Phased Array System Toolbox™ Release Notes

How to Contact MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

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

For contact information about worldwide offices, see the MathWorks Web site.

Phased Array System Toolbox™ Release Notes

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.

Summary by Version

This table provides quick access to what's new in each version. For clarification, see "Using Release Notes" on page 1.

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Latest Version V1.2 (R2012a)	Yes Details	Yes Summary	Bug Reports
V1.1 (R2011b)	Yes Details	Yes Summary	Bug Reports
V1.0 (R2011a)	Yes Details	No	Bug Reports

Using Release Notes

Use release notes when upgrading to a newer version to learn about:

- New features
- Changes
- Potential impact on your existing files and practices

Review the release notes for other MathWorks® products required for this product (for example, MATLAB® or Simulink®). Determine if enhancements, bugs, or compatibility considerations in other products impact you.

If you are upgrading from a software version other than the most recent one, review the current release notes and all interim versions. For example, when you upgrade from V1.0 to V1.2, review the release notes for V1.1 and V1.2.

What Is in the Release Notes

New Features and Changes

- New functionality

- Changes to existing functionality

Version Compatibility Considerations

When a new feature or change introduces a reported incompatibility between versions, the **Compatibility Considerations** subsection explains the impact.

Compatibility issues reported after the product release appear under Bug Reports at the MathWorks Web site. Bug fixes can sometimes result in incompatibilities, so review the fixed bugs in Bug Reports for any compatibility impact.

Fixed Bugs and Known Problems

MathWorks offers a user-searchable Bug Reports database so you can view Bug Reports. The development team updates this database at release time and as more information becomes available. Bug Reports include provisions for any known workarounds or file replacements. Information is available for bugs existing in or fixed in Release 14SP2 or later. Information is not available for all bugs in earlier releases.

Access Bug Reports using your MathWorks Account.

Documentation on the MathWorks Web Site

Related documentation is available on `mathworks.com` for the latest release and for previous releases:

- Latest product documentation
- Archived documentation

Version 1.2 (R2012a) Phased Array System Toolbox Software

This table summarizes what's new in Version 1.2 (R2012a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes Summary	Bug Reports

New features and changes introduced in this version are

Replicated Subarrays and Partitioned Array Apertures

A *subarray* is an accessible subset of array elements. The following new System objects enable you to create arrays that contain subarrays:

- `phased.ReplicatedSubarray`
- `phased.PartitionedArray`

The following table lists existing System objects that now support operations on arrays that contain subarrays. Each of the objects in the table has a property called `SensorArray` or `Sensor`. You can set that property to an array object that contains subarrays. Also, some of the objects in the table support subarray steering through a new input argument, `STEERANGLE`, in the `step` method.

System Object	SensorArray or Sensor Can Contain Subarrays	step Syntax Can Include Subarray Steering
`phased.AngleDopplerResponse`	Yes	Not applicable
`phased.ArrayGain`	Yes	Yes
`phased.ArrayResponse`	Yes	Yes
`phased.Collector`	Yes	Yes
`phased.ConstantGammaClutter`	Yes	Yes
`phased.MVDRBeamformer`	Yes	Not applicable
`phased.PhaseShiftBeamformer`	Yes	Not applicable
`phased.Radiator`	Yes	Yes
`phased.STAPSMIBeamformer`	Yes	Not applicable
`phased.SteeringVector`	Yes	Yes
`phased.SubbandPhaseShiftBeamformer`	Yes	Not applicable
`phased.WidebandCollector`	Yes	Yes

For more information about using subarrays, see “Subarrays Within Arrays” or Subarrays in Phased Array Antennas.

Compatibility Considerations

The `IncludeElementResponse` property of the `phased.SteeringVector` System object is no longer tunable in V1.2 (R2012a). This change facilitates support for arrays containing subarrays.

You may have code from an earlier version that tunes the value of the `IncludeElementResponse` property of a locked steering vector object. If so, the code will produce an error message in R2012a. You can avoid the error message by calling `release` to unlock the object, or by not changing the value of the `IncludeElementResponse` property.

Stretch Processing

The following new features help you perform pulse compression on linear frequency modulation (FM) waveforms using stretch processing:

- `phased.StretchProcessor` System object
- `stretchfreq2rng` function
- `getStretchProcessor` method of `phased.LinearFMWaveform` System object

Stretch processing is sometimes called *deramping* or *dechirping*.

For more information about using stretch processing, see "Stretch Processing" or Range Estimation Using Stretch Processing.

U/V Space and Phi/Theta Angles

Several enhancements facilitate performing operations in the *u*/*v* coordinate system or in a spherical coordinate system that describes angles using φ and θ instead of azimuth and elevation:

- Visualize radiation patterns in *u*/*v* space using the `plotResponse` method for arrays, antenna elements, and microphone elements. To use this feature, include `'Format','UV'` in the `plotResponse` syntax.
- Convert coordinates from one coordinate system to another using these new functions:
  - `uv2azel`

  - `azel2uv`
  - `phitheta2azel`
  - `azel2phitheta`
  - `uv2phitheta`
  - `phitheta2uv`
- Convert antenna radiation patterns from one coordinate system to another using these new functions:
  - `uv2azelpat`
  - `azel2uvpat`
  - `phitheta2azelpat`
  - `azel2phithetapat`
  - `uv2phithetapat`
  - `phitheta2uvpat`

For examples, see Antenna Radiation Pattern in U/V Coordinates and Antenna Array Analysis with Custom Radiation Pattern. For background information about the coordinate systems, see "Spherical Coordinates".

Multiple-Beam Beamformers

The following beamformers support multiple beamforming directions:

- `phased.PhaseShiftBeamformer`
- `phased.SubbandPhaseShiftBeamformer`
- `phased.MVDRBeamformer`

You can use this capability to model switched-beam systems.

To indicate multiple beamforming directions, use a matrix instead of a vector for the `Direction` property of the beamformer object or the `ANG` input argument of `step`. In earlier versions, the value required a vector. Now, when you specify multiple beamforming directions, the `Y` and `W` outputs of `step` have an extra matrix dimension.

Compatibility Considerations

In V1.2 (R2012a), you can create a MAT-file that stores a beamformer variable specifying multiple directions in the `Direction` property. However, you cannot load that variable from the MAT-file in an earlier version. As an alternative, you can re-create the variable in the earlier version and specify only one beamforming direction.

Support for Wideband Beam Pattern Analysis

The `plotResponse` method for arrays, antenna elements, and microphone elements has enhancements for use with wideband beamforming applications.

- For arrays or elements, `plotResponse` can plot multiple frequency responses in a three-dimensional waterfall plot. To use this feature, include `'OverlayFreq',false` in the `plotResponse` syntax. The `OverlayFreq` argument is new.
- For arrays, `plotResponse` can apply weights independently to each frequency in the plot. For example, you can use beamformer weights as in "Visualization of Wideband Beamformer Performance". To use this feature, include `'Weights',Value` in the `plotResponse` syntax, where `Value` is a vector or matrix. R2011b required that the weights be the same for all frequencies in the plot and that `Value` be a vector.

In the `phased.ArrayResponse` and `phased.ArrayGain` System objects, the `step` method permits the `WEIGHTS` input argument to be a vector or a matrix. In earlier releases, `WEIGHTS` is a vector.

Beamformer Option to Preserve Power

The phase shift beamformer offers options for normalizing the beamformer weights. To select an option, set the new `WeightsNormalization` property of the `phased.PhaseShiftBeamformer` object to one of these values:

- `'Distortionless'` — The gain toward the beamforming direction is 0 dB. This choice is the default and matches the behavior in earlier versions.
- `'Preserve power'` — The norm of the weights is 1.

Compatibility Considerations

In V1.2 (R2012a), if you save a phase shift beamformer variable in a MAT-file, you cannot load that variable from the MAT-file in an earlier version. Instead, re-create the variable in the earlier version.

Symmetric Sweeping of Linear FM Waveform

You can create a linear FM waveform to sweep in an interval that is symmetric about 0 or positive only. To choose the location of the FM sweep interval, set the new `SweepInterval` property of the `phased.LinearFMWaveform` object to one of these values:

- `'Positive'` — The waveform sweeps between 0 and B, where B is the sweep bandwidth. This choice is the default and matches the behavior in earlier versions.
- `'Symmetric'` — The waveform sweeps between $-B/2$ and $B/2$.

Toolbox Location of dutycycle Function

The `dutycycle` function in the Signal Processing Toolbox™ product replaces the earlier `dutycycle` function in the Phased Array System Toolbox™ product. The new function includes both the capabilities of the earlier function and additional new capabilities.

New System Object Option on File Menu

The File menu on the MATLAB desktop now includes a **New > System object** menu item. This option opens a System object class template, which you can use to define a System object class.

Variable-Size Input Support for System Objects

System objects that you define now support inputs that change size at runtime.

Data Type Support for System Objects

System objects that you define now support all MATLAB data types as inputs and outputs.

New Property Attribute to Define States

R2012a adds the new `DiscreteState` attribute for properties in your System object class definition file. Discrete states are values calculated during one step of an object's algorithm that are needed during future steps.

New Methods to Validate Properties and Get States from System Objects

The following methods have been added:

- `validateProperties` – Checks that the System object is in a valid configuration. This applies only to objects that have a defined `validatePropertiesImpl` method
- `getDiscreteState` – Returns a `struct` containing a System object's properties that have the `DiscreteState` attribute

matlab.system.System changed to matlab.System

The base System object class name has changed from `matlab.system.System` to `matlab.System`.

Compatibility Considerations

The previous `matlab.system.System` class will remain valid for existing System objects. When you define new System objects, your class file should inherit from the `matlab.System` class.

Version 1.1 (R2011b) Phased Array System Toolbox Software

This table summarizes what's new in Version 1.1 (R2011b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes Summary	Bug Reports

New features and changes introduced in this version are

Constant Gamma Clutter Modeling

The new `phased.ConstantGammaClutter` System object helps you model surface clutter using the constant gamma model. You can use this object when simulating a radar system or estimating its performance statistically.

For more information, see these resources:

- "Clutter Modeling"
- `phased.ConstantGammaClutter`

- Ground Clutter Mitigation with Moving Target Indication (MTI) Radar

Clutter Modeling Utilities

These new utility functions can help you implement custom clutter models:

- `billingsleyicm`
- `depressionang`
- `effearthradius`
- `grazingang`
- `horizonrange`
- `surfclutterrcs`
- `surfacegamma`

Phase-Coded Waveforms

The new `phased.PhaseCodedWaveform` System object generates samples of a phase-coded pulse waveform. This object supports these code types:

- Barker
- Frank
- P1
- P2
- P3
- P4
- Px
- Zadoff-Chu

For more information, see "Phase-Coded Waveforms" and `phased.PhaseCodedWaveform`.

Spectrum Weighting Options in Matched Filter

The `phased.MatchedFilter` System object supports spectrum weighting using these window types:

- Hamming
- Chebyshev
- Hann
- Kaiser
- Taylor

You can also specify a custom window. To do so, write a function that takes the window length as an input argument and returns window coefficients in an output argument.

For more information, see "Matched Filtering" or `phased.MatchedFilter`.

Compatibility Considerations

If you save a `phased.MatchedFilter` object in a MAT-file in V1.1 (R2011b) and then load the MAT-file in V1.0 (R2011a), the object does not perform spectrum weighting. The Command Window shows this warning:

```
Warning: While loading an object of class 'phased.MatchedFilter':
No public field SpectrumWindow exists for class phased.MatchedFilter.
```

If you write code in V1.1 (R2011b) that sets or reads any of the following properties of `phased.MatchedFilter` object, the code produces an error message in V1.0 (R2011a).

- `SpectrumWindow`
- `CustomSpectrumWindow`
- `SpectrumRange`
- `SampleRate`
- `SidelobeAttenuation`
- `Beta`

- `Nbar`

Expanded Lattice Options in Uniform Rectangular Array

The `phased.URA` System object supports both triangular lattices and rectangular lattices. You use the `Lattice` property to select the lattice type.

In V1.0 (R2011a), `phased.URA` supports only rectangular lattices and does not have a `Lattice` property.

Compatibility Considerations

If you save a `phased.URA` object in a MAT-file in V1.1 (R2011b) and then load the MAT-file in V1.0 (R2011a), the object uses a rectangular lattice. The Command Window shows this warning:

```
Warning: While loading an object of class 'phased.URA':
No public field Lattice exists for class phased.URA.
```

If you write code in V1.1 (R2011b) that sets or reads the `Lattice` property of a `phased.URA` object, the code produces an error message in V1.0 (R2011a).

Enhanced Plots Show Multiple Frequency Responses

The `plotResponse` method can plot multiple frequency responses along an azimuth cut or elevation cut. This method is available for the System objects for array design, antenna elements, and microphone elements. To create a plot of multiple frequency responses, use a `plotResponse` syntax in which:

- `FREQ` is a row vector.
- `RespCut` either does not appear explicitly, or has the value `'Az'` or `'El'`.

The affected System objects are:

- `phased.ConformalArray`
- `phased.CosineAntennaElement`
- `phased.CustomAntennaElement`
- `phased.CustomMicrophoneElement`

- `phased.IsotropicAntennaElement`
- `phased.OmnidirectionalMicrophoneElement`
- `phased.ULA`
- `phased.URA`

In V1.0 (R2011a), `FREQ` must be a scalar. The resulting plot shows one frequency response.

Custom Antenna Removes Restriction on Radiation Pattern

The `phased.CustomAntennaElement` System object now permits more general radiation patterns. The main beam of the pattern is no longer required to point to 0 degrees azimuth and 0 degrees elevation.

Storing States When Saving or Cloning Objects

The `save` and `clone` operations now store all states of the System objects in the `phased` package. As a result, calling the `step` method on a loaded or cloned object resumes processing from the state where the original object left off. In V1.0 (R2011a), the loaded or cloned object is unlocked and uninitialized.

Compatibility Considerations

If your legacy code exploits the unlocked, uninitialized state of a loaded or cloned object, you should update the code in V1.1 (R2011b). You can use the `release` method to unlock objects.

Custom System Objects

You can now create custom System objects in MATLAB. This capability allows you to define your own System objects for time-based and data-driven algorithms, I/O, and visualizations. The System object API provides a set of implementation and service methods that you incorporate into your code to implement your algorithm. See “Define New System Objects” in the DSP System Toolbox™ documentation for more information.

Conversion of Error and Warning Message Identifiers

For version 1.1 (R2011b), some error and warning message identifiers have changed in Phased Array System Toolbox software.

Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages, or in code that uses a try/catch statement and performs an action based on a specific error identifier.

For example, the `'phased:phased:RootWSFEstimator:ZeroSourceNumber'` identifier and the `'phased:phased:RootMUSICEstimator:ZeroSourceNumber'` identifier have both changed to `'phased:phased:doa:ZeroSourceNumber'`. If your code checks for one of the earlier values, you must update it to check for `'phased:phased:doa:ZeroSourceNumber'` instead.

To determine the identifier for a warning, run the following command just after you see the warning:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable `MSGID`.

To determine the identifier for an error, run the following command just after you see the error:

```
exception = MException.last;
MSGID = exception.identifier;
```

Note Warning messages indicate a potential issue with your code. While you can turn off a warning, a suggested alternative is to change your code so it runs warning-free.

Version 1.0 (R2011a) Phased Array System Toolbox Software

This table summarizes what's new in Version 1.0 (R2011a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	No	Bug Reports

Introducing the Phased Array System Toolbox

Phased Array System Toolbox provides algorithms and tools for the design, simulation, and analysis of phased array signal processing systems. These capabilities are provided as MATLAB functions and MATLAB System objects. The system toolbox includes algorithms for waveform generation, beamforming, direction of arrival estimation, target detection, and space-time adaptive processing. The system toolbox lets you build monostatic, bistatic, and multistatic architectures for a variety of array geometries. You can model these architectures on stationary or moving platforms. Array analysis and visualization tools help you evaluate spatial, spectral, and temporal performance. The system toolbox lets you model an end-to-end phased array system or use individual algorithms to process acquired data.

Features

Key features of Phased Array System Toolbox Version 1.0 include:

- Algorithms available as MATLAB functions and MATLAB System objects
- Monostatic, bistatic, and multistatic phased array system modeling
- Array analysis and 3D visualization; physical array modeling for uniform linear arrays, uniform rectangular arrays, and arbitrary conformal arrays on platforms with motion

- Broadband and narrowband digital beamforming functions, including MVDR/Capon, LCMV, time delay, Frost, time delay LCMV, and subband phase shift
- Space-time adaptive processing algorithms, including displaced phase center array (DPCA), adaptive DPCA, sample matrix inversion (SMI) beamforming, and angle-Doppler response visualization
- Direction of arrival algorithms, including MVDR, ESPRIT, Beamscan, Root MUSIC, and monopulse tracking
- Waveform synthesis functions for pulsed CW, linear FM, stepped FM, and staggered PRF signals, and waveform visualization tools for ambiguity function and matched filter response
- Algorithms for TVG, pulse compression, coherent and non-coherent integration, CFAR processing, plotting ROC curves, and estimating range and Doppler

Compatibility Summary for Phased Array System Toolbox

This table summarizes new features and changes that might cause incompatibilities when you upgrade from an earlier version, or when you use files on multiple versions. Details are provided in the description of the new feature or change.

Version (Release)	New Features and Changes with Version Compatibility Impact
Latest Version V1.2 (R2012a)	• "Replicated Subarrays and Partitioned Array Apertures" on page 4 • "Multiple-Beam Beamformers" on page 6 • "Beamformer Option to Preserve Power" on page 7 • "matlab.system.System changed to matlab.System" on page 9
V1.1 (R2011b)	• "Spectrum Weighting Options in Matched Filter" on page 12 • "Expanded Lattice Options in Uniform Rectangular Array" on page 13 • "Storing States When Saving or Cloning Objects" on page 14 • "Conversion of Error and Warning Message Identifiers" on page 15
V1.0 (R2011a)	None