Optimization Toolbox Release Notes

The "Optimization Toolbox 2.2 Release Notes" on page 1-1 describe the changes introduced in the latest version of the Optimization Toolbox. The following topics are discussed in these Release Notes:

- "New Features" on page 1-2
- "Major Bug Fixes" on page 1-4
- "Upgrading from an Earlier Release" on page 1-5

If you are upgrading from a release earlier than Release 12.1, you should also see these sections:

- "Optimization Toolbox 2.1.1 Release Notes" on page 2-1
- "Optimization Toolbox 2.1 Release Notes" on page 3-1

If you are upgrading from a release prior to Release 11.1, see the Release 11 New Features Guide. Note that this is a PDF document.

Printing the Release Notes

If you would like to print the Release Notes, you can link to a PDF version.

Contents

1 Optimization Toolbox 2.2 Release Notes

2 Optimization Toolbox 2.1.1 Release Notes

3 Optimization Toolbox 2.1 Release Notes

1

Optimization Toolbox 2.2 Release Notes

New Features

This section summarizes the new features and enhancements introduced in the Optimization Toolbox 2.2.

If you are upgrading from a release earlier than Release 12.1, then you should also see "New Features" on page 2-2 of the Optimization Toolbox 2.1.1 Release Notes.

New fsolve Default Algorithm

The `fsolve` function, which is used to solve systems of nonlinear equations, has a new default algorithm for medium-scale systems where the number of equations is equal to the number of variables. The new algorithm uses a trust-region dogleg method that has improved convergence properties over the previous default method. It is based on the algorithm described in [1].

Medium-Scale Is Now the Default Method

In keeping with the new default trust-region dogleg algorithm, `fsolve` now defaults to the medium-scale method. The default for the `'LargeScale'` parameter is changed to `'off'`.

To use the large-scale `fsolve` method, which was the default in Versions 2.0 and 2.1, specify the large-scale method using code similar to

```
options = optimset('LargeScale','on');
xsol = fsolve(myfun,x0,options);
```

New 'NonlEqnAlgorithm' Parameter

A new `'NonlEqnAlgorithm'` `fsolve` parameter enables you to choose the Levenberg-Marquardt or Gauss-Newton algorithm over the trust-region dogleg algorithm. `'NonlEqnAlgorithm'` has three options: `'dogleg'` (default), `'lm'` (Levenburg-Marquart), and `'gn'` (Gauss-Newton). Specify the use of an algorithm other than the default with code similar to

```
options = optimset('NonlEqnAlgorithm','gn');
xsol = fsolve(myfun,x0,options);
```

See "Upgrading from an Earlier Release" on page 1-5 for more information.

References

[1] J.J. Moré, B.S. Garbow, and K.E. Hillstrom, “User Guide for MINPACK - 1,” Argonne National Laboratory, Rept. ANL-80-74, 1980

Function Summary

Version 2.2 of the Optimization Toolbox provides the following functions with new or changed capabilities.

Optimization Functions with New or Changed Capabilities

Function	Enhancement or Change
`fsolve`	The trust-region dogleg algorithm is now the default for medium-scale systems of equations where the number of equations is equal to the number of variables. See “New fsolve Default Algorithm” on page 1-2 for more information.
`optimset`	A new `'NonlEqnAlgorithm'` parameter, which applies only to `fsolve`, enables you to choose the Levenberg-Marquardt or Gauss-Newton algorithm over the new default trust-region dogleg algorithm.

Major Bug Fixes

The Optimization Toolbox 2.2 includes several bug fixes made since Version 2.1.1. This section describes the particularly important Version 2.2 bug fixes.

If you are viewing these Release Notes in PDF form, please refer to the HTML form of the Release Notes, using either the Help browser or the MathWorks Web site and use the link provided.

Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Optimization Toolbox 2.1.1 to Version 2.2.

If you are upgrading from a version earlier than 2.1.1, then you should also see "Upgrading from an Earlier Release" on page 3-5 in the Optimization Toolbox 2.1. Release Notes.

fsolve Now Defaults to New Medium-Scale Method

In keeping with the new default trust-region dogleg algorithm, `fsolve` now defaults to the medium-scale method. The default for the `'LargeScale'` parameter is changed to `'off'`.

Using the Large-Scale Method

If you want to continue using the large-scale `fsolve` method, which was the default in Versions 2.0 and 2.1, you must explicitly specify the large-scale method using code similar to

```
options = optimset('LargeScale','on');
xsol = fsolve(myfun,x0,options);
```

Using the Gauss-Newton Method

In prior releases to get the Gauss-Newton method you did:

```
options = optimset('LargeScale','off',...
                   'LevenbergMarquardt','off');
```

For this release you do:

```
options = optimset('LargeScale','off',...
                   'NonlEqnAlgorithm','gn');
```

Note Strictly speaking, it is unnecessary to set `'LargeScale'` to `'off'`, unless you have previously set it `'on'`.

Using the Levenberg-Marquardt Method

In prior releases to get the Levenberg-Marquardt method you did:

```
options = optimset('LargeScale','off',...
                   'LevenbergMarquardt','on');
```

For this release you do:

```
options = optimset('largescale','off',...
                   'NonlEqnAlgorithm','lm');
```

Use playshow Command to Run Slideshow Style Demos

Starting in Release 13, to run slideshow style demos from the command line, you must use the playshow command. For example,

```
playshow circustent
```

You can continue to run other styles of demos from the command line by typing just the demo name. Optimization Toolbox demos that are affected: `circustent`, `molecule`, `optdeblur`.

2

Optimization Toolbox 2.1.1 Release Notes

New Features

This section introduces the new features and enhancements added in the Optimization Toolbox 2.1.1 since the Optimization Toolbox 2.1 (Release 12.0).

For information about Optimization Toolbox features that are incorporated from Version 2.1, see “New Features” on page 3-2.

Support for Large Problems that Are Not Well-Scaled

In Version 2.1.1, large-scale finite differencing is improved numerically to handle cases when an optimization problem is not well-scaled. These changes potentially improve the speed and accuracy of results when using the large-scale versions of `lsqnonlin`, `lsqcurvefit`, `fsolve`, `fmincon` and `fminunc`, particularly if the objective function is not well-scaled.

Major Bug Fixes

The Optimization Toolbox includes several bug fixes, including the following descriptions of particularly important bug fixes for both Version 2.1 (Release 12.0) and Version 2.1.1 (Release 12.1).

Upgrading from an Earlier Release

There are no upgrade issues moving from the Optimization Toolbox 2.1 to the Optimization Toolbox 2.1.1.

See “Upgrading from an Earlier Release” on page 3-5 for upgrade issues involved in moving from the Optimization Toolbox 2.0 (Release 11.0) to the Optimization Toolbox 2.1 (Release 12.0).

3

Optimization Toolbox 2.1 Release Notes

New Features

This section introduces the new features and enhancements added in the Optimization Toolbox 2.1 since the Optimization Toolbox 2.0 (Release 11.0):

- Overall toolbox speed is improved.
- Functions that take a function as an argument now support the use of function handles.
- Large structured problems are supported.
- Several other existing functions have been enhanced.

Toolbox Speed

By improving the speed of `optimset` and `optimget`, the overall speed of the toolbox is improved. Applications that call these functions repeatedly should exhibit improved execution time.

Function Handles

For any optimization function that expects a function as an argument, you can now specify that argument as a function handle. These optimization functions also accept additional parameters, which they pass to the passed-in function.

For information about function handles, see the `function_handle` (@), `func2str`, and `str2func` reference pages, and the "Function Handles" section of "Programming and Data Types" in the MATLAB documentation.

Large Structured Problems

The functions `fmincon`, `fminunc`, `fsolve`, `lsqcurvefit`, `lsqlin`, `lsqnonlin`, and `quadprog` now support solving large structured problems, i.e., problems that have large dense Hessian or Jacobian matrices that you do not want to form explicitly, but for which Hessian-matrix (or Jacobian-matrix) products are efficient to compute.

Two new `options` parameters, `HessMult` and `JacobMult`, provide access to this new feature.

See "Large Scale Examples" in the *Optimization Toolbox User's Guide*, and the respective function reference pages for more information.

Functions with New or Changed Capabilities

Function	New or Changed Capability
`fminbnd`, `fminsearch`, `fzero`, `lsqnonneg`	A new `Display` options parameter value, `'notify'`, displays output only if the function does not converge. For these functions, `'notify'` is the new default.
`fmincon`, `fminunc`, `quadprog`	A new options parameter, `HessMult`, enables you to provide a function that computes the Hessian-matrix product for large structured problems.
`fsolve`, `lsqcurvefit`, `lsqlin`, `lsqnonlin`	A new options parameter, `JacobMult`, enables you to provide a function that computes the Jacobian-matrix product for large structured problems.

Major Bug Fixes

The Optimization Toolbox includes several bug fixes, including the following descriptions (online only) of particularly important bug fixes.

Upgrading from an Earlier Release

The Optimization Toolbox uses an options structure to access various algorithm options. The Version 2.0 (Release 11) version of the options structure may be incompatible with the 2.1 (Release 12.0) version in some cases. In particular, if you have saved an options structure in a MAT-file from Release 11 and load it into Release 12.0 you may get an error similar to

```
??? Error using ==> subsref
Reference to non-existent field 'MaxSQPIter'.
```

To avoid this error, wrap a call to `optimset` around the old options structure to update it. For example,

```
optionsnew = optimset(optionsold);
```

and then use the `optionsnew` options structure.

If you were using `optimset` to create your options structure, as opposed to loading it from an older MAT-file, you should not need to make any changes.