Satellite Communications Toolbox Release Notes

MATLAB®

MathWorks®

How to Contact MathWorks

Latest news:	`www.mathworks.com`
Sales and services:	`www.mathworks.com/sales_and_services`
User community:	`www.mathworks.com/matlabcentral`
Technical support:	`www.mathworks.com/support/contact_us`
Phone:	508-647-7000

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

Satellite Communications Toolbox Release Notes

Contents

R2023a

Version: 1.4

New Features

Satellite scenario enhancements

- Show or hide body axes for all satellite scenario assets

  The `satellite`, `groundStation`, `gimbal`, `conicalSensor`, `transmitter`, and `receiver` object functions now support a new property, `CoordinateAxes`, and a new object function called `coordinateAxes`. You can use the `coordinateAxes` function to create a `CoordinateAxes` object to visualize the body axes of all the satellite scenario assets.
- Calculate Doppler shift and latency in a satellite scenario
  - The `Satellite` object, the `GroundStation` object, and the `Transmitter` object now support the `dopplershift` function. Use the `dopplershift` function to calculate:
    - Doppler shift at a target asset
    - Relative velocity
    - Doppler rate
  - The `Satellite` object and the `GroundStation` object now support the `latency` function. Use the `latency` function to calculate the propagation delay from one asset to another asset.

  For more information about calculating Doppler shift and latency in a satellite scenario, see the “Calculate Latency and Doppler in a Satellite Scenario” example.

Satellite Link Budget Analyzer app enhancements

The **Satellite Link Budget Analyzer** app now supports these features.

- Custom 2-D contour plots for sensitivity analysis. For more information about creating custom 2-D contour plots, see the **Satellite Link Budget Analyzer** app.
- Generation of MATLAB scripts for link budget analysis.

Support for satellite navigation systems

Use the `gnssSignalAcquirer` System object™ to acquire global navigation satellite system (GNSS) signals. Currently, Satellite Communications Toolbox supports these GNSS systems.

- Global Positioning System (GPS)
- Navigation with Indian Constellation (NavIC)
- Quasi-Zenith Satellite System (QZSS)

Support for deep space optical communications Poisson channel

Use the `dsocPoissonChannel` System object to create and configure a deep space optical communications (DSOC) Poisson channel.

Scenario generation and visualization examples

The “Location-Based Analysis of Visible GPS Satellites” shows how to analyze which Global Positioning System (GPS) satellites are visible from a particular location by using `satelliteScenario`.

Link budget analysis examples

- “Sensitivity Analysis Using Exported Script From Satellite Link Budget Analyzer” — This example shows how to perform sensitivity analysis by using a MATLAB script exported from the **Satellite Link Budget Analyzer** app
- “NB-IoT NTN Link Budget Analysis” — This example shows hows how to compute the link budget for narrowband Internet-of-Things (NB-IoT) equipment in non-terrestrial networks (NTN) using the parameter sets defined in 3GPP TR 36.763. To improve the radio coverage in NB-IoT systems, the transmitter repeats the same signal over additional periods. This example calculates and reports the minimum number of additional repetitions required for an NB-IoT transmission.

Signal transmission examples

The “GPS L1C Waveform Generation” shows you how to generate Global Positioning System (GPS) civil navigation data on L1C with binary offset carrier (BOC) modulation.

End-to-End simulation examples

- “End-to-End CCSDS High Photon Efficiency Telemetry Optical Link Simulation” — This example shows how to measure the block error rate in a Consultative Committee for Space Data Systems (CCSDS) high photon efficiency (HPE) telemetry (TM) end-to-end optical link with timing offset impairment.
- “NB-IoT NTN NPDSCH Throughput” — This example shows how to perform a narrowband Internet of Things (NB-IoT) narrowband physical downlink shared channel (NPDSCH) throughput simulation in a non-terrestrial network (NTN) channel.
- “End-to-End DVB-RCS2 Simulation with RF Impairments and Corrections for TC-LM Bursts” — This example shows how to measure the packet error rate (PER) of a Second Generation DVB Interactive Satellite System (DVB-RCS2) link that has a constant waveform ID for Turbo-Coding Linear Modulation (TC-LM) reference bursts.
- “NR NTN PDSCH Throughput” — This example now allows to model a power amplifier with or without memory.

Over-the-Air testing examples

- “GPS Signal Transmission, Acquisition and Tracking using PlutoSDR” — This example shows how to use an ADALM-PLUTO radio for over-the-air transmission and reception of a GPS waveform generated using Satellite Communications Toolbox.
- “Capture Satellite Data Using AWS Ground Station” — This example now uses an Amazon Simple Storage Service (S3) bucket for storing the contact data from Amazon® Web Services (AWS) Ground Station and processes the captured data using a Consultative Committee for Space Data Systems (CCSDS) Telemetry (TM) receiver.

Improved performance for states function

The `states` function of the `Satellite` object shows performance improvement if you use the geographic or Earth-centered, Earth-fixed (ECEF) coordinate system. For example, consider this code that finds the positions of 1000 satellites.

```
% Create a satellite scenario object. The specified start,
% stop, and sample times result in 1441 time samples in the
% scenario.
startTime = datetime(2021,7,8);
stopTime = startTime + days(1);
sampleTime = 60;
scTBK = satelliteScenario(startTime,stopTime,sampleTime);

% Use the Keplerian elements to add 1000 satellites to each
% scenario with the orbit propagator: Two-Body-Keplerian (TBK)
semiMajorAxis = ones(1,1000)*10000000;
eccentricity = ones(1,1000)*0.1;
inclination = ones(1,1000)*60;
rightAscensionOfAscendingNode = repmat(0:18:360 - 18,1,50);
argumentOfPeriapsis = zeros(1,1000);
trueAnomaly = sort(repmat(0:7.2:360 - 7.2,1,20));

satTBK = satellite(scTBK, ...
      semiMajorAxis, ...
      eccentricity, ...
      inclination, ...
      rightAscensionOfAscendingNode, ...
      argumentOfPeriapsis, ...
      trueAnomaly, ...
      OrbitPropagator="two-body-keplerian");

% Simulate the scenario first using the aer method
aer(satTBK(1),satTBK(2));

% Time the geographic simulation of the states method with Two-Body-Keplerian.
tic
positionsGeographic = states(satTBK,CoordinateFrame="geographic");
% Store the execution time.
executionTime = toc;

% Display the execution time.
disp(['Time to compute the states using TBK with geographic = ', ...
    num2str(executionTime),' seconds.'])

% Time the ECEF simulation of the states method with Two-Body-Keplerian.
tic
positionsECEF = states(satTBK,CoordinateFrame="ecef");
% Store the execution time.
executionTime = toc;

% Display the execution time.
disp(['Time to compute the satellite states using TBK with ECEF = ', ...
    num2str(executionTime),' seconds.'])
```

These are approximate code execution times for R2022b and R2023a.

R2022b

- For geographic, the execution time is 85.1046 seconds.
- For ECEF, the execution time is 82.2593 seconds.

R2023a

- For geographic, the execution time is 8.0648 seconds. This is 10.5x faster compared to the previous release.

- For ECEF, the execution time is 2.9357 seconds. This is 28x faster compared to the previous release.

The code was timed on a Windows 10, Intel® Xeon® W-2133 CPU @ 3.60 GHz test system by running this script.

Support for HDL code generation examples

The “GPS HDL Data Decode” example shows how to perform bit synchronization and frame synchronization, and decode the tracked global positioning system (GPS) legacy navigation (LNAV) signal generated using the “GPS HDL Acquisition and Tracking Using C/A Code” example.

These examples are designed using Simulink® blocks and support HDL code generation with HDL Coder™.

R2022b

Version: 1.3

New Features

Satellite scenario enhancements

- New default graphics style and colors in satellite scenario viewer

  A new default graphics style and colors enhance the visibility and contrast of `satelliteScenarioViewer` visualizations.
- Create satellite constellations using RINEX data

  The `satellite` object function now supports receiver independent exchange format (RINEX) data containing GPS or Galileo navigation data, which you can obtain by using the `rinexread` function in Navigation Toolbox™.
- Received power calculation for communication links in satellite scenario

  The `Link` object now supports a new object function, `sigstrength`, that returns the received isotropic power and the power at the receiver input.
- Specify pre-receiver loss in satellite scenario

  The `Receiver` object now supports a new property, `PreReceiverLoss`, that specifies the total loss before the receiver input in the receiver system.

Improved performance for creating satellites

The `satellite` function shows improved performance when creating thousands of satellites. For example, compared to the previous release, this code that creates 5000 satellites is about 10x faster.

```
function timeSatelliteCreation
% Create a satellite scenario object.
sc = satelliteScenario(datetime(2022,03,22,9,0,0), ...
                       datetime(2022,03,22,10,0,0),60);
numSats = 5000;

% Set up orbital elements for satellites
semiMajorAxis = ones(1,numSats)*1e7;
eccentricity = zeros(1,numSats);
inclination = linspace(0,179,numSats);
rightAscensionOfAscendingNode = zeros(1,numSats);
argumentOfPeriapsis = zeros(1,numSats);
trueAnomaly = zeros(1,numSats);

% Create 5000 satellites and time the result
tic
sats = satellite(sc, ...
    semiMajorAxis, ...
    eccentricity, ...
    inclination, ...
    rightAscensionOfAscendingNode, ...
    argumentOfPeriapsis, ...
    trueAnomaly);
executionTime = toc;
end
```

The approximate execution times are:

R2022a: 145 s

R2022b: 14 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by running this script.

This improvement scales up with a higher number of satellites. With fewer satellites, the improvement is less noticeable.

Improved performance for satellite scenario viewer

The `satelliteScenarioViewer` and `play` functions show improved performance with satellite scenarios that contain thousands of satellites. For example, compared to the previous release, this code that visualizes and plays 5000 satellites is about 6x faster.

```
function timeSatelliteVisualizationAndPlay
% Create satellite scenario object.
sc = satelliteScenario(datetime(2022,03,22,9,0,0), ...
                       datetime(2022,03,22,10,0,0),60);
numSats = 5000;

% Set up orbital elements for satellites
semiMajorAxis = ones(1,numSats)*1e7;
eccentricity = zeros(1,numSats);
inclination = linspace(0,179,numSats);
rightAscensionOfAscendingNode = zeros(1,numSats);
argumentOfPeriapsis = zeros(1,numSats);
trueAnomaly = zeros(1,numSats);

% Create 5000 satellites
sats = satellite(sc, ...
    semiMajorAxis, ...
    eccentricity, ...
    inclination, ...
    rightAscensionOfAscendingNode, ...
    argumentOfPeriapsis, ...
    trueAnomaly);

% Visualize and animate the satellites. Time the execution.
tic
viewer = satelliteScenarioViewer(sc);
play(sc);
executionTime = toc;
end
```

The approximate execution times are:

R2022a: 801 s

R2022b: 124 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by running this script.

Satellite Link Budget Analyzer app enhancements

The **Satellite Link Budget Analyzer** app now supports custom 2-D line plots for sensitivity analysis. For more information, see **Satellite Link Budget Analyzer**.

Compute C/N ratio for satellite link budget parameters

Use these features to compute the carrier-to-noise (C/N) ratio for specified satellite link budget parameters.

Function	Description
`satelliteCNRConfig`	C/N configuration parameters
`satelliteCNR`	Compute C/N ratio

Support for CCSDS-based SCPPM codes for optical communications links

Use these features to encode and decode the Consultative Committee for Space Data Systems (CCSDS)-based serially concatenated pulse position modulation (SCPPM) codes. These codes are recommended for optical communications coding and synchronization of signals used in free-space optical communications systems for space missions.

Function	Description
`ccsdsSCPPMEncode`	Encode CCSDS-based SCPPM codes
`ccsdsSCPPMDecode`	Decode CCSDS-based SCPPM codes

For an example of how to perform BER analysis for the CCSDS SCPPM end-to-end chain using a deep space Poisson channel, see End-to-End CCSDS SCPPM Simulation Using Deep Space Poisson Channel.

CCSDS optical high photon efficiency telemetry waveform generation

The CCSDS Optical High Photon Efficiency Telemetry Waveform Generation example shows how to generate a high photon efficiency (HPE) waveform for the free space optical communications systems, as defined in CCSDS 142.0-B-1 section 3.

Support for satellite navigation systems

- Use the `gnssBitSynchronize` function to perform bit synchronization for global navigation satellite systems (GNSS) receivers that use code division multiple access (CDMA) schemes. These GNSS receivers include GPS, NavIC, and QZSS.
- End-to-End GPS Legacy Navigation Receiver Using C/A-Code — This example estimates the GPS receiver position using a multi-satellite GPS baseband waveform.

Support for Lutz channel model

Use the `lutzLMSChannel` object to create and configure a frequency-flat fading Lutz land mobile-satellite (LMS) channel, as defined in the IEEE paper — "The Land Mobile-Satellite Communication Channel-Recording, Statistics, and Channel Model".

For more information on the fading characteristics of this channel, see Simulate and Visualize Land Mobile-Satellite Channel.

End-to-End Simulation example update

The NR NTN PDSCH Throughput example now shows how to perform the receiver-end Doppler compensation in the throughput measurement of a physical downlink shared channel (PDSCH) for a 5G New Radio (NR) link in a non-terrestrial network (NTN) channel.

Support for DVB-S2X bit recovery

Use the `dvbs2xBitRecover` function for bit recovery of Digital Video Broadcasting Satellite Second Generation extended (DVB-S2X) physical layer frames.

Receive and analyze captured satellite data in VITA 49 format

Use the `vita49Reader` object to create a VMEbus International Trade Association (VITA) 49 file reader object to read VITA 49 packets and the associated metadata from an input VITA 49 file.

Support for HDL code generation examples

- DVB-S2 HDL Transmitter — This example shows how to implement a Digital Video Broadcasting Satellite Second Generation (DVB-S2) waveform generator using Simulink blocks optimized for HDL code generation and hardware implementation. The generated transmitter waveform is compatible with the DVB-S2 HDL Receiver reference application.

  This design is ready for deployment to an FPGA or ASIC.
- GPS HDL Acquisition and Tracking Using C/A Code — This example shows how to perform acquisition and tracking on a GPS waveform to extract legacy navigation (LNAV) data using Simulink blocks optimized for HDL code generation and hardware implementation.

These examples are designed using Simulink blocks and support HDL code generation with HDL Coder.

R2022a

Version: 1.2

New Features

Satellite scenario enhancements

- Support for beam steering of phased array objects from Phased Array System Toolbox™ and loop simulation in satellite scenario

  Manually step through a satellite scenario simulation, modify the asset parameters, and manage electronic beam steering of phased array antennas in the middle of a satellite scenario simulation.

  - `satelliteScenario` has three new properties — `SimulationTime`, `SimulationStatus`, and `AutoSimulate`, and two new object functions — `advance` and `restart`.
  - `ConicalSensor`, `Gimbal`, `Transmitter`, and `Receiver` objects now support the `aer` object function.
  - `Transmitter` and `Receiver` objects now support the `pointAt` object function.

  For more information on beam steering with phased arrays, see Interference from Satellite Constellation on Communications Link.
- GPS orbit propagator and SEM almanac import

  You can now use system effectiveness model (SEM) almanacs in a satellite scenario simulation. Use the GPS orbit propagator to propagate an orbit or calculate a satellite position.
- Vectorization of satellite scenario setup and analysis

  You can now vectorize the satellite scenario setup workflow and the analysis workflow to gain performance. Vectorization enables you to add multiple conical sensors, gimbals, transmitters, receivers, accesses, and links in a single line of code.
- Non-cluttered satellite scenario constellation visualization

  Create a design using the `ShowDetails` property of `satelliteScenarioViewer` to visualize the constellation in a easy-to-read, non-cluttered manner.
- Specify custom orientations for satellite and gimbal objects

  The satellite and gimbal object function `pointAt` now supports custom orientations. For an example, see Modeling Custom Satellite Attitude and Gimbal Steering.
- Interference from a satellite constellation on a communications link

  The Interference from Satellite Constellation on Communications Link example now supports the use of a parabolic reflector and uniform rectangular array with Antenna Toolbox™ and Phased Array System Toolbox, respectively.
- Multi-hop path selection through large satellite constellation

  The Multi-Hop Path Selection Through Large Satellite Constellation example determines the path through a large constellation consisting of 1000 low-Earth orbit satellites to gain access between two ground stations. It also demonstrates how to calculate the intervals during the next three hour period when this path can be used.
- MATLAB® Compiler™ now supports all satellite scenario generation capabilities.

Improved performance for link analysis when using antennas from Antenna Toolbox

The `Link` object functions (`linkPercentage`, `linkIntervals`, `linkStatus`, and `ebno`) show improved performance when you assign transmitters and receiver antennas from Antenna Toolbox. For example, compared to the previous release, the following code is about 319x faster.

```
function timeLink
% Create a satellite scenario object.
startTime = datetime(2020,10,11);
stopTime = startTime + days(1);
sampleTime = 60;
sc = satelliteScenario(startTime,stopTime,sampleTime);

% Add a satellite to the scenario.
sat = satellite(sc,'eccentricOrbitSatellite.tle');

% Add a ground station to the scenario.
gs = groundStation(sc);

% Add a gimbal to the ground station.
gim = gimbal(gs);

% Point the satellite at the ground station
% and the ground station gimbal at the satellite.
pointAt(sat,gs);
pointAt(gim,sat);

% Create a parabolic reflector antenna, assign it
% to a transmitter, and attach it to the gimbal.
frequency = 14e9; % Hz
anGim = design(reflectorParabolic,frequency);
tx = transmitter(gim,'Frequency',frequency, ...
    'Power',20, ... dBW
    'Antenna',anGim);

% Create a slot antenna, assign it to a receiver,
% and attach it to the satellite.
anSat = design(slot,frequency);
rx = receiver(sat, ...
    'RequiredEbNo', 4, ...
    "Antenna",anSat);

% Create a link analysis between the transmitter and the receiver.
lnk = link(tx,rx);

% Determine the intervals using linkIntervals and time the execution.
tic;
intvls = linkIntervals(lnk);
executionTime = toc;
disp("Execution of linkIntervals = " + executionTime + " seconds.");
end
```

The approximate execution times are:

R2021b: 231.99 s

R2022a: 0.73 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by running this script.

Support for BOC modulation

Use the `bocmod` function to modulate the data bits with a square wave by using binary offset carrier (BOC) modulation. This function supports modernized GPS waveform on an L1C carrier.

Enhanced support for DVB-S2 and DVB-S2X waveform generation

`dvbs2WaveformGenerator` and `dvbs2xWaveformGenerator` objects now support waveform generation with dummy frames.

Support for ITU-R P.681-11 LMS channel

Use the `p681LMSChannel` object to create and configure an ITU-R P.681-11 frequency-flat fading LMS channel, as defined in ITU-R Recommendation P.681-11 Section 6.2.

To observe the fading characteristics of this channel, see Simulate and Visualize a Land Mobile-Satellite Channel.

Model 5G NR NTN channel

The Model NR NTN Channel example shows how to model these two types of New Radio (NR) non-terrestrial network (NTN) channels.

- Flat fading narrowband channel using the `p681LMSChannel` object from Satellite Communications Toolbox
- Frequency selective fading tapped delay line (TDL) channel using `nrTDLChannel` object from 5G Toolbox™

Antenna size analysis using ITU-R P.618 propagation model

The Antenna Size Analysis Using ITU-R P.618 Propagation Model example shows how to select a parabolic antenna diameter for a particular ground location using the `p618PropagationLosses` function.

End-to-End Simulation examples

- GPS Receiver Acquisition and Tracking Using C/A-Code — This example shows how to generate a legacy GPS intermediate frequency (IF) waveform from multiple satellites, add noise to the composite signal, perform initial synchronization, and track the code phase and carrier frequency of the available satellites detected from the initial synchronization operation.
- GPS Data Decode — This example shows how to perform bit and frame synchronization and decode GPS legacy navigation (LNAV) data.
- NR NTN PDSCH Throughput — This example shows how to measure the physical downlink shared channel (PDSCH) throughput of a 5G NR link in an NTN channel, as defined by the 3GPP NR standard.

- Support for Digital Video Broadcasting Satellite Second Generation extended (DVB-S2X) super-frame (SF) generators
  - DVB-S2X Super-Frame Generation for Formats 0 and 1 — This example shows how to generate an SF complex baseband waveform for formats 0 and 1.
  - DVB-S2X Super-Frame Generation for Formats 2 and 3 — This example shows how to generate an SF complex baseband waveform for formats 2 and 3.
- End-to-End DVB-S2X Simulation with RF Impairments and Corrections in Wideband Mode — This example shows how to measure the bit error rate (BER) and packet error rate (PER) of a single stream DVB-S2X link that has constant coding and modulation for wideband mode using time slicing.

Receive and analyze captured satellite data

Read the signal time data packets and the associated metadata (context packet data) from a VITA 49.2 format file into the MATLAB workspace. For details, see VITA 49 File Reader.

Support for HDL code generation examples

- DVB-S2 HDL PL Header Recovery — This example shows how to implement Digital Video Broadcasting Satellite Second Generation (DVB-S2) time, frequency, and phase synchronization, and recover physical layer (PL) header using Simulink blocks optimized for HDL code generation and hardware implementation.
- DVB-S2 HDL Receiver — This example builds on the *DVB-S2 HDL PL Header Recovery* example and shows how to implement an end-to-end receiver for real-time applications that conform to the DVB-S2 standard.

These examples are designed using Simulink blocks and support HDL code generation with HDL Coder.

R2021b

Version: 1.1

New Features

Satellite scenario enhancements

- Satellite radiation pattern visualization

  The `pattern` function enables visualization of satellite and ground station antenna radiation patterns in the `satelliteScenarioViewer`.
- Calculate latency and Doppler in a satellite scenario

  The Calculate Latency and Doppler in a Satellite Scenario example calculates the latency, Doppler frequency, and latency and Doppler rates of changes between an orbiting satellite and a ground station.
- Interference from a satellite constellation on a communications link

  The Interference from Satellite Constellation on Communications Link example analyzes downlink interference by calculating signal-to-interference-plus-noise ratio (SINR) for a geosynchronous satellite to a ground station located in the Pacific Ocean, with a constellation of multiple low Earth orbit (LEO) satellites causing interference.

Improved performance for computing satellite orbits

The `satelliteScenario` object shows improved simulation performance when you use Two-Body-Keplerian, SGP4, or SDP4 orbit propagators. The performance improvement is observable when the scenario involves large satellite constellations involving 40 or more satellites and/or when the scenario contains more than 1000 time samples between start and stop times.

For example, compared to the previous release, the following code demonstrates that the computation of the position history of 1000 satellites using Two-Body-Keplerian, SGP4, and SDP4 orbit propagators is about 51x, 64x, and 50x faster, respectively.

```
% Create 3 satellite scenario objects. The specified start,
% stop, and sample times result in 1441 time samples in the
% scenario. Each scenario tests a specific orbit propagator.
startTime = datetime(2021,7,8);
stopTime = startTime + days(1);
sampleTime = 60;
scTBK = satelliteScenario(startTime,stopTime,sampleTime);
scSGP4 = satelliteScenario(startTime,stopTime,sampleTime);
scSDP4 = satelliteScenario(startTime,stopTime,sampleTime);

% Use the Keplerian elements to add 1000 satellites to each
% scenario with the orbit propagator in order:
% Two-Body-Keplerian, SGP4, and SDP4.
semiMajorAxis = ones(1,1000)*10000000;
eccentricity = ones(1,1000)*0.1;
inclination = ones(1,1000)*60;
rightAscensionOfAscendingNode = repmat(0:18:360-18,1,50);
argumentOfPeriapsis = zeros(1,1000);
trueAnomaly = sort(repmat(0:7.2:360-7.2,1,20));
satTBK = satellite(scTBK, ...
    semiMajorAxis, ...
    eccentricity, ...
    inclination, ...
    rightAscensionOfAscendingNode, ...
    argumentOfPeriapsis, ...
    trueAnomaly, ...
    "OrbitPropagator","two-body-keplerian");
satSGP4 = satellite(scSGP4, ...
    semiMajorAxis, ...
    eccentricity, ...
    inclination, ...
    rightAscensionOfAscendingNode, ...
    argumentOfPeriapsis, ...
    trueAnomaly, ...
    "OrbitPropagator","sgp4");
satSDP4 = satellite(scSDP4, ...
    semiMajorAxis, ...
    eccentricity, ...
    inclination, ...
    rightAscensionOfAscendingNode, ...
    argumentOfPeriapsis, ...
    trueAnomaly, ...
    "OrbitPropagator","sdp4");

% Initialize an array to store the satellite position in the
% GCRF frame.
numTimeSamples = ceil(seconds(stopTime - startTime) ...
                      /sampleTime) + 1;
positionsTBK = zeros(3,numTimeSamples,1000);
positionsSGP4 = zeros(3,numTimeSamples,1000);
positionsSDP4 = zeros(3,numTimeSamples,1000);

% Time the simulation with Two-Body-Keplerian.
tic

% Fill out the position array using the states function.
% The first call to states simulates the scenario.
for idx = 1:1000
    positionsTBK(:,:,idx) = states(satTBK(idx));
end

% Store the execution time.
executionTime = toc;

% Display the execution time.
disp("Time to compute the satellite states using ...
      Two-Body-Keplerian = " + executionTime + " seconds.");

% Time the simulation with SGP4.
tic

% Fill out the position array using the states function.
% The first call to states simulates the scenario.
for idx = 1:1000
    positionsSGP4(:,:,idx) = states(satSGP4(idx));
end

% Store the execution time.
executionTime = toc;

% Display the execution time.
disp("Time to compute the satellite states using SGP4 = " ...
                             + executionTime + " seconds.");
```

```
% Time the simulation with SDP4.
tic

% Fill out the position array using the states function.
% The first call to states simulates the scenario.
for idx = 1:1000
    positionsSDP4(:,:,idx) = states(satSDP4(idx));
end

% Store the execution time.
executionTime = toc;

% Display the execution time.
disp("Time to compute the satellite states using SDP4 = " ...
                                + executionTime + " seconds.");
```

The approximate execution times are:

R2021a: Two-Body-Keplerian – 393.49 s, SGP4 – 515.55 s, and SDP4 – 734.12 s

R2021b: Two-Body-Keplerian – 7.63 s, SGP4 – 7.95 s, and SDP4 – 14.53 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by running this script.

You can observe the performance improvement in all functions related to satellite scenario objects.

Improved performance for access analysis

The `Access` object functions (`accessPercentage`, `accessIntervals`, and `accessStatus`) show improved performance, which is observable when the number of access analysis objects in the scenario is 40 or more, and/or when the scenario contains more than 1000 time samples between start and stop times.

For example, compared to the previous release, the following code is about 29x faster.

```
% Create a satellite scenario object.
startTime = datetime(2021,7,8);
stopTime = startTime + days(1);
sampleTime = 60;
sc = satelliteScenario(startTime,stopTime,sampleTime);

% Add 40 satellites.
sat = satellite(sc,'leoSatelliteConstellation.tle');

% Point the satellites at a specified geographical coordinate.
for idx = 1:numel(sat)
    pointAt(sat(idx),[0;0;0]);
end

% Add gimbals to the satellites, and then add a conical sensor
% to the gimbal.
for idx = 1:numel(sat)
    gim = gimbal(sat(idx));
    conicalSensor(gim);
end
```

```
% Add a ground station.
gs = groundStation(sc);

% Point the gimbals at the ground station.
gim = [sat.Gimbals];
for idx = 1:numel(gim)
    pointAt(gim(idx),gs);
end

% Add access analysis between each conical sensor and
% the ground station.
sensor = [gim.ConicalSensors];
for idx = 1:numel(sensor)
    access(sensor(idx),gs);
end

% Retrieve the access objects.
ac = [sensor.Accesses];

% Time the computation of the access intervals for all the
% access objects.
tic;
intvls = accessIntervals(ac);
executionTime = toc;
disp("Execution of accessIntervals = " + executionTime + ...
     " seconds.");
```

The approximate execution times are:

R2021a: 48.18 s

R2021b: 1.64 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by running this script.

Improved performance for satellite scenario viewer

The graphic updates on the `satelliteScenarioViewer` are observed to be faster when the satellite scenario contains 50 or more satellites.

For example, compared to the previous release, the following code for updating the visualization of 1000 satellites on the satellite scenario viewer is about 5x faster.

```
% Create a satellite scenario object.
startTime = datetime(2021,7,8);
stopTime = startTime + days(1);
sampleTime = 60;
sc = satelliteScenario(startTime,stopTime,sampleTime);

% Add 1000 satellites to the scenario using Keplerian elements.
semiMajorAxis = ones(1,1000)*10000000;
eccentricity = ones(1,1000)*0.1;
inclination = ones(1,1000)*60;
rightAscensionOfAscendingNode = repmat(0:18:360-18,1,50);
argumentOfPeriapsis = zeros(1,1000);
```

```
trueAnomaly = sort(repmat(0:7.2:360-7.2,1,20));
sat = satellite(sc, ...
    semiMajorAxis, ...
    eccentricity, ...
    inclination, ...
    rightAscensionOfAscendingNode, ...
    argumentOfPeriapsis, ...
    trueAnomaly);

% Before launching the satellite scenario viewer, hide the
% satellite labels.
for idx = 1:numel(sat)
    sat(idx).ShowLabel = false;
end

% Launch the satellite scenario viewer and hide the satellites.
v = satelliteScenarioViewer(sc);
hide(sat);

% Show the satellites and time, again.
tic;
show(sat);
updateTime = toc;
disp("Time to update the graphics = " + updateTime + ...
     " seconds.");
```

The approximate execution times are:

R2021a: 77.76 s

R2021b: 14.15 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by running this script.

Improved performance for field of view visualization

The visualization using the `fieldOfView` function of `ConicalSensor` object is observed to be faster. The performance improvement is observable even if only one field-of-view contour is displayed and becomes pronounced as you add more contours and/or the number of time samples between the satellite scenario start and stop times is greater than 1000.

For example, compared to the previous release, the following code for visualizing 40 field-of-view contours with 86,401 sample times is about 2.7x faster.

```
% Create a satellite scenario object.
startTime = datetime(2021,7,8);
stopTime = startTime + days(1);
sampleTime = 60;
sc = satelliteScenario(startTime,stopTime,sampleTime);

% Add 40 satellites.
sat = satellite(sc,"leoSatelliteConstellation.tle");

% Add conical sensors to each satellite.
for idx = 1:numel(sat)
    conicalSensor(sat(idx));
```

```
end

% Point the satellites at a specified geographical location.
for idx = 1:numel(sat)
    pointAt(sat(idx),[0;0;0]);
end

% Add field-of-view visualization to the conical sensors.
sensors = [sat.ConicalSensors];
fieldOfView(sensors);

% Launch the satellite scenario viewer.
v = satelliteScenarioViewer(sc);

% Play the scenario to visualize the evolution of field-of-view (FOV)
% contours, and time it.
tic;
play(sc);
executionTime = toc;
disp(['Time to calculate and visualize FOV contours = ', ...
    num2str(executionTime), ' seconds.']);
```

The approximate execution times are:

R2021a: 102.52 s

R2021b: 37.56 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by running this script.

Improved performance for link analysis when using Gaussian antennas

The `Link` object functions (`linkPercentage`, `linkStatus`, and `linkIntervals`) show improved performance when using Gaussian antennas. This improvement is observable when the number of link analysis objects in the scenario is 40 or more, and/or when the scenario contains more than 1000 time samples between start and stop times.

For example, compared to the previous release, the following code is about 34x faster.

```
% Create a satellite scenario object.
startTime = datetime(2021,7,8);
stopTime = startTime + days(1);
sampleTime = 60;
sc = satelliteScenario(startTime,stopTime,sampleTime);

% Add 40 satellites.
sat = satellite(sc,'leoSatelliteConstellation.tle');

% Point the satellites at a specified geographical coordinate.
for idx = 1:numel(sat)
    pointAt(sat(idx),[0;0;0]);
end

% Add gimbals to the satellites, and then add a transmitter
% to the gimbal.
for idx = 1:numel(sat)
    gim = gimbal(sat(idx));
    transmitter(gim);
end

% Add a ground station, and then add a receiver to it.
gs = groundStation(sc);
rx = receiver(gs,"MountingAngles",[0;180;0]);

% Point the gimbals at the ground station.
gim = [sat.Gimbals];
for idx = 1:numel(gim)
    pointAt(gim(idx), gs);
end

% Add link analysis between each satellite transmitter and the
% ground station receiver.
tx = [gim.Transmitters];
for idx = 1:numel(tx)
    link(tx(idx),rx);
end

% Retrieve the link objects.
lnk = [tx.Links];

% Time the computation of the link intervals for all the
% link objects.
tic;
intvls = linkIntervals(lnk);
executionTime = toc;
disp("Execution of linkIntervals = " + executionTime + " seconds.");
```

The approximate execution times are:

R2021a: 49.75 s

R2021b: 1.43 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by running this script.

Satellite Link Budget Analyzer app enhancements

The **Satellite Link Budget Analyzer** app now supports link availability analysis with P.618 propagation data and prediction methods. For more information, see Satellite Link Budget Analyzer.

Support for DVB-RCS2

Use the features listed in this table to encode and decode Digital Video Broadcasting Second Generation Return Channel over Satellite (DVB-RCS2) duo-binary turbo codes, generate a DVB-RCS2 standard-based waveform, and recover the signal.

Function	Description
`dvbrcs2TurboEncode`	Encode DVB-RCS2-compliant turbo codes
`dvbrcs2TurboDecode`	Decode DVB-RCS2-compliant turbo codes
`dvbrcs2WaveformGenerator`	Generate DVB-RCS2 waveform
`dvbrcs2RecoveryConfig`	DVB-RCS2 receiver configuration parameters
`dvbrcs2BitRecover`	Recover bits for DVB-RCS2 waveform

Support for satellite navigation systems

Use the features listed in this table to generate coarse acquisition codes (C/A-codes) and precision codes (P-codes).

Function	Description
`gnssCACode`	Generate C/A-code for GPS, NavIC, and QZSS satellites
`gpsPCode`	Generate P-code for GPS satellites

For more information on how to configure and generate a GPS waveform using C/A and P-codes, see GPS Waveform Generation.

End-to-End Simulation examples

- End-to-End DVB-S2X Simulation with RF Impairments and Corrections for VL-SNR Frames — This example shows how to measure the bit error rate (BER) and packet error rate (PER) of a single stream Digital Video Broadcasting Satellite Second Generation extended (DVB-S2X) link for very low signal-to-noise ratio (VL-SNR) frames.
- DVB-S2 Bent Pipe Simulation with RF Impairments and Corrections — This Simulink example models a DVB-S2 bent-pipe satellite link and calculates the end-to-end PER and a low density parity check (LDPC) coding BER.
- End-to-End CCSDS Flexible Advanced Coding and Modulation Simulation with RF Impairments and Corrections — This example shows how to measure the BER of the end-to-end chain of the Consultative Committee for Space Data Systems (CCSDS) flexible advanced coding and modulation (FACM) scheme for a high-rate telemetry (TM) applications system.

Receive and analyze captured satellite data

Use the Amazon Web Services (AWS) Ground Station service with MATLAB to receive data from an Earth observation satellite. Set up access to AWS, configure and set up data capture, schedule contact with the satellite, and download the received satellite data. For details, see Capture Satellite Data Using AWS Ground Station.

MATLAB Compiler Support

MATLAB Compiler now supports Satellite Communications Toolbox software. For information about product limitations, see MATLAB Compiler and Simulink Compiler.

MATLAB Online Support

MATLAB Online™ now supports Satellite Communications Toolbox software. For information about product limitations, see Specifications and Limitations.

R2021a

Version: 1.0

New Features

Introducing Satellite Communications Toolbox

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.

Satellite Scenario: Simulate, analyze, and visualize satellites in orbit

Use functions and reference examples that enable you to simulate, analyze, and visualize satellites in orbits. Use the `satelliteScenario` object to:

- Define satellites and their orbits.
- Define ground stations.
- Visualize satellites in orbit and ground tracks.
- Visualize satellite field-of-view on Earth.
- Analyze line-of-sight access between satellites and ground stations.
- Analyze the closure of communications links between satellites and ground stations.
- View a satellite scenario with playback animation.

For more information, see Satellite Scenario Basics and Scenario Generation and Visualization.

Satellite link budget analysis

Use the **Satellite Link Budget Analyzer** app to analyze, design, and visualize the link budget for satellite communications. For more information, see Link Budget Analysis.

Standards-based waveform generation

Use standards-based functions and reference examples to design, model, and verify satellite communications and navigation systems. The toolbox offers these standards-based waveforms:

- Consultative Committee for Space Data Systems (CCSDS) Telecommand (TC)
- CCSDS Telemetry (TM)
- Digital Video Broadcasting Satellite Second Generation (DVB-S2)
- Digital Video Broadcasting Satellite Second Generation extended (DVB-S2X)
- Global Positioning System (GPS)

For more information, see Generate Waveforms and Signal Transmission.

Channel models and RF propagation loss

Use ITU-R P.618 functions to design Earth-space links by calculating the signal propagation loss through the atmosphere. With these toolbox features, you can also model and visualize noisy single input single output (SISO) channels that have Rician or Land Mobile Satellite (LMS) fading profiles. For more information, see RF Propagation and Channel Models.

Signal recovery

Use receiver functions to decode and demodulate waveforms. Use reference examples to model end-to-end communications links and analyze link performance. For more information, see End-to-End Simulation.

C and C++ code generation support

Satellite Communications Toolbox supports ANSI®/ISO® compliant C/C++ code generation. For an alphabetized list of features that support C/C++ code generation, see Satellite Communications Toolbox – Functions and Objects Filtered by C/C++ Code Generation.