Radar Toolbox Release Notes

MATLAB® & SIMULINK®

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

Radar Toolbox Release Notes

Contents

R2021b

R2021a

R2022b

Range-Doppler radar cross section

The new clutterSurfaceRangeDopplerRCS function generates radar cross sections (RCS) of flat surfaces by decomposing the RCS into range and Doppler cells. Applicable to monostatic radar, the function sums over many range-Doppler resolution cells to compute radar cross sections of extended regions.

Scenario surface and clutter visualization

This release introduces two new plotting capabilities to Radar Toolbox for use with a theaterPlot object. Both capabilities are implemented using a plotter configuration object, a plotting function, and a data structure. You can plot land and sea surfaces and regions that generate clutter.

Scenario surface plotter

- Use the surfacePlotter object method to create a SurfacePlotter plotter object for theaterPlot.
- Then use the plotSurface function to create a surface plot from surface data and the SurfacePlotter object.
- Use the surfacePlotterData function to create a data structure to use as input to the plotSurface function. The structure lets you plot surfaces managed by a SurfaceManager object.

Scenario clutter plotter

- Use the clutterRegionPlotter object method to create a ClutterRegionPlotter plotter object for theaterPlot.
- Then use the plotClutterRegion function to create a clutter plot from clutter data and the ClutterRegionPlotter object.
- The clutterRegionData function creates a data structure that you can use as input to plotClutterRegion function.

Atmospheric refracted signal paths

The new llarangeangle function enables you to compute the propagation range of a signal between two points. Specify the points using latitude, longitude, and altitude. Additionally, you can use the function to determine the angle of departure of the signal from the first point and the angle of arrival at the second point. Because of atmospheric refraction, propagation paths follow a curved trajectory. The refraction model uses the effective Earth radius approximation. You can specify the effective Earth radius or use its default value of 4/3.

The slant2range function returns the refracted propagation range between a target and a sensor. You can specify two different models of path refraction. Choose 'Curved' to use an effective earth radius model for refraction or 'CRPL' to use an exponential reference atmosphere for refraction.

The new atmosphere object function models atmospheric refraction effects from within the radarScenario object. The effearthradius function computes the effective Earth radius from an atmosphere model.

New plot capabilities for Radar Designer app

Using the **Radar Designer** app, you can now generate Probability of detection (Pd) versus signal-to-noise ratio (SNR) plots. You can change radar parameters on the **Radar**, **Target**, or **Environment** panels and view the changes on the plot. For example, you can change the Probability of false alarm (Pfa) value on the **Radar** panel or change the `Swerling` target model on the **Target** panel.

In addition, you can now export Pd versus range plots, environmental loss plots, and link budget plots by using the **Export Detectability Analysis** MATLAB® script.

Radar budget plot

You can visualize a radar link budget as a waterfall chart using the `radarbudgetplot` function. The waterfall chart shows the contributions of individual losses and gains present in a radar system to the total energy required by the radar to produce a detection. The total gains and losses is called the radar detectability factor.

Radar resource management: Workflow enhancements for radarDataGenerator

This release introduces three changes to `radarDataGenerator` System object™ to facilitate radar resource management (RRM) workflows.

There is now a custom scan mode for the object. Set the `ScanMode` property to `'Custom'` and then you can set the `LookAngle` property during a simulation. When you set the scan mode to `'Custom'`, there is no distinction between electronic and mechanical scanning. In this mode, you can use only monostatic workflows with no radar emissions are allowed. This scan mode disables the following properties:

`MaxAzimuthScanRate`	`MaxElevationScanRate`	`MechanicalAzimuthLimits`
`MechanicalElevationLimits`	`ElectronicAzimuthLimits`	`ElectronicElevationLimits`
`MechanicalAngle`	`ElectronicAngle`	`FieldOfView`
`EmissionsInputPort`		

Several properties of the `radarDataGenerator` System object are now tunable. These property modifications let you simulate RRM workflows that manage and dynamically change detection, waveform, and antenna array parameters. You can tune the properties related to these parameters in all scan modes.

`CenterFrequency`	`Bandwidth`	`DetectionProbability`
`ReferenceRange`	`ReferenceRCS`	`FalseAlarmRate`
`RangeLimits`	`RangeRateLimits`	`MaxUnambiguousRange`
`MaxUnambiguousRadialSpeed`	`AzimuthResolution`	`ElevationResolution`
`ElevationResolution`	`RangeRateResolution`	

This release also adds a new `BeamShape` property to the `radarDataGenerator` when you use the custom scan mode. You can specify `BeamShape` as `'Rectangular'` or `'Gaussian'`. Previously, when a radar beam scanned a target, the detection probability from the target remained the same regardless of where the target was located within the beam. Now, when setting `BeamShape` to `'Gaussian'`, the detection probability is greatest when the target is centered on the beam and falls off as the beam moves away from the target. Using `BeamShape` enables RRM workflows that optimize beam placement, beam spacing, revisit times, and dwell times to improve the detection probability.

Use motion model name to obtain track position, velocity, and covariance

By using the `getTrackPositions` and `getTrackVelocities` functions, you can now obtain positions, velocities, and associated covariances of tracks by specifying the motion model name as an input. For example,

```
[positions,covariances] = getTrackPositions(tracks,"constvel")
```

returns position and position covariances in tracks based on the constant velocity model defined in the `constvel` function.

Previously, you could only use the position selector or velocity selector input to obtain the position and velocity states. For example,

```
positionSelector = [1 0 0 0 0 0 0 0 0;
                    0 0 0 1 0 0 0 0 0;
                    0 0 0 0 0 0 1 0 0];
[positions,covariances] = getTrackPositions(tracks,positionSelector)
```

Confirm tracks in radarTracker

You can now confirm a track using the `confirmTrack` object function of the `radarTracker` System object. The function confirms tracks for any specified `TrackID`.

Specify wait and reverse motion for waypoint trajectory

You can now specify wait and reverse motion using the `waypointTrajectory` System object.

- To let the trajectory wait at a specific waypoint, simply repeat the waypoint coordinate in two consecutive rows when specifying the `Waypoints` property.
- To render reverse motion, separate positive (forward) and negative (backward) groundspeed values by a zero value in the `GroundSpeed` property.

Applications: Radar models, clutter, and atmospheric effects

The release introduces several new application examples.

- “Simulate a Maritime Radar PPI” creates a plan position indicator (PPI) radar image for a rotating antenna array in a maritime environment. You can configure a maritime radar scenario which includes a spectral sea surface model and an extended target.
- The “Maritime Clutter Removal with Neural Networks” example shows how to train and evaluate a convolutional neural network to remove clutter returns from maritime radar PPI images using the Deep Learning Toolbox™.

- “Multibeam Radar for Adaptive Search and Track” shows how to use the `radarDataGenerator` in a closed-loop simulation of a multifunction phased array radar (MPAR) that tracks multiple maneuvering targets. The radar performs volume search, cued search, and tracking.
- “FMCW Radar Altimeter Simulation” illustrates how to model a radar altimeter and how to evaluate its performance.
- “Predict Surface Clutter Power in Range-Doppler Space” calculates the radar cross section (RCS) of surface clutter seen by a pulse-Doppler radar system.
- “Simulating Radar Signals with Atmospheric Refraction Effects” discusses environmental factors that can result in signal loss and errors in target parameter estimation.
- The “Simulate FMCW Interference Between Automotive Radars” example shows how to simulate interference between two FMCW automotive radars in a highway scenario.

R2022a

Land and sea surface modeling

This release gives you the capability to add land and sea surfaces to a `radarScenario`. Use the `landSurface` object function to create a `LandSurface` object and the `seaSurface` object function to create a `SeaSurface` object.

- The `LandSurface` object defines the terrain, boundary, reflectivity, and reference height of land surfaces. The object enables you to create surfaces from imported DTED files. The `radarScenario` object enables terrain occlusions when you use the `detect` object function of a `radarScenario` object, or the `detect` object function of a `Platform` object in the scenario. If the `IsEarthCentered` property of the `radarScenario` is set to `true`, the scenario models the Earth surface using the WGS84 model.
- The `SeaSurface` object defines the boundary, sea surface motion, reflectivity, and reference height of sea surfaces. You can use the `seaSpectrum` object to generate surface heights and control the evolution of the surface over time. The object implements the Elfouhaily model for creating an omnidirectional and wind-dependent spectrum. The object also allows the use to specify a custom sea spectrum. The `radarScenario` object enables sea surface occlusions when you use the `detect` object function of the scenario, or the `detect` object function of a `Platform` object in the scenario. The spectral model is only available for non-Earth-centered scenarios.
- In this release, you can simulate clutter returns by using the `clutterGenerator` object function of the `radarScenario` object. With `clutterGenerator`, you can control how clutter returns are generated by the `radarDataGenerator` and `radarTransceiver` System objects. By default, clutter returns are limited to the radar beam illumination area. Use the new `ringClutterRegion` object function to define custom regions for the clutter generator.
- You can add multiple `LandSurface` objects and `SeaSurface` objects to a scenario. You generally cannot not overlap `LandSurface` or `SeaSurface` objects. However, two surfaces can overlap as long as at least one of them does not have height data. This means that for a `LandSurface`, the `Terrain` property is empty and for a `SeaSurface`, the `SpectralModel` property is empty. Then you can overlap a land surface with terrain or sea surface with a spectral model with an unbounded flat background surface and place a land or a sea surface over it in some region. This will generate clutter returns from background if the radar points outside the bounds of the surface.
- You can manage land and sea surfaces using the `SurfaceManager`.
  - You can determine if the line-of-sight between two positions in the scenario are occluded by surfaces using the `occlusion` object function of the `SurfaceManager` object. To disable surface occlusion, specify the `UseOcclusion` property of the `SurfaceManager` object as `false`.
  - You can obtain the height of land surfaces and sea surfaces in the scenario by using the `height` object function of the `SurfaceManager` object.

Land and sea reflectivity

This release introduces four surface reflectivity objects and also updates the sea and land reflectivity functions:

- The Radar Toolbox now has four new System objects designed to compute normalized reflectivity when used within the `radarScenario` framework. These objects allow users to model reflectivity for constant gamma, land, sea, and custom-specified surface.

    - `surfaceReflectivity`
    - `surfaceReflectivityLand`
    - `surfaceReflectivitySea`
    - `surfaceReflectivityCustom`
- The `landreflectivity` function now supports seven land reflectivity models: Barton, APL, Billingsley, GIT, Morchin, Nathanson, and Ulaby-Dobson.
- The `seareflectivity` function now supports nine sea reflectivity models: NRL, APL, GIT, Hybrid, Masuko, Nathanson, RRE, Sittrop, and TSC.

RCS fluctuation modeling

This release enhances the radar cross-section model created by the `rcsSignature` class. The new `FluctuationModel` property enables a statistical RCS fluctuation model that improves the fidelity of the statistical radar model. The `rcsSignature` class is used by the `radarDataGenerator` when simulating detection statistics.

Array scan loss

The new `HasScanLoss` property of the `radarDataGenerator` enables you to include off-broadside losses that arise from electronic scanning. This property models the effect of antenna array beam broadening when the radar points at an off-broadside angle. To enable this feature, set the `HasScanLoss` property to `true` and set the `ScanMode` property to `'Custom'`.

Wrap measurements in tracking filters to prevent filter divergence

You can enable measurement wrapping for these tracking filter objects:

- `trackingEKF`
- `trackingUKF`

First, specify the `MeasurementWrapping` property as `true`, and then specify the `MeasurementFcn` property as a measurement function with two outputs: the measurement and the measurement wrapping bound. With this setup, the filter wraps the measurement residuals according to the measurement bounds, which helps prevent the filter from diverging due to incorrect measurement residual values.

These measurement functions have predefined wrapping bounds:

- `cvmeas`
- `cameas`
- `ctmeas`

For these functions, the wrapping bounds are [–180, 180] degrees for azimuth angle measurements and [–90, 90] degrees for elevation angle measurements. Other measurements are unwrapped.

You can also customize a wrapping-enabled measurement function by returning the wrapping bounds as the second output of the measurement function.

Changes to behavior of trackingKF object

Behavior change

As of R2022a, the `trackingKF` filter object has these behavior changes:

- The object now accepts and uses the process noise specified using the `ProcessNoise` property. Previously, the object ignored the process noise specified in the `ProcessNoise` property.
- If you set the `MotionModel` property to a predefined state transition model, such as `"1D Constant Velocity"`, you can no longer specify the control model for the filter. To use a control model, specify the `MotionModel` property as `"Custom"`.
- To use a control model, you must:
  - Specify the `MotionModel` property as `"Custom"` and use a customized motion model.
  - Specify the control model when creating the filter.

  Also, you cannot change the size of the control model matrix.
- You can no longer change the size of the measurement model matrix specified in the `MeasurementModel` property.
- The dimension of the process matrix set through the `ProcessNoise` property now differentiates between a predefined motion model and a customized motion model.
  - If the specified motion model is a predefined motion model, specify the `ProcessNoise` property as a *D*-by-*D* matrix, where *D* is the dimension of the motion. For example, $D = 2$ for the `"2D Constant Velocity"` motion model.
  - If the specified motion model is a customized motion model, specify the `ProcessNoise` property as an *N*-by-*N* matrix, where *N* is the dimension of the state. For example, $N = 6$ if you customize a 3-D motion model in which the state is (x, v_x, y, v_y, z, v_z).
- The orientation of the state now matches that of the state vector that you specified when creating the filter. For example, if you set the initial state in the filter as a row vector, the filter displays the filter state as a row vector and outputs the state as a row vector when using the `predict` or `correct` object functions. Previously, the filter displayed and output the filter state as a column vector regardless of the initial state.
- You can generate efficient C/C++ code without dynamic memory allocation for `trackingKF`.

Applications: Surface and clutter simulation, multifunction radar, SAR image formation

The release introduces several new application examples.

- Introduction to Radar Scenario Clutter Simulation shows how to generate monostatic surface clutter signals and detections in a radar scenario using `radarDataGenerator` and `radarTransceiver`.
- The Simulating Radar Returns from Moving Sea Surfaces example simulates an X-band radar system used on a fixed offshore platform for oceanographic studies of sea states.
- In the Simulated Land Scenes for Synthetic Aperture Radar Image Formation example, a synthetic aperture radar (SAR) system uses platform motion to mimic a longer aperture to improve cross-range resolution.

- Simulate Radar Detections of Surface Targets in Clutter simulates the detection of targets in surface clutter including the effect of Doppler separation and line-of-sight occlusions on detectability.
- Generate Clutter and Target Returns for MTI Radar shows how to generate surface clutter and target returns in the simulation of moving target indicator (MTI) radar.
- The Quality-of-Service Optimization for Radar Resource Management example sets up a resource management scheme for a multifunction phased array radar (MPAR) surveillance based on a quality-of-service (QoS) optimization.
- The Design and Simulate an FMCW Long-Range Radar (LRR) example configures a `radarDataGenerator` from a radar design exported from the **Radar Designer** app.
- The ERS SAR Raw Data Extraction And Image Formation example shows how to extract European Remote Sensing (ERS) Synthetic Aperture Radar (SAR) system parameters and unfocused raw data and then form a focused image from raw data using a range migration image formation algorithm.
- Processing Radar Reflections Acquired with the Demorad Radar Sensor Platform demonstrates how to process and visualize FMCW radar echoes acquired via the Demorad Radar Sensor Platform with the Phased Array System Toolbox™ and Simulink®.

R2021b

Radar Designer App: Plot vertical coverage diagrams

Starting this release, the **Radar Designer** app can plot vertical coverage diagrams. Vertical coverage diagrams, also known as range-height-angle charts or Blake charts, show the relationship between the range to a target, the height of the target, and the initial elevation angle of the transmitted rays for the sensor. This visualization enables users to understand the propagation characteristics of electromagnetic waves through the Earth's atmosphere.

This release also introduces four functions related to Blake charts:

- `range2height` computes the target height based on the initial elevation angle at the radar, the antenna height, and the propagated range.
- `height2range` computes the propagated range based on the target height, the antenna height, and the elevation angle at the radar.
- `height2grndrange` computes the ground range based on the target height, the antenna height, and the elevation angle at the radar.
- `refractionexp` computes the refraction exponent, which is the decay constant of the Cosmic Ray Physics Laboratory (CRPL) exponential reference atmosphere model.

Synthetic Aperture Radar: Convert between ground range resolution and slant range resolution

This release introduces the `grnd2slantrngres` and `slant2grndrngres` functions.

- `grnd2slantrngres` returns the slant range resolutions corresponding to a set of ground range resolutions and a set of grazing angles.
- `slant2grndrngres` returns the ground range resolutions corresponding to a set of slant range resolutions and a set of grazing angles.

Radar Data Generator block: Generate radar data in Simulink

This release introduces the Radar Data Generator block. The block implements a statistical radar sensor model that generates synthetic data and provides the option to generate tracks, detections, and clustered detections. Radar Data Generator maintains the properties of the `radarDataGenerator` System object.

New custom scan mode for radarDataGenerator

Starting this release, the `radarDataGenerator` System object has a custom scan mode that enables users to point the radar beam in a specific direction.

Merge detections into clustered detections using mergeDetections

Use the `mergeDetections` function to merge detections that share the same cluster labels into clustered detections. By default, the function uses a Gaussian mixture merging algorithm, but you can customize your own detection merging algorithm.

Generate more memory-efficient C/C++ code from tracking filters

These objects now support strict single-precision and static memory allocation code generation:

- `trackingEKF`
- `trackingUKF`

See the Extended Capabilities section on each object reference page for its code generation limitations.

Applications: AI, SAR, Tracking, Radar Coverage, Environment Effects

The release introduces several new application examples:

- Hand Gesture Classification Using Radar Signals and Deep Learning (Deep Learning Toolbox) shows how to classify ultra-wideband (UWB) impulse radar signal data using a multiple-input, single-output convolutional neural network (CNN).
- Introduction to SAR Target Classification Using Deep Learning lets you create and train a simple convolution neural network to classify SAR targets using deep learning.
- Automatic Target Recognition (ATR) in SAR Images shows how to train a Region-based Convolutional Neural Networks (R-CNN) for target recognition in large scene Synthetic Aperture Radar (SAR) images using Deep Learning Toolbox and Parallel Computing Toolbox™.
- Lidar and Radar Fusion in an Urban Air Mobility Scenario shows how to simulate radar and lidar data and how to use multi-object trackers to track various unmanned aerial vehicles (UAVs) in an urban environment.
- Extended Target Tracking with Multipath Radar Reflections in Simulink shows how to model and mitigate multipath radar reflections in a highway driving scenario in Simulink.
- Radar Vertical Coverage over Terrain shows how to visualize 3-D vertical radar coverage over terrain in the presence of heavy clutter.
- Modeling Target Position Estimation Errors discusses some of the environmental factors that can result in detection losses and errors in target parameter estimation.
- Introduction to Scanning and Processing Losses in Pulse Radar demonstrates how various parameters influence the losses that must be included in the radar detectability factor when evaluating the radar equation. These include losses caused by the pulse eclipsing effect, off-broadside scanning with an electronic beam, and MTI processing, CFAR loss, and filter matching loss.
- Introduction to Pulse Integration and Fluctuation Loss in Radar illustrates how to compute gains for several pulse integration techniques. It also demonstrates computation of the losses due to the target's RCS fluctuation.

R2021a

Version: 1.0

New Features

New Radar Toolbox

The new Radar Toolbox features a comprehensive set of algorithms and tools for designing, simulating, analyzing, and testing multifunction radar systems.

Radar Designer App: Model radar gains and losses and assess performance in different environments

The **Radar Designer** app is an interactive tool that assists engineers and system analysts with high-level design and assessment of radar systems at the early stage of radar development. Using the app, you can:

- Assess and compare multiple radar designs in a single session
- Add smart radar, environment, and target Radar Designer Configurations to jump-start your analysis
- Incorporate environmental effects due to Earth's curvature, atmosphere, terrain, and precipitation
- Add custom target radar cross-sections, antenna/array models, and both range-independent and range-dependent losses
- Export and save results, sessions, models, and plots to continue your analysis

Evaluate Radar Performance

Radar Toolbox gives you tools to evaluate the performance of radar systems. You can find these tools in Radar Systems Engineering. Significant capabilities allow you to:

- Evaluate the radar received signal-to-noise ratio as a function of transmitted power and target range (radar equation)
- Derive detection and tracking statistics
- Evaluate antenna and receiver gains and losses
- Compute attenuation losses due to atmospheric effects, clutter, and weather
- Compute signal processing gains and losses for synthetic aperture radars

Create Radar Scenarios

Use the Radar Toolbox to create realistic radar scenarios. Functions for creating radar scenarios can be found in Scenario Generation. You can:

- Model platform motion and orientation based on waypoints and trajectories or by simulating inertial navigation systems
- Use `radarScenario` and other functions to create realistic radar scenarios for airborne, ground-based, and shipborne platforms and targets
- Employ plotting functions to visualize the evolution of the radar scenario over time

Simulate Radar Data

The toolbox helps you create simulated radar data using the functions in Data Synthesis. With these functions you can:

- Simulate radar data at probabilistic or signal levels.
- Generate signal and track data and object detections
- Simulate signal data including effects of multipath propagation, clutter, and interference
- Simulate target echoes from simple geometric shapes or complex structures such as a walking pedestrian or a moving bicyclist

Signal and Data Processing

The toolbox lets you perform signal processing operations on simulated radar data. See Signal and Data Processing for a description of the signal and data processing functions. Among the capabilities are:

- Perform matched filtering and stretch-processing, pulse compression, coherent and noncoherent pulse integration
- Estimate target range, Doppler and angle
- Employ constant false alarm rate (CFAR) techniques to reduce false detections
- Cluster neighboring detections into single extended detections
- Create, delete, and manage tracks for multiple objects

Applications

Radar Applications

- Simulate Radar Ghosts due to Multipath Return
- Highway Vehicle Tracking with Multipath Radar Reflections
- Radar Signal Simulation and Processing for Automated Driving
- Track-to-Track Fusion for Automotive Safety Applications
- Adaptive Tracking of Maneuvering Targets with Managed Radar
- Labeling Radar Signals with Signal Labeler
- Spaceborne Synthetic Aperture Radar Performance Prediction
- Airborne SAR System Design
- Synthetic Aperture Radar System Simulation and Image formation

Radar System Engineering

- Radar Architecture: Part 1 – System components and requirements allocation
- Radar Architecture: Part 2 – Test automation and requirements traceability
- Radar Link Budget Analysis
- Modeling Radar Detectability Factors
- MTI Improvement Factor for a Land-Based Radar System
- Sea Clutter Simulation for a Maritime Radar System
- Introduction to Modeling the Propagation of Radar Signals
- Receiver Operating Characteristic to Tracker Operating Characteristic

Scenario Generation

- Radar Scenario Tutorial
- Radar Performance Analysis Over Terrain

Data Synthesis

- Simulating a Scanning Radar
- Simulating Passive Radar Sensors and Radar Interferences
- Transitioning From Statistical to Physics Based Radar Models