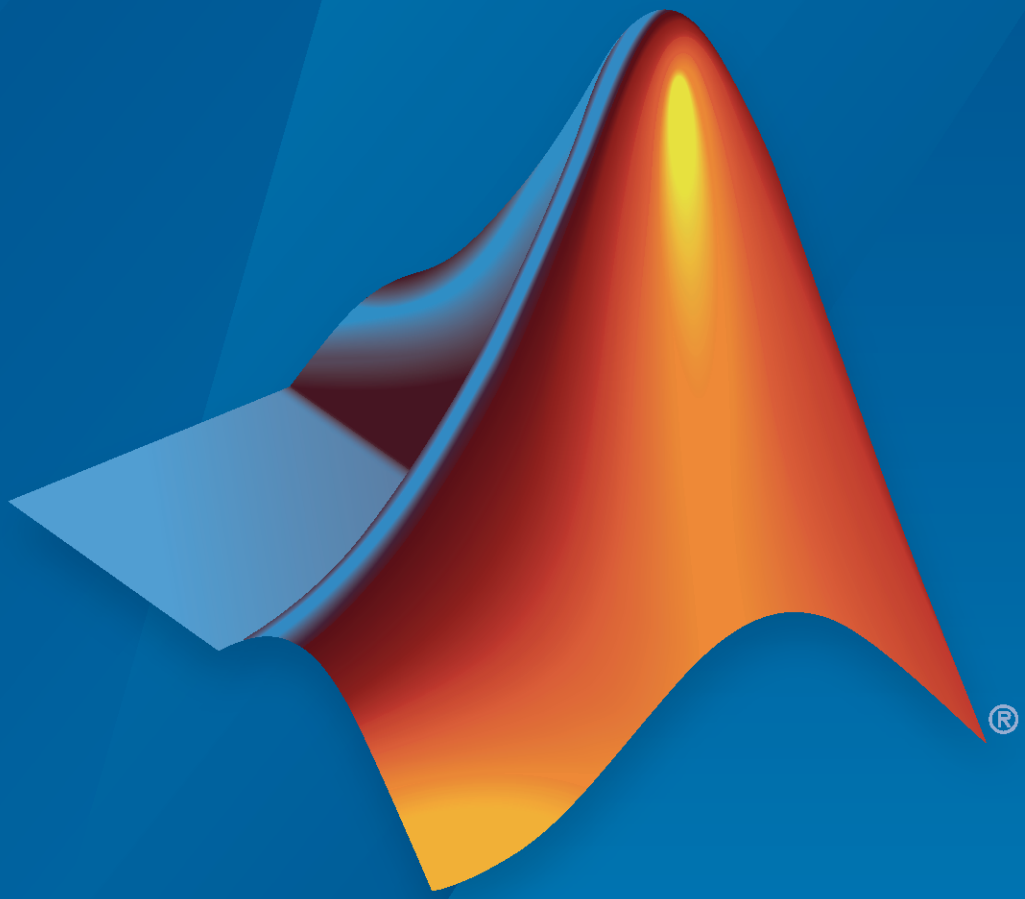
Satellite Communications Toolbox

Getting Started Guide



MATLAB®

R2023a

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.
1 Apple Hill Drive
Natick, MA 01760-2098

Satellite Communications Toolbox Getting Started Guide

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

March 2021	Online only	New for Version 1.0 (Release 2021a)
September 2021	Online only	Revised for Version 1.1 (Release 2021b)
March 2022	Online only	Revised for Version 1.2 (Release 2022a)
September 2022	Online only	Revised for Version 1.3 (Release 2022b)
March 2023	Online only	Revised for Version 1.4 (Release 2023a)

Contents

1 About Satellite Communication

2 Waveform Generation

3 Satellite Link Budget Analyzer App

4 Scenario Generation and Visualization

5 Channel Modeling

About Satellite Communication

Satellite Communications Toolbox Product Description

Simulate, analyze, and test satellite communications systems and links

Satellite Communications Toolbox provides standards-based tools for designing, simulating, and verifying satellite communications systems and links. The toolbox enables you to model and visualize satellite orbits and perform link analysis and access calculations. You can also design physical layer algorithms together with RF components and ground station receivers, generate test waveforms, and perform golden reference design verification.

With the toolbox you can configure, simulate, measure, and analyze end-to-end satellite communications links. You can also create and reuse tests to verify that your designs, prototypes, and implementations comply with satellite communications and navigations standards, including DVB-S2X, DVB-S2, CCSDS, and GPS.

Waveform Generation

Waveform Generation Workflow

Satellite Communications Toolbox provides these standards-based time-domain waveform generation functionalities.

- Consultative Committee for Space Data Systems (CCSDS) Telecommand (TC)
- CCSDS Telemetry (TM) synchronization and channel coding
- CCSDS Flexible advanced coding and modulation schemes for high rate TM applications
- Digital Video Broadcasting Satellite Second Generation (DVB-S2)
- Digital Video Broadcasting Satellite Second Generation extended (DVB-S2X)
- Digital Video Broadcasting Second Generation Return Channel over Satellite (DVB-RCS2)

Satellite Communications Toolbox objects define and configure format-specific and function-specific properties for these standard-based waveforms. The reference page of each object contains descriptions, valid settings, ranges, and other information about the object properties.

Follow these steps to generate the waveforms.

1 Select the associated object for creating the desired waveform.

This table shows the mapping between these standard-based waveforms and their associated Satellite Communications Toolbox objects.

Standard-Based Waveform	Object
CCSDS TC	`ccsdsTCConfig`
CCSDS TM synchronization and channel coding	`ccsdsTMWaveformGenerator`
CCSDS Flexible advanced coding and modulation schemes for high rate TM applications	
DVB-S2	`dvbs2WaveformGenerator`
DVB-S2X	`dvbs2xWaveformGenerator`
DVB-RCS2	`dvbrcs2WaveformGenerator`

2 Create the configuration or System object™ for the waveform type.

You can set properties by using either name-value arguments during object creation or dot notation after object creation. For details on this step, see “Create Configuration or System Objects” on page 2-3.

3 Use the created object to generate and visualize the desired waveform. For details, see “Generate Waveforms” on page 2-12.

See Also

More About

- “Create Configuration or System Objects” on page 2-3
- “Generate Waveforms” on page 2-12

Create Configuration or System Objects

Satellite Communications Toolbox configuration and System object initialize, store, and validate object properties. These properties correspond to parameters that define the standards-specific waveforms.

After you create the various objects described here, you can use them to generate waveforms. The functions in the toolbox initialize parameter settings for waveform transmission and reception by using the relevant object properties.

Create CCSDS TC Configuration Object

This example shows how to create a Consultative Committee for Space Data Systems (CCSDS) Telecommand (TC) configuration object. It also shows how to change the default property settings by using dot notation or by overriding the default settings by using `Name,Value` pairs when creating the object.

Create Object and Then Modify Properties

Create a CCSDS TC configuration object with default settings.

```
cfg = ccsdsTCConfig
```

```
cfg =
  ccsdsTCConfig with properties:

            DataFormat: "CLTU"
         ChannelCoding: "BCH"
         HasRandomizer: 1
            Modulation: "PCM/PSK/PM"
             PCMFormat: "NRZ-L"
       ModulationIndex: 0.4000
    SubcarrierFrequency: 16000
            SymbolRate: 4000
       SamplesPerSymbol: 10

   Read-only properties:
     SubcarrierWaveform: "sine"
```

Modify the defaults to specify for BPSK modulation scheme.

```
cfg.Modulation = "BPSK"
```

```
cfg =
  ccsdsTCConfig with properties:

       DataFormat: "CLTU"
    ChannelCoding: "BCH"
    HasRandomizer: 1
       Modulation: "BPSK"
```

Override Default Property Values During Object Creation

Create a CCSDS TC configuration object, using `Name,Value` pairs to specify LDPC codes with codeword length of 512.

```
cfg = ccsdsTCConfig("ChannelCoding","LDPC","LDPCCodewordLength",512)
```

```
cfg =
  ccsdsTCConfig with properties:

             DataFormat: "CLTU"
          ChannelCoding: "LDPC"
      LDPCCodewordLength: 512
             Modulation: "PCM/PSK/PM"
              PCMFormat: "NRZ-L"
         ModulationIndex: 0.4000
     SubcarrierFrequency: 16000
              SymbolRate: 4000
        SamplesPerSymbol: 10

   Read-only properties:
     SubcarrierWaveform: "sine"
```

Create CCSDS TM Object

This example shows how to create a Consultative Committee for Space Data Systems (CCSDS) Telemetry (TM) System object. It also shows how to change the default property settings by using dot notation or by overriding the default settings by using `Name,Value` pairs when creating the object.

The System object `ccsdsTMWaveformGenerator` supports these two CCSDS TM standards, depending on the type of input to `WaveformSource` property.

- CCSDS TM synchronization and channel coding standard (CCSDS 131.0-B-3)
- CCSDS flexible advanced coding and modulation scheme for high rate telemetry standard (CCSDS 131.2-B-1)

The default standard for this object is CCSDS TM synchronization and channel coding.

Create Object and Then Modify Properties

Create a CCSDS TM System object with default settings.

```
tmWaveGen = ccsdsTMWaveformGenerator
```

```
tmWaveGen =
  ccsdsTMWaveformGenerator with properties:

          WaveformSource: "synchronization and channel coding"
           HasRandomizer: true
                  HasASM: true
               PCMFormat: "NRZ-L"

   Channel coding
           ChannelCoding: "RS"
```

```
         RSMessageLength: 223
      RSInterleavingDepth: 1
     IsRSMessageShortened: false

   Digital modulation and filter
              Modulation: "QPSK"
     PulseShapingFilter: "root raised cosine"
           RolloffFactor: 0.3500
      FilterSpanInSymbols: 10
         SamplesPerSymbol: 10

  Show all properties
```

Modify the defaults to specify for turbo codes and QPSK modulation.

```
tmWaveGen.ChannelCoding = "turbo";
tmWaveGen.Modulation = "QPSK"

tmWaveGen =
  ccsdsTMWaveformGenerator with properties:

               WaveformSource: "synchronization and channel coding"
                HasRandomizer: true
                       HasASM: true
                    PCMFormat: "NRZ-L"

   Channel coding
                ChannelCoding: "turbo"
    NumBitsInInformationBlock: 7136
                     CodeRate: "1/2"

   Digital modulation and filter
                   Modulation: "QPSK"
           PulseShapingFilter: "root raised cosine"
                RolloffFactor: 0.3500
          FilterSpanInSymbols: 10
             SamplesPerSymbol: 10

  Show all properties
```

Override Default Property Values During Object Creation

Create a CCSDS TM System object, using `Name,Value` pairs, to specify the object for flexible advanced coding and modulation scheme for high rate TM applications standard, and specifying the ACM format as 9.

```
tmWaveGen = ccsdsTMWaveformGenerator("WaveformSource","flexible advanced coding and modulation","

tmWaveGen =
  ccsdsTMWaveformGenerator with properties:

              WaveformSource: "flexible advanced coding and modulation"
                   ACMFormat: 9
    NumBytesInTransferFrame: 223

   Channel coding
```

```
    No properties.

   Digital modulation and filter
         PulseShapingFilter: "root raised cosine"
              RolloffFactor: 0.3500
          FilterSpanInSymbols: 10
             SamplesPerSymbol: 10
                    HasPilots: false
        ScramblingCodeNumber: 0

  Show all properties
```

Create DVB-S2 Object

This example shows how to create a Digital Video Broadcasting Satellite Second Generation (DVB-S2) System object. It also shows how to change the default property settings by using dot notation or by overriding the default settings by using `Name,Value` pairs when creating the object.

To create a DVB-S2 System object, you must use MAT-files with LDPC parity matrices. If the MAT-files are not available on the path, download and unzip the MAT-files by entering this code at the MATLAB command prompt.

```
if ~exist('dvbs2xLDPCParityMatrices.mat','file')
    if ~exist('s2xLDPCParityMatrices.zip','file')
        url = 'https://ssd.mathworks.com/supportfiles/spc/satcom/DVB/s2xLDPCParityMatrices.zip';
        websave('s2xLDPCParityMatrices.zip',url);
        unzip('s2xLDPCParityMatrices.zip');
    end
addpath('s2xLDPCParityMatrices');
end
```

Create Object and Then Modify Properties

Create a DVB-S2 System object with default settings.

```
s2WaveGen = dvbs2WaveformGenerator
```

```
s2WaveGen =
  dvbs2WaveformGenerator with properties:

           StreamFormat: "TS"
        NumInputStreams: 1
                FECFrame: "normal"
                  MODCOD: 1
                     DFL: 15928
                HasPilots: 0
           RolloffFactor: 0.3500
    FilterSpanInSymbols: 10
       SamplesPerSymbol: 4

  Show all properties
```

Modify the defaults to specify multi-input generic stream and the data field length (DFL) for each stream.

```
s2WaveGen.StreamFormat = "GS";
s2WaveGen.NumInputStreams = 3;
s2WaveGen.DFL = [44500 51387 42960]
```

```
s2WaveGen =
  dvbs2WaveformGenerator with properties:

           StreamFormat: "GS"
         NumInputStreams: 3
                     UPL: 0
                FECFrame: "normal"
                  MODCOD: 1
                     DFL: [44500 51387 42960]
              HasPilots: 0
           RolloffFactor: 0.3500
     FilterSpanInSymbols: 10
        SamplesPerSymbol: 4
```

Override Default Property Values During Object Creation

Create a DVB-S2 System object, using `Name,Value` pairs for a single-input transport stream with short FEC frame format, and specified modulation scheme and FEC rate (MODCOD).

```
s2WaveGen = dvbs2WaveformGenerator("FECFrame","short","MODCOD",10)
```

```
s2WaveGen =
  dvbs2WaveformGenerator with properties:

           StreamFormat: "TS"
         NumInputStreams: 1
                FECFrame: "short"
                  MODCOD: 10
                     DFL: 15928
              HasPilots: 0
           RolloffFactor: 0.3500
     FilterSpanInSymbols: 10
        SamplesPerSymbol: 4

  Show all properties
```

Create a DVB-S2X Object

This example shows how to create a Digital Video Broadcasting Satellite Second Generation extended (DVB-S2X) System object. It also shows how to change the default property settings by using dot notation or by overriding the default settings by using Name,Value pairs when creating the object.

To create a DVB-S2X System object, you must use MAT-files with LDPC parity matrices. If the MAT-files are not available on the path, download and unzip the MAT-files by entering this code at the MATLAB command prompt.

```
if ~exist('dvbs2xLDPCParityMatrices.mat','file')
    if ~exist('s2xLDPCParityMatrices.zip','file')
        url = 'https://ssd.mathworks.com/supportfiles/spc/satcom/DVB/s2xLDPCParityMatrices.zip';
        websave('s2xLDPCParityMatrices.zip',url);
```

```
        unzip('s2xLDPCParityMatrices.zip');
    end
addpath('s2xLDPCParityMatrices');
end
```

Create Object and Then Modify Properties

Create a DVB-S2X System object with default settings.

```
s2xWaveGen = dvbs2xWaveformGenerator
```

```
s2xWaveGen =
  dvbs2xWaveformGenerator with properties:

           StreamFormat: "TS"
         HasTimeSlicing: false
        NumInputStreams: 1
         PLSDecimalCode: 132
                    DFL: 18448
      PLScramblingIndex: 0
          RolloffFactor: 0.3500
    FilterSpanInSymbols: 10
       SamplesPerSymbol: 4

  Show all properties
```

Modify the defaults to specify multi-input transport stream with time slicing enabled.

```
s2xWaveGen.NumInputStreams = 4;
s2xWaveGen.HasTimeSlicing = true
```

```
s2xWaveGen =
  dvbs2xWaveformGenerator with properties:

           StreamFormat: "TS"
         HasTimeSlicing: true
        NumInputStreams: 4
         PLSDecimalCode: 132
                    DFL: 18448
      PLScramblingIndex: 0
          RolloffFactor: 0.3500
    FilterSpanInSymbols: 10
       SamplesPerSymbol: 4
                  ISSYI: false

  Show all properties
```

Override Default Property Values During Object Creation

Create a DVB-S2X System object, using Name,Value pairs to specify the very low signal to noise ratio (VL-SNR) frame set 2, and specfying the modulation scheme and code rate as BPSK 1/5.

```
s2xWaveGen = dvbs2xWaveformGenerator("PLSDecimalCode",131,"CanonicalMODCODName","BPSK 1/5")
```

```
s2xWaveGen =
  dvbs2xWaveformGenerator with properties:
```

```
          StreamFormat: "TS"
        HasTimeSlicing: false
       NumInputStreams: 1
        PLSDecimalCode: 131
    CanonicalMODCODName: "BPSK 1/5"
                   DFL: 18448
      PLScramblingIndex: 0
          RolloffFactor: 0.3500
    FilterSpanInSymbols: 10
       SamplesPerSymbol: 4

  Show all properties
```

Create DVB-RCS2 Object

This example shows how to create a Digital Video Broadcasting Second Generation Return Channel over Satellite (DVB-RCS2) System object. It also shows how to change the default property settings by using dot notation or by overriding the default settings by using `Name,Value` pairs when creating the object.

Create Object and Then Modify Properties

Create a DVB-RCS2 System object with default settings.

```
wg = dvbrcs2WaveformGenerator
```

```
wg =
  dvbrcs2WaveformGenerator with properties:

      TransmissionFormat: "TC-LM"
             ContentType: "traffic"
        IsCustomWaveform: false
              WaveformID: 1
     PreBurstGuardLength: 0
    PostBurstGuardLength: 0
     FilterSpanInSymbols: 10
        SamplesPerSymbol: 4

   Read-only:
         FramePDULength: 272
```

Modify the defaults to specify the transmission format and the burst content type.

```
wg.TransmissionFormat = "SS-TC-LM";
wg.ContentType = "logon"
```

```
wg =
  dvbrcs2WaveformGenerator with properties:

      TransmissionFormat: "SS-TC-LM"
             ContentType: "logon"
        IsCustomWaveform: false
              WaveformID: 1
     PreBurstGuardLength: 0
```

```
    PostBurstGuardLength: 0
     FilterSpanInSymbols: 10
        SamplesPerSymbol: 4

   Read-only:
          FramePDULength: 784
```

Override Default Property Values During Object Creation

Create a DVB-RCS2 System object, using `Name,Value` pairs, to specify the object for a custom waveform with pre burst guard length as 4.

```
wg = dvbrcs2WaveformGenerator("IsCustomWaveform",true,"PreBurstGuardLength",4)
```

```
wg =
  dvbrcs2WaveformGenerator with properties:

       TransmissionFormat: "TC-LM"
              ContentType: "traffic"
         IsCustomWaveform: true
      PreBurstGuardLength: 4
     PostBurstGuardLength: 0
      FilterSpanInSymbols: 10
         SamplesPerSymbol: 4
     PayloadLengthInBytes: 10

   Coding and Modulation:
            MappingScheme: "pi/2-BPSK"
                 CodeRate: "1/3"
    PermutationParameters: [9 0 0 0 0]

   Unique Word:
           PreambleLength: 8
          PostambleLength: 8
              PilotPeriod: 0
        PilotBlockLength: 1
              UniqueWord: "FFFF"

   Read-only:
           FramePDULength: 48
```

References

[1] TM Synchronization and Channel Coding. *Recommendation for Space Data System Standards*. CCSDS 131.0-B-3. Blue Book. Issue 3. Washington, D.C.: CCSDS, September 2017.

[2] Flexible Advanced Coding and Modulation Scheme for High Rate Telemetry Applications. *Recommendation for Space Data System Standards*. CCSDS 131.2-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, March 2012.

See Also

`ccsdsTCConfig` | `ccsdsTMWaveformGenerator` | `dvbs2WaveformGenerator` | `dvbs2xWaveformGenerator` | `dvbrcs2WaveformGenerator`

More About

- “Waveform Generation Workflow” on page 2-2
- “Generate Waveforms” on page 2-12

Generate Waveforms

After you create the necessary objects described in “Create Configuration or System Objects” on page 2-3, you can use these objects to generate the desired waveform. Vary the object parameters and plot the waveforms.

In each section of these examples, you:

- Create a format-specific configuration object or a System object.
- Create a column vector or cell array of column vectors of the information bits for the waveform generation.
- Generate the format-specific waveform and plot it.

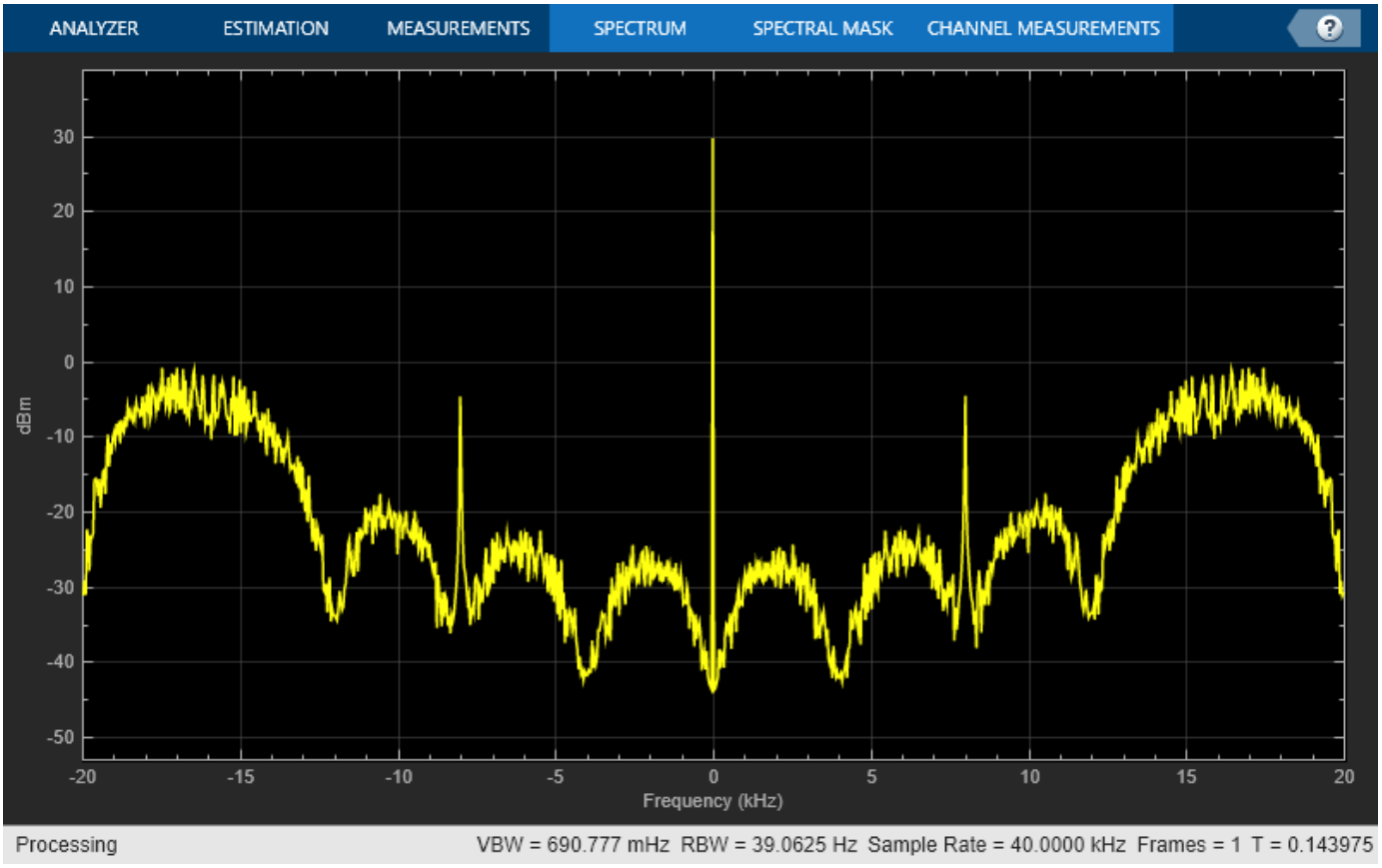
Generate CCSDS TC Waveform

Create a Consultative Committee for Space Data Systems (CCSDS) Telecommand (TC) configuration object and the waveform. Use `Name,Value` pairs to specify the transmission parameters.

```
cfg = ccsdsTCConfig ('ChannelCoding', "LDPC", 'LDPCCodewordLength', 512);
TFLength = 12;                      % Transfer frame length
bits = randi([0 1],8*TFLength,1);  % Bits in TC transfer frame
waveform = ccsdsTCWaveform(bits,cfg);
```

Create a spectrum analyzer System object to display the signal spectrum of the generated CCSDS TC waveform.

```
scope = spectrumAnalyzer;
scope.SampleRate = cfg.SamplesPerSymbol*cfg.SymbolRate;
scope(waveform)
```



Generate CCSDS TM Waveform

Create a Consultative Committee for Space Data Systems (CCSDS) Telemetry (TM) System object and the waveform.

The object `ccsdsTMWaveformGenerator` supports these two CCSDS TM standards, depending on the type of input to `WaveformSource` property.

- CCSDS TM synchronization and channel coding standard (CCSDS 131.0-B-3)
- CCSDS flexible advanced coding and modulation scheme for high rate telemetry standard (CCSDS 131.2-B-1)

The default standard for this object is CCSDS TM synchronization and channel coding.

Create the System object and Generate Waveform for CCSDS TM Synchronization and Channel Coding Scheme

```
tmWaveGen = ccsdsTMWaveformGenerator % CCSDS TM object with defualt properties
```

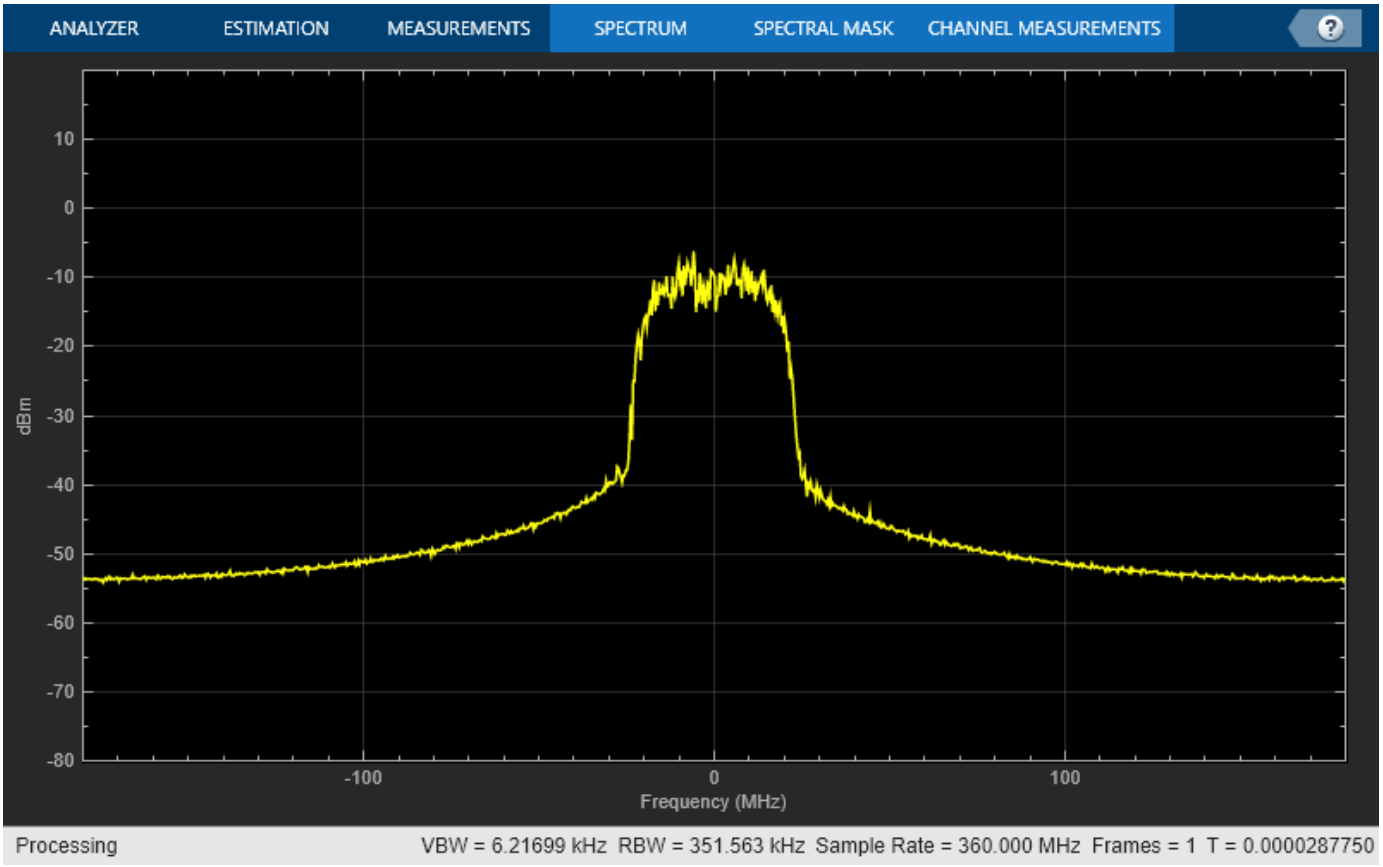
```
tmWaveGen =
  ccsdsTMWaveformGenerator with properties:

          WaveformSource: "synchronization and channel coding"
```

```
          HasRandomizer: true
                HasASM: true
             PCMFormat: "NRZ-L"

   Channel coding
          ChannelCoding: "RS"
        RSMessageLength: 223
     RSInterleavingDepth: 1
    IsRSMessageShortened: false

   Digital modulation and filter
             Modulation: "QPSK"
      PulseShapingFilter: "root raised cosine"
           RolloffFactor: 0.3500
     FilterSpanInSymbols: 10
        SamplesPerSymbol: 10

  Show all properties
```

```
bits = randi([0 1], tmWaveGen.NumInputBits,1); % Input information bits
waveform = tmWaveGen(bits);
```

Create a spectrum analyzer System object to display the frequency spectrum of the generated CCSDS TM time-domain waveform.

```
BW = 36e6;     % Typical satellite channel bandwidth
Fsamp = tmWaveGen.SamplesPerSymbol*BW;
scope = spectrumAnalyzer('SampleRate',Fsamp,'AveragingMethod','Exponential');
scope(waveform)
```



Create the System object and Generate Waveform for CCSDS TM Flexible Advanced Coding and Modulation Scheme

For this example, use dot notation to specify the transmission parameters.

```
tmWaveGen = ccsdsTMWaveformGenerator;
tmWaveGen.WaveformSource = "flexible advanced coding and modulation";
tmWaveGen.ACMFormat = 14;   % 16APSK
```

Calculate the number of transfer frames in one physical layer frame. Generate the waveform using the information bits, `data`.

```
NumTFInOnePLFrame = tmWaveGen.MinNumTransferFrames*16  % One PL frame consists of 16 codewords, a
```

```
NumTFInOnePLFrame = 192
```

```
waveform = [];  % Initialize waveform as null
% Generate waveform
for iTF = 1:NumTFInOnePLFrame
   bits = randi([0 1], tmWaveGen.NumInputBits, 1);
   waveform = [waveform;tmWaveGen(bits)];
end
```

Generate DVB-S2 Waveform

This example uses MAT-files with LDPC parity matrices. If the MAT-files are not available on the path, download and unzip the MAT-files by entering this code at the MATLAB command prompt.

```
if ~exist('dvbs2xLDPCParityMatrices.mat','file')
    if ~exist('s2xLDPCParityMatrices.zip','file')
        url = 'https://ssd.mathworks.com/supportfiles/spc/satcom/DVB/s2xLDPCParityMatrices.zip';
        websave('s2xLDPCParityMatrices.zip',url);
        unzip('s2xLDPCParityMatrices.zip');
    end
addpath('s2xLDPCParityMatrices');
end
```

Create a Digital Video Broadcasting Satellite Second Generation (DVB-S2) System Object and the waveform. Use `Name,Value` pairs to specify the transmission parameters.

```
s2WaveGen = dvbs2WaveformGenerator("NumInputStreams",2,"MODCOD",[6 19],"RolloffFactor",0.25 );
disp(s2WaveGen)
```

```
  dvbs2WaveformGenerator with properties:

          StreamFormat: "TS"
       NumInputStreams: 2
               FECFrame: "normal"
                 MODCOD: [6 19]
                    DFL: 15928
          ScalingMethod: "outer radius as 1"
              HasPilots: 0
          RolloffFactor: 0.2500
    FilterSpanInSymbols: 10
       SamplesPerSymbol: 4
                  ISSYI: false

  Show all properties
```

Initialize the simulation parameters.

```
numFramesPerStream = 1;                                          % Number of PL frames generated p
syncBits = [0 1 0 0 0 1 1 1]';                                   % Sync byte of TS packet (47 HEX)
pktLen = 1496;                                                   % User packet (UP) length withou
numPktsPerStream = s2WaveGen.MinNumPackets*numFramesPerStream;   % Number of packets required to
```

Generate transport stream (TS) packets per stream.

```
data =  cell(s2WaveGen.NumInputStreams,1);
for i = 1:s2WaveGen.NumInputStreams
    txRawPkts = randi([0 1],pktLen,numPktsPerStream(i));
    txPkts = [repmat(syncBits,1,numPktsPerStream(i)); txRawPkts];
    data{i} = txPkts(:);
end
```

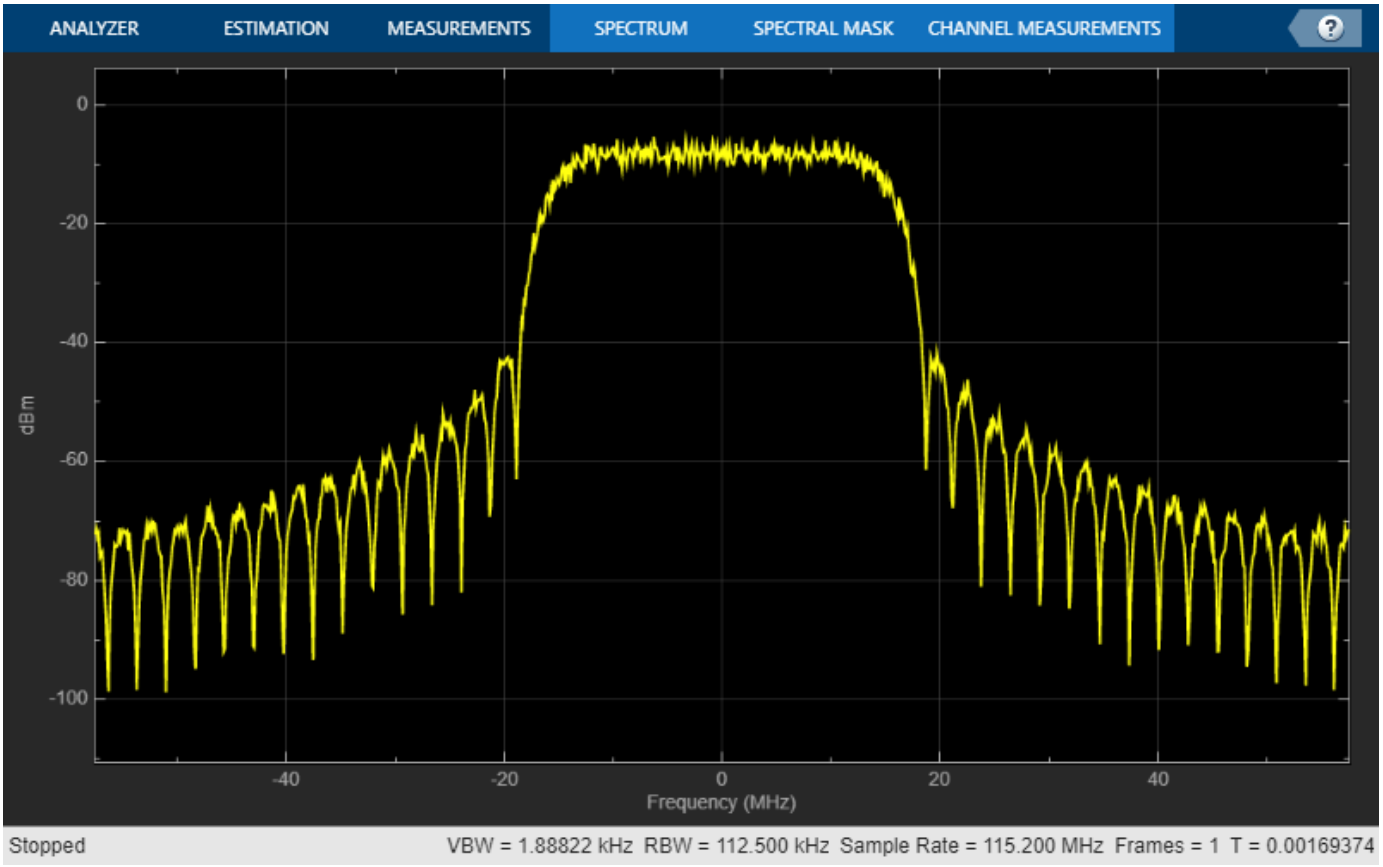
Generate the DVB-S2 time-domain waveform using the information bits, `data`.

```
txWaveform = s2WaveGen(data);
```

Create a spectrum analyzer System object to display the signal spectrum of the generated DVB-S2 time-domain waveform.

```
BW = 36e6;                                          % Typical satellite channel bandwidth
Fsym = BW/(1+s2WaveGen.RolloffFactor);
```

```
Fsamp = Fsym*s2WaveGen.SamplesPerSymbol;
spectrum = spectrumAnalyzer('SampleRate',Fsamp);
spectrum(txWaveform);
release(spectrum);
```



Generate DVB-S2X Waveform

This example uses MAT-files with LDPC parity matrices. If the MAT-files are not available on the path, download and unzip the MAT-files by entering this code at the MATLAB command prompt.

```
if ~exist('dvbs2xLDPCParityMatrices.mat','file')
    if ~exist('s2xLDPCParityMatrices.zip','file')
        url = 'https://ssd.mathworks.com/supportfiles/spc/satcom/DVB/s2xLDPCParityMatrices.zip';
        websave('s2xLDPCParityMatrices.zip',url);
        unzip('s2xLDPCParityMatrices.zip');
    end
addpath('s2xLDPCParityMatrices');
end
```

Create a Digital Video Broadcasting Satellite Second Generation extended (DVB-S2X) System object and the waveform.

```
s2xWaveGen = dvbs2xWaveformGenerator % DVB-S2X object with defualt properties
```

```
s2xWaveGen =
  dvbs2xWaveformGenerator with properties:

          StreamFormat: "TS"
        HasTimeSlicing: false
       NumInputStreams: 1
        PLSDecimalCode: 132
                   DFL: 18448
     PLScramblingIndex: 0
         RolloffFactor: 0.3500
   FilterSpanInSymbols: 10
      SamplesPerSymbol: 4

  Show all properties
```

Initialize the simulation parameters.

```
numFrames = 3;                                  % Number of PL frames generated per stream
syncBits = [0 1 0 0 0 1 1 1]';                  % Sync byte of TS packet (47 HEX)
pktLen = 1496;                                  % User packet (UP) length without sync bits is 149
numPkts = s2xWaveGen.MinNumPackets*numFrames;  % Number of packets required to fill data field pe
```

Generate transport stream (TS) packets per stream.

```
txRawPkts = randi([0 1], pktLen, numPkts);
txPkts = [repmat(syncBits, 1, numPkts);txRawPkts];
data = txPkts(:);
```

Generate the DVB-S2X time-domain waveform using the information bits, `data`.

```
txWaveform = s2xWaveGen(data);
```

Generate DVB-RCS2 Waveform

Create a Digital Video Broadcasting Second Generation Return Channel over Satellite (DVB-RCS2) System object and the waveform. Use `Name,Value` pairs to specify the transmission parameters.

```
wg = dvbrcs2WaveformGenerator("WaveformID",2,"PreBurstGuardLength",6,"SamplesPerSymbol",6);
disp(wg)
```

```
  dvbrcs2WaveformGenerator with properties:

      TransmissionFormat: "TC-LM"
             ContentType: "traffic"
       IsCustomWaveform: false
             WaveformID: 2
    PreBurstGuardLength: 6
   PostBurstGuardLength: 0
    FilterSpanInSymbols: 10
       SamplesPerSymbol: 6

   Read-only:
         FramePDULength: 80
```

Generate a frame protocol data unit (PDU).

```
framePDU = randi([0 1],wg.FramePDULength,1);
```

Generate the DVB-RCS2 burst samples.

```
txWaveform = wg(framePDU);
```

References

[1] TM Synchronization and Channel Coding. *Recommendation for Space Data System Standards*. CCSDS 131.0-B-3. Blue Book. Issue 3. Washington, D.C.: CCSDS, September 2017.

[2] Flexible Advanced Coding and Modulation Scheme for High Rate Telemetry Applications. *Recommendation for Space Data System Standards*. CCSDS 131.2-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, March 2012.

See Also

Objects

`ccsdsTCConfig` | `ccsdsTMWaveformGenerator` | `dvbs2WaveformGenerator` | `dvbs2xWaveformGenerator` | `dvbrcs2WaveformGenerator`

Functions

`ccsdsTCWaveform`

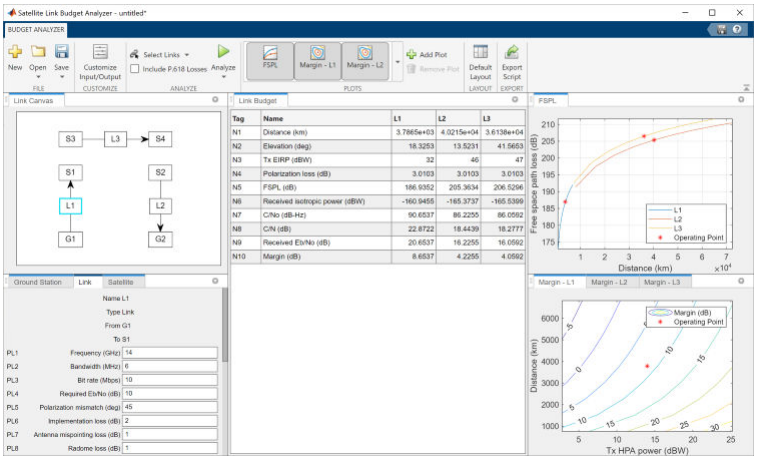
More About

- “Waveform Generation Workflow” on page 2-2
- “Create Configuration or System Objects” on page 2-3

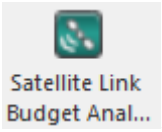
Satellite Link Budget Analyzer App

Get Started with Satellite Link Budget Analyzer App

The **Satellite Link Budget Analyzer** app is an interactive tool for creating, configuring, analyzing, and visualizing link budgets for satellite communications. This section describes the some of the app features. For more information and examples that demonstrate use of the app, see the **Satellite Link Budget Analyzer** documentation page.



Open Satellite Link Budget Analyzer App

On the **Apps** tab in the MATLAB® Toolstrip, under **Signal Processing and Communications**, click the app icon 

. Alternatively, you can open the app by entering `satelliteLinkBudgetAnalyzer` at the MATLAB command prompt.

Budget Analyzer Toolstrip

The app opens to show the **BUDGET ANALYZER** toolstrip and docked tabs. The **BUDGET ANALYZER** toolstrip contains **FILE**, **CUSTOMIZE**, **ANALYZE**, **PLOTS**, and **LAYOUT** ribbons. The default main work area tabs are arranged to show the link canvas, entity (ground station, link, and satellite) properties, results, and plots. You can move the tabbed panes by dragging to rearrange as desired.

Link Canvas

In the **Link Canvas** pane, you can select an individual ground station, link, or satellite entity to bring it into focus in the associated property pane. The default canvas defines these three satellite communication links.

- Link *L1* is an uplink between ground station *G1* and satellite *S1*
- Link *L3* is an crosslink between satellite *S3* and satellite *S4*
- Link *L2* is a downlink between satellite *S2* and ground station *G2*

Properties

Select the **Ground Station**, **Link**, or **Satellite** tab to view the pane and adjust settings of the input properties for each type of entity. Select the entity of interest in the **Link Canvas** tab to bring focus on that entity in the associated property tab. You can adjust the property settings for the entity in focus only.

Link Budget

To compute the link budget results and display them in the **Link Budget** pane, click **Analyze**. The factory default results compute distance, elevation, transmit effective radiated power (EIRP), polarization loss, free-space pathloss (FSPL), received isotropic power, carrier-to-receiver-noise-density ratio (C/N_0), received energy per bit to noise power spectral density ratio (E_b/N_0), and overall link margin.

To open the view or customize the configured input properties and output computations, click the **Customize Input/Output** button.

Visualization Options

View free-space pathloss in 2-D line plots and link margins in 2-D contour plots that characterize the links. You can also create customized 2-D line and 2-D contour plots. When opened, the app shows the default scenario, including link budget results and these plots:

- Free-space pathloss in a 2-D line plot labeled **FSPL** (if created, custom 2-D line plots appear next to the **FSPL** plot)
- Link margin for links *L1*, *L2*, and *L3* in individual 2-D contour plots labeled **Margin - L1**, **Margin - L2**, and **Margin - L3**, respectively

Customize Input/Output Toolstrip

On the **BUDGET ANALYZER** toolstrip, click the **Customize Input/Output** button to open the **CUSTOMIZE INPUT/OUTPUT** toolstrip. The **CUSTOMIZE INPUT/OUTPUT** toolstrip enables you to review, add, delete, and customize the properties for entity and the formulas for the computation of results.

In the **ADD NEW PROPERTY** ribbon, you can customize the app by defining and adding new properties.

In the **ADD NEW RESULT** ribbon, you can customize the app by defining new formulas and adding them as new results.

In the **Properties** pane, you can adjust or delete properties.

In the **Results** pane, you can adjust or delete results.

Generate MATLAB Script

To generate the equivalent MATLAB code for sensitivity analysis, click **Export Script**. You can modify the exported script to study the impact of link budget parameters on link performance.

See Also

Satellite Link Budget Analyzer

Scenario Generation and Visualization

Model, Visualize, and Analyze Satellite Scenario

This example shows how to model satellites in orbit, analyze access between the satellites and the ground stations, and visualize the fields of view and ground tracks of the satellites.

Create Satellite Scenario

Create a satellite scenario with the start time of 02-June-2020 8:23:00 AM UTC and a stop time of five hours later. Set the simulation sample time to `60` seconds.

```
startTime = datetime(2020,6,02,8,23,0);
stopTime = startTime + hours(5);
sampleTime = 60;
sc = satelliteScenario(startTime,stopTime,sampleTime);
```

Add Satellites to Scenario

Add satellites to the scenario from the `threeSatelliteConstellation` TLE file.

```
sat = satellite(sc,"threeSatelliteConstellation.tle");
```

Show the satellites in orbit and plot their ground tracks over 20 minutes.

```
show(sat)
groundTrack(sat,"LeadTime",1200);
```

Return Orbital Elements and Position of Satellites

Display the orbital elements of each satellite in the scenario.

```
ele1 = orbitalElements(sat(1))
```

```
ele1 = struct with fields:
                      MeanMotion: 9.1649e-04
                    Eccentricity: 1.0000e-03
                     Inclination: 55
    RightAscensionOfAscendingNode: 175.0000
              ArgumentOfPeriapsis: 100
                      MeanAnomaly: 174.9900
                           Period: 6.8557e+03
                            Epoch: 02-Jun-2020 18:43:16
                            BStar: 1.0000e-04
```

```
ele2 = orbitalElements(sat(2))
```

```
ele2 = struct with fields:
                      MeanMotion: 8.5025e-04
                    Eccentricity: 1.0000e-03
                     Inclination: 55
    RightAscensionOfAscendingNode: 350.0000
              ArgumentOfPeriapsis: 90
                      MeanAnomaly: 310.0877
                           Period: 7.3898e+03
                            Epoch: 02-Jun-2020 18:33:26
                            BStar: 1.0000e-04
```

```
ele3 = orbitalElements(sat(3))
```

```
ele3 = struct with fields:
                       MeanMotion: 8.6605e-04
                     Eccentricity: 1.0000e-03
                      Inclination: 55
    RightAscensionOfAscendingNode: 270
              ArgumentOfPeriapsis: 95
                      MeanAnomaly: 119.9007
                           Period: 7.2550e+03
                            Epoch: 02-Jun-2020 18:37:40
                            BStar: 1.0000e-04
```

Return the latitude, longitude, and altitude of the first satellite at time 02-June-2020 12:30:00 PM UTC.

```
time = datetime(2020,6,02,12,30,0);
pos = states(sat(1),time,"CoordinateFrame","geographic")
```

```
pos = 3×1
10^6 ×

    0.0000
   -0.0001
    1.4212
```

Add Ground Stations

Specify the latitudes and longitudes of the Madrid and Canberra Deep Space Communications Complexes as ground stations of interest.

```
name = ["Madrid Deep Space Communications Complex", ...
      "Canberra Deep Space Communications Complex"];
lat = [40.43139, -35.40139];
lon = [-4.24806, 148.98167];
gs = groundStation(sc,"Name",name,"Latitude",lat, ...
    "Longitude", lon);
```

Return Azimuth Angle, Elevation Angle, and Range at Specified Time

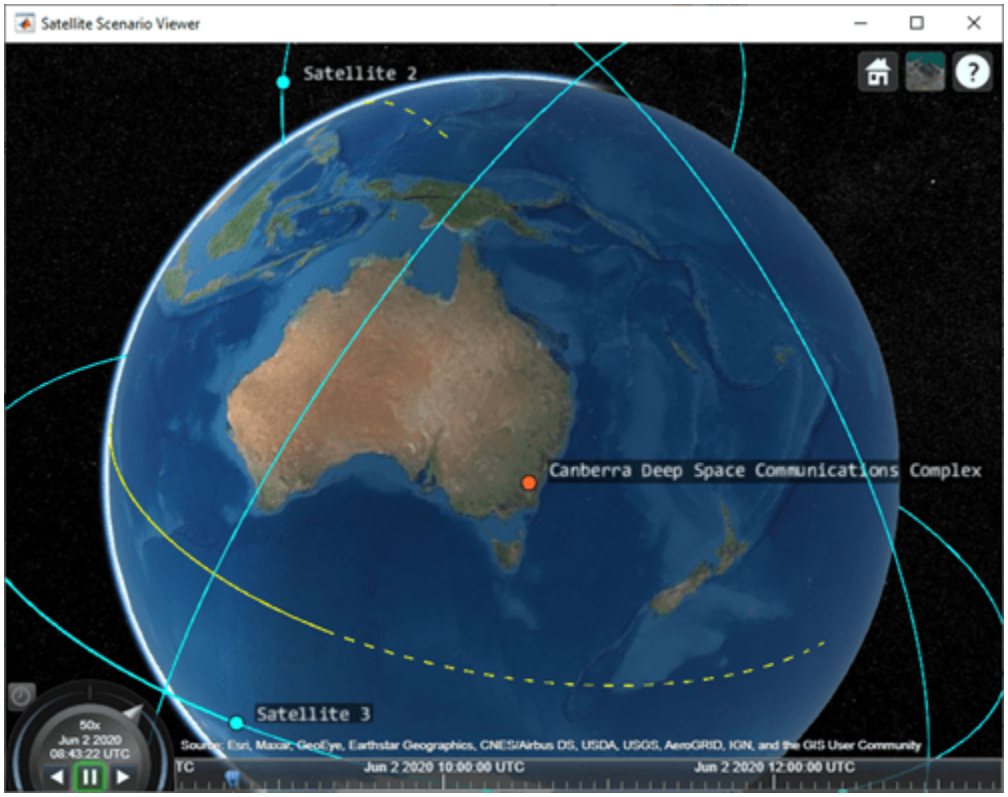
Return the azimuth angle, elevation angle and range of the first satellite with respect to the Madrid Deep Space Communications Complex at time 02-June-2020 12:30:00 PM UTC.

```
time = datetime(2020,6,02,12,30,0);
[az,elev,r] = aer(gs(1),sat(1),time)
```

```
az = 264.2457

elev = -34.0669

r = 9.3088e+06
```

Play the satellite scenario with the satellites and ground stations.

```
play(sc)
```



See Also

Objects
`satelliteScenario` | `satellite` | `Access` | `GroundStation` | `satelliteScenarioViewer` | `ConicalSensor` | `Transmitter` | `Receiver`

Functions
`show` | `play` | `hide`

Related Examples

- “Multi-Hop Satellite Communications Link Between Two Ground Stations”
- “Satellite Constellation Access to Ground Station”
- “Comparison of Orbit Propagators”
- “Modeling Satellite Constellations Using Ephemeris Data”
- “Estimate GNSS Receiver Position with Simulated Satellite Constellations”

More About

- “Satellite Scenario Key Concepts” on page 4-5
- “Satellite Scenario Basics” on page 4-15

Satellite Scenario Key Concepts

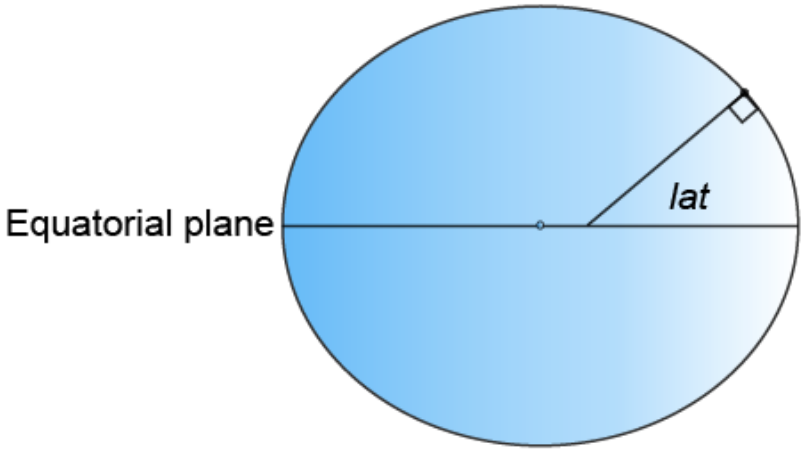
The Satellite Communications Toolbox `satelliteScenario` object provides the ability to model and visualize satellites in orbit, compute access with ground stations, and visualize and analyze communication links. This topic provides an overview of the technical terms frequently encountered in scenario visualization.

Coordinate Systems

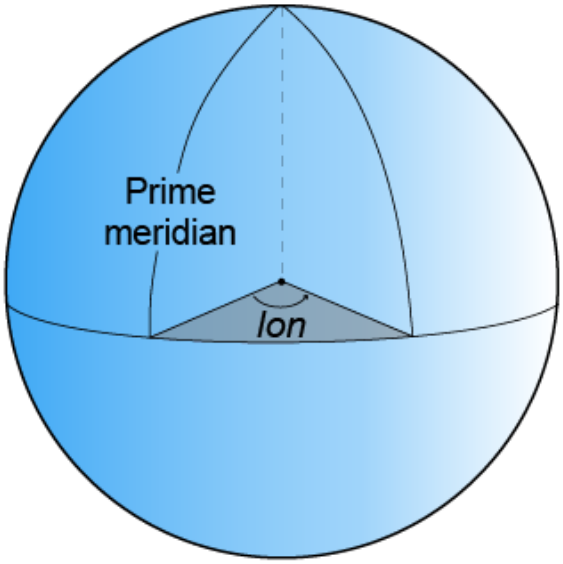
Geodetic Coordinates

A geodetic system uses coordinates (*lat*, *lon*, *h*) to represent position relative to a reference ellipsoid. All geodetic coordinates in a satellite scenario use the World Geodetic System, 1984 (WGS 84), as the reference ellipsoid. The coordinate origin of WGS 84 is located at the center of mass of the Earth.

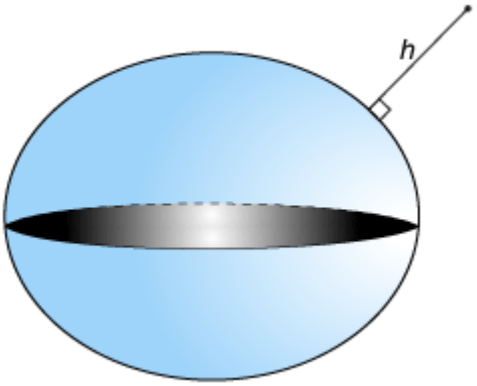
- *lat* is the latitude, which originates at the equator. More specifically, the latitude of a point is the angle normal to the ellipsoid at that point makes with the equatorial plane, which contains the center and equator of the ellipsoid. An angle of latitude is in the range [–90°, 90°]. Positive latitudes correspond to North and negative latitudes correspond to South.



- *lon* is the longitude, which originates at the prime meridian. More specifically, the longitude of a point is the angle that a plane containing the ellipsoid center and the meridian containing that point makes with the plane containing the ellipsoid center and prime meridian. Positive longitudes are measured in a counterclockwise direction from a vantage point above the North Pole. Typically, longitude is in the range [–180°, 180°] or [0°, 360°].



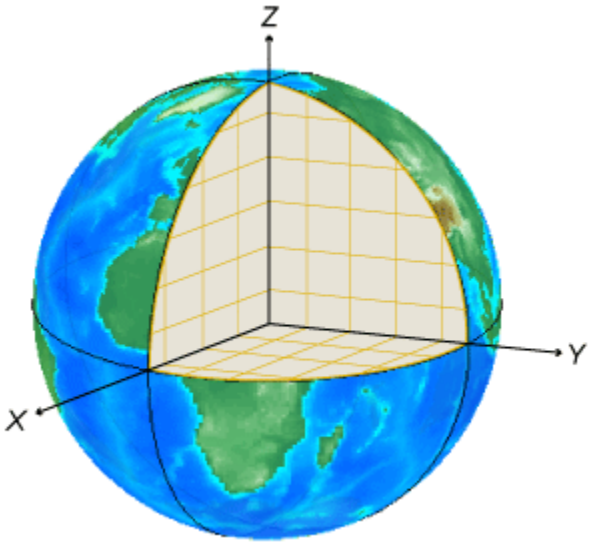
- *h* is the ellipsoidal height, which is measured along a normal of the reference spheroid.



Earth-Centered Earth-Fixed Coordinates

An Earth-centered Earth-fixed (ECEF) system uses Cartesian coordinates (*X*, *Y*, *Z*) to represent position relative to the center of the reference ellipsoid. The distance between the center of the ellipsoid and the center of the Earth depends on the reference ellipsoid. The Satellite scenario uses the WGS 84 reference ellipsoid, which has a center that coincides with the center of mass of the Earth.
[1]

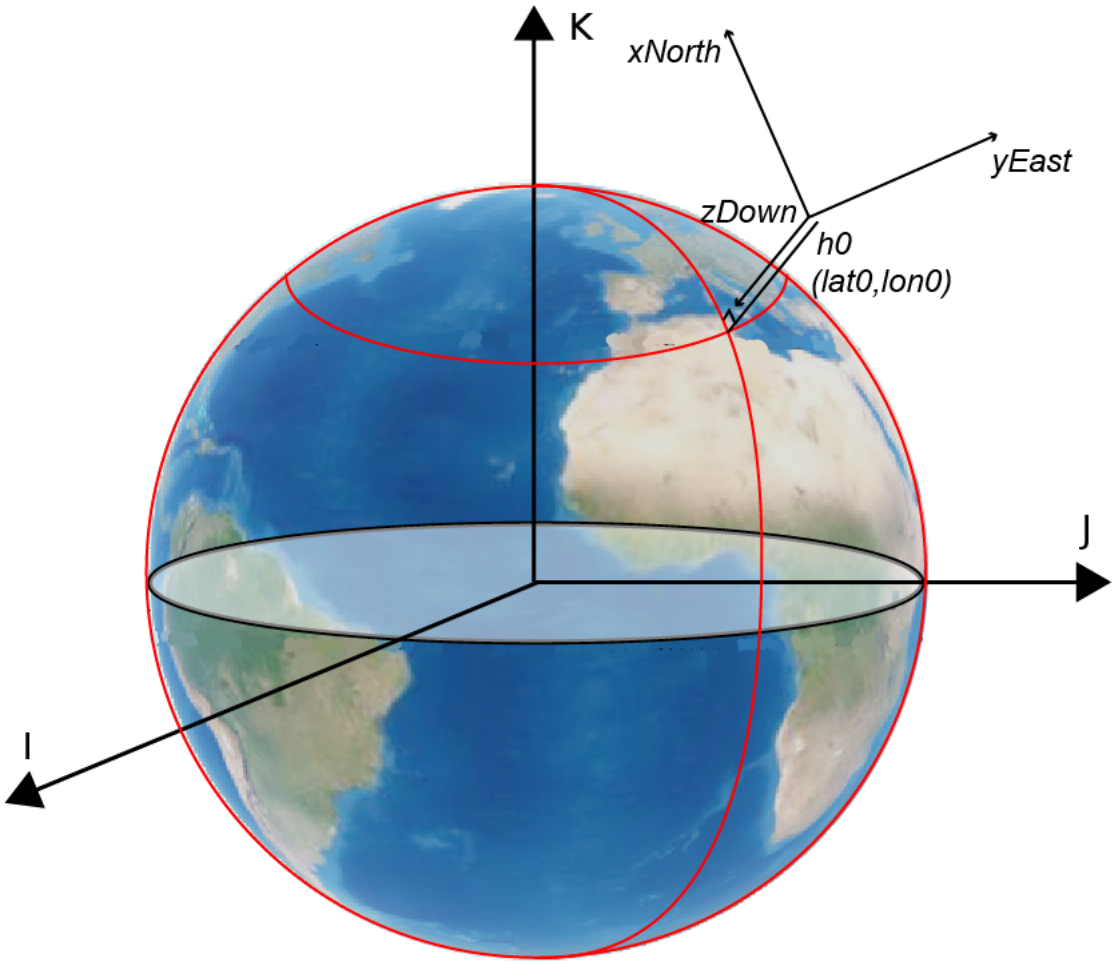
1 Alignment of boundaries and region labels are a presentation of the feature provided by the data vendors and do not imply endorsement by MathWorks®.



- The positive *X*-axis intersects the surface of the ellipsoid at 0° latitude and 0° longitude, where the equator meets the prime meridian.
- The positive *Y*-axis intersects the surface of the ellipsoid at 0° latitude and 90° longitude.
- The positive *Z*-axis intersects the surface of the ellipsoid at 90° latitude.

Frame of Reference and North East Down (NED) Frame

To describe a point in space, you need a frame of reference that does not rotate with respect to the stars. The Geocentric Celestial Reference Frame (GCRF), with the origin at the center of the Earth and orthogonal vectors **I**, **J**, and **K**, is used as the frame of reference when you add `Satellite` objects to a `satelliteScenario` object. The fundamental plane is the **IJ**-plane, which is closely aligned with the equator with a small offset that changes over time because of precession and nutation of the rotation axis of the Earth. When you use orbital elements to add satellites to the satellite scenario, the coordinates are assumed to be defined in the geocentric celestial reference frame.



When you refer to the position, velocity, acceleration, orientation, and angular velocity, you must mention the coordinate system in which these quantities are expressed. Global systems such as GCRF and geodetic systems describe the position of an object using a triplet of coordinates. Local systems such as NED and Azimuth Elevation Range (AER) systems require two triplets of coordinates: one triplet describes the location of the origin, and the other triplet describes the location of the object with respect to the origin.

An NED system uses Cartesian coordinates (*xNorth*, *yEast*, *zDown*) to represent position relative to a local origin. The local origin is described by geodetic coordinates (*lat0*, *lon0*, *h0*). Typically, the local origin of an NED system is above the surface of the Earth.

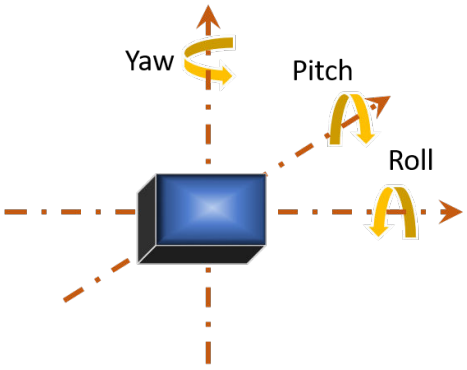
- The positive *xNorth*-axis points North along the meridian of longitude containing *lon0*.
- The positive *yEast*-axis points East along the parallel of latitude containing *lat0*.
- The positive *zDown*-axis points South along the ellipsoid normal.

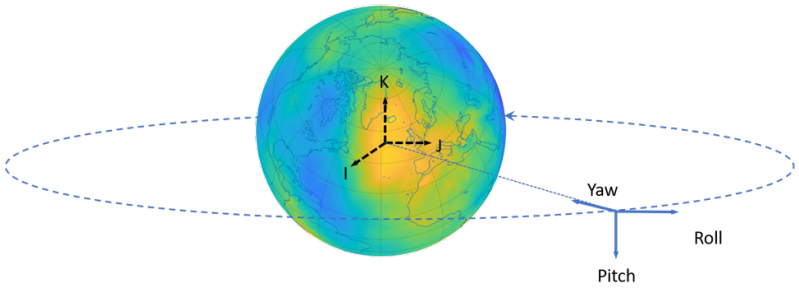
An NED coordinate system is commonly used to specify location relative to a moving satellite. In this case, the coordinates are not fixed to the frame of the satellite, but to the point on the surface of the WGS 84 ellipsoid corresponding to the latitude and longitude of the satellite.

Roll, Pitch, and Yaw

Three lines run through a satellite and intersect at right angles at the center of mass of the satellite. These axes are fixed with respect to the satellite and are used to define the orientation of the satellite with respect to the NED frame. The orientation is defined by rotations in this sequence:

1. Rotation about the yaw axis by the yaw angle
2. Rotation about the pitch axis by the pitch angle
3. Rotation about the roll axis by the roll angle



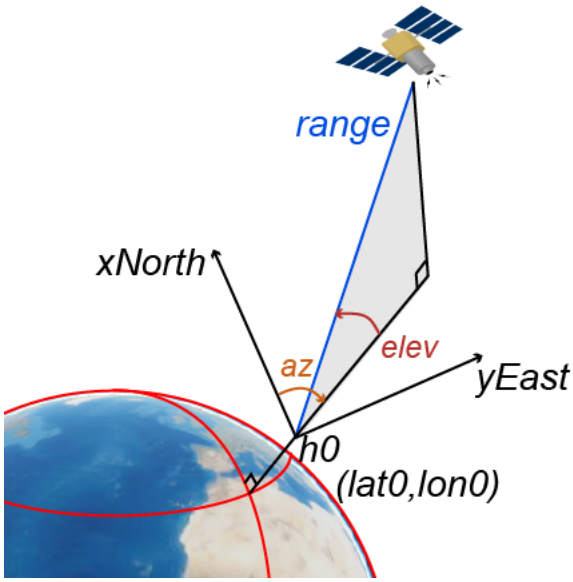


The yaw, pitch, and roll angles have positive clockwise directions when looking in the positive direction of the axes. Unless otherwise specified, the Satellite Communications Toolbox uses the yaw-pitch-roll rotation order for these angles by default.

Azimuth-Elevation-Range (AER) Coordinates

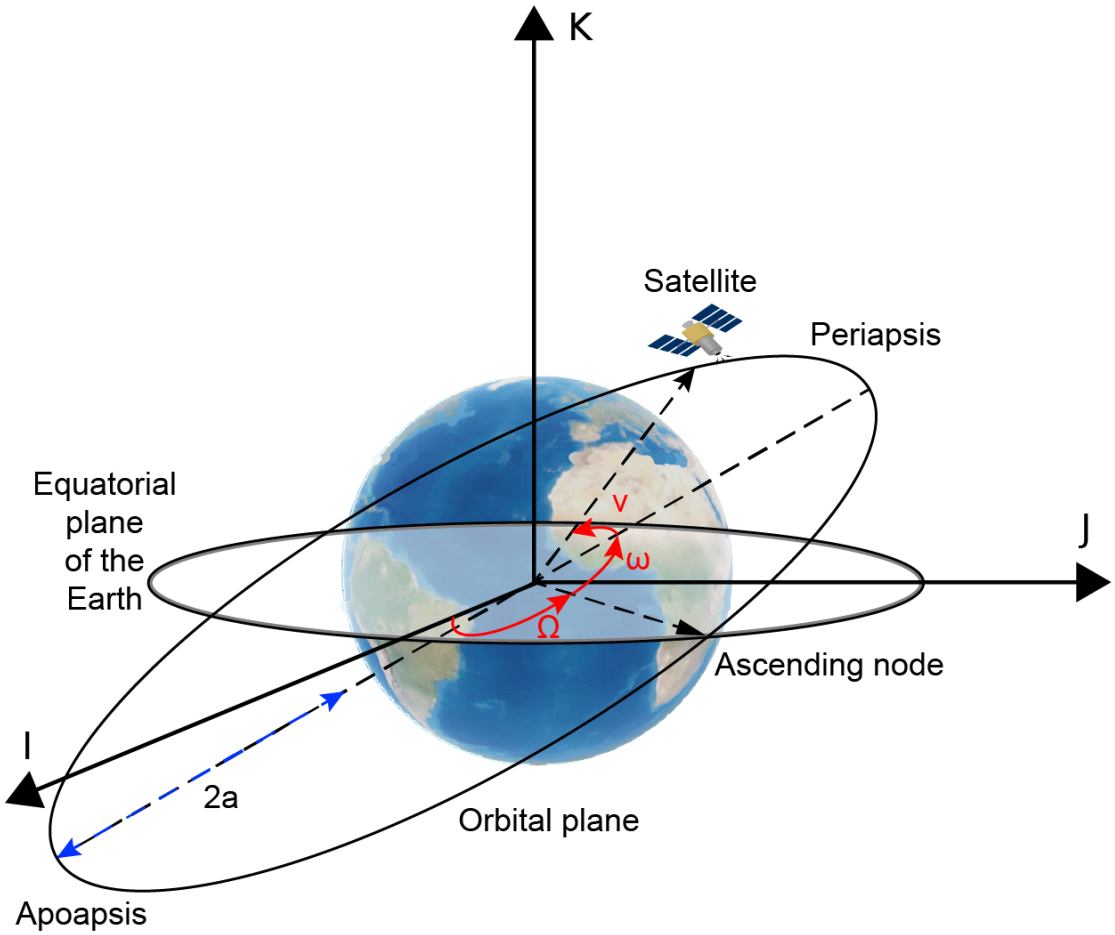
An AER system uses spherical coordinates (*az*, *elev*, *range*) to represent position relative to a local origin. The local origin is described by geodetic coordinates (*lat0*, *lon0*, *h0*). Azimuth, elevation, and slant ranges are dependent on a local Cartesian system (for example, an NED system) with the origin centered at the center of mass of the satellite.

- *az* is the azimuth, which is the clockwise angle in the (*x*North)(*y*East)-plane from the positive *y*East-axis to the projection of the object into the plane.
- *elev* is the elevation, which is the angle from the (*x*North)(*y*East)-plane to the object.
- *range* is the slant range, which is the Euclidean distance between the object and the local origin.



Orbital Elements

Orbital elements are parameters required to uniquely identify a specific orbit. Uniquely defining an orbit and the position of a satellite within satellite orbit requires at least six parameters. Three of the parameters describe what the orbital plane looks like and the position of the satellite in the ellipse. The other three parameters describe how that plane is oriented in the celestial inertial reference frame and the location of the satellite in that plane. These six parameters are called the Keplerian elements or orbital elements.



In this diagram, the orbital plane (yellow) intersects a reference plane (gray). For Earth-orbiting satellites, the reference plane is usually the **IJ**-plane of the GCRF.

These two elements define the shape and size of the ellipse:

- Eccentricity (*e*) — Describe how elongated the shape of the ellipse is when compared to a circle.
- Semimajor axis (*a*) — The sum of the periapsis and apoapsis distances divided by 2. Periapsis is the point at which an orbiting object is closest to the center of mass of the body it is orbiting. Apoapsis is the point at which an orbiting object is farthest away from the center of mass of the body it is orbiting. For classic two-body orbits, the semimajor axis is the distance between the centers of the bodies.

These two elements define the orientation of the orbital plane in which the ellipse is embedded:

- Inclination (*i*) — The vertical tilt of the ellipse with respect to the reference plane measured at the ascending node. The ascending node is the location where the orbit passes upward through the reference plane (the green angle *i* in the diagram). The tilt angle is measured perpendicular to the line of intersection between orbital plane and the reference plane. Any three points on an ellipse defines the ellipse orbital plane.

Starting with an equatorial orbit, the orbital plane can be tilted up. The angle it is tilted up from the equator is referred to as the inclination angle, *i*, in the range [0,180]. Because the center of the Earth must always be in the orbital plane, the point in the orbit where the satellite passes the equator on its way up north-bound through the orbit is the ascending node, and the point where the satellite passes the equator on its way down south-bound is the descending node. Drawing a line through these two points on the equator defines the line of nodes.

- Right ascension of ascending node (Ω) — The horizontal orientation of the ascending node of the ellipse (where the orbit passes upward through the reference plane) with respect to the **I**-axis of the reference frame. The rotation of the right ascension of the ascending node (RAAN) can be any number between 0° and 360°.

The remaining two elements are:

- Argument of periapsis (ω) — The orientation of the ellipse in the orbital plane, as an angle measured from the ascending node to the periapsis in the range [0, 360).
- True Anomaly (*ν*) — The position of the orbiting body along the ellipse at a specific time. The position of the satellite on the path is measured counterclockwise from the periapsis and is called the true anomaly, *ν*, in the range [0, 360).

Two Line Element (TLE) Files

The Satellite Communications Toolbox `satellite` function accepts a TLE file as a possible input to initialize the satellite. To download TLE files, see the Space track website.

A TLE set is a data format encoding a list of orbital elements of an Earth orbiting object for a given point in time, the *epoch*. Orbital element parameters can be encoded as text in a variety of formats. The most common format is the National Aeronautics and Space Administration (NASA) or North American Aerospace Defense Command (NORAD) TLE format. In this format, each satellite has three rows: the first row contains the name of the satellite, and the next two rows are the standard TLE.

Data for each satellite consists of three lines, as this example shows.

```
Satellite 1
1 25544U 98067A   04236.56031392  .00020137  00000-0  16538-3 0  9993
2 25544  51.6335 344.7760 0007976 126.2523 325.9359 15.70406856328906
```

- Row 1 is an eleven-character satellite name.
- Rows 2 and 3 are the standard TLE set format identical to that used by NORAD and NASA.

  This table describes the columns in row 2.

Column	Description	Example
1	Line number	`1`
3–7	Satellite number	`25544`
8	Elset classification	`U`
10–17	International designator	`98067A`
19–32	Element set epoch (UTC)	`04236.56031392`
34–43	First derivative of the mean motion with respect to time	`.00020137`

Column	Description	Example
45–52	Second derivative of the mean motion with respect to time (decimal point assumed)	`00000-0`
54–61	BSTAR drag term	`16538-3`
63	Element set type	`0`
65–68	Element number	`999`
69	Check sum (modulo 10)	`3`

This table describes the columns in row 3.

Column	Description	Example
1	Line number of element data	`2`
3–7	Satellite number	`25544`
9–16	Inclination (in degrees)	`51.6335`
18–25	Right ascension of the ascending node (in degrees)	`344.7760`
27–33	Eccentricity (leading decimal point assumed)	`0007976`
35–42	Argument of perigee (in degrees)	`126.2523`
44–51	Mean anomaly (in degrees)	`325.9359`
53–63	Mean motion (in revs per day)	`15.70406856`
64–68	Revolution number at epoch (in revs)	`32890`
69	Check sum (modulo 10)	`6`

Depending on the application and object orbit, the data derived from TLEs older than 30 days can become unreliable. Calculate the orbital positions from TLEs using the Simplified General Perturbations-4 (SGP4) and Simplified Deep-Space Perturbations-4 (SDP4) algorithms.

References

[1] “Basics of Space Flight” https://solarsystem.nasa.gov/basics/chapter5-1/.

[2] “CelesTrak. Frequently Asked Questions: Two-Line Element Set Format.” Accessed March 26, 2016. https://celestrak.com/columns/v04n03/.

See Also

Objects
`satelliteScenario` | `satellite` | `access` | `groundStation` | `satelliteScenarioViewer` | `conicalSensor`

Functions
`show` | `play` | `hide`

Related Examples

- “Multi-Hop Satellite Communications Link Between Two Ground Stations”

- “Satellite Constellation Access to Ground Station”
- “Comparison of Orbit Propagators”
- “Modeling Satellite Constellations Using Ephemeris Data”
- “Estimate GNSS Receiver Position with Simulated Satellite Constellations”
- “Model, Visualize, and Analyze Satellite Scenario” on page 4-2
- “Satellite Scenario Basics” on page 4-15

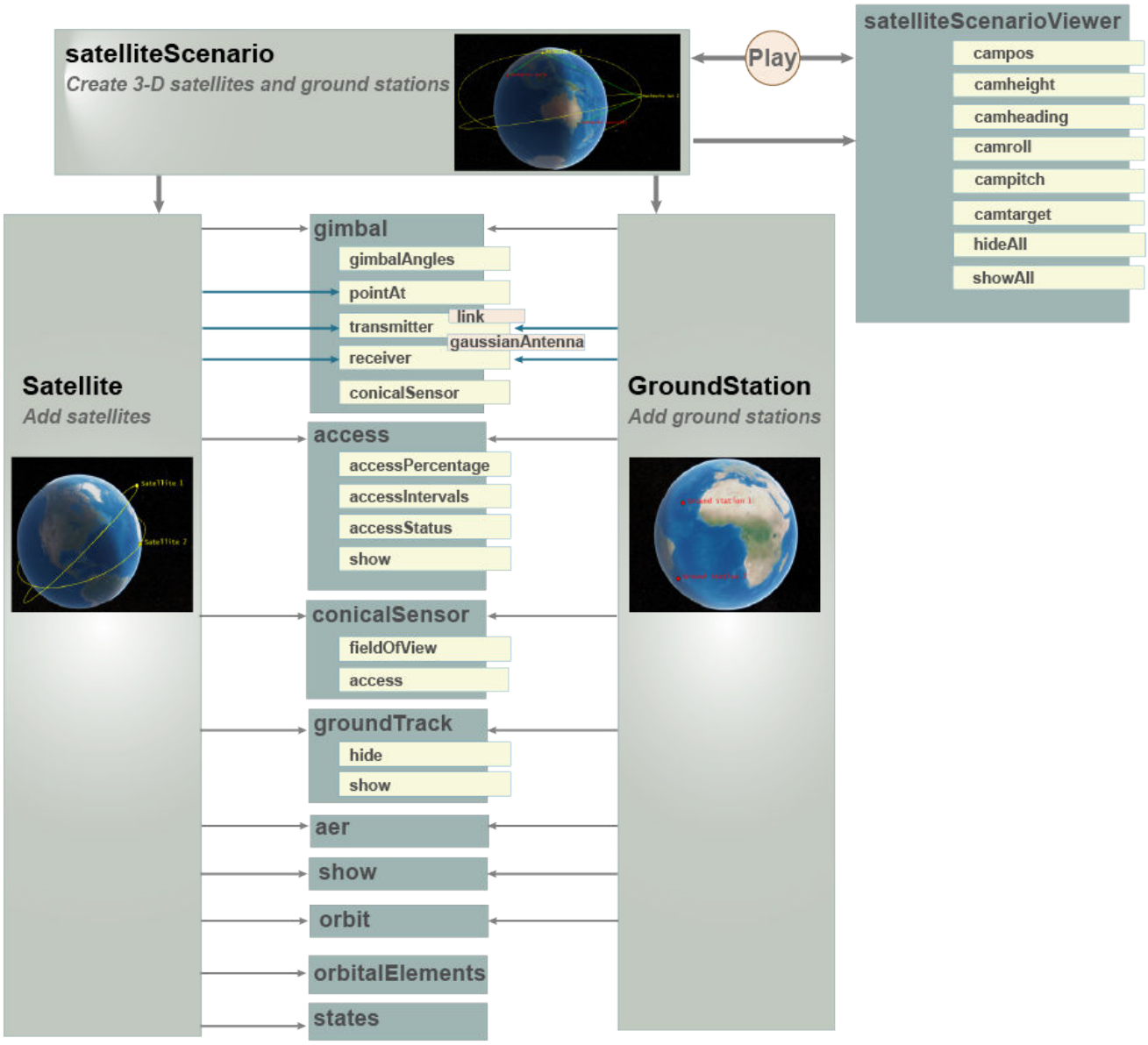
Satellite Scenario Basics

You can build a complete satellite scenario simulation by using the Satellite Communications Toolbox functions and objects. The workflow for satellite scenario simulation consists of these five main components:

- The `satelliteScenario` object represents a 3-D arena consisting of satellites, ground stations, and the interactions between them. Use this object to model satellite constellations, model ground station networks, perform access analysis between the satellites and ground stations, and visualize the results.
- The `satellite` function adds satellites to the scenario using two line element (TLE) files or orbital elements. For more details on orbital elements and TLE files, see “Two Line Element (TLE) Files” on page 4-12.
- The `groundStation` object adds ground stations to the scenario using default parameters or the specified latitude and longitude.
- The `satelliteScenarioViewer` function creates a 3D viewer for the scenario.
- The `play` function simulates the satellite scenario and plays the results in the visualization window specified by `satelliteScenarioViewer`.

This diagram shows the architecture of satellite scenario simulation. Some of the object in the simulation share object functions.

- The `Satellite`, `GroundStation`, and `ConicalSensor` objects share the `access` object function.
- The `Satellite`, `GroundStation`, `GroundTrack`, and `Access` objects share the `show` and `hide` object functions.
- The `Satellite`, `GroundStation`, and `Gimbal` objects share the `conicalSensor` object function.
- The `Satellite` and `GroundStation` objects share the `transmitter` and `receiver` object functions.
- The `Transmitter` and `Receiver` objects share the `link` object function.



See Also

Objects

`satelliteScenario`

Functions

`show` | `play` | `hide` | `satellite` | `access` | `groundStation` | `satelliteScenarioViewer` | `conicalSensor` | `transmitter` | `receiver`

Related Examples

- “Multi-Hop Satellite Communications Link Between Two Ground Stations”

- “Satellite Constellation Access to Ground Station”
- “Comparison of Orbit Propagators”
- “Modeling Satellite Constellations Using Ephemeris Data”
- “Estimate GNSS Receiver Position with Simulated Satellite Constellations”
- “Model, Visualize, and Analyze Satellite Scenario” on page 4-2
- “Satellite Scenario Key Concepts” on page 4-5

5

Channel Modeling

Earth-Space Propagation Losses

Propagation environments have significant effects on the design of Earth-space links for satellite communications systems. The ionospheric effects on Earth-space links become significant at frequencies below 1 GHz. Effects of the nonionized atmosphere become critical above about 1 GHz and for low elevation angles. Recommendation ITU-R P.618 [1] predicts propagation parameters, which are required in the planning of Earth-space systems operating in either the Earth-to-space or space-to-Earth direction. P.618 deals with only the effects of the troposphere such as rain attenuation, gaseous attenuation, precipitation and cloud attenuation, and attenuation due to tropospheric scintillation.

In some cases, you might want to provide continuous, high-quality transmission of voice, data, and television signals. In these cases, you can use the `p618Config` object to model tropospheric effects such as rain attenuation, gaseous attenuation, cloud and fog attenuation, and attenuation due to tropospheric scintillation. You can then use the `p618PropagationLosses` function, which initializes configuration parameters settings, to calculate the Earth-space propagation losses, cross-polarization discrimination, and sky noise temperature of the ground station antenna.

Heavy rainfall causes large attenuation values on an Earth-space link. Site diversity enables the rerouting of link traffic to alternate Earth stations, which improves the system reliability. Site diversity systems are classified as one of these options.

- Balanced: The attenuation thresholds on the two links are equal.
- Unbalanced: The attenuation thresholds on the two links are not equal.

In the case of two Earth stations existing, you can use the `p618SiteDiversityConfig` object to model the parameters required for the calculation of the outage probability due to rain attenuation. Intense rain can cause large attenuation values on an Earth-space link. Then, you can use the `p618SiteDiversityOutage` function, which initializes configuration parameters settings, to calculate the outage probability due to rain attenuation with site diversity.

Earth-Space Propagation Losses

This example requires MAT-files with digital maps from International Telecommunication Union (ITU) documents. If the files are not available on the path, execute these commands to download and untar the MAT-files.

```
maps = exist('maps.mat','file');
p836 = exist('p836.mat','file');
p837 = exist('p837.mat','file');
p840 = exist('p840.mat','file');
matFiles = [maps p836 p837 p840];
if ~all(matFiles)
    if ~exist('ITURDigitalMaps.tar.gz','file')
        url = 'https://www.mathworks.com/supportfiles/spc/P618/ITURDigitalMaps.tar.gz';
        websave('ITURDigitalMaps.tar.gz',url);
        untar('ITURDigitalMaps.tar.gz');
    else
        untar('ITURDigitalMaps.tar.gz');
    end
    addpath(cd);
end
```

This example shows how to parameterize and calculate propagation losses for the design of Earth-space systems.

The propagation losses calculated by the `p618PropagationLosses` function are:

- Attenuation by atmospheric gases
- Attenuation by rain
- Precipitation and cloud attenuation
- Attenuation due to tropospheric scintillation
- Total atmospheric attenuation

Earth-space propagation losses are modeled to depend on the frequency, geographic location, and elevation angle. Based on the propagation conditions, at elevation angles above 10°, only the attenuations cause by atmospheric gases, rain, cloud, and tropospheric scintillation are significant.

Configure the Earth-Space Propagation Parameters

Create a default P.618 configuration object. Change property values, and then display the object properties.

```
cfg = p618Config;
cfg.Frequency = 25e9;                % Signal frequency in Hz
cfg.ElevationAngle = 45;
cfg.Latitude = 30;                   % North direction
cfg.Longitude = 120;                 % East direction
cfg.TotalAnnualExceedance = 0.001;   % Time percentage of excess for the total
                                     % Attenuation per annum
cfg.AntennaEfficiency = 0.65;
disp(cfg);
```

```
  p618Config with properties:

                     Frequency: 2.5000e+10
                ElevationAngle: 45
                      Latitude: 30
                     Longitude: 120
           GasAnnualExceedance: 1
         CloudAnnualExceedance: 1
          RainAnnualExceedance: 1
  ScintillationAnnualExceedance: 1
         TotalAnnualExceedance: 1.0000e-03
         PolarizationTiltAngle: 0
               AntennaDiameter: 1
            AntennaEfficiency: 0.6500
```

Calculate the Propagation Losses in Light Rainfall

Find the propagation losses (`pl`), cross-polarization discrimination (`xpd`), and sky noise temperature (`tsky`) in light rainfall of 1 mm/hr, specifying an Earth station height of 0.5 km.

The fields of propagation losses, `pl`, describe these attenuations.

- `Ag`: Gaseous attenuation (in dB)
- `Ac`: Cloud and fog attenuation (in dB)

- `Ar`: Rain attenuation (in dB)
- `As`: Attenuation due to tropospheric scintillation (in dB)
- `At`: Total atmospheric attenuation (in dB)

```
[pl,xpd,tsky] =  p618PropagationLosses(cfg, ...
                              'StationHeight',0.5, ...
                              'WaterVaporDensity',2.8, ...
                              'TotalColumnarContent',1.4, ...
                              'RainRate',1)
```

```
pl = struct with fields:
    Ag: 1.6393
    Ac: 1.2010
    Ar: 0.0811
    As: 0.3010
    At: 6.6514


xpd = 73.1657

tsky = 214.6132
```

Outage Probability due to Rain Attenuation with Site Diversity

This example requires MAT-files with digital maps from ITU documents. If the files are not available on the path, execute these commands to download and untar the MAT-files.

```
maps = exist('maps.mat','file');
p836 = exist('p836.mat','file');
p837 = exist('p837.mat','file');
p840 = exist('p840.mat','file');
matFiles = [maps p836 p837 p840];
if ~all(matFiles)
    if ~exist('ITURDigitalMaps.tar.gz','file')
        url = 'https://www.mathworks.com/supportfiles/spc/P618/ITURDigitalMaps.tar.gz';
        websave('ITURDigitalMaps.tar.gz',url);
        untar('ITURDigitalMaps.tar.gz');
    else
        untar('ITURDigitalMaps.tar.gz');
    end
    addpath(cd);
end
```

This example shows how to calculate outage probability due to rain attenuation with site diversity.

The `p618SiteDiversityOutage` function applies to unbalanced and balanced systems and computes the joint probability of exceeding attenuation thresholds.

Configure P.618 Site Diversity Parameters

Create a default P.618 site diversity configuration object. Change property values, and then display the object properties.

```
cfgSD = p618SiteDiversityConfig;
cfgSD.Frequency = 25e9;                 % Signal frequency in Hz
cfgSD.Latitude = [30 60];               % North direction
```

```
cfgSD.Longitude = [120 150];            % East direction
cfgSD.PolarizationTiltAngle = [-90 90];
cfgSD.AttenuationThreshold =  [7 7];    % Attenuation threshold on the two links
cfgSD.SiteDistance = 50;                % Separation between the two sites
disp(cfgSD);
```

```
  p618SiteDiversityConfig with properties:

               Frequency: 2.5000e+10
          ElevationAngle: [52.4099 52.4852]
                Latitude: [30 60]
               Longitude: [120 150]
    PolarizationTiltAngle: [-90 90]
             SiteDistance: 50
     AttenuationThreshold: [7 7]
```

Calculate Outage Probability

Calculate the outage probability due to rain attenuation for the specified site diversity configuration.

```
outage = p618SiteDiversityOutage(cfgSD, ...
                                 'RainAnnualExceedances',[0.01 0.01 0.03 0.05 0.1 0.2], ...
                                 'RainProbability1',0.3, ...
                                 'RainProbability2',0.4);
disp(outage);
```

```
    0.0030
```

Attenuation due to Atmospheric Gases

This example requires MAT-files with digital maps from ITU documents. If the files are not available on the path, execute these commands to download and untar the MAT-files.

```
maps = exist('maps.mat','file');
p836 = exist('p836.mat','file');
p837 = exist('p837.mat','file');
p840 = exist('p840.mat','file');
matFiles = [maps p836 p837 p840];
if ~all(matFiles)
    if ~exist('ITURDigitalMaps.tar.gz','file')
        url = 'https://www.mathworks.com/supportfiles/spc/P618/ITURDigitalMaps.tar.gz';
        websave('ITURDigitalMaps.tar.gz',url);
        untar('ITURDigitalMaps.tar.gz');
    else
        untar('ITURDigitalMaps.tar.gz');
    end
    addpath(cd);
end
```

This example shows how to calculate gaseous attenuation for a specified range of frequencies when designing the Earth-space systems.

Gaseous attenuation is modeled to depend on the signal frequency, elevation angle, Earth station height, and water vapor density. Based on the propagation conditions, gaseous attenuations can be significant with frequencies above 10 GHz and neglected at frequencies below 10 GHz.

Configure P.618 Propagation Parameters

Create a default P.618 configuration object. Change property values, and then display the object properties.

```
cfg = p618Config;
cfg.Latitude = 51.5;          % North direction
cfg.Longitude = -0.14;        % West direction
cfg.GasAnnualExceedance = 10; % Time percentange of excess for the gaseous attenuation per annum
cfg.ElevationAngle = 31.076;
```

Set the signal frequency range to the interval from 5 to 55 GHz.

```
freq_range = 5e9:1e9:55e9;
```

Calculate Gaseous Attenuation

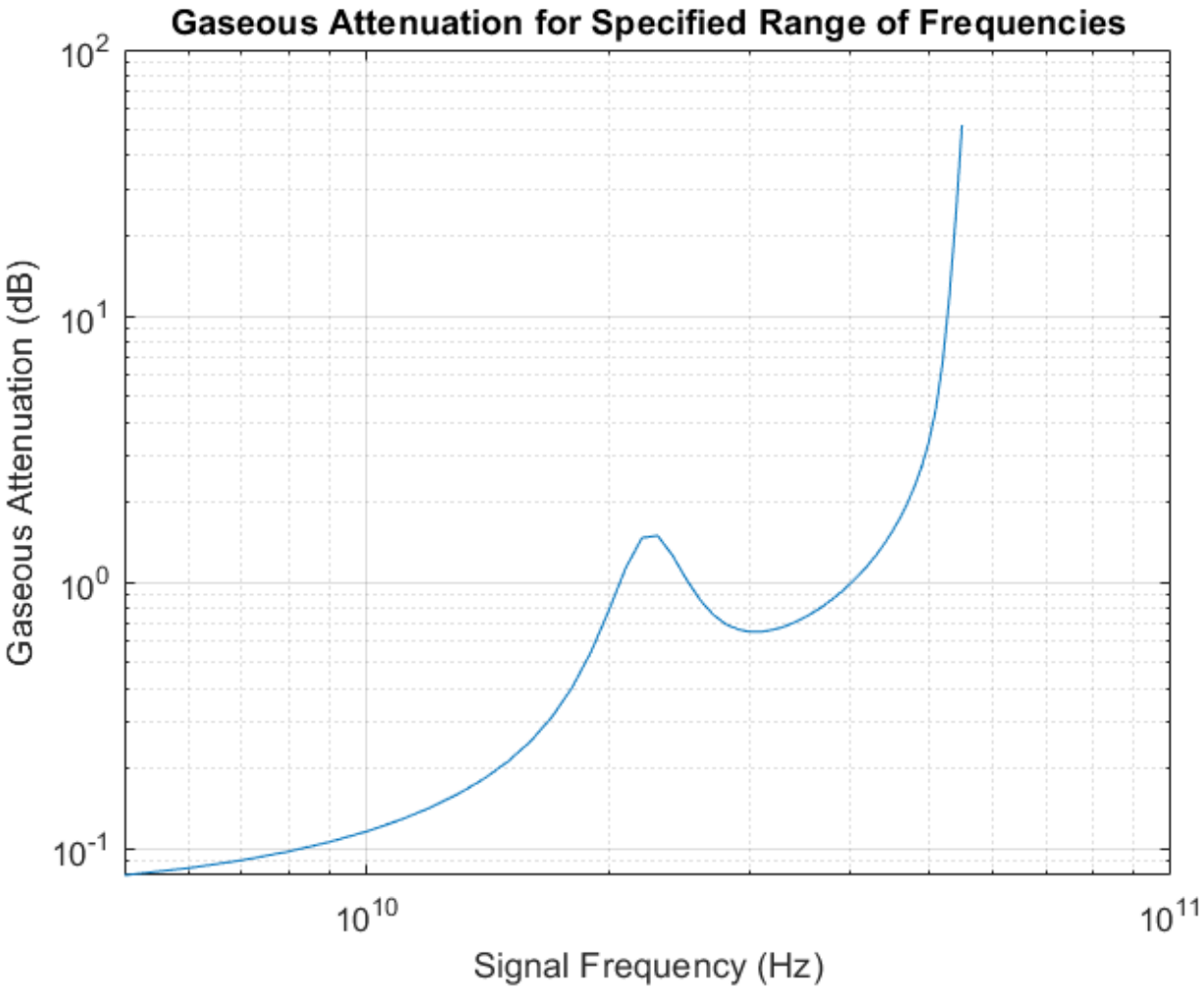
Calculate the attenuation due to atmospheric gases for the specified configuration parameters.

```
gaseous_attenuation = zeros(size(freq_range));
for n = 1:numel(freq_range)
    cfg.Frequency = freq_range(n);
    pl =  p618PropagationLosses(cfg, ...
                            'StationHeight',0.031, ...
                            'Temperature',283.6, ...
                            'Pressure',1009.48, ...
                            'WaterVaporDensity',13.79);
    gaseous_attenuation(n) = pl.Ag;
end
```

Plot Gaseous Attenuation

On a logarithmic scale, plot the gaseous attenuations for the given range of frequencies.

```
loglog(freq_range,gaseous_attenuation);
grid on;
xlabel('Signal Frequency (Hz)');
ylabel('Gaseous Attenuation (dB)');
title('Gaseous Attenuation for Specified Range of Frequencies');
```



Propagation Losses for Specified Range of Elevation Angle

This example requires MAT-files with digital maps from ITU documents. If the files are not available on the path, execute these commands to download and untar the MAT-files.

```
maps = exist('maps.mat','file');
p836 = exist('p836.mat','file');
p837 = exist('p837.mat','file');
p840 = exist('p840.mat','file');
matFiles = [maps p836 p837 p840];
if ~all(matFiles)
    if ~exist('ITURDigitalMaps.tar.gz','file')
        url = 'https://www.mathworks.com/supportfiles/spc/P618/ITURDigitalMaps.tar.gz';
        websave('ITURDigitalMaps.tar.gz',url);
        untar('ITURDigitalMaps.tar.gz');
    else
        untar('ITURDigitalMaps.tar.gz');
    end
```

```
    addpath(cd);
end
```

This example shows how to parameterize and calculate Earth-space propagation losses for a specified range of elevation angles when designing Earth-space systems.

Configure P.618 Propagation Parameters

Create a default P.618 configuration object. Change property values, and then display the object properties.

```
cfg = p618Config;
cfg.Frequency = 14.25e9;  % Signal frequency in Hz
cfg.Latitude = 51.5;      % North direction
cfg.Longitude = -0.14;    % West direction
```

Set the elevation angle range to the interval from 5 to 90 degrees.

```
elev_range = 5:5:90;
```

Calculate Earth-Space Propagation Losses

Calculate the Earth-space propagation losses for the specified configuration parameters.

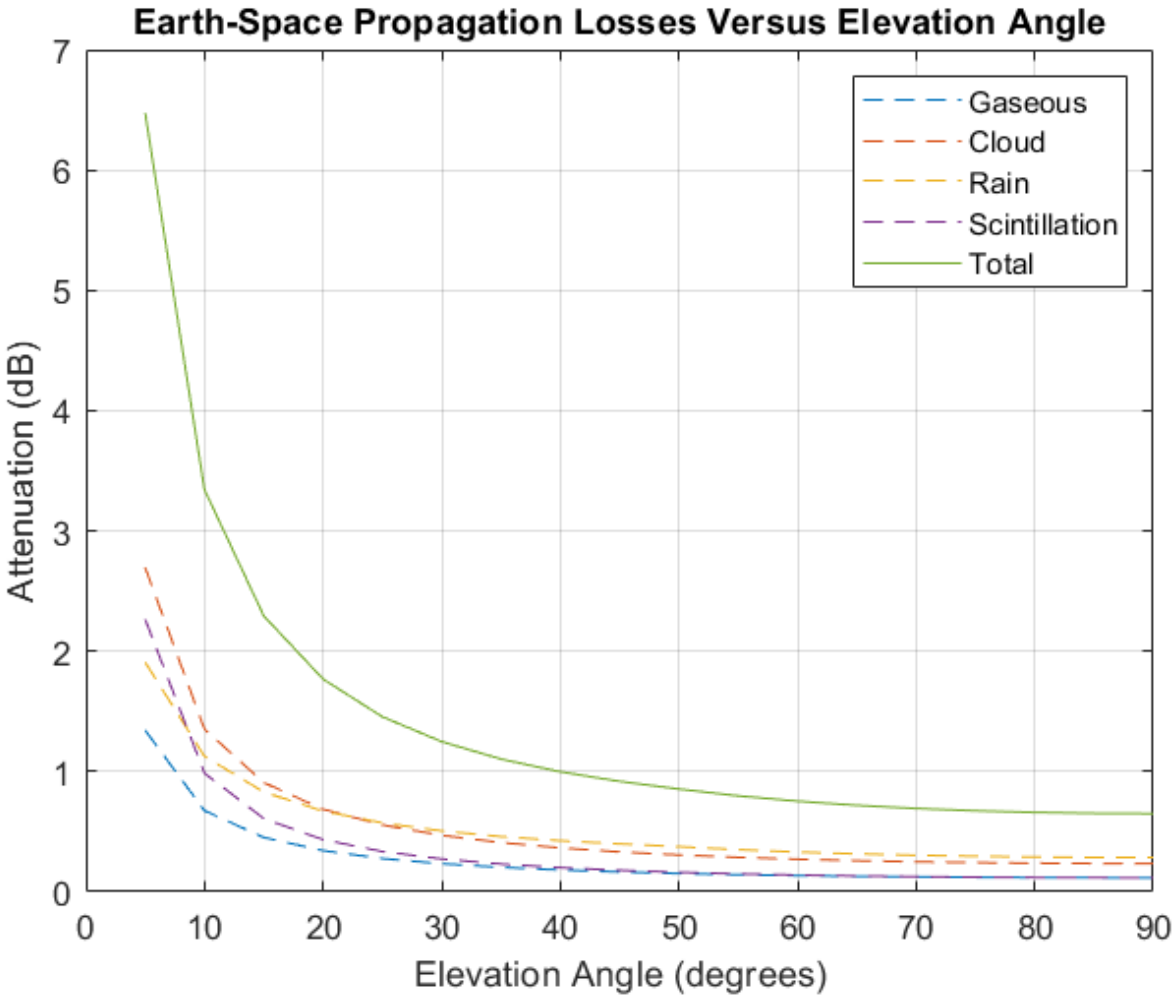
```
elevation_angle = size(elev_range);
gaseous_attenuation = zeros(elevation_angle);
cloud_attenuation = zeros(elevation_angle);
rain_attenuation = zeros(elevation_angle);
scintillation_attenuation = zeros(elevation_angle);
total_attenuation = zeros(elevation_angle);
for  n = 1:numel(elev_range)
     cfg.ElevationAngle = elev_range(n);
     pl =  p618PropagationLosses(cfg, ...
                           'StationHeight',0.031, ...
                           'Temperature',283.6, ...
                           'Pressure',1009.48, ...
                           'WaterVaporDensity',13.79);
    gaseous_attenuation(n) = pl.Ag;
    cloud_attenuation(n) = pl.Ac;
    rain_attenuation(n) = pl.Ar;
    scintillation_attenuation(n) = pl.As;
    total_attenuation(n) = pl.At;
end
```

Plot the Earth-space propagation losses

Plot the various propagation losses (in dB) for the specified range of elevation angles.

```
plot(elev_range,gaseous_attenuation,'--');
hold on;
plot(elev_range,cloud_attenuation,'--');
hold on;
plot(elev_range,rain_attenuation,'--');
hold on;
plot(elev_range,scintillation_attenuation,'--');
hold on;
plot(elev_range,total_attenuation);
legend('Gaseous','Cloud','Rain','Scintillation','Total');
```

```
grid on;
xlabel('Elevation Angle (degrees)');
ylabel('Attenuation (dB)');
title('Earth-Space Propagation Losses Versus Elevation Angle');
```



Propagation Losses with Time Percentage of Exceedance

This example requires MAT-files with digital maps from ITU documents. If the files are not available on the path, execute these commands to download and untar the MAT-files.

```
maps = exist('maps.mat','file');
p836 = exist('p836.mat','file');
p837 = exist('p837.mat','file');
p840 = exist('p840.mat','file');
matFiles = [maps p836 p837 p840];
if ~all(matFiles)
    if ~exist('ITURDigitalMaps.tar.gz','file')
        url = 'https://www.mathworks.com/supportfiles/spc/P618/ITURDigitalMaps.tar.gz';
        websave('ITURDigitalMaps.tar.gz',url);
```

```
        untar('ITURDigitalMaps.tar.gz');
    else
        untar('ITURDigitalMaps.tar.gz');
    end
    addpath(cd);
end
```

This example shows how to parameterize and calculate Earth-space propagation losses when the percentage of time attenuation value is exceeded when designing the Earth-space systems.

Configure Earth-Space Propagation Parameters

Create a P.618 configuration object. Change property values, and then display the object properties.

```
cfg = p618Config;
cfg.Frequency = 19.5e9;          % Signal frequency in Hz
cfg.ElevationAngle = 36.6142654;
cfg.Latitude = 46.2208;          % North direction
cfg.Longitude = 6.137;           % East direction
cfg.AntennaDiameter = 1.2;
cfg.AntennaEfficiency = 0.65;
```

Set the time percentage of excess for the gaseous attenuation, cloud attenuation, rain attenuation, scintillation, and total atmospheric attenuation.

```
annual_exceedance =[5 3 2 1 0.5 0.3 0.2 0.1 0.05 0.03 0.02 0.01 0.005 0.003 0.002 0.001];
```

Calculate Earth-Space Propagation Losses

Calculate the Earth-space propagation losses for the specified configuration parameters.

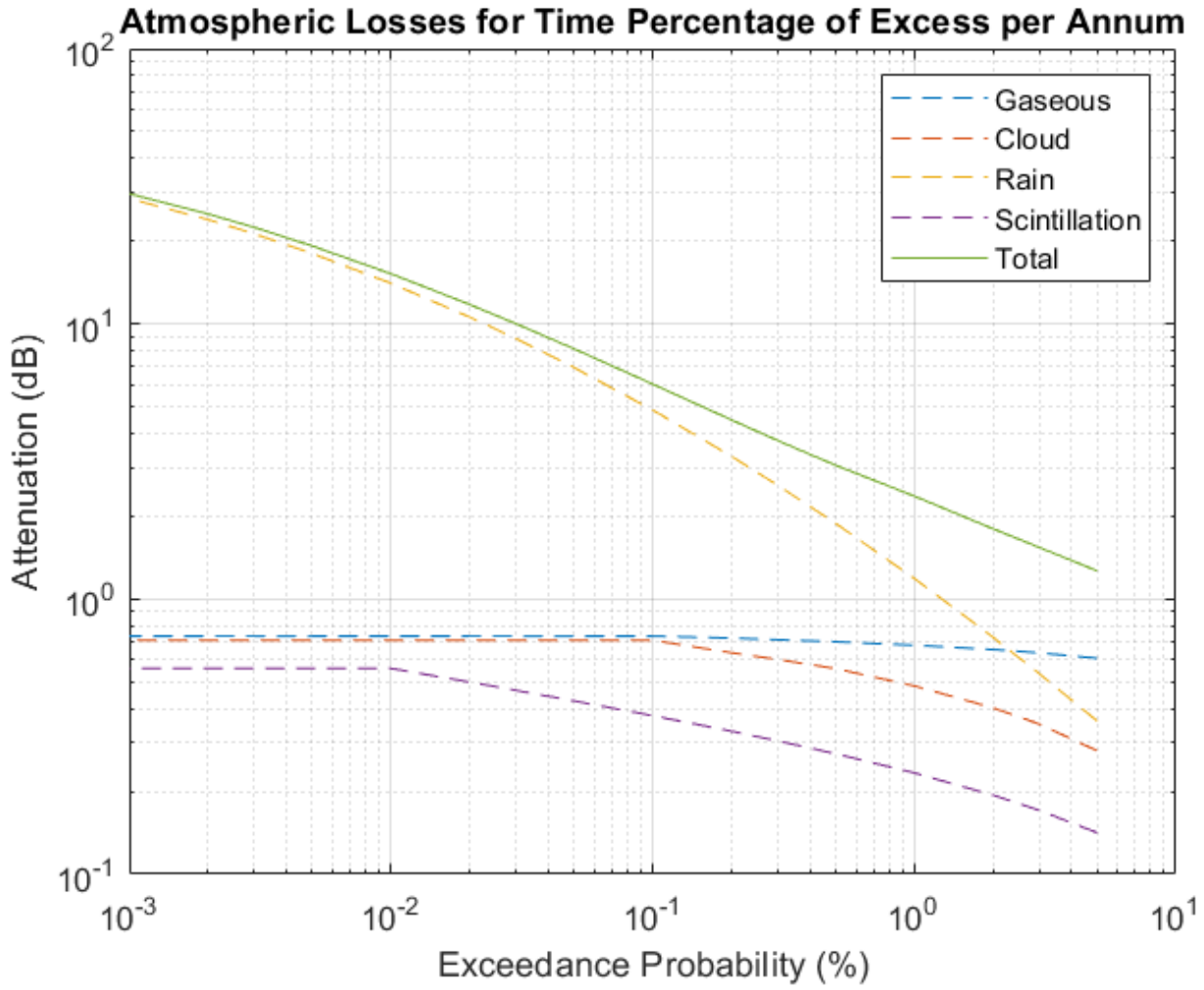
```
excess=size(annual_exceedance);
gaseous_attenuation = zeros(excess);
cloud_attenuation = zeros(excess);
rain_attenuation = zeros(excess);
scintillation=zeros(excess);
total_attenuation = zeros(excess);
for n = 1:numel(annual_exceedance)
    exceedance_value = annual_exceedance(n);
    cfg.GasAnnualExceedance = max(exceedance_value,0.1);            % Supported range is 0.1% to
    cfg.CloudAnnualExceedance = max(exceedance_value,0.1);          % Supported range is 0.1% to
    cfg.RainAnnualExceedance = exceedance_value;                    % Supported range is 0.001% t
    cfg.ScintillationAnnualExceedance = max(exceedance_value,0.01); % Supported range is 0.01% to
    cfg.TotalAnnualExceedance = exceedance_value;                   % Supported range is 0.001% t
    pl =  p618PropagationLosses(cfg,'StationHeight',0.412);
    gaseous_attenuation(n) = pl.Ag;
    cloud_attenuation(n) = pl.Ac;
    rain_attenuation(n) = pl.Ar;
    scintillation(n) = pl.As;
    total_attenuation(n) = pl.At;
end
```

Plot Earth-Space Propagation Losses

Plot the various propagation losses (in dB) when the percentage of time attenuation value is exceeded.

```
loglog(annual_exceedance,gaseous_attenuation,'--');
hold on;
```

```
loglog(annual_exceedance,cloud_attenuation,'--');
hold on;
loglog(annual_exceedance,rain_attenuation,'--');
hold on;
loglog(annual_exceedance,scintillation,'--');
hold on;
loglog(annual_exceedance,total_attenuation);
legend('Gaseous','Cloud','Rain','Scintillation','Total');
grid on;
xlabel('Exceedance Probability (%)');
ylabel('Attenuation (dB)');
title('Atmospheric Losses for Time Percentage of Excess per Annum');
```



References

[1] International Telecommunication Union, ITU-R Recommendation P.618 (12/2017)

[2] International Telecommunication Union, ITU-R Recommendation P.676 (08/2019)

[3] International Telecommunication Union, ITU-R Recommendation P.1511 (08/2019)

[4] International Telecommunication Union, ITU-R Recommendation P.1510 (06/2017)

[5] International Telecommunication Union, ITU-R Recommendation P.835 (12/2017)

[6] International Telecommunication Union, ITU-R Recommendation P.836 (12/2017)

[7] International Telecommunication Union, ITU-R Recommendation P.840 (08/2019)

[8] International Telecommunication Union, ITU-R Recommendation P.837 (06/2017)

[9] International Telecommunication Union, ITU-R Recommendation P.453 (08/2019)

[10] International Telecommunication Union, ITU-R Recommendation P.839 (09/2013)

[11] International Telecommunication Union, ITU-R Recommendation P.838 (03/2005)

[12] Validation examples for Study Group 3 Earth-Space propagation prediction methods, Version: 5.0 (P), ITU Radio communication Study Groups.

See Also

Objects
`p618Config` | `p618SiteDiversityConfig`

Functions
`p618PropagationLosses` | `p618SiteDiversityOutage`