

Satellite Communications Toolbox Release Notes



MATLAB[®]



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

© COPYRIGHT 2021–2022 by The MathWorks, Inc.

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

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

Trademarks

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

Patents

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

R2022a

Satellite scenario enhancements	1-2
Improved performance for link analysis when using antennas from Antenna Toolbox	1-3
Support for BOC modulation	1-4
Enhanced support for DVB-S2 and DVB-S2X waveform generation	1-4
Support for ITU-R P.681-11 LMS channel	1-4
Model 5G NR NTN channel	1-4
Antenna size analysis using ITU-R P.618 propagation model	1-4
End-to-End Simulation examples	1-4
Receive and analyze captured satellite data	1-5
Support for HDL code generation examples	1-5

R2021b

Satellite scenario enhancements	2-2
Improved performance for computing satellite orbits	2-2
Improved performance for access analysis	2-4
Improved performance for satellite scenario viewer	2-5
Improved performance for field of view visualization	2-6
Improved performance for link analysis when using Gaussian antennas	2-7
Satellite Link Budget Analyzer app enhancements	2-8
Support for DVB-RCS2	2-8

Support for satellite navigation systems	2-9
End-to-End Simulation examples	2-9
Receive and analyze captured satellite data	2-9
MATLAB Compiler Support	2-9
MATLAB Online Support	2-10

R2021a

Introducing Satellite Communications Toolbox	3-2
Satellite Scenario: Simulate, analyze, and visualize satellites in orbit ...	3-2
Satellite link budget analysis	3-2
Standards-based waveform generation	3-2
Channel models and RF propagation loss	3-3
Signal recovery	3-3
C and C++ code generation support	3-3

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)*100000000;
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
% contours, and time it.
tic;
play(sc);
executionTime = toc;
disp("Time taken to calculate and visualize the ...
    field-of-view contours = " + 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 P618 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](#).

