

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.
1 Apple Hill Drive
Natick, MA 01760-2098

Global Optimization Toolbox Release Notes

© COPYRIGHT 2005–2022 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.

R2022a

Multiobjective Global Optimization Toolbox Solvers in Problem-Based Workflow: Solve multiobjective problems using optimization variables	1-2
Multiple-Start Global Optimization Toolbox Solvers in Problem-Based Workflow: Search for a global optimum using multiple local solves . . .	1-2
Problem-Based Optimize Live Editor Task: Solve optimization problems using a visual interface	1-2
Functionality being removed or changed	1-3
Updated direction generators in ga, gamultiobj, paretosearch, patternsearch, and surrogateopt	1-3
surrogateopt creates adaptive points differently for bound-constrained problems	1-3
surrogateopt MinSampleDistance default change	1-3
paretosearch default poll method change	1-3

R2021b

Global Optimization Toolbox Solvers in Problem-Based Workflow: Solve nonsmooth or global optimization problems using optimization variables	2-2
Integer Constraints in gamultiobj: Solve multiobjective problems with integer, bound, linear, and nonlinear inequality constraints	2-2
Integer Genetic Algorithm: Solve problems with linear equality constraints or nondefault operators	2-2
Multiobjective Plots: Create 3-D gaplotpareto plots, and view Pareto plot tips with an index for finding control variables	2-3
Functionality being removed or changed	2-3
surrogateopt measures infeasibilities differently	2-3
surrogateopt with integer constraints iterates change	2-3
Genetic algorithm iterative display gives more detail	2-3

R2021a

Linear Constraints in surrogateopt: Solve problems with linear constraints, integer constraints, and nonlinear inequality constraints	3-2
Vectorized surrogateopt: Customize parallel optimization	3-2
Functionality Being Removed or Changed	3-2
Optimization App Removed	3-2

R2020b

Optimize Live Editor Task: Create and run optimization problems using a visual interface	4-2
Parallel Computation: Solve problems faster using ga, gamultiobj, particleswarm, and patternsearch when objective or constraint functions use large amounts of data	4-2
Hybrid Functions for ga and particleswarm: Obtain hybrid function exit flag	4-2
Functionality Being Removed or Changed	4-2
Solvers Warn on Parallel Error	4-2

R2020a

surrogateopt Nonlinear Constraints: Solve problems with nonlinear inequality constraints and time-consuming objective function	5-2
Functionality Being Removed or Changed	5-2
surrogateopt default plot function changes	5-2

R2019b

Integer Constraints in Surrogate Optimization: Solve mixed-integer bound-constrained problems with time-consuming objective functions	6-2
--	------------

Functionality Being Removed or Changed	6-2
ga Performs Fewer Fitness Function Evaluations	6-2
surrogateopt Handles Fixed Variables	6-2
Simplified Parallel Deployment	6-2

R2019a

Surrogate Optimization Checkpoints: Store the solver state at each iteration, and resume optimization from a stored state	7-2
Functionality Being Removed or Changed	7-2
Parallel patternsearch automatically sets complete poll	7-2

R2018b

surrogateopt Solver: Solve time-consuming, bound-constrained optimization problems using fewer objective function evaluations ...	8-2
paretosearch Multiobjective Solver: Find Pareto sets quickly and accurately for problems with bound, linear, or smooth nonlinear constraints	8-2
Parallel Computation: Accelerate surrogateopt and paretosearch functions (using Parallel Computing Toolbox)	8-2
Functionality Being Removed or Changed	8-2
gaoptimset, psoptimset, and saoptimset are not recommended	8-2
gaoptimget, psoptimget, and saoptimget are not recommended	8-2
patternsearch Search Function: @searchga inherits evaluation options from patternsearch	8-3

R2018a

Automatic Code Suggestions and Completions: Specify options and arguments by making selections from a list	9-2
---	------------

R2017b

Bug Fixes

R2017a

String Arguments: Solvers accept strings	11-2
gamultiobj: Performance improvement	11-2
gamultiobj Plot Function: Maximum nonlinear constraint violation plot using @gaplotmaxconstr	11-2
ga, gamultiobj, and simulannealrnd: Updated rngstate field in output structure	11-2

R2016b

Linear Constraints in ga and patternsearch: Different algorithm	12-2
--	------

R2016a

optimoptions function: Set solver options and get a comprehensive display of your settings	13-2
Renamed Options: Use more expressive and consistent names for options	13-2
Hybrid functions no longer switch automatically to a compatible solver	13-2

R2015b

ga measures timing using tic and toc	14-2
gatool and psearchtool have been removed	14-2

R2015a

Bug Fixes

R2014b

particleswarm solver for particle swarm optimization	16-2
Nonlinear constraints in gamultiobj multiobjective genetic algorithm solver	16-2
Nonlinear constraint algorithm option in ga genetic algorithm solver can speed solutions	16-2

R2014a

Genetic algorithm changes	17-2
Parallel option change	17-2
Default fmincon algorithm for GlobalSearch and MultiStart	17-2

R2013b

No New Features or Changes

R2013a

Vectorized scalar patternsearch assumes row orientation	19-2
--	-------------

R2012b

Example of mixed integer programming using ga	20-2
--	-------------

R2012a

No New Features or Changes

R2011b

Mixed Integer Nonlinear Programming in Genetic Algorithm Solver . .	22-2
New Demo	22-2
Conversion of Error and Warning Message Identifiers	22-2

R2011a

“History to New Window” Output Functions Removed	23-2
---	-------------

R2010b

Output Functions and Plot Functions for GlobalSearch and MultiStart	24-2
Demo Removed	24-2

R2010a

Toolbox Renamed and Expanded	25-2
New GlobalSearch and MultiStart Solver Objects	25-2
New patternsearch Poll Method	25-2
New Demo	25-3
threshacceptbnd Function Removed	25-3

R2009b

threshacceptbnd Function Deprecated	26-2
--	-------------

R2009a

New Demo	27-2
-----------------------	-------------

R2008b

Optimization Tool Enables Parallel Functionality	28-2
---	-------------

R2008a

Parallel Computing Toolbox Support	29-2
Genetic Algorithm Tool and Pattern Search Tool Combined Into Optimization Tool	29-2
New Optimization Tool Support for gamultiobj, simulannealbnd, and threshacceptbnd	29-2
New Automatic Population Generation in ga and gamultiobj	29-2
New Default StallTimeLimit Option = Inf in Genetic Algorithm	29-2

R2007b

Multiobjective Optimization with Genetic Algorithm	30-2
Multiobjective Optimization with Genetic Algorithm and Custom Data Types	30-2
Hybrid Multiobjective Optimization Combining Genetic Algorithm with Optimization Toolbox	30-2

Vectorized Function Inputs with Nonlinear Constraints	30-2
New Demos	30-2

R2007a

New Functions for Simulated Annealing and Threshold Acceptance ...	31-2
ga Output Argument exitflag Returns Numeric Value	31-2

R2006b

New Syntax for Search Method Option in Pattern Search Algorithm Improves Speed and Memory	32-2
--	-------------

R2006a

Bug Fixes

R14SP3

Both the Genetic Algorithm and the Pattern Search Algorithm Now Accept Nonlinear Constraints	34-2
Direct Search Now Implements Two Algorithms — Generalized Pattern Search Algorithm (GPS) and Mesh Adaptive Search Algorithm (MADS)	34-2
New Options Available in the Genetic Algorithm	34-2
New Options Available in the Pattern Search Algorithm	34-2
New Demos	34-3

R2022a

Version: 4.7

New Features

Bug Fixes

Version History

Multiobjective Global Optimization Toolbox Solvers in Problem-Based Workflow: Solve multiobjective problems using optimization variables

The problem-based functions `solve` and `prob2struct` now allow you to solve multiobjective problems using the `gamultiobj` and `paretosearch` solvers. By default, `solve` and `prob2struct` call `gamultiobj` to solve multiobjective problems.

To hold multiobjective results, `solve` returns the solution as an `OptimizationValues` object.

To plot multiobjective results, use `paretoplot` on the returned `OptimizationValues` object. For an example, see the function reference page or “Pareto Front for Multiobjective Optimization, Problem-Based”.

To provide initial points for multiobjective solvers and for other solvers that take multiple start points (`ga`, `MultiStart`, `particleswarm`, and `surrogateopt`), specify the start points in the `x0` argument as an `OptimizationValues` object using the `optimvalues` function. For examples, see “Specify Start Points for `MultiStart`, Problem-Based” and “Initial Points for Global Optimization Toolbox Solvers”.

For examples using the new workflow, see the “Multiobjective Optimization” category. In particular, the example “Plan Nuclear Fuel Disposal Using Multiobjective Optimization” shows a nonlinear mixed-integer problem with hundreds of optimization variables and three objective functions.

Multiple-Start Global Optimization Toolbox Solvers in Problem-Based Workflow: Search for a global optimum using multiple local solves

The problem-based function `solve` now allows you to search for a global optimum using multiple local solves using the `GlobalSearch` and `MultiStart` solvers.

To use `GlobalSearch` or `MultiStart`, call `solve` with one of these objects as the third input argument:

```
ms = MultiStart; % Or gs = GlobalSearch
[sol,fval] = solve(prob,x0,ms)
```

To specify the minimum number of start points for `MultiStart`, set the new `MinNumStartPoints` argument.

```
[sol,fval] = solve(prob,x0,ms,MinNumStartPoints=50)
```

For examples using the new workflow, see the “Global or Multiple Starting Point Search” category.

Problem-Based Optimize Live Editor Task: Solve optimization problems using a visual interface

The **Optimize** Live Editor task now supports the “Problem-Based Optimization Workflow” for both single-objective and multiobjective optimization. You can let the task choose an appropriate solver, or you can choose the `ga`, `gamultiobj`, `paretosearch`, `particleswarm`, `patternsearch`, `simulannealbnd`, or `surrogateopt` Global Optimization Toolbox solvers. Using problem-based **Optimize** you can:

- Create optimization variables, expressions, and problems.

-
- Create linear or nonlinear objectives or constraints using built-in function templates.
 - Specify the solver and options.
 - Run the solver directly from the task.
 - Export the resulting MATLAB® code.

To launch the task, create a Live Editor window, then select **Task > Optimize** or **Insert > Task > Optimize**. Then choose the problem-based approach.

For an example, see “Get Started with Problem-Based Optimize Live Editor Task”. For tips, see “Use Problem-Based Optimize Live Editor Task Effectively”.

Functionality being removed or changed

Updated direction generators in `ga`, `gamultiobj`, `paretosearch`, `patternsearch`, and `surrogateopt`

Behavior change

Several solvers use direction generators internally to create feasible points from existing points. Linear equality constraints and integer constraints can cause difficulties for the generators. The generators were updated for reliability and to provide the widest possible range of feasible directions. Previously, multiple constraints could cause the generators to provide only a subset of the feasible directions.

There is no provision for using the previous direction generators.

`surrogateopt` creates adaptive points differently for bound-constrained problems

Behavior change

The `surrogateopt` solver creates adaptive points differently than in previous releases for bound-constrained problems. Now, `surrogateopt` uses the same algorithm for all constraint types. For algorithm details, see “Surrogate Optimization Algorithm”.

To use the previous algorithm for bound-constrained problems, set the undocumented `AdaptiveSampler` option to "random". For example,

```
options = optimoptions("surrogateopt", AdaptiveSampler="random");  
[x, fval, exitflag, output] = surrogateopt(fun, lb, ub, options)
```

`surrogateopt` `MinSampleDistance` default change

Behavior change

The `surrogateopt` `MinSampleDistance` default value is now $1e-6$. Previously, the default value was $1e-3$.

The change has two effects:

- `surrogateopt` takes more iterations between surrogate resets.
- `surrogateopt` finds a more accurate value of a local minimum.

To obtain the previous behavior, use `optimoptions` to set the `MinSampleDistance` option to $1e-3$

`paretosearch` default poll method change

Behavior change

The `paretosearch` default poll method is now `'GPSPositiveBasis2Np2'`. Previously, the default poll method was `'GSSPositiveBasis2np2'`.

Often, this change causes `paretosearch` to converge more quickly.

To obtain the previous behavior, use `optimoptions` to set the `PollMethod` option to `'GSSPositiveBasis2np2'`.

R2021b

Version: 4.6

New Features

Bug Fixes

Version History

Global Optimization Toolbox Solvers in Problem-Based Workflow: Solve nonsmooth or global optimization problems using optimization variables

The problem-based functions `solve` and `prob2struct` now allow you to choose most Global Optimization Toolbox solvers for single-objective problems. The problem-based workflow uses optimization variables and expressions to formulate the objective and constraints. For a comparison of the new problem-based workflow with the traditional solver-based workflow, see [Decide Between Problem-Based and Solver-Based Approach](#). Currently, you cannot choose `MultiStart` or `GlobalSearch` in the problem-based workflow.

These category pages now list both problem-based examples and solver-based examples:

- [Direct Search](#)
- [Genetic Algorithm](#)
- [Particle Swarm](#)
- [Surrogate Optimization](#)
- [Simulated Annealing](#)

For details of using Global Optimization Toolbox solvers in the problem-based approach, see [Problem-Based Optimization Setup](#).

Integer Constraints in `gamultiobj`: Solve multiobjective problems with integer, bound, linear, and nonlinear inequality constraints

`gamultiobj` now accepts integer constraints in addition to bounds, linear constraints, and nonlinear inequality constraints. As with `ga`, a problem cannot have both integer constraints and nonlinear equality constraints, only nonlinear inequalities. For details, see the function reference page. For an example, see [Integer `gamultiobj`](#).

Integer Genetic Algorithm: Solve problems with linear equality constraints or nondefault operators

Mixed-integer `ga` and `gamultiobj` have fewer restrictions than in previous versions. When used with integer constraints, these solvers have the following behavior:

- You can specify linear equality constraints. Previously, `ga` supported only linear inequality constraints for mixed-integer problems.
- You can specify nondefault values or provide custom functions for the options `CreationFcn`, `CrossoverFcn`, `MutationFcn`, and `SelectionFcn`. `ga` no longer overrides these options. The default values for these options are the same as in previous versions, and are now visible as editable named functions.
- When using nondefault or custom functions, the solvers ensure that the population remains feasible at all iterations with respect to integer constraints, linear constraints, and bounds. In other words, custom functions do not have to ensure feasibility. In contrast, the solvers do not ensure that intermediate iterations are feasible with respect to nonlinear inequality constraints. Also, when the problem has no integer constraints and you use nondefault functions, the solvers do not enforce that linear constraints are satisfied.

To help you understand which functions are used, the iterative display now lists the creation, crossover, mutation, and selection functions being used for `ga` and `gamultiobj`.

Version History

When a problem has both linear and integer constraints, the internal algorithms for linear feasibility can cause integer variables to be slightly different from integer values, and can cause bounds to be slightly violated. The tolerance for integer feasibility is $1e-5$. You can use the `round` function to restore exact integer feasibility. However, doing so can cause linear constraints to become slightly infeasible.

Multiobjective Plots: Create 3-D `gaplotpareto` plots, and view Pareto plot tips with an index for finding control variables

The `gaplotpareto` plot function now supports 3-D plotting. For an example, see Plot 3-D Pareto Front.

The plot functions for Pareto sets in `gamultiobj` (`gaplotpareto`) and `paretosearch` (`psplotparetof`) now include data tips with an index that enables you to find the solution variables associated with plotted function values. See Plot 3-D Pareto Front.

Functionality being removed or changed

surrogateopt measures infeasibilities differently

Behavior change

When a point is infeasible, `surrogateopt` now measures the maximum absolute value of the nonlinear constraint functions as the measure of infeasibility. This measure is the same measure used by other nonlinear solvers. Previously, `surrogateopt` counted the number of infeasible nonlinear constraint functions at the point as the primary measure to decrease, and used maximum infeasibility as a secondary criterion. In other words, previously a point would be considered better if fewer nonlinear constraint functions were violated, even if the maximum violation increased.

Because of this change, `surrogateopt` can have different iterations for some nonlinearly constrained problems. There is no provision for obtaining the previous behavior.

surrogateopt with integer constraints iterates change

Behavior change

When a problem has integer constraints, `surrogateopt` internally uses a branch-and-bound algorithm to search for improved feasible points. In previous versions, this algorithm had an internal time limit. Because timing depends on a variety of conditions for the computer, the time limit led to nonreproducible behavior for `surrogateopt` iterations.

This time limit has been removed for the current software. Instead the algorithm uses an internal iteration limit to control the time spent in the branch-and-bound algorithm. Therefore, `surrogateopt` iterations for integer-constrained problems are now reproducible, but can differ from iterations in previous software versions.

Genetic algorithm iterative display gives more detail

Behavior change

The `ga` and `gamultiobj` iterative display now contains information on the problem type and which creation, mutation, crossover, and selection functions the solver is using. Obtain the iterative display by setting the `Display` option to `"iter"` or `"diagnose"` using `optimoptions`. For an example of the expanded display, see [Constrained Minimization Using the Genetic Algorithm](#).

R2021a

Version: 4.5

New Features

Bug Fixes

Version History

Linear Constraints in surrogateopt: Solve problems with linear constraints, integer constraints, and nonlinear inequality constraints

surrogateopt now accepts linear inequality constraints and linear equality constraints in addition to the other types it already accepted: bounds, nonlinear inequality constraints, and integer constraints. For details, see the function reference page.

Vectorized surrogateopt: Customize parallel optimization

surrogateopt gains two options for speeding computations in some cases: UseVectorized and BatchUpdateInterval. These options allow you to control batch objective and constraint evaluations, enabling you to perform custom parallel optimization and to control the surrogate update interval. For details, see the example Vectorized Surrogate Optimization for Custom Parallel Simulation and the Surrogate Optimization Options topic.

Functionality Being Removed or Changed

Optimization App Removed

Errors

The Optimization app (optimtool) has been removed. For a visual interface to solvers, use the **Optimize** Live Editor task.

R2020b

Version: 4.4

New Features

Bug Fixes

Version History

Optimize Live Editor Task: Create and run optimization problems using a visual interface

The Optimize Live Editor task helps you formulate and run optimization and equation-solving problems using a visual interface. **Optimize** applies to all MATLAB optimization solvers, all Optimization Toolbox™ solvers, and, with a license, all Global Optimization Toolbox solvers, except for `fseminf`, `GlobalSearch`, and `MultiStart`. For an example, see [Constrained Minimization Using `patternsearch` and Optimize Live Editor Task](#). For details, see the [Optimize](#) reference page.

Version History

The **Optimize** Live Editor task replaces the Optimization app (`optimtool`) as a visual interface for creating and solving optimization problems. As noted since R2015b, the Optimization app will be removed in a future release.

Optimize provides only two outputs from the solver, typically called `x` and `fval` in the documentation. Specifically, **Optimize** does not provide the `exitflag` and output structure information. To obtain more information from the solver, use the generated code to rerun the solver while requesting more outputs.

Parallel Computation: Solve problems faster using `ga`, `gamultiobj`, `particleswarm`, and `patternsearch` when objective or constraint functions use large amounts of data

Internal routines for parallel computation in `ga`, `gamultiobj`, `particleswarm`, and `patternsearch` now dispatch data more efficiently. When objective or nonlinear constraint function handles contain a large amount of data, computations are noticeably faster.

Hybrid Functions for `ga` and `particleswarm`: Obtain hybrid function exit flag

When you set the `HybridFcn` option for the `ga` and `particleswarm` solvers, they return the exit flag from the hybrid function in the `output.hybridflag` field. This field encodes the reason that the hybrid function stops. For examples, see [Hybrid Scheme in the Genetic Algorithm and Tune Particle Swarm Optimization Process](#).

Functionality Being Removed or Changed

Solvers Warn on Parallel Error

Behavior change

When you pass the `'UseParallel'` option as `true` but your solver cannot open a parallel pool or connect to a parallel pool, the solver now issues a warning and then computes in serial. Previously, solver behavior varied:

- Nonlinear Optimization Toolbox solvers, `ga`, `patternsearch`, `particleswarm`, and `gamultiobj` did not issue a warning, and silently computed in serial.
- If Parallel Computing Toolbox™ was not installed, `surrogateopt` and `paretosearch` issued an error and did not compute.

-
- If Parallel Computing Toolbox was installed but could not open or connect to a parallel pool, `surrogateopt` did not warn before computing in serial, and `paretosearch` did warn before computing in serial (the current behavior).

R2020a

Version: 4.3

New Features

Bug Fixes

Version History

surrogateopt Nonlinear Constraints: Solve problems with nonlinear inequality constraints and time-consuming objective function

The `surrogateopt` solver now accepts nonlinear inequality constraints. The constraints must be written in a form that is different from the form for other solvers. For details, see [Solve Problem with Nonlinear Constraints and Convert Nonlinear Constraints Between surrogateopt Form and Other Solver Forms](#).

Functionality Being Removed or Changed

surrogateopt default plot function changes

Behavior change

The default plot function in `surrogateopt` is now `'optimplotfvalconstr'` instead of `'optimplotfval'`. When you do not have nonlinear constraints, the behavior of the new plot function is similar to the previous one. When you have nonlinear constraints, `'optimplotfvalconstr'` colors infeasible points red. Also, the new plot function changes the title of the plot figure for feasibility problems. For details, see [Plot Function](#).

To use the previous default plot function, set the `PlotFcn` option to `'optimplotfval'` using `optimoptions`.

R2019b

Version: 4.2

New Features

Bug Fixes

Version History

Integer Constraints in Surrogate Optimization: Solve mixed-integer bound-constrained problems with time-consuming objective functions

The `surrogateopt` solver now accepts integer constraints. Specify variable components that are integer-valued using the `intcon` argument. For details, see the function reference page, the Surrogate Optimization Algorithm description, or the example Mixed-Integer Surrogate Optimization.

Functionality Being Removed or Changed

ga Performs Fewer Fitness Function Evaluations

Behavior change

When the fitness function is deterministic, `ga` does not reevaluate the fitness function on elite (current best) individuals. You can override this behavior by setting the new `state.EvalElites` field to `true` in a custom output function or custom plot function. See Custom Output Function for Genetic Algorithm or Custom Plot Function.

Similarly, when `ga` creates duplicate members in the initial population, `ga` evaluates each unique member only once. You can override this behavior by setting the new `state.HaveDuplicates` field to `false` in a custom plot function or custom output function.

For details, see The State Structure.

surrogateopt Handles Fixed Variables

Behavior change

`surrogateopt` now internally removes fixed variables from an optimization. Fixed variables are variables with equal lower and upper bounds. A solver cannot optimize these variables. Previously, in order to have a nonsingular surrogate, `surrogateopt` would slightly perturb any equal bounds.

When *all* lower bound arguments are equal to the corresponding upper bound arguments, there is only one feasible point. In this case, `surrogateopt` performs no optimization and returns exit flag 10 (unless the feasible point has objective value less than `options.ObjectiveLimit`, in which case the exit flag is 1).

Simplified Parallel Deployment

Behavior change

If you deploy code that calls an optimization solver, and want the solver to use parallel computing, you no longer need to create a parallel pool explicitly in your code.

R2019a

Version: 4.1

New Features

Bug Fixes

Version History

Surrogate Optimization Checkpoints: Store the solver state at each iteration, and resume optimization from a stored state

The `CheckpointFile` option causes `surrogateopt` to maintain a file that you can use to resume an optimization from the last function evaluation. This feature enables you to recover an optimization from a crash or other stoppage. You can change some options when restarting, including plot functions and stopping criteria. For details, see [Work with Checkpoint Files and Checkpoint File](#).

Functionality Being Removed or Changed

Parallel patternsearch automatically sets complete poll

Behavior change

When you set the `patternsearch` 'UseParallel' option to true, `patternsearch` internally sets the 'UseCompletePoll' option to true, overriding any other option setting. This change prevents the previous behavior, where you set 'UseParallel' to true but `patternsearch` would compute in serial because the default value of 'UseCompletePoll' is false.

R2018b

Version: 4.0

New Features

Bug Fixes

Version History

surrogateopt Solver: Solve time-consuming, bound-constrained optimization problems using fewer objective function evaluations

To search for a global minimum of an objective function that takes a long time to evaluate, use the `surrogateopt` solver. `surrogateopt` accepts only bound constraints. For details, see [Surrogate Optimization](#).

paretosearch Multiobjective Solver: Find Pareto sets quickly and accurately for problems with bound, linear, or smooth nonlinear constraints

`paretosearch` uses a direct search algorithm to create Pareto sets for multiobjective problems. For ease of trying different solvers on multiobjective problems, the syntaxes of `paretosearch` and `gamultiobj` are the same. For details, see [Multiobjective Optimization](#).

Parallel Computation: Accelerate surrogateopt and paretosearch functions (using Parallel Computing Toolbox)

If you have Parallel Computing Toolbox, the `surrogateopt` and `paretosearch` solvers evaluate objective and nonlinear constraint functions in parallel when you set the `UseParallel` option to `true`. This setting is especially beneficial for the expensive objective functions for which `surrogateopt` is designed. For details, see [Parallel Computing](#).

Functionality Being Removed or Changed

gaoptimset, psoptimset, and saoptimset are not recommended

Still runs

For setting options, the `gaoptimset`, `psoptimset`, and `saoptimset` functions are not recommended. Instead, use `optimoptions`. The only difference between using `optimoptions` and the other functions is, for `optimoptions`, you include the solver name as the first argument. For example, to set iterative display in `ga`,

```
options = optimoptions('ga','Display','iter');  
% instead of  
options = gaoptimset('Display','iter');
```

`optimoptions` has several advantages over the other functions.

- `optimoptions` has better automatic code suggestions and completions, especially in the Live Editor.
- You can use a single option-setting function instead of a variety of functions.

There are no plans to remove `gaoptimset`, `psoptimset`, and `saoptimset` at this time.

gaoptimget, psoptimget, and saoptimget are not recommended

Still runs

For querying options, the `gaoptimget`, `psoptimget`, and `saoptimget` functions are not recommended. Instead, use dot notation. For example, to see the setting of the `Display` option in `opts`,

```
displayopt = opts.Display
% instead of
displayopt = gaoptimget(opts, 'Display')
```

Using automatic code completions, dot notation takes fewer keystrokes: `displayopt = opts.DTab`.

There are no plans to remove `gaoptimget`, `psoptimget`, and `saoptimget` at this time.

patternsearch Search Function: @searchga inherits evaluation options from patternsearch
Behavior change

By default, the `@searchga` search method of `patternsearch` now uses the same settings of the `UseParallel` and `UseVectorized` options as `patternsearch`. Previously, by default, `@searchga` used the default options for `ga`. The new behavior is consistent with typical usage of search functions.

To obtain the previous behavior, set the `@searchga` options explicitly:

```
optionsGA = optimoptions('ga');
iterlim = 1; % Recommended setting
options = optimoptions('patternsearch', 'SearchFcn', ...
    {@searchga, iterlim, optionsGA});
```


R2018a

Version: 3.4.4

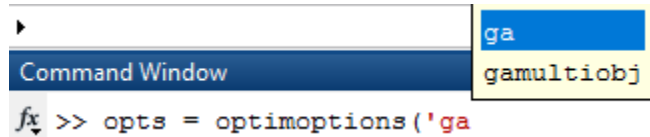
New Features

Bug Fixes

Automatic Code Suggestions and Completions: Specify options and arguments by making selections from a list

The `resetoptions` and `optimoptions` functions present you with a list of choices for nonnumeric entries. To get the list at the command line or MATLAB Editor, enter a single or double quote and some letters, and then press **Tab**.

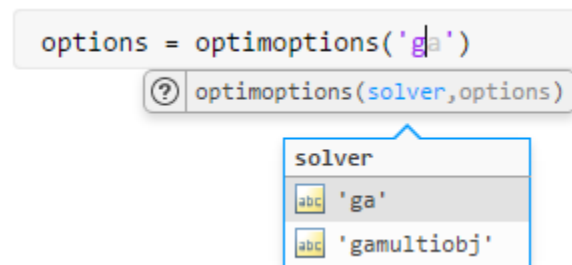
```
options = optimoptions('gTab
```



```
options = optimoptions('ga','DTab
```

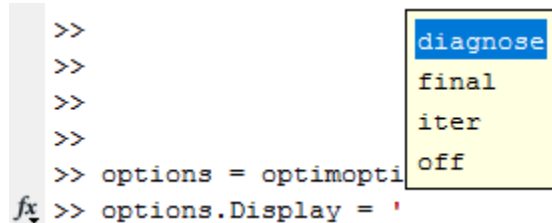
```
opts = optimoptions('ga','Display'
```

The Live Editor presents these choices without requiring you to press **Tab**.



You can also press **Tab** to display a list of options when using dot notation.

```
options = optimoptions('ga');
options.Display = 'Tab
```



R2017b

Version: 3.4.3

Bug Fixes

R2017a

Version: 3.4.2

New Features

Bug Fixes

Version History

String Arguments: Solvers accept strings

You can use a string wherever you would previously have used a character vector in setting options or passing arguments to solvers. For example,

```
solver = string('patternsearch');  
name1 = string('ScaleMesh');  
value1 = false;  
name2 = string('PollMethod');  
value2 = string('GSSPositiveBasis2N');  
options = optimoptions(solver,name1,value1,name2,value2);
```

gamultiobj: Performance improvement

gamultiobj runs faster than before due to an updated internal algorithm.

gamultiobj Plot Function: Maximum nonlinear constraint violation plot using @gaplotmaxconstr

The @gaplotmaxconstr plot function now plots the maximum nonlinear constraint violation for gamultiobj. Specify plot functions using the PlotFcn option in optimoptions or the PlotFcns option in gaoptimset.

ga, gamultiobj, and simulannealbnd: Updated rngstate field in output structure

The rngstate field in the output structure of ga, gamultiobj, and simulannealbnd is now directly compatible with the corresponding fields in the structure that rng returns.

Version History

To avoid errors, update any script that uses the output.rngstate.state or output.rngstate.type fields to use output.rngstate.State or output.rngstate.Type.

R2016b

Version: 3.4.1

Bug Fixes

Version History

Linear Constraints in ga and patternsearch: Different algorithm

When there are linear constraints in a problem, the `ga` and `patternsearch` solvers internally use `linprog` to ensure feasibility. In previous releases, the solvers internally used the `linprog` 'active-set' and 'simplex' algorithms. However, these algorithms were removed in R2016b. So the solvers now use the `linprog` 'dual-simplex' and 'interior-point' algorithms.

Version History

When your problem has linear constraints, your `ga` and `patternsearch` results can differ from previous versions. This difference is likely to be inconsequential.

R2016a

Version: 3.4

New Features

Bug Fixes

Version History

optimoptions function: Set solver options and get a comprehensive display of your settings

`optimoptions` can now set options for all solvers except `GlobalSearch` and `MultiStart`. `optimoptions` is now the recommended function for setting options.

The Optimization app now exports `optimoptions` objects instead of option structures for `ga`, `gamultiobj`, `patternsearch`, and `simulannealbnd`.

Version History

The previous functions for setting options, `gaoptimset`, `psoptimset`, and `saoptimset`, continue to work. However, the previous functions use legacy option names, while `optimoptions` uses the current names and also supports the legacy names. For details, see [Options Changes in R2016a](#).

Earlier toolbox versions do not support the new options objects for `ga`, `gamultiobj`, `patternsearch`, and `simulannealbnd`.

`optimoptions` displays only current names, so if you set a legacy name, it displays the equivalent current name. `optimoptions` no longer displays some options. For details, see [View Options](#).

Options exported from the Optimization app use the new option names.

Renamed Options: Use more expressive and consistent names for options

Some options have different names than before. The renaming gives a more consistent set of option names that match those in Optimization Toolbox.

Similarly, some `GlobalSearch` and `MultiStart` property names changed, and the `DimStartPoints` property of a `CustomStartPointSet` object is now named `StartPointsDimension`.

Version History

Previously-created options continue to work in all solvers. For details on which names changed, see [Options Changes in R2016a](#). For changes in Optimization Toolbox solvers, see [Current and Legacy Option Name Tables](#).

Hybrid functions no longer switch automatically to a compatible solver

Hybrid functions for `ga`, `particleswarm`, and `simulannealbnd` check that their optional hybrid function is compatible with the problem constraints. If it is not, the solvers immediately throw an error. For example, if there are bound constraints, solvers do not run `fminunc` as a hybrid function, because `fminunc` does not accept constraints.

Version History

Previously, a solver switched its hybrid function automatically to one compatible with the problem constraints. Now, such an incompatible hybrid function causes the solver not to run. To work around

this behavior, change any script that calls an incompatible hybrid function to call a compatible hybrid function.

R2015b

Version: 3.3.2

Bug Fixes

Version History

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.

Version History

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.

Version History

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

Version History

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</code> \leq 5, 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.

Version History

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.

Version History

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

Version History

Vectorized scalar patternsearch assumes row orientation

When x_0 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.

Version History

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

Version History

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.

Version History

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

Version History

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](#).

Version History

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	<code>GlobalSearch</code> or <code>MultiStart</code> 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

Version History

Toolbox Renamed and Expanded

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

Version History

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.

Version History

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

Version History

threshacceptbnd Function Deprecated

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

Version History

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

Version History

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.

Version History

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`.

Version History

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.

Version History

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 30-2.

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

Version History

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.

Version History

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

Version History

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.

Version History

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

