Optimization Options Reference

On this page…
Options Structure Output Function Plot Functions

Options Structure

The following table describes fields in the optimization options structure `options`. You can set values of these fields using the function `optimset`. The column labeled L, M, B indicates whether the option applies to large-scale methods, medium scale methods, or both:

- L — Large-scale methods only
- M — Medium-scale methods only
- B — Both large- and medium-scale methods
- I — Interior-point method only

See the individual function reference pages for information about available option values and defaults.

The default values for the options vary depending on which optimization function you call with `options` as an input argument. You can determine the default option values for any of the optimization functions by entering `optimset` followed by the name of the function. For example,

```
optimset fmincon
```

returns a list of the options and the default values for `fmincon`. Options with default values listed as `[]` are either not used by the function, or have different default values depending on the algorithms the solver uses.

Optimization Options

Option Name	Description	L, M, B, I	Used by Functions
`Algorithm`	Chooses the algorithm used by the solver.	B, I	`fmincon`, `fsolve`, `lsqcurvefit`, `lsqnonlin`, `quadprog`
`AlwaysHonorConstraints`	The default `'bounds'` ensures that bound constraints are satisfied at every iteration. Turn off by setting to `'none'`.	M, I	`fmincon`
`BranchStrategy`	Strategy `bintprog` uses to select branch variable.	M	`bintprog`

`DerivativeCheck`	Compare user-supplied analytic derivatives (gradients or Jacobian, depending on the selected solver) to finite differencing derivatives.	B, I	`fgoalattain`, `fmincon`, `fminimax`, `fminunc`, `fseminf`, `fsolve`, `lsqcurvefit`, `lsqnonlin`
`Diagnostics`	Display diagnostic information about the function to be minimized or solved.	B	All but `fminbnd`, `fminsearch`, `fzero`, and `lsqnonneg`
`DiffMaxChange`	Maximum change in variables for finite differencing.	B, I	`fgoalattain`, `fmincon`, `fminimax`, `fminunc`, `fseminf`, `fsolve`, `lsqcurvefit`, `lsqnonlin`
`DiffMinChange`	Minimum change in variables for finite differencing.	B, I	`fgoalattain`, `fmincon`, `fminimax`, `fminunc`, `fseminf`, `fsolve`, `lsqcurvefit`, `lsqnonlin`
`Display`	Level of display. • `'off'` displays no output. • `'iter'` displays output at each iteration, and gives the default exit message. • `'iter-detailed'` displays output at each iteration, and gives the technical exit message. • `'notify'` displays output only if the function does not converge, and gives the default exit message. • `'notify-detailed'` displays output only if the function does not converge, and gives the technical exit message. • `'final'` displays just the final output, and gives the default exit message. • `'final-detailed'` displays just the final output, and gives the technical exit message.	B, I	All. See the individual function reference pages for the values that apply.

FinDiffRelStep	Scalar or vector step size factor. When you set FinDiffRelStep to a vector v, forward finite differences delta are
	`delta = v.*sign(x).*max(abs(x),TypicalX);`
	and central finite differences are
	`delta = v.*max(abs(x),TypicalX);`
	Scalar FinDiffRelStep expands to a vector. The default is sqrt(eps) for forward finite differences, and eps^(1/3) for central finite differences.
		B, I	fgoalattain, fmincon, fminimax, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin

FinDiffType	Finite differences, used to estimate gradients, are either 'forward' (the default) , or 'central' (centered), which takes twice as many function evaluations but should be more accurate. 'central' differences might violate bounds during their evaluation in fmincon interior-point evaluations if the AlwaysHonorConstraints option is set to 'none'.	B, I	fgoalattain, fmincon, fminimax, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin
FunValCheck	Check whether objective function and constraints values are valid. 'on' displays an error when the objective function or constraints return a value that is complex, NaN, or Inf. **Note** FunValCheck does not return an error for Inf when used with fminbnd, fminsearch, or fzero, which handle Inf appropriately. 'off' displays no error.	B, I	fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, fzero, lsqcurvefit, lsqnonlin
GoalsExactAchieve	Specify the number of objectives required for the objective fun to equal the goal goal. Objectives should be partitioned into the first few elements of F.	M	fgoalattain
GradConstr	User-defined gradients for the nonlinear constraints.	M, I	fgoalattain, fmincon, fminimax
GradObj	User-defined gradients for the objective functions.	B, I	fgoalattain, fmincon, fminimax, fminunc, fseminf
HessFcn	Function handle to a user-supplied Hessian (see Hessian).	I	fmincon

Hessian	If 'user-supplied', function uses user-defined Hessian or Hessian information (when using HessMult), for the objective function. If 'off', function approximates the Hessian using finite differences.	L, I	fmincon, fminunc
HessMult	Handle to a user-supplied Hessian multiply function. For fmincon, ignored unless Hessian is 'user-supplied' or 'on'.	L, I	fmincon, fminunc, quadprog
HessPattern	Sparsity pattern of the Hessian for finite differencing. The size of the matrix is n-by-n, where n is the number of elements in x0, the starting point.	L	fmincon, fminunc
HessUpdate	Quasi-Newton updating scheme.	M	fminunc
InitBarrierParam	Initial barrier value.	I	fmincon
InitialHessMatrix	Initial quasi-Newton matrix.	M	fminunc
InitialHessType	Initial quasi-Newton matrix type.	M	fminunc
InitTrustRegionRadius	Initial radius of the trust region.	I	fmincon
Jacobian	If 'on', function uses user-defined Jacobian or Jacobian information (when using JacobMult), for the objective function. If 'off', function approximates the Jacobian using finite differences.	B	fsolve, lsqcurvefit, lsqnonlin
JacobMult	User-defined Jacobian multiply function. Ignored unless Jacobian is 'on' for fsolve, lsqcurvefit, and lsqnonlin.	L	fsolve, lsqcurvefit, lsqlin, lsqnonlin
JacobPattern	Sparsity pattern of the Jacobian for finite differencing. The size of the matrix is m-by-n, where m is the number of values in the first argument returned by the user-specified function fun, and n is the number of elements in x0, the starting point.	L	fsolve, lsqcurvefit, lsqnonlin
LargeScale	Use large-scale algorithm if possible.	B	fminunc, fsolve, linprog, lsqcurvefit, lsqlin, lsqnonlin
MaxFunEvals	Maximum number of function evaluations allowed.	B, I	fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin

MaxIter	Maximum number of iterations allowed.	B, I	All but fzero and lsqnonneg
MaxNodes	Maximum number of possible solutions, or *nodes*, the binary integer programming function bintprog searches.	M	bintprog
MaxPCGIter	Maximum number of iterations of preconditioned conjugate gradients method allowed.	L	fmincon, fminunc, fsolve, lsqcurvefit, lsqlin, lsqnonlin, quadprog
MaxProjCGIter	A tolerance for the number of projected conjugate gradient iterations; this is an inner iteration, not the number of iterations of the algorithm.	I	fmincon
MaxRLPIter	Maximum number of iterations of linear programming relaxation method allowed.	M	bintprog
MaxSQPIter	Maximum number of iterations of sequential quadratic programming method allowed.	M	fgoalattain, fmincon, fminimax
MaxTime	Maximum amount of time in seconds allowed for the algorithm.	M	bintprog
MeritFunction	Use goal attainment/minimax merit function (multiobjective) vs. fmincon (single objective).	M	fgoalattain, fminimax
MinAbsMax	Number of $F(x)$ to minimize the worst case absolute values.	M	fminimax
NodeDisplayInterval	Node display interval for bintprog.	M	bintprog
NodeSearchStrategy	Search strategy that bintprog uses.	M	bintprog
ObjectiveLimit	If the objective function value goes below ObjectiveLimit and the iterate is feasible, then the iterations halt.	M, I	fmincon, fminunc, quadprog
OutputFcn	Specify one or more user-defined functions that the optimization function calls at each iteration. See Output Function.	B, I	fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, fzero, lsqcurvefit, lsqnonlin

PlotFcns	Plots various measures of progress while the algorithm executes, select from predefined plots or write your own. • @optimplotx plots the current point • @optimplotfunccount plots the function count • @optimplotfval plots the function value • @optimplotconstrviolation plots the maximum constraint violation • @optimplotresnorm plots the norm of the residuals • @optimplotfirstorderopt plots the first-order of optimality • @optimplotstepsize plots the step size See Plot Functions.	B, I	fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, fzero, lsqcurvefit, and lsqnonlin. See the individual function reference pages for the values that apply.
PrecondBandWidth	Upper bandwidth of preconditioner for PCG. Setting to 'Inf' uses a direct factorization instead of CG.	L	fmincon, fminunc, fsolve, lsqcurvefit, lsqlin, lsqnonlin, quadprog
RelLineSrchBnd	Relative bound on line search step length.	M	fgoalattain, fmincon, fminimax, fseminf
RelLineSrchBndDuration	Number of iterations for which the bound specified in RelLineSrchBnd should be active.	M	fgoalattain, fmincon, fminimax, fseminf
ScaleProblem	For fmincon interior-point and sqp algorithms, 'obj-and-constr' causes the algorithm to normalize all constraints and the objective function by their initial values. Disable by setting to the default 'none'. For the other solvers, when using the Algorithm option 'levenberg-marquardt', setting the ScaleProblem option to 'jacobian' sometimes helps the solver on badly-scaled problems.	L, I	fmincon, fsolve, lsqcurvefit, lsqnonlin, quadprog
Simplex	If 'on', function uses the simplex algorithm.	M	linprog
SubproblemAlgorithm	Determines how the iteration step is calculated.	I	fmincon

TolCon	Termination tolerance on the constraint violation.	B, I	bintprog, fgoalattain, fmincon, fminimax, fseminf, quadprog
TolConSQP	Constraint violation tolerance for the inner SQP iteration.	M	fgoalattain, fmincon, fminimax, fseminf
TolFun	Termination tolerance on the function value.	B, I	bintprog, fgoalattain, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, linprog (L only), lsqcurvefit, lsqlin (L only), lsqnonlin, quadprog
TolPCG	Termination tolerance on the PCG iteration.	L	fmincon, fminunc, fsolve, lsqcurvefit, lsqlin, lsqnonlin, quadprog
TolProjCG	A relative tolerance for projected conjugate gradient algorithm; this is for an inner iteration, not the algorithm iteration.	I	fmincon
TolProjCGAbs	Absolute tolerance for projected conjugate gradient algorithm; this is for an inner iteration, not the algorithm iteration.	I	fmincon
TolRLPFun	Termination tolerance on the function value of a linear programming relaxation problem.	M	bintprog
TolX	Termination tolerance on *x*.	B, I	All functions except the medium-scale algorithms for linprog, lsqlin, and quadprog
TolXInteger	Tolerance within which bintprog considers the value of a variable to be an integer.	M	bintprog

TypicalX	Array that specifies typical magnitude of array of parameters x. The size of the array is equal to the size of x0, the starting point.	B, I	fgoalattain, fmincon, fminimax, fminunc, fsolve, lsqcurvefit, lsqlin, lsqnonlin, quadprog
UseParallel	When 'always', applicable solvers estimate gradients in parallel. Disable by setting to 'never'.	M, I	fgoalattain, fmincon, fminimax.

Back to Top

Output Function

The OutputFcn field of the options structure specifies one or more functions that an optimization function calls at each iteration. Typically, you might use an output function to plot points at each iteration or to display optimization quantities from the algorithm. Using an output function you can view, but not set, optimization quantities. To set up an output function, do the following:

1. Write the output function as a function file or subfunction.
2. Use optimset to set the value of OutputFcn to be a function handle, that is, the name of the function preceded by the @ sign. For example, if the output function is outfun.m, the command

   ```
   options = optimset('OutputFcn', @outfun);
   ```

   specifies OutputFcn to be the handle to outfun. To specify more than one output function, use the syntax

   ```
   options = optimset('OutputFcn',{@outfun, @outfun2});
   ```

3. Call the optimization function with options as an input argument.

See Output Functions for an example of an output function.

Passing Extra Parameters explains how to parameterize the output function OutputFcn, if necessary.

Structure of the Output Function

The function definition line of the output function has the following form:

```
stop = outfun(x, optimValues, state)
```

where

- x is the point computed by the algorithm at the current iteration.
- optimValues is a structure containing data from the current iteration. Fields in optimValues describes the structure in detail.
- state is the current state of the algorithm. States of the Algorithm lists the possible values.
- stop is a flag that is true or false depending on whether the optimization routine should quit or continue. See Stop Flag for more information.

The optimization function passes the values of the input arguments to outfun at each iteration.

Fields in optimValues

The following table lists the fields of the `optimValues` structure. A particular optimization function returns values for only some of these fields. For each field, the Returned by Functions column of the table lists the functions that return the field.

Corresponding Output Arguments. Some of the fields of `optimValues` correspond to output arguments of the optimization function. After the final iteration of the optimization algorithm, the value of such a field equals the corresponding output argument. For example, `optimValues.fval` corresponds to the output argument `fval`. So, if you call `fmincon` with an output function and return `fval`, the final value of `optimValues.fval` equals `fval`. The Description column of the following table indicates the fields that have a corresponding output argument.

Command-Line Display. The values of some fields of `optimValues` are displayed at the command line when you call the optimization function with the `Display` field of `options` set to `'iter'`, as described in Iterative Display. For example, `optimValues.fval` is displayed in the `f(x)` column. The Command-Line Display column of the following table indicates the fields that you can display at the command line.

In the following table, L, M, and B indicate:

- L — Function returns a value to the field when using a large-scale algorithm.
- M — Function returns a value to the field when using a medium-scale algorithm.
- B — Function returns a value to the field when using both large- and medium-scale algorithms.

optimValues Fields

OptimValues Field (optimValues.field)	Description	Returned by Functions	Command-Line Display
`attainfactor`	Attainment factor for multiobjective problem. For details, see Goal Attainment Method.	`fgoalattain` (M)	None
`cgiterations`	Number of conjugate gradient iterations at current optimization iteration.	`fmincon` (L), `fsolve` (L), `lsqcurvefit` (L), `lsqnonlin` (L)	`CG-iterations` See Iterative Display.
`constrviolation`	Maximum constraint violation.	`fgoalattain` (M), `fmincon` (B I), `fminimax` (M), `fseminf` (M)	`max constraint` See Iterative Display.
`degenerate`	Measure of degeneracy. A point is *degenerate* if The partial derivative with respect to one of the variables is 0 at the point. A bound constraint is active for that variable at the point. See Degeneracy.	`fmincon` (L), `fsolve` (L), `lsqcurvefit` (L), `lsqnonlin` (L)	None

directionalderivative	Directional derivative in the search direction.	fgoalattain (M), fmincon (M), fminimax (M), fminunc (M), fseminf (M), fsolve (M), lsqcurvefit (M), lsqnonlin (M)	Directional derivative See Iterative Display.
firstorderopt	First-order optimality (depends on algorithm). Final value equals optimization function output output.firstorderopt.	fgoalattain (M), fmincon (B,I), fminimax (M), fminunc (M), fseminf (M), fsolve (B), lsqcurvefit (B), lsqnonlin (B)	First-order optimality See Iterative Display.
funccount	Cumulative number of function evaluations. Final value equals optimization function output output.funcCount.	fgoalattain (M), fminbnd (B), fmincon (B), fminimax (M), fminsearch (B), fminunc (B), fsolve (B), fzero (B), fseminf (M), lsqcurvefit (B), lsqnonlin (B)	F-count See Iterative Display.
fval	Function value at current point. Final value equals optimization function output fval.	fgoalattain (M), fminbnd (B), fmincon (B), fminimax (M), fminsearch (B), fminunc (B), fseminf (M), fsolve (B), fzero (B)	f(x) See Iterative Display.
gradient	Current gradient of objective function — either analytic gradient if you provide it or finite-differencing approximation. Final value equals optimization function output grad.	fgoalattain (M), fmincon (B), fminimax (M), fminunc (M), fseminf (M), fsolve (B), lsqcurvefit (B), lsqnonlin (B)	None
iteration	Iteration number — starts at 0. Final value equals optimization function output output.iterations.	fgoalattain (M), fminbnd (B),fmincon (B), fminimax (M), fminsearch (B), fminunc (B), fsolve (B), fseminf (M), fzero (B), lsqcurvefit (B), lsqnonlin (B)	Iteration See Iterative Display.
lambda	The Levenberg-Marquardt parameter, lambda, at the current iteration. See Levenberg-Marquardt Method.	fsolve (L, Levenberg-Marquardt algorithm), lsqcurvefit (L, Levenberg-Marquardt algorithm), lsqnonlin (L, Levenberg-Marquardt algorithm)	Lambda
maxfval	Maximum function value	fminimax (M)	None

positivedefinite	0 if algorithm detects negative curvature while computing Newton step. 1 otherwise.	fmincon (L), fsolve (L), lsqcurvefit (L), lsqnonlin (L)	None
procedure	Procedure messages.	fgoalattain (M), fminbnd (B), fmincon (M), fminimax (M), fminsearch (B), fseminf (M), fzero (B)	Procedure See Iterative Display.
ratio	Ratio of change in the objective function to change in the quadratic approximation.	fmincon (L), fsolve (L), lsqcurvefit (L), lsqnonlin (L)	None
residual	The residual vector. For fsolve, residual means the 2-norm of the residual squared.	lsqcurvefit (B), lsqnonlin (B), fsolve (B)	Residual See Iterative Display.
resnorm	2-norm of the residual squared.	lsqcurvefit (B), lsqnonlin (B)	Resnorm See Iterative Display.
searchdirection	Search direction.	fgoalattain (M), fmincon (M), fminimax (M), fminunc (M), fseminf (M), fsolve (M), lsqcurvefit (M), lsqnonlin (M)	None
stepaccept	Status of the current trust-region step. Returns true if the current trust-region step was successful, and false if the trust-region step was unsuccessful.	fsolve (L, NonlEqnAlgorithm='dogleg')	None
stepsize	Current step size (displacement in x). Final value equals optimization function output output.stepsize.	fgoalattain (M), fmincon (B), fminimax (M), fminunc (B), fseminf (M), fsolve (B), lsqcurvefit (B), lsqnonlin (B)	Step-size or Norm of Step See Iterative Display.
trustregionradius	Radius of trust region.	fmincon (L), fsolve (L, M), lsqcurvefit, lsqnonlin (L)	Trust-region radius See Iterative Display.

Degeneracy. The value of the field degenerate, which measures the degeneracy of the current optimization point x, is defined as follows. First, define a vector r, of the same size as x, for which r(i) is the minimum distance from x(i) to the *i*th entries of the lower and upper bounds, lb and ub. That is,

```
r = min(abs(ub-x, x-lb))
```

Then the value of `degenerate` is the minimum entry of the vector `r + abs(grad)`, where `grad` is the gradient of the objective function. The value of `degenerate` is 0 if there is an index `i` for which both of the following are true:

- `grad(i) = 0`
- `x(i)` equals the *i*th entry of either the lower or upper bound.

States of the Algorithm

The following table lists the possible values for `state`:

State	Description
`'init'`	The algorithm is in the initial state before the first iteration.
`'interrupt'`	The algorithm is in some computationally expensive part of the iteration. In this state, the output function can interrupt the current iteration of the optimization. At this time, the values of `x` and `optimValues` are the same as at the last call to the output function in which `state=='iter'`.
`'iter'`	The algorithm is at the end of an iteration.
`'done'`	The algorithm is in the final state after the last iteration.

The following code illustrates how the output function might use the value of `state` to decide which tasks to perform at the current iteration:

```
switch state
    case 'iter'
          % Make updates to plot or guis as needed
    case 'interrupt'
          % Probably no action here. Check conditions to see
          % whether optimization should quit.
    case 'init'
          % Setup for plots or guis
    case 'done'
          % Cleanup of plots, guis, or final plot
otherwise
end
```

Stop Flag

The output argument `stop` is a flag that is `true` or `false`. The flag tells the optimization function whether the optimization should quit or continue. The following examples show typical ways to use the `stop` flag.

Stopping an Optimization Based on Data in optimValues. The output function can stop an optimization at any iteration based on the current data in `optimValues`. For example, the following code sets `stop` to `true` if the directional derivative is less than `.01`:

```
function stop = outfun(x,optimValues,state)
stop = false;
% Check if directional derivative is less than .01.
if optimValues.directionalderivative < .01
    stop = true;
end
```

Stopping an Optimization Based on GUI Input. If you design a GUI to perform optimizations, you can make the output function stop an optimization when a user clicks a **Stop** button on the GUI. The following code shows how to do this, assuming that the **Stop** button callback stores the value `true` in the `optimstop` field of a `handles` structure called `hObject`:

```
function stop = outfun(x,optimValues,state)
stop = false;
% Check if user has requested to stop the optimization.
stop = getappdata(hObject,'optimstop');
```

Back to Top

Plot Functions

The `PlotFcns` field of the `options` structure specifies one or more functions that an optimization function calls at each iteration to plot various measures of progress while the algorithm executes. The structure of a plot function is the same as that for an output function. For more information on writing and calling a plot function, see Output Function. For an example of using built-in plot functions, Example: Using a Plot Function.

To view and modify a predefined plot function listed for `PlotFcns`, you can open it in the MATLAB Editor. For example, to view the file corresponding to the norm of residuals, enter:

```
edit optimplotresnorm.m
```

Back to Top