

LTE System Toolbox™
Getting Started Guide



MATLAB®

R2017b

 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™ Getting Started Guide

© COPYRIGHT 2013–2017 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

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)
March 2015	Online only	Revised for Version 2.0 (Release 2015a)
September 2015	Online only	Revised for Version 2.1 (Release 2015b)
March 2016	Online only	Revised for Version 2.2 (Release 2016a)
September 2016	Online only	Revised for Version 2.3 (Release 2016b)
March 2017	Online only	Revised for Version 2.4 (Release 2017a)
September 2017	Online only	Revised for Version 2.5 (Release 2017b)

1	Getting Started with LTE System Toolbox Software	
	LTE System Toolbox Product Description	1-2
	Key Features	1-2
	What Is LTE?	1-3
	Long-Term Evolution	1-3
	LTE Releases	1-4
	LTE Physical Layer	1-7
	Limitations	1-12
	Code Generation Support	1-12
	Fixed-Point Support	1-12
	Block Support	1-12
	LTE-Advanced Functionality	1-13
	Release 9 Positioning Reference Signal	1-13
	Release 9 Dual-Layer UE-Specific Beamforming	1-14
	Release 10 Downlink Enhanced MIMO	1-15
	Release 10 Uplink MIMO	1-16
	Release 10 Spatial Orthogonal Resource Transmit Diversity (SORTD)	1-17
	Release 10 PUCCH Format 3	1-18
	Release 11 Enhanced Physical Downlink Control Channel (EPDCCH)	1-18
	Release 12 Carrier Aggregation	1-18
	Release 12 Alternative Codebook	1-18
	Data Structures	1-19
	Overview	1-19
	Multidimensional Arrays	1-19
	Creating an Empty Resource Grid	1-21

Resource Grid Indexing	1-23
Overview	1-23
Subframe Resource Grid Size	1-24
Creating an Empty Resource Grid	1-24
Resource Grid Indexing	1-25
Linear Indices and Subscripts	1-26
Converting Between Linear Indices and Subscripts	1-30
Multi-Antenna Linear Indices	1-30
Index Base	1-32
Resource Blocks	1-32
Parameterization	1-35
Parameter Structures	1-35
Create a Cell-Wide Settings Structure	1-36
Cell-Wide Parameters	1-37
Optional Output Formats	1-37
UL-SCH Parameterization	1-41
Set UL-SCH Parameters in Scalar Structure	1-41
Set UL-SCH Parameters in Structure Array	1-43
Obsolete LTE Toolbox Interface	1-46

High-Level Examples

2

Transmit-Receive Chain Processing	2-2
Transmit-Receive Chain	2-2

System Toolboxes

3

What Is a System Toolbox?	3-2
--	------------

Getting Started with LTE System Toolbox Software

- “LTE System Toolbox Product Description” on page 1-2
- “What Is LTE?” on page 1-3
- “Limitations” on page 1-12
- “LTE-Advanced Functionality” on page 1-13
- “Data Structures” on page 1-19
- “Resource Grid Indexing” on page 1-23
- “Parameterization” on page 1-35
- “UL-SCH Parameterization” on page 1-41
- “Obsolete LTE Toolbox Interface” on page 1-46

LTE System Toolbox Product Description

Simulate the physical layer of LTE and LTE-Advanced wireless communications systems

LTE System Toolbox provides standard-compliant functions and apps for the design, simulation, and verification of LTE and LTE-Advanced communications systems. The system toolbox accelerates LTE algorithm and physical layer (PHY) development, supports golden reference verification and conformance testing, and enables test waveform generation. With the system toolbox, you can configure, simulate, measure, and analyze end-to-end communications links. You can also create and reuse a conformance test bench to verify that designs, prototypes, and implementations comply with the LTE standard.

Key Features

- Standard-compliant models for LTE and LTE-Advanced (Releases 8, 9, 10, and 11)
- Link-level transmit and receive processing functions, support for downlink transmission modes 1 to 10, and reference designs, including coordinated multipoint (CoMP)
- Test models (E-TM) and reference measurement channel (RMC) for LTE, LTE-A, and UMTS waveform generation
- Interactive tools for conformance and BER testing
- Waveform transmission and reception with radio devices and instruments for over-the-air testing
- System and control parameter recovery from captured signals, including cell identity, MIB, and SIB1
- Channel estimation, synchronization, MIMO receiver functions, and propagation channel models

What Is LTE?

In this section...

“Long-Term Evolution” on page 1-3

“LTE Releases” on page 1-4

“LTE Physical Layer” on page 1-7

Long-Term Evolution

Long-Term Evolution (LTE) is the air interface supporting fourth generation cellular networks. LTE is specifically designed for packet data communications, where the emphasis of the technology is high spectral efficiency, high peak data rates, low latency, and frequency flexibility. The LTE specifications were developed by the Third Generation Partnership Project (3GPP).

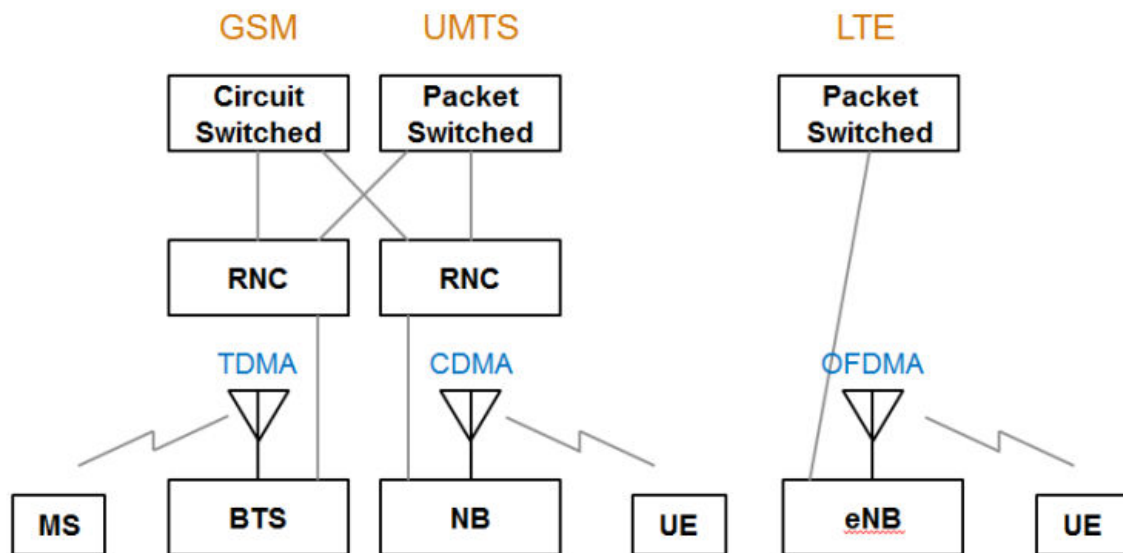
GSM and UMTS are the predecessors of the LTE air interface and are referred to as second generation (2G) and third generation (3G) technologies, respectively. GSM was developed as a circuit switched network meaning that radio services are configured at the user's request and resources remain allocated until terminated by the network controller. This type of operation is well suited to supporting voice calls. Eventually, GSM was enhanced to support low data rate services with packet switching capability but data rates were limited by GSM's air interface, time division multiple access (TDMA). In TDMA, each user is assigned to a particular channel (frequency band) and time slot which serves to limit capacity as the channel spacing is only 200 kHz.

UMTS uses code division multiple access (CDMA) as its air interface. In CDMA, active users transmit simultaneously over the allocated bandwidth, typically 5 MHz. Signals are separated from each other by the use of orthogonal variable spreading factor (OVSF) spreading codes. The advantage of OVSF codes is that resources can be allocated asymmetrically among the active users. UMTS supports both circuits switched services for voice calls and packet switched for data sessions. Due to its larger bandwidth and superior spectral efficiency, UMTS can support higher data rates than GSM.

Unlike GSM and UMTS, LTE is a purely packet switched network in which both voice and data services are carried by IP. LTE uses orthogonal frequency division multiple access (OFDMA) in which the spectrum is divided into resource blocks (RB) that are composed of twelve 15 kHz subcarriers. By dividing the spectrum in such a way, complicated equalizers are no longer necessary to mitigate frequency selective fading.

LTE supports higher order modulation schemes up to 64-QAM along with bandwidth allocations that can be as large as 20 MHz. In addition, LTE makes use of MIMO so that very high theoretical data rates can be achieved (75 Mbps in the uplink and 300 Mbps in the downlink for Release 8).

Second and third generation cellular networks consist of an interface to the public telephone or IP network, a radio network controller (RNC) that allocates radio resources among the users, a base station (referred to as a Node B in UMTS) that transmits and receives signals to and from the users, and user devices (MS for GSM and UE for UMTS). The LTE access network is similar with the exception that the RNC functionality has been pushed down into the enhanced Node B (eNB). The flatter architecture reduces the time required to establish data services resulting in lower latency. The architecture is shown below.



LTE Releases

Initially standardized in 3GPP Release 8, the LTE standards continue to evolve over multiple releases to capture requirements that lead to improved data throughput, lower latencies, and increasingly flexible configurations. After the release is frozen, 3GPP continues revisions of the associated standards to correct errors and fill in omissions, but no new features are introduced.

LTE Release 8

Release 8 introduced LTE for the first time. The Release 8 functionality set was frozen in the March 2009 standards release (SA#43). The release consisted of a completely new radio interface and core network, which enabled substantially improved data performance compared with previous systems. Highlights from Release 8 include:

- Up to 300 Mbps downlink and 75 Mbps uplink
- Latency as low as 10 ms
- Bandwidth sized in 1.4, 3, 5, 10, 15, or 20 MHz blocks to allow for a variety of deployment scenarios
- Orthogonal frequency domain multiple access (OFDMA) downlink
- Single-carrier frequency domain multiple access (SC-FDMA) uplink
- Multiple-input multiple-output (MIMO) antennas
- Flat radio network architecture, with no equivalent to the GSM base station controller (BSC) or UMTS radio network controller (RNC), and functionality distributed among the base stations (enhanced NodeBs)
- All IP core network, the System Architecture Evolution (SAE)

LTE Release 9

The Release 9 functionality set was frozen in the March 2010 standards release (SA#47). LTE Release 9 brought refinements to LTE Release 8, and it introduced some new service features and network architecture improvements. Highlights from Release 9 include:

- Evolved multimedia broadcast and multicast service (eMBMS) for the efficient delivery of the same multimedia content to multiple destinations
- Location services (LCS) to pinpoint the location of a mobile device by using assisted GPS (A-GPS), observed time difference of arrival (OTDOA), enhanced cell-ID (E-CID).
- Dual layer beamforming

LTE Release 10

The Release 10 functionality set was frozen in the June 2011 standards release (SA#52). LTE Release 10 is considered to be the start of LTE-Advanced. It significantly improved data throughput and extended cell coverage. Highlights from Release 10 include:

- Higher order MIMO antenna configurations supporting up to 8×8 downlinks and 4×4 uplinks
- Data throughput of up to 3 Gbps downlink and 1.5 Gbps uplink
- Carrier aggregation (CA), allowing the combination of up to five separate carriers to enable bandwidths of up to 100 MHz
- Relay nodes to support Heterogeneous Networks (HetNets) containing a wide variety of cell sizes
- Enhanced intercell interference coordination (eICIC) to improve performance toward the edge of cells

LTE Release 11

The Release 11 functionality set was frozen in the March 2013 standards release (SA#59). LTE Release 11 included refinements to existing Release 10 capabilities, including:

- Enhancements to Carrier Aggregation, MIMO, relay nodes, and eICIC.
- Coordinated multipoint transmission and reception (CoMP) to enable simultaneous communication with multiple cells.
- Enhanced PDCCH (EPDCCH), which uses PDSCH resources for transmitting control information. Previously, from Release 8, control information could only be transmitted in the PDCCH region of subframes.
- Introduction of new frequency bands.

LTE Release 12

The Release 12 functionality set was frozen in the March 2015 standards release (SA#67). Highlights from Release 12 include:

- Enhanced small cells for LTE, introducing a number of features to improve the support of HetNets.
- Intersite carrier aggregation to coordinate the capabilities and backhaul of adjacent cells.
- Machine-to-machine (M2M) communication, also referred to as machine-type communication (MTC).
- Device-to-device (D2D) interface to support public safety communications systems, and proximity services (ProSe) for discovery and group communications. The LTE D2D interface is called a *sidelink*.

- Interworking between LTE and WiFi or HSPDA.
- Higher order modulation schemes of up to 64-QAM.
- LTE operation in unlicensed spectrum.

LTE Release 13

The LTE standard releases from 13 onwards are known as LTE-Advanced Pro. Highlights from Release 13 include:

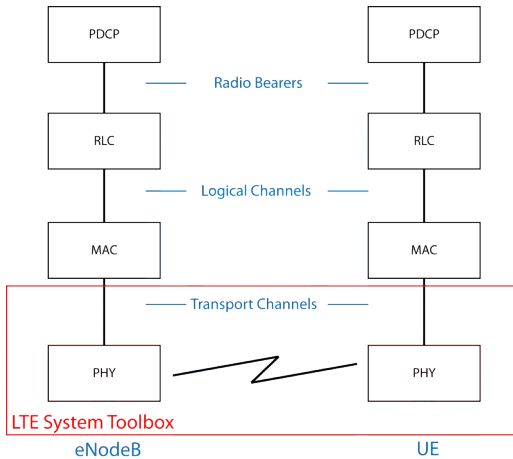
- New working group introduced for programming of mission critical applications.
- Carrier aggregation (CA), allowing the combination of up to 32 separate carriers to enable bandwidths of up to 640 MHz.
- LTE operation in a combination of licensed and unlicensed spectrum.
- Intersite carrier aggregation, to coordinate the capabilities and backhaul of adjacent cells.
- Enhanced machine-to-machine (M2M) communication, also referred to as machine-type communication (MTC).
- Interworking with Wi-Fi, licensed assisted access (at 5 GHz).
- Indoor positioning.
- Further enhancements of public safety features, such as D2D and ProSe, which include small-cell dual-connectivity and architecture changes.
- Single-cell point to multipoint.
- New antenna techniques, such as 3D/FD-MIMO, which include study of high-order MIMO systems with up to 64 antenna ports.
- Advanced receivers to maximize the potential of large cells.
- Work on latency reduction.

LTE Physical Layer

The LTE radio access network is comprised of the following protocol entities.

- Packet Data Convergence Protocol (PDCP)
- Radio Link Control (RLC)
- Medium Access Control (MAC)
- The Physical Layer (PHY)

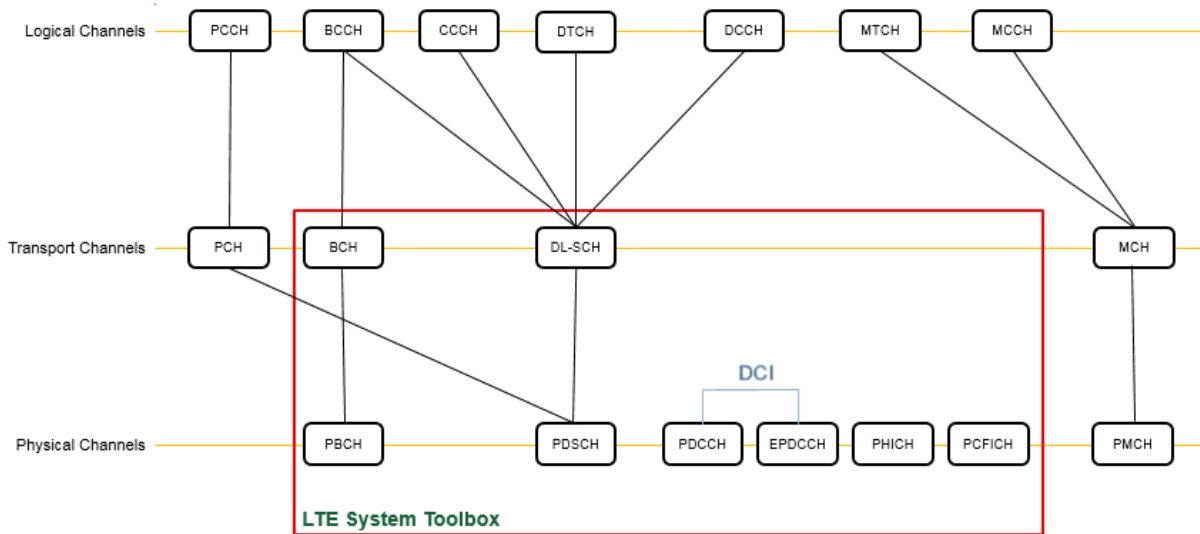
The first three protocol entities handle tasks such as header compression, ciphering, segmentation and concatenation, and multiplexing and demultiplexing. The physical layer handles coding and decoding, modulation and demodulation, and antenna mapping. The figure shows the delineation between the physical layer and higher layers.



LTE System Toolbox focuses on the physical layer, which is highlighted in red in the preceding figure. It also supports interfacing with portions of the RLC and MAC layers, which are highlighted in blue. The primary features of the LTE physical layer are OFDM modulation, including the time-frequency structure of the resource blocks, adaptive modulation and coding, hybrid-ARQ, and MIMO.

Downlink Channel Mapping

System downlink data follows the indicated mapping between logical channels, transport channels, and physical channels. The red outline contains LTE System Toolbox downlink functionality for physical channels, transport channels, and control information.

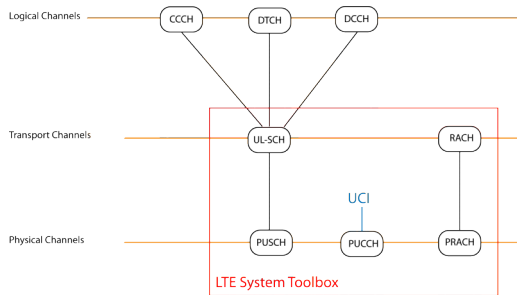


For more information, see “Downlink Channels” or the specific channel category of interest:

- “Physical Signals”
- “Physical Channels”
- “Control Information”
- “Transport Channels”

Uplink Channel Mapping

System uplink data follows the indicated mapping between logical channels, transport channels, and physical channels. The red outline contains LTE System Toolbox uplink functionality for physical channels, transport channels, and control information.

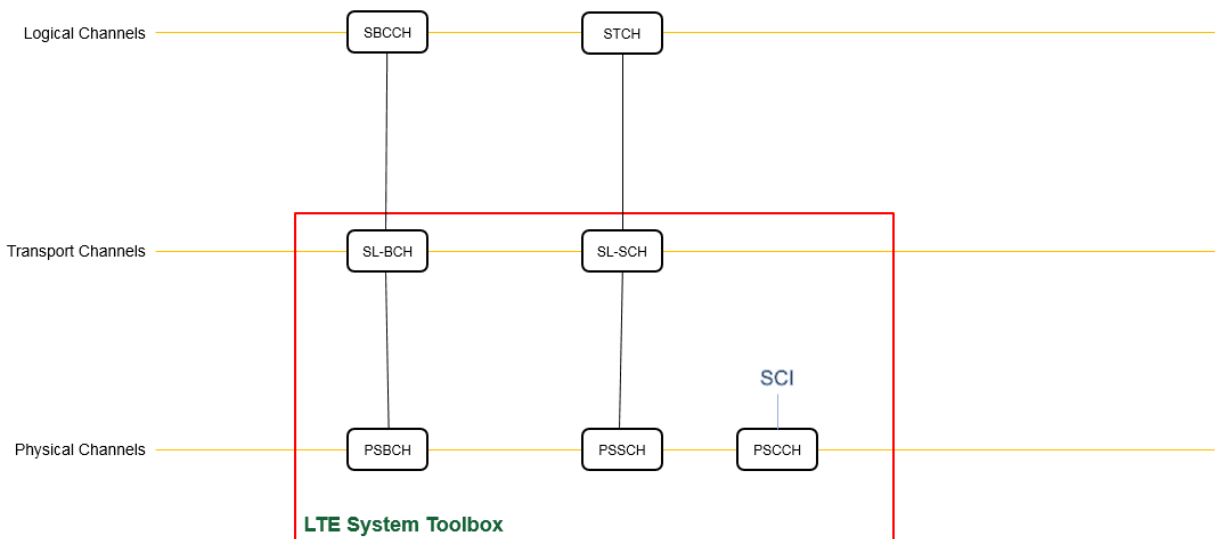


For more information, see “Uplink Channels” or the specific channel category of interest:

- “Physical Signals”
- “Physical Channels”
- “Transport Channels”
- “Control Information”

Sidelink Channel Mapping

System sidelink data follows the indicated mapping between logical channels, transport channels, and physical channels. The red outline contains LTE System Toolbox sidelink functionality for physical channels, transport channels, and control information.



For more information, see “Sidelink Channels” or the specific channel category of interest:

- “Physical Signals”
- “Physical Channels”
- “Transport Channels”
- “Control Information”

References

- [1] Nohrborg, Magdalena, for 3GPP. “LTE Overview.” *3GPP, A Global Initiative, THE Mobile Broadband Standard*, August 2013. <http://www.3gpp.org/LTE>.
- [2] Dahlman, E., Parkvall, S., and Sköld, J.. *4G LTE / LTE-Advanced for Mobile Broadband*. Kidlington, Oxford: Academic Press, 2011. pp. 112–118.

Limitations

Code Generation Support

LTE System Toolbox does not support automatic generation of C code or HDL code.

Fixed-Point Support

LTE System Toolbox does not support fixed-point data types.

Block Support

LTE System Toolbox does not contain Simulink® blocks.

LTE-Advanced Functionality

In this section...

“Release 9 Positioning Reference Signal” on page 1-13

“Release 9 Dual-Layer UE-Specific Beamforming” on page 1-14

“Release 10 Downlink Enhanced MIMO” on page 1-15

“Release 10 Uplink MIMO” on page 1-16

“Release 10 Spatial Orthogonal Resource Transmit Diversity (SORTD)” on page 1-17

“Release 10 PUCCH Format 3” on page 1-18

“Release 11 Enhanced Physical Downlink Control Channel (EPDCCH)” on page 1-18

“Release 12 Carrier Aggregation” on page 1-18

“Release 12 Alternative Codebook” on page 1-18

The LTE System Toolbox supports enhancements to the LTE Release 8 and 9 offerings. LTE-Advanced builds upon these earlier releases. A brief description of significant Release 9 updates is provided here prior to discussing the LTE-Advanced functionality. 3GPP defines LTE-Advanced functionality in Releases 10, 11, and 12 of the LTE Standard.

Release 9 Positioning Reference Signal

Release 9 defines several changes to the provision for positioning within the LTE standard. These changes, which enable the network to compute the position of the UE, include:

- UE reception of a new downlink positioning reference signal (PRS) transmitted by the eNodeB.
- Transmission of the time difference of arrival to the eNodeB as a measurement.

LTE System Toolbox supports the PRS with the `ltePRS` and `ltePRSIndices` functions.

To learn how to use the PRS to perform time-difference of arrival (TDOA) position estimation, see “Time Difference Of Arrival Positioning Using PRS”.

Release 9 Dual-Layer UE-Specific Beamforming

Release 9 provides a dual-layer UE-specific beamforming mode. It defines two UE-specific reference signals (antenna ports 7 and 8). Two independent streams of data can be sent, one on each layer. These streams of data can be to a single UE (one rank 2 transmission) or to two UEs (two rank 1 transmissions).

LTE System Toolbox supports the antenna port 5, 7, and 8 reference signals with the `lteDMRS` and `lteDMRSIndices` functions. These functions support the transmission of UE-specific reference signals for Release 8, 9 and 10. The particular UE-specific reference signals created are controlled by the transmission scheme parameter, `TxScheme`. For Release 9, you have the option to set `TxScheme` to these transmission schemes.

Parameter Setting	Description
'Port7-8'	Release 9 single-antenna port, port 7 (if <code>NLayers=1</code>)
	Release 9 dual layer transmission, ports 7 and 8 (if <code>NLayers=2</code>)
'Port8'	Single-antenna port, port 8

PDSCH transmissions, associated with antenna ports 7 and 8 (or any transmission scheme), can be made using the `ltePDSCH` and `ltePDSCHIndices` functions. These functions accept settings for the `TxScheme` parameter as described in the preceding table.

The UE-specific beamforming of the reference signals and PDSCH transmission is specified by the parameter `W` provided to `lteDMRS` and `ltePDSCH`. The `lteDMRSIndices` and `ltePDSCHIndices` functions use the `NTxAnts` parameter to specify the number of transmission antennas. See the function reference pages for details.

At the receiver, `ltePDSCHDecode` decodes PDSCH transmissions made on ports 7 and 8, under the assumption that the input will be equalized back to the transmission layers. Hence, no deprecoding is required. This behavior is consistent with the operation of `lteDLChannelEstimate`, which cannot assume knowledge of the UE-specific beamforming used at the transmitter when it produces the channel matrices between transmission layers and receive antennas. Therefore, the MMSE equalization carried out within `ltePDSCHDecode` outputs the PDSCH layers, which are then layer demapped, demodulated, and descrambled to produce soft bit estimates.

Note Several other functions are aware of the transmission scheme and process signals in compliance with Release 9 UE-specific beamforming, including `lteDLDeprecode`, `lteDLPrecode`, `lteDLSCH`, `lteDLSCHDecode`, `lteRateMatchTurbo`, and `lteRateRecoverTurbo`.

Release 9 transmissions on antenna ports 7 and 8 are associated with DCI Format 2B, which is supported by the `lteDCI`, `lteDCIDecode`, `lteDCIInfo`, `ltePDCCHSearch`, and `lteDCIResourceAllocation` functions.

Release 10 Downlink Enhanced MIMO

Release 10 provides a further extension to downlink UE-specific beamforming with reference signals (antenna ports) for up to 8 layers. These reference signals are called demodulation reference signals (DM-RS) in the standard. To support channel estimation for up to 8 layers (noting the cell-specific reference signals support only 4 antenna ports) a new channel state information reference signal (CSI-RS) set has been added, with 8 antenna ports specifically designed for CSI estimation.

The DM-RS antenna ports are numbers 7 through 14, with ports 7 and 8 being compatible with the dual-layer UE-specific beamforming capability of Release 9. LTE System Toolbox supports these reference signals with the `lteDMRS` and `lteDMRSIndices` functions. These functions support the transmission of UE-specific reference signals for Release 8, 9 and 10. The particular UE-specific reference signals created are controlled by the transmission scheme parameter, `TxScheme`. For Release 10, you have the option to set `TxScheme` to this transmission scheme.

Parameter Setting	Description
'Port7-14'	Release 10 up to 8-layer transmission, ports 7–14 (<code>NLayers=1...8</code>)

PDSCH transmissions, associated with antenna ports 7 through 14 (or any transmission scheme), can be made using the `ltePDSCH` and `ltePDSCHIndices` functions. These functions accept settings for the `TxScheme` parameter as described in the preceding table.

The UE-specific beamforming of the reference signals and PDSCH transmission is specified by the parameter `W` provided to `lteDMRS` and `ltePDSCH`. The `lteDMRSIndices` and `ltePDSCHIndices` functions use the `NTxAnts` parameter to specify the number of transmission antennas. See the function reference pages for details.

At the receiver, `ltePDSCHDecode` decodes PDSCH transmissions made on ports 7 through 14, under the assumption that the input will be equalized back to the transmission layers. Hence, no deprecoding is required. This behavior is consistent with the operation of `lteDLChannelEstimate`, which cannot assume knowledge of the UE-specific beamforming used at the transmitter when it produces the channel matrices between transmission layers and receive antennas. Therefore, the MMSE equalization carried out within `ltePDSCHDecode` outputs the PDSCH layers, which are then layer demapped, demodulated, and descrambled to produce soft bit estimates.

For PMI feedback, `lteDLChannelEstimate` can optionally perform channel estimation against the CSI-RS. To do so, set the `Reference` parameter to `'CSIRS'`. Then, provide this channel estimate to `ltePMISelect` to perform PMI selection based on the codebook for CSI reporting, which is implemented using the `lteCSICodebook` function.

Note Several other functions are aware of the transmission scheme and process signals in compliance with Release 10, including `lteDLDeprecode`, `lteDLPrecode`, `lteDLSCH`, `lteDLSCHDecode`, `lteRateMatchTurbo`, and `lteRateRecoverTurbo`.

Release 10 transmissions on antenna ports 7 through 14 are associated with DCI Format 2C, which is supported by the `lteDCI`, `lteDCIDecode`, `lteDCIInfo`, `ltePDCCHSearch`, and `lteDCIResourceAllocation` functions.

Release 10 Uplink MIMO

Release 10 supports uplink MIMO, with 2 codewords transmitted on up to 4 layers on 4 antennas for the PUSCH. LTE System Toolbox supports uplink MIMO similar to how it supports downlink MIMO, using cell arrays to represent multiple codeword vectors, and using multiple column matrices to represent multiple layers and transmission antennas.

Uplink MIMO transmission is provided by the `ltePUSCH` and `lteULSCH` functions. In the receiver, the timing offset function, `lteULFrameOffset`, searches its input across all configured DM-RS signals. By default, `lteULChannelEstimate` provides channel estimates to the precoded DM-RS signals, or transmission antennas, and `ltePUSCHDecode` uses knowledge of the precoding matrices used to perform MIMO equalization. Alternatively, you can configure `lteULChannelEstimate` to provide channel estimates to the DRS layers. To do so, set the `Reference` parameter to `'Layers'`. In this case, `ltePUSCHDecode` equalizes back to transmission layers.

The `ltePUSCHPrecode` and `ltePUSCHDeprecode` functions perform MIMO precoding and deprecoding for the PUSCH.

The `lteLayerMap` and `lteLayerDemap` functions provide support for the uplink and downlink.

The `lteACKDecode`, `lteACKEncode`, `lteRIDecode`, and `lteRIEncode` functions support the increased number of bits that can be coded in Release 10.

To learn how to create and simulate an uplink MIMO PUSCH performance test, see “Release 10 PUSCH Multiple Codeword Transmit and Receive Modeling”.

Release 10 Spatial Orthogonal Resource Transmit Diversity (SORTD)

Release 10 incorporates spatial orthogonal resource transmit diversity (SORTD) transmission on the PUCCH and SRS channels. SORTD transmits independent versions of an encoded and modulated signal on each transmission antenna by using a different orthogonal resource for each transmission antenna. For the PUCCH, the different orthogonal resources are different PUCCH resource indices, $n1_{\text{PUCCH}}$, $n2_{\text{PUCCH}}$, and $n3_{\text{PUCCH}}$. For the SRS, the different orthogonal resources are different reference signal cyclic shifts, *alpha*.

SORTD transmission is supported by the `ltePUCCH1`, `ltePUCCH1DRS`, `ltePUCCH1DRSIndices`, `ltePUCCH1Indices`, `ltePUCCH2`, `ltePUCCH2DRS`, `ltePUCCH2DRSIndices`, `ltePUCCH2Indices`, `lteSRS`, and `lteSRSIndices` functions.

For PUCCH formats 1 and 2 and their DM-RS signals, specify SORTD using the `ResourceIdx` parameter. For Release 10 this parameter is a vector of indices, rather than a scalar index as for Release 8. For the SRS, specify SORTD using the `NTxAnts` parameter.

In the receiver, the timing offset functions, `lteULFrameOffsetPUCCH1` and `lteULFrameOffsetPUCCH2`, search their input across all configured DM-RS signals. The channel estimators, `lteULChannelEstimatePUCCH1` and `lteULChannelEstimatePUCCH2`, make a channel estimate against all DM-RS signals, or transmission antennas. If using a pilot averaging frequency window size, orthogonal despreading of different DM-RS signals is supported. The pilot averaging frequency window size is always a multiple of 12.

Release 10 PUCCH Format 3

Release 10 introduces a new PUCCH format, format 3, designed to transmit a large number of ACK indications in a single subframe. The LTE System Toolbox implements PUCCH format 3 with the `ltePUCCH3`, `ltePUCCH3Decode`, `ltePUCCH3DRS`, `ltePUCCH3DRSIndices`, `ltePUCCH3Indices`, `ltePUCCH3PRBS`, `lteULChannelEstimatePUCCH3` and `lteULFrameOffsetPUCCH3` functions.

Release 11 Enhanced Physical Downlink Control Channel (EPDCCH)

Release 11 introduces the Enhanced Physical Downlink Control Channel, EPDCCH, which is designed to achieve improved spectral reuse of control channel resources. It supports CoMP, downlink MIMO, beamforming and frequency domain inter-cell interference coordination (ICIC). LTE System Toolbox implements EPDCCH with the `lteEPDCCH`, `lteEPDCCHIndices`, `lteEPDCCHDMRS`, `lteEPDCCHDMRSIndices`, and `lteEPDCCHPRBS` functions.

Coordinated multipoint (CoMP) operation in LTE Release 11 takes advantage of low latency and high capacity backhaul between base stations within a cooperating set. For a demonstration of CoMP in a dynamic point selection (DPS) scheme, see “CoMP Dynamic Point Selection with Multiple CSI Processes”.

Release 12 Carrier Aggregation

Release 12 introduced intersite carrier aggregation to coordinate the capabilities and backhaul of adjacent cells. For a demonstration on how to create a signal covering multiple LTE carriers using carrier aggregation, see “Release 12 Downlink Carrier Aggregation Waveform Generation, Demodulation and Analysis”.

Release 12 Alternative Codebook

Release 12 introduced an alternative codebook for channel state information reporting. For more information, see `lteCSICodebook`.

Data Structures

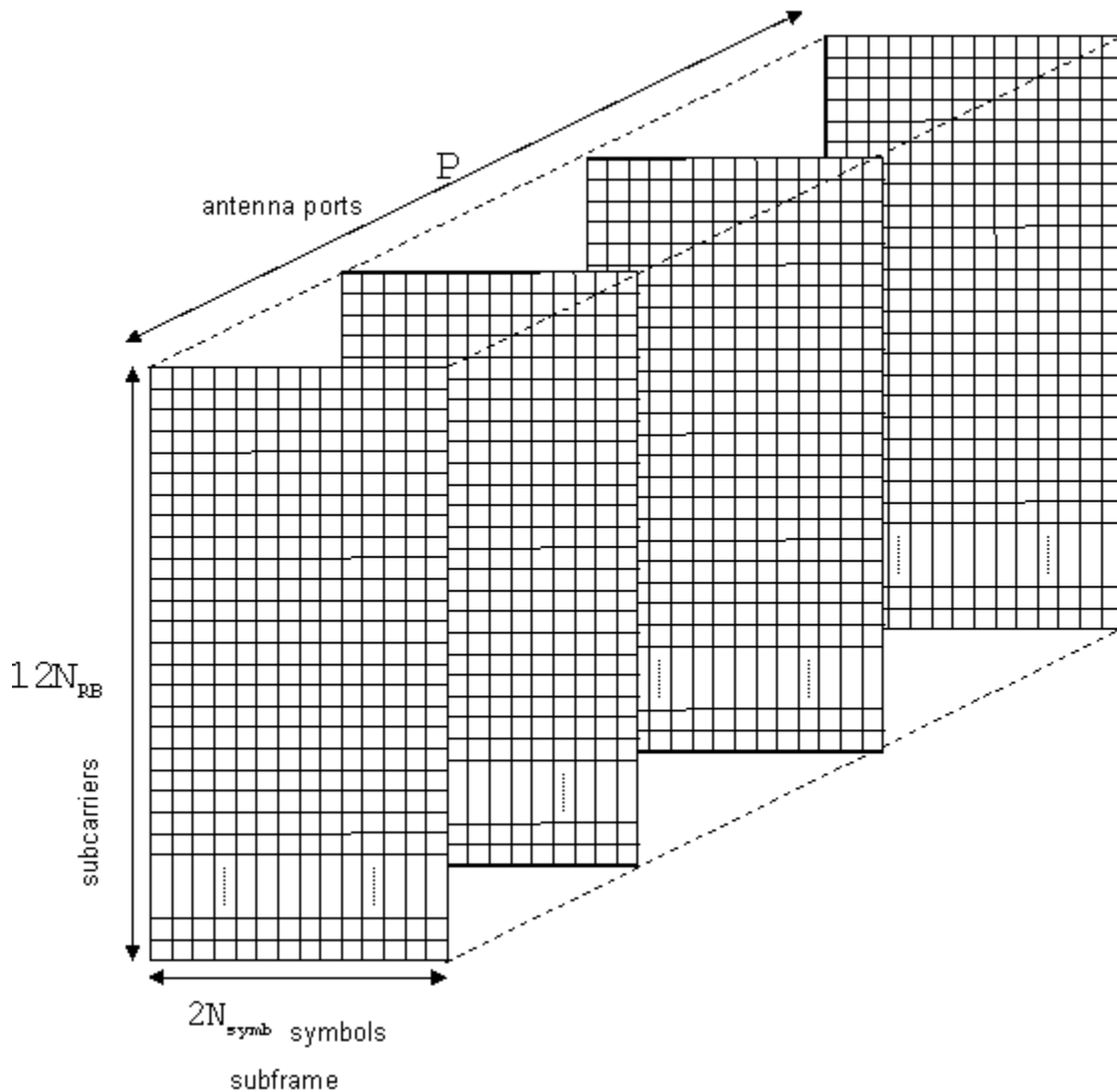
In this section...
“Overview” on page 1-19
“Multidimensional Arrays” on page 1-19
“Creating an Empty Resource Grid” on page 1-21

Overview

This section describes the data structures used to represent the resource grid in the LTE System Toolbox.

Multidimensional Arrays

Before OFDM modulation (IFFT), the physical channels and signals in LTE are mapped to different portions of the resource grid. LTE System Toolbox represents the resource grid as a multidimensional array.



The rows of this array represent the subcarrier. The columns map the OFDM or SC-FDMA symbols in the downlink and uplink, respectively. The third dimension or plane represents the antenna ports. In the LTE System Toolbox, the resource grid spans a subframe in the time-domain, instead of a slot. Hence, the documentation uses the term

subframe resource grid. The size of this multidimensional array is $12N_{RB} \times 2N_{symp} \times P$, where N_{RB} is the number of resource blocks spanning the available bandwidth, N_{symp} is the number of OFDM, or SC-FDMA in the uplink, symbols per slot, and P is the number of antenna ports. Therefore, the resource grid represents a subframe, two slots, and whole bandwidth since there are 12 subcarriers per resource block. For the single antenna case, you can work with a two-dimensional array of size $12N_{RB} \times 2N_{symp}$.

Creating an Empty Resource Grid

Create an empty downlink resource grid using two different methods. Valid and equivalent subframe resource grids can be created using the `lteDLResourceGrid` function or the MATLAB® `zeros` function.

Initialize required parameters

Create the parameter structure for normal cyclic prefix, nine downlink resource blocks, and one transmit antenna. Also define seven symbols per slot for use in the `zeros` function only.

```
enb.CyclicPrefix = 'Normal';
enb.NDLRB = 9;
enb.CellRefP = 1;
noSymbolsSlot = 7;
```

Create two empty resource grids

Create an empty subframe resource grid, using each method.

```
resourceGrid1 = lteDLResourceGrid(enb);
resourceGrid2 = zeros(enb.NDLRB*12, noSymbolsSlot*2, enb.CellRefP);
```

Confirm the grids are equal

Compare the two grid variables for equality using the MATLAB `isequal` function.

```
isequal(resourceGrid1, resourceGrid2)
```

```
ans =
    logical
     1
```

Both approaches generate the same result. Use either approach to create an empty downlink resource grid. Similarly, an empty uplink resource grid could be created using `lteULResourceGrid` function or the MATLAB `zeros` function.

See Also

`isequal` | `lteDLResourceGrid` | `lteResourceGrid` | `lteULResourceGrid` | `zeros`

Related Examples

- “Resource Grid Indexing” on page 1-23

Resource Grid Indexing

In this section...

“Overview” on page 1-23

“Subframe Resource Grid Size” on page 1-24

“Creating an Empty Resource Grid” on page 1-24

“Resource Grid Indexing” on page 1-25

“Linear Indices and Subscripts” on page 1-26

“Converting Between Linear Indices and Subscripts” on page 1-30

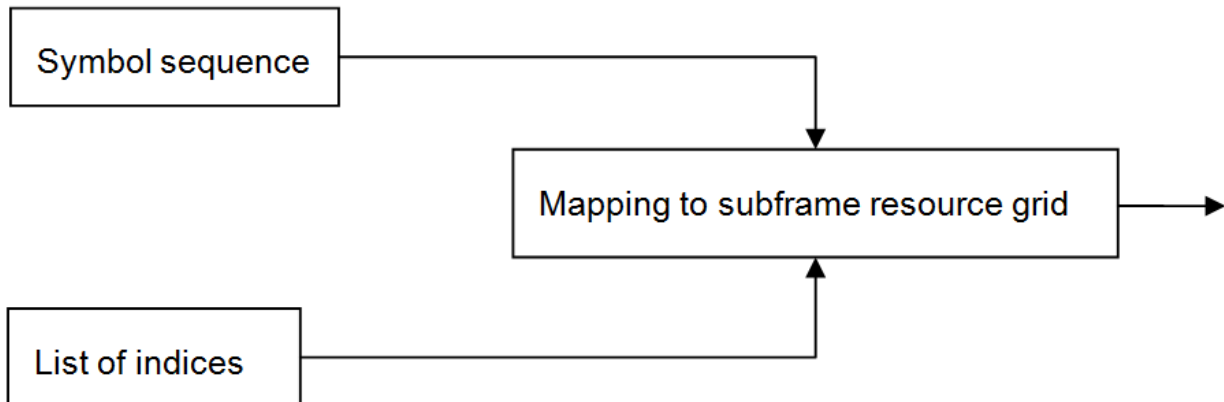
“Multi-Antenna Linear Indices” on page 1-30

“Index Base” on page 1-32

“Resource Blocks” on page 1-32

Overview

The LTE System Toolbox provides facilities to generate sequences of symbols corresponding to the physical channels and signals. Indices for the mapping of these sequences to the resource grid are also generated. For convenience, the LTE System Toolbox uses the MATLAB® linear indexing style to represent these indices.



Subframe Resource Grid Size

Before applying OFDM modulation (IFFT), the physical channels and signals in LTE are mapped to different portions of the subframe resource grid. The subframe resource grid is represented in the LTE System Toolbox as a multidimensional array of the following size.

$$12N_{RB} \times 2N_{symp} \times P$$

In the preceding expression, N_{RB} is the number of resource blocks spanning the available bandwidth, N_{symp} is the number of OFDM (or SC-FDMA in the uplink) symbols per slot, and P is the number of antenna ports. Therefore, the resource grid represents a subframe (two slots) and whole bandwidth, since there are 12 subcarriers per resource block. For the single antenna case, a resource grid can be a two-dimensional matrix of the following size.

$$12N_{RB} \times 2N_{symp}$$

Creating an Empty Resource Grid

Create an empty downlink resource grid using two different methods. Valid and equivalent subframe resource grids can be created using the `lteDLResourceGrid` function or the MATLAB® `zeros` function.

Initialize required parameters

Create the parameter structure for normal cyclic prefix, nine downlink resource blocks, and one transmit antenna. Also define seven symbols per slot for use in the `zeros` function only.

```
enb.CyclicPrefix = 'Normal';  
enb.NDLRB = 9;  
enb.CellRefP = 1;  
noSymbolsSlot = 7;
```

Create two empty resource grids

Create an empty subframe resource grid, using each method.

```
resourceGrid1 = lteDLResourceGrid(enb);  
resourceGrid2 = zeros(enb.NDLRB*12, noSymbolsSlot*2, enb.CellRefP);
```

Confirm the grids are equal

Compare the two grid variables for equality using the MATLAB `isequal` function.

```
isequal(resourceGrid1,resourceGrid2)
```

```
ans =  
  
    logical  
  
     1
```

Both approaches generate the same result. Use either approach to create an empty downlink resource grid. Similarly, an empty uplink resource grid could be created using `lteULResourceGrid` function or the MATLAB `zeros` function.

Resource Grid Indexing

Generate a reference signal and map it to an empty resource grid for the single antenna case. The LTE System Toolbox™ has been designed to facilitate the mapping of physical channels and signals in the resource grid.

Set up the cell-wide settings. Create a structure and specify the cell-wide settings as its fields.

```
enb.CyclicPrefix = 'Normal';  
enb.NDLRB = 6;  
enb.CellRefP = 1;  
enb.NCellID = 1;  
enb.NSubframe = 0;  
enb.DuplexMode = 'FDD';  
antPort = 0;
```

The `enb` structure now contains the parameters required by the functions to be called next.

Use `lteDLResourceGrid` to create an empty subframe resource grid, then populate it with reference symbols. To do so, call the `lteCellRSIndices` and `lteCellRS` functions.

```
resourceGrid = lteDLResourceGrid(enb);  
ind = lteCellRSIndices(enb,antPort);
```

```
rs = lteCellRS(enb,antPort);  
resourceGrid (ind) = rs;
```

A call to the function `lteCellRSIndices` generates a list of indices identifying to where the reference signal is to be mapped, whereas the call to `lteCellRS` generates the reference signal symbols.

Linear Indices and Subscripts

Generate indices in linear and subscript form. All of the LTE System Toolbox™ index generation functions can produce either linear or subscript formats by setting the appropriate options string. The default is linear indexing style, which allows access to any element of a matrix with a single index value. Subscripted matrix element indexing is also available. Using subscripted indexing on a 2-D matrix, you can access each element with a set of two elements representing the row and column equivalents.

Note, the linear indexing style allows you to conveniently map the reference sequence symbols to the appropriate location in the resource grid with just one line of code. Mapping reference symbols to the resource grid using subscripted indices would require more finesse.

Create a structure specifying the cell-wide settings as its fields. Assign zero as the antenna port number.

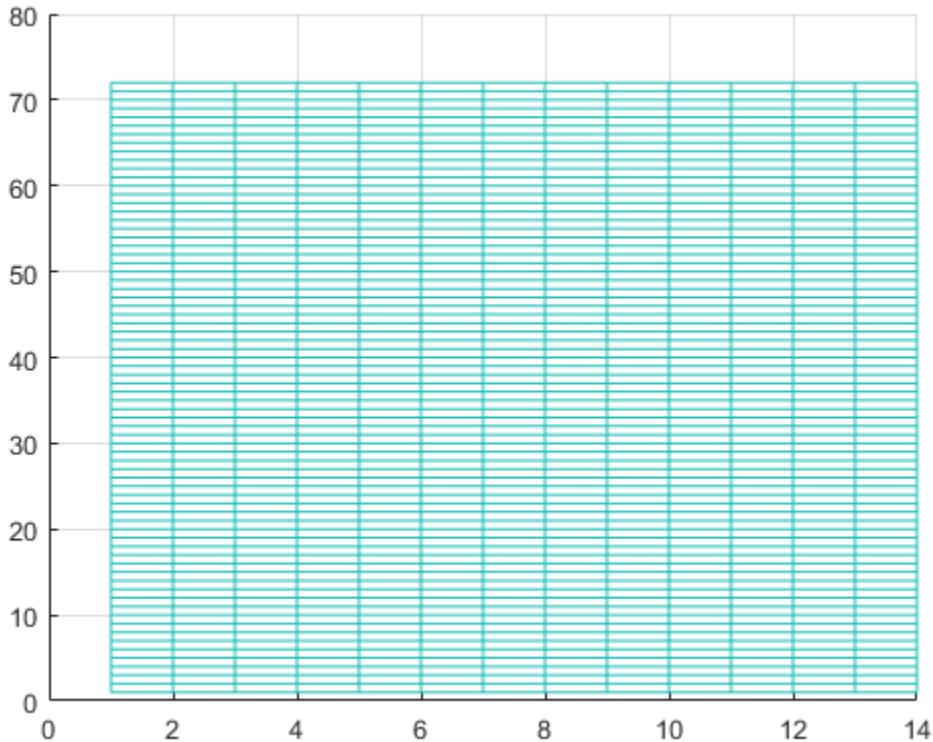
```
enb.CyclicPrefix = 'Normal';  
enb.NDLRB = 6;  
enb.CellRefP = 1;  
enb.NCellID = 1;  
enb.NSubframe = 0;  
enb.DuplexMode = 'FDD';  
  
antPort = 0;
```

The `enb` structure now contains the parameters required by the functions to be called next.

Use `lteDLResourceGrid` to create an empty subframe resource grid and `lteCellRS` to create reference signal symbols. View the empty resource grid.

```
resourceGrid = lteDLResourceGrid(enb);  
rs = lteCellRS(enb,antPort);
```

```
mesh(abs(resourceGrid))  
view(2)
```



After the indices are generated and the reference symbols are mapped to the resource grid, the resource grid is replotted to show the reference symbol mapping.

Generate linear indices. Since linear indexing is the default, the following two calls are equivalent.

```
ind_lin = lteCellRSIndices(enb,antPort);  
ind_lin = lteCellRSIndices(enb,antPort,'ind')
```

```
ind_lin = 48x1 uint32 column vector
```

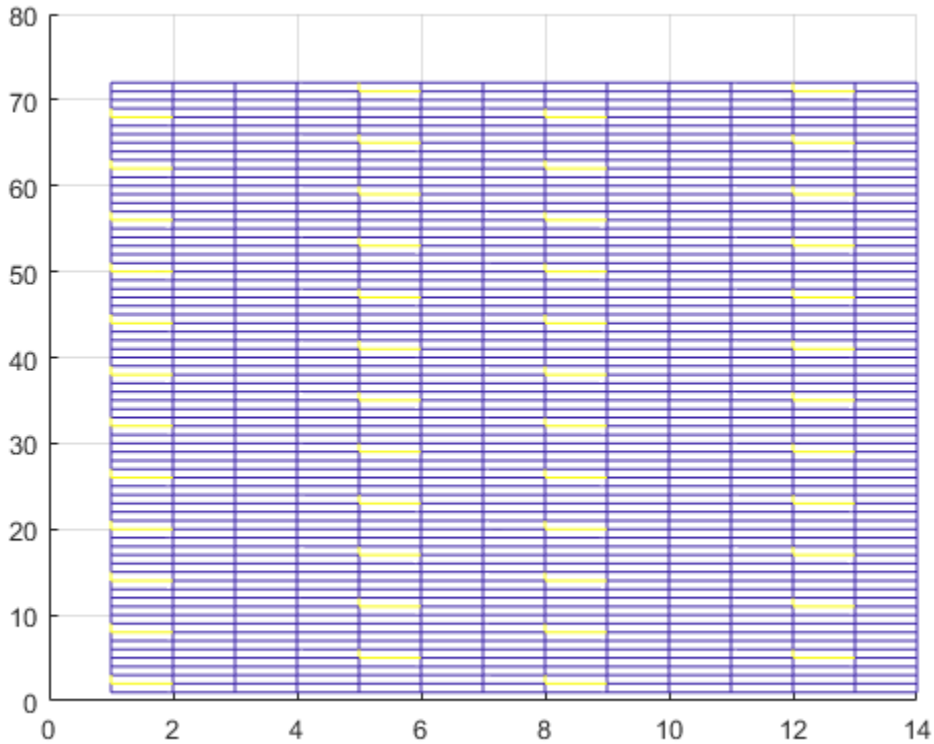
```
2  
8  
14  
20  
26  
32  
38  
44  
50  
56
```

Mapping the reference signal symbols to the resource grid using linear indices, `ind_lin`, simply requires you to assign the reference signal symbols to the resource grid indicating the active indices.

```
resourceGrid(ind_lin) = rs;
```

Replotting the resource grid shows the reference symbols active for the cell-wide settings assigned in `enb`.

```
figure  
mesh(abs(resourceGrid))  
view(2)
```

Alternatively, generate indices in subscript form by providing the 'sub' option string for `lteCellRSIndices`.

```
ind_sub = lteCellRSIndices(enb,antPort,'sub')
```

```
ind_sub = 48x3 uint32 matrix
```

```

     2     1     1
     8     1     1
    14     1     1
    20     1     1
    26     1     1
    32     1     1
    38     1     1
```

```
44    1    1
50    1    1
56    1    1
```

In this case, the output argument, `ind_sub`, is a [subcarrier, OFDM symbol, antenna port] matrix format of the resource grid. These indices are calculated for antenna port 0.

Converting Between Linear Indices and Subscripts

Conversion between linear indices and subscripts can be achieved using the MATLAB® `ind2sub` and `sub2ind` functions. Alternatively, all index generation functions in the LTE System Toolbox can produce both formats.

Multi-Antenna Linear Indices

Generate indices in multi-antenna linear form. This form is a variant of the MATLAB® linear indexing style in which the indices corresponding for each antenna port are in a different column. However, all indices are still in linear form. Several toolbox functions return indices in multi-antenna linear form.

To illustrate this, call the function `ltePDSCH` for the four antenna case.

```
enb.CellRefP = 4;
enb.CFI = 1;
enb.NCellID = 1;
enb.NSubframe = 0;
enb.NDLRB = 6;
enb.CyclicPrefix = 'Normal';
enb.DuplexMode = 'FDD';

pdsch.TxScheme = 'TxDiversity';
pdsch.Modulation = 'QPSK';
pdsch.RNTI = 1;
pdsch.PRBSset = (0:5).';

data = ones(768,1);
symb = ltePDSCH(enb,pdsch,data);
size(symb)

ans =
```

```
384      4
```

```
symb(1:10, :)
```

```
ans =
-0.5000 - 0.5000i    0.0000 + 0.0000i   -0.5000 - 0.5000i    0.0000 + 0.0000i
 0.5000 - 0.5000i    0.0000 + 0.0000i   -0.5000 + 0.5000i    0.0000 + 0.0000i
 0.0000 + 0.0000i   -0.5000 - 0.5000i    0.0000 + 0.0000i    0.5000 - 0.5000i
 0.0000 + 0.0000i   -0.5000 - 0.5000i    0.0000 + 0.0000i   -0.5000 + 0.5000i
 0.5000 - 0.5000i    0.0000 + 0.0000i    0.5000 + 0.5000i    0.0000 + 0.0000i
-0.5000 + 0.5000i    0.0000 + 0.0000i    0.5000 + 0.5000i    0.0000 + 0.0000i
 0.0000 + 0.0000i    0.5000 + 0.5000i    0.0000 + 0.0000i   -0.5000 + 0.5000i
 0.0000 + 0.0000i    0.5000 + 0.5000i    0.0000 + 0.0000i    0.5000 - 0.5000i
-0.5000 - 0.5000i    0.0000 + 0.0000i    0.5000 + 0.5000i    0.0000 + 0.0000i
-0.5000 + 0.5000i    0.0000 + 0.0000i   -0.5000 + 0.5000i    0.0000 + 0.0000i
```

The output argument, `symb`, is a matrix with four columns, in which each column corresponds to each antenna port.

In a similar format, generate the indices for the PDSCH by calling `ltePDSCHIndices`.

```
pdschIndices = ltePDSCHIndices(enb,pdsch,pdsch.PRBSets);
size(pdschIndices)
```

```
ans =
```

```
384      4
```

```
pdschIndices(1:10, :)
```

```
ans = 10x4 uint32 matrix
```

```
145    1153    2161    3169
146    1154    2162    3170
147    1155    2163    3171
148    1156    2164    3172
149    1157    2165    3173
150    1158    2166    3174
151    1159    2167    3175
152    1160    2168    3176
153    1161    2169    3177
154    1162    2170    3178
```

Again, each column corresponds to each of the four antenna ports. The concatenation of all four columns produces a column vector of indices using the MATLAB linear indexing style.

Index Base

Generate either zero-based or one-based indices. All mapping operations in the LTE technical specification (TS) documents refer to zero-based indexing. However, MATLAB® indices must be one-based. LTE System Toolbox™ generates one-based indices by default, but you can generate zero-based indices by setting the appropriate options string.

Create a cell-wide setting structure and assign an antenna port number.

```
enb.NDLRB = 6;  
enb.NCellID = 1;  
enb.CyclicPrefix = 'Normal';  
enb.DuplexMode = 'FDD';
```

```
antPort = 0;
```

Since one-based indexing is the default, you can generate one-based indices by specifying the '1based' flag or leaving it out.

```
ind = lteCellRSIndices(enb,antPort);  
ind = lteCellRSIndices(enb,antPort,'1based');
```

Generate zero-based indices by specifying the '0based' flag.

```
ind = lteCellRSIndices(enb,antPort,'0based');
```

Resource Blocks

The 3GPP documents describes a *resource block* to be a group of resource elements spanning 12 consecutive subcarriers in the frequency domain and one slot in the time domain. For processing efficiency, the LTE System Toolbox™, operates on a subframe (two timeslot) basis and describes a *resource block pair* to represent 12 consecutive subcarriers spanning in the frequency domain and one subframe (two slots) in the time domain. For example, the command `ltePDSCHIndices` uses the parameter `PRBSet` to define the set of physical resource block (PRB) indices for a subframe of data.

Create the cell-wide settings structure and define the PDSCH configuration.

```
enb.CellRefP = 4;
enb.CFI = 1;
enb.NCellID = 1;
enb.NSubframe = 0;
enb.NDLRB = 6;
enb.CyclicPrefix = 'Normal';
enb.DuplexMode = 'FDD';

pdsch.TxScheme = 'TxDiversity';
pdsch.Modulation = 'QPSK';
pdsch.RNTI = 1;
pdsch.PRBSets = (0:5).';
```

Create a set of PDSCH PRB indices for the initialized configuration.

```
pdschIndices = ltePDSCHIndices(enb,pdsch,pdsch.PRBSets);
size(pdschIndices)
pdschIndices(1:10,:)
pdsch
```

```
ans =
```

```
    384     4
```

```
ans =
```

```
10x4 uint32 matrix
```

```
    145    1153    2161    3169
    146    1154    2162    3170
    147    1155    2163    3171
    148    1156    2164    3172
    149    1157    2165    3173
    150    1158    2166    3174
    151    1159    2167    3175
    152    1160    2168    3176
    153    1161    2169    3177
    154    1162    2170    3178
```

```
pdsch =
```

```
struct with fields:

    TxScheme: 'TxDiversity'
    Modulation: 'QPSK'
    RNTI: 1
    PRBSet: [6x1 double]
```

`pdsch.PRBSet` can be either a column vector or a two-column matrix. If you provide a column vector, the resource allocation is the same in both slots of the subframe, which means the set of resource indices applies to both subframe time slots. On the other hand, if you provide a two-column matrix, the PRB indices refer to each slot individually.

See Also

`lteCellRS` | `lteCellRSIndices` | `lteDLResourceGrid` | `ltePDSCH` |
`ltePDSCHIndices` | `zeros`

Related Examples

- “Parameterization” on page 1-35

Parameterization

In this section...

“Parameter Structures” on page 1-35

“Create a Cell-Wide Settings Structure” on page 1-36

“Cell-Wide Parameters” on page 1-37

“Optional Output Formats” on page 1-37

Some of the functions in the LTE System Toolbox require a large number of parameters. To simplify the process, the LTE System Toolbox groups relevant parameters together into structures.

Parameter Structures

Consider, as an example, the task of generating PCFICH symbols and mapping indices. For this task, you can call the functions `ltePCFICH`, and `ltePCFICHIndices`. The `ltePCFICH` function also requires `cw`, an input bit vector. For this input, you can call the `lteCFI` function. All three functions require a parameter structure, `enb`, that represents the eNodeB cell-wide settings.

The function `ltePCFICH` requires `enb` to have at least the following fields.

- `NCellID` — Physical layer cell identity
- `CellRefP` — Number of cell-specific reference signal antenna ports. Valid values are 1, 2, and 4.
- `NSubframe` — Subframe number

In comparison, the function `ltePCFICHIndices` requires `enb` to have at least the following fields.

- `NCellID` — Physical layer cell identity
- `NDLRB` — Number of downlink resource blocks

Finally, the function `lteCFI` only requires `enb` to have one field, `CFI`. In all cases, if additional fields are present and not required, the function ignores them.

Create a Cell-Wide Settings Structure

This example shows how to create a cell-wide settings structure. In particular, you can create a parameter structure, `enb`, that has all the fields required by the `lteCFI`, `ltePCFICH`, and `ltePCFICHIndices` functions.

Create a new parameter structure, `enb`, with only one field, `CFI`.

```
enb.CFI = 1;
```

Create a 32-element bit vector, `cw`, representing the rate 1/16 block encoding of the control format indicator (CFI) value. To do so, call the `lteCFI` function. Provide `enb` as an input argument.

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

Add additional fields to the parameter structure, `enb`.

```
enb.NCellID = 0;  
enb.CellRefP = 1;  
enb.NSubframe = 0;  
enb.NDLRB = 9;
```

Generate the PCFICH complex symbols by calling the `ltePCFICH` function, providing the `enb` structure and the `cw` bit vector as input arguments.

```
sym = ltePCFICH(enb,cw);
```

Although `ltePCFICH` does not require that `enb` have the `NDLRB` field, this does not cause a problem. In this case, the function ignores any non-required fields.

Generate the PCFICH mapping indices by calling the `ltePCFICHIndices` function, providing the `enb` structure as an input argument.

```
ind = ltePCFICHIndices(enb);
```

Although `ltePCFICHIndices` does not require the `enb.NSubframe` field, it does not cause a problem. The function ignores any fields that it does not require.

You can remove fields from a structure using the MATLAB® `rmfield` function but, as shown, removing the field is not necessary.

Cell-Wide Parameters

Many functions in the LTE System Toolbox require a parameter structure called `enb`. This parameter represents the eNodeB, or cell-wide, settings which are common to all user equipments (UEs) in the cell. This structure can include the following fields, which are among the most common.

- `NCellID` — Physical layer cell identity
- `CellRefP` — Number of cell-specific reference signal antenna ports. Valid values are 1, 2, and 4.
- `CyclicPrefix` — Length of cyclic prefix. Valid values are 'Normal' and 'Extended'.
- `NSubframe` — Subframe number
- `NDLRB` — Number of downlink resource blocks

Different functions require different fields. Not all functions that require the `enb` structure need all the fields listed above. Some functions require only a subset of those listed above. In this case, any non-required fields are ignored.

When optional parameter fields are not specified, a function in the LTE System Toolbox may assume default settings. In this case, the toolbox produces warning messages to specify the default values that it is using. You may control these warnings using the `lteWarning` function.

Optional Output Formats

This example shows how to pass optional inputs to certain functions to change the output format provided from the function.

Create a new parameter structure, `enb`.

```
enb.NCellID = 0;  
enb.CellRefP = 1;  
enb.NSubframe = 0;  
enb.NDLRB = 9;  
enb.Ng = 'Sixth';
```

For example, consider the case where a list of indices for a certain physical channel is generated using `ltePCFICHIndices`.

The input argument, `enb`, is a structure with the appropriate fields. By default, these indices are one-based, as opposed to the zero-based indices specified in the technical specification (TS) documentation.

```
ind = ltePCFICHIndices(enb);  
firstIndex = ind(1)
```

```
firstIndex =  
  
    uint32  
  
    2
```

Change the base number used in the index generation by providing an additional optional input argument. Specify `'0based'` to generate zero-based indices or `'1based'` to generate one-based indices.

```
ind = ltePCFICHIndices(enb, '0based');  
firstIndex_0based = ind(1)  
ind = ltePCFICHIndices(enb, '1based');  
firstIndex_1based = ind(1)
```

```
firstIndex_0based =  
  
    uint32  
  
    1
```

```
firstIndex_1based =  
  
    uint32  
  
    2
```

The first index generated when no optional argument is provided matches the first index when `'1based'` is specified. The optional input is not required. If you do not specify an optional input, the function uses the default value.

Specify multiple output format options for a function by providing a cell array input argument, `opts`.

```
opts = {'sub', 'lbased', 'reg'};  
pcfichInd = ltePCFICHIndices(enb, opts)
```

```
pcfichInd =
```

```
4x3 uint32 matrix
```

```
    1    1    1  
   25    1    1  
   55    1    1  
   79    1    1
```

The generated PCFICH indices are in subscript indexing style, one-based, and refer to resource element groups. The cell array of options that you specify indicates the format for the returned indices.

Varying the order of the `opts` cell entries produces the same result.

```
opts = {'lbased', 'sub', 'reg'};  
pcfichInd = ltePCFICHIndices(enb, opts)
```

```
pcfichInd =
```

```
4x3 uint32 matrix
```

```
    1    1    1  
   25    1    1  
   55    1    1  
   79    1    1
```

The order in which you provide the `opts` inputs is not relevant. Both cases produce the same values in the output argument, `pcfichInd`.

See Also

`lteCFI` | `ltePCFICH` | `ltePCFICHIndices` | `lteWarning` | `rmfield`

Related Examples

- “LTE Parameterization for Waveform Generation and Simulation”
- “UL-SCH Parameterization” on page 1-41

UL-SCH Parameterization

In this section...

“Set UL-SCH Parameters in Scalar Structure” on page 1-41

“Set UL-SCH Parameters in Structure Array” on page 1-43

A number of the uplink shared channel (UL-SCH) and PUSCH related functions offer two different ways of parameterizing multiple codewords in the UL-SCH or PUSCH-specific parameter structure. As with many functions in the LTE System Toolbox, the parameters associated with codewords can be combined together in the individual fields of a single scalar (1-by-1) structure. However, many UL-SCH-specific functions also allow each codeword to be defined by separate independent elements of a (1-by-2) structure array. This feature offers additional flexibility and results in more compact code when explicit fine-grained parameterization of the individual processing steps is required.

Set UL-SCH Parameters in Scalar Structure

This example shows how to parameterize an UL-SCH or PUSCH-specific parameter structure using two different representations. Consider creating a parameter structure for the `lteULSCHDeinterleave` function.

When UCI is being transmitted on the UL-SCH, the deinterleaving and UCI demultiplexing operations require explicit knowledge of number of control channel symbols within the codeword. For example, for a single LTE Release 8 codeword, the UL-SCH specific parameters can be defined by a scalar (1-by-1) structure.

```
ulsch1.Modulation = 'QPSK';
ulsch1.QdCQI = 4;
ulsch1.QdRI = 2;
ulsch1.QdACK = 2;
```

In this case, there are four CQI, two RI, and two HARQ-ACK symbols within the QPSK-modulated codeword.

When moving to a full LTE-Advanced uplink transmission, you must consider a second possible codeword and the impact of the additional PUSCH layering. This layering can be achieved either by adding values in the structure field values above or by using a 1-by-2 element structure array to define the codeword pair. For example, transmit a second 16-QAM-modulated codeword also, which now carries the CQI and both codewords are sent on a total of three spatial layers.

```
ulsch2.Modulation = {'QPSK', '16QAM'};  
ulsch2.NLayers = 3;  
ulsch2.QdCQI = [0,4];  
ulsch2.QdRI = 2;  
ulsch2.QdACK = 2;
```

Since the CQI should only be transmitted on one of the codewords (the second one here) this symbol allocation is signaled by setting `ulsch2.QdCQI = [0,4]`.

You must explicitly specify some parameters for each codeword. However, in general, when using a single 1-by-1 structure for multi-codeword parameterization, scalar parameter field values are assigned to all codewords. The structure `ulsch2` sets the number of RI and HARQ-ACK coded modulation symbols per layer per codeword to two. Make this number of symbols explicit for each codeword by defining the `QdRI` and `QdACK` fields as 1-by-2 vectors.

```
ulsch2.QdRI = [2,2];  
ulsch2.QdACK = [2,2];
```

One special case is the parameter field which controls the number of spatial layers, `NLayers`, which has slightly different semantics. If this field value is scalar, it defines the total number of layers across all codewords. Following the LTE standard formulae, when you set the total number of layers to three, the LTE System Toolbox™ partitions one layer for the first codeword and two layers for the second codeword. Make this layer allocation per codeword explicit by defining the `NLayers` field as a 1-by-2 vector.

```
ulsch2.NLayers = [1,2];
```

In summary, you can write the overall parameter structure by declaring all of the parameter fields at once.

```
ulsch2.Modulation = {'QPSK', '16QAM'};  
ulsch2.NLayers = [1,2];  
ulsch2.QdCQI = [0,4];  
ulsch2.QdRI = [2,2];  
ulsch2.QdACK = [2,2];
```

This structure is equivalent to the ones created earlier.

Set UL-SCH Parameters in Structure Array

This example shows how to parameterize an UL-SCH or PUSCH-specific parameter structure using two different representations. Consider creating a parameter structure for the `lteULSCHDeinterleave` function.

When UCI is being transmitted on the UL-SCH, the deinterleaving and UCI demultiplexing operations require explicit knowledge of number of control channel symbols within the codeword. For example, for a single LTE Release 8 codeword, the UL-SCH specific parameters can be defined by a scalar (1-by-1) structure.

```
ulsch1.Modulation = 'QPSK';
ulsch1.QdCQI = 4;
ulsch1.QdRI = 2;
ulsch1.QdACK = 2;
```

The UL-SCH-specific structure also allows each codeword to be defined by separate, independent elements of a 1-by-2 structure array. In this case, the important distinction is that no parameter field values are implicitly shared between the codewords. Each field value applies only to the codeword associated with that structure array element. For example, redefine the single codeword structure by creating a new 1-by-2 structure array containing two identical elements.

```
ulsch2(1:2) = ulsch1

ulsch2 =

    1x2 struct array with fields:

        Modulation
        QdCQI
        QdRI
        QdACK
```

Update only the parameters which are different for each codeword.

```
ulsch2(1).QdCQI = 0;
ulsch2(2).Modulation = '16QAM';
```

Add the explicit number of layers per codeword parameter, `NLayers`, to the elements of the structure array.

```
[ulsch2.NLayers] = deal(1,2);
```

View the first element of the final `ulsch2` structure array.

```
ulsch2(1)
```

```
ans =
```

```
struct with fields:

  Modulation: 'QPSK'
    QdCQI: 0
    QdRI: 2
    QdACK: 2
  NLayers: 1
```

View the second element of the final `ulsch2` structure array.

```
ulsch2(2)
```

```
ans =
```

```
struct with fields:

  Modulation: '16QAM'
    QdCQI: 4
    QdRI: 2
    QdACK: 2
  NLayers: 2
```

Both of these forms of UL-SCH parameter representation can be used in many of the UL-SCH- and PUSCH-related functions. In addition, the `lteULSCHInfo` function can return its output structure in either form:

- To receive a structure array, set the second element of the 1-by-2 `opts` cell array to `'cwseparate'`.

- To receive a scalar structure, set it to 'cwcombined'.

See Also


`lteULSCHDeinterleave` | `lteULSCHInfo`

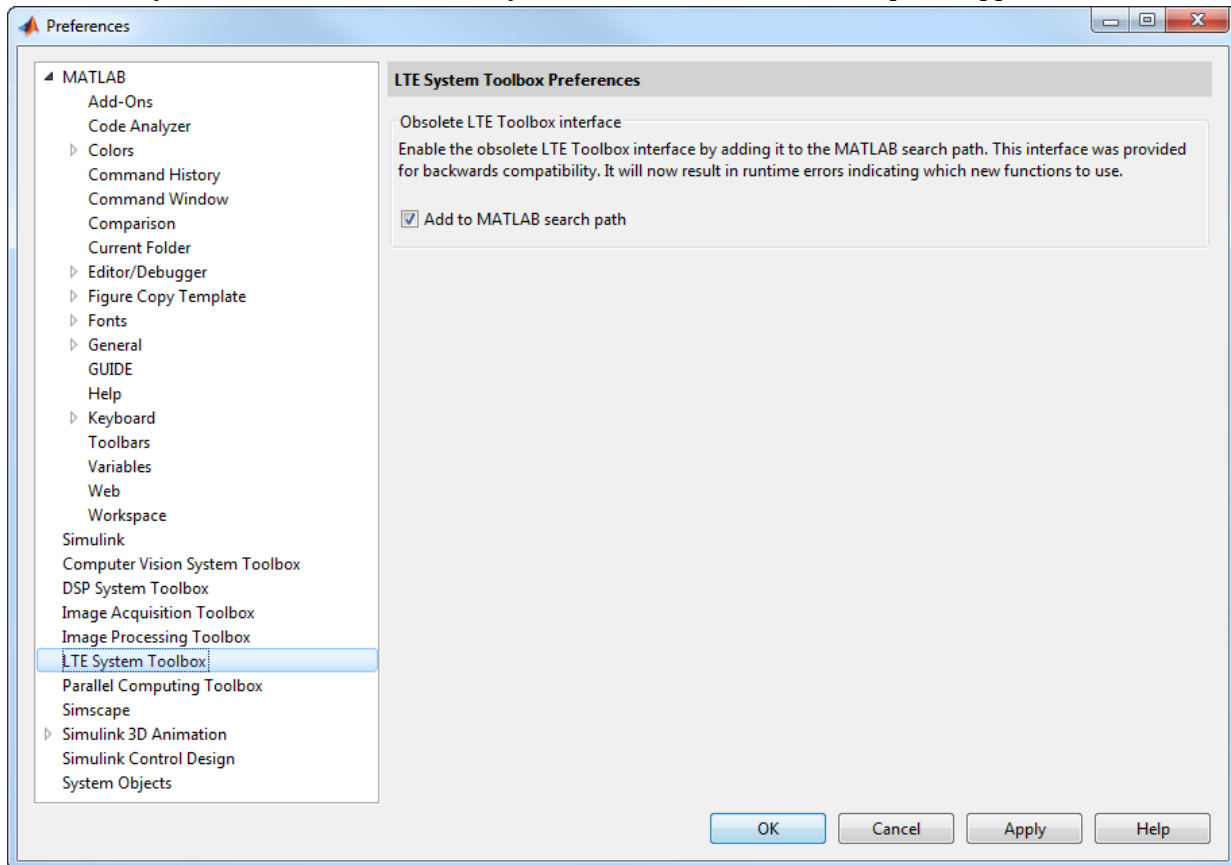
Related Examples

- “Parameterization” on page 1-35

Obsolete LTE Toolbox Interface

Previous versions of the LTE System Toolbox contained a different set of function names. The term Obsolete LTE Toolbox interface refers to these previous versions, prior to version 1.0. If you wrote scripts using any of the old function names used in previous versions, you should modify the scripts to use the new function names. Also, you should modify many of the scripts to expect column vectors for output arguments where row vectors were previously returned.

Alternatively, to enable the Obsolete LTE Toolbox interface, in the MATLAB toolstrip, select the Preferences button ( Preferences). Then, in the left navigation bar, select **LTE System Toolbox**. The **LTE System Toolbox Preferences** panel appears.



To enable the Obsolete LTE Toolbox interface, select the **Add to MATLAB search path** check box . This setting is the default and is equivalent to running the `addlteobsolete` function. To disable the Obsolete LTE Toolbox interface, clear the **Add to MATLAB search path** check box . This setting is equivalent to running the `rmlteobsolete` function. Use these functions to modify the product preferences programmatically; do not use the `setpref` function.

Note The Obsolete LTE Toolbox interface is provided for backwards compatibility. . It will now result in runtime errors indicating which new functions to use.

When **Add to MATLAB search path** is selected, all the functions listed in the **Previous Function Name** column of the following table are available on the MATLAB path. Refer to the following table for a mapping of the previous function names to their new equivalent function names.

Previous Function Name	New Function Name
LteACKDecode	lteACKDecode
LteACKEncode	lteACKEncode
LteBCH	lteBCH
LteBCHDecode	lteBCHDecode
LteCFI	lteCFI
LteCFIDeCode	lteCFIDeCode
LteCQIDeCode	lteCQIDeCode
LteCQIEncode	lteCQIEncode
LteCRC	lteCRCEncode
LteCRCDeCode	lteCRCDeCode
LteCSICodebook	lteCSICodebook
LteCSIRS	lteCSIRS
LteCSIRSIndices	lteCSIRSIndices
LteCellRS	lteCellRS
LteCellRSIndices	lteCellRSIndices
LteCellSearch	lteCellSearch

Previous Function Name	New Function Name
LteCodeBlkDeseg	lteCodeBlockDesegment
LteCodeBlkSeg	lteCodeBlockSegment
LteConvCode	lteConvolutionalEncode
LteConvDecode	lteConvolutionalDecode
LteDCI	lteDCI
LteDCIDecode	lteDCIDecode
LteDCIDims	lteDCIInfo
LteDCIEncode	lteDCIEncode
LteDLChannelEstimation	lteDLChannelEstimate
LteDLConformanceTestBench	lteDLConformanceTestTool
LteDLDeprecoder	lteDLDeprecode
LteDLFrameOffset	lteDLFrameOffset
LteDLPerfectChannelEstimation	lteDLPerfectChannelEstimate
LteDLPrecoder	lteDLPrecode
LteDLResourceGrid	lteDLResourceGrid
LteDLResourceGridDims	lteDLResourceGridSize
LteDLSCH	lteDLSCH
LteDLSCHDecode	lteDLSCHDecode
LteDLSCHDims	lteDLSCHInfo
LteDMRS	lteDMRS
LteDMRSIndices	lteDMRSIndices
LteDuplexDims	lteDuplexingInfo
LteEVM	lteEVM
LteEqualizeMIMO	lteEqualizeMIMO
LteEqualizeMMSE	lteEqualizeMMSE
LteEqualizeULMIMO	lteEqualizeULMIMO
LteEqualizeZF	lteEqualizeZF
LteFadingChan	lteFadingChannel

Previous Function Name	New Function Name
LteFreqCorrect	lteFrequencyCorrect
LteFreqOffset	lteFrequencyOffset
LteHSTChan	lteHSTChannel
LteLayerDemapper	lteLayerDemap
LteLayerMapper	lteLayerMap
LteMIB	lteMIB
LteMovingChan	lteMovingChannel
LteOFDM	lteOFDMModulate
LteOFDMDemod	lteOFDMDemodulate
LteOFDMDims	lteOFDMInfo
LtePBCH	ltePBCH
LtePBCHDecode	ltePBCHDecode
LtePBCHIndices	ltePBCHIndices
LtePBCHPRBS	ltePBCHPRBS
LtePCFICH	ltePCFICH
LtePCFICHDecode	ltePCFICHDecode
LtePCFICHDims	ltePCFICHInfo
LtePCFICHIndices	ltePCFICHIndices
LtePCFICHPRBS	ltePCFICHPRBS
LtePDCCH	ltePDCCH
LtePDCCHDecode	ltePDCCHDecode
LtePDCCHDeinterleave	ltePDCCHDeinterleave
LtePDCCHDims	ltePDCCHInfo
LtePDCCHIndices	ltePDCCHIndices
LtePDCCHInterleave	ltePDCCHInterleave
LtePDCCHPRBS	ltePDCCHPRBS
LtePDCCHSearch	ltePDCCHSearch
LtePDCCHSpace	ltePDCCHSpace

Previous Function Name	New Function Name
LtePDSCH	ltePDSCH
LtePDSCHDecode	ltePDSCHDecode
LtePDSCHIndices	ltePDSCHIndices
LtePDSCHPRBS	ltePDSCHPRBS
LtePHICH	ltePHICH
LtePHICHDecode	ltePHICHDecode
LtePHICHDeprecoder	ltePHICHDeprecode
LtePHICHDims	ltePHICHInfo
LtePHICHIndices	ltePHICHIndices
LtePHICHPRBS	ltePHICHPRBS
LtePHICHPrecoder	ltePHICHPrecode
LtePHICHTxDivDecode	ltePHICHTransmitDiversityDecode
LtePMIDims	ltePMIInfo
LtePMISelection	ltePMISelect
LtePRACH	ltePRACH
LtePRACHDetect	ltePRACHDetect
LtePRACHDims	ltePRACHInfo
LtePRBFromDCI	lteDCIResourceAllocation
LtePRBS	ltePRBS
LtePRS	ltePRS
LtePRSIndices	ltePRSIndices
LtePSS	ltePSS
LtePSSIndices	ltePSSIndices
LtePUCCH1	ltePUCCH1
LtePUCCH1DRS	ltePUCCH1DRS
LtePUCCH1DRSIndices	ltePUCCH1DRSIndices
LtePUCCH1Decode	ltePUCCH1Decode
LtePUCCH1Indices	ltePUCCH1Indices

Previous Function Name	New Function Name
LtePUCCH2	ltePUCCH2
LtePUCCH2DRS	ltePUCCH2DRS
LtePUCCH2DRSDecode	ltePUCCH2DRSDecode
LtePUCCH2DRSIndices	ltePUCCH2DRSIndices
LtePUCCH2Decode	ltePUCCH2Decode
LtePUCCH2Indices	ltePUCCH2Indices
LtePUCCH2PRBS	ltePUCCH2PRBS
LtePUCCH3	ltePUCCH3
LtePUCCH3DRS	ltePUCCH3DRS
LtePUCCH3DRSIndices	ltePUCCH3DRSIndices
LtePUCCH3Decode	ltePUCCH3Decode
LtePUCCH3Indices	ltePUCCH3Indices
LtePUCCH3PRBS	ltePUCCH3PRBS
LtePUSCH	ltePUSCH
LtePUSCHDRS	ltePUSCHDRS
LtePUSCHDRSIndices	ltePUSCHDRSIndices
LtePUSCHDecode	ltePUSCHDecode
LtePUSCHDeprecoder	ltePUSCHDeprecode
LtePUSCHIndices	ltePUSCHIndices
LtePUSCHPrecoder	ltePUSCHPrecode
LteRIDecode	lteRIDecode
LteRIEncode	lteRIEncode
LteRMCDL	lteRMCDL
LteRMCDLTool	lteRMCDLTool
LteRMCUL	lteRMCUL
LteRMCULTool	lteRMCULTool
LteRateMatchConv	lteRateMatchConvolutional
LteRateMatchTurbo	lteRateMatchTurbo

Previous Function Name	New Function Name
LteRateRecoverConv	lteRateRecoverConvolutional
LteRateRecoverTurbo	lteRateRecoverTurbo
LteResourceGrid	lteResourceGrid
LteResourceGridDims	lteResourceGridSize
LteSCFDMA	lteSCFDMAModulate
LteSCFDMADemod	lteSCFDMADemodulate
LteSCFDMADims	lteSCFDMAInfo
LteSRS	lteSRS
LteSRSDims	lteSRSInfo
LteSRSIndices	lteSRSIndices
LteSSS	lteSSS
LteSSSIndices	lteSSSIndices
LteSymbolDemod	lteSymbolDemodulate
LteSymbolMod	lteSymbolModulate
LteTBS	lteTBS
LteTestModel	lteTestModel
LteTestModelTool	lteTestModelTool
LteTurboCode	lteTurboEncode
LteTurboDecode	lteTurboDecode
LteTxDiversityDecode	lteTransmitDiversityDecode
LteUCI3Decode	lteUCI3Decode
LteUCI3Encode	lteUCI3Encode
LteUCIDecode	lteUCIDecode
LteUCIEncode	lteUCIEncode
LteUeRS	Removed. Use lteDMRS instead.
LteUeRSIndices	Removed. Use lteDMRSIndices instead.
LteULChannelEstimation	lteULChannelEstimate
LteULChannelEstimationPUCCH1	lteULChannelEstimatePUCCH1

Previous Function Name	New Function Name
LteULChannelEstimationPUCCH2	lteULChannelEstimatePUCCH2
LteULChannelEstimationPUCCH3	lteULChannelEstimatePUCCH3
LteULDeprecoder	lteULDeprecode
LteULDescrambler	lteULDescramble
LteULFrameOffset	lteULFrameOffset
LteULFrameOffsetPUCCH1	lteULFrameOffsetPUCCH1
LteULFrameOffsetPUCCH2	lteULFrameOffsetPUCCH2
LteULFrameOffsetPUCCH3	lteULFrameOffsetPUCCH3
LteULPMIDims	lteULPMIInfo
LteULPMISelection	lteULPMISelect
LteULPerfectChannelEstimation	lteULPerfectChannelEstimate
LteULPrecoder	lteULPrecode
LteULResourceGrid	lteULResourceGrid
LteULResourceGridDims	lteULResourceGridSize
LteULSCH	lteULSCH
LteULSCHDecode	lteULSCHDecode
LteULSCHDeinterleave	lteULSCHDeinterleave
LteULSCHDims	lteULSCHInfo
LteULSCHInterleave	lteULSCHInterleave
LteULScrambler	lteULScramble
LteVersion	Removed. Use the MATLAB version function instead.
LteWarning	lteWarning
LteZadoffChu	Removed. Use lteZadoffChuSeq in the Communications System Toolbox™ instead.

See Also

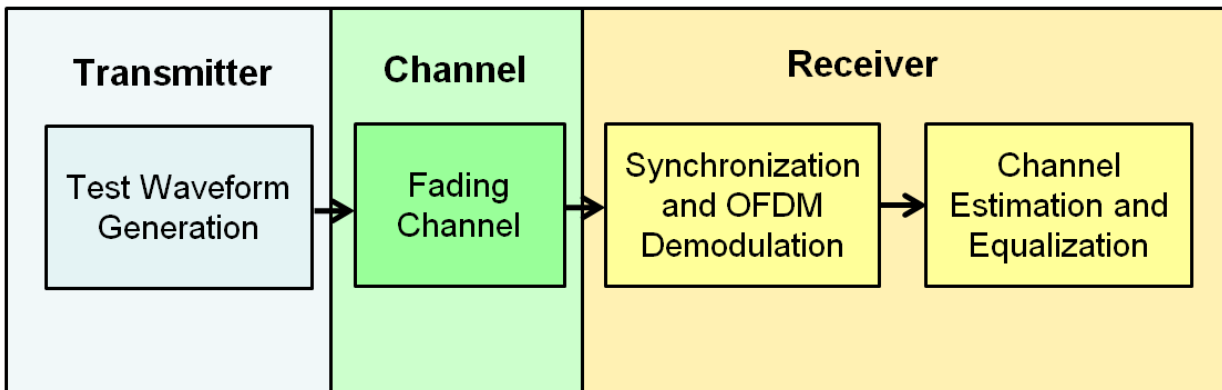
addlteobsolete | rmlteobsolete | setpref

High-Level Examples

Transmit-Receive Chain Processing

Transmit-Receive Chain

This example shows how to implement an LTE transmit and receive chain, as shown in this figure.

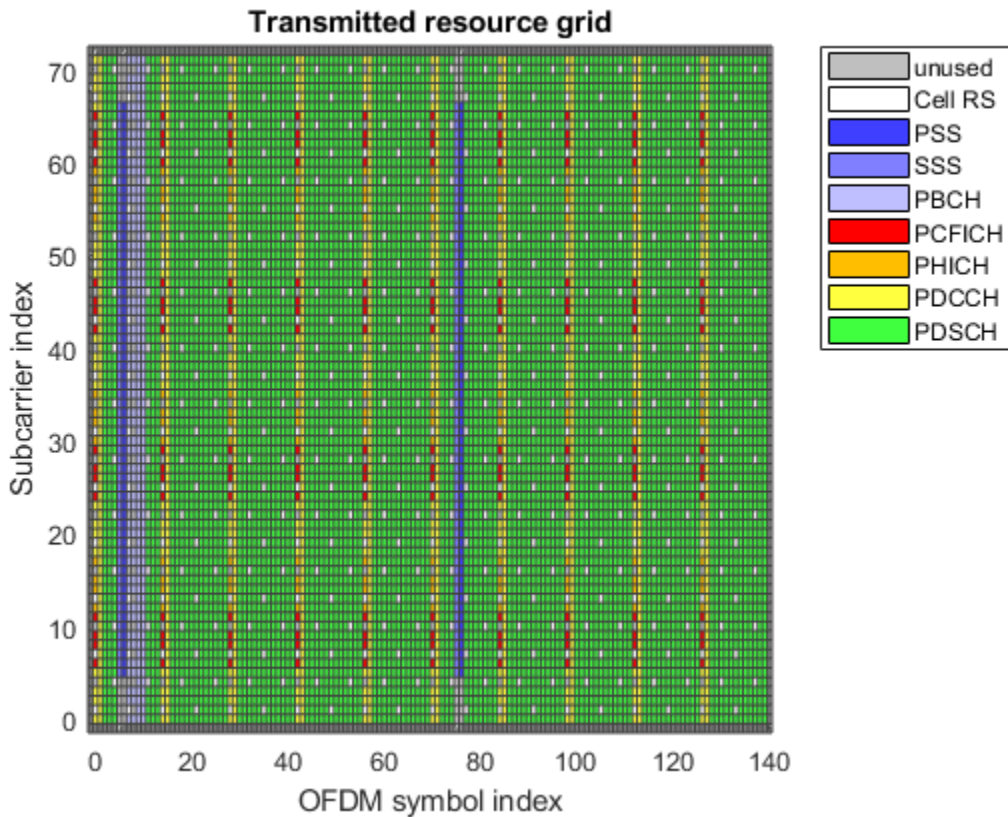


Generate an E-UTRA test model (E-TM) configuration. Use this configuration to generate the waveform and populate the resource grid.

```
enb = lteTestModel('1.1', '1.4MHz');  
[txwave, txgrid, info] = lteTestModelTool(enb);
```

Plot a graphical representation of the transmit resource grid.

```
figure('Color', 'w');  
helperPlotTransmitResourceGrid(enb, txgrid);
```



The figure shows a resource grid populated with E-TM 1.1 resource elements.

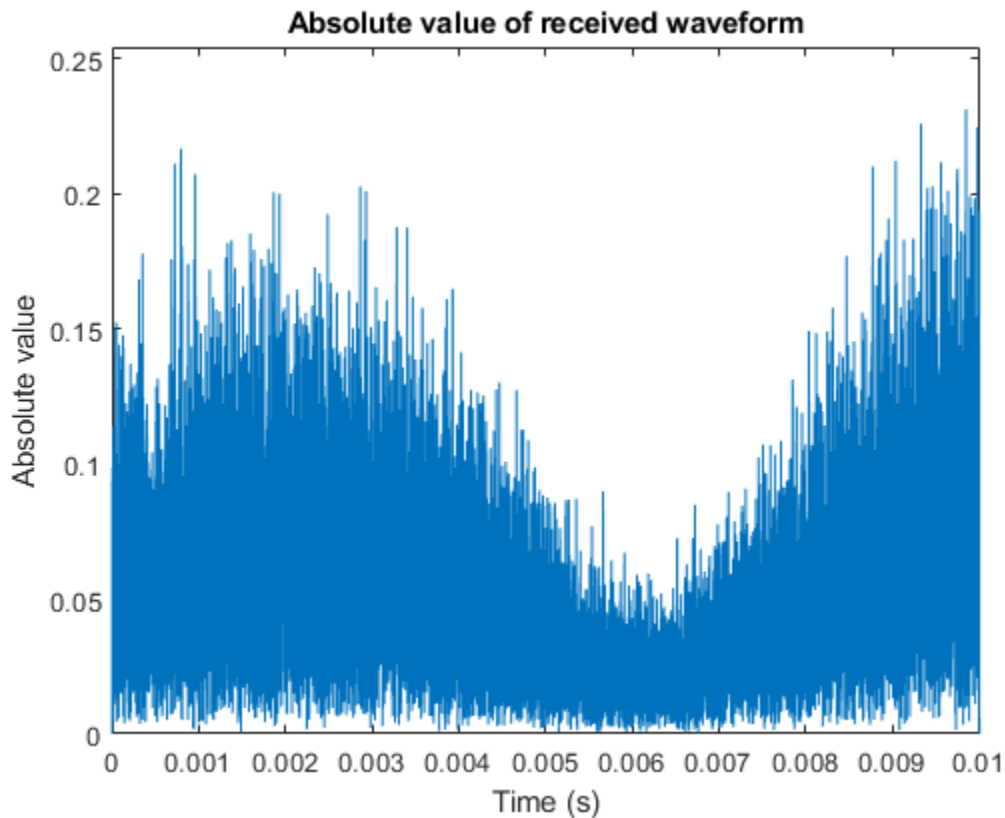
Simulate transmission through a fading channel propagation model.

```
channel.ModelType = 'GMEDS';
channel.DelayProfile = 'EVA';
channel.DopplerFreq = 70;
channel.MIMOCorrelation = 'Medium';
channel.NRxAnts = 1;
channel.InitTime = 0;
channel.InitPhase = 'Random';
channel.Seed = 17;
channel.NormalizePathGains = 'On';
channel.NormalizeTxAnts = 'On';
```

```
channel.SamplingRate = info.SamplingRate;  
channel.NTerms = 16;  
rxwave = lteFadingChannel(channel, [txwave; zeros(25,1)]);
```

Plot the time-varying power of the received waveform.

```
figure('Color','w');  
helperPlotReceiveWaveform(info, rxwave);
```



This plot shows the waveform power variation over time.

Perform frame synchronization.

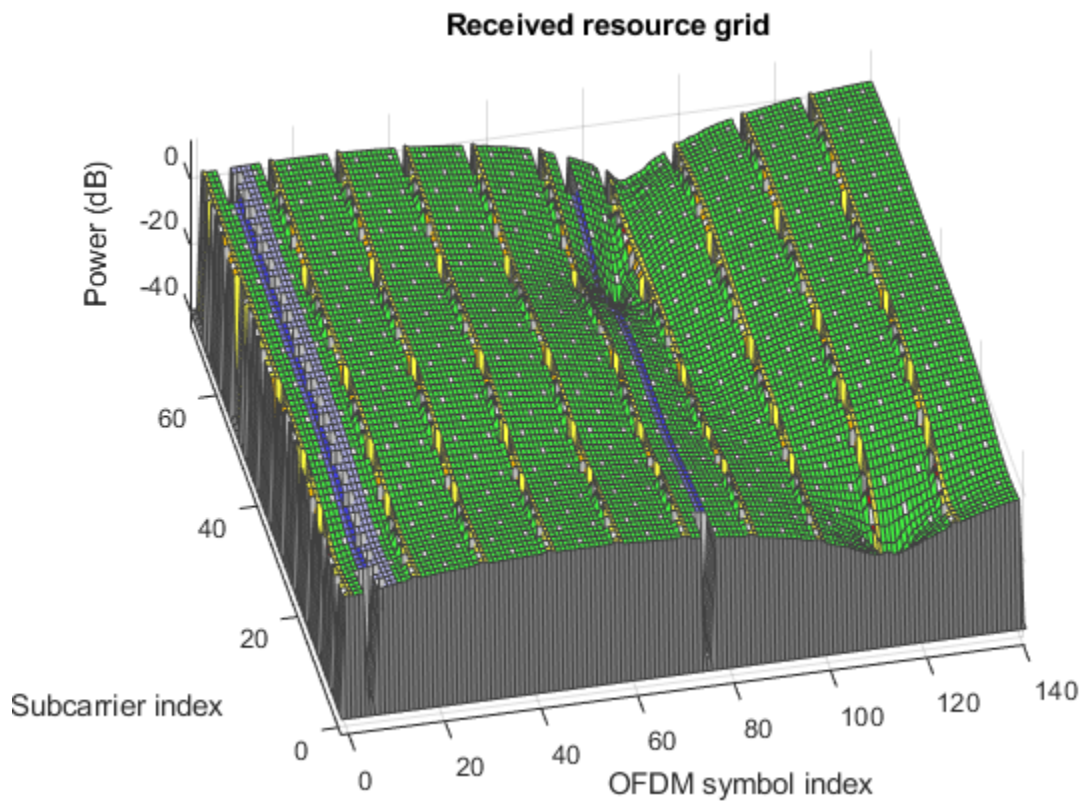
```
offset = lteDLFrameOffset(enb, rxwave);  
rxwave = rxwave(offset:end, :);
```

Perform OFDM demodulation.

```
rxgrid = lteOFDMDemodulate(enb, rxwave);
```

Create a surface plot showing the power of the received grid for each subcarrier and OFDM symbol.

```
figure('Color','w');  
helperPlotReceiveResourceGrid(enb, rxgrid);
```



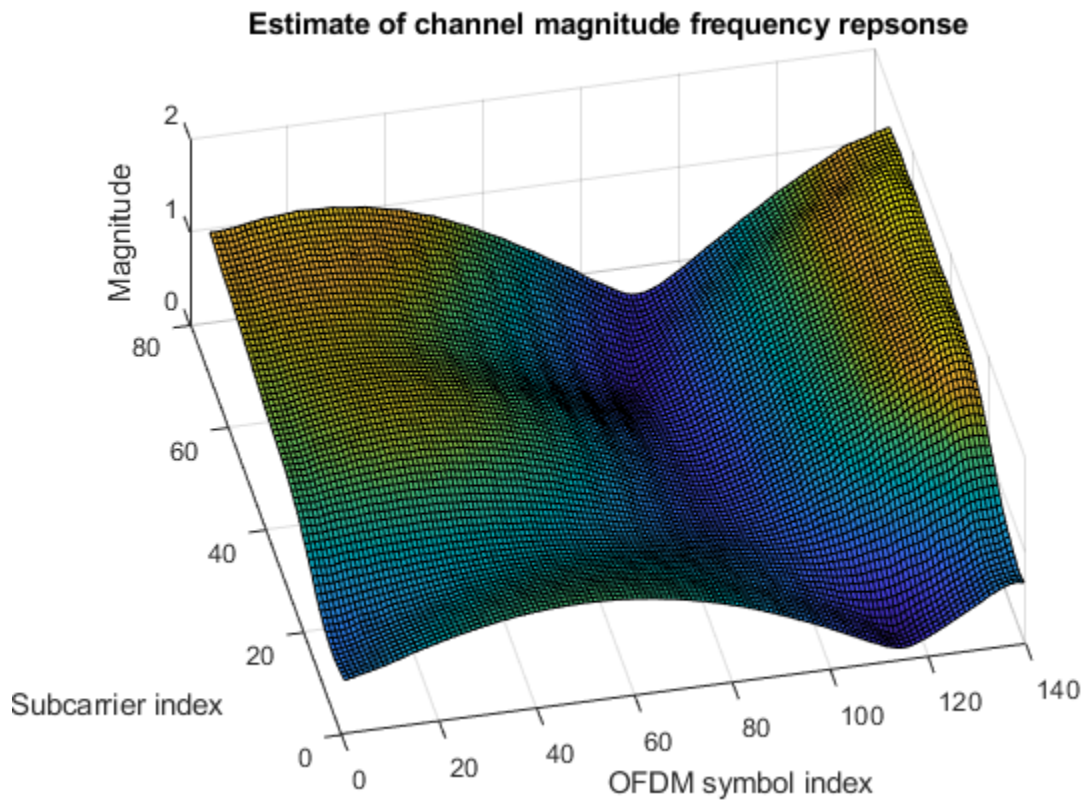
This plot shows the received grid power.

Estimate the channel and noise.

```
cec.PilotAverage = 'UserDefined';  
cec.FreqWindow = 9;  
cec.TimeWindow = 9;  
cec.InterpType = 'Cubic';  
cec.InterpWindow = 'Centered';  
cec.InterpWinSize = 3;  
[hest, nest] = lteDLChannelEstimate(enb, cec, rxgrid);
```

Create a surface plot showing the magnitude of the channel estimate for each OFDM symbol across the subcarriers.

```
figure('Color', 'w');  
helperPlotChannelEstimate(hest);
```



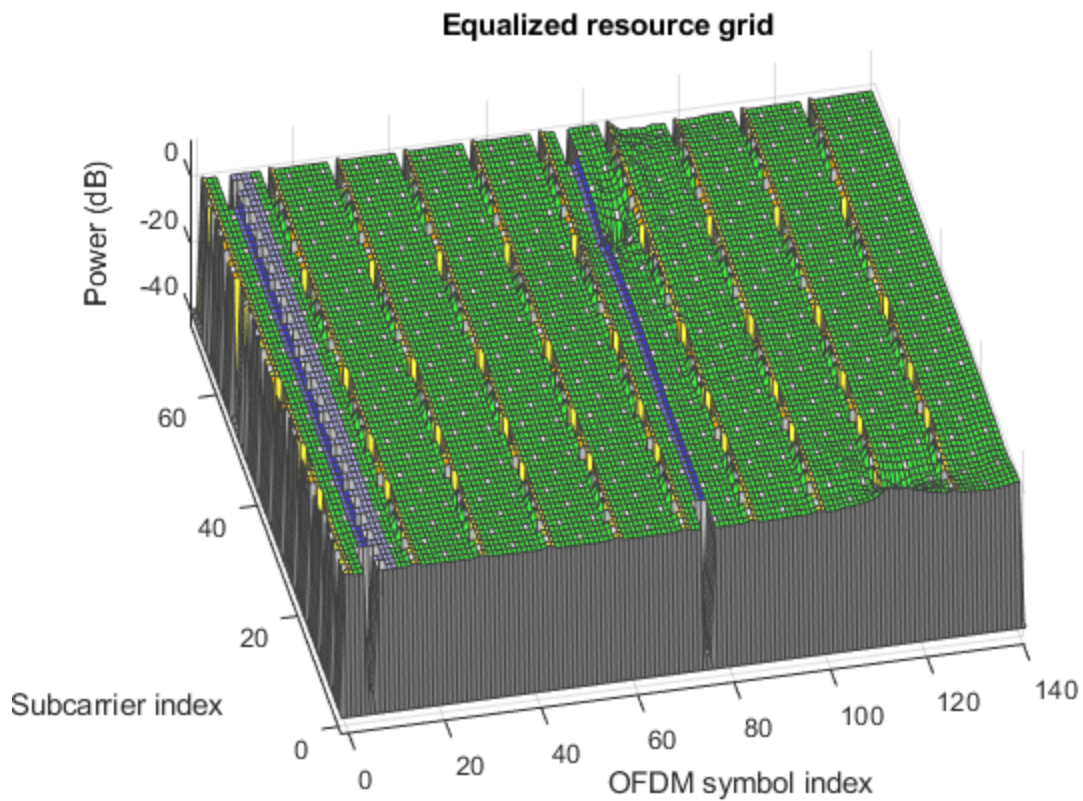
This figure shows an estimate of channel magnitude frequency response.

Finally, perform minimum mean-square error (MMSE) equalization on the received grid.

```
eqgrid = lteEqualizeMMSE(rxgrid,hest,nest);
```

Create a surface plot of the power of the equalized resource grid, in dB.

```
figure('Color','w');  
helperPlotEqualizedResourceGrid(enb,eqgrid);
```



As can be seen the equalization flattened the power response across the resource grid.

See Also

`lteDLChannelEstimate` | `lteDLFrameOffset` | `lteEqualizeMMSE` |
`lteFadingChannel` | `lteOFDMDemodulate` | `lteTestModel` | `lteTestModelTool`

Related Examples

- “Generate a Test Model”
- “Simulate Propagation Channels”
- “Find Channel Impulse Response”

More About

- “Propagation Channel Models”
- “Channel Estimation”

System Toolboxes

What Is a System Toolbox?

System Toolbox products provide algorithms and tools for designing, simulating, and deploying dynamic systems in MATLAB and Simulink. These toolboxes contain MATLAB functions, System objects, and Simulink blocks that deliver the same design and verification capabilities across MATLAB and Simulink, enabling more effective collaboration among system designers. Available System Toolbox products include:

- DSP System Toolbox
- Communications System Toolbox
- Computer Vision System Toolbox
- LTE System Toolbox
- Phased Array System Toolbox

System Toolboxes support floating-point and fixed-point streaming data simulation for both sample- and frame-based data. They provide a programming environment for defining and executing code for various aspects of a system, such as initialization and reset. System Toolboxes also support code generation for a range of system development tasks and workflows, such as:

- Rapid development of reusable IP and test benches
- Sharing of component libraries and systems models across teams
- Large system simulation
- C-code generation for embedded processors
- Finite wordlength effects modeling and optimization
- Ability to prototype and test on real-time hardware