

Global Optimization 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.
3 Apple Hill Drive
Natick, MA 01760-2098

Global Optimization Toolbox Release Notes

© COPYRIGHT 2005–2015 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.

R2015b

ga measures timing using tic and toc	1-2
gatool and psearchtool have been removed	1-2

R2015a

Bug Fixes

R2014b

particleswarm solver for particle swarm optimization	3-2
Nonlinear constraints in gamultiobj multiobjective genetic algorithm solver	3-2
Nonlinear constraint algorithm option in ga genetic algorithm solver can speed solutions	3-2

R2014a

Genetic algorithm changes	4-2
Parallel option change	4-3
Default fmincon algorithm for GlobalSearch and MultiStart	4-3

R2013b

No New Features or Changes

R2013a

Vectorized scalar patternsearch assumes row orientation ..	6-2
---	------------

R2012b

Example of mixed integer programming using ga	7-2
--	------------

R2012a

No New Features or Changes

R2011b

Mixed Integer Nonlinear Programming in Genetic Algorithm Solver	9-2
New Demo	9-2
Conversion of Error and Warning Message Identifiers	9-2

R2011a

“History to New Window” Output Functions Removed	10-2
---	------

R2010b

Output Functions and Plot Functions for GlobalSearch and MultiStart	11-2
Demo Removed	11-2

R2010a

Toolbox Renamed and Expanded	12-2
New GlobalSearch and MultiStart Solver Objects	12-2
New patternsearch Poll Method	12-3
New Demo	12-3

threshacceptbnd Function Removed	12-3
---	-------------

R2009b

threshacceptbnd Function Deprecated	13-2
--	-------------

R2009a

New Demo	14-2
-----------------------	-------------

R2008b

Optimization Tool Enables Parallel Functionality	15-2
---	-------------

R2008a

Parallel Computing Toolbox Support	16-2
---	-------------

Genetic Algorithm Tool and Pattern Search Tool Combined Into Optimization Tool	16-2
---	-------------

New Optimization Tool Support for gamultiobj, simulannealbnd, and threshacceptbnd	16-2
--	-------------

New Automatic Population Generation in ga and gamultiobj	16-2
---	-------------

New Default StallTimeLimit Option = Inf in Genetic Algorithm	16-3
---	-------------

R2007b

Multiobjective Optimization with Genetic Algorithm	17-2
Multiobjective Optimization with Genetic Algorithm and Custom Data Types	17-2
Hybrid Multiobjective Optimization Combining Genetic Algorithm with Optimization Toolbox	17-2
Vectorized Function Inputs with Nonlinear Constraints ..	17-2
New Demos	17-2

R2007a

New Functions for Simulated Annealing and Threshold Acceptance	18-2
ga Output Argument exitflag Returns Numeric Value	18-2

R2006b

New Syntax for Search Method Option in Pattern Search Algorithm Improves Speed and Memory	19-2
--	-------------

Bug Fixes

Both the Genetic Algorithm and the Pattern Search Algorithm Now Accept Nonlinear Constraints	21-2
Direct Search Now Implements Two Algorithms — Generalized Pattern Search Algorithm (GPS) and Mesh Adaptive Search Algorithm (MADS)	21-2
New Options Available in the Genetic Algorithm	21-2
New Options Available in the Pattern Search Algorithm . .	21-3
New Demos	21-3

R2015b

Version: 3.3.2

Bug Fixes

Compatibility Considerations

ga measures timing using tic and toc

Two `ga` options, `TimeLimit` and `StallTimeLimit`, have changed their internal time measurement functions. Instead of using `cputime`, now both options use `tic` and `toc` to measure time.

Compatibility Considerations

The behavior of `ga` can change, because time is now measured differently.

In particular, when running `ga` in parallel, the behavior of the `TimeLimit` option is different than before. Previously, the parallel workers used most of the processing time, so the client `ga` used almost no `cputime`. Therefore, `ga` almost never exceeded the `TimeLimit` setting. Now, `ga` stops when the total elapsed time exceeds `TimeLimit`.

gatool and psearchtool have been removed

Two specialized interfaces to the Optimization app, namely `gatool` and `psearchtool`, have been removed.

Compatibility Considerations

Use the `optimtool` command to launch the Optimization app. Alternatively, click the Optimization app in the Apps tab. Choose **Solver: `ga` - Genetic Algorithm** or **`patternsearch` - Pattern Search**.

R2015a

Version: 3.3.1

Bug Fixes

R2014b

Version: 3.3

New Features

Bug Fixes

particleswarm solver for particle swarm optimization

The `particleswarm` function solves unconstrained or bound-constrained optimization problems using the particle swarm optimization algorithm. `particleswarm` is not available in the Optimization app. For details and examples, see Particle Swarm.

Nonlinear constraints in gamultiobj multiobjective genetic algorithm solver

`gamultiobj` now accepts nonlinear constraints. For details, see the function reference page.

Nonlinear constraint algorithm option in ga genetic algorithm solver can speed solutions

You can choose a new nonlinear constraint algorithm in `ga` by setting the `NonlinConAlgorithm` option to `'penalty'`. Frequently, the `'penalty'` algorithm is faster than the default `'auglag'` algorithm. For details, see Nonlinear Constraint Solver Algorithms.

R2014a

Version: 3.2.5

New Features

Bug Fixes

Compatibility Considerations

Genetic algorithm changes

Several default settings in the `ga` function changed for noninteger problems.

Option	Old Default	New Default
EliteCount	2	<code>floor(0.05*PopulationSize)</code>
Generations	100	<code>100*numberOfVariables</code>
PopInitRange	[0;1]	[-10;10], shifted and scaled to match any finite bounds
PopulationSize	20	50 when <code>numberOfVariables <= 5</code> , and 200 otherwise

Other changes:

- `ga` now stops when the average relative change in best fitness value over `StallGenLimit` generations is less than the `TolFun` tolerance. Previously, the stopping criterion was a *weighted* average relative change, where the weighting factor was $(1/2)^n$ for the n th prior iteration. This change usually causes `ga` to take more iterations. The new `StallTest` option, with default value `'totalChange'`, controls the stopping criterion. Set `StallTest` to `'geometricWeighted'` to recover the previous behavior.
- When there are linear constraints, the default crossover function is now `'crossoverintermediate'`. Previously, the default crossover function was `'crossoverscattered'` for all constraint types.
- The `gamultiobj` function has the same new default values as `ga` for the `PopInitRange` and `PopulationSize` options.

Compatibility Considerations

When using its default options, `ga` generally runs longer than before, and obtains different (and often better) results.

If you want `ga` to run as before, set its options to their old defaults using `gaoptimset`.

Parallel option change

The `UseParallel` option now accepts the values `true` and `false`. The option also accepts the former values `'always'` and `'never'`, and scalar values `1` and `0`.

The affected solvers are `MultiStart`, `patternsearch`, `ga`, and `gamultiobj`. Also, the local or hybrid solver `fmincon` has the same change to its `UseParallel` option.

Default `fmincon` algorithm for `GlobalSearch` and `MultiStart`

The default `fmincon` algorithm is now `'interior-point'`. Previously, the default `fmincon` algorithm for `GlobalSearch` was `'active-set'`. Similarly, the default `fmincon` algorithm for `MultiStart` was `'active-set'` when you set `fmincon` as the local solver.

Compatibility Considerations

When you do not set an `fmincon` algorithm, `GlobalSearch` and `MultiStart` can obtain different results than before, and take different iteration steps. To reproduce their previous behavior, set the `fmincon` `Algorithm` option to `'active-set'` using `optimoptions`, and include the option using the `createOptimProblem` function. For example,

```
opts = optimoptions('fmincon','Algorithm','active-set');  
problem = createOptimProblem('fmincon','options',opts,...);
```


R2013b

Version: 3.2.4

No New Features or Changes

R2013a

Version: 3.2.3

Bug Fixes

Compatibility Considerations

Vectorized scalar `patternsearch` assumes row orientation

When `x0` is a scalar, `patternsearch` now expects your objective function to output a column vector of values. In addition, any nonlinear constraint functions need to output column vectors as well.

Compatibility Considerations

Previously, `patternsearch` assumed that scalar problems were in column form, so expected a vectorized objective function to output a row vector.

R2012b

Version: 3.2.2

Bug Fixes

Example of mixed integer programming using `ga`

There is an updated example of mixed integer programming using `ga`. View the example [here](#). To run the example at the MATLAB® command line:

```
echodemo steppedCantileverExample
```

This example replaces a similar example, `weldedBeamDemo`.

R2012a

Version: 3.2.1

No New Features or Changes

R2011b

Version: 3.2

New Features

Compatibility Considerations

Mixed Integer Nonlinear Programming in Genetic Algorithm Solver

The `ga` function now allows you to specify that certain variables are integer valued. When you include integer constraints, you can have any objective function, bounds, and inequality constraints, but you cannot directly include equality constraints. To try to circumvent this limitation, see [No Equality Constraints](#).

For details on mixed-integer programming, see the [ga](#) function reference page or [Mixed Integer Optimization](#).

New Demo

There is a new demo of `ga` for mixed integer programming. Run the demo at the MATLAB command line by entering `echodemo weldedBeamDemo`.

Conversion of Error and Warning Message Identifiers

For R2011b, error and warning message identifiers have changed in Global Optimization Toolbox.

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 `'globaloptim:EQNSOLV:sparseToFull'` identifier has changed to `'globaloptim:eqnsolv:eqSparseToFull'`. If your code checks for `'globaloptim:EQNSOLV:sparseToFull'`, you must update it to check for `'globaloptim:eqnsolv:eqSparseToFull'` 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;
```

Tip 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.

R2011a

Version: 3.1.1

New Features

Bug Fixes

“History to New Window” Output Functions Removed

The `patternsearch` and `ga` **History to new window** output functions (`@psoutputhistory` and `@gaoutputgen`) have been removed. Obtain the same functionality by setting one of the following:

- **Display** option to `'iter'` with `psoptimset` or `gaoptimset`
- **Level of display** to `iterative` in the **Display to command window** part of the Optimization Tool **Options** pane

R2010b

Version: 3.1

New Features

Bug Fixes

Compatibility Considerations

Output Functions and Plot Functions for GlobalSearch and MultiStart

Use output functions or plot functions with GlobalSearch and MultiStart to report and plot information on algorithm progress during runs. You can also stop the solvers according to criteria you set. For more information, see Output Functions for GlobalSearch and MultiStart and Plot Functions for GlobalSearch and MultiStart.

Compatibility Considerations

In order to make exit flags have more uniform meaning across solvers, two GlobalSearch and MultiStart exit flags have different meanings than in R2010a:

Exit Flag	Meaning
-1	GlobalSearch or MultiStart stopped by an output function or plot function (regardless of local solver exit flag)
-8	No solution found; all local solver runs had exit flag -1 or smaller

Demo Removed

The demo titled “Using the Genetic Algorithm with the Parallel Computing Toolbox™” was removed from the toolbox. The demo used more complex parallelization techniques than those in the Optimization Toolbox™ demo titled “Minimizing an Expensive Optimization Problem Using Parallel Computing Toolbox™.”

R2010a

Version: 3.0

New Features

Bug Fixes

Compatibility Considerations

Toolbox Renamed and Expanded

Former Genetic Algorithm and Direct Search Toolbox™ functions are now part of Global Optimization Toolbox software.

Compatibility Considerations

Error and warning IDs now use the `globaloptim` name instead of the `gads` name. For example, to turn off the `sahybrid:unconstrainedHybridFcn` warning, instead of

```
warning('off','gads:sahybrid:unconstrainedHybridFcn')
```

use the statement

```
warning('off','globaloptim:sahybrid:unconstrainedHybridFcn')
```

New GlobalSearch and MultiStart Solver Objects

`GlobalSearch` and `MultiStart` run a local solver (such as `fmincon`) from a variety of start points. The goal is to find a global minimum, or multiple local minima. The chief differences between the solver objects are:

- `GlobalSearch` uses a scatter-search mechanism for generating start points. `MultiStart` uses uniformly distributed start points within bounds, or user-supplied start points.
- `GlobalSearch` analyzes start points and rejects those that are unlikely to improve the best local minimum found so far. `MultiStart` runs all start points.
- `MultiStart` gives a choice of local solver: `fmincon`, `fminunc`, `lsqcurvefit`, or `lsqnonlin`. `GlobalSearch` uses `fmincon`.
- `MultiStart` can be run in parallel, distributing start points to multiple processors. `GlobalSearch` does not run in parallel.

These solver objects come with a variety of new objects, functions, and methods:

- `createOptimProblem` — Function for creating optimization problem structure
- `CustomStartPointSet` and `RandomStartPointSet` — Objects for `MultiStart` multiple start points
- `GlobalOptimSolution` — Object for holding results of multiple runs of local solver

-
- `list` — Method for obtaining start points from a `CustomStartPointSet` or `RandomStartPointSet`
 - `run` — Method for running `GlobalSearch` or `MultiStart` objects with optimization problem structures

For more information, see [Using GlobalSearch and MultiStart in the Global Optimization Toolbox User's Guide](#).

New `patternsearch` Poll Method

A new poll method generates search directions faster and more reliably in `patternsearch` for linearly constrained problems. Use this poll method at the command line by setting the `PollMethod` option to `'GSSPositiveBasis2N'` or `'GSSPositiveBasisNp1'` with `psoptimset`. With the Optimization Tool, set **Options > Poll > Poll method** to **GSS Positive basis 2N** or **GSS Positive basis Np1**.

For more information, see [Poll Options in the Global Optimization Toolbox User's Guide](#).

New Demo

There is a new demo showing how to use `GlobalSearch` and `MultiStart` to find a global optimum or several local optima. Run the demo at the MATLAB command line by entering `echodemo opticalInterferenceDemo`.

`threshacceptbnd` Function Removed

The `threshacceptbnd` function has been removed.

Compatibility Considerations

Use `simulannealbnd` for similar functionality. To obtain results using a threshold acceptance algorithm, write a custom acceptance function for `simulannealbnd`—see `AcceptanceFcn` in Algorithm Settings.

R2009b

Version: 2.4.2

New Features

Compatibility Considerations

threshacceptbnd Function Deprecated

The `threshacceptbnd` function will be removed in a future release.

Compatibility Considerations

The `threshacceptbnd` function now warns that it will be removed in a future release. Use `simulannealbnd` for similar functionality. To obtain results using a threshold acceptance algorithm, write a custom acceptance function for `simulannealbnd`—see `AcceptanceFcn` in Algorithm Settings.

R2009a

Version: 2.4.1

New Features

Bug Fixes

New Demo

There is a new demo showing graphically how patternsearch works. To see the demo, enter `echodemo mtwashdemo` at the MATLAB command line.

R2008b

Version: 2.4

New Features

Optimization Tool Enables Parallel Functionality

You can now access built-in parallel functionality in Optimization Tool for relevant Genetic Algorithm and Direct Search Toolbox solvers. The option is available when you have a license for Parallel Computing Toolbox functions.

R2008a

Version: 2.3

New Features

Compatibility Considerations

Parallel Computing Toolbox Support

The functions `ga`, `gamultiobj`, and `patternsearch` can take advantage of parallel computing. Furthermore, applicable hybrid functions can use parallel computing. For more information, see the Parallel Processing chapter in the User's Guide.

Genetic Algorithm Tool and Pattern Search Tool Combined Into Optimization Tool

The Genetic Algorithm Tool and Pattern Search Tool GUIs have been combined into the Optimization Toolbox Optimization Tool GUI. To access these GUIs, enter `optimtool` at the command line and choose the appropriate solver.

Compatibility Considerations

The functions `gatool` and `psearchtool` continue to work, calling `optimtool` with the appropriate solver selected (`ga` or `patternsearch`). However, the functions `gatool` and `psearchtool` are no longer listed in the documentation.

New Optimization Tool Support for `gamultiobj`, `simulannealbnd`, and `threshacceptbnd`

The Optimization Tool GUI now includes the functions `gamultiobj`, `simulannealbnd`, and `threshacceptbnd`. Therefore, all Genetic Algorithm and Direct Search Toolbox solvers are supported in Optimization Tool. To access these GUIs, enter `optimtool` at the command line and choose the appropriate solver.

New Automatic Population Generation in `ga` and `gamultiobj`

`ga` and `gamultiobj` can now create populations satisfying bounds and linear constraints, with well-dispersed populations, using the function `gacreationlinearfeasible`.

Compatibility Considerations

The previous creation function, `gacreationuniform`, is accessible by using `gaoptimset` to set `CreationFcn` to `@gacreationuniform`. The new default behavior is to use `gacreationlinearfeasible` when there are linear constraints, and `gacreationuniform` when there are bounds or no constraints.

New Default StallTimeLimit Option = Inf in Genetic Algorithm

The default value of `StallTimeLimit` in `ga` used to be 20. It was changed to `Inf` in order to avoid time-outs when using computationally intensive fitness functions.

Compatibility Considerations

Change `StallTimeLimit` to 20 using `gaoptimset` to get the previous behavior.

R2007b

Version: 2.2

New Features

Bug Fixes

Multiobjective Optimization with Genetic Algorithm

Multiobjective optimization, with linear and bound constraints, is now available through the new function `gamultiobj`. This function determines optimal Pareto fronts from specified criteria, including Pareto fronts that are nonconvex, disconnected, or both.

Optimization Toolbox also contains multiobjective functionality, but cannot reliably generate optimal Pareto fronts if these are nonconvex or disconnected.

Two new demos illustrate this feature. See “New Demos” on page 17-2.

Multiobjective Optimization with Genetic Algorithm and Custom Data Types

The new function `gamultiobj` also supports multiobjective optimization with custom data types, including binary.

Hybrid Multiobjective Optimization Combining Genetic Algorithm with Optimization Toolbox

To determine multiobjective optimizations more accurately, you can now combine the new function `gamultiobj` with the existing function `fgoalattain` from Optimization Toolbox.

Vectorized Function Inputs with Nonlinear Constraints

The functions `ga` and `patternsearch` now accept vectorized function inputs with nonlinear constraints. The new function `gamultiobj` does as well.

New Demos

Two accompanying demos illustrate the use of the new multiobjective genetic algorithm function `gamultiobj`:

- `gamultiobjfitness` uses `gamultiobj` to solve a simple problem with one decision variable and two objectives.
- `gamultiobjoptionsdemo` shows how to set options for multiobjective optimization with a simple genetic algorithm problem.

R2007a

Version: 2.1

New Features

Bug Fixes

Compatibility Considerations

New Functions for Simulated Annealing and Threshold Acceptance

The following functions have been added for simulated annealing and threshold acceptance:

<code>simulannealbnd</code>	Perform unconstrained or bound-constrained minimization of a function of several variables using simulated annealing. The default algorithm uses adaptive annealing, but options can be changed to use Boltzmann annealing, fast annealing, and other variants.
<code>threshacceptbnd</code>	Perform unconstrained or bound-constrained minimization of a function of several variables using threshold acceptance.
<code>saoptimset</code>	Create or modify optimization options for <code>simulannealbnd</code> or <code>threshacceptbnd</code> .
<code>saoptimget</code>	Access options for <code>simulannealbnd</code> or <code>threshacceptbnd</code> .

If you are viewing this documentation in the Help browser, the following demos are available:

- [Minimization Using Simulated Annealing And Threshold Acceptance Algorithms](#)
- [Simulated Annealing and Threshold Acceptance Options](#)
- [Custom Data Type Optimization Using Simulated Annealing](#)

ga Output Argument `exitflag` Returns Numeric Value

The third output argument returned by the `ga` function is now a numeric value. This change is consistent with other optimization solvers in MATLAB and makes it easier to programmatically determine the reason the solver stopped. As in previous versions, the fourth output argument is a structure with the field `message` containing a string that indicates the reason the solver stopped.

The new syntax is as follows:

```
[x,fval,exitflag,output] = ga(fitnessfcn, ...)
```

For more information, including a description of the messages that correspond to the numeric values for each `exitflag` value, see the `ga` function reference page in the Genetic Algorithm and Direct Search Toolbox User's Guide for more information.

Compatibility Considerations

In previous versions, the third output argument returned by `ga` is a string describing the reason the solver stopped.

```
[x,fval,reason] = ga(fitnessfcn, ...)
```

If you used the third output argument of the `ga` function programmatically in a previous release, for example, to compare the value to a string, this code will now produce an error.

R2006b

Version: 2.0.2

New Features

Bug Fixes

Compatibility Considerations

New Syntax for Search Method Option in Pattern Search Algorithm Improves Speed and Memory

The new syntax is more efficient both with speed and memory. This is done by changing the way linear and bound constraints are stored and passed to a search function. The following describes the new calling syntax:

```
function [successSearch,xBest,fBest,funccount] =  
searchfcnTemplate(fun,x,A,b,Aeq,beq,lb,ub, ...  
    optimValues,options)
```

For more information on how to use the new search function syntax, see Structure of the Search Function in the Genetic Algorithm and Direct Search Toolbox User's Guide. To see a template that you can view and edit, type

```
edit searchfcnTemplate  
at the Command Window.
```

Compatibility Considerations

In previous versions, a search function required the following calling syntax:

```
function [successSearch,nextIterate,optimState] =  
searchfcnTemplate(fun,iterate,tol,A,L,U, ...  
    funeval,maxfun,searchoptions,objfcnarg, ...  
    iterlimit,factors)
```

If you have a search function written for use in a previous release, the function performs correctly in Version 2.0.2 but returns a warning. Custom search functions written in a previous version need to be updated with the new syntax. In later versions, this syntax may cause a warning or error.

The `searchConversion` utility function is provided to convert your search functions from previous releases to the new syntax of Version 2.0.2. For more information on obtaining and using the conversion function, see this technical support solution.

R2006a

Version: 2.0.1

Bug Fixes

R14SP3

Version: 2.0

New Features

Bug Fixes

Both the Genetic Algorithm and the Pattern Search Algorithm Now Accept Nonlinear Constraints

Previously, the genetic algorithm solver only solved unconstrained optimization problems, and the pattern search solver solved unconstrained optimization problems as well as those with linear constraints and bounds. Now, both solvers have the ability to solve general nonlinear optimization problems with linear constraints, bounds, and nonlinear constraints by accepting a nonlinear constraint function. The M-file for the nonlinear constraint function is accepted as an input argument at the command line for both the `ga` and `patternsearch` functions, as well as in the **Constraints** panel of `psearchtool` and `gatool`.

Direct Search Now Implements Two Algorithms – Generalized Pattern Search Algorithm (GPS) and Mesh Adaptive Search Algorithm (MADS)

The GPS algorithm is the pattern search algorithm implemented in previous versions of the toolbox. The MADS algorithm is a modification of the GPS algorithm. The algorithms differ in how the set of points forming the mesh is computed. The GPS algorithm uses fixed direction vectors, whereas the new MADS algorithm uses a random selection of vectors to define the mesh.

New Options Available in the Genetic Algorithm

The following options are available in the `gatool` and when using the `ga` function at the command prompt:

- The new **Constraints** panel has a **Nonlinear constraint function** field in addition to fields for linear constraints and bounds for solving constrained optimization problems
- New **Max constraint** (`@gaplotmaxconstr`) option in the **Plot** pane to plot the maximum nonlinear constraint violation at each generation
- New crossover function, **Arithmetic** (`@crossoverarithmetic`), available in the **Crossover** panel that creates children that are the weighted arithmetic mean of two parents
- New mutation function, **Adaptive Feasible** (`mutationadaptfeasible`), available in the **Crossover** panel that randomly generates directions that are adaptive with respect to the last successful or unsuccessful generation. This function is the default for constrained problems

-
- New **Algorithm settings** panel for selecting algorithm specific parameters, such as the penalty parameters, **Initial penalty** and **Penalty factor**, for a nonlinear constraint algorithm
 - New **Hybrid function**, `fmincon`, for constrained problems
 - New **Nonlinear constraint tolerance** parameter in **Stopping criteria**

New Options Available in the Pattern Search Algorithm

The following options are available in the `psearchtool` and when using the `patternsearch` function at the command prompt:

- **Constraints** now has a **Nonlinear constraint function** option to solve for constrained optimization problems
- New **Max constraint** (`@psplotmaxconstr`) option in the **Plot** pane to plot the maximum nonlinear constraint violation at each generation
- Updated **Poll method** and **Search method** options for selecting the GPS or MADS algorithms
- New **Algorithm settings** panel for selecting algorithm specific parameters, such as the penalty parameters, **Initial penalty** and **Penalty factor**, for a nonlinear constraint algorithm
- New **Time limit** and **Nonlinear constraint tolerance** parameters in **Stopping criteria**

New Demos

The Genetic Algorithm and Direct Search Toolbox contains the following new demos for Version 2.0:

- Optimization of Non-smooth Objective Function
- Constrained Minimization Using the Genetic Algorithm
- Constrained Minimization Using the Pattern Search
- Optimization of Stochastic Objective Function
- Using the Genetic Algorithm and Direct Search Toolbox

