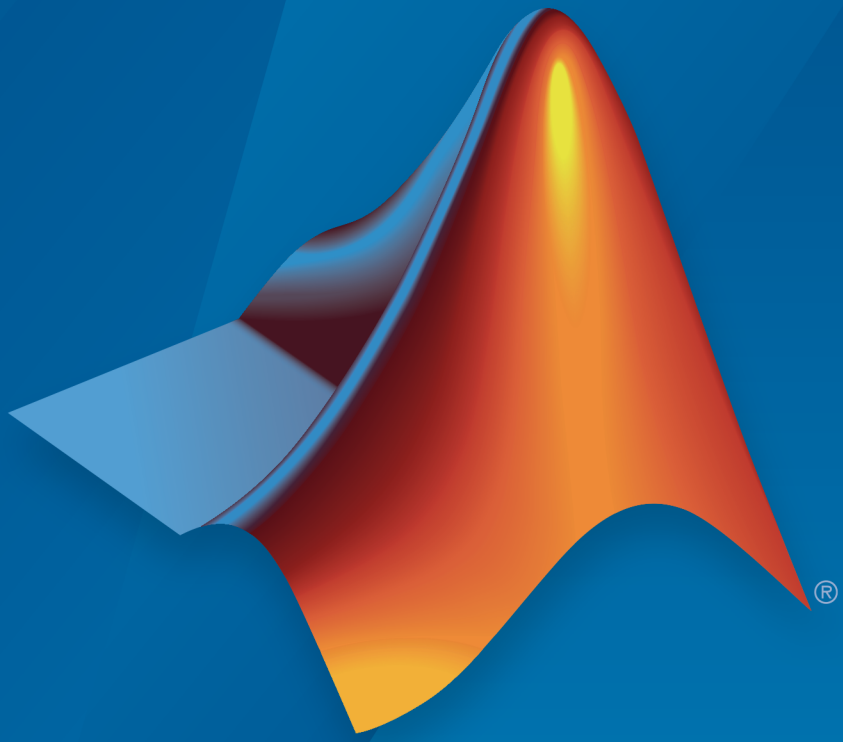
Partial Differential Equation Toolbox™ Release Notes



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.
3 Apple Hill Drive
Natick, MA 01760-2098

Partial Differential Equation Toolbox™ Release Notes

Contents

R2016a

R2015b

R2015a

R2014b

R2014a

R2013b

R2013a

R2012b

R2016a

PDE Solvers: Use `solvepde` and `solvepdeeig` functions to solve PDE equations and eigenvalue problems

New solver functions for `PDEModel`:

- `solvepde` replaces `assempde`, `pdenonlin`, `hyperbolic`, and `parabolic`.
- `solvepdeeig` replaces `pdeeig`.

To use the new solvers, include PDE coefficients in your model using `specifyCoefficients`. Include initial conditions in your model using `setInitialConditions`.

The new solvers return results as one of the three new objects:

- `StationaryResults` — Returned by `solvepde` for a stationary PDE model.
- `TimeDependentResults` — Returned by `solvepde` for a time-dependent PDE model.
- `EigenResults` — Returned by `solvepdeeig`.

`StationaryResults` and `TimeDependentResults` objects contain solution gradients at the nodes.

PDE Coefficients: Specify equation coefficients as a property of PDEModel

The `specifyCoefficients` function specifies equation coefficients as a property of `PDEModel`.

Initial Conditions: Set initial conditions or initial guess as a property of PDEModel

The `setInitialConditions` function specifies initial conditions as a property of `PDEModel`.

Quadratic Elements for 2-D Mesh: Generate 2-D mesh using quadratic triangular elements

Create a quadratic mesh for 2-D problems using `generateMesh` with `GeometricOrder` set to `'quadratic'`.

Gradient of PDE Solution: Evaluate solution gradient at arbitrary 2-D or 3-D points

The `evaluateGradient` function enables you to interpolate the gradient of a `StationaryResults` or `TimeDependentResults` object at arbitrary points in the geometry.

Finite Element Matrices: Use `assembleFEMatrices` to assemble finite element matrices

The `assembleFEMatrices` function assembles finite element matrices for independent factoring and solution using linear algebra methods. It replaces `assempde`, `assema`, and `assemb` for matrix assembly.

PDE Results for Plotting and Postprocessing: New result objects depend on the type of PDE

The `createPDEResults` function returns results as one of the three new objects, depending on the type of the PDE problem.

- A `StationaryResults` object for a stationary PDE model. `StationaryResults` contains the solution of PDE and its gradients at the nodal locations.
- A `TimeDependentResults` object for a time-dependent PDE model. `TimeDependentResults` contains the solution of PDE and its gradients at the nodal locations.
- A `EigenResults` object for an eigenvalue problem.

Compatibility Considerations

`createPDEResults` no longer creates an object of type `PDEResults`.

The syntax of `createPDEResults` has changed to accommodate creating the new result types for time-dependent and eigenvalue problems.

- To create the `TimeDependentResults` object for a time-dependent problem, use the syntax `createPDEResults(pdem,u,utimes,'time-dependent')`, where `utimes` is a vector of solution times.
- To create the `EigenResults` object for an eigenvalue problem, use the syntax `createPDEResults(pdem,eigenvectors,eigenvalues,'eigen')`.

`EigenResults` has different property names than `PDEResults`. Update any eigenvalue scripts that use `PDEResults` property names.

Functionality being removed or changed

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
`wbound`	Still runs	No replacement	New features and the recommended workflow are not compatible with this function. For the recommended workflow, see "Solve Problems Using PDEModel Objects".
`pdeadgsc` and `pdeadworst`	Still runs	No replacement	New features and the recommended workflow are not compatible with these functions. For the recommended workflow, see "Solve Problems Using PDEModel Objects".
`assempde`, `assema`, `assemb`, `hyperbolic`, `parabolic`, `pdenonlin`	Still runs	`solvepde` and `assembleFEMatrices`	To solve PDE problems, use `solvepde`. To obtain finite element matrices, use `assembleFEMatrices`. For the recommended workflow, see "Solve Problems Using PDEModel Objects".
`pdeeig` and `sptarn`	Still runs	`solvepdeeig`	To solve PDE eigenvalue problems, use `solvepdeeig`. For the recommended workflow, see "Solve Problems Using PDEModel Objects".

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
`poimesh`, `poiasma`, `poicalc`, `poiindex`, `poisolv`	Still runs	`solvepde`	To solve Poisson's equations, use `solvepde`. For details, see "Solve Problems Using PDEModel Objects".
`pdejmps`	Still runs	No replacement	New features and the recommended workflow are not compatible with this function. For the recommended workflow, see "Solve Problems Using PDEModel Objects".
`pdesmech`	Still runs	PDE app	Use the PDE app instead of `pdesmech`.
`dst` and `idst`	Still runs	No replacement	Remove all instances of `dst` and `idst`.
`tri2grid` and `pdeintrp`	Still runs	`interpolateSolution`	Use the interpolation function `interpolateSolution` provided by `StationaryResults`, `TimeDependentResults` and `EigenResults`.
`pdeprtni`	Still runs	No replacement	`NodalSolution` is a property of `StationaryResults` and `TimeDependentResults`. `Eigenvectors` is the corresponding property of `EigenResults`.
`pdemdlcv`	Still runs	No replacement	Remove all instances of `pdemdlcv`.

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
`pde`	Warns	`createpde`	Use `createpde` to create a `PDEModel` that holds the PDE analysis data. The `pde` class was a value class. The replacement `PDEModel` class is a handle class.
`pdeGeometryFromEdg`	Warns	`geometryFromEdges`	Use `geometryFromEdges` instead. Although `pdeBoundaryConditions` still runs with a warning, its returned type has changed from a `pdeGeometry` object to an `AnalyticGeometry` object. The `pdeGeometry` class was a value class. The replacement `AnalyticGeometry` class is a handle class.

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
`pdeBoundaryConditi`	Warns	`applyBoundaryConditio`	Replace all instances of `pdeBoundaryConditions(ApplicationReg` with `applyBoundaryCondition(model,'edge',` Although `pdeBoundaryConditions` still runs with a warning, its returned type has changed from a `pdeBoundaryConditions` object to a `BoundaryCondition` object. The `pdeBoundaryConditions` class was a value class. The replacement `BoundaryCondition` class is a handle class.
Loading a `pdeBoundaryConditi` object from an R2014b MAT file.	Errors	`applyBoundaryConditio`	Recreate the Boundary Conditions using `applyBoundaryCondition`.
Function handle for specifying nonconstant boundary conditions and coefficients of the form `@f(problem, region,state)`.	Still runs.	`@f(region,state)`	Use `specifyCoefficients` and define a function handle that takes only two arguments: `@f(region,state)`

R2015b

3-D geometry creation from a finite element mesh using the `geometryFromMesh` function

The geometryFromMesh function creates 3-D geometry from a finite element mesh, or from a triangulated surface mesh. For details, see the function reference page or Create and View 3-D Geometry.

Data structure to represent solutions using the `createPDEResults` function

The createPDEResults function converts a PDE solution into a PDEResults object. The `PDEResults` object allows you to interpolate the solution using interpolateSolution. For details, see the reference pages.

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`pdesmech`	Still runs	PDE app	Use the PDE app instead of `pdesmech`

R2015a

3-D finite element analysis

You can now solve partial differential equations with 3-D geometry. To do so, there is a new workflow that combines the geometry, mesh, and boundary conditions into a PDEModel object. You can also use this workflow for 2-D geometry. For details, see Solve Problems Using PDEModel Objects.

Equation coefficients and boundary conditions for 3-D problems

To specify problem coefficients or boundary conditions in 3-D geometry, you can use strings with a syntax similar to that of a 2-D problem. There is a new way of writing functions for coefficients in 3-D geometries. For details, see PDE Coefficients and Boundary Conditions.

Compatibility Considerations

To accommodate both 2-D and 3-D geometry, the format of boundary condition objects changed from that introduced in R2014b. The new object is BoundaryCondition Properties, and calling `pdeBoundaryConditions` now warns that it will be removed in a future release. If you saved a `pdeBoundaryConditions` object in an R2014b-format MAT file, then loading that file in R2015a can produce an error. Additionally, the syntax for specifying nonconstant boundary conditions has changed. Functions written in the previous syntax continue to work for now.

R2014b Syntax	R2015a Syntax
`function bcMatrix = myfun(problem,region,state)`	`function bcMatrix = myfun(region,state)`

For details, see Changes to Boundary Conditions Object From R2014b.

Elliptic, parabolic, hyperbolic, nonlinear, eigenvalue solvers for 3-D problems

The main toolbox solvers now support problems with 3-D geometry. For a listing of functions that do or do not support 3-D geometry, see Functions That Support 3-D Geometry. Solvers take a `model` argument instead of the previous `b`, `p`, `e`, `t` arguments. For details, see the function reference pages.

3-D geometry import from STL files

Import the geometry for a 3-D problem in the STL file format using the importGeometry function. For details, see Create and View 3-D Geometry.

3-D unstructured meshing using tetrahedra

Create finite element meshes using the generateMesh function. For 3-D geometry, the meshes consist of tetrahedra. See Mesh Data for [p,e,t] Triples: 3-D.

Plot function to inspect 3-D solutions

The pdeplot3D function plots solutions on the boundaries of 3-D geometry. For details, see Plot 3-D Solutions.

Featured examples with 3-D geometry

There are two new featured examples related to linear elasticity that have 3-D geometry:

- Deflection Analysis of a Bracket
- Vibration of a Square Plate

There is also a new example of plotting slices through a 3-D solution: Contour Slices Through a 3-D Solution.

To run the examples at the MATLAB® command line:

```
echodemo StrainedBracketExample
echodemo Eigenvaluesofa3DPlateExample
echodemo ContourSlices3DExample
```

`pdebound` and `pdegeom` reference pages removed

The `pdebound` and `pdegeom` reference pages have been replaced by the Boundary Conditions and 2-D Geometry documentation categories.

R2014b

Version: 1.5

New Features

Bug Fixes

Functions for modular definition of boundary conditions

To specify PDE boundary conditions in a modular fashion, per edge or set of edges, use a pdeBoundaryConditions specification. For details, see Steps to Specify a Boundary Conditions Object.

`pdeInterpolant` object for solution interpolation

Interpolate a PDE solution to a set of points using evaluate on an interpolant. Create the interpolant using pdeInterpolant.

R2014a

Version: 1.4

New Features

Bug Fixes

Damping option for `hyperbolic` solver

You can include damping in the hyperbolic solver in matrix form. There is a new example of dynamics of a damped cantilever beam that shows how to use this feature.

R2013b

Version: 1.3

New Features

Bug Fixes

Display option in `hyperbolic` and `parabolic` solvers

You can disable the display of internal ODE solution details that the hyperbolic and parabolic solvers report. To disable the display, set the `Stats` name-value pair to `'off'`.

Eigenvalue example

There is a new example of eigenvalues of a circular membrane. View the example here. To run the example at the MATLAB command line:

```
echodemo eigsExample
```

R2013a

Version: 1.2

New Features

Bug Fixes

Performance and robustness enhancements in meshing algorithm

The meshing (geometry triangulation) functions in initmesh and adaptmesh provide an enhancement option for increased meshing speed and robustness. Choose the enhanced algorithm by setting the `MesherVersion` name-value pair to `'R2013a'`. The default `MesherVersion` value of `'preR2013a'` gives the same mesh as previous toolbox versions.

The enhancement is available in`pdetool` from the **Mesh > Parameters > Mesher version** menu.

New example

There is a new example of heat distribution in a radioactive rod. View the example here. To run the example at the MATLAB command line:

```
echodemo radioactiveRod
```

R2012b

Version: 1.1

New Features

Compatibility Considerations

Coefficients of parabolic and hyperbolic PDEs that can be functions of the solution and its gradient

You can now solve parabolic and hyperbolic equations whose coefficients depend on the solution u or on the gradient of u. Use the parabolic or hyperbolic commands, or solve the equations using `pdetool`. For details, see the function reference pages.

Graphics export from `pdetool`

You can save the current pdetool figure in a variety of image formats. Save the figure using the **File > Export Image** menu. See File Menu.

`pdegplot` labels edges and subdomains

`pdegplot` now optionally labels:

- The edges in the geometry
- The subdomains in the geometry

To obtain these labels, set the `edgeLabels` or `subdomainLabels` name-value pairs to `'on'`. For details, see the pdegplot reference page.

New examples

There is a new example of uniform pressure load on a thin plate. View the example here. To run the example at the MATLAB command line:

```
echodemo clampedSquarePlateExample
```

There is a new example of nonlinear heat transfer in a thin plate. View the example here. To run the example at the MATLAB command line:

```
echodemo heatTransferThinPlateExample
```

There is a new example of a system of coupled PDEs. View the example here. To run the example at the MATLAB command line:

```
echodemo deflectionPiezoelectricActuator
```

pdesmech shear strain calculation change

The pdesmech function now calculates shear strain according to the engineering shear strain definition. This has always been the documented behavior. However, the previous calculation was performed according to the tensor shear strain calculation, which gives half the value of the engineering shear strain.

Compatibility Considerations

`pdesmech` now returns shear strain values exactly twice as large as before.