Choosing a Solver

On this page…

Optimization Decision Table

The following table is designed to help you choose a solver. It does not address multiobjective optimization or equation solving. There are more details on all the solvers in Problems Handled by Optimization Toolbox Functions.

Use the table as follows:

1. Identify your objective function as one of five types:
   - Linear
   - Quadratic
   - Sum-of-squares (Least squares)
   - Smooth nonlinear
   - Nonsmooth
2. Identify your constraints as one of five types:
   - None (unconstrained)
   - Bound
   - Linear (including bound)
   - General smooth
   - Discrete (integer)
3. Use the table to identify a relevant solver.

In this table:

- * means relevant solvers are found in Global Optimization Toolbox functions (licensed separately from Optimization Toolbox solvers).
- `fmincon` applies to most smooth objective functions with smooth constraints. It is not listed as a preferred solver for least squares or linear or quadratic programming because the listed solvers are usually more efficient.
- The table has suggested functions, but it is not meant to unduly restrict your choices. For example, `fmincon` can be effective on some nonsmooth problems.
- The Global Optimization Toolbox `ga` function can address mixed integer programming problems.

Solvers by Objective and Constraint

Constraint Type	Objective Type				
	Linear	**Quadratic**	**Least Squares**	**Smooth nonlinear**	**Nonsmooth**
None	n/a (f = const, or min = $-\infty$)	`quadprog`, Theory, Examples	`\`, `lsqcurvefit`, `lsqnonlin`, Theory, Examples	`fminsearch`, `fminunc`, Theory, Examples	`fminsearch`, *

Bound	linprog, Theory, Examples	quadprog, Theory, Examples	lsqcurvefit, lsqlin, lsqnonlin, lsqnonneg, Theory, Examples	fminbnd, fmincon, fseminf, Theory, Examples	fminbnd, *
Linear	linprog, Theory, Examples	quadprog, Theory, Examples	lsqlin, Theory, Examples	fmincon, fseminf, Theory, Examples	*
General smooth	fmincon, Theory, Examples	fmincon, Theory, Examples	fmincon, Theory, Examples	fmincon, fseminf, Theory, Examples	*
Discrete	bintprog, *, Theory, Example	*	*	*	*

Note This table does not list multiobjective solvers nor equation solvers. See Problems Handled by Optimization Toolbox Functions for a complete list of problems addressed by Optimization Toolbox functions.

Note Some solvers have several algorithms. For help choosing, see Choosing the Algorithm.

Back to Top

Choosing the Algorithm

- fmincon Algorithms
- fsolve Algorithms
- fminunc Algorithms
- Least Squares Algorithms
- Linear Programming Algorithms
- Quadratic Programming Algorithms

fmincon Algorithms

`fmincon` has four algorithm options:

- `'interior-point'`
- `'sqp'`
- `'active-set'`
- `'trust-region-reflective'` (default)

Use `optimset` to set the `Algorithm` option at the command line.

Recommendations

- Use the 'interior-point' algorithm first.

  For help if the minimization fails, see When the Solver Fails or When the Solver Might Have Succeeded.
- To run an optimization again to obtain more speed on small- to medium-sized problems, try 'sqp' next, and 'active-set' last.
- Use 'trust-region-reflective' when applicable. Your problem must have: objective function includes gradient, only bounds, or only linear equality constraints (but not both).

Reasoning Behind the Recommendations.

- 'interior-point' handles large, sparse problems, as well as small dense problems. The algorithm satisfies bounds at all iterations, and can recover from NaN or Inf results. It is a large-scale algorithm; see Large-Scale vs. Medium-Scale Algorithms. The algorithm can use special techniques for large-scale problems. For details, see Interior-Point Algorithm.
- 'sqp' satisfies bounds at all iterations. The algorithm can recover from NaN or Inf results. It is not a large-scale algorithm; see Large-Scale vs. Medium-Scale Algorithms.
- 'active-set' can take large steps, which adds speed. The algorithm is effective on some problems with nonsmooth constraints. It is not a large-scale algorithm; see Large-Scale vs. Medium-Scale Algorithms.
- 'trust-region-reflective' requires you to provide a gradient, and allows only bounds or linear equality constraints, but not both. Within these limitations, the algorithm handles both large sparse problems and small dense problems efficiently. It is a large-scale algorithm; see Large-Scale vs. Medium-Scale Algorithms. The algorithm can use special techniques to save memory usage, such as a Hessian multiply function. For details, see Trust-Region-Reflective Algorithm.
  - 'trust-region-reflective' is the default algorithm for historical reasons. It is effective when applicable, but it has many restrictions, so is not always applicable.

fsolve Algorithms

fsolve has three algorithms:

- 'trust-region-dogleg' (default)
- 'trust-region-reflective'
- 'levenberg-marquardt'

Use optimset to set the Algorithm option at the command line.

Recommendations

- Use the 'trust-region-dogleg' algorithm first.

  For help if fsolve fails, see When the Solver Fails or When the Solver Might Have Succeeded.
- To solve equations again if you have a Jacobian multiply function, or want to tune the internal algorithm (see Trust-Region-Reflective Algorithm Only), try 'trust-region-reflective'.
- Try timing all the algorithms, including 'levenberg-marquardt', to find the algorithm that works best on your problem.

Reasoning Behind the Recommendations.

- 'trust-region-dogleg' is the only algorithm that is specially designed to solve nonlinear equations. The others attempt to minimize the sum of squares of the function.
- The 'trust-region-reflective' algorithm is effective on sparse problems. It can use special

techniques such as a Jacobian multiply function for large-scale problems.

fminunc Algorithms

`fminunc` has two algorithms:

- Large-scale
- Medium-scale

Choose between them at the command line by using `optimset` to set the `LargeScale` option to:

- `'on'` (default) for Large-scale
- `'off'` for Medium-scale

Recommendations
• If your objective function includes a gradient, use `'LargeScale' = 'on'`. • Otherwise, use `'LargeScale' = 'off'`. For help if the minimization fails, see When the Solver Fails or When the Solver Might Have Succeeded.

Least Squares Algorithms

lsqlin. `lsqlin` has two algorithms:

- Large-scale
- Medium-scale

Choose between them at the command line by using `optimset` to set the `LargeScale` option to:

- `'on'` (default) for Large-scale
- `'off'` for Medium-scale

Recommendations
• If you have no constraints or only bound constraints, use `'LargeScale' = 'on'`. • If you have linear constraints, use `'LargeScale' = 'off'`. For help if the minimization fails, see When the Solver Fails or When the Solver Might Have Succeeded.

lsqcurvefit and lsqnonlin. `lsqcurvefit` and `lsqnonlin` have two algorithms:

- `'trust-region-reflective'` (default)
- `'levenberg-marquardt'`

Use `optimset` to set the `Algorithm` option at the command line.

Recommendations
• If you have no constraints or only bound constraints, use `'trust-region-reflective'`. • If your problem is underdetermined (fewer equations than dimensions), use `'levenberg-marquardt'`. For help if the minimization fails, see When the Solver Fails or When the Solver Might Have Succeeded.

Linear Programming Algorithms

`linprog` has three algorithms:

- Large-scale interior-point
- Medium-scale active set
- Medium-scale Simplex

Choose between them at the command line by using `optimset` to set the `LargeScale` option to:

- `'on'` (default) for large-scale interior-point
- `'off'` for one of the other two

  When `LargeScale` is `'off'`, choose between the two remaining algorithms by setting the `Simplex` option to:
  - `'on'` for Simplex
  - `'off'` (default) for active set

Recommendations
Use `'LargeScale' = 'on'` For help if the minimization fails, see When the Solver Fails or When the Solver Might Have Succeeded.

Quadratic Programming Algorithms

`quadprog` has three algorithms:

- `trust-region-reflective` (formerly `LargeScale = 'on'`), the default
- `active-set` (formerly `LargeScale = 'off'`)
- `interior-point-convex`

Use `optimset` to set the `Algorithm` option at the command line.

Recommendations
• If you have a convex problem, or if you don't know whether your problem is convex, use `interior-point-convex`. • If you have only bounds, or only linear equalities, use `trust-region-reflective`. • If you have a nonconvex problem that does not satisfy the restrictions of `trust-region-reflective`, use `active-set`. For help if the minimization fails, see When the Solver Fails or When the Solver Might Have Succeeded.

Back to Top

Problems Handled by Optimization Toolbox Functions

The following tables show the functions available for minimization, equation solving, multiobjective optimization, and solving least-squares or data-fitting problems.

Minimization Problems

Type	Formulation	Solver

Scalar minimization	$\min_{x} f(x)$ such that $l < x < u$ (x is scalar)	fminbnd
Unconstrained minimization	$\min_{x} f(x)$	fminunc, fminsearch
Linear programming	$\min_{x} f^T x$ such that $A \cdot x \le b$, $Aeq \cdot x = beq$, $l \le x \le u$	linprog
Quadratic programming	$\min_{x} \frac{1}{2} x^T H x + c^T x$ such that $A \cdot x \le b$, $Aeq \cdot x = beq$, $l \le x \le u$	quadprog
Constrained minimization	$\min_{x} f(x)$ such that $c(x) \le 0$, $ceq(x) = 0$, $A \cdot x \le b$, $Aeq \cdot x = beq$, $l \le x \le u$	fmincon
Semi-infinite minimization	$\min_{x} f(x)$ such that $K(x,w) \le 0$ for all w, $c(x) \le 0$, $ceq(x) = 0$, $A \cdot x \le b$, $Aeq \cdot x = beq$, $l \le x \le u$	fseminf
Binary integer programming	$\min_{x} f^T x$ such that $A \cdot x \le b$, $Aeq \cdot x = beq$, x binary	bintprog

Multiobjective Problems

Type	Formulation	Solver
Goal attainment	$\min_{x,\gamma} \gamma$ such that $F(x) - w \cdot \gamma \le$ goal, $c(x) \le 0$, $ceq(x) = 0$, $A \cdot x \le b$, $Aeq \cdot x = beq$, $l \le x \le u$	fgoalattain
Minimax	$\min_{x} \max_{i} F_i(x)$ such that $c(x) \le 0$, $ceq(x) = 0$, $A \cdot x \le b$, $Aeq \cdot x = beq$, $l \le x \le u$	fminimax

Equation Solving Problems

Type	Formulation	Solver
Linear equations	$C \cdot x = d$, n equations, n variables	\ (matrix left division)
Nonlinear equation of one variable	$f(x) = 0$	fzero
Nonlinear equations	$F(x) = 0$, n equations, n variables	fsolve

Least-Squares (Model-Fitting) Problems

Type	Formulation	Solver
Linear least-squares	$$\min_x \frac{1}{2}\|C\cdot x-d\|_2^2$$ *m* equations, *n* variables	\ (matrix left division)
Nonnegative linear-least-squares	$$\min_x \frac{1}{2}\|C\cdot x-d\|_2^2$$ such that $x \geq 0$	`lsqnonneg`
Constrained linear-least-squares	$$\min_x \frac{1}{2}\|C\cdot x-d\|_2^2$$ such that $A\cdot x \leq b$, $Aeq\cdot x = beq$, $lb \leq x \leq ub$	`lsqlin`
Nonlinear least-squares	$$\min_x \|F(x)\|_2^2 = \min_x \sum_i F_i^2(x)$$ such that $lb \leq x \leq ub$	`lsqnonlin`
Nonlinear curve fitting	$$\min_x \|F(x,xdata)-ydata\|_2^2$$ such that $lb \leq x \leq ub$	`lsqcurvefit`

Back to Top

Was this topic helpful? Yes No

© 1984-2012 The MathWorks, Inc. • Terms of Use • Patents • Trademarks • Acknowledgments