cell array of strings

Symbol generation option strings, specified as a string or a cell array of strings that can contain the following values.

Option	Values	Description
Symbol style	`'ind'` (default), `'mat'`	Style for the returned symbols, specified as one of the following: • `'ind'` — returns the symbols as a column vector (default) • `'mat'` — returns the symbols as a matrix in which each column contains symbols for an active antenna port from the set p = 107...110
Symbol format	`'rsonly'` (default), `'rs+unused'`	Format of the returned symbols. • `'rsonly'` — returns only active DM-RS symbols (default) • `'rs+unused'` — also returns zeros for the RE locations, which should be unused because of EPDCCH DM-RS transmissions on other antenna ports

Data Types: `char` | `cell`

Output Arguments

`sym` — EPDCCH DM-RS symbols

numeric column vector | numeric matrix

EPDCCH demodulation reference signal symbols, returned as a column vector containing the non-precoded DM-RS symbol sequences concatenated for all active PRB pairs and antenna ports. Optionally, the function returns sym as a numeric matrix, where each column contains symbols for an active antenna port.

Data Types: `double`

See Also

`lteCellRS` | `lteCSIRS` | `lteEPDCCH` | `lteEPDCCHDMRSIndices` | `lteEPDCCHPRBS`

lteEPDCCHDMRSIndices

EPDCCH DM-RS resource element indices

Syntax

```
ind = lteEPDCCHDMRSIndices(enb,chs)
ind = lteEPDCCHDMRSIndices(enb,chs,opts)
```

Description

`ind = lteEPDCCHDMRSIndices(enb,chs)` returns the indices of the Enhanced Physical Downlink Control Channel Demodulation Reference Signal (EPDCCH DM-RS) resource elements (RE) associated with an EPDCCH transmission candidate in a subframe. By default, ind is a column vector of indices in 1-based linear indexing form. You can use this indexing form to directly index the EPDCCH DM-RS REs of an *N*-by-*M*-by-4 array that represents the subframe resource grid across the 4 possible EPDCCH antenna ports (p = 107...110). You can also generate alternative index representations. The order of the indices is the same as required for the complex EPDCCH DM-RS symbols mapping. These symbols are generated by `lteEPDCCHDMRS`.

The indices are parameterized in terms of a configured PRB pair set which defines:

- The overall set of possible EPDCCH candidates.
- The aggregation of one or more consecutive enhanced control channel elements (ECCE). This aggregation identifies the specific EPDCCH instance that the DM-RS will be associated with.

The DM-RS indices are created only for the specific PRB pairs and antenna ports that the corresponding EPDCCH is mapped to. They do not account for any external precoding operations.

For a localized EPDCCH transmission, the EPDCCH are associated with a single antenna port from p = 107...110, dependent on the `RNTI` and ECCEs selected. Thus, the DM-RS antenna port indices (1...4 respectively if 1-based) are output for that single port.

For a distributed transmission, the EPDCCH is mapped to two antenna ports in an alternating fashion. Therefore, the DM-RS indices are generated for the PRBs in both

ports: p = 107,109 for normal cyclic prefix and p = 107,108 for extended cyclic prefix. The output is ordered so that the symbols for the lowest antenna index plane come first. These indices are suitable for indexing an N-by-M-by-4 array that represents the subframe resource grid across the 4 possible EPDCCH antenna ports (p = 107...110).

This syntax returns an NRE length column vector of 1-based linear indices for the DM-RS resource elements associated with a particular EPDCCH candidate. The function is initialized according to the cell-wide settings enb and the EPDCCH transmission configuration chs.

`ind = lteEPDCCHDMRSIndices(enb,chs,opts)` enables additional control over the contents and format of the indices through a cell array of option strings, opts.

Examples

Generate EPDCCH DM-RS Indices

Specify the cell-wide settings and channel transmission configuration in parameter structures `enb` and `chs`.

```
enb = struct('CyclicPrefix','Normal','DuplexMode','FDD');
enb.NDLRB = 6;
enb.NSubframe = 0;
chs.EPDCCHECCE = [0 7];
chs.EPDCCHType = 'Localized';
chs.EPDCCHPRBSet = 2:3;
chs.RNTI = 1;
```

Create the EPDCCH DM-RS indices for an EPDCCH having eight ECCEs.

```
ind = lteEPDCCHDMRSIndices(enb,chs)
```

```
ind =

        1898
        1903
        1908
        1910
        1915
        1920
        1970
        1975
```

```
1980
1982
1987
1992
```

Input Arguments

enb — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
NDLRB	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
CyclicPrefix	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
NSubframe	Required	Nonnegative scalar integer	Subframe number
DuplexMode	Optional	`'FDD'` (default), `'TDD'`	Duplexing mode, specified as one of the following: • `'FDD'` — Frequency division duplex (default) • `'TDD'` — Time division duplex
The following parameters apply when `DuplexMode` is set to `'TDD'`.			
TDDConfig	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
SSC	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)

chs — Channel-specific transmission configuration

structure

Channel-specific transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`EPDCCHECCE`	Required	1- or 2-element vector specifying the 0-based ECCE index or inclusive [`begin`, `end`] ECCE index range according to the aggregation level `L` (`L = end – begin + 1`). The number of ECCEs in the candidate must be a power of 2. If no transmission is required, leave this parameter empty.	The set of one or several consecutive ECCEs defining the EPDCCH transmission candidate in the overall EPDCCH set.
`EPDCCHType`	Required	`'Localized'`, `'Distributed'`	EPDCCH transmission type
`EPDCCHPRBSet`	Required	Vector of 0-based indices for the PRB pairs corresponding to the EPDCCH PRB set. The number of PRB pair indices must be a power of 2. If no transmission is required, leave this parameter empty.	EPDCCH PRB pair indices
The following parameters apply when `EPDCCHType` is set to `'Localized'`.			
`RNTI`	Required	Scalar integer	Radio network temporary identifier (RNTI) value (16 bits)

`opts` — Index generation options

string | cell array of strings

Index generation options, specified as a string or a cell array of strings that can contain the following values.

Option	Values	Description
Indexing style	`'ind'` (default), `'mat'`, `'sub'`	Style for the returned indices, specified as one of the following: • `'ind'` — returns the indices in linear index form as a column vector (default) • `'mat'` — returns the indices in linear index form as a matrix, where each column contains indices for an individual port. • `'sub'` — returns the indices in `[subcarrier, symbol, antenna]` subscript row style. The number of rows in the output, ind, is the number of resource elements (NRE). Thus, ind is an NRE-by-3 matrix.
Index base	`'1based'` (default), `'0based'`	Base value of the returned indices. Specify `'1based'` to generate indices where the first value is one. Specify `'0based'` to generate indices where the first value is zero.
Indexing format	`'rsonly'` (default), `'rs+unused'`	RE locations mode of the returned indices. • `'rsonly'` — returns only active DM-RS locations (default) • `'rs+unused'` — also includes all RE locations, which should be unused because of DM-RS transmission on another antenna ports

Data Types: `char` | `cell`

Output Arguments

`ind` — EPDCCH DM-RS RE indices

numeric column vector | numeric matrix

EPDCCH DM-RS resource element indices, returned by default as a numeric vector of length NRE-by-1. Optionally, for subscript-specific indexing style [`subcarrier`, `symbol`, `antenna`], ind is returned as an NRE-by-3 numeric matrix. NRE is the number of subframe resource elements. You can also return the indices in a linear indexing matrix, where each column contains indices for an individual antenna port. By default,

the indices are returned in 1-based linear indexing form, which you can use to directly index the EPDCCH DM-RS resource elements.

Data Types: `double`

See Also

`lteDMRSIndices` | `lteEPDCCHDMRS` | `lteEPDCCHIndices`

lteEPDCCHIndices

Enhanced physical downlink control channel (EPDCCH) resource element indices

Syntax

```
[ind,info] = lteEPDCCHIndices(enb,chs)
[ind,info] = lteEPDCCHIndices(enb,chs,opts)
```

Description

`[ind,info] = lteEPDCCHIndices(enb,chs)` returns the subframe resource element (RE) indices for the Enhanced Physical Downlink Control Channel (EPDCCH). By default, ind is a vector of indices in a 1-based linear indexing style. You can use this style of indexing to directly index elements of an *N*-by-*M*-by-4 array representing the subframe resource grid across the 4 possible EPDCCH antenna ports (p = 107...110). The indices are for a single transmission instance of the EPDCCH. The order of the indices is the same as required for the complex EPDCCH symbols mapping. These symbols are generated by `lteEPDCCH`.

The indices are parameterized in terms of a configured PRB pair set that defines:

- The overall set of possible EPDCCH candidates.
- The aggregation of one or more consecutive enhanced control channel elements (ECCE). This aggregation identifies the specific EPDCCH instance within the set of EPDCCH candidates.

The EPDCCH can use either localized or distributed transmission, differing in the mapping of ECCEs to REs, active PRB pairs, and antenna ports.

This syntax returns an NRE-by-1 vector of 1-based linear indexing RE indices, given the cell-wide settings structure enb and the EPDCCH transmission configuration chs.

`[ind,info] = lteEPDCCHIndices(enb,chs,opts)` enables additional control over the format of the returned indices through a cell array of option strings, opts.

Examples

Generate RE Indices of Localized Transmission

This example generates RE Indices of localized transmission in default and subscripted formats.

Specify the cell-wide settings in parameter structure, `enb`.

```
enb.NDLRB = 6;
enb.NSubframe = 0;
enb.NCellID = 0;
enb.CellRefP = 1;
enb.CyclicPrefix = 'Normal';
enb.DuplexMode = 'FDD';
enb.NFrame = 0;
enb.CSIRSPeriod = 'Off';
enb.ZeroPowerCSIRSPeriod = 'Off';
```

Specify the channel transmission configuration in parameter structure, `chs`.

```
chs.EPDCCHECCE = [0 7];
chs.EPDCCHType = 'Localized';
chs.EPDCCHPRBSet = 2:3;
chs.EPDCCHStart = 2;
chs.RNTI = 1;
```

Generate 1-based linear resource element indices of a localized transmission.

```
[ind,info] = lteEPDCCHIndices(enb,chs);
size(ind)
```

```
ans =

   228     1
```

Display the size and the first 10 indices of `ind`.

```
ind(1:10)
```

```
ans =

        1177
        1178
        1179
        1180
        1181
        1182
        1183
        1184
        1185
        1186
```

Generate 1-based resource element indices in the subscript format [`subcarrier`, `symbol`, `antenna`].

```
[ind,info] = lteEPDCCHIndices(enb,chs,'sub');
size(ind)
```

```
ans =

   228     3
```

Display the size and the first 10 indices of `ind`.

```
ind(1:10,:)
```

```
ans =

          25           3           2
          26           3           2
          27           3           2
          28           3           2
          29           3           2
          30           3           2
          31           3           2
          32           3           2
          33           3           2
          34           3           2
```

Input Arguments

enb — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
NDLRB	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
NCellID	Required	Nonnegative scalar integer (0, ..., 503)	Physical layer cell identity
CyclicPrefix	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
CellRefP	Required	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
NSubframe	Required	Nonnegative scalar integer	Subframe number
The following parameters apply when `chs.EPDCCHStart` is absent.			
CFI	Required	1, 2, 3	Control format indicator (CFI) value
DuplexMode	Optional	`'FDD'` (default), `'TDD'`	Duplexing mode, specified as one of the following: • `'FDD'` — Frequency division duplex (default) • `'TDD'` — Time division duplex
The following parameters apply when `DuplexMode` is set to `'TDD'`.			
TDDConfig	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration

Parameter Field	Required or Optional	Values	Description
SSC	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)
NFrame	Optional	0 (default), Nonnegative scalar integer	Frame number
CSIRSPeriod	Optional	'Off' (default), 'On', Icsi-rs, (0,...,154), [Tcsi-rs Dcsi-rs]	CSI-RS subframe configuration
The following parameters apply when CSIRSPeriod is set to any value but 'Off'.			
CSIRSConfig	Required	Nonnegative scalar integer	CSI-RS configuration index. See table 6.10.5.2-1 in TS 36.211.
CSIRefP	Required	1 (default), 2, 4, 8	Number of CSI-RS antenna ports
ZeroPowerCSIRS	Optional	'Off' (default), 'On', Icsi-rs, (0,...,154), [Tcsi-rs Dcsi-rs]	Zero-power CSI-RS subframe configuration
The following parameters apply when ZeroPowerCSIRSPeriod is set to any value but 'Off'.			
ZeroPowerCS	Required	16-bit bitmap string (truncated if not 16 bits or '0' MSB extended), numerical list of CSI-RS configuration indices. See table 6.10.5.2-1 (4 CSI reference signal column) in TS 36.211.	Zero-power CSI-RS configuration index list. See table 6.10.5.2 in TS 36.211.

chs — EPDCCH-specific channel transmission configuration

structure

EPDCCH-specific channel transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
EPDCCHECCE	Required	1- or 2- element vector specifying the 0-based ECCE index	The set of one of several consecutive ECCEs defining the EPDCCH

Parameter Field	Required or Optional	Values	Description
		or inclusive [`begin`, `end`] ECCE index range according to the aggregation level `L` (`L = end – begin + 1`). The number of ECCEs in the candidate must be a power of 2. If no transmission is required, leave this parameter empty.	transmission candidate in the overall EPDCCH set.
`EPDCCHType`	Required	`'Localized'`, `'Distributed'`	EPDCCH transmission type
`EPDCCHPRBSet`	Required	Vector of 0-based indices for the PRB pairs corresponding to the EPDCCH PRB set. The number of PRB pair indices must be a power of 2. If no transmission is required, leave this parameter empty.	EPDCCH PRB pair indices
`EPDCCHStart`	Optional	0...14 If this parameter is not present, then the cell-wide CFI parameter is used for the starting symbol.	EPDCCH starting symbol
The following parameters apply when `EPDCCHType` is set to `'Localized'`.			
`RNTI`	Required only for the `'Localized'` transmission type	Scalar integer	Radio network temporary identifier (RNTI) value (16 bits)

`opts` — Index generation options

string | cell array of strings

Index generation options, specified as a string or a cell array of strings that can contain the following values.

Option	Values	Description
Indexing style	`'ind'` (default), `'sub'`	Style for the returned indices, specified as one of the following options. • `'ind'` — returns the indices in linear index form as a column vector (default) • `'sub'` — returns the indices in `[subcarrier, symbol, antenna]` subscript row style. The number of rows in the output, ind, is the number of resource elements (NRE). Thus, ind is an NRE-by-3 matrix.
Index base	`'1based'` (default), `'0based'`	Base value of the returned indices. Specify `'1based'` to generate indices where the first value is one. Specify `'0based'` to generate indices where the first value is zero.

Whether in linear or subscript format style, the indices are always formed out of `[subcarrier, symbol, antenna]` subscripts. These subscripts identify the used resource elements in each subframe resource grid per antenna port.

For the EPDCCH, the antenna subscripts have the possible range 1...4 (if `1-based`) which represents antenna ports p = 107...110. For a localized EPDCCH transmission, the antenna subscripts are a single value out of 1...4, dependent on the RNTI and ECCEs selected. For a distributed EPDCCH transmission, the antenna subscripts alternate between one of two values: {1,3} (p = 107,109) for normal cyclic prefix and {1,2} (p = 107,108) for extended cyclic prefix. See section 6.8A.5 in TS 36.211. These indices are suitable for indexing an N-by-M-by-4 array that represents the subframe resource grid across the 4 possible EPDCCH antenna ports (p = 107...110).

Data Types: `char` | `cell`

Output Arguments

`ind` — Subframe EPDCCH RE indices

numeric column vector | 3-column numeric matrix

EPDCCH subframe resource element indices, returned by default as a numeric column vector of length NRE-by-1. Optionally, for subscript-specific indexing style [`subcarrier`, `symbol`, `antenna`], ind is returned as a numeric matrix of size NRE-by-3. NRE is the number of subframe resource elements. By default, the indices are returned in 1-based linear indexing form, which you can use to directly index the EPDCCH subframe resource elements.

Data Types: `double`

`info` — Information related to EPDCCH indices

scalar structure

Information related to EPDCCH indices, returned as a scalar structure. info is a structure having the following fields.

Parameter Field	Description	Values	Data Type
`EPDCCHG`	EPDCCH data bit capacity	Integer	int32
`EPDCCHGd`	EPDCCH QPSK symbol capacity	Integer	int32
`nEPDCCH`	Number of REs in a PRB pair configured for possible EPDCCH transmission. See section 6.8A.1 in TS 36.211.	Integer	int32
`NECCE`	Number of ECCE available for transmission of EPDCCHs in the PRB pair set	Integer	int32
`NECCEPerPRB`	Number of ECCE per PRB pair	Integer	int32
`NEREGPerECCE`	Number of EREG per ECCE	Integer	int32

See Also

`lteEPDCCH` | `lteEPDCCHDMRSIndices` | `lteEPDCCHIndices`

lteEPDCCHPRBS

EPDCCH pseudorandom scrambling sequence

Syntax

```
seq = lteEPDCCHPRBS(enb,chs,n)
seq = lteEPDCCHPRBS(enb,chs,n,mapping)
```

Description

`seq = lteEPDCCHPRBS(enb,chs,n)` returns the first n outputs of the Enhanced Physical Downlink Control Channel (EPDCCH) scrambling sequence. The function is initialized according to the cell-wide settings structure enb and the channel transmission configuration structure chs.

`seq = lteEPDCCHPRBS(enb,chs,n,mapping)` allows additional control over the format of the returned sequence, seq, through the string mapping. Valid formats are `'binary'` (default) and `'signed'`. The `'binary'` format maps true to 1 and false to 0. The `'signed'` format maps `true` to –1 and `false` to 1.

Examples

Generate the EPDCCH Scrambling Sequence

Specify the cell-wide settings and channel transmission configuration in parameter structures enb and chs.

```
enb.NSubframe = 0;
chs.EPDCCHNID = 0;
```

Create the codeword and generate the EPDCCH scrambling sequence.

```
cw = randi([0 1],100,1);
prbs = lteEPDCCHPRBS(enb,chs,length(cw));
```

Scramble the DCI coded bits.

```
scrambled = xor(prbs,cw);
prbs(1:20)

ans =

     0
     0
     0
     0
     0
     0
     1
     0
     0
     0
     0
     1
     1
     0
     1
     0
     0
     0
     0
     1
```

Generate the EPDCCH scrambling sequence using the `'signed'` sequence format.

```
prbs = lteEPDCCHPRBS(enb,chs,length(cw),'signed');
prbs(1:20)

ans =

     1
     1
     1
     1
     1
     1
    -1
     1
     1
     1
     1
    -1
```

```
    -1
     1
    -1
     1
     1
     1
     1
    -1
```

Input Arguments

enb — eNodeB cell-wide settings
structure

eNodeB cell-wide settings, specified as a structure. This argument contains the following parameter field.

NSubframe — Subframe number
nonnegative scalar integer

Subframe number, specified as a nonnegative scalar integer.

Data Types: `double`

Data Types: `struct`

chs — Channel-specific transmission configuration
structure

Channel-specific transmission configuration, specified as a structure. This argument contains the following parameter field.

EPDCCHNID — EPDCCH scrambling sequence initialization
nonnegative scalar integer

EPDCCH nID parameter for scrambling sequence initialization, specified as a nonnegative scalar integer.

Data Types: `double`

Data Types: `struct`

n — Number of elements in returned sequence
numeric scalar

Number of elements in returned sequence seq, specified as a numeric scalar.

Data Types: `double`

`mapping` — Output sequence format

`'binary'` (default) | `'signed'`

Format of the returned sequence seq, specified as the string `'binary'` or `'signed'`.

- `'binary'` maps `true` to 1 and `false` to 0.
- `'signed'` maps `true` to –1 and `false` to 1.

Data Types: `char`

Output Arguments

`seq` — EPDCCH pseudorandom scrambling sequence

logical column vector | numeric column vector

EPDCCH pseudorandom scrambling sequence, returned as a logical column vector or a numeric column vector. This argument contains the first n outputs of the EPDCCH scrambling sequence, when initialized according to the cell-wide settings structure enb and the channel transmission configuration chs. If you set mapping to `'signed'`, the output data type is `double`. Otherwise, the output data type is `logical`.

Data Types: `logical` | `double`

See Also

`lteEPDCCH` | `lteEPDCCHIndices`

lteEVM

Error vector magnitude calculation

Syntax

```
evm = lteEVM(x,r)
evm = lteEVM(ev)
```

Description

`evm = lteEVM(x,r)` returns a structure, evm, containing error vector magnitude (EVM) information for the input vector, x, given the reference signal vector, r. The EVM is defined using the error, or difference, between the input values, x, and the reference signal, r.

The EVM values in the `RMS` and `Peak` structure fields are linear EVM, not EVM as a percentage. To obtain EVM as a percentage, multiply the value of the `RMS` and `Peak` structure fields by 100.

`evm = lteEVM(ev)` returns a structure, evm, for the input vector, ev, which is taken to be the normalized error vector given by the expression `ev=(x-r)/sqrt(mean(abs(r.^2)))`. This syntax allows for peak and RMS EVM calculation for preexisting normalized error vectors. For example, it can be used to calculate the EVM across an array of previous EVM results, by extracting and concatenating the EV fields from the array to form the ev input vector.

Examples

Measure EVM of Random QPSK Constellation

Create a random QPSK constellation.

```
txSym = lteSymbolModulate(randi([0,1],10000,1),'QPSK');
```

Specify an error vector magnitude (EVM) percentage, `evmPercent`.

```
evmPc = 14.0;
```

Create noisy received symbols, using the EVM percentage. Then, measure the EVM.

```
noise = complex(randn(size(txSym)),randn(size(txSym)))*(evmPc/100)/sqrt(2);
rxSym = txSym + noise;
```

Finally, measure the EVM. Display the information structure and the root mean square (RMS) EVM as a percentage.

```
evm = lteEVM(rxSym,txSym)
evm.RMS*100
```

```
    Peak: 0.4249
     RMS: 0.1378
      EV: [5000x1 double]

   13.7831
```

Input Arguments

x — Input vector

numeric column vector

Input vector, specified as a numeric column vector.

Data Types: `double | single`
Complex Number Support: Yes

r — Reference signal vector

numeric column vector

Reference signal vector, specified as a numeric column vector.

Data Types: `double | single`
Complex Number Support: Yes

ev — Normalized error vector

numeric column vector

Normalized error vector, specified as a numeric column vector.

Data Types: `double | single`

Complex Number Support: Yes

Output Arguments

`evm` — EVM information

structure

EVM information, returned as structure. evm contains the following fields.

`RMS` — Root mean square (RMS) EVM

positive numeric scalar

Root mean square (RMS) EVM, specified as a positive numeric scalar. It is the square root of the mean of the squares of all the values of the EVM.

Data Types: `double | single`

`Peak` — Peak EVM

positive numeric scalar

Peak EVM, returned as a positive numeric scalar. It is the largest single EVM value calculated across all input values.

Data Types: `double | single`

`EV` — Normalized error vector

numeric column vector

Normalized error vector, returned as a numeric column vector.

Data Types: `double | single`
Complex Number Support: Yes

Data Types: `struct`

See Also

`lteSymbolDemodulate`

lteEqualizeMIMO

MMSE-based joint downlink equalization and combining

Syntax

```
[out,csi] = lteEqualizeMIMO(enb,chs,in,hest,noiseest)
```

Description

`[out,csi] = lteEqualizeMIMO(enb,chs,in,hest,noiseest)` performs joint equalization and combining of the received PDSCH symbols in in, given cell-wide settings structure, enb, PDSCH configuration structure, chs, channel estimate, hest, and noise power estimate, noiseest. MMSE equalization is performed on the product of the channel matrix and precoding matrices. Thus, it performs MMSE equalization between transmit and receive layers and returns the result, out.

Examples

Equalize and Deprecode PDSCH Symbols

Generate a resource grid for RMC R.11 in a MIMO configuration.

```
enb = lteRMCDL(struct('RC','R.11'),2);
enb.TotSubframes = 1;
[~,txGrid] = lteRMCDLTool(enb,{[1;0],[0;1]});
```

Extract the PDSCH symbols from this transmit grid.

```
[ind,indInfo] = ltePDSCHIndices(enb,enb.PDSCH,enb.PDSCH.PRBSet);
pdschSym = txGrid(ind);
```

Create an ideal, or identity, channel estimate and an ideal, or zero, noise estimate.

```
hest = permute(repmat(eye(enb.CellRefP),[1,1,indInfo.Gd]),[3,1,2]);
nest = 0.0;
```

Equalize and deprecode the PDSCH symbols, using the channel and noise estimates.

```
[out,csi] = lteEqualizeMIMO(enb,enb.PDSCH,pdschSym,hest,nest);
```

Input Arguments

`enb` — Cell-wide settings
structure

Cell-wide settings, specified as a structure with the following fields. The parameter fields TDDConfig and SSC are only required if DuplexMode is set to `'TDD'`. The CFI parameter field is only required if chs.TxScheme is set to `'SpatialMux'` or `'MultiUser'`.

`NDLRB` — Number of downlink resource blocks
positive scalar integer

Number of downlink resource blocks, specified as a positive scalar integer.

Data Types: `double`

`NCellID` — Physical layer cell identity
nonnegative scalar integer

Physical layer cell identity, specified as a nonnegative scalar integer.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`CellRefP` — Number of cell-specific reference signal antenna ports
1 | 2 | 4

Number of cell-specific reference signal antenna ports, specified as 1, 2, or 4.

Data Types: `double`

`NSubframe` — Subframe number
nonnegative scalar integer

Subframe number, specified as a nonnegative scalar integer.

Data Types: `double`

`DuplexMode` — Duplexing mode

`'FDD'` (default) | Optional | `'TDD'`

Duplexing mode, specified as a string. Optional.

Data Types: `char`

`TDDConfig` — TDD duplex mode subframe configuration

0 (default) | Optional | nonnegative scalar integer (0...6)

TDD duplex mode subframe configuration, specified as a nonnegative scalar integer between 0 and 6. Optional. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

`SSC` — Uplink and downlink special subframe configuration

0 (default) | Optional | nonnegative scalar integer (0...9)

Uplink and downlink special subframe configuration, specified as a nonnegative scalar integer between 0 and 9. Optional. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

`CFI` — Control format indicator value

1 | 2 | 3

Control format indicator value, returned as a positive scalar integer. Valid values are 1, 2, and 3. Only required if chs.TxScheme is set to `'SpatialMux'` or `'MultiUser'`.

Data Types: `double`

Data Types: `struct`

`chs` — PDSCH configuration

structure

PDSCH configuration, specified as a structure that can contain the following fields. The PMISet and PRBSet parameter fields are only required if chs.TxScheme is set to `'SpatialMux'` or `'MultiUser'`.

`NLayers` — Number of transmission layers

1 | 2 | 3 | 4

Number of transmission layers, specified as a positive scalar integer. Possible values are 1, 2, 3, or 4.

Data Types: `double`

`Modulation` — Modulation scheme

`'QPSK'` | `'16QAM'` | `'64QAM'` | cell array of one or two strings

Modulation scheme, specified as a string or cell array of one or two strings, with the modulation formats for one or two codewords. Two codewords is valid only for transmission schemes `'CDD'` and `'SpatialMux'`.

Data Types: `cell` | `char`

`RNTI` — Radio network temporary identifier

numeric scalar

Radio network temporary identifier, 16-bit, specified as a numeric scalar.

Data Types: `double`

`TxScheme` — Transmission scheme

`'CDD'` | `'SpatialMux'` | `'MultiUser'`

Transmission scheme, specified as a string. Its values and their descriptions are shown in the following table.

String	Description
`'CDD'`	Large delay CDD scheme
`'SpatialMux'`	Closed-loop spatial multiplexing scheme
`'MultiUser'`	Multi-user MIMO scheme

Data Types: `char`

`PMISet` — Precoder matrix indication set

numeric scalar (0...15) | numeric vector (0...15)

Precoder matrix indication set, specified as a numeric scalar or a numeric vector with values ranging between 0 and 15. Only required if TxScheme is set to `'SpatialMux'` or `'MultiUser'`. The field may contain either a single value, corresponding to single PMI mode, or multiple values, corresponding to multiple or subband PMI mode.

Data Types: `double`

`PRBSet` — Set of 0-based PRB indices

1- or 2-column numeric matrix

Set of 0-based PRB indices, specified as a 1- or 2-column numeric matrix. Only required if TxScheme is set to `'SpatialMux'` or `'MultiUser'`. The physical resource block indices (PRBs) correspond to the resource allocations for this PDSCH. If a column vector is provided, the resource allocation is the same in both slots of the subframe. Use the 2-column matrix to specify differing PRBs for each slot in a subframe. The PRB indices are 0-based.

Data Types: `double`

Data Types: `struct`

`in` — Received PDSCH input symbols

numeric matrix

Received PDSCH input symbols, specified as a numeric matrix of size *M*-by-`NRxAnts`, where *M* is the number of received symbols for each of `NRxAnts` receive antennas.

Data Types: `double`
Complex Number Support: Yes

`hest` — Channel estimate

3-D numeric array

Channel estimate, specified as a 3-D numeric array of size *M*-by-`NRxAnts`-by-CellRefP, where *M* is the number of received symbols in in, `NRxAnts` is the number of receive antennas, and CellRefP is the number of cell-specific reference signal antenna ports, given in the enb structure.

Data Types: `double`

`noiseest` — Noise power estimate

numeric scalar

Noise power estimate, specified as a numeric scalar. This argument is an estimate of the noise power spectral density per RE on `rxgrid`. Such an estimate is provided by the `lteDLChannelEstimate` function.

Data Types: `double`

Output Arguments

`out` — Equalized output symbols

numeric matrix

Equalized output symbols, returned as a numeric matrix of size *M*-by-*NU*, where *M* is the number of received symbols for each receive antenna and *NU* is the number of transmit layers.

Data Types: `double`
Complex Number Support: Yes

`csi` — Soft channel state information

numeric matrix

Soft channel state information, returned as a numeric matrix of size *M*-by-`NU`, the same size as out. This argument contains soft channel state information and provides an estimate, via MMSE, of the received gain for each received layer.

Data Types: `double`

See Also

`lteDLChannelEstimate` | `lteDLPrecode` | `lteEqualizeMMSE` | `lteEqualizeULMIMO` | `lteEqualizeZF` | `ltePDSCHDecode`

lteEqualizeMMSE

MMSE equalization

Syntax

```
[out,csi] = lteEqualizeMMSE(rxgrid,channelest,noiseest)
```

Description

`[out,csi] = lteEqualizeMMSE(rxgrid,channelest,noiseest)` returns equalized data in multidimensional array, out. MMSE equalization is applied to the received data resource grid in the matrix, rxgrid, using the channel information in the channelest matrix. noiseest is an estimate of the received noise power spectral density.

Alternatively, the input channelest can be provided as a 3-D array of size *NRE*-by-`NRxAnts`-by-*P*, and the input rxgrid can be provided as a matrix of size *NRE*-by-`NRxAnts`. In this case, the first two dimensions have been reduced to one dimension by appropriate indexing through the frequency and time locations of the resource elements of interest, typically for a single physical channel. The outputs, out and csi, are of size (*N*×*M*)-by-*P*.

Examples

Equalize MMSE for RMC R.5

This example applies MMSE equalization on the received signal for reference measurement channel (RMC) R.5, after channel estimation.

Set the DL reference measurement channel to R.5

```
enb = lteRMCDL('R.5');
```

Set channel estimator configuration `PilotAverage` field to `UserDefined`. as follows: averaging window of 9 resource elements in both frequency and time domain, cubic interpolation with a casual window.

```
cec = struct('FreqWindow',9,'TimeWindow',9,'InterpType','cubic');
cec.PilotAverage = 'UserDefined';
cec.InterpWinSize = 1;
cec.InterpWindow = 'Causal';
```

Generate the `txWaveform`.

```
txWaveform = lteRMCDLTool(enb,[1;0;0;1]);
n = length(txWaveform);
```

Apply some random noise to the transmitted signal and save as the `rxWaveform`.

```
rxWaveform = repmat(txWaveform,1,2)+complex(randn(n,2),randn(n,2))*1e-3;
```

Next, demodulate the received data.

```
rxGrid = lteOFDMDemodulate(enb,rxWaveform);
```

Then, perform channel estimation.

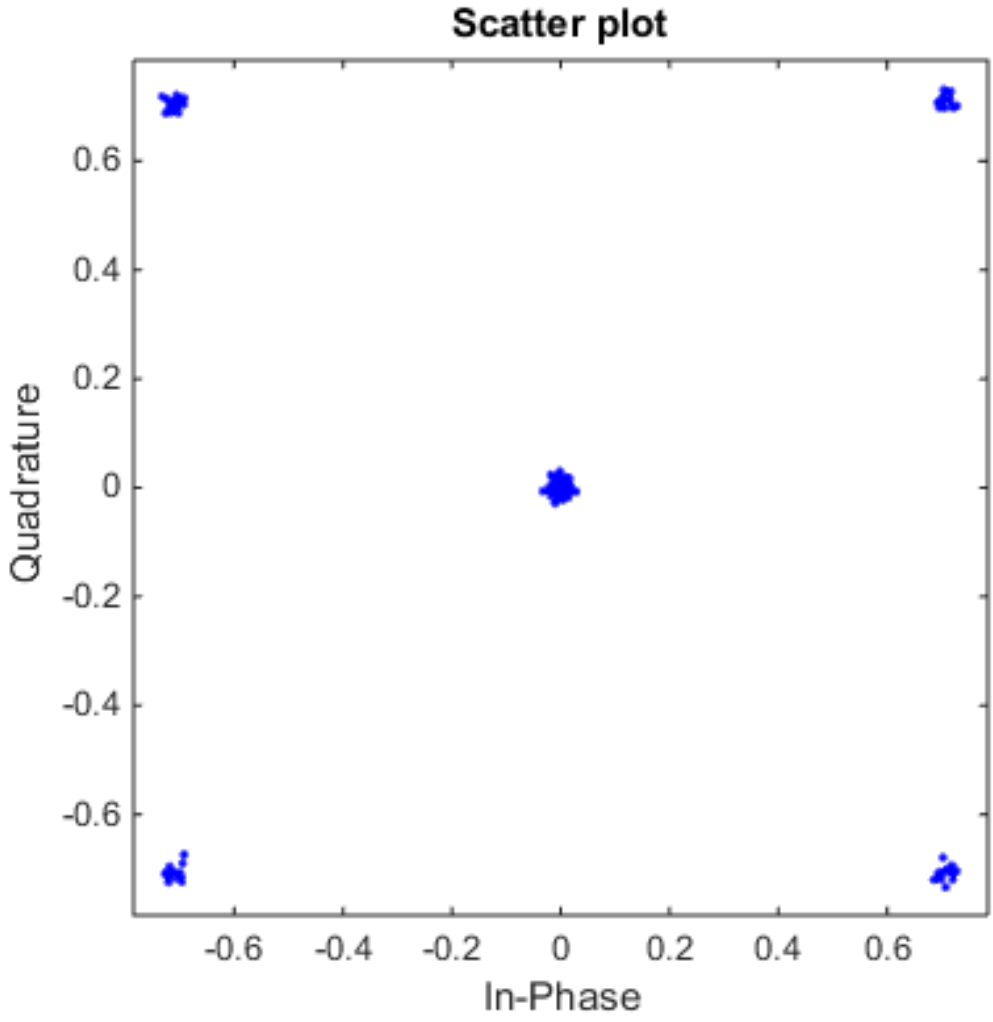
```
[hest,nO] = lteDLChannelEstimate(enb,cec,rxGrid);
```

Finally, apply the MMSE equalization.

```
out = lteEqualizeMMSE(rxGrid,hest,nO);
```

Show scatter plot of one component carrier.

```
scatterplot(out(:,1))
```



Input Arguments

`rxgrid` — Received data resource grid
3-D numeric array | 2-D numeric matrix

Received data resource grid, specified as a 3-D numeric array or a 2-D numeric matrix. As a 3-D numeric array, it has size *N*-by-*M*-by-`NRxAnts`, where *N* is the number of subcarriers, *M* is the number of OFDM symbols, and `NRxAnts` is the number of receive antennas.

Alternatively, as a 2-D numeric matrix, it has size *NRE*-by-`NRxAnts`. In this case, the first two dimensions have been reduced to one dimension by appropriate indexing through the frequency and time locations of the resource elements of interest, typically for a single physical channel.

Data Types: `double`
Complex Number Support: Yes

`channelest` — Channel information

4-D numeric array | 3-D numeric array

Channel information, specified as a 4-D numeric array or a 3-D numeric array. As a 4-D numeric array, it has size *N*-by-*M*-by-`NRxAnts`-by-*P*. *N* is the number of subcarriers, *M* is the number of OFDM symbols, `NRxAnts` is the number of receive antennas, and *P* is the number of transmit antennas. Each element is a complex number representing the narrowband channel for each resource element and for each link between transmit and receive antennas. This matrix can be obtained using the channel estimation command `lteDLChannelEstimate`.

Alternatively, as a 3-D numeric array, it has size *NRE*-by-`NRxAnts`-by-*P*. In this case, the first two dimensions have been reduced to one dimension by appropriate indexing through the frequency and time locations of the resource elements of interest, typically for a single physical channel.

Data Types: `double`
Complex Number Support: Yes

`noiseest` — Noise power estimate

numeric scalar

Noise power estimate, specified as a numeric scalar. It is an estimate of the received noise power spectral density per RE on `rxgrid`.

Data Types: `double`

Output Arguments

`out` — Equalized output data

3-D numeric array | 2-D numeric matrix

Equalized output data, returned as a 3-D numeric array or a 2-D numeric matrix. As a 3-D numeric array, it has size *N*-by-*M*-by-*P*, where *N* is the number of subcarriers, *M* is the number of OFDM symbols, and *P* is the number of transmit antennas.

Alternatively, if channelest is provided as a 3-D array, out is a 2-D numeric matrix of size (*N*×*M*)-by-*P*. In this case, the first two dimensions have been reduced to one dimension by appropriate indexing through the frequency and time locations of the resource elements of interest, typically for a single physical channel.

Data Types: `double`
Complex Number Support: Yes

`csi` — Soft channel state information

3-D numeric array | 2-D numeric matrix

Soft channel state information, returned as a 3-D numeric array of the same size as out. As a 3-D numeric array, it has size *N*-by-*M*-by-*P*, where *N* is the number of subcarriers, *M* is the number of OFDM symbols, and *P* is the number of transmit antennas. csi provides an estimate (via MMSE) of the received RE gain for each received RE.

Alternatively, if channelest is provided as a 3-D array, csi is a 2-D numeric matrix of size (*N*×*M*)-by-*P*. In this case, the first two dimensions have been reduced to one dimension by appropriate indexing through the frequency and time locations of the resource elements of interest, typically for a single physical channel.

Data Types: `double`

See Also

`lteDLChannelEstimate` | `lteEqualizeMIMO` | `lteEqualizeULMIMO` | `lteEqualizeZF` | `lteOFDMDemodulate` | `lteSCFDMADemodulate` | `lteULChannelEstimate`

lteEqualizeULMIMO

MMSE-based joint uplink equalization and combining

Syntax

```
[out,csi] = lteEqualizeULMIMO(ue,chs,in,hest,noiseest)
```

Description

`[out,csi] = lteEqualizeULMIMO(ue,chs,in,hest,noiseest)` performs joint equalization and combining of the received PUSCH symbols in in, given UE-specific settings structure, ue, PUSCH configuration structure, chs, channel estimate, hest and noise power estimate, noiseest. MMSE equalization is performed on the product of the channel matrix and precoding matrices, thus performing MMSE equalization between transmit and receive layers and returning the result in out.

Examples

Equalize and Deprecode PUSCH Symbols

Extract, equalize, and deprecode PUSCH symbols from an RMC A3-2 grid.

Generate a resource grid using multiple antennas to transmit a single PUSCH codeword.

```
ue = lteRMCUL('A3-2');
ue.TotSubframes = 1;
ue.NTxAnts = 2;
ue.PUSCH.NLayers = 2;
[~,txGrid] = lteRMCULTool(ue,[1;0;0;1]);
```

Extract the PUSCH symbols from this transmit grid.

```
[ind,indInfo] = ltePUSCHIndices(ue,ue.PUSCH);
puschSym = txGrid(ind);
```

Create an ideal, or identity, channel estimate and an ideal, or zero, noise estimate.

```
hest = permute(repmat(eye(ue.NTxAnts),[1,1,indInfo.Gd]),[3,1,2]);
```

```
nest = 0.0;
```

Equalize and deprecode the PUSCH symbols, using the channel and noise estimates.

```
[out,csi] = lteEqualizeULMIMO(ue,ue.PUSCH,puschSym,hest,nest);
```

Input Arguments

ue — UE-specific settings

structure

UE-specific settings, specified as a structure that can contain the following fields.

NTxAnts — Number of transmission antennas

1 (default) | Optional | 2 | 4

Number of transmission antennas, specified as a positive scalar integer. Optional. Valid values are 1, 2, and 4.

Data Types: `double`

Data Types: `struct`

chs — PUSCH configuration structure

structure

PUSCH configuration structure, specified as a structure that can contain the following fields. The PMI parameter field is only required if ue.NTxAnts is set to 2 or 4.

NLayers — Number of transmission layers

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers, specified as 1, 2, 3, or 4. Optional.

Data Types: `double`

PMI — Precoder matrix indication

nonnegative scalar integer (0...23)

Precoder matrix indication, specified as a nonnegative scalar integer between 0 and 23. Only required if ue.NTxAnts is set to 2 or 4. This PMI is to be used during precoding of the DRS reference symbols. For more information, see `lteULPMIInfo`.

Data Types: `double`

Data Types: `struct`

`in` — Received PUSCH input symbols

numeric matrix

Received PUSCH input symbols, specified as a numeric matrix of size *M*-by-`NRxAnts`, where *M* is the number of received symbols for each of the `NRxAnts` receive antennas.

Data Types: `double`
Complex Number Support: Yes

`hest` — Channel estimate

3-D numeric array

Channel estimate, specified as a 3-D numeric array of size *M*-by-`NRxAnts`-by-`NTxAnts`, where *M* is the number of received symbols in in, `NRxAnts` is the number of receive antennas, and `NTxAnts` is the number of transmit antenna ports, given by `ue.NTxAnts`.

Data Types: `double`

`noiseest` — Noise power estimate

numeric scalar

Noise power estimate as power spectral density per RE on `rxgrid`, specified as a numeric scalar. Such an estimate is provided by the `lteULChannelEstimate` function.

Data Types: `double`

Output Arguments

`out` — Equalized output symbols

complex-valued numeric matrix

Equalized output symbols, returned as a complex-valued numeric matrix of size *M*-by-*NU*, where *M* is the number of received symbols for each receive antenna and *NU* is the number of transmit layers.

Data Types: `double`
Complex Number Support: Yes

`csi` — Soft channel state information

numeric matrix

Soft channel state information, returned as a numeric matrix of the same size as out, *M*-by-*NU*. This output provides an estimate, via MMSE, of the received gain for each received layer.

Data Types: `double`

See Also

`lteEqualizeMIMO` | `lteEqualizeMMSE` | `lteEqualizeZF` | `ltePUSCHDecode` | `ltePUSCHPrecode` | `lteULChannelEstimate`

lteEqualizeZF

Zero-forcing equalization

Syntax

```
[out,csi] = lteEqualizeZF(rxgrid,channelest)
```

Description

`[out,csi] = lteEqualizeZF(rxgrid,channelest)` returns equalized data in multidimensional array, out, by applying MIMO zero-forcing equalization to the received data resource grid in matrix rxgrid, using the channel information in the channelest input matrix.

For each resource element, the function calculates the pseudoinverse of the channel and equalizes the corresponding received signal.

Alternatively, the channelest input can be provided as a 3-D array of size *NRE*-by-*NRxAnts*-by-*P* and the rxgrid input can be provided as a matrix of size *NRE*-by-*NRxAnts*. In this case, the first two dimensions have been reduced to one dimension by appropriate indexing through the frequency and time locations of the resource elements of interest, typically for a single physical channel. The outputs, out and csi, are of size (*N*×*M*)-by-*P*.

Examples

Perform Zero-Forcing Equalization for RMC R.5

Perform zero-forcing equalization for a reference measurement channel (RMC) R.5 after channel estimation.

```
enb = lteRMCDL('R.5');
cec = struct('FreqWindow',9,'TimeWindow',9,'InterpType','cubic');
cec.PilotAverage = 'UserDefined';
cec.InterpWinSize = 1;
```

```
cec.InterpWindow = 'Causal';
txWaveform = lteRMCDLTool(enb,[1;0;0;1]);
n = length(txWaveform);
rxWaveform = repmat(txWaveform,1,2)+complex(randn(n,2),randn(n,2))*1e-4;
rxGrid = lteOFDMDemodulate(enb,rxWaveform);
hest = lteDLChannelEstimate(enb,cec,rxGrid);
out = lteEqualizeZF(rxGrid,hest);
```

Input Arguments

`rxgrid` — Received data resource grid

3-D numeric array | 2-D numeric matrix

Received data resource grid, specified as a 3-D numeric array or a 2-D numeric matrix. As a 3-D numeric array, it has size *N*-by-*M*-by-`NRxAnts`, where *N* is the number of subcarriers, *M* is the number of OFDM symbols, and `NRxAnts` is the number of receive antennas.

Alternatively, as a 2-D numeric matrix, it has size *NRE*-by-`NRxAnts`. In this case, the first two dimensions have been reduced to one dimension by appropriate indexing through the frequency and time locations of the resource elements of interest, typically for a single physical channel.

Data Types: `double`
Complex Number Support: Yes

`channelest` — Channel information

4-D numeric array | 3-D numeric array

Channel information, specified as a 4-D numeric array or a 3-D numeric array. As a 4-D numeric array, it has size *N*-by-*M*-by-`NRxAnts`-by-*P*. *N* is the number of subcarriers, *M* is the number of OFDM symbols, `NRxAnts` is the number of receive antennas, and *P* is the number of transmit antennas. Each element is a complex number representing the narrowband channel for each resource element and for each link between transmit and receive antennas. This matrix can be obtained using a channel estimation function, such as `lteDLChannelEstimate`.

Alternatively, as a 3-D numeric array, it has size *NRE*-by-`NRxAnts`-by-*P*. In this case, the first two dimensions have been reduced to one dimension by appropriate indexing through the frequency and time locations of the resource elements of interest, typically for a single physical channel.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`out` — Equalized output data

3-D numeric array | 2-D numeric matrix

Equalized output data, returned as a 3-D numeric array or a 2-D numeric matrix. As a 3-D numeric array, it has size *N*-by-*M*-by-*P*. *N* is the number of subcarriers, *M* is the number of OFDM symbols, and *P* is the number of transmit antennas.

Alternatively, if channelest is provided as a 3-D array, out is a 2-D numeric matrix of size (*N*×*M*)-by-*P*. In this case, the first two dimensions have been reduced to one dimension by appropriate indexing through the frequency and time locations of the resource elements of interest, typically for a single physical channel.

Data Types: `double`
Complex Number Support: Yes

`csi` — Soft channel state information

3-D numeric array | 2-D numeric matrix

Soft channel state information, returned as a 3-D numeric array or a 2-D numeric matrix of the same size as out. As a 3-D numeric array, it has size *N*-by-*M*-by-*P*. *N* is the number of subcarriers, *M* is the number of OFDM symbols, and *P* is the number of transmit antennas. csi provides an estimate of the received RE gain for each received RE.

Alternatively, if channelest is provided as a 3-D array, csi is a 2-D numeric matrix of size (*N*×*M*)-by-*P*. In this case, the first two dimensions have been reduced to one dimension by appropriate indexing through the frequency and time locations of the resource elements of interest, typically for a single physical channel.

Data Types: `double`

See Also

`lteDLChannelEstimate` | `lteEqualizeMIMO` | `lteEqualizeMMSE` | `lteEqualizeULMIMO` | `lteOFDMDemodulate` | `lteSCFDMADemodulate` | `lteULChannelEstimate`

lteExtractResources

Resource elements extraction

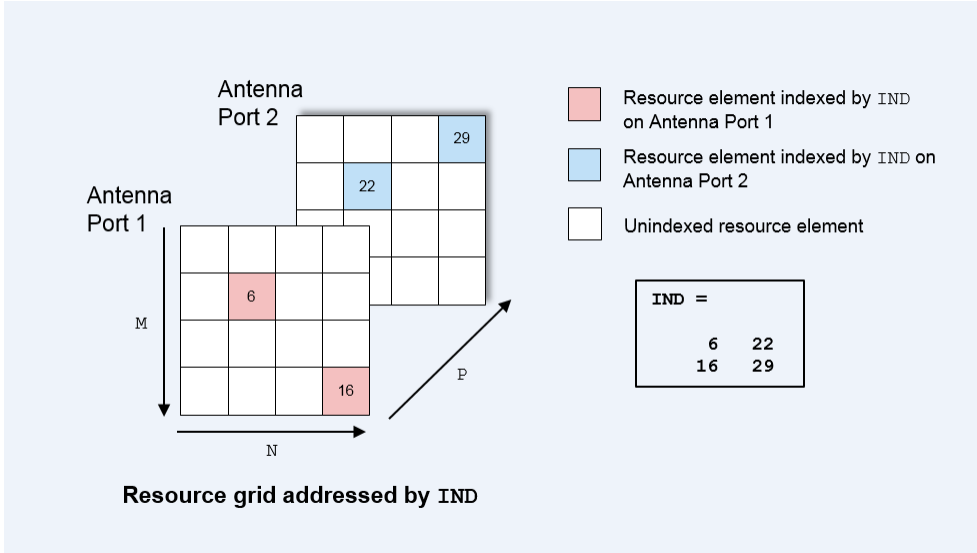
Syntax

```
[re,reind] = lteExtractResources(ind,grid)
[re1, re2,...,reN,reind1, reind2, ...,reindN]= lteExtractResources(
ind,grid1,grid2, ....,gridN)
re = lteExtractResources(....,opts)
```

Description

`[re,reind] = lteExtractResources(ind,grid)` returns resource elements `re` and indices of the extracted resource elements `reind` from a resource array, `grid`, using resource elements indices `ind`. You can extract resource elements from a resource grid with different dimensionality than the resource grid addressed by the indices. The indices specified and returned are in 1–based linear indexing form. Other indexing options are available. The resource extraction process is further explained in “Algorithms” on page 1-257.

In LTE System Toolbox, indices are generated for mapping sequences of physical channel and signal symbols to a resource grid. These indices are generated using channel-or signal-specific functions and address resource elements in an array sized, *M*-by-*N*-by-*P*. *M* is the number of subcarriers, *N* is the number of OFDM or SC-FDMA symbols and *P* is the number of planes. The diagram highlights the resource elements of a resource grid addressed by indices, `ind`. The indices are in a 1–based linear indexing form. $P = 2$ is the number of antenna ports.



Typically the resource array extracts resource elements from one of the following:

- A 3–D received grid, sized *M*-by-*N*-by-*NRxAnts*. *NRxAnts* is the number of receive antennas. This grid is created after OFDM or SC-FDMA demodulation.
- A 4–D channel estimation grid, sized *M*-by-*N*-by-*NRxAnts*-by-*P*. This grid is created by channel estimation functions (refer “Channel Estimation”).

You can describe the size of the 3D received grid as a 4D grid that has a trailing singleton dimension.

`[re1, re2,...,reN,reind1, reind2, ...,reindN]= lteExtractResources( ind,grid1,grid2, ....,gridN)` extracts resource elements from multiple resource arrays using the indices `ind`.

`re = lteExtractResources(....,opts)` enables control over the format of the indices and the extraction method used through a cell array of option strings, `opts`.

Examples

Extract PDCCH Symbols and Channel Estimates for Decoding

Create a transmit waveform for one subframe.

```
enb = lteRMCDL('R.12');
enb.TotSubframes = 1;
txWaveform = lteRMCDLTool(enb,[1;0;0;1]);
```

Receive sum of transmit antenna waveforms on three receive antennas.

```
NRxAnts = 3;
rxWaveform = repmat(sum(txWaveform,2),1,NRxAnts);
rxGrid = lteOFDMDemodulate(enb,rxWaveform);
```

Compute channel estimate.

```
cec.FreqWindow = 1;
cec.TimeWindow = 1;
cec.InterpType = 'cubic';
cec.PilotAverage = 'UserDefined';
cec.InterpWinSize = 3;
cec.InterpWindow = 'Causal';
[hEstGrid,nEst] = lteDLChannelEstimate(enb,cec,rxGrid);
```

Generate PDCCH indices. Extract symbols from received and channel estimate grids in preparation for PDCCH decoding.

```
ind = ltePDCCHIndices(enb);
[pdcchRxSym,pdcchHestSym] = lteExtractResources(ind,rxGrid,hEstGrid);
```

`pdcchRxSym` is sized *NRE* -by- *NRxAnts* and `pdcchHestSym` is sized *NRE* -by- *NRxAnts*-by-`CellRefP`.

```
rxSymSize = size(pdcchRxSym)
hestSymSize = size(pdcchHestSym)
```

```
rxSymSize =

   212     3


hestSymSize =

   212     3     4
```

Decode PDCCH with extracted resource elements

```
pdcchBits = ltePDCCHDecode(enb,pdcchRxSym,pdcchHestSym,nEst);
```

The output is in the MATLAB Workspace.

Extract Resources From 3D Receive Grid and 4D Channel Estimate Grid

Extract resources from a 3D receive grid and 4D channel estimate grid. Show the location of the indices within the grid. This replicates the diagrams shown in "Algorithms" on page 1-257.

Setup sizes of the grids.

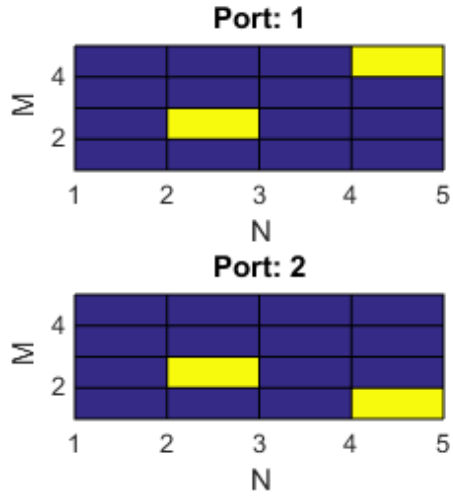
```
M = 4;       % Subcarriers
N = 4;       % OFDM symbols
P = 2;       % Antenna ports
NRxAnts = 3; % Receive antennas
```

Create indices and show the locations within the transmit grid addressed by these indices. As you will notice, different resource elements are addressed on each antenna port.

```
ind = [6 22; 16 29];
txGrid = zeros(M,N,P);
txGrid(ind) = 1; % Addressed resource element locations contain 1
```

Visualize locations of indexed resource elements in the transmit grid.

```
visualizeGrid = zeros(M+1,N+1,P);
visualizeGrid(1:M,1:N,:) = txGrid;
figure;
subplot(321);
pcolor(visualizeGrid(:,:,1));
title('Port: 1'); xlabel('N'); ylabel('M');
subplot(323);
pcolor(visualizeGrid(:,:,2));
title('Port: 2'); xlabel('N'); ylabel('M');
```



Create a 3D received grid to extract resource elements.

```
rxGrid = zeros(M,N,NRxAnts);
```

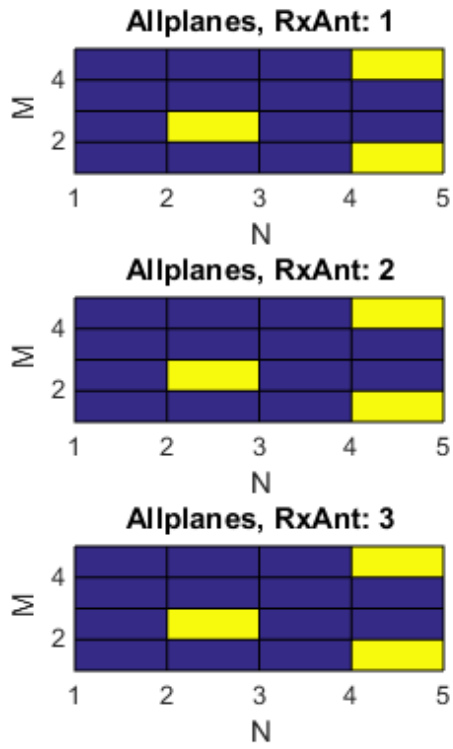
Extract resource elements from the received grid. Show the locations of these extracted resource elements.

```
[re, indOut] = lteExtractResources(ind,rxGrid);
rxGrid(indOut) = 1; % Addressed resource element locations contain 1
```

Visualize locations of indexed resource elements in the receive grid.

```
figure;
visualizeGrid = zeros(M+1,N+1,NRxAnts);
```

```
visualizeGrid(1:M,1:N,:) = rxGrid;
subplot(321); pcolor(visualizeGrid(:,:,1));
title('Allplanes, RxAnt: 1'); xlabel('N'); ylabel('M');
subplot(323); pcolor(visualizeGrid(:,:,2));
title('Allplanes, RxAnt: 2'); xlabel('N'); ylabel('M');
subplot(325); pcolor(visualizeGrid(:,:,3));
title('Allplanes, RxAnt: 3'); xlabel('N'); ylabel('M');
```



Create a 4D channel estimate grid to extract resource elements.

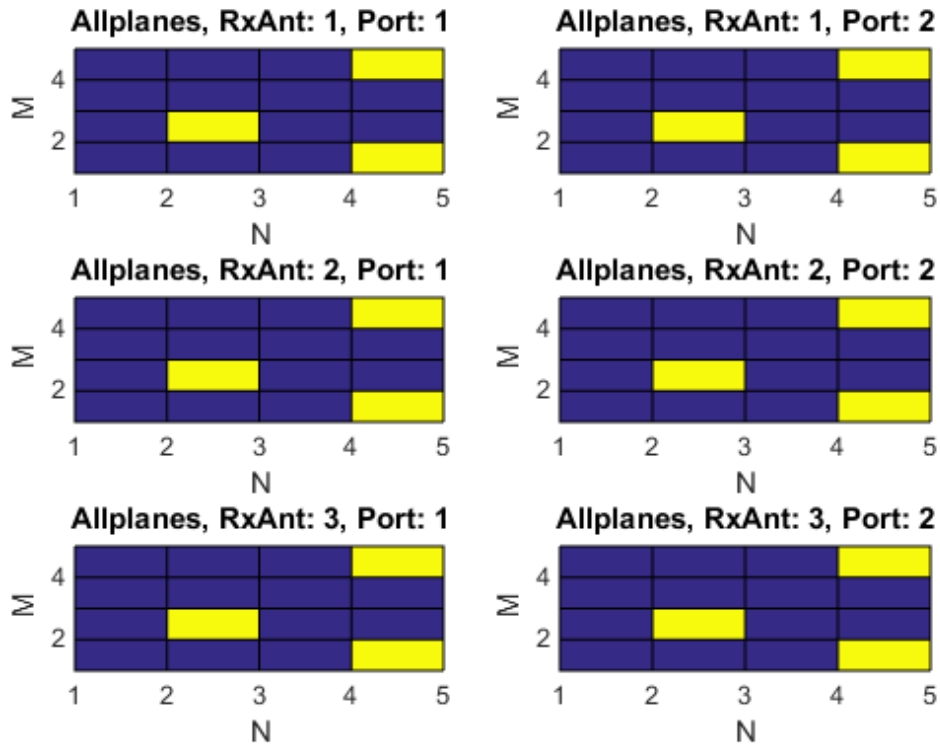
```
hEstGrid = zeros(M,N,NRxAnts,P);
```

Extract resource elements from the channel estimate grid. Show the locations of these extracted resource elements.

```
[re, indOut] = lteExtractResources(ind,hEstGrid);
hEstGrid(indOut) = 1; % Addressed resource element locations contain 1
```

Visualize locations of the resource elements extracted using `'allplanes'` mode from 3D receive grid.

```
figure;
visualizeGrid = zeros(M+1,N+1,NRxAnts,P);
visualizeGrid(1:M,1:N,:,:) = hEstGrid;
subplot(321); pcolor(visualizeGrid(:,:,1,1));
title('Allplanes, RxAnt: 1, Port: 1'); xlabel('N'); ylabel('M');
subplot(323); pcolor(visualizeGrid(:,:,2,1));
title('Allplanes, RxAnt: 2, Port: 1'); xlabel('N'); ylabel('M');
subplot(325); pcolor(visualizeGrid(:,:,3,1));
title('Allplanes, RxAnt: 3, Port: 1'); xlabel('N'); ylabel('M');
subplot(322); pcolor(visualizeGrid(:,:,1,2));
title('Allplanes, RxAnt: 1, Port: 2'); xlabel('N'); ylabel('M');
subplot(324); pcolor(visualizeGrid(:,:,2,2));
title('Allplanes, RxAnt: 2, Port: 2'); xlabel('N'); ylabel('M');
subplot(326); pcolor(visualizeGrid(:,:,3,2));
title('Allplanes, RxAnt: 3, Port: 2'); xlabel('N'); ylabel('M');
```



Create a 4D channel estimate grid to extract resource elements.

```
hEstGridDirect = zeros(M,N,NRxAnts,P);
```

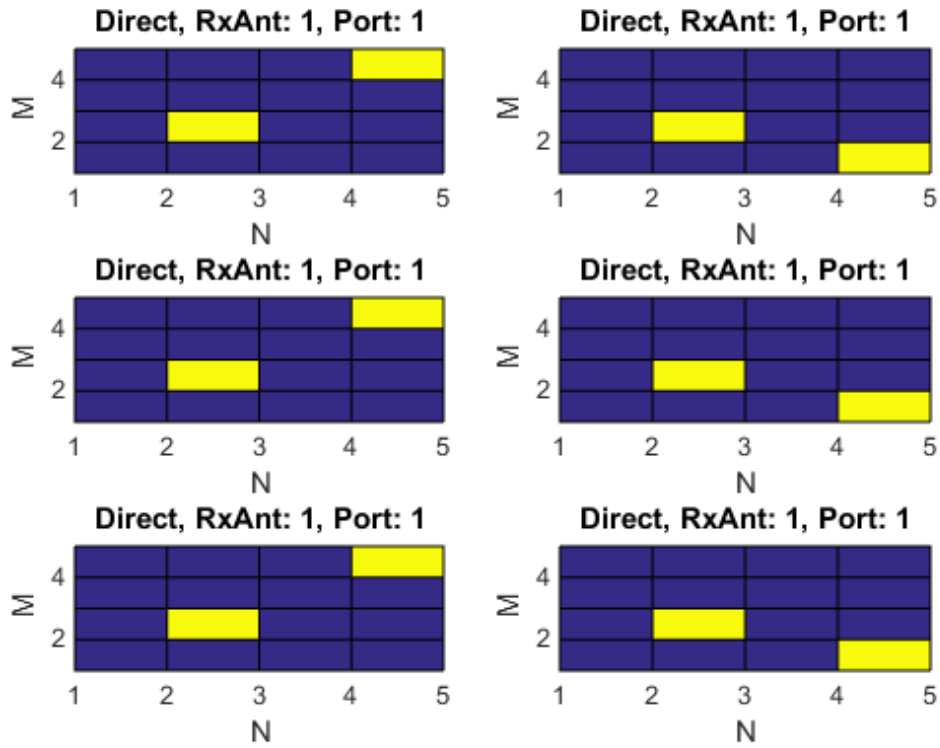
Extract resource elements from the channel estimate grid using `'direct'` extraction mode. Show the locations of these extracted resource elements.

```
[re, indOut] = lteExtractResources(ind,hEstGridDirect,'direct');
hEstGridDirect(indOut) = 1;% Addressed resource element locations contain 1
```

Visualize locations of the resource elements extracted using `'direct'` mode from 4D channel estimate grid.

```
figure;
visualizeGrid = zeros(M+1,N+1,NRxAnts,P);
```

```
visualizeGrid(1:M,1:N,:,:) = hEstGridDirect;
subplot(321); pcolor(visualizeGrid(:,:,1,1));
title('Direct, RxAnt: 1, Port: 1'); xlabel('N'); ylabel('M');
subplot(323); pcolor(visualizeGrid(:,:,2,1));
title('Direct, RxAnt: 1, Port: 1'); xlabel('N'); ylabel('M');
subplot(325); pcolor(visualizeGrid(:,:,3,1));
title('Direct, RxAnt: 1, Port: 1'); xlabel('N'); ylabel('M');
subplot(322); pcolor(visualizeGrid(:,:,1,2));
title('Direct, RxAnt: 1, Port: 1'); xlabel('N'); ylabel('M');
subplot(324); pcolor(visualizeGrid(:,:,2,2));
title('Direct, RxAnt: 1, Port: 1'); xlabel('N'); ylabel('M');
subplot(326); pcolor(visualizeGrid(:,:,3,2));
title('Direct, RxAnt: 1, Port: 1'); xlabel('N'); ylabel('M');
```



Extract Cell-Specific Reference Signal (CRS) Symbols

Use `'direct'` and `'allplanes'` extraction methods and subscript indices to extract cell-specific reference signal (CRS) symbols in subcarrier 7 from `grid`.

Generate a resource grid and CRS indices in the subscript form: [subcarrier, OFDM symbol, CRS port].

```
enb = lteRMCDL('R.12');
enb.TotSubframes = 1;
enb.CellRefP = 2;
enb.PDSCH.NLayers = 2;
[waveform,grid] = lteRMCDLTool(enb,[1;0;0;1]);
crsInd = lteCellRSIndices(enb,'sub');
```

There are 2 resource elements used on CRS ports 1 & 2; all are on different OFDM symbols (1, 5, 8, 12).

```
crsIndSC7 = crsInd(crsInd(:,1)==7,:)
```

```
crsIndSC7 =

           7           1           1
           7           8           1
           7           5           2
           7          12           2
```

Use `'direct'` method to extract resource elements. The extracted resource element indices are same as the generated CRS indices as the resource array indexed by `crsInd` in `grid`.

```
[dirREs,dirInd] = lteExtractResources(crsInd,grid,{'direct','sub'});
directIndSC7 = dirInd(dirInd(:,1)==7,:)
```

```
directIndSC7 =

           7           1           1
           7           8           1
           7           5           2
           7          12           2
```

Use `'allplanes'` method to extract resource elements. There are 4 extracted CRS indices as per the CRS port on subcarrier 7. Indices addressing unique OFDM symbols in the indexed resource grid are used to extract resource elements from all the CRS ports in `'grid`. Therefore indices are extracted at OFDM symbols (1, 5, 8,12) on both CRS ports.

```
[apREs,apInd] = lteExtractResources(crsInd,grid,{'allplanes','sub'});
allPlanesIndSC7 = apInd(apInd(:,1)==7,:)
```

```
allPlanesIndSC7 =

           7           1           1
           7           8           1
           7           5           1
           7          12           1
           7           1           2
```

```
     7           8           2
     7           5           2
     7          12           2
```

Input Arguments

`ind` — Resource elements indices

numeric array

Resource elements indices, specified as a numeric array. The indices address elements of a *N*-by-*M*-by-*P* resource array. *M* is the number of subcarriers, *N* is the number of OFDM or SC-FDMA symbols, and *P* is the number of planes.

`grid` — Resource array

3–D numeric array (default) | 4–D numeric array

Resource array, specified as a 3-D or 4-D numeric array. Typically the resource array to extract resource elements from in one of the following:

- A 3–D received grid, sized *M*-by-*N*-by-*NRxAnts*. *NRxAnts* is the number of receive antennas. This grid is created after OFDM or SC-FDMA demodulation.
- A 4–D channel estimation grid, sized *M*-by-*N*-by-*NRxAnts*-by-*P*. This grid is created by channel estimation functions (refer “Channel Estimation”).

You can describe the size of the 3D received grid as a 4D grid that has a trailing singleton dimension.

Data Types: `double`

`opts` — Resource elements extraction options

string | cell array of strings

Resource elements extraction options, specified as a string or cell array of strings. `opts` can contain the following values:

Parameter Field	Required or Optional	Values	Description
`IndexingStyle`	Required	`'ind'` (default) or `'sub'`	Indexing style of the specified or returned indices, `ind` and `reind`, specified as one of the following options:

Parameter Field	Required or Optional	Values	Description
			• `'ind'` — linear index form • `'sub'` — subscript form
`Index Base`	Required	`'1based'` (default) or `'0based'`	Base value of the specified or returned indices, `ind` and `reind`, specified as one of the following options: • `'1based'` — the first value of index sequence is one • `'0based'` — the first value of the index sequence is zero
`Extraction Method`	Required	`'allplanes'` (default) or `'direct'`	Resource element extraction methods. The methods are described in "Algorithms" on page 1-257. • `'allplanes'` — uses indices addressing unique subcarrier and symbol location over all planes of the indexed resource array for extraction. • `'direct'` — only resource elements relevant to each plane of the indexed resource grid are extracted.

Output Arguments

`re` — Extracted resource elements

column vector | numeric array

Extracted resource elements, returned as a column vector or numeric array.

When `'allplanes'` extraction method is used, the extracted resource elements array is of size *NRE*-by-*NRxAnts*-by-*P* where:

- *NRE* is the number of resource elements per *M*-by-*N* plane of `grid`.
- *M* is the number of subcarriers.

- *N* is the number of OFDM or SC-FDMA symbols.
- *P* is the number of planes.

When using `'direct'` extraction method, the size of the extracted resource elements array, `re`, depends on the number of indices addressing each plane of the indexed source grid:

- If the same number of indices address each plane then `re` is of size *NRE*-by-*NRxAnts*-by-*P*.
- If a different number of indices address each plane then `re` is a column vector containing all extracted resource elements.

`reind` — Indices of extracted resource elements

numeric array

Indices of extracted resource elements within `grid`, returned as numeric array. `reind` is the same size as extracted resource elements array `re`.

More About

Algorithms

`lteExtractResources` can extract resource elements using one of two methods. The `'allplanes'` method is used by default. You can optionally specify `'direct'` extraction method.

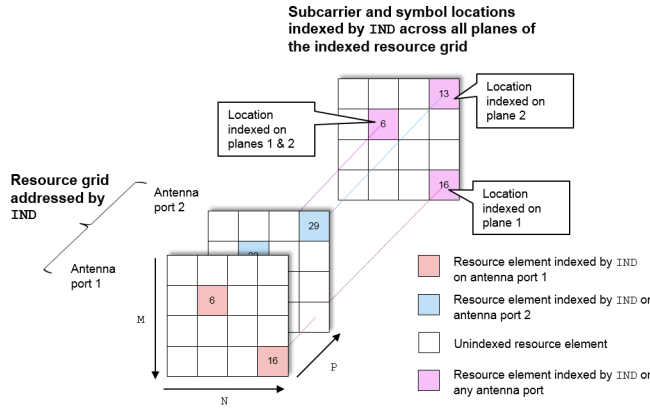
All Planes Extraction Method

The `'allplanes'` method extracts resource elements from each *M*-by-*N* plane within `grid` using indices that address unique subcarrier and symbol locations over all the planes of the indexed resource array.

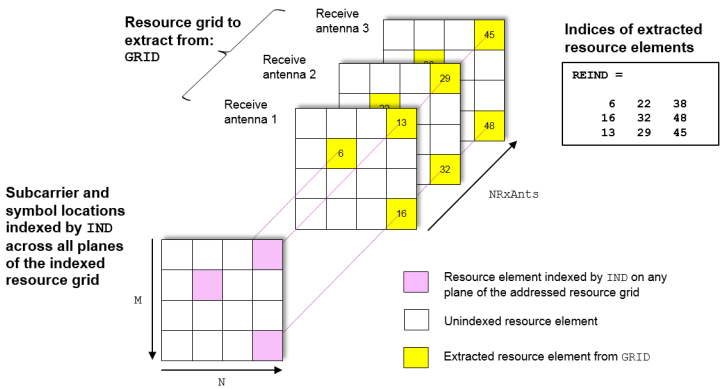
The following diagrams illustrate the resource extraction process for a 3D received grid and a 4D channel estimation grid. The example, “Extract Resources From 3D Receive Grid and 4D Channel Estimate Grid” on page 1-247 recreates these diagrams.

Indices addressed by unique subcarrier and symbol locations across all planes of the indexed resource gird are used for the extraction. The diagram highlights the indices

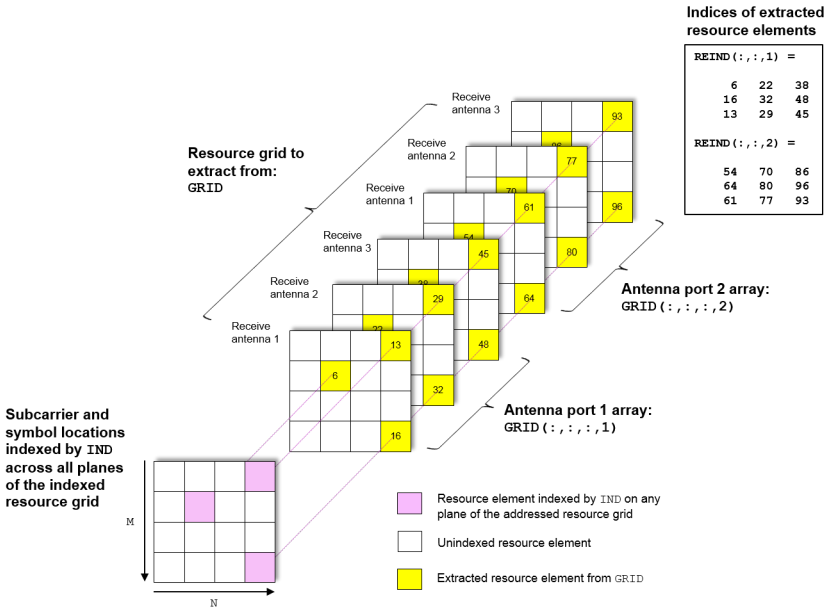
used to extract resource elements address the resource grid with *P* = *2*. In this case, *P* is the number of antenna ports.



Resource elements are extracted from `grid` at the symbol and subcarrier locations. The following diagrams illustrate the resource element extraction from a 3D received grid, `grid`, with *NRxAnts* = *3*.



The following diagram shows the extraction process for a 4D channel estimate grid, `grid`, with *NRxAnts* = *3* and *P*= *2*. In this case, *P* is the number for antenna ports. The 4D resource grid consists of *P* *M*-by-*N*-by-*NRxAnts* arrays, each associated with an antenna port. Resource elements are extracted from all planes within these arrays.



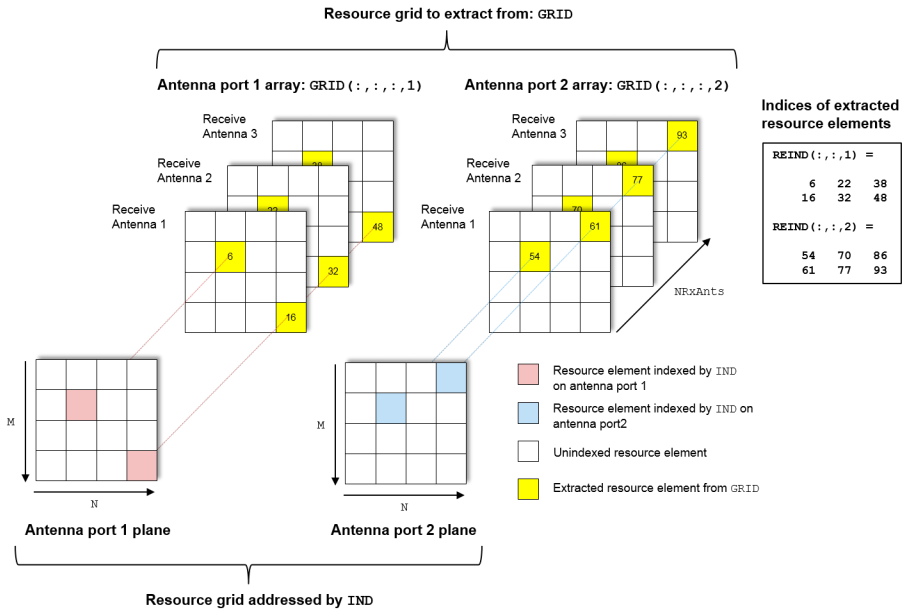
Direct Extraction Method

The `'direct'` method extracts resource elements from `grid` with the assumption that third and fourth dimension of the `grid` represents the same property as the planes of the indexed resource array such as antenna ports, layers, transmit antennas. Therefore the only resource elements relevant to each plane of the indexed resource grid are extracted:

- For a 3D `grid`, the `'direct'` method extracts elements from each *M*-by-*N* plane of `grid` using indices addressing the same plane of the indexed resource array. This is the same as the standard MATLAB operation `re = grid(ind)`. Therefore `reind = ind`.
- For a 4D `grid`, the `'direct'` method extracts elements from each *M*-by-*N*-by-*NRxAnts* array of `grid` using indices addressing the same plane of the indexed resource array. Therefore it is assumed the property represented by the planes of the indexed resource array is the same as the fourth dimension of `grid`.

The extraction of a 4D estimation grid, `grid`, using the `'direct'` method is illustrated in the following diagram with *NRxAnts* = *3* and *P*= *2*, which is the number of antenna ports. The 4D resource grid consists of *P* *M*-by-*N*-by-*NRxAnts* arrays, each associated

with an antenna port. Therefore the indices corresponding to each individual antenna port in the indexed resource array are used to extract resource elements from each of these arrays. The example, “Extract Resources From 3D Receive Grid and 4D Channel Estimate Grid” on page 1-247 creates a version of this diagram.



See Also

`lteCellRSIndices` | `lteDLChannelEstimate` | `lteDLResourceGrid` | `lteOFDMDemodulate` | `ltePBCHDecode` | `ltePBCHIndices` | `ltePCFICHDecode` | `ltePCFICHIndices` | `ltePDCCHDecode` | `ltePDCCHDecode` | `ltePDCCHIndices` | `ltePDCCHIndices` | `ltePDSCHDecode` | `ltePDSCHIndices` | `ltePHICHDecode` | `ltePHICHIndices` | `ltePUCCH1Decode` | `ltePUCCH1Indices` | `ltePUCCH2Decode` | `ltePUCCH2Indices` | `ltePUCCH3Decode` | `ltePUCCH3Indices` | `ltePUSCHDecode` | `ltePUSCHIndices` | `lteSCFDMADemodulate` | `lteULChannelEstimate` | `lteULChannelEstimatePUCCH1` | `lteULChannelEstimatePUCCH2` | `lteULChannelEstimatePUCCH3` | `lteULResourceGrid`

lteFadingChannel

Multipath fading MIMO channel propagation conditions

Syntax

```
[out,info] = lteFadingChannel(model,in)
```

Description

`[out,info] = lteFadingChannel(model,in)` implements the MIMO multipath fading channel model, as specified in [1] and [2]. Each column of the input matrix, in, corresponds to the waveform at each of the transmit antennas. These are filtered by the multipath Rayleigh fading channel model specified in the structure model. The delay profile of model is resampled to match the input signal sampling rate. out is the channel output signal matrix. Each column of out corresponds to the waveform at each of the receive antennas. out has the same number of rows as in. The structure, info, contains information about the channel modeling.

Examples

Transmit Two Consecutive Frames over Fading Channel

Transmit two consecutive frames over a fading propagation channel.

Generate the first frame of a reference measurement channel (RMC) R.10 transmit waveform. Set up the fading propagation channel configuration settings.

```
rmc = lteRMCDL('R.10');
[txWaveform,txGrid,info] = lteRMCDLTool(rmc,[1;0;1]);
chcfg = struct('Seed',1,'DelayProfile','EPA','NRxAnts',1);
chcfg.DopplerFreq = 5.0;
chcfg.MIMOCorrelation = 'Low';
chcfg.SamplingRate = info.SamplingRate;
```

Initialize the channel `InitTime` parameter to 0 seconds, $t = 0$, assuming that the first subframe is transmitted at time 0 seconds.

```
chcfg.InitTime = 0;
```

Transmit the first frame of the waveform over a fading propagation channel, using the channel configuration structure, `chcfg`.

```
rxWaveform = lteFadingChannel(chcfg,txWaveform);
```

The returned output, `rxWaveform`, is the first frame of the received waveform.

Increment the frame number and generate the second frame of an RMC R.10 transmit waveform.

```
rmc.NFrame = 1;
[txWaveform,txGrid] = lteRMCDLTool(rmc,[1;0;1]);
```

Initialize the channel fading process to a time of 10 milliseconds, $t = 10\times10^{-3}$, since the second frame is transmitted at this time. This setting guarantees continuity of the fading process between the end of the first frame and the beginning of the second frame.

```
chcfg.InitTime = 10e-3;
```

Transmit the second frame of the waveform over the fading propagation channel, using the updated channel configuration structure, `chcfg`.

```
rxWaveform = lteFadingChannel(chcfg,txWaveform);
```

The returned output, `rxWaveform`, is the second frame of the received waveform.

Input Arguments

`model` — Multipath fading channel model

structure

Multipath fading channel model, specified as a structure. model must contain the following fields.

`NRxAnts` — Number of receive antennas

positive scalar integer

Number of receive antennas, specified as a positive scalar integer. `NRxAnts` must be equal to or greater than 1.

Data Types: `double`

`MIMOCorrelation` — Correlation between UE and eNodeB antennas

`'Low' | 'Medium' | 'UplinkMedium' | 'High' | 'Custom'`

Correlation between UE and eNodeB antennas, specified as a string. A `'Low'` correlation is equivalent to no correlation between antennas. The `'Medium'` correlation level is applicable to tests defined in [1]. The `'UplinkMedium'` correlation level is applicable to tests defined in [2].

Data Types: `char`

`NormalizeTxAnts` — Transmit antenna number normalization

`'On'` (default) | Optional | `'Off'`

Transmit antenna number normalization, specified as a string. Optional. If the value is `'On'`, the model output is normalized by `1/sqrt(P)`, where P is the number of transmit antennas. This ensures that the output power per receive antenna is unaffected by the number of transmit antennas. If the value is `'Off'`, the output is not normalized by this factor.

Data Types: `char`

`DelayProfile` — Delay profile model

`'EPA' | 'EVA' | 'ETU' | 'Custom' | 'Off'`

Delay profile model, specified as a string. `DelayProfile` set to `'Off'` switches off fading completely and implements a static MIMO channel model. In that case, the antenna geometry is given by the number of transmit antennas (i.e., the number of columns in the input in), the number of receive antennas, `model.NRxAnts`, and the MIMO correlation, `model.MIMOCorrelation`. The temporal part of the model for each link between transmit and receive antennas consists of a single path with zero delay and constant unit gain.

Data Types: `char`

`DopplerFreq` — Doppler frequency

scalar value

Doppler frequency, specified as a scalar value expressed in hertz. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`.

Data Types: `double`

`SamplingRate` — Input signal sampling rate

numeric scalar

Input signal sampling rate, specified as a numeric scalar. It is the rate of each sample in the rows of the input matrix, in. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`.

Data Types: `double`

`InitTime` — Fading process time offset

numeric scalar

Fading process time offset, in seconds, specified as a numeric scalar. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`.

Data Types: `double`

`NTerms` — Number of oscillators used in fading path modeling

16 (default) | Optional | scalar power of 2

Number of oscillators used in fading path modeling, specified as a scalar power of 2. Optional. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`.

Data Types: `double`

`ModelType` — Rayleigh fading model type

`'GMEDS'` (default) | Optional | `'Dent'`

Rayleigh fading model type, specified as `'Dent'` or `'GMEDS'`. Optional. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`.

When `ModelType` is set to `'GMEDS'`, the Rayleigh fading is modeled using the Generalized Method of Exact Doppler Spread (GMEDS), as described in [4]. When `ModelType` is set to `'Dent'`, the Rayleigh fading is modeled using the modified Jakes fading model described in [3].

Data Types: `char`

`NormalizePathGains` — Model output normalization

`'On'` (default) | Optional | `'Off'`

Model output normalization, specified as `'On'` or `'Off'`. Optional. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`. If `NormalizePathGains` is set to `'On'`, the model output is normalized such that the average power is unity. If it is set to `'Off'`, the average output power is the sum of the powers of the taps of the delay profile.

Data Types: `char`

`InitPhase` — Phase initialization for the sinusoidal components of the model

`'Random'` (default) | Optional | scalar value | *N*-D array

Phase initialization for the sinusoidal components of the model, specified as `'Random'`, a scalar value, or an *N*-D array. Optional. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`. If `InitPhase` is set to `'Random'`, the phases are randomly initialized according to the seed value, `Seed`. If it is set to a scalar, that value, assumed to be in radians, is used to initialize the phases of all components. Otherwise, it is necessary to provide an array of size *N*-by-*L*-by-*P*-by-`NRxAnts` to initialize explicitly the phase in radians of each component. In that case, *N* is the number of phase initialization values per path, *L* is the number of paths, *P* is the number of transmit antennas, and `NRxAnts` is the number of receive antennas. $N = 2 \times$ `NTerms` when `ModelType` is set to `'GMEDS'` and $N =$ `NTerms` when `ModelType` is set to `'Dent'`.

Data Types: `char`

`Seed` — Random number generator seed

scalar value

Random number generator seed, specified as a scalar value. Set `Seed` to zero if you want a random seed. This parameter is required only if `DelayProfile` is set to a value other than `'Off'`. It is not required if `InitPhase` is not set to `'Random'`.

Data Types: `double`

`AveragePathGaindB` — Average gains of the discrete paths

vector

Average gains of the discrete paths, specified as a vector and expressed in dB. This parameter is required when `DelayProfile` is set to `'Custom'`.

Data Types: `double`

`PathDelays` — Delays of the discrete paths

vector

Delays of the discrete paths, specified as a vector and expressed in seconds. This parameter is required when `DelayProfile` is set to `'Custom'`. It must have the same size as `AveragePathGaindB`.

Data Types: `double`

`TxCorrelationMatrix` — Correlation between transmit antennas

matrix

Correlation between transmit antennas, specified as a P-by-P complex matrix. This parameter is required when `MIMOCorrelation` is set to `'Custom'`.

Data Types: `double | single`
Complex Number Support: Yes

`RxCorrelationMatrix` — Correlation between receive antennas

matrix

Correlation between receive antennas, specified as a complex matrix of size `NRxAnts`-by-`NRxAnts`. This parameter is required when `MIMOCorrelation` is set to `'Custom'`.

Data Types: `double | single`

Data Types: `struct`

`in` — Input samples

numeric matrix

Input samples, specified as a numeric matrix of size T-by-P, where P is the number of transmit antennas and T is the number of time-domain samples. Each column of in corresponds to the waveform at each of the transmit antennas.

Data Types: `double | single`
Complex Number Support: Yes

Output Arguments

`out` — Channel output signal

numeric matrix

Channel output signal, returned as a numeric matrix. Each column of out corresponds to the waveform at each of the receive antennas. out has the same number of rows as the input, in.

Data Types: `double | single`
Complex Number Support: Yes

`info` — Channel modeling information

structure

Channel modeling information, returned as a structure. info contains the following fields.

`ChannelFilterDelay` — Implementation delay of the internal channel filtering
scalar value

Implementation delay of the internal channel filtering, returned as a scalar value.

Data Types: `double`

`PathGains` — Complex gain of the discrete channel paths
N-D numeric array

Complex gain of the discrete channel paths, specified as an *N*-D numeric array of size *T*-by-*L*-by-*P*-by-`NRxAnts`, where *T* is the number of output samples, *L* is the number of paths, *P* is the number of transmit antennas, and `NRxAnts` is the number of receive antennas.

Data Types: `double`
Complex Number Support: Yes

`PathSampleDelays` — Delays of the discrete channel paths
row vector

Delays of the discrete channel paths, returned as a row vector. The delays are expressed in samples at the sampling rate specified in `model.SamplingRate`.

Data Types: `double`

Data Types: `struct`

References

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[3] Dent, P., G. E. Bottomley, and T. Croft. "Jakes Fading Model Revisited." *Electronics Letters*. Vol. 29, 1993, Number 13, pp. 1162–1163.

[4] Pätzold, Matthias, Cheng-Xiang Wang, and Bjørn Olav Hogstad. "Two New Sum-of-Sinusoids-Based Methods for the Efficient Generation of Multiple Uncorrelated Rayleigh Fading Waveforms." *IEEE Transactions on Wireless Communications*. Vol. 8, 2009, Number 6, pp. 3122–3131.

See Also

`lteHSTChannel` | `lteMovingChannel` | `lteOFDMModulate` | `lteSCFDMAModulate`

lteFrequencyCorrect

Frequency offset correction

Syntax

```
out = lteFrequencyCorrect(cfg,in,foffset)
```

Description

`out = lteFrequencyCorrect(cfg,in,foffset)` corrects for a specified frequency offset, foffset, in the time-domain waveform, in, by performing simple frequency modulation (FM). The parameters of the waveform, in, are specified in a settings structure, cfg, which must contain either the field `NDLRB` or `NULRB` to control whether a downlink or uplink signal is expected in in.

The input, foffset is the frequency offset, in hertz, present on the waveform, in. Therefore, the correction applied is FM modulation by –foffset.

Examples

Correct for Specified Frequency Offset

Perform frequency offset estimation and correction on an uplink signal, to which a frequency offset has been applied.

Generate uplink RMC A3-2.

```
[txWaveform,rgrid,cfg] = lteRMCULTool('A3-2',[1;0;0;1],'Fdd',2);
```

Apply an arbitrary frequency offset of 51.2 Hz.

```
t = (0:length(txWaveform)-1).'/cfg.SamplingRate;
txWaveform = txWaveform .* exp(1i*2*pi*51.2*t);
```

Estimate and display the frequency offset.

```
offset = lteFrequencyOffset(cfg,txWaveform);
```

```
fprintf('Frequency offset: %0.2fHz\n',offset);
```

```
Frequency offset: 51.20Hz
```

Correct for the frequency offset.

```
rxWaveform = lteFrequencyCorrect(cfg,txWaveform,offset);
```

Finally, perform SC-FDMA demodulation.

```
rxGrid = lteSCFDMADemodulate(cfg,rxWaveform);
```

Input Arguments

`cfg` — Waveform parameter settings

structure

Waveform parameter settings, specified as a structure. `cfg` must contain either the field NDLRB, to specify a downlink configuration, or the field NULRB, to specify an uplink configuration.

`NDLRB` — Number of downlink resource blocks

positive scalar integer

Number of downlink resource blocks, specified as a positive scalar integer. You must set this parameter field to specify a downlink configuration.

Data Types: `double`

`CyclicPrefix` — Downlink cyclic prefix length

`'Normal'` (default) | `'Extended'`

Downlink cyclic prefix length, specified as a string. Only set this parameter field if you are specifying a downlink configuration.

Data Types: `char`

`NULRB` — Number of uplink resource blocks

positive scalar integer

Number of uplink resource blocks, specified as a positive scalar integer. You must set this parameter field to specify a uplink configuration.

Data Types: `double`

`CyclicPrefixUL` — Uplink cyclic prefix length

`'Normal'` (default) | `'Extended'`

Uplink cyclic prefix length, specified as a string. Only set this parameter field if you are specifying an uplink configuration.

Data Types: `char`

Data Types: `struct`

`in` — Time-domain waveform

numeric column vector

Time-domain waveform, specified as a numeric column vector.

Data Types: `double` | `single`
Complex Number Support: Yes

`foffset` — Waveform frequency offset

scalar value

Waveform frequency offset, specified as a scalar value expressed in Hertz. The correction applied to `in` is FM modulation by –`foffset`.

Data Types: `double`

Output Arguments

`out` — Offset-corrected waveform

numeric column vector

Offset-corrected waveform, returned as a numeric column vector.

Data Types: `double` | `single`
Complex Number Support: Yes

See Also

`lteCellSearch` | `lteDLFrameOffset` | `lteFrequencyOffset` | `lteOFDMDemodulate` | `lteSCFDMADemodulate` | `lteULFrameOffset`

lteFrequencyOffset

Frequency offset estimation using cyclic prefix

Syntax

```
foffset = lteFrequencyOffset(cfgdl,waveform)
foffset = lteFrequencyOffset(cfgul,waveform)
[foffset, corr] = lteFrequencyOffset( ___ )
[foffset, corr] = lteFrequencyOffset( ___ ,toffset)
```

Description

`foffset = lteFrequencyOffset(cfgdl,waveform)` estimates the average frequency offset, foffset, of the time-domain waveform, waveform, by calculating correlation of the cyclic prefix. The parameters of waveform are given in the downlink settings structure, cfgdl. cfgdl must contain the field `NDLRB` to specify that a downlink signal is expected in waveform.

The output, foffset, is the measured frequency offset in hertz.

`foffset = lteFrequencyOffset(cfgul,waveform)` estimates the average frequency offset, foffset, of the time-domain waveform, waveform, by calculating correlation of the cyclic prefix. The parameters of waveform are given in the uplink settings structure, cfgul. cfgul must contain the field `NULRB` to specify that an uplink signal is expected in waveform.

The output, foffset, is the measured frequency offset in hertz.

`[foffset, corr] = lteFrequencyOffset( ___ )` also returns a complex matrix, corr, spanning one slot and containing the same number of antennas, or columns, as waveform. corr is the signal used to extract the timing of the correlation for the estimation of the frequency offset.

`[foffset, corr] = lteFrequencyOffset( ___ ,toffset)` provides control over the position in the correlator output used to estimate the frequency offset. If toffset is absent, or empty, the position in the correlator output used for frequency offset

estimation is the position of the peak magnitude of the correlator output. If toffset is present, toffset is the timing offset in samples from the start of the correlator output to the position used for the frequency offset estimation. This input allows for example a timing offset to be calculated externally on a signal of longer duration than the input waveform here, and then obtain a short-term frequency offset estimate while retaining the benefit of a longer-term timing estimate.

Note: If toffset is absent, the quality of the internal timing estimate is subject to the length and signal quality of the input waveform and, therefore, it may result in inaccurate frequency offset measurements.

Examples

Estimate Frequency Offset

Perform frequency offset estimation and correction on an uplink signal, to which a frequency offset has been applied.

Generate uplink RMC A3-2.

```
[txWaveform,rgrid,cfg] = lteRMCULTool('A3-2',[1;0;0;1],'Fdd',2);
```

Apply an arbitrary frequency offset of 51.2 Hz.

```
t = (0:length(txWaveform)-1).'/cfg.SamplingRate;
txWaveform = txWaveform .* exp(1i*2*pi*51.2*t);
```

Estimate and display the frequency offset.

```
offset = lteFrequencyOffset(cfg,txWaveform);
fprintf('Frequency offset: %0.2fHz\n',offset);
```

```
Frequency offset: 51.20Hz
```

Correct for frequency offset.

```
rxWaveform = lteFrequencyCorrect(cfg,txWaveform,offset);
```

Perform SC-FDMA demodulation.

```
rxGrid = lteSCFDMADemodulate(cfg,rxWaveform);
```

Input Arguments

`cfgdl` — Downlink configuration

structure

Downlink configuration, specified as a structure having the following fields. The parameter fields TDDConfig, SSC, and NSubframe are only required if DuplexMode is set to `'TDD'`.

`NDLRB` — Number of downlink resource blocks

positive scalar integer

Number of downlink resource blocks, specified as a positive scalar integer. You must set this parameter field to specify a downlink configuration.

Data Types: `double`

`CyclicPrefix` — Downlink cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Downlink cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`DuplexMode` — Duplex mode

`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, specified as a string. Optional.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration

0 (default) | Optional | nonnegative scalar integer (0...6)

Uplink or downlink configuration, specified as a nonnegative scalar integer between 0 and 6. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

`SSC` — Special subframe configuration

0 (default) | Optional | nonnegative scalar integer (0...9)

Special subframe configuration, specified as a nonnegative scalar integer between 0 and 9. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

`NSubframe` — Subframe number

scalar integer

Subframe number, specified as a scalar integer. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

Data Types: `struct`

`cfgUl` — Uplink configuration

structure

Uplink configuration, specified as a structure having the following fields. The parameter fields TDDConfig, , and NSubframe are only required if DuplexMode is set to `'TDD'`.

`NULRB` — Number of uplink resource blocks

positive scalar integer

Number of uplink resource blocks, specified as a positive scalar integer. You must set this parameter field to specify an uplink configuration.

Data Types: `double`

`CyclicPrefixUL` — Uplink cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Uplink cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`DuplexMode` — Duplex mode

`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, specified as a string. Optional.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration

0 (default) | Optional | nonnegative scalar integer (0...6)

Uplink or downlink configuration, specified as a nonnegative scalar integer between 0 and 6. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

`SSC` — Special subframe configuration

0 (default) | Optional | nonnegative scalar integer (0...9)

Special subframe configuration, specified as a nonnegative scalar integer between 0 and 9. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

`NSubframe` — Subframe number

scalar integer

Subframe number, specified as a scalar integer. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

Data Types: `struct`

`waveform` — Input time-domain waveform

numeric column vector

Input time-domain waveform, specified as a numeric column vector.

Data Types: `double | single`
Complex Number Support: Yes

`toffset` — Timing offset

scalar value

Timing offset, specified as a scalar value expressed in samples. Use `TOFFSET` to control the position in the correlator output used to estimate the frequency offset. If `TOFFSET` is absent, or empty, the position of the peak magnitude of the correlator output is used.

Data Types: `double | single`

Output Arguments

`foffset` — Average frequency offset estimate

scalar value

Average frequency offset estimate, returned as a scalar value expressed in hertz.

Data Types: `double` | `single`

`corr` — Correlation timing signal

numeric matrix

Correlation timing signal, returned as a numeric matrix. corr is a complex matrix that spans 1 slot and contains the same number of antennas, or columns, as waveform. It is the signal used to extract the timing of the correlation for the frequency offset estimation.

Data Types: `double` | `single`
Complex Number Support: Yes

See Also

`lteCellSearch` | `lteDLFrameOffset` | `lteFrequencyCorrect` | `lteOFDMDemodulate` | `lteSCFDMADemodulate` | `lteULFrameOffset`

lteHSTChannel

High speed train MIMO channel propagation conditions

Syntax

```
out = lteHSTChannel(model,in)
```

Description

`out = lteHSTChannel(model,in)` implements the high speed train (HST) MIMO channel model specified in [1] and [2]. The high speed train propagation condition is composed of a non-fading single path of unit amplitude and zero phase with a changing Doppler shift. The columns of matrix in correspond to the channel input waveforms at each transmit antenna. These are filtered by the channel model with the characteristics specified in structure model. The filtered waveform is stored in matrix, out, each of whose columns corresponds to the waveform at one of the receive antennas.

Examples

Model High Speed Train Propagation Channel

Filter the input subframes through an LTE high-speed train (HST) channel model. The first subframe is transmitted at time 0 seconds, t=0. Hence, initialize the channel by setting the `InitTime` parameter to 0 seconds.

```
rmc = lteRMCDL('R.10');
[txWaveform,txGrid,info] = lteRMCDLTool(rmc,[1;0;1]);
chcfg = struct('NRxAnts',1,'Ds',100,'Dmin',500,'Velocity',350);
chcfg.DopplerFreq = 5.0;
chcfg.SamplingRate = info.SamplingRate;
chcfg.InitTime = 0;
```

```
rxWaveform = lteHSTChannel(chcfg,txWaveform);
```

Input Arguments

`model` — High speed train propagation channel model
structure

High speed train propagation channel model, specified as a structure. model must contain the following fields.

`NRxAnts` — Number of receive antennas
positive integer scalar

Number of receive antennas, specified as a positive integer scalar. It must be equal to or greater than 1.

Data Types: `double`

`Ds` — Train-to-eNodeB double initial distance
numeric scalar

Train-to-eNodeB double initial distance, in meters, specified as a numeric scalar. `Ds/2` is the initial distance between the train and eNodeB.

Data Types: `double`

`Dmin` — eNodeB to railway track distance
scalar value

eNodeB to railway track distance, specified as a scalar value expressed in metres.

Data Types: `double`

`Velocity` — Train velocity
scalar value

Train velocity, specified as a scalar value expressed in kilometers per hour (km/h).

Data Types: `double`

`DopplerFreq` — Maximum Doppler frequency
scalar value

Maximum Doppler frequency, specified as a scalar value expressed in hertz.

Data Types: `double`

`SamplingRate` — Input signal sampling rate

scalar value

Input signal sampling rate, specified as a scalar value. It is the rate of each sample in the rows of the input matrix, in.

Data Types: `double`

`InitTime` — Doppler shift timing offset

scalar value

Doppler shift timing offset, specified as a scalar value expressed in seconds.

Data Types: `double`

`NormalizeTxAnts` — Transmit antenna number normalization

`'On'` (default) | Optional | `'Off'`

Transmit antenna number normalization, specified as a string. Optional. If the value is `'On'`, the model output is normalized by `1/sqrt(P)`, where P is the number of transmit antennas. This ensures that the output power per receive antenna is unaffected by the number of transmit antennas. If the value is `'Off'`, the output is not normalized by this factor.

Data Types: `char`

Data Types: `struct`

`in` — Channel input waveforms at transmit antennas

numeric matrix

Channel input waveforms at transmit antennas, specified as a numeric matrix. in has size T-by-P, where P is the number of antennas and T is the number of time-domain samples. Since each column of in corresponds to a transmit antenna, the number of columns must be 1, 2, or 4.

Data Types: `double | single`
Complex Number Support: Yes

Output Arguments

`out` — Filtered waveform

numeric matrix

Filtered waveform, returned as a numeric matrix. Each column of out corresponds to the waveform at one of the receive antennas.

Data Types: `double` | `single`
Complex Number Support: Yes

References

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteFadingChannel` | `lteMovingChannel` | `lteOFDMDemodulate` | `lteSCFDMADemodulate`

lteLayerDemap

Layer demapping onto scrambled and modulated codewords

Syntax

```
out = lteLayerDemap(in,ncw)
out = lteLayerDemap(in,ncw,txscheme)
out = lteLayerDemap(chs,in)
```

Description

`out = lteLayerDemap(in,ncw)` performs the layer demapping required to undo the processing described in [1], sections 5.3.2A and 6.3.3. The function returns `out`, a cell array containing one or two vectors of modulation symbols, one for each codeword. The function demaps the `NU` layers specified in the *M*-by-`NU` matrix `in` into `ncw` codewords using `'Port0'` transmission scheme if `NU = 1` and `'SpatialMux'` transmission scheme otherwise.

`out = lteLayerDemap(in,ncw,txscheme)` performs the layer demapping using the transmission scheme, txscheme.

`out = lteLayerDemap(chs,in)` demaps the *NU* layers specified in the *M*-by-*NU* matrix in into codeword vectors out, according to the parameters specified in the channel transmission configuration structure, chs.

The number of codewords is established from the number of modulation formats in the `Modulation` field. This value enables you to return the correct number of codewords when using chs specified in `ltePDSCH` or `ltePUSCH` on the transmit side. Alternatively, you can specify the number of codewords directly in the `NCodewords` field. The `NCodewords` field takes precedence if present.

Examples

Demap Codeword for Transmit Diversity

Map a codeword onto 4 symbols for `'TxDiversity'` transmission scheme.

```
nCodewords = 1;
codeword = ones(16,1);
nLayers = 4;
txScheme = 'TxDiversity';
layerMap = lteLayerMap(codeword,nLayers,txScheme);
```

Recover the codeword by demapping the 4 layers onto 1 codeword.

```
out = lteLayerDemap(layerMap,nCodewords,txScheme)
```

```
out =

    [16x1 double]
```

Demap Codeword for Spatial Multiplexing

Demap 4 symbols on four layers onto one codeword for the spatial multiplexing transmission scheme.

```
layerMap2 = lteLayerMap(ones(16,1),4);
out2 = lteLayerDemap(layerMap2,1)
```

```
out2 =

    [16x1 double]
```

Input Arguments

`in` — Modulation symbols

numeric matrix

Modulation symbols, specified as a *M*-by-*NU* numeric matrix consisting of *M* modulation symbols for *NU* transmission layers. You can generate this matrix by `lteDLDeprecode` or `ltePUSCHDeprecode`.

Data Types: `double`
Complex Number Support: Yes

ncw — Number of codewords

1 | 2

Number of codewords, specified as 1 or 2.

Data Types: `double`

txscheme — Transmission scheme

`'Port0'` (default) | `'TxDiversity'` | `'CDD'` | `'SpatialMux'` | `'MultiUser'` | `'Port5'` | `'Port7-8'` | `'Port8'` | `'Port7-14'`

Transmission scheme, specified as one of the following options.

- `'SpatialMux'` — Closed-loop spatial multiplexing.
- `'Port0'` — Single-antenna port, port 0.
- `'TxDiversity'` — Transmit diversity scheme.
- `'CDD'` — Large delay CDD scheme.
- `'MultiUser'` — Multiuser MIMO scheme.
- `'Port5'` — Single-antenna port, port 5.
- `'Port7-8'` — Single-antenna port, port 7 (**`NLayers`** = 1). Dual layer transmission, ports 7 and 8 (**`NLayers`** = 2).
- `'Port8'` — Single-antenna port, port 8.
- `'Port7-14'` — Up to 8-layer transmission, ports 7–14.

Data Types: `char`

chs — Channel-specific transmission configuration

structure

Channel-specific transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`TxScheme`	Optional	`'Port0'` (default), `'TxDiversity'`, `'CDD'`,	Transmission scheme, specified as one of the following options. • `'SpatialMux'` — Closed-loop spatial multiplexing.

Parameter Field	Required or Optional	Values	Description
		`'SpatialMux'` (default), `'MultiUser'`, `'Port5'`, `'Port7-8'`, `'Port8'`, `'Port7-14'`	• `'Port0'` — Single-antenna port, port 0. • `'TxDiversity'` — Transmit diversity scheme. • `'CDD'` — Large delay CDD scheme. • `'MultiUser'` — Multiuser MIMO scheme. • `'Port5'` — Single-antenna port, port 5. • `'Port7-8'` — Single-antenna port, port 7 (**`NLayers`** = 1). Dual layer transmission, ports 7 and 8 (**`NLayers`** = 2). • `'Port8'` — Single-antenna port, port 8. • `'Port7-14'` — Up to 8-layer transmission, ports 7–14.
Additionally, you must specify one of the following fields.			
`Modulation`	Required if `NCodewords` is not set	`'QPSK'`, `'16QAM'`, `'64QAM'`, cell array of strings	Modulation type, specified as a string or cell array of strings. If 2 blocks, each cell is associated with a transport block.
`NCodewords`	Required if `Modulation` is not set	1, 2	Number of codewords

Output Arguments

`out` — Modulation symbols

cell array of one or two vectors

Modulation symbols, specified as a cell array of one or two vectors. The cell array contains one or two vectors of symbols, one for each codeword.

Data Types: `cell`

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`lteDLDeprecode` | `lteLayerMap` | `ltePHICHDeprecode` | `ltePUSCHDeprecode` | `lteSymbolDemodulate` | `lteULDeprecode`

lteLayerMap

Layer mapping of modulated and scrambled codewords

Syntax

```
out = lteLayerMap(in,nu)
out = lteLayerMap(in,nu,txscheme)
out = lteLayerMap(chs,in)
```

Description

`out = lteLayerMap(in,nu)` performs layer mapping of codeword or codewords, in, onto nu layers. It carries out the layer mapping according to Section 5.3.2A and 6.3.3 of [1]. The function returns an *M*-by-nu matrix consisting of the modulation symbols for transmission upon nu layers. These transmission layers are formed by multiplexing the modulation symbols from either one or two codewords. The overall operation of the layer mapper is the transpose of that defined in the specification. In other words, the symbols for layers lie in columns rather than rows.

`out = lteLayerMap(in,nu,txscheme)` performs layer mapping using the transmission scheme, txscheme.

`out = lteLayerMap(chs,in)` performs layer mapping of codeword or codewords, in, according to the parameters in the channel transmission configuration structure, chs.

Examples

Map Codeword for Transmit Diversity

Map one codeword to four layers for the transmit diversity transmission scheme.

```
out1 = lteLayerMap(ones(40,1),4,'TxDiversity');
size(out1)
```

```
    10     4
```

Map Codeword for Spatial Multiplexing

Map one codeword to four layers for the spatial multiplexing transmission scheme.

```
out2 = lteLayerMap(ones(40,1),4);
size(out2)
```

```
    10     4
```

Input Arguments

`in` — Scrambled and modulated codeword or codewords

numeric vector | cell array of numeric vectors

Scrambled and modulated codeword or codewords, specified as a numeric vector or a cell array of numeric vectors. As a cell array, in contains one or two vectors of modulation symbols that result from the scrambling and modulation of DL-SCH or UL-SCH codewords.

`nu` — Number of transmission layers

1,...,8

Number of transmission layers, specified as a positive scalar integer between 1 and 8.

Data Types: `double`

`txscheme` — Transmission scheme

`'Port0'` (default) | `'TxDiversity'` | `'CDD'` | `'SpatialMux'` | `'MultiUser'` | `'Port5'` | `'Port7-8'` | `'Port8'` | `'Port7-14'`

Transmission scheme, specified as one of the following option strings.

- `'Port0'` — Single-antenna port, port 0. Default if `NLayers` is 1.
- `'TxDiversity'` — Transmit diversity scheme
- `'CDD'` — Large delay CDD scheme
- `'SpatialMux'` — Closed-loop spatial multiplexing. Default if `NLayers` is not 1.
- `'MultiUser'` — Multiuser MIMO scheme

- `'Port5'` — Single-antenna port, port 7, if `NLayers` is 1. Dual layer transmission, port 7 and 8, if `NLayers` is 2.
- `'Port7-8'` — Single-antenna port, port 0
- `'Port8'` — Single-antenna port, port 8
- `'Port7-14'` — Up to 8-layer transmission, ports 7–14

Data Types: `char`

chs — Channel-specific transmission configuration

structure

Channel-specific transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`NLayers`	Required	1, ..., 8	Number of transmission layers
`TxScheme`	Optional	`'Port0'` (default), `'TxDiversity'`, `'CDD'`, `'SpatialMux'` (default), `'MultiUser'`, `'Port5'`, `'Port7-8'`, `'Port8'`, `'Port7-14'`	Transmission scheme, specified as one of the following option strings. • `'Port0'` — Single-antenna port, port 0. Default if `NLayers` is 1. • `'TxDiversity'` — Transmit diversity scheme • `'CDD'` — Large delay CDD scheme • `'SpatialMux'` — Closed-loop spatial multiplexing. Default if `NLayers` is not 1. • `'MultiUser'` — Multiuser MIMO scheme • `'Port5'` — Single-antenna port, port 7, if `NLayers` is 1. Dual layer transmission, port 7 and 8, if `NLayers` is 2. • `'Port7-8'` — Single-antenna port, port 0 • `'Port8'` — Single-antenna port, port 8 • `'Port7-14'` — Up to 8-layer transmission, ports 7–14

Output Arguments

`out` — Modulation symbols

numeric matrix

Modulation symbols, returned as a numeric matrix. `out` is an *M*-by-nu matrix consisting of the modulation symbols for transmission upon nu layers.

Data Types: `double`
Complex Number Support: Yes

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDLPrecode` | `lteLayerDemap` | `ltePHICHPrecode` | `ltePUSCHPrecode` | `lteSymbolModulate` | `lteULPrecode`

lteMCS

Modulation and code scheme lookup

Syntax

```
[itbs,mod]= lteMCS(...)
[itbs,mod]= lteMCS()
[itbs,mod]= lteMCS(imcs)
```

Description

`[itbs,mod]= lteMCS(...)` returns the relevant transport block size index or indices, `itbs`, and constellation modulation order or orders, `mod`, after looking up the modulation and coding scheme table of [1], Table 7.1.7.1–1.

`[itbs,mod]= lteMCS()` returns the complete table of 32 rows. Each row gives the `itbs` and modulation orders for the `imcs` values 0, ..., 31. For the reserved values of `itbs` present in the table, `itbs` is set to `NaN`.

`[itbs,mod]= lteMCS(imcs)` returns one or more rows of the table specified in the vector `imcs`. The values of `imcs` must be in the range 0, ..., 31. If `imcs = –1`, the function interprets the value as a discontinuous transmission (DTX), in [2], Annex A.4.

Examples

Return the Transport Block Size Modulation Order

Use `lteMCS` to return the transport block size index and modulation order for `IMCS = 17`.

```
[ITBS,Modulation] = lteMCS(17)
```

```
ITBS =

    15
```

```
Modulation =

64QAM
```

Input Arguments

`imcs` — Modeling and coding scheme indices

vector | 0, ..., 31 | –1

Modeling and coding scheme indices, specified as a vector of values in the range 0, ..., 31.

If `imcs=–1`, the function interprets the value as a discontinuous transmission (DTX) where `itbs=-1` and `mod='QPSK'`. If `imcs` is scalar, `mod` is returned as a single string instead of a 1-element cell array of strings.

Data Types: `double`

Output Arguments

`itbs` — Transport block size indices

column vector

Transport block size indices, returned as a column vector. If `imcs=-1`, the function interprets the value as a discontinuous transmission (DTX), where `itbs=-1`, for which `lteTBS` yields `tbs = 0`.

`mod` — Modulation orders

1–column cell array of strings | 'QPSK' | '16QAM' | '64QAM'

Modulation orders, returned as a 1-column cell array of strings. If the value of `imcs=-1`, this is interpreted as a discontinuous transmission (DTX), and the value of `mod='QPSK'`. If `imcs` is a scalar, `mod` is returned as a single string.

References

[1] 3GPP TS 36.213. "Physical layer procedures." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.101. "User Equipment (UE) radio transmission and reception" *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDLSCH` | `lteTBS`

lteMIB

Master information block encoding and decoding

Syntax

```
mib = lteMIB(enb)
enb = lteMIB(mib)
enb = lteMIB(mib,enb)
```

Description

`mib = lteMIB(enb)` allows encoding and decoding of the master information block (MIB) broadcast control channel (BCCH) message from cell-wide settings.

It creates the 24-bit-long MIB message, mib, from the fields of cell-wide settings structure, enb.

`enb = lteMIB(mib)` performs the inverse processing of the preceding syntax, taking as input the MIB message bits, mib, and creating the cell-wide settings structure, enb.

`enb = lteMIB(mib,enb)` includes in the enb output structure any fields contained in the enb input structure. For any of the fields already present in the input structure, the value decoded from the MIB replaces the existing value.

Note: Within the MIB, the system frame number (SFN) is stored as `floor(SFN/4)`. Therefore, when enb is created from an MIB bit sequence, `enb.NFrame` satisfies `mod(enb.NFrame,4)==0` and the frame number modulo 4 must be established by other means. For example, this can be done by using the `nfmod4` output of `ltePBCHDecode`.

Examples

Decode MIB Message Bits

Decode the a set of master information block (MIB) message bits.

Decode the MIB message bits in the column vector, `mib`.

```
mib = [0,1,0,0,1,0,1,1,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0].';
enb = lteMIB(mib)
```

```
            NDLRB: 25
    PHICHDuration: 'Normal'
               Ng: 'One'
           NFrame: 828
```

Input Arguments

`enb` — Cell-wide settings

structure

Cell-wide settings, specified as a structure. enb can contain the following fields.

`NDLRB` — Number of downlink resource blocks

scalar value (6...110)

Number of downlink resource blocks, specified as a positive integer scalar value. NDLRB must be between 6 and 110.

Note: If NDLRB is a nonstandard bandwidth, not one of the set {6,15,25,50,75,100}, all ones are inserted into the first 3 bits, the *dl-Bandwidth* bit field, of the MIB message, mib.

Data Types: `double`

`Ng` — HICH group multiplier

`'Sixth' | 'Half' | 'One' | 'Two'`

HICH group multiplier, specified as a string.

Data Types: `char`

`NFrame` — Frame number

0 (default) | Optional | nonnegative scalar integer

Frame number, specified as a nonnegative scalar integer. Optional.

Data Types: `double`

`PHICHDuration` — PHICH duration

`'Normal'` (default) | Optional | `'Extended'`

PHICH duration, specified as a string. Optional.

Data Types: `char`

Data Types: `struct`

`mib` — MIB message bit sequence

24-bit column vector

MIB message bit sequence, specified as a 24-bit column vector.

Note: If the first 3 bits, the *dl-Bandwidth* bit field, of the MIB message do not contain the equivalent of a decimal between 0 and 5 (MSB first, corresponding to the RB set {6,15,25,50,75,100}), the returned NDLRB is 0.

Data Types: `double` | `int8` | `logical`

Output Arguments

`mib` — MIB message

24-bit column vector

MIB message, returned as a 24-bit column vector.

Note: If the enb`.`NDLRB input parameter field is a nonstandard bandwidth, not one of the set {6,15,25,50,75,100}, the first 3 bits of mib, the *dl-Bandwidth* bit field, are all ones.

Data Types: `int8`

`enb` — Cell-wide settings created from MIB

structure

Cell-wide settings created from MIB, returned as a structure. enb contains the following fields.

`NDLRB` — Number of downlink resource blocks

nonnegative scalar integer

Number of downlink resource blocks, returned as a nonnegative scalar integer.

Note: If the first 3 bits, the *dl-Bandwidth* bit field, of the input MIB message, mib, do not contain the equivalent of a decimal between 0 and 5 (MSB first, corresponding to the RB set {6,15,25,50,75,100}), NDLRB is 0. The MIB message should have 24 bits. Longer messages are truncated to 24 elements, while shorter messages are zero padded.

Data Types: `int32`

`PHICHDuration` — PHICH duration

`'Normal' | 'Extended'`

PHICH duration, returned as a string.

Data Types: `char`

`Ng` — HICH group multiplier

`'Sixth' | 'Half' | 'One' | 'Two'`

HICH group multiplier, specified as a string.

Data Types: `char`

`NFrame` — Frame number

scalar value

Frame number, specified as a scalar value.

Data Types: `int32`

Data Types: `struct`

See Also

`lteBCH | lteBCHDecode`

lteMovingChannel

Moving channel propagation conditions

Syntax

```
out = lteMovingChannel(model,in)
```

Description

`out = lteMovingChannel(model,in)` implements the moving propagation conditions specified in [1]. The columns of matrix in correspond to the channel input waveforms at each transmit antenna. These are filtered by the moving channel model specified in structure model. The filtered waveform is stored in matrix out, where each column corresponds to the waveform at each of the receive antennas.

in is a T-by-P matrix of input samples, where P is the number of antennas and T is the number of the time-domain samples. The number of columns must be 1, 2, or 4. The input waveforms are filtered with the delay profiles specified in the parameter structure, model. These delay profiles are resampled to match the input signal sampling rate.

This process introduces delay on top of the channel group delay.

As specified in [1], Scenario 1 implements an extended typical urban with 200 Hz Doppler shift (ETU200) Rayleigh fading model with changing delays. The Rayleigh fading model can be modeled using two different methods. The `'Dent'` method uses the modified Jakes fading model described in [2]. The `'GMEDS'` method uses the Generalized Method of Exact Doppler Spread (GMEDS) described in [3].

Scenario 2 consists of a single non-fading path with unit amplitude and zero phase degrees with changing delay. No AWGN is introduced internally in this model.

The time difference between the first multipath component and the reference time (assumed to be 0) follows a sinusoidal characteristic.

$$\Delta\tau = \frac{A}{2}(1+\sin(\Delta\omega(t+t_0)))$$

where the offset t_0 is

$$t_0 = InitTime + \frac{3\pi}{2(\Delta\omega)}$$

If `InitTime` is 0, the delay of the first multipath component is 0. If $t = 0$, $\Delta\tau = 0$. Relative delay between all multipath components is fixed.

For Scenario 1, `InitTime` also controls the fading process timing offset. Changing this value produces parts of the fading process at different points in time.

Examples

Model Moving Propagation Channel

Model a moving propagation channel and filter an input signal through it.

Filter the input signal through an LTE moving channel model. The waveform is transmitted at time 0 seconds, *t*=0. Hence, initialize the channel by setting the `InitTime` parameter to 0 seconds.

```
txWaveform = ones(500,1);
chcfg = struct('Seed',1,'NRxAnts',1,'MovingScenario','Scenario1');
chcfg.SamplingRate = 100000;
chcfg.InitTime = 0;
rxWaveform = lteMovingChannel(chcfg,txWaveform);
```

Input Arguments

`model` — Moving channel model

structure

Moving channel model, specified as a structure. model must contain the following fields.

`Seed` — Random number generator seed

scalar value

Random number generator seed, specified as a scalar value. Set `Seed` to zero if you want a random seed.

Data Types: `double`

`NRxAnts` — Number of receive antennas

positive integer scalar

Number of receive antennas, specified as a positive integer scalar. `NRxAnts` must be equal to or greater than 1.

Data Types: `double`

`MovingScenario` — Moving channel scenario

`'Scenario1' | 'Scenario2'`

Moving channel scenario, specified as a string.

Data Types: `char`

`SamplingRate` — Input signal sampling rate

scalar value

Input signal sampling rate, specified as a scalar value. It is the rate of each sample in the rows of the input matrix, in.

Data Types: `double`

`InitTime` — Fading process and timing adjustment offset

scalar value

Fading process and timing adjustment offset, specified as a scalar value expressed in seconds.

Data Types: `double`

`NormalizeTxAnts` — Transmit antenna number normalization

`'On'` (default) | Optional | `'Off'`

Transmit antenna number normalization, specified as a string. Optional. If the value is `'On'`, the model output is normalized by `1/sqrt(P)`, where P is the number of transmit antennas. This ensures that the output power per receive antenna is unaffected by the number of transmit antennas. If the value is `'Off'`, the output is not normalized by this factor.

Data Types: `char`

`NTerms` — Number of oscillators used in fading path modeling

16 (default) | Optional | scalar power of 2

Number of oscillators used in fading path modeling, specified as a scalar power of 2. Optional. This parameter is required only if `MovingScenario` is set to `'Scenario1'`.

Data Types: `double`

`ModelType` — Rayleigh fading model type

`'Dent'` (default) | Optional | `'GMEDS'`

Rayleigh fading model type, specified as `'Dent'` or `'GMEDS'`. Optional. This parameter is required only if `MovingScenario` is set to `'Scenario1'`.

Data Types: `char`

`NormalizePathGains` — Model output normalization

`'On'` (default) | Optional | `'Off'`

Model output normalization, specified as `'On'` or `'Off'`. Optional. This parameter is required only if `MovingScenario` is set to `'Scenario1'`. If `NormalizePathGains` is set to `'On'`, the model output is normalized such that the average power is unity. If it is set to `'Off'`, the average output power is the sum of the powers of the taps of the delay profile.

Data Types: `char`

Data Types: `struct`

`in` — Input samples

numeric matrix

Input samples, specified as a numeric matrix. in has size T-by-P, where P is the number of transmit antennas and T is the number of time-domain samples. The number of columns must be 1, 2, or 4. Each column of in corresponds to the waveform at each of the transmit antennas.

Data Types: `double` | `single`
Complex Number Support: Yes

Output Arguments

`out` — Filtered waveform

numeric matrix

Filtered waveform, returned as a numeric matrix. Each column of out corresponds to the waveform at each of the receive antennas.

Data Types: `double` | `single`
Complex Number Support: Yes

References

[1] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] Dent, P., G. E. Bottomley, and T. Croft. "Jakes Fading Model Revisited." *Electronics Letters*. Vol. 29, 1993, Number 13, pp. 1162–1163.

[3] Pätzold, Matthias, Cheng-Xiang Wang, and Bjørn Olav Hogstad. "Two New Sum-of-Sinusoids-Based Methods for the Efficient Generation of Multiple Uncorrelated Rayleigh Fading Waveforms." *IEEE Transactions on Wireless Communications*. Vol. 8, 2009, Number 6, pp. 3122–3131.

See Also

`lteFadingChannel` | `lteHSTChannel` | `lteOFDMModulate` | `lteSCFDMAModulate`

lteOFDMDemodulate

OFDM demodulation

Syntax

```
grid = lteOFDMDemodulate(enb,waveform)
grid = lteOFDMDemodulate(enb,waveform,cpfraction)
```

Description

`grid = lteOFDMDemodulate(enb,waveform)` performs OFDM demodulation of the time-domain waveform, waveform, given the cell-wide settings structure, enb.

The demodulation performs one FFT operation per received OFDM symbol in order to recover the received subcarrier values. These values are then used to construct each column of the output resource array, grid. The FFT is positioned partway through the cyclic prefix to allow for a certain degree of channel delay spread while avoiding the overlap between adjacent OFDM symbols. The particular position of the FFT chosen here avoids the OFDM symbol overlapping used in `lteOFDMModulate`. Since the FFT is performed away from the original zero phase point on the transmitted subcarriers, a phase correction is applied to each subcarrier after the FFT. Then, the received subcarriers are extracted from the FFT bins, skipping unused frequency bins at either end of the spectrum and the central DC frequency bin. These extracted subcarriers form the columns of the output grid.

The sampling rate of the time-domain waveform, waveform, must be the same as used in `lteOFDMModulate` for the specified number of resource blocks, `NDLRB`. waveform must also be time-aligned such that the first sample is the first sample of the cyclic prefix of the first OFDM symbol in a subframe. This alignment can be achieved by using `lteDLFrameOffset`.

`grid = lteOFDMDemodulate(enb,waveform,cpfraction)` allows specification of the position of the demodulation through the cyclic prefix.

Examples

Perform OFDM Demodulation

Perform modulation and demodulation of Test Model 1.1 5MHz.

```
cfg = lteTestModel('1.1','5MHz');
txWaveform = lteTestModelTool(cfg);
rxGrid = lteOFDMDemodulate(cfg,txWaveform);
```

Input Arguments

`enb` — Cell-wide settings
structure

Cell-wide settings, specified as a structure. enb contains the following fields.

`NDLRB` — Number of downlink resource blocks
positive integer scalar

Number of downlink resource blocks, specified as a positive integer scalar.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

Data Types: `struct`

`waveform` — Time-domain waveform
numeric matrix

Time-domain waveform, specified as a numeric matrix of size T-by-P, where P is the number of antennas and T is the number of time-domain samples. T is given by $T = K \times 30720 / 2048 \times N_{\text{fft}}$, where N_{fft} is the IFFT size and K is the number of subframes in the input, grid. waveform must be time-aligned such that the first sample is the first sample of the cyclic prefix of the first OFDM symbol in a subframe.

Data Types: `double`
Complex Number Support: Yes

`cpfraction` — Demodulation position

0.55 (default) | scalar value

Demodulation position, specified as a scalar between 0 and 1, with 0 representing the start of the cyclic prefix and 1 representing the end of the cyclic prefix. The default value, 0.55, allows for the default level of windowing in `lteOFDMModulate`

Data Types: `double`

Output Arguments

`grid` — Resource elements

3-D numeric array

Resource elements, returned as a 3-D numeric array. `grid` stores the resource elements for a number of subframes across all configured antenna ports. It is an $M \times N \times P$ array, where M is the number of subcarriers, N is the number of OFDM symbols, and P is the number of antennas.

Data Types: `double`
Complex Number Support: Yes

See Also

`lteDLChannelEstimate` | `lteDLFrameOffset` | `lteDLPerfectChannelEstimate` | `lteOFDMInfo` | `lteOFDMModulate`

lteOFDMModulate

OFDM modulation

Syntax

```
[waveform,info] = lteOFDMModulate(enb,grid)
[waveform,info] = lteOFDMModulate(enb,grid,windowing)
```

Description

`[waveform,info] = lteOFDMModulate(enb,grid)` performs DC subcarrier insertion, inverse fast Fourier transform (IFFT) calculation, cyclic prefix insertion, and optional raised cosine windowing and overlapping of adjacent OFDM symbols of the complex symbols in the resource array, grid. grid is a 3-D array containing the resource elements (REs) for a number of subframes across all configured antenna ports, as described in "Data Structures". It could also be multiple concatenated matrices to give multiple subframes, using concatenation across the columns or second dimension. Each of the antenna planes in grid is OFDM modulated and together comprise the columns of waveform.

grid is an $M \times N \times P$ array, where M is the number of subcarriers, N is the number of OFDM symbols, and P is the number of antennas. M must be 12 ×enb.NDLRB, where enb.NDLRB must be between 6 and 110. N must be a multiple of the number of symbols in a subframe, L, where L is 14 for normal cyclic prefix and 12 for extended cyclic prefix.

waveform is a T-by-P matrix, where P is the number of antennas and T is the number of the time-domain samples.

grid can span multiple subframes. Windowing and overlapping are applied between all adjacent OFDM symbols, including the last of one subframe and the first of the next. Therefore, a different result is obtained than if `lteOFDMModulate` is called on individual subframes and then those time-domain waveforms are concatenated. In that case, the resulting waveform has discontinuities at the start or end of each subframe. It is recommended that all subframes for OFDM modulation first be concatenated prior to calling `lteOFDMModulate` on the resulting multi-subframe array. However, individual subframes can be OFDM modulated and the resulting multisubframe time-domain waveform created by manual overlapping.

If enb.Windowing is absent, a default value for the number of windowed and overlapped samples is used. The default value is chosen as a function of enb.NDLRB to compromise between the effective duration of cyclic prefix, and thus the channel delay spread tolerance, and the spectral characteristics of the transmitted signal, not considering any additional FIR filtering. The value used is returned in enb.Windowing. If `enb.Windowing` is present, it must be even. The issues concerning concatenation of subframes before OFDM modulation do not apply when enb.Windowing is zero.

`[waveform,info] = lteOFDMModulate(enb,grid,windowing)` allows control of the number of windowed and overlapped samples used in the time-domain windowing, specified by the windowing parameter. The value of enb.Windowing, if present, is ignored, and the output, info.Windowing is set to windowing.

Examples

Perform OFDM Modulation

Perform OFDM modulation of one subframe of random uniformly-distributed noise using a 10MHz two-antenna configuration.

```
enb = struct('NDLRB',50,'CyclicPrefix','Normal','CellRefP',2);
d = lteDLResourceGridSize(enb);
rgrid = complex(rand(d)-0.5,rand(d)-0.5);
waveform = lteOFDMModulate(enb,rgrid);
```

Input Arguments

`enb` — Cell-wide settings
structure

Cell-wide settings, specified as a structure. enb can contain the following fields.

`NDLRB` — Number of downlink resource blocks
positive scalar integer

Number of downlink resource blocks, specified as a positive scalar integer.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`Windowing` — OFDM sample span

Optional | even scalar integer

OFDM sample span, specified as an even scalar integer. Optional. Windowing is the number of time-domain samples over which windowing and overlapping of OFDM symbols are applied.

Data Types: `double`

Data Types: `struct`

`grid` — Resource elements

3-D numeric array

Resource elements, specified as a 3-D numeric array. `grid` stores the resource elements for a number of subframes across all configured antenna ports. `grid` is an $M \times N \times P$ array, where M is the number of subcarriers, N is the number of OFDM symbols, and P is the number of antennas.

Data Types: `double`
Complex Number Support: Yes

`windowing` — OFDM sample span

even scalar integer

OFDM sample span, specified as an even scalar integer. This input argument controls the number of windowed and overlapped samples used in the time-domain windowing. This value overwrites the value of the parameter field `enb.Windowing`, if present.

Data Types: `double`

Output Arguments

`waveform` — OFDM modulated waveform

numeric matrix

OFDM modulated waveform, returned as a numeric matrix of size T-by-P, where P is the number of antennas and T is the number of time-domain samples.

$T = K \times 30720 / 2048 \times N_{\text{fft}}$ where N_{fft} is the IFFT size and K is the number of subframes in the input grid. N_{fft} is a function of the number of resource blocks (NRB), as shown in the following table.

NRB	N_{fft}
6	128
15	256
25	512
50	1024
75	2048
100	2048

In general, N_{fft} is the smallest power of 2 greater than or equal to `12*NRB/0.85`. It is the smallest FFT that spans all subcarriers and results in a bandwidth occupancy, `12*NRB/Nfft`, of no more than 85%.

Data Types: `double`
Complex Number Support: Yes

`info` — OFDM modulated waveform information

structure

OFDM modulated waveform information, returned as a structure. info contains the following fields.

`SamplingRate` — Time-domain waveform sampling rate

scalar value

Time-domain waveform sampling rate, returned as a scalar value.
`SamplingRate` = 30.72 MHz / 2048 × N_{fft}.

Data Types: `double`

`Nfft` — Number of FFT points

scalar power of 2

Number of FFT points, returned as a scalar power of 2. `Nfft` is the smallest power of 2 greater than or equal to 12 × `NDLRB` / 0.85. It is the smallest FFT that spans all

subcarriers and results in a bandwidth occupancy ($12 \times$ NDLRB / N_{fft}) of no more than 85%.

Data Types: `uint32`

`Windowing` — OFDM sample span
even integer scalar

OFDM sample span, returned as an even integer scalar. This parameter is the number of time-domain samples over which windowing and overlapping of OFDM symbols are applied.

Data Types: `int32`

Data Types: `struct`

More About

Algorithms

Windowing

The use of the IFFT within the OFDM modulator constitutes the use of a rectangular pulse shape. This means that discontinuities occur from one OFDM symbol to the next, resulting in out of band emissions. (Alternatively, considering the frequency domain, the frequency response of this rectangular pulse shape is a sinc pulse.) The discontinuities between OFDM symbols can be reduced by using windowing, which smooths the transitions between OFDM symbols. Within LTE System Toolbox, the windowing is performed as follows:

For **`Windowing`** = N samples, the cyclic prefix added to the nominal OFDM symbol is further extended by N samples.

This extended waveform is windowed by pointwise multiplication in the time domain with a raised cosine window, which applies a taper to the first N and last N samples, with all other values being 1. The values, y, in the first N samples are given by:

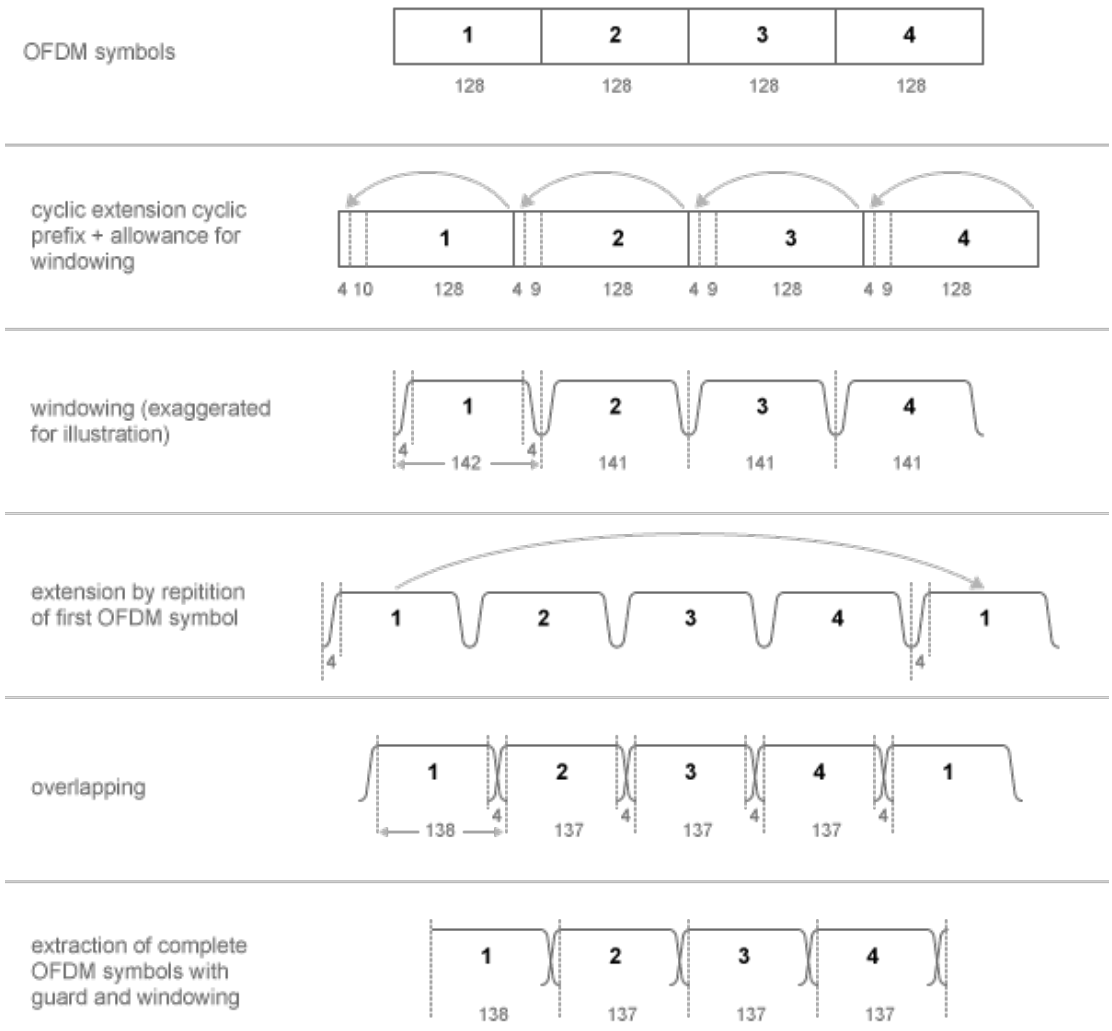
$$y = \frac{1}{2}\left(1 - \sin\left(\pi \frac{N + 1 - 2i}{2N}\right)\right), \text{ where } i = 1 \ldots N$$

The values in the last *N* samples are the same values in reverse order.

The windowed OFDM symbols are then overlapped by commencing transmission of each windowed OFDM symbol *N* samples prior to the end of the previous OFDM symbol. This ensures that the time between OFDM symbols is maintained as required by the standard. Note that the taper at the start of the first OFDM symbol for transmission is removed and is overlapped with the taper at the end of the last OFDM symbol.

Processing

The processing performed by this function is illustrated in the following diagram.



The number of samples used for windowing depends on the cyclic prefix length, normal or extended, and the number of resource blocks. The number of samples is chosen in accordance with the *maximum* values implied by tables F.5.3-1 and F.5.4-1 in [1].

Number of resource blocks (NRB)	Windowing samples for normal cyclic prefix	Windowing samples for extended cyclic prefix
6	4	4
15	6	6
25	4	4
50	6	6
75	8	8
100	8	8

The number of windowing samples is a compromise between the effective duration of cyclic prefix, and therefore the channel delay spread tolerance, and the spectral characteristics of the transmitted signal, not considering any additional FIR filtering. For a larger amount of windowing, the effective duration of the cyclic prefix is reduced but the transmitted signal spectrum has smaller out-of-band emissions.

References

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDLResourceGrid` | `lteFadingChannel` | `lteHSTChannel` | `lteMovingChannel` | `lteOFDMDemodulate` | `lteOFDMInfo`

lteOFDMInfo

OFDM modulation related information

Syntax

```
info = lteOFDMInfo(enb)
```

Description

`info = lteOFDMInfo(enb)` provides information related to the OFDM modulation performed by `lteOFDMModulate`, given the cell-wide settings structure, enb.

Examples

Get Information Related to OFDM Modulation

Find the sampling rate of a 50RB, or 10MHz, waveform after OFDM modulation.

```
enb = struct('NDLRB',50,'CyclicPrefix','Normal');
lteOFDMInfo(enb)
```

```
    SamplingRate: 15360000
            Nfft: 1024
       Windowing: 6
```

Input Arguments

enb — Cell-wide settings

structure

Cell-wide settings, specified as a structure. enb contains the following fields.

NDLRB — Number of downlink resource blocks

scalar value

Number of downlink resource blocks, specified as a positive integer scalar value.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`Windowing` — OFDM sample span

Optional | even integer scalar

OFDM sample span, specified as an even integer scalar. Number of time-domain samples over which windowing and overlapping of OFDM symbols are applied. Optional.

Data Types: `double` | `single` | `uint8` | `uint16` | `uint32` | `uint64` | `int8` | `int16` | `int32` | `int64`

Data Types: `struct`

Output Arguments

`info` — OFDM information

structure

OFDM information, returned as a structure. info contains the following fields.

`SamplingRate` — Sampling rate of the OFDM modulator

integer scalar value

Sampling rate of the OFDM modulator, returned as an integer scalar value.

Data Types: `double`

`Nfft` — Number of FFT points

scalar power of 2

Number of FFT points used in the OFDM modulator, returned as a scalar power of 2.

Data Types: `uint32`

`Windowing` — OFDM sample span

even integer scalar

OFDM sample span, returned as an even integer scalar Number of time-domain samples over which windowing and overlapping of OFDM symbols are applied.

If `enb.Windowing` is absent, `info.Windowing` returns a default value chosen as a function of `enb.NDLRB` to compromise between the effective duration of cyclic prefix (and therefore the channel delay spread tolerance) and the spectral characteristics of the transmitted signal (not considering any additional FIR filtering). See `lteOFDMModulate` for details.

Data Types: `int32`

Data Types: `struct`

See Also

`lteDLResourceGridSize` | `lteOFDMModulate`

ltePBCH

Physical broadcast channel

Syntax

```
sym = ltePBCH(enb,cw)
```

Description

`sym = ltePBCH(enb,cw)` returns a matrix containing the complex symbols of the Physical Broadcast Channel (PBCH) for cell-wide settings structure, enb, and codeword, cw. The function performs all physical channel processing steps, including the stages of scrambling, QPSK modulation, layer mapping, and precoding. The size of the output matrix, sym, is *N*-by-`CellRefP`, where *N* is the number of modulation symbols for one antenna port and `CellRefP` is the number of antenna ports.

The BCH transport channel consumes information bits every 40 ms. The coded transport block is then passed to PBCH for physical channel processing. The PBCH is transmitted in the first subframe of every frame, so four successive frames are required to transmit one transport block. As the scrambling sequence is initialized at the boundary of every 40 ms, this function expects 40 ms worth of data. For example, it expects 1920 bits for normal cyclic prefix, or 1728 bits for extended cyclic prefix. The output of this function then needs to be demultiplexed into quarter length blocks for carriage on the first subframe in each 10 ms frame.

cw is a vector containing the bit values of the PBCH codeword for modulation.

Examples

Generate PBCH Symbols

Generate physical broadcast channel (PBCH) symbols using the master information block (MIB).

First, generate the MIB for the reference measurement channel (RMC) R.0, as specified in [1], for cell-wide settings, `enb`. Pass the MIB through broadcast channel (BCH) transport channel coding. Finally, generate and display the PBCH symbols.

```
enb = lteRMCDL('R.0');
mib = lteMIB(enb);
bchCoded = lteBCH(enb,mib);
pbchSymbols = ltePBCH(enb,bchCoded);
pbchSymbols(1:10)
```

```
   0.7071 + 0.7071i
   0.7071 - 0.7071i
   0.7071 + 0.7071i
  -0.7071 + 0.7071i
  -0.7071 + 0.7071i
   0.7071 + 0.7071i
  -0.7071 + 0.7071i
  -0.7071 + 0.7071i
  -0.7071 + 0.7071i
  -0.7071 - 0.7071i
```

Input Arguments

`enb` — Cell-wide settings
structure

Cell-wide settings, specified as a structure. enb must contain the following fields.

`NCellID` — Physical layer cell identity
scalar integer

Physical layer cell identity, specified as a scalar integer.

Data Types: `double`

`CellRefP` — Number of cell-specific reference signal antenna ports
1 | 2 | 4

Number of cell-specific reference signal antenna ports, specified as 1, 2, or 4.

Data Types: `double`

Data Types: `struct`

`cw` — PBCH codeword
vector

PBCH codeword, specified as a vector. cw contains the bit values of the PBCH codeword for modulation.

Data Types: `double` | `single` | `uint8` | `uint16` | `uint32` | `uint64` | `int8` | `int16` | `int32` | `int64`

Output Arguments

`sym` — PBCH symbols

numeric matrix

PBCH symbols, returned as a numeric matrix. sym contains the complex symbols of the Physical Broadcast Channel (PBCH) for cell-wide settings, enb, and codeword, cw. Its size is *N*-by-`CellRefP`, where *N* is the number of modulation symbols for one antenna port and `CellRefP` is the number of antenna ports.

Data Types: `double`
Complex Number Support: Yes

References

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteBCH` | `ltePBCHDecode` | `ltePBCHIndices` | `ltePBCHPRBS`

ltePBCHDecode

Physical broadcast channel decoding

Syntax

```
[bits,symbols,nfmod4,trblk,cellrefp] = ltePBCHDecode(enb,sym)
[bits,symbols,nfmod4,trblk,cellrefp] = ltePBCHDecode(enb,sym,hest,
noiseest)
[bits,symbols,nfmod4,trblk,cellrefp] = ltePBCHDecode(enb,sym,hest,
noiseest,alg)
```

Description

`[bits,symbols,nfmod4,trblk,cellrefp] = ltePBCHDecode(enb,sym)` returns a vector of soft bits, bits, a vector of received constellation complex symbols, symbols, frame number (modulo 4), nfmod4, decoded BCH information bits, trblk, and number of cell-specific reference signal antenna ports, cellrefp, resulting from performing the inverse of Physical Broadcast Channel (PBCH) processing. See [2] and `ltePBCH` for details.

The function performs the inverse of PBCH processing on the matrix of complex modulated PBCH symbols, sym, given a cell-wide settings structure, enb. The channel inverse processing includes deprecoding, symbol demodulation, and descrambling.

sym must be a matrix of `NRE`-by-`NRxAnts` complex modulated PBCH symbols. `NRE` is the number of QPSK symbols per antenna assigned to the PBCH and `NRxAnts` is the number of receive antennas. sym can contain 1 to 4 subframes' worth of PBCH data. The function is capable of decoding PBCH data scrambled with any scrambling sequence phase. Thus, there is no restriction on the input sym to be aligned at the 40 ms boundary.

bits is a vector of soft bits and symbols is a vector of received constellation complex symbols. bits is optionally scaled by channel state information (CSI) calculated during the equalization process. nfmod4 is the system frame number modulo 4, `mod(NFrame,4)`, obtained when determining the scrambling phase of the input PBCH symbols, sym. trblk is the decoded BCH information bits (24 bits) and cellrefp is the number of cell-specific reference signal antenna ports determined during the BCH decoding process. nfmod4, trblk, and cellrefp are determined by successful

synchronization with the scrambling sequence, which gets initialized every 40 ms. The true number of transmitted cell-specific reference signals is returned in cellrefp, and is searched for by attempting decoding with cellrefp equal to 1, 2, or 4. `enb.CellRefP` is attempted first to ensure that symbols contains the expected constellation and bits contains the expected soft bit estimates for the specified value. Decoding may succeed with a different value of cellrefp under good conditions, but may give unexpected bits and symbols. `enb.CellRefP` is optional; if it is not provided, the true number of transmitted cell-specific reference signals is established by the search and returned in cellrefp.

`[bits,symbols,nfmod4,trblk,cellrefp] = ltePBCHDecode(enb,sym,hest, noiseest)` performs the decoding of the complex PBCH symbols, sym, using cell-wide settings, enb, the channel estimate, hest, and the noise estimate, noiseest. For the `TxDiversity` transmission scheme (cellrefp = 2 or cellrefp = 4), the reception is performed using an OSFBC (Orthogonal Space Frequency Block Code) decoder. For the `Port0` transmission scheme (cellrefp = 1), the reception is performed using MMSE equalization.

hest is a 3-D `NRE`-by-`NRxAnts`-by-*P* array, where `NRE` are the frequency and time locations corresponding to the PBCH resource element positions (a total of `NRE` positions), `NRxAnts` is the number of receive antennas, and *P* is the number of cell-specific reference signal antennas.

noiseest is an estimate of the noise power spectral density per resource element on the received subframe. This estimate is provided by `lteDLChannelEstimate`.

`[bits,symbols,nfmod4,trblk,cellrefp] = ltePBCHDecode(enb,sym,hest, noiseest,alg)` provides control over weighting the output soft bits, bits, with channel state information (CSI) calculated during the equalization stage using the algorithmic configuration structure, alg.

Examples

Encode and Decode BCH and PBCH Data

Perform encoding and decoding of BCH and PBCH data.

```
x = round(rand(24,1));
enb = lteRMCDL('R.0');
bch = lteBCH(enb,x);
pbch = ltePBCH(enb,bch);
```

```
demod = ltePBCHDecode(enb,pbch);
decoded = lteBCHDecode(enb,demod);
```

Decode PBCH Symbols Generated from MIB

Decode PBCH symbols that were generated from the master information block (MIB) for RMC R.0.

Use the MIB for RMC R.0, as specified in [1], to generate the PBCH symbols for cell-wide settings, `enb`. Then, decode the PBCH symbols and find the number of cell-specific reference signal antenna ports, `cellrefp`.

```
enb = lteRMCDL('R.14');
mib = lteMIB(enb);
bchBits = lteBCH(enb,mib);
quarterLen = length(bchBits)/4;
pbchSymbols = ltePBCH(enb,bchBits(1:quarterLen));
[bits,~,~,~,cellrefp] = ltePBCHDecode(enb,pbchSymbols);
cellrefp
```

```
           4
```

Input Arguments

`enb` — Cell-wide settings

structure

Cell-wide settings, specified as a structure. enb contains the following fields.

`NCellID` — Physical layer cell identity

scalar integer

Physical layer cell identity, specified as a scalar integer.

Data Types: `double`

`CellRefP` — Number of cell-specific reference signal antenna ports

Optional | 1 | 2 | 4

Number of cell-specific reference signal antenna ports, specified as 1, 2, or 4. Optional. The default is to establish CellRefP by decoding the input symbols, sym.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length

`'normal'` (default) | Optional | `'Extended'` | `'Extended'`

Cyclic prefix length, specified as a string.

Data Types: `struct`

`sym` — Complex modulated PBCH symbols

numeric matrix

Complex modulated PBCH symbols, specified as a numeric matrix. sym must have size `NRE`-by-`NRxAnts`, where `NRE` is the number of QPSK symbols per antenna assigned to the PBCH and `NRxAnts` is the number of receive antennas. sym can contain 1 to 4 subframes' worth of PBCH data.

Data Types: `double`
Complex Number Support: Yes

`hest` — Channel estimate

3-D array

Channel estimate, specified as a 3-D array of size `NRE`-by-`NRxAnts`-by-P, where `NRE` are the frequency and time locations corresponding to the PBCH resource element positions (a total of `NRE` positions), `NRxAnts` is the number of receive antennas, and P is the number of cell-specific reference signal antennas.

Data Types: `double`
Complex Number Support: Yes

`noiseest` — Noise estimate

numeric scalar

Noise estimate, specified as a numeric scalar. It is an estimate of the noise power spectral density per resource element on the received subframe. This estimate is provided by the `lteDLChannelEstimate` function.

Data Types: `double`

`alg` — Algorithmic configuration

structure

Algorithmic configuration, specified as a structure. The structure must have the following field.

`CSI` — Channel state information setting

`'On'` (default) | Optional | `'Off'`

Channel state information weighting setting, specified as `'On'` or `'Off'`. Optional. It controls whether bits is weighted with channel state information (CSI) calculated during the equalization stage.

Data Types: `char`

Data Types: `struct`

Output Arguments

`bits` — Decoded PBCH soft bits

numeric column vector

Decoded PBCH soft bits, returned as a numeric column vector. bits is optionally scaled by channel state information.

Data Types: `double`

`symbols` — Received constellation of complex symbols

complex numeric column vector

Received constellation of complex symbols, returned as a complex numeric column vector.

Data Types: `double`
Complex Number Support: Yes

`nfmod4` — System frame number modulo 4

integer scalar

System frame number modulo 4, `mod(NFrame,4)`, returned as an integer scalar. nfmod4 is obtained when determining the scrambling phase of the input PBCH symbols, sym.

Data Types: `double`

`trblk` — Decoded BCH information bits

24-by-1 numeric column vector

Decoded BCH information bits, returned as a 24-by-1 numeric column vector.

Data Types: `int8`

`cellrefp` — Number of cell-specific reference signal antenna ports

Optional | 1 | 2 | 4

Number of cell-specific signal antenna ports, returned as 1, 2, or 4. Optional. The number of ports is determined during the BCH decoding.

Data Types: `uint32`

References

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteBCHDecode` | `ltePBCH` | `ltePBCHIndices` | `ltePBCHPRBS`

ltePBCHIndices

PBCH resource element indices

Syntax

```
ind = ltePBCHIndices(enb)
ind = ltePBCHIndices(enb,opts)
```

Description

`ind = ltePBCHIndices(enb)` returns an *N*-by-`CellRefP` matrix of resource element (RE) indices for the Physical Broadcast Channel (PBCH) given the parameter fields of structure enb. By default, the indices are returned in 1-based linear indexing form that can directly index elements of a 3-D array representing the subframe resource grid for `CellRefP` antennas. These indices are ordered as the PBCH modulation symbols should be mapped. Alternative indexing formats can also be generated. The PBCH is only transmitted in first subframe of each frame.

`ind = ltePBCHIndices(enb,opts)` allows control of the format of the returned indices through a cell array, opts, of option strings.

Examples

Generate PBCH RE Indices

Generate 0-based physical broadcast channel (PBCH) resource element (RE) indices in linear form.

```
enb = lteRMCDL('R.14');
ind = ltePBCHIndices(enb,{'Obased'});
ind(1:4,:)
```

```
        4465       12865       21265       29665
        4466       12866       21266       29666
        4468       12868       21268       29668
        4469       12869       21269       29669
```

For the 4 antenna case, the result has four columns.

Input Arguments

`enb` — Cell-wide settings
structure

Cell-wide settings, specified as a structure. enb contains the following fields.

`NDLRB` — Number of downlink resource blocks
scalar value

Number of downlink resource blocks, specified as a scalar value.

`NCellID` — Physical layer cell identity
scalar integer

Physical layer cell identity, specified as a scalar integer.

`CyclicPrefix` — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

`CellRefP` — Number of cell-specific reference signal antenna ports
1 (default) | Optional | 2 | 4

Number of cell-specific reference signal antenna ports, specified as 1, 2, or 4. Optional.

`NSubframe` — Subframe number
0 (default) | Optional | scalar integer

Subframe number, specified as a scalar integer. Optional.

Data Types: `struct`

`opts` — Index formatting options
cell array

Index formatting options, specified as a cell array. opts can contain the following option strings.

`Indexing style` — Form of returned indices
`'ind'` (default) | `'sub'`

Form of returned indices, specified as `'ind'` for linear indexing or `'sub'` for [subcarrier, symbol, antenna] subscript row style.

`Index base` — Index base of returned indices
`'1based'` (default) | `'0based'`

Index base of returned indices, specified as `'1based'` for 1-based indices or `'0based'` for 0-based indices.

Data Types: `char` | `cell`

Output Arguments

`ind` — PBCH resource element indices
numeric matrix

PBCH resource element indices, returned as a numeric matrix of size *N*-by-`CellRefP`.

Data Types: `double`

See Also
`ltePBCH` | `ltePBCHDecode` | `ltePBCHPRBS`

ltePBCHPRBS

PBCH pseudorandom scrambling sequence

Syntax

```
seq = ltePBCHPRBS(enb,n)
seq = ltePBCHPRBS(enb,n,mapping)
```

Description

`seq = ltePBCHPRBS(enb,n)` returns a vector with the first n outputs of the Physical Broadcast Channel (PBCH) scrambling sequence when initialized with the structure enb.

`seq = ltePBCHPRBS(enb,n,mapping)` allows control over the format of the returned sequence, seq, through the string mapping. Valid formats are `'binary'`, the default, and `'signed'`. The `'binary'` format maps true to 1 and false to 0. The `'signed'` format maps true to –1 and false to 1.

Examples

Compare Pseudorandom Scrambling Sequences

Compare the PBCH scrambling sequence generated using both generic and PBCH-specific pseudorandom binary sequence generators.

```
enb = lteRMCDL('R.0');
prbsSeq = ltePRBS(enb.NCellID,5);
pbchPrbsSeq = ltePBCHPRBS(enb,5);
isequal(prbsSeq,pbchPrbsSeq)
```

```
     1
```

The generic pseudorandom binary scrambling sequence equals the PBCH-specific pseudorandom binary scrambling sequence.

Generate PBCH Pseudorandom Scrambling Sequence from MIB

Generate the PBCH scrambling sequence for a cell-wide settings structure, `enb`. The length of the scrambling sequence is the number of BCH transport bits.

```
enb = lteRMCDL('R.0');
mib = lteMIB(enb);
bchBits=lteBCH(enb,mib);
pbchPrbsSeq = ltePBCHPRBS(enb,length(bchBits));
pbchPrbsSeq(1:10)
```

```
     0
     0
     0
     0
     0
     0
     1
     0
     0
     0
```

Input Arguments

`enb` — Cell-wide settings
structure

Cell-wide settings, specified as a structure. enb must contain the following field.

`NCellID` — Physical layer cell identity
scalar integer

Physical layer cell identity, specified as a scalar integer.

Data Types: `double`

Data Types: `struct`

`n` — Number of outputs
numeric scalar

Number of outputs, specified as a numeric scalar.

Data Types: `double`

`mapping` — Output sequence formatting
`'binary'` (default) | `'signed'`

Output sequence formatting, specified as a string. Valid formats are `'binary'` and `'signed'`. `'binary'` maps true to 1 and false to 0. `'signed'` maps true to –1 and false to 1.

Data Types: `char`

Output Arguments

`seq` — PBCH pseudorandom scrambling sequence

logical column vector | numeric column vector

PBCH pseudorandom scrambling sequence, specified as a logical column vector or a numeric column vector. seq contains the first n outputs of the physical broadcast channel (PBCH) scrambling sequence. If n is set to `'signed'`, seq is a vector of data type double. Otherwise, it is a vector of data type logical.

Data Types: `logical` | `double`

See Also

`ltePBCH` | `ltePBCHDecode` | `ltePBCHIndices`

ltePCFICH

Physical control format indicator channel

Syntax

```
sym = ltePCFICH(enb,cw)
```

Description

`sym = ltePCFICH(enb,cw)` returns the matrix of complex modulation symbols generated by the Physical Control Format Indicator Channel (PCFICH). The channel processing includes the stages of scrambling, QPSK modulation, layer mapping and precoding. Given input bit vector cw, each column of the 16-by-`CellRefP` matrix sym contains the 16 QPSK symbols carried by the PCFICH on each of `CellRefP` transmit antenna ports. The channel is parameterized by structure enb.

The PCFICH is intended to carry the 32-bit block encoding of the CFI. For more information, see `lteCFI`. The channel expects the input bit vector, cw, to be 32 elements in length. If `length(cw) < 32`, cw is padded with zeros prior to channel processing. If `length(cw) > 32`, only the first 32 elements are used.

Examples

Generate PCFICH Symbols

Generate PCFICH symbols, using a control format indicator (CFI) value of 1 and using 2 antenna ports for transmit diversity.

```
enb = struct('CellRefP',2,'NCellID',0,'NSubframe',0);
cfiCodeword = lteCFI(struct('CFI',1));
pcfichSymbols = ltePCFICH(enb,cfiCodeword);
size(pcfichSymbols)
```

```
16       2
```

Input Arguments

enb — Cell-wide settings structure
Scalar structure

`enb` is a structure having the following fields.

NCellID — Physical layer cell identity
0...503

Physical layer cell identity, specified as an integer in the range of [0,503].

CellRefP — Number of cell-specific reference signal (CRS) antenna ports
1 (default) | 2 | 4

Number of cell-specific reference signal (CRS) antenna ports, specified as one of the set (1, 2, 4).

NSubframe — Subframe number
scalar

Subframe number, specified as an integer.

Data Types: `struct`

cw — Input bit vector
vector

Input bit vector that is 32 elements in length, specified as a vector. If `length(cw) < 32`, `cw` is padded with zeros prior to channel processing. If `length(cw) > 32`, only the first 32 elements are used.

Example: `cw = lteCFI(struct('CFI',1));`

Data Types: `int8`

Output Arguments

sym — Complex modulation symbols generated by the PCFICH
numeric matrix

Complex modulation symbols generated by the PCFICH, returned as a numeric matrix of size 16-by-`CellRefP`. The channel processing includes the stages of scrambling, QPSK modulation, layer mapping and precoding. Given input bit vector `cw`, each column of the 16-by-`CellRefP` matrix `sym` contains the 16 QPSK symbols carried by the PCFICH on each of the `CellRefP` transmit antenna ports. The channel is parameterized by structure `enb`.

Data Types: `double`
Complex Number Support: Yes

See Also

`lteCFI` | `ltePCFICHDecode` | `ltePCFICHIndices` | `ltePCFICHInfo` | `ltePCFICHPRBS`

ltePCFICHDecode

Physical control format indicator channel decoding

Syntax

```
[bits,symbols] = ltePCFICHDecode(enb,sym)
[bits,symbols] = ltePCFICHDecode(enb,sym,hest,noiseest)
[bits,symbols] = ltePCFICHDecode(enb,sym,hest,noiseest,alg)
```

Description

`[bits,symbols] = ltePCFICHDecode(enb,sym)` performs the inverse of Physical Control Format Indicator Channel (PCFICH) processing on the matrix of complex modulated PCFICH symbols, sym, and cell-wide settings structure, enb. The channel inverse processing includes deprecoding, symbol demodulation, and descrambling. It returns a column vector of soft bits, bits, and received constellation of complex symbol vector, symbols, resulting from performing the inverse of PCFICH processing. See section 6.7 of [1] or `ltePCFICH` for details. bits is optionally scaled by channel state information (CSI) calculated during the equalization process.

sym must be a matrix of `NRE`-by-`NRxAnts` complex modulated PCFICH symbols. `NRE` is the number of QPSK symbols per antenna assigned to the PCFICH (16) and `NRxAnts` is the number of receive antennas.

`[bits,symbols] = ltePCFICHDecode(enb,sym,hest,noiseest)` performs the decoding of the complex PCFICH symbols, sym, using cell-wide settings, enb, the channel estimate, hest, and the noise estimate, noiseest. For the `'TxDiversity'` transmission scheme, when CellRefP is 2 or 4, the reception is performed using an orthogonal space frequency block code (OSFBC) decoder. For the `'Port0'` transmission scheme, when CellRefP is 1, the reception is performed using MMSE equalization.

hest is a 3-D `NRE`-by-`NRxAnts`-by-`CellRefP` array, where `NRE` are the frequency and time locations corresponding to the PCFICH RE positions, a total of `NRE` positions, `NRxAnts` is the number of receive antennas, and `CellRefP` is the number of cell-specific reference signal antennas, given by `enb.CellRefP`.

noiseest is an estimate of the noise power spectral density per RE on received subframe. This estimate is produced by the `lteDLChannelEstimate` function.

`[bits,symbols] = ltePCFICHDecode(enb,sym,hest,noiseest,alg)` is the same as above except it provides control over weighting the output soft bits, bits, with channel state information (CSI) calculated during the equalization stage using algorithmic configuration structure, alg.

Examples

Decode PCFICH Symbols

Perform encoding, modulation, demodulation and decoding of the CFI value given in cell-wide settings, `enb`.

```
enb = struct('NCellID',0,'NSubframe',0,'CellRefP',1);
enb.CFI = 3;
bits = ltePCFICHDecode(enb,ltePCFICH(enb,lteCFI(enb)));
rxVal = lteCFIDecode(bits)
```

```
           3
```

Input Arguments

`enb` — Cell-wide settings

scalar structure

Cell-wide settings, specified as a scalar structure. enb contains the following fields.

`NCellID` — Physical layer cell identity

0...503

Physical layer cell identity, specified as a nonnegative scalar integer in the range of 0 to 503.

Data Types: `double`

`CellRefP` — Number of cell-specific reference signal (CRS) antenna ports

1 | 2 | 4

Number of cell-specific reference signal (CRS) antenna ports, specified as either 1, 2, or 4.

Data Types: `double`

`NSubframe` — Subframe number

positive scalar integer

Subframe number, specified as a positive scalar integer greater than 0.

Data Types: `double`

Data Types: `struct`

`sym` — Complex modulated PCFICH symbols

numeric matrix

Complex modulated PCFICH symbols, specified as a numeric matrix of size `NRE`-by-`NRxAnts`. `NRE` is the number of QPSK symbols per antenna assigned to the PCFICH (16). `NRxAnts` is the number of receive antennas.

Data Types: `double`
Complex Number Support: Yes

`hest` — Channel estimate

3-D numeric array

Channel estimate, specified as a 3-D numeric array of size `NRE`-by-`NRxAnts`-by-`CellRefP`. `NRE` are the frequency and time locations corresponding to the PCFICH RE positions (a total of `NRE` positions), `NRxAnts` is the number of receive antennas, and `CellRefP` is the number of cell-specific reference signal antennas, given by `enb.CellRefP`.

Data Types: `double`
Complex Number Support: Yes

`noiseest` — Noise estimate

scalar

Estimate of the noise power spectral density per RE on received subframe. Such an estimate is provided by the `lteDLChannelEstimate` function.

Data Types: `double`

`alg` — Algorithmic configuration

structure

Algorithmic configuration, specified as a structure. It contains the following fields.

`CSI` — Channel state information weighting

`'On'` (default) | Optional | `'Off'`

Channel state information weighting, specified as a string having the value `'On'` or `'Off'`. Optional. This parameter determines if soft bits should be weighted by CSI.

Data Types: `char`

Data Types: `struct`

Output Arguments

`bits` — Soft bits

numeric column vector

Soft bits, returned as a numeric column vector. bits is optionally scaled by channel state information (CSI) calculated during the equalization process.

Data Types: `double`

`symbols` — Received constellation symbols

complex numeric column vector

Received constellation symbols, returned as a complex numeric column vector.

Data Types: `double`
Complex Number Support: Yes

References

[1] 3GPP TS 36.211. “Physical Channels and Modulation.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteCFIDecode` | `ltePCFICH` | `ltePCFICHIndices` | `ltePCFICHInfo` | `ltePCFICHPRBS`

ltePCFICHIndices

PCFICH resource element indices

Syntax

```
ind = ltePCFICHIndices(enb)
ind = ltePCFICHIndices(enb,opts)
```

Description

`ind = ltePCFICHIndices(enb)` returns the 16-by-`CellRefP` matrix of subframe resource element (RE) indices for the physical control format indicator channel (PCFICH), given the enb input structure. By default, the indices are returned in 1-based linear indexing form that directly indexes elements of a 3-D array representing the subframe resource grid for `CellRefP` antenna ports. Each column of ind contains per-antenna indices for 16 resource elements in one of the `CellRefP` array planes. The rows are ordered as the PCFICH modulation symbols should be mapped. The indices can also be returned in a number of alternative indexing formats.

The PCFICH is always transmitted on 16 resource elements, or 4 resource element groups (REG), in the first OFDM symbol of a subframe however their locations depend on the `NCellID` and `NDLRB` parameters.

`ind = ltePCFICHIndices(enb,opts)` allows control of the format of the returned indices through opts, a cell array of option strings.

Examples

Generate PCFICH RE Indices

This example generates physical CFI channel (PCFICH) resource element (RE) indices for two different physical layer cell identity values.

To show the effects of the physical layer cell identity, `NCellID` on the indices, first set it to 0. Then, generate and display the PCFICH indices.

```
enb.NDLRB = 50;
```

```
enb.NCellID = 0;
enb.CyclicPrefix = 'Normal';
enb.CellRefP = 1;
ind = ltePCFICHIndices(enb,{'0based','reg'})
```

```
ind =

           0
         150
         300
         450
```

```
Next, set the physical layer cell identity, |NCellID|, to 1. Regenerate
and display the PCFICH indices.
```

```
enb.NCellID = 1;
ind = ltePCFICHIndices(enb,{'0based','reg'})
```

```
ind =

           6
         156
         306
         456
```

Input Arguments

enb — eNodeB cell-wide settings

scalar structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
NDLRB	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)

Parameter Field	Required or Optional	Values	Description
`NCellID`	Required	Nonnegative scalar integer (0, …, 503)	Physical layer cell identity
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
`CellRefP`	Optional	1 (default), 2, 4	Number of cell-specific reference signal (CRS) antenna ports

`opts` — Index generation options

cell array

Index generation options, specified as a string or a cell array of strings that can contain the following values.

Option	Values	Description
Indexing style	`'ind'` (default), `'sub'`	Style for the returned indices, specified as one of the following options. • `'ind'` — returns the indices in linear index form as a column vector (default) • `'sub'` — returns the indices in `[subcarrier, symbol, antenna]` subscript row style. The number of rows in the output, ind, is the number of resource elements (NRE). Thus, ind is an NRE-by-3 matrix.
Index base	`'1based'` (default), `'0based'`	Base value of the returned indices. Specify `'1based'` to generate indices where the first value is one. Specify `'0based'` to generate indices where the first value is zero.
Indexing unit	`'re'` (default), `'reg'`	Unit of the returned indices. Specify `'re'` to indicate that the returned values correspond to individual resource elements (REs). Specify `'reg'` to indicate that the returned values correspond to resource element groups (REGs).

Data Types: `char` | `cell`

Output Arguments

`ind` — Subframe PCFICH RE indices

numeric matrix

Subframe PCFICH RE indices, returned as a numeric matrix of size 16-by-`CellRefP`. Each column of ind contains per-antenna indices for 16 resource elements in one of the `CellRefP` array planes. The rows are ordered as the PCFICH modulation symbols should be mapped.

See Also

`lteDLResourceGrid` | `ltePCFICH` | `ltePCFICHInfo` | `ltePDCCHIndices` | `ltePHICHIndices`

ltePCFICHInfo

PCFICH resource information

Syntax

```
info = ltePCFICHInfo
```

Description

`info = ltePCFICHInfo` returns a structure info containing the Physical Control Format Indicator Channel (PCFICH) subframe resources.

For the PCFICH, `NREG` = 4 and `NRE` = 16 = 4×`NREG`. These values are fixed for the system.

Examples

Get PCFICH Resource Information

Display information about the PCFICH subframe resources.

```
info = ltePCFICHInfo
```

```
    NREG: 4
     NRE: 16
```

Output Arguments

`info` — PCFICH resource information

scalar structure

PCFICH resource information, returned as a scalar structure. It can contain the following fields.

Parameter Field	Description	Values	Data Type
NRE	Number of resource elements (REs) assigned to PCFICH (4×`NREG`)	Nonnegative scalar integer	`uint64`
NREG	Number of resource element groups (REGs) assigned to PCFICH	Nonnegative scalar integer	`uint64`

See Also

`ltePCFICH` | `ltePCFICHDecode` | `ltePCFICHIndices` | `ltePCFICHPRBS`

ltePCFICHPRBS

PCFICH pseudorandom scrambling sequence

Syntax

```
seq = ltePCFICHPRBS(enb,n)
seq = ltePCFICHPRBS(enb,n,mapping)
```

Description

`seq = ltePCFICHPRBS(enb,n)` returns a vector containing the first n outputs of the physical control format indicator channel (PCFICH) scrambling sequence when initialized according to cell-wide settings structure, enb.

`seq = ltePCFICHPRBS(enb,n,mapping)` allows control over the format of the returned sequence, seq, through the mapping string.

Examples

Generate PCFICH Pseudorandom Scrambling Sequence

Generate a pseudorandom scrambling sequence for the PCFICH. Each resource element (RE) in the PCFICH is QPSK-modulated, resulting in two bits-per-symbol mapping on each resource element.

```
enb = lteRMCDL('R.14');
info = ltePCFICHInfo;
pcfichPrbsSeq = ltePCFICHPRBS(enb,info.NRE*2);
size(pcfichPrbsSeq)
pcfichPrbsSeq(1:10)
```

```
    32     1

     0
     1
     0
```

```
     0
     0
     0
     0
     1
     1
     0
```

The returned output is a 32-by-1 logical column vector containing zeros and ones.

Generate Signed PCFICH Pseudorandom Scrambling Sequence

Generate a signed pseudorandom scrambling sequence for the PCFICH. Each resource element (RE) in the PCFICH is QPSK-modulated, resulting in two bits-per-symbol mapping on each resource element.

```
enb = lteRMCDL('R.14');
info = ltePCFICHInfo;
pcfichPrbsSeq = ltePCFICHPRBS(enb,info.NRE*2,'signed');
size(pcfichPrbsSeq)
pcfichPrbsSeq(1:10)
```

```
    32     1

     1
    -1
     1
     1
     1
     1
     1
    -1
    -1
     1
```

The returned output is a 32-by-1 numeric column vector containing ones and negative ones.

Input Arguments

enb — Cell-wide settings

scalar structure

Cell-wide settings, specified as a scalar structure. enb contains the following fields.

`NCellID` — Physical layer cell identity

0...503

Physical layer cell identity, specified as a nonnegative scalar integer in the range of 0 to 503.

Data Types: `double`

`NSubframe` — Subframe number

positive scalar integer

Subframe number, specified as a positive scalar integer greater than 0.

Data Types: `double`

Data Types: `struct`

`n` — Length of scrambling sequence

positive scalar integer

Length of scrambling sequence, specified as a positive scalar integer greater than 0. This argument determines the number of elements in the output vector, seq.

Data Types: `double`

`mapping` — Controls the format of the returned sequence

`'binary'` (default) | `'signed'`

Controls the format of the returned sequence, specified as either binary or signed. Valid formats are (`'binary'` (default), `'signed'`). `'binary'` maps true to 1 and false to 0, and `'signed'` maps true to –1 and false to 1.

Data Types: `char`

Output Arguments

`seq` — PCFICH pseudorandom scrambling sequence

logical column vector | numeric column vector

PCFICH pseudorandom scrambling sequence, returned as a logical column vector or a numeric column vector. This argument contains the first n outputs of the PCFICH

scrambling sequence when initialized according to cell-wide settings structure, enb. If you set mapping to `'signed'`, the output is of data type double. Otherwise, it is of data type logical.

Data Types: `logical` | `double`

See Also

`ltePCFICH` | `ltePCFICHDecode` | `ltePCFICHIndices` | `ltePCFICHInfo` | `ltePRBS`

ltePDCCH

Physical downlink control channel

Syntax

```
[sym,info] = ltePDCCH(enb,cw)
[sym,info] = ltePDCCH(enb,cw,NREG)
[sym,info] = ltePDCCH(enb,cw,NREG,CCEGAINS)
```

Description

`[sym,info] = ltePDCCH(enb,cw)` returns an `NRE`-by-`CellRefP` complex matrix sym of modulation symbols given the input bit vector cw.

The function returns a matrix (sym) of complex modulation symbols generated by the set of Physical Downlink Control Channels (PDCCH) in a subframe. The channel processing includes the stages of scrambling, QPSK modulation, layer mapping and precoding, followed by REG interleaving and cyclic shifting. For a given input bit vector (typically the PDCCH multiplex), the output matrix sym will contain the QPSK symbols in column-wise antenna form. Any input bits with value < 0 are turned into `<NIL> ('0')` symbols. The optional structure info returns control resourcing information about the output symbols (see `ltePDCCHInfo` for details).

`[sym,info] = ltePDCCH(enb,cw,NREG)` returns matrix `sym`. With this syntax, the number of output QPSK symbols, `NRE`, is fixed by the NREG input value (`NRE`=4×NREG) instead of being calculated from the parameters of the enb structure. These sizes are reflected in the info structure.

`[sym,info] = ltePDCCH(enb,cw,NREG,CCEGAINS)` returns matrix `sym`. With this syntax, the CCEGAINS vector parameter allows control of the QPSK symbol gains on a per CCE basis.

Examples

Generate PDCCH Symbols

Generate complex modulated symbols for the PDCCH. The length of the input bit vector, `cw`, is the maximum number of input bits that can be transmitted on the PDCCH.

```
enb = lteRMCDL('R.0');
pdcchInfo = ltePDCCHInfo(enb);
cw = randi([0,1],pdcchInfo.MTot,1);
[pdcchSym,info] = ltePDCCH(enb,cw);
size(pdcchSym)
pdcchSym(1:10)
```

```
   452     1

   0.7071 - 0.7071i
  -0.7071 - 0.7071i
   0.7071 - 0.7071i
  -0.7071 + 0.7071i
   0.7071 - 0.7071i
   0.7071 - 0.7071i
   0.7071 + 0.7071i
   0.7071 - 0.7071i
  -0.7071 - 0.7071i
  -0.7071 + 0.7071i
```

The first output, `pdcchSym`, is a 452-by-1 column vector of complex modulated PDCCH symbols.

Input Arguments

`enb` — Cell-wide settings

scalar structure

Cell-wide settings, specified as a scalar structure. enb contains the following fields.

`CellRefP` — Number of cell-specific reference signal (CRS) antenna ports

1 | 2 | 4

Number of cell-specific reference signal (CRS) antenna ports, specified as 1, 2, or 4. Optional.

Data Types: `double`

`NCellID` — Physical layer cell identity

0...503

Physical layer cell identity, specified as an integer in the range of 0 to 503.

`NSubframe` — Subframe number

scalar

Subframe number, specified as an integer greater than 0.

`NDLRB` — Number of downlink resource blocks

6...110

Number of downlink resource blocks, specified as integer within the range 6 to 110.

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`CFI` — Control format indicator value

1 | 2 | 3

Control format indicator value, specified as 1, 2, or 3.

Data Types: `double`

`Ng` — PHICH group multiplier

Sixth | Half | One | Two

PHICH group multiplier, specified as one of `Sixth`, `Half`, `One`, or `Two`.

Data Types: `char`

`DuplexMode` — Duplexing mode

`'FDD'` (default) | Optional | `'TDD'`

Duplexing mode, specified as either `'FDD'` or `'TDD'`. `'FDD'` signifies frequency division duplexing. `'TDD'` signifies time division duplexing. Optional.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration

0 (default) | Optional | 0...6

Uplink or downlink configuration, specified as an integer between 0 and 6. Optional. Required for `'TDD'` duplex mode only.

Data Types: `struct`

`cw` — Input bit vector

vector

Input bit vector that is 32 elements in length, specified as a vector. If `length(cw) < 32`, `cw` is padded with zeros prior to channel processing. If `length(cw) > 32`, only the first 32 elements are used.

Example: `cw = lteCFI(struct('CFI',1));`

Data Types: `int8`

`NREG` — Resource element groups (REGs) assigned to PDCCH

scalar

Resource element groups (REGs) assigned to PDCCH.

`CCEGAINS` — Vector that controls the QPSK symbol gains on a per CCE basis

vector

Vector that controls the QPSK symbol gains on a per CCE basis. Each CCE (Control Channel Element) is a group of 36 QPSK symbols (72 bits) and is the minimum unit that a single coded DCI can be mapped to. The number of complete CCE, NCCE = floor(NREG/9), is available via the `NCCE` field in `info`. Each element of CCEGAINS acts as a linear multiplier to all 36 symbols generated from the associated block of 72 input bits. If CCEGAINS does not cover all the NREG symbols i.e. length(CCEGAINS) < `NCCE`, then the uncovered CCE will receive zero power. Note that all symbols are interleaved prior to output.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`sym` — PDCCH modulation symbols

Complex matrix

PDCCH modulation symbols, given the input bit vector `cw`, returned as a `NRE`-by-`CellRefP` complex matrix. `NRE` is the number of QPSK symbols per antenna and `CellRefP` is the number of TX antenna ports. `NRE` corresponds to the number of control region resource elements assigned to the PDCCH given the structure enb.

Data Types: `double`
Complex Number Support: Yes

`info` — Information for various PDCCH resourcing quantities

Structure

Information for various PDCCH resourcing quantities, returned as a structure. It contains fields including `NRE`, NREG and `MTot`.

`MTot` is the maximum number of input bits that can be transmitted on the `NRE` symbols (`MTot` = 2×`NRE` = 8×NREG). If `length(cw)<MTot`, the input is padded with (`MTot`-length(cw)) `<NIL>` elements which translate to 0 valued symbols. Any elements of input vector cw valued < 0 are also treated as `<NIL>` elements. If length(cw) > `MTot` then only the first `MTot` bits are used.

Data Types: `struct`

See Also

`lteDCIEncode` | `ltePDCCHDecode` | `ltePDCCHIndices` | `ltePDCCHInfo` | `ltePDCCHInterleave` | `ltePDCCHPRBS` | `ltePDCCHSearch` | `ltePDCCHSpace`

ltePDCCHDecode

Physical downlink control channel decoding

Syntax

```
[bits,symbols] = ltePDCCHDecode(enb,sym)
[bits,symbols] = ltePDCCHDecode(enb,sym,hest,noiseest)
[bits,symbols] = ltePDCCHDecode(enb,sym,hest,noiseest,alg)
```

Description

`[bits,symbols] = ltePDCCHDecode(enb,sym)` performs the inverse of Physical Downlink Control Channel (PDCCH) processing on the matrix of complex modulated PDCCH symbols, sym, and cell-wide settings structure, enb. The channel inverse processing includes resource element group deinterleaving and cyclic shifting, deprecoding, symbol demodulation, and descrambling.

The function returns a column vector of soft bits, bits, and received constellation of complex symbol vector, symbols, resulting from performing the inverse of PDCCH processing. See [1] or `ltePDCCH` for details.

`[bits,symbols] = ltePDCCHDecode(enb,sym,hest,noiseest)` performs the decoding of the complex PDCCH symbols, sym, using cell-wide settings, enb, the channel estimate, hest, and the noise estimate, noiseest. For the `TxDiversity` transmission scheme, when `CellRefP` is 2 or 4, the reception is performed using an orthogonal space frequency block code (OSFBC) decoder. For the `Port0` transmission scheme, when `CellRefP` is 1, the reception is performed using minimum mean square error (MMSE) equalization.

`[bits,symbols] = ltePDCCHDecode(enb,sym,hest,noiseest,alg)` provides control over weighting the output soft bits, bits, with channel state information (CSI) calculated during the equalization stage using algorithmic configuration structure, alg. When `alg.CSI` is `'On'`, bits is scaled by channel state information calculated during the equalization process.

Examples

Decode PDCCH Symbols

Generate and decode the complex PDCCH modulated symbols for RMC R.0 from cell-wide settings structure, `enb`.

```
enb = lteRMCDL('R.0');
pdcchInfo = ltePDCCHInfo(enb);
codewordBits = randi([0,1],pdcchInfo.MTot,1);
pdcchSym = ltePDCCH(enb,codewordBits);
[softBits,symbols] = ltePDCCHDecode(enb,pdcchSym);
```

Input Arguments

`enb` — Cell-wide settings
scalar structure

Cell-wide settings, specified as a scalar structure. enb contains the following fields.

`CellRefP` — Number of cell-specific reference signal (CRS) antenna ports
1 | 2 | 4

Number of cell-specific reference signal (CRS) antenna ports, specified as either 1, 2, or 4.

Data Types: `double`

`NCellID` — Physical layer cell identity
nonnegative scalar integer [0...503]

Physical layer cell identity, specified as a nonnegative scalar integer in the range of 0 to 503.

Data Types: `double`

`NSubframe` — Subframe number
scalar integer

Subframe number, specified as a scalar integer.

Data Types: `double`

Data Types: `struct`

sym — PDCCH modulation symbols

complex numeric matrix

PDCCH modulation symbols, specified as a complex numeric matrix of size NRE-by-NRxAnts. NRE is the number of QPSK symbols per antenna assigned to the PDCCH (i.e. the number of control region resource elements) and NRxAnts is the number of receive antennas.

Data Types: double
Complex Number Support: Yes

hest — Channel estimate

3-D numeric array

Channel estimate, specified as a 3-D numeric array of size NRE-by-NRxAnts-by-CellRefP. NRE are the frequency and time locations corresponding to the PDCCH RE positions (a total of NRE positions), NRxAnts is the number of receive antennas, and CellRefP is the number of cell-specific reference signal antennas, given by enb.CellRefP.

Data Types: double
Complex Number Support: Yes

noiseest — Noise estimate

numeric scalar

Noise estimate, specified as a numeric scalar. This input argument is an estimate of the noise power spectral density per RE on received subframe. This estimate is produced by the lteDLChannelEstimate function.

Data Types: double

alg — Algorithmic configuration to calculate CSI for weighting soft bits

structure

Algorithmic configuration to calculate CSI for weighting soft bits, specified as a structure having the following fields.

CSI — Channel state information

'On' (default) | Optional | 'Off'

Channel state information, specified as a string having the value 'On' or 'Off'. Optional. This parameter determines if soft bits should be weighted by CSI.

Data Types: `char`

Data Types: `struct`

Output Arguments

`bits` — Soft bits

numeric column vector

Soft bits, returned as a numeric column vector. bits is the received PDCCH payload containing coded downlink control information (DCI) messages. It is optionally scaled by channel state information (CSI) calculated during the equalization process.

Data Types: `double`

`symbols` — Received constellation symbols

complex numeric column vector

Received constellation symbols, returned as a complex numeric column vector.

Data Types: `double`
Complex Number Support: Yes

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDCIDecode` | `ltePDCCH` | `ltePDCCHDeinterleave` | `ltePDCCHIndices` | `ltePDCCHInfo` | `ltePDCCHPRBS` | `ltePDCCHSearch` | `ltePDCCHSpace`

ltePDCCHDeinterleave

PDCCH deinterleaving and cyclic shifting

Syntax

```
out = ltePDCCHDeinterleave(enb,in)
```

Description

`out = ltePDCCHDeinterleave(enb,in)` performs the PDCCH Resource Element Groups (REGs) deinterleaving and cyclic shifting to undo the processing described in section 6.8.5 of [1]. This function takes PDCCH complex modulated symbols in an *M*-by-*P* matrix in. *M* is the number of modulated symbols and *P* is the number of transmit antennas. The *M* modulated symbols specified in input matrix in must be a concatenation of symbol quadruplets. If the input in is a vector, it deinterleaves the elements of the vector. If in is a matrix, it deinterleaves the rows.

The cyclic shifting process is the reverse of the `NCellID` dependent cyclic shift carried out during PDCCH coding to avoid intercell interference. The de-interleaving is performed to reverse the permutation operation described in section 5.1.4.2.1 of [2] while making the exception that “bits” are replaced by “symbol quadruplets.”

Examples

Deinterleave PDCCH Symbols

Perform PDCCH resource element group (REG) deinterleaving.

First, interleave the PDCCH symbol bits, `cw`. Then, deinterleave the output and compare it with the input bit vector, `cw`.

```
enb = lteRMCDL('R.0');
pdcchInfo = ltePDCCHInfo(enb);
cw = randi([0,1],pdcchInfo.MTot,1);
interleavedBits = ltePDCCHInterleave(enb,cw);
```

```
deinterleavedBits  = ltePDCCHDeinterleave(enb,interleavedBits);
isequal(cw,deinterleavedBits)
```

```
     1
```

Input Arguments

`enb` — Cell-wide settings

scalar structure

Cell-wide settings, specified as a scalar structure. enb can contain the following fields.

`NCellID` — Physical layer cell identity

integer (0...503)

Physical layer cell identity, specified as an integer in the range of 0 to 503.

Data Types: `struct`

`in` — PDCCH complex modulated input symbols

numeric matrix

PDCCH complex modulated input symbols, specified as a numeric matrix of size *M*-by-*P*, where *M* is the number of modulated symbols and *P* is the number of transmit antennas. The *M* modulated symbols specified in input matrix in must be a concatenation of symbol quadruplets. If the input in is a vector, it deinterleaves the elements of the vector. If in is a matrix, it deinterleaves the rows.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`out` — Deinterleaved output

numeric column vector

Deinterleaved output, returned as a numeric column vector.

Data Types: `double`
Complex Number Support: Yes

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

[2] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`ltePDCCH` | `ltePDCCHDecode` | `ltePDCCHIndices` | `ltePDCCHInfo` | `ltePDCCHInterleave` | `ltePDCCHPRBS` | `ltePDCCHSearch` | `ltePDCCHSpace`

ltePDCCHIndices

PDCCH resource element indices

Syntax

```
ind = ltePDCCHIndices(enb)
ind = ltePDCCHIndices(enb,opts)
ind = ltePDCCHIndices(enb,exreg,opts)
```

Description

`ind = ltePDCCHIndices(enb)` returns a `NRE`-by-`CellRefP` matrix of 1-based linear indexing RE indices given the structure enb. It returns the subframe resource element (RE) indices for the physical downlink control channels (PDCCH). By default, ind is a `NRE`-by-`CellRefP` matrix of indices in a 1-based linear indexing style which can directly index elements of a *N*-by-*M*-by-`CellRefP` array representing the subframe grid across `CellRefP` antenna ports. Each column of ind identifies the same set of `NRE` subframe resource elements but with indices offset to select them in a different antenna "page" of the 3-D resource array.

The `NRE` indices returned cover all PDCCH resources in the control region not already assigned to PCFICH or PHICH (see `ltePDCCHInfo`). They are ordered as the complete block of padded, interleaved and shifted PDCCH modulation symbols that should then be mapped, as described in section 6.8.5 of [1].

`ind = ltePDCCHIndices(enb,opts)` allows control of the format of the returned indices through a cell array opts.

`ind = ltePDCCHIndices(enb,exreg,opts)` will return a matrix of indices as above where the resources not to be assigned to PDCCH are defined explicitly by vector exreg rather than taken to be the PCFICH and PHICH configured by enb. The exreg must contain valid resource element group (REG) indices but can be either 0 or 1-based throughout, and indices which do not fall within the control region are ignored.

Examples

Get PDCCH Resource Element Indices

Retrieve PDCCH resource element (RE) indices.

Create an RMC R.0 configuration structure and find its PDCCH RE indices. Display the size of the indices.

```
enb = lteRMCDL('R.0');
ind = ltePDCCHIndices(enb);
size(ind)
```

```
   452     1
```

Get PDCCH Indices and Exclude Resources

Explicitly exclude resources when retrieving PDCCH indices.

First, do not exclude any resources. Call the `ltePDCCHIndices` function and provide an empty matrix for the argument exreg.

```
enb = lteRMCDL('R.0');
ind = ltePDCCHIndices(enb,[],'re');
size(ind)
```

```
   480     1
```

All RE indices are returned in the required mapping order.

Next, explicitly exclude the PCFICH and PHICH indices.

```
enb = lteRMCDL('R.0');
exreg = [ltePCFICHIndices(enb,'reg'); ltePHICHIndices(enb,'reg')];
ind = ltePDCCHIndices(enb,exreg,'re');
size(ind)
```

```
   452     1
```

This call returns the same result as the default syntax call, `ltePDCCHIndices(enb)`.

Input Arguments

`enb` — Cell-wide settings

structure

enb is a structure having the following fields.

`NDLRB` — Number of downlink resource blocks
6...110

Number of downlink resource blocks, specified as an integer between 6 and 110.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`CellRefP` — Number of cell-specific reference signal (CRS) antenna ports
1 | 2 | 4

Number of cell-specific reference signal (CRS) antenna ports, specified as 1, 2, or 4.

Data Types: `double`

`CFI` — Control format indicator value
1 | 2 | 3

Control format indicator value, specified as 1, 2, or 3.

Data Types: `double`

`Ng` — HICH group multiplier
Sixth | Half | One | Two

HICH group multiplier, specified as one of `Sixth`, `Half`, `One`, or `Two`.

Data Types: `char`

`PHICHDuration` — PHICH duration
`'Normal'` (default) | Optional | `'Extended'`

PHICH duration, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`DuplexMode` — Duplexing mode
`'FDD'` (default) | Optional | `'TDD'`

Duplexing mode, specified as a string. Possible values are either frequency division duplex, `'FDD'`, or time division duplex, `'TDD'`. Optional.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration

0 (default) | Optional | 0...6

Uplink or downlink configuration, specified as a nonnegative scalar integer between 0 and 6. Optional. Required for `'TDD'` duplex mode only.

`NSubframe` — Subframe number

positive scalar integer

Subframe number, specified as a positive scalar integer greater than 0.

Data Types: `double`

Data Types: `struct`

`opts` — Option strings for symbol formatting

string | cell array of strings

Option strings for symbol formatting, specified as a string or a cell array of strings. opts can contain the following option strings.

`Indexing unit` — Format of the returned indices

`'re'` (default) | `'reg'`

Format of the returned indices, specified as a string. This parameter indicates that the returned values correspond to either individual resource elements (REs) or resource element groups (REGs).

Data Types: `char`

`Indexing style` — Style of the returned indices

`'ind'` (default) | `'sub'`

Style of the returned indices, specified as a string. This parameter indicates the style of returned indices as either linear index form or [*subcarrier*, *symbol*, *antenna*] subscript row form.

Data Types: `char`

`Index base` — Index base of the returned indices

`'1based'` (default) | `'0based'`

Index base of the returned indices, specified as a string. This parameter indicates the index base of returned indices as either 1-based or 0-based.

Data Types: `char`

Data Types: `char` | `cell`

`exreg` — Resources excluded from PDCCH

vector

Resources excluded from PDCCH, specified as a vector. This vector explicitly defines those resources not to be assigned to PDCCH. exreg must contain valid resource element group (REG) indices but can be either 0- or 1-based throughout. Indices which do not fall within the control region are ignored.

Data Types: `double`

Output Arguments

`ind` — PDCCH RE indices

numeric matrix

PDCCH RE indices, returned as a numeric matrix of size `NRE`-by-`CellRefP`. The matrix contains 1-based linear indexing RE indices. Each column of ind identifies the same set of `NRE` subframe resource elements but with indices offset to select them in a different antenna "page" of the 3-D resource array.

Data Types: `double`

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePDCCH` | `ltePDCCHDecode` | `ltePDCCHDeinterleave` | `ltePDCCHInfo` | `ltePDCCHInterleave` | `ltePDCCHSearch` | `ltePDCCHSpace`

ltePDCCHInfo

PDCCH resource information

Syntax

```
info = ltePDCCHInfo(enb)
```

Description

`info = ltePDCCHInfo(enb)` returns a structure info containing information about the Physical Downlink Control Channel (PDCCH) subframe resources.

Within a non-MBMS downlink subframe, the first `info.NSymbols` OFDM symbols represent its *control region* and carry the PCFICH, PHICH and PDCCH. The non-reference resource elements (RE) not assigned to the PCFICH or PHICH are associated with PDCCH transmission and their number is given by `info.NRE`. These resources carry the set of PDCCH where each PDCCH carries a single encoded DCI message. Each PDCCH can be transmitted on 1,2,4 or 8 Control Channel Elements (CCE) where 1 CCE = 9 REG = 36 RE = 72 bits. As such, not all of the `NRE` elements can carry actual PDCCH instances, with (`info.NRE` — `info.NREUsed`) being unavailable for PDCCH transmission.

Examples

Get PDCCH Information

Get information about the PDCCH subframe resources for RMC R.0.

```
enb = lteRMCDL('R.0');
info = ltePDCCHInfo(enb)
```

```
        NREG: 113
         NRE: 452
        NCCE: 12
    NREGUsed: 108
```

```
     NREUsed: 432
        MTot: 904
    NSymbols: 3
```

Input Arguments

`enb` — Cell-wide settings structure

scalar structure

Cell-wide settings, specified as a scalar structure. enb is a structure having the following fields.

`NDLRB` — Number of downlink resource blocks

6...110

Number of downlink resource blocks, specified as integer within the range 6 to 110.

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char` |

`CellRefP` — Number of cell-specific reference signal antenna ports

1 | 2 | 4

Number of cell-specific reference signal antenna ports, specified as 1, 2, or 4.

Data Types: `double`

`CFI` — Control format indicator value

1 | 2 | 3

Control format indicator value, specified as 1, 2, or 3.

Data Types: `double`

`Ng` — PHICH group multiplier

Sixth | Half | One | Two

PHICH group multiplier, specified as one of `Sixth`, `Half`, `One`, or `Two`.

Data Types: `char`

`DuplexMode` — Duplexing mode

`'FDD'` (default) | Optional | `'TDD'`

Duplexing mode, specified as either `'FDD'` or `'TDD'`. Optional. Options signify frequency division duplex (FDD) or time division duplex (TDD) mode.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration

0 (default) | Optional | 0...6

Uplink or downlink configuration, specified as an integer in the range 0 to 6. Optional. Required for `'TDD'` duplex mode only.

`NSubframe` — Subframe number

integer

Subframe number, specified as an integer.

Data Types: `struct`

Output Arguments

`info` — PDCCH subframe resource information

structure

PDCCH subframe resource information, returned as a structure. info contains the following fields.

`NRE` — Total resource elements associated with PDCCHs

numeric scalar

Total resource elements associated with PDCCHs, returned as a numeric scalar (4×`NREG`).

Data Types: `double`

`NREG` — Total resource element groups associated with PDCCHs

numeric scalar

Total resource element groups associated with PDCCHs, returned as a numeric scalar (4×NRE).

Data Types: double

MTot — Total number of bits associated with PDCCHs

numeric scalar

Total number of bits associated with PDCCHs, returned as a numeric scalar (8×NREG). MTot is the maximum number of input bits that can be transmitted on the NRE symbols (MTot = 2×NRE = 8×NREG).

Data Types: double

NCCE — Number of control channel elements available for actual PDCCH usage

numeric scalar

Number of control channel elements available for actual PDCCH usage, returned as a numeric scalar.

Data Types: double

NREGUsed — Number of Resource Element Groups available for actual PDCCH usage

numeric scalar

Number of Resource Element Groups available for actual PDCCH usage, returned as a numeric scalar.

Data Types: double

NREUsed — Number of Resource Elements available for actual PDCCH usage

numeric scalar

Number of Resource Elements available for actual PDCCH usage, returned as a numeric scalar.

Data Types: double

NSymbols — Number of OFDM symbols spanned by the PDCCH

numeric scalar

Number of OFDM symbols spanned by the PDCCH, returned as a numeric scalar.

Data Types: double

Data Types: `struct`

See Also

`ltePDCCH` | `ltePDCCHDecode` | `ltePDCCHDeinterleave` | `ltePDCCHIndices` | `ltePDCCHInterleave` | `ltePDCCHPRBS` | `ltePDCCHSearch` | `ltePDCCHSpace`

ltePDCCHInterleave

PDCCH interleaving and cyclic shift

Syntax

```
out = ltePDCCHInterleave(enb,in)
```

Description

`out = ltePDCCHInterleave(enb,in)` performs the interleaving and cyclic shifting on PDCCH resource element groups (REGs) as described in section 6.8.5 of [1]. This function takes PDCCH complex modulated symbols in an *M*-by-*P* matrix, in. *M* is the number of modulated symbols and *P* is the number of transmit antennas. The *M* modulated symbols, specified in input matrix in, must be a concatenation of symbol quadruplets. If the input, in, is a vector, it interleaves the elements of the vector. Otherwise, if the input, in, is a matrix, it interleaves the rows.

The permutation, or interleaving, operation is performed as described in section 5.1.4.2.1 of [2], with the exception that "bits" are replaced by "symbol quadruplets." Then, the block of PDCCH-modulated symbol quadruplets are cyclically shifted with `NCellID` to avoid intercell interference.

Examples

Perform PDCCH Interleaving

Interleave a sequential input.

```
enb = lteRMCDL('R.0');
info = ltePDCCHInfo(enb);
out = ltePDCCHInterleave(enb,(1:info.MTot).');
out(1:10)
```

```
    13
    14
    15
```

```
  16
 141
 142
 143
 144
 269
 270
```

The resulting output is a vector concatenated input quadruplets.

Input Arguments

`enb` — Cell-wide settings

scalar structure

Cell-wide settings, specified as a scalar structure. enb contains the following fields.

`NCellID` — Physical layer cell identity

0...503

Physical layer cell identity, specified as an integer in the range of [0,503].

Data Types: `struct`

`in` — PDCCH complex modulated input symbols

complex-valued numeric matrix | numeric vector

PDCCH complex modulated input symbols, specified in a complex-valued numeric matrix or a numeric vector. As a matrix, its size is *M*-by-*P*, where *M* is the number of modulated symbols and *P* is the number of transmit antennas. The *M* modulated symbols specified in input matrix, in, must be a concatenation of symbol quadruplets. If the input, in, is a vector, it interleaves the elements of the vector. If in is a matrix, it interleaves the rows.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`out` — Interleaved output

Numeric vector

Interleaved output, returned as a numeric vector.

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePDCCH` | `ltePDCCHDecode` | `ltePDCCHDeinterleave` | `ltePDCCHIndices` | `ltePDCCHInfo` | `ltePDCCHPRBS` | `ltePDCCHSearch` | `ltePDCCHSpace`

ltePDCCHPRBS

PDCCH pseudorandom scrambling sequence

Syntax

```
seq = ltePDCCHPRBS(enb,n)
seq = ltePDCCHPRBS(enb,n,mapping)
```

Description

`seq = ltePDCCHPRBS(enb,n)` returns a column vector containing the first n outputs of the Physical Downlink Control Channel (PDCCH) scrambling sequence when initialized according to cell-wide settings structure, enb.

`seq = ltePDCCHPRBS(enb,n,mapping)` allows control over the format of the returned sequence, seq, through the mapping string . Valid formats are `'binary'`, which is the default, and `'signed'`. `'binary'` maps `true` to 1 and `false` to 0. `'signed'` maps `true` to –1 and `false` to 1.

Examples

Return Signed PDCCH Scrambling Sequence

```
enb = lteRMCDL('R.0');
seq = ltePDCCHPRBS(enb,7,'signed')
```

```
     1
     1
     1
     1
     1
     1
    -1
```

Return Binary PDCCH Scrambling Sequence

```
enb = lteRMCDL('R.0');
```

```
seq = ltePDCCHPRBS(enb,7)

     0
     0
     0
     0
     0
     0
     1
```

Input Arguments

`enb` — Cell-wide settings
structure

Cell-wide settings, specified as a structure. This argument contains the following fields.

`NCellID` — Physical layer cell identity
0...503

Physical layer cell identity, specified as a nonnegative scalar integer in the range of 0 to 503.

Data Types: `double`

`NSubframe` — Subframe number
positive scalar integer

Subframe number, specified as a positive scalar integer greater than 0.

Data Types: `double`

Data Types: `struct`

`n` — Number of elements in returned sequence
numeric scalar

Number of elements in returned sequence, seq, specified as a numeric scalar.

Data Types: `double`

`mapping` — Sequence format
`'binary'` (default) | `'signed'`

String that controls the format of the returned sequence seq. The string `'binary'` maps `true` to 1 and `false` to 0. The string `'signed'` maps `true` to –1 and `false` to 1.

Data Types: `char`

Output Arguments

`seq` — PDCCH pseudorandom scrambling sequence

logical column vector | numeric column vector

PDCCH pseudorandom scrambling sequence, returned as a logical column vector or a numeric column vector. This argument contains the first n outputs of the PDCCH scrambling sequence when initialized according to cell-wide settings structure, enb. If you set mapping to `'signed'`, the output data type is double. Otherwise, the output data type is logical.

Data Types: `logical` | `double`

See Also

`ltePDCCH` | `ltePDCCHDecode` | `ltePDCCHDeinterleave` | `ltePDCCHIndices` | `ltePDCCHInfo` | `ltePDCCHInterleave` | `ltePDCCHSearch` | `ltePDCCHSpace`

ltePDCCHSearch

PDCCH downlink control information search

Syntax

```
[dcistr,dcibits] = ltePDCCHSearch(enb,chs,softbits)
```

Description

`[dcistr,dcibits] = ltePDCCHSearch(enb,chs,softbits)` recovers downlink control information (DCI) message structures, dcistr, and corresponding vectors of DCI message bits, dcibits, after blind decoding the multiplexed physical downlink control channels (PDCCHs) within the control region given by input vector of soft bits, softbits, cell-wide settings enb and user equipment (UE)-related configuration structure, chs.

The UE is required to monitor multiple PDCCHs within the control region. The UE is only informed of the width, in OFDM symbols, of the control region within a subframe, and is not aware of the exact location of PDCCHs relevant to it. The UE finds the PDCCHs relevant to it by monitoring a set of PDCCH candidates, a set of consecutive control candidate elements (CCEs) on which PDCCH could be mapped, in every subframe. For details, see `ltePDCCHSpace`. This process is referred to as *blind decoding*.

To simplify the decoding task at the UE, the whole control region is subdivided into common and UE-specific search spaces which the UE monitors (monitor implies attempting to decode each PDCCH). Each search space comprises 2, 4 or 6 PDCCH candidates whose data length depends on its corresponding PDCCH format; each PDCCH must be transmitted on 1, 2, 4 or 8 CCE (1 CCE = 72 bits). The common search space is limited to only two aggregation levels, 4 and 8, while the UE-specific search space can have an aggregation level of 1, 2, 4, or 8.

The common search space carries control information common to all UEs and is therefore monitored by all UEs within a cell. The common control information carries initial important information including paging information, system information and random access procedures. The UE monitors the common search space by de-masking each PDCCH candidate with different RNTIs e.g. P-RNTI, SI-RNTI, RA-RNTI etc.

In the UE-specific search space the UE finds the PDCCH relevant to it by monitoring a set of PDCCH candidates in every subframe. If no CRC error is detected when the UE de-

masks a PDCCH candidate with its RNTI (16-bit C-RNTI value), the UE determines that the PDCCH candidate carries its own control information.

The number and location of candidates within a search space is different for each PDCCH format. There are four PDCCH formats (i.e., 0, 1, 2 or 3). If the UE fails to decode any PDCCH candidates for a given PDCCH format, it tries to decode candidates for another PDCCH format.

This function returns the DCI messages in two forms:

- dcistr is a cell array of structures where each structure represents a successfully decoded DCI whose fields match those of the associated DCI format. Each structure contains the fields associated with one or more decoded DCI messages. As multiple PDCCHs can be transmitted in a subframe, the UE has to monitor all possible PDCCHs directed at it. If more than one PDCCHs is directed to the UE or successfully decoded then DCISTR will contain that number of decoded DCI messages.
- dcibits is a cell array containing one or more vectors where each vector contains the bit stream of a recovered DCI message including any zero-padding. Each vector of bit values corresponds to successfully decoded DCI messages. For details, see `lteDCI`.

Examples

Get DCI Message Structure and Bits

Extract the PDCCH symbols from the transmit resource grid and decode them.

```
rmc = lteRMCDL('R.0');
[~,txGrid] = lteRMCDLTool(rmc,[1;0;0;1]);
pdcchSymbols = txGrid(ltePDCCHIndices(rmc));
rxPdcchBits = ltePDCCHDecode(rmc,pdcchSymbols);
```

The UE monitors the common search space by demasking the PDCCH candidate with RNTIs. Perform blind decoding of the multiplexed PDCCH within the control region. Recover the DCI cell array, `rxDCI`, and the corresponding vector of DCI message bits, `rxDCIBits`.

```
ueConfig.RNTI = rmc.PDSCH.RNTI;
[rxDCI,rxDCIBits] = ltePDCCHSearch(rmc,ueConfig,rxPdcchBits);
```

Display the first successfully decoded DCI, `decDCI`, a structure whose fields match those of the associated DCI format. Also, get the bit stream of the first recovered DCI message, including any zero-padding.

```
decDCI = rxDCI{1}
decDCIBits = rxDCIBits{1};
```

```
         DCIFormat: 'Format1'
               CIF: 0
    AllocationType: 1
        Allocation: [1x1 struct]
         ModCoding: 14
            HARQNo: 0
           NewData: 1
                RV: 0
          TPCPUCCH: 0
          TDDIndex: 0
```

Input Arguments

`enb` — Cell-wide settings

structure

Cell-wide settings, specified as a structure with these fields.

`NDLRB` — Number of downlink resource blocks

6 | 15 | 25 | 50 | 75 | 100

Number of downlink resource blocks, specified as a numeric scalar value.

Data Types: `double`

`NSubframe` — Subframe number

0 (default) | nonnegative scalar integer

Subframe number, specified as a nonnegative scalar integer.

Data Types: `double`

`CellRefP` — Number of cell-specific reference signal antenna ports

1 | 2 | 4

Number of cell-specific reference signal antenna ports, specified as a numeric scalar value.

Data Types: `double`

`DuplexMode` — Duplexing mode

`'FDD'` (default) | `'TDD'`

Duplexing mode, specified as a string.

Data Types: `double`

Data Types: `struct`

`chs` — User Equipment (UE) related configuration

structure

User Equipment (UE) related configuration, specified as a structure containing the following field.

`RNTI` — Radio network temporary identifier

1 (default) | numeric scalar

Radio network temporary identifier value, specified as a numeric scalar.

Data Types: `double`

Data Types: `struct`

`softbits` — Input vector of soft bits

numeric column vector

Input vector of soft bits, specified as a column vector of double values.

Data Types: `double`

Output Arguments

`dcistr` — DCI message structures

cell array of structures

DCI message structures, returned as a cell array of structures where each structure represents a successfully decoded DCI whose fields match those of the associated DCI format. Each structure contains the fields associated with one or more decoded DCI messages. Since multiple PDCCHs can be transmitted in a subframe, the UE must monitor all possible PDCCHs directed at it. If more than one PDCCH is directed to UE or successfully decoded, `DCISTR` contains that number of decoded DCI messages.

Each cell contains a structure with these fields.

`DCIFormat` — Downlink control information (DCI) format type
`'Format0' | 'Format1' | 'Format1A' | 'Format1B' | 'Format1C' | 'Format1D' | 'Format2' | 'Format2A' | 'Format2B' | 'Format2C' | 'Format3' | 'Format3A' | 'Format4'`

Downlink control information (DCI) format type, specified as a string. The following table presents the fields associated with each DCI format.

DCI Formats	DCISTR Fields	Size	Description
`'Format0'`	`DCIFormat`	-	`'Format0'`
	`FreqHopping`	1-bit	PUSCH frequency hopping flag
	`Allocation`	variable	Resource block assignment/ allocation
	`ModCoding`	5-bits	Modulation, coding scheme and redundancy version
	`NewData`	1-bit	New data indicator
	`TPC`	2-bits	PUSCH TPC command
	`CShiftDMRS`	3-bits	Cyclic shift for DM RS
	`CQIReq`	1-bit	CQI request
	`TDDIndex`	2-bits	For TDD config 0, this field is the Uplink Index. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format1'`	`DCIFormat`	-	`'Format1'`
	`AllocationType`	1-bit	Resource allocation header: type 0, type 1 (only if downlink bandwidth is >10 PRBs)
	`Allocation`	variable	Resource block assignment/ allocation
	`ModCoding`	5-bits	Modulation and coding scheme

DCI Formats	DCISTR Fields	Size	Description
	HARQNo	3-bits (FDD) 4-bits (TDD)	HARQ process number
	NewData	1-bit	New data indicator
	RV	2-bits	Redundancy version
	TPCPUCCH	2-bits	PUCCH TPC command
	TDDIndex	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
'Format1A'	DCIFormat	-	'Format1A'
	AllocationType	1-bit	VRB assignment flag: 0 (localized), 1 (distributed)
	Allocation	variable	Resource block assignment/ allocation
	ModCoding	5-bits	Modulation and coding scheme
	HARQNo	3-bits (FDD) 4-bits (TDD)	HARQ process number
	NewData	1-bit	New data indicator
	RV	2-bits	Redundancy version
	TPCPUCCH	2-bits	PUCCH TPC command
	TDDIndex	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
'Format1B'	DCIFormat	-	'Format1B'

DCI Formats	DCISTR Fields	Size	Description
	`AllocationType`	1-bit	VRB assignment flag: 0 (localized), 1 (distributed)
	`Allocation`	variable	Resource block assignment/ allocation
	`ModCoding`	5-bits	Modulation and coding scheme
	`HARQNo`	3-bits (FDD) 4-bits (TDD)	HARQ process number
	`NewData`	1-bit	New data indicator
	`RV`	2-bits	Redundancy version
	`TPCPUCCH`	2-bits	PUCCH TPC command
	`TPMI`	2-bits (2 antennas) 4-bits (4 antennas)	PMI information
	`PMI`	1-bit	PMI confirmation
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format1C'`	`DCIFormat`	-	`'Format1C'`
	`Allocation`	variable	Resource block assignment/ allocation
	`ModCoding`	5-bits	Modulation and coding scheme
`'Format1D'`	`DCIFormat`	-	`'Format1D'`
	`AllocationType`	1-bit	VRB assignment flag: 0 (localized), 1 (distributed)
	`Allocation`	variable	Resource block assignment/ allocation

DCI Formats	DCISTR Fields	Size	Description
	`ModCoding`	5-bits	Modulation and coding scheme
	`HARQNo`	3-bits (FDD) 4-bits (TDD)	HARQ process number
	`NewData`	1-bit	New data indicator
	`RV`	2-bits	Redundancy version
	`TPCPUCCH`	2-bits	PUCCH TPC command
	`TPMI`	2-bits (2 antennas) 4-bits (4 antennas)	Precoding TPMI information
	`DlPowerOffset`	1-bit	Downlink power offset
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format2'`	`DCIFormat`	-	`'Format2'`
	`AllocationType`	1-bit	Resource allocation header: type 0, type 1 (only if downlink bandwidth is >10 PRBs)
	`Allocation`	variable	Resource block assignment/ allocation
	`TPCPUCCH`	2-bits	PUCCH TPC command
	`HARQNo`	3-bits (FDD) 4-bits (TDD)	HARQ process number
	`SwapFlag`	1-bit	Transport block to codeword swap flag

DCI Formats	DCISTR Fields	Size	Description
	`ModCoding1`	5-bits	Modulation and coding scheme for transport block 1
	`NewData1`	1-bit	New data indicator for transport block 1
	`RV1`	2-bits	Redundancy version for transport block 1
	`ModCoding2`	5-bits	Modulation and coding scheme for transport block 2
	`NewData2`	1-bit	New data indicator for transport block 2
	`RV2`	2-bits	Redundancy version for transport block 2
	`PrecodingInfo`	3-bits (2-antennas) 6-bits (4-antennas)	Precoding information
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format2A'`	`DCIFormat`	-	`'Format2A'`
	`AllocationType`	1-bit	Resource allocation header: type 0, type 1 (only if downlink bandwidth is >10 PRBs)
	`Allocation`	variable	Resource block assignment/ allocation
	`TPCPUCCH`	2-bits	PUCCH TPC command

DCI Formats	DCISTR Fields	Size	Description
	HARQNo	3-bits (FDD) 4-bits (TDD)	HARQ process number
	SwapFlag	1-bit	Transport block to codeword swap flag
	ModCoding1	5-bits	Modulation and coding scheme for transport block 1
	NewData1	1-bit	New data indicator for transport block 1
	RV1	2-bits	Redundancy version for transport block 1
	ModCoding2	5-bits	Modulation and coding scheme for transport block 2
	NewData2	1-bit	New data indicator for transport block 2
	RV2	2-bits	Redundancy version for transport block 2
	PrecodingInfo	0-bits (2 antennas) 2-bits (4 antennas)	Precoding information
	TDDIndex	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
'Format2B'	DCIFormat	-	'Format2B'
	AllocationType	1-bit	Resource allocation header: type 0, type 1 (only if downlink bandwidth is >10 PRBs)

DCI Formats	DCISTR Fields	Size	Description
	`Allocation`	variable	Resource block assignment/ allocation
	`TPCPUCCH`	2-bits	PUCCH TPC command
	`HARQNo`	3-bits (FDD) 4-bits (TDD)	HARQ process number
	`ScramblingId`	1-bit	Scrambling identity
	`ModCoding1`	5-bits	Modulation and coding scheme for transport block 1
	`NewData1`	1-bit	New data indicator for transport block 1
	`RV1`	2-bits	Redundancy version for transport block 1
	`ModCoding2`	5-bits	Modulation and coding scheme for transport block 2
	`NewData2`	1-bit	New data indicator for transport block 2
	`RV2`	2-bits	Redundancy version for transport block 2
	`TDDIndex`	2-bits	For TDD config 0, this field is not used. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
`'Format3'`	`DCIFormat`	-	`'Format3'`
	`TPCCommands`	variable	TPC commands for PUCCH and PUSCH
`'Format3A'`	`DCIFormat`	-	`'Format3A'`
	`TPCCommands`	variable	TPC commands for PUCCH and PUSCH
`'Format4'`	`DCIFormat`	-	`'Format4'`

DCI Formats	DCISTR Fields	Size	Description
	`CIF`	variable	Carier indicator
	`Allocation`	variable	Resource block assignment/ allocation
	`TPC`	2-bits	PUSCH TPC command
	`CShiftDMRS`	3-bits	Cyclic shift for DM RS
	`TDDIndex`	2-bits	For TDD config 0, this field is Uplink Index. For TDD Config 1-6, this field is the Downlink Assignment Index. Not present for FDD.
	`CQIReq`	variable	CQI request
	`SRSRequest`	2-bits	SRS request
	`AllocationType`	1-bits	Resource allocation header: non-hopping PUSCH resource allocation type 0, type 1
	`ModCoding`	5-bits	Modulation, coding scheme and redundancy version
	`NewData`	1-bits	New data indicator
	`ModCoding1`	5-bits	Modulation and coding scheme for transport block 1
	`NewData1`	1-bits	New data indicator for transport block 1
	`ModCoding2`	5-bits	Modulation and coding scheme for transport block 2
	`NewData2`	1-bits	New data indicator for transport block 2
	`PrecodingInfo`	3-bits (2 antennas) 6-bits (4 antennas)	Precoding information

Data Types: `char`

Data Types: `cell`

`dcibits` — DCI message bits

cell array of numeric vectors

DCI message bits, returned as a cell array of one or more numeric vectors. Each vector contains the bit stream of a recovered DCI message including any zero-padding. Each vector of bit values corresponds to successfully decoded DCI messages. For details, see `lteDCI`.

Data Types: `cell`

See Also

`ltePDCCH` | `ltePDCCHDecode` | `ltePDCCHDeinterleave` | `ltePDCCHIndices` | `ltePDCCHInfo` | `ltePDCCHInterleave` | `ltePDCCHPRBS` | `ltePDCCHSpace`

ltePDCCHSpace

PDCCH search space candidates

Syntax

```
ind = ltePDCCHSpace(enb,ue)
ind = ltePDCCHSpace(enb,ue,opts)
```

Description

`ind = ltePDCCHSpace(enb,ue)` returns the (0,2,4,6)-by-2 matrix ind of search space PDCCH candidate indices given the structures enb and ue. Depending on input parameters each search space will contain (0,2,4,6) PDCCH candidate locations defined by the rows of ind. Each two-element row contains the inclusive `[begin,end]` indices of a single PDCCH candidate location. By default, the 1-based indices define the PDCCH locations in the block of all multiplexed PDCCH data bits to be transmitted in that subframe. The indices can also be returned in a number of alternative formats.

The control region of a downlink subframe comprises the multiplexing of all PDCCHs bits into a single block of data which is subsequently processed and interleaved prior to PDCCH resource mapping. A UE has to blindly decode individual PDCCH directed at it and this task is simplified by subdividing the whole region into common and UE-specific search spaces which the UE should monitor. Each space comprises 2, 4, or 6 PDCCH candidates whose data length depends on its PDCCH format; each PDCCH must be transmitted on 1, 2, 4, or 8 control channel elements (CCE) (1 CCE = 72 bits).

The returned search space is of the UE-specific type unless the `RNTI` field is missing from the structure ue when a common search space is returned. The search space will always contain 2, 4, or 6 candidates; therefore, ind has 2, 4, or 6 rows, unless the parameter combinations are not valid, in which case ind will be empty. For more information, see section 9.1.1 of [1]. The candidates in a space need not be unique, especially for smaller bandwidths.

`ind = ltePDCCHSpace(enb,ue,opts)` allows control of the format of the returned indices through a cell array of option strings, opts.

Examples

Get PDCCH Search Space Candidates

Find and use PDCCH search space candidates.

To illustrate the search space structuring of the PDCCH, set up a cell wide parameter structure, `enb`, with the following field values.

```
enb.NDLRB = 50;
enb.CFI = 2;
enb.CellRefP = 2;
enb.Ng = 'Sixth';
enb.NSubframe = 0;
```

This configuration defines a control region with the following information.

```
resInfo = ltePDCCHInfo(enb)
```

```
        NREG: 240
         NRE: 960
        NCCE: 26
    NREGUsed: 234
     NREUsed: 936
        MTot: 1920
    NSymbols: 2
```

The entire data block of padded, multiplexed PDCCHs needs to be 1920 bits, `resInfo.MTot`, in length. Using –1 to represent <NIL> padding bits, create an "empty" multiplex.

```
pdcchs = -1*ones(1,resInfo.MTot);
```

Suppose you want to transmit all zeros in the first candidate of the UE-specific search space for PDCCH format 2 and the UE's `RNTI` = 1. For this format, a PDCCH spans 4 CCE or 288 bits, and the UE-specific search space contains two PDCCH candidates.

```
candidates = ltePDCCHSpace(enb,struct('PDCCHFormat',2,'RNTI',1))
```

```
        1441        1728
           1         288
```

These location values arise for `enb.NSubframe = 0`. They change in a pseudorandom fashion as the subframe number increases. Since the default candidate indices define

inclusive, 1-based bounds, we can use them to index the PDCCH data multiplex directly by using the MATLAB colon operator.

```
pdcchs(candidates(1,1):candidates(1,2)) = 0;
```

This command sets the 288 bits of the first PDCCH candidate to all zeros. The second candidate actually falls within the common search space also.

Input Arguments

`enb` — Cell-wide settings

structure

Cell-wide settings, specified as a structure with these fields.

`NDLRB` — Number of downlink resource blocks

6 | 15 | 25 | 50 | 75 | 100

Number of downlink resource blocks, specified as a numeric scalar value.

Data Types: `double`

`CFI` — Control format indicator value

1 | 2 | 3

Control format indicator value, specified as a double value.

Data Types: `double`

`CellRefP` — Number of cell-specific reference signal antenna ports

1 | 2 | 4

Number of cell-specific reference signal antenna ports, specified as 1, 2, or 4.

Data Types: `double`

`Ng` — HICH group multiplier

`'Sixth'` (default) | `'Half'` | `'One'` | `'Two'`

HICH group multiplier, returned as a string.

Data Types: `char`

`NSubframe` — Subframe number

0 (default) | nonnegative scalar integer

Subframe number, specified as a nonnegative scalar integer.

Data Types: `double`

`NREG` — Total number of resource element groups (REGs) associated with PDCCHs

Optional | nonnegative scalar integer

Total number of resource element groups (REGs) associated with PDCCHs, specified as a nonnegative scalar integer. Optional. If the `NREG` field is absent, enb must contain the following fields.

- `CyclicPrefix`
- `CellRefP`
- `CFI`
- `Ng`
- `DuplexMode`
- `TDDConfig`, but only for `'TDD'` duplex mode

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length for downlink

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for downlink, specified as a string. Optional.

Data Types: `char`

`DuplexMode` — Duplexing mode

`'FDD'` (default) | Optional | `'TDD'`

Duplexing mode, specified as a string. Optional.

Data Types: `double`

`TDDConfig` — Uplink or downlink configuration

Optional | 0...6

Uplink or downlink configuration, specified as a nonnegative scalar integer between 0 and 6. Optional. Required only for `'TDD'` duplex mode.

Data Types: `double`

Data Types: `struct`

`ue` — UE-specific cell-wide settings

structure

UE-specific cell-wide settings, specified as a structure withthe following fields.

`PDCCHFormat` — PDCCH format

0 | 1 | 2 | 3

PDCCH format, specified as a double value.

Data Types: `double`

`RNTI` — Radio Network Temporary Identifier value

1 (default) | numeric scalar

Radio Network Temporary Identifier value, specified as a numeric scalar value.

Data Types: `double`

Data Types: `struct`

`opts` — Format of the returned indices

cell array of strings

Format of the returned indices, specified as a cell array of these option strings.

`Index base` — Index base of returned indices

`'1based'` (default) | `'0based'`

Index base of returned indices, specified as a string.

Data Types: `char`

`Indexing unit` — Index unit of returned values

`'bits'` (default) | `'cce'`

Index unit of the returned values, specified as a string corresponding to bit indices or control channel elements (CCEs) indices.

Data Types: `char`

Data Types: `cell`

Output Arguments

`ind` — Search space PDCCH candidate indices

(0,2,4,6)-by-2 matrix

Search space PDCCH candidate indices, returned as a (0,2,4,6)-by-2 matrix given the structures enb and ue. It is a matrix of indices identifying a common or UE-specific PDCCH search space. Each 2-element row contains the inclusive [*begin*,*end*] indices of a single PDCCH candidate location. By default, the 1-based indices define the PDCCH locations in the block of all multiplexed PDCCH data bits to be transmitted in that subframe. The indices can also be returned in a number of alternative formats.

Data Types: `double`

References

[1] 3GPP TS 36.213. "Physical layer procedures." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePDCCH` | `ltePDCCHDecode` | `ltePDCCHDeinterleave` | `ltePDCCHIndices` | `ltePDCCHInfo` | `ltePDCCHInterleave` | `ltePDCCHPRBS` | `ltePDCCHSearch`

ltePDSCH

Physical downlink shared channel

Syntax

```
sym = ltePDSCH(enb,chs,cws)
```

Description

`sym = ltePDSCH(enb,chs,cws)` returns a matrix containing the physical downlink shared channel (PDSCH) complex symbols for cell-wide settings, enb, channel transmission configuration, chs, and the codeword or codewords contained in cws. The channel processing includes the stages of scrambling, symbol modulation, layer mapping, and precoding.

Examples

Generate PDSCH Symbols for Test Model E-TM1.1 10 Mhz

Generate the configuration structure for Test Model E-TM1.1 10 MHz, as specified in [2].

```
tm = lteTestModel('1.1','10MHz');
tm.PDSCH.RNTI = 0;
```

Generate information related to PDSCH indices.

```
prbset = (0:tm.NDLRB-1)';
[~,info] = ltePDSCHIndices(tm,tm.PDSCH,prbset);
```

Generate the PDSCH symbols.

```
pdschSym = ltePDSCH(tm,tm.PDSCH,zeros(info.G,1));
pdschSym(1:4)
```

```
   0.7071 + 0.7071i
   0.7071 + 0.7071i
   0.7071 + 0.7071i
```

```
-0.7071 + 0.7071i
```

Generate PDSCH for Test 5.3

Specify cell-wide settings in parameter structure `enb`.

```
enb.NDLRB = 50;
enb.NCellID = 1;
enb.CellRefP = 2;
enb.NSubframe = 0;
enb.CFI = 2;
```

Specify channel configuration settings in `cfg`.

```
cfg.TxScheme = 'SpatialMux';
cfg.NLayers = 2;
cfg.Modulation = {'16QAM' '16QAM'};
cfg.PMISet = [0,1,0,1,0,1,0,1,0];
cfg.PRBSet = [(0:49)',(0:49)'];
cfg.RNTI = 1;
cfg.RV = [0,0];
```

Perform DL-SCH coding and PDSCH modulation for Test 5.3: RMC FDD R.11 10MHz 16QAM 2-layer spatial multiplexing, frequency selective precoding (Subframe 0), as specified in [1].

```
trData = {round(rand(1,12960)),round(rand(1,12960))};
cws = lteDLSCH(enb,cfg,[24768,24768],trData);
sym = ltePDSCH(enb,cfg,cws);
```

Input Arguments

enb — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`NCellID`	Required	Nonnegative scalar integer (0, …, 503)	Physical layer cell identity

Parameter Field	Required or Optional	Values	Description
NSubframe	Required	Nonnegative scalar integer	Subframe number
CellRefP	Required	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
DuplexMode	Optional	'FDD' (default), 'TDD'	Duplex mode
The following parameters are dependent upon the condition that DuplexMode is set to 'TDD'.			
TDDConfig	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
SSC	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)
The following parameter fields are dependent upon the condition that chs.TxScheme is set to 'SpatialMux' or 'MultiUser'.			
CFI	Required	1, 2, 3	Control format indicator (CFI) value
NDLRB	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
CyclicPrefix	Optional	'Normal' (default), 'Extended'	Cyclic prefix length

chs — Channel-specific transmission configuration

structure

Channel-specific transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
Modulation	Required	'QPSK', '16QAM', '64QAM', cell array of strings	Modulation type, specified as a string or cell array of strings. If 2 blocks, each cell is associated with a transport block.

Parameter Field	Required or Optional	Values	Description
RNTI	Required	Scalar integer	Radio network temporary identifier (RNTI) value (16 bits)
TxScheme	Required	'SpatialMux' (default), 'Port0', 'TxDiversity', 'CDD', 'MultiUser', 'Port5', 'Port7-8', 'Port8', 'Port7-14'	Transmission scheme, specified as one of the following options. • 'SpatialMux' — Closed-loop spatial multiplexing. • 'Port0' — Single-antenna port, port 0. • 'TxDiversity' — Transmit diversity scheme. • 'CDD' — Large delay CDD scheme. • 'MultiUser' — Multiuser MIMO scheme. • 'Port5' — Single-antenna port, port 5. • 'Port7-8' — Single-antenna port, port 7 (**NLayers** = 1). Dual layer transmission, ports 7 and 8 (**NLayers** = 2). • 'Port8' — Single-antenna port, port 8. • 'Port7-14' — Up to 8-layer transmission, ports 7–14.
NLayers	Required	1,...,8, depending on TxScheme	Number of transmission layers (downlink modulation)
The following parameters are dependent upon the condition that TxScheme is set to 'SpatialMux' or 'MultiUser'.			

Parameter Field	Required or Optional	Values	Description
PMISet	Required	Integer vector (0, ..., 15)	Precoder matrix indication (PMI) set. It can contain either a single value, corresponding to single PMI mode, or multiple values, corresponding to multiple or subband PMI mode. The number of values depends on the CellRefP, transmission layers and TxScheme.
PRBSet	Required	1- or 2-column integer matrix	0-based physical resource block (PRB) indices corresponding to the resource allocations for this PDSCH. As a column vector, the resource allocation is the same in both slots of the subframe. As a 2-column matrix, this parameter specifies different PRBs for each slot in a subframe.
The following parameters are dependent upon the condition that `TxScheme` is set to `'Port5'`, `'Port7-8'`, `'Port8'`, or `'Port7-14'`.			
W	Optional	Numeric matrix, `[]` (default)	Precoding matrix for the UE-specific beamforming of the PDSCH symbols, specified as a numeric matrix of size `NLayers`-by-`NTxAnts`. The default value is an empty matrix, `[]`, which signifies that there is no precoding.

cws — Codeword or codewords

numeric vector | cell array

Codeword or codewords, specified as a vector of bit values for one codeword to be modulated, or a cell array containing one or two vectors of bit values corresponding to the one or two codewords to be modulated.

Output Arguments

sym — PDSCH symbols

complex numeric matrix

PDSCH symbols, returned as a complex numeric matrix. It has size *N*-by-*P*, where *N* is the number of modulation symbols for one antenna port and *P* is the number of transmission antennas. The complex symbols are generated using cell-wide settings, enb, channel transmission configuration, chs, and the codeword or codewords contained in cws.

Data Types: `double`
Complex Number Support: Yes

References

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.141. "Base Station (BS) conformance testing." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDLSCH` | `ltePDSCHDecode` | `ltePDSCHIndices` | `ltePDSCHPRBS`

ltePDSCHDecode

Physical downlink shared channel decoding

Syntax

```
[cws,symbols] = ltePDSCHDecode(enb,chs,sym)
[cws,symbols] = ltePDSCHDecode(enb,chs,sym,hest,noiseest)
[cws,symbols] = ltePDSCHDecode(enb,chs,rxgrid,hest,noiseest)
```

Description

`[cws,symbols] = ltePDSCHDecode(enb,chs,sym)` performs the inverse of physical downlink shared channel (PDSCH) processing on the matrix of complex modulated PDSCH symbols, sym, cell-wide settings structure, enb, and channel-specific configuration structure, chs. The channel inverse processing includes inverting the channel precoding, layer demapping and codeword separation, soft demodulation, and descrambling. Inverting the precoding is accomplished by matrix pseudoinversion of the precoding matrices. It returns a cell array, cws, of soft bit vectors, or codewords, and a cell array, symbols, of received constellation symbol vectors resulting from performing the inverse of Physical Downlink Shared Channel (PDSCH) processing. For details, see section 6.4 of [1] and `ltePDSCH`. cws is optionally scaled by channel state information (CSI) calculated during the equalization process.

`[cws,symbols] = ltePDSCHDecode(enb,chs,sym,hest,noiseest)` performs the decoding of the complex modulated PDSCH symbols sym using cell-wide settings, enb, channel-specific configuration, chs, channel estimate, hest, and the noise estimate, noiseest.

The behavior varies based on the chs`.TxScheme` setting. For the `TxDiversity` transmission scheme, the precoding inversion is performed using an orthogonal space frequency block code (OSFBC) decoder. For the `SpatialMux`, `CDD`, and `MultiUser` transmission schemes, the precoding inversion is performed using a multiple-input, multiple-output (MIMO) minimum mean square error (MMSE) equalizer, equalizing between transmitted and received layers. For the `'Port0'`, `'Port5'`, `'Port7-8'`, `'Port8'`, and `'Port7-14'` transmission schemes, the reception is performed using MMSE equalization. The input channel estimate, hest, is assumed to be with reference

to the transmission layers, using the UE-specific reference signals, so the MMSE equalization will produce MMSE equalized layers.

noiseest is an estimate of the noise power spectral density per RE on the received subframe. This estimate is provided by the `lteDLChannelEstimate` function.

`[cws,symbols] = ltePDSCHDecode(enb,chs,rxgrid,hest,noiseest)` accepts the full received resource grid, rxgrid, for one subframe, in place of the sym input; the decoder will internally extract the PDSCH REs to obtain the complex modulated PDSCH symbols. rxgrid is a 3-D *M*-by-*N*-by-`NRxAnts` array of resource elements, where *M* and *N* are the number of subcarriers and symbols for one subframe for cell-wide settings enb and `NRxAnts` is the number of receive antennas. In this case, hest is a 4-D *M*-by-*N*-by-`NRxAnts`-by-`CellRefP` array where *M* and *N* are the number of subcarriers and symbols for one subframe for cell-wide settings enb, `NRxAnts` is the number of receive antennas, and `CellRefP` is the number of cell-specific reference signal antenna ports, given by enb`.CellRefP`. hest is processed to extract the channel estimates relevant to the PDSCH, those in the time and frequency locations corresponding to the PDSCH REs in rxgrid.

Examples

Decode PDSCH Symbols

Generate the complex PDSCH modulated symbols for RMC R.0. Then, decode them using the cell-wide settings structure, `enb`.

```
enb = lteRMCDL('R.0');
codewordBits = randi([0,1],enb.PDSCH.CodedTrBlkSizes(1),1);
pdschSym = ltePDSCH(enb,enb.PDSCH,codewordBits);
[rxCws,symbols] = ltePDSCHDecode(enb,enb.PDSCH,pdschSym);
```

Input Arguments

enb — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`NCellID`	Required	Nonnegative scalar integer (0, …, 503)	Physical layer cell identity
`NSubframe`	Required	Nonnegative scalar integer	Subframe number
`CellRefP`	Required	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplex mode
The following parameters are dependent upon the condition that `DuplexMode` is set to `'TDD'`.			
`TDDConfig`	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
`SSC`	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)
The following parameter fields are dependent upon the condition that chs`.TxScheme` is set to `'SpatialMux'` or `'MultiUser'`.			
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`CFI`	Required	1, 2, 3	Control format indicator (CFI) value
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length

`chs` — Channel-specific transmission configuration

structure

Channel-specific transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`Modulation`	Required	`'QPSK'`, `'16QAM'`, `'64QAM'`, cell array of strings	Modulation type, specified as a string or cell array of strings. If 2 blocks, each cell is associated with a transport block.
`RNTI`	Required	Scalar integer	Radio network temporary identifier (RNTI) value (16 bits)
`TxScheme`	Required	`'SpatialMux'` (default), `'Port0'`, `'TxDiversity'`, `'CDD'`, `'MultiUser'`, `'Port5'`, `'Port7-8'`, `'Port8'`, `'Port7-14'`	Transmission scheme, specified as one of the following options. • `'SpatialMux'` — Closed-loop spatial multiplexing. • `'Port0'` — Single-antenna port, port 0. • `'TxDiversity'` — Transmit diversity scheme. • `'CDD'` — Large delay CDD scheme. • `'MultiUser'` — Multiuser MIMO scheme. • `'Port5'` — Single-antenna port, port 5. • `'Port7-8'` — Single-antenna port, port 7 (**`NLayers`** = 1). Dual layer transmission, ports 7 and 8 (**`NLayers`** = 2). • `'Port8'` — Single-antenna port, port 8. • `'Port7-14'` — Up to 8-layer transmission, ports 7–14.
`NLayers`	Required	1,...,8, depending on `TxScheme`	Number of transmission layers (downlink modulation)
`CSI`	Optional	`'Off'` (default), `'On'`	CSI weighting for soft bits flag. If on, soft bits are weighted by CSI.

Parameter Field	Required or Optional	Values	Description
The following parameters are dependent upon the condition that `TxScheme` is set to `'SpatialMux'` or `'MultiUser'`.			
`PMISet`	Required	Integer vector (0, ..., 15)	Precoder matrix indication (PMI) set. It can contain either a single value, corresponding to single PMI mode, or multiple values, corresponding to multiple or subband PMI mode. The number of values depends on the CellRefP, transmission layers and TxScheme.
`PRBSet`	Required	1- or 2-column integer matrix	0-based physical resource block (PRB) indices corresponding to the resource allocations for this PDSCH. As a column vector, the resource allocation is the same in both slots of the subframe. As a 2-column matrix, this parameter specifies different PRBs for each slot in a subframe.
The following parameters are dependent upon the condition that `TxScheme` is set to `'Port5'`, `'Port7-8'`, `'Port8'`, or `'Port7-14'`.			
`W`	Optional	Numeric matrix, `[]` (default)	Precoding matrix for the UE-specific beamforming of the PDSCH symbols, specified as a numeric matrix of size `NLayers`-by-`NTxAnts`. The default value is an empty matrix, `[]`, which signifies that there is no precoding.

`sym` — Complex modulated PDSCH symbols

Numeric matrix

Complex modulated PDSCH symbols, specified as a numeric matrix of size `NRE`-by-`NRxAnts`. `NRE` is the number of QAM symbols per antenna assigned to the PDSCH and `NRxAnts` is the number of receive antennas.

Data Types: `double`
Complex Number Support: Yes

`hest` — Channel estimate

3-D numeric array | 4-D numeric array

Channel estimate, specified as a 3-D or 4-D numeric array. For the `'Port0'`, `'TxDiversity'`, `'SpatialMux'`, `'CDD'`, and `'MultiUser'` transmission schemes, the array size is `NRE`-by-`NRxAnts`-by-`CellRefP`, where `NRE` is the number of QAM symbols per antenna assigned to the PDSCH, `NRxAnts` is the number of receive antennas, and `CellRefP` is the number of cell-specific reference signal antennas, given by `enb.CellRefP`. For the `'Port5'`, `'Port7-8'`, `'Port8'`, and `'Port7-14'` transmission schemes, the array size is `NRE`-by-`NRxAnts`-by-`NLayers`, where `NLayers` is the number of transmission layers given by `chs.NLayers`.

When rxgrid is supplied, hest is a 4-D numeric array of size *M*-by-*N*-by-`NRxAnts`-by-`CellRefP`, where *M* and *N* are the number of subcarriers and symbols for one subframe for cell-wide settings, enb, `NRxAnts` is the number of receive antennas, and `CellRefP` is the number of cell-specific reference signal antenna ports, given by `enb.CellRefP`.

Data Types: `double`

`noiseest` — Noise estimate

numeric array

Noise estimate of the noise power spectral density per RE on the received subframe, specified as a numeric array.

Data Types: `double`

`rxgrid` — Full received resource grid

numeric array

Full received resource grid, specified as a 3-D *M*-by-*N*-by-`NRxAnts` array of resource elements, where *M* and *N* are the number of subcarriers and symbols for one subframe for cell-wide settings enb and `NRxAnts` is the number of receive antennas.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`cws` — Codeword or codewords

numeric vector | cell array

Codeword or codewords, returned as a vector of bit values for one codeword to be modulated, or a cell array containing one or two vectors of bit values corresponding to the one or two codewords to be modulated.

Data Types: `double`

`symbols` — Received constellation symbols

Cell array of column vectors

Received constellation symbols, returned as a cell array of complex double column vectors, resulting from performing the inverse of PDSCH processing.

Data Types: `cell`

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`lteDLSCHDecode` | `ltePDSCH` | `ltePDSCHIndices` | `ltePDSCHPRBS`

ltePDSCHIndices

Physical downlink shared channel (PDSCH) resource element indices

Syntax

```
[ind,info] = ltePDSCHIndices(enb,chs,prbset)
[ind,info] = ltePDSCHIndices(enb,chs,prbset,opts)
```

Description

`[ind,info] = ltePDSCHIndices(enb,chs,prbset)` returns a matrix, ind, containing physical downlink shared channel (PDSCH) resource element (RE) indices and a structure, info, containing information related to the PDSCH indices. By default, the indices are returned in 1-based linear indexing form that can directly index elements of a 3-D array representing the subframe resource grid for all antenna ports. This function is initialized with cell-wide settings enb, channel transmission configuration chs, and physical resource block indices prbset.

prbset contains the Physical Resource Block (PRB) indices corresponding to the resource allocation for this PDSCH transmission. You can specify prbset as either a column vector or a two-column matrix. If you specify a column vector, the resource allocation is the same in both slots of the subframe. You can use the two-column matrix to specify PRBs if the PRBs in the first and second slots of the subframe differ. Note that the PRB indices are 0-based.

Each column of the returned *N*-by-*P* matrix, ind, contains the per-antenna indices for the *N* resource elements in each of the *P* resource array planes. For the `'Port0'`, `'TxDiversity'`, `'CDD'`, `'SpatialMux'`, and `'MultiUser'` transmission schemes, *P* = `enb.CellRefP`. For the other transmission schemes, *P* = `chs.NTxAnts`. If `chs.NTxAnts = 0` or is absent, the matrix ind is of size *N*-by-*NU* and contains the per-layer indices for the *N* resource elements in each of *NU* resource array planes associated with the layers, where *NU* = `chs.NLayers`. You can also return the indices in alternative indexing formats using the opts argument.

The info structure contains parameter fields `G` and `Gd`. `info.G` provides the appropriate size of the DL-SCH coder output, which is required as the parameter `outlen` provided

to the `lteDLSCH` function. `info.Gd` is the number of coded and rate-matched DL-SCH data symbols, equal to the number of rows in the PDSCH indices. To provide accurate information in info, the `Modulation`, `TxScheme`, and `Nlayers` fields are required in chs. These extra fields are required only if the info output is assigned when you call the function.

`[ind,info] = ltePDSCHIndices(enb,chs,prbset,opts)` enables control over the format of the returned indices through option strings opts.

Examples

Generate PDSCH RE Indices

This example generates the 0-based PDSCH resource element (RE) indices mapping in linear index form for the 4-antenna case.

Create the cell-wide settings structure, `enb`.

```
enb = lteRMCDL('R.14');
enb.NDLRB = 6;
enb.CFI = 1;
enb.PDSCH.PRBSet = (1:enb.NDLRB-1).';
```

Generate PDSCH RE indices, specifying the 0-based and linear options.

```
ind = ltePDSCHIndices(enb,enb.PDSCH, ...
    enb.PDSCH.PRBSet,{'0based','ind'});
ind(1:10,:)
```

```
ans =

         156        1164        2172        3180
         157        1165        2173        3181
         158        1166        2174        3182
         159        1167        2175        3183
         160        1168        2176        3184
         161        1169        2177        3185
         162        1170        2178        3186
         163        1171        2179        3187
         164        1172        2180        3188
         165        1173        2181        3189
```

The result, `ind`, is a matrix of 0-based mapping indices in linear index form. Since this is example is for the 4-antenna case, `ind`, has 4 columns.

Input Arguments

`enb` — eNodeB cell-wide settings

structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`NCellID`	Required	Nonnegative scalar integer (0, …, 503)	Physical layer cell identity
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
`CellRefP`	Required	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
`NSubframe`	Required	Nonnegative scalar integer	Subframe number
`CFI`	Required	1, 2, 3	Control format indicator (CFI) value
`DuplexMode`	Optional	`'FDD'` (default), `'TDD'`	Duplexing mode, specified as one of the following: • `'FDD'` — Frequency division duplex (default) • `'TDD'` — Time division duplex
The following apply when `DuplexMode` is set to `'TDD'`.			
`TDDConfig`	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration

Parameter Field	Required or Optional	Values	Description
SSC	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)

The following parameters apply to the `enb` structure when the transmission scheme `chs.TxScheme` is set to `'Port7-14'`.

Parameter Field	Required or Optional	Values	Description
NFrame	Optional	0 (default), Nonnegative scalar integer	Frame number
CSIRSPeriod	Optional	`'Off'` (default), `'On'`, `Icsi-rs`, (0,...,154), `[Tcsi-rs Dcsi-rs]`	CSI-RS subframe configuration
The following parameters apply when `CSIRSPeriod` is set to any value but `'Off'`.			
CSIRSConfig	Required	Nonnegative scalar integer	CSI-RS configuration index. See table 6.10.5.2-1 in TS 36.211.
CSIRefP	Required	1 (default), 2, 4, 8	Number of CSI-RS antenna ports
ZeroPowerCSIRSPerio	Optional	`'Off'` (default), `'On'`, `Icsi-rs`, (0,...,154), `[Tcsi-rs Dcsi-rs]`	Zero-power CSI-RS subframe configuration
The following parameters apply when `ZeroPowerCSIRSPeriod` is set to any value but `'Off'`.			
ZeroPowerCSIRSC	Required	16-bit bitmap string (truncated if not 16 bits or `'0'` MSB extended), numerical list of CSI-RS configuration indices. See table 6.10.5.2-1 (4 CSI reference signal column) in TS 36.211.	Zero-power CSI-RS configuration index list. See table 6.10.5.2 in TS 36.211.

Data Types: `struct`

chs — PDSCH-specific channel transmission configuration

structure

PDSCH-specific channel transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`TxScheme`	Optional	`'Port0'` (default), `'TxDiversity'`, `'CDD'`, `'SpatialMux'`, `'MultiUser'`, `'Port5'`, `'Port7-8'`, `'Port8'`, `'Port7-14'`	Transmission scheme, specified as one of the following options. • `'Port0'` — Single-antenna port, port 0 (default). • `'TxDiversity'` — Transmit diversity scheme. • `'CDD'` — Large delay CDD scheme. • `'SpatialMux'` — Closed-loop spatial multiplexing. • `'MultiUser'` — Multiuser MIMO scheme. • `'Port5'` — Single-antenna port, port 5. • `'Port7-8'` — Single-antenna port, port 7 (**`NLayers`** = 1). Dual layer transmission, ports 7 and 8 (**`NLayers`** = 2). • `'Port8'` — Single-antenna port, port 8. • `'Port7-14'` — Up to 8-layer transmission, ports 7–14.
The following parameters apply when `TxScheme` is set to `'Port5'`, `'Port7-8'`, `'Port8'`, or `'Port7-14'`.			
`NTxAnts`	Optional	Nonnegative integer, 0 (default)	Number of transmission antenna ports. This argument is only present for UE-specific demodulation reference symbols.

Parameter Field	Required or Optional	Values	Description
To provide accurate information in info, you are required to define `TxScheme` and the following additional parameters.			
`Modulatio`	Optional	`'QPSK'` (default), `'16QAM'`, `'64QAM'`, cell array of strings	Codeword modulation format, specified as a string or a cell array of one or two strings. To specify the modulation format for one codeword, use a string. To specify the modulation formats for two codewords, use a cell array of two strings.
`NLayers`	Optional	1 (default), 2, 3, 4, 5, 6, 7, 8	Number of transmission layers

Data Types: `struct`

`prbset` — Physical resource block indices

column vector | 2-column numeric matrix

Physical resource block indices, specified as a column vector or a 2-column numeric matrix. This argument contains the Physical Resource Block (PRB) indices corresponding to the resource allocation for this PDSCH transmission. If you specify a column vector, the resource allocation is the same in both slots of the subframe. You can use the 2-column matrix to specify PRBs if the PRBs in the first and second slots of the subframe differ. The PRB indices are 0-based.

Data Types: `double`

`opts` — Index generation options

string | cell array of strings

Index generation options, specified as a string or a cell array of strings that can contain the following values.

Option	Values	Description
Indexing style	`'ind'` (default), `'sub'`	Style for the returned indices, specified as one of the following options. • `'ind'` — returns the indices in linear index form as a column vector (default)

Option	Values	Description
		• `'sub'` — returns the indices in `[subcarrier, symbol, antenna]` subscript row style. The number of rows in the output, ind, is the number of resource elements (NRE). Thus, ind is an NRE-by-3 matrix.
Index base	`'1based'` (default), `'0based'`	Base value of the returned indices. Specify `'1based'` to generate indices where the first value is one. Specify `'0based'` to generate indices where the first value is zero.

Data Types: `char` | `cell`

Output Arguments

`ind` — Physical downlink shared channel (PDSCH) resource element (RE) indices

matrix

Physical downlink shared channel (PDSCH) resource element (RE) indices, specified as a matrix. Each column of the *N*-by-*P* matrix, ind, contains the per-antenna indices for the *N* resource elements in each of the *P* resource array planes. For the `'Port0'`, `'TxDiversity'`, `'CDD'`, `'SpatialMux'`, and `'MultiUser'` transmission schemes, *P* = `enb.CellRefP`. For the other transmissions schemes, *P* = `chs.NTxAnts`. If `chs.NTxAnts = 0` or is absent, the matrix, ind, is of size *N*-by-*NU* and contains the per-layer indices for the *N* resource elements in each of *NU* resource array planes associated with the layers, where *NU* = `chs.NLayers`. You can also return the indices in alternative indexing formats using the opts argument.

Note: The active or zero-power CSI-RS resource elements are excluded from the output indices only for the Release 10, `'Port7-14'` transmission scheme. For all other schemes, the CSI-RS resource element indices are not avoided, which results in a Release 8/9 compatible PDSCH. Any active or zero-power CSI-RS would overwrite the associated PDSCH REs later in the subframe construction.

`info` — Information related to PDSCH indices

structure

Information related to PDSCH indices, returned as a structure. To provide accurate information in info, the channel transmission configuration structure, chs, must contain

the fields `TxScheme`, `Modulation`, and `NLayers`. The structure info has the following fields.

Parameter Field	Description	Values	Data Type
`G`	Number of coded and rate-matched DL-SCH data bits for each codeword.	1- or 2-element vector	`int32`
`Gd`	Number of coded and rate-matched DL-SCH data symbols.	Integer equal to the number of rows in the PDSCH indices	`uint32`

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePDSCH` | `ltePDSCHDecode` | `ltePDSCHPRBS`

ltePDSCHPRBS

PDSCH pseudorandom scrambling sequence

Syntax

```
seq = ltePDSCHPRBS(enb,rnti,ncw,n)
seq = ltePDSCHPRBS(enb,rnti,ncw,n,mapping)
```

Description

`seq = ltePDSCHPRBS(enb,rnti,ncw,n)` returns a column vector containing the first n outputs of the Physical Downlink Shared Channel (PDSCH) scrambling sequence when initialized according to cell-wide settings, enb, 16-bit rnti, and ncw, which is either 0 or 1, indicating which codeword this sequence would scramble.

`seq = ltePDSCHPRBS(enb,rnti,ncw,n,mapping)` allows control over the format of the returned sequence seq through the string mapping. Valid formats are `'binary'`, the default, and `'signed'`. The `'binary'` format maps true to 1 and false to 0. The `'signed'` format maps true to –1 and false to 1.

Examples

Generate PDSCH Pseudorandom Binary Scrambling Sequence

Generate a physical downlink shared channel (PDSCH) pseudorandom scrambling sequence for two different codeword numbers.

When transmitting multiple codewords via spatial multiplexing, each codeword uses a different scrambling sequence. Generate the first 10 values of the PRBS sequence for the first codeword. Set `ncw` to 0, and set `rnti` to 11.

```
enb = lteRMCDL('R.0');
pdschPrbsSeq1 = ltePDSCHPRBS(enb,11,0,10)
```

```
     0
     1
```

```
     1
     1
     0
     1
     0
     1
     0
     1
```

Next, generate the first 10 values of the PRBS sequence for the second codeword. To do so, set `ncw` to 1.

```
pdschPrbsSeq2 = ltePDSCHPRBS(enb,11,1,10)
```

```
     0
     0
     1
     1
     0
     0
     0
     1
     0
     1
```

Input Arguments

`enb` — Cell-wide settings
structure

Cell-wide settings, specified as a structure. enb contains the following fields.

`NCellID` — Physical layer cell identity
nonnegative integer

Physical layer cell identity, specified as a nonnegative integer.

Data Types: `double`

`NSubframe` — Subframe number
nonnegative integer

Subframe number, specified as a nonnegative integer.

Data Types: `double`

Data Types: `struct`

`rnti` — Radio network temporary identifier

nonnegative integer

Radio network temporary identifier, specified as nonnegative integer.

Data Types: `double`

`ncw` — Number of codewords

0 | 1

Number of codewords, specified as a 0 or 1. This input indicates which codeword this sequence scrambles.

Data Types: `double`

`n` — Length of scrambling sequence

positive integer

Length of scrambling sequence, specified as a positive integer.

Data Types: `double`

`mapping` — Format of returned sequence

`'binary'` (default) | `'signed'`

Format of returned sequence, specified as a string. This input controls the format of the returned sequence, seq. The string `'binary'` maps true to 1 and false to 0. The string `'signed'` maps true to –1 and false to 1.

Data Types: `char`

Output Arguments

`seq` — PDSCH scrambling sequence

logical column vector | numeric column vector

PDSCH scrambling sequence, returned as a numeric column vector. This output argument contains the first n outputs of the PDSCH pseudorandom scrambling sequence.

Data Types: `logical` | `double`

See Also

`ltePDSCH` | `ltePDSCHDecode` | `ltePDSCHIndices`

ltePHICH

Physical hybrid ARQ indicator channel

Syntax

```
[sym,info] = ltePHICH(enb,hiset)
```

Description

`[sym,info] = ltePHICH(enb,hiset)` returns a matrix, sym, of complex modulation symbols generated by the set of Physical Hybrid ARQ Indicator Channels (PHICH) in a subframe. The channel processing includes the stages of BPSK modulation, scrambling, orthogonal sequence spreading, REG alignment, layer mapping and precoding, followed by PHICH group summation and mapping unit creation. The optional returned structure, info, contains control resourcing information about the output symbols. See `ltePHICHInfo` for details and further background.

The control region of a subframe can contain up to *NPHICH* separate PHICH, each carrying a single hybrid ARQ (HARQ) acknowledgment (ACK) or negative acknowledgment (NACK). The number of rows of sym is the number of resource elements (NRE). Each column of the NRE-by-`CellRefP` matrix sym contains the per-antenna symbols for the combined set of PHICHs, where *NPHICH*, returned as info`.`NPHICH, is defined via the settings in the enb structure.

The *R*-by-3 input matrix, hiset, configures up to NPHICH individual PHICH that are combined together in the output. Each row of hiset defines a single PHICH in terms of `[nGroup,nSeq,hi]`, where `nGroup` is the PHICH group index number, `nSeq` is the sequence index number within the group, and `hi` is 1 or 0 representing ACK or NACK, respectively. These indices are 0-based. In terms of info structure fields, the following conditions apply.

- *R* < info`.`NPHICH
- `nGroup` < info`.`NGroups
- `nSeq` < info`.`NSequences

The output matrix, sym, always contains info`.`NRE symbols to map into the total PHICH resource allocation, even if less than the full set of NPHICH channels are configured.

Examples

Generate PHICH Complex Modulation Symbols

Generate physical HARQ indicator channel (PHICH) complex modulation symbols for three different HARQ indicator (HI) sets. An HI set is comprised of the PHICH group index number, the sequence number within the group, and an ACK/NACK.

In a system subframe (normal CP) with `enb.NDLRB = 50` and `enb.Ng = 'Half'`, 16 PHICH are available split between 2 PHICH groups of 8 sequences. These are mapped to `NRE`=24 resource elements. Modulate a NACK (`hi`=0) onto the third orthogonal sequence (`nSeq`=2) of the second group (`nGroup`=1).

```
enb = lteRMCDL('R.7');
out1 = ltePHICH(enb,[1,2,0]);
out1(1:4)
```

```
     0
     0
     0
     0
```

Next, add in an ACK on the last sequence of the first group.

```
enb = lteRMCDL('R.7');
out2 = ltePHICH(enb,[1,2,0;0,7,1]);
out2(1:4)
```

```
   0.7071 - 0.7071i
   0.7071 - 0.7071i
  -0.7071 + 0.7071i
   0.7071 - 0.7071i
```

Finally, provide an empty matrix for hiset.

```
enb = lteRMCDL('R.7');
out3 = ltePHICH(enb,[]);
size(out3)
```

```
    24     1
```

The result is a 24-by-`enb.CellRefP` matrix of zeros.

Input Arguments

`enb` — Cell-wide settings

scalar structure

Cell-wide settings, specified as a scalar structure. enb contains the following fields. The TDDConfig parameter field is only required if DuplexMode is set to `'TDD'`.

`NDLRB` — Number of downlink resource blocks

positive integer

Number of downlink resource blocks, specified as a positive integer.

Data Types: `double`

`NCellID` — Physical layer cell identity

nonnegative integer

Physical layer cell identity, specified as a nonnegative integer.

Data Types: `double`

`CellRefP` — Number of cell-specific reference signal antenna ports

1 | 2 | 4

Number of cell-specific reference signal antenna ports, specified as 1, 2, or 4.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`NSubframe` — Subframe number

nonnegative integer

Subframe number, specified as a nonnegative integer.

Data Types: `double`

`Ng` — HICH group multiplier
`'Sixth'` | `'Half'` | `'One'` | `'Two'`

HICH group multiplier, specified as a string.

Data Types: `char`

`DuplexMode` — Duplex mode
`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, specified as `'FDD'` or `'TDD'`. Optional.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration
0 (default) | Optional | 1 | 2 | 3 | 4 | 5 | 6

Uplink or downlink configuration, specified as a nonnegative scalar integer between 0 and 6. Optional. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

Data Types: `struct`

`hiset` — HARQ indicator set
numeric matrix

HARQ indicator set, specified as a numeric matrix of size *R*-by-3. With this input matrix, you can configure up to NPHICH individual PHICH that are combined together in the output. Each row of hiset defines a single PHICH in terms of `[nGroup,nSeq,hi]`, where `nGroup` is the PHICH group index number, `nSeq` is the sequence index number within the group and `hi` is 1 or 0 representing ACK or NACK, respectively. These indices are 0-based.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`sym` — PHICH complex modulation symbols
numeric matrix

PHICH complex modulation symbols, returned as a numeric matrix. Each column of this NRE-by-CellRefP matrix contains the per-antenna symbols for the combined set of PHICHs, where info.NPHICH is defined via the settings in the enb structure.

Data Types: `double`
Complex Number Support: Yes

`info` — Control resourcing information about output symbols
scalar structure

Control resourcing information about the output symbols, returned as a scalar structure. info contains the following fields.

`NRE` — Number of resource elements assigned to all PHICH
positive integer

Number of resource elements assigned to all PHICH, returned as a positive integer.

Data Types: `uint64`

`NREG` — Number of resource element groups assigned to all PHICH
positive integer

Number of resource element groups assigned to all PHICH, returned as a positive integer.

Data Types: `uint64`

`NPHICH` — Number of individual PHICH available
positive integer

Number of individual PHICH available, returned as a positive integer.

Data Types: `uint64`

`NGroups` — Number of PHICH groups
positive integer

Number of PHICH groups, returned as a positive integer.

Data Types: `int8`

`NMappingUnits` — Number of PHICH mapping units
positive integer

Number of PHICH mapping units, returned as a positive integer.

Data Types: `int8`

`NSequences` — Number of orthogonal sequences in each PHICH group
positive integer

Number of orthogonal sequences in each PHICH group, returned as a positive integer.

Data Types: `int8`

`PHICHDuration` — PHICH duration
integer

PHICH duration, returned as an integer.

Data Types: `int8`

Data Types: `struct`

See Also

`ltePHICHDecode` | `ltePHICHIndices` | `ltePHICHInfo` | `ltePHICHPRBS` | `ltePHICHPrecode`

ltePHICHDecode

Physical hybrid ARQ indicator channel decoding

Syntax

```
[hi,symbols] = ltePHICHDecode(enb,hires,sym)
[hi,symbols] = ltePHICHDecode(enb,hires,sym,hest,noiseest)
[hi,symbols] = ltePHICHDecode(enb,hires,sym,hest,noiseest,alg)
```

Description

`[hi,symbols] = ltePHICHDecode(enb,hires,sym)` performs the inverse of Physical Hybrid ARQ Indicator Channel (PHICH) processing on the matrix of complex modulated PHICH symbols, sym, PHICH resources, hires, and cell-wide settings structure, enb. The channel inverse processing includes deprecoding, symbol demodulation and descrambling. It returns a column vector of soft hybrid ARQ indicator values hi and received constellation of complex symbol vector symbols resulting from performing the inverse of Physical Hybrid ARQ Indicator Channel (PHICH) processing. For details, see section 6.9 of [1] and `ltePHICH`. hi is optionally scaled by channel state information (CSI) calculated during the equalization process.

hires is an *R*-by-2 matrix configuring up to NPHICH individual PHICH that are decoded. Each row of HIRES defines a single PHICH in terms of [`nGroup`, `nSeq`] where `nGroup` is the PHICH group index number and `nSeq` is the sequence index number. These indices are 0-based. In terms of `ltePHICHInfo` `info` structure fields, the following conditions apply:
R < `info.NPHICH`
`nGroup` < `info.NGroups`
`nSeq` < `info.NSequences`

sym must be a matrix of `NRE`-by-`NRxAnts` complex modulated PHICH symbols. `NRE` is the number of BPSK symbols per antenna assigned to the PHICH and `NRxAnts` is the number of receive antennas. The input matrix sym should always contain `info.NRE` symbols corresponding to the total PHICH resource allocation, even if less than the full set of NPHICH channels were configured.

`[hi,symbols] = ltePHICHDecode(enb,hires,sym,hest,noiseest)` performs the decoding of the complex PHICH symbols sym using cell-wide settings enb, PHICH resources hires, the channel estimate hest and the noise estimate noiseest. For the `TxDiversity` transmission scheme (`CellRefP`=2 or `CellRefP`=4), the reception is performed using an Orthogonal Space Frequency Block Code (OSFBC) decoder. For the `'Port0'` transmission scheme (`CellRefP`=1), the reception is performed using MMSE equalization.

hest is a 3-D `NRE`-by-`NRxAnts`-by-`CellRefP` array where `NRE` are the frequency and time locations corresponding to the PHICH RE positions (a total of `NRE` positions), `NRxAnts` is the number of receive antennas, and `CellRefP` is the number of cell-specific reference signal antennas, given by `enb.CellRefP`.

noiseest is an estimate of the noise power spectral density per RE on received subframe; such an estimate is provided by the `lteDLChannelEstimate` function.

`[hi,symbols] = ltePHICHDecode(enb,hires,sym,hest,noiseest,alg)` is the same as above except it provides control over weighting the output soft bits with Channel State Information (CSI) calculated during the equalization stage using algorithmic configuration structure, alg.

Examples

Decode PHICH Symbols

Generate the complex modulation symbols for RMC R.1 by the set of PHICH in a subframe. Define an individual PHICH with each row of the matrix, `hiset`.

```
enb = lteRMCDL('R.1');
hiset = [1,1,1;1,2,0];
phichSym = ltePHICH(enb,hiset);
```

Decode the transmitted Hybrid ARQ indicator value, using `hiset`.

```
hi = ltePHICHDecode(enb,hiset(:,1:2),phichSym);
isequal(hi,hiset(:,3))
```

1

Input Arguments

`enb` — Cell-wide settings

scalar structure

Cell-wide settings, specified as a scalar structure. enb is a structure having the following fields.

`NCellID` — Physical layer cell identity

nonnegative integer

Physical layer cell identity, specified as a nonnegative integer.

Data Types: `double`

`CellRefP` — Number of cell-specific reference signal antenna ports

1 | 2 | 4

Number of cell-specific reference signal antenna ports, specified as 1, 2, or 4.

Data Types: `double`

`NSubframe` — Subframe number

nonnegative integer

Subframe number, specified as a nonnegative integer.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

Data Types: `struct`

`hires` — PHICH resources

numeric matrix

PHICH resources, specified as a numeric matrix of size *R*-by-2. This matrix configures up to *NPHICH* individual PHICH that are decoded. Each row of hires defines a single PHICH in terms of `[nGroup,nSeq]`, where `nGroup` is the PHICH group index number and `nSeq` is the sequence index number. These indices are 0-based.

Data Types: `double`
Complex Number Support: Yes

`sym` — Complex modulated PHICH symbols

numeric matrix

Complex modulated PHICH symbols, specified as a numeric matrix of size `NRE`-by-`NRxAnts`. `NRE` is the number of BPSK symbols per antenna assigned to the PHICH and `NRxAnts` is the number of receive antennas. The matrix should always contain `info.NRE` symbols corresponding to the total PHICH resource allocation, even if less than the full set of NPHICH channels were configured.

Data Types: `double`
Complex Number Support: Yes

`hest` — Channel estimate

3-D numeric array

Channel estimate, specified as a 3-D numeric array of size `NRE`-by-`NRxAnts`-by-`CellRefP`, where `NRE` are the frequency and time locations corresponding to the PHICH RE positions (a total of `NRE` positions), `NRxAnts` is the number of receive antennas, and `CellRefP` is the number of cell-specific reference signal antennas, given by `enb.CellRefP`.

Data Types: `double`
Complex Number Support: Yes

`noiseest` — Noise estimate

numeric scalar

Noise estimate, specified as a numeric scalar. This input argument is an estimate of the noise power spectral density per RE on received subframe. Such an estimate is provided by the `lteDLChannelEstimate` function.

Data Types: `double`

`alg` — Weighting algorithm

structure

Weighting algorithm, specified as a structure. This input argument controls weighting output soft bits, `bits`, with CSI. alg contains the following fields.

`CSI` — CSI weighting flag

`'On'` (default) | Optional | `'Off'`

CSI weighting flag, specified as `'On'` or `'Off'`. Optional. This field determines if soft bits should be weighted by CSI calculated during the equalization stage.

Data Types: `char`

Data Types: `struct`

Output Arguments

`hi` — Soft hybrid ARQ indicator values

numeric column vector

Soft hybrid ARQ indicator values, returned as a numeric column vector.

Data Types: `double`

`symbols` — Received constellation of complex symbols

numeric matrix

Received constellation of complex symbols, returned as a numeric matrix.

Data Types: `double`
Complex Number Support: Yes

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePHICH` | `ltePHICHDeprecode` | `ltePHICHIndices` | `ltePHICHInfo` | `ltePHICHPRBS` | `ltePHICHTransmitDiversityDecode`

ltePHICHDeprecode

PHICH deprecoding

Syntax

```
out = ltePHICHDeprecode(in,cp,ngroup)
out = ltePHICHDeprecode(enb,ngroup,in)
```

Description

`out = ltePHICHDeprecode(in,cp,ngroup)` performs deprecoding of the *N*-by-*P* matrix of antennas, in, onto *NU*=*P* layers, given cyclic prefix length, cp, which can be either `'Normal'` or `'Extended'`, and PHICH group, ngroup. *N* is the number of symbols per antenna. It performs PHICH deprecoding using matrix pseudoinversion to undo the processing described in section 6.9.2 of [1]. This function returns out, an *M*-by-*NU* matrix, where *NU* is the number of transmission layers and *M* is the number of symbols per layer.

`out = ltePHICHDeprecode(enb,ngroup,in)` performs deprecoding of the *N*-by-*P* matrix of antennas, in, onto *NU*=*P* layers, for PHICH group, ngroup, using the cell-wide settings structure, enb.

Examples

Deprecode PHICH Symbols

Precode and deprecode a set of physical HARQ indicator channel (PHICH) symbols for RMC R.11.

Precode an arbitrary set of input symbols, specified as a matrix of size *M*-by-`nLayers`, for a reference measurement channel (RMC) R.11, PHICH group 1.

```
enb = lteRMCDL('R.11');
nLayers = enb.PDSCH.NLayers;
symbols = complex(randi([0,1],40,nLayers),randi([0,1],40,nLayers));
precodedSym = ltePHICHPrecode(symbols,enb.CyclicPrefix,1);
```

Then, deprecode the precoded symbols, using the cell-wide settings structure, `enb`, and PHICH group 1.

```
out = ltePHICHDeprecode(precodedSym,enb.CyclicPrefix,1);
isequal(symbols,out)
```

```
     1
```

Input Arguments

`in` — Precoded input symbols

complex-valued numeric matrix

Precoded input symbols, specified as a complex-valued numeric matrix of antennas. It has size N-by-P, where N is the number of symbols per antenna and P is the number of antennas. The number of input symbols, N, must be a multiple of the number of antennas, P.

Data Types: `double`
Complex Number Support: Yes

`cp` — Cyclic prefix length

`'Normal' | 'Extended'`

Cyclic prefix length, specified as a string.

Data Types: `char`

`ngroup` — PHICH group number

positive scalar integer (≥1)

PHICH group number, specified as a positive scalar integer of 1 or more.

Data Types: `double`

`enb` — Cell-wide settings

scalar structure

Cell-wide settings, specified as a scalar structure. enb contains the following fields.

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

Data Types: `struct`

Output Arguments

`out` — Deprecoded output symbols

numeric matrix

Deprecoded output symbols, returned as a numeric matrix. It has size *M*-by-*NU*, where *M* is the number of symbols per layer and *NU* is the number of transmission layers.

Data Types: `double`
Complex Number Support: Yes

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteLayerDemap` | `ltePHICHDecode` | `ltePHICHIndices` | `ltePHICHInfo` | `ltePHICHPRBS` | `ltePHICHPrecode` | `ltePHICHTransmitDiversityDecode`

ltePHICHIndices

PHICH resource element indices

Syntax

```
ind = ltePHICHIndices(enb)
ind = ltePHICHIndices(enb,opts)
```

Description

`ind = ltePHICHIndices(enb)` returns the subframe resource element (RE) indices, ind, for the Physical Hybrid-ARQ Indicator Channels (PHICH), given the parameter fields of cell-wide settings structure, enb. By default, the number of rows of ind is the number of resource elements (NRE), and ind is an NRE-by-CellRefP matrix of indices in a 1-based linear indexing style. These indices can directly index elements of an *N*-by-*M*-by-CellRefP array that represents the subframe resource grid across CellRefP antenna ports. Each column of ind identifies the same set of NRE resource elements, but with indices offset to select them in a different antenna "page" of the 3-D resource array.

The indices returned are for all PHICH groups in a subframe, where the number of groups depends on the bandwidth and PHICH Ng parameter. See `ltePHICHInfo` for details. The indices are ordered as the modulation symbols should be mapped for the set of consecutive PHICH groups. The PHICH resources are normally all assigned in the first OFDM symbol of a subframe, unless the PHICH duration is of the extended type.

`ind = ltePHICHIndices(enb,opts)` allows control of the format of the returned indices through a cell array of option strings, opts.

Examples

Generate PHICH Indices

Generate PHICH resource element (RE) indices in linear form and resource element group (REG) indices in subscript form.

Get 1-based PHICH resource element (RE) indices in linear form.

```
enb = lteRMCDL('R.14');
enb.NDLRB = 6;
ind = ltePHICHIndices(enb,{'ind','re'})
```

```
     8        1016        2024        3032
     9        1017        2025        3033
    11        1019        2027        3035
    12        1020        2028        3036
    26        1034        2042        3050
    27        1035        2043        3051
    29        1037        2045        3053
    30        1038        2046        3054
    50        1058        2066        3074
    51        1059        2067        3075
    53        1061        2069        3077
    54        1062        2070        3078
```

Get 0-based PHICH resource element group (REG) indices in subscript form, where each column of `ind` corresponds to a dimension of the 3-D resource grid array—subcarrier, OFDM symbol, and antenna port.

```
ind = ltePHICHIndices(enb,{'0based','sub','reg'})
```

```
     6           0           0
    24           0           0
    48           0           0
     6           0           1
    24           0           1
    48           0           1
     6           0           2
    24           0           2
    48           0           2
     6           0           3
    24           0           3
    48           0           3
```

Input Arguments

`enb` — Cell-wide settings

scalar structure

Cell-wide settings, specified as a scalar structure. enb can contain the following fields. The TDDConfig and NSubframe parameter fields are only required if DuplexMode is set to `'TDD'`.

`NDLRB` — Number of downlink resource blocks
positive integer

Number of downlink resource blocks, specified as a positive integer.

Data Types: `double`

`NCellID` — Physical layer cell identity
nonnegative integer

Physical layer cell identity, specified as a nonnegative integer.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`CellRefP` — Number of cell-specific reference signal antenna ports
1 | 2 | 4

Number of cell-specific reference signal antenna ports, specified as 1, 2, or 4.

Data Types: `double`

`Ng` — HICH group multiplier
`'Sixth'` | `'Half'` | `'One'` | `'Two'`

HICH group multiplier, specified as a string.

Data Types: `char`

`PHICHDuration` — PHICH duration
`'Normal'` (default) | Optional | `'Extended'`

PHICH duration, specified as a string. Optional.

Data Types: `char`

`DuplexMode` — Duplex mode

`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, specified as a string. Optional.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration

0 (default) | Optional | 1 | 2 | 3 | 4 | 5 | 6

Uplink or downlink configuration, specified as a nonnegative scalar integer between 0 and 6. Optional. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

`NSubframe` — Subframe number

nonnegative integer

Subframe number, specified as a nonnegative integer. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

Data Types: `struct`

`opts` — Format options of returned indices

string | cell array of strings

Format options of returned indices, specified as a string or a cell array of strings. opts can contain the following option strings.

Example: `{'0based','ind'}`

Example: `'0based'`

`Indexing unit` — Indexing unit of returned indices

`'re'` (default) | `'reg'`

Indexing unit of returned indices, specified as `'re'` or `'reg'`. If `'re'`, the returned index values correspond to resource elements (REs). If `'reg'`, the returned index values correspond to resource element groups (REGs).

Data Types: `char`

`Indexing style` — Indexing style of returned indices

`'ind'` (default) | `'sub'`

Indexing style of returned indices, specified as `'ind'` or `'sub'`. If `'ind'`, the form of the returned indices is linear index form. If `'sub'`, the form of the returned indices is `[subcarrier, symbol, antenna]` subscript row form.

Data Types: `char`

`Index base` — Index base of returned indices

`'1based'` (default) | `'0based'`

Index base of returned indices, specified as `'1based'` or `'0based'`. If `'1based'`, the minimum index value is 1. If `'0based'`, the minimum index value is 0.

Data Types: `char`

Data Types: `char | cell`

Output Arguments

`ind` — PHICH resource element indices

numeric matrix

PHICH resource element indices, returned as a numeric matrix. The size of the matrix is NRE-by-CellRefP. By default, it contains 1-based linear indexing RE indices.

Data Types: `uint32`

See Also

`ltePHICH | ltePHICHDecode | ltePHICHDeprecode | ltePHICHInfo | ltePHICHPRBS | ltePHICHPrecode`

ltePHICHInfo

PHICH resource information

Syntax

```
info = ltePHICHInfo(enb)
```

Description

`info = ltePHICHInfo(enb)` returns a structure info containing information about the physical hybrid ARQ indicator channel (PHICH) subframe resources.

Examples

Get PHICH Resource Information

Get PHICH resource information from a manually constructed cell-wide settings structure.

Show that in a system subframe, normal cyclic prefix (CP) with enb`.NDLRB` set to 50 and enb`.Ng` set to `'Sixth'`, 16 PHICH are available, split between two PHICH groups of 8 sequences.

```
enb.NDLRB = 50;
enb.Ng = 'Sixth';
info = ltePHICHInfo(enb)
```

```
info =
             NREG: 6
              NRE: 24
           NPHICH: 16
          NGroups: 2
    NMappingUnits: 2
       NSequences: 8
```

```
    PHICHDuration: 1
```

Get PHICH Resource Information from RMC

Get PHICH resource information for downlink reference measurement channel (RMC) R.14.

Show that for RMC R.14, 16 PHICH are available, split between two PHICH groups of 8 sequences.

```
enb = lteRMCDL('R.14');
info = ltePHICHInfo(enb)
```

```
info =
             NREG: 6
              NRE: 24
           NPHICH: 16
          NGroups: 2
    NMappingUnits: 2
       NSequences: 8
    PHICHDuration: 1
```

Input Arguments

`enb` — eNodeB cell-wide settings

scalar structure

eNodeB cell-wide settings, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`Ng`	Required	`'Sixth'`, `'Half'`, `'One'`, `'Two'`	HICH group multiplier
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length

Parameter Field	Required or Optional	Values	Description
DuplexMode	Optional	'FDD' (default), 'TDD'	Duplex mode
The following parameters are dependent upon the condition that DuplexMode is set to 'TDD'.			
TDDConfig	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
NSubframe	Optional	Nonnegative scalar integer	Subframe number

Output Arguments

info — PHICH subframe resource information

scalar structure

PHICH subframe resource information, returned as a scalar structure. info contains the following fields.

Parameter Field	Description	Values	Data Type
NRE	Number of resource elements (REs) assigned to all PHICH	Nonnegative scalar integer	uint64
NREG	Number of resource element groups assigned to all PHICH	Nonnegative scalar integer	uint64
NPHICH	Number of individual PHICH available	Nonnegative scalar integer	uint64
NGroups	Number of PHICH groups	Nonnegative scalar integer	int8
NMappingUnits	Number of PHICH mapping units	Nonnegative scalar integer	int8
NSequences	Number of orthogonal sequences in each PHICH group	Nonnegative scalar integer	int8
PHICHDuration	PHICH duration	Nonnegative scalar integer	int8

The control region of a subframe can contain up to `NPHICH` separate PHICHs with each carrying a single hybrid ARQ ACK or NACK. Multiple PHICHs can be mapped to the same set of resource elements through PHICH groups where each PHICH in a group is carried on one of `NSequences` orthogonal sequences. For the purpose of mapping to resources, the groups are combined into mapping units where each unit spans 3 resource element groups. Thus, `NREG` is 3×`NMappingUnits` and `NRE` is 4×3×`NMappingUnits`. The `Ng` parameter controls the number of groups available for a given bandwidth.

See Also

`ltePHICH` | `ltePHICHDecode` | `ltePHICHDeprecode` | `ltePHICHIndices` | `ltePHICHPRBS` | `ltePHICHPrecode` | `ltePHICHTransmitDiversityDecode`

ltePHICHPRBS

PHICH pseudorandom scrambling sequence

Syntax

```
seq = ltePHICHPRBS(enb,n)
seq = ltePHICHPRBS(enb,n,mapping)
```

Description

`seq = ltePHICHPRBS(enb,n)` returns a column vector containing the first n outputs of the Physical Hybrid ARQ Indicator Channel (PHICH) scrambling sequence when initialized according to cell-wide settings structure, enb.

`seq = ltePHICHPRBS(enb,n,mapping)` allows control over the format of the returned sequence, seq, through the mapping string. Valid formats are `'binary'`, which is the default, and `'signed'`. `'binary'` maps true to 1 and false to 0. `'signed'` maps true to –1 and false to 1.

Examples

Generate PHICH Pseudorandom Scrambling Sequence

Generate the pseudorandom scrambling sequence for the PHICH. Using RMC R.0 results in 12 BPSK modulated symbols, where 1 bit per symbol is mapped onto a single resource element (RE).

```
enb = lteRMCDL('R.0');
phichInfo = ltePHICHInfo(enb);
phichPrbsSeq = ltePHICHPRBS(enb,phichInfo.NRE);
```

Input Arguments

enb — Cell-wide settings

scalar structure

Cell-wide settings, specified as a scalar structure. enb contains the following fields.

`NCellID` — Physical layer cell identity
nonnegative integer

Physical layer cell identity, specified as a nonnegative integer.

Data Types: `double`

`NSubframe` — Subframe number
nonnegative integer

Subframe number, specified as a nonnegative integer.

Data Types: `double`

Data Types: `struct`

`n` — Length of PHICH scrambling sequence
positive scalar integer

Length of PHICH scrambling sequence, specified as a positive scalar integer of 1 or more.

Data Types: `double`

`mapping` — Format of returned sequence
`'binary'` (default) | `'signed'`

Format of returned sequence, specified as a string. This argument controls the format of the returned sequence, seq. The string `'binary'` maps true to 1 and false to 0. The string `'signed'` maps true to –1 and false to 1.

Data Types: `char`

Output Arguments

`seq` — PHICH pseudorandom scrambling sequence
logical column vector | numeric column vector

PHICH pseudorandom scrambling sequence, returned as a logical column vector or a numeric column vector. This argument contains the first n outputs of the PHICH scrambling sequence.

Data Types: `logical` | `double`

See Also

`ltePHICH` | `ltePHICHDecode` | `ltePHICHDeprecode` | `ltePHICHIndices` | `ltePHICHInfo` | `ltePHICHPrecode` | `ltePHICHTransmitDiversityDecode`

ltePHICHPrecode

PHICH precoding

Syntax

```
out = ltePHICHPrecode(in,cp,ngroup)
out = ltePHICHPrecode(enb,ngroup,in)
```

Description

`out = ltePHICHPrecode(in,cp,ngroup)` performs precoding of the *N*-by-*NU* matrix of layers, in, onto *P*=*NU* antennas, given cyclic prefix length, cp, which can be `'Normal'` or `'Extended'`, and PHICH group, ngroup. It performs PHICH precoding according to section 6.9.2 of [1]. This function returns an *M*-by-*P* matrix, where *P* is the number of transmission antennas and *M* is the number of symbols per antenna.

`out = ltePHICHPrecode(enb,ngroup,in)` performs precoding of the *N*-by-*NU* matrix of layers, in, onto *P*=*NU* antennas for PHICH group, ngroup, using the cell-wide settings structure, enb.

Examples

Precode PHICH Symbols

Precode an arbitrary set of PHICH symbols, specified as a matrix of size *M*-by-`nLayers`, for reference measurement channel (RMC) R.11, PHICH group 1.

```
enb = lteRMCDL('R.11');
nLayers = enb.PDSCH.NLayers;
phichSym = complex(randi([0,1],40,nLayers),randi([0,1],40,nLayers));
out = ltePHICHPrecode(phichSym,enb.CyclicPrefix,1);
```

Input Arguments

`in` — PHICH input symbols

complex-valued numeric matrix

PHICH input symbols, specified as a complex-valued numeric matrix. in is a matrix of *N*-by-*NU* layers.

Data Types: `double`
Complex Number Support: Yes

`cp` — Cyclic prefix length

`'Normal' | 'Extended'`

Cyclic prefix length, specified as a string.

Data Types: `char`

`ngroup` — PHICH group

positive scalar integer

PHICH group number, specified as a positive scalar integer of 1 or more.

Data Types: `double`

`enb` — Cell-wide settings

scalar structure

Cell-wide settings, specified as a scalar structure. enb can contain the following field.

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

Data Types: `struct`

Output Arguments

`out` — Precoded output

numeric matrix

Precoded output, returned as a numeric matrix of size *M*-by-*P*, where *P* is the number of transmission antennas and *M* is the number of symbols per antenna.

Data Types: `double`

Complex Number Support: Yes

References

[1] 3GPP TS 36.211. “Physical Channels and Modulation.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteLayerMap` | `ltePHICH` | `ltePHICHDeprecode` | `ltePHICHIndices` | `ltePHICHInfo` | `ltePHICHPRBS`

ltePHICHTransmitDiversityDecode

PHICH OSFBC decoding

Syntax

```
[out,CSI]=ltePHICHTransmitDiversityDecode(in,cp,ngroup,hest)
```

Description

`[out,CSI]=ltePHICHTransmitDiversityDecode(in,cp,ngroup,hest)` returns Orthogonal Space Frequency Block Code (OSFBC) decoded symbols out and channel state information CSI given received PHICH symbols in along with cyclic prefix length cp (`'Normal'`,`'Extended'`), PHICH resource group number ngroup and channel estimate hest.

out is of size *M*-by-1, where *M* is the number of received symbols for each receive antenna. CSI is of size *M*-by-1, the same as out, containing soft channel state information; the CSI provides an estimate of the received RE gain for each received RE. in is an *M*-by-`NRxAnts` matrix of received symbols, where *M* is the number of received symbols for each of `NRxAnts` receive antennas.

hest is a 3-D *M*-by-`NRxAnts`-by-`NTxAnts` array, where *M* is the number of received symbols in in, `NRxAnts` is the number of receive antennas, and `NTxAnts` is the number of transmit antennas.

Examples

Deprecode PHICH Symbols

This example generates the PHICH symbols for multiple antennas using RMC R.11.

Initialize cell-wide settings.

```
enb = lteRMCDL('R.11');
phichInfo = ltePHICHInfo(enb);
hiset = [1,1,1;1,2,0];
phichSym = ltePHICH(enb,hiset);
```

Create an ideal, or identity, channel estimate.

```
hest = permute(repmat(eye(enb.CellRefP), ...
    [1,1,phichInfo.NRE]),[3,1,2]);
```

Deprecode the received symbols, using the channel estimates.

```
ng = phichInfo.NGroups;
out = ltePHICHTransmitDiversityDecode(phichSym, ...
    enb.CyclicPrefix,ng,hest)
```

```
out =

   0.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
  -1.4142 - 1.4142i
  -1.4142 - 1.4142i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   1.4142 + 1.4142i
  -1.4142 - 1.4142i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   1.4142 + 1.4142i
  -1.4142 - 1.4142i
   0.0000 + 0.0000i
```

Input Arguments

`in` — Received PHICH symbols

numeric matrix

Received PHICH symbols, specified as a numeric matrix of size *M*-by-`NRxAnts`, where *M* is the number of received symbols for each of `NRxAnts` receive antennas.

Data Types: `double`
Complex Number Support: Yes

`cp` — Cyclic prefix length

`'Normal' | 'Extended'`

Cyclic prefix length, specified as a string.

Data Types: `char`

`ngroup` — PHICH group number

positive scalar integer (≥1)

PHICH group number, specified as a positive scalar integer of 1 or more.

Data Types: `double`
Complex Number Support: Yes

`hest` — Channel estimate

3-D numeric array

Channel estimate, specified as a 3-D numeric array of size *M*-by-`NRxAnts`-by-`NTxAnts`, where *M* is the number of received symbols in in, `NRxAnts` is the number of receive antennas, and `NTxAnts` is the number of transmit antennas.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`out` — OSFBC decoded symbols

numeric matrix

OSFBC decoded symbols, returned as a numeric matrix of size *M*-by-1, where *M* is the number of received symbols for each receive antenna.

Data Types: `double`
Complex Number Support: Yes

`CSI` — Soft channel state information

numeric matrix

Soft channel state information, returned as a numeric matrix of size *M*-by-1. It provides an estimate of the received RE gain for each received RE.

Data Types: `double`
Complex Number Support: Yes

See Also

`ltePHICH` | `ltePHICHDecode` | `ltePHICHDeprecode` | `ltePHICHIndices` | `ltePHICHInfo` | `ltePHICHPRBS`

ltePMIInfo

Precoder matrix indication reporting information

Syntax

```
info = ltePMIInfo(enb,chs)
```

Description

`info = ltePMIInfo(enb,chs)` returns a structure, info, containing information related to precoder matrix indication (PMI) reporting. This information is in terms of the subband size in resource blocks, the number of subbands for PMI reporting, the maximum permitted PMI value for the given configuration, and the size of the codebook subset restriction bitmap.

You can use `info.NSubbands` to determine the correct size of the `PMISet` vector required for closed-loop spatial multiplexing operation. `PMISet` is a column vector with `info.NSubbands` rows. For CSI reporting, `(CSIRefP =8)`, `info.NSubbands` indicates the number of second codebook indices, *i2*, in the report. The first codebook index, *i1*, is always chosen in a wideband fashion, and is therefore a scalar. `PMIMode = 'Wideband'` corresponds to PUSCH reporting Mode 1-2 or PUCCH reporting Mode 1-1 (PUCCH Report Type 2). `PMIMode = 'Subband'` corresponds to PUSCH reporting Mode 3-1.

Examples

PMI Reporting Information

Get the PMI reporting information for RMC R.13.

```
enb = lteRMCDL('R.13');
pmiInfo = ltePMIInfo(enb, enb.PDSCH)
```

```
pmiInfo =

                    k: 50
            NSubbands: 1
               MaxPMI: 15
```

```
    CodebookSubsetSize: 16
```

Input Arguments

enb — Cell-wide settings

structure

Cell-wide settings, specified as a scalar structure. The structure contains the following fields:

Parameter Field	Required or Optional	Values	Description
NDLRB	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
CellRefP	Optional	1 (default), 2, 4	Number of cell-specific reference signal (CRS) antenna ports
The following parameters apply when chs.Txscheme is set to `Port7–14`.			
CSIRefP	Optional	1 (default), 2, 4, 8	Number of CSI-RS antenna ports

Data Types: `struct`

chs — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure containing the following fields.

Parameter Field	Required or Optional	Values	Description
NLayers	Optional	1 (default), 2, 3, 4, 5, 6, 7, 8	Number of transmission layers
PMIMode	Optional	`'Wideband'` (default), `'Subband'`	PMI reporting mode

Parameter Field	Required or Optional	Values	Description
`TxScheme`	Optional	`'SpatialMux'` (default), `'Port0'`, `'TxDiversity'`, `'CDD'`, `'MultiUser'`, `'Port5'`, `'Port7-8'`, `'Port8'`, `'Port7-14'`	Transmission scheme, specified as one of the following options. • `'SpatialMux'` — Closed-loop spatial multiplexing. • `'Port0'` — Single-antenna port, port 0. • `'TxDiversity'` — Transmit diversity scheme. • `'CDD'` — Large delay CDD scheme. • `'MultiUser'` — Multiuser MIMO scheme. • `'Port5'` — Single-antenna port, port 5. • `'Port7-8'` — Single-antenna port, port 7 (**`NLayers`** = 1). Dual layer transmission, ports 7 and 8 (**`NLayers`** = 2). • `'Port8'` — Single-antenna port, port 8. • `'Port7-14'` — Up to 8-layer transmission, ports 7–14.

Data Types: `struct`

Output Arguments

`info` — Information related to PMI reporting

structure

Information related to PMI reporting, returned as a structure containing the following fields:

Parameter Field	Description	Values
k	Subband size, in resource blocks (equal to `NRB` for wideband PMI reporting or transmission schemes without PMI reporting).	numeric scalar
NSubbands	Number of subbands for PMI reporting (equal to 1 for wideband PMI reporting) or transmission schemes without PMI reporting.	numeric scalar
MaxPMI	Maximum permitted PMI value for the given configuration. Valid PMI values range from 0 to `MaxPMI`. For CSI reporting, when `CSIRefP = 8`, `MaxPMI` is a 2–element vector, indicating the maximum permissible values of *i1* and *i2*, the first and second codebook indices. For transmission schemes without PMI reporting, `MaxPMI = 0`.	nonnegative numeric scalar
CodeBookSub	Size of the codebook subset restriction bitmap. For transmission schemes without PMI reporting, `CodebookSubsetSize=0`.	scalar

See Also

`lteCSICodebook` | `lteDLPrecode` | `ltePDSCH` | `ltePDSCHDecode` | `ltePMISelect`

ltePMISelect

PDSCH precoder matrix indicator calculation

Syntax

```
[pmiset,info,sinrs,subbandsinrs] = ltePMISelect(enb,chs,hest,
noiseest)
```

Description

`[pmiset,info,sinrs,subbandsinrs] = ltePMISelect(enb,chs,hest, noiseest)` performs PDSCH precoder matrix indication (PMI) set calculation for the given cell-wide settings, `enb`, channel configuration structure, `chs`, channel estimate resource array, `hest`, and receiver noise variance, `noiseest`.

The PMI selection is performed using the codebooks specified in [2], section 7.2.4. For the `'Port7-14'` transmission scheme, the CSI reporting codebook is used when CSI-RS ports are equal to 8. For other numbers of CSI-RS ports in the `'Port7-14'` transmission scheme and for other transmission schemes, the PMI selection is performed using the codebook for closed-loop spatial multiplexing. This term is defined in [1], Tables 6.3.4.2.3-1 and 6.3.4.2.3-2.

`PMIMode= 'Wideband'` corresponds to PUSCH reporting Mode 1-2 or PUCCH reporting Mode 1-1 (PUCCH Report Type 2). `PMIMode= 'Subband'` corresponds to PUSCH reporting Mode 3-1. PMI selection is based on the rank indicated by `chs.NLayers`, except for the `'TxDiversity'` transmission scheme, where the rank is 1. In PUCCH reporting Mode 1-1, you can achieve codebook subsampling for submode 2 ([2], Table 7.2.2-1D) with an appropriate `chs.CodebookSubset`.

Examples

Calculate PMI

Populate an empty resource grid for RMC R.13 with cell-specific reference signals symbols.

```
enb = lteRMCDL('R.13');
```

```
enb.PDSCH.PMIMode = 'Subband';
reGrid = lteResourceGrid(enb);
reGrid(lteCellRSIndices(enb)) = lteCellRS(enb);
```

Modulate the signal, filter it through the fading channel, and demodulate it.

```
txWaveform = lteOFDMModulate(enb,reGrid);
chcfg.SamplingRate = 15360000;
chcfg.DelayProfile = 'EPA';
chcfg.NRxAnts = 4;
chcfg.DopplerFreq = 5;
chcfg.MIMOCorrelation = 'Low';
chcfg.InitTime = 0;
chcfg.Seed = 1;
rxWaveform = lteFadingChannel(chcfg,txWaveform);
rxSubframe = lteOFDMDemodulate(enb,rxWaveform);
```

```
Warning: Using default value for parameter field InitPhase (Random)
Warning: Using default value for parameter field ModelType (GMEDS)
Warning: Using default value for parameter field NTerms (16)
Warning: Using default value for parameter field NormalizeTxAnts (On)
Warning: Using default value for parameter field NormalizePathGains (On)
```

Estimate the corresponding channel and the noise power spectral density on the reference signal subcarriers.

```
cec.FreqWindow = 1;
cec.TimeWindow = 31;
cec.InterpType = 'cubic';
cec.PilotAverage = 'UserDefined';
cec.InterpWinSize = 1;
cec.InterpWindow = 'Centered';
[hest, noiseEst] = lteDLChannelEstimate(enb,cec,rxSubframe);
```

Calculate PMI.

```
pmi = ltePMISelect(enb,enb.PDSCH,hest,noiseEst)
enb.PDSCH.PMISet = pmi; % configure transmitter PMI
txWaveform = lteRMCDLTool(enb,[1;0;0;1]);
```

```
Warning: Using default value for parameter field CodebookSubset
(1111111111111111)

pmi =

     1
```

```
 1
 6
 2
12
12
12
12
12
```

Input Arguments

enb — Cell-wide settings

structure

Cell-wide settings, specified as a structure containing the following fields:

Parameter Field	Required or Optional	Values	Description
NDLRB	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
NCellID	Required	Nonnegative scalar integer (0, …, 503)	Physical layer cell identity
CellRefP	Required	1 (default), 2, 4, 8	Number of cell-specific reference signal (CRS) antenna ports
CyclicPrefix	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length
DuplexMode	Optional	`'FDD'` (default), `'TDD'`	Duplex mode
The following parameters apply when `DuplexMode` is set to `'TDD'`.			
TDDConfig	Optional	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
SSC	Optional	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)
NSubframe	Required	Nonnegative scalar integer	Subframe number

Parameter Field	Required or Optional	Values	Description
The following parameters apply when `chs.TxScheme` is set to `'Port7–14'` transmission scheme.			
`CSIRefP`	Required	1, 2, 4	Number of CSI-RS antenna ports
`CSIRSConf`	Required	scalar integer	CSI-RS configuration index. See table 6.10.5.2-1 in TS 36.211.
`CSIRSPeri`	Optional	`'On'` (default), `'Off'`, `Icsi-rs` (0, ..., 154), `[Tcsi-rs Dcsi-rs]`	CSI-RS subframe configuration
`Nframe`	Optional	0 (default), Nonnegative scalar integer	Frame number

Data Types: `struct`

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure containing the following fields:

Parameter Field	Required or Optional	Values	Description
`NLayers`	Required	1, 2, 3, 4, 5, 6, 7, 8	Number of transmission layers
`PMIMode`	Optional	`'Wideband'` (default), `'Subband'`	PMI reporting mode
`TxScheme`	Optional	`'SpatialMux'` (default), `'Port0'`, `'TxDiversity'`, `'CDD'`, `'MultiUser'`, `'Port5'`, `'Port7-8'`, `'Port8'`, `'Port7-14'`	Transmission scheme, specified as one of the following options. • `'SpatialMux'` — Closed-loop spatial multiplexing. • `'Port0'` — Single-antenna port, port 0. • `'TxDiversity'` — Transmit diversity scheme. • `'CDD'` — Large delay CDD scheme.

Parameter Field	Required or Optional	Values	Description
			• `'MultiUser'` — Multiuser MIMO scheme. • `'Port5'` — Single-antenna port, port 5. • `'Port7-8'` — Single-antenna port, port 7 (**`NLayers`** = 1). Dual layer transmission, ports 7 and 8 (**`NLayers`** = 2). • `'Port8'` — Single-antenna port, port 8. • `'Port7-14'` — Up to 8-layer transmission, ports 7–14.
`CodebookSubs`	Optional	String or vector, all ones (default)	Codebook subset restriction, specified as a string bitmap. The default values are all ones, permitting all PMI values. This parameter is configured by higher layers and indicates the values of PMI that can be reported. The bitmap, defined in TS36.213, Section 7.2, is arranged a_A-1,a_A-2,...a_0. For example, the element CodebookSubset(1) corresponds to a_A-1 and the element CodebookSubset(end) corresponds to a_0. The length of the bitmap is given by the info.CodebookSubsetSize field returned by `ltePMIInfo`. You can also specify the bitmap in a hexadecimal form by prefixing the string with `'0x'`. Alternatively, you can specify a numeric array identical to the PMISET output, indicating to restrict the selection to only those PMISET values. Specifying the parameter in this way enables you to obtain SINR estimates against an existing reported PMI for RI and CQI selection. If this parameter field is defined but is empty, no codebook subset restriction is applied. `(codebookSubsetRestriction)`

Data Types: `struct`

`hest` — Channel estimate
multidimensional array

Channel estimate, specified as a multidimensional array of size *K*-by-*L*-by-*NRxAnts*-by-*P* where:

- *K* is the number of subcarriers.
- *L* is the number of OFDM symbols.
- *NRxAnts* is the number of received antennas.
- *P* is the number of planes.

Data Types: `double`
Complex Number Support: Yes

`noiseest` — Receiver noise variance

numeric scalar

Receiver noise variance, specified as a numeric scalar. This input argument specifies an estimate of the received noise power spectral density.

Data Types: `double`

Output Arguments

`pmiset` — PMI set selected

column vector | integer

Precoder matrix indications (PMI) set selected, returned as a column vector or an integer. For the `'Port7-14'` transmission scheme with 8 CSI-RS ports, `pmiset` has `info.NSubbands + 1` rows. The first row indicates wideband codebook index *i1*. The subsequent `info.NSubbands` rows indicate the subband codebook indices *i2* or if `info.NSubbands=1`, the wideband codebook index *i2*. For other numbers of CSI-RS ports in the `'Port7-14'` transmission scheme, and for other transmission schemes, `pmiset` has `'info.NSubbands'` rows. Each row gives the subband codebook index for that subband. For wideband reporting (`info.NSubbands = 1`), `pmiset` is a scalar specifying the selected wideband codebook index.

`info` — Information related to PMI reporting

structure

Information related to PMI reporting, returned as a scalar structure. `info` contains the following fields:

Parameter Field	Description	Values
k	Subband size, in resource blocks (equal to NRB for wideband PMI reporting or transmission schemes without PMI reporting).	numeric scalar
NSubbands	Number of subbands for PMI reporting (equal to 1 for wideband PMI reporting) or transmission schemes without PMI reporting.	numeric scalar
MaxPMI	Maximum permitted PMI value for the given configuration. Valid PMI values range from 0 to MaxPMI. For CSI reporting, when CSIRefP = 8, MaxPMI is a 2–element vector, indicating the maximum permissible values of *i1* and *i2*, the first and second codebook indices. For transmission schemes without PMI reporting, MaxPMI = 0.	nonnegative numeric scalar
CodeBookSubs	Size of the codebook subset restriction bitmap. For transmission schemes without PMI reporting, CodebookSubsetSize=0.	scalar

sinrs — Signal-to-interference plus noise ratios
multidimensional array

Signal to interference plus noise ratios, returned as a multidimensional array of size *K*-by-*L*-by-*N1*-by-*N2*, where:

- *K* is the number of subcarriers.
- *L* is the number of OFDM symbols.

For the 'Port7-14' transmission scheme with 8 CSI-RS ports:

- *N1* is info.MaxPMI (1)+1
- *N2* is info.MaxPMI (2)+1

i.e the number of possible first and second codebook indices.

For other numbers of CSI-RS ports in the 'Port7-14' transmission scheme, and for other transmission schemes:

- *N1* is 1.
- *N2* is info.MaxPMI +1.

The array contains non-`NaN` values in the time and frequency locations (first two dimensions) of the reference signal REs. This array is used for PMI estimation for all possible codebook indices (last two dimensions). These values are the calculated sinrs in the reference signal RE locations for each codebook index combination. You can obtain the values using a linear MMSE SINR metric. All locations not corresponding to a reference signal RE are set to `NaN`.

`subbandsinrs` — Subband signal-to-interference plus noise ratios
multidimensional array

Subband signal-to-interference plus noise ratios (`sinrs`), returned as a multidimensional array. The size of the array is *'INFO.NSubbands'*-by-*N1*-by-*N2*-by-*chs.NLayers*. This array indicates the average linear SINR in the subband specified for each possible PMI value (*N1* and *N2* dimensions) and each layer. The `sinrs` output is formed by summing a 5–dimensional *K*-by-*L*-by-*N1*-by-*N2*-by-*chs.NLayers* estimate of the `sinrs` across all of the layers. `subbandsinrs` is formed by averaging that same 5-dimensional estimate across each subband that is in the appropriate region of the K dimension and across the L dimension.

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.213. "Physical layer procedures." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteCQISelect` | `lteCSICodebook` | `lteDLPrecode` | `ltePDSCH` | `ltePDSCHDecode` | `ltePMIInfo` | `lteRISelect`

ltePRACH

Physical random access channel

Syntax

```
[waveform,info]=ltePRACH(ue,chs)
```

Description

`[waveform,info]=ltePRACH(ue,chs)` returns a column vector, waveform, containing complex symbols of the Physical Random Access Channel given UE-specific settings structure, ue, and channel transmission configuration structure, chs. PRACH information is returned in a structure, info, as described in `ltePRACHInfo`. waveform is *N*-by-1, where *N*=info.SamplingRate×info.TotSubframes×0.001, and contains the time-domain PRACH signal spanning info.TotSubframes, as described in section 5.7 of [2]. The waveform consists of a period of zeros (for the case of a time offset or Preamble Format 4), a cyclic prefix, the "useful" part of the PRACH signal, and a period of zeros to extend the waveform to span info.TotSubframes. The duration of the PRACH is a function of the Preamble Format as described in table 5.7.1-1 of [2]. Depending on the configuration given in ue and chs, it is possible that no PRACH are generated; in this case info.PRBSet is empty to signal this condition, and waveform consists of all zeros. The conditions under which no PRACH are generated are described in the help for `ltePRACHInfo`.

Although the parameters Format and ConfigIdx are both described as optional, at least one of these parameters must be specified.

The parameter TimingOffset is not a genuine parameter of the PRACH generation as defined in the standard. It is provided to allow easy generation of a delayed PRACH output for use in testing, to simulate the effect of the distance between UE and eNodeB. The maximum value of TimingOffset that yields a complete PRACH transmission in the output waveform is a timing offset equal to the duration of the last field of info.Fields; this timing offset corresponds to the maximum cell size and hence maximum distance between UE and eNodeB. If this maximum timing offset is exceeded, part of the PRACH signal is lost. The end of the useful part of the PRACH signal is out with the span of waveform.

Note that in the `ltePRACHInfo` and `ltePRACHDetect` functions, chs.PreambleIdx can be a vector. This assists with modelling of an eNodeB receiver searching for multiple preambles. However, this function only generates a single PRACH and therefore chs.PreambleIdx should be a scalar. If chs.PreambleIdx is a vector, the first element is used.

For convenience the output, waveform, is sampled by default at the same sampling rate as for any uplink channel (PUCCH, PUSCH and SRS) using the `lteSCFDMAModulate` modulator, for the given value of ue.NULRB.

If the value of chs.PreambleIdx is such that an insufficient quantity of cyclic shifts are available at the configured logical root index, chs.SeqIdx, the logical root index number needs to be incremented. As such, the physical root used, info.RootSeq, differs from the physical root configured by chs.SeqIdx. The cyclic shift corresponding to chs.PreambleIdx can be found in info.CyclicShift. For High Speed mode, if the value of info.CyclicShift is –1, used to indicate that there are no cyclic shifts in the restricted set, then the PRACH waveform is generated with no cyclic shift.

Examples

Generate PRACH Symbols

This example generates PRACH symbols of format 0 in an ue.NULRB=9 bandwidth, leaving all other parameters at their default values.

Initialize ue-specific settings and channel transmission configuration.

```
ue.DuplexMode = 'FDD';
ue.NULRB = 6;
chs.Format = 0;
chs.HighSpeed = 0;
chs.CyclicShiftIdx = 0;
chs.FreqOffset = 0;
chs.SeqIdx = 0;
chs.PreambleIdx = [ 0 ];
```

Generate PRACH symbols and PRACH info.

```
[prachSym,prachInfo] = ltePRACH(ue,chs);
prachInfo
```

```
prachInfo =

                  NZC: 839
    SubcarrierSpacing: 1250
                  Phi: 7
                    K: 12
         TotSubframes: 1
               Fields: [0 3168 24576 2976]
               PRBSet: [6x1 uint32]
                  NCS: 0
          CyclicShift: 0
              RootSeq: 129
         SamplingRate: 1920000
           BaseOffset: 0
```

Input Arguments

ue — UE-specific settings

scalar structure

UE-specific settings, specified as a scalar structure. ue can contain the following fields. The TDDConfig and SSC parameter fields are only required if DuplexMode is set to `'TDD'`. The CyclicPrefix parameter field is only required if chs.Format is set to 4.

NULRB — Number of uplink resource blocks

positive integer

Number of uplink resource blocks, specified as a positive integer.

Data Types: `double`

NSubframe — Subframe number

0 (default) | Optional | nonnegative integer

Subframe number, specified as a nonnegative integer. Optional.

Data Types: `double`

NFrame — Frame number

0 (default) | Optional | nonnegative integer

Frame number, specified as a nonnegative integer. Optional.

Data Types: `double`

`DuplexMode` — Duplex mode

`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, specified as a string. Optional.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration

0 (default) | Optional | 1 | 2 | 3 | 4 | 5 | 6

Uplink or downlink configuration, specified as a nonnegative scalar integer between 0 and 6. Optional. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

`SSC` — Special subframe configuration

0 (default) | Optional | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Special subframe configuration, specified as a nonnegative scalar integer between 0 and 9. Optional. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional. Only required if chs`.`Format is set to 4, for preamble format 4.

Data Types: `char`

Data Types: `struct`

`chs` — Channel transmission configuration

scalar structure

Channel transmission configuration, specified as a scalar structure. chs can contain the following fields. The FreqOffset parameter field is only required if ue`.`DuplexMode is set to `'TDD'`. The FreqOffset parameter field is only required if Format is set to a value between 0 and 3.

`Format` — Preamble format

determined by ConfigIdx (default) | Optional | nonnegative scalar integer (0...4)

Preamble format, specified as a nonnegative scalar integer between 0 and 4. Optional. However, the `Format` field must be specified if the ConfigIdx field is not specified. Default value is determined by ConfigIdx field, if present.

Data Types: `double`

`SeqIdx` — Logical root sequence index

0 (default) | Optional | nonnegative scalar integer (0...837)

Logical root sequence index, specified as a nonnegative scalar integer between 0 and 837. Optional. (*RACH_ROOT_SEQUENCE*).

Data Types: `double`

`ConfigIdx` — PRACH Configuration Index

determined by Format (default) | Optional | nonnegative scalar integer (0...63)

PRACH Configuration Index, specified as a nonnegative scalar integer between 0 and 63. Optional. However, the `ConfigIdx` field must be specified if the Format field is not specified. Default value is determined by Format field, if present. (*prach-ConfigurationIndex*)

Data Types: `double`

`PreambleIdx` — Scalar preamble index within cell

0 (default) | Optional | nonnegative scalar integer (0...63)

Scalar preamble index within cell, specified as a nonnegative scalar integer between 0 and 63. Optional. (*ra-PreambleIndex*).

Data Types: `double`

`CyclicShiftIdx` — Cyclic shift configuration index

0 (default) | Optional | nonnegative scalar integer (0...15)

Cyclic shift configuration index, specified as a nonnegative scalar integer between 0 and 15. Optional. (*zeroCorrelationZoneConfig*, yields *N_CS*)

Data Types: `double`

`HighSpeed` — High speed flag

0 (default) | Optional | 1

High speed flag, specified as a 0 or 1. Optional. A value of 1 signifies a restricted set. A value of 0 signifies an unrestricted set. (*highSpeedFlag*)

Data Types: `logical` | `double`

`TimingOffset` — PRACH timing offset

0.0 (default) | Optional | numeric scalar

PRACH timing offset, in microseconds, specified as a numeric scalar. Optional.

Data Types: `double`

`FreqIdx` — Frequency resource index

0 (default) | Optional | 0...5

Frequency resource index, specified as a nonnegative scalar integer between 0 and 5. Optional. Only required if ue`.`DuplexMode is set to `'TDD'`. (*f_RA*)

Data Types: `double`

`FreqOffset` — PRACH frequency offset

0 (default) | Optional | 0...94

PRACH frequency offset, specified as a nonnegative scalar integer between 0 and 94. Optional. Only required if Format is set to a value between 0 and 3, for preamble formats 0, 1, 2, and 3. (*n_PRBoffset*)

Data Types: `double`

Data Types: `struct`

Output Arguments

`waveform` — PRACH waveform symbols

complex-valued numeric column vector

PRACH waveform symbols, returned as a complex-valued numeric column vector. It has size *N*-by-1, where *N*=(info`.SamplingRate` × info`.TotSubframes` × 0.001). It contains the time-domain PRACH signal spanning info`.TotSubframes`.

Data Types: `double`
Complex Number Support: Yes

`info` — PRACH information

scalar structure

PRACH information, returned as a scalar structure. It contains the following fields.

`NZC` — Zadoff-Chu sequence length
positive integer

Zadoff-Chu sequence length, returned as a positive integer. (*N_ZC*)

Data Types: `double`

`SubcarrierSpacing` — Subcarrier spacing of PRACH preamble
positive integer

Subcarrier spacing of PRACH preamble, in Hz, returned as a positive integer. (*deltaf_RA*)

Data Types: `double`

`Phi` — Frequency-domain location offset
positive integer

Frequency-domain location offset, returned as a positive integer. (*phi*)

Data Types: `double`

`K` — Ratio of uplink data to PRACH subcarrier spacing
numeric scalar

Ratio of uplink data to PRACH subcarrier spacing, returned as a numeric scalar. (*K*)

Data Types: `double`

`TotSubframes` — Number of subframes duration of PRACH
numeric scalar

Number of subframes duration of the PRACH, returned as a numeric scalar. Each subframe lasts 30720 fundamental periods, therefore `TotSubframes` is `ceil(sum(Fields)/30720)`, the number of subframes required to hold the entire PRACH waveform. The duration of the PRACH is a function of the Preamble Format as described in table 5.7.1-1 of [2].

Data Types: `double`

`Fields` — PRACH field lengths
1-by-4 numeric vector

PRACH field lengths, returned as a 1-by-4 numeric vector. The elements are [*OFFSET T_CP T_SEQ GUARD*]. *T_CP* and *T_SEQ* are the lengths in fundamental time periods (*T_s*), of cyclic prefix and PRACH sequence, respectively. *OFFSET* is the number of fundamental time periods from the start of configured subframe to the start of the cyclic prefix, and is nonzero only for TDD special subframes. *GUARD* is the number of fundamental time periods from the end of the PRACH sequence to the end of the number of subframes spanned by the PRACH.

Data Types: `double`

`PRBSet` — PRBs occupied by PRACH preamble

nonnegative integer column vector

PRBs occupied by PRACH preamble, returned as a nonnegative integer column vector. (starts at *n_PRB*, 0-based).

- An empty `info.PRBSet` indicates that the PRACH is not present and the waveform generated by ltePRACH consists of all zeros.
- An `info.PRBSet` that contains six consecutive Physical Resource Block numbers indicates the frequency domain location of the PRACH.

Note: The PRACH uses a different SC-FDMA symbol construction from the other channels (PUCCH, PUSCH and SRS) and therefore the PRBSet indicates the frequency range (180kHz per RB) that the PRACH occupies, it does not occupy the set of 12 subcarriers in each RB in the same fashion as other channels. The PRACH occupies a bandwidth approximately equal to 1.08MHz = 6RBs.

Data Types: `uint32`

`NCS` — Length of zero correlation zone plus 1

positive integer

Length of zero correlation zone plus 1, specified as a positive integer (*N_CS*). `NCS` corresponds to the complete extent of autocorrelation lags (0 and *N_CS*–1 nonzero) that exhibit perfect correlation properties (1 at 0 lag, 0 at nonzero lags). `NCS` is expressed directly, as in the standard, related to the fundamental Zadoff-Chu sequence construction. The actual sample span of the zero correlation zone in the waveform generated by `ltePRACH` is a function of the sampling rate.

Data Types: `double`

`CyclicShift` — Cyclic shift or shifts of Zadoff-Chu sequence
numeric row vector

Cyclic shift or shifts of Zadoff-Chu sequence, returned as a numeric row vector. (C_v). For High Speed mode, any element of `CyclicShift` equal to –1 indicates that there are no cyclic shifts in the restricted set for the corresponding preamble index.

Data Types: `double`

`RootSeq` — Physical root Zadoff-Chu sequence index or indices
numeric row vector

Physical root Zadoff-Chu sequence index or indices, returned as a numeric row vector. (u)

Data Types: `double`

`CyclicOffset` — Cyclic shift or shifts corresponding to Doppler Shift
vector

Cyclic shift or shifts corresponding to Doppler Shift of ($1/T\_SEQ$), returned as a vector. This parameter is present for High Speed mode. (d_u)

Data Types: `double`

`SamplingRate` — Sampling rate of PRACH modulator
numeric scalar

Sampling rate of the PRACH modulator, returned as a numeric scalar.

Data Types: `double`

`BaseOffset` — Base timing offset
numeric scalar

Base timing offset, in microseconds, returned as a numeric scalar. This parameter field is used for the detection test in [1]. (duration of $N\_CS/2$)

Data Types: `double`

Data Types: `struct`

References

[1] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access*

Network; Evolved Universal Terrestrial Radio Access (E-UTRA). URL: http://www.3gpp.org.

[2] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePRACHDetect` | `ltePRACHInfo`

ltePRACHDetect

Physical random access channel detection

Syntax

```
[indout,offset] = ltePRACHDetect(ue,chs,waveform,indin)
```

Description

`[indout,offset] = ltePRACHDetect(ue,chs,waveform,indin)` performs PRACH detection given UE-specific settings structure, ue, channel configuration structure, chs, received signal potentially containing a PRACH transmission, waveform, and range of preamble indices for which to search, specified in indin. The detector performs each distinct correlation required to cover all preamble indices, specified in indin, and searches the output of the correlations for peaks which exceed a detection threshold. The position of the peak in the correlator output is used to determine the preamble index that was detected and its associated timing offset, with the preamble index and timing offset being returned in indout and offset respectively.

Although the parameters `Format` and `ConfigIdx` are both described as optional, at least one of these parameters must be specified.

waveform is an N-by-P matrix containing the received time-domain signal in which to search for PRACH transmissions, where N is the number of time-domain samples and P is the number of receive antennas. Such a waveform is generated by the `ltePRACH` function, with P=1. Waveforms with, for example, P=2 or P=4 antennas can be created using one of the channel model functions, `lteFadingChannel`, `lteHSTChannel`, or `lteMovingChannel`. Any other waveform provided must be sampled at the same sampling rate as `ltePRACH` would produce for the same configuration i.e. the same value of `ue.NULRB` as configured here. The appropriate sampling rate can be found in the `SamplingRate` field of the output of `ltePRACHInfo`. It is assumed that any PRACH signal in waveform is synchronized such that the first sample of waveform corresponds to the start of an uplink subframe, therefore any delay from the start of waveform to the first sample of the PRACH therein (except for the case of the appropriate delay to position the transmission of Preamble Format 4 in the UpPTS for TDD special subframes) is interpreted by the detector as a timing offset.

indin is a column vector of preamble indices within the cell for which to search. indin can be between 1 and 64 in length, containing values between 0 and 63.

Output indout is a column vector of indices (a subset of indin) indicating which preambles were found during the search.

Output offset is a corresponding column vector of timing offsets expressed in samples at the input sampling rate.

The detector first calls `info=ltePRACHInfo` to establish the set of root sequences `info.RootSeq` required to cover all preamble indices in indin. A correlation is then performed for each distinct value in `info.RootSeq`, with the inputs to the correlation being the input waveform and a locally generated PRACH waveform. The correlation is performed in the frequency domain. Multiplication of the FFT of the useful part of the locally generated PRACH waveform by a portion of the input waveform extracted with the same timing as the useful part of the locally generated PRACH waveform, followed by an IFFT to give the correlation. Further fields from `info` are then used to establish the length of the window of the correlator output that corresponds to each preamble index, the zero correlation zone. The preamble index is established by testing of the position of the peak in the correlator output to determine if it lies in the window of the correlator output given by the cyclic shift for each preamble index in turn, with the offset within the window being used to compute the timing offset.

Examples

Detect PRACH Preamble

Detect a PRACH preamble that has been delayed by 7 samples.

```
ue = struct('NULRB',9);
config = struct('Format',0,'CyclicShiftIdx',1,'PreambleIdx',44);
tx = ltePRACH(ue,config);
rx = [zeros(7,1); tx];
[index,offset] = ltePRACHDetect(ue,config,rx,(0:63).')
```

```
    44
```

```
7.1895
```

Input Arguments

ue — UE-specific settings

structure array

UE-specific settings, specified as a structure array. ue contains the following fields.

NULRB — Number of uplink resource blocks

positive integer

Number of uplink resource blocks, specified as a positive integer.

Data Types: `double`

NSubframe — Subframe number

`0` (default) | Optional | nonnegative integer

Subframe number, specified as a nonnegative integer. Optional.

Data Types: `double`

NFrame — Frame number

0 (default) | Optional | nonnegative integer

Frame number, specified as a nonnegative integer. Optional.

Data Types: `double`

DuplexMode — Duplex mode

`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, specified as a string. Optional.

Data Types: `char`

TDDConfig — Uplink or downlink configuration

0 (default) | Optional | 1 | 2 | 3 | 4 | 5 | 6

Uplink or downlink configuration, specified as a nonnegative integer between 0 and 6. Optional. Required only for `'TDD'` duplex mode.

Data Types: `double`

`SSC` — Special subframe configuration

0 (default) | Optional | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Special subframe configuration, specified as a nonnegative integer between 0 and 9. Optional. This argument is available when DuplexMode is `'TDD'`.

Data Types: `char`

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional. Required only for Preamble Format 4.

Data Types: `char`

Data Types: `struct`

`chs` — Channel transmission configuration

structure array

Channel transmission configuration, specified as a structure array. chs contains the following fields.

`Format` — Preamble format

Optional | 0...4

Preamble format, specified as a nonnegative integer between 0 and 4. Optional. However, the `Format` field must be specified if the `ConfigIdx` field is not specified. Default is determined by `ConfigIdx` field, if present.

Data Types: `double`

`SeqIdx` — Logical root sequence index

0 (default) | Optional | 0...837

Logical root sequence index, specified as a nonnegative integer between 0 and 837. Optional. (*RACH_ROOT_SEQUENCE*).

Data Types: `double`

`ConfigIdx` — PRACH configuration index

Optional | 0...63

PRACH configuration index, specified as a nonnegative integer between 0 and 63. Optional. However, the `ConfigIdx` field must be specified if the `Format` field is not specified. Default is determined by `Format` field, if present. (*prach-ConfigurationIndex*)

Data Types: `double`

`CyclicShiftIdx` — Cyclic shift configuration index

0 (default) | Optional | 0...15

Cyclic shift configuration index, specified as a nonnegative integer between 0 and 15. Optional. (*zeroCorrelationZoneConfig*, yields *N_CS*)

Data Types: `double`

`HighSpeed` — High speed flag

0 (default) | Optional | 1

High speed flag, specified as a 0 or 1. Optional. A value of 1 signifies a restricted set, and a value of 0 signifies an unrestricted set. (*highSpeedFlag*)

Data Types: `logical` | `double`

`FreqIdx` — Frequency resource index

0 (default) | Optional | 0...5

Frequency resource index, specified as a nonnegative integer between 0 and 5. Optional. Required only for `'TDD'` duplex mode. (*f_RA*)

Data Types: `double`

`FreqOffset` — PRACH frequency offset

0 (default) | Optional | 0...94

PRACH frequency offset, specified as a nonnegative integer between 0 and 94. Optional. Required only for `'TDD'` duplex mode. (*n_PRBoffset*)

Data Types: `double`

Data Types: `struct`

`waveform` — Received signal potentially containing PRACH transmission

numeric matrix

Received signal potentially containing PRACH transmission, specified as a numeric matrix of size *N*-by-*P*. This matrix contains the received time-domain signal in which

to search for PRACH transmissions. *N* is the number of time-domain samples. *P* is the number of receive antennas.

Data Types: `double`
Complex Number Support: Yes

`indin` — Range of preamble indices within the cell for which to search

column vector

Range of preamble indices within the cell for which to search, specified as a column vector. It can be between 1 and 64 in length, containing values between 0 and 63.

Data Types: `double`

Output Arguments

`indout` — Preamble index

column vector

Preamble index, returned as a column vector. These indices, a subset of indin, indicate which preambles were found during the search.

Data Types: `double`

`offset` — Timing offsets

column vector

Timing offsets, in samples, returned as a column vector. This argument contains timing offsets expressed in samples at the input sampling rate.

Data Types: `double`

See Also

`ltePRACH` | `ltePRACHInfo`

ltePRACHInfo

PRACH resource information

Syntax

```
info = ltePRACHInfo(ue,chs)
```

Description

`info = ltePRACHInfo(ue,chs)` returns a structure, info, containing PRACH resource information given UE-specific settings structure, ue, and channel transmission configuration structure, chs.

Although the parameters Format and ConfigIdx are both described as optional, at least one of these parameters must be specified.

The structure info contains PRACH resource information.

info.PRBSet is empty, indicating that the PRACH is not present, or contains 6 consecutive physical resource block (PRB) numbers, indicating the frequency-domain location of the PRACH. The PRACH uses a different SC-FDMA symbol construction from the other channels, PUCCH, PUSCH, and SRS. Therefore, the PRBSet indicates the frequency range, 180kHz per RB, that the PRACH occupies; it does not occupy the set of 12 subcarriers in each RB in the same fashion as other channels. The PRACH occupies a bandwidth approximately equal to 1.08MHz, or 6RBs.

The parameters “PRACH Mask Index” and “PRACH Resource Index,” as described in [3], are not explicit in the configuration, but are implicit in the choice of ue.NSubframe and ue.NFrame.

If chs.ConfigIdx is absent, chs.Format must be specified to indicate the desired preamble format and the PRACH is always generated provided it fits with the overall duplexing arrangement. Specifically, the PRACH is generated in any subframe for FDD, and for TDD the PRACH is generated only in special subframes for Preamble Format 4, and in uplink subframes for Preamble Format 0-3, provided there are info.TotSubframes consecutive uplink subframes for the chosen TDD configuration starting from the current subframe. If chs.ConfigIdx is present, further validation is used to comply with table 5.7.1-2 for FDD and table 5.7.1-4 for TDD of [2]. Specifically, chs.Format, if present, is

validated against chs.ConfigIdx and a preamble is only generated in appropriate frames and subframes; for contradictory values of chs.Format and chs.ConfigIdx, a warning is issued. If chs.Format is absent, it is inferred, if possible, from chs.ConfigIdx. If the entry in table 5.7.1-2 for FDD or table 5.7.1-4 for TDD of [2] indicates "N/A" for the preamble format, an error is issued. For TDD, `chs.FreqIdx` corresponds to the first entry in the quadruples in table 5.7.1-4 of [2]. The other three entries (*t_RA0*,*t_RA1*, *t_RA2*) in the quadruple are completely specified by ue.NSubframe and ue.NFrame. If a combination of chs.ConfigIdx, ue.TDDConfig, *t_RA0*, *t_RA1*, *t_RA2* given by ue.NSubframe and ue.NFrame and chs.FreqIdx appears in table 5.7.1-4, the PRACH is generated.

In accordance with the above logic, ue.NSubframe and ue.NFrame are not required at all for FDD if chs.ConfigIdx is absent. In the case that a preamble is not generated under the above rules, info.PRBSet is empty and the waveform generated by `ltePRACH` consists of all zeros.

If chs.PreambleIdx is a vector, info.RootSeq is a vector containing the physical root Zadoff-Chu sequence index required to generate the PRACH for each of the configured set of preamble indices. info.CyclicShift is also a vector indicating the cyclic shift for each of the configured set of preamble indices. If chs.PreambleIdx is a scalar, info.RootSeq and info.CyclicShift are also scalars, indicating the physical root Zadoff-Chu sequence index and cyclic shift for the single preamble index, respectively. For High Speed mode, the info.CyclicOffset field is present, containing a vector of cyclic offset values for each of the configured set of preamble indices. Again, if chs.PreambleIdx is a scalar, info.CyclicOffset is a scalar. For High Speed mode, any element of info.CyclicOffset equal to –1 indicates that there are no cyclic shifts in the restricted set for the corresponding preamble index.

info.NCS is the length of the zero correlation zone, plus one, corresponding to the complete extent of autocorrelation lags (0 and *N_CS-1* nonzero) that exhibits perfect correlation properties (1 at 0 lag, 0 at nonzero lags). info.NCS is expressed directly as in the standard, related to the fundamental Zadoff-Chu sequence construction. The actual sample span of the zero correlation zone in the waveform generated by `ltePRACH` is a function of the sampling rate.

Examples

Find Root Zadoff-Chu Sequences from PRACH Information

Find the set of root Zadoff-Chu sequences required for all preamble indices (0...63) in a cell.

```
ue.NULRB = 6;
config.Format = 0;
config.CyclicShiftIdx = 8;
config.PreambleIdx = (0:63);
prachInfo = ltePRACHInfo(ue,config);
unique(prachInfo.RootSeq)
```

```
   129   140   699   710
```

Input Arguments

ue — UE-specific settings

structure array

UE-specific settings, specified as a structure array. ue can contain the following fields.

NULRB — Number of uplink resource blocks

positive integer

Number of uplink resource blocks, specified as a positive integer.

Data Types: `double`

NSubframe — Subframe number

`0` (default) | Optional | nonnegative scalar integer

Subframe number, specified as a nonnegative scalar integer. Optional.

Data Types: `double`

NFrame — Frame number

0 (default) | Optional | nonnegative scalar integer

Frame number, specified as a nonnegative scalar integer. Optional.

Data Types: `double`

DuplexMode — Duplex mode

`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, specified as a string. Optional.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration

0 (default) | Optional | 1 | 2 | 3 | 4 | 5 | 6

Uplink or downlink configuration, specified as a nonnegative scalar integer between 0 and 6. Optional. Required only for `'TDD'` duplex mode.

Data Types: `double`

`SSC` — Special subframe configuration

0 (default) | Optional | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Special subframe configuration, specified as a nonnegative scalar integer between 0 and 9. Optional. This argument is available when DuplexMode is `'TDD'`.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional. Required only for Preamble Format 4.

Data Types: `char`

Data Types: `struct`

`chs` — Channel transmission configuration

scalar structure

Channel transmission configuration, specified as a scalar structure. chs contains the following fields.

`Format` — Preamble format

Optional | 0...4

Preamble format, specified as a nonnegative scalar integer between 0 and 4. Optional. However, the `Format` field must be specified if the `ConfigIdx` field is not specified. Default is determined by `ConfigIdx` field, if present.

Data Types: `double`

`SeqIdx` — Logical root sequence index

0 (default) | Optional | 0...837

Logical root sequence index, specified as a nonnegative scalar integer between 0 and 837. Optional. (*RACH_ROOT_SEQUENCE*).

Data Types: `double`

`ConfigIdx` — PRACH configuration index

Optional | 0...63

PRACH configuration index, specified as a nonnegative scalar integer between 0 and 63. Optional. However, the `ConfigIdx` field must be specified if the `Format` field is not specified. Default is determined by `Format` field, if present. (*prach-ConfigurationIndex*)

Data Types: `double`

`PreambleIdx` — Scalar preamble index within cell

0 (default) | Optional | 0...63

Scalar preamble index within cell, specified as a nonnegative scalar integer between 0 and 63. Optional. (*ra-PreambleIndex*).

Data Types: `double`

`CyclicShiftIdx` — Cyclic shift configuration index

0 (default) | Optional | 0...15

Cyclic shift configuration index, specified as a nonnegative scalar integer between 0 and 15. Optional. (*zeroCorrelationZoneConfig*, yields *N_CS*).

Data Types: `double`

`HighSpeed` — High speed flag

0 (default) | Optional | 1

High speed flag, specified as a 0 or 1. Optional. A value of 1 signifies a restricted set. A value of 0 signifies an unrestricted set. (*highSpeedFlag*)

Data Types: `logical` | `double`

`FreqOffset` — PRACH frequency offset

0 (default) | Optional | 0...94

PRACH frequency offset, specified as a nonnegative scalar integer between 0 and 94. Optional. Required only for `'TDD'` duplex mode. (*n_PRBoffset*)

Data Types: `double`

`FreqIdx` — Frequency resource index

0 (default) | Optional | 0...5

Frequency resource index, specified as a nonnegative scalar integer between 0 and 5. Optional. Required only for `'TDD'` duplex mode. (*f_RA*).

Data Types: `double`

Data Types: `struct`

Output Arguments

`info` — PRACH resource information

scalar structure

PRACH resource information, returned as a scalar structure. info contains the following fields.

`NZC` — Zadoff-Chu sequence length

positive integer

Zadoff-Chu sequence length, returned as a positive integer. (*N_ZC*)

Data Types: `double`

`SubcarrierSpacing` — Subcarrier spacing of PRACH preamble

positive integer

Subcarrier spacing of PRACH preamble, in Hz, returned as a positive integer. (*deltaf_RA*)

Data Types: `double`

`Phi` — Frequency-domain location offset

positive integer

Frequency-domain location offset, returned as a positive integer. (*phi*)

Data Types: `double`

`K` — Ratio of uplink data to PRACH subcarrier spacing

numeric scalar

Ratio of uplink data to PRACH subcarrier spacing, returned as a numeric scalar. (*K*)

Data Types: `double`

`TotSubframes` — Number of subframes duration of PRACH

numeric scalar

Number of subframes duration of the PRACH, returned as a numeric scalar. Each subframe lasts 30720 fundamental periods, therefore `TotSubframes` is `ceil(sum(Fields)/30720)`, the number of subframes required to hold the entire PRACH waveform. The duration of the PRACH is a function of the Preamble Format as described in table 5.7.1-1 of [2].

Data Types: `double`

`Fields` — PRACH field lengths

1-by-4 numeric vector

PRACH field lengths, returned as a 1-by-4 numeric vector. The elements are [*OFFSET T_CP T_SEQ GUARD*]. *T_CP* and *T_SEQ* are the lengths in fundamental time periods (*T_s*), of cyclic prefix and PRACH sequence, respectively. *OFFSET* is the number of fundamental time periods from the start of configured subframe to the start of the cyclic prefix, and is non-zero only for TDD special subframes. *GUARD* is the number of fundamental time periods from the end of the PRACH sequence to the end of the number of subframes spanned by the PRACH.

Data Types: `double`

`PRBSet` — PRBs occupied by PRACH preamble

nonnegative integer column vector

PRBs occupied by PRACH preamble, returned as a nonnegative integer column vector. (starts at *n_PRB*, 0-based).

- An empty `info.PRBSet` indicates that the PRACH is not present and the waveform generated by ltePRACH consists of all zeros.
- An `info.PRBSet` that contains six consecutive Physical Resource Block numbers indicates the frequency domain location of the PRACH.

Note: The PRACH uses a different SC-FDMA symbol construction from the other channels (PUCCH, PUSCH and SRS) and therefore the PRBSet indicates the frequency range (180kHz per RB) that the PRACH occupies, it does not occupy the set of 12

subcarriers in each RB in the same fashion as other channels. The PRACH occupies a bandwidth approximately equal to 1.08MHz = 6RBs.

Data Types: `uint32`

`NCS` — Length of zero correlation zone plus 1

positive integer

Length of zero correlation zone plus 1, specified as a positive integer (*N_CS*). `NCS` corresponds to the complete extent of autocorrelation lags (0 and *N_CS*–1 non-zero) that exhibit perfect correlation properties (1 at 0 lag, 0 at non-zero lags). `NCS` is expressed directly, as in the standard, related to the fundamental Zadoff-Chu sequence construction. The actual sample span of the zero correlation zone in the waveform generated by `ltePRACH` is a function of the sampling rate.

Data Types: `double`

`CyclicShift` — Cyclic shift or shifts of Zadoff-Chu sequence

numeric row vector

Cyclic shift or shifts of Zadoff-Chu sequence, returned as a numeric row vector. (*C_v*).

For High Speed mode, any element of `CyclicShift` equal to –1 indicates that there are no cyclic shifts in the restricted set for the corresponding preamble index.

Data Types: `double`

`RootSeq` — Physical root Zadoff-Chu sequence index or indices

numeric row vector

Physical root Zadoff-Chu sequence index or indices, returned as a numeric row vector. (*u*)

Data Types: `double`

`CyclicOffset` — Cyclic shift or shifts corresponding to Doppler Shift

vector

For High Speed mode, Cyclic shift or shifts corresponding to a Doppler Shift of 1/*T_SEQ* (*d_u*).

For High Speed mode, the field `CyclicOffset` is present, containing a vector of cyclic offset values for each of the configured set of preamble indices. If `chs.PreambleIdx` is a scalar, `CyclicOffset` is a scalar.

Data Types: `double`

`SamplingRate` — Sampling rate of PRACH modulator
numeric scalar

Sampling rate of the PRACH modulator, returned as a numeric scalar.

Data Types: `double`

`BaseOffset` — Base timing offset
numeric scalar

Base timing offset, in microseconds. This field is used for the detection test in [1]. (duration of *N_CS*/2)

Data Types: `double`

Data Types: `struct`

References

[1] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[3] 3GPP TS 36.321. "Medium Access Control (MAC) protocol specification." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePRACH` | `ltePRACHDetect`

ltePRBS

Pseudorandom binary sequence

Syntax

```
seq = ltePRBS(cinit,n)
seq = ltePRBS(cinit,n,mapping)
```

Description

`seq = ltePRBS(cinit,n)` returns an n-element column vector containing the first n elements of the pseudorandom binary sequence (PRBS) generator when initialized with 32-bit integer, cinit. PRBS sequences are used for scrambling of physical channels for interference mitigation.

`seq = ltePRBS(cinit,n,mapping)` allows control over the format of the returned sequence seq through the string mapping. Valid formats are `'binary'` and `'signed'`. The `'binary'` format maps `true` to 1 and `false` to 0. The `'signed'` format maps `true` to –1 and `false` to 1.

Examples

Generate Pseudorandom Binary Sequence

Generate an unsigned pseudorandom binary sequence.

```
seq = ltePRBS(162,4);
seq(1:4)
```

```
     1
     0
     1
     1
```

Generate Signed Pseudorandom Binary Sequence

Generate a signed pseudorandom binary sequence.

```
seq = ltePRBS(162,4,'signed');
seq(1:4)
```

```
    -1
     1
    -1
    -1
```

Generate PRBS from Physical Layer Cell Identity

Generate a pseudorandom binary sequence based on physical layer cell identity for RMC R.0.

Create a reference measurement channel (RMC) for configuration R.0. Use the physical layer cell identity, `NCellID`, as an initial value to generate the pseudorandom binary sequence.

```
enb = lteRMCDL('R.0');
prbsSeq = ltePRBS(enb.NCellID,5)
```

```
     0
     0
     0
     0
     0
```

Input Arguments

`cinit` — Initialization value

32-bit integer

32-bit integer initialization value

Data Types: `int32` | `uint32` | `double`

`n` — Number of outputs

positive scalar integer

Number of outputs, specified as a positive scalar integer

Data Types: `double`

`mapping` — Format of returned sequence

`'binary'` (default) | `'signed'`

Format of returned sequence, specified as a string. This string controls the format of the returned sequence, seq. The string `'binary'` maps true to 1 and false to 0. The string `'signed'` maps true to –1 and false to 1.

Data Types: `char`

Output Arguments

`seq` — Pseudorandom binary sequence

logical column vector | numeric column vector

Pseudorandom binary sequence, returned as a logical column vector or a numeric column vector. The vector contains the first n elements of the PRBS generator, when initialized with 32-bit integer cinit. If the sequence is binary, the values are of data type logical. If the sequence is signed, the values are of data type double.

Data Types: `logical` | `double`

See Also

`ltePBCHPRBS` | `ltePCFICHPRBS` | `ltePDCCHPRBS` | `ltePDSCHPRBS` | `ltePHICHPRBS` | `ltePUCCH2PRBS` | `ltePUCCH3PRBS`

ltePRS

Positioning reference signal

Syntax

```
sym = ltePRS(enb)
```

Description

`sym = ltePRS(enb)` returns a column vector containing the positioning reference signal (PRS) symbols for transmission in a single subframe on antenna port 6. These symbols are ordered as they should be mapped into the resource elements along with `ltePRSIndices`. If, according to the PRS subframe configuration and duplex mode, there are no PRS scheduled in the subframe, then the output vector us empty.

The optional `PRSPeriod` parameter controls the downlink subframes in which PRS is present. See the `ltePRSIndices` function reference page for details.

Examples

Generate Positioning Reference Signal Symbols

Generate the PRS symbols for subframe 0 of a 10MHz downlink.

```
rmc = lteRMCDL('R.2');
rmc.NPRSRB = rmc.NDLRB;
rmc.PRSPeriod = 0;
prsSymbols = ltePRS(rmc);
prsSymbols(1:4)
```

```
   0.7071 + 0.7071i
   0.7071 + 0.7071i
   0.7071 + 0.7071i
```

```
0.7071 + 0.7071i
```

Input Arguments

`enb` — Cell-wide settings
structure

Cell-wide settings, specified as a structure. enb contains the following parameter fields.

The parameters `TDDConfig` and `SSC` are only required if `DuplexMode` is set to `'TDD'`.

`NDLRB` — Number of downlink resource blocks
positive scalar integer (6...110)

Number of downlink resource blocks, specified as a positive scalar integer between 6 and 110.

Example: 45

Data Types: `double`

`CellRefP` — Number of cell-specific reference signal antenna ports
1 | 2 | 4

Number of cell-specific reference signal antenna ports, specified as a 1, 2, or 4.

Example: 1

Data Types: `double`

`NCellID` — Physical layer cell identity number
nonnegative scalar integer

Physical layer cell identity number, specified as a nonnegative scalar integer.

Example: 4

Data Types: `double`

`NSubframe` — Subframe number
nonnegative scalar integer

Subframe number, specified as nonnegative scalar integer.

Example: 5

Data Types: `double`

`NFrame` — Frame number

0 (default) | Optional | nonnegative scalar integer

Frame number, specified as nonnegative scalar integer. Optional.

Example: 6

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional. Valid values include `'Normal'` or `'Extended'`.

Data Types: `char`

`DuplexMode` — Duplex mode type

`'FDD'` (default) | Optional | `'TDD'`

Duplex mode type, specified as `'FDD'` or `'TDD'`. Optional. Used for separating the transmission signals.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration for TDD

0 (default) | Optional | nonnegative scalar integer (0...6)

Uplink or downlink configuration for TDD, specified as a nonnegative scalar integer between 0 and 6. Optional. Required only for `'TDD'` duplex mode.

Example: 4

Data Types: `double`

`SSC` — Special subframe configuration

0 (default) | Optional | nonnegative scalar integer (0...9)

Special subframe configuration, specified as nonnegative scalar integer between 0 and 9. Optional. Required only for `'TDD'` duplex mode.

Example: 6

Data Types: `double`

`NPRSRB` — Number of PRS physical resource blocks

0...`NDLRB`

Number of PRS physical resource blocks, specified as nonnegative scalar integer between 0 and `NDLRB`.

Example: 8

Data Types: `double`

`PRSPeriod` — Positioning reference signal (PRS) period

`'On'` (default) | Optional | `'Off'` | `[Iprs]` | `[Tprs Dprs]`

Positioning reference signal (PRS) period, specified as `'On'`, `'Off'`, a numeric scalar, or a 1-by-2 vector. Optional. This parameter controls the downlink subframes in which PRS will be present. For details, see `ltePRSIndices`.

Example: 0

Example: [160 0]

Data Types: `char` | `double`

Output Arguments

`sym` — Positioning Reference Signal (PRS) symbols

complex numeric column vector

Positioning Reference Signal (PRS) symbols, returned as complex numeric column vector, for transmission in a single subframe on antenna port 6.

Example: 0.7071 + 0.7071i

Data Types: `double`
Complex Number Support: Yes

See Also

`lteCellRS` | `lteCSIRS` | `lteDMRS` | `ltePRSIndices`

ltePRSIndices

PRS resource element indices

Syntax

```
ind = ltePRSIndices(enb)
ind = ltePRSIndices(enb,opts)
```

Description

`ind = ltePRSIndices(enb)` returns a column vector of 1-based linear indices for the PRS elements in the subframe, given the cell-wide settings parameter structure, enb. The length of ind is the number of resource elements (NRE). It returns the indices for the Positioning Reference Signal (PRS) resource element (RE) locations transmitted on antenna port 6. By default, these indices are in 1-based linear indexing form that can directly index elements in a matrix representing a single subframe of the port 6 resource grid. Other index representations can also be created. These indices are ordered as the complex PRS symbols should be mapped and will not include any elements allocated to PBCH, PSS, and SSS. A PRS subframe configuration schedule can be defined as required. If the subframe contains no PRS, ind is an empty vector.

The optional enb`.PRSPeriod` parameter controls the downlink subframes in which PRS will be present, either always `'On'` or `'Off'`, or defined by the scalar subframe configuration index, *Iprs* (0...2399), or the explicit subframe periodicity and offset pair, `[Tprs Dprs]`, as listed in section 6.10.4.3 of [1]. The PRS containing subframes are located in conjunction with the parameters enb`.NSubframe` and optional enb`.NFrame`. `NSubframe` can be greater than 10; thus, setting `NSubframe` to 11 is equivalent to setting `NSubframe` to 1 and `NFrame` to 1.

`ind = ltePRSIndices(enb,opts)` allows control of the format of the returned indices through a cell array opts of option strings.

Examples

Generate PRS Resource Element Indices

Generate the PRS resource element (RE) indices for subframe 0 of a 10 MHz downlink.

```
rmc = lteRMCDL('R.2');
rmc.NPRSRB = rmc.NDLRB;
rmc.PRSPeriod = 0;
prsIndices = ltePRSIndices(rmc,'ind');
prsIndices(1:4)
```

```
        1804
        1810
        1816
        1822
```

Input Arguments

`enb` — Cell-wide settings

structure

Cell-wide settings, specified as a structure. enb contains the following fields.

The parameters `TDDConfig` and `SSC` are only required if `DuplexMode` is set to `'TDD'`.

`NDLRB` — Number of downlink resource blocks

6...110

Number of downlink resource blocks, specified as a nonnegative scalar integer between 6 and 110.

Example: 50

Data Types: `double`

`CellRefP` — Number of cell-specific reference signal antenna ports

1 (default) | 2 | 4

Number of cell-specific reference signal antenna ports, specified as 1, 2, or 4.

Example: 1

Data Types: `double`

`NCellID` — Physical layer cell identity

nonnegative scalar integer

Physical layer cell identity, specified as a nonnegative scalar integer.

Example: 3

Data Types: `double`

`NSubframe` — Subframe number
nonnegative scalar integer

Subframe number, specified as a nonnegative scalar integer.

Example: 3

Data Types: `double`

`NFrame` — Frame number
0 (default) | Optional | nonnegative scalar integer

Frame number, specified as a nonnegative scalar integer. Optional.

Example: 3

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`DuplexMode` — Duplex mode type
`'FDD'` (default) | Optional | `'TDD'`

Duplex mode type, specified as `'FDD'` or `'TDD'`. Optional.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration for TDD
0 (default) | Optional | 0...6

Uplink or downlink configuration for TDD, specified as a nonnegative scalar integer between 0 and 6. Optional. Required only for `'TDD'` duplex mode.

Example: 4

Data Types: `double`

`SSC` — Special subframe configuration for TDD
0 (default) | Optional | 0...9

Example: 5

Special subframe configuration for TDD, specified as a nonnegative scalar integer between 0 and 9. Optional. Required only for `'TDD'` duplex mode.

Data Types: `double`

`NPRSRB` — Number of PRS physical resource blocks

0...`NDLRB`

Number of PRS physical resource blocks, specified as a nonnegative scalar integer between 0 and `NDLRB`.

Example: 32

Data Types: `double`

`PRSPeriod` — Positioning reference signal (PRS) period

`'On'` (default) | Optional | `'Off'` | `[Iprs]` | `[Tprs Dprs]`

Positioning reference signal (PRS) period, specified as `'On'`, `'Off'`, a numeric scalar, or a 1-by-2 vector. Optional. This parameter controls the downlink subframes in which PRS will be present. For details, see `ltePRSIndices`.

Example: 0

Example: [160 0]

Data Types: `char` | `double`

`opts` — Options to control format of returned indices

string | cell array of strings

Options to control format of returned indices, specified as a string or a cell array of strings. opts can contain the following option strings.

`Indexing style` — Indexing style of returned indices

`'ind'` (default) | `'sub'`

Cell array option, specified as `'ind'` or `'sub'`. If `'ind'`, the returned indices are in linear index form. If `'sub'`, the indices are in `[subcarrier, symbol, antenna]` subscript form. In this case, ind is a matrix of size *NRE*-by-3.

Data Types: `char`

`Index base` — Index base of returned indices

`'1based'` (default) | `'0based'`

Index base of returned indices, specified as `'1based'` or `'0based'`. This option controls whether the returned indices are 1-based or 0-based.

Data Types: `char`

Data Types: `char` | `cell`

Output Arguments

`ind` — PRS resource element indices

integer column vector | integer matrix

PRS resource element indices, returned as an integer column vector of length *NRE* or an integer matrix of size *NRE*-by-3. These indices are for the PRS resource element (RE) locations transmitted on antenna port 6.

Example: 1804

Data Types: `uint32`

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`lteCellRSIndices` | `lteCSIRSIndices` | `lteDMRSIndices` | `ltePRS`

ltePSS

Primary synchronization signal

Syntax

```
s = ltePSS(enb)
```

Description

`s = ltePSS(enb)` returns a complex column vector containing the primary synchronization signal (PSS) values for cell-wide settings in the enb structure.

This signal is only defined for subframes 0 and 5 in FDD, and subframes 1 and 6 in TDD. Therefore, an empty vector is returned for other values of `NSubframe`. This behavior allows this function and the corresponding sequence function `ltePSSIndices` to index the resource grid for any subframe number as described in "Resource Grid Indexing". However, the resource grid is only modified in subframes 0 and 5 in FDD, or subframes 1 and 6 in TDD.

Examples

Generate Primary Synchronization Signal Values

Generate the primary synchronization signal (PSS) values, providing cell-wide settings.

```
pss = ltePSS(struct('NCellID',1,'NSubframe',0));
pss(1:4)
```

```
   1.0000 + 0.0000i
  -0.9691 - 0.2468i
  -0.7331 - 0.6802i
   0.0747 + 0.9972i
```

Input Arguments

enb — Cell-wide settings

structure

Cell-wide settings, specified as a structure. enb contains the following fields.

`NCellID` — Physical layer cell identity number
nonnegative scalar integer

Physical layer cell identity number, specified as a nonnegative scalar integer.

Example: 6

Data Types: `double`

`NSubframe` — Subframe number
0 (default) | Optional | nonnegative scalar integer

Subframe number, specified as nonnegative scalar integer. Optional.

Example: 8

Data Types: `double`

`DuplexMode` — Duplex mode type
`'FDD'` (default) | Optional | `'TDD'`

Duplex mode type, specified as `'FDD'` or `'TDD'`. Optional. Used for separating the transmission signals.

Data Types: `char`

Output Arguments

`s` — Primary synchronization signal (PSS) values
complex-valued numeric column vector

Primary synchronization signal (PSS) values, returned as a complex-valued numeric column vector. These value are created for the cell-wide settings in the enb structure.

Example: 1.0000 + 0.0000i

Data Types: `double`
Complex Number Support: Yes

See Also

`ltePSSIndices` | `lteSSS`

ltePSSIndices

PSS resource element indices

Syntax

```
ind = ltePSSIndices(enb)
ind = ltePSSIndices(enb,port)
ind = ltePSSIndices(enb,port,opts)
```

Description

`ind = ltePSSIndices(enb)` returns a column vector, ind, of resource element (RE) indices, Port 0 oriented, for the Primary Synchronization Signal (PSS), using the cell-wide settings structure, enb. By default, the indices are returned in 1-based linear indexing form that can directly index elements of a 3-D array representing the resource array. These indices are ordered as the PSS modulation symbols should be mapped. Alternative indexing formats can also be generated.

`ind = ltePSSIndices(enb,port)` returns indices appropriate for antenna port, port, which must be either 0, 1, 2, or 3.

`ind = ltePSSIndices(enb,port,opts)` allows control of the format of the returned indices through a cell array opts of option strings.

These indices are only defined for subframes 0 and 5 in FDD, and subframes 1 and 6 in TDD. Therefore, an empty vector is returned for other values of `NSubframe`. This behavior allows this function and the corresponding sequence function `ltePSS` to index the resource grid for any subframe number as described in “Resource Grid Indexing”. However, the resource grid is only modified in subframes 0 and 5 in FDD, or subframes 1 and 6 in TDD.

Examples

Get PSS Resource Element Indices

Get 0-based PSS resource element indices in linear form for antenna port 0.

```
enb = lteRMCDL('R.4');
ind = ltePSSIndices(enb,0,{'0based','ind'});
ind(1:4)
```

```
         437
         438
         439
         440
```

Input Arguments

`enb` — Cell-wide settings
structure

Cell-wide settings, specified as a structure. enb contains the following fields.

`NDLRB` — Number of downlink resource blocks
6...110

Number of downlink resource blocks, specified as nonnegative scalar integer.

Example: 9

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`NSubframe` — Subframe number
0 (default) | Optional | nonnegative scalar integer

Subframe number, specified as nonnegative scalar integer. Optional.

Example: 9

Data Types: `double`

`DuplexMode` — Duplex mode type
`'FDD'` (default) | Optional | `'TDD'`

Duplex mode type, specified as `'FDD'` or `'TDD'`. Optional.

Data Types: `char`

`port` — Antenna port number

0 (default) | 1 | 2 | 3

Antenna port number, specified as nonnegative scalar integer.

Example: 0

Data Types: `double`

`opts` — Options to control format of returned indices

string | cell array of strings

Options to control format of returned indices, specified as a string or a cell array of strings. opts can contain the following option strings.

`Indexing style` — Indexing style of returned indices

`'ind'` (default) | `'sub'`

Cell array option, specified as `'ind'` or `'sub'`. If `'ind'`, the returned indices are in linear index form. If `'sub'`, the indices are in `[subcarrier, symbol, antenna]` subscript form.

Data Types: `char`

`Index base` — Index base of returned indices

`'1based'` (default) | `'0based'`

Index base of returned indices, specified as `'1based'` or `'0based'`. This option string controls whether the returned indices are 1-based or 0-based.

Data Types: `char`

Data Types: `char | cell`

Output Arguments

`ind` — PSS resource element indices

integer column vector | 3-column integer matrix

PSS resource element indices, returned as an integer column vector or a 3-column integer matrix. This output is generated using the cell-wide settings structure, enb.

Data Types: `uint32`

See Also

`ltePSS` | `lteSSSIndices`

ltePUCCH1

Physical uplink control channel format 1

Syntax

```
[sym,info] = ltePUCCH1(ue,chs,ack)
```

Description

`[sym,info] = ltePUCCH1(ue,chs,ack)` returns a complex matrix, sym, containing Physical Uplink Control Channel (PUCCH) format 1 symbols and information structure array, info, for UE-specific settings, ue, channel transmission configuration, chs, and vector of hybrid ARQ (HARQ) indicator values, ack. The symbols for each antenna are in the columns of sym, with the number of columns determined by the number of PUCCH resource indices specified in the chs structure.

ack is a vector containing 0, 1 or 2 hybrid ARQ indicator values; the corresponding PUCCH transmission will be of format 1, 1a, or 1b, respectively. This vector is denoted "block of bits $b(0),...,b(M_{bit}-1)$" in section 5.4.1 of [1]. An M_{bit} value of 0, 1, or 2 corresponds to PUCCH format 1, 1a, or 1b, respectively, as described in table 5.4-1 of [1].

For shortened transmissions, when `ue.Shortened` is 1, the second column of `info.OrthSeq` has a zero in the last row because in this case, the spreading factor for the second slot is 3, rather than 4.

Examples

Generate PUCCH Format 1 Symbols

Generate the physical uplink control channel (PUCCH) format 1 symbols for UE-specific settings.

```
ue = struct('NCellID',1,'NSubframe',0);
pucch1Sym = ltePUCCH1(ue,struct('ResourceIdx',0),[]);
pucch1Sym(1:4)
```

```
  0.7071 + 0.7071i
 -0.9659 + 0.2588i
  0.9659 + 0.2588i
 -0.7071 + 0.7071i
```

Input Arguments

`ue` — UE-specific settings

structure

UE-specific settings, specified as a structure. ue can contain the following fields.

`NCellID` — Physical layer cell identity

nonnegative scalar integer

Physical layer cell identity, specified as a nonnegative scalar integer.

Example: 4

Data Types: `double`

`NSubframe` — Subframe number

nonnegative scalar integer

Position reference signal subframe number, specified as a nonnegative scalar integer.

Example: 8

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length for uplink channels

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink channels, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`Hopping` — Uplink frequency hopping

`'Off'` (default) | Optional | `'Group'`

Uplink frequency hopping, specified as `'Off'` or `'Group'`. Optional.

Data Types: `char`

`Shortened` — Shorten subframe flag

0 (default) | Optional | 1

Flag to shorten the subframe, specified as 0 or 1. Optional. This is required for subframes with possible SRS transmission. If 1, the last symbol of the subframe is not used.

Data Types: `double`

Data Types: `struct`

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure. chs contains the following fields.

`ResourceIdx` — PUCCH resource indices

0 (default) | Optional | nonnegative vector (0...2047)

PUCCH resource indices, specified as nonnegative vector, one for each transmission antenna. These determine the cyclic shift and orthogonal cover used for transmission.

Example: 78

Data Types: `double`

`DeltaShift` — Delta shift

1 (default) | Optional | 2 | 3

Delta shift, specified as 1, 2, or 3. Optional. (*delta_shift*)

Data Types: `double`

`DeltaOffset` — Delta offset

0 (default) | Deprecated | Optional | 1 | 2

Note: Warning: The use of the `'DeltaOffset'` parameter field is deprecated. This parameter may be removed in a future release.

Delta offset, specified as 0, 1, or 2. Optional. (*delta_offset*)

Data Types: `double`

`CyclicShifts` — Number of cyclic shifts for format 1 resource blocks

0 (default) | Optional | nonnegative scalar integer (0...7)

Number of cyclic shifts for format 1 resource blocks, in RBs, specified as nonnegative scalar integer between 0 and 7. Optional. This parameter can be used in a mixture of format 1 and format 2 PUCCH. (*N1cs*)

Data Types: `double`

`ack` — Hybrid ARQ indicator values

0 (default) | nonnegative integer vector (0, 1, 2)

Hybrid ARQ indicator values, specified as nonnegative integer vector with values of 0, 1, or 2. The corresponding PUCCH transmission is of format 1, 1a, or 1b, respectively. This vector is denoted "block of bits $b(0),...,b(M_{bit}-1)$" in section 5.4.1 of [1]. An M_{bit} value of 0, 1, or 2 corresponds to PUCCH format 1, 1a, or 1b, respectively, as described in table 5.4-1 of [1].

Example: [0,1,2]

Data Types: `double`

Output Arguments

`sym` — PUCCH format 1 symbols

complex numeric column vector

PUCCH format 1 symbols, returned as complex numeric column vector.

Example: 0.7071 + 0.7071i

Data Types: `double`
Complex Number Support: Yes

`info` — PUCCH format 1 resource information

structure array

PUCCH format 1 resource information, returned as a structure array. info is a structure array having the following fields.

`Alpha` — Reference signal cyclic shift for OFDM symbol

two-column row vector

Reference signal cyclic shift, returned as two-column row vector, for each OFDM symbol.

Data Types: double

SeqGroup — PUCCH base sequence group number for each slot
two-column row vector

PUCCH base sequence group number for each slot, returned as two-column row vector. (u)

Data Types: double

SeqIdx — PUCCH base sequence group number indices
two-column row vector

PUCCH base sequence group number indices, returned as two-column row vector, for each slot. (v)

Data Types: double

NResourceIdx — PUCCH resource indices for each slot
two-column row vector

PUCCH resource indices for each slot, returned as a two-column row vector. (n')

Data Types: double

NCellCyclicShift — Cell-specific cyclic shift for each OFDM symbol
row vector

Cell-specific cyclic shift for each OFDM symbol, returned as a row vector. (*ncell_cs*)

Data Types: double

OrthSeqIdx — Orthogonal sequence index for each slot
row vector

Orthogonal sequence index for each slot, returned as a two-column row vector. (n_oc)

Data Types: double

Symbols — Modulated data symbols for each OFDM symbol
row vector

Modulated data symbols for each OFDM symbol, returned as row vector. (*d(0)*)

Data Types: `double`

`OrthSeq` — Orthogonal sequence of each slot

numeric matrix

Orthogonal sequence of each slot, returned as numeric matrix. Each column in the matrix contains the orthogonal sequence (*w_n_oc*) for each slot

Data Types: `double`

`ScrambSeq` — Scrambling value

two-column row vector

Scrambling value for each slot (*S*), returned as two-column row vector.

Data Types: `double`

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePUCCH1Decode` | `ltePUCCH1DRS` | `ltePUCCH1DRSIndices` | `ltePUCCH1Indices` | `ltePUCCH2` | `ltePUCCH3`

ltePUCCH1Decode

Physical uplink control channel format 1 decoding

Syntax

```
ack = ltePUCCH1Decode(ue,chs,oack,sym)
```

Description

`ack = ltePUCCH1Decode(ue,chs,oack,sym)` returns a vector of hybrid-ARQ (HARQ) indicator values, ack, obtained by performing PUCCH Format 1 decoding of the complex matrix sym. The decoder uses a maximum likelihood (ML) approach, assuming that sym has already been equalized to best restore the original transmitted complex values. The symbols for each antenna are in the columns of sym, and the number of columns should match the number of PUCCH resource indices specified in the structure chs.

The input argument oack specifies the number of Hybrid ARQ indicator values expected. It can be either 1 for PUCCH Format 1A or 2 for PUCCH Format 1B.

The output argument ack is a vector containing oack hybrid-ARQ indicator values.

Examples

Decode PUCCH Format 1B Symbols

Decode a received PUCCH format 1B symbol vector.

Create a UE-specific configuration structure and use it to generate PUCCH symbols. Then decode the symbols.

```
ue = struct('NCellID',0,'NSubframe',0,'CyclicPrefixUL','Normal');
config = struct();
txAck = [0;1];
pucchSym = ltePUCCH1(ue,config,txAck);
```

```
rxAck = ltePUCCH1Decode(ue,config,length(txAck),pucchSym)

     0
     1
```

Input Arguments

ue — UE-specific settings
structure

UE-specific settings, specified as a structure. ue contains the following fields.

NCellID — Cell identity number
0 (default) | nonnegative scalar integer

Physical layer cell identity number, specified as a nonnegative scalar integer.

Example: 4

Data Types: `double`

NSubframe — Subframe number
0 (default) | nonnegative scalar integer

Position reference signal subframe number, specified as a nonnegative scalar integer.

Example: 8

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length for uplink channels
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink channels, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

Hopping — Uplink frequency hopping
`'Off'` (default) | Optional | `'Group'`

Uplink frequency hopping, specified as `'Off'` or `'Group'`. Optional.

Data Types: `char`

`Shortened` — Shorten subframe flag

0 (default) | 1

Flag to shorten the subframe, specified as `0` or `1`. This is required for subframes with possible `Sound Referencing Signal (SRS)` transmission. If `1`, the last symbol of the subframe is not used.

Data Types: `double`

Data Types: `struct`

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure. chs can contain the following fields.

`ResourceIdx` — PUCCH resource indices

nonnegative integer vector (0...2047)

PUCCH resource indices, specified as nonnegative integer vector with values ranging between 0 and 2047. There is one element for each transmission antenna. These indices determine the cyclic shift and orthogonal cover used for transmission. This input argument is optional.

Example: 78

Data Types: `double`

`DeltaShift` — Delta shift

1 (default) | Optional | 2 | 3

Delta shift, specified as 1, 2, or 3. Optional. (*delta_shift*)

Data Types: `double`

`DeltaOffset` — Delta offset

0 (default) | Deprecated | Optional | 1 | 2

Note: Warning: The use of the `'DeltaOffset'` parameter field is deprecated. This parameter may be removed in a future release.

Delta offset, specified as 0, 1, or 2. Optional. (*delta_offset*)

Data Types: `double`

`CyclicShifts` — Number of cyclic shifts for format 1 resource blocks

0 (default) | Optional | 0...7

Number of cyclic shifts for format 1 resource blocks, in RBs, specified as a nonnegative scalar integer between 0 and 7. Optional. This can be used in a mixture of format 1 and format 2 PUCCH. (*N1cs*)

Data Types: `double`

Data Types: `struct`

`oack` — Number of uncoded HARQ-ACK bits

0 (default) | nonnegative integer vector

Uncloded HARQ-ACK bits, specified as a nonnegative integer vector. The corresponding PUCCH transmission is of format 1, 1a, or 1b.

Data Types: `double`

`sym` — Symbols of each antenna

complex numeric matrix

Symbols for each antenna, specified as complex numeric matrix. The number of columns should match the number of PUCCH resource indices specified in the structure, chs.

Example: 0.25881 + 0.9659i

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`ack` — Hybrid ARQ indicator values

logical column vector

Hybrid ARQ indicator values, specified as a logical column vector. This vector is obtained by performing PUCCH Format 1 decoding of the complex matrix, sym.

Data Types: `logical`

See Also

`ltePUCCH1` | `ltePUCCH1DRS` | `ltePUCCH1DRSIndices` | `ltePUCCH1Indices` | `ltePUCCH2Decode` | `ltePUCCH3Decode`

ltePUCCH1DRS

PUCCH format 1 demodulation reference signal

Syntax

```
[seq,info] = ltePUCCH1DRS(ue,chs)
```

Description

`[seq,info] = ltePUCCH1DRS(ue,chs)` returns a complex matrix, seq, containing PUCCH Format 1 demodulation reference signal (DRS) values and information structure array, info, for UE-specific settings, ue, and channel transmission configuration, chs. The symbols for each antenna are in the columns of seq, with the number of columns determined by the number of PUCCH resource indices specified in the chs structure.

Examples

Generate PUCCH Format 1 DRS

Generate PUCCH format 1 demodulation reference signal (DRS) values for UE-specific settings.

```
ue = struct('NCellID',1,'NSubframe',0);
drsSeq = ltePUCCH1DRS(ue,struct('ResourceIdx',0));
drsSeq(1:4)
```

```
   0.7071 + 0.7071i
   0.7071 + 0.7071i
  -0.7071 + 0.7071i
  -0.7071 + 0.7071i
```

Input Arguments

ue — UE-specific settings

structure

UE-specific settings, specified as a structure. ue contains the following fields.

`NCellID` — Physical layer cell identity

0 (default) | nonnegative scalar integer

Physical layer cell identity number, specified as a nonnegative scalar integer.

Example: 4

Data Types: `double`

`NSubframe` — Subframe number

0 (default) | nonnegative scalar integer

Position reference signal subframe number, specified as a nonnegative scalar integer.

Example: 8

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length for uplink channels

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink channels, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`Hopping` — Uplink frequency hopping

`'Off'` (default) | Optional | `'Group'`

Uplink frequency hopping, specified as `'Off'` or `'Group'`. Optional.

Data Types: `char`

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure. chs contains the following fields.

`ResourceIdx` — PUCCH resource indices

0...2047

PUCCH resource indices, specified as nonnegative numeric vector with values between 0 and 2047. There is one index for each transmission antenna. These indices determine the cyclic shift and orthogonal cover used for transmission. This input argument is optional.

Example: 78

Data Types: `double`

`DeltaShift` — Delta shift

1 (default) | Optional | 2 | 3

Delta shift, specified as a 1, 2, or 3. Optional. (*delta_shift*)

Data Types: `double`

`DeltaOffset` — Delta offset

0 (default) | Deprecated | Optional | 1 | 2

Note: Warning: The use of the `'DeltaOffset'` parameter field is deprecated. This parameter may be removed in a future release.

Delta offset, specified as 0, 1, or 2. Optional. (*delta_offset*)

Data Types: `double`

`CyclicShifts` — Number of cyclic shifts for format 1 resource blocks

0 (default) | Optional | 0...7

Number of cyclic shifts for format 1 resource blocks, in RBs, specified as nonnegative scalar integer between 0 and 7. Optional. This parameter can be used in a mixture of format 1 and format 2 PUCCH. (*N1cs*)

Data Types: `double`

Output Arguments

`seq` — PUCCH format 1 DRS values

complex numeric matrix

PUCCH format 1 DRS values, returned as a complex numeric matrix. The symbols for each antenna are in the columns of seq, with the number of columns determined by the number of PUCCH resource indices specified in the structure, chs.

Data Types: `double`
Complex Number Support: Yes

`info` — PUCCH format 1 DRS information

structure array

PUCCH format 1 DRS information, returned as a structure array. info contains the following fields.

`Alpha` — Reference signal cyclic shift for each OFDM symbol

two-column row vector

Reference signal cyclic shift for each OFDM symbol, returned as a two-column row vector.

Data Types: `double`

`SeqGroup` — PUCCH base sequence group number for each slot

two-column row vector

PUCCH base sequence group number for each slot, returned as two-column row vector. (u)

Data Types: `double`

`SeqIdx` — PUCCH base sequence number for each slot

two-column row vector

PUCCH base sequence number for each slot, returned as two-column row vector. (v)

Data Types: `double`

`NResourceIdx` — PUCCH resource indices for each slot

row vector

PUCCH resource indices for each slot, returned as two column row vector. (n')

Data Types: `double`

`NCellCyclicShift` — Cell-specific cyclic shift for each OFDM symbol

row vector

Cell-specific cyclic shift for each OFDM symbol, returned as row vector. (*ncell_cs*)

Data Types: `double`

`OrthSeqIdx` — Orthogonal sequence index for each slot

two-column row vector

Orthogonal sequence index for each slot, returned as two-column row vector. (*n_oc*)

Data Types: `double`

`Symbols` — Modulated data symbols

row vector

Modulated data symbols, returned as row vector. There is one element for each OFDM symbol.

Data Types: `double`
Complex Number Support: Yes

`OrthSeq` — Orthogonal sequence for each slot

numeric matrix

Orthogonal sequence for each slot, returned as a numeric matrix. (*wbar*)

Data Types: `double`
Complex Number Support: Yes

See Also

`ltePUCCH1` | `ltePUCCH1Decode` | `ltePUCCH1DRSIndices` | `ltePUCCH1Indices` | `ltePUCCH2DRS` | `ltePUCCH3DRS`

ltePUCCH1DRSIndices

PUCCH format 1 DRS resource element indices

Syntax

```
[ind,info] = ltePUCCH1DRSIndices(ue,chs)
[ind,info] = ltePUCCH1DRSIndices(ue,chs,opts)
```

Description

`[ind,info] = ltePUCCH1DRSIndices(ue,chs)` returns a matrix of resource element indices given the UE-specific settings structure ue and channel transmission configuration chs. It returns a matrix of resource element (RE) indices and information structure array info for the demodulation reference signal (DRS) associated with PUCCH format 1 transmission. By default the indices are returned in 1-based linear indexing form that can directly index elements of a resource matrix. These indices are ordered as the PUCCH format 1 DRS modulation symbols should be mapped. Alternative indexing formats can also be generated. The indices for each antenna are in the columns of ind, with the number of columns determined by the number of PUCCH Resource Indices specified.

`[ind,info] = ltePUCCH1DRSIndices(ue,chs,opts)` allows control of the format of the returned indices through a cell array opts of option strings.

Examples

Generate PUCCH Format 1 DRS Indices

Generate PUCCH format 1 demodulation reference signal (DRS) resource element (RE) indices for 1.4 MHz bandwidth, PUCCH resource index 0, and using default values for all other parameters.

```
ind = ltePUCCH1DRSIndices(struct('NULRB',6),struct('ResourceIdx',0));
ind(1:4)
```

```
    145
```

```
146
147
148
```

Input Arguments

ue — UE-specific settings

structure

ue is a structure having the following fields.

NULRB — Number of uplink resource blocks

0 (default) | nonnegative scalar integer

Number of uplink resource blocks, specified as a nonnegative scalar integer.

Example: 4,5,...

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as `'Normal'` or `'Extended'`, for uplink channels. Optional.

Data Types: `char`

chs — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure. chs contains the following fields.

ResourceIdx — PUCCH resource indices

(default) | 0...2047

PUCCH resource indices, specified as nonnegative vector, with one element for each transmission antenna. These indices determine the cyclic shift and orthogonal cover used for transmission. This input argument is optional.

Example: 78

Data Types: `double`

`ResourceSize` — Size of resources allocated to PUCCH format 2

0 (default) | Optional | 0...63

Size of resources allocated to PUCCH format 2, specified as nonnegative scalar integer between 0 and 63. This parameter affects location of this transmission. Optional. (*N2RB*)

Data Types: `double`

`DeltaShift` — Delta shift

1 (default) | Optional | 2 | 3

Delta shift, specified as 1, 2, or 3. Optional. (*delta_shift*)

Data Types: `double`

`CyclicShifts` — Number of cyclic shifts for format 1 resource blocks

0 (default) | Optional | nonnegative scalar integer

Number of cyclic shifts for format 1 resource blocks, in RBs, specified as a nonnegative scalar integer. Optional. This parameter can be used in a mixture of format 1 and format 2 PUCCH. (*N1cs*)

Example: 0...7

Data Types: `double`

Data Types: `struct`

`opts` — Options to control format of returned indices

string | cell array of strings

Options to control format of returned indices, specified as a string or a cell array of strings. It can contain the following option strings.

`Indexing style` — Indexing style of returned indices

`'ind'` (default) | `'sub'`

Indexing style of returned indices, specified as `'ind'` or `'sub'`. If `'ind'`, the returned indices are in linear index form. If `'sub'`, the returned indices are in [*subcarrier*, *symbol*, *antenna*] subscript form.

Data Types: `char`

`Index base` — Index base of returned indices

`'1based'` (default) | `'0based'`

Index base of returned indices, specified as `'1based'` or `'0based'`. This parameter controls whether the returned indices are 1-based or 0-based.

Data Types: `char`

Data Types: `char` | `cell`

Output Arguments

`ind` — PUCCH format 1 resource element indices

integer column vector | 3-column integer matrix

PUCCH format 1 resource element indices, returned as an integer column vector or a 3-column integer matrix. By default, the indices are returned in 1-based linear indexing form, which can be used to directly index elements of a resource grid.

Example: 145,146.147....

Data Types: `uint32`

`info` — PUCCH format 1 DRS information

structure array

PUCCH format 1 DRS information, returned as a structure array. info can contain the following fields.

`PRBSet` — Indices occupied by PRB in each slot of subframe

numeric row vector

Indices occupied by PRB in each slot of subframe, returned as a numeric row vector. By default, the indices are 0-based.

Data Types: `double`

`RBIdx` — PUCCH logical resource block index

numeric scalar

PUCCH logical resource block index, returned as a numeric scalar. (m)

Data Types: `double`

Data Types: `struct`

See Also

`ltePUCCH1` | `ltePUCCH1Decode` | `ltePUCCH1DRS` | `ltePUCCH1Indices` | `ltePUCCH2DRSIndices` | `ltePUCCH3DRSIndices`

ltePUCCH1Indices

PUCCH format 1 resource element indices

Syntax

```
[ind,info] = ltePUCCH1Indices(ue,chs)
[ind,info] = ltePUCCH1Indices(ue,chs,opts)
```

Description

`[ind,info] = ltePUCCH1Indices(ue,chs)` returns a matrix of resource element indices given the UE-specific settings structure ue and channel transmission configuration chs. ue and chs must be structures. It returns a matrix of resource element (RE) indices and information structure array info for the Physical Uplink Control Channel (PUCCH) format 1. By default, the indices are returned in 1-based linear indexing form that can directly index elements of a resource matrix. These indices are ordered as the PUCCH format 1 modulation symbols should be mapped. Alternative indexing formats can also be generated. The indices for each antenna are in the columns of ind, with the number of columns determined by the number of PUCCH Resource Indices specified.

`[ind,info] = ltePUCCH1Indices(ue,chs,opts)` allows control of the format of the returned indices through a cell array of option strings, opts.

Examples

Create PUCCH Format 1 Indices

Create PUCCH format 1 resource element indices for 1.4 MHz bandwidth, PUCCH Resource Index 0.

Set the number of uplink RBs to 6 and the resource index to 0. Use default values for all other parameters.

```
ind = ltePUCCH1Indices(struct('NULRB',6),struct('ResourceIdx',0));
```

```
ind(1:4)
```

```
           1
           2
           3
           4
```

Input Arguments

ue — UE-specific settings
structure

ue is a structure having the following fields.

NULRB — Number of uplink resource blocks
0 (default) | nonnegative scalar integer

Number of uplink resource blocks, specified as nonnegative scalar integer.

Example: 4

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length for uplink channels
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink channels, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

Shortened — Shorten subframe flag
0 (default) | Optional | 1

Shorten subframe flag, specified as a 0 or 1. Optional. This parameter is required for subframes with possible SRS transmission. If 1, the last symbol of the subframe is not used.

Data Types: `double`

Data Types: `struct`

chs — Channel transmission configuration
structure

Channel transmission configuration, specified as a structure. chs contains the following fields.

`ResourceIdx` — PUCCH resource indices

nonnegative integer vector (0...2047)

PUCCH resource indices, specified as nonnegative vector. There is one index for each transmission antenna. These indices determine the cyclic shift and orthogonal cover used for transmission. This input argument is optional.

Example: 78

Data Types: `double`

`ResourceSize` — Size of resources allocated to PUCCH format 2

0 (default) | Optional | 0...63

Size of resources allocated to PUCCH format 2, specified as nonnegative scalar integer between 0 and 63. This parameter affects location of this transmission. Optional. (*N2RB*)

Data Types: `double`

`DeltaShift` — Delta shift

1 (default) | Optional | 2 | 3

Delta shift, specified as 1, 2, or 3. Optional. (*delta_shift*)

Data Types: `double`

`CyclicShifts` — Number of cyclic shifts for format 1 resource blocks

0 (default) | Optional | nonnegative scalar integer (0...7)

Number of cyclic shifts for format 1 resource blocks, in RBs, specified as a nonnegative scalar integer. This parameter can be used in a mixture of format 1 and format 2 PUCCH. Optional. (*N1cs*)

Data Types: `double`

`opts` — Format options for control of the returned indices

string | cell array of strings

Format options for control of the returned indices, specified as a string or a cell array of strings. It may include the following option strings.

`Indexing style` — Controls the indexing style of the indices

`'ind'` (default) | `'sub'`

Cell array option, specified as `'ind'` or `'sub'`, returns the indices linear index form or [sub-carrier, symbol, antenna] subscript form.

Data Types: `char`

`Index base` — Controls the index base of the indices

`'1based'` (default) | `'0based'`

Cell array option, specified as `1based` or `0based`, returns the indices as 1-based or 0-based.

Data Types: `char`

Output Arguments

`ind` — Resource element indices

linear matrix

Resource element (RE) indices, returned as 1 linear matrix. By default the indices are returned in 1-based linear indexing form that can directly index elements of a resource matrix.

Example: 1,2,3,4...

Data Types: `double`

`info` — PUCCH format 1 information

structure array

PUCCH format 1 information, returned as a structure array. info is a structure array having the following fields.

`PRBSet` — Set of PRB indices

one- or two-column matrix

Set of PRB indices, returned as a one- or two-column matrix. This field contains the physical resource block indices (PRBs) corresponding to the resource allocations. If a column vector, the resource allocation is the same in both slots of the subframe; the two-column matrix can be used to specify differing PRBs for each slot in a subframe. The PRB indices are 0-based.

Example: [0,5]

Data Types: `double`

`RBIdx` — PUCCH logical resource block index

nonnegative scalar integer

PUCCH logical resource block index (m).

Example: 0

Data Types: `double`

See Also

`ltePUCCH1` | `ltePUCCH1Decode` | `ltePUCCH1DRS` | `ltePUCCH1DRSIndices` | `ltePUCCH2Indices` | `ltePUCCH3Indices`

ltePUCCH2

Physical uplink control channel format 2

Syntax

```
[sym,info] = ltePUCCH2(ue,chs,bits)
```

Description

`[sym,info] = ltePUCCH2(ue,chs,bits)` returns a complex matrix sym containing Physical Uplink Control Channel Format 2 values and information structure array info for UE-specific settings ue, channel transmission configuration chs, and vector of coded CQI/PMI or RI bits bits. The symbols for each antenna are in the columns of sym, with the number of columns determined by the number of PUCCH Resource Indices specified in the structure chs.

bits is a vector of coded CQI/PMI or RI bits (coded UCI), formed by performing UCI encoding of a bit vector representing the CQI/PMI or RI information fields described in section 5.2.3.3 of [2]. This coded bit vector must be 20 bits long. This vector is denoted "block of bits $b(0),...,b(19)$" in section 5.4.2 of [1]. If M_{bit} is 21 or 22, corresponding to PUCCH format 2a or 2b, respectively, as described in table 5.4-1 of [1], the further bits, $b(20),...,b(M_{bit}-1)$, should be provided as input to the `ltePUCCH2DRS` function for transmission. An M_{bit} value of 20 corresponds to PUCCH format 2, with no additional bits being transmitted on the PUCCH format 2 DRS.

Examples

Create PUCCH Format 2 Symbols

Generate PUCCH format 2 symbol values, setting `NCellID` to 1 and `NSubframe` to 0.

```
ue = struct('NCellID',1,'NSubframe',0,'RNTI',1);
sym = ltePUCCH2(ue,struct('ResourceIdx',0),ones(20,1));
sym(1:4)
```

```
   0.0000 + 1.0000i
  -0.5000 - 0.8660i
```

```
-0.5000 + 0.8660i
-0.0000 - 1.0000i
```

Input Arguments

ue — UE-specific settings

structure

UE-specific settings, specified as a structure. ue contains the following fields.

NCellID — Physical layer cell identity

nonnegative scalar integer

Physical layer cell identity, specified as a nonnegative scalar integer.

Example: 4

Data Types: `double`

NSubframe — Subframe number

nonnegative scalar integer

Subframe number, specified as nonnegative scalar integer.

Example: 8

Data Types: `double`

RNTI — Radio network temporary identifier

numeric scalar

Radio network temporary identifier (16-bit), specified as a numeric scalar.

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length for uplink channels

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink channels, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

Hopping — Uplink frequency hopping

`'Off'` (default) | Optional | `'Group'`

Uplink frequency hopping, specified as `'Off'` or `'Group'`. Optional.

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure. chs contains the following fields.

`ResourceIdx` — PUCCH resource indices

0...1185

PUCCH resource indices, specified as a nonnegative vector with one element for each transmission antenna. These indices determine the cyclic shift and orthogonal cover used for transmission. This input argument is optional.

Example: 78

Data Types: `double`

`ResourceSize` — Size of resources allocated to PUCCH format 2

0 (default) | Optional | 0...63

Size of resources allocated to PUCCH format 2, specified as nonnegative scalar integer between 0 and 63. Optional. This parameter affects location of this transmission. (*N2RB*)

Data Types: `double`

`CyclicShifts` — Number of cyclic shifts for format 1 resource blocks

0 (default) | Optional | nonnegative scalar integer (0...7)

Number of cyclic shifts for format 1 resource blocks, in RBs, specified as a nonnegative scalar integer between 0 and 7. Optional. This parameter can be used in a mixture of format 1 and format 2 PUCCH. (*N1cs*)

Example: 7

Data Types: `double`

`bits` — Coded CQI/PMI or RI bits

vector

Coded CQI/PMI or RI bits, specified as vector, formed by performing UCI encoding of a bit vector representing the CQI/PMI or RI information fields. This coded bit vector must be 20 bits long. This coded bit vector must be 20 bits long. This vector is denoted

"block of bits $b(0),...,b(19)$" in section 5.4.2 of [1]. If M_{bit} is 21 or 22, corresponding to PUCCH format 2a or 2b, respectively, as described in table 5.4-1 of [1], the further bits, $b(20),...,b(M_{bit}-1)$, should be provided as input to the `ltePUCCH2DRS` function for transmission. An M_{bit} value of 20 corresponds to PUCCH format 2, with no additional bits being transmitted on the PUCCH format 2 DRS.

Data Types: `logical | double`

Output Arguments

`sym` — PUCCH format 2 symbols

complex numeric column vector

PUCCH format 2 symbols, returned as complex numeric column vector.

Example: 0.7071 + 0.7071i

Data Types: `double`
Complex Number Support: Yes

`info` — PUCCH format 2 information

structure array

PUCCH format 2 information, returned as a structure array. info is a structure array having the following fields.

`Alpha` — Reference signal cyclic shift for each OFDM symbol

two-column row vector

Reference signal cyclic shift for each OFDM symbol, returned as two-column row vector.

Data Types: `double`

`SeqGroup` — PUCCH base sequence group number for each slot

two-column row vector

PUCCH base sequence group number, returned as a two-column row vector. (u)

Data Types: `double`

`SeqIdx` — PUCCH base sequence group number indices for each slot

two-column row vector

PUCCH base sequence group number indices for each slot, returned as a two-column row vector. (v)

Data Types: `double`

`NResourceIdx` — PUCCH resource indices for each slot

two-column row vector

PUCCH resource indices for each slot, returned as two-column row vector. (n')

Data Types: `double`

`NCellCyclicShift` — Cell-specific cyclic shift

numeric row vector

Cell-specific cyclic shift, returned as a numeric row vector. There is one element for each OFDM symbol. (*ncell_cs*)

Data Types: `double`

`Symbols` — Modulated data symbols

complex numeric row vector

Modulated data symbols, returned as a complex numeric row vector. There is one element for each OFDM symbol. (*d(0)*)

Data Types: `double`
Complex Number Support: Yes

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

[2] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`ltePUCCH1` | `ltePUCCH2Decode` | `ltePUCCH2DRS` | `ltePUCCH2DRSIndices` | `ltePUCCH2Indices` | `ltePUCCH3` | `lteUCIEncode`

ltePUCCH2DRS

PUCCH format 2 demodulation reference signal

Syntax

```
[seq,info] = ltePUCCH2DRS(ue,chs,ack)
```

Description

`[seq,info] = ltePUCCH2DRS(ue,chs,ack)` returns a complex matrix seq containing PUCCH Format 2 Demodulation Reference Signal (DRS) values and information structure array, info, for UE-specific settings, ue, channel transmission configuration, chs, and vector of hybrid ARQ (HARQ) indicator values, ack. The symbols for each antenna are in the columns of seq, with the number of columns determined by the number of PUCCH Resource Indices specified in the structure chs.

ack is a vector containing 0, 1, or 2 hybrid ARQ indicator values; the corresponding PUCCH transmission is of format 2, 2a, or 2b, respectively. The standard does not support format 2a or 2b transmission with extended cyclic prefix; in these cases, the function returns an empty matrix. This vector is denoted "block of bits $b(20),...,b(M_{\text{bit}}-1)$" in section 5.4.2 of [1]. An M_{bit} value of 20, 21, or 22 corresponds to PUCCH format 2, 2a, or 2b, respectively, as described in table 5.4-1 of [1].

When configured for format 2a or 2b transmission with extended cyclic prefix, the info structure contains all fields, but each field is empty.

Examples

Generate PUCCH Format 2 DRS

Generate PUCCH format 2 demodulation reference signal (DRS) values for UE-specific settings.

```
ue = struct('NCellID',1,'NSubframe',0);
pucchSeq = ltePUCCH2DRS(ue,struct('ResourceIdx',0),[]);
pucchSeq(1:4)
```

```
   0.7071 + 0.7071i
  -0.2588 + 0.9659i
  -0.2588 - 0.9659i
   0.7071 - 0.7071i
```

Input Arguments

ue — UE-specific settings
structure

UE-specific settings, specified as a structure. ue contains the following fields.

NCellID — Physical layer cell identity
nonnegative scalar integer

Physical layer cell identity, specified as a nonnegative scalar integer.

Example: 4

Data Types: `double`

NSubframe — Subframe number
nonnegative scalar integer

Subframe number, specified as nonnegative scalar integer.

Example: 8

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length for uplink channels
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink channels, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

Hopping — Uplink frequency hopping
`'Off'` (default) | Optional | `'Group'`

Uplink frequency hopping, specified as `'Off'` or `'Group'`. Optional.

Data Types: `char`

Data Types: `struct`

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure. chs contains the following fields.

`ResourceIdx` — PUCCH resource indices

nonnegative vector (0...2047)

PUCCH resource indices, specified as a nonnegative vector with one element for each transmission antenna. These determine the cyclic shift and orthogonal cover used for transmission. (*n2_pucch*). This input argument is optional.

Example: 78

Data Types: `double`

`ResourceSize` — Size of resources allocated to PUCCH format 2

0 (default) | Optional | 0...63

Size of resources allocated to PUCCH format 2, specified as nonnegative scalar integer. Optional. This parameter affects location of this transmission. (*N2RB*)

Data Types: `double`

`CyclicShifts` — Number of cyclic shifts for format 1 resource blocks

0 (default) | Optional | nonnegative scalar integer (0...7)

Number of cyclic shifts for format 1 resource blocks, in RBs, specified as a nonnegative scalar integer between 0 and 7. Optional. This can be used in a mixture of format 1 and format 2 PUCCH. (*N1cs*)

Example: 7

Data Types: `double`

`ack` — Hybrid ARQ indicator values

nonnegative integer vector (0, 1, 2)

Hybrid ARQ indicator values, specified as nonnegative integer vector with values of 0, 1, or 2. The corresponding PUCCH transmission is of format 2, 2a, or 2b, respectively. The standard does not support format 2a or 2b transmission with extended cyclic prefix; in these cases, the function returns an empty matrix. This vector is denoted “block of

bits $b(20),...,b(M_{bit}-1)$" in section 5.4.2 of [1]. An M_{bit} value of 20, 21, or 22 corresponds to PUCCH format 2, 2a, or 2b, respectively, as described in table 5.4-1 of [1].

Example: 0,1,2

Data Types: `double`

Output Arguments

`seq` — PUCCH Format 1 DRS sequence

complex numeric matrix

PUCCH Format 1 DRS sequence, returned as a complex numeric matrix. The symbols for each antenna are in the columns of seq, with the number of columns determined by the number of PUCCH Resource Indices specified in the structure chs.

Data Types: `double`
Complex Number Support: Yes

`info` — PUCCH format 2 information

structure array

PUCCH format 2 information, returned as a structure array. info contains the following fields.

`Alpha` — Reference signal cyclic shift for OFDM symbol

two-column row vector

Reference signal cyclic shift for OFDM symbol, returned as two-column row vector, for each OFDM symbol.

Data Types: `double`

`SeqGroup` — PUCCH base sequence group number for each slot

two-column row vector

PUCCH base sequence group number for each slot, returned as two-column row vector. (u)

Data Types: `double`

`SeqIdx` — PUCCH base sequence group number indices for each slot

two-column row vector

PUCCH base sequence group number indices for each slot, returned as two-column row vector. (v)

Data Types: `double`

`NResourceIdx` — PUCCH resource indices
two-column row vector

PUCCH resource indices, returned as two-column row vector, for each slot. (n')

Data Types: `double`

`NCellCyclicShift` — Cell-specific cyclic shift
row vector

Cell-specific cyclic shift for each OFDM symbol, returned as row vector. (*ncell_cs*)

Data Types: `double`

`Symbols` — Modulated data symbols
row vector

Modulated data symbols, returned as a row vector for each OFDM symbol. (*d(0)*)

Data Types: `double`

`OrthSeq` — Orthogonal sequence of each slot
4-by-2 numeric matrix

Orthogonal sequence of each slot, returned as a 4-by-2 numeric matrix.

Data Types: `double`

Data Types: `struct`

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`ltePUCCH1DRS` | `ltePUCCH2` | `ltePUCCH2Decode` | `ltePUCCH2DRSDecode` | `ltePUCCH2DRSIndices` | `ltePUCCH2Indices` | `ltePUCCH3DRS`

ltePUCCH2DRSDecode

PUCCH format 2 DRS decoding

Syntax

```
ack = ltePUCCH2DRSDecode(ue,chs,oack,sym)
```

Description

`ack = ltePUCCH2DRSDecode(ue,chs,oack,sym)` returns a vector of hybrid automatic repeat request (HARQ) indicator values, ack, obtained by performing PUCCH format 2 DRS decoding of the complex matrix, sym. The decoder uses a maximum likelihood (ML) approach, assuming that sym has already been equalized, to best restore the originally transmitted complex values. The symbols for each antenna are in the columns of sym, and the number of columns should match the number of PUCCH resource indices specified in the chs structure.

oack specifies the number of HARQ indicator values expected. They can be either 1 or 2, for PUCCH format 2A or 2B, respectively.

ack is a column vector containing oack HARQ indicator (HI) values.

Examples

Decode PUCCH Format 2A DRS

Decode a PUCCH format 2A DRS from a synchronized and equalized resource array.

First, configure the UE and PUCCH format 2 parameter structures.

```
ue = struct('NULRB',6,'NCellID',0,'NSubframe',0,'Hopping','Off');
pucch2 = struct('ResourceIdx',0);
```

For the transmitter, create the PUCCH format 2A DRS.

```
rgrid = lteULResourceGrid(ue);
```

```
drsIndices = ltePUCCH2DRSIndices(ue,pucch2);
txAck = [1;0];
rgrid(drsIndices) = ltePUCCH2DRS(ue,pucch2,txAck);
```

On the receiver side, decode the PUCCH format 2 DRS symbols.

```
drsSymbols = rgrid(drsIndices);
rxAck = ltePUCCH2DRSDecode(ue,pucch2,length(txAck),drsSymbols)
```

```
     1
     0
```

Input Arguments

ue — UE-specific settings

structure

UE-specific settings, specified as a structure. ue can contain the following fields.

NCellID — Physical layer cell identity

nonnegative scalar integer

Physical layer cell identity number, specified as a nonnegative scalar integer.

Example: 4

Data Types: `double`

NSubframe — Subframe number

nonnegative scalar integer

Subframe number, specified as a nonnegative scalar integer.

Example: 8

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length for uplink channels

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink channels, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`Hopping` — Uplink frequency hopping

`'Off'` (default) | Optional | `'Group'`

Uplink frequency hopping, specified as `'Off'` or `'Group'`. Optional.

Data Types: `char`

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure. chs contains the following fields.

`ResourceIdx` — PUCCH resource indices

nonnegative vector (0...2047)

PUCCH resource indices, specified as nonnegative vector with one element for each transmission antenna and values ranging from 0 to 2047. These indices determine the cyclic shift and orthogonal cover used for transmission. This input argument is optional.

Example: 78

Data Types: `double`

`ResourceSize` — Size of resources allocated to PUCCH format 2

0 (default) | Optional | 0...63

Size of resources allocated to PUCCH format 2, specified as nonnegative scalar integer between 0 and 63. Optional. This parameter affects location of this transmission. *N2RB*)

Data Types: `double`

`CyclicShifts` — Number of cyclic shifts for format 1 resource blocks

0 (default) | Optional | nonnegative scalar integer (0...7)

Number of cyclic shifts for format 1 resource blocks, in RBs, specified as a nonnegative scalar integer between 0 and 7. Optional. This parameter can be used in a mixture of format 1 and format 2 PUCCH. (*N1cs*)

Example: 7

Data Types: `double`

`oack` — Number of uncoded HARQ-ACK bits

0 (default) | nonnegative integer vector

Number of uncoded HARQ-ACK bits, specified as nonnegative integer vector. The corresponding PUCCH transmission is of format 1, 1A or 1B

Data Types: `double`

`sym` — Symbols of each antenna

complex numeric matrix

Symbols for each antenna, specified as complex numeric matrix. The number of columns should match the number of PUCCH resource indices specified in the chs structure.

Example: 0.25881 + 0.9659i

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`ack` — Hybrid ARQ indicator values

logical column vector

Hybrid ARQ indicator values, returned as a logical column vector. This output is obtained by performing PUCCH format 1 decoding of the complex matrix, sym.

Data Types: `logical`

See Also

`ltePUCCH1DRS` | `ltePUCCH2` | `ltePUCCH2Decode` | `ltePUCCH2DRS` | `ltePUCCH2DRSIndices` | `ltePUCCH2Indices` | `ltePUCCH2PRBS` | `ltePUCCH3DRS`

ltePUCCH2DRSIndices

PUCCH format 2 DRS resource element indices

Syntax

```
[ind,info] = ltePUCCH2DRSIndices(ue,chs)
[ind,info] = ltePUCCH2DRSIndices(ue,chs,opts)
```

Description

`[ind,info] = ltePUCCH2DRSIndices(ue,chs)` returns a matrix of resource element indices given the UE-specific settings structure ue, and channel transmission configuration chs. It returns a matrix of resource element (RE) indices and information structure array info for the Demodulation Reference Signal (DRS) associated with PUCCH Format 2 transmission. By default the indices are returned in 1-based linear indexing form that can directly index elements of a resource matrix. These indices are ordered as the PUCCH format 2 DRS modulation symbols should be mapped. Alternative indexing formats can also be generated. The indices for each antenna are in the columns of ind, with the number of columns determined by the number of PUCCH Resource Indices specified.

`[ind,info] = ltePUCCH2DRSIndices(ue,chs,opts)` allows control of the format of the returned indices through a cell array of option strings, opts.

Examples

Generate PUCCH Format 2 DRS Indices

Generate PUCCH format 2 demodulation reference signal (DRS) indices for 1.4 MHz bandwidth, PUCCH resource index 0.

```
ind = ltePUCCH2DRSIndices(struct('NULRB',6),struct('ResourceIdx',0));
ind(1:4)
```

```
74
75
76
```

Input Arguments

ue — UE-specific settings

structure

UE-specific settings, specified as a structure. ue contains the following fields.

NULRB — Number of uplink resource blocks

nonnegative scalar integer

Number of uplink resource blocks, specified as a nonnegative scalar integer.

Example: 4

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length for uplink channels

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink channels, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

chs — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure. chs contains the following fields.

ResourceIdx — PUCCH resource indices

nonnegative vector (0...1185)

PUCCH resource indices, specified as nonnegative vector with one element for each transmission antenna. These indices determine the cyclic shift and orthogonal cover used for transmission. (*n2_pucch*). This input argument is optional.

Example: 78

Data Types: `double`

`opts` — Format options for control of the returned indices

string | cell array of strings

Format options for control of the returned indices, specified as a string or a cell array of strings. It can contain the following option strings.

`Indexing style` — Indexing style of the returned indices

`'ind'` (default) | `'sub'`

Indexing style of the returned indices, specified as `'ind'` or `'sub'`. If `'ind'`, the returned indices are in linear index form. If `'sub'`, the returned indices are in `[subcarrier, symbol, antenna]` subscript form.

Data Types: `char`

`Index base` — Index base of the returned indices

`'1based'` (default) | `'0based'`

Index base of the returned indices, specified as `'1based'` or `'0based'`. If `'1based'`, the lowest index is 1. If `'0based'`, the lowest index is 0.

Data Types: `char`

Output Arguments

`ind` — Resource element indices

linear matrix

Resource element (RE) indices, returned as 1 linear matrix. By default the indices are returned in 1-based linear indexing form that can directly index elements of a resource matrix.

Example: 145,146.147....

Data Types: `double`

`info` — PUCCH format 2 DRS information

structure array

PUCCH format 2 DRS information, returned as a structure array. info is a structure array having these fields:

`PRBSet` — PRBs occupied by indices in each slot of the subframe
numeric row vector

PRBs occupied by indices in each slot of the subframe, returned as a numeric row vector.

Data Types: `double`

`RBIdx` — PUCCH logical resource block index
numeric scalar

PUCCH logical resource block index, returned as a numeric scalar. (*m*)

Data Types: `double`

Data Types: `struct`

See Also

`ltePUCCH1DRSIndices` | `ltePUCCH2` | `ltePUCCH2Decode` | `ltePUCCH2DRS` | `ltePUCCH2DRSDecode` | `ltePUCCH2Indices` | `ltePUCCH2PRBS` | `ltePUCCH3DRSIndices`

ltePUCCH2Decode

Physical uplink control channel format 2 decoding

Syntax

```
out = ltePUCCH2Decode(ue,chs,sym)
```

Description

`out = ltePUCCH2Decode(ue,chs,sym)` performs decoding of the PUCCH format 2 given UE-specific settings ue and channel transmission configuration chs. out is a soft bit vector consisting of 20 bits, formed by decoding complex symbol matrix sym, performing demodulation with the PUCCH format 2 reference sequence, QPSK demodulation, and descrambling. The symbols for each antenna are in the columns of sym, and the number of columns should match the number of PUCCH Resource Indices specified in the structure, chs.

Examples

Decode PUCCH Format 2 Signal

Decode a PUCCH format 2 signal from an equalized resource array, `grid`.

First, create a UE configuration structure, `ue`, and a PUCCH configuration structure, `pucch2`.

```
ue = struct('NULRB',6,'NCellID',0,'NSubframe',0,'RNTI',1);
pucch2 = struct('ResourceIdx',0);
```

For the transmitter, create a PUCCH format 2 resource grid.

```
rgrid = lteULResourceGrid(ue);
pucch2Indices = ltePUCCH2Indices(ue,pucch2);
tx = [1;0;0;0;0;1];
encoded = lteUCIEncode(tx);
rgrid(pucch2Indices) = ltePUCCH2(ue,pucch2,encoded);
```

On the receiver side, decode the PUCCH format 2 signal contained in equalized resource array, `grid`. Also decode the UCI bits.

```
rx = ltePUCCH2Decode(ue,pucch2,rgrid(pucch2Indices));
decoded = lteUCIDecode(rx,length(tx))
```

```
     1
     0
     0
     0
     0
     1
```

Input Arguments

`ue` — UE-specific settings
structure

ue is a structure having the following fields.

`NCellID` — Cell identity number
0 (default)

Physical layer cell identity number, specified as a nonnegative scalar integer.

Example: 4

Data Types: `double`

`NSubframe` — Subframe number
0 (default)

Position reference signal subframe number, specified as a nonnegative scalar integer.

Example: 8

Data Types: `double`

`RNTI` — Radio network temporary identifier (16-bit)
scalar integer

Radio network temporary identifier (16-bit), specified as a scalar integer.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length for uplink channels

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink channels, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`Hopping` — Uplink frequency hopping

`'Off'` (default) | Optional | `'Group'`

Uplink frequency hopping, specified as `'Off'` or `'Group'`. Optional.

Data Types: `char`

Data Types: `struct`

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure. chs contains the following fields.

`ResourceIdx` — PUCCH resource indices

0...2047

PUCCH resource indices, specified as nonnegative vector, one for each transmission antenna. These determine the cyclic shift and orthogonal cover used for transmission. This input argument is optional.

Example: 78

Data Types: `double`

`ResourceSize` — Size of resources allocated

0 (default) | Optional | 0...63

Size of resources allocated to `PUCCH` Format 2 `(N2RB)`, specified as nonnegative scalar integer. This parameter affects location of this transmission. Optional.

Data Types: `double`

`CyclicShifts` — Number of cyclic shifts for format 1 resource blocks

0 (default) | Optional | nonnegative scalar integer (0...7)

Number of cyclic shifts for format 1 resource blocks, in RBs, specified as a nonnegative scalar integer between 0 and 7. Optional. This parameter can be used in a mixture of format 1 and format 2 PUCCH. (*N1cs*)

Example: 7

Data Types: `double`

`sym` — Symbols of each antenna

complex numeric matrix

Symbols for each antenna, specified as complex numeric matrix. The number of columns should match the number of PUCCH resource indices specified in the channel transmission configuration structure, chs.

Example: 0.25881 + 0.9659i

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`out` — PUCCH format 2 decoded soft bit output

logical column vector

PUCCH format 2 decoded soft bit output, returned as a logical column vector. This output contains the result of decoding `sym`.

Data Types: `logical`

See Also

`ltePUCCH1Decode` | `ltePUCCH2` | `ltePUCCH2DRS` | `ltePUCCH2DRSDecode` | `ltePUCCH2DRSIndices` | `ltePUCCH2Indices` | `ltePUCCH2PRBS` | `ltePUCCH3Decode` | `lteUCIDecode`

ltePUCCH2Indices

PUCCH format 2 resource element indices

Syntax

```
[ind,info] = ltePUCCH2Indices(ue,chs)
[ind,info] = ltePUCCH2Indices(ue,chs,opts)
```

Description

`[ind,info] = ltePUCCH2Indices(ue,chs)` returns a matrix of resource element indices given the UE-specific settings structure ue and channel transmission configuration chs. It returns a matrix of resource element (RE) indices and information structure array info for the Physical Uplink Control Channel (PUCCH) Format 2. By default the indices are returned in 1-based linear indexing form that can directly index elements of a resource matrix. These indices are ordered as the PUCCH format 2 modulation symbols should be mapped. Alternative indexing formats can also be generated. The indices for each antenna are in the columns of ind, with the number of columns determined by the number of PUCCH Resource Indices specified.

`[ind,info] = ltePUCCH2Indices(ue,chs,opts)` allows control of the format of the returned indices through a cell array opts of option strings.

Examples

Generate PUCCH Format 2 Indices

Create indices for 1.4 MHz bandwidth, PUCCH resource index 0, using default values for all other parameters.

```
ind = ltePUCCH2Indices(struct('NULRB',6),struct('ResourceIdx',0));
ind(1:4)
```

```
     1
     2
```

```
3
4
```

Input Arguments

ue — UE-specific settings

structure

UE-specific settings, specified as a structure. ue can contain the following fields.

NULRB — Number of uplink resource blocks

nonnegative scalar integer

Number of uplink resource blocks, specified as a nonnegative scalar integer.

Example: 4,5,...

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length for uplink channels

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink channels, specified as `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

chs — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure. chs contains the following fields.

ResourceIdx — PUCCH resource indices

nonnegative integer vector | 0...2047

PUCCH resource indices, specified as nonnegative integer vector between 0 and 2047. There is one index for each transmission antenna. These indices determine the cyclic shift and orthogonal cover used for transmission.

Example: 78

Data Types: `double`

`opts` — Options to control the format of the returned indices

cell array of strings | string

Options to control the format of the returned indices, specified as a string or a cell array of strings. The following option strings are accepted.

`Indexing style` — Form of returned indices

`'ind'` (default) | `'sub'`

Form of returned indices, specified as `'ind'` or `'sub'`. `'ind'` denotes linear index form. `'sub'` denotes [*subcarrier*, *symbol*, *antenna*] subscript row style.

Data Types: `char`

`Index base` — Index base of returned indices

`'1based'` (default) | `'0based'`

Index base of returned indices, specified as `'1based'` or `'0based'`. This option controls whether indices start at 1 or 0.

Data Types: `char`

Output Arguments

`ind` — Resource element indices

linear matrix

Resource element (RE) indices, returned as 1 linear matrix. By default the indices are returned in 1-based linear indexing form that can directly index elements of a resource matrix.

Example: 1,2,3,4...

Data Types: `double`

`info` — PUCCH format 2 information

structure array

PUCCH format 2 information, returned as a structure array. info contains the following fields.

`PRBSet` — Set of PRBs occupied by the indices in each slot of the subframe

numeric row vector

Set of PRBs occupied by the indices in each slot of the subframe, returned as a numeric row vector.

Data Types: `double`

`RBIdx` — PUCCH logical resource block index

numeric scalar

PUCCH logical resource block index, returned as a numeric scalar. (m)

Data Types: `double`

See Also

`ltePUCCH1Indices` | `ltePUCCH2` | `ltePUCCH2Decode` | `ltePUCCH2DRS` | `ltePUCCH2DRSDecode` | `ltePUCCH2DRSIndices` | `ltePUCCH2PRBS` | `ltePUCCH3Indices`

ltePUCCH2PRBS

PUCCH format 2 pseudorandom scrambling sequence

Syntax

```
seq = ltePUCCH2PRBS(ue,n)
seq = ltePUCCH2PRBS(ue,n,mapping)
```

Description

`seq = ltePUCCH2PRBS(ue,n)` returns a column vector containing the first n outputs of the Physical Uplink Control Channel (PUCCH) Format 2 scrambling sequence when initialized according to UE-specific settings, ue.

`seq = ltePUCCH2PRBS(ue,n,mapping)` allows control over the format of the returned sequence, seq, through the mapping string. Valid formats are `'binary'`, which is the default, and `'signed'`. The `'binary'` format maps true to 1 and false to 0. The `'signed'` format maps true to –1 and false to 1.

Examples

Generate PUCCH Format 2 Pseudorandom Scrambling Sequence

This example shows the generatation of unsigned and signed PUCCH format 2 pseudorandom scrambling sequences.

Generate a PUCCH format 2 pseudorandom scrambling sequence.

```
seq = ltePUCCH2PRBS(struct('NCellID',1,'NSubframe', ...
    0,'RNTI',1),5)
```

```
seq =

     1
     1
     0
     0
```

```
     1
```

Generate a signed PUCCH format 2 pseudorandom scrambling sequence.

```
seq = ltePUCCH2PRBS(struct('NCellID',1,'NSubframe', ...
    0,'RNTI',1),5,'signed')
```

```
seq =

    -1
    -1
     1
     1
    -1
```

Input Arguments

ue — UE-specific settings
structure

UE-specific settings, specified as a structure. ue contains the following fields.

NCellID — Physical layer cell identity number
nonnegative scalar integer

Physical layer cell identity number, specified as a nonnegative scalar integer.

Example: 1

Data Types: `double`

NSubframe — Subframe number
nonnegative scalar integer

Subframe number, specified as a nonnegative scalar integer.

Example: 0

Data Types: `double`

RNTI — Radio network temporary identifier
scalar integer

Radio network temporary identifier (16-bit), specified as a scalar integer.

Example: 1

Data Types: `double`

`n` — Length of PUCCH format 2 scrambling sequence

nonnegative scalar integer

Length of PUCCH format 2 scrambling sequence, specified as a nonnegative scalar integer.

Example: 10

Data Types: `double`

`mapping` — Control format of the returned sequence

`'binary'` (default) | `'signed'`

Control format of the returned sequence, specified as a string. Valid formats are `'binary'`, which is the default, and `'signed'`. The `'binary'` format maps true to 1 and false to 0. The `'signed'` format maps true to –1 and false to 1.

Data Types: `char`

Output Arguments

`seq` — PUCCH format 2 pseudorandom scrambling sequence

logical column vector | numeric column vector

PUCCH format 2 pseudorandom scrambling sequence, returned as a logical column vector or a numeric column vector. The length of this sequence is given by the input argument, n. If you set mapping to `'signed'`, the output data type is double. Otherwise, the output data type is logical.

Data Types: `logical` | `double`

See Also

`ltePUCCH2` | `ltePUCCH2Decode` | `ltePUCCH2DRS` | `ltePUCCH2DRSDecode` | `ltePUCCH2DRSIndices` | `ltePUCCH2Indices` | `ltePUCCH3Indices` | `ltePUCCH3PRBS`

ltePUCCH3

Physical uplink control channel format 3

Syntax

```
[sym,info] = ltePUCCH3(ue,chs,bits)
```

Description

`[sym,info] = ltePUCCH3(ue,chs,bits)` returns a complex matrix sym containing Physical Uplink Control Channel (PUCCH) format 3 symbols and information structure array, info, for UE-specific settings, ue, channel transmission configuration, chs and vector of coded hybrid ARQ (HARQ) values, bits. The symbols for each antenna are in the columns of sym, with the number of columns determined by the number of PUCCH Resource Indices specified in the chs structure.

bits is a vector of coded HARQ-ACK bits (coded UCI), formed by performing UCI encoding of the HARQ-ACK, as described in section 5.2.3.1 of [2]. This coded bit vector must be 48 bits long. This vector is denoted "block of bits $b(0)...b(M_{bit}-1)$" in section 5.4.2A of [1]. For this PUCCH format, M_{bit} is 48, as given in table 5.4-1 of [1].

For shortened transmissions, when `ue.Shortened` is one, the second column of `info.OrthSeq` has a zero in the last row because the spreading factor for the second slot is 4 instead of 5.

Examples

Generate PUCCH Format 3 Symbols

Generate PUCCH format 3 modulated symbols.

```
ue = struct('NCellID',0,'RNTI',1);
pucch3Sym = ltePUCCH3(ue,struct('ResourceIdx',0),ones(48,1));
pucch3Sym(1:4)
```

```
   1.6330 - 1.2247i
  -0.7071 + 0.7071i
```

```
-0.5577 + 0.1494i
 0.4082 - 0.0000i
```

Input Arguments

ue — UE-specific settings
structure

UE-specific settings, specified as a structure. ue can contain the following fields.

NCellID — Physical layer cell identity
nonnegative integer

Physical layer cell identity, specified as a nonnegative integer.

Data Types: `double`

RNTI — Radio Network Temporary Identifier (16-bit)
nonnegative integer.

Radio Network Temporary Identifier (16-bit), specified as an nonnegative integer.

Data Types: `double`

NSubframe — Subframe number
0 (default) | Optional | nonnegative integer

Subframe number, specified as a nonnegative integer. Optional.

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length for uplink
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink, specified as either `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

Shortened — Shortened subframe flag
0 (default) | Optional | 1

Shortened subframe flag, specified as 0 or 1. Optional. When the value is 1, the last symbol of the subframe is not used. This value is required for subframes with possible SRS transmission.

Data Types: `double`

Data Types: `struct`

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure having the following fields.

`ResourceIdx` — PUCCH Resource Indices

column vector of integers with values from 0...549

PUCCH Resource Indices, specified as a column vector of integers with values from 0...549. There is one entry for each transmission antenna. These determine the cyclic shift and orthogonal cover used for transmission (*n3_pucch*). This input argument is optional.

Data Types: `double`

Data Types: `struct`

`bits` — Coded HARQ-ACK bits

nonnegative integer column vector of length 48

Coded HARQ-ACK bits, specified as a nonnegative integer column vector of length 48. Values are formed by performing UCI encoding of the HARQ-ACK as described in section 5.2.3.1 of [2]. The coded bit vector must be 48 bits long. This vector is denoted "block of bits $b(0)...b(M_{bit}–1)$" in section 5.4.2A of [1]. For this PUCCH format, M_{bit} is 48, as given in table 5.4-1 of [1].

Data Types: `double`

Output Arguments

`sym` — PUCCH format 3 symbols

complex-valued matrix

PUCCH format 3 symbols, returned as complex-valued matrix. The symbols for each antenna are in the columns of sym, with the number of columns determined by the number of PUCCH Resource Indices specified in the structure chs.

Data Types: `double`

Complex Number Support: Yes

`info` — PUCCH format 3 information

structure

PUCCH format 3 information, returned as a structure having the following fields.

`NCellCyclicShift` — Cell-specific cyclic shift

row vector of integers

Cell-specific cyclic shift for each OFDM symbol, returned as a row vector of integers, (*ncell_cs*).

Data Types: `double`

`OrthSeqIdx` — Orthogonal sequence index

row vector of integers

Orthogonal sequence index , returned as a row vector of integers. The vector contains one entry for each slot, (*n_oc*).

Data Types: `double`

`Symbols` — Modulated data symbols

complex-valued row vector

Modulated data symbols, returned as a complex-valued row vector, (*d*).

Data Types: `double`

`OrthSeq` — Orthogonal sequence for each slot

complex-valued matrix

Orthogonal sequence, returned as a complex-valued matrix. Each column contains the orthogonal sequence for each slot, (*w_n_oc*).

Data Types: `double`

`NSymbSlot` — Number of OFDM symbols in each slot

row vector of integers

The number of OFDM symbols in each slot, (`[N_SF,O_PUCCH N_SF,1_PUCCH]`), returned as a row vector.

Data Types: `double`

Data Types: `struct`

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

[2] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`ltePUCCH3Decode` | `ltePUCCH3DRS` | `ltePUCCH3DRSIndices` | `ltePUCCH3Indices` | `ltePUCCH3PRBS` | `lteUCI3Encode`

ltePUCCH3Decode

Physical uplink control channel format 3 decoding

Syntax

```
out = ltePUCCH3Decode(ue,chs,sym)
```

Description

`out = ltePUCCH3Decode(ue,chs,sym)` performs decoding of the PUCCH format 3 given UE-specific settings, ue, and channel transmission configuration, chs. out is a soft bit vector consisting of 48 bits, formed by decoding complex symbol matrix, sym, performing SC-FDMA deprecoding, demodulation with the PUCCH Format 3 reference sequence, QPSK demodulation, and descrambling. The symbols for each antenna are in the columns of sym, and the number of columns should match the number of PUCCH Resource Indices specified in the chs structure.

Examples

Decode PUCCH Format 3 Signal

Decode a PUCCH format 3 signal contained in an equalized resource array.

Set up the UE configuration structure, `ue`, and the PUCCH format 3 configuration structure, `pucch3`.

```
ue = struct('NULRB',6,'NCellID',0,'RNTI',1);
pucch3 = struct('ResourceIdx',0);
```

To model the transmitter, create an uplink resource grid and populate it with PUCCH format 3 symbols.

```
rgrid = lteULResourceGrid(ue);
pucch3Indices = ltePUCCH3Indices(ue,pucch3);
tx = [1;0;0;1;1;1];
```

```
encoded = lteUCI3Encode(tx);
rgrid(pucch3Indices) = ltePUCCH3(ue,pucch3,encoded);
```

To model the receiver, decode the PUCCH format 3 symbols contained in an equalized resource array. Decode and display the UCI.

```
rx = ltePUCCH3Decode(ue,pucch3,rgrid(pucch3Indices));
decoded = lteUCI3Decode(rx,length(tx))
```

```
     1
     0
     0
     1
     1
     1
```

Input Arguments

ue — UE-specific settings

structure

UE-specific settings, specified as a structure. ue can contain the following fields.

NCellID — Physical layer cell identity

nonnegative integer

Physical layer cell identity, specified as a nonnegative integer.

Data Types: `double`

RNTI — Radio Network Temporary Identifier (16-bit)

nonnegative integer.

Radio Network Temporary Identifier (16-bit), specified as an nonnegative integer.

Data Types: `double`

NSubframe — Subframe number

0 (default) | Optional | nonnegative integer

Subframe number, specified as a nonnegative integer. Optional.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length for uplink

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink, specified as either `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`Shortened` — Shortened subframe flag

0 (default) | Optional | 1

Shortened subframe flag, specified as 0 or 1. Optional. When the value is 1, the last symbol of the subframe is not used. This value is required for subframes with possible SRS transmission.

Data Types: `double`

Data Types: `struct`

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure having the following fields.

`ResourceIdx` — PUCCH resource indices

column vector of integers with values from 0...549

PUCCH resource indices, specified as a column vector of integers with values from 0...549. There is one entry for each transmission antenna. These determine the cyclic shift and orthogonal cover used for transmission (*n3_pucch*). This input argument is optional.

Data Types: `double`

Data Types: `struct`

`sym` — PUCCH format 3 symbols

complex-valued matrix

PUCCH format 3 symbols, specified as a complex-valued matrix. The symbols for each antenna are in the columns of sym, and the number of columns should match the number of PUCCH resource indices specified in the structure chs.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`out` — Soft bit vector

48-by-1 real-valued column vector

Soft bit vector consisting of 48 bits, returned as a 48-by-1 real-valued column vector.

Data Types: `double`

See Also

`ltePUCCH3` | `ltePUCCH3DRS` | `ltePUCCH3DRSIndices` | `ltePUCCH3Indices` | `ltePUCCH3PRBS` | `lteUCI3Decode`

ltePUCCH3DRS

PUCCH format 3 demodulation reference signal

Syntax

```
[seq,info] = ltePUCCH3DRS(ue,chs)
```

Description

`[seq,info] = ltePUCCH3DRS(ue,chs)` returns a complex matrix seq containing PUCCH format 3 demodulation reference signal (DRS) values and information structure array, info, for UE-specific settings, ue, and channel transmission configuration, chs. The symbols for each antenna are in the columns of seq, with the number of columns determined by the number of PUCCH resource indices specified in the structure chs.

Examples

Generate PUCCH Format 3 DRS

Generate PUCCH format 3 demodulation reference signal (DRS) values for UE-specific settings.

```
ue = struct('NCellID',1,'NSubframe',0);
pucch3RefSig = ltePUCCH3DRS(ue,struct('ResourceIdx',0));
pucch3RefSig(1:4)
```

```
   0.7071 + 0.7071i
   0.2588 + 0.9659i
  -0.9659 - 0.2588i
  -0.7071 - 0.7071i
```

Input Arguments

ue — UE-specific settings

structure

UE-specific settings, specified as a structure. ue can contain the following fields.

`NCellID` — Physical layer cell identity
nonnegative integer

Physical layer cell identity, specified as a nonnegative integer.

Data Types: `double`

`NSubframe` — Subframe number
0 (default) | Optional | nonnegative integer

Subframe number, specified as a nonnegative integer. Optional.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length for uplink
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink, specified as either `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`Hopping` — Hopping control
`'Off'` (default) | Optional | `'Group'`

Hopping control, specified as a string. Optional.

Data Types: `char`

`Shortened` — Shortened subframe flag
0 (default) | Optional | 1

Shortened subframe flag, specified as 0 or 1. Optional. When the value is 1, the last symbol of the subframe is not used. This value is required for subframes with possible SRS transmission.

Data Types: `double`

Data Types: `struct`

`chs` — Channel transmission configuration
structure

Channel transmission configuration, specified as a structure having the following fields.

`ResourceIdx` — PUCCH Resource Indices

column vector of integers with values from 0...549

PUCCH Resource Indices, specified as a column vector of integers with values from 0...549. There is one entry for each transmission antenna. These determine the cyclic shift and orthogonal cover used for transmission (*n3_pucch*). This input argument is optional.

Data Types: `double`

Data Types: `struct`

Output Arguments

`seq` — PUCCH format 3 DRS values

complex-valued matrix

PUCCH format 3 DRS values, returned as a complex-valued matrix. The symbols for each antenna are in the columns of seq, with the number of columns determined by the number of PUCCH Resource Indices specified in the structure chs.

Data Types: `double`

`info` — Information structure array

structure array

Information structure array, returned as a structure array having the following fields.

`Alpha` — Reference signal cyclic shift for each OFDM symbol

real-valued row vector

Reference signal cyclic shift for each OFDM symbol, returned as a real-valued row vector. (*alpha*)

Data Types: `double`

`SeqGroup` — Base sequence group number for each slot

integer row vector

Base sequence group number for each slot, returned as an integer row vector. (*u*)

Data Types: `double`

`SeqIdx` — Base sequence number for each slot
integer row vector

Base sequence number for each slot, returned as an integer row vector. (v)

Data Types: `double`

`NResourceIdx` — Resource indices for each slot
integer row vector

Resource indices for each slot, returned as an integer row vector. (n')

Data Types: `double`

`NCellCyclicShift` — Cell-specific cyclic shift for each OFDM symbol
integer row vector

Cell-specific cyclic shift for each OFDM symbol, returned as an integer row vector. ($ncell\_cs$)

Data Types: `double`

`OrthSeqIdx` — Orthogonal sequence index for each slot
integer row vector

Orthogonal sequence index for each slot, returned as an integer row vector. (n_oc)

Data Types: `double`

`Symbols` — Modulated data symbols for each OFDM symbol
complex-valued row vector

Modulated data symbols for each OFDM symbol, returned as a complex-valued row vector. (z)

Data Types: `double`
Complex Number Support: Yes

`OrthSeq` — Orthogonal sequence
complex-valued matrix

Orthogonal sequence, returned as a complex-valued matrix. Each column contains the orthogonal sequence for each slot. ($w\ bar$)

Data Types: `double`

`NSymbSlot` — Number of OFDM symbols in each slot

integer row vector

Number of OFDM symbols in each slot, returned as an integer row vector. ([*N_SF,0_PUCCH*, *N_SF,1_PUCCH*])

Data Types: `double`

Data Types: `struct`

See Also

`ltePUCCH1DRS` | `ltePUCCH2DRS` | `ltePUCCH3` | `ltePUCCH3Decode` | `ltePUCCH3DRSIndices` | `ltePUCCH3Indices` | `ltePUCCH3PRBS`

ltePUCCH3DRSIndices

PUCCH format 3 DRS resource element indices

Syntax

```
[ind,info] = ltePUCCH3DRSIndices(ue,chs)
[ind,info] = ltePUCCH3DRSIndices(ue,chs,opts)
```

Description

`[ind,info] = ltePUCCH3DRSIndices(ue,chs)` returns a matrix of resource element indices, ind, given the UE-specific settings structure, ue, and channel transmission configuration, chs. It returns a matrix of resource element indices, ind, and information structure array, info, for the Demodulation Reference Signal (DRS) associated with PUCCH Format 3 transmission. By default, the indices are returned in 1-based linear indexing form that can directly index elements of a resource matrix. These indices are ordered as the PUCCH format 3 DRS modulation symbols should be mapped. Alternative indexing formats can also be generated. The indices for each antenna are in the columns of ind, with the number of columns determined by the number of PUCCH resource indices specified.

`[ind,info] = ltePUCCH3DRSIndices(ue,chs,opts)` allows control of the format of the returned indices through a cell array, opts, of option strings.

Examples

Generate PUCCH Format 3 DRS Indices

Generate indices for the demodulation reference signal (DRS) associated with a PUCCH format 3 transmission of 1.4 MHz bandwidth, using PUCCH Resource Index 0.

```
ind = ltePUCCH3DRSIndices(struct('NULRB',6),struct('ResourceIdx',0));
ind(1:4)
```

```
74
75
76
```

Input Arguments

ue — UE-specific settings
structure

UE-specific settings, specified as a structure. ue can contain the following fields.

NULRB — Number of uplink resource blocks
scalar integer

Number of uplink resource blocks, specified as a scalar integer.

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink, specified as either `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

Data Types: `struct`

chs — Channel transmission configuration
structure

Channel transmission configuration, specified as a structure having the following fields.

ResourceIdx — PUCCH Resource Indices
column vector of integers with values ranging from 0 to 549

PUCCH Resource Indices, specified as a column vector of integers with values from 0...549. There is one index for each transmission antenna. The indices determine the physical resource blocks used for transmission. (*n3_pucch*). This input argument is optional.

Data Types: `double`

Data Types: `struct`

`opts` — Format control for returned element resource indices

`'ind'`, `'1based'` (default) | `'sub'` | `'0based'`

Format control for returned element resource indices, specified as a string or a cell array of strings. The option strings can contain values from the *Indexing Style* and *Index base* categories.

Category	Options	Description
Indexing style	`'ind'` (default)	The returned indices are in linear index form.
	`'sub'`	The returned indices are in [subcarrier, symbol, port] subscript row form.
Index base	`'1based'` (default)	Indices returned are 1-based.
	`'0based'`	Indices returned are 0-based.

Data Types: `char` | `cell`

Output Arguments

`ind` — Resource element indices

matrix of integer values

Resource element indices, returned as a matrix of integer values. The indices for each antenna form the columns of ind, with the number of columns determined by the number of PUCCH Resource Indices specified.

Data Types: `uint32`

`info` — Information associated with PUCCH format 3 transmission

scalar structure | structure array

Information associated with PUCCH format 3 transmission, returned as a scalar structure or a structure array. It contains the following fields.

`PRBSet` — PRBs occupied by indices

row vector of integers

PRBs occupied by the indices in each slot of the subframe (0-based), specified as a row vector of integers.

Data Types: `double`

`RBIdx` — PUCCH logical resource block index

scalar integer

PUCCH logical resource block index (m), specified as a scalar integer.

Data Types: `double`

Data Types: `struct`

See Also

`ltePUCCH1DRSIndices` | `ltePUCCH2DRSIndices` | `ltePUCCH3` | `ltePUCCH3Decode` | `ltePUCCH3DRS` | `ltePUCCH3Indices` | `ltePUCCH3PRBS`

ltePUCCH3Indices

PUCCH format 3 resource element indices

Syntax

```
[ind,info] = ltePUCCH3Indices(ue,chs)
[ind,info] = ltePUCCH3Indices(ue,chs,opts)
```

Description

`[ind,info] = ltePUCCH3Indices(ue,chs)` returns a column vector of resource element indices given the UE-specific settings structure, ue, and channel transmission configuration, chs. It returns a matrix of resource element (RE) indices and information structure array, info, for the Physical Uplink Control Channel (PUCCH) format 3.
By default, the indices are returned in 1-based linear indexing form that can directly index elements of a resource matrix. These indices are ordered as the PUCCH format 3 modulation symbols, `sym`, should be mapped. Alternative indexing formats can also be generated. The indices for each antenna are in the columns of `sym`, with the number of columns determined by the number of PUCCH resource indices specified.

`[ind,info] = ltePUCCH3Indices(ue,chs,opts)` allows control of the format of the returned indices through a cell array, opts, of option strings.

Examples

Generate PUCCH Format 3 RE Indices

Generate the indices of resource elements for PUCCH format 3, using 1.4MHz bandwidth and PUCCH Resource Index 0.

```
ind = ltePUCCH3Indices(struct('NULRB',6),struct('ResourceIdx',0));
ind(1:4)
```

```
     1
     2
```

```
3
4
```

Input Arguments

ue — UE-specific settings
structure

UE-specific settings, specified as a structure. ue can contain the following fields.

NULRB — Number of uplink resource blocks
integer

Number of uplink resource blocks, specified as a integer.

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length for uplink
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as either `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

Shortened — Shortened subframe flag
0 (default) | Optional | 1

Shortened subframe flag, specified as 0 or 1. Optional. When the value is 1, the last symbol of the subframe is not used. This value is required for subframes with possible SRS transmission.

Data Types: `double`

Data Types: `struct`

chs — Channel transmission configuration
structure

Channel transmission configuration, specified as a structure having the following fields.

ResourceIdx — PUCCH resource indices
column vector of integers with values ranging from 0 to 549

PUCCH resource indices, specified as a column vector of integers with values from 0...549. There is one index for each transmission antenna. The indices determine the Physical Resource Blocks used for transmission (*n3_pucch*). This input argument is optional.

Data Types: `double`

Data Types: `struct`

`opts` — Format control for returned element resource indices

string | cell array of strings

Format control for returned element resource indices, specified as a string or a cell array of strings. The option strings can contain values from the *Indexing Style* and *Index base* categories.

Category	Options	Description
Indexing style	`'ind' (default)`	The returned indices are in linear index form
	`'sub'`	The returned indices are in `[subcarrier, symbol, port]` subscript row form
Index base	`'1based' (default)`	Indices returned are 1-based
	`'0based'`	Indices returned are 0-based

Data Types: `char | cell`

Output Arguments

`ind` — Resource element indices

column vector of integer values

Resource element indices, returned as a column vector of integer values.

Data Types: `int32`

`info` — Information associated with PUCCH format 3 transmission

scalar structure | structure array

Information associated with PUCCH format 3 transmission, returned as a scalar structure or a structure array. It contains the following fields.

`PRBSet` — PRBs occupied by indices in each slot of the subframe

integer row vector

PRBs occupied by the indices in each slot of the subframe (0-based), specified as an integer row vector.

Data Types: `double`

`RBIdx` — PUCCH logical resource block index

scalar integer

PUCCH logical resource block index, specified as a scalar integer. (*m*)

Data Types: `double`

`NSymbSlot` — Number of OFDM symbols in each slot

integer row vector

Number of OFDM symbols in each slot, specified as an integer row vector. (*[N_SF,0_PUCCH N_SF,1_PUCCH]*)

Data Types: `double`

Data Types: `struct`

See Also

`ltePUCCH1Indices` | `ltePUCCH2Indices` | `ltePUCCH3` | `ltePUCCH3Decode` | `ltePUCCH3DRS` | `ltePUCCH3DRSIndices` | `ltePUCCH3PRBS`

ltePUCCH3PRBS

PUCCH format 3 pseudorandom scrambling sequence

Syntax

```
seq = ltePUCCH3PRBS(ue,n)
seq = ltePUCCH3PRBS(ue,n,mapping)
```

Description

`seq = ltePUCCH3PRBS(ue,n)` returns a column vector containing the first n outputs of the Physical Uplink Control Channel (PUCCH) format 3 scrambling sequence when initialized according to UE-specific settings, ue, which must be a structure.

`seq = ltePUCCH3PRBS(ue,n,mapping)` allows control over the format of the returned sequence, seq, through the string mapping. Valid formats are `'binary'`, the default, and `'signed'`. The `'binary'` format maps true to 1 and false to 0. The `'signed'` format maps true to –1 and false to 1.

Examples

Generate PUCCH Format 3 Pseudorandom Scrambling Sequence

This example shows the generatation of unsigned and signed PUCCH format 3 pseudorandom scrambling sequences.

Initialize ue specific parameters.

```
ue = struct('NCellID',1,'NSubframe',0,'RNTI',1);
```

Generate a PUCCH format 3 pseudorandom scrambling sequence.

```
pucch3Seq = ltePUCCH3PRBS(ue,5)
```

```
pucch3Seq =
```

```
     1
     1
     0
     0
     1
```

Generate a signed PUCCH format 3 pseudorandom scrambling sequence.

```
pucch3Seq = ltePUCCH3PRBS(ue,5,'signed')
```

```
pucch3Seq =

    -1
    -1
     1
     1
    -1
```

Input Arguments

`ue` — UE-specific settings
structure

UE-specific settings, specified as a structure having the following fields.

`NCellID` — Physical layer cell identity
integer (0...503)

Physical layer cell identity, specified as an integer between 0 and 503.

Data Types: `double`

`NSubframe` — Subframe number
integer

Subframe number, specified as an integer.

Data Types: `double`

`RNTI` — Radio Network Temporary Identifier
integer

Radio Network Temporary Identifier (16-bit), specified as an integer.

Data Types: `double`

Data Types: `struct`

`n` — Length of PUCCH Format 3 scrambling sequence

integer

Length of PUCCH Format 3 scrambling sequence, specified as an integer. This parameter determines the length of the output argument vector, seq.

Example: 3

Data Types: `double`

`mapping` — Format control of the returned sequence

`'binary'` (default) | `'signed'`

Format control of the returned sequence, specified as either the string `'binary'` or the string `'signed'`. The format `'binary'` maps `true` to 1 and `false` to 0. The format `'signed'` maps `true` to –1 and `false` to 1.

Example: `'signed'`

Data Types: `char`

Output Arguments

`seq` — PUCCH format 3 scrambling sequence

logical column vector | numeric column vector

PUCCH format 3 scrambling sequence, returned as a logical column vector or a numeric column vector of size n-by-1. The number of elements returned is determined by the argument n. If you set mapping to `'signed'`, the output data type is double. Otherwise, the output data type is logical.

Data Types: `logical` | `double`

See Also

`ltePRBS` | `ltePUCCH2PRBS` | `ltePUCCH3` | `ltePUCCH3Decode` | `ltePUCCH3DRS` | `ltePUCCH3DRSIndices` | `ltePUCCH3Indices`

ltePUSCH

Physical uplink shared channel

Syntax

```
sym=ltePUSCH(ue,chs,cws)
```

Description

`sym=ltePUSCH(ue,chs,cws)` returns a vector containing the Physical Uplink Shared Channel (PUSCH) complex symbols for UE-specific settings, ue, PUSCH channel-specific configuration, chs, and the codeword or codewords contained in cws. The size of the matrix sym is *N*-by-*P*, where *N* is the number of modulation symbols for one antenna port and *P* is the number of transmission antennas.

The input argument, cws, is a vector of bit values for one codeword to be modulated, or a cell array containing one or two vectors of bit values corresponding to the one or two codewords to be modulated.

For `PRBSet`, if a column vector is provided, the resource allocation is the same in both slots of the subframe. The two-column matrix can be used to specify differing PRBs for each slot in a subframe. The PRB indices are 0-based.

Examples

Create PUSCH Complex Symbols

Create a PUSCH transmission for an uplink FRC A3-3 3 MHz, as specified in [1].

```
ue = struct('NCellID',1,'NSubframe',0,'RNTI',1);
pusch = struct('Modulation','QPSK','PRBSet',(0:14).','RV',0);
frc = lteRMCUL('A3-3');
cw = lteULSCH(ue,pusch,randi([0,1],frc.PUSCH.TrBlkSizes(1),1));
puschSym = ltePUSCH(ue,pusch,cw);
puschSym(1:4)
```

```
   0.1054 - 0.5270i
```

```
-0.3431 - 0.0884i
-0.3386 + 0.1531i
-0.0003 - 0.7208i
```

Input Arguments

`ue` — UE-specific settings

structure

UE-specific settings, specified as a structure having the following fields.

`NCellID` — Physical layer cell identity

nonnegative integer

Physical layer cell identity, specified as a nonnegative integer.

Data Types: `double`

`NSubframe` — Subframe number

integer

Subframe number, specified as an integer.

Data Types: `double`

`RNTI` — Radio network temporary identifier

integer

Radio network temporary identifier (16-bit), specified as an integer.

Data Types: `double`

`NTxAnts` — Number of transmission antennas

1 (default) | Optional | 2 | 4

Number of transmission antennas, specified as either 1, 2, or 4. Optional.

Data Types: `double`

Data Types: `struct`

`chs` — PUSCH channel-specific configuration

structure

PUSCH channel-specific configuration, specified as a structure having the following fields.

`Modulation` — Modulation format

`'QPSK'` (default) | `'16QAM'` | `'64QAM'`

Modulation format, specified as a character string for one codeword or a cell array of one or more strings for one or two codewords. Valid string values can be selected from `'QPSK'`, `'16QAM'`, or `'64QAM'`.

Data Types: `char`

`PRBSet` — PRB indices

column vector | 2-column matrix

PRB indices, specified as a column vector or 2-column matrix, containing the Physical Resource Block indices (PRBs) corresponding to the resource allocations for this PUSCH.

Data Types: `double`

`NLayers` — Number of transmission layers

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers, specified as an integer from 1 to 4. Optional.

Data Types: `double`

`PMI` — Precoder matrix indication

numeric scalar (0...23)

Precoder matrix indication, specified as a numeric scalar between 0 and 23. Only required if `ue.NTxAnts` is set to 2 or 4. The scalar PMI is used during precoding.

Data Types: `double`

Data Types: `struct`

`cws` — Codeword bit values

vector | cell array

Codeword bit values specified as a vector of bit values for one codeword to be modulated, or a cell array containing one or two vectors of bit values corresponding to the one or two codewords to be modulated.

Data Types: `double`

Output Arguments

`sym` — PUSCH symbols
complex-valued numeric matrix

PUSCH symbols, returned as a complex-valued numeric matrix of size *N*-by-*P*. *N* is the number of modulation symbols for one antenna port. *P* is the number of transmission antennas.

Data Types: `double`
Complex Number Support: Yes

References

[1] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePUSCHDecode` | `ltePUSCHDRS` | `ltePUSCHDRSIndices` | `ltePUSCHIndices` | `ltePUSCHPrecode` | `lteULPrecode` | `lteULSCH` | `lteULScramble`

ltePUSCHDecode

Physical uplink shared channel decoding

Syntax

```
[cws,symbols] = ltePUSCHDecode(ue,chs,sym)
[cws,symbols] = ltePUSCHDecode(ue,chs,sym,hest,noiseest)
[cws,symbols] = ltePUSCHDecode(ue,chs,sym,hest,noiseest,alg)
```

Description

`[cws,symbols] = ltePUSCHDecode(ue,chs,sym)` returns a soft bit vector, or cell array of soft bit vectors, cws, containing the received codeword estimates and received constellation of complex symbol vector, symbols, resulting from decoding of the Physical Uplink Shared Channel (PUSCH) complex symbols, sym, for UE-specific settings, ue, and channel transmission configuration structure or structure array, chs. It performs the inverse of the Physical Uplink Shared Channel (PUSCH) processing. See section 5.3 of [1] or `ltePUSCH` for details. Codeword or codewords cws are optionally scaled by channel state information (CSI) calculated during the equalization process.

If UCI is present in the transmitted PUSCH to be decoded, the following optional fields should be configured in the chs structure—`ORI`, `OACK`, `QdRI`, and `QdACK`.

Multiple codewords can be parameterized by two different forms of the chs structure. Each codeword can be defined by separate elements of a 1-by-2 structure array, or the codeword parameters can be combined together in the fields of a single scalar, or 1-by-1, structure. Any scalar field values apply to both codewords and a scalar `NLayers` is the total number. See “UL-SCH Parameterization” for further details.

If UCI control information, such as RI or HARQ-ACK, is present in the received complex PUSCH symbols, then this function performs the descrambling of the placeholder bits by establishing the correct locations with the help of the UCI related parameters present in chs.

For `PRBSet`, if a column vector is provided, the resource allocation is the same in both slots of the subframe; the 2-column matrix can be used to specify differing PRBs for each slot in a subframe. Note that the PRB indices are 0-based.

sym is an *M*-by-*P* matrix or *M*-by-*NU* matrix, where *M* is the number of symbols per antenna or layer, *P* is the number of transmission antennas, `NTxAnts`, and *NU* is the number of transmission layers, `NLayers`. For a single-antenna transmission, when `NTxAnts` is 1, both *P* and *NU* are 1 and sym must be *M*-by-1 and contain the single-antenna PUSCH symbols for decoding. When *P* is greater than 1 and sym is *M*-by-*P*, decoding is performed using pseudoinverse-based deprecoding for spatial multiplexing. When *P* is greater than 1 and sym is *M*-by-*NU*, decoding is performed without deprecoding; the input sym is assumed to already be deprecoded. For example, by having performed channel estimation against the transmit layer DRS sequences and performing equalization of the received symbols using that channel estimate to yield sym. The case where *P*>1 and *P*=*NU* is ambiguous as to whether deprecoding is required here; in that case, this function does apply deprecoding.

`[cws,symbols] = ltePUSCHDecode(ue,chs,sym,hest,noiseest)` performs the decoding of the complex PUSCH symbols, sym, using UE-specific settings, ue, channel transmission configuration, chs, the channel estimate, hest, and the noise estimate noiseest. In this case, sym is an *M*-by-`NRxAnts` matrix, where *M* is the number of symbols per antenna and `NRxAnts` is the number of receive antennas. When `ue.NTxAnts` is greater than 1, the reception is performed using an MMSE equalizer, equalizing between transmitted and received layers. When `ue.NTxAnts` is 1, the reception is performed using MMSE equalization on the received antennas.

hest is a 3-D *M*-by-`NRxAnts`-by-`NTxAnts` array, where *M* is the number of symbols per antenna, `NRxAnts` is the number of receive antennas, and `NTxAnts` is the number of transmit antennas ports, given by `ue.NTxAnts`.

noiseest is an estimate of the noise power spectral density per RE on received subframe; such an estimate is provided by the `lteULChannelEstimate` function.

`[cws,symbols] = ltePUSCHDecode(ue,chs,sym,hest,noiseest,alg)` provides control over weighting the output soft bits with Channel State Information (CSI) calculated during the equalization stage using algorithmic configuration structure, alg.

Examples

Decode PUSCH Symbols from FRC

Decode the PUSCH modulation symbols contained in the output of a fixed reference channel (FRC).

```
frc = lteRMCUL('A3-2');
trdata = randi([0,1],frc.PUSCH.TrBlkSizes(1),1);
[waveform,rgrid] = lteRMCULTool(frc,trdata);
puschIndices = ltePUSCHIndices(frc,frc.PUSCH);
rxCw = ltePUSCHDecode(frc,frc.PUSCH,rgrid(puschIndices));
```

Input Arguments

`ue` — UE-specific settings

structure

UE-specific settings, specified as a structure having the following fields.

`NCellID` — Physical layer cell identity

integer

Physical layer cell identity, specified by an integer

Data Types: `double`

`NSubframe` — Subframe number

integer

Subframe number, specified as an integer.

Data Types: `double`

`RNTI` — Radio network temporary identifier

integer

Radio network temporary identifier (16-bit), specified as an integer.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length for uplink

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink, specified as either `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`NTxAnts` — Number of transmission antennas

1 (default) | Optional | 2 | 4

Number of transmission antennas specified as either 1, 2, or 4. Optional.

Data Types: `double`

`Shortened` — Shortened subframe flag

0 (default) | Optional | 1

Shortened subframe flag, specified as 0 or 1. Optional. When the value is `1`, the last symbol of the subframe is not used. This value is required for subframes with possible SRS transmission.

Data Types: `logical`

Data Types: `struct`

`chs` — Channel transmission configuration

scalar structure | structure array

Channel transmission configuration, specified as a scalar structure or a structure array. chs is the PUSCH channel-specific structure having these fields. If `UCI` is present in the transmitted PUSCH to be decoded, the optional fields, `ORI`, `OACK`, `QdRI`, and `QdACK`, should be configured in the chs structure.

`Modulation` — Modulation format

`'QPSK'` | `'16QAM'` | `'64QAM'`

Modulation format, specified as a string.

Data Types: `char`

`PRBSet` — Physical resource block indices

column vector | 2-column matrix

Physical resource block indices (PRBs), specified as a column vector or 2-column matrix. The PRBs correspond to the resource allocations for this PUSCH.

Data Types: `double`

`NLayers` — Number of transmission layers

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers, total or per codeword, specified as an integer from 1 to 4. Optional.

Data Types: `double`

`PMI` — Precoder matrix indication

0 (default) | Optional | numeric scalar (0...23)

Precoder matrix indication, specified as a numeric scalar between 0 and 23. Optional. Only required if ue.NTxAnts is set to 2 or 4. The scalar PMI is used during precoding.

Data Types: `double`

`ORI` — Number of uncoded RI bits

0 (default) | Optional | integer

Number of uncoded RI bits, specified as an integer. Optional.

Data Types: `double`

`OACK` — Number of uncoded HARQ-ACK bits

0 (default) | Optional | integer

Number of uncoded HARQ-ACK bits, specified as an integer. Optional.

Data Types: `double`

`QdRI` — Number of coded RI symbols

0 (default) | Optional | integer

Number of coded RI symbols in UL-SCH, specified as an integer. Optional. (*Q'_RI*)

Data Types: `double`

`QdACK` — Number of coded HARQ-ACK symbols

0 (default) | Optional | integer

Number of coded HARQ-ACK symbols in UL-SCH (*Q'_ACK*), specified as an integer. Optional.

Data Types: `double`

Data Types: `struct`

`sym` — PUSCH symbols

complex-valued numeric matrix

PUSCH symbols, specified as a complex-valued numeric matrix of size *M*-by-*P* or *M*-by-*NU*, where *M* is the number of symbols per antenna or layer, *P* is the number

of transmission antennas, `NTxAnts`, and *NU* is the number of transmission layers, `NLayers`.

Data Types: `double`
Complex Number Support: Yes

`hest` — Channel estimate
3-D numeric array

Channel estimate, specified as a 3-D numeric array of size *M*-by-`NRxAnts`-by-`NTxAnts`, where *M* is the number of symbols per antenna, `NRxAnts` is the number of receive antennas, and `NTxAnts` is the number of transmit antennas ports, given by `ue.NTxAnts`.

Data Types: `double`

`noiseest` — Noise estimate
numeric scalar

Noise estimate, specified as a numeric scalar. This argument is an estimate of the noise power spectral density per RE on received subframe.

Data Types: `double`

`alg` — Algorithmic configuration
structure

Algorithmic configuration, specified as a structure having the following field.

`CSI` — Channel state information
`'On'` (default) | Optional | `'Off'`

Channel state information, specified as `'On'` or `'Off'`. Optional. This argument determines if soft bits should be weighted by CSI.

Data Types: `char`

Data Types: `struct`

Output Arguments

`cws` — Codewords
column vector | cell array of column vectors

Codewords, returned as a column vector or a cell array of column vectors. The soft bit vectors contain the received codeword estimates.

Data Types: `double`

`symbols` — Received constellation of symbols

complex-valued column vector

Received constellation of symbols, received as a complex-valued column vector.

Data Types: `double`

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePUSCH` | `ltePUSCHDeprecode` | `ltePUSCHDRS` | `ltePUSCHDRSIndices` | `ltePUSCHIndices` | `lteULDeprecode` | `lteULDescramble` | `lteULSCHDecode`

ltePUSCHDeprecode

PUSCH MIMO deprecoding onto transmission layers

Syntax

```
out = ltePUSCHDeprecode(in,nu,codebook)
out = ltePUSCHDeprecode(chs,in)
```

Description

`out = ltePUSCHDeprecode(in,nu,codebook)` performs deprecoding of the precoded symbol matrix, in, onto nu layers. nu can be 1, 2, 3, or 4. It performs deprecoding using matrix pseudoinversion, to undo the processing described in Section 5.3.3A of [1]. This function returns a *M*-by-nu matrix, out, containing nu layers with *M* symbols in each layer. The overall operation of the deprecoder is the transpose of that defined in the specification; the symbols for layers and antennas lie in columns rather than rows. The input argument in is an *N*-by-*P* matrix, where *P* is the number of transmission antennas and *N* is the number of symbols per antenna. When *P* is 2 or 4, deprecoding for spatial multiplexing is applied with the scalar codebook index, codebook. When *P* is 1, this input argument is ignored. The codebook matrix corresponding to a particular index can be found in section 5.3.3A of [1].

`out = ltePUSCHDeprecode(chs,in)` performs deprecoding of the precoded symbol matrix, in, according to channel transmission configuration, chs.

Examples

Deprecode PUSCH Symbols

Deprecode PUSCH symbols, using a precoded identity matrix as input.

First, precode an identity matrix. Then, deprecode the precoded matrix. By precoding with an identity matrix, you gain access to the precoding matrices. Generate the precoded matrix with codebook index 0 for 4 layers and 4 antennas.

```
precodingMatrix = ltePUSCHPrecode(eye(4),4,0);
```

```
out = ltePUSCHDeprecode(precodingMatrix,4,0);
isequal(eye(4),out)
```

```
     1
```

The resulting output, `out`, is an identity matrix.

Input Arguments

`in` — Precoded symbol input

complex-valued numeric matrix

Precoded symbol input, specified as an N-by-P complex-valued numeric matrix, where P is the number of transmission antennas and N is the number of symbols per antenna.

Example: `[0.5000 + 0.0000i 0.5000 + 0.0000i; 0.5000 + 0.0000i -0.5000 + 0.0000i]`

Data Types: `double`
Complex Number Support: Yes

`nu` — Number of layers

1 | 2 | 3 | 4

Number of layers, specified as 1, 2, 3, or 4.

Example: 2

Data Types: `double`

`codebook` — Codebook index

numeric scalar

Codebook index, specified as a numeric scalar.

Data Types: `double`

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure having the following fields.

`NLayers` — Number of transmission layers

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers, specified as an integer between 1 and 4. Optional.

Data Types: `double`

`PMI` — Precoder matrix indication
numeric scalar (0...23)

Precoder matrix indication, specified as a numeric scalar between 0 and 23. Only required if the number of transmission antennas, *P*, is 2 or 4. Acceptable values for PMI depend upon *P* and the number of transmission layers, NLayers. The scalar PMI is used during deprecoding.

Data Types: `double`

Data Types: `struct`

Output Arguments

`out` — Deprecoded output
numeric matrix

Deprecoded output, returned as an *M*-by-nu matrix, containing nu layers with *M* symbols in each layer.

Data Types: `double`

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePUSCH` | `ltePUSCHDecode` | `ltePUSCHDRS` | `ltePUSCHDRSIndices` | `ltePUSCHIndices` | `ltePUSCHPrecode` | `lteULDeprecode`

ltePUSCHDRS

PUSCH demodulation reference signal

Syntax

```
[seq,info,layers] = ltePUSCHDRS(ue,chs)
```

Description

`[seq,info,layers] = ltePUSCHDRS(ue,chs)` returns complex matrices, seq and layers containing PUSCH Demodulation Reference Signal (DRS) values, and information structure, info, for UE-specific settings, ue, and channel transmission configuration, chs. seq is of size *M*-by-*P*, where *M* is the number of DRS symbols per antenna and *P* is the number of transmission antennas. When *P* is greater than 1, the DRS is precoded using spatial multiplexing. layers is a matrix of size *M*-by-*NU*, where *M* is the number of DRS symbols per layer and *NU* is the number of transmission layers.

For short base reference sequences, such as those used with PUSCH allocations of 1 or 2 PRBs, Zadoff-Chu sequences are not used. In this case, `RootSeq` and `NZC` are set to –1. For cases where the seq output is empty, such as when the input `PRBSet` is empty, the info structure contains all fields, but each field is either empty for vector fields or –1 for scalar fields.

Examples

Generate PUSCH DRS

Generate a PUSCH Demodulation Reference Signal (DRS) for UE-specific settings.

```
ue = struct('NCellID',1,'NSubframe',0);
puschSeq = ltePUSCHDRS(ue,struct('PRBSet',(0:5).'));
puschSeq(1:4)
```

```
   1.0000 + 0.0000i
  -0.0810 + 0.9967i
  -0.9610 + 0.2766i
```

```
-0.8839 - 0.4677i
```

Input Arguments

`ue` — UE-specific settings

structure

UE-specific settings, specified as a structure having the following fields.

`NCellID` — Physical layer cell identity

nonnegative integer

Physical layer cell identity, specified as a nonnegative integer.

Data Types: `double`

`NSubframe` — Subframe number

integer

Subframe number, specified as an integer.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length for uplink

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink, specified as either `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

`NTxAnts` — Number of transmission antennas

1 (default) | Optional | 2 | 4

Number of transmission antennas specified as either 1, 2, or 4. Optional.

Data Types: `double`

`Hopping` — Frequency hopping method

`'Off'` (default) | Optional | `'Group'` | `'Sequence'`

Frequency hopping method, specified as a string. Optional. Possible values are `'Off'`, `'Group'`, or `'Sequence'`.

Data Types: `char`

`SeqGroup` — PUSCH sequence group number

0 (default) | Optional | 0...29

PUSCH sequence group number, specified as an integer from 0 to 29. Optional. (*delta_SS*)

Data Types: `double`

`CyclicShift` — Number of cyclic shifts used for PUSCH DRS

0 (default) | Optional | 0...7

Number of cyclic shifts used for PUSCH DRS, specified as an integer from 0 to 7. Optional. (yields *n1_DMRS*)

Data Types: `double`

Data Types: `struct`

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure having the following fields.

`PRBSet` — PRB set of indices

column vector | 2-column matrix

PRB set of indices, specified as a column vector or a 2-column matrix. It contains the Physical Resource Block indices (PRBs) corresponding to the resource allocations for this PUSCH.

Data Types: `double`

`NLayers` — Number of transmission layers

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers, specified as an integer from 1 to 4. Optional.

Data Types: `double`

`DynCyclicShift` — Cyclic Shift for DMRS

0 (default) | Optional | 0...7

Cyclic Shift for DMRS, specified as an integer from 0 to 7. Optional. (yields *n2_DMRS*)

Data Types: `double`

`OrthCover` — Orthogonal cover sequence flag

`'Off'` (default) | Optional | `'On'`

Orthogonal cover sequence flag, specified as a string. Optional. It can either be `'On'`, to apply, or `'Off'`, the default, to not apply. (*Activate-DMRS-with OCC*).

Data Types: `char`

`PMI` — Precoder matrix indication

0 (default) | Optional | numeric scalar (0...23)

Precoder matrix indication, specified as a numeric scalar between 0 and 23. Optional. Only required if ue`.`NTxAnts is set to 2 or 4. Acceptable values for PMI depend upon ue`.`NTxAnts and NLayers. The scalar PMI is used during precoding.

Data Types: `double`

Data Types: `struct`

Output Arguments

`seq` — PUSCH DRS sequence

M-by-*P* complex-valued matrix

PUSCH DRS sequence values, returned as an *M*-by-*P* complex-valued matrix where *M* is the number of DRS symbols per antenna and *P* is the number of transmission antennas.

Data Types: `double`
Complex Number Support: Yes

`info` — Information about PUSCH DRS

structure array

Information about PUSCH DRS, returned as a structure array, with one element per transmission layer, having the following fields.

`Alpha` — Reference signal cyclic shift

row vector

Reference signal cyclic shift (*alpha*) for each slot, specified as a row vector.

Data Types: `double`

`SeqGroup` — Base sequence group number
row vector

Base sequence group number (u) for each slot, specified as a row vector.

Data Types: `double`

`SeqIdx` — Base sequence number
row vector

Base sequence number (v) for each slot, specified as a row vector.

Data Types: `double`

`RootSeq` — Root Zadoff-Chu sequence index
row vector

Root Zadoff-Chu sequence index (q) for each slot, specified as a row vector.

Data Types: `double`

`NZC` — Zadoff-Chu sequence length
integer

Zadoff-Chu sequence length (*NRS_ZC*), specified as an integer.

Data Types: `double`

`N1DMRS` — Component of reference signal cyclic shift
integer

Component of the reference signal cyclic shift, signaled from higher layers, (*n1_DMRS*), specified as an integer.

Data Types: `double`

`N2DMRS` — Component of the reference signal cyclic shift
integer

Component of the reference signal cyclic shift, signalled from the most recent DCI format 0 message, (*n2_DMRS*), specified as an integer.

Data Types: `double`

`NPRS` — Cell-specific component of reference signal cyclic shift

row vector

Cell-specific component of the reference signal cyclic shift (n_PRS) for each slot, specified as a row vector.

Data Types: `double`

`OrthSeq` — Orthogonal cover value

row vector

Vector containing the orthogonal cover value (w) for each slot, specified as a row vector.

Data Types: `double`

Data Types: `struct`

`layers` — PUSCH DRS sequence by layers

M-by-NU complex-valued matrix

PUSCH DRS sequence by layers, returned as an M-by-NU complex matrix where M is the number of DRS symbols per layer and NU is the number of transmission layers. If P is greater than 1, the DRS is precoded using spatial multiplexing.

Data Types: `double`
Complex Number Support: Yes

See Also

`ltePUSCH` | `ltePUSCHDecode` | `ltePUSCHDeprecode` | `ltePUSCHDRSIndices` | `ltePUSCHIndices` | `ltePUSCHPrecode`

ltePUSCHDRSIndices

PUSCH DRS resource element indices

Syntax

```
ind = ltePUSCHDRSIndices(ue,chs)
ind = ltePUSCHDRSIndices(ue,chs,opts)
```

Description

`ind = ltePUSCHDRSIndices(ue,chs)` returns a matrix of resource element (RE) indices for the Demodulation Reference Signal (DRS) associated with PUSCH transmission, ind, given the UE-specific settings structure, ue, and channel transmission configuration, chs. By default, the indices are returned in 1-based linear indexing form that can directly index elements of a resource matrix. These indices are ordered as the PUSCH DRS modulation symbols should be mapped. Alternative indexing formats can also be generated. ind is of size *M*-by-*P*, where *M* is the number of DRS indices per antenna and *P* is the number of transmission antennas.

If indices for a number of layers, *NU*, are required, rather than indices for `NTxAnts`, the first *NU* columns of the output can be used. The first *NU* columns of ind are the appropriate indices for the `layers` output from the `ltePUSCHDRS` function.

`ind = ltePUSCHDRSIndices(ue,chs,opts)` allows control of the format of the returned indices through a cell array, opts, of option strings.

Examples

Generate PUSCH DRS RE Indices

Generate the 0-based PUSCH DRS resource element indices in linear form for the uplink reference measurement channel (RMC), A3-3.

```
ue = lteRMCUL('A3-3');
puschInd = ltePUSCHDRSIndices(ue,ue.PUSCH,{'Obased','ind'});
```

```
puschInd(1:4)
```

```
         540
         541
         542
         543
```

Input Arguments

ue — UE-specific settings
structure

UE-specific settings, specified as a structure having the following fields.

NULRB — Number of uplink resource blocks
integer

Number of uplink resource blocks, specified as a integer.

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length for uplink
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as either `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

NTxAnts — Number of transmission antennas
1 (default) | Optional | 2 | 4

Number of transmission antennas specified as either 1, 2, or 4. Optional.

Data Types: `double`

Data Types: `struct`

chs — Channel transmission configuration
structure

Channel transmission configuration, specified as a structure having the following fields.

PRBSet — Physical Resource Block indices
column vector | 2-column matrix

Physical Resource Block indices (PRBs), specified as a column vector or 2-column matrix. The PRBs correspond to the resource allocations for this PUSCH.

Data Types: `double`

Data Types: `struct`

`opts` — Format control for returned element resource indices

`'ind'`, `'1based'` (default) | `'sub'` | `'0based'`

Format control for returned element resource indices, specified as a string or a cell array of strings. The option strings can contain values from the *Indexing Style* and *Index base* categories.

Category	Options	Description
Indexing style	`'ind'` (default)	The returned indices are in linear index form
	`'sub'`	The returned indices are in [subcarrier, symbol, port] subscript row form
Index base	`'1based'` (default)	Indices returned are 1-based
	`'0based'`	Indices returned are 0-based

Data Types: `char` | `cell`

Output Arguments

`ind` — PUSCH DRS resource element indices

M-by-*P* numeric matrix

PUSCH DRS resource element indices, returned as an *M*-by-*P* numeric matrix, where *M* is the number of DRS indices per antenna and *P* is the number of transmission antennas. The resource element (RE) indices for the Demodulation Reference Signal (DRS) are associated with PUSCH transmission. By default, the indices are returned in 1-based linear indexing form that can directly index elements of a resource matrix. These indices are ordered as the PUSCH DRS modulation symbols should be mapped. Alternative indexing formats can also be generated.

Data Types: `uint32`

See Also

`ltePUSCH` | `ltePUSCHDecode` | `ltePUSCHDeprecode` | `ltePUSCHDRS` | `ltePUSCHIndices` | `ltePUSCHPrecode`

ltePUSCHIndices

PUSCH resource element indices

Syntax

```
[ind,info] = ltePUSCHIndices(ue,chs)
[ind,info] = ltePUSCHIndices(ue,chs,opts)
```

Description

`[ind,info] = ltePUSCHIndices(ue,chs)` returns a column vector of resource element indices given the UE-specific settings structure, ue, and channel transmission configuration, chs. It returns a column vector of Physical Uplink Shared Channel (PUSCH) resource element (RE) indices and a structure, info, containing information related to the PUSCH indices. By default, the indices are returned in 1-based linear indexing form that can directly index elements of a resource matrix. These indices are ordered as the PUSCH modulation symbols should be mapped. Alternative indexing formats can also be generated.

Support of PUSCH frequency hopping is provided by the function `lteDCIResourceAllocation`, which creates `PRBSet` from a DCI Format 0 message.

`[ind,info] = ltePUSCHIndices(ue,chs,opts)` allows control of the format of the returned indices through a cell array, opts, of option strings.

Examples

Generate PUSCH RE Indices

Generate 0-based PUSCH resource element (RE) indices in linear form.

```
frc = lteRMCUL('A1-1');
puschIndices = ltePUSCHIndices(frc,frc.PUSCH,{'Obased','ind'});
puschIndices(1:4)
```

```
     0
     1
```

```
2
3
```

Input Arguments

ue — UE-specific settings

structure

UE-specific settings, specified as a structure having the following fields.

NULRB — Number of uplink resource blocks

integer

Number of uplink resource blocks, specified as an integer

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as either `'Normal'` or `'Extended'`. Optional.

Data Types: `char`

NTxAnts — Number of transmission antennas

1 (default) | Optional | 2 | 4

Number of transmission antennas specified as either 1, 2, or 4. Optional.

Data Types: `double`

Shortened — Shortened subframe flag

0 (default) | Optional | 1

Shortened subframe flag, specified as 0 or 1. Optional. When the value is 1, the last symbol of the subframe is not used. This value is required for subframes with possible SRS transmission.

Data Types: `double`

Data Types: `struct`

chs — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure. It contains the following fields.

`PRBSet` — PRB indices

column vector | 2-column matrix

PRB indices, specified as a column vector or a 2-column matrix, containing the Physical Resource Block indices (PRBs) corresponding to the resource allocations for this PUSCH.

Data Types: `double`

`Modulation` — Modulation format

`'QPSK'` (default) | Optional | `'16QAM'` | `'64QAM'`

Modulation format, specified as a character string for one codeword or a cell array of one or more strings for one or two codewords. Optional. Possible string values are `'QPSK'`, `'16QAM'`, or `'64QAM'`.

Data Types: `char`

`NLayers` — Number of transmission layers

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers, specified as an integer from 1 to 4. Optional.

Data Types: `double`

Data Types: `struct`

`opts` — Format control for returned element resource indices

`'ind'`, `'1based'` (default) | `'sub'` | `'0based'`

Format control for returned element resource indices, specified as a string or a cell array of strings. The option strings can contain values from the *Indexing Style* and *Index base* categories.

Category	Options	Description
Indexing style	`'ind'` (default)	The returned indices are in linear index form
	`'sub'`	The returned indices are in [subcarrier, symbol, port] subscript row form
Index base	`'1based'` (default)	Indices returned are 1-based

Category	Options	Description
	`'0based'`	Indices returned are 0-based

Data Types: `cell`

Output Arguments

`ind` — PUSCH resource element indices
column vector of integers

PUSCH resource element (RE) indices, returned as column vector of integers.

Data Types: `uint32`

`info` — Information related to PUSCH indices
structure

Information related to the PUSCH indices, returned as a structure having the following fields.

`G` — Number of coded and rate matched UL-SCH data bits
1- or 2-element vector of integers

Number of coded and rate matched UL-SCH data bits for each codeword, returned as a 1- or 2-element vector of integers.

Data Types: `uint32`

`Gd` — Number of coded and rate matched UL-SCH data symbols
integer

Number of coded and rate matched UL-SCH data symbols, returned as an integer. This field equals the number of rows in the PUSCH indices.

Data Types: `uint32`

Data Types: `struct`

See Also
`lteDCIResourceAllocation` | `ltePUSCH` | `ltePUSCHDecode` | `ltePUSCHDeprecode` | `ltePUSCHDRS` | `ltePUSCHDRSIndices` | `ltePUSCHPrecode`

ltePUSCHPrecode

PUSCH MIMO precoding of transmission layers

Syntax

```
out = ltePUSCHPrecode(in,p,codebook)
out = ltePUSCHPrecode(ue,chs,in)
```

Description

`out = ltePUSCHPrecode(in,p,codebook)` performs precoding of the matrix of layers, in, onto p antennas. p can be 1, 2, or 4. When p is 2 or 4, precoding for spatial multiplexing is applied with the scalar codebook index, codebook. It performs precoding according to section 5.3.3A of [1]. This function returns an *M*-by-*P* matrix, where *M* is the number of symbols per antenna and *P* is the number of transmission antennas. The overall operation of the precoder is the transpose of that defined in the specification; the symbols for layers and antennas lie in columns rather than rows.

in is an *N*-by-*NU* matrix consisting of the *N* modulation symbols for transmission upon *NU* layers. Such a matrix is generated by the `lteLayerMap` function.

codebook is a scalar integer specifying the codebook index to be used during precoding. This input is ignored when p is 1. The codebook matrix corresponding to a particular index can be found in section 5.3.3A of [1].

`out = ltePUSCHPrecode(ue,chs,in)` performs precoding of the matrix of layers, in, according to UE-specific settings, ue, and channel transmission configuration, chs.

Examples

Perform PUSCH MIMO Precoding

Perform physical uplink shared channel (PUSCH) multiple-input, multiple-output (MIMO) precoding on an identity matrix.

Obtain the precoding matrix with codebook index 1 for 3 layers and 4 antennas. By precoding an identity matrix, we can gain access to the precoding matrices themselves.

```
out = ltePUSCHPrecode(eye(3),4,1)
```

```
   0.5000 + 0.0000i  -0.5000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.5000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.5000 + 0.0000i
```

Input Arguments

`in` — Transmission input layers

numeric matrix

Transmission input layers, specified as a numeric matrix of size *N*-by-*NU*. in consists of the *N* modulation symbols for transmission upon *NU* layers.

Example: [1 0 0; 0 1 0; 0 0 1]

Data Types: `double`
Complex Number Support: Yes

`p` — Number of transmission antennas

1 | 2 | 4

Number of transmission antennas, specified as an integer having the values 1, 2, or 4.

Example: 1

Data Types: `double`

`codebook` — Codebook index

scalar integer

Codebook index, specified as a scalar integer. This argument specifies the codebook index used during precoding.

Data Types: `double`

`ue` — UE-specific settings

structure

UE-specific settings, specified as a structure having the following fields.

`NTxAnts` — Number of transmission antennas

1 (default) | Optional | 2 | 4

Number of transmission antennas, specified as 1, 2, or 4. Optional.

Data Types: `double`

Data Types: `struct`

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified a structure. chs can contain the following field. The PMI parameter field is only required if ue.NTxAnts is set to 2 or 4.

`PMI` — Precoder matrix indication

numeric scalar (0...23)

Precoder matrix indication, specified as a numeric scalar between 0 and 23. Only required if ue.NTxAnts is set to 2 or 4. Acceptable values for PMI depend upon ue.NTxAnts and the number of layers, *NU*. The scalar PMI is used during precoding.

Data Types: `double`

Data Types: `struct`

Output Arguments

`out` — Precoded output symbols

M-by-*P* numeric matrix

Precoded output symbols, returned as a numeric matrix of size *M*-by-*P*, where *M* is the number of symbols per antenna and *P* is the number of transmission antennas.

Data Types: `double`
Complex Number Support: Yes

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePUSCH` | `ltePUSCHDecode` | `ltePUSCHDeprecode` | `ltePUSCHDRS` | `ltePUSCHDRSIndices` | `ltePUSCHIndices` | `lteULPrecode`

lteRIDecode

Rank indication channel decoding

Syntax

```
out = lteRIDecode(chs,in)
```

Description

`out = lteRIDecode(chs,in)` performs the block decoding on soft input data, in. The input is assumed to be encoded using the procedure defined for RI in section 5.2.2.6 of [1] for given channel transmission configuration, chs. The function returns the decoded output, out, as a vector of length ORI, the number of uncoded RI bits transmitted.

The block decoding will be performed separately on each soft input data using a maximum likelihood (ML) approach, assuming that in has been demodulated and equalized to best restore the originally transmitted values.

The RI decoder performs different type of block decoding depending upon the number of uncoded RI bits to be recovered. For ORI less than 3 bits, the decoder assumed the bits are encoded using the procedure defined in section 5.2.2.6 of [1]. For decoding 3 to 11 RI bits, the decoder assumes the bits are block encoded using the procedure defined in section 5.2.2.6.4 of [1]. For decoding greater than 11 bits, the decoder performs the inverse procedure described in section 5.2.2.6.5 of [1].

Examples

Decode RI Bits for 64QAM

Decode the soft input bits, `codedRi`, for a 64QAM channel transmission configuration.

```
ri = [1;0;1];
chs = struct('Modulation','64QAM','QdRI',1,'ORI',length(ri));
codedRI = lteRIEncode(chs,ri);
codedRI(codedRI==0) = -1;
```

```
decRI = lteRIDecode(chs,codedRI)

     1
     0
     1
```

Input Arguments

chs — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure. Multiple codewords can be parameterized by two different forms of the chs structure. Each codeword can be defined by separate elements of a 1-by-2 structure array, or the codeword parameters can be combined together in the fields of a single scalar, or 1-by-1, structure. Any scalar field values apply to both codewords and a scalar `NLayers` is the total number. See "UL-SCH Parameterization" for further details.

Modulation — Modulation format

`'QPSK'` | `'16QAM'` | `'64QAM'` | cell array of strings

Modulation scheme type, specified as a string or cell array of strings. If there are two blocks and you provide a cell array with two strings, each string is associated with a transport block.

Data Types: `char` | `cell`

ORI — Number of uncoded RI bits

0 (default) | Optional | nonnegative integer

Number of uncoded RI bits, specified as a nonnegative integer. Optional. The RI decoder performs different type of block decoding depending upon the number of uncoded RI bits to be recovered.

For `ORI` less than 3 bits, the decoder assumed the bits are encoded using the procedure defined in section 5.2.2.6 of [1].

For decoding 3 to 11 RI bits, the decoder assumes the bits are block encoded using the procedure defined in section 5.2.2.6.4 of [1]. For decoding greater than 11 bits, the decoder performs the inverse procedure described in section 5.2.2.6.5 of [1].

Data Types: `double`

`NLayers` — Number of transmission layers

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers, specified as a positive scalar integer. Possible values are 1, 2, 3, or 4. Optional.

Data Types: `double`

Data Types: `struct`

`in` — RI input bits

numeric vector | cell array of numeric vectors

RI input bits, specified as a numeric vector or a cell array of numeric vectors. The block decoding will be performed separately on each soft input data using a maximum likelihood (ML) approach assuming that in has been demodulated and equalized to best restore the originally transmitted values.

Data Types: `double` | `cell`

Output Arguments

`out` — Decoded output

logical column vector

Decoded output, returned as a logical column vector. The vector length is determined by the value of `ORI`.

Data Types: `logical`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteACKDecode` | `lteCQIDecode` | `lteRIEncode` | `lteULSCHDecode` | `lteULSCHDeinterleave`

lteRIEncode

Rank indication channel encoding

Syntax

```
out = lteRIEncode(chs,in)
```

Description

`out = lteRIEncode(chs,in)` returns the coded rank indication (RI) bits after performing block coding, as defined for RI in section 5.2.2.6 of [1]. in should be a vector or cell array containing up to 15 RI bits. out contains the encoded bits in the same form.

Multiple codewords can be parameterized by two different forms of the chs structure. Each codeword can be defined by separate elements of a 1-by-2 structure array, or the codeword parameters can be combined together in the fields of a single scalar, or 1-by-1, structure. Any scalar field values apply to both codewords and a scalar `NLayers` is the total number. See "UL-SCH Parameterization" for further details.

Since the RI bits are carried on all defined codewords, a single input will result in a cell array of encoded outputs if multiple codewords are parameterized. This allows for easy integration with the other functions.

The RI coder performs different types of block coding depending upon the number of RI bits in vector in. If in consists of one element, it uses table 5.2.2.6-3 of [1]. If in consists of two elements, it uses table 5.2.2.6-4 of [1] for encoding. The placeholder bits, x and y in the tables, are represented by –1 and –2, respectively.

Similarly, for 3 to 11 bits, the RI encoding is performed as per section 5.2.2.6.4 of [1]. For greater than 11 bits, the encoding is performed as described in section 5.2.2.6.5 of [1].

Examples

Encode RI Bits for One Codeword

Generate the coded rank indication (RI) bits for a single codeword.

```
riBit = 0;
chsRi = struct('Modulation','64QAM','QdRI',1);
codedRi = lteRIEncode(chsRi,riBit)
```

```
     0
    -2
    -1
    -1
    -1
    -1
```

Encode RI Bits for Two Codewords

Generate the coded rank indication (RI) bits for two codewords on three layers.

```
riBit = 0;
chsRi = struct('Modulation',{{'64QAM' '64QAM'}},'QdRI',1,'NLayers',3);
codedRi = lteRIEncode(chsRi,riBit)
```

```
    [6x1 int8]    [12x1 int8]
```

Input Arguments

`chs` — PUSCH-specific parameter structure

scalar structure | structure array

PUSCH-specific parameter structure, specified as a scalar structure or a structure array. chs contains the following fields.

`QdRI` — Number of coded RI symbols

nonnegative numeric scalar | nonnegative numeric vector

Number of coded RI symbols, specified as a nonnegative numeric scalar or vector (Q'_RI).

Data Types: `double`

`Modulation` — Modulation format

`'QPSK'` | `'16QAM'` | `'64QAM'` | cell array of strings

Modulation format, specified as a string or cell array of strings. If there are 2 blocks, each string in the cell array is associated with each transport block.

Data Types: `char` | `cell`

`NLayers` — Number of transmission layers

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers, specified as a positive numeric scalar. Optional.

Data Types: `double`

Data Types: `struct`

`in` — RI input bits

logical vector of length 1 to 15 | cell array of logical vectors

RI input bits, specified as a logical vector of length 1 to 15 or a cell array of logical vectors. Each vector can contain up to 15 RI bits apiece.

Data Types: `cell` | `double`

Output Arguments

`out` — Encoded output bits

integer column vector | cell array of integer column vectors

Encoded output bits, returned as a integer column vector or a cell array of integer column vectors, in same form as `in`. If the PUSCH-specific parameter structure chs defines multiple codewords, out is a cell array.

Data Types: `int8` | `cell`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteACKEncode` | `lteCQIEncode` | `lteRIDecode` | `lteULSCH` | `lteULSCHInterleave`

lteRMCDL

Downlink reference measurement channel configuration

Syntax

```
rmccfgout = lteRMCDL(rc,duplexmode,totsubframes)
rmccfgout = lteRMCDL(rmccfg,ncodewords)
```

Description

`rmccfgout = lteRMCDL(rc,duplexmode,totsubframes)` returns a configuration structure for the reference channel defined by `rc`. This structure uses a channel-specific default configuration. The structure contains the configuration parameters required to generate a given reference channel waveform using the reference measurement channel (RMC) generator tool, `lteRMCDLTool`. The field names and default values comply with the definition found in [1], annex A.3. `duplexmode` and `totsubframes` are optional input parameters that define the duplexing mode and total number of subframes to generate.

`rmccfgout = lteRMCDL(rmccfg,ncodewords)` returns a configuration structure for the reference channel partially, or wholly, defined by the input structure, `rmccfg`. The input structure, `rmccfg`, can define any, or all, of the listed fields or substructure fields. The output structure `rmccfgout`, retains the defined fields. The undefined fields are given the appropriate default values. The RMC generator tool can use the `rmccfgout` structure to generate a waveform. `rmccfg` must contain at least the `rc` field.

`ncodewords` is an optional input argument defining the number of PDSCH codewords to modulate. `ncodewords` can be 1 or 2. The default used is the value defined in [1] for the RMC configuration given by `rc`.

Examples

Generate an RMC Configuration from a List of Reference Channels

Set the input arguments.

```
rc = 'R.12';
duplex = 'TDD';
nSubFrames = 5;
```

Generate the configuration structure.

```
rmc = lteRMCDL(rc,duplex,nSubFrames)
```

```
rmc =

              RC: 'R.12'
           NDLRB: 6
         CellRefP: 4
          NCellID: 0
     CyclicPrefix: 'Normal'
             CFI: 3
              Ng: 'Sixth'
    PHICHDuration: 'Normal'
           NFrame: 0
       NSubframe: 0
     TotSubframes: 5
            OCNG: 'Off'
       Windowing: 0
      DuplexMode: 'TDD'
            PDSCH: [1x1 struct]
             SSC: 4
       TDDConfig: 1
```

Display the PDSCH structure.

```
rmc.PDSCH
```

```
ans =

          TxScheme: 'TxDiversity'
        Modulation: {'QPSK'}
           NLayers: 4
               Rho: 0
              RNTI: 1
             RVSeq: [0 1 2 3]
                RV: 0
     NHARQProcesses: 7
```

```
       NTurboDecIts: 5
             PRBSet: [6x1 double]
         TrBlkSizes: [208 0 0 0 408 0 0 0 0 408]
    CodedTrBlkSizes: [624 0 0 0 1248 0 0 0 0 1248]
            CSIMode: 'PUCCH 1-1'
            PMIMode: 'Wideband'
```

Generate an RMC Configuration from an RMC Structure Variable

Create a configuration structure for RC R.11, as specified in [1].

```
rmc.RC = 'R.11';
rmc.NCellID = 100;
rmc.PDSCH.TxScheme = 'SpatialMux';
rmcOut = lteRMCDL(rmc,2)
```

```
rmcOut =

              RC: 'R.11'
           NDLRB: 50
         CellRefP: 2
          NCellID: 100
     CyclicPrefix: 'Normal'
              CFI: 2
               Ng: 'Sixth'
    PHICHDuration: 'Normal'
           NFrame: 0
        NSubframe: 0
     TotSubframes: 10
             OCNG: 'Off'
       Windowing: 0
      DuplexMode: 'FDD'
            PDSCH: [1x1 struct]
```

Display the contents of the PDSCH substructure.

```
rmcOut.PDSCH
```

```
ans =
```

```
          TxScheme: 'SpatialMux'
        Modulation: {'16QAM'  '16QAM'}
           NLayers: 2
               Rho: 0
              RNTI: 1
             RVSeq: [2x4 double]
                RV: [0 0]
    NHARQProcesses: 8
      NTurboDecIts: 5
            PRBSet: [50x2 double]
       TrBlkSizes: [2x10 double]
    CodedTrBlkSizes: [2x10 double]
           CSIMode: 'PUCCH 1-1'
           PMIMode: 'Wideband'
            PMISet: 0
```

Input Arguments

rc — Reference measurement channel number

`'R.0' | 'R.1' | 'R.2' | 'R.3' | 'R.4' | 'R.5' | 'R.6' | 'R.7' | 'R.8' | 'R.9' | 'R.10' | 'R.11' | 'R.12' | 'R.13' | 'R.14' | 'R.25' | 'R.26' | 'R.27' | 'R.28' | 'R.6-27RB' | 'R.12-9RB' | 'R.11-45RB'`

Reference measurement channel number as defined in [1], specified as a one of the strings listed in the table. The given RC defaults to the transmission scheme shown.

RMC Number	Transmission Scheme
`'R.0'`, `'R.1'`, `'R.2'`, `'R.3'`, `'R.4'`, `'R.5'`, `'R.6'`, `'R.7'`, `'R.8'`, `'R.9'`, `'R.6-27RB'`	`'Port0'` — Single-antenna port, Port 0
`'R.10'`, `'R.11'`, `'R.11-45RB'`, `'R.12'`, `'R.12-9RB'`	`'TxDiversity'` — Transmit diversity scheme
`'R.13'`, `'R.14'`	`'SpatialMux'` — Closed-loop spatial multiplexing scheme
`'R.25'`, `'R.26'`, `'R.27'`, `'R.28'`	`'Port5'` — Single-antenna port, Port 5 (UE-specific beamforming)

Data Types: `char`

`duplexmode` — Duplexing mode

`'FDD'` (default) | `'TDD'`

Duplexing mode, specified as a string `'FDD'` or `'TDD'`. This field is optional. It represents the frame structure type.

Data Types: `char`

`totsubframes` — Total number of subframes

10 (default) | numeric scalar

Total number of subframes, specified as a numeric scalar. This field is optional. This argument specifies the total number of subframes that form the resource grid, `lteRMCDLTool`, to generate the waveform.

Data Types: `double`

`rmccfg` — Reference channel configuration

structure

Reference channel configuration, specified as a structure. The structure defines any, or all, of the fields or subfields. The function retains the defined fields in the output structure, rmccfgout. The undefined fields are given appropriate default values. rmccfg must contain at least the RC field.

`RC` — Reference measurement channel number

`'R.0'` | `'R.1'` | `'R.2'` | `'R.3'` | `'R.4'` | `'R.5'` | `'R.6'` | `'R.7'` | `'R.8'` | `'R.9'` | `'R.10'` | `'R.11'` | `'R.12'` | `'R.13'` | `'R.14'` | `'R.25'` | `'R.26'` | `'R.27'` | `'R.28'` | `'R.6-27RB'` | `'R.11-45RB'` | `'R.12-9RB'`

Reference measurement channel number, specified as a string.

Data Types: `char`

Data Types: `struct`

`ncodewords` — Number of PDSCH codewords to modulate

1 | 2

Number of PDSCH codewords to modulate, specified as 1 or 2. This field is optional. The default used is the value defined in [1] for the RMC configuration given by RC.

Data Types: `double`

Output Arguments

`rmccfgout` — RMC configuration output

structure

RMC configuration output, returned as a scalar structure. The RMC generator tool can use the `rmccfgout` structure to generate a waveform. rmccfgout contains the following fields.

Parameter Field	Values	Description
`RC`	`'R0'` (default), `'R1'`, `'R2'`, `'R3'`, `'R4'`, `'R5'`, `'R6'`, `'R7'`, `'R8'`, `'R9'`, `'R10'`, `'R11'`, `'R12'`, `'R13'`, `'R14'`, `'R25'`, `'R26'`, `'R27'`, `'R28'`, `'R6-27RB'`, `'R11-45RB'`, `'R12-9RB'`	Reference measurement channel (RMC) number or type, as specified in TS36.101, Annex A.3
`NDLRB`	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`CellREfP`	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
`NCellD`	Nonnegative scalar integer (0, ..., 503)	Physical layer cell identity
`CyclicPrefi`	`'Normal'` (default), `'Extended'`	Cyclic prefix length
`CFI`	1, 2, 3	Control format indicator (CFI) value
`Ng`	`'Sixth'`, `'Half'`, `'One'`, `'Two'`	HICH group multiplier
`PHICHDurati`	Nonnegative scalar integer	PHICH duration

Parameter Field	Values	Description
NFrame	0 (default), Nonnegative scalar integer	Frame number
NSubFrame	Nonnegative scalar integer	Subframe number
TotSubFrames	Nonnegative scalar integer	Total number of subframes to generate
OCNG	`'Off'`, `'On'`	OFDMA channel noise generator
Windowing	Nonnegative scalar integer	Number of time-domain samples over which windowing and overlapping of OFDM symbols is applied
DuplexMode	`'FDD'` (default), `'TDD'`	Duplex mode
TDDConfig	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
SSC	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)
PDSCH	Scalar structure	PDSCH transmission configuration structure

The substructure PDSCH relates to the physical channel configuration and contains the following fields:

Parameter Field	Values	Description
TxScheme	`'SpatialMux'` (default), `'Port0'`, `'TxDiversity'`, `'CDD'`, `'MultiUser'`, `'Port5'`, `'Port7-8'`, `'Port8'`, `'Port7-14'`	Transmission scheme, specified as one of the following options. • `'SpatialMux'` — Closed-loop spatial multiplexing. • `'Port0'` — Single-antenna port, port 0. • `'TxDiversity'` — Transmit diversity scheme. • `'CDD'` — Large delay CDD scheme. • `'MultiUser'` — Multiuser MIMO scheme. • `'Port5'` — Single-antenna port, port 5.

Parameter Field	Values	Description
		• `'Port7-8'` — Single-antenna port, port 7 (**`NLayers`** = 1). Dual layer transmission, ports 7 and 8 (**`NLayers`** = 2). • `'Port8'` — Single-antenna port, port 8. • `'Port7-14'` — Up to 8-layer transmission, ports 7–14.
`Modulation`	`'QPSK'`, `'16QAM'`, `'64QAM'`, cell array of strings	Modulation type, specified as a string or cell array of strings. If 2 blocks, each cell is associated with a transport block.
`NLayers`	1 (default), 2, 3, 4, 5, 6, 7, 8	Number of transmission layers
`NTxAnts`	Nonnegative scalar integer	Number of transmission antenna ports. This argument is only present for UE-specific demodulation reference symbols.
`Rho`	0 (default), Scalar	PDSCH resource element power allocation, in dB
`RNTI`	Scalar integer	Radio network temporary identifier (RNTI) value (16 bits)
`RVSeq`	Integer vector (0,1,2,3)	Redundancy version (RV) sequence, applied to all subframes
`RV`	0, 1, 2, 3, 2-element integer vector	Redundancy version indicators, specified as a vector of 1 or 2 values. Each value can be an integer between 0 and 3.
`NHARQProces`	Nonnegative scalar integer	Number of HARQ processes
`NSubFrame`	Nonnegative scalar integer	Subframe number
`NTurboDecit`	Nonnegative scalar integer	Number of turbo decoder iteration cycles

Parameter Field	Values	Description
PRBSet	1- or 2-column integer matrix	0-based physical resource block (PRB) indices corresponding to the resource allocations for this PDSCH. As a column vector, the resource allocation is the same in both slots of the subframe. As a 2-column matrix, this parameter specifies different PRBs for each slot in a subframe.
TrBlkSizes	1- or 2-row matrix	Transport block sizes for each subframe in a frame
CodedTrBlkS	1- or 2-row numeric matrix	Coded transport block sizes for one or two codewords. This parameter field is only for informational purposes.
PMISet	Integer vector (0, ..., 15)	Precoder matrix indication (PMI) set. It can contain either a single value, corresponding to single PMI mode, or multiple values, corresponding to multiple or subband PMI mode. The number of values depends on the CellRefP, transmission layers and TxScheme.
CSIMode	`'PUCCH 1-0'`, `'PUCCH 1-1'`, `'PUSCH 1-2'`, `'PUSCH 3-0'`, `'PUSCH 3-1'`	CSI reporting mode
PMIMode	`'Wideband'` (default), `'Subband'`	PMI reporting mode

References

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteRMCDLTool` | `lteRMCUL` | `lteTestModel`

lteRISelect

PDSCH rank indication calculation

Syntax

```
[ri,pmiset] = lteRISelect(enb,chs,hest,noiseest)
```

Description

`[ri,pmiset] = lteRISelect(enb,chs,hest,noiseest)` calculates PDSCH RI (Rank Indication) for a given cell wide `enb`, channel configuration structure `chs`, channel estimate resource array `hest` , and receiver noise variance `noiseest`.

The RI selection process consists of using `ltePMISelect` for each possible value of NLayers (the number of transmission layers) and then selecting the NLayers value that maximizes the SINR for the selected PMI. On the PUCCH, RI selection corresponds to Report Type 3 (for reporting Mode 1-0 or Mode 1-1). On the PUSCH, RI selection corresponds to reporting Mode 1-2 or Mode 3-1 .

Perform PMI selection using the codebooks specified in [1], Section 7.2.4. For the `'Port 7-14'` transmission scheme, the CSI reporting codebook is used when the number of CSI-RS ports is 8. For other numbers of CSI-RS ports in the `'Port 7-14'` transmission scheme and for other transmission schemes, in the PMI selection will be performed using the codebook for closed-loop spatial multiplexing defined in [2], Tables 6.3.4.2.3–1 and 6.3.4.2.3–2. `PMIMode = 'Wideband'` corresponds to PUSCH reporting Mode 1–2 or PUCCH reporting Mode 1–1 (PUCCH Report Type 2). `PMIMode = 'Subband'` corresponds to PUSCH reporting Mode 3–1.

Examples

Calculate RI

Populate an empty resource grid for RMC R.13 with cell-specific reference signal symbols. Filter the signal through a channel and demodulate the signal using OFDM. Use estimates of the channel and noise power spectral density for RI and PMI calculation.

Open an empty resource grid RMC R.13. Populate the resource grid with cell specific reference signal symbols.

```
enb = lteRMCDL('R.13');
enb.PDSCH.CodebookSubset = '111111111111111';
reGrid = lteResourceGrid(enb);
reGrid(lteCellRSIndices(enb)) = lteCellRS(enb);
txWaveform = lteOFDMModulate(enb,reGrid);
chcfg.DelayProfile = 'EPA';
chcfg.NRxAnts = 4;
chcfg.DopplerFreq = 5;
chcfg.MIMOCorrelation = 'Low';
chcfg.SamplingRate = 15360000;
chcfg.Seed = 1;
chcfg.InitPhase = 'Random';
chcfg.ModelType = 'GMEDS';
chcfg.NTerms = 16;
chcfg.NormalizeTxAnts = 'On';
chcfg.NormalizePathGains = 'On';
chcfg.InitTime = 0;
```

Filter the signal through a channel and demodulate it.

```
rxWaveform = lteFadingChannel(chcfg,txWaveform);
rxSubframe = lteOFDMDemodulate(enb,rxWaveform);
cec.FreqWindow = 1;
cec.TimeWindow = 31;
cec.InterpType = 'cubic';
cec.PilotAverage = 'UserDefined';
cec.InterpWinSize = 1;
cec.InterpWindow = 'Centered';
```

Estimate the corresponding channel, including the noise spectral density and reference signal subcarriers.

```
[hest, noiseEst] = lteDLChannelEstimate(enb,cec,rxSubframe);
```

Calculate the RI and PMI.

```
[ri,pmi] = lteRISelect(enb,enb.PDSCH,hest,noiseEst)
```

```
ri =

     3
```

```
pmi =

    13
```

Input Arguments

enb — eNodeB cell-wide settings

structure

eNodeB cell wide settings, specified as a structure containing the following parameter fields:

Parameter Field	Required or Optiona	Values	Description
NDLRB	Require	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
NCellID	Require	Nonnegative scalar integer (0, …, 503)	Physical layer cell identity
CellRefP	Require	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
CyclicPrefi	Optiona	`'Normal'` (default), `'Extended'`	Cyclic prefix length
DuplexMode	Optiona	`'FDD'` (default), `'TDD'`	Duplex mode
The following parameters apply when `DuplexMode` is set to, `TDD`.			
TDDConf	Optiona	0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration
SSC	Optiona	0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)

Parameter Field	Required or Optional	Values	Description
NSubfra	Require	Nonnegative scalar integer	Subframe number
The following parameters apply when `chs.TxScheme` is set to `'Port7–14'`.			
CSIRefP	Require	1 (default), 2, 4, 8	Number of CSI-RS antenna ports
CSIRSCo	Require	scalar integer	CSI-RS configuration index. See table 6.10.5.2-1 in TS 36.211.
CSIRSPe	Optional	`'On'` (default), `'Off'`, `Icsi-rs` (0, ..., 154), `[Tcsi-rs Dcsi-rs]`	CSI-RS subframe configuration
NFrame	Optional	0 (default), Nonnegative scalar integer	Frame number

`chs` — Channel-specific transmission configuration

structure | structure array

Channel specific transmission configuration, specified as scalar structure or structure array containing the following parameter fields:

Parameter Field	Required or Optional	Values	Description
PMIMode	Optional	`'Wideband'` (default), `'Subband'`	PMI reporting mode
TxScheme	Optional	`'SpatialMux'` (default), `'Port0'`, `'TxDiversity'`, `'CDD'`, `'MultiUser'`, `'Port5'`, `'Port7-8'`, `'Port8'`, `'Port7-14'`	Transmission scheme, specified as one of the following options. • `'SpatialMux'` — Closed-loop spatial multiplexing. • `'Port0'` — Single-antenna port, port 0. • `'TxDiversity'` — Transmit diversity scheme.

Parameter Field	Required or Optional	Values	Description
			• `'CDD'` — Large delay CDD scheme. • `'MultiUser'` — Multiuser MIMO scheme. • `'Port5'` — Single-antenna port, port 5. • `'Port7-8'` — Single-antenna port, port 7 (**`NLayers`** = 1). Dual layer transmission, ports 7 and 8 (**`NLayers`** = 2). • `'Port8'` — Single-antenna port, port 8. • `'Port7-14'` — Up to 8-layer transmission, ports 7–14.

Parameter Field	Required or Optional	Values	Description
`CodebookSub`	Optional	String or vector, all ones (default)	Codebook subset restriction, specified as a string bitmap. The default values are all ones, permitting all PMI values. This parameter is configured by higher layers and indicates the values of PMI that can be reported. The bitmap, defined in TS36.213, Section 7.2, is arranged a_A-1,a_A-2,...a_0. For example, the element CodebookSubset(1) corresponds to a_A-1 and the element CodebookSubset(end) corresponds to a_0. The length of the bitmap is given by the info.CodebookSubsetSize field returned by `ltePMIInfo`. You can also specify the bitmap in a hexadecimal form by prefixing the string with `'0x'`. Alternatively, you can specify a numeric array identical to the PMISET output, indicating to restrict the selection to only those PMISET values. Specifying the parameter in this way enables you to obtain SINR estimates against an existing reported PMI for RI and CQI selection. If this parameter field is defined but is empty, no codebook subset restriction is applied. `(codebookSubsetRestriction)`

`hest` — Channel estimate

multidimensional array

Channel estimate, specified as a multidimensional array of size *K*-by-*L*-by-*NRxAnts*-by-*P* where:

- *K* is the number of sub.carriers.
- *L* is the number of OFDM symbols
- *NRxAnts* is the number of receive antennas

- P is the number of transmit antennas.

Data Types: `double`
Complex Number Support: Yes

`noiseest` — Receiver noise variance
numeric scalar

Receiver noise variance, specified as numeric scalar. It is an estimate of the received noise power spectral density.

Data Types: `double`

Output Arguments

`ri` — Rank indication
scalar

Rank indication, returned as a scalar, indicating the optimal number of transmission layers to transmit for maximum SINR.

`pmiset` — Precoder matrix indications
column vector

Precoder matrix indications, returned as a column vector.

For the `'Port7-14'` transmission scheme with 8 CSI-RS ports, PMISet has `INFO.NSubbands +1` rows. The first row indicates wideband codebook index *i1*. The subsequent `INFO.NSubbands` rows indicate the subband codebook indices *i2* or, if `INFO.NSubbands = 1`, wideband codebook index *i2*. For other numbers of CSI-RS ports in the `'Port7-14'` transmission scheme, and for other transmission schemes, PMISet has `INFO.NSubbands` rows, each row returns the subband codebook index for that subband. For wideband reporting (`INFO.NSubbands` = 1), PMISet is a scalar specifying the selected wideband codebook index.

References

[1] 3GPP TS 36.213. "Physical layer procedures." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.211. “Physical channels and modulation.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteCQISelect` | `ltePMISelect`

lteRMCDLTool

Downlink RMC waveform generation

Syntax

```
[waveform,grid,rmccfgout] = lteRMCDLTool
[waveform,grid,rmccfgout] = lteRMCDLTool(rc,trdata,duplexmode,
totsubframes)
[waveform,grid,rmccfgout] = lteRMCDLTool(rmccfg,trdata)
```

Description

`[waveform,grid,rmccfgout] = lteRMCDLTool` launches a graphical user interface (GUI) for the parameterization and generation of the reference measurement channel (RMC) waveforms. The main function outputs are specified in the GUI but can also be assigned to variables. waveform, the generated reference measurement channel waveform, is a T-by-P matrix, where T is the number of time-domain samples and P is the number of antennas. grid represents the populated resource grid for all the physical channels specified in annex A.3 of [1]. It is a 3-D array of resource elements for a number of subframes across all configured antenna ports, as described in "Data Structures". rmccfgout is a structure containing information about the OFDM modulated waveform, as described in `lteOFDMInfo`, and the RMC-specific configuration parameters, as described in `lteRMCDL`.

`[waveform,grid,rmccfgout] = lteRMCDLTool(rc,trdata,duplexmode, totsubframes)` returns waveform, grid, and rmccfgout for the default reference measurement channel defined by rc, using the information bits trdata. duplexmode and totsubframes are optional input arguments, and define the duplex mode of the generated waveform and total number of subframes that make up the grid.

`[waveform,grid,rmccfgout] = lteRMCDLTool(rmccfg,trdata)` generates the waveform, grid, and rmccfgout in the same way as above except it takes the user-defined reference channel structure rmccfg as input parameter. The reference configuration structure with default parameters can easily be created with the function `lteRMCDL` which is designed to generate the various RMC configuration structures as per annex A.3

of [1]. This configuration structure then can be modified as per requirement and can be used in the generation of waveform.

Examples

Generate Downlink Reference Measurement Channel Waveform

Generate a time-domain signal, waveform, and a 3-D array of the resource elements, grid, for RC modified R.12, as specified in [1]. This transmission uses 16QAM modulation scheme, instead of QPSK.

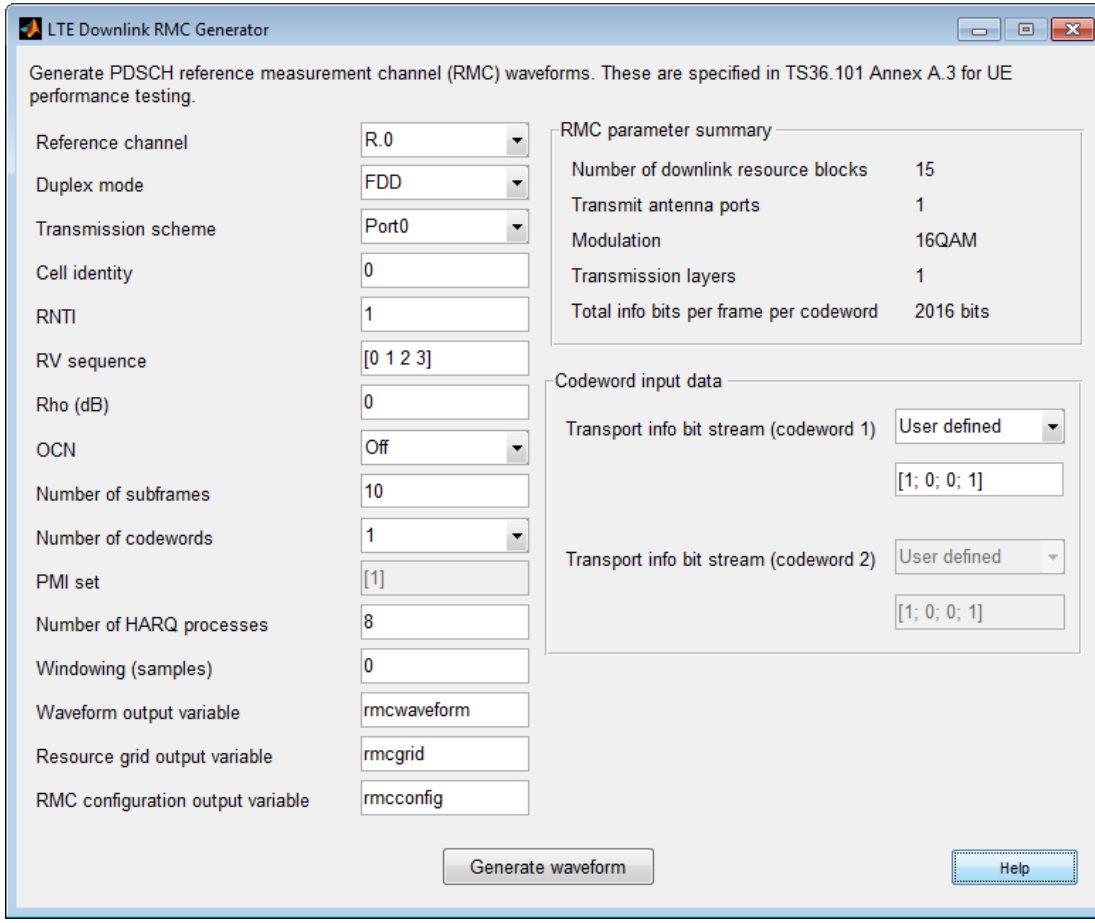
```
rmc = lteRMCDL('R.12');
rmc.PDSCH.Modulation = '16QAM';
[waveform,rgrid,rmccFgOut] = lteRMCDLTool(rmc,[1;0;0;1]);
```

Launch LTE Downlink RMC Generator Tool

Launch the tool to generate a downlink reference measurement channel waveform.

```
lteRMCDLTool;
```

The LTE Downlink RMC Generator dialog box appears.



- “Generate LTE Downlink RMC Waveforms”

Input Arguments

`rc` — Reference channel

```
'R.0' | 'R.1' | 'R.2' | 'R.3' | 'R.4' | 'R.5' | 'R.6' | 'R.7' | 'R.8' | 'R.9'
| 'R.10' | 'R.11' | 'R.12' | 'R.13' | 'R.14' | 'R.25' | 'R.26' | 'R.27' |
'R.28' | 'R.6-27RB' | 'R.12-9RB' | 'R.11-45RB'
```

Reference channel, specified as a string. This argument identifies the reference measurement channel (RMC) number, as specified in [1]. A given rc takes a default transmission scheme as shown in the following table.

RMC Number	Transmission Scheme

`'R.0'`
`'R.1'`
`'R.2'`
`'R.3'`
`'R.4'`
`'R.5'`
`'R.6'`
`'R.7'`
`'R.8'`
`'R.9'`
`'R.6-27RB'`
	`'Port0'` — Single-antenna port, Port 0

`'R.10'`
`'R.11'`
`'R.11-45RB'`
`'R.12'`
`'R.12-9RB'`
	`'TxDiversity'` — Transmit diversity scheme

`'R.13'` `'R.14'`	`'SpatialMux'` — Closed-loop spatial multiplexing scheme
`'R.25'` `'R.26'`	`'Port5'` — Single-antenna port, Port 5 (UE-specific beamforming)

RMC Number	Transmission Scheme
'R.27' 'R.28'	

Data Types: char

trdata — Information bits

vector | cell array containing one or two vectors

Information bits, specified as a vector or cell array containing one or two vectors of bit values. Each vector contains the information bits stream to be coded across the duration of the generation, which represents multiple concatenated transport blocks. Internally, these vectors are looped if the number of bits required across all subframes of the generation exceeds the length of the vectors provided. This feature allows you to enter a short pattern, such as [1;0;0;1], that is repeated as the input to the transport coding. In each subframe of generation, the number of data bits taken from this stream is given by the elements of the TrBlkSizes matrix, a field of the PDSCH substructure of the RMC configuration structure rmccfgout.

When the trdata input contains empty vectors, it indicates no transport data. This results in no PDSCH and its corresponding PDCCH transmission. In other words, the transmission of PDSCH and its corresponding PDCCH can be skipped in the waveform when the trdata contains empty vectors. The other physical channels and signals are transmitted as normal in generated waveform.

Example: [1;0;0;1]

Data Types: double | cell
Complex Number Support: Yes

duplexmode — Duplexing mode

'FDD' (default) | Optional | 'TDD'

Duplexing mode, specified as a string. Optional. This string represents the frame structure type of the generated waveform.

Data Types: char

totsubframes — Total number of subframes

10 (default) | Optional | positive numeric scalar

Total number of subframes, specified as a numeric scalar. Optional. This argument specifies the total number of subframes that form the resource grid.

Data Types: `double`

`rmccfg` — Reference channel configuration

structure

Reference channel configuration, specified as a structure. The structure defines any (or all) of the fields or subfields.

Data Types: `struct`

Output Arguments

`waveform` — Generated RMC time-domain waveform

numeric matrix

Generated RMC time-domain waveform, returned as a numeric matrix of size T-by-P, where T is the number of time-domain samples and P is the number of antennas.

Data Types: `double`
Complex Number Support: Yes

`grid` — Populated resource grid

numeric 3-D array

Populated resource grid, returned as a numeric 3-D array of resource elements for a number of subframes across all configured antenna ports.

Data Types: `double`
Complex Number Support: Yes

`rmccfgout` — Reference channel configuration

scalar structure

Reference channel configuration, returned as a scalar structure. This argument contains information about the OFDM-modulated waveform and RMC-specific configuration parameters.

Data Types: `struct`

References

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDLConformanceTestTool` | `lteRMCDL` | `lteRMCULTool` | `lteTestModelTool`

lteRMCUL

Uplink reference measurement channel or FRC configuration

Syntax

```
rmccfgout = lteRMCUL(rc,duplexmode,totsubframes)
rmccfgout = lteRMCUL(rmccfg)
```

Description

`rmccfgout = lteRMCUL(rc,duplexmode,totsubframes)` returns rmccfgout, a configuration structure for the reference channel defined by rc using a channel-specific default configuration. It returns rmccfgout, a structure containing the configuration parameters required to generate a given reference channel waveform using fixed reference channel (FRC) generator tool, `lteRMCULTool`. The field names and default values of FRCs are in accordance with the definition found in annex A of [1]. duplexmode and totsubframes are optional input parameters which define the duplexing mode and total number of subframes to be generated.

`rmccfgout = lteRMCUL(rmccfg)` returns rmccfgout, a configuration structure for the reference channel partially, or wholly, defined by the input structure, rmccfg. The input structure, rmccfg, can define any, or all, of the parameters or substructure parameters, and the output structure, rmccfgout retains the defined parameters. The undefined fields are given appropriate default values. The rmccfgout structure can be used by the FRC generator tool to generate a waveform. rmccfg must contain at least the `RC` field.

Examples

Create Uplink RMC Configuration

Create a configuration structure for RC A1-1.

```
rmc.RC = 'A1-1';
rmc.NCellID = 100;
rmc.PUSCH.Modulation = 'QPSK';
```

```
rmcOut = lteRMCUL(rmc)
```

Next, display the PUSCH substructure.

```
rmcOut.PUSCH
```

Input Arguments

rc — Reference channel number

```
'A1-1' | 'A1-2' | 'A1-3' | 'A1-4' | 'A1-5' | 'A2-1' | 'A2-2' | 'A2-3' |
'A3-1' | 'A3-2' | 'A3-3' | 'A3-4' | 'A3-5' | 'A3-6' | 'A3-7' | 'A4-1' |
'A4-2' | 'A4-3' | 'A4-4' | 'A4-5' | 'A4-6' | 'A4-7' | 'A4-8' | 'A5-1' |
'A5-2' | 'A5-3' | 'A5-4' | 'A5-5' | 'A5-6' | 'A5-7' | 'A7-1' | 'A7-2' |
'A7-3' | 'A7-4' | 'A7-5' | 'A7-6' | 'A8-1' | 'A8-2' | 'A8-3' | 'A8-4' |
'A8-5' | 'A8-6' | 'A3-2-9RB' | 'A4-3-9RB'
```

Reference channel number, specified as a string. This argument represents the reference measurement channel (RMC) number, or fixed reference channel (FRC), as described in [1].

Data Types: `char`

duplexmode — Duplexing mode

`'FDD'` (default) | Optional | `'TDD'`

Duplexing mode, specified as a string. Optional. It represents the frame structure type.

Data Types: `char`

totsubframes — Total number of subframes

10 (default) | Optional | positive numeric scalar

Total number of subframes, specified as a numeric scalar. Optional. This argument specifies the total number of subframes that form the resource grid.

Data Types: `double`

rmccfg — RMC configuration

Structure

RMC configuration, specified as a structure. rmccfg must contain at least the RC field. rmccfg can contain the following fields.

RC — Reference channel number

`'A1-1' | 'A1-2' | 'A1-3' | 'A1-4' | 'A1-5' | 'A2-1' | 'A2-2' | 'A2-3' | 'A3-1' | 'A3-2' | 'A3-3' | 'A3-4' | 'A3-5' | 'A3-6' | 'A3-7' | 'A4-1' | 'A4-2' | 'A4-3' | 'A4-4' | 'A4-5' | 'A4-6' | 'A4-7' | 'A4-8' | 'A5-1' | 'A5-2' | 'A5-3' | 'A5-4' | 'A5-5' | 'A5-6' | 'A5-7' | 'A7-1' | 'A7-2' | 'A7-3' | 'A7-4' | 'A7-5' | 'A7-6' | 'A8-1' | 'A8-2' | 'A8-3' | 'A8-4' | 'A8-5' | 'A8-6' | 'A3-2-9RB' | 'A4-3-9RB'`

Reference channel number, specified as a string. This argument identifies the reference measurement channel number as described in [1].

Data Types: `char`

SRS — Enable SRS configuration parameters

`'off'` (default) | Optional | `'on'`

Enable SRS configuration parameters, specified as a string. Optional. Set `SRS` to `'on'` to enable SRS related configuration parameters for RMCs which optionally support SRS, or a complete or part SRS structure. If SRS is `'off'`, no SRS configuration is created.

Data Types: `char`

Data Types: `struct`

Output Arguments

rmccfgout — Configuration parameters

Structure

Configuration parameters, returned as a structure. rmccfgout contains the following fields.

RC — Reference channel number

`'A1-1' | 'A1-2' | 'A1-3' | 'A1-4' | 'A1-5' | 'A2-1' | 'A2-2' | 'A2-3' | 'A3-1' | 'A3-2' | 'A3-3' | 'A3-4' | 'A3-5' | 'A3-6' | 'A3-7' | 'A4-1' | 'A4-2' | 'A4-3' | 'A4-4' | 'A4-5' | 'A4-6' | 'A4-7' | 'A4-8' | 'A5-1' | 'A5-2' | 'A5-3' | 'A5-4' | 'A5-5' | 'A5-6' | 'A5-7' | 'A7-1' | 'A7-2' | 'A7-3' | 'A7-4' | 'A7-5' | 'A7-6' | 'A8-1' | 'A8-2' | 'A8-3' | 'A8-4' | 'A8-5' | 'A8-6' | 'A3-2-9RB' | 'A4-3-9RB'`

Reference channel number, returned as a string.

Data Types: `char`

`NULRB` — Number of uplink resource blocks
scalar integer

Number of uplink resource blocks, returned as a scalar integer.

Data Types: `double`

`NCellID` — Physical layer cell identity
scalar integer

Physical layer cell identity, returned as a scalar integer. Physical layer cell identity is `0` for standard and `10` for non-standard bandwidth RC.

Data Types: `double`

`NSubframe` — Initial subframe number
Scalar integer

Initial subframe number, returned as a scalar integer.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length
`'Normal'` (default) | `'Extended'`

Cyclic prefix length, returned as a string.

Data Types: `char`

`CyclicShift` — Cyclic shift
`0...7`

Cyclic shift, returned as a nonnegative integer scalar between `0` and `7`. This argument yields *n1_DMRS*.

Data Types: `double`

`Shortened` — Subframe shortened flag
`0` | `1`

Subframe shortened flag, returned as `0` or `1`. If the function sets the flag to `1`, the last symbol of the subframe is not used. Subframes with possible SRS transmission requires this flag to be set.

Data Types: `logical`

`Hopping` — Hopping type
`'Off' | 'Group' | 'Sequence'`

Hopping type, returned as a string.

Data Types: `char`

`SeqGroup` — PUSCH sequence group number
`0...29`

PUSCH sequence group number, returned as a nonnegative scalar integer between `0` and `29`. (*delta_SS*)

Data Types: `double`

`TotSubframes` — Total number of subframes
`10` (default) | Optional | positive scalar integer

Total number of subframes to be generated, specified as a numeric scalar. Optional. This argument specifies the total number of subframes that form the resource grid.

Data Types: `double`

`RNTI` — Radio network temporary identifier
1 (default) | numeric scalar

Radio network temporary identifier, returned as a numeric scalar (16-bit).

Data Types: `double`

`NTxAnts` — Number of transmission antenna ports
`1 | 2 | 4`

Number of transmission antenna ports, returned as a positive scalar integer. Valid values are `1`, `2`, and `4`.

Data Types: `double`

`Windowing` — Number of windowing samples
positive scalar integer

Number of windowing samples, returned as a positive scalar integer. This argument represents the number of time-domain samples over which windowing and overlapping of SC-FDMA symbols is applied.

Data Types: `double`

`DuplexMode` — Duplex mode
`'FDD'` (default) | `'TDD'`

Duplex mode, returned as a string. Valid values are `'FDD'` and `'TDD'`. It represents the frame structure type.

Data Types: `char`

`PUSCH` — PUSCH transmission configuration
structure

PUSCH transmission configuration, returned as a structure. The PUSCH input represents a substructure relating to the physical channel configuration and contains the following fields.

`Modulation` — Modulation format
`'QPSK'` | `'16QAM'` | `'64QAM'`

Modulation format, returned as a string.

Data Types: `char`

`NLayers` — Number of transmission layers
(default) | `1` | `2` | `3` | `4`

Number of transmission layers, returned as a positive scalar integer. Valid values are `1`, `2`, `3`, and `4`.

Data Types: `double`

`DynCyclicShift` — DMRS cyclic shift
`0...7` | positive scalar integer

DMRS cyclic shift, returned as a positive scalar integer between `0` and `7`. This argument yields *n2_DMRS*.

Data Types: `double`

`NBundled` — HARQ-ACK bundling scrambling sequence index
`0...9` | positive scalar integer

HARQ-ACK bundling scrambling sequence index, returned as a positive scalar integer between `0` and `9`.

Data Types: `double`

`BetaACK` — Modulation and coding scheme (MCS) offset for HARQ-ACK bits
scalar integer

Modulation and coding scheme (MCS) offset for HARQ-ACK bits, returned as a scalar integer.

Data Types: `double`

`BetaCQI` — Modulation and coding scheme (MCS) offset for CQI and PMI bits
scalar integer

Modulation and Coding Scheme (MCS) offset for CQI and PMI bits, returned as a scalar integer.

Data Types: `double`

`BetaRI` — Modulation and Coding Scheme (MCS) offset for RI bits
scalar integer

Modulation and Coding Scheme (MCS) offset for RI bits, returned as a scalar integer.

Data Types: `double`

`NHARQProcesses` — Number of HARQ processes
`1...8`

Number of HARQ processes, returned as a positive scalar integer between `1` and `8`.

Data Types: `double`

`RVSeq` — Redundancy version (RV) indicator
numeric matrix

Redundancy version (RV) indicator, returned as a numeric matrix. This argument is a 1- or 2-row matrix that specifies the redundancy version (RV) indicator for one or two codewords. The RV indicator specified in each column is applied to the transmission in a HARQ process. The number of transmissions in a HARQ process equals the number of columns in `RVSeq`, where the row defines the RV indicator for one or two codewords. In a two-codeword transmission, if `RVSeq` is a row vector, the same RV indicator is applied to both codewords.

Data Types: `double`

`RV` — Redundancy version (RV) indicator in initial subframe
numeric matrix

Redundancy version (RV) indicator in initial subframe, returned as a numeric matrix. This argument is a 1- or 2-column vector that specifies the redundancy version for one or two codewords used in the initial subframe number, `NSubframe`. This parameter field is only for informational purposes and is read-only.

Data Types: `double`

`NTurboDecIts` — Number of turbo decoder iteration cycles

positive scalar integer

Number of turbo decoder iteration cycles, returned as a positive scalar integer.

Data Types: `double`

`OrthCover` — Orthogonal cover sequence flag

`'On' | 'Off'`

Orthogonal cover sequence flag, returned as a string. If set to `'On'`, it applies an orthogonal cover sequence, *w* (*Activate-DMRS-with-OCC*). If set to `'Off'`, it does not apply an orthogonal cover sequence, *w* (*Activate-DMRS-with-OCC*).

Data Types: `char`

`PMI` — Precoder matrix indication

numeric scalar (0...23)

Precoder matrix indication, returned as a numeric scalar between 0 and d23. This argument is used during precoding.

Data Types: `double`

`PRBSet` — Physical resource block set of indices

numeric matrix

Physical resource block set of indices, returned as a numeric matrix. This argument is a 1- or 2-column matrix that contains the 0-based physical resource block indices (PRBs) corresponding to the resource allocations for this PUSCH.

Data Types: `double`

`TrBlkSizes` — Transport block sizes for each subframe in a frame

numeric vector

Transport block sizes for each subframe in a frame, returned as a numeric vector.

Data Types: `double`

`CodedTrBlkSizes` — Coded transport block sizes for each a subframe in a frame

numeric vector

Coded transport block sizes for each a subframe in a frame, returned as a numeric vector. This parameter field is only for informational purposes and is read-only.

Data Types: `double`

Data Types: `struct`

`SRS` — SRS configuration

scalar structure

SRS configuration, returned as a scalar structure. This substructure contains the following fields.

`NTxAnts` — Number of transmission antennas

`1 | 2 | 4`

Number of transmission antennas, returned as a positive scalar integer. Valid values are `1`, `2`, and `4`.

Data Types: `double`

`BWConfig` — Cell-specific SRS bandwidth configuration

`0...7` | positive scalar integer

Cell-specific SRS bandwidth configuration, returned as a positive scalar integer between `0` and `7`. (*C_SRS*)

Data Types: `double`

`BW` — UE-specific SRS bandwidth

`0...3` | positive scalar integer

UE-specific SRS bandwidth, returned as a positive scalar integer between `0` and `3`. (*B_SRS*)

Data Types: `double`

`ConfigIdx` — Configuration index for UE-specific periodicity and subframe offset

`0...644` | positive scalar integer

Configuration index for UE-specific periodicity (*T_SRS*) and subframe offset (*T_offset*), returned as a positive scalar integer between `0` and `644`. (*I_SRS*)

Data Types: `double`

`CyclicShift` — UE-specific cyclic shift
`0...7` | positive scalar integer

UE-specific cyclic shift, returned as a positive scalar integer between 0 and 7. (*n_SRS^cs*)

Data Types: `double`

`SeqGroup` — PUCCH sequence group number
`0...29` | positive scalar integer

PUCCH sequence group number, returned as a positive scalar integer between 0 and 29. (*u*)

Data Types: `double`

`SeqIdx` — Base sequence number
`0` | `1`

Base sequence number, returned as a 0 or 1. (*v*)

Data Types: `double`

`ConfigIdx` — Configuration index for UE-specific periodicity and subframe offset
`0...644`

Configuration index for UE-specific periodicity (*T_SRS*) and subframe offset (*T_offset*), returned as a nonnegative scalar integer between 0 and 644. (*I_SRS*)

Data Types: `double`

`TxComb` — Transmission comb
`0` | `1`

Transmission comb, returned as a 0 or 1. This argument controls SRS positions; SRS is transmitted in 6 carriers per resource block on odd (1) and even (0) resource indices.

Data Types: `double`

`HoppingBW` — SRS frequency hopping configuration index
`0...3` | positive scalar integer

SRS frequency hopping configuration index, returned as a positive scalar integer between 0 and 3. (*b_hop*)

Data Types: `double`

`FreqPosition` — Frequency-domain position
`0...23` | positive scalar integer

Frequency-domain position, returned as a positive numeric scalar between `0` and `23`. (*n_RRC*)

Data Types: `double`

`NF4RachPreambles` — Number of Format 4 in UpPTS RACH preamble frequency resources
`0...6` | positive scalar integer

Number of Format 4 in UpPTS RACH preamble frequency resources, returned as a positive scalar integer. This argument is only returned for `'TDD'` duplex mode.

Data Types: `double`

`OffsetIdx` — SRS subframe offset choice for 2ms SRS periodicity
0...1 | positive numeric scalar

SRS subframe offset choice forf 2ms SRS periodicity, returned as a numeric scalar between `0` and `1`. This argument is only returned for `'TDD'` duplex mode. This parameter indexes the 2 SRS subframe offset entries in the row of table 8.2-2 [2] for the SRS configuration index specified by the `ConfigIdx` parameter.

Data Types: `double`

References

[1] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.213. "Physical layer procedures." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteRMCDL` | `lteRMCULTool` | `lteTestModel`

lteRMCULTool

Uplink RMC or FRC waveform generation

Syntax

```
[waveform,grid,rmccfgout] = lteRMCULTool
[waveform,grid,rmccfgout] = lteRMCULTool(rc,trdata,duplexmode,
totsubframes)
[waveform,grid,rmccfgout] = lteRMCULTool(rmccfg,trdata)
[waveform,grid,rmccfgout] = lteRMCULTool(rmccfg,trdata,cqi,ri,ack)
```

Description

`[waveform,grid,rmccfgout] = lteRMCULTool` launches a graphical user interface (GUI) for the parameterization and generation of the reference measurement channel (RMC) waveforms. The main function output variables are specified in the GUI but they can also be assigned to the waveform, grid, and rmccfgout function output variables, representing the time-domain waveform, resource grid, and RMC configuration structure, respectively. If assigning output variables, it returns waveform, the generated reference measurement channel waveform and grid representing the populated resource grid for all the physical channels specified in annex A of [1]. rmccfgout is a structure containing information about the SC-FDMA modulated waveform as described in `lteSCFDMAInfo` in addition to the RMC specific configuration parameters as described in `lteRMCUL`. The RMC waveform can be configured via a GUI or by passing the required input parameters in a function call.

waveform is a T-by-P matrix, where T is the number of time-domain samples and P is the number of antennas. grid is a 3-D array of resource elements for a number of subframes across all configured antenna ports, as described in “Data Structures”. rmccfgout is a structure containing information about the SC-FDMA modulated waveform as well as RMC configuration parameters.

`[waveform,grid,rmccfgout] = lteRMCULTool(rc,trdata,duplexmode, totsubframes)` returns the waveform, grid, and rmccfgout for the default reference measurement channel defined by rc, using the information bits, trdata. duplexmode and

totsubframes are optional input arguments and define the duplex mode of the generated waveform and total number of subframes that make up the grid.

`[waveform,grid,rmccfgout] = lteRMCULTool(rmccfg,trdata)` generates the waveform, grid and rmccfgout in the same way as above except it takes the user defined reference channel structure rmccfg as input parameter. The reference configuration structure with default parameters can easily be created with the function `lteRMCUL` which is designed to generate the various RMC configuration structures as per annex A of [1]. This configuration structure then can be modified as per requirement and can be used in the generation of waveform.

`[waveform,grid,rmccfgout] = lteRMCULTool(rmccfg,trdata,cqi,ri,ack)` generates the waveform, grid and rmccfgout in the same way as above but with support for control information transmission on PUSCH specified in vectors cqi, ri, and ack. Together, these three fields form UCI. The vectors cqi, ri, and ack can be empty vectors if these particular control information bits are not present in this transmission. The UCI is encoded for PUSCH transmission using the processing defined in section 5.2.4 of [2], consisting of UCI coding and channel interleaving. The vectors cqi, ri, and ack are not treated as data streams. Thus, each subframe will contain the same CQI, RI, and ACK information bits.

Examples

Generate Uplink RMC Waveform

Generate a time-domain signal, waveform, and a 2-D array of the resource elements, grid, for the modified RC, A1-1, as specified in [1]. This transmission uses 16QAM modulation scheme, instead of QPSK.

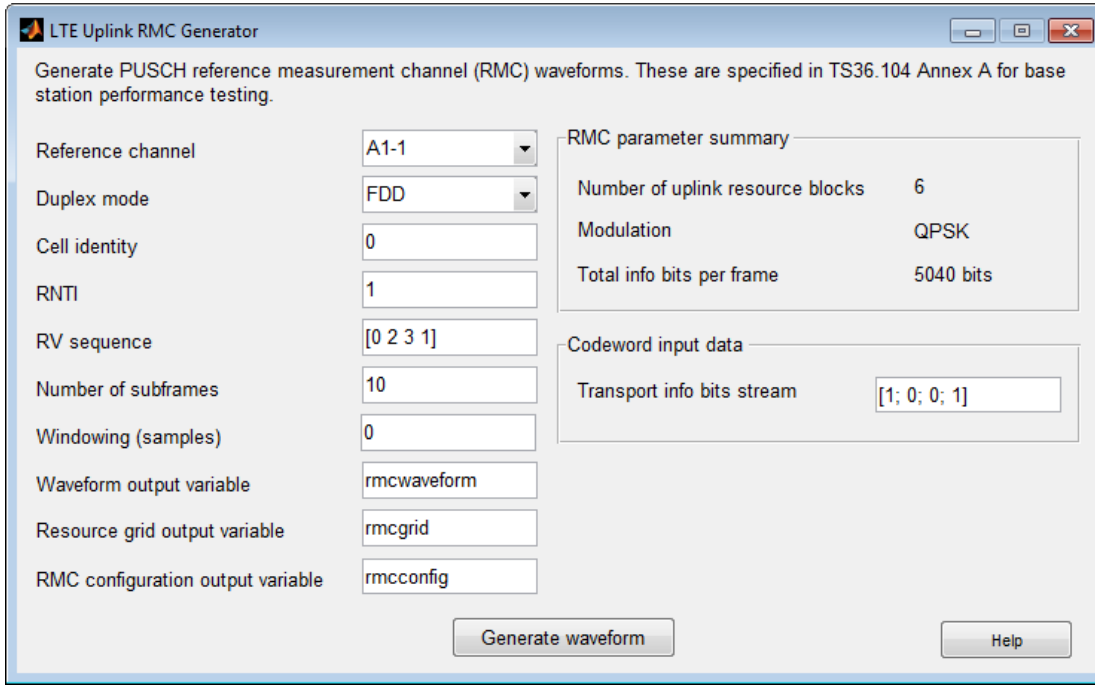
```
frc = lteRMCUL('A1-1');
frc.PUSCH.Modulation = '16QAM';
[waveform,rgrid,rmccFgOut] = lteRMCULTool(frc,[1;0;0;1]);
```

Launch LTE Uplink RMC Generator Tool

Launch the tool to generate an uplink reference measurement channel waveform.

```
lteRMCULTool;
```

The LTE Uplink RMC Generator dialog box appears.



- “Generate LTE Uplink RMC Waveforms”

Input Arguments

`rc` — Reference measurement channel

```
'A1-1' | 'A1-2' | 'A1-3' | 'A1-4' | 'A1-5' | 'A2-1' | 'A2-2' | 'A2-3' |
'A3-1' | 'A3-2' | 'A3-3' | 'A3-4' | 'A3-5' | 'A3-6' | 'A3-7' | 'A4-1' |
'A4-2' | 'A4-3' | 'A4-4' | 'A4-5' | 'A4-6' | 'A4-7' | 'A4-8' | 'A5-1' |
'A5-2' | 'A5-3' | 'A5-4' | 'A5-5' | 'A5-6' | 'A5-7' | 'A7-1' | 'A7-2' |
'A7-3' | 'A7-4' | 'A7-5' | 'A7-6' | 'A8-1' | 'A8-2' | 'A8-3' | 'A8-4' |
'A8-5' | 'A8-6' | 'A3-2-9RB' | 'A4-3-9RB'
```

String representing reference measurement channel number, as specified in [1].

Data Types: `char`

`trdata` — Information bits

column vector | cell array of one or two column vectors

Information bits, specified as a column vector or a cell array containing one or two column vectors of bit values. Each vector contains the information bits stream to be coded across the duration of the generation, which represents multiple concatenated transport blocks. Internally these vectors are looped if the number of bits required across all subframes of the generation exceeds the length of the vectors provided. This allows for the user to enter a short pattern, such as `[1;0;0;1]`, that is repeated as the input to the transport coding. In each subframe of generation, the number of data bits taken from this stream is given by the elements of the `TrBlkSizes` matrix, a field of the PUSCH substructure of the RMC configuration structure, rmccfgout.

Data Types: `double` | `cell`

`duplexmode` — Duplexing mode

`'FDD'` (default) | Optional | `'TDD'`

Duplexing mode, specified as a string. Optional. This string represents the frame structure type of the generated waveform.

Data Types: `char`

`totsubframes` — Total number of subframes

10 (default) | Optional | positive numeric scalar

Total number of subframes, specified as a numeric scalar. Optional. This argument specifies the total number of subframes that form the resource grid.

Data Types: `double`

`rmccfg` — Reference channel configuration

structure

Reference channel configuration, specified as a structure. The structure defines any, or all, of the fields or subfields.

Data Types: `struct`

`cqi` — CQI information bits

numeric vector

CQI information bits, specified as a numeric vector. CQI stands for channel quality information. cqi can be empty if these particular control information bits are not present in the transmission. cqi is not treated as a data stream, and thus each subframe will contain the same CQI information bits.

Data Types: `double`

`ri` — RI information bits

numeric vector

RI information bits, specified as a numeric vector. RI stands for rank indication. ri can be empty if these particular control information bits are not present in the transmission. ri is not treated as a data stream, and thus each subframe will contain the same RI information bits.

Data Types: `double`

`ack` — ACK information bits

numeric vector

ACK information bits, specified as a numeric vector. ACK stands for acknowledgement in automatic repeat request (ARQ) protocols. ack can be empty if these particular control information bits are not present in the transmission. ack is not treated as a data stream, and thus each subframe will contain the same ACK information bits.

Data Types: `double`

Output Arguments

`waveform` — Generated RMC time-domain waveform

numeric matrix

Generated RMC time-domain waveform, returned as a numeric matrix of size T-by-P, where T is the number of time-domain samples and P is the number of antennas.

Data Types: `double`
Complex Number Support: Yes

`grid` — Populated resource grid

numeric 3-D array

Populated resource grid, returned as a numeric 3-D array of resource elements for a number of subframes across all configured antenna ports.

Data Types: `double`
Complex Number Support: Yes

`rmccfgout` — RMC configuration
structure

RMC configuration, returned as a structure. The structure contains information about the SC-FDMA modulated waveform, as described in `lteSCFDMAInfo`, and the RMC-specific configuration parameters, as described in `lteRMCUL`. The RMC waveform can be configured via a graphical user interface (GUI) or by passing the required input parameters in a function call.

Data Types: `struct`

References

[1] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDLConformanceTestTool` | `lteRMCDLTool` | `lteRMCUL` | `lteTestModelTool`

lteRateMatchConvolutional

Convolutional rate matching

Syntax

```
out = lteRateMatchConvolutional(in,outlen)
```

Description

`out = lteRateMatchConvolutional(in,outlen)` rate matches the input data vector, in, to create an output vector, out, of length outlen. This function includes the stages of subblock interleaving, bit collection and bit selection, and pruning defined for convolutionally encoded data. For more information, see section 5.1.4.2 of [1]. The input data is assumed to comprise a concatenation of 3 subblocks, each of which is then interleaved prior to virtual circular buffer creation. No special processing is given to input filler bits.

Examples

Perform Convolutional Rate Matching

Perform convolutional rate matching of a coded block vector of length 132, with the output length set to 50.

```
rateMatched = lteRateMatchConvolutional(ones(132,1),50);
size(rateMatched)
```

```
ans =

    50     1
```

Input Arguments

`in` — Input data

numeric column vector

Input data, specified as a complex-valued column vector. Input data is assumed to comprise a concatenation of 3 subblocks each of which is then interleaved prior to virtual circular buffer creation. No special processing is given to input filler bits.

Example: `ones(5,1)`

Data Types: `double | uint8 | uint16 | uint32 | uint64 | int8 | int16 | int32 | int64`

`outlen` — Output vector length

nonnegative scalar integer

Output vector length, specified as a nonnegative scalar integer.

Data Types: `double`

Output Arguments

`out` — Rate matched output

numeric column vector

Rate matched output, returned as numeric column vector.

Data Types: `double | uint8 | uint16 | uint32 | uint64 | int8 | int16 | int32 | int64`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteBCH | lteConvolutionalEncode | lteDCIEncode | lteRateMatchTurbo | lteRateRecoverConvolutional`

lteRateMatchTurbo

Turbo rate matching

Syntax

```
out = lteRateMatchTurbo(in,outlen,rv)
out = lteRateMatchTurbo(in,outlen,rv,chs)
```

Description

`out = lteRateMatchTurbo(in,outlen,rv)` performs rate matching of the input data, in, to create the output vector, out, of length outlen. The input data could be a vector or a cell array. This function includes the stages of subblock interleaving, bit collection and bit selection, and pruning defined for turbo encoded data. For more information, see section 5.1.4.1 of [1].

The input data can be a single vector or a cell array of vectors assumed to be code blocks. In the cell array case, each vector is rate matched separately and the results are concatenated into the single output vector, out. The length of each nonempty input vector must be an integer multiple of 3. The redundancy version of the output is controlled by the rv parameter. rv can be 0, 1, 2, or 3. The bit selection stage assumes a QPSK transmission mapped onto a single layer. It also assumes no restriction on the number of soft bits, as in an uplink transport channel.

`out = lteRateMatchTurbo(in,outlen,rv,chs)` allows additional control of the bit selection stage through selection of parameters for the soft buffer size and physical channel configuration in the chs input structure.

For downlink turbo coded transport channels, the soft buffer dimensions can be controlled by one of the following sets of optional chs fields, in order of precedence.

- NIR — Soft buffer size for entire input transport block
- All of NSoftbits, TxScheme, and DuplexMode. If `DuplexMode` is `'TDD'`, also specify TDDConfig and NSubframe.

If neither the NIR nor NSoftbits fields are present, the function assumes an uplink turbo coded transport channel and places no limit on the number of soft bits.

Examples

Perform Turbo Rate Matching

Perform turbo rate matching on a coded block vector of length 132, with the output length set to 100 and the RV parameter set to 0.

```
codedBlklen = 132;
rmatched = lteRateMatchTurbo(ones(codedBlklen,1),100,0);
size(rmatched)
```

```
ans =

   100     1
```

Input Arguments

`in` — Input data

vector | cell array of vectors

Input data, specified as a single vector or a cell array of vectors, assumed to be code blocks. In the cell array case, each vector is rate matched separately and the results are concatenated into the single output vector, out. The length of each nonempty input vector must be an integer multiple of 3.

Example: `ones(132,1)`

Data Types: `double` | `uint8` | `uint16` | `uint32` | `uint64` | `int8` | `int16` | `int32` | `int64` | `cell`

`outlen` — Output vector length

nonnegative integer

Output vector length, specified as a nonnegative integer.

Example: 3

Data Types: `double`

`rv` — Redundancy version control

0 | 1 | 2 | 3

Redundancy version control, specified as 0, 1, 2, or 3.

Example: 1

Data Types: `double`

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure. It allows additional control of the bit selection stage through parameters for the soft buffer size and physical channel configuration. For downlink turbo coded transport channels, the soft buffer dimensions can be controlled by one of the following sets of optional chs fields, in order of precedence.

- NIR — Soft buffer size for entire input transport block
- All of NSoftbits, TxScheme, and DuplexMode. If DuplexMode is `'TDD'`, also specify TDDConfig and NSubframe.

If neither the NIR nor NSoftbits fields are present, the function assumes an uplink turbo coded transport channel and places no limit on the number of soft bits.

chs can contain the following fields.

`Modulation` — Modulation format

`'QPSK' | '16QAM' | '64QAM'`

Modulation format, specified as a string. Accepted values are `'QPSK'`, `'16QAM'`, and `'64QAM'`.

Data Types: `char`

`NLayers` — Number of transmission layers for transport block

1 | 2 | 3 | 4

Number of transmission layers for transport block, specified as 1, 2, 3, or 4. Not necessary if TxScheme is set to `'Port0'`, `'TxDiversity'`, or `'Port5'`.

Data Types: `double`

`TxScheme` — Transmission scheme

`'Port0'` (default) | Optional | `'TxDiversity'` | `'CDD'` | `'SpatialMux'` | `'MultiUser'` | `'Port5'` | `'Port7-8'` | `'Port8'` | `'Port7-14'`

Transmission scheme, specified as a string. Optional. Accepted values and their descriptions are shown in the following table.

Transmission scheme	Description
`'Port0'`	Single antenna port, port 0. Default.
`'TxDiversity'`	Transmit diversity
`'CDD'`	Large delay CDD
`'SpatialMux'`	Closed loop spatial multiplexing
`'MultiUser'`	Multi-user MIMO
`'Port5'`	Single-antenna port, Port 5
`'Port7-8'`	Single-antenna port, port 7, if NLayers is set to 1. Dual layer transmission, port 7 and 8, if NLayers is set to 2.
`'Port8'`	Single-antenna port, Port 8
`'Port7-14'`	Up to 8 layer transmission, ports 7-14

`NIR` — Soft buffer size for entire input transport block

nonnegative integer

Soft buffer size for entire input transport block, specified as a nonnegative integer.

Data Types: `double`

`NSoftbits` — Total number of soft channel bits

nonnegative integer

Total number of soft channel bits, specified as a nonnegative integer.

Data Types: `double`

`DuplexMode` — Duplex mode

`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, specified as a string. Optional. Accepted values are `'FDD'` and `'TDD'`.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration

0 (default) | Optional | nonnegative scalar integer (0...6)

Uplink or downlink configuration, specified as a nonnegative scalar integer between 0 and 6. Optional. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

`NSubframe` — Subframe number
integer

Subframe number, specified as an integer. Only required if DuplexMode is set to `'TDD'`

Data Types: `double`

Data Types: `struct`

Output Arguments

`out` — Turbo rate matched output
numeric column vector

Turbo rate matched output, returned as a numeric column vector.

Data Types: `double` | `uint8` | `uint16` | `uint32` | `uint64` | `int8` | `int16` | `int32` | `int64`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDLSCH` | `lteRateMatchConvolutional` | `lteRateRecoverTurbo` | `lteTurboEncode` | `lteULSCH`

lteRateRecoverConvolutional

Convolutional rate matching recovery

Syntax

```
out = lteRateRecoverConvolutional(in,outlen)
```

Description

`out = lteRateRecoverConvolutional(in,outlen)` performs rate recovery of the input data vector, in, to create an output vector, out, of length outlen. This function is the inverse of the rate matching operation for convolutionally encoded data. For more information, see `lteRateMatchConvolutional`. This function includes the inverses of the subblock interleaving, bit collection and bit selection, and pruning stages. This function also implements additive soft combining of the input data elements in the case where repetition occurred during the original rate matching.

Examples

Perform Convolutional Rate Recovery

Perform rate recovery after rate matching. The returned vector has the same length as the input to rate matching.

```
codedBlklen = 132;
rateMatched = lteRateMatchConvolutional(ones(codedBlklen ,1),50);
rateRecovered = lteRateRecoverConvolutional(rateMatched,codedBlklen);
size(rateRecovered)
```

```
   132     1
```

The output variable, `rateRecovered`, is a vector of the same length as the input to rate matching.

Input Arguments

`in` — Input data

numeric column vector

Input data, specified as a numeric column vector.

Data Types: `double` | `uint8` | `uint16` | `uint32` | `uint64` | `int8` | `int16` | `int32` | `int64`

`outlen` — Output vector length

nonnegative scalar integer

Output vector length, specified as a nonnegative scalar integer.

Example: 50

Data Types: `double`

Output Arguments

`out` — Rate recovered output

numeric column vector

Rate recovered output, returned as a numeric column vector.

Data Types: `double` | `uint8` | `uint16` | `uint32` | `uint64` | `int8` | `int16` | `int32` | `int64`

See Also

`lteBCHDecode` | `lteConvolutionalDecode` | `lteDCIDecode` | `lteRateMatchConvolutional` | `lteRateRecoverTurbo`

lteRateRecoverTurbo

Turbo rate recovery

Syntax

```
out = lteRateRecoverTurbo(in,trblklen,rv)
out = lteRateRecoverTurbo(in,trblklen,rv,chs,cbsbuffers)
```

Description

`out = lteRateRecoverTurbo(in,trblklen,rv)` performs rate recovery of the input vector, in, to create a cell array of vectors, out. out represents the turbo encoded code blocks prior to concatenation. This function is the inverse of the rate matching operation for turbo encoded data. For more information, see `lteRateMatchTurbo` and section 5.1.4.1 of [1]. This function includes the inverses of the subblock interleaving, bit collection and bit selection and pruning stages. The dimensions of out are deduced from trblklen, which represents the length of the original encoded transport block. This parameterization is required in order to recover the original number of code blocks, their encoded lengths and the locations of any filler bits. The redundancy version used to recover the data is controlled by the rv parameter. rv can be 0, 1, 2, or 3. The bit selection recovery assumes a QPSK transmission mapped onto a single layer. It also assumes no restriction on the number of soft bits, as in an uplink transport channel.

`out = lteRateRecoverTurbo(in,trblklen,rv,chs,cbsbuffers)` behaves as above except the chs input structure allows additional control of the bit selection recovery stage through parameters for the soft buffer size and physical channel configuration. It also allows combining with preexisting soft information for the HARQ process in cbsbuffers.

For downlink turbo coded transport channels, the soft buffer dimensions can be controlled by one of the following sets of optional chs fields, in order of precedence.

- NIR — Soft buffer size for entire input transport block
- All of NSoftbits, TxScheme, and DuplexMode. If DuplexMode is `'TDD'`, also specify TDDConfig and NSubframe.

If neither the NIR nor NSoftbits fields are present, the function assumes an uplink turbo coded transport channel and places no limit on the number of soft bits.

Examples

Perform Turbo Rate Recovery

Perform turbo rate matching and recovery.

```
trBlkLen = 135;
codewordLen = 450;
RV = 0;
trblockwithcrc = lteCRCEncode(zeros(trBlkLen,1),'24A');
codeblocks = lteCodeBlockSegment(trblockwithcrc);
turbocodedblocks = lteTurboEncode(codeblocks);
codeword = lteRateMatchTurbo(turbocodedblocks,codewordLen,RV);
rateRecovered = lteRateRecoverTurbo(codeword,trBlkLen,RV)
```

```
    rateRecovered =

    [492x1 int8]
```

The output variable, `rateRecovered`, is a cell array with a single code block of length 492.

Input Arguments

`in` — Input data

numeric vector

Input data, specified as a numeric vector.

Data Types: `double` | `uint8` | `uint16` | `uint32` | `uint64` | `int8` | `int16` | `int32` | `int64`

`trblklen` — Length of original encoded transport block prior to encoding

numeric value

Length of the original encoded transport block prior to encoding, specified as a numeric value.

Data Types: `double`

`rv` — Redundancy version used to recover data

0 | 1 | 2 | 3

Redundancy version used to recover data, specified as 0, 1, 2, or 3. The bit selection recovery assumes a QPSK transmission mapped onto a single layer. It also assumes no restriction on the number of soft bits i.e. an uplink transport channel.

Data Types: `double`

`chs` — Channel transmission configuration

structure

Channel transmission configuration, specified as a structure. It allows additional control of the bit selection stage through parameters for the soft buffer size and physical channel configuration. For downlink turbo coded transport channels, the soft buffer dimensions can be controlled by one of the following sets of optional chs fields, in order of precedence.

- NIR — Soft buffer size for entire input transport block
- All of NSoftbits, TxScheme, and DuplexMode. If DuplexMode is `'TDD'`, also specify TDDConfig and NSubframe.

If neither the NIR nor NSoftbits fields are present, the function assumes an uplink turbo coded transport channel and places no limit on the number of soft bits.

chs can contain the following fields.

`Modulation` — Modulation scheme

`'QPSK' | '16QAM' | '64QAM'`

Modulation scheme, specified as a string. Accepted values are `'QPSK'`, `'16QAM'`, and `'64QAM'`.

Data Types: `char`

`NLayers` — Number of transmission layers for transport block

1 | 2 | 3 | 4

Number of transmission layers for transport block, specified as 1, 2, 3, or 4. Not necessary if TxScheme is set to `'Port0'`, `'TxDiversity'`, or `'Port5'`.

Data Types: `double`

`TxScheme` — Transmission scheme

`'Port0'` (default) | Optional | `'TxDiversity'` | `'CDD'` | `'SpatialMux'` | `'MultiUser'` | `'Port5'` | `'Port7-8'` | `'Port8'` | `'Port7-14'`

Transmission scheme, specified as a string. Optional. Accepted values and their descriptions are shown in the following table

Transmission scheme	Description
`'Port0'`	Single antenna port, port 0. Default
`'TxDiversity'`	Transmit diversity
`'CDD'`	Large delay CDD
`'SpatialMux'`	Closed loop spatial multiplexing
`'MultiUser'`	Multi-user MIMO
`'Port5'`	Single-antenna port, Port 5
`'Port7-8'`	Single-antenna port, port 7, if NLayers is set to 1. Dual layer transmission, port 7 and 8, if NLayers is set to 2.
`'Port8'`	Single-antenna port, Port 8
`'Port7-14'`	Up to 8 layer transmission, ports 7-14

Data Types: `char`

`NIR` — Soft buffer size for entire input transport block

nonnegative integer

Soft buffer size for entire input transport block, specified as a nonnegative integer.

Data Types: `double`

`NSoftbits` — Total number of soft channel bits

nonnegative integer

Total number of soft channel bits, specified as a nonnegative integer.

Data Types: `double`

`DuplexMode` — Duplex mode

`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, specified as a string. Optional. Accepted values are `'FDD'` and `'TDD'`.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration

0 (default) | Optional | nonnegative scalar integer (0...6)

Uplink or downlink configuration, specified as a nonnegative scalar integer between 0 and 6. Optional. Only required if DuplexMode is set to `'TDD'`.

Data Types: `double`

`NSubframe` — Subframe number

integer

Subframe number, specified as an integer. Only required if DuplexMode is set to `'TDD'`

Data Types: `double`

Data Types: `struct`

`cbsbuffers` — Code block soft information buffers

cell array of numeric vectors | empty cell array | cell array of numeric scalar elements

Code block soft information buffers, specified as a cell array. This input argument represents any preexisting code block-oriented soft information that should be additively combined with the recovered turbo encoded code blocks. This input argument allows the direct soft combining of consecutive HARQ retransmissions and it would normally have been returned by a previous call to the function to recover an earlier transmission of the same transport block. It should be a cell array of the same dimensionality as the code blocks output, out, or can be empty to represent the processing of an initial HARQ transmission. The cell array elements can also be scalar to add a constant offset to all the deinterleaved soft data in a code block.

Data Types: `cell`

Output Arguments

`out` — Turbo encoded code blocks prior to concatenation

cell array of numeric column vectors

Turbo encoded code blocks prior to concatenation, returned as a cell array of numeric column vectors. The dimensions of out are deduced from trblklen, which represents the length of the original encoded transport block.

Data Types: `cell`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`lteDLSCHDecode` | `lteDLSCHInfo` | `lteRateMatchTurbo` | `lteRateRecoverConvolutional` | `lteTurboDecode` | `lteULSCHDecode` | `lteULSCHInfo`

lteResourceGrid

Subframe resource array

Syntax

```
grid = lteResourceGrid(cfg)
grid = lteResourceGrid(cfg,p)
```

Description

`grid = lteResourceGrid(cfg)` returns an empty resource array generated from the configuration settings structure, cfg. To create a downlink resource array, cfg must contain the `NDLRB` and `CellRefP` fields. To create an uplink resource array, cfg must contain the `NULRB` field. The presence of field `NDLRB` takes precedence over the field `NULRB`. To specifically create a downlink or uplink resource array, use `lteDLResourceGrid` or `lteULResourceGrid`. .

`lteResourceGrid` returns an empty multidimensional array used to represent the resource elements for one subframe across all configured antenna ports, as described in "Data Structures".

The size of grid is *N*-by-*M*-by-*P*. The variable *N* is the number of subcarriers, 12×`NDLRB`. The variable *M* is the number of OFDM or SC-FDMA symbols in a subframe, 14 for normal cyclic prefix and 12 for extended cyclic prefix. The variable *P* is the number of transmit antenna ports, which is cfg`.CellRefP` in the downlink and cfg`.NTxAnts` in the uplink.

`grid = lteResourceGrid(cfg,p)` returns a resource array where the number of antenna planes in the array is specified directly by the parameter p.

Examples

Create Downlink Subframe Resource Array

Create an empty resource array that represents the downlink resource elements for 10MHz bandwidth, one subframe, and two antennas.

```
griddl = lteResourceGrid(struct('NDLRB',50,'CellRefP',2));
size(griddl)
```

```
   600    14     2
```

Create Uplink Subframe Resource Array

Create an empty resource array that represents the uplink resource elements for 10MHz bandwidth, one subframe, and two antennas.

```
gridul = lteResourceGrid(struct('NULRB',50,'NTxAnts',2));
size(gridul)
```

```
   600    14     2
```

Input Arguments

`cfg` — Configuration settings

scalar structure

Configuration settings, specified as a scalar structure. For the downlink, cfg can contain the following fields.

Parameter Field	Required or Optional	Values	Description
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`CellRefP`	Required	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length

For the uplink, cfg can contain the following fields.

Parameter Field	Required or Optional	Values	Description
`NULRB`	Required	Positive scalar integer	Number of uplink (UL) resource blocks (RBs)

Parameter Field	Required or Optional	Values	Description
`CyclicPrefixUL`	Optional	`'Normal'` (default), `'Extended'`	Current cyclic prefix length
`NTxAnts`	Optional	1 (default), 2, 4	Number of transmission antenna ports. This argument is only present for UE-specific demodulation reference symbols.

p — Number of antenna planes in the output array

nonnegative scalar integer

Number of antenna planes in the output array, specified as a nonnegative scalar integer.

Data Types: `double`

Output Arguments

grid — Empty multidimensional resource grid

3-D numeric array

Empty multidimensional resource grid, returned as a 3-D numeric array. This argument is generated from the settings structure. It represents the resource elements for one subframe across all configured antenna ports. Its size is *N*-by-*M*-by-*P*. The variable *N* is the number of subcarriers, 12×`NDLRB`. The variable *M* is the number of OFDM or SC-FDMA symbols in a subframe, 14 for normal cyclic prefix and 12 for extended cyclic prefix. The variable *P* is the number of transmit antenna ports, cfg`.CellRefP` in the downlink and cfg`.NTxAnts` in the uplink.

Data Types: `double`

See Also

`lteDLResourceGrid` | `lteOFDMModulate` | `lteResourceGridSize` | `lteSCFDMAModulate` | `lteULResourceGrid`

lteResourceGridSize

Size of subframe resource array

Syntax

```
d = lteResourceGridSize(cfg)
d = lteResourceGridSize(cfg,p)
```

Description

`d = lteResourceGridSize(cfg)` returns a 3-element row vector of dimension lengths for the resource array generated from the settings structure, cfg. To create a downlink resource array, cfg must contain the `NDLRB` and `CellRefP` fields. To create an uplink resource array, cfg must contain the `NULRB` field. If both `NDLRB` and `NULRB` fields are defined, the presence of the field `NDLRB` takes precedence over the field `NULRB`. To get the dimension lengths specifically for a downlink or uplink resource array, use the function lteDLResourceGridSize or lteULResourceGridSize respectively. The function returns a 3-element row vector of dimension lengths for multidimensional array used to represent the resource elements for one subframe across all configured antenna ports, as described in “Data Structures”.

The vector d is [*N M P*]. The variable *N* is the number of subcarriers, 12×`NDLRB`. The variable *M* is the number of OFDM symbols in a subframe, 14 for normal cyclic prefix and 12 for extended cyclic prefix. The variable *P* is the number of transmit antenna ports, which is cfg`.CellRefP` in the downlink and cfg`.NTxAnts` in the uplink.

`d = lteResourceGridSize(cfg,p)` returns a 3-element row vector where the number of antenna planes in the array is specified directly by parameter p.

Examples

Get Downlink Subframe Resource Array Size

Get the downlink subframe resource array size from a downlink configuration structure. Then, use the returned vector to directly create a MATLAB array.

```
cfgdl = struct('NDLRB',6,'CellRefP',2,'CyclicPrefix','Normal');
griddl = zeros(lteResourceGridSize(cfgdl));
size(griddl)
```

```
    72    14     2
```

The result, `griddl`, is a resource array. This resource array could also be obtained in a similar manner using the `lteResourceGrid` function.

Get Uplink Subframe Resource Array Size

Get the uplink subframe resource array size from an uplink configuration structure. Then, use the returned vector to directly create a MATLAB array.

```
cfgul = struct('NULRB',6,'NTxAnts',2,'CyclicPrefixUL','Normal');
gridul = zeros(lteResourceGridSize(cfgul));
size(gridul)
```

```
    72    14     2
```

The result, `gridul`, is an uplink resource array. This resource array could also be obtained in a similar manner using the `lteResourceGrid` function.

Input Arguments

`cfg` — Configuration settings

scalar structure

Configuration settings, specified as a scalar structure. For the downlink, cfg can contain the following fields.

Parameter Field	Required or Optional	Values	Description
`NDLRB`	Required	Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`CellRefP`	Required	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
`CyclicPrefix`	Optional	`'Normal'` (default), `'Extended'`	Cyclic prefix length

For the uplink, cfg can contain the following fields.

Parameter Field	Required or Optional	Values	Description
`NULRB`	Required	Positive scalar integer	Number of uplink (UL) resource blocks (RBs)
`CyclicPrefixUL`	Optional	`'Normal'` (default), `'Extended'`	Current cyclic prefix length
`NTxAnts`	Optional	1 (default), 2, 4	Number of transmission antenna ports. This argument is only present for UE-specific demodulation reference symbols.

p — Number of antenna planes

positive scalar integer

Number of antenna planes, specified as a positive scalar integer.

Data Types: `double`

Output Arguments

d — Dimension lengths of resource grid

numeric vector

Dimension lengths of resource grid, returned as a numeric vector of length 3. The elements are [*N M P*]. The variable *N* is the number of subcarriers, 12×`NDLRB`. The variable *M* is the number of OFDM or SC-FDMA symbols in a subframe, 14 for normal cyclic prefix and 12 for extended cyclic prefix. The variable *P* is the number of transmit antenna ports, cfg`.CellRefP` in the downlink and cfg`.NTxAnts` in the uplink.

Data Types: `double`

See Also

`lteDLResourceGridSize` | `lteResourceGrid` | `lteULResourceGridSize`

lteSCFDMADemodulate

SC-FDMA demodulation

Syntax

```
grid = lteSCFDMADemodulate(ue,waveform)
grid = lteSCFDMADemodulate(ue,waveform,cpfraction)
```

Description

`grid = lteSCFDMADemodulate(ue,waveform)` performs SC-FDMA demodulation of the time-domain waveform, waveform, given UE-specific settings structure, ue.

The demodulation performs one FFT operation per received SC-FDMA symbol. It recovers the received subcarrier values, which are then used to construct each column of the output resource array, grid. The FFT is positioned partway through the cyclic prefix, to allow for a certain degree of channel delay spread while avoiding the overlap between adjacent OFDM symbols. The input FFT is also shifted by half of one subcarrier. The particular position of the FFT chosen here avoids the SC-FDMA symbol overlapping used in the `lteSCFDMAModulate` function. Since the FFT is performed away from the original zero phase point on the transmitted subcarriers, a phase correction is applied to each subcarrier after the FFT.

`grid = lteSCFDMADemodulate(ue,waveform,cpfraction)` allows the specification of the position of the demodulation through the cyclic prefix.

Examples

Perform SC-FDMA Demodulation

Perform SC-FDMA demodulation of uplink fixed reference channel (FRC) A3-2.

```
frc = lteRMCUL('A3-2');
waveform = lteRMCULTool(frc,randi([0,1],frc.PUSCH.TrBlkSizes(1),1));
```

```
rgrid = lteSCFDMADemodulate(frc,waveform);
```

Input Arguments

ue — UE-specific settings

structure

UE-specific settings, specified as a structure. ue contains the following fields.

NULRB — Number of uplink resource blocks

positive scalar integer

Number of uplink resource blocks, specified as a positive scalar integer.

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length for uplink

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink, specified as a string. Optional. Valid values are `'Normal'` and `'Extended'`.

Data Types: `char`

Data Types: `struct`

waveform — Time-domain waveform

numeric matrix

Time-domain waveform, specified as a numeric matrix. The sampling rate of the time domain waveform waveform must be the same as used in the `lteSCFDMAModulate` modulator function for the specified number of resource blocks NULRB. waveform must also be time-aligned such that the first sample is the first sample of the cyclic prefix of the first SC-FDMA symbol in a subframe.

Data Types: `double`
Complex Number Support: Yes

cpfraction — Cyclic prefix fraction

`0.55` (default) | positive numeric scalar

Cyclic prefix fraction, specified as a positive numeric scalar between `0` and `1`. This argument specifies the position of the demodulation through the cyclic prefix. A value

of `0` represents the start of the cyclic prefix. A value of `1` represents the end of the cyclic prefix. The default value of `0.55` allows for the default level of windowing in the `lteSCFDMAModulate` function.

Data Types: `double`

Output Arguments

`grid` — Output resource array

numeric matrix

Output resource array, returned as a numeric matrix.

Data Types: `double`
Complex Number Support: Yes

See Also

`lteSCFDMAInfo` | `lteSCFDMAModulate` | `lteULChannelEstimate` | `lteULChannelEstimatePUCCH1` | `lteULChannelEstimatePUCCH2` | `lteULChannelEstimatePUCCH3` | `lteULFrameOffset` | `lteULFrameOffsetPUCCH1` | `lteULFrameOffsetPUCCH2` | `lteULFrameOffsetPUCCH3` | `lteULPerfectChannelEstimate`

lteSCFDMAModulate

SC-FDMA modulation

Syntax

```
[waveform,info] = lteSCFDMAModulate(ue,grid)
[waveform,info] = lteSCFDMAModulate(ue,grid,windowing)
```

Description

`[waveform,info] = lteSCFDMAModulate(ue,grid)` performs IFFT calculation, half-subcarrier shifting, and cyclic prefix insertions. It optionally performs raised-cosine windowing and overlapping of adjacent SC-FDMA symbols of the complex symbols in the resource array, grid.

For a block diagram that illustrates the steps in SC-FDMA modulation, see “Algorithms” on page 1-696.

`[waveform,info] = lteSCFDMAModulate(ue,grid,windowing)` allows control of the number of windowed and overlapped samples used in the time-domain windowing. If the value in `ue.Windowing` is present, it is ignored and the output, `info.Windowing`, equals windowing.

Examples

Perform SC-FDMA Modulation

Perform SC-FDMA modulation of one subframe of random uniformly-distributed noise, using a 10MHz configuration.

```
ue = struct('NULRB',50);
d = lteULResourceGridSize(ue);
rgrid = complex(rand(d)-0.5,rand(d)-0.5);
```

```
waveform = lteSCFDMAModulate(ue,rgrid);
```

Input Arguments

ue — UE-specific settings
structure

UE-specific settings, specified as a structure. If `ue.Windowing` is absent, `info.Windowing` will return a default value chosen as a function of `ue.NULRB` to compromise between the effective duration of cyclic prefix (and therefore the channel delay spread tolerance) and the spectral characteristics of the transmitted signal (not considering any additional FIR filtering). `ue.Windowing` must be even. With a value of zero, the issues above concerning concatenation of subframes before SC-FDMA modulation do not apply. ue contains the following fields.

NULRB — Number of uplink resource blocks
positive scalar integer

Number of uplink resource blocks, specified as a positive scalar integer.

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length for uplink
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink, specified as a string. Optional. Possible values are `'Normal'` and `'Extended'`.

Data Types: `char`

Windowing — Number of windowing samples
Absent (default) | Optional | positive scalar integer

Number of windowing samples, specified as a positive scalar integer. Optional. This argument is the number of time-domain samples over which windowing and overlapping of SC-FDMA symbols is applied. Default value is absent.

The number of samples used for windowing depends on the cyclic prefix length (normal or extended) and the number of resource blocks, and is chosen in accordance with the maximum values implied by tables E.5.1-1 and E.5.1-2 of [1]. The number of windowing samples is a compromise between the effective duration of cyclic prefix (and therefore

the channel delay spread tolerance) and the spectral characteristics of the transmitted signal (not considering any additional FIR filtering). For a larger amount of windowing, the effective duration of the cyclic prefix is reduced but the transmitted signal spectrum will have smaller out of band emissions.

Number of resource blocks `NULRB`	Windowing samples for normal cyclic prefix	Windowing samples for extended cyclic prefix
6	4	4
15	6	6
25	4	4
50	6	6
75	8	8
100	8	8

Data Types: `double`

Data Types: `struct`

`grid` — Resource grid

M-by-*N*-by-*P* numeric array

Resource grid, specified as an *M*-by-*N*-by-*P* numeric array array. The grid input contains *M* number of subcarriers, *N* number of SC-FDMA symbols, and *P* number of transmission antennas. The array contains resource elements (REs) for a number of subframes across all configured antenna ports, as described in "Data Structures". Alternatively, it contains multiple such matrices concatenated to give multiple subframes (concatenation across the columns or 2nd dimension). The antenna planes in grid are each OFDM modulated to give the columns of waveform.

Dimension *M* must be 12 ×`NULRB` where `NULRB` must be (`6...110`). Dimension *N* must be a multiple number of symbols in a subframe *L*, where *L*=14 for normal cyclic prefix and *L*=12 for extended cyclic prefix. Dimension *P* must be (`1,2,4`).

Note that grid can span multiple subframes and windowing and overlapping is applied between all adjacent SC-FDMA symbols, including the last of one subframe and the first of the next. Therefore a different result is obtained than if `lteSCFDMAModulate` is called on individual subframes and then those time-domain waveforms concatenated. The resulting waveform in that case would have discontinuities at the start/end of each subframe. Therefore it is recommended that all subframes for SC-FDMA modulation first

be concatenated prior to calling `lteSCFDMAModulate` on the resulting multi-subframe array. However, individual subframes can be OFDM modulated and the resulting multi-subframe time-domain waveform created by manually overlapping.

Data Types: `double`
Complex Number Support: Yes

`windowing` — Number of windowed and overlapped samples

positive scalar integer

Number of windowed and overlapped samples, specified as a positive scalar integer. This argument controls the number of windowed and overlapped samples used in time-domain windowing.

Data Types: `double`

Output Arguments

`waveform` — SC-FDMA modulated waveform

Numeric matrix

SC-FDMA modulated waveform, returned as a numeric matrix of size T-by-P, where T is the number of time-domain samples and P is the number of transmission antennas. $T = K \times 30720 / 2048 \times N_{fft}$ where N_{fft} is the IFFT size and K is the number of subframes in the input grid. N_{fft} is a function of the Number of Resource Blocks (NRB).

NRB	N_{fft}
6	128
15	256
25	512
50	1024
75	2048
100	2048

In general, N_{fft} is the smallest power of 2 greater than or equal to `12×NRB/0.85`. It is the smallest FFT that spans all subcarriers and results in a bandwidth occupancy ($12\times\text{NRB}/N_{\text{fft}}$) of no more than 85%.

Data Types: `double`
Complex Number Support: Yes

`info` — Information about SC-FDMA modulated waveform

Scalar structure

Information about SC-FDMA modulated waveform, returned as a scalar structure. info contains the following fields.

`SamplingRate` — Sampling rate of time-domain waveform

positive numeric scalar

Sampling rate of time-domain waveform, waveform, returned as a positive numeric scalar. This argument is given by the equation: $SamplingRate = 30.72\text{MHz} / 2048 \times N_{\text{fft}}$.

Data Types: `double`

`Nfft` — Number of FFT points

positive scalar integer

Number of FFT points, returned as a positive scalar integer.

Data Types: `double`

`Windowing` — Number of time-domain samples over which windowing and overlapping of SC-FDMA symbols is applied

positive scalar integer

Number of time-domain samples over which windowing and overlapping of SC-FDMA symbols is applied, returned as a positive scalar integer.

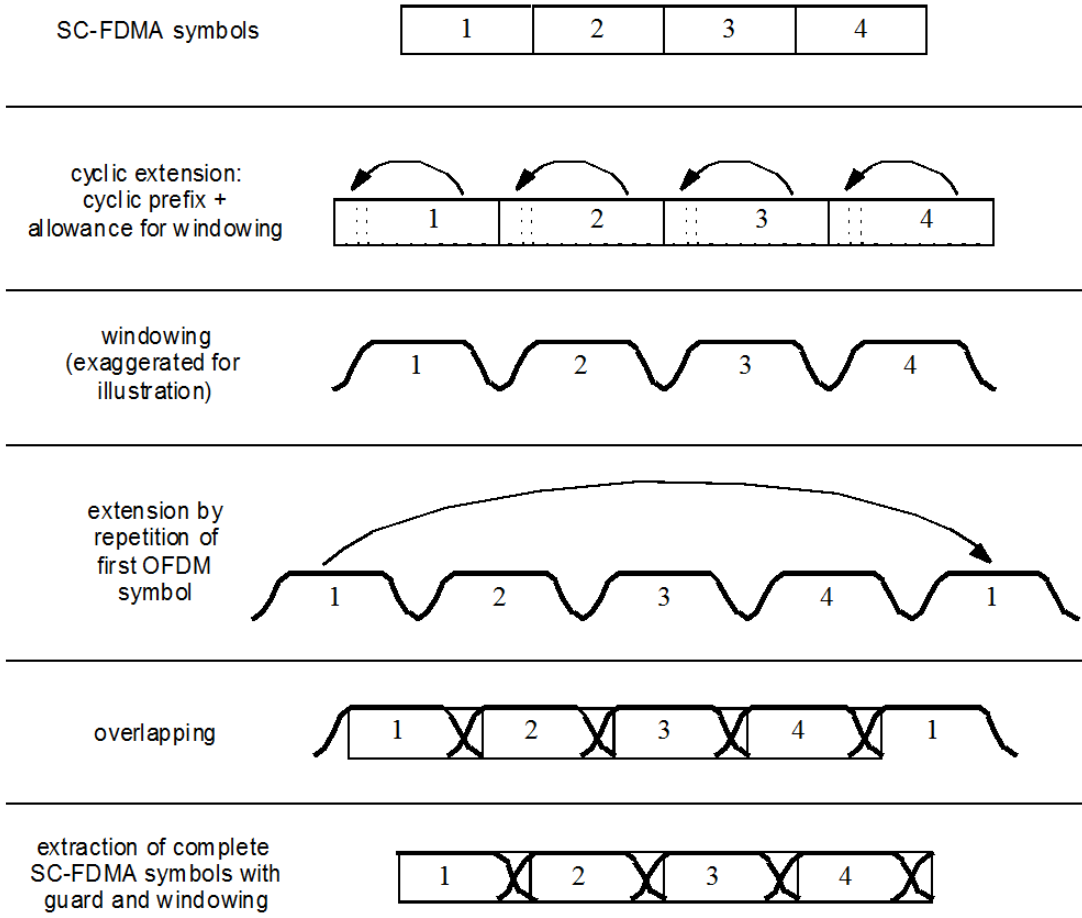
Data Types: `double`

Data Types: `struct`

More About

Algorithms

The following diagram illustrates the processing performed by SC-FDMA modulation.



References

[1] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteFadingChannel` | `lteHSTChannel` | `lteMovingChannel` | `lteSCFDMADemodulate` | `lteSCFDMAInfo` | `lteULResourceGrid` | `lteULResourceGridSize`

lteSCFDMAInfo

SC-FDMA modulation information

Syntax

```
info = lteSCFDMAInfo(ue)
```

Description

`info = lteSCFDMAInfo(ue)` provides information related to the SC-FDMA modulation performed by `lteSCFDMAModulate`, given the UE-specific settings structure, ue.

Examples

Get SC-FDMA Modulation Information

Find the sampling rate of a 50RB, or 10MHz, waveform after SC-FDMA modulation.

```
ue = struct('NULRB',50);
info = lteSCFDMAInfo(ue);
info.SamplingRate
```

```
    15360000
```

Input Arguments

ue — UE-specific settings

structure

UE-specific settings, specified as a structure. If the `ue.Windowing` field is absent, `info.Windowing` returns a default value chosen as a function of the `ue.NULRB` field. This mechanism acts as a compromise between the effective duration of cyclic prefix (and therefore the channel delay spread tolerance) and the spectral

characteristics of the transmitted signal (not considering any additional FIR filtering). See `lteSCFDMAModulate` for details. ue is a structure having these fields.

`NULRB` — Number of uplink resource blocks
scalar

Number of uplink resource blocks, specified as a scalar value.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`NTxAnts` — Number of transmission antennas
1 (default) | Optional | 2 | 4

Number of transmission antennas, specified as 1, 2, or 4. Optional.

Data Types: `double`

`Windowing` — Number of windowing samples
Optional | scalar integer

Number of windowing samples, specified as a scalar integer. Optional. This parameter is the number of time-domain samples over which windowing and overlapping of SC-FDMA symbols is applied. The default depends on previous parameters. See `lteSCFDMAModulate` for details.

Data Types: `double`

Data Types: `struct`

Output Arguments

`info` — Information related to SC-FDMA modulation
structure array

Information related to SC-FDMA modulation, returned as a structure array. The structure contains these fields.

`SamplingRate` — Sampling rate

numeric scalar

Sampling rate, returned as a numeric scalar. The function computes the sampling rate of the time domain waveform using the following equation: SamplingRate = 30720000 ÷ 2048 ×N_{fft}

Data Types: `double`

`Nfft` — Number of FFT points

numeric scalar

Number of FFT points used in the SC-FDMA modulator, returned as a numeric scalar.

Data Types: `double`

`Windowing` — Number of time-domain windowing samples

scalar integer

Number of time-domain windowing samples, returned as a scalar integer. This field represents the number of time-domain samples over which windowing and overlapping of SC-FDMA symbols is applied.

Data Types: `double`

Data Types: `struct`

See Also

`lteOFDMInfo` | `lteSCFDMADemodulate` | `lteSCFDMAModulate` | `lteULResourceGridSize`

lteSRS

Uplink sounding reference signal

Syntax

```
[seq,info] = lteSRS(ue,chs)
```

Description

`[seq,info] = lteSRS(ue,chs)` returns a complex matrix, seq, containing Uplink sounding reference signal (SRS) values and information structure array, info. The function returns these values for UE-specific settings, ue, and signal transmission configuration, chs. The symbols for each antenna are in the columns of seq, with the number of columns determined by the number of transmission antennas configured.

If type 1 triggered SRS transmission is intended (aperiodic SRS triggered by DCI formats 0/4/1A for FDD or TDD and DCI formats 2B/2C for TDD), parameter ConfigIdx indexes trigger type 1 UE-specific periodicity *T_SRS,1* and subframe offset *T_offset,1*. In this case, the valid range of ConfigIdx (*I_SRS*) is 0...16 for FDD and 0...24 for TDD.

If the NTxAnts parameter field is present in the chs structure, the function uses its value for the number of transmission antennas. Otherwise, the function uses the value of the same parameter field in the ue structure instead.

For short base reference sequences, used with SRS transmissions spanning between 1 and 4 PRBs, Zadoff-Chu sequences are not used. In this case, `RootSeq` and `NZC` are set to –1. For cases where the seq output is empty, when the SRS is not scheduled for transmission in this subframe, the info structure will contain all fields, but each field is set to –1.

Examples

Generate Uplink SRS Values

This example generates SRS values for 1.4 MHz bandwidth using the default SRS configuration.

Set the signal transmission configuration, chs structure fields.

```
chs.BWConfig = 7;
chs.BW = 0;
chs.CyclicShift = 0;
chs.SeqGroup = 0;
chs.SeqIdx = 0;
chs.ConfigIdx = 7;
```

Set ue structure fields.

```
ue.DuplexMode = 'FDD';
ue.CyclicPrefixUL = 'Normal';
ue.NTxAnts = 1;
ue.NFrame = 0;
ue.NULRB = 6;
ue.NSubframe = 0;
```

Generate Uplink SRS resource element values.

```
srs = lteSRS(ue,chs);
srs(1:4)
```

```
ans =

   0.7071 - 0.7071i
  -0.7071 + 0.7071i
   0.7071 + 0.7071i
  -0.7071 - 0.7071i
```

Input Arguments

ue — UE-specific settings

structure

UE-specific settings, specified as a structure. The structure contains the following fields.

NULRB — Number of uplink resource blocks

positive scalar integer

Number of uplink resource blocks, specified as a positive scalar integer.

Data Types: `double`

`NSubframe` — Number of subframes
0 (default) | Optional | scalar integer

Number of subframes, specified as a scalar integer. Optional.

Data Types: `double`

`NTxAnts` — Number of transmission antennas
1 (default) | Optional | 2 | 4

Number of transmission antennas, specified as 1, 2, or 4. Optional.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length for uplink
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink, specified as a string. Optional.

Data Types: `char`

`NFrame` — Initial frame number
0 (default) | Optional | numeric scalar

Initial frame number, specified as a numeric scalar. Optional.

Data Types: `double`

`DuplexMode` — Duplexing mode
`'FDD'` (default) | Optional | `'TDD'`

Duplexing mode, specified as a string. Optional. This string represents the frame structure type of the generated waveform.

Example: `'TDD'`

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration
0 (default) | Optional | nonnegative numeric scalar (0...6)

Uplink or downlink configuration, specified as a nonnegative numeric scalar between 0 and 6. Optional. Only required for `TDD` duplex mode.

Data Types: `double`

`SSC` — Special subframe configuration

0 (default) | Optional | nonnegative numeric scalar (0...9)

Special subframe configuration, specified as a nonnegative numeric scalar between 0 and 9. Optional. Only required for `TDD` duplex mode.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length in the downlink

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length in the downlink, specified as a string. Optional.

Data Types: `char`

Data Types: `struct`

`chs` — Signal transmission configuration

structure

Signal transmission configuration, specified as a structure. The structure contains these fields.

`NTxAnts` — Number of transmission antennas

1 (default) | Optional | 2 | 4

Number of transmission antennas, specified as 1, 2, or 4. Optional.

Data Types: `double`

`BWConfig` — SRS bandwidth configuration

7 (default) | Optional | 0...7

SRS bandwidth configuration, specified as a nonnegative integer between 0 and 7. Optional. The default is 7. (*C_SRS*)

Data Types: `double`

`BW` — UE-specific SRS bandwidth

0 (default) | Optional | 0...3

UE-specific SRS bandwidth, specified as a nonnegative integer between 0 and 3. Optional. The default is 0. (*B_SRS*)

Data Types: `double`

`ConfigIdx` — Configuration index for UE-specific periodicity

7 (default) | Optional | 0...644

Configuration index for UE-specific periodicity, specified as a nonnegative integer between 0 and 644. Optional. This parameter contains the configuration index for UE-specific periodicity (*T_SRS*) and subframe offset (*T_offset*).

Data Types: `double`

`CyclicShift` — UE-specific cyclic shift

0 (default) | Optional | 0...7

UE-specific cyclic shift, specified as a nonnegative integer between 0 and 7. Optional. The default is 0. (*n_SRS^cs*)

Data Types: `double`

`SeqGroup` — PUCCH sequence group number

0 (default) | Optional | 0...29

PUCCH sequence group number, specified as a numeric value. Optional. (*u*)

Data Types: `double`

`SeqIdx` — Base sequence number

0 (default) | Optional | 1

Base sequence number, specified as either 0 or 1. Optional. (*v*)

Data Types: `double` | `logical`

`OffsetIdx` — SRS subframe offset

0 (default) | Optional | nonnegative numeric scalar (0...1)

SRS subframe offset choice for 2ms SRS periodicity, specified as a nonnegative numeric scalar between 0 and 1. Optional. Only required for `'TDD'` duplex mode. This parameter indexes the 2 SRS subframe offset entries in the row of table 8.2-2 of [1] for the SRS configuration index specified by the `ConfigIdx` parameter.

Data Types: `double`

Data Types: `struct`

Output Arguments

`seq` — Uplink SRS values
complex matrix

Uplink SRS values, returned as a complex matrix. The symbols for each antenna are in the columns of the matrix, seq. The number of columns is determined by the number of transmission antennas configured.

Data Types: `double`
Complex Number Support: Yes

`info` — Information related to SRS
structure

Information related to SRS, returned as a structure. The structure contains the following fields.

`Alpha` — Reference signal cyclic shift
numeric scalar

Reference signal cyclic shift, returned as a numeric scalar. (*alpha*)

Data Types: `double`

`SeqGroup` — Base sequence group number
nonnegative numeric scalar (0...29)

Base sequence group number, returned as a nonnegative numeric scalar between 0 and 29. (*u*)

Data Types: `double`

`SeqIdx` — Base sequence number
0 | 1

Base sequence number, returned as a 0 or 1. (*v*)

Data Types: `double`

`RootSeq` — Root Zadoff-Chu sequence index
numeric scalar

Root Zadoff-Chu sequence index, returned as numeric scalar. (*q*)

Data Types: `double`

`NZC` — Zadoff-Chu sequence length
numeric scalar

Zadoff-Chu sequence length, returned as a numeric scalar. (*NRS_ZC*)

Data Types: `double`

Data Types: `struct`

References

[1] 3GPP TS 36.213. "Physical layer procedures." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteCellRS` | `lteCSIRS` | `lteDMRS` | `ltePRS` | `lteSRSIndices` | `lteSRSInfo`

lteSRSIndices

Uplink SRS resource element indices

Syntax

```
[ind,info] = lteSRSIndices(ue,chs)
[ind,info] = lteSRSIndices(ue,chs,opts)
```

Description

`[ind,info] = lteSRSIndices(ue,chs)` returns a column vector of resource element (RE) indices given the UE-specific settings structure, ue, and signal transmission configuration, chs. It returns a matrix of resource element (RE) indices and information structure array, info, for the Uplink sounding reference signal (SRS). By default, the indices are returned in 1-based linear indexing form that can directly index elements of a resource matrix. These indices are ordered as the SRS modulation symbols should be mapped. Alternative indexing formats can also be generated. The indices for each antenna are in the columns of ind, with the number of columns determined by the number of transmission antennas configured.

If type 1 triggered SRS transmission is intended, aperiodic SRS triggered by DCI formats 0/4/1A for FDD or TDD and DCI formats 2B/2C for TDD, parameter chs.ConfigIdx indexes trigger type 1 UE-specific periodicity, *T_SRS*, of 1 and subframe offset, *T_offset*, of 1. In this case, the valid range of chs.ConfigIdx (*I_SRS*) is 0 to 16 for FDD and 0...24 for TDD.

For type 1 triggered SRS transmission, frequency hopping is not permitted. Therefore, chs.HoppingBW (*b_hop*) should be set such that chs.HoppingBW`>=BW` (*B_SRS*).

The UE-specific SRS periodicity, info.UePeriod, and subframe offset, info.UeOffset, are distinct from the cell-specific SRS periodicity and subframe offset returned from the `lteSRSInfo` function.

For SRS transmission on multiple antennas, when chs.NTxAnts is set to 2 or 4, the value of info.Port simply matches the position in the structure array, 0...`NTxAnts`–1. If chs.NTxAnts is set to 1,info.Port is used to indicate the port chosen by SRS transmit antenna selection. info.Port is 0 or 1 indicating the selected antenna port. If the

NTxAnts parameter field is not present in the chs structure, the function uses the value of the same parameter field in the ue structure instead. If the NTxAnts parameter field is not present in the ue structure, the function assumes one antenna.

`[ind,info] = lteSRSIndices(ue,chs,opts)` allows control of the format of the returned indices through a cell array, opts, of option strings.

Examples

Generate Uplink SRS Indices

This example creates SRS indices for 3 MHz bandwidth using the default SRS configuration.

Set the signal transmission configuration, `chs` structure fields.

```
chs.NTxAnts = 1;
chs.BWConfig = 7;
chs.BW = 0;
chs.ConfigIdx = 7;
chs.TxComb = 0;
chs.HoppingBW = 0;
chs.FreqPosition = 0;
```

Set `ue` structure fields.

```
ue.DuplexMode = 'FDD';
ue.CyclicPrefixUL = 'Normal';
ue.NFrame = 0;
ue.NULRB = 15;
ue.NSubframe = 0;
```

Generate the Uplink SRS resource element indices.

```
srsIndices = lteSRSIndices(ue,chs);
srsIndices(1:4)
```

```
ans =

        2401
        2403
        2405
```

2407

Input Arguments

`ue` — UE-specific settings
structure

UE-specific settings, specified as a structure. The structure contains the following fields.

`NULRB` — Number of uplink resource blocks
positive scalar integer

Number of uplink resource blocks, specified as a positive scalar integer.

Data Types: `double`

`NSubframe` — Number of subframes
0 (default) | Optional | numeric scalar

Number of subframes, specified as a numeric scalar. Optional.

Data Types: `double`

`NTxAnts` — Number of transmission antennas
1 (default) | Optional | 2 | 4

Number of transmission antennas, specified as a 1, 2, or 4. Optional.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`NFrame` — Initial frame number
0 (default) | Optional | numeric scalar

Initial frame number, returned as a numeric scalar. Optional.

Data Types: `double`

`DuplexMode` — Duplexing mode

`'FDD'` (default) | Optional | `'TDD'`

Duplexing mode, specified as a string. Optional. This string represents the frame structure type of the generated waveform.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration

0 (default) | Optional | nonnegative numeric scalar (0...6)

Uplink or downlink configuration, returned as a nonnegative numeric scalar between 0 and 6. Optional. Only required for `'TDD'` duplex mode.

Data Types: `double`

`SSC` — Special subframe configuration

0 (default) | Optional | nonnegative numeric scalar (0...9)

Special subframe configuration, returned as a numeric scalar between 0 and 9. Optional. Only required for `'TDD'` duplex mode.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, returned as a string. Optional. Only required for `'TDD'` duplex mode.

Data Types: `char`

Data Types: `struct`

`chs` — Signal transmission configuration

structure

Signal transmission configuration, specified as a structure. The structure contains these fields.

`NTxAnts` — Number of transmission antennas

1 (default) | Optional | 2 | 4

Number of transmission antennas, specified as a 1, 2, or 4. Optional.

Data Types: `double`

`BWConfig` — Cell-specific SRS bandwidth configuration

7 (default) | Optional | 0...7

Cell-specific SRS bandwidth configuration, specified as a nonnegative scalar integer. Optional. (*C_SRS*)

Data Types: `double`

`BW` — UE-specific SRS bandwidth

0 (default) | Optional | 0...3

UE-specific SRS bandwidth, specified as a nonnegative integer between 0 and 3. Optional. (*B_SRS*)

Data Types: `double`

`ConfigIdx` — Configuration index for UE-specific periodicity

7 (default) | Optional | 0...644

Configuration index for UE-specific periodicity, specified as a nonnegative integer between 0 and 644. Optional. This parameter contains the configuration index for UE-specific periodicity (*T_SRS*) and subframe offset (*T_offset*).

Data Types: `double`

`TxComb` — Transmission comb

0 (default) | Optional | 1

Transmission comb, specified as a 0 or 1. Optional. This parameter controls SRS positions. SRS is transmitted in 6 carriers per resource block on odd (1) and even (0) resource indices.

Data Types: `double` | `logical`

`HoppingBW` — SRS frequency hopping configuration index

0 (default) | Optional | 0...3

SRS frequency hopping configuration index, specified as a nonnegative scalar integer between 0 and 3. Optional. (*b_hop*)

Data Types: `double`

`FreqPosition` — Frequency-domain position

0 (default) | Optional | 0...23

Frequency-domain position, specified as a nonnegative scalar integer between 0 and 23. Optional. (*n_RRC*)

Data Types: `double`

`CyclicShift` — UE-specific cyclic shift
0 (default) | Optional | 0...7

UE-specific cyclic shift, specified as a numeric value between 0 and 7. Optional. This parameter applies only when `NTxAnts` is 4. (*n_SRS^cs*)

Data Types: `double`

`NF4RachPreambles` — Number of RACH preamble frequency resources of format 4 in UpPTS
0 (default) | Optional | 0...6

Number of RACH preamble frequency resources of format 4 in UpPTS, specified as a nonnegative scalar integer between 0 and 6. Optional. Only required for `'TDD'` duplex mode.

Data Types: `double`

`OffsetIdx` — SRS subframe offset
0 (default) | Optional | numeric scalar (0...1)

SRS subframe offset choice for 2ms SRS periodicity, specified as a numeric scalar between 0 and 1. Optional. Only required for `'TDD'` duplex mode. This parameter indexes the 2 SRS subframe offset entries in the row of table 8.2-2 of [1] for the SRS configuration index specified by the `ConfigIdx` parameter.

Data Types: `double`

`MaxUpPts` — Reconfiguration enable and disable
1 (default) | Optional | 0

Reconfiguration enable and disable, specified as 0 or 1. Optional. Only required for `'TDD'` duplex mode. Enables (1) or disables (0) reconfiguration of *m_SRS,0^max* in UpPTS. (*srs-MaxUpPts*)

Data Types: `double` | `logical`

Data Types: `struct`

`opts` — Option strings for format of returned indices
string | cell array of strings

Option strings for format of returned indices, specified as a string or a cell array of strings. It can include the following option strings.

`Indexing style` — Form of returned indices

`'ind'` (default) | `'sub'`

Form of returned indices, specified as `'ind'` or `'sub'`. The options are linear index form or [*subcarrier*, *symbol*, *port*] subscript row form.

Data Types: `char`

`Index base` — Index base of returned indices

`'1based'` (default) | `'0based'`

Index base of returned indices, specified as `'1based'` or `'0based'`.

Data Types: `char`

Data Types: `char | cell`

Output Arguments

`ind` — Antenna indices

numeric matrix

Antenna indices, returned as a numeric matrix. Each column of RE indices corresponds to a transmission antenna.

Data Types: `uint32`

`info` — Information related to SRS

structure array

Information related to SRS, returned as a structure array. The structure contains the following fields.

`UePeriod` — UE-specific SRS periodicity

2 | 5 | 10 | 20 | 40 | 80 | 160 | 320

UE-specific SRS periodicity, in ms, returned as a positive scalar integer.

Data Types: `double`

`UeOffset` — UE-specific SRS offset
`0...319`

UE-specific SRS offset, returned as a nonnegative scalar integer between `0` and `319`.

Data Types: `double`

`PRBSet` — Physical resource block set
numeric vector

Physical resource block set, returned as a numeric vector. This vector specifies the PRBs occupied by the indices in each slot of the subframe (0-based).

Data Types: `double`

`FreqStart` — Frequency-domain starting position
numeric scalar

Frequency-domain starting position (*k_0*), returned as a numeric scalar. This argument is the 0-based subcarrier index of the lowest SRS subcarrier.

Data Types: `double`

`KTxComb` — Offset to the frequency-domain starting position
numeric scalar

Offset to the frequency-domain starting position (*k_TC*), returned as a numeric scalar. This argument is a function of the transmission comb parameter.

Data Types: `double`

`BaseFreq` — Base frequency-domain starting position
numeric scalar

Base (cell-specific) frequency-domain starting position (*k_0 bar*), returned as a numeric scalar. This UE-specific SRS is offset as a function of the UE-specific SRS bandwidth value, *B_SRS*. UE-specific SRS configuration cannot result in a frequency-domain starting position (*k_0*) lower than this value, given the cell-specific SRS bandwidth configuration value, *C_SRS*.

Data Types: `double`

`Freqidx` — Frequency position index
numeric vector

Frequency position index, returned as a numeric vector. This argument specifies the frequency position index (n_b) for each b in the range 0...B_SRS.

Data Types: `double`

`HoppingOffset` — Offset term due to frequency hopping
numeric vector

Offset term due to frequency hopping, returned as a numeric vector. This argument specifies the offset term due to frequency hopping (F_b), used in the calculation of n_b.

Data Types: `double`

`NSRSTx` — Number of UE-specific SRS transmissions
positive scalar integer

Number of UE-specific SRS transmissions (n_SRS), returned as a positive scalar integer.

Data Types: `double`

`Port` — Antenna port number used for transmission
positive scalar integer

Antenna port number used for transmission (p), returned as a positive scalar integer.

Data Types: `double`

Data Types: `struct`

References

[1] 3GPP TS 36.213. “Physical layer procedures.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also
`lteCellRSIndices` | `lteCSIRSIndices` | `lteDMRSIndices` | `ltePRSIndices` | `lteSRS` | `lteSRSInfo`

lteSRSInfo

Uplink SRS information

Syntax

```
info = lteSRSInfo(ue,chs)
```

Description

`info = lteSRSInfo(ue,chs)` returns information related to the sounding reference signal (SRS) configuration determined by UE-specific settings, ue, and signal transmission configuration, chs. The information returned relates to the cell-specific SRS subframe configuration as described in section 5.5.3.3 of [1].

This function returns information related to cell-specific SRS subframe configuration as described in section 5.5.3.3 of [1]. Information relating to a particular UE, such as UE-specific SRS configuration defined in section 8.2 of [2], is captured by the operation of the `lteSRSIndices` and `lteSRS` components. For a given configuration, if either of these components returns an empty vector, the SRS is not transmitted for that UE in the specified subframe.

Examples

Get Information Related to SRS

Correctly control PUCCH format 1 shortening on the basis of the cell-wide SRS configuration.

Set up the SRS and PUCCH format 1 to be consistent with "*Simultaneous-ACK/NACK-and-SRS*=True" from section 8.2 of [2].

```
ue = lteRMCUL('A1-1');
srs = struct('SubframeConfig',0);
srsInfo = lteSRSInfo(ue,srs);
ue.Shortened = srsInfo.IsSRSSubframe;
pucchSymbols = ltePUCCH1(ue,ue.PUSCH,[]);
```

For the case of "*Simultaneous-ACK/NACK-and-SRS*=False," the PUCCH would be transmitted with `ue.Shortened` equal to 0, which is the default case.

Input Arguments

`ue` — UE-specific settings
structure

UE-specific settings, specified as a structure. ue contains the following fields.

`NSubframe` — Subframe number
Optional | numeric scalar

Subframe number, specified as a numeric scalar. Optional.

Data Types: `double`

`DuplexMode` — Duplex mode
`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, specified as a string. Optional.

Data Types: `char`

`TDDConfig` — Uplink or downlink configuration
0 (default) | Optional | 0...6

Uplink or downlink configuration, specified as a nonnegative scalar integer between 0 and 6. Optional. Only required for `'TDD'` duplex mode.

Data Types: `double`

`SSC` — Special subframe configuration
0 (default) | Optional | 0...9

Special subframe configuration, specified as a nonnegative scalar integer between 0 and 9. Optional. Only required for `'TDD'` duplex mode.

Data Types: `double`

Data Types: `struct`

`chs` — Signal transmission configuration
structure

Signal transmission configuration, specified as a structure. chs contains the following fields.

`SubframeConfig` — SRS subframe configuration
0...15

SRS subframe configuration, specified as a nonnegative scalar integer between 0 and 15.

Data Types: `double`

Data Types: `struct`

Output Arguments

`info` — Information related to the SRS configuration
structure

Information related to the SRS configuration, returned as a structure. info contains the following fields.

`CellPeriod` — Cell-specific SRS periodicity
1 | 2 | 5 | 10

Cell-specific SRS periodicity, in ms, returned as a positive scalar integer. Valid values are 1, 2, 5, and 10.

Data Types: `uint32`

`CellOffset` — Cell-specific SRS offsets
0...9

Cell-specific SRS offsets, returned as a nonnegative scalar integer between 0 and 9.

Data Types: `int32`

`IsSRSSubframe` — SRS subframe flag
1 | 0

SRS subframe flag, returned as 1 or 0. Present only if ue contains `NSubframe`. If `NSubframe` satisfies the expression `mod(NSubframe,CellPeriod)==CellOffset`, this value is 1. Otherwise, this value is 0.

Data Types: `uint32`

Data Types: `struct`

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.213. "Physical layer procedures." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteSRS` | `lteSRSIndices`

lteSSS

Secondary synchronization signal

Syntax

```
sss = lteSSS(enb)
```

Description

`sss = lteSSS(enb)` returns a complex column vector containing the secondary synchronization signal (SSS) values for cell-wide settings in structure enb.

This signal is only defined for subframes 0 and 5; therefore, an empty vector is returned for other values of `NSubframe`. This allows this function and the corresponding indices function `lteSSSIndices` to be used to index the resource grid, as described in "Resource Grid Indexing", for any subframe number, but the resource grid is only modified in subframes 0 and 5.

Examples

Generate SSS Values

Generate secondary synchronization signal (SSS) values for a physical layer cell identity of 1.

```
sss = lteSSS(struct('NCellID',1,'NSubframe',0));
sss(1:4)
```

```
     1
    -1
     1
     1
```

Input Arguments

enb — Cell-wide settings

structure

Cell-wide settings, specified as a structure. This structure can contain the following fields.

`NCellID` — Physical layer cell identity
integer

Physical layer cell identity, specified as an integer.

Data Types: `double`

`NSubframe` — Subframe number
0 (default) | Optional | integer

Subframe number, specified as an integer.

Data Types: `double`

Data Types: `struct`

Output Arguments

`sss` — Secondary synchronization signal
complex column vector

Secondary synchronization signal (SSS), returned as a complex column vector. The vector contains the SSS values for cell-wide settings in the enb structure.

Data Types: `double`
Complex Number Support: Yes

See Also

`ltePSS` | `lteSSSIndices`

lteSSSIndices

SSS resource element indices

Syntax

```
ind = lteSSSIndices(enb)
ind = lteSSSIndices(enb,port)
ind = lteSSSIndices(___,opts)
```

Description

`ind = lteSSSIndices(enb)` returns a column vector of resource element indices, port 0 oriented, given the parameter fields of structure, enb. It returns a column vector of resource element (RE) indices for the Secondary Synchronization Signal (SSS). By default, the indices are returned in 1-based linear indexing form that can directly index elements of a 3-D array representing the resource array. These indices are ordered as the SSS modulation symbols should be mapped. Alternative indexing formats can also be generated.

These indices are only defined for subframes 0 and 5; therefore, an empty vector is returned for other values of `NSubframe`. This allows this function and the corresponding sequence function, `lteSSS`, to be used to index the resource grid, as described in "Resource Grid Indexing", for any subframe number. However, the resource grid is only modified in subframes 0 and 5.

`ind = lteSSSIndices(enb,port)` returns indices appropriate for an antenna port, port, which must be either 0, 1, 2, or 3.

`ind = lteSSSIndices(___,opts)` allows control of the format of the returned indices through a cell array, opts, of option strings.

Examples

Generate SSS RE Indices in Linear Form

Get 0-based SSS resource element indices in linear form for antenna port 0.

```
enb = lteRMCDL('R.4');
sssIndices = lteSSSIndices(enb,0,{'0based','ind'});
sssIndices(1:4)
```

```
         365
         366
         367
         368
```

Generate SSS RE Indices in Subscript Form

Generate 0-based SSS resource element indices in subscript form for antenna port 0. In this case, a matrix is generated where each row has three columns representing subcarrier, symbol, and antenna port.

Generate 0-based SSS resource element indices in subscript form for antenna port 0.

```
enb = lteRMCDL('R.4');
sssIndices = lteSSSIndices(enb,0,{'0based','sub'});
sssIndices(1:4,:)
```

```
           5           5           0
           6           5           0
           7           5           0
           8           5           0
```

The first column of the output represents subcarrier. The second column represents symbol. The third column represents antenna port.

Input Arguments

`enb` — Cell-wide settings

structure

Cell-wide settings, specified as a structure. It contains the following fields.

`NDLRB` — Number of downlink resource blocks

positive integer (6...110)

Number of downlink resource blocks, specified as an integer between 6 and 110.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`NSubframe` — Subframe number
0 (default) | Optional | integer

Subframe number, specified as an integer. Optional.

Data Types: `double`

`DuplexMode` — Duplex mode
`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, specified as `'FDD'` or `'TDD'`. Optional.

Data Types: `char`

Data Types: `struct`

`port` — Antenna port
0 | 1 | 2 | 3

Antenna port, specified as 0, 1, 2, or 3.

Data Types: `double`

`opts` — Format options for control of returned indices
string | cell array of strings

Format options for control of returned indices, specified as a string or a cell array strings. It can contain the following option strings.

`Indexing style` — Form of returned indices
`'ind'` (default) | `'sub'`

Form of returned indices, specified as `'ind'` or `'sub'`. The `'ind'` options represents the linear index form, and the `'sub'` string represents the subcarrier, symbol, antenna subscript row form.

Data Types: `char`

`Index base` — Index base of returned indices

`'1based'` (default) | `'0based'`

Index base of returned indices, specified as `'1based'` or `'0based'`.

Data Types: `char`

Data Types: `char` | `cell`

Output Arguments

`ind` — SSS resource element (RE) indices

column vector

SSS resource element (RE) indices, returned as a column vector. By default, the indices are returned in 1-based linear indexing form that can directly index elements of a 3D array representing the resource array. These indices are ordered as the SSS modulation symbols should be mapped. Alternative indexing formats can also be generated.

Data Types: `double`

See Also

`ltePSSIndices` | `lteSSS`

lteSymbolDemodulate

Demodulation and symbol to bit conversion

Syntax

```
out = lteSymbolDemodulate(in,mod)
out = lteSymbolDemodulate(in,mod,dec)
```

Description

`out = lteSymbolDemodulate(in,mod)` returns a column vector containing bits resulting from soft constellation demodulation of complex values in vector in. The modulation format, mod, can be either `'QPSK'`, `'16QAM'`, or `'64QAM'`.

Demodulation is performed assuming the constellations given in the LTE specification, including the power normalization factors specified. The constellation expected is given by the output of `lteSymbolModulate`, including the following factors.

- `1/sqrt(2)` for `'QPSK'`
- `1/sqrt(10)` for `'16QAM'`
- `1/sqrt(42)` for `'64QAM'`

The demodulation algorithm assumes the in vector to contain coherently received symbols that are normalized to fall on these constellation points.

`out = lteSymbolDemodulate(in,mod,dec)` allows the decision mode, dec, to be specified as either `'Hard'` or `'Soft'`. For hard-decision decoding, the output, out, contains the bit sequences corresponding to the closest constellation points to the input, in. For soft-decision decoding, the output, out, indicates the bit values using the sign (–*ve* for 0, +*ve* for 1) and the magnitude of the output gives a piecewise linear approximation to the log likelihood ratio (LLR) of the demodulated bits. The algorithm used for the LLR approximation is described in [1]. The returned LLRs are scaled such that for a input signal lying on the constellation points in the preceding description, the output values lie on the points with the following magnitudes.

- `1/sqrt(2)` for `'QPSK'`
- `[1 3]/sqrt(10)` for `'16QAM'`

- `[1 3 5 7]/sqrt(42)` for `'64QAM'`

Examples

Demodulate Complex-Valued Symbols

Demodulate complex-valued symbols.

```
out = lteSymbolDemodulate([0.7-0.7i;-0.7+0.7i],'QPSK','Hard')
```

```
     0
     1
     1
     0
```

Input Arguments

`in` — Input symbols to demodulate

numeric column vector

Input symbols to demodulate, specified as a column vector of complex numeric values. This vector is assumed to contain coherently received symbols that are normalized to fall on the constellation points.

Example: `[0.7 - 0.7i; -0.7 + 0.7i]`

Data Types: `double`
Complex Number Support: Yes

`mod` — Modulation format

`'QPSK' | '16QAM' | '64QAM'`

Modulation format, specified as a string.

Data Types: `char`

`dec` — Decision mode

`'Hard' | 'Soft'`

Decision mode, specified as a string. `'Hard'` decision mode results in the output containing the bit sequences corresponding to the closest constellation points to the input.

`'Soft'` decision mode results in the output indicating the bit values using the sign (`-ve` for 0, `+ve` for 1). For `'Soft'` decision mode, the magnitude of the output gives a piecewise linear approximation to the log likelihood ratio (LLR) of the demodulated bits. The algorithm used for the LLR approximation is described in [1]. The returned LLRs are scaled such that for a input signal lying on the constellation points in the preceding description, the output values lie on the points with the following magnitudes.

- `1/sqrt(2)` for `'QPSK'`
- `[1 3]/sqrt(10)` for `'16QAM'`
- `[1 3 5 7]/sqrt(42)` for `'64QAM'`

Data Types: `char`

Output Arguments

`out` — Demodulated output bits

numeric column vector

Demodulated output bits, returned as a numeric column vector. This argument contains bits resulting from soft constellation demodulation of complex values vector, in.

Data Types: `double`

References

[1] Tosato, F., and Bisaglia, P. "Simplified soft-output demapper for binary interleaved COFDM with application to HIPERLAN/2." *IEEE International Conference on Communications (ICC) 2002, Vol. 2*. pp. 664-668.

See Also

`lteLayerDemap` | `lteSymbolModulate` | `lteULDescramble`

lteSymbolModulate

Symbol modulation

Syntax

```
out = lteSymbolModulate(in,mod)
```

Description

`out = lteSymbolModulate(in,mod)` maps the bit values, in, to complex modulation symbols with the modulation scheme specified in the mod string.

Examples

Generate QPSK Modulated Symbols

Map bit values to QPSK modulated symbols.

```
out = lteSymbolModulate([0;1;1;0],'QPSK')
```

```
   0.7071 - 0.7071i
  -0.7071 + 0.7071i
```

Input Arguments

`in` — Input bits

column vector

Input bits, specified as a column vector, where each bit is either 0 or 1. The vector length must be a multiple of 2 for `QPSK` modulation, 4 for `16QAM` modulation, and 6 for `64QAM` modulation . The bit values must be 0 or 1.

Data Types: `double`

`mod` — Modulation scheme

`'QPSK' | '16QAM' | '64QAM'`

Modulation scheme, specified as a string. Valid values are `QPSK`, `16QAM`, and `64QAM`.

Data Types: `char`

Output Arguments

`out` — Complex modulated output symbols

column vector

Complex modulated output symbols, returned as a column vector. The symbols use the modulation scheme specified in mod.

Data Types: `double`
Complex Number Support: Yes

See Also

`lteDLPrecode` | `lteLayerMap` | `lteSymbolDemodulate` | `lteULScramble`

lteTBS

Transport block size lookup

Syntax

```
tbs = lteTBS(nprb)
tbs = lteTBS(nprb,itbs)
tbs = lteTBS(nprb,itbs,smnlayer)
```

Description

`tbs = lteTBS(nprb)` looks up the transport block size (TBS) table, as defined in table 7.1.7.2.1-1 of [1], and returns the relevant TBS or sizes. It returns the column of the table appropriate to the specified value of nprb, the number of physical resource blocks. Values for nprb are between 1 and 110. The returned column, tbs, has 27 rows, corresponding to transport block size indices from 0 to 26.

`tbs = lteTBS(nprb,itbs)` returns a vector of values of the table appropriate to the specified value of nprb, the number of physical resource blocks, and itbs, a vector of transport block size indices. Values for itbs are between 0 and 26. If a value in itbs is –1, this is interpreted as a discontinuous transmission (DTX). In this case, the function produces a corresponding tbs value of 0.

`tbs = lteTBS(nprb,itbs,smnlayer)` returns a vector of values of the table appropriate to the specified values of nprb and itbs as above, but with smnlayer indicating the number of spatial multiplexing layers to which this transport block is mapped. Values for smnlayer are between 1 and 4. For 2-layer spatial multiplexing, the function follows the rules in section 7.1.7.2.2 of [1]. For 3-layer spatial multiplexing, the function follows the rules in section 7.1.7.2.4 of [1]. For 4-layer spatial multiplexing, the function follows the rules in section 7.1.7.2.5 of [1].

The syntax tbs = `lteTBS`(nprb,itbs,`sm2layer`) was formerly provided, with `sm2layer` specifying whether the transport block was mapped to two-layer spatial multiplexing. A value of 1 specified that the block was mapped to two-layer spatial multiplexing and 0 specified that it was not. The `sm2layer` argument has been replaced with smnlayer with a value of 1 being interpreted as a single layer and a value of 0 no longer being valid. If you input a value of 0, the function produces a warning to indicate the change of syntax.

Examples

Generate Transport Block Sizes

Look up the valid transport block sizes (TBSs) for a single physical resource block (PRB) allocation.

```
tbs = lteTBS(1)
```

```
tbs =

          16
          24
          32
          40
          56
          72
         328
         104
         120
         136
         144
         176
         208
         224
         256
         280
         328
         336
         376
         408
         440
         488
         520
         552
         584
         616
         712
```

Input Arguments

`nprb` — Number of physical resource blocks

1,...,110

Number of physical resource blocks, specified as a positive scalar integer from 1 to 110.

Data Types: `double`

`itbs` — Transport block size indices

numeric vector

Transport block size indices, specified as a numeric vector.

Data Types: `double`

`smnlayer` — Number of spatial multiplexing layers to which transport block is mapped

1,...,4

Number of spatial multiplexing layers to which transport block is mapped, specified as a positive scalar integer from 1 to 4.

Data Types: `double`

Output Arguments

`tbs` — Transport block size or sizes

column vector

Transport block size or sizes, returned as a column vector from the transport block size table. This vector contains 27 rows, corresponding to TBS indices from 0 to 26.

Data Types: `int32`

References

[1] 3GPP TS 36.213. "Physical layer procedures." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDLSCH` | `ltePDSCH`

lteTestModel

Downlink test model configuration structure

Syntax

```
tm = lteTestModel(tmn,bw)
tm = lteTestModel(tmn,bw,ncellid,duplexmode)
tm = lteTestModel(tmcfg)
```

Description

`tm = lteTestModel(tmn,bw)` returns the E-UTRA test model (E-TM) configuration structure, tm for given test model number, tmn, and bandwidth, bw, as specified in clause 6 of [1]. tm is a structure that contains the configuration parameters required to generate a given downlink E-TM waveform using the generator tool, `lteTestModelTool`. The field names and default values are in accordance with the definition found in clause 6 of [1].

The `PDSCH` is a substructure relating to the physical channel configuration and contains the fields `NLayers`, `TxScheme`, and `Modulation`.

`tm = lteTestModel(tmn,bw,ncellid,duplexmode)` controls the configuration of the physical cell identity, ncellid, and duplex mode, duplexmode.

`tm = lteTestModel(tmcfg)` returns tm, a configuration structure for the test model partially (or wholly) defined by the input structure tmcfg. The input structure tmcfg can define any (or all) of the parameters or substructure parameters and the output structure tm retains the defined parameters. The undefined fields are given appropriate default values.

The tm structure can be used in conjunction with the E-TM generator tool to generate a waveform. tm must contain at least the `TMN` and `BW` fields.

Examples

Create Downlink E-TM Configuration Structure

Create a configuration structure for Test Model E-TM 3.2 20 MHz, as specified in [1].

```
tm = lteTestModel('3.2','20MHz')
```

```
                    TMN: '3.2'
                     BW: '20MHz'
                  NDLRB: 100
                CellRefP: 1
                 NCellID: 1
            CyclicPrefix: 'Normal'
                    CFI: 1
                     Ng: 'Sixth'
           PHICHDuration: 'Normal'
               NSubframe: 0
            TotSubframes: 10
               Windowing: 0
              DuplexMode: 'FDD'
                   PDSCH: [1x1 struct]
             CellRSPower: 0
                PSSPower: 2.4260
                SSSPower: 2.4260
               PBCHPower: 2.4260
             PCFICHPower: 0
      NAllocatedPDCCHREG: 180
               PDCCHPower: 1.1950
        PDSCHPowerBoosted: 2.4260
     PDSCHPowerDeboosted: -3
```

Next, display the PDSCH substructure.

```
tm.PDSCH
```

```
      TxScheme: 'Port0'
    Modulation: {'16QAM'  'QPSK'}
       NLayers: 1
```

Input Arguments

`tmn` — Test model number

`'1.1' | '1.2' | '2' | '3.1' | '3.2' | '3.3'`

Test model number, specified as a string. See [1] for information on the types.

Data Types: `char`

`bw` — Channel bandwidth type

`'1.4MHz' | '3MHz' | '5MHz' | '10MHz' | '15MHz' | '20MHz' | '9RB' | '11RB' | '27RB' | '45RB' | '64RB' | '91RB'`

Channel bandwidth type, specified as a string. You can set the nonstandard bandwidths, `'9RB'`, `'11RB'`, `'27RB'`, `'45RB'`, `'64RB'`, and `'91RB'`, only when `TMN` is `'1.1'`. These nonstandard bandwidths specify custom test models.

Data Types: `char`

`ncellid` — Physical layer cell identity

1 or 10 (default) | Optional | scalar integer

Physical layer cell identity, specified as a scalar integer. Optional. If not specified, default is 1 for standard bandwidths and 10 for nonstandard bandwidths.

Data Types: `double`

`duplexmode` — Duplex mode

`'FDD'` (default) | `'TDD'`

Duplex mode, specified as a string.

Data Types: `char`

`tmcfg` — Test model configuration

Scalar structure

Test model configuration, specified as a scalar structure. Refer to the output tm for structure fields. If you define any or all listed parameters or substructure parameters in the input structure, tmcfg, the output structure, tm, retains the defined parameters. The undefined fields are given the appropriate default values.

Data Types: `struct`

Output Arguments

`tm` — E-UTRA test model (E-TM) configuration

Scalar structure

E-UTRA test model (E-TM) configuration, returned as a scalar structure. tm contains the following fields.

`TMN` — Test model number type

`'1.1' | '1.2' | '2' | '3.1' | '3.2' | '3.3'`

Test model number type, returned as a string.

Data Types: `char`

`BW` — Channel bandwidth type

`'1.4MHz' | '3MHz' | '5MHz' | '10MHz' | '15MHz' | '20MHz' | '9RB' | '11RB' | '27RB' | '45RB' | '64RB' | '91RB'`

Channel bandwidth type, in MHz, returned as a string. Non-standard bandwidths, `'9RB'`,`'11RB'`,`'27RB'`,`'45RB'`,`'64RB'`, and `'91RB'`, specify custom test models.

Data Types: `char`

`NDLRB` — Number of downlink resource blocks

nonnegative scalar integer

Number of downlink resource blocks, returned as a nonnegative scalar integer.

Data Types: `double`

`CellRefP` — Number of cell-specific reference signal antenna ports

1

Number of cell-specific reference signal antenna ports, returned as 1. This argument is for informational purposes and is read-only.

Data Types: `double`

`NCellID` — Physical layer cell identity

scalar integer

Physical layer cell identity, returned as a scalar integer.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length

`'Normal'`

Cyclic prefix length, returned as the string, `'Normal'`. This argument is for informational purposes and is read-only.

Data Types: `char`

`CFI` — Control format indicator value
`1 | 2 | 3`

Control format indicator value, returned as a positive scalar integer. Valid values are 1, 2, and 3.

Data Types: `double`

`Ng` — HICH group multiplier
`'Sixth' | 'Half' | 'One' | 'Two'`

HICH group multiplier, returned as a string.

Data Types: `char`

`PHICHDuration` — PHICH duration
`'Normal' | 'Extended'`

PHICH duration, returned as a string.

Data Types: `char`

`NSubframe` — Subframe number
nonnegative scalar integer

Subframe number, returned as a nonnegative scalar integer.

Data Types: `double`

`TotSubframes` — Total number of subframes to be generated
nonnegative scalar integer

Total number of subframes to be generated, returned as a nonnegative scalar integer.

Data Types: `double`

`Windowing` — Number of windowing samples
nonnegative scalar integer

Number of windowing samples, returned as a nonnegative scalar integer. This argument is the number of time-domain samples over which windowing and overlapping of OFDM symbols is applied.

Data Types: `double`

`DuplexMode` — Duplex mode
`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, returned as a string. Optional.

Data Types: `char`

`CellRSPower` — Cell-specific reference symbol power adjustment
numeric scalar

Cell-specific reference symbol power adjustment, in dB, returned as a numeric scalar.

Data Types: `double`

`PSSPower` — PSS symbol power adjustment
numeric scalar

PSS symbol power adjustment, in dB, returned as a numeric scalar.

Data Types: `double`

`SSSPower` — SSS symbol power adjustment
numeric scalar

SSS symbol power adjustment, in dB, returned as a numeric scalar.

Data Types: `double`

`PBCHPower` — PBCH symbol power adjustment
numeric scalar

PBCH symbol power adjustment, in dB, returned as a numeric scalar.

Data Types: `double`

`PCFICHPower` — PCFICH symbol power adjustment
numeric scalar

PCFICH symbol power adjustment, in dB, returned as a numeric scalar.

Data Types: `double`

`NAllocatedPDCCHREG` — Number of allocated PDCCH REGs
nonnegative scalar integer

Number of allocated PDCCH REGs, returned as a nonnegative scalar integer. This argument is derived from TMN and BW.

Data Types: `double`

`PDCCHPower` — PDCCH symbol power adjustment
numeric scalar

PDCCH symbol power adjustment, in dB, returned as a numeric scalar.

Data Types: `double`

`PDSCHPowerBoosted` — PDSCH symbol power adjustment for boosted PRBs
numeric scalar

PDSCH symbol power adjustment for boosted PRBs, in dB, returned as a numeric scalar.

Data Types: `double`

`PDSCHPowerDeboosted` — PDSCH symbol power adjustment for de-boosted PRBs
numeric scalar

PDSCH symbol power adjustment for de-boosted PRBs, in dB, returned as a numeric scalar.

Data Types: `double`

`TDDConfig` — Present only for TDD duplex mode. Uplink/Downlink Configuration
0 (default) | numeric scalar | 1...6

`TDDConfig` enumerates the subframe uplink-downlink configuration to be used in this frame. Clause 4.2 of [2] specifies uplink-downlink configurations (uplink, downlink and special subframe combinations).

Data Types: `double`

`SSC` — Present only for TDD duplex mode. Special Subframe Configuration
0 (default) | numeric scalar | 1...9

`SSC`enumerates the special subframe configuration. Clause 4.2 of [2] specifies the special subframe configurations (lengths of DwPTS, GP and UpPTS).

Data Types: `double`

`PDSCH` — PDSCH transmission configuration
scalar structure

PDSCH transmission configuration, returned as a scalar structure. PDSCH contains the following fields.

`NLayers` — Number of transmission layers

1

Number of transmission layers, returned as 1. This argument is for informational purposes and is read-only.

Data Types: `double`

`TxScheme` — Transmission scheme

`'Port0'`

Transmission scheme, returned as the string `'Port0'`. This argument is for informational purposes and is read-only. The E-TMs have a single antenna port, Port 0.

Data Types: `char`

`Modulation` — Modulation formats

`{'QPSK'}` | `{'16QAM'}` | `{'64QAM'}` | Cell array of two strings

Modulation formats, returned as a cell array of one or two strings specifying the modulation formats for boosted and deboosted PRBs.

Data Types: `cell`

Data Types: `struct`

Data Types: `struct`

References

[1] 3GPP TS 36.141. "Base Station (BS) conformance testing." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.211. "Physical channels and modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteRMCDL` | `lteRMCUL` | `lteTestModelTool`

lteTestModelTool

Downlink test model waveform generation

Syntax

```
[waveform,grid,tm] = lteTestModelTool
[waveform,grid,tm] = lteTestModelTool(tmn,bw,ncellid,duplexmode)
[waveform,grid,tm] = lteTestModelTool(tm)
```

Description

`[waveform,grid,tm] = lteTestModelTool` launches a graphical user interface (GUI) for the parameterization and generation of the E-UTRA test model (E-TM) waveforms. The main function outputs are specified in the GUI but can also be assigned to variables. waveform represents the generated E-TM time-domain waveform. It is a T-by-P array, where T is the number of time-domain samples and P is the number of antennas. Clause 6 of [1] fixes $P = 1$, making waveform a T-by-1 column vector. grid, the populated resource grid, is a 2-D array of resource elements for a number of subframes (10 for FDD and 20 for TDD) across a single antenna port, as specified in clause 6 of [1] and described in "Data Structures". tm is a structure containing information about the OFDM modulated waveform as described in `lteOFDMInfo` and test model specific configuration parameters as described in `lteTestModel`.

`[waveform,grid,tm] = lteTestModelTool(tmn,bw,ncellid,duplexmode)` returns the waveform, grid and tm for the test model number tmn and channel bandwidth bw. The ncellid and duplexmode are the optional input parameters and define the physical cell identity and duplex mode of the generated waveform, respectively.

The input tmn is a string identifying the test model number as per [1].

`[waveform,grid,tm] = lteTestModelTool(tm)` generates the same waveform, grid and tm in the same way as above except it takes the user defined test model configuration structure tm as input parameter. You can create the test model configuration structure with default parameters by using the function `lteTestModel` which is designed to generate the various tm configuration structures as per Clause 6 of

[1]. This configuration structure then can be modified as per requirements and can be used to generate the waveform.

Examples

Generate Downlink E-TM Waveform

Generate a time domain signal, waveform, and a 2-D array of the resource elements (REs) in grid for Test Model E-TM 3.2, 15MHz, in [1].

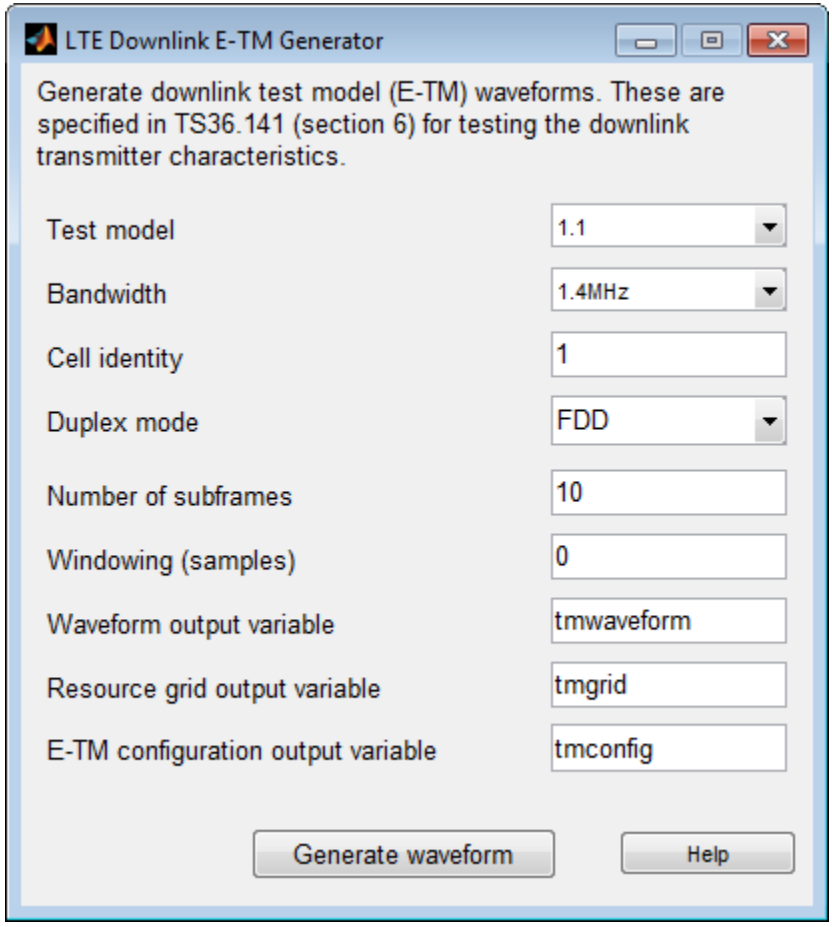
```
[waveform,rgrid,tm] = lteTestModelTool(lteTestModel('3.2','15MHz'));
```

Launch LTE Test Model Generator Tool

Launch the tool to generate an E-UTRA Test Model (E-TM) waveform.

```
lteTestModelTool;
```

The LTE Downlink E-TM Generator dialog box appears.



- "Generate a Test Model"
- "Generate LTE Test Model Waveforms"

Input Arguments

`tmn` — Test model number

`'1.1' | '1.2' | '2' | '3.1' | '3.2' | '3.3'`

Test model number, specified as a string. For more information on these test model numbers, see [1].

Example: `'3.2'`

Data Types: `char`

bw — Channel bandwidth

`'1.4MHz' | '3MHz' | '5MHz' | '10MHz' | '15MHz' | '20MHz' | '9RB' | '11RB' | '27RB' | '45RB' | '64RB' | '91RB'`

Channel bandwidth, specified as a string. You can set the nonstandard bandwidths, `'9RB'`,`'11RB'`,`'27RB'`,`'45RB'`,`'64RB'`, and `'91RB'`, only when `tmn` is `'1.1'`. These nonstandard bandwidths specify custom test models.

Example: `'15MHz'`

Data Types: `char`

ncellid — Physical layer cell identity

1 or 10 (default) | Optional | scalar integer

Physical layer cell identity, specified as a scalar integer. Optional. If you do not specify this argument, the default is 1 for standard bandwidths and 10 for non-standard bandwidths.

Example: 1

Data Types: `double`

duplexmode — Duplex mode of the generated waveform

`'FDD'` (default) | Optional | `'TDD'`

Duplex mode of the generated waveform, specified as a string. Optional.

Example: `'FDD'`

Data Types: `char`

tm — User-defined test model configuration

scalar structure

User-defined test model configuration, specified as a scalar structure. You can create the test model configuration structure with default parameters by calling `lteTestModel`, which is designed to generate the various tm configuration structures as per clause 6 of

[1]. You can then modify this configuration structure, as per requirements, and use it to generate waveform.

Data Types: `struct`

Output Arguments

`waveform` — Generated E-TM time-domain waveform

numeric matrix

Generated E-TM time-domain waveform, returned as a numeric matrix of size T-by-P, where P is the number of antennas and T is the number of time-domain samples.

Data Types: `double`
Complex Number Support: Yes

`grid` — Resource grid

numeric matrix

Resource grid, returned as a numeric matrix. This argument is a 2-D array of resource elements for a number of subframes across a single antenna port.

Data Types: `double`
Complex Number Support: Yes

`tm` — Test model configuration

scalar structure

Test model configuration, returned as a scalar structure. This argument contains information about the OFDM modulated waveform and test model configuration parameters .tm contains the following fields.

`TMN` — Test model number type

`'1.1' | '1.2' | '2' | '3.1' | '3.2' | '3.3'`

Test model number type, returned as a string.

Data Types: `char`

`BW` — Channel bandwidth type

`'1.4MHz' | '3MHz' | '5MHz' | '10MHz' | '15MHz' | '20MHz' | '9RB' | '11RB' | '27RB' | '45RB' | '64RB' | '91RB'`

Channel bandwidth type, in MHz, returned as a string. Non-standard bandwidths, `'9RB'`,`'11RB'`,`'27RB'`,`'45RB'`,`'64RB'`, and `'91RB'`, specify custom test models.

Data Types: `char`

`NDLRB` — Number of downlink resource blocks

100 | nonnegative scalar integer

Number of downlink resource blocks, returned as a nonnegative scalar integer.

Data Types: `double`

`CellRefP` — Number of cell-specific reference signal antenna ports

1

Number of cell-specific reference signal antenna ports, returned as 1. This argument is for informational purposes and is read-only.

Data Types: `double`

`NCellID` — Physical layer cell identity

scalar integer

Physical layer cell identity, returned as a scalar integer.

Data Types: `double`

`CyclicPrefix` — Cyclic prefix length

`'Normal'`

Cyclic prefix length, returned as the string, `'Normal'`. This argument is for informational purposes and is read-only.

Data Types: `char`

`CFI` — Control format indicator value

1 | 2 | 3

Control format indicator value, returned as a positive scalar integer. Valid values are 1, 2, and 3.

Data Types: `double`

`Ng` — HICH group multiplier

`'Sixth' | 'Half' | 'One' | 'Two'`

HICH group multiplier, returned as a string.

Data Types: `char`

`PHICHDuration` — PHICH duration
`'Normal' | 'Extended'`

PHICH duration, returned as a string.

Data Types: `char`

`NSubframe` — Subframe number
nonnegative scalar integer

Subframe number, returned as a nonnegative scalar integer.

Data Types: `double`

`TotSubframes` — Total number of subframes to be generated
nonnegative scalar integer

Total number of subframes to be generated, returned as a nonnegative scalar integer.

Data Types: `double`

`Windowing` — Number of windowing samples
nonnegative scalar integer

Number of windowing samples, returned as a nonnegative scalar integer. This argument is the number of time-domain samples over which windowing and overlapping of OFDM symbols is applied.

Data Types: `double`

`DuplexMode` — Duplex mode
`'FDD'` (default) | Optional | `'TDD'`

Duplex mode, returned as a string. Optional.

Data Types: `char`

`TDDConfig` — Present only for TDD duplex mode. Uplink/Downlink Configuration
0 (default) | numeric scalar | 1...6

`TDDConfig` enumerates the subframe uplink-downlink configuration to be used in this frame. Clause 4.2 of [2] specifies uplink-downlink configurations (uplink, downlink and special subframe combinations).

Data Types: `double`

`SSC` — Present only for TDD duplex mode. Special Subframe Configuration

0 (default) | numeric scalar | 1...9

`SSC` enumerates the special subframe configuration. Clause 4.2 of [2] specifies the special subframe configurations (lengths of DwPTS, GP and UpPTS).

Data Types: `double`

`PDSCH` — PDSCH transmission configuration

scalar structure

PDSCH transmission configuration, returned as a scalar structure. PDSCH contains the following fields.

`NLayers` — Number of transmission layers

1

Number of transmission layers, returned as 1. This argument is for informational purposes and is read-only.

Data Types: `double`

`TxScheme` — Transmission scheme

`'Port0'`

Transmission scheme, returned as the string `'Port0'`. This argument is for informational purposes and is read-only. The E-TMs have a single antenna port, Port 0.

Data Types: `char`

`Modulation` — Modulation formats

`{'QPSK'}` | `{'16QAM'}` | `{'64QAM'}` | Cell array of two strings

Modulation formats, returned as a cell array of one or two strings specifying the modulation formats for boosted and deboosted PRBs.

Data Types: `cell`

Data Types: `struct`

`CellRSPower` — Cell-specific reference symbol power adjustment

numeric scalar

Cell-specific reference symbol power adjustment, in dB, returned as a numeric scalar.

Data Types: `double`

`PSSPower` — PSS symbol power adjustment
numeric scalar

PSS symbol power adjustment, in dB, returned as a numeric scalar.

Data Types: `double`

`SSSPower` — SSS symbol power adjustment
numeric scalar

SSS symbol power adjustment, in dB, returned as a numeric scalar.

Data Types: `double`

`PBCHPower` — PBCH symbol power adjustment
numeric scalar

PBCH symbol power adjustment, in dB, returned as a numeric scalar.

Data Types: `double`

`PCFICHPower` — PCFICH symbol power adjustment
numeric scalar

PCFICH symbol power adjustment, in dB, returned as a numeric scalar.

Data Types: `double`

`NAllocatedPDCCHREG` — Number of allocated PDCCH REGs
nonnegative scalar integer

Number of allocated PDCCH REGs, returned as a nonnegative scalar integer. This argument is derived from TMN and BW.

Data Types: `double`

`PDCCHPower` — PDCCH symbol power adjustment
numeric scalar

PDCCH symbol power adjustment, in dB, returned as a numeric scalar.

Data Types: `double`

`PDSCHPowerBoosted` — PDSCH symbol power adjustment for boosted PRBs
numeric scalar

PDSCH symbol power adjustment for boosted PRBs, in dB, returned as a numeric scalar.

Data Types: `double`

`PDSCHPowerDeboosted` — PDSCH symbol power adjustment for de-boosted PRBs
numeric scalar

PDSCH symbol power adjustment for de-boosted PRBs, in dB, returned as a numeric scalar.

Data Types: `double`

`AllocatedPRB` — Allocated PRB list
numeric scalar

Allocated PRB list, returned as a numeric scalar array.

Data Types: `double`

`SamplingRate` — Sampling rate
numeric scalar

Sampling rate, returned as a numeric scalar value.

Data Types: `double`

`Nfft` — Number of FFT points
128 (default) | numeric scalar

Number of FFT points, returned as an unsigned integer.

Data Types: `uint32`

Data Types: `struct`

References

[1] 3GPP TS 36.141. "Base Station (BS) conformance testing." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.211. "Physical channels and modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteDLConformanceTestTool` | `lteRMCDLTool` | `lteRMCULTool` | `lteTestModel`

lteTurboDecode

Turbo decoding

Syntax

```
out = lteTurboDecode(in)
out = lteTurboDecode(in,nturbodecits)
```

Description

`out = lteTurboDecode(in)` returns the result of turbo decoding the input data in. The function can decode single data vectors or cell arrays of data vectors. In the case of cell array input, the output is a cell array containing the separately decoded input array vectors. The input data is assumed to be soft bit data that has been encoded with the parallel concatenated convolutional code (PCCC), as defined in section 5.1.3.2 of [1]. Each input data vector is assumed to be structured as three encoded parity streams concatenated in a block-wise fashion, `[S P1 P2]`, where `S` is the vector of systematic bits, `P1` is the vector of encoder 1 bits, and `P2` is the vector of encoder 2 bits. The decoder uses a default value of 5 iteration cycles. It returns the decoded bits in output vector out after performing turbo decoding using a sub-log-MAP (Max-Log-MAP) algorithm.

`out = lteTurboDecode(in,nturbodecits)` provides control over the number of turbo decoding iteration cycles via parameter nturbodecits. The nturbodecits is an optional parameter. If it is not provided, it uses the default value of 5 iteration cycles.

Examples

Turbo Decode Input Data

Perform turbo decoding for a cell array input.

```
turboEncoded = lteTurboEncode({ones(40,1),ones(6144,1)});
turboDecoded = lteTurboDecode(turboEncoded)
```

```
    [40x1 int8]    [6144x1 int8]
```

Input Arguments

`in` — Soft bit input data

numeric vector | numeric cell array of vectors

Soft bit input data, specified as a numeric vector or a cell array of vectors. The decoder expects the input bits to be encoded with the parallel concatenated convolutional code (PCCC), as defined in section 5.1.3.2 of [1].

Data Types: `int8` | `double` | `cell`

`nturbodecits` — Number of turbo decoding iteration cycles

5 (default) | Optional | positive scalar integer (1...30)

Number of turbo decoder iteration cycles, specified as a positive scalar integer between 1 and 30. Optional.

Data Types: `double`

Output Arguments

`out` — Turbo decoded bits

integer column vector | cell array of integer column vectors

Turbo decoded bits, returned as an integer column vector or a cell array of integer column vectors.

Data Types: `int8` | `cell`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteCodeBlockDesegment` | `lteConvolutionalDecode` | `lteDLSCHDecode` | `lteRateRecoverTurbo` | `lteTurboEncode` | `lteULSCHDecode`

lteTurboEncode

Turbo encoding

Syntax

```
out = lteTurboEncode(in)
```

Description

`out = lteTurboEncode(in)` returns the result of turbo encoding the input data, in. Only a finite number of acceptable data vector lengths can be coded. For more information, see table 5.1.3-3 of [1]. Filler bits are supported through negative input values.

The encoder is a parallel concatenated convolutional code (PCCC) with two 8-state constituent encoders and a contention-free interleaver. The coding rate of turbo encoder is 1/3. The three encoded parity streams are concatenated block-wise to form the encoded output, `[S P1 P2]`, where `S` is the vector of systematic bits, `P1` is the vector of encoder 1 bits, and `P2` is the vector of encoder 2 bits. To support the correct processing of filler bits, negative input bit values are specially processed. They are treated as logical 0 at the input to both encoders but their negative values are passed directly through to the associated output positions in subblocks `S` and `P1`.

Examples

Turbo Encode Input Data

Perform turbo encoding for a cell array input.

```
turboEncoded = lteTurboEncode({ones(40,1),ones(6144,1)})
```

```
[132x1 int8]    [18444x1 int8]
```

Input Arguments

`in` — Input data

numeric vector | numeric cell array of vectors

Input data, specified as a numeric vector or a cell array of vectors.

Data Types: `int8` | `double` | `cell`

Output Arguments

`out` — Turbo encoded bits

integer column vector | cell array of integer column vectors

Turbo encoded bits, returned as an integer column vector or a cell array of integer column vectors. If the input is a cell array, the output is a cell array containing the separately encoded input array vectors.

Data Types: `int8` | `cell`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`lteCodeBlockSegment` | `lteConvolutionalEncode` | `lteDLSCH` | `lteRateMatchTurbo` | `lteTurboDecode` | `lteULSCH`

lteTransmitDiversityDecode

Orthogonal space frequency block code decoding

Syntax

```
[out,csi] = lteTransmitDiversityDecode(in,hest)
```

Description

`[out,csi] = lteTransmitDiversityDecode(in,hest)` performs orthogonal space frequency block code (OSFBC) decoding of received symbols, in, and channel estimate, hest, returning the result in out.

Examples

Perform OSFBC Decoding of PDSCH Symbols

Perform orthogonal space frequency block code (OSFBC) decoding of PDSCH symbols, using ideal channel estimates.

First, generate a resource grid using multiple antennas to transmit a single PDSCH codeword.

```
enb = lteRMCDL('R.11');
enb.TotSubframes = 1;
[~,txGrid] = lteRMCDLTool(enb,[1;0;0;1]);
```

Extract the PDSCH symbols from this transmit grid.

```
[ind,indInfo] = ltePDSCHIndices(enb,enb.PDSCH,enb.PDSCH.PRBSet);
pdschSym = txGrid(ind);
```

Create an ideal, or identity, channel estimate.

```
hest = permute(repmat(eye(enb.CellRefP),[1,1,indInfo.Gd]),[3,1,2]);
```

Deprecode the PDSCH symbols, using the channel estimates.

```
[out,csi] = lteTransmitDiversityDecode(pdschSym,hest);
```

Input Arguments

`in` — Received input symbols

numeric matrix

Received input symbols, specified as a numeric matrix. It has size *M*-by-*NRxAnts*, where *M* is the number of received symbols for each of *NRxAnts* receive antennas.

Data Types: `double`
Complex Number Support: Yes

`hest` — Channel estimate

3-D numeric array

Channel estimate, specified as a 3-D numeric array. It has size *M*-by-*NRxAnts*-by-*CellRefP*, where *M* is the number of received symbols in in, *NRxAnts* is the number of receive antennas, and *CellRefP* is the number of cell-specific reference signal antenna ports.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`out` — Decoded received symbols

complex-valued numeric column vector

Decoded received symbols, returned as a complex-valued numeric column vector. It has size *M*-by-1, where *M* is the number of received symbols for each receive antenna.

Data Types: `double`
Complex Number Support: Yes

`csi` — Soft channel state information

numeric column vector

Soft channel state information (CSI), returned as an numeric column vector. It has size *M*-by-1, where *M* is the number of received symbols for each receive antenna. csi provides an estimate of the received RE gain for each received RE.

Data Types: `double`

See Also

`lteDLDeprecode`

lteUCI3Decode

PUCCH format 3 transmission UCI decoding

Syntax

```
ackbits = lteUCI3Decode(cw,n)
```

Description

`ackbits = lteUCI3Decode(cw,n)` returns a column vector of decoded ACK bits, ackbits, resulting from decoding the soft bit column vector, cw, where the output vector ackbits is expected to contain n bits. ackbits is a vector of HARQ-ACK bits (UCI), representing the HARQ-ACK information as described in section 5.2.3.1 of [1]. n must be a scalar integer between 1 and 22, where ackbits will be empty if no HARQ-ACK bits are detected.

The decoder uses a maximum likelihood (ML) approach, assuming that cw has been demodulated using `ltePUCCH3Decode`, whose input had already been equalized to best restore the originally transmitted complex values. Specifically, this function assumes that cw is properly scaled to reflect a QPSK constellation (± 0.7071 amplitude for real and imaginary parts). If multiple decoded UCI bit vectors have a likelihood equal to the maximum, ackbits will be a matrix where each column represents one of the equally likely bit vectors. If a minimum likelihood threshold is not met, ackbits will be empty.

Examples

Encoding and decoding HARQ-ACK feedback for PUCCH Format 3

This example shows how to encode and decode an ACK using PUCCH format 3 transmission UCI decoding.

Create a Tx ACK vector.

```
txAck = [1;0;0;1];
```

Encode the vector using PUCCH format 3.

```
cw = lteUCI3Encode(txAck);
```

Convert the logical bits into soft data.

```
cw = (double(cw)-0.5)*sqrt(2.0);
```

Decode the received data using the PUCCH format 3 UCI decoder. Verify that the Rx ACK vector matches the Tx ACK vector.

```
rxAck = lteUCI3Decode(cw,length(txAck))
```

```
rxAck =

     1
     0
     0
     1
```

Input Arguments

cw — Soft bits to decode

numeric column vector

Soft bits to decode, specified as a numeric column vector.

Data Types: `int8` | `double`

n — Number of bits to return

positive scalar integer (1...22)

Number of bits to return, specified as a positive scalar integer between 1 and 22.

Data Types: `double`

Output Arguments

ackbits — Decoded HARQ-ACK bits

logical column vector

Decoded HARQ-ACK bits (UCI), returned as a logical column vector.

Data Types: `logical`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePUCCH3Decode` | `lteUCI3Encode`

lteUCI3Encode

PUCCH format 3 transmission UCI encoding

Syntax

```
cw = lteUCI3Encode(ackbits)
```

Description

`cw = lteUCI3Encode(ackbits)` returns a column vector of coded UCI bits, cw, resulting from processing of control information, ackbits. The ackbits is a vector of HARQ-ACK bits (UCI), representing the HARQ-ACK information as described in section 5.2.3.1 of [1]. ackbits must be a vector containing up to 22 bits.

The UCI HARQ-ACK processing is defined in section 5.2.3.1 of [1], and consists of a (32,*O*) block code, where *O* is the number of bits in ackbits. The coded bit vector, cw, is 48 bits long.

Examples

Encode HARQ-ACK Feedback for PUCCH Format 3

Encode and decode HARQ-ACK feedback for PUCCH format 3.

```
txAck = [1;0;0;1];
cw = lteUCI3Encode(txAck);
cw = (double(cw)-0.5)*sqrt(2.0);
rxAck = lteUCI3Decode(cw,length(txAck))
```

```
     1
     0
     0
```

1

Input Arguments

ackbits — HARK-ACK control information bits

logical vector of length 1 to 22

HARK-ACK control information bits, specified as a logical vector of length 1 to 22. This vector contains up to 22 HARQ-ACK bits (UCI).

Data Types: `logical` | `double`

Output Arguments

cw — Coded UCI bits

48-by-1 integer column vector

Coded UCI bits, returned as a 48-by-1 integer column vector. This coded bit vector is 48 bits long.

Data Types: `int8`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`ltePUCCH3` | `lteUCI3Decode`

lteUCIDecode

PUCCH format 2, 2a, and 2b transmission UCI decoding

Syntax

```
ucibits = lteUCIDecode(cw,n)
```

Description

`ucibits = lteUCIDecode(cw,n)` returns a vector of decoded UCI bits, ucibits, resulting from decoding the soft bit column vector, cw, where the output vector, ucibits, is expected to contain n bits. ucibits is a column vector of CQI/PMI or RI bits (UCI), representing the CQI/PMI or RI information fields described in section 5.2.3.3 of [1]. n must be between 1 and 13. The decoder uses a maximum likelihood approach assuming that cw has been demodulated using `ltePUCCH2Decode` whose input had already been equalized to best restore the originally transmitted complex values. If multiple decoded UCI bit vectors have a likelihood equal to the maximum, UCIBITS will be a matrix where each column represents one of the equally likely bit vectors

Examples

Decode UCI Bits for PUCCH Format 2 Transmission

Decode UCI bits representing an RI value of 3.

For more information, see table 5.2.3.3.1-3 of [1].

```
uci = lteUCIDecode(ones(20,1)/sqrt(2),2)
```

```
     1
     0
```

Input Arguments

cw — Codeword of soft bits

numeric column vector

Codeword of soft bits, specified as a numeric column vector.

Data Types: `logical` | `double`

n — Number of bits

1...11

Number of bits, specified as a scalar integer from 1 to 11.

Data Types: `double`

Output Arguments

ucibits — Decoded UCI bits

logical column vector

Decoded UCI bits, returned as a logical column vector. UCI bits are CQI/PMI or RI information.

Data Types: `logical`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePUCCH2Decode` | `lteUCI3Encode` | `lteUCIEncode`

lteUCIEncode

PUCCH format 2, 2a, and 2b transmission UCI encoding

Syntax

```
cw = lteUCIEncode(ucibits)
```

Description

`cw = lteUCIEncode(ucibits)` returns a column vector of coded UCI bits, cw, resulting from processing of control information, ucibits. ucibits is a column vector of CQI/PMI or RI bits (UCI), representing the CQI/PMI or RI information fields described in section 5.2.3.3 of [1]. ucibits should be a vector containing up to 13 bits. For PUCCH formats 2a and 2b with extended cyclic prefix, this vector should also contain the appended 1 or 2 HARQ-ACK bits for joint encoding.

The UCI processing is defined in section 5.2.3 of [1], and consists of a (20,*A*) block code, where *A* is the number of bits in ucibits. The coded bit vector, cw, is 20 bits long.

Examples

Encode UCI Bits for PUCCH Format 2 Transmission

Encode UCI bits that represent an RI value of 3.

For more information, see table 5.2.3.3.1-3 of [1].

```
cw = lteUCIEncode([1;0])
```

```
     1
     1
     1
     1
     1
     1
     1
     1
     1
```

```
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
```

Input Arguments

`ucibits` — Control information bits

logical vector of length 1 to 13

Control information bits, specified as a logical vector of length 1 to 13. This vector contains the CQI/PMI or RI logical bits (UCI), representing the CQI/PMI or RI information fields. It should be up to 13 bits in length. For PUCCH formats 2a and 2b with extended cyclic prefix, this vector should also contain the appended 1 or 2 HARQ-ACK bits for joint encoding.

Data Types: `logical`

Output Arguments

`cw` — Coded UCI bits

20-by-1 integer column vector

Coded UCI bits, returned as a 20-by-1 integer column vector. The coded bit vector is 20 bits long.

Data Types: `int8`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePUCCH2` | `lteUCI3Decode` | `lteUCIDecode`

lteULChannelEstimate

PUSCH uplink channel estimation

Syntax

```
[hest noiseest] = lteULChannelEstimate(ue,chs,rxgrid)
[hest noiseest] = lteULChannelEstimate(ue,chs,cec,rxgrid)
[hest noiseest] = lteULChannelEstimate(ue,chs,cec,rxgrid,refgrid)
[hest noiseest] = lteULChannelEstimate(ue,chs,rxgrid,refgrid)
```

Description

`[hest noiseest] = lteULChannelEstimate(ue,chs,rxgrid)` returns an estimate for the channel by averaging the least squares estimates of the reference symbols across time and copying these across the allocated resource elements within the time frequency grid. It returns the estimated channel between each transmit and receive antenna and an estimate of the noise power spectral density.

`[hest noiseest] = lteULChannelEstimate(ue,chs,cec,rxgrid)` returns the estimated channel using the method and parameters defined by the user in the channel estimator configuration cec structure.

The `'TestEVM'` pilot averaging ignores other structure fields in cec, and the method follows that described in annex F of [1] for the purposes of transmitter EVM testing.

The `'UserDefined'` pilot averaging uses a rectangular kernel of size cec.FreqWindow-by-cec.TimeWindow and performs a 2-D filtering operation on the pilots. Pilots near the edge of the resource grid are averaged less because they have no neighbors outside the grid. For cec.FreqWindow = 12×*X*, any multiple of 12, and cec.TimeWindow = 1, the estimator enters a special case where an averaging window of (12×*X*)-in-frequency is used to average the pilot estimates. The averaging is always applied across (12×*X*) subcarriers, even at the upper and lower band edges; therefore, the first (6×*X*) symbols at the upper and lower band edge have the same channel estimate. This operation ensures that averaging is always done on 12, or a multiple of 12, symbols. This provides the appropriate despreading operation required for the case of uplink MIMO where the DMRS signals associated with each layer occupy the same time/frequency locations but use different orthogonal covers to allow them to be differentiated at the receiver.

Setting cec.Reference to `'Antennas'` uses the PUSCH DMRS after precoding onto the transmission antennas as the reference for channel estimation. In this case, the precoding matrix indicated in chs.PMI is used to precode the DMRS layers onto antennas, and the channel estimate, hest, is a matrix of size *M*-by-*N*-by-*NRxAnts*-by-*NTxAnts* where *NTxAnts* is given by chs.NTxAnts. Setting cec.Reference to `'Layers'` uses the PUSCH DMRS without precoding as the reference for channel estimation. The channel estimate, hest, is of size *M*-by-*N*-by-*NRxAnts*-by-*NLayers* where *NLayers* is given by chs.NLayers. Setting cec.Reference to `'None'` generates no internal reference signals, and the channel estimation can be performed on arbitrary known REs as given by the refgrid argument below. This approach can be used to provide a refgrid containing for example the SRS signals created on all *NTxAnts*, allowing for full-rank channel estimation for the purposes of PMI selection when the PUSCH is transmitted with less than full rank.

`[hest noiseest] = lteULChannelEstimate(ue,chs,cec,rxgrid,refgrid)` returns the estimated channel using the method and parameters defined by the channel estimation configuration structure and the additional information about the transmitted symbols found in refgrid. A typical use for refgrid is to provide values of the SRS transmitted at some point during the time span of rxgrid, so the channel estimation can be enhanced.

The channel estimator configuration structure cec contains the additional field `Window`.

`[hest noiseest] = lteULChannelEstimate(ue,chs,rxgrid,refgrid)` returns the estimated channel using the estimation method as described in annex F4 of [1]. The method described utilizes extra channel information obtained through information of the transmitted symbols found in refgrid. This additional information allows for an improved estimate of the channel and is required for accurate EVM measurements. rxgrid and refgrid must only contain a whole subframe worth of OFDM symbols. For normal cyclic prefix, each subframe contains 14 OFDM symbols. Therefore, *N* is 14.

Examples

Estimate Channel Characteristics for PUSCH

Estimate the channel characteristics for a received resource grid.

```
ue = lteRMCUL('A3-2');
ue.TotSubframes = 1;
cec = struct('FreqWindow',7,'TimeWindow',1,'InterpType','cubic');
txWaveform = lteRMCULTool(ue,[1;0;0;1]);
```

```
rxWaveform = txWaveform;
rxGrid = lteSCFDMADemodulate(ue,rxWaveform);
hest = lteULChannelEstimate(ue,ue.PUSCH,cec,rxGrid);
```

Input Arguments

ue — UE-specific configuration

structure

UE-specific configuration, specified as a structure. ue can contain the following fields.

NULRB — Number of uplink resource blocks

6 | 15 | 25 | 50 | 75 | 100

Number of uplink resource blocks, specified as a positive scalar integer.

Data Types: `double`

NCellID — Physical layer cell identity

nonnegative scalar integer

Physical layer cell identity, specified as a nonnegative scalar integer.

Data Types: `double`

NSubframe — Subframe number

nonnegative scalar integer

Subframe number, specified as a nonnegative scalar integer.

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length for uplink

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink, specified as a string. Optional.

Data Types: `char`

NTxAnts — Number of transmission antennas

1 (default) | Optional | 2 | 4

Number of transmission antennas, specified as a positive scalar integer. Optional. Valid values are 1, 2, and 4.

Data Types: `double`

Data Types: `struct`

`chs` — PUSCH channel settings

structure

PUSCH channel settings, specified as a structure that can contain the following fields. The parameter field PMI is only required if `ue.NTxAnts` is set to 2 or 4.

`PRBSet` — Physical resource block set

1- or 2-column numeric matrix

Physical resource block set, specified as a 1- or 2-column matrix. This parameter field contains the physical resource block indices (PRBs) corresponding to the resource allocations for this PUSCH.

Data Types: `double`

`DynCyclicShift` — Cyclic shift for DMRS

0 (default) | Optional | 0...7

Cyclic shift for DMRS, specified as a positive scalar integer between 0 and 7. Optional. (yields *n2_DMRS*)

Data Types: `double`

`NLayers` — Number of transmission layers

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers, specified as a scalar value. Optional.

Data Types: `double`

`PMI` — Precoder matrix indication

0 (default) | Optional | nonnegative scalar integer (0...23)

Precoder matrix indication, specified as a nonnegative scalar integer between 0 and 23. Only required if `ue.NTxAnts` is set to 2 or 4. This PMI value is to be used during precoding of the DRS reference symbols,

Data Types: `double`

Data Types: `struct`

`rxgrid` — Resource elements grid

3-D numeric array

Resource elements grid, specified as a 3-D numeric array of size *M*-by-*N*-by-`NRxAnts`. The second dimension of rxgrid contains a whole subframe worth of OFDM symbols. For example, for normal cyclic prefix, each subframe contains 14 OFDM symbols. Therefore, *N* is 14 when `TotalNoSubframes` is 1. If `TotalNoSubframes` is greater than one, the correct region is extracted from the returned hest array.

Data Types: `double`
Complex Number Support: Yes

`cec` — Channel estimator configuration

structure

Channel estimator configuration, specified as a structure with these fields.

`FreqWindow` — Size of window used for averaging over frequency

scalar integer

Size of window used for averaging over frequency, in resource elements, specified as a scalar integer. The window size must be either an odd number or a multiple of 12.

Data Types: `double`

`TimeWindow` — Size of window used for averaging over time

scalar integer

Size of window used for averaging over time, in resource elements, specified as a scalar integer. The window size must be an odd number.

Data Types: `double`

`InterpType` — 2-D interpolation type

`'nearest' | 'linear' | 'natural' | 'cubic' | 'v4'`

2-D interpolation type, specified as a string. For details, see `griddata`. Supported choices are shown in the following table.

String	Description
`'nearest'`	Nearest neighbor interpolation
`'linear'`	Linear interpolation

String	Description
`'natural'`	Natural neighbor interpolation
`'cubic'`	Cubic interpolation
`'v4'`	MATLAB 4 `griddata` method

Data Types: `char`

`PilotAverage` — Type of pilot averaging
`'UserDefined'` (default) | Optional | `'TestEVM'`

Type of pilot averaging, specified as a string. Optional.

Data Types: `char`

`Reference` — Point of reference for channel estimation
`'Antennas'` (default) | Optional | `'Layers'` | `'None'`

Point of reference for channel estimation, specified as a string. Optional. This parameter field specifies the signals to internally generate.

Data Types: `char`

`Window` — Position of subframe containing desired channel estimate
Optional | `'Left'` | `'Right'` | `'Centred'` | `'Centered'`

Position of the subframe containing desired channel estimate, from rxgrid and refgrid, specified as a string. This input argument is optional unless rxgrid contains more than one subframe. Specifying a value for Window results in channel estimates for only the designated subframe, but channel estimation is aided by reference symbols in the other subframes. The value of Window can be one of `'Left'`, `'Right'`, `'Centred'` or `'Centered'`. For `'Centred'` or `'Centered'`, the window size must be odd.

Data Types: `char`

Data Types: `struct`

`refgrid` — Reference array of known transmitted data symbols in their correct locations
3-D numeric array

Reference array of known transmitted data symbols in their correct locations, specified as a 3-D numeric array of size *M*-by-*N*-by-*NTxAnts*. All other locations, such as DRS Symbols and unknown data symbol locations, should be represented by a `NaN`. rxgrid and refgrid must have the same dimensions.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`hest` — Channel estimate between each transmit and receive antenna

4-D numeric array

Channel estimate between each transmit and receive antenna, returned as a 4-D numeric array of size *M*-by-*N*-by-*NRxAnts*-by-*NTxAnts*. *M* is the total number of subcarriers, *N* is the number of OFDM symbols, *NRxAnts* is the number of receive antennas, and *NTxAnts* is the number of transmit antennas. Optionally, the channel estimator can be configured to use the DMRS layers as the reference signal. In this case, the array is an *M*-by-*N*-by-*NRxAnts*-by-*NLayers* array, where *NLayers* is the number of transmission layers.

Data Types: `double`
Complex Number Support: Yes

`noiseest` — Noise estimate

numeric scalar

Noise estimate, returned as the power spectral density of the noise present on the estimated channel response coefficients, returned as a numeric scalar.

Data Types: `double`

More About

Algorithms

The channel estimation algorithm is described in the following steps.

1. Extract the demodulation reference signals, or pilot symbols, for a transmit-receive antenna pair from the allocated physical resource blocks within the received subframe.
2. Average the least-squares estimates to reduce any unwanted noise from the pilot symbols.
3. Using the cleaned pilot symbol estimates, interpolate to obtain an estimate of the channel for the entire number of subframes passed into the function.

Least-Squares Estimation

The least-squares estimates of the reference signals are obtained by dividing the received pilot symbols by their expected value. The least-squares estimates are affected by any system noise. This noise needs to be removed or reduced to achieve a reasonable estimation of the channel at pilot symbol locations.

Noise Reduction and Interpolation

To minimize the effects of noise on the pilot symbol estimates, the least-squares estimates are averaged. This simple method produces a substantial reduction in the level of noise found on the pilot symbols. The pilot symbol averaging method uses an averaging window defined by the user. The averaging window size is measured in resource elements; any pilot symbols located within the window are used to average the value of the pilot symbol found at the center of the window.

Then, the averaged pilot symbol estimates are used to perform a 2-D interpolation across allocated physical resource blocks. The location of pilot symbols within the subframe is not ideally suited to interpolation. To account for this positioning, virtual pilots are created and placed out with the area of the current subframe. This placement allows complete and accurate interpolation to be performed.

Note: The PUSCH channel estimator is only able to deal with contiguous allocation of resource blocks in time and frequency.

References

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`griddata` | `lteEqualizeMIMO` | `lteEqualizeMMSE` | `lteEqualizeZF` | `lteSCFDMADemodulate` | `lteULFrameOffset` | `lteULPerfectChannelEstimate`

lteULChannelEstimatePUCCH1

PUCCH format 1 uplink channel estimation

Syntax

```
[hest,noiseest] = lteULChannelEstimatePUCCH1(ue,chs,rxgrid)
[hest,noiseest] = lteULChannelEstimatePUCCH1(ue,chs,cec,rxgrid)
[hest,noiseest] = lteULChannelEstimatePUCCH1(ue,chs,cec,rxgrid,
refgrid)
[hest,noiseest] = lteULChannelEstimatePUCCH1(ue,chs,rxgrid,refgrid)
```

Description

`[hest,noiseest] = lteULChannelEstimatePUCCH1(ue,chs,rxgrid)` returns an estimate for the channel by averaging the least squares estimates of the reference symbols across time and copying these across the allocated resource elements within the time frequency grid. `lteULChannelEstimatePUCCH1` returns hest, the estimated channel between each transmit and receive antenna and noiseest, an estimate of the noise power spectral density.

`[hest,noiseest] = lteULChannelEstimatePUCCH1(ue,chs,cec,rxgrid)` returns the estimated channel using the method and parameters defined by the user in the channel estimator configuration structure cec.

The `'TestEVM'` pilot averaging ignores other structure fields in cec, and the method follows that described in annex F of [1] for the purposes of transmitter EVM testing.

The `'UserDefined'` pilot averaging uses a rectangular kernel of size `cec.FreqWindow`-by-`cec.TimeWindow` and performs a 2-*D* filtering operation upon the pilots. Note that pilots near the edge of the resource grid are averaged less as they have no neighbors outside of the grid. For `cec.FreqWindow` = 12×*X* (i.e. any multiple of 12) and `cec.TimeWindow` = 1 the estimator enters a special case where an averaging window of (12×*X*)-in-frequency is used to average the pilot estimates; the averaging is always applied across (12×*X*) subcarriers, even at the upper and lower band edges; therefore the first (6×*X*) symbols at the upper and lower band edge have the same channel estimate. This operation ensures that averaging is always done on 12, or a multiple of 12, symbols.

This provides the appropriate “despreading” operation required for the case multi-antenna transmission where the DRS signals associated with each antenna occupy the same time/frequency locations but use different orthogonal covers to allow them to be differentiated at the receiver.

`[hest,noiseest] = lteULChannelEstimatePUCCH1(ue,chs,cec,rxgrid, refgrid)` returns the estimated channel using the method and parameters defined by the channel estimation configuration structure and the additional information about the transmitted symbols found in refgrid.

The configuration structure cec contains the additional field, `Window`.

`[hest,noiseest] = lteULChannelEstimatePUCCH1(ue,chs,rxgrid,refgrid)` returns the estimated channel using the estimation method as described in annex F4 of [1]. The method described utilizes extra channel information obtained through information of the transmitted symbols found in refgrid. This additional information allows for an improved estimate of the channel and is required for accurate EVM measurements. rxgrid and refgrid must only contain a whole subframe worth of OFDM symbols i.e. for normal cyclic prefix each subframe contains 14 OFDM symbols. Therefore, N is 14.

Examples

Estimate Channel Characteristics for PUCCH Format 1

Estimate the channel characteristics for a received resource grid.

```
ue = struct('NULRB',6,'NCellID',0,'NSubframe',0,'Hopping','Off');
pucch1 = struct('ResourceIdx',0);
rgrid = lteULResourceGrid(ue);
rgrid(ltePUCCH1DRSIndices(ue,pucch1)) = ltePUCCH1DRS(ue,pucch1);
cec = struct('FreqWindow',12,'TimeWindow',1,'InterpType','cubic');
hest = lteULChannelEstimatePUCCH1(ue,pucch1,cec,rgrid);
```

Input Arguments

ue — UE-specific configuration settings

structure

UE-specific configuration settings, specified as a structure that can contain the following fields.

`NULRB` — Number of uplink resource blocks

6 | 15 | 25 | 50 | 75

Number of uplink resource blocks, specified as a positive scalar integer.

Data Types: `double`

`NCellID` — Physical layer cell identity

nonnegative scalar integer

Physical layer cell identity, specified as a nonnegative scalar integer.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length for uplink

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink, specified as a string. Optional.

Data Types: `char`

`Hopping` — Frequency hopping method

`'Off'` (default) | Optional | `'Group'`

Frequency hopping method, specified as a string. Optional.

Data Types: `char`

Data Types: `struct`

`chs` — PUCCH settings

structure

PUCCH channel settings, specified as a structure that can contain the following fields. The parameter DeltaOffset is deprecated and may be removed in a later release.

`ResourceIdx` — PUCCH resource indices

nonnegative integer vector (0...2047)

PUCCH resource indices, specified as nonnegative integer vector with values ranging between 0 and 2047. There is one element for each transmission antenna. These indices

determine the cyclic shift and orthogonal cover used for transmission. (*n1_pucch*). This input argument is optional.

Data Types: `double`

`DeltaShift` — Delta shift

1 (default) | Optional | 2 | 3

Delta shift, specified as a 1, 2, or 3. Optional. (*delta_shift*)

Data Types: `double`

`DeltaOffset` — Delta offset

0 (default) | Deprecated | Optional | 1 | 2

Note: Warning: The use of the `'DeltaOffset'` parameter field is deprecated. This parameter may be removed in a future release.

Delta offset, specified as 0, 1, or 2. Optional. (*delta_offset*)

Data Types: `double`

`CyclicShifts` — Number of cyclic shifts used for format 1

0 (default) | Optional | nonnegative scalar integer (0...7)

Number of cyclic shifts used for format 1, in resource blocks (RBs), with a mixture of format 1 and format 2 PUCCH, specified as a nonnegative scalar integer between 0 and 7. Optional. (*N1cs*)

Data Types: `double`

Data Types: `struct`

`rxgrid` — Received resource element grid

3-D numeric array

Received resource element grid, specified as a 3-D numeric array of size *M*-by-*N*-by-*NRxAnts*. The second dimension of rxgrid contains a whole subframe worth of OFDM symbols e.g. for normal cyclic prefix each subframe contains 14 OFDM symbols, therefore *N* is equal to 14 when `TotalNoSubframes` = 1. If `TotalNoSubframes` is greater than one, the correct region is extracted from the returned hest array.

Data Types: `double`

Complex Number Support: Yes

`cec` — Channel estimator configuration

structure

Channel estimator configuration, specified as a structure with these fields.

`FreqWindow` — Size of window used to average over frequency

scalar integer

Size of window used to average over frequency, specified as a scalar integer. The window size must be either an odd number or a multiple of 12.

Data Types: `double`

`TimeWindow` — Size of window used to average over time

scalar integer

Size of window used to average over time, specified as a scalar integer. The window size must be an odd number.

Data Types: `double`

`InterpType` — 2-D interpolation type

`'nearest'` | `'linear'` | `'natural'` | `'cubic'` | `'v4'`

2-D interpolation type, specified as a string. For details, see `griddata`. Supported choices are shown in the following table.

String	Description
`'nearest'`	Nearest neighbor interpolation
`'linear'`	Linear interpolation
`'natural'`	Natural neighbor interpolation
`'cubic'`	Cubic interpolation
`'v4'`	MATLAB 4 `griddata` method

Data Types: `char`

`PilotAverage` — Type of pilot averaging

`'UserDefined'` (default) | Optional | `'TestEVM'`

Type of pilot averaging, specified as a string. Optional.

Data Types: `char`

`Window` — Position of subframe containing the desired channel estimate

Optional | `'Left'` | `'Right'` | `'Centred'` | `'Centered'`

Position of the subframe containing the desired channel estimate, from rxgrid and refgrid, specified as a string. Optional. If more than one subframe is input, this parameter is required to indicate that only channel estimates for this subframe is returned. The window can be either `'Left'`, `'Right'`, `'Centred'`, or `'Centered'`. For `'Centred'` or `'Centered'`, the window size must be odd.

Data Types: `char`

Data Types: `struct`

`refgrid` — Reference array of known transmitted data symbols

3-D numeric array

Reference array of known transmitted data symbols in their correct locations, specified as a 3-D numeric array of size *M*-by-*N*-by-*NTxAnts*. All other locations, such as DRS symbols and unknown data symbol locations, should be represented by a `NaN`. rxgrid and refgrid must have the same dimensions.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`hest` — Channel estimate between each transmit and receive antenna

3-D numeric array

Channel estimate between each transmit and receive antenna, returned as a 3-D numeric array of size *M*-by-*N*-by-*NRxAnts*. *M* is the total number of subcarriers, *N* is the number of OFDM symbols, and *NRxAnts* is the number of receive antennas.

Data Types: `double`

`noiseest` — Noise estimate

numeric scalar

Noise estimate, returned as a numeric scalar. It is the power spectral density of the noise present on the estimated channel response coefficients.

Data Types: `double`

More About

Algorithms

The channel estimation algorithm functions as described in the following steps.

1. Extract the PUCCH format 1 demodulation reference signals (DRS), or pilot symbols, for a transmit-receive antenna pair from the allocated physical resource blocks within the received subframe.
2. Average the least-squares estimates to reduce any unwanted noise from the pilot symbols.
3. Using the cleaned pilot symbol estimates, interpolate to obtain an estimate of the channel for the allocated subframe slot passed into the function.

Least-Squares Estimation

The least-squares estimates of the reference signals are obtained by dividing the received pilot symbols by their expected value. The least-squares estimates are affected by any system noise. This noise needs to be removed or reduced to achieve a reasonable estimation of the channel at pilot symbol locations.

Noise Reduction and Interpolation

To minimize the effects of noise on the pilot symbol estimates, the least-squares estimates are averaged. This simple method produces a substantial reduction in the level of noise found on the pilot symbols. The pilot symbol averaging method uses an averaging window defined by the user. The averaging window size is measured in resource elements; any pilot symbols located within the window are used to average the value of the pilot symbol found at the center of the window.

Then, the averaged pilot symbol estimates are used to perform a 2-D interpolation across the slot of the subframe that was allocated to the PUCCH format 1 data. The location of pilot symbols within the subframe is not ideally suited to interpolation. To account for this positioning, virtual pilots are created and placed out with the area of the current subframe. This placement allows complete and accurate interpolation to be performed.

References

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`griddata` | `lteSCFDMADemodulate` | `lteULChannelEstimate` | `lteULFrameOffsetPUCCH1` | `lteULPerfectChannelEstimate`

lteULChannelEstimatePUCCH2

PUCCH format 2 uplink channel estimation

Syntax

```
[hest noiseest] = lteULChannelEstimatePUCCH2(ue,chs,rxgrid,rxack2)
[hest noiseest] = lteULChannelEstimatePUCCH2(ue,chs,cec,rxgrid,
rxack2)
[hest noiseest] = lteULChannelEstimatePUCCH2(ue,chs,cec,rxgrid,
rxack2,refgrid)
[hest noiseest] = lteULChannelEstimatePUCCH2(ue,chs,rxgrid,rxack2,
refgrid)
```

Description

`[hest noiseest] = lteULChannelEstimatePUCCH2(ue,chs,rxgrid,rxack2)` returns an estimate for the channel by averaging the least squares estimates of the reference symbols across time and copying these across the allocated resource elements within the time frequency grid. It returns hest, the estimated channel between each transmit and receive antenna and noiseest, an estimate of the noise power spectral density.

`[hest noiseest] = lteULChannelEstimatePUCCH2(ue,chs,cec,rxgrid,rxack2)` returns the estimated channel using the method and parameters defined by the user in the channel estimator configuration structure cec.

The `'TestEVM'` pilot averaging ignores other structure fields in cec, and the method follows that described in annex F of [1] for the purposes of transmitter EVM testing.

The `'UserDefined'` pilot averaging uses a rectangular kernel of size `cec.FreqWindow`-by-`cec.TimeWindow` and performs a 2-D filtering operation upon the pilots. Pilots near the edge of the resource grid are averaged less as they have no neighbors outside of the grid. For `cec.FreqWindow` = 12×*X* (i.e. any multiple of 12) and `cec.TimeWindow` = 1 the estimator enters a special case where an averaging window of (12×*X*)-in-frequency is used to average the pilot estimates; the averaging is always applied across (12×*X*)

subcarriers, even at the upper and lower band edges; therefore the first (6×*X*) symbols at the upper and lower band edge have the same channel estimate. This operation ensures that averaging is always done on 12 (or a multiple of 12) symbols. This provides the appropriate despreading operation required for the case multi-antenna transmission where the DRS signals associated with each antenna occupy the same time/frequency locations but use different orthogonal covers to allow them to be differentiated at the receiver.

`[hest noiseest] = lteULChannelEstimatePUCCH2(ue,chs,cec,rxgrid, rxack2,refgrid)` returns the estimated channel using the method and parameters defined by the channel estimation configuration structure and the additional information about the transmitted symbols found in refgrid.

The configuration structure cec contains the additional field `Window`.

`[hest noiseest] = lteULChannelEstimatePUCCH2(ue,chs,rxgrid,rxack2, refgrid)` returns the estimated channel using the estimation method, as described in annex F4 of [1]. The method described utilizes extra channel information obtained through information of the transmitted symbols found in refgrid. This additional information allows for an improved estimate of the channel and is required for accurate EVM measurements. rxgrid and refgrid must only contain a whole subframe worth of OFDM symbols i.e. for normal cyclic prefix each subframe contains 14 OFDM symbols. Therefore, *N* is 14.

Examples

Estimate Channel Characteristics for PUCCH Format 2

Estimate the channel characteristics for a received resource grid.

```
ue = struct('NULRB',6,'NCellID',0,'NSubframe',0,'Hopping','Off');
pucch2 = struct('ResourceIdx',0);
txGrid = lteULResourceGrid(ue);
txAck = [1;1];
drsIndices = ltePUCCH2DRSIndices(ue,pucch2);
txGrid(drsIndices) = ltePUCCH2DRS(ue,pucch2,txAck);
rxGrid = txGrid;
rxAck = ltePUCCH2DRSDecode(ue,pucch2,length(txAck),rxGrid(drsIndices));
cec = struct('FreqWindow',12,'TimeWindow',1,'InterpType','cubic');
```

```
hest = lteULChannelEstimatePUCCH2(ue,pucch2,cec,rxGrid,rxAck);
```

Input Arguments

`ue` — UE-specific configuration settings

structure

UE-specific configuration settings, specified as a structure that can contain the following fields.

`NULRB` — Number of uplink resource blocks

6 | 15 | 25 | 50 | 75

Number of uplink resource blocks, specified as a positive scalar integer.

Data Types: `double`

`NCellID` — Physical layer cell identity

nonnegative scalar integer

Physical layer cell identity, specified as a nonnegative scalar integer.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length for uplink

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink, specified as a string. Optional.

Data Types: `char`

`Hopping` — Frequency hopping method

`'Off'` (default) | Optional | `'Group'`

Frequency hoping method, specified as a string. Optional.

Data Types: `char`

Data Types: `struct`

`chs` — PUCCH channel settings

structure

PUCCH channel settings, specified as a structure that can contain the following fields.

`ResourceIdx` — PUCCH resource indices
nonnegative integer vector (0...2047)

PUCCH resource indices, specified as a nonnegative integer vector with values ranging between 0 and 2047. There is one element for each transmission antenna. These indices determine the cyclic shift and orthogonal cover used for transmission. (*n2_pucch*). This input argument is optional.

Data Types: `double`

`ResourceSize` — Size of resources allocated to PUCCH format 2
0 (default) | Optional | nonnegative scalar integer (0...63)

Size of resources allocated to PUCCH format 2, specified as a nonnegative scalar integer between 0 and 63. Optional. (*N2RB*)

Data Types: `double`

`CyclicShifts` — Number of cyclic shifts used for format 1
0 (default) | Optional | nonnegative scalar integer (0...7)

Number of cyclic shifts used for format 1, in resource blocks (RBs), with a mixture of format 1 and format 2 PUCCH, specified as a nonnegative scalar integer between 0 and 7. Optional. (*N1cs*)

Data Types: `double`

Data Types: `struct`

`rxgrid` — Received resource element grid
3-D numeric array

Received resource element grid, specified as a 3-D numeric array of size *M*-by-*N*-by-*NRxAnts*. The second dimension of rxgrid contains a whole subframe worth of OFDM symbols e.g. for normal cyclic prefix each subframe contains 14 OFDM symbols, therefore *N* is equal to 14 when `TotalNoSubframes` = 1. If `TotalNoSubframes` is greater than one, the correct region is extracted from the returned hest array.

Data Types: `double`
Complex Number Support: Yes

`rxack2` — Hybrid ARQ indicators
logical value

Hybrid ARQ indicators, specified as a row vector of either 1 or 2 indicators, decoded from the PUCCH Format 2 DRS. This is required as the channel estimator uses the PUCCH Format 2 DRS to estimate the channel. rxack2 can be obtained for example by using the `lteULFrameOffsetPUCCH2` function.

Data Types: `logical`

`cec` — Channel estimator configuration

structure

Channel estimator configuration, specified as a structure with these fields.

`FreqWindow` — Size of window used to average over frequency

scalar integer

Size of window used to average over frequency, in resource elements (REs), specified as a scalar integer. The window size must be either an odd number or a multiple of 12.

Data Types: `double`

`TimeWindow` — Size of window used to average over time

scalar integer

Size of window used to average over time, in resource elements (REs), specified as a scalar integer. The window size must be an odd number.

Data Types: `double`

`InterpType` — 2-D interpolation type

`'nearest'` | `'linear'` | `'natural'` | `'cubic'` | `'v4'`

2-D interpolation type, specified as a string. For details, see `griddata`. Supported choices are shown in the following table.

String	Description
`'nearest'`	Nearest neighbor interpolation
`'linear'`	Linear interpolation
`'natural'`	Natural neighbor interpolation
`'cubic'`	Cubic interpolation
`'v4'`	MATLAB 4 `griddata` method

Data Types: `char`

`PilotAverage` — Type of pilot averaging

`'UserDefined'` (default) | Optional | `'TestEVM'`

Type of pilot averaging, specified as a string. Optional.

Data Types: `char`

`Window` — Position of subframe containing desired channel estimate

Optional | `'Left'` | `'Right'` | `'Centred'` | `'Centered'`

Position of the subframe, from rxgrid and refgrid, containing desired channel estimate, specified as a string. Optional. If more than one subframe is input, this parameter is required to indicate that only channel estimates for this subframe are returned. The window can be either `'Left'`, `'Right'`, `'Centred'`, or `'Centered'`. For `'Centred'` or `'Centered'`, the window size must be odd.

Data Types: `char`

Data Types: `struct`

`refgrid` — Reference array of known transmitted data symbols in their correct locations

3-D numeric array

Reference array of known transmitted data symbols in their correct locations, specified as a 3-D numeric array of size *M*-by-*N*-by-*NTxAnts*. All other locations, such as DRS Symbols and unknown data symbol locations, should be represented by a `NaN`. rxgrid and refgrid must have the same dimensions.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`hest` — Channel estimate between each transmit and receive antenna

3-D numeric array

Channel estimate between each transmit and receive antenna, returned as a 3-D numeric array of size *M*-by-*N*-by-*NRxAnts*. *M* is the total number of subcarriers, *N* is the number of OFDM symbols, and *NRxAnts* is the number of receive antennas.

Data Types: `double`

`noiseest` — Noise estimate
numeric scalar

Noise estimate, returned as a numeric scalar. This output is the power spectral density of the noise present on the estimated channel response coefficients.

Data Types: `double`

More About

Algorithms

The channel estimation algorithm functions as described in the following steps.

1. Extract the PUCCH format 2 demodulation reference signals (DRS), or pilot symbols, for a transmit-receive antenna pair from the allocated physical resource blocks within the received subframe.
2. Average the least-squares estimates to reduce any unwanted noise from the pilot symbols.
3. Using the cleaned pilot symbol estimates, interpolate to obtain an estimate of the channel for the allocated subframe slot.

Least-Squares Estimation

The least-squares estimates of the reference signals are obtained by dividing the received pilot symbols by their expected value. The least-squares estimates are affected by any system noise. This noise needs to be removed or reduced to achieve a reasonable estimation of the channel at pilot symbol locations.

Noise Reduction and Interpolation

To minimize the effects of noise on the pilot symbol estimates, the least-squares estimates are averaged. This simple method produces a substantial reduction in the level of noise found on the pilot symbols. The pilot symbol averaging method uses an averaging window defined by the user. The averaging window size is measured in resource elements; any pilot symbols located within the window are used to average the value of the pilot symbol found at the center of the window.

Then, the averaged pilot symbol estimates are used to perform a 2-D interpolation across the slot of the subframe that was allocated to the PUCCH format 2 data. The location of pilot symbols within the subframe is not ideally suited to interpolation. To account for this positioning, virtual pilots are created and placed out with the area of the current subframe. This placement allows complete and accurate interpolation to be performed.

References

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`griddata` | `lteSCFDMADemodulate` | `lteULChannelEstimate` | `lteULFrameOffsetPUCCH2` | `lteULPerfectChannelEstimate`

lteULChannelEstimatePUCCH3

PUCCH format 3 uplink channel estimation

Syntax

```
[hest noiseest] = lteULChannelEstimatePUCCH3(ue,chs,rxgrid)
[hest noiseest] = lteULChannelEstimatePUCCH3(ue,chs,cec,rxgrid)
[hest noiseest] = lteULChannelEstimatePUCCH3(ue,chs,cec,rxgrid,
refgrid)
[hest noiseest] = lteULChannelEstimatePUCCH3(ue,chs,rxgrid,refgrid)
```

Description

`[hest noiseest] = lteULChannelEstimatePUCCH3(ue,chs,rxgrid)` returns an estimate for the channel by averaging the least squares estimates of the reference symbols across time and copying these across the allocated resource elements within the time frequency grid. It returns hest, the estimated channel between each transmit and receive antenna and noiseest, an estimate of the noise power spectral density.

`[hest noiseest] = lteULChannelEstimatePUCCH3(ue,chs,cec,rxgrid)` returns the estimated channel using the method and parameters defined by the user in the channel estimator configuration structure cec.

The `'TestEVM'` pilot averaging ignores other structure fields in cec, and the method follows that described in annex F of [1] for the purposes of transmitter EVM testing.

The `'UserDefined'` pilot averaging uses a rectangular kernel of size `cec.`FreqWindow-by-`cec.`TimeWindow and performs a 2-D filtering operation upon the pilots. Pilots near the edge of the resource grid are averaged less because they have no neighbors outside the grid. For `cec.`FreqWindow = 12×*X* (i.e. any multiple of 12) and `cec.`TimeWindow = 1 the estimator enters a special case where an averaging window of (12×*X*)-in-frequency is used to average the pilot estimates; the averaging is always applied across (12×*X*) subcarriers, even at the upper and lower band edges; therefore the first (6×*X*) symbols at the upper and lower band edge have the same channel estimate. This operation ensures that averaging is always done on 12 (or a multiple of 12) symbols. This provides the appropriate “despreading” operation required for the multi-antenna transmission case

where the DRS signals associated with each antenna occupy the same time/frequency locations, but use different orthogonal covers to allow them to be differentiated at the receiver.

`[hest noiseest] = lteULChannelEstimatePUCCH3(ue,chs,cec,rxgrid, refgrid)` returns the estimated channel using the method and parameters defined by the channel estimation configuration structure and the additional information about the transmitted symbols found in refgrid. rxgrid and refgrid must have the same dimensions.

The configuration structure, cec, contains the additional field, Window.

`[hest noiseest] = lteULChannelEstimatePUCCH3(ue,chs,rxgrid,refgrid)` returns the estimated channel using the estimation method, as described in annex F4 of [1]. The method described utilizes extra channel information obtained through information of the transmitted symbols found in refgrid. This additional information allows for an improved estimate of the channel and is required for accurate EVM measurements. rxgrid and refgrid must have the same dimensions. rxgrid and refgrid must only contain a whole subframe worth of OFDM symbols. For example, for normal cyclic prefix, each subframe contains 14 OFDM symbols. Therefore, N is 14.

Examples

Estimate Channel Characteristics for PUCCH Format 3

Estimate the channel characteristics for a received resource grid.

```
ue = struct('NULRB',6,'NCellID',0,'NSubframe',0,'Hopping','Off');
pucch3 = struct('ResourceIdx',0);
txGrid = lteULResourceGrid(ue);
txGrid(ltePUCCH3DRSIndices(ue,pucch3)) = ltePUCCH3DRS(ue,pucch3);
rxGrid = txGrid;
cec = struct('FreqWindow',12,'TimeWindow',1,'InterpType','cubic');
hest = lteULChannelEstimatePUCCH3(ue,pucch3,cec,rxGrid);
```

Input Arguments

ue — UE-specific cell-wide settings

structure

UE-specific cell-wide settings, specified as a structure with the following fields.

`NCellID` — Physical layer cell identity
nonnegative scalar integer

Physical layer cell identity, specified as a nonnegative scalar integer.

Data Types: `double`

`NSubframe` — Subframe number
0 (default) | Optional | nonnegative scalar integer

Subframe number, specified as a nonnegative scalar integer. Optional.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length for uplink
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length for uplink, specified as a string. Optional.

Data Types: `char`

`Hopping` — Frequency hopping method
`'Off'` (default) | Optional | `'Group'`

Frequency hopping method, specified as a string. Optional.

Data Types: `char`

`Shortened` — Shortened flag
0 (default) | Optional | 1

Shortened flag, specified as a 0 or 1. Optional. This parameter field indicates whether to use last symbol of the subframe. If 1, the last symbol of the subframe is not used. This behavior is required for subframes with possible SRS transmission.

Data Types: `logical` | `double`

Data Types: `struct`

`chs` — PUCCH channel settings
structure

PUCCH channel settings, specified as a structure with the following fields.

`ResourceIdx` — PUCCH resource indices
nonnegative integer vector (0...549)

PUCCH resource indices, specified as a nonnegative integer vector with values ranging between 0 and 549. There is one index for each transmission antenna. These indices determine the physical resource blocks used for transmission. (*n3_pucch*). This input argument is optional.

Data Types: `double`

Data Types: `struct`

`rxgrid` — Received resource element grid

3-D numeric array

Received resource element grid, specified as a 3-D numeric array of size *M*-by-*N*-by-*NRxAnts*. The second dimension of rxgrid contains a whole subframe worth of SC-FDMA symbols. For example, for normal cyclic prefix, each subframe contains 14 SC-FDMA symbols. Therefore, *N* is 14 when `TotalNoSubframes` is 1. If `TotalNoSubframes` is greater than one, the correct region is extracted from the returned hest array.

If refgrid is also input, rxgrid and refgrid must have the same dimensions.

Data Types: `double`
Complex Number Support: Yes

`cec` — Channel estimator configuration

structure

Channel estimator configuration, specified as a structure that can contain the following fields.

`FreqWindow` — Size of window used to average over frequency

scalar integer

Size of window used to average over frequency, in resource elements (REs), specified as a scalar integer. The window size must be either an odd number or a multiple of 12.

Data Types: `double`

`TimeWindow` — Size of window used to average over time

scalar integer

Size of window used to average over time, in resource elements, specified as a scalar integer. The window size must be an odd number.

Data Types: `double`

`InterpType` — 2-D interpolation type
`'nearest'` | `'linear'` | `'natural'` | `'cubic'` | `'v4'`

2-D interpolation type, specified as a string. For details, see `griddata`. Supported choices are shown in the following table.

String	Description
`'nearest'`	Nearest neighbor interpolation
`'linear'`	Linear interpolation
`'natural'`	Natural neighbor interpolation
`'cubic'`	Cubic interpolation
`'v4'`	MATLAB 4 `griddata` method

Data Types: `char`

`PilotAverage` — Type of pilot averaging
`'UserDefined'` (default) | Optional | `'TestEVM'`

Type of pilot averaging, specified as a string. Optional.

Data Types: `char`

`Window` — Position of subframe containing the desired channel estimate
Optional | `'Left'` | `'Right'` | `'Centred'` | `'Centered'`

Position of the subframe, from rxgrid and refgrid, containing the desired channel estimate, specified as a string. If more than one subframe is input, this parameter is required to indicate that only channel estimates for this subframe is returned. The window can be either `'Left'`, `'Right'`, `'Centred'`, or `'Centered'`. For `'Centred'` or `'Centered'`, the window size must be odd.

Data Types: `char`

Data Types: `struct`

`refgrid` — Reference array of known transmitted data symbols in their correct locations
3-D numeric array

Reference array of known transmitted data symbols in their correct locations, specified as a 3-D numeric array of size *M*-by-*N*-by-*NTxAnts*. All other locations, such as DRS

Symbols and unknown data symbol locations, should be represented by a `NaN`. rxgrid and refgrid must have the same dimensions.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`hest` — Channel estimate between each transmit and receive antenna

3-D numeric array

Channel estimate between each transmit and receive antenna, returned as a 3-D numeric array of size *M*-by-*N*-by-*NRxAnts*. *M* is the total number of subcarriers, *N* is the number of OFDM symbols, and *NRxAnts* is the number of receive antennas.

Data Types: `double`

`noiseest` — Noise estimate

numeric scalar

Noise estimate, returned as a numeric scalar. This output is the power spectral density of the noise present on the estimated channel response coefficients, returned as a numeric array.

Data Types: `double`

More About

Algorithms

The channel estimation algorithm functions as described in the following steps.

1. Extract the PUCCH format 3 demodulation reference signals (DRS), or pilot symbols, for a transmit-receive antenna pair from the allocated physical resource blocks within the received subframe.
2. Average the least-squares estimates to reduce any unwanted noise from the pilot symbols.
3. Using the cleaned pilot symbol estimates, interpolate to obtain an estimate of the channel for the allocated subframe slot.

Least-Squares Estimation

The least-squares estimates of the reference signals are obtained by dividing the received pilot symbols by their expected value. The least-squares estimates are affected by any system noise. This noise needs to be removed or reduced to achieve a reasonable estimation of the channel at pilot symbol locations.

Noise Reduction and Interpolation

To minimize the effects of noise on the pilot symbol estimates, the least-squares estimates are averaged. This simple method produces a substantial reduction in the level of noise found on the pilot symbols. The pilot symbol averaging method uses an averaging window defined by the user. The averaging window size is measured in resource elements; any pilot symbols located within the window are used to average the value of the pilot symbol found at the center of the window.

Then, the averaged pilot symbol estimates are used to perform a 2-D interpolation across the slot of the subframe that was allocated to the PUCCH format 3 data. The location of pilot symbols within the subframe is not ideally suited to interpolation. To account for this positioning, virtual pilots are created and placed out with the area of the current subframe. This placement allows complete and accurate interpolation to be performed.

References

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`griddata` | `lteSCFDMADemodulate` | `lteULChannelEstimate` | `lteULFrameOffsetPUCCH3` | `lteULPerfectChannelEstimate`

lteULDeprecode

SC-FDMA deprecoding

Syntax

```
out = lteULDeprecode(in,nrb)
```

Description

`out = lteULDeprecode(in,nrb)` performs SC-FDMA deprecoding of the complex modulation symbols in for PUSCH configuration with a bandwidth of nrb resource blocks. in is a *Msymb*-by-*NLayers* matrix of values and out has the same dimensions.

Examples

Deprecode PUSCH Symbols

Perform the deprecoding of precoded PUSCH symbols, `precodedSym`, providing the number of uplink resource blocks (RBs), `ue.NULRB`.

```
ue = lteRMCUL('A3-2');
ueDim = lteULResourceGridSize(ue);
mSymb = ue.NULRB*ueDim(1)*ueDim(2);
realIn = randi([0,1],mSymb,ue.PUSCH.NLayers);
imagIn = randi([0,1],mSymb,ue.PUSCH.NLayers);
precodedSym = lteULPrecode(complex(realIn,imagIn),ue.NULRB);
dePrecodedSym = lteULDeprecode(precodedSym,ue.NULRB);
```

Input Arguments

`in` — Complex modulation symbols

numeric matrix

Complex modulation symbols, specified as a numeric matrix of size *Msymb*-by-*NLayers*.

Data Types: `double` | `single`
Complex Number Support: Yes

`nrb` — Number of resource blocks
scalar integer

Number of resource blocks, specified as a scalar integer.

Data Types: `double`

Output Arguments

`out` — Deprecoded PUSCH output symbols
numeric matrix

Deprecoded PUSCH output symbols, returned as a numeric matrix of size *Msymb*-by-*NLayers*.

Data Types: `double`
Complex Number Support: Yes

See Also
`lteLayerDemap` | `ltePUSCHDecode` | `ltePUSCHDeprecode` | `lteULPrecode`

lteULDescramble

PUSCH descrambling

Syntax

```
out = lteULDescramble(ue,chs,in)
out = lteULDescramble(ue,in)
out = lteULDescramble(in,nsubframe,cellid,rnti)
```

Description

`out = lteULDescramble(ue,chs,in)` performs PUSCH descrambling of the soft bit vector, in, or cell array in case of two codewords, according to UE-specific settings in the ue structure and UL-SCH related parameters in the chs structure. It performs PUSCH descrambling to undo the processing described in section 5.3.1 of [1] and returns a soft bit vector or cell array of vectors, out. This syntax supports the descrambling of control information bits if they are present in the soft bits in in conjunction with information bits. The descrambling of the control information bits is done by establishing the correct locations of placeholder bits with the help of UL-SCH-related parameters present in chs. The descrambler skips the *'x'* placeholder bits to undo the processing defined in section 5.3.1 of [1].

Multiple codewords can be parameterized by two different forms of the chs structure. Each codeword can be defined by separate elements of a 1-by-2 structure array, or the codeword parameters can be combined together in the fields of a single scalar, 1-by-1, structure. In the latter case, any scalar field values apply to both codewords and a scalar `NLayers` is the total number. For further details, see "UL-SCH Parameterization".

`out = lteULDescramble(ue,in)` performs PUSCH descrambling of the soft bit input, in, but takes only the UE-specific settings in the ue structure. The in input should contain only the scrambled data bits resulting in descrambling of transport data only. The ue structure must include the NCellID, NSubframe, and RNTI fields.

`out = lteULDescramble(in,nsubframe,cellid,rnti)` performs PUSCH descrambling of soft bits, in, for subframe number, nsubframe, cell identity, cellid, and specified radio network temporary identifier (RNTI), rnti. This syntax performs only

block descrambling and expects the input, in, to contain only the scrambled data bits. If the in vector contains placeholder bits, they are not descrambled correctly because the placeholder bits are not skipped during the descrambling process. Thus, this function syntax descrambles only the transport data bits.

Examples

Scramble and Descramble PUSCH Vector

Perform scrambling and descrambling of a PUSCH soft bit vector.

Scramble a vector of ones, `x`. Then, descramble it and display the results.

```
x = ones(10,1);
ue = struct('NCellID',100,'NSubframe',0,'RNTI',61);
scram = double(lteULScramble(ue,x))-0.5;
descram = lteULDescramble(ue,scram)
```

```
    0.5000
    0.5000
    0.5000
    0.5000
    0.5000
    0.5000
    0.5000
    0.5000
    0.5000
    0.5000
```

Input Arguments

`ue` — UE-specific settings

scalar structure

UE-specific settings, specified as a scalar structure that can contain the following fields.

`NCellID` — Physical layer cell identity

scalar integer

Physical layer cell identity, specified as a scalar integer.

Data Types: `double`

`NSubframe` — Subframe number
scalar integer

Subframe number, specified as a scalar integer.

Data Types: `double`

`RNTI` — Radio network temporary identifier
numeric scalar

Radio network temporary identifier, 16-bit, specified as a numeric scalar.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`Shortened` — Shorten subframe flag
0 (default) | Optional | 1

Shorten subframe flag, specified as 0 or 1. Optional. If 1, the last symbol of the subframe is not used and rate matching is adjusted accordingly. This setting is required for subframes with possible SRS transmission.

Data Types: `logical` | `double`

Data Types: `struct`

`chs` — UL-SCH channel-specific settings
structure

UL-SCH channel-specific settings, specified as a structure that can contain the following fields.

`Modulation` — Modulation scheme associated with each transport block
`'QPSK'` | `'16QAM'` | `'64QAM'`

Modulation scheme associated with each transport block, specified as a string.

Data Types: `char`

`NLayers` — Number of transmission layers

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers, total or per codeword, specified as 1, 2, 3, or 4. Optional.

Data Types: `double`

`ORI` — Number of uncoded RI bits

0 (default) | Optional | nonnegative scalar integer

Number of uncoded RI bits, specified as a nonnegative scalar integer. Optional.

Data Types: `double`

`OACK` — Number of uncoded HARQ-ACK bits

0 (default) | Optional | nonnegative scalar integer

Number of uncoded HARQ-ACK bits, specified as a nonnegative scalar integer. Optional.

Data Types: `double`

`QdRI` — Number of coded RI symbols in UL-SCH

0 (default) | Optional | nonnegative scalar integer

Number of coded RI symbols in UL-SCH, specified as a nonnegative scalar integer. Optional. (*Q'_RI*)

Data Types: `double`

`QdACK` — Number of coded HARQ-ACK symbols in UL-SCH

0 (default) | Optional | nonnegative scalar integer

Number of coded HARQ-ACK symbols in UL-SCH, specified as a nonnegative scalar integer. Optional. (*Q'_ACK*)

Data Types: `double`

Data Types: `struct`

`in` — Soft bit input data

numeric column vector | cell array of numeric column vectors

Soft bit input data, specified as a numeric column vector or cell array of numeric column vectors. This argument contains one or two vectors corresponding to the number of codewords to be scrambled.

Data Types: `double | cell`

`nsubframe` — Subframe number
scalar integer

Subframe number, specified as a scalar integer.

Data Types: `double`

`cellid` — Physical layer cell identity
scalar integer

Physical layer cell identity, specified as a scalar integer.

Data Types: `double`

`rnti` — Radio network temporary identifier
numeric scalar

Radio network temporary identifier, 16-bit, specified as a numeric scalar.

Data Types: `double`

Output Arguments

`out` — PUSCH descrambled output bits
numeric column vector | cell array of numeric column vectors

PUSCH descrambled output bits, returned as a numeric column vector or cell array of numeric column vectors.

Data Types: `double`

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also
`ltePUSCHDecode | lteSymbolDemodulate | lteULScramble`

lteULFrameOffset

PUSCH DRS uplink frame timing estimate

Syntax

```
offset = lteULFrameOffset(ue,chs,waveform)
[offset,corr] = lteULFrameOffset(ue,chs,waveform)
```

Description

`offset = lteULFrameOffset(ue,chs,waveform)` performs synchronization using PUSCH DRS signals for the time-domain waveform, waveform, given UE-specific settings, ue, and PUSCH configuration, chs.

waveform must be a T-by-P matrix, where T is the number of time-domain samples and P is the number of receive antennas. This matrix can be generated by SC-FDMA modulation of a resource matrix using the `lteSCFDMAModulate` function, or by using one of the channel model functions, `lteFadingChannel`, `lteHSTChannel`, or `lteMovingChannel`.

The returned value offset indicates the number of samples from the start of the waveform, waveform, to the position in that waveform where the first subframe begins. offset is computed by extracting the timing of the peak of the correlation between waveform and internally generated reference waveforms containing DRS signals. The correlation is performed separately for each antenna and the antenna with the strongest correlation is used to finally compute offset.

offset provides subframe timing; frame timing can be achieved by using offset in conjunction with the subframe number `NSubframe` in UE-specific settings structure ue. This is consistent with real-world operation, since the base station knows when, or in which subframe, to expect uplink transmissions.

`[offset,corr] = lteULFrameOffset(ue,chs,waveform)` also returns a complex matrix corr of the same dimensions as waveform which is the signal used to extract the timing offset. That is, offset is the position of `max(abs(corr))`.

Examples

Synchronize and Demodulate Delayed Transmission

Perform synchronization and demodulation of a transmission that has been delayed by 5 samples.

```
ue = lteRMCUL('A3-2');
waveform = lteRMCULTool(ue,[1;0;0;1]);
tx = [0;0;0;0;0;waveform];
offset = lteULFrameOffset(ue,ue.PUSCH,tx)
rxGrid = lteSCFDMADemodulate(ue,tx(1+offset:end));
```

```
     5
```

Input Arguments

`ue` — UE-specific settings
scalar structure

UE-specific settings, specified as a scalar structure with the following fields.

`NULRB` — Number of uplink resource blocks
scalar integer

Number of uplink (UL) resource blocks (RBs), specified as a scalar integer.

Data Types: `double`

`NCellID` — Physical layer cell identity
scalar integer

Physical layer cell identity, specified as a scalar integer.

Data Types: `double`

`NSubframe` — Subframe number
scalar integer

Subframe number, specified as a scalar integer.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`Hopping` — Frequency hopping method

`'Off'` (default) | Optional | `'Group'` | `'Sequence'`

Frequency hopping method, specified as a string. Optional.

Data Types: `char`

`SeqGroup` — PUSCH sequence group number

0 (default) | Optional | 0...29

PUSCH sequence group number, specified as a nonnegative scalar integer between 0 and 29. Optional. (*delta_SS*)

Data Types: `double`

`CyclicShift` — Number of cyclic shifts used for PUSCH DRS

0 (default) | Optional | 0...7

Number of cyclic shifts used for PUSCH DRS, specified as a nonnegative scalar integer between 0 and 7. Optional. (yields *n1_DMRS*)

Data Types: `double`

Data Types: `struct`

`chs` — PUSCH configuration

scalar structure

PUSCH configuration, specified as a scalar structure with the following fields.

`PRBSet` — Set of physical resource block indices

1- or 2-column integer matrix

Set of physical resource block indices, specified as a 1- or 2-column integer matrix. PRBSet contains the 0-based physical resource block indices (PRBs) corresponding to the resource allocations for this PUSCH.

Data Types: `double`

`DynCyclicShift` — Cyclic Shift for DMRS

0 (default) | Optional | nonnegative scalar integer (0...7)

Cyclic Shift for DMRS, specified as a nonnegative scalar integer between 0 and 7. Optional. (yields *n2_DMRS*)

Data Types: `double`

Data Types: `struct`

`waveform` — Time-domain waveform

numeric matrix

Time-domain waveform, specified as a numeric matrix. waveform must be a T-by-P matrix where T is the number of time-domain samples, and P is the number of receive antennas.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`offset` — Offset number of samples

scalar integer

Offset number of samples, returned as a scalar integer. This output is the number of samples from the start of the waveform to the position in that waveform where the first subframe begins. `offset` is computed by extracting the timing of the peak of the correlation between waveform and internally generated reference waveforms containing DRS signals. The correlation is performed separately for each antenna and the antenna with the strongest correlation is used to finally compute offset.

Data Types: `double`

`corr` — Signal used to extract the timing offset

complex-valued numeric matrix

Signal used to extract the timing offset, returned as a complex-valued numeric matrix. corr has the same dimensions as waveform.

Data Types: `double`
Complex Number Support: Yes

See Also

`lteFadingChannel` | `lteFrequencyCorrect` | `lteFrequencyOffset` | `lteHSTChannel` | `lteMovingChannel` | `lteSCFDMADemodulate`

lteULFrameOffsetPUCCH1

PUCCH format 1 DRS uplink frame timing estimate

Syntax

```
offset = lteULFrameOffsetPUCCH1(ue,chs,waveform)
[offset,corr] = lteULFrameOffsetPUCCH1(ue,chs,waveform)
```

Description

`offset = lteULFrameOffsetPUCCH1(ue,chs,waveform)` performs synchronization using PUCCH format 1 demodulation reference signals (DRS) for the time-domain waveform, waveform, given UE-specific settings, ue, and PUCCH format 1 configuration, chs.

waveform must be a *T*-by-*P* matrix, where *T* is the number of time-domain samples and *P* is the number of receive antennas; such a matrix can be generated by SC-FDMA modulation of a resource matrix using `lteSCFDMAModulate` function, or by using one of the channel model functions (`lteFadingChannel`, `lteHSTChannel`, or `lteMovingChannel`).

The returned value offset indicates the number of samples from the start of the waveform waveform to the position in that waveform where the first subframe begins. offset is computed by extracting the timing of the peak of the correlation between waveform and internally generated reference waveforms containing DRS signals. The correlation is performed separately for each transmit/receive antenna pair and the pair with the strongest correlation is used to finally compute offset.

offset provides subframe timing; frame timing can be achieved by using offset in conjunction with the subframe number, NSubframe, in UE-specific settings structure, ue. This behavior is consistent with real-world operation because the base station knows when, or in which subframe, to expect uplink transmissions.

`[offset,corr] = lteULFrameOffsetPUCCH1(ue,chs,waveform)` also returns a complex matrix corr of the same dimensions as waveform which is the signal used to extract the timing offset. That is, offset is the position of `max(abs(corr))`.

Examples

Synchronize and Demodulate Delayed Transmission

Perform synchronization and demodulation of a transmission that has been delayed by 4 samples.

```
ue = struct('NULRB',6,'NCellID',0,'NSubframe',0,'Hopping','Off');
pucch1 = struct('ResourceIdx',0);
rgrid = lteULResourceGrid(ue);
rgrid(ltePUCCH1DRSIndices(ue,pucch1)) = ltePUCCH1DRS(ue,pucch1);
waveform = lteSCFDMAModulate(ue,rgrid);
tx = [0;0;0;0;waveform];
offset = lteULFrameOffsetPUCCH1(ue,pucch1,tx)
rxGrid = lteSCFDMADemodulate(ue,tx(1+offset:end));
```

```
     4
```

Input Arguments

ue — UE-specific settings
scalar structure

UE-specific settings, specified as a scalar structure with the following fields.

NULRB — Number of uplink (UL) resource blocks (RBs)
scalar integer

Number of uplink (UL) resource blocks (RBs), specified as a scalar integer.

NTxAnts — Number of transmit antennas
Optional | scalar positive integer

Number of transmit antennas, specified as a positive integer. Optional.

NCellID — Physical layer cell identity
scalar integer

Physical layer cell identity, specified as a scalar integer.

NSubframe — Subframe number
scalar integer

Subframe number, specified as a scalar integer.

`CyclicPrefixUL` — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

`Hopping` — Frequency hopping method
`'Off'` (default) | Optional | `'Group'`

Frequency hopping method, specified as a string. Optional.

Data Types: `struct`

`chs` — PUCCH Format 1 configuration
scalar structure

PUCCH Format 1 configuration, specified as a scalar structure with the following fields.

`ResourceIdx` — PUCCH Resource Indices (*n1_pucch*)
0...2047

PUCCH Resource Indices, specified as a vector, one for each transmission antenna, which determine the cyclic shift and orthogonal cover used for transmission. (*n1_pucch*). This input argument is optional.

`DeltaShift` — Delta shift
1 (default) | Optional | 2 | 3

Delta shift, specified as a 1, 2, or 3. Optional. (*delta_shift*)

`DeltaOffset` — Delta offset
0 (default) | Deprecated | Optional | 1 | 2

Note: Warning: The use of the `'DeltaOffset'` parameter field is deprecated. This parameter may be removed in a future release.

Delta offset, specified as 0, 1, or 2. Optional. (*delta_offset*)

`CyclicShifts` — Number of cyclic shifts used for format 1
0 (default) | Optional | 0...7

Number of cyclic shifts used for format 1, in resource blocks (RBs), with a mixture of format 1 and format 2 PUCCH, specified as a nonnegative scalar integer between 0 and 7. Optional. (*N1cs*)

Data Types: `struct`

`waveform` — Time-domain waveform

numeric matrix

Time-domain waveform, specified as a numeric matrix. waveform must be a *T*-by-*P* matrix where *T* is the number of time-domain samples and *P* is the number of receive antennas.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`offset` — Number of samples from the start of the waveform to the position in that waveform where the first subframe begins

scalar integer

Number of samples from the start of the waveform to the position in that waveform where the first subframe begins, returned as a scalar integer. `offset` is computed by extracting the timing of the peak of the correlation between waveform and internally generated reference waveforms containing DRS signals. The correlation is performed separately for each antenna and the antenna with the strongest correlation is used to finally compute offset.

`corr` — Signal used to extract the timing offset

numeric matrix

Signal used to extract the timing offset, returned as a complex numeric matrix. corr has the same dimensions as waveform.

See Also

`lteFadingChannel` | `lteHSTChannel` | `lteMovingChannel` | `lteSCFDMADemodulate` | `lteULFrameOffset` | `lteULFrameOffsetPUCCH2` | `lteULFrameOffsetPUCCH3`

lteULFrameOffsetPUCCH2

PUCCH format 2 DRS uplink frame timing estimate

Syntax

```
offset = lteULFrameOffsetPUCCH2(ue,chs,waveform,oack)
[offset ack] = lteULFrameOffsetPUCCH2(ue,chs,waveform,oack)
[offset ack corr] = lteULFrameOffsetPUCCH2(ue,chs,waveform,oack)
```

Description

`offset = lteULFrameOffsetPUCCH2(ue,chs,waveform,oack)` performs synchronization using PUCCH format 2 demodulation reference signals (DRS) for the time-domain waveform, waveform, given UE-specific settings, ue, PUCCH format 2 configuration chs, and the number of Hybrid ARQ indicators oack, which may be 1 or 2.

waveform must be a T-by-P matrix where T is the number of time-domain samples and P is the number of receive antennas. This matrix can be generated by SC-FDMA modulation of a resource matrix using the `lteSCFDMAModulate` function, or by using one of the channel model functions, `lteFadingChannel`, `lteHSTChannel`, or `lteMovingChannel`.

The returned value offset indicates the number of samples from the start of the waveform waveform to the position in that waveform where the first subframe begins. offset is computed by extracting the timing of the peak of the correlation between waveform and internally generated reference waveforms containing DRS signals. The correlation is performed separately for each transmit/receive antenna pair and the pair with the strongest correlation is used to finally compute offset. This process is repeated for each possible combination of Hybrid ARQ indicators, for either 1 or 2 Hybrid ARQ indicators as specified by the parameter oack; this amounts to a maximum likelihood (ML) decoding of the Hybrid ARQ indicators which are conveyed on the PUCCH Format 2 DRS.

offset provides subframe timing; frame timing can be achieved by using offset in conjunction with the subframe number, NSubframe, in UE-specific settings structure ue. This behavior is consistent with real-world operation because the base station knows when, in which subframe, to expect uplink transmissions.

`[offset ack] = lteULFrameOffsetPUCCH2(ue,chs,waveform,oack)` also returns a vector ack of decoded PUCCH Format 2 Hybrid ARQ indicators.

`[offset ack corr] = lteULFrameOffsetPUCCH2(ue,chs,waveform,oack)` also returns a complex matrix corr of the same dimensions as waveform which is the signal used to extract the timing offset. offset is the position of `max(abs(corr))`.

Examples

Synchronize and Demodulate Delayed Transmission

This example performs synchronization and demodulation of a transmission that has been delayed by 5 samples.

Initialize ue specific parameter structure, PUCCH2 structure, UL resource grid and txAck parameter.

```
ue.NULRB = 6;
ue.NCellID = 0;
ue.NSubframe = 0;
ue.Hopping = 'Off';
ue.CyclicPrefixUL = 'Normal';
ue.NTxAnts = 1;
pucch2.ResourceIdx = 0;
pucch2.ResourceSize = 0;
pucch2.CyclicShifts = 0;
rgrid = lteULResourceGrid(ue);
txAck = [1;1];
rgrid(ltePUCCH2DRSIndices(ue,pucch2)) = ...
    ltePUCCH2DRS(ue,pucch2,txAck);
```

Generate modulated waveform

```
waveform = lteSCFDMAModulate(ue,rgrid);
tx = [0;0;0;0;0;waveform];
```

Estimate UL frame offset timing

```
offset = lteULFrameOffsetPUCCH2(ue,pucch2,tx,length(txAck))
```

```
offset =
```

```
    5
```

Perform demodulation

```
rxGrid = lteSCFDMADemodulate(ue,tx(1+offset:end));
```

Input Arguments

ue — UE-specific settings
scalar structure

UE-specific settings, specified as a scalar structure with the following fields.

NULRB — Number of uplink (UL) resource blocks (RBs)
scalar integer

Number of uplink (UL) resource blocks (RBs), specified as a scalar integer.

Data Types: `double`

NCellID — Physical layer cell identity
scalar integer

Physical layer cell identity, specified as a scalar integer.

Data Types: `double`

NSubframe — Subframe number
scalar integer

Subframe number, specified as a scalar integer.

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length
`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

Hopping — Frequency hopping method
`'Off'` (default) | Optional | `'Group'`

Frequency hopping method, specified as a string. Optional.

Data Types: `char`

Data Types: `struct`

`chs` — PUCCH Format 2 configuration

scalar structure

PUCCH Format 2 configuration, specified as a scalar structure with the following fields.

`ResourceIdx` — PUCCH Resource Indices (*n2_pucch*)

0...1185

PUCCH Resource Indices, specified as a vector, one for each transmission antenna, which determine the cyclic shift and orthogonal cover used for transmission. (*n2_pucch*). This input argument is optional.

Data Types: `double`

`ResourceSize` — Size of resources allocated to PUCCH format 2

0 (default) | Optional | 0...63

Size of resources allocated to PUCCH format 2, specified as a nonnegative scalar integer between 0 and 63. Optional. (*N2RB*)

Data Types: `double`

`CyclicShifts` — Number of cyclic shifts used for format 1

0 (default) | Optional | 0...7

Number of cyclic shifts used for format 1, in resource blocks (RBs), with a mixture of format 1 and format 2 PUCCH. Optional. (*N1cs*)

Data Types: `double`

Data Types: `struct`

`waveform` — Time-domain waveform

numeric matrix

Time-domain waveform, specified as a numeric matrix. waveform must be a *T*-by-*P* matrix where *T* is the number of time-domain samples and *P* is the number of receive antennas.

Data Types: `double`

Complex Number Support: Yes

`oack` — Number of uncoded Hybrid ARQ bits

1 | 2

Number of uncoded Hybrid ARQ bits.

Data Types: `double`

Output Arguments

`offset` — Number of samples from the start of the waveform to the position in that waveform where the first subframe begins

scalar integer

Number of samples from the start of the waveform to the position in that waveform where the first subframe begins, returned as a scalar integer. `offset` is computed by extracting the timing of the peak of the correlation between waveform and internally generated reference waveforms containing DRS signals. The correlation is performed separately for each antenna and the antenna with the strongest correlation is used to finally compute offset.

`ack` — Decoded PUCCH Format 2 Hybrid ARQ bits

numeric vector

Decoded PUCCH Format 2 Hybrid ARQ bits, returned as a vector.

Data Types: `double`

`corr` — Signal used to extract the timing offset

numeric matrix

Signal used to extract the timing offset, returned as a complex numeric matrix. corr has the same dimensions as waveform.

See Also

`lteFadingChannel` | `lteHSTChannel` | `lteMovingChannel` | `lteSCFDMADemodulate` | `lteULFrameOffset` | `lteULFrameOffsetPUCCH1` | `lteULFrameOffsetPUCCH3`

lteULFrameOffsetPUCCH3

PUCCH format 3 DRS uplink frame timing estimate

Syntax

```
offset = lteULFrameOffsetPUCCH3(ue,chs,waveform)
[offset corr] = lteULFrameOffsetPUCCH3(ue,chs,waveform)
```

Description

`offset = lteULFrameOffsetPUCCH3(ue,chs,waveform)` performs synchronization using PUCCH format 3 demodulation reference signals (DRS) for the time-domain waveform, waveform, given UE-specific settings, ue, and PUCCH format 3 configuration, chs.

waveform must be a T-by-P matrix, where T is the number of time-domain samples and P is the number of receive antennas; such a matrix can be generated by SC-FDMA modulation of a resource matrix using the `lteSCFDMAModulate` function, or by using one of the channel model functions, `lteFadingChannel`, `lteHSTChannel` or `lteMovingChannel`.

The returned value, offset, indicates the number of samples from the start of the waveform, waveform, to the position in that waveform where the first subframe begins. offset is computed by extracting the timing of the peak of the correlation between waveform and internally generated reference waveforms containing DRS signals. The correlation is performed separately for each antenna and the antenna with the strongest correlation is used to finally compute offset.

offset provides subframe timing; frame timing can be achieved by using offset in conjunction with the subframe number, NSubframe, in UE-specific settings structure, ue. This behavior is consistent with real-world operation because the base station knows when, or in which subframe, to expect uplink transmissions.

`[offset corr] = lteULFrameOffsetPUCCH3(ue,chs,waveform)` also returns a complex-valued matrix corr of the same dimensions as waveform which is the signal used to extract the timing offset. offset is the position of `max(abs(corr))`.

Examples

Synchronize and Demodulate Delayed Transmission

Synchronize and demodulate a transmission that has been delayed by 5 samples.

```
ue = struct('NULRB',6,'NCellID',0,'NSubframe',0,'Hopping','Off');
pucch3 = struct('ResourceIdx',0);
rgrid = lteULResourceGrid(ue);
rgrid(ltePUCCH3DRSIndices(ue,pucch3)) = ltePUCCH3DRS(ue,pucch3);
waveform = lteSCFDMAModulate(ue,rgrid);
tx = [0;0;0;0;0;0;0;waveform];
offset = lteULFrameOffsetPUCCH3(ue,pucch3,tx)
rxGrid = lteSCFDMADemodulate(ue,tx(1+offset:end));
```

```
     7
```

Input Arguments

`ue` — UE-specific settings
structure

UE-specific settings, specified as a structure with the following fields.

`NULRB` — Number of uplink resource blocks
scalar integer

Number of uplink (UL) resource blocks (RBs), specified as a scalar integer.

Data Types: `double`

`NCellID` — Physical layer cell identity
scalar integer

Physical layer cell identity, specified as a scalar integer.

Data Types: `double`

`NSubframe` — Subframe number
scalar integer

Subframe number, specified as a scalar integer.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`Hopping` — Frequency hopping method

`'Off'` (default) | Optional | `'Group'`

Frequency hopping method, specified as a string. Optional.

Data Types: `char`

`Shortened` — Shorten subframe flag

0 (default) | Optional | 1

Shorten subframe flag, specified as 0 or 1. Optional. If 1, the last symbol of the subframe is not used and rate matching is adjusted accordingly. This setting is required for subframes with possible SRS transmission.

Data Types: `logical` | `double`

Data Types: `struct`

`chs` — PUCCH Format 3 configuration

scalar structure

PUCCH Format 3 configuration, specified as a scalar structure with the following fields.

`ResourceIdx` — PUCCH resource indices

integer vector (0...549)

PUCCH resource indices, specified as an integer vector with values between 0 and 549. There is one element for each transmission antenna, which determine the physical resource blocks used for transmission. (*n3_pucch*). This input argument is optional.

Data Types: `double`

Data Types: `struct`

`waveform` — Time-domain waveform

numeric matrix

Time-domain waveform, specified as a numeric matrix. waveform must be a T-by-P matrix where T is the number of time-domain samples and P is the number of receive antennas.

Data Types: `double`
Complex Number Support: Yes

Output Arguments

`offset` — Number of samples from the start of the waveform to the position in that waveform where the first subframe begins

scalar integer

Number of samples from the start of the waveform to the position in that waveform where the first subframe begins, returned as a scalar integer. `offset` is computed by extracting the timing of the peak of the correlation between waveform and internally generated reference waveforms containing DRS signals. The correlation is performed separately for each antenna and the antenna with the strongest correlation is used to finally compute offset.

Data Types: `double`

`corr` — Signal used to extract the timing offset

numeric matrix

Signal used to extract the timing offset, returned as a numeric matrix. corr has the same dimensions as waveform.

Data Types: `double`
Complex Number Support: Yes

See Also

`lteFadingChannel` | `lteHSTChannel` | `lteMovingChannel` | `lteSCFDMADemodulate` | `lteULFrameOffset` | `lteULFrameOffsetPUCCH1` | `lteULFrameOffsetPUCCH2`

lteULPMIInfo

PUSCH precoder matrix indication reporting information

Syntax

```
info=lteULPMIInfo(ue,chs)
```

Description

`info=lteULPMIInfo(ue,chs)` returns a structure info containing information related to precoder matrix indication (PMI) reporting.

You can use info.NSubbands to determine the correct size of the vector PMI required for closed-loop spatial multiplexing operation. PMI is a column vector with info.NSubbands rows. Currently, only wideband PMI reporting is defined by the standard. Thus, the number of subbands, info.NSubbands, is always 1. This field and info.k are provided for consistency with the downlink version of this function, `ltePMIInfo`.

Examples

Get PUSCH PMI Reporting Information

Display the PMI reporting information for RMC A3-2.

```
ue = lteRMCUL('A3-2');
pmiInfo = lteULPMIInfo(ue,ue.PUSCH)
```

```
            k: 6
    NSubbands: 1
       MaxPMI: 0
```

Input Arguments

ue — UE-specific configuration

structure

UE-specific configuration, specified as a structure. ue can contain the following fields.

`NULRB` — Number of uplink resource blocks

6 | 15 | 25 | 50 | 75 | 100

Number of uplink resource blocks, specified as a positive scalar integer.

Data Types: `double`

`NTxAnts` — Number of transmission antennas

1 (default) | Optional | 2 | 4

Number of transmission antennas, specified as a positive scalar integer. Optional. Valid values are 1, 2, and 4.

Data Types: `double`

Data Types: `struct`

`chs` — PUSCH channel settings

structure

PUSCH channel settings, specified as a structure with the following fields.

`NLayers` — Number of transmission layers

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers, specified as 1, 2, 3, or 4. Optional.

Data Types: `double`

Data Types: `struct`

Output Arguments

`info` — Information related to PMI reporting

structure

Information related to PMI reporting, returned as a structure with these fields.

`k` — Subband size

scalar integer

Subband size, in resource blocks (RBs), returned as a scalar integer. This parameter is equal to NULRB.

Data Types: `double`

`NSubbands` — Number of subbands for PMI reporting

scalar integer

Number of subbands for PMI reporting, returned as a scalar integer. This parameter is equal to 1 for wideband PMI.

Data Types: `double`

`MaxPMI` — Maximum permitted PMI value for the given configuration

scalar integer

Maximum permitted PMI value for the given configuration, returned as a scalar integer. Valid PMI values range from 0 to MaxPMI.

Data Types: `double`

See Also

`ltePUSCH` | `ltePUSCHPrecode` | `lteULPMISelect`

lteULPMISelect

PUSCH precoder matrix indication calculation

Syntax

```
pmi = lteULPMISelect(ue,chs,hest,noiseest)
pmi = lteULPMISelect(ue,chs,hest,noiseest,refgrid)
pmi = lteULPMISelect(ue,chs,hest,noiseest,refgrid,cec)
```

Description

`pmi = lteULPMISelect(ue,chs,hest,noiseest)` performs PUSCH precoder matrix indication (PMI) calculation for given UE-specific settings, ue, channel configuration structure, chs, channel estimate resource array, hest, and receiver noise variance, noiseest. The output, pmi, is a scalar containing the PMI selected for closed-loop transmission.

hest is a 4-D array of size *M*-by-*N*-by-`NRxAnts`-by-`NTxAnts`, where *M* is the number of subcarriers, *N* is the number of SC-FDMA symbols, `NRxAnts` is the number of receive antennas, and `NTxAnts` is the number of transmit antennas.

noiseest is a scalar, an estimate of the received noise power spectral density.

`pmi = lteULPMISelect(ue,chs,hest,noiseest,refgrid)` provides an additional input refgrid, a 3-D *M*-by-*N*-by-`NTxAnts` array containing known transmitted data symbols in their correct locations. All other locations i.e. DRS Symbols and unknown data symbol locations should be represented by a `NaN`. This is the same array as the additional refgrid input described for the `lteULChannelEstimate` function. For PMI selection the symbols in refgrid are ignored, but the non-`NaN` RE locations are used as RE locations at which to sample the channel estimate and perform PMI estimation. This approach can be used to provide a refgrid containing for example the SRS RE locations created on all `NTxAnts`, allowing for full-rank channel estimation for the purposes of PMI selection when the PUSCH is transmitted with less than full rank.

`pmi = lteULPMISelect(ue,chs,hest,noiseest,refgrid,cec)` accepts channel estimator configuration structure cec containing the field `Reference`.

`Reference` = `'None'` will generate no internal reference signals, and the PMI estimation can be performed on arbitrary known REs as given by the refgrid argument. This approach can be used to provide a refgrid containing for example the SRS signals created on all `NTxAnts`, allowing for full-rank PMI estimation for the purposes of PMI selection when the PUSCH is transmitted with less than full rank. `Reference` = `'Antennas'` or `Reference` = `'Layers'` will use the PUSCH DMRS RE indices as reference locations for PMI estimation; additional references can still be provided in refgrid.

Examples

Calculate PUSCH PMI

This example creates an empty resource grid for RMC A3-2 and amend it for MIMO configuration.

Initialize ue specific parameter structure and create an empty resource grid for RMC A3-2 and amend it for MIMO configuration.

```
ue = lteRMCUL('A3-2');
ue.NTxAnts = 4;
ue.PUSCH.NLayers = 2;
rgrid = lteULResourceGrid(ue);
rgrid(ltePUSCHDRSIndices(ue,ue.PUSCH)) = ...
    ltePUSCHDRS(ue,ue.PUSCH);
```

Generate modulated waveform.

```
txWaveform = lteSCFDMAModulate(ue,rgrid);
```

Configure a fading channel.

```
chcfg.Seed = 100;
chcfg.DelayProfile = 'EPA';
chcfg.NRxAnts = 2;
chcfg.InitTime = 100;
chcfg.InitPhase = 'Random';
chcfg.ModelType = 'GMEDS';
chcfg.NTerms = 16;
chcfg.NormalizeTxAnts = 'On';
chcfg.NormalizePathGains = 'On';
chcfg.DopplerFreq = 50.0;
```

```
chcfg.MIMOCorrelation = 'Low';
chcfg.SamplingRate = 15360000;
```

Filter the transmit waveform through a fading channel and perform SC-FDMA demodulation.

```
rxWaveform = lteFadingChannel(chcfg,txWaveform);
rxSubframe = lteSCFDMADemodulate(ue,rxWaveform);
```

Estimate the corresponding channel and the noise power spectral density on the reference signal subcarriers.

```
cec = struct('FreqWindow',12,'TimeWindow',1,'InterpType','cubic');
cec.PilotAverage = 'UserDefined';
cec.Reference = 'Antennas';

[hest,noiseEst] = lteULChannelEstimate(ue,ue.PUSCH,cec,rxSubframe);
```

Use this estimate to calculate the precoder matrix indication (PMI).

```
pmi = lteULPMISelect(ue,ue.PUSCH,hest,noiseEst)
```

```
pmi =

     4
```

Input Arguments

ue — UE-specific settings

scalar structure

UE-specific settings, specified as a scalar structure with the following fields.

NULRB — Number of uplink (UL) resource blocks (RBs)

scalar integer

Number of uplink (UL) resource blocks (RBs), specified as a scalar integer.

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`NTxAnts` — Number of transmission antennas

1 (default) | Optional | 2 | 4

Number of transmission antennas, specified as 1, 2, or 4. Optional.

Data Types: `double`

Data Types: `struct`

`chs` — Channel configuration structure

scalar structure

Channel configuration structure, specified as a scalar structure with the following fields.

`PRBSet` — Physical Resource Block indices

numeric column matrix

Physical Resource Block indices, specified as a numeric column matrix. PRBSet can be a 1- or 2-column matrix, containing the 0-based Physical Resource Block indices (PRBs) corresponding to the resource allocations for this PUSCH.

Data Types: `double`

`NLayers` — Number of transmission layers

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers, specified as 1, 2, 3, or 4. Optional.

Data Types: `double`

Data Types: `struct`

`hest` — Channel estimate

4-D numeric array

Channel estimate, specified as a 4-D numeric array of size *M*-by-*N*-by-`NRxAnts`-by-`NTxAnts`. *M* is the number of subcarriers, *N* is the number of SC-FDMA symbols, `NRxAnts` is the number of receive antennas and `NTxAnts` is the number of transmit antennas.

Data Types: `double`

Complex Number Support: Yes

`noiseest` — Receiver noise variance

numeric scalar

Receiver noise variance, specified as a numeric scalar. It is an estimate of the received noise power spectral density.

Data Types: `double`

`refgrid` — Transmitted data symbols

3-D numeric array

Transmitted data symbols, specified as a 3-D numeric array. `refgrid` is an *M*-by-*N*-by-`NTxAnts` array containing known symbols in their correct locations.

Data Types: `double`
Complex Number Support: Yes

`cec` — Channel estimator configuration

scalar structure

Channel estimator configuration, specified as a scalar structure with the following fields.

`Reference` — Point of reference (indices to internally generate) for PMI estimation

`'Antennas'` (default) | Optional | `'Layers'` | `'None'`

Point of reference (indices to internally generate) for PMI estimation. `Reference` = `'None'` generates no internal reference signals, and the PMI estimation can be performed on arbitrary known REs as given by the refgrid argument. `Reference` = `'Antennas'` or `Reference` = `'Layers'` uses the PUSCH DMRS RE indices as reference locations for PMI estimation; additional references can still be provided in refgrid. Optional.

Data Types: `char`

Data Types: `struct`

Output Arguments

`pmi` — Precoder matrix indication selected for closed-loop transmission

numeric scalar (0...23)

Precoder matrix indication selected for closed-loop transmission, returned as a numeric scalar between 0 and 23.

See Also

`ltePUSCH` | `ltePUSCHPrecode` | `lteULPMIInfo`

lteULPerfectChannelEstimate

Uplink perfect channel estimation

Syntax

```
h = lteULPerfectChannelEstimate(ue,chs)
h = lteULPerfectChannelEstimate(ue,chs,toffset)
```

Description

`h = lteULPerfectChannelEstimate(ue,chs)` performs perfect channel estimation for a system configuration given by UE-specific settings, ue, and channel configuration structure, chs. It produces a perfect channel estimate, h, which is a 4-D array of size *M*-by-*N*-by-NRxAnts-by-NTxAnts, where *M* is the number of subcarriers, *N* is the number of OFDM symbols, NRxAnts is the number of receive antennas, and NTxAnts is the number of transmit antennas. The perfect channel estimates are only produced for the fading channel model created using the `lteFadingChannel` toolbox function. This function provides a perfect MIMO channel estimate after SC-FDMA modulation. This is achieved by setting the channel with the desired configuration and sending a set of known symbols through it, for each transmit antenna in turn.

chs must also include sufficient fields for `lteFadingChannel` to be configured. Prior to execution of the channel itself, chs will have the field `SamplingRate` set to the sampling rate of the time-domain waveform that is passed to it for channel filtering.

`h = lteULPerfectChannelEstimate(ue,chs,toffset)` takes the additional parameter, toffset, which specifies the timing offset from the start of the output of the channel to where SC-FDMA demodulation should be performed. This parameter allows h to be the precise channel that results in the case that the receiver is precisely synchronized, as for example by using the `lteULFrameOffset` function.

Examples

Estimate Perfect Uplink Channel

Perform perfect channel estimation using the `lteFadingChannel` function.

```
ue = struct('NULRB',6,'NTxAnts',2,'CyclicPrefixUL','Normal');
ue.TotSubframes = 1;
chs = struct('Seed',1,'DelayProfile','EPA','NRxAnts',4,'InitTime',0.0);
chs.DopplerFreq = 5.0;
chs.MIMOCorrelation = 'Low';
h = lteULPerfectChannelEstimate(ue,chs);
size(h)

    72    14     4     2
```

Input Arguments

ue — UE-specific settings

scalar structure

UE-specific settings, specified as a scalar structure that can contain the following fields.

NULRB — Number of uplink (UL) resource blocks (RBs)

scalar integer

Number of uplink (UL) resource blocks (RBs), specified as a scalar integer.

Data Types: `double`

CyclicPrefixUL — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

NTxAnts — Number of transmission antennas

1 | 2 | 4

Number of transmission antennas, specified as 1, 2, or 4.

Data Types: `double`

TotSubframes — Total number of subframes to be generated

scalar integer

Total number of subframes to be generated, specified as a scalar integer.

Data Types: `double`

Data Types: `struct`

`chs` — Channel configuration structure

scalar structure

Channel configuration structure, specified as a scalar structure. chs must contain the following fields and must also include sufficient fields for `lteFadingChannel` to be configured. Prior to execution of the channel, chs must have the field `SamplingRate` set to the sampling rate of the time-domain waveform that is passed to it for channel filtering.

`NRxAnts` — Number of receive antennas

1 | 2 | 4

Number of receive antennas, specified as 1, 2, or 4.

Data Types: `double`

Data Types: `struct`

`toffset` — Timing offset

scalar integer

Timing offset, specified as a scalar integer. Specifies the timing offset from the start of the output of the channel to where SC-FDMA demodulation should be performed.

Data Types: `double`

Output Arguments

`h` — Perfect channel estimate

4-D complex-valued numeric array

Perfect channel estimate, returned as a 4-D complex-valued numeric array of size *M*-by-*N*-by-NRxAnts-by-NTxAnts, where *M* is the number of subcarriers, *N* is the number of OFDM symbols, `NRxAnts` is the number of receive antennas, and `NTxAnts` is the number of transmit antennas.

Data Types: `double`
Complex Number Support: Yes

See Also

`lteULChannelEstimate` | `lteULChannelEstimatePUCCH1` | `lteULChannelEstimatePUCCH2` | `lteULChannelEstimatePUCCH3`

lteULPrecode

SC-FDMA precoding

Syntax

```
out = lteULPrecode(in,nrb)
```

Description

`out = lteULPrecode(in,nrb)` performs SC-FDMA precoding of the complex modulation symbols in for PUSCH configuration with a bandwidth of nrb resource blocks. in is a *MSymb*-by-*NLayers* matrix of values and out has the same dimensions.

Examples

Perform SC-FDMA Precoding on Complex Modulation Symbols

Generate the complex modulated symbols for RMC A3-2. The symbol matrix has size *MSymb*-by-*NLayers*, where *MSymb* represents the total number of RE in a grid. Then, perform SC-FDMA precoding on the symbols for PUSCH configuration.

```
ue = lteRMCUL('A3-2');
ueDim = lteULResourceGridSize(ue);
mSymb = ue.NULRB*ueDim(1)*ueDim(2);
realIn = randi([0,1],mSymb,ue.PUSCH.NLayers);
imagIn = randi([0,1],mSymb,ue.PUSCH.NLayers);
out = lteULPrecode(complex(realIn,imagIn),ue.NULRB);
size(out)
```

```
        6048           1
```

Input Arguments

`in` — Complex modulation symbols

numeric matrix

Complex modulation symbols, specified as a numeric matrix of size *MSymb*-by-*NLayers*.

Data Types: `double` | `single` | `uint8` | `uint16` | `uint32` | `uint64` | `int8` | `int16` | `int32` | `int64`
Complex Number Support: Yes

`nrb` — Number of resource blocks
scalar integer

Number of resource blocks, specified as a scalar integer.

Data Types: `double`

Output Arguments

`out` — Precoded PUSCH output symbols
numeric matrix

Precoded PUSCH output symbols, returned as a numeric matrix of size *MSymb*-by-*NLayers*.

Data Types: `double` | `single` | `uint8` | `uint16` | `uint32` | `uint64` | `int8` | `int16` | `int32` | `int64`
Complex Number Support: Yes

See Also
`lteLayerMap` | `ltePUSCH` | `ltePUSCHPrecode` | `lteULDeprecode`

lteULResourceGrid

Uplink subframe resource array

Syntax

```
grid = lteULResourceGrid(ue)
grid = lteULResourceGrid(ue,P)
```

Description

`grid = lteULResourceGrid(ue)` returns an empty resource array generated from the UE-specific settings structure ue. It returns an empty multidimensional array used to represent the resource elements for one subframe across all configured antenna ports, as described in “Data Structures”.

The size of grid is *N*-by-*M*-by-*P*, where *N* is the number of subcarriers, 12×`NULRB`, *M* is the number of SC-FDMA symbols in a subframe, 14 for normal cyclic prefix and 12 for extended cyclic prefix, and *P* is the number of transmission antennas.

`grid = lteULResourceGrid(ue,P)` returns a resource array as above, except the number of antenna planes in the array is specified directly by parameter P. In this case, `NTxAnts` is not required as a structure field of ue.

Examples

Create Resource Array

Create an empty resource array representing the resource elements for 10 MHz bandwidth.

```
rgrid = lteULResourceGrid(struct('NULRB',50));
```

Input Arguments

ue — UE-specific settings

scalar structure

UE-specific settings, specified as a scalar structure with the following fields.

`NULRB` — Number of uplink (UL) resource blocks (RBs)

scalar integer

Number of uplink (UL) resource blocks (RBs), specified as a scalar integer.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`NTxAnts` — Number of transmission antennas

1 (default) | Optional | 2 | 4

Number of transmission antennas, specified as a positive scalar integer. Optional.

Data Types: `double`

Data Types: `struct`

`P` — Number of antenna planes in the return array

scalar integer

Number of antenna planes in the return array, specified as a scalar integer.

Data Types: `double`

Output Arguments

`grid` — Empty resource grid

multidimensional numeric array

Empty resource grid, returned as a multidimensional numeric array. It is used to represent the resource elements for one subframe across all configured antenna ports. Its size is *N*-by-*M*-by-*P*. *N* is the number of subcarriers (12×`NULRB`), *M* is the number of SC-FDMA symbols in a subframe, 14 for normal cyclic prefix and 12 for extended cyclic prefix, and *P* is the number of transmission antennas.

Data Types: `double`

See Also

`lteDLResourceGrid` | `lteResourceGrid` | `lteSCFDMAModulate` | `lteULResourceGridSize`

lteULResourceGridSize

Size of uplink subframe resource array

Syntax

```
D = lteULResourceGridSize(ue)
D = lteULResourceGridSize(ue,P)
```

Description

`D = lteULResourceGridSize(ue)` returns a 3-element row vector of dimension lengths for the resource array generated from the UE-specific settings structure ue. It returns a 3-element row vector of dimension lengths for the multidimensional array used to represent the resource elements for one subframe across all configured antenna ports, as described in “Data Structures”.

The vector D is [*N M P*], where *N* is the number of subcarriers, 12×`NULRB`, *M* is the number of SC-FDMA symbols in a subframe, 14 for normal cyclic prefix and 12 for extended cyclic prefix, and *P* is the number of transmission antennas.

`D = lteULResourceGridSize(ue,P)` returns a 3-element row vector as above, except the number of antenna planes in the array is specified directly by parameter P. In this case, `NTxAnts` is not required as a structure field of ue.

Examples

Get Size of Uplink Subframe Resource Array

Use the returned vector to directly create MATLAB® arrays.

```
ue = struct('NULRB',6,'CyclicPrefixUL','Normal','NTxAnts',1);
rgrid = zeros(lteULResourceGridSize(ue));
size(rgrid)
```

```
ans =
```

```
    72    14
```

This result is the same matrix as the one obtained using the `lteULResourceGrid` function.

Input Arguments

`ue` — UE-specific settings

scalar structure

UE-specific settings, specified as a scalar structure with the following fields.

`NULRB` — Number of uplink (UL) resource blocks (RBs)

scalar integer

Number of uplink (UL) resource blocks (RBs), specified as a scalar integer.

Data Types: `double`

`CyclicPrefixUL` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`NTxAnts` — Number of transmission antennas

1 (default) | Optional | 2 | 4

Number of transmission antennas, specified as 1, 2, or 4. Optional.

Data Types: `double`

Data Types: `struct`

`P` — Number of antenna planes in the return array

scalar integer

Number of antenna planes in the return array, specified as a scalar integer.

Data Types: `double`

Output Arguments

D — Dimension lengths

3-element row vector

Dimension lengths, returned as a 3-element row vector [*N M* P]. *N* is the number of subcarriers (12×NULRB), *M* is the number of SC-FDMA symbols in a subframe (14 for normal cyclic prefix and 12 for extended cyclic prefix) and P is the number of transmission antennas.

Data Types: `double`

See Also

`lteDLResourceGridSize` | `lteResourceGridSize` | `lteULResourceGrid`

lteULSCH

Uplink shared channel

Syntax

```
[cwout,chinfo] = lteULSCH(ue,chs,trblkin)
[cwout,chinfo] = lteULSCH(ue,chs,trblkin,opts)
[cwout,chinfo] = lteULSCH(ue,chs,trblkin,cqi,ri,ack,opts)
```

Description

`[cwout,chinfo] = lteULSCH(ue,chs,trblkin)` performs complete UL-SCH transport coding and UCI coding on the input information bits, trblkin, and returns the complete codewords in the output, cwout. It encodes both a single transport block or pair of blocks, contained in a cell array, for the case of spatial multiplexing schemes transmitting two codewords, represented by input trblkin without any UCI data. The lowest order information bit of trblkin should be mapped to the most significant bit of the transport block, as defined in section 6.1.1 of [3]. The encoding process also includes the channel interleaving. It The transport encoding includes type-24A CRC calculation, code block segmentation and type-24B CRC attachment, turbo encoding, rate matching, block concatenation, and channel interleaving. For more information, see sections 5.2.2.1 to 5.2.2.5 and 5.2.2.8 of [2]. Parameter information relating to the underlying UL-SCH and UCI coding is available in structure chinfo.

The output chinfo is a structure containing information related to the UL-SCH coding process.

For multiple transport blocks, each structure in the array corresponds to one of the blocks. This output is also available from the `lteULSCHInfo` function.

`[cwout,chinfo] = lteULSCH(ue,chs,trblkin,opts)` allows for the merging of the input chs structure fields into chinfo at the output.

If the UL-SCH encoding is for a retransmission of a previously sent transport block, use the three “Init” fields, `'InitPRBSet'`, `'InitCyclicPrefixUL'`, and `'InitShortened'`. If any of these fields are absent, their values are assumed to be the same as the values for the associated current subframe fields, `'PRBSet'`, `'CyclicPrefixUL'`, and `'Shortened'`.

opts is an optional input string, or cell array, parameter which enables the concatenation or merging of the chs input structure fields into the fields returned by chinfo. This parameter is useful for combining the high-level configuration parameters with the fine-grained ones used in the encoding process.

opts allows additional control of the contents and format of the chinfo output through a cell array of option strings.

`[cwout,chinfo] = lteULSCH(ue,chs,trblkin,cqi,ri,ack,opts)` encodes and multiplexes the UCI input data, CQI, RI, and ACK, along with the information bits, trblkin, in the codeword, cwout. For more information, see sections 5.2.2.6 to 5.2.2.8 of [2]. Any of the trblkin, cqi, ri, or ack vectors can be empty if that data is not present. If trblkin is empty, only UCI on UL-SCH/PUSCH is processed, according to section 5.2.4 of [2]. The coding of the UCI can be controlled through the additional fields, `BetaACK`, `BetaCQI`, `BetaRI`, and `NBundled`, in the chs input structure. Setting `NBundled` to 0 disables the TDD HARQ-ACK bundling scrambling; therefore, it is off by default.

Examples

Create Coded Information Bits for UL-SCH

Create coded information bits for uplink FRC A3-3, 3MHz, as specified in [1].

```
rmc = lteRMCUL('A3-3');
rmc.PUSCH.PRBSet = (0:14).';
cw = lteULSCH(struct(),rmc.PUSCH,randi([0,1],rmc.PUSCH.TrBlkSizes(1),1));
```

Input Arguments

ue — UE-specific settings

structure

UE-specific settings, specified as a structure with the following fields.

Parameter Field	Required or Optional	Values	Description
`CyclicPrefixUL`	Optional	`'Normal'` (default), `'Extended'`	Current cyclic prefix length

Parameter Field	Required or Optional	Values	Description
`Shortened`	Optional	0 (default), 1	Shorten subframe flag. If 1, the last symbol of the subframe is not used. It should be set if the current subframe contains a possible SRS transmission.

`chs` — Channel-specific transmission configuration

scalar structure

Channel-specific transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`Modulation`	Required	`'QPSK'`, `'16QAM'`, `'64QAM'`, cell array of strings	Modulation type, specified as a string or cell array of strings. If 2 blocks, each cell is associated with a transport block.
`NLayers`	Optional	1 (default), 2, 3, 4	Total number of transmission layers associated with the transport block or blocks.
`RV`	Required	0, 1, 2, 3, 2-element integer vector	Redundancy version indicators, specified as a vector of 1 or 2 values. Each value can be an integer between 0 and 3. This parameter field is not required if trblkin is `[]`, which signifies that the UL-SCH is carrying only UCI and no transport data.
`PRBSet`	Required	1- or 2-column integer matrix	0-based physical resource block indices (PRBs) for the slots of the current PUSCH resource allocation. As a

Parameter Field	Required or Optional	Values	Description
			column vector, the resource allocation is the same in both slots of the subframe. As a two-column matrix, it specifies different PRBs for each slot in a subframe.
The following three `'Init'` fields should be used if the UL-SCH encoding is for a retransmission of a previously sent transport block. If any of these fields are absent, its value is assumed to be the same as the value for its associated current subframe field.			
`InitPRBSet`	Optional	1- or 2-column integer matrix, `PRBSet` (default)	PRB indices used in the initial transmission PUSCH allocation. If this field is absent, its value is assumed to be the same as the value for the associated current subframe field, `PRBSet`.
`InitCyclicPrefixU`	Optional	`'Normal'`, `'Extended'`, `CyclicPrefixUL` (default)	Cyclic prefix length of initial transmit subframe. This is the length used during the first transmission of this transport block. If this field is absent, its value is assumed to be the same as the value for the associated current subframe field, `CyclicPrefixUL`.
`InitShortened`	Optional	0, 1, `Shortened` (default)	Initial transmit subframe shortened flag. If 1, the initial transmit subframe was shortened for possible SRS. If this field is absent, its value is assumed to be the same as the value for the associated current subframe field, `Shortened`.
The coding of the UCI can be controlled through the following additional fields.			

Parameter Field	Required or Optional	Values	Description
`BetaCQI`	Optional	numeric scalar, 2.0 (default)	Modulation and coding scheme (MCS) offset for CQI and PMI bits
`BetaRI`	Optional	numeric scalar, 2.0 (default)	Modulation and coding scheme (MCS) offset for RI bits
`BetaACK`	Optional	numeric scalar, 2.0 (default)	Modulation and coding scheme (MCS) offset for HARQ-ACK bits. This field was previously named `BetaHI`; if this field is absent but `BetaHI` is present, it is used as before.
`NBundled`	Optional	0 (default), 1, …, 9	TDD HARQ-ACK bundling scrambling sequence index. When set to 0, the function disables the TDD HARQ-ACK bundling scrambling. Therefore, it is off by default.

`trblkin` — Input transport blocks

numeric vector | cell array of numeric vectors

Input transport blocks, specified as a numeric vector or a cell array of numeric vectors.

Data Types: `double` | `cell`

`opts` — Options to control output contents and format

string | cell array of strings

Options to control output contents and format, specified as a string or a cell array of strings. You may choose any of the option strings listed in the following table.

Option	Values	Description
Channel parameter merging	`'nochsconcat'` (default)	Do not concatenate `chs` input structure into `chinfo`.
	`'chsconcat'`	Concatenate `chs` input structure into `chinfo`.

Option	Values	Description
Output structure format	`'cwseparate'` (default)	Information values for each codeword are output in separate elements of the 1-by-*ncodewords* structure array `chinfo`.
	`'cwcombined'`	Information values for each codeword are combined into the fields of a single scalar, or 1-by-1, structure.

Data Types: `char` | `cell`

`cqi` — CQI input data
numeric vector

CQI input data, specified as a numeric vector. Part of the UCI data.

Data Types: `double`

`ri` — RI input data
numeric vector

RI input data, specified as a numeric vector. Part of the UCI data.

Data Types: `double`

`ack` — HARQ-ACK input data
numeric vector

HARQ-ACK input data, specified as a numeric vector. Part of the UCI data.

Data Types: `double`

Output Arguments

`cwout` — Complete output codewords
integer column vector | cell array of integer column vectors

Complete output codewords, returned as an integer column vector or a cell array of integer column vectors.

Data Types: `int8` | `cell`

`chinfo` — Parameter information relating to the underlying UL-SCH and UCI coding
structure | structure array

Parameter information relating to the underlying UL-SCH and UCI coding, returned as a structure or a structure array. If two transport blocks are encoded, chinfo is a structure array of two elements, one for each block. Alternatively, you can create code block segmentation fields in this structure independently, by calling the `lteULSCHInfo` function. chinfo contains the following fields.

Parameter Field	Description	Values	Data Type
`C`	Total number of code blocks	nonnegative scalar integer	`int32`
`Km`	Lower code block size ($K–$)	nonnegative scalar integer	`int32`
`Cm`	Number of code blocks of size `Km` ($C–$)	nonnegative scalar integer	`int32`
`Kp`	Upper code block size ($K+$)	nonnegative scalar integer	`int32`
`Cp`	Number of code blocks of size `Kp` ($C+$)	nonnegative scalar integer	`int32`
`F`	Number of filler bits in first block	nonnegative scalar integer	`int32`
`L`	Number of segment cyclic redundancy check (CRC) bits	nonnegative scalar integer	`int32`
`Bout`	Total number of bits in all segments	nonnegative scalar integer	`int32`
`G`	Number of coded and rate matched UL-SCH data bits	nonnegative scalar integer	`int32`
`Qm`	Number of bits per symbol	nonnegative scalar integer	`int32`
`Gd`	Number of coded and rate matched UL-SCH data symbols (G')	nonnegative scalar integer	`int32`
`OCQI`	Number of uncoded channel quality information (CQI) bits	nonnegative scalar integer	`int32`
`ORI`	Number of uncoded symbols for RI	nonnegative scalar integer	`int32`
`OACK`	Number of uncoded symbols for ACK/NACK	nonnegative scalar integer	`int32`

Parameter Field	Description	Values	Data Type
QdCQI	Number of coded symbols for CQI (*Q'_CQI*)	nonnegative scalar integer	int32
QdRI	Number of coded symbols for RI (*Q'_RI*)	nonnegative scalar integer	int32
QdACK	Number of coded symbols for ACK/NACK (*Q'_ACK*)	nonnegative scalar integer	int32
NRE	Number of resource elements (REs) used for PUSCH transmission	nonnegative scalar integer	int32
NLayers	Number of layers associated with one codeword	nonnegative scalar integer	int32
Modulation	Modulation scheme associated with one codeword	'QPSK', '16QAM', '64QAM'	char
RV	RV value associated with one codeword	scalar integer	int32

References

[1] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[3] 3GPP TS 36.321. "Medium Access Control (MAC) protocol specification." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`ltePUSCH` | `lteULSCHDecode` | `lteULSCHInfo` | `lteULSCHInterleave`

lteULSCHDecode

Uplink shared channel decoding

Syntax

```
[trblkout,blkcrc,stateout] = lteULSCHDecode(ue,chs,trblklen,cwin,
statein)
```

Description

`[trblkout,blkcrc,stateout] = lteULSCHDecode(ue,chs,trblklen,cwin, statein)` returns the information bits trblkout decoded from the input soft LLR codewords data cwin. The UL-SCH decoder includes channel deinterleaver, rate recovery, turbo decoding, block concatenation and CRC calculations. The function also returns the type-24A transport block CRC decoding result in blkcrc and the HARQ process decoding state in stateout. The initial HARQ process state can be input via the optional statein parameter. The function is capable of processing both a single codeword or pairs of codewords, contained in a cell array, for the case of spatial multiplexing schemes transmitting two codewords. The type of the return variable trblkout is the same as input cwin. If cwin is a cell array containing one or two codewords, trblkout returns a cell array of one or two transport blocks. If cwin is a vector of soft data, trblkout returns a vector too. If decoding a pair of codewords, pairs of modulation schemes and RV indicators are required to be defined in the associated parameter fields below. This function only decodes the information bits, but supports the presence of UCI data, CQI, RI, and HARQ-ACK, in the input codeword. UCI should be demultiplexed then decoded separately.

Strictly speaking, because all the fields in structure ue are optional, it is legal for this parameter to be an empty structure.

Multiple codewords can be parameterized by two different forms of the chs structure. Each codeword can be defined by separate elements of a 1-by-2 structure array, or the codeword parameters can be combined together in the fields of a single scalar, or 1-by-1, structure. Any scalar field values apply to both codewords and a scalar `NLayers` is the total number. See “UL-SCH Parameterization” for more details.

trblklen is an input vector (one or two elements in length) defining the transport block lengths that the input code blocks should be rate recovered and decoded to.

cwin is an input parameter containing the floating point soft LLR data of the codewords to be decoded. It can either be a single vector or a cell array containing one or two vectors. If the latter, then all rate matching calculations assume that the pair were transmitting on a single PUSCH, distributed across the total number of layers defined in chs, as per [1].

statein is an optional input structure array (empty or one or two elements) which can input the current decoder buffer state for each transport block in an active HARQ process. If statein is not an empty array and it contains a non-empty field `CBSBuffers` then this field should contain a cell array of vectors representing the LLR soft buffer states for the set of code blocks at the input to the turbo decoder i.e. after explicit rate recovery. The updated buffer states after decoding are returned in the `CBSBuffers` field in the output parameter stateout. The statein array would normally be generated and recycled from the stateout of previous calls to lteULSCHDecode as part of a sequence of HARQ transmissions.

trblkout is the output parameter containing the decoded information bits. It is either a single vector or a cell array containing one or two vectors, depending on the class and dimensionality of cwin.

blkcrc is an output array (one or two elements) containing the result of the type-24A transport block CRC decoding for the transport blocks.

stateout, the final output parameter, is a one element structure array containing the internal state of each transport block decoder in the fields `CBSBuffers`, `CBSCRC`, `blkcrc`.

The stateout array would normally be reapplied via the statein variable of subsequent `lteULSCHDecode` function calls as part of a sequence of HARQ retransmissions.

Examples

Decode UL-SCH

Generate and decode 2 transmissions, the first with `RV` set to 0 and the second with `RV` set to 2, as part of a single codeword HARQ process for the A3-3 FRC.

First, get the definition of FRC A3-3. Create a codeword with `RV` set to 0. Turn logical bits into LLR data. Initialize the decoder states for the first HARQ transmission. Decode the first transmission.

```
nsf = 1;
frc = lteRMCUL('A3-3');
trBlkLen = frc.PUSCH.TrBlkSizes(nsf);
trBlkData = randi([0,1],trBlkLen,1);
frc.PUSCH.RV = 0;
cw = lteULSCH(frc,frc.PUSCH,trBlkData);
cw(cw == 0) = -1;
decState =[];
[rxTrBlk,~,decState] = lteULSCHDecode(frc,frc.PUSCH,trBlkLen,cw,decState);
```

Create a second, retransmitted codeword with RV set to 2. Turn logical bits into LLR data. Decode the second transmission.

```
frc.PDSCH.RV = 2;
cWord = lteULSCH(frc,frc.PUSCH,trBlkData);
cWord(cWord == 0) = -1;
rxTrBlk = lteULSCHDecode(frc,frc.PUSCH,trBlkLen,cWord,decState);
```

Input Arguments

ue — UE-specific settings

scalar structure

UE-specific settings, specified as a scalar structure with the following fields. Because all the fields in structure ue are optional, this parameter can be an empty structure.

CyclicPrefixUL — Current cyclic prefix length

'Normal' (default) | Optional | 'Extended'

Current cyclic prefix length, specified as a string. Optional.

Data Types: char

Shortened — Shorten subframe flag

0 (default) | Optional | 1

Shorten subframe flag, specified as 0 or 1. Optional. If 1, the last symbol of the subframe is not used. It should be set if the current subframe contains a possible SRS transmission.

Data Types: logical

Data Types: `struct`

`chs` — UL-SCH related parameters

scalar structure

UL-SCH related parameters, specified as a scalar structure with the following fields.

`Modulation` — Modulation scheme associated with each transport block

`'QPSK' | '16QAM' | '64QAM'`

Modulation scheme associated with each transport block, specified as a string or, if there are 2 blocks, as a cell array of strings.

Data Types: `char | cell`

`NLayers` — Number of transmission layers

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers (total or per codeword), specified as a positive scalar integer. Optional.

Data Types: `double`

`RV` — Redundancy version indicators

0 | 1 | 2 | 3

Redundancy version indicators, specified as a vector of 1 or 2 indicators.

Data Types: `double`

`QdCQI` — Number of coded CQI symbols

0 (default) | Optional | nonnegative scalar integer

Number of coded CQI symbols, specified as a nonnegative scalar integer. Optional. (*Q'_CQI*)

Data Types: `double`

`QdRI` — Number of coded RI symbols

0 (default) | Optional | nonnegative scalar integer

Number of coded RI symbols, specified as a nonnegative scalar integer. Optional. (*Q'_RI*)

Data Types: `double`

`QdACK` — Number of coded ACK symbols

0 (default) | Optional | nonnegative scalar integer

Number of coded ACK symbols, specified as a nonnegative scalar integer. Optional. (*Q'_ACK*)

Data Types: `double`

`NTurboDecIts` — Number of turbo decoder iteration cycles

5 (default) | Optional | scalar integer

Number of turbo decoder iteration cycles, specified as a scalar integer. Optional.

Data Types: `double`

Data Types: `struct`

`trblklen` — Transport block length

numeric vector

Transport block length, specified as a numeric vector (one or two elements in length) defining the transport block lengths that the input code blocks should be rate recovered and decoded to.

Data Types: `double`

`cwin` — Soft LLR codeword data

numeric column vector | cell array of one or two column vectors

Soft LLR codeword data, specified as a numeric column vector or a cell array of one or two column vectors. This argument contains the floating point soft LLR data of the codeword or codewords to be decoded. It can either be a single vector or a cell array containing one or two vectors. If a cell array, all rate matching calculations assume that the pair were transmitting on a single PUSCH, distributed across the total number of layers defined in chs, as specified in [1].

Data Types: `int8` | `cell`

`statein` — Initial HARQ process state

Optional | structure array

Initial HARQ process state, specified as a structure array. Optional. It can be empty or have one or two elements, which can input the current decoder buffer state for each transport block in an active HARQ process.

Data Types: `double`

Output Arguments

`trblkout` — Decoded information bits

numeric vector | cell array

Decoded information bits, returned as a numeric vector or cell array. trblkout contains decoded transport data blocks. It is either a single vector or a cell array containing one or two vectors, depending on the class and dimensionality of cwin.

Data Types: `double` | `cell`

`blkcrc` — Type 24A transport block CRC decoding result

0 or 1

Type 24A transport block CRC decoding result, returned as 0 or 1.

Data Types: `logical`

`stateout` — HARQ process decoding state

structure | structure array

HARQ process decoding state, returned as a one-element structure array. It contains the internal state of each transport block decoder. It contains the following parameter fields.

Parameter Field	Description	Values	Data Type
`CBSBuffers`	LLR soft buffer states for the set of code blocks associated with a single transport block. The buffers are positioned at the input to the turbo decoder, or after explicit rate recovery.	Cell array of numeric vectors	`cell`
`CBSCRS`	Type-24B code block set CRC decoding results	Numeric vector	`int8`
`BLKCRC`	Type-24A transport block CRC decoding error	One- or two-element numeric vector	`logical`

References

[1] 3GPP TS 36.211. “Physical Channels and Modulation.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`ltePUSCHDecode` | `lteULSCH` | `lteULSCHDeinterleave` | `lteULSCHInfo`

lteULSCHDeinterleave

UL-SCH deinterleaving

Syntax

```
[cdata,ccqi,cri,cack] = lteULSCHDeinterleave(ue,chs,in)
```

Description

`[cdata,ccqi,cri,cack] = lteULSCHDeinterleave(ue,chs,in)` returns the deinterleaved data vector cdata, encoded UCI vectors, ccqi,cri, and cack, or cell array of vectors, after performing the demultiplexing and UL-SCH channel deinterleaving to undo the processing described in sections 5.2.2.7 and 5.2.2.8 of [1] for UE-specific settings, ue, and UL-SCH channel specific configuration, chs.

The function expects the input in to be multiplexed and interleaved as per defined in sections 5.2.2.7 and 5.2.2.8 of [1]. This input can be a vector or a cell array of vectors, deinterleaved and demultiplexed separately, and the outputs are of the same form. The size of the coded CQI symbols and codeword number with it is multiplexed, to correctly perform the demultiplexing, are deduced using the channel specific structure chs via the `Modulation` and `QdCQI` parameters. The presence or absence of ccqi in the transmission is signaled via `QdCQI` parameter with nonzero (number of coded CQI symbols) or zero value, respectively.

Multiple codewords can be parameterized by two different forms of the chs structure. Each codeword can be defined by separate elements of a 1-by-2 structure array, or the codeword parameters can be combined together in the fields of a single scalar, or 1-by-1, structure. Any scalar field values apply to both codewords and a scalar `NLayers` is the total number. See “UL-SCH Parameterization” for more details.

Examples

Deinterleave Vector of Interleaver Input Bit Indices

Perform back-to-back interleaving and deinterleaving of a vector of interleaver input bit indices.

```
chs = struct('Modulation','QPSK');
interleave = lteULSCHInterleave(struct(),chs,(1:288).');
deinterleave = lteULSCHDeinterleave(struct(),chs,interleave);
deinterleave(1:10)
```

```
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
```

For verification purposes, see that the result, `deint`, has the same values as the ones you input.

Input Arguments

`ue` — UE-specific settings

structure

UE-specific settings, specified as a structure with the following fields.

`CyclicPrefixUL` — Cyclic prefix length

`'Normal'` (default) | Optional | `'Extended'`

Cyclic prefix length, specified as a string. Optional.

Data Types: `char`

`Shortened` — Shorten subframe flag

0 (default) | Optional | 1

Shorten subframe flag, specified as 0 or 1. Optional. If 1, the last symbol of the subframe is not used. It should be set if the current subframe contains a possible SRS transmission.

Data Types: `logical` | `double`

Data Types: `struct`

`chs` — UL-SCH related parameters

structure

UL-SCH related parameters, specified as a structure with the following fields.

`Modulation` — Modulation scheme type

`'QPSK'` | `'16QAM'` | `'64QAM'` | cell array of strings

Modulation format string, specified as a string or cell array of strings (if 2 blocks) associated with each transport block.

Data Types: `char`

`NLayers` — Number of transmission layers

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers, total or per codeword, specified as a positive scalar integer. Optional.

Data Types: `double`

`QdCQI` — Number of coded symbols for CQI

0 (default) | Optional | nonnegative scalar integer

Number of coded symbols for CQI, specified as a nonnegative scalar integer. Optional. (*Q'_CQI*)

Data Types: `double`

`QdRI` — Number of coded symbols for RI

0 (default) | Optional | nonnegative scalar integer

Number of coded symbols for RI, specified as a nonnegative scalar integer. Optional. (*Q'_RI*)

Data Types: `double`

`QdACK` — Number of coded symbols for ACK/NACK

0 (default) | Optional | nonnegative scalar integer

Number of coded symbols for ACK/NACK, specified as a nonnegative scalar integer. Optional. (*Q'_ACK*)

Data Types: `double`

Data Types: `struct`

`in` — Input data

column vector | cell array of column vectors

Input data specified as a column vector or a cell array of column vectors.

Data Types: `double` | `cell`

Output Arguments

`cdata` — Deinterleaved data

column vector | cell array of column vectors

Deinterleaved data, returned as a column vector or cell array of column vectors.

Data Types: `double` | `cell`

`ccqi` — Encoded UCI

vector | cell array of vectors

Encoded UCI, returned as a vector or cell array of vectors.

Data Types: `double` | `cell`

`cri` — Encoded UCI

vector | cell array of vectors

Encoded UCI, returned as a vector or cell array of vectors.

Data Types: `double` | `cell`

`cack` — Encoded UCI

vector | cell array of vectors

Encoded UCI, returned as a vector or cell array of vectors.

Data Types: `double` | `cell`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also

`lteACKDecode` | `lteCQIDecode` | `lteRateRecoverTurbo` | `lteRIDecode` | `lteULSCH` | `lteULSCHInfo` | `lteULSCHInterleave`

lteULSCHInfo

UL-SCH coding information

Syntax

```
info = lteULSCHInfo(ue,chs,blklen)
info = lteULSCHInfo(ue,chs,blklen,opts)
info = lteULSCHInfo(ue,chs,blklen,ocqi,ori,oack)
info = lteULSCHInfo(ue,chs,blklen,ocqi,ori,oack,opts)
```

Description

`info = lteULSCHInfo(ue,chs,blklen)` provides information related to the entire UL-SCH coding process, for UL-SCH data without UCI. It returns a structure array with fields covering the transport channel (TrCH) encoding and UCI multiplexing. When UCI is present, it includes the coded symbol capacities given UCI sizes, PUSCH resource allocations, and *Beta* offset values, which can be useful in a number of UL-SCH- and PUSCH-related functions. These symbol capacities are calculated from the Q' formulae in sections 5.2.2.6 and 5.2.4.1 of [1]. The one- or two-element vector, blklen, defines the length of the transmitted transport blocks.

By default, in the case of multiple transport blocks or codewords, each structure in the array corresponds to one of the blocks. Note that the `NLayers`, `Modulation`, and `RV` fields at the output contain only the value for the associated codeword and therefore have a different form to those given in the input. In the case of `NLayers` the output returns the number of layers per codeword where the input field represents the total number of transmission layers across all codewords.

If the UL-SCH encoding is for a retransmission of a previously sent transport block, use the three "Init" fields, `'InitPRBSet'`, `'InitCyclicPrefixUL'`, and `'InitShortened'`. If any of these fields are absent, their values are assumed to be the same as the values for the associated current subframe fields, `'PRBSet'`, `'CyclicPrefixUL'`, and `'Shortened'`.

`info = lteULSCHInfo(ue,chs,blklen,opts)` controls the contents and format of the output info through a cell array of option strings, opts. The optional parameter opts allows for the merging of the input chs structure fields into info at the output.

`info = lteULSCHInfo(ue,chs,blklen,ocqi,ori,oack)` supports the multiplexing of both transport and UCI data, CQI, RI, and HARQ-ACK, or UCI only. The number of uncoded UCI bits is given by ocqi, ori and oack respectively. Any of the data length parameters can be zero if the associated data is not present. The coding of the UCI can be controlled through the additional `BetaACK`, `BetaCQI`, and `BetaRI` fields in the chs input structure.

`info = lteULSCHInfo(ue,chs,blklen,ocqi,ori,oack,opts)` supports the multiplexing of both transport and UCI data (CQI, RI, HARQ-ACK) or UCI only, and controls the contents and format of the output info through a cell array of option strings, opts. The optional parameter opts allows for the merging of the input chs structure fields into info at the output.

Examples

Obtain UL-SCH Information for One Transport Block

Obtain information for UL-SCH coding of a single transport block of length 6712 bits.

```
pusch = struct('Modulation','QPSK','NLayers',1,'PRBSet',(0:74).');
info = lteULSCHInfo(struct(),pusch,6712)
```

```
info =

             C: 2
            Km: 3328
            Cm: 0
            Kp: 3392
            Cp: 2
             F: 0
             L: 24
          Bout: 6784
             G: 21600
            Qm: 2
            Gd: 10800
          OCQI: 0
           ORI: 0
          OACK: 0
         QdCQI: 0
          QdRI: 0
         QdACK: 0
           NRE: 10800
```

```
       NLayers: 1
    Modulation: 'QPSK'
```

Obtain UL-SCH Information for 2 Transport blocks in Structure Array

Obtain information for the UL-SCH coding of two transport blocks with UCI, specifying 3-bit RI and 2-bit HARQ-ACK.

```
pusch.Modulation = {'QPSK','16QAM'};
pusch.NLayers = 3;
pusch.PRBSet = (0:74).';
info = lteULSCHInfo(struct(),pusch,[6712,6712],0,3,2)
```

```
info =

1x2 struct array with fields:

    C
    Km
    Cm
    Kp
    Cp
    F
    L
    Bout
    G
    Qm
    Gd
    OCQI
    ORI
    OACK
    QdCQI
    QdRI
    QdACK
    NRE
    NLayers
    Modulation
```

Obtain UL-SCH Information for 2 Transport Blocks in Scalar Structure

Obtain information in a single scalar structure for the UL-SCH coding of two transport blocks with UCI, specifying 3-bit RI and 2-bit HARQ-ACK.

Obtain dimension information for UL-SCH coding of a single transport block of length 6712 bits.

```
pusch.Modulation = {'QPSK','16QAM'};
pusch.NLayers = 3;
pusch.PRBSet = (0:74).';
info = lteULSCHInfo(struct(),pusch,[6712,6712],0,3,2,'cwcombined')

info =

            C: [2 2]
           Km: [3328 3328]
           Cm: [0 0]
           Kp: [3392 3392]
           Cp: [2 2]
            F: [0 0]
            L: [24 24]
        Bout: [6784 6784]
           G: [21590 86360]
          Qm: [2 4]
          Gd: [10795 21590]
        OCQI: 0
         ORI: 3
        OACK: 2
       QdCQI: [0 0]
        QdRI: [5 5]
       QdACK: [4 4]
         NRE: [10800 21600]
    NLayers: [1 2]
Modulation: {'QPSK'   '16QAM'}
```

Input Arguments

ue — UE-specific configuration settings

structure

UE-specific configuration settings, specified as a structure that can contain the following fields.

Parameter Field	Required or Optional	Values	Description
`CyclicPrefixUL`	Optional	`'Normal'` (default), `'Extended'`	Current cyclic prefix length

Parameter Field	Required or Optional	Values	Description
`Shortened`	Optional	0 (default), 1	Shorten subframe flag. If 1, the last symbol of the subframe is not used. It should be set if the current subframe contains a possible SRS transmission.

`chs` — Channel-specific transmission configuration

structure

Channel-specific transmission configuration, specified as a structure that can contain the following parameter fields.

Parameter Field	Required or Optional	Values	Description
`Modulation`	Required	`'QPSK'`, `'16QAM'`, `'64QAM'`, cell array of strings	Modulation type, specified as a string or cell array of strings. If 2 blocks, each cell is associated with a transport block.
`NLayers`	Optional	1 (default), 2, 3, 4	Total number of transmission layers associated with the transport block or blocks.
`PRBSet`	Required	1- or 2-column integer matrix	0-based physical resource block indices (PRBs) for the slots of the current PUSCH resource allocation. As a column vector, the resource allocation is the same in both slots of the subframe. As a two-column matrix, it specifies different PRBs for each slot in a subframe.
`RV`	Required	0, 1, 2, 3, 2-element integer vector	Redundancy version indicators, specified as a vector of 1 or 2 values. Each value can be an integer between 0 and 3.

Parameter Field	Required or Optional	Values	Description
The following three `'Init'` fields should be used if the UL-SCH encoding is for a retransmission of a previously sent transport block. If any of these fields are absent, its value is assumed to be the same as the value for its associated current subframe field.			
`InitPRBSet`	Optional	1- or 2-column integer matrix, `PRBSet` (default)	PRB indices used in the initial transmission PUSCH allocation. If this field is absent, its value is assumed to be the same as the value for the associated current subframe field, `PRBSet`.
`InitCyclicPrefixU`	Optional	`'Normal'`, `'Extended'`, `CyclicPrefixUL` (default)	Cyclic prefix length of initial transmit subframe. This is the length used during the first transmission of this transport block. If this field is absent, its value is assumed to be the same as the value for the associated current subframe field, `CyclicPrefixUL`.
`InitShortened`	Optional	0, 1, `Shortened` (default)	Initial transmit subframe shortened flag. If 1, the initial transmit subframe was shortened for possible SRS. If this field is absent, its value is assumed to be the same as the value for the associated current subframe field, `Shortened`.
The coding of the UCI can be controlled through the following additional fields.			
`BetaCQI`	Optional	numeric scalar, 2.0 (default)	Modulation and coding scheme (MCS) offset for CQI and PMI bits
`BetaRI`	Optional	numeric scalar, 2.0 (default)	Modulation and coding scheme (MCS) offset for RI bits

Parameter Field	Required or Optional	Values	Description
`BetaACK`	Optional	numeric scalar, 2.0 (default)	Modulation and coding scheme (MCS) offset for HARQ-ACK bits. This field was previously named `BetaHI`; if this field is absent but `BetaHI` is present, it is used as before.

`blklen` — Length of transmitted transport blocks

numeric vector

Length of the transmitted transport blocks, specified as a one or two element numeric vector.

Data Types: `double`

`opts` — Format options

string | cell array of strings

Format options of `chinfo` output, specified as a string or a cell array of strings.

Option	Values	Description
Channel parameter merging	`'nochsconcat'` (default)	Do not concatenate `chs` input structure into `chinfo`.
	`'chsconcat'`	Concatenate `chs` input structure into `chinfo`.
Output structure format	`'cwseparate'` (default)	Information values for each codeword are output in separate elements of the 1-by-ncodewords structure array `chinfo`.
	`'cwcombined'`	Information values for each codeword are combined into the fields of a single scalar, or 1-by-1, structure.

Data Types: `char | cell`

`ocqi` — Number of uncoded CQI bits

numeric scalar

Number of uncoded CQI bits, specified as a numeric scalar.

Data Types: `double`

`ori` — Number of uncoded RI bits

numeric scalar

Number of uncoded RI bits, specified as a numeric scalar.

Data Types: `double`

`oack` — Number of uncoded HARQ-ACK bits

numeric scalar

Number of uncoded HARQ-ACK bits, specified as a numeric scalar.

Data Types: `double`

Output Arguments

`info` — UL-SCH information

structure | structure array

UL-SCH information, returned as a structure or a structure array. If two transport blocks are encoded, info is a structure array of two elements, one for each block. , It contains the following parameter fields.

Parameter Field	Description	Values	Data Type
`C`	Total number of code blocks	nonnegative scalar integer	`int32`
`Km`	Lower code block size (K–)	nonnegative scalar integer	`int32`
`Cm`	Number of code blocks of size `Km` (C–)	nonnegative scalar integer	`int32`
`Kp`	Upper code block size (K+)	nonnegative scalar integer	`int32`
`Cp`	Number of code blocks of size `Kp` (C+)	nonnegative scalar integer	`int32`
`F`	Number of filler bits in first block	nonnegative scalar integer	`int32`

Parameter Field	Description	Values	Data Type
`L`	Number of segment cyclic redundancy check (CRC) bits	nonnegative scalar integer	`int32`
`Bout`	Total number of bits in all segments	nonnegative scalar integer	`int32`
`G`	Number of coded and rate matched UL-SCH data bits	nonnegative scalar integer	`int32`
`Qm`	Number of bits per symbol	nonnegative scalar integer	`int32`
`Gd`	Number of coded and rate matched UL-SCH data symbols (G')	nonnegative scalar integer	`int32`
`OCQI`	Number of uncoded channel quality information (CQI) bits	nonnegative scalar integer	`int32`
`ORI`	Number of uncoded symbols for RI	nonnegative scalar integer	`int32`
`OACK`	Number of uncoded symbols for ACK/NACK	nonnegative scalar integer	`int32`
`QdCQI`	Number of coded symbols for CQI (Q'_CQI)	nonnegative scalar integer	`int32`
`QdRI`	Number of coded symbols for RI (Q'_RI)	nonnegative scalar integer	`int32`
`QdACK`	Number of coded symbols for ACK/NACK (Q'_ACK)	nonnegative scalar integer	`int32`
`NRE`	Number of resource elements (REs) used for PUSCH transmission	nonnegative scalar integer	`int32`
`NLayers`	Number of layers associated with one codeword	nonnegative scalar integer	`int32`
`Modulation`	Modulation scheme associated with one codeword	`'QPSK'`, `'16QAM'`, `'64QAM'`	`char`
`RV`	RV value associated with one codeword	scalar integer	`int32`

References

[1] 3GPP TS 36.212. “Multiplexing and channel coding.” *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`lteACKEncode` | `lteCQIEncode` | `ltePUSCHDecode` | `lteRIEncode` | `lteULSCH` | `lteULSCHDecode` | `lteULSCHInterleave`

lteULSCHInterleave

UL-SCH interleaving

Syntax

```
out = lteULSCHInterleave(ue,chs,cdata)
out = lteULSCHInterleave(ue,chs,cdata,ccqi,cri,cack)
```

Description

`out = lteULSCHInterleave(ue,chs,cdata)` performs the UL-SCH channel interleaving on input cdata containing encoded transport channel (TrCH) data without UCI. It performs the UL-SCH data and UCI multiplexing and interleaving as defined in sections 5.2.2.7 and 5.2.2.8 of [1]. This input can be a vector or a cell array of vectors, interleaved separately, and the output is of the same form.

Multiple codewords can be parameterized by two different forms of the chs structure. Each codeword can be defined by separate elements of a 1-by-2 structure array, or the codeword parameters can be combined together in the fields of a single scalar, or 1-by-1, structure. Any scalar field values apply to both codewords and a scalar `NLayers` is the total number. See “UL-SCH Parameterization” for more details.

`out = lteULSCHInterleave(ue,chs,cdata,ccqi,cri,cack)` is as above except it also supports UL-SCH channel interleaving on both cdata and encoded UCI in ccqi, cri and cack. If any of these inputs are cell arrays, the output has the same form and any vector inputs are interleaved into the first cell of the output only. Any of the input cells or arrays can be empty if the associated input is not transmitted on one or more codewords.

Examples

Show Interleaving Order

Show the interleaving order for one resource block, which contains 288 PUSCH REs, with QPSK.

```
chs = struct('Modulation','QPSK');
```

```
interleave = lteULSCHInterleave(struct(),chs,(1:288).',[],[],[]);
interleave(1:10)
```

```
     1
     2
    25
    26
    49
    50
    73
    74
    97
    98
```

Input Arguments

ue — UE-specific settings
structure

UE-specific settings, specified as a structure with the following fields.

CyclicPrefixUL — Cyclic prefix length
'Normal' (default) | Optional | 'Extended'

Cyclic prefix length, specified as a string. Optional.

Data Types: char

Shortened — Shorten subframe
0 (default) | Optional | 1

Shorten subframe, specified as 0 or 1. Optional. If 1, the last symbol of the subframe is not used. It should be set if the current subframe contains a possible SRS transmission.

Data Types: logical

Data Types: struct

chs — UL-SCH related parameters
scalar structure

UL-SCH related parameters, specified as a scalar structure with the following fields.

`Modulation` — Modulation format string

`'QPSK' | '16QAM' | '64QAM'`

Modulation format string, specified as a string .

Data Types: `char`

`NLayers` — Number of transmission layers (total or per codeword)

1 (default) | Optional | 2 | 3 | 4

Number of transmission layers (total or per codeword), specified as a positive scalar integer. Optional.

Data Types: `double`

Data Types: `struct`

`cdata` — Encoded TrCH data

column vector | cell array of column vectors

Encoded TrCH data, specified as a column vector or a cell array of column vectors.

Data Types: `double | cell`

`ccqi` — Encoded CQI

vector

Encoded CQI, specified as a vector.

Data Types: `double`

`cri` — Encoded RI

vector

Encoded RI, specified as a vector.

Data Types: `double`

`cack` — Encoded ACK

vector

Encoded ACK, specified as a vector.

Data Types: `double`

Output Arguments

`out` — Interleaved UL-SCH output

numeric column vector | cell array of numeric column vectors

Interleaved UL-SCH output, returned as a numeric column vector or a cell array of numeric column vectors.

Data Types: `double` | `cell`

References

[1] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA)*. URL: http://www.3gpp.org.

See Also

`lteACKEncode` | `lteCQIEncode` | `lteRateMatchTurbo` | `lteRIEncode` | `lteULSCH` | `lteULSCHDeinterleave` | `lteULSCHInfo`

lteULScramble

PUSCH scrambling

Syntax

```
out = lteULScramble(in,nsubframe,cellid,rnti)
out = lteULScramble(ue,in)
```

Description

`out = lteULScramble(in,nsubframe,cellid,rnti)` performs PUSCH scrambling of bit vector, in, for subframe number, nsubframe, cell identity, cellid, and specified RNTI, rnti. It performs PUSCH scrambling according to section 5.3.1 of [1]. Placeholder bits, denoted by *x*, are represented by –1 in the input vector or cell array of vectors. Repetition placeholder bits, *y*, are represented by –2. This function substitutes these placeholders as part of its scrambling operation.

in is a vector or a cell array containing one or two vectors corresponding to the number of codewords to be scrambled.

`out = lteULScramble(ue,in)` performs PUSCH scrambling of the in according to UE-specific settings in structure, ue.

Examples

Perform PUSCH Scrambling

```
in = ones(10,1);
bits = lteULScramble(struct('NCellID',100,'NSubframe',0,'RNTI',61),in)
```

```
     0
     1
     0
     0
     0
     1
```

```
0
0
1
1
```

Input Arguments

`in` — Bit input data

numeric column vector | cell array of numeric column vectors

Bit input data, specified as a numeric column vector or cell array of numeric column vectors. This argument contains one or two vectors corresponding to the number of codewords to be scrambled.

Data Types: `double | cell`
Complex Number Support: Yes

`nsubframe` — Subframe number

numeric scalar

Subframe number, specified as a numeric scalar.

Data Types: `double`

`cellid` — Physical layer cell identity

numeric scalar

Physical layer cell identity, specified as a numeric scalar.

Data Types: `double`

`rnti` — Radio Network Temporary Identifier (16-bit)

numeric scalar

Radio Network Temporary Identifier (16-bit). specified as a numeric scalar.

Data Types: `double`

`ue` — UE-specific settings

structure

UE-specific settings, specified as a structure with the following fields.

`NCellID` — Physical layer cell identity
numeric scalar

Physical layer cell identity, specified as a numeric scalar.

Data Types: `double`

`NSubframe` — Subframe number
numeric scalar

Subframe number, specified as a numeric scalar.

Data Types: `double`

`RNTI` — Radio Network Temporary Identifier (16-bit)
numeric scalar

Radio Network Temporary Identifier (16-bit). specified as a numeric scalar.

Data Types: `double`

Data Types: `struct`

Output Arguments

`out` — PUSCH scrambled output bits
Numeric column vector | Cell array of numeric column vectors

PUSCH scrambled output bits, returned as a numeric column vector or a cell array of numeric column vectors.

References

[1] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

See Also
`ltePUSCH` | `lteSymbolModulate` | `lteULDescramble`

lteWarning

Enable and disable warnings for LTE System Toolbox product

Syntax

```
lteWarning('OFF',msgname)
lteWarning('ON',msgname)
```

Description

`lteWarning('OFF',msgname)` and `lteWarning('ON',msgname)` disable and enable the display of any warning messages of type msgname, specific to the LTE System Toolbox product.

Examples

Disable and Enable Warnings

Disable and enable warnings about using default values for optional parameters.

Turn off the warning about default values.

```
lteWarning('off','DefaultValue')
bch = lteBCH(struct('CellRefP',1),ones(24,1));
bch(1:4)
```

```
     1
     1
     1
     0
```

Notice that the warning about default values is not displayed.

Next, turn on the warning about default values again.

```
lteWarning('on','DefaultValue')
bch = lteBCH(struct('CellRefP',1),ones(24,1));
```

```
bch(1:4)

Warning: Using default value for parameter field CyclicPrefix (Normal)
    1
    1
    1
    0
```

Notice that the warning about default values is displayed again.

Input Arguments

msgname — Message name

`'defaultValue' | 'deprecated' | 'obsoleteFunction'`

Message name, specified as a string. The following strings are valid values for msgname.

Option	Warning Message
`'defaultValue'`	This warning occurs when an optional parameter value or parameter structure field is not defined and the default value is used instead.
`'deprecated'`	This warning occurs for deprecated functionality of the LTE System Toolbox product. For example, it occurs when the user calls a function using a deprecated syntax option. This warning indicates that the functionality may be removed in a later release.
`'obsoleteFunction'`	This warning occurs when a function from the obsolete LTE Toolbox interface is called. This interface is provided for backwards compatibility and may be removed in a later release.

Data Types: `char`

See Also

`warning`

rmlteobsolete

Remove obsolete LTE Toolbox interface from search path

Syntax

```
rmlteobsolete
```

Description

`rmlteobsolete` removes the `matlabroot\toolbox\lte\lteobsolete` directory from the MATLAB path, which disables the obsolete LTE Toolbox interface. This interface is provided for backwards compatibility and will be removed in a later release.

Note: You can undo your changes by calling the `addlteobsolete` function.

Examples

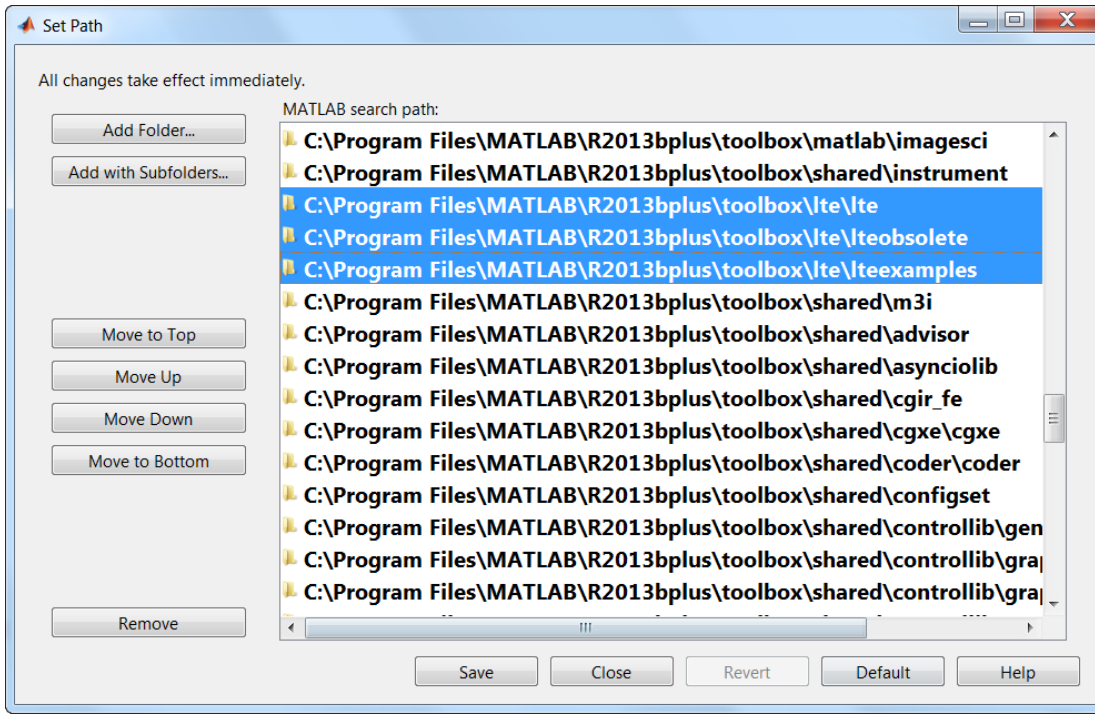
Remove Obsolete LTE Toolbox Interface from Search Path

Remove the directory associated with the obsolete LTE Toolbox interface from the MATLAB path.

View the current MATLAB path by calling the `pathtool` command.

```
pathtool
```

The Set Path dialog box appears. Scroll down through the listings to find the LTE System Toolbox product directories, as shown in the following figure.



If you do not see the listing `matlabroot\toolbox\lte\lteobsolete`, you do not need to run the `rmlteobsolete` function. Exit this example. Otherwise, click **Close** to close the Set Path dialog box.

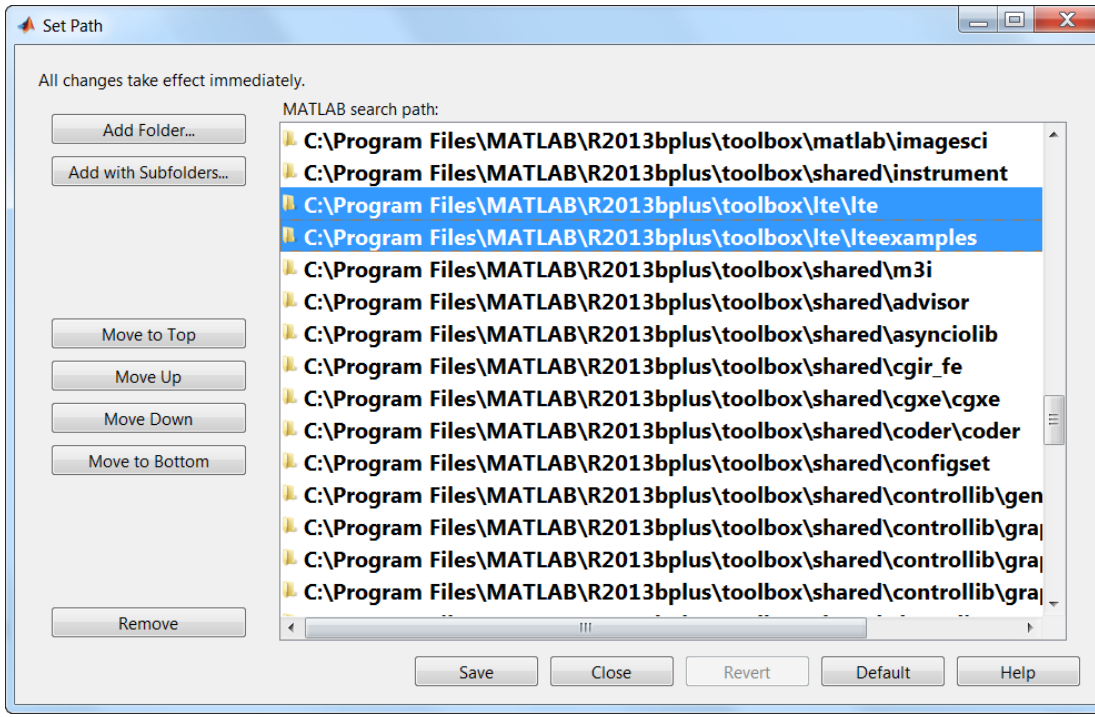
Remove the obsolete LTE Toolbox interface from the path.

```
rmlteobsolete
```

To confirm the changes, call the `pathtool` command again.

```
pathtool
```

The Set Path dialog box appears. Scroll down through the listings to find the LTE System Toolbox product directories, as shown in the following figure.



The absence of the listing `matlabroot\toolbox\lte\lteobsolete` shows that you have successfully remove the obsolete LTE Toolbox interface from the search path. Click **Close** again to close the Set Path dialog box.

See Also

`addlteobsolete`

2

Other Reference Pages

Parameter Field Names

Find names of fields to populate parameter structures

List of Structure Field Names

An alphabetical list of all field names used within the LTE System Toolbox product functions, their descriptions, their valid values, default values, and in which functions they are used is shown in the following table.

Field name	Used by Functions	Values	Description
`Alpha`	`lteSRS`		Reference signal cyclic shift (*alpha*)
	`lteSRS`, `ltePUCCH1`, `ltePUCCH1DRS`, `ltePUCCH2`, `ltePUCCH2DRS`		A row vector, containing the reference signal cyclic shift (*alpha*) for each OFDM symbol
	`ltePUSCHDRS`		A two-column row vector, containing the reference signal cyclic shift (*alpha*) for each slot
`BaseFreq`			Base (cell-specific) frequency domain starting position (*k_0 bar*), from which this UE-specific SRS is offset as a function of the UE-specific SRS Bandwidth value, *B_SRS*. UE-specific SRS configuration cannot result in a frequency domain starting position (*k_0*) lower than this, given the cell-specific SRS bandwidth configuration value, *C_SRS*.
`BaseOffset`			Base timing offset, in microseconds, for detection test in TS 36.104 (duration of *N_CS*/2)

Field name	Used by Functions	Values	Description
`BetaACK`		numeric scalar, 2.0 (default)	Modulation and coding scheme (MCS) offset for HARQ-ACK bits. This field was previously named `BetaHI`; if this field is absent but `BetaHI` is present, it is used as before.
`BetaCQI`		numeric scalar, 2.0 (default)	Modulation and coding scheme (MCS) offset for CQI and PMI bits
`BetaHI`			Modulation and Coding Scheme (MCS) offset for HARQ-ACK bits. **Note:** The field name `BetaHI` is deprecated and should be replaced with `BetaACK`.
`BetaRI`		numeric scalar, 2.0 (default)	Modulation and coding scheme (MCS) offset for RI bits
`BLKCRC`			Type-24A transport block CRC decoding error
`Bout`			Total number of bits in all segments
`BW`	`lteSRS`, `lteSRSIndices`, `lteRMCUL`, `lteRMCULTool`	0…3	UE-specific SRS Bandwidth value (*B_SRS*)
	`lteTestModelTool`, `lteTestModel`		Channel bandwidth type string when generating test model waveforms
`BWConfig`		0…7	Cell-specific SRS Bandwidth Configuration value (*C_SRS*)
`C`			Total number of code blocks
`CarrierFreq`			Carrier frequency, in hertz

Field name	Used by Functions	Values	Description
`CBSBuffers`			Cell array of vectors representing the LLR soft buffer states for the set of code blocks associated with a single transport block. The buffers are positioned at the input to the turbo decoder, after explicit rate recovery.
`CBSCRC`			Array of type-24B code block set CRC decoding results
`CellOffset`		0…9	Cell-specific sounding reference signal (SRS) offsets
`CellPeriod`		1, 2, 5, 10	Cell-specific sounding reference signal (SRS) periodicity, in milliseconds
`CellRefP`	`lteBCH`	1, 2, 4	Number of cell-specific reference signal (CRS) antenna ports
`CellRS`		`'Off'`, `'OmitEdgeRBs'`, `'On'`	Cell-specific reference signal (CRS) correlation mode
`CellRSPower`			Cell-specific reference symbol power adjustment, in dB
`CFI`		1, 2, 3	Control format indicator (CFI) value
`ChannelFilterDelay`			The implementation delay of the internal channel filtering
`Cm`			Number of code blocks of size `Km` (*C*–)
`CodebookIdx`		0…15	Codebook index used during precoding

Field name	Used by Functions	Values	Description
`CodedTrBlkSizes`			Coded transport block sizes for one or two codewords. This parameter field is only for informational purposes.
`ConfigIdx`	`lteSRS`, `lteSRSIndices`, `lteRMCUL`, `lteRMCULTool`	0...636	Configuration index for UE-specific periodicity and subframe offset
	`ltePRACH`, `ltePRACHInfo`	0...63	PRACH Configuration Index (*prach-ConfigurationIndex*)
`Cp`			Number of code blocks of size `Kp` (*C+*)
`CSI`		`'Off'`, `'On'`	CSI weighting for soft bits flag. If on, soft bits are weighted by CSI.
`CSIRefP`		1, 2, 4, 8	Number of CSI-RS antenna ports
`CSIRSConfig`			CSI-RS configuration index. See table 6.10.5.2-1 in TS 36.211.
`CSIRSPeriod`		`'On'`, `'Off'`, `Icsi-rs`, `[Tcsi-rs Dcsi-rs]`	CSI-RS subframe configuration
`CyclicOffset`			For High Speed mode, cyclic shift or shifts corresponding to a Doppler Shift of 1/*T_SEQ* (*d_u*)
`CyclicPrefix`	Downlink functions (`lteBCH`, `lteBCHDecode`))	`'Normal'` (default), `'Extended'`	Cyclic prefix length
	Uplink functions	`'Normal'` (default), `'Extended'`	Cyclic prefix length in the downlink

Field name	Used by Functions	Values	Description
`CyclicPrefixUL`	Uplink functions	`'Normal'` (default), `'Extended'`	Current cyclic prefix length
`CyclicShift`	`ltePUSCHDRS`, `lteRMCUL`, `lteRMCULTool`, `lteULFrameOffset`	0...7	Number of cyclic shifts used for PUSCH DRS (yields *n1_DMRS*)
	`ltePRACH`, `ltePRACHInfo`		Cyclic shift or shifts of Zadoff-Chu sequence (*C_v*)
	`lteSRS`, `lteSRSIndices`, `lteRMCUL`, `lteRMCULTool`	0...7	UE-specific cyclic shift (*n_SRS^cs*)
`CyclicShiftIdx`		0...15	Cyclic shift configuration index (*zeroCorrelationZoneConfig*, yields *N_CS*)
`CyclicShifts`		0...7	Number of cyclic shifts used for Format 1 in resource blocks (RBs) with a mixture of Format 1 and Format 2 PUCCH (*N1cs*)
`DCIFormat`			Downlink control information (DCI) format type string
`DelayProfile`		`'EPA'`, `'EVA'`, `'ETU'`	Delay profile model
`DeltaOffset`		0, 1, 2	Warning: The use of this parameter field is deprecated. This parameter may be removed in a future release. (*delta_offset*)
`DeltaShift`		1, 2, 3	(*delta_shift*)
`Dmin`			eNodeB to railway track distance, in meters

Field name	Used by Functions	Values	Description
`DopplerFreq`			*Doppler* frequency, in hertz
`Ds`			`Ds`/2 is initial distance between train and eNodeB, in meters
`DuplexMode`		`'FDD'` (default), `'TDD'`	Duplex mode
`DynCyclicShift`		0…7	Cyclic shift for DMRS (yields *n2_DMRS*)
`EV`			Normalized error vector
`F`			Number of filler bits in first block
`Fields`			A 1-by-4 vector of PRACH field lengths, [*OFFSET T_CP T_SEQ GUARD*], where *T_CP* and *T_SEQ* are the length of cyclic prefix and PRACH sequence in fundamental time periods (*T_s*), *OFFSET* is the number of fundamental time periods from the start of configured subframe to the start of the cyclic prefix (non-zero only for TDD special subframes), and *GUARD* is the number of fundamental time periods from the end of the PRACH sequence to the end of the number of subframes spanned by the PRACH
`Format`		0…4	Preamble format
`FreqIdx`	`ltePRACH`, `ltePRACHDetect`, `ltePRACHInfo`	0…5	Frequency resource index (*f_RA*). Only required for `'TDD'` duplexing mode.

Field name	Used by Functions	Values	Description
	`lteSRSIndices`	0...*B_SRS*	A vector specifying the frequency position index (*n_b*) for each *b*
`FreqOffset`		0...94	PRACH frequency offset (*n_PRBoffset*). Only required for Preamble format 0–3.
`FreqPosition`		0...23	Frequency domain position (*n_RRC*)
`FreqStart`			Frequency domain starting position (*k_0*), the 0-based subcarrier index of the lowest SRS subcarrier
`FreqWindow`			Size of window in resource elements used to average over frequency during channel estimation
`G`	`ltePDSCHIndices`		A one- or two-element vector, specifying the number of coded and rate matched DL-SCH data bits for each codeword
	`ltePUSCHIndices`, `lteULSCH`, `lteULSCHInfo`		A one- or two-element vector, specifying the number of coded and rate matched UL-SCH data bits for each codeword
`Gd`	`ltePDSCHIndices`		Number of coded and rate matched DL-SCH data symbols, equal to the number of rows in the PDSCH indices
	`ltePUSCHIndices`		Number of coded and rate matched UL-SCH data symbols, equal to the number of rows in the PUSCH indices
	`lteULSCH`, `lteULSCHInfo`		Number of coded and rate matched UL-SCH data symbols

Field name	Used by Functions	Values	Description
`HighSpeed`		0...1	High Speed flag (*highSpeedFlag*). A value of 1 signifies a restricted set. A value of 0 signifies an unrestricted set.
`Hopping`	`ltePUCCH1`, `ltePUCCH2`, `ltePUCCH3`	`'Off'`, `'Group'`	Frequency hopping method
	`ltePUSCH`	`'Off'`, `'Group'`, `'Sequence'`	Frequency hopping method
`HoppingBW`		0...3	SRS Frequency hopping configuration index (*b_hop*)
`HoppingOffset`			A vector specifying the offset term due to frequency hopping (*F_b*), used in calculation of *n_b*
`InitCyclicPrefixUL`		`'Normal'`, `'Extended'`, `CyclicPrefixUL` (default)	Cyclic prefix length of initial transmit subframe. This is the length used during the first transmission of this transport block. If this field is absent, its value is assumed to be the same as the value for the associated current subframe field, `CyclicPrefixUL`.
`InitPRBSet`		1- or 2-column integer matrix, `PRBSet` (default)	PRB indices used in the initial transmission PUSCH allocation. If this field is absent, its value is assumed to be the same as the value for the associated current subframe field, `PRBSet`.

<table>
<tr><th>Field name</th><th>Used by Functions</th><th>Values</th><th>Description</th></tr>
<tr><td>InitShortened</td><td></td><td>0, 1, Shortened (default)</td><td>Initial transmit subframe shortened flag. If 1, the initial transmit subframe was shortened for possible SRS. If this field is absent, its value is assumed to be the same as the value for the associated current subframe field, Shortened.</td></tr>
<tr><td rowspan="3">InitTime</td><td>lteFadingChannel</td><td></td><td>Fading process time offset, in seconds</td></tr>
<tr><td>lteHSTChannel</td><td></td><td>Doppler shift timing offset, in seconds</td></tr>
<tr><td>lteMovingChannel</td><td></td><td>Fading process and timing adjustment offset, in seconds</td></tr>
<tr><td>InterpType</td><td>lteDLChannelEstima
lteULChannelEstima
lteULChannelEstima
lteULChannelEstima
lteULChannelEstima</td><td>'nearest',
'linear',
'natural',
'cubic', 'v4'</td><td>Type of 2-D interpolation used during interpolation. For details, see griddata. Supported choices are shown in the following table.
<table>
<tr><th>String</th><th>Description</th></tr>
<tr><td>'neares</td><td>Nearest neighbor interpolation</td></tr>
<tr><td>'linear</td><td>Linear interpolation</td></tr>
<tr><td>'natura</td><td>Natural neighbor interpolation</td></tr>
<tr><td>'cubic'</td><td>Cubic interpolation</td></tr>
<tr><td>'v4'</td><td>MATLAB 4 griddata method</td></tr>
</table></td></tr>
</table>

Field name	Used by Functions	Values	Description
`InterpWindow`		`'Causal'`, `'Non-causal'`, `'Centred'`	Interpolation window type used during channel estimation. Options `'Centred'` and `'Centered'` are equivalent.
`InterpWinSize`			Window size across which to interpolate. The interpolation window size is specified in number of subframes.
`IsSRSSubframe`			This field is present only if the UE contains `NSubframe`. Value is 1 if `NSubframe` satisfies the equation: mod(`NSubframe`, `CellPeriod`)=`CellOffset` Value is 0 otherwise.
`k`			Subband size, in resource blocks (equal to `NRB` for wideband PMI reporting or transmission schemes without PMI reporting).
`K`			Ratio of uplink data to PRACH subcarrier spacing (*K*)
`Km`			Lower code block size (*K*–)
`Kp`			Upper code block size (*K*+)
`KTxComb`			Offset to the frequency domain starting position (*k_TC*), a function of the transmission comb parameter
`L`			Number of segment cyclic redundancy check (CRC) bits
`MacTxNumber`		0…27	Number of the current MAC (re-)transmission, *CURRENT_TX_NB*

Field name	Used by Functions	Values	Description
`MaxPMI`	`ltePMIInfo`		Maximum permitted PMI value for the given configuration. Valid PMI values range from 0 to `MaxPMI`. For CSI reporting, when `CSIRefP = 8`, `MaxPMI` is a 2–element vector, indicating the maximum permissible values of *i1* and *i2*, the first and second codebook indices. For transmission schemes without PMI reporting, `MaxPMI = 0`.
	`lteULPMIInfo`		Indicates the maximum permitted PMI value for the given configuration. Valid PMI values range from 0 to `MaxPMI`.
`MIMOCorrelation`		`'Low'`, `'Medium'`, `'High'`	Correlation between UE and eNodeB antennas
`ModelType`	`lteFadingChannel`	`'Dent'`, `'GMEDS'`	*Rayleigh* fading model type
`Modulation`	`lteACKDecode`, `lteACKEncode`	`'QPSK'`, `'16QAM'`, `'64QAM'`, cell array of strings	Modulation type, specified as a string or cell array of strings. If 2 blocks, each cell is associated with a transport block.
		`'QPSK'`, `'16QAM'`, `'64QAM'`	Modulation scheme associated with one codeword
`MovingScenario`		`'Scenario1'`, `'Scenario2'`	Moving channel scenario

Field name	Used by Functions	Values	Description
`MTot`			Total number of bits associated with PDCCHs (8×`NRE`)
`NAllocatedPDCCHREG`			Number of allocated PDCCH REGs as per Test Model number and BW
`N1DMRS`			Component of the reference signal cyclic shift, signaled from higher layers (*n1_DMRS*)
`N2DMRS`			Component of the reference signal cyclic shift, signaled from the most recent DCI format 0 message (*n2_DMRS*)
`NBundled`	`lteACKDecode`, `lteACKEncode`	0 (default), 1, ..., 9	TDD HARQ-ACK bundling scrambling sequence index. When set to 0, the function disables the TDD HARQ-ACK bundling scrambling. Therefore, it is off by default.
`NCCE`			Number of control channel elements available for actual PDCCH usage
`NCellCyclicShift`			A row vector, containing the cell-specific cyclic shift (*ncell_cs*) for each OFDM symbol
`NCellID`			Physical layer cell identity
`NCodewords`		1, 2	Number of codewords
`NCS`			Length of zero correlation zone, plus 1 (*N_CS*)

Field name	Used by Functions	Values	Description
`NDLRB`		Scalar integer (6, ..., 110). Standard bandwidth values are 6, 15, 25, 50, 75, and 100.	Number of downlink (DL) resource blocks (RBs)
`NF4RachPreambles`		0...27	Number of RACH preamble frequency resources of Format 4 in *UpPTS*
`Nfft`			Number of fast *Fourier* transform (FFT) points
`NFrame`			Frame number
`Ng`		`'Sixth'`, `'Half'`, `'One'`, `'Two'`	HICH group multiplier
`Ngroups`			Number of PHICH groups
`NHARQProcesses`		1...8	Number of HARQ processes
`NIR`			Number of soft bits associated with transport block. Soft buffer size for entire input transport block
`NL`			Number of layers used in rate matching calculation
`NLayers`	`lteDLSCH`, `lteDLSCHDecode`, `lteULSCH`, `lteULSCHDecode`	1 (default), 2, 3, 4, 5, 6, 7, 8	Total number of transmission layers associated with the transport block or blocks.
	Downlink modulation	1,...,8, depending on `TxScheme`	Number of transmission layers (downlink modulation)
	Uplink modulation (`lteACKDecode`, `lteACKEncode`)	1 (default), 2, 3, 4	Number of transmission layers, total or per codeword

Field name	Used by Functions	Values	Description
NMappingUnits			Number of PHICH mapping units
NPHICH			Number of individual PHICH available
NPRS			A two-column row vector, containing the cell-specific component of the reference signal cyclic shift (*n_PRS*) for each slot
NPRSRB		0...NDLRB	Number of PRS physical resource blocks
NRE	ltePCFICHInfo		Number of resource elements (REs) assigned to PCFICH (4×NREG)
	ltePDCCHInfo		Total number of resource elements (REs) associated with PDCCHs (4×NREG)
	ltePHICHInfo		Number of resource elements (REs) assigned to all PHICH
	lteULSCH, lteULSCHInfo		Number of resource elements (REs) used for PUSCH transmission
NREG	ltePCFICHInfo		Number of resource element groups (REGs) assigned to PCFICH
	ltePDCCHInfo, ltePDCCHSpace		Total number of resource element groups (REGs) associated with PDCCHs (4×NRE)
	ltePHICHInfo		Number of resource element groups assigned to all PHICH
NREGUsed			Number of resource element groups (REGs) available for actual PDCCH usage

Field name	Used by Functions	Values	Description
`NResourceIdx`			A two-column row vector, containing the resource indices (n') for each slot
`NREUsed`			Number of resource elements (REs) available for actual PDCCH usage
`NRxAnts`		1 or more	Number of receive antennas
`NSCID`		0, 1	Scrambling code identity (ID)
`NSequences`			Number of orthogonal sequences in each PHICH group
`NSoftbits`		Nonnegative scalar integer (default 0)	Total number of soft buffer bits. The default setting of 0 signifies that there is no buffer limit.
`NSRSTx`			Number of UE-specific SRS transmissions (n_SRS)
`NSubbands`	`ltePMIInfo`, `lteULPMIInfo`		Number of subbands for PMI reporting (equal to 1 for wideband PMI reporting) or transmission schemes without PMI reporting.
`NSubframe`			Subframe number
`NSymbols`	`lteDuplexingInfo`		Total number of symbols in the subframe
	`ltePDCCHInfo`		Total number of OFDM symbols spanned by the PDCCH
`NSymbolsDL`			Number of symbols used for transmission in the downlink (DL)
`NSymbolsGuard`			Number of symbols in the guard period

Field name	Used by Functions	Values	Description
`NSymbolsUL`			Number of symbols used for transmission in the uplink (UL)
`NSymbSlot`			A vector indicating the number of OFDM symbols in each slot ([*N_SF*,*0_PUCCH N_SF*,*1_PUCCH*])
`NTerms`		power of 2	Number of oscillators used in fading path modeling
`NTurboDecIts`			Number of turbo decoder iteration cycles
`NTxAnts`	`lteDMRSIndices`, `ltePDSCHIndices`	0 or more	Number of transmission antenna ports. This argument is only present for UE-specific demodulation reference symbols.
	Uplink modulation	1. 2, 4	Number of transmission antennas
`NULRB`			Number of uplink (UL) resource blocks (RBs)
`NZC`	`ltePRACHDetect`, `ltePRACHInfo`		Zadoff-Chu sequence length (*N_ZC*)
	`ltePUSCHDRS`, `lteSRS`		Zadoff-Chu sequence length (*NRS_ZC*)
`OACK`	`lteACKDecode`	nonnegative scalar integer, 0 (default)	Number of uncoded HARQ-ACK bits.
			Number of uncoded symbols for ACK/NACK
`OCQI`	`lteCQIDecode`	nonnegative scalar integer, 0 (default)	Number of uncoded channel quality information (CQI) bits

Field name	Used by Functions	Values	Description
			Number of coded channel quality information (CQI) symbols in UL-SCH
`OCNG`		`'Disable'`, `'Enable'`	OFDMA channel noise generator
`OdACK`			Number of coded symbols for ACK/NACK (*Q'_ACK*)
`OdCQI`			Number of coded symbols for CQI (*Q'_CQI*)
`OdRI`			Number of coded symbols for RI (*Q'_RI*)
`OffsetIdx`		0...1	Choice of SRS Subframe Offset in the case of 2 ms SRS periodicity. This parameter indexes the two SRS Subframe Offset entries in the row specified by the `ConfigIdx` parameter in table 8.2-2 of TS 36.213 for the SRS Configuration Index.
`ORI`			Number of uncoded symbols for RI
			Number of uncoded RI bits
`OrthCover`		`'On'`, `'Off'`	Applies (`'On'`), or does not apply (`'Off'`), orthogonal cover sequence, *w* (*Activate-DMRS-with OCC*)
`OrthSeq`	`ltePUCCH1`, `ltePUCCH3`		A 4-by-2 matrix where each column contains the orthogonal sequence (*w_n_oc*) for each slot

Field name	Used by Functions	Values	Description
	`ltePUCCH1DRS`, `ltePUCCH2DRS`, `ltePUCCH3DRS`		A two-column matrix where each column contains the orthogonal sequence (*w bar*) for each slot
	`ltePUSCHDRS`		A vector containing the orthogonal cover value (*w*) for each slot
`OrthSeqIdx`	`ltePUCCH1`, `ltePUCCH3`, `ltePUCCH3DRS`		A two-column row vector, containing the orthogonal sequence index (*n_oc*) for each slot
	`ltePUCCH1DRS`		A two-column row vector, containing the orthogonal sequence index (*n_oc bar*) for each slot
`PBCHPower`			PBCH symbol power adjustment, in dB
`PCFICHPower`			PCFICH symbol power adjustment, in dB
`PDCCHFormat`		0, 1, 2, 3	PDCCH format
`PDCCHPower`			PDCCH symbol power adjustment, in dB
`PDSCH`			PDSCH transmission configuration structure
`PDSCHPowerBoosted`			PDSCH symbol power adjustment, in dB, for the boosted physical resource blocks (PRBs)
`PDSCHPowerDeboosted`			PDSCH symbol power adjustment, in dB, for the de-boosted physical resource blocks (PRBs)

Field name	Used by Functions	Values	Description
`Peak`			Peak error vector magnitude (EVM), the largest single EVM value calculated across all input values
`Phi`			Frequency domain location offset (phi)
`PHICHDuration`		`'Normal'`, `'Extended'`	PHICH duration
`PilotAverage`		`'TestEVM'`, `'UserDefined'`	Type of pilot averaging
`PMI`		0...23	Scalar precoder matrix indication (PMI) to be used during precoding
`PMIMode`		`'Wideband'`, `'Subband'`	PMI reporting mode
`PMISet`		Integer vector (0, ..., 15)	Precoder matrix indication (PMI) set. It can contain either a single value, corresponding to single PMI mode, or multiple values, corresponding to multiple or subband PMI mode. The number of values depends on the CellRefP, transmission layers and TxScheme.
`Port`			Antenna port number used for transmission (*p*)

Field name	Used by Functions	Values	Description
`PRBSet`	Downlink modulation	1- or 2-column integer matrix	0-based physical resource block (PRB) indices corresponding to the resource allocations for this PDSCH. As a column vector, the resource allocation is the same in both slots of the subframe. As a 2-column matrix, this parameter specifies different PRBs for each slot in a subframe.
	Uplink modulation	1- or 2-column integer matrix	0-based physical resource block indices (PRBs) for the slots of the current PUSCH resource allocation. As a column vector, the resource allocation is the same in both slots of the subframe. As a two-column matrix, it specifies different PRBs for each slot in a subframe.
	`ltePRACHInfo`		PRBs occupied by PRACH preamble (starts at *n_PRB*, 0-based)
	`lteSRSIndices`		A vector specifying the PRBs occupied by the indices in each slot of the subframe (0-based)
`PreambleIdx`		0...63	Scalar preamble index within cell (*ra-PreambleIndex*)
`PRSPeriod`		`'On'`, `'Off'`, `Iprs`, `[Tprs Dprs]`	Positioning reference signal (PRS) subframe configuration
`PSS`		`'Off'`, `'On'`	Primary synchronization signal (PSS) correlation mode

Field name	Used by Functions	Values	Description
`PSSPower`			Primary synchronization signal (PSS) symbol power adjustment, in dB
`PUSCH`			PUSCH transmission configuration structure
`PUSCHHopping`		`'Inter'`, `'InterAndIntra`	Uplink subframe hopping mode
`PUSCHHoppingOffset`		0...98	PUSCH hopping offset
`QdACK`	`lteACKEncode`	nonnegative scalar integer	Number of coded HARQ-ACK symbols for ACK or NACK (*Q'_ACK*)
`QdCQI`	`lteCQIEncode`	nonnegative scalar integer	Number of coded channel quality information (CQI) symbols (*Q'_CQI*)
`QdRI`			Number of coded symbols for RI (*Q'_RI*)
`Qm`			Bits per symbol variable used in rate matching calculation
			Number of bits per symbol
`RBIdx`			PUCCH logical resource block index (*m*)
`RC`			Reference measurement channel (RMC) number or type, as specified in TS36.101, Annex A.3
`Reference`	`lteDLChannelEstima`	`'DMRS'`, `'CSIRS'`	Specifies point of reference (signals to internally generate) for channel estimation
	`lteULChannelEstima` `lteULPMISelect`	`'Antennas'`, `'Layers'`, `'None'`	Specifies point of reference (signals to internally generate)

Field name	Used by Functions	Values	Description
`ResourceIdx`	`ltePUCCH1`, `ltePUCCH1Decode`, `ltePUCCH1DRS`, `ltePUCCH1DRSIndice` `ltePUCCH1Indices`, `lteULChannelEstima` `lteULFrameOffsetPU`	0…2047	A vector of PUCCH Resource Indices, one for each transmission antenna, which determine the cyclic shift and orthogonal cover used for transmission (*n1_pucch*)
	`ltePUCCH2`, `ltePUCCH2Decode`, `ltePUCCH2DRS`, `ltePUCCH2DRSDecode` `ltePUCCH2DRSIndice` `ltePUCCH2Indices`, `lteULChannelEstima` `lteULFrameOffsetPU`	0…1185	A vector of PUCCH Resource Indices, one for each transmission antenna, which determine the cyclic shift and orthogonal cover used for transmission (*n2_pucch*)
	`ltePUCCH3`, `ltePUCCH3Decode`, `ltePUCCH3DRS`, `ltePUCCH3DRSIndice` `ltePUCCH3Indices`, `lteULChannelEstima` `lteULFrameOffsetPU`	0…549	A vector of PUCCH Resource Indices, one for each transmission antenna, which determine the cyclic shift and orthogonal cover used for transmission (*n3_pucch*)
`ResourceSize`		0…63	Size of resource allocated to PUCCH Format 2 (*N2RB*)
`Rho`			PDSCH resource element power allocation, in dB
`RMS`			Root mean square (RMS) error vector magnitude (EVM), the square root of the mean square of the EVM across all input values
`RNTI`			Radio network temporary identifier (RNTI) value (16 bits)

Field name	Used by Functions	Values	Description
`RootSeq`	`ltePRACH`, `ltePRACHDetect`, `ltePRACHInfo`		Physical root Zadoff-Chu sequence index or indices (u)
	`lteSRS`		Root Zadoff-Chu sequence index (q)
	`ltePUSCHDRS`		A two-column row vector, containing the Root Zadoff Chu sequence index (q) for each slot
`RV`		0, 1, 2, 3, 2-element integer vector	Redundancy version indicators, specified as a vector of 1 or 2 values. Each value can be an integer between 0 and 3.
		0, 1, 2, 3	RV value associated with one codeword
`RVSeq`			Redundancy version (RV) sequence, applied to all subframes
`SamplingRate`	`lteOFDMModulate`, `lteOFDMInfo`, `lteSCFDMAModulate`, `lteSCFDMAInfo`		Sampling rate of the time-domain waveform
	`ltePRACH`, `ltePRACHDetect`, `ltePRACHInfo`		Sampling rate of the PRACH modulator
	`lteSRS`		Input signal sampling rate, the rate of each sample in the rows of the input matrix, *IN*
`ScrambSeq`			A two-column row vector, containing the scrambling value (S) for each slot

Field name	Used by Functions	Values	Description
`Seed`			Random number generator seed for channel models. Set to zero for a random seed.
`SeqGroup`	`lteSRS`	0…29	PUCCH sequence group number (u)
	`ltePUSCHDRS`, `lteRMCUL`, `lteRMCULTool`, `lteULFrameOffset`	0…29	PUSCH sequence group number (*delta_SS*)
	`ltePUCCH1`, `ltePUCCH1DRS`, `ltePUCCH2`, `ltePUCCH2DRS`, `ltePUCCH3DRS`, `ltePUSCHDRS`		A two-column row vector, containing the base sequence group number (u) for each slot
	`lteSRS`		Base sequence group number (u)
`SeqIdx`	`lteSRS`	0, 1	Base sequence number (v)
	`ltePUCCH1`, `ltePUCCH1DRS`, `ltePUCCH2`, `ltePUCCH2DRS`, `ltePUCCH3DRS`, `ltePUSCHDRS`		A two-column row vector, containing the base sequence number (v) for each slot
	`ltePRACH`, `ltePRACHDetect`, `ltePRACHInfo`	0…837	Logical root sequence index (*RACH_ROOT_SEQUENCE*)
`Shortened`		0 (default), 1	Shorten subframe flag. If 1, the last symbol of the subframe is not used. It should be set if the current subframe contains a possible SRS transmission.

Field name	Used by Functions	Values	Description
`SRS`		`'on'`	String to enable SRS related configuration parameters (set SRS to `'on'`) for RMCs which optionally support SRS, or a complete or part SRS structure. If absent, no SRS configuration is created.
`SSC`		0 (default), 1, 2, 3, 4, 5, 6, 7, 8, 9	Special subframe configuration (SSC)
`SSS`		`'Off'`, `'On'`	Secondary synchronization signal (SSS) correlation mode
`SSSPower`			Secondary synchronization signal (SSS) symbol power adjustment, in dB
`SubcarrierSpacing`			Subcarrier spacing of PRACH preamble, in Hz (*deltaf_RA*)
`SubframeConfig`		0...15	Sounding reference signal (SRS) subframe configuration
`SubframeType`		`'Downlink'`, `'Uplink'`, `'Special'`	Type of subframe
`Symbols`	`ltePUCCH1`		A row vector containing the modulated data symbols (*d(0)*) for each OFDM symbol
	`ltePUCCH1DRS`, `ltePUCCH2DRS`, `ltePUCCH3DRS`		A row vector containing the modulated data symbols (*z*) for each OFDM symbol
	`ltePUCCH2`, `ltePUCCH3`		A row vector containing the modulated data symbols (*d*) for each OFDM symbol
`TDDConfig`		0 (default), 1, 2, 3, 4, 5, 6	Uplink or downlink configuration

Field name	Used by Functions	Values	Description
`TimeWindow`			Size of window in resource elements used to average over time during channel estimation
`TimingOffset`			PRACH timing offset, in microseconds
`TMN`			Test model number, as specified in TS 36.141, a type string used when generating test model waveforms
`TotSubframes`	`lteDLPerfectChanne` `lteRMCDL`, `lteRMCDLTool`, `lteRMCUL`, `lteRMCULTool`, `lteULPerfectChanne`		Total number of subframes to generate
	`ltePRACHInfo`		The number of subframes duration of the PRACH. Each subframe lasts 30,720 fundamental periods; therefore, `TotSubframes` is ceil(sum(*Fields*)/30720), the number of subframes required to hold the entire PRACH waveform. The duration of the PRACH is a function of the Preamble Format as described in table 5.7.1-1 of TS 36.211.
`TrBlkSizes`			Transport block sizes for each subframe in a frame
`TxComb`		0, 1	Transmission comb. Controls SRS positions; SRS is transmitted in 6 carriers per resource block on odd (1) and even (0) resource indices.

Field name	Used by Functions	Values	Description
`TxScheme`		`'SpatialMux'` (default), `'Port0'`, `'TxDiversity'`, `'CDD'`, `'MultiUser'`, `'Port5'`, `'Port7-8'`, `'Port8'`, `'Port7-14'`	Transmission scheme, specified as one of the following options. • `'SpatialMux'` — Closed-loop spatial multiplexing. • `'Port0'` — Single-antenna port, port 0. • `'TxDiversity'` — Transmit diversity scheme. • `'CDD'` — Large delay CDD scheme. • `'MultiUser'` — Multiuser MIMO scheme. • `'Port5'` — Single-antenna port, port 5. • `'Port7-8'` — Single-antenna port, port 7 (**`NLayers`** = 1). Dual layer transmission, ports 7 and 8 (**`NLayers`** = 2). • `'Port8'` — Single-antenna port, port 8. • `'Port7-14'` — Up to 8-layer transmission, ports 7–14.
`UeOffset`		0…319	UE-specific SRS offset
`UePeriod`		2, 5, 10, 20, 40, 80, 160, 320	UE-specific SRS periodicity, in milliseconds
`Velocity`	`lteHSTChannel`		Train velocity, in kilometers per hour

Field name	Used by Functions	Values	Description
`W`	`lteDMRS`		Precoding matrix for the UE-specific beamforming of the DM-RS, of size `NLayers`-by-`NTxAnts`. An empty matrix, `[]`, signifies no precoding.
	`ltePDSCH`	Numeric matrix, `[]` (default)	Precoding matrix for the UE-specific beamforming of the PDSCH symbols, specified as a numeric matrix of size `NLayers`-by-`NTxAnts`. The default value is an empty matrix, `[]`, which signifies that there is no precoding.
`Window`		`'Left'`, `'Right'`, `'Centred'`, `'Centered'`	If more than one subframe is input this parameter is required to indicate the position of the subframe from *RXGRID* and *REFGRID* containing the desired channel estimate. Only channel estimates for this subframe will be returned. For the `'Centred'` and `'Centered'` settings, the window size must be odd.
`Windowing`	`lteOFDMModulate`, `lteOFDMInfo`, `lteRMCDL`, `lteTestModel`		Number of time-domain samples over which windowing and overlapping of OFDM symbols is applied
	`lteRMCUL`, `lteSCFDMAModulate`, `lteSCFDMAInfo`		The number of time-domain samples over which windowing and overlapping of SC-FDMA symbols is applied

References

[1] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.141. "Base Station (BS) conformance testing." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[3] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[4] 3GPP TS 36.213. "Physical layer procedures." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

Related Examples

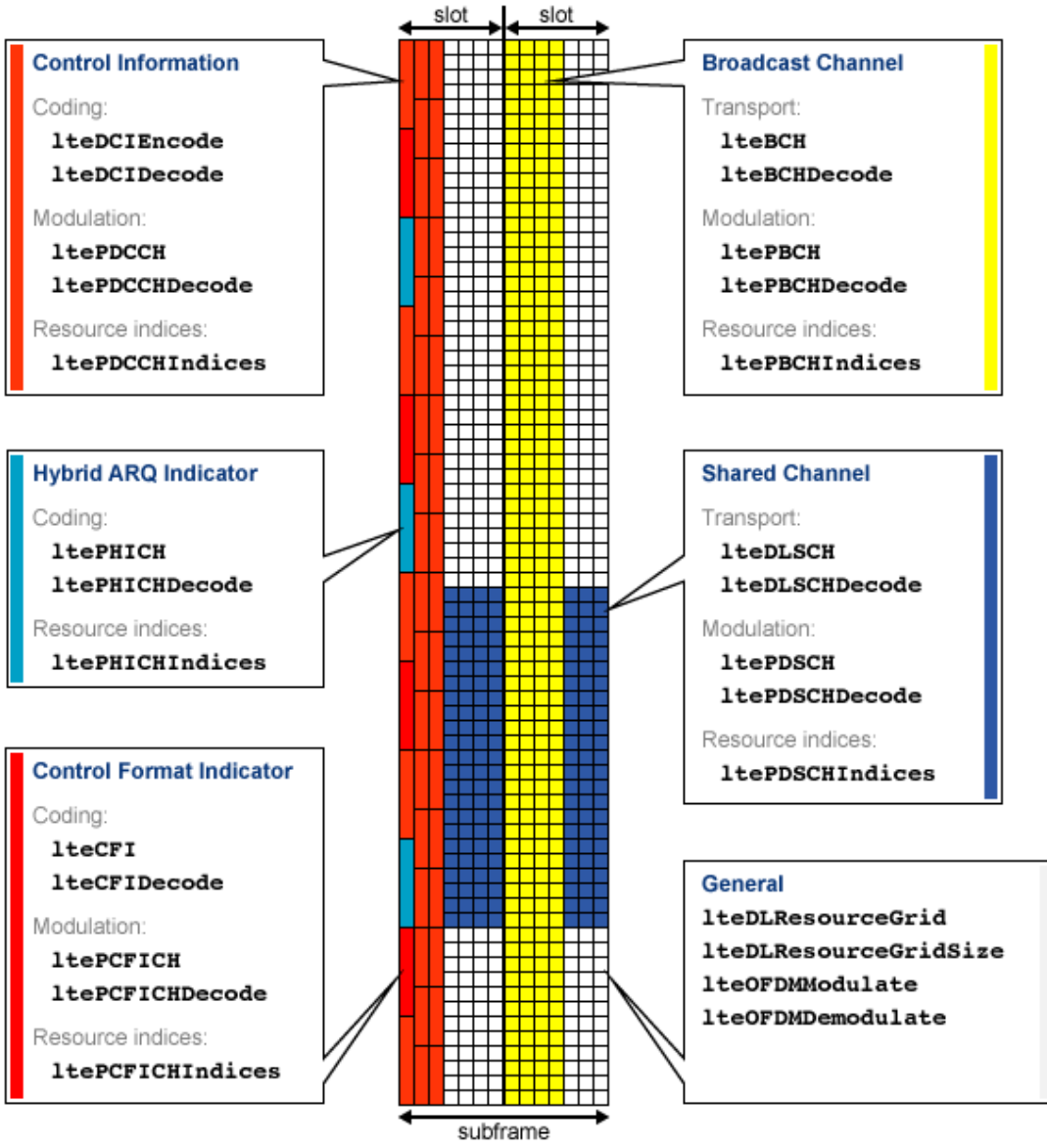
- "Parameterization"
- "UL-SCH Parameterization"

3

Resource Grid and Block Diagrams

Downlink Physical Channels Grid

The downlink physical channels, their associated functions, and their locations on the resource grid are shown in the following figure.



- Control Information

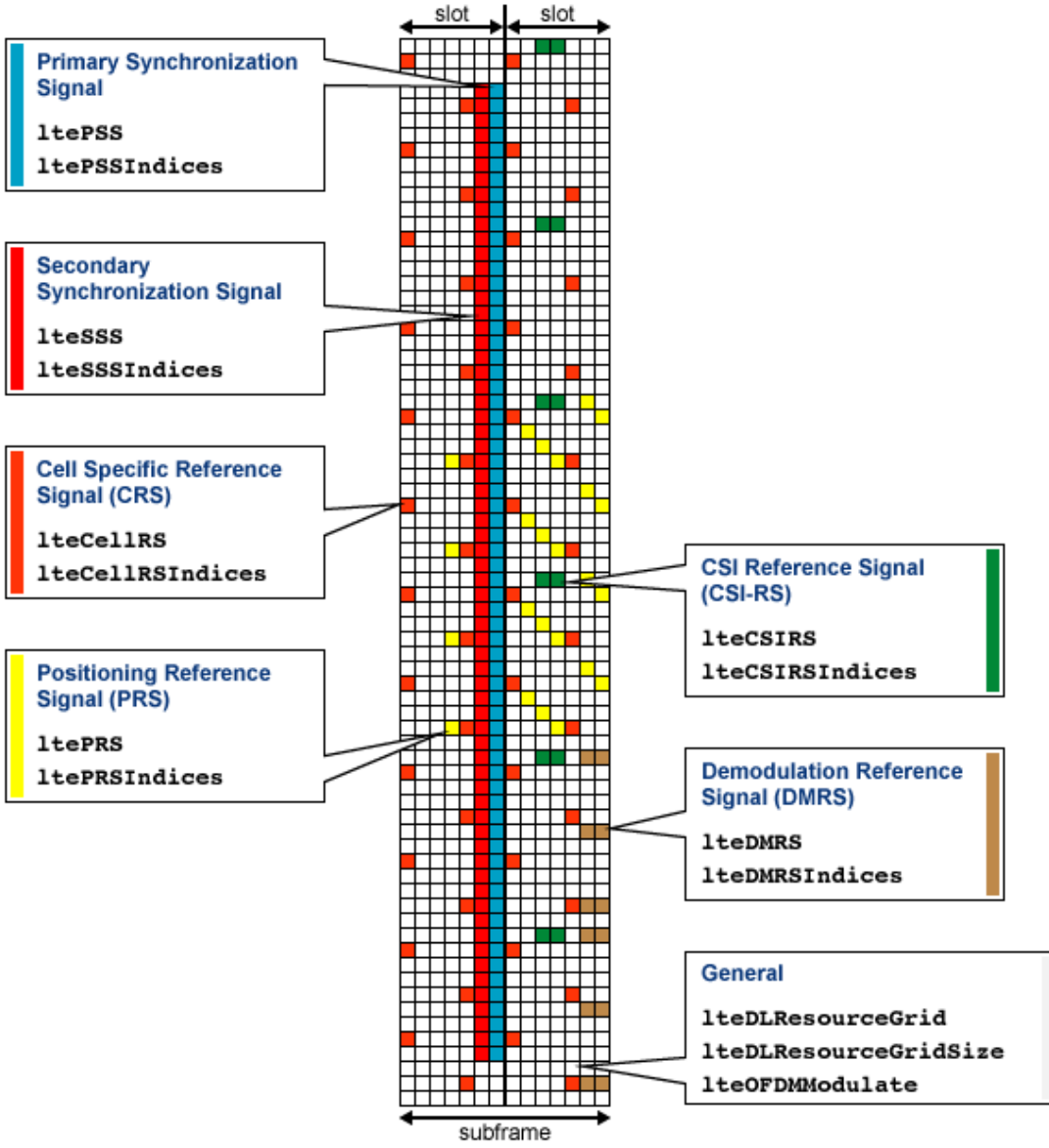
- Coding
  - `lteDCIEncode`
  - `lteDCIDecode`
- Modulation
  - `ltePDCCH`
  - `ltePDCCHDecode`
- Resource Indices — `ltePDCCHIndices`

- Hybrid ARQ Indicator
  - Coding
    - `ltePHICH`
    - `ltePHICHDecode`
  - Resource Indices — `ltePHICHIndices`
- Control Format Indicator
  - Coding
    - `lteCFI`
    - `lteCFIDecode`
  - Modulation
    - `ltePCFICH`
    - `ltePCFICHDecode`
  - Resource Indices — `ltePCFICHIndices`
- Broadcast Channel
  - Transport
    - `lteBCH`
    - `lteBCHDecode`
  - Modulation
    - `ltePBCH`

    - `ltePBCHDecode`
  - Resource Indices — `ltePBCHIndices`
- Shared Channel
  - Transport
    - `lteDLSCH`
    - `lteDLSCHDecode`
  - Modulation
    - `ltePDSCH`
    - `ltePDSCHDecode`
  - Resource Indices — `ltePDSCHIndices`
- General
  - `lteDLResourceGrid`
  - `lteDLResourceGridSize`
  - `lteOFDMModulate`
  - `lteOFDMDemodulate`

Downlink Physical Signals Grid

The downlink physical signals, their associated functions, and their locations on the resource grid are shown in the following figure.

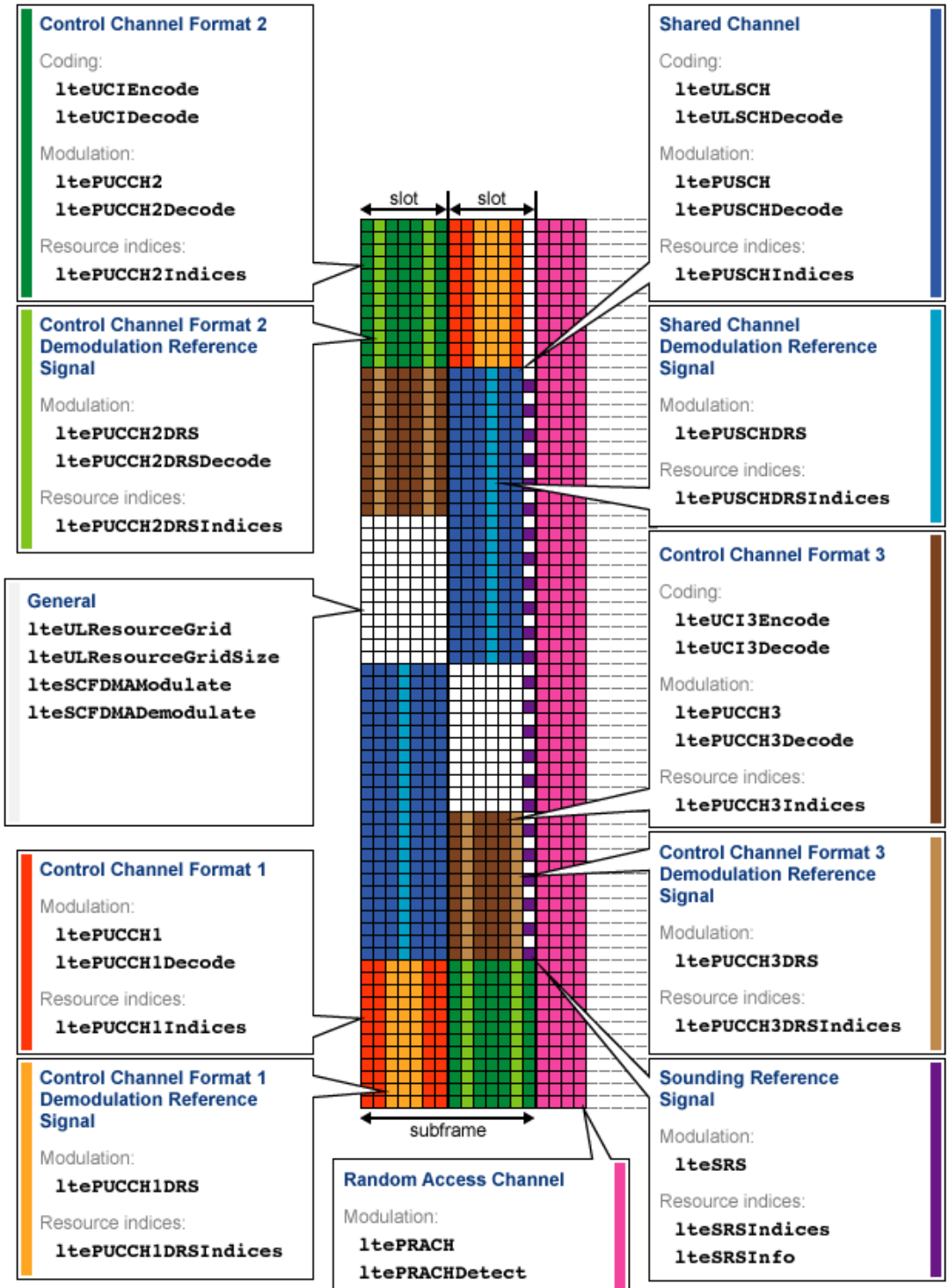


- Primary Synchronization Signal (PSS)

    - `ltePSS`
    - `ltePSSIndices`
- Secondary Synchronization Signal (SSS)
    - `lteSSS`
    - `lteSSSIndices`
- Cell-specific Reference Signal (CRS)
    - `lteCellRS`
    - `lteCellRSIndices`
- Positioning Reference Signal (PRS)
    - `ltePRS`
    - `ltePRSIndices`
- Channel State Information Reference Signal (CSI-RS)
    - `lteCSIRS`
    - `lteCSIRSIndices`
- Demodulation Reference Signal (DMRS)
    - `lteDMRS`
    - `lteDMRSIndices`
- General
    - `lteDLResourceGrid`
    - `lteDLResourceGridSize`
    - `lteOFDMModulate`

Uplink Physical Channels and Signals Grid

The uplink physical channels and signals, their associated functions, and their locations on the resource grid are shown in the following figure.


Control Channel Format 2
Coding:
lteUCIEncode
lteUCIDecode
Modulation:
ltePUCCH2
ltePUCCH2Decode
Resource indices:
ltePUCCH2Indices
Control Channel Format 2 Demodulation Reference Signal
Modulation:
ltePUCCH2DRS
ltePUCCH2DRSDecode
Resource indices:
ltePUCCH2DRSIndices
General
lteULResourceGrid
lteULResourceGridSize
lteSCFDMAModulate
lteSCFDMADemodulate
Control Channel Format 1
Modulation:
ltePUCCH1
ltePUCCH1Decode
Resource indices:
ltePUCCH1Indices
Control Channel Format 1 Demodulation Reference Signal
Modulation:
ltePUCCH1DRS
Resource indices:
ltePUCCH1DRSIndices
slot
slot
subframe
Random Access Channel
Modulation:
ltePRACH
ltePRACHDetect
Shared Channel
Coding:
lteULSCH
lteULSCHDecode
Modulation:
ltePUSCH
ltePUSCHDecode
Resource indices:
ltePUSCHIndices
Shared Channel Demodulation Reference Signal
Modulation:
ltePUSCHDRS
Resource indices:
ltePUSCHDRSIndices
Control Channel Format 3
Coding:
lteUCI3Encode
lteUCI3Decode
Modulation:
ltePUCCH3
ltePUCCH3Decode
Resource indices:
ltePUCCH3Indices
Control Channel Format 3 Demodulation Reference Signal
Modulation:
ltePUCCH3DRS
Resource indices:
ltePUCCH3DRSIndices
Sounding Reference Signal
Modulation:
lteSRS
Resource indices:
lteSRSIndices
lteSRSInfo

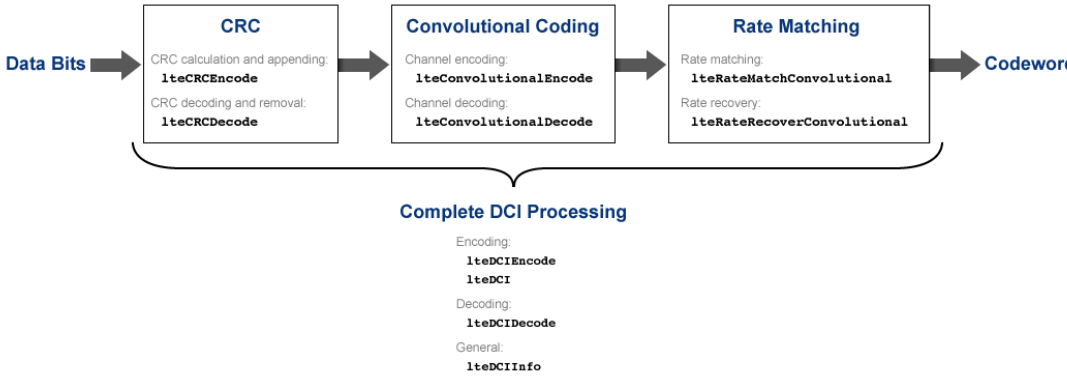
- Control Channel Format 1
  - Modulation
    - `ltePUCCH1`
    - `ltePUCCH1Decode`
  - Resource Indices — `ltePUCCH1Indices`
- Control Channel Format 2
  - Coding
    - `lteUCIEncode`
    - `lteUCIDecode`
  - Modulation
    - `ltePUCCH2`
    - `ltePUCCH2Decode`
  - Resource Indices — `ltePUCCH2Indices`
- Control Channel Format 3
  - Coding
    - `lteUCI3Encode`
    - `lteUCI3Decode`
  - Modulation
    - `ltePUCCH3`
    - `ltePUCCH3Decode`
  - Resource Indices — `ltePUCCH3Indices`
- Random Access Channel
  - Modulation
    - `ltePRACH`
    - `ltePRACHDetect`
  - General — `ltePRACHInfo`
- Shared Channel

    - Transport
        - `lteULSCH`
        - `lteULSCHDecode`
    - Modulation
        - `ltePUSCH`
        - `ltePUSCHDecode`
    - Resource Indices
        - `ltePUSCHIndices`
- Control Channel Format 1 Demodulation Reference Signal
    - Modulation — `ltePUCCH1DRS`
    - Resource Indices — `ltePUCCH1DRSIndices`
- Control Channel Format 2 Demodulation Reference Signal
    - Modulation
        - `ltePUCCH2DRS`
        - `ltePUCCH2DRSDecode`
    - Resource Indices — `ltePUCCH2DRSIndices`
- Control Channel Format 3 Demodulation Reference Signal
    - Modulation — `ltePUCCH3DRS`
    - Resource Indices — `ltePUCCH3DRSIndices`
- Sounding Reference Signal
    - Modulation — `lteSRS`
    - Resource Indices — `lteSRSIndices`
    - General — `lteSRSInfo`
- Shared Channel Demodulation Reference Signal
    - Modulation — `ltePUSCHDRS`
    - Resource Indices — `ltePUSCHDRSIndices`
- General

- `lteULResourceGrid`
- `lteULResourceGridSize`
- `lteSCFDMAModulate`
- `lteSCFDMADemodulate`

DCI Processing Functions

The complete downlink control information process and associated low-level and mid-level DCI functions are shown in the following block diagram.



- Cyclic redundancy check (CRC)
  - CRC calculation and appending — `lteCRCEncode`
  - CRC decoding and removal — `lteCRCDecode`
- Convolutional channel coding
  - Channel encoding — `lteConvolutionalEncode`
  - Channel decoding — `lteConvolutionalDecode`
- Rate matching and recovery
  - Rate matching — `lteRateMatchConvolutional`
  - Rate recovery — `lteRateRecoverConvolutional`
- Complete DCI processing
  - Encoding
    - `lteDCIEncode`
    - `lteDCI`
  - Decoding — `lteDCIDecode`

- General — `lteDCIInfo`

Related Examples

- "Model DCI and PDCCH"

More About

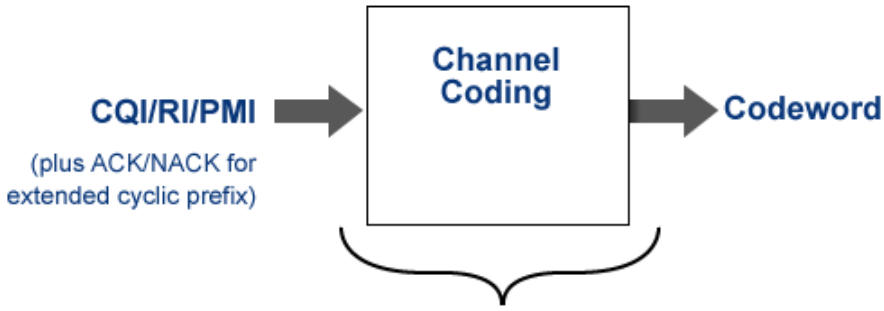
- "Downlink Control Channel"

UCI Processing Functions

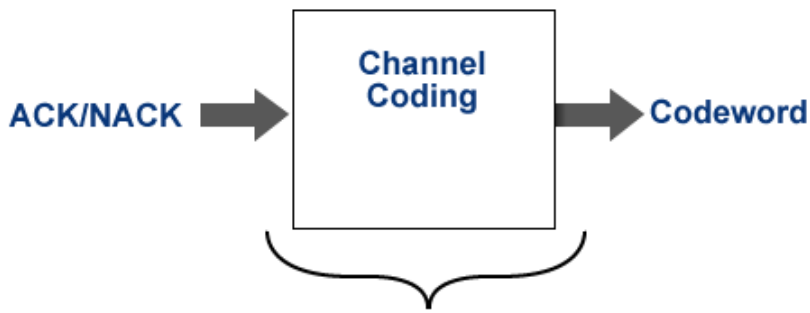
The uplink control information process for PUCCH format 1, 2, and 3 and associated mid-level UCI functions are shown in the following block diagram.

PUCCH Format 1:

PUCCH Format 2:



PUCCH Format 3:



- PUCCH format 2 complete UCI processing
  - `lteUCIEncode`
  - `lteUCIDecode`
- PUCCH format 3 complete UCI processing
  - `lteUCI3Encode`
  - `lteUCI3Decode`

Related Examples

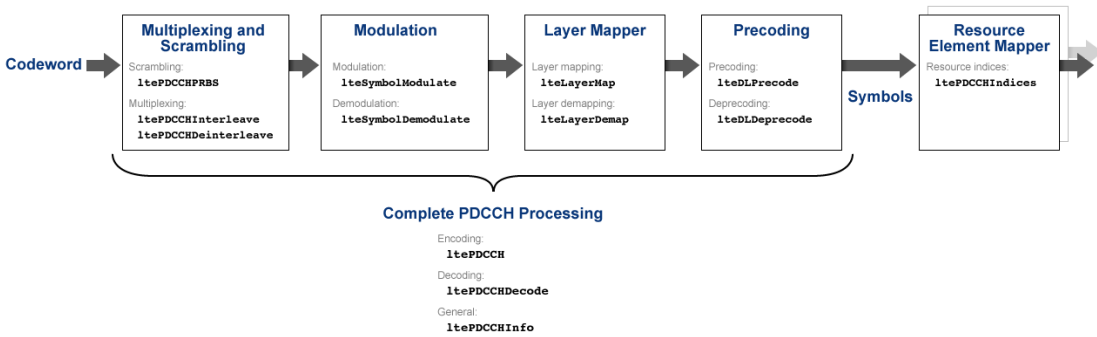
- "Model PUCCH Format 2"

More About

- "Uplink Control Channel Format 1"
- "Uplink Control Channel Format 2"

PDCCH Processing Functions

The complete physical downlink control channel process and associated low-level and mid-level PDCCH functions are shown in the following block diagram.



- Multiplexing and scrambling
  - Scrambling — `ltePDCCHPRBS`
  - Multiplexing
    - `ltePDCCHInterleave`
    - `ltePDCCHDeinterleave`
- Symbol modulation and demodulation
  - Modulation — `lteSymbolModulate`
  - Demodulation — `lteSymbolDemodulate`
- Layer mapper and de-mapper
  - Layer mapping — `lteLayerMap`
  - Layer demapping — `lteLayerDemap`
- Precoding and deprecoding
  - Precoding — `lteDLPrecode`
  - Deprecoding — `lteDLDeprecode`
- Resource element mapper

  - Resource indices — `ltePDCCHIndices`
- Complete PDCCH processing
  - Encoding — `ltePDCCH`
  - Decoding — `ltePDCCHDecode`
  - General — `ltePDCCHInfo`

Related Examples

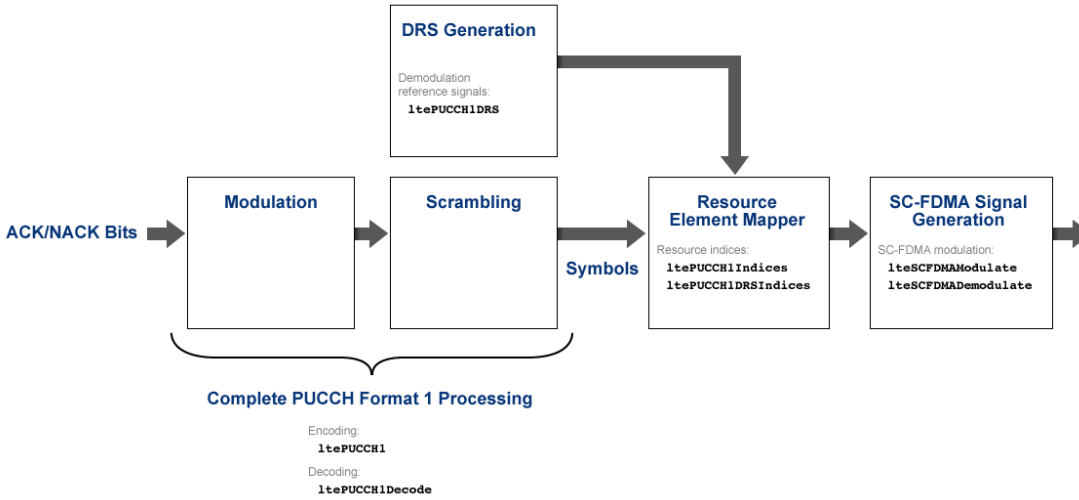
- "Model DCI and PDCCH"

More About

- "Downlink Control Channel"

PUCCH Format 1 Processing Functions

The complete physical uplink control channel format 1 process and associated low-level and mid-level PUCCH format 1 functions are shown in the following block diagram.



- Demodulation reference signal (DRS) generation
  - Demodulation reference signals — `ltePUCCH1DRS`
- Resource element mapper
  - Resource indices — `ltePUCCH1Indices`
  - DRS resource indices — `ltePUCCH1DRSIndices`
- SC-FDMA signal generation
  - SC-FDMA modulation — `lteSCFDMAModulate`
  - SC-FDMA demodulation — `lteSCFDMADemodulate`
- Complete PUCCH format 1 processing
  - Encoding — `ltePUCCH1`
  - Decoding — `ltePUCCH1Decode`

Related Examples

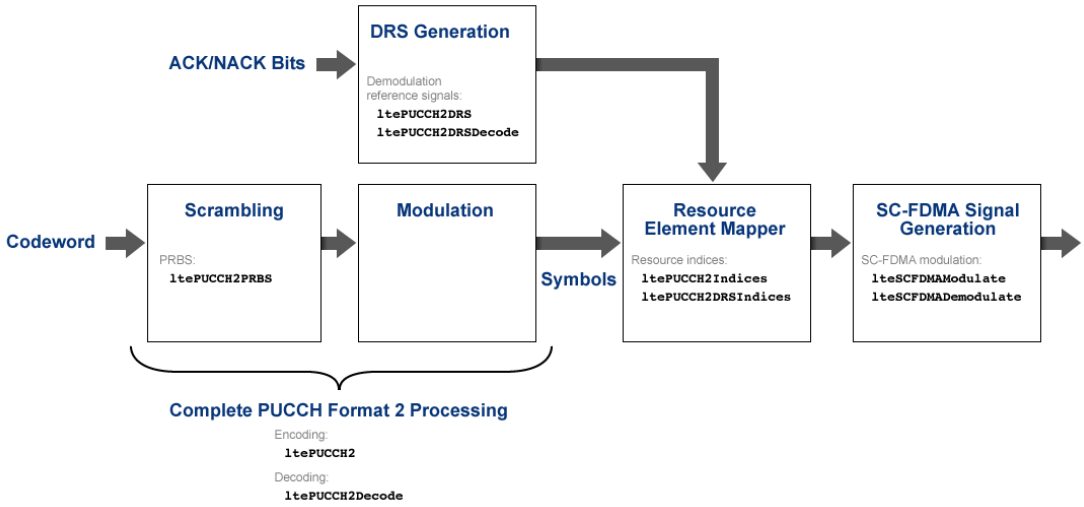
- “Model PUCCH Format 1”
- “PUCCH1a ACK Missed Detection Probability Conformance Test”
- “PUCCH1a Multi User ACK Missed Detection Probability Conformance Test”

More About

- “Uplink Control Channel Format 1”

PUCCH Format 2 Processing Functions

The complete physical uplink control channel format 2 process and associated low-level and mid-level PUCCH format 2 functions are shown in the following block diagram.



- Scrambling
  - Pseudo-random binary sequence (PRBS) — `ltePUCCH2PRBS`
- Demodulation reference signal (DRS) generation
  - Demodulation reference signals — `ltePUCCH2DRS`
  - Demodulation reference signal decoding — `ltePUCCH2DRSDecode`
- Resource element mapper
  - Resource indices — `ltePUCCH2Indices`
  - DRS resource indices — `ltePUCCH2DRSIndices`
- SC-FDMA signal generation
  - SC-FDMA modulation — `lteSCFDMAModulate`
  - SC-FDMA demodulation — `lteSCFDMADemodulate`
- Complete PUCCH format 2 processing

- Encoding — `ltePUCCH2`
- Decoding — `ltePUCCH2Decode`

Related Examples

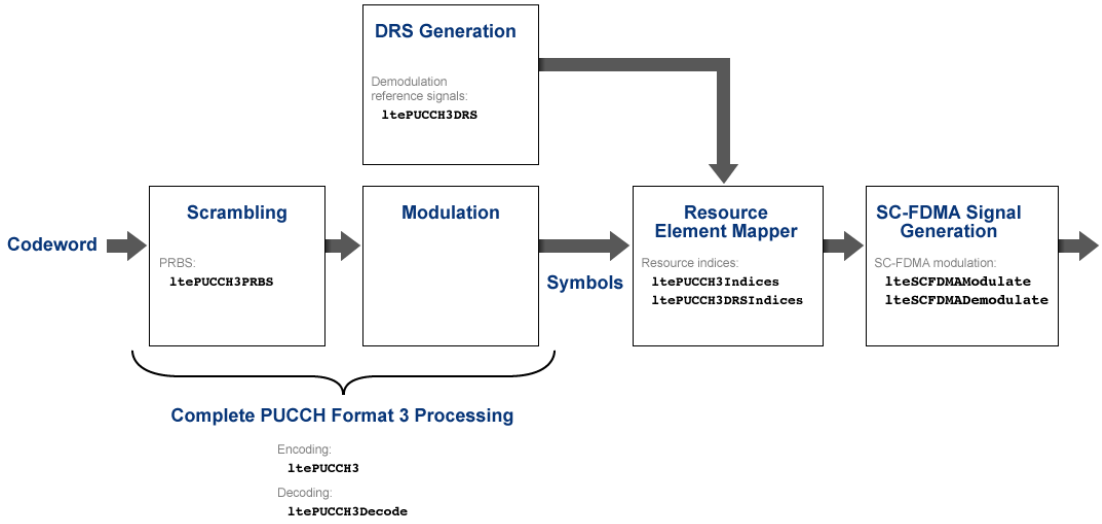
- "Model PUCCH Format 2"

More About

- "Uplink Control Channel Format 2"

PUCCH Format 3 Processing Functions

The complete physical uplink control channel format 3 process and associated low-level and mid-level PUCCH format 3 functions are shown in the following block diagram.

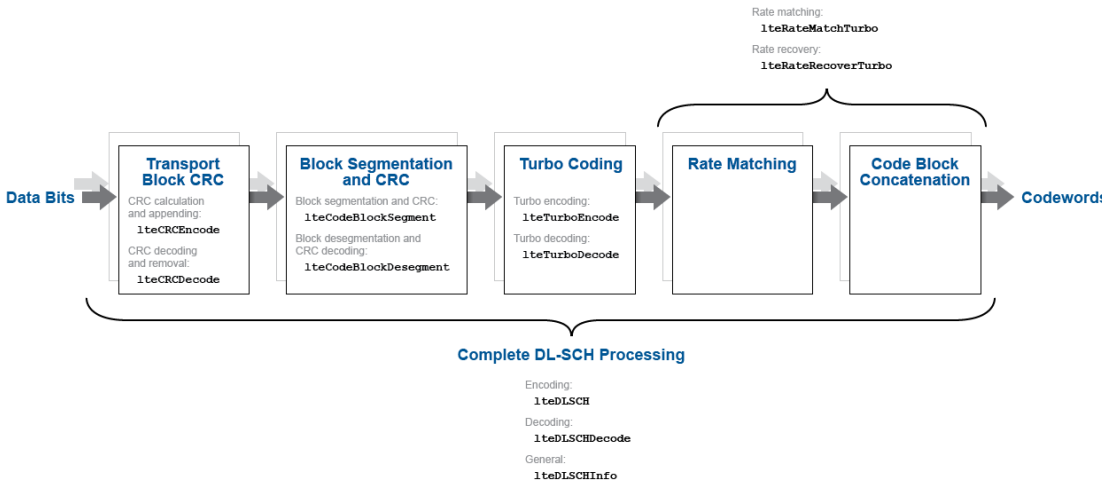


- Scrambling
  - Pseudo-random binary sequence (PRBS) — `ltePUCCH3PRBS`
- Demodulation reference signal (DRS) generation
  - Demodulation reference signals — `ltePUCCH3DRS`
- Resource element mapper
  - Resource indices — `ltePUCCH3Indices`
  - DRS resource indices — `ltePUCCH3DRSIndices`
- SC-FDMA signal generation
  - SC-FDMA modulation — `lteSCFDMAModulate`
  - SC-FDMA demodulation — `lteSCFDMADemodulate`
- Complete PUCCH format 3 processing

- Encoding — `ltePUCCH3`
- Decoding — `ltePUCCH3Decode`

DL-SCH Processing Functions

The complete downlink shared channel process and associated low-level and mid-level DL-SCH functions are shown in the following block diagram.



- Transport block cyclic redundancy check (CRC)
  - CRC calculation and appending — `lteCRCEncode`
  - CRC decoding and removal — `lteCRCDecode`
- Block segmentation and CRC
  - Block segmentation and CRC attachment — `lteCodeBlockSegment`
  - Block desegmentation and CRC decoding — `lteCodeBlockDesegment`
- Turbo encoding and decoding
  - `lteTurboEncode`
  - `lteTurboDecode`
- Rate matching and recovery
  - Rate matching — `lteRateMatchTurbo`
  - Rate recovery — `lteRateRecoverTurbo`
- Complete DL-SCH processing

- Encoding — `lteDLSCH`
- Decoding — `lteDLSCHDecode`
- General — `lteDLSCHInfo`

Related Examples

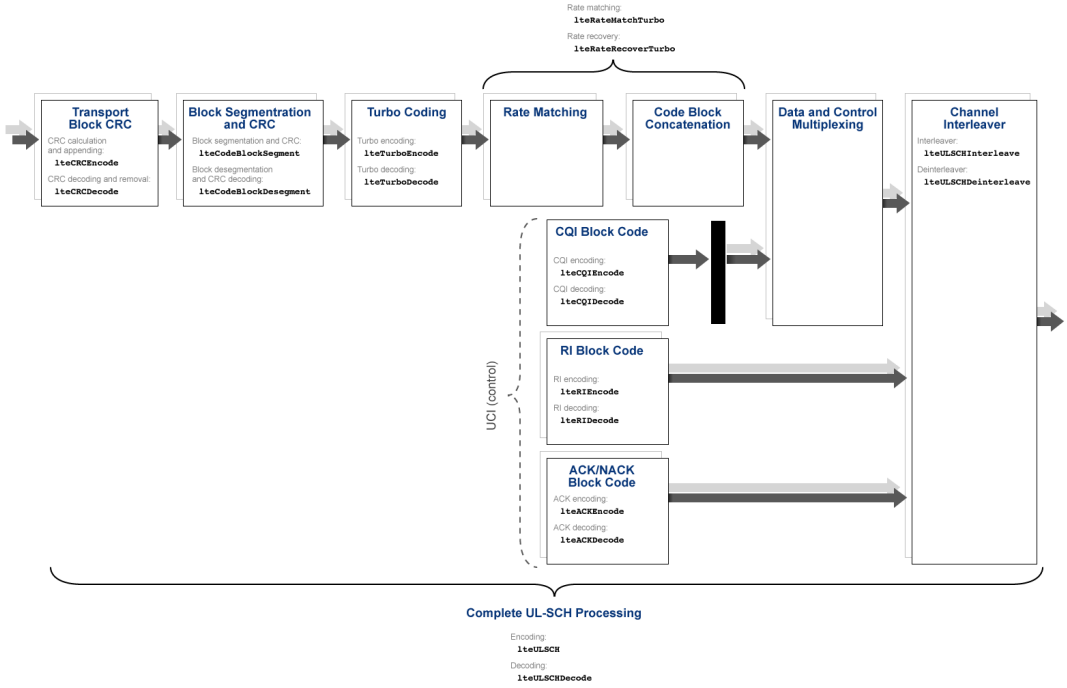
- "Model DL-SCH and PDSCH"
- "DL-SCH HARQ Modeling"

More About

- "Downlink Shared Channel"

UL-SCH Processing Functions

The complete uplink shared channel process and associated low-level and mid-level UL-SCH functions are shown in the following block diagram.



- Transport block cyclic redundancy check (CRC)
  - CRC calculation and appending — `lteCRCEncode`
  - CRC decoding and removal — `lteCRCDecode`
- Block segmentation and CRC
  - Block segmentation and CRC attachment — `lteCodeBlockSegment`
  - Block desegmentation and CRC decoding — `lteCodeBlockDesegment`
- Turbo encoding and decoding
  - `lteTurboEncode`

    - `lteTurboDecode`
- Rate matching and recovery
    - Rate matching — `lteRateMatchTurbo`
    - Rate recovery — `lteRateRecoverTurbo`
- Uplink control information (UCI)
    - Channel quality information (CQI) block code
        - CQI encoding — `lteCQIEncode`
        - CQI decoding — `lteCQIDecode`
    - Rank indicator (RI) block code
        - RI encoding — `lteRIEncode`
        - RI decoding — `lteRIDecode`
    - Acknowledgement (ACK) or Negative acknowledgement (NACK) block code
        - ACK encoding — `lteACKEncode`
        - ACK decoding — `lteACKDecode`
- Channel interleaver
    - Interleaver — `lteULSCHInterleave`
    - Deinterleaver — `lteULSCHDeinterleave`
- Complete UL-SCH processing
    - Encoding — `lteULSCH`
    - Decoding — `lteULSCHDecode`

Related Examples

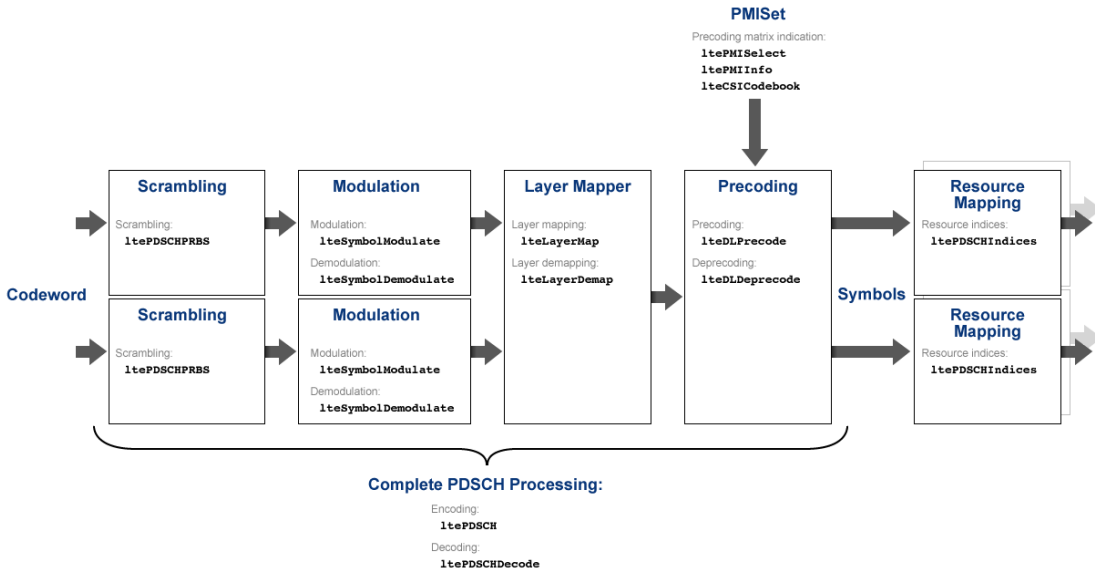
- "Model UL-SCH and PUSCH"

More About

- "Uplink Shared Channel"

PDSCH Processing Functions

The complete physical downlink shared channel process and associated low-level and mid-level PDSCH functions are shown in the following block diagram.



- Scrambling — `ltePDSCHPRBS`
- Symbol modulation and demodulation
  - Modulation — `lteSymbolModulate`
  - Demodulation — `lteSymbolDemodulate`
- Layer mapping
  - Layer mapping — `lteLayerMap`
  - Layer demapping — `lteLayerDemap`
- Precoding and deprecoding
  - Precoding — `lteDLPrecode`
  - Deprecoding — `lteDLDeprecode`
- Downlink precoding matrix indication (PMI)

  - `ltePMISelect`
  - `ltePMIInfo`
  - `lteCSICodebook`
- Resource mapping
  - Resource indices — `ltePDSCHIndices`
- Complete PDSCH processing
  - Encoding — `ltePDSCH`
  - Decoding — `ltePDSCHDecode`

Related Examples

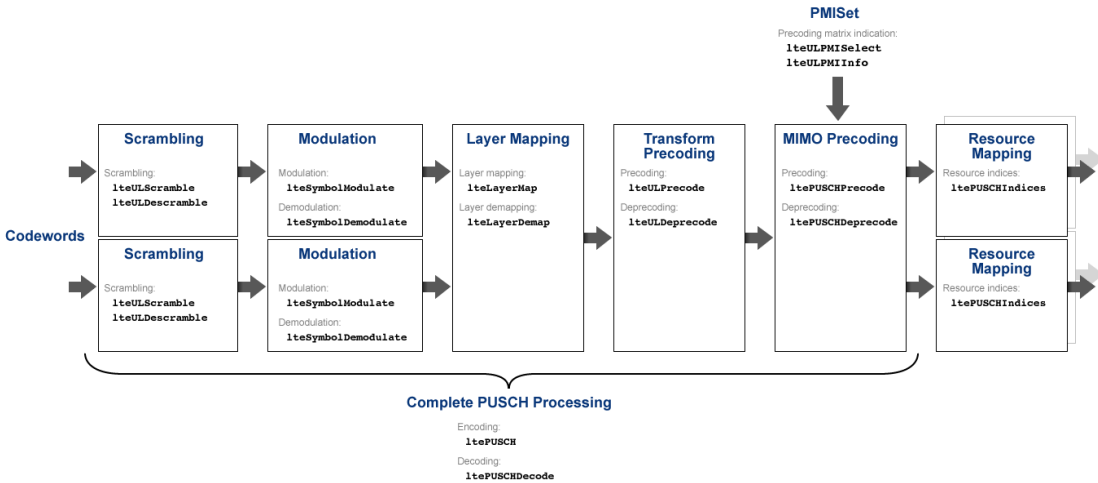
- "Model DL-SCH and PDSCH"
- "PDSCH Bit Error Rate Curve Generation"

More About

- "Downlink Shared Channel"

PUSCH Processing Functions

The complete physical uplink shared channel process and associated low-level and mid-level PUSCH functions are shown in the following block diagram.



- Scrambling
  - Scrambling — `lteULScramble`
  - Descrambling — `lteULDescramble`
- Symbol modulation and demodulation
  - Modulation — `lteSymbolModulate`
  - Demodulation — `lteSymbolDemodulate`
- Layer mapping
  - Layer mapping — `lteLayerMap`
  - Layer demapping — `lteLayerDemap`
- Transform precoding and deprecoding
  - Transform precoding — `lteULPrecode`
  - Transform deprecoding — `lteULDeprecode`

- Multiple-input multiple-output (MIMO) precoding and deprecoding
  - MIMO precoding — `ltePUSCHPrecode`
  - MIMO deprecoding — `ltePUSCHDeprecode`
- Uplink (UL) precoding matrix indication (PMI)
  - `lteULPMISelect`
  - `lteULPMIInfo`
- Resource mapping
  - Resource indices — `ltePUSCHIndices`
- Complete PUSCH processing
  - Encoding — `ltePUSCH`
  - Decoding — `ltePUSCHDecode`

Related Examples

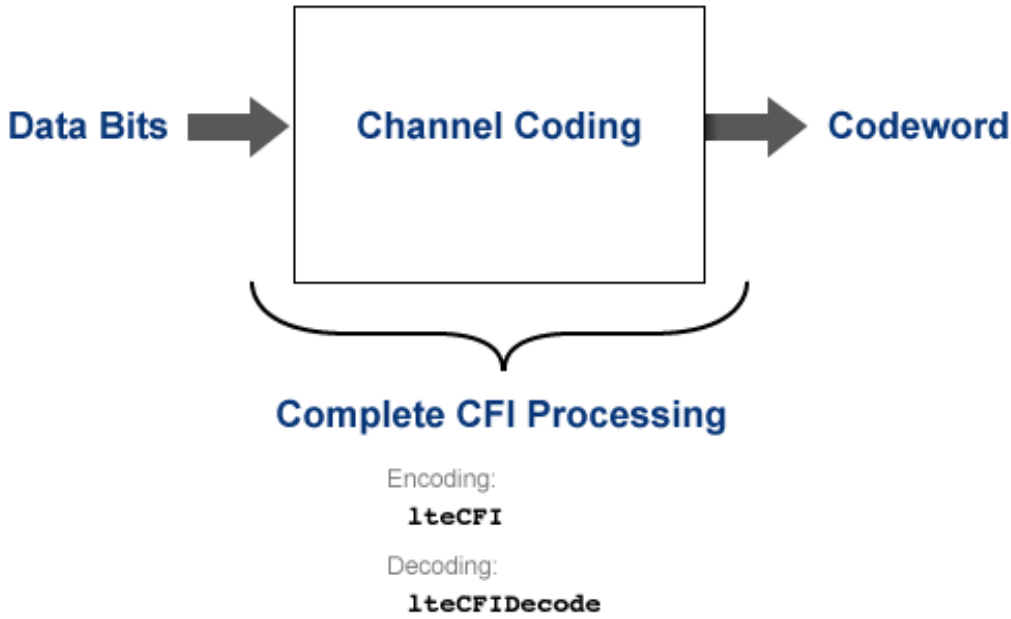
- "Model UL-SCH and PUSCH"

More About

- "Uplink Shared Channel"

CFI Processing Functions

The complete control format information process and associated low-level and mid-level CFI functions are shown in the following block diagram.



- Complete CFI processing
  - Encoding — `lteCFI`
  - Decoding — `lteCFIDecode`

Related Examples

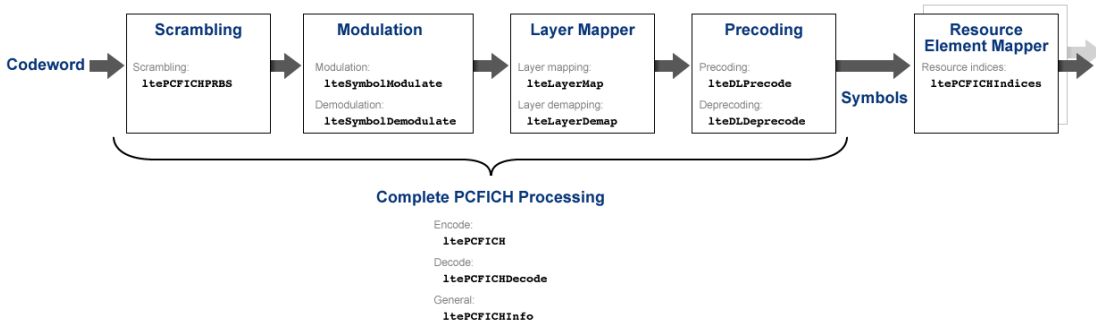
- "Model CFI and PCFICH"

More About

- "Control Format Indicator (CFI) Channel"

PCFICH Processing Functions

The complete physical downlink control format indicator channel process and associated low-level and mid-level PCFICH functions are shown in the following block diagram.



- Scrambling — `ltePCFICHPRBS`
- Symbol modulation and demodulation
  - Modulation — `lteSymbolModulate`
  - Demodulation — `lteSymbolDemodulate`
- Layer mapper and demapper
  - Layer mapping — `lteLayerMap`
  - Layer demapping — `lteLayerDemap`
- Precoding and deprecoding
  - Precoding — `lteDLPrecode`
  - Deprecoding — `lteDLDeprecode`
- Resource element mapper
  - Resource indices — `ltePCFICHIndices`
- Complete PCFICH processing
  - Encoding — `ltePCFICH`
  - Decoding — `ltePCFICHDecode`
  - General — `ltePCFICHInfo`

Related Examples

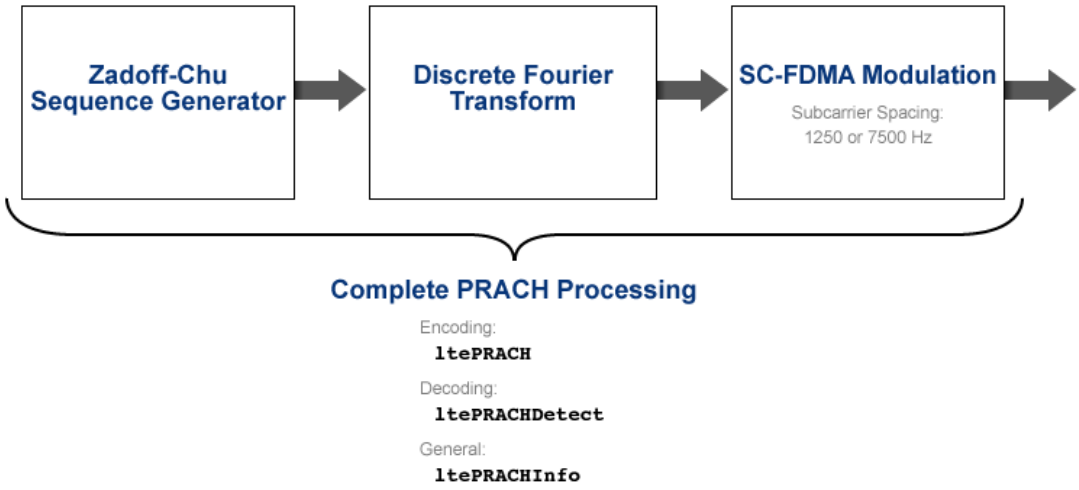
- “Model CFI and PCFICH”

More About

- “Control Format Indicator (CFI) Channel”

PRACH Processing Functions

The random access channel process and associated PRACH functions are shown in the following block diagram.



- Complete PRACH processing
  - Encoding — `ltePRACH`
  - Decoding — `ltePRACHDetect`
  - General — `ltePRACHInfo`

Related Examples

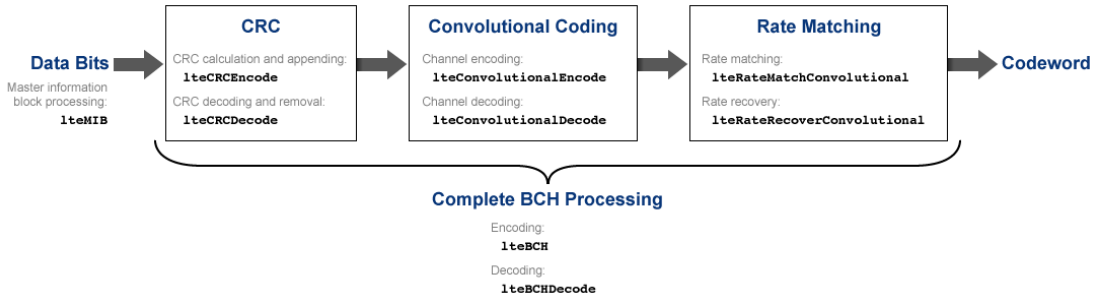
- "PRACH False Alarm Probability Conformance Test"
- "PRACH Detection Conformance Test"

More About

- "Random Access Channel"

BCH Processing Functions

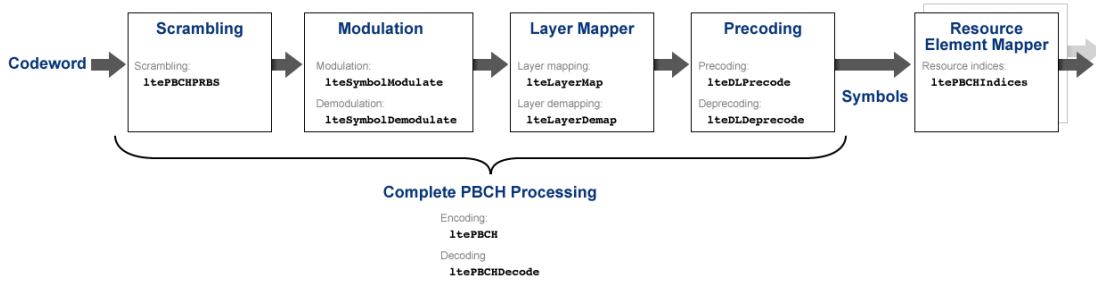
The complete broadcast channel process and associated low-level and mid-level BCH functions are shown in the following block diagram.



- Cyclic redundancy check (CRC)
  - CRC calculation and appending — `lteCRCEncode`
  - CRC decoding and removal — `lteCRCDecode`
- Convolutional channel encoding and decoding
  - `lteConvolutionalEncode`
  - `lteConvolutionalDecode`
- Rate matching and recovery
  - Rate matching — `lteRateMatchConvolutional`
  - Rate recovery — `lteRateRecoverConvolutional`
- Master information block (MIB) processing — `lteMIB`
- Complete BCH processing
  - Encoding — `lteBCH`
  - Decoding — `lteBCHDecode`

PBCH Processing Functions

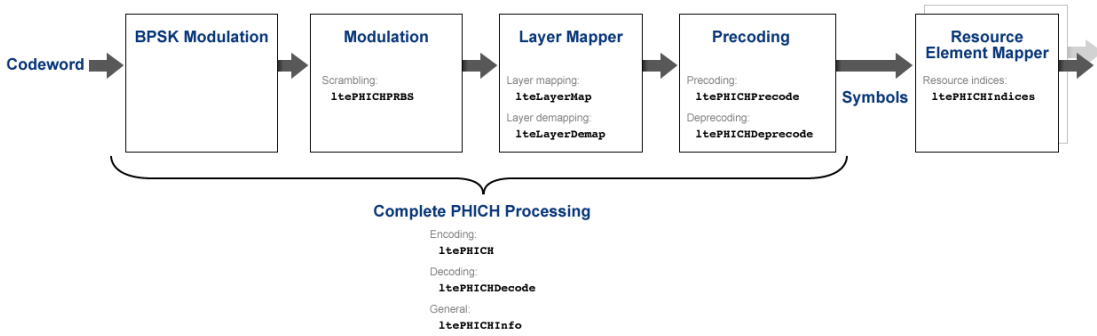
The complete physical broadcast channel process and associated low-level and mid-level PBCH functions are shown in the following block diagram.



- Scrambling — `ltePBCHPRBS`
- Symbol modulation and demodulation
  - Modulation — `lteSymbolModulate`
  - Demodulation — `lteSymbolDemodulate`
- Layer mapper and demapper
  - Layer mapping — `lteLayerMap`
  - Layer demapping — `lteLayerDemap`
- Precoding and deprecoding
  - Precoding — `lteDLPrecode`
  - Deprecoding — `lteDLDeprecode`
- Resource element mapper
  - Resource indices — `ltePBCHIndices`
- Complete PBCH processing
  - Encoding — `ltePBCH`
  - Decoding — `ltePBCHDecode`

PHICH Processing Functions

The complete physical hybrid automatic repeat request (HARQ) indicator channel process and associated low-level and mid-level PHICH functions are shown in the following block diagram.



- Scrambling — `ltePHICHPRBS`
- Layer mapper and demapper
  - Layer mapping — `lteLayerMap`
  - Layer demapping — `lteLayerDemap`
- Precoding and deprecoding
  - Precoding — `ltePHICHPrecode`
  - Deprecoding — `ltePHICHDeprecode`
- Resource element mapper
  - Resource indices — `ltePHICHIndices`
- Complete PBCH processing
  - Encoding — `ltePHICH`
  - Decoding — `ltePHICHDecode`
  - General — `ltePHICHInfo`

Related Examples

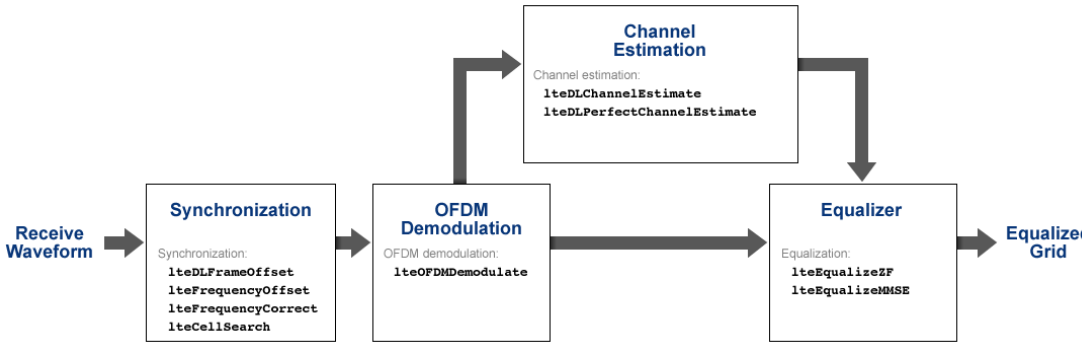
- "Model HARQ Indicator and PHICH"

More About

- “HARQ Indicator (HI) Channel”

Downlink Receiver Functions

The complete downlink (DL) receiver process and associated functions are shown in the following block diagram.



- Synchronization
  - `lteDLFrameOffset`
  - `lteFrequencyOffset`
  - `lteFrequencyCorrect`
  - `lteCellSearch`
- OFDM demodulation — `lteOFDMDemodulate`
- Channel estimation
  - `lteDLChannelEstimate`
  - `lteDLPerfectChannelEstimate`
- Equalization
  - `lteEqualizeZF`
  - `lteEqualizeMMSE`

Related Examples

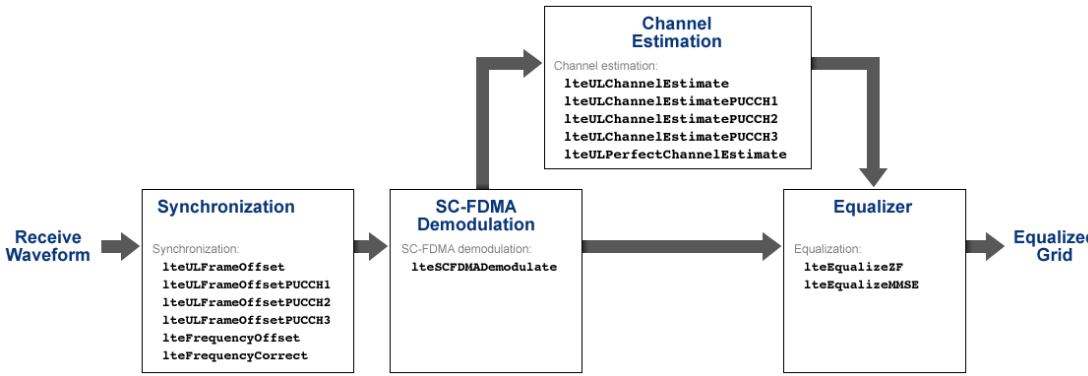
- "LTE Downlink Channel Estimation and Equalization"

More About

- “Channel Estimation”

Uplink Receiver Functions

The complete uplink (UL) receiver process and associated functions are shown in the following block diagram.



- Synchronization
  - `lteULFrameOffset`
  - `lteULFrameOffsetPUCCH1`
  - `lteULFrameOffsetPUCCH2`
  - `lteULFrameOffsetPUCCH3`
  - `lteFrequencyOffset`
  - `lteFrequencyCorrect`
- SC-FDMA demodulation — `lteSCFDMADemodulate`
- Channel estimation
  - `lteULChannelEstimate`
  - `lteULChannelEstimatePUCCH1`
  - `lteULChannelEstimatePUCCH2`
  - `lteULChannelEstimatePUCCH3`
  - `lteULPerfectChannelEstimate`
- Equalization

- `lteEqualizeZF`
- `lteEqualizeMMSE`

Related Examples

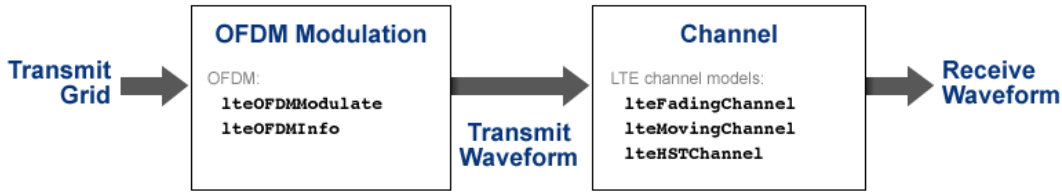
- "PUCCH2 CQI BLER Conformance Test"

More About

- "Channel Estimation"

OFDM Modulation and Propagation Channel Models

The orthogonal frequency-division multiplexing (OFDM) modulation process, propagation channel models, and their associated functions are shown in the following block diagram.



- OFDM modulation
  - `lteOFDMModulate`
  - `lteOFDMInfo`
- LTE propagation channel models
  - `lteFadingChannel`
  - `lteMovingChannel`
  - `lteHSTChannel`

Related Examples

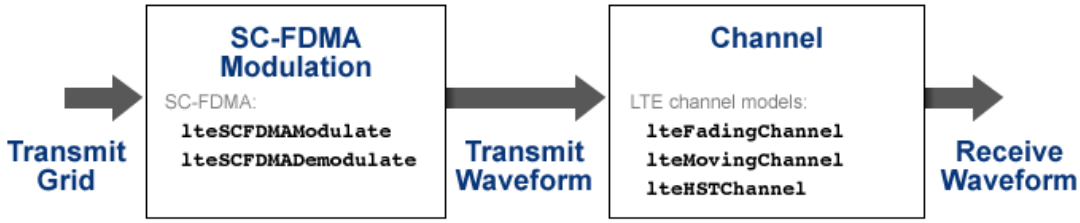
- "Simulate Propagation Channels"
- "Find Channel Impulse Response"

More About

- "Propagation Channel Models"

SC-FDMA Modulation and Propagation Channel Models

The single-carrier frequency-division multiple access (SC-FDMA) modulation process, propagation channel models, and their associated functions are shown in the following block diagram.



- SC-FDMA modulation
  - `lteSCFDMAModulate`
  - `lteSCFDMADemodulate`
- LTE propagation channel models
  - `lteFadingChannel`
  - `lteMovingChannel`
  - `lteHSTChannel`

Related Examples

- "Simulate Propagation Channels"
- "Find Channel Impulse Response"

More About

- "Propagation Channel Models"

Glossary

This list defines commonly used LTE abbreviations and acronyms. These terms may appear in some or all of the documents that describe MathWorks® products that model communications and LTE.

3G	Third Generation
3GPP	Third Generation Partnership Project
4G	Fourth Generation
ACK	Acknowledgement — in ARQ protocols
ACLR	Adjacent Channel Leakage Power Ratio
	Adjacent Channel Leakage Ratio
ARQ	Automatic Repeat Request
AWGN	Additive White Gaussian Noise
BCCH	Broadcast Control Channel
BCH	Broadcast Channel
BER	Bit Error Rate
	Bit Error Ratio
BLER	Block Error Rate
	Block Error Ratio
BPSK	Binary Phase-Shift Keying
BS	Base Station
BW	Bandwidth
CA	Carrier Aggregation
CDMA	Code Division Multiple Access
CellRS	Cell-specific Reference Signal

CFI	Control Format Indicator
CP	Cyclic Prefix
CQI	Channel Quality Indicator
	Channel Quality Information
CRC	Cyclic Redundancy Check
CRS	Cell-specific Reference Signal
CSI	Channel State Information
CSI-RS	CSI Reference Signals
DCI	Downlink Control Information
DL	Downlink
DL-SCH	Downlink Shared Channel
DM-RS	Demodulation Reference Signal
DRS	Demodulation Reference Signal
DwPTS	Downlink Pilot Time Slot — the downlink part of the special subframe, for TDD operation
eICIC	Enhanced Inter-Cell Interference Coordination
eMBMS	Evolved Multimedia Broadcast and Multicast Service
eNB	eNodeB
eNodeB	E-UTRAN NodeB
EPA	Extended Pedestrian A
EPC	Evolved Packet Core
EPDCCH	Enhanced Physical Downlink Control Channel
EPS	Evolved Packet System

E-TM	E-UTRA Test Model
ETU	Extended Typical Urban model
E-UTRA	Evolved UTRA
E-UTRAN	Evolved UTRAN
EVA	Extended Vehicular A
EVM	Error Vector Magnitude
FDD	Frequency Division Duplex
FDM	Frequency Division Multiplex
FDMA	Frequency Division Multiple Access
FFT	Fast Fourier Transform
FRC	Fixed Reference Channel
GP	Guard Period
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HARQ	Hybrid ARQ
HetNets	Heterogeneous Networks
HI	HARQ Indicator
HST	High Speed Train
ICIC	Inter-Cell Interference Coordination
IFFT	Inverse Fast Fourier Transform
IP	Internet Protocol
LCS	Location Services

LLR	Log-Likelihood Ratio
LTE	Long Term Evolution
MAC	Medium Access Control
MIB	Master Information Block
MIMO	Multiple-Input Multiple-Output
ML	Maximum Likelihood
MMSE	Minimum Mean Square Error
NACK	Negative Acknowledgement — in ARQ protocols
NAK	Negative Acknowledgement — in ARQ protocols
NodeB	A logical node handling transmission and reception in multiple cells. A NodeB commonly, but not necessarily, corresponds to a base station.
NRE	Number of Resource Elements
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access
OSFBC	Orthogonal Space Frequency Block Code
PBCH	Physical Broadcast Channel
PCFICH	Physical Control Format Indicator Channel
PDCCH	Physical Downlink Control Channel
PDCP	Packet Data Convergence Protocol
PDSCH	Physical Downlink Shared Channel
PHICH	Physical Hybrid-ARQ Indicator Channel
PHY	Physical layer

PLMN	Public Land Mobile Network
PMI	Precoding Matrix Indicator
PRACH	Physical Random Access Channel
PRB	Physical Resource Block
PRBS	Pseudorandom Binary Sequence
PRS	Positioning Reference Signal
PSD	Power Spectral Density
PSS	Primary Synchronization Signal
PUCCH	Physical Uplink Control Channel
PUSCH	Physical Uplink Shared Channel
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
	Quaternary Phase Shift Keying
RACH	Random Access Channel
RB	Resource Block
RE	Resource Element
REG	Resource Element Group
RI	Rank Indication
	Rank Indicator
RLC	Radio Link Control
RMC	Reference Measurement Channel
RMS	Root Mean Square

RNTI	Radio Network Temporary Identifier
RS	Reference Symbol
RV	Redundancy Version
RX	Receive
	Receiver
SC-FDMA	Single Carrier Frequency Division Multiple Access
SFN	System Frame Number
SI	System Information
SIB	System Information Block
SIB1	System Information Block type 1
SRS	Sounding Reference Signal
SSS	Secondary Synchronization Signal
TBS	Transport Block Size
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
TDOA	Time Difference of Arrival
TPC	Transmit Power Control
TS	Technical Specification
TX	Transmit
	Transmitter
UCI	Uplink Control Information
UE	User Equipment — the 3GPP name for the mobile terminal

UE-RS	UE-specific Reference Signal
UL	Uplink
UL-SCH	Uplink Shared Channel
UMTS	Universal Mobile Telecommunication System
UpPTS	Uplink Pilot Time Slot — the uplink part of the special subframe, for TDD operation
UTRA	Universal Terrestrial Radio Access
UTRAN	Universal Terrestrial Radio Access Network
WiMAX	Worldwide interoperability for Microwave Access
ZF	Zero Forcing

Selected Bibliography

[1] 3GPP TS 36.101. "User Equipment (UE) Radio Transmission and Reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[2] 3GPP TS 36.104. "Base Station (BS) radio transmission and reception." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[3] 3GPP TS 36.141. "Base Station (BS) conformance testing." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[4] 3GPP TS 36.211. "Physical Channels and Modulation." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[5] 3GPP TS 36.212. "Multiplexing and channel coding." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[6] 3GPP TS 36.213. "Physical layer procedures." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[7] 3GPP TS 36.214. "Physical layer — Measurements." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[8] 3GPP TS 36.321. "Medium Access Control (MAC) protocol specification." *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).* URL: http://www.3gpp.org.

[9] Chu, D. C. “Polyphase codes with good periodic correlation properties.” *IEEE Trans. Inf. Theory*. Vol. 18, Number 4, July 1972, pp. 531–532.

[10] Dahlman, E., Parkvall, S., and Sköld, J.. *4G LTE / LTE-Advanced for Mobile Broadband*. Kidlington, Oxford: Academic Press, 2011. p. 112.

[11] Dent, P., G. E. Bottomley, and T. Croft. “Jakes Fading Model Revisited.” *Electronics Letters*. Vol. 29, 1993, Number 13, pp. 1162–1163.

[12] Nohrborg, Magdalena, for 3GPP. “LTE Overview.” *3GPP, A Global Initiative, THE Mobile Broadband Standard*, August 2013. http://www.3gpp.org/LTE.

[13] Pätzold, Matthias, Cheng-Xiang Wang, and Bjørn Olav Hogstad. “Two New Sum-of-Sinusoids-Based Methods for the Efficient Generation of Multiple Uncorrelated Rayleigh Fading Waveforms.” *IEEE Transactions on Wireless Communications*. Vol. 8, 2009, Number 6, pp. 3122–3131.

[14] Strang, Gilbert. *Linear Algebra and Its Application*. Academic Press, 1980. 2nd Edition.