Optimization Algorithms in MATLAB

Maria G Villarreal

ISE Department

The Ohio State University

February 03, 2011

Outline

- Problem Description
- Optimization Problem that can be solve in MATLAB
- Optimization Toolbox solvers
- Non Linear Optimization
- Multobjective Optimization

Problem Description

- **Objective:**
  - Determine the values of the controllable process variables (factors) that improve the output of a process (or system).
- **Facts:**
  - Have a computer simulator (input/output black-box) to represent the output of a process (or system).
  - The simulation program takes very long time to run.
- **Procedure:**
  - Evaluate a set (usually small) of input combination (DOE) into the computer code and obtain an output value for each one.
  - Construct a mathematical model to relate inputs and outputs, which is easier and faster to evaluate then the actual computer code.
  - Use this model (metamodel), and via an optimization algorithm obtained the values of the controllable variables (inputs/factors) that optimize a particular output (s).

Optimization Problem that can be solve in MATLAB (Optimization Toolbox)

- **Constrained and Unconstrained continues and discrete**
  - Linear
  - Quadratic
  - Binary Integer
  - **Nonlinear**
  - **Multiobjective Problems**

Optimization Toolbox solvers

- **Minimizers**

This group of solvers attempts to find a local minimum of the objective function near a starting point x_0.

If you have a Global Optimization Toolbox license, use the GlobalSearch or MultiStart solvers. These solvers automatically generate random start points within bounds.

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`

Optimization Toolbox solvers

- **Multiobjective minimizers**

  This group of solvers attempts to either minimize the maximum value of a set of functions (fminimax), or to find a location where a collection of functions is below some prespecified values (fgoalattain).

Multiobjective Problems

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

Non Linear Optimization

A General Problem (GP) description is stated as

$$\min_x f(x), \tag{6-1}$$

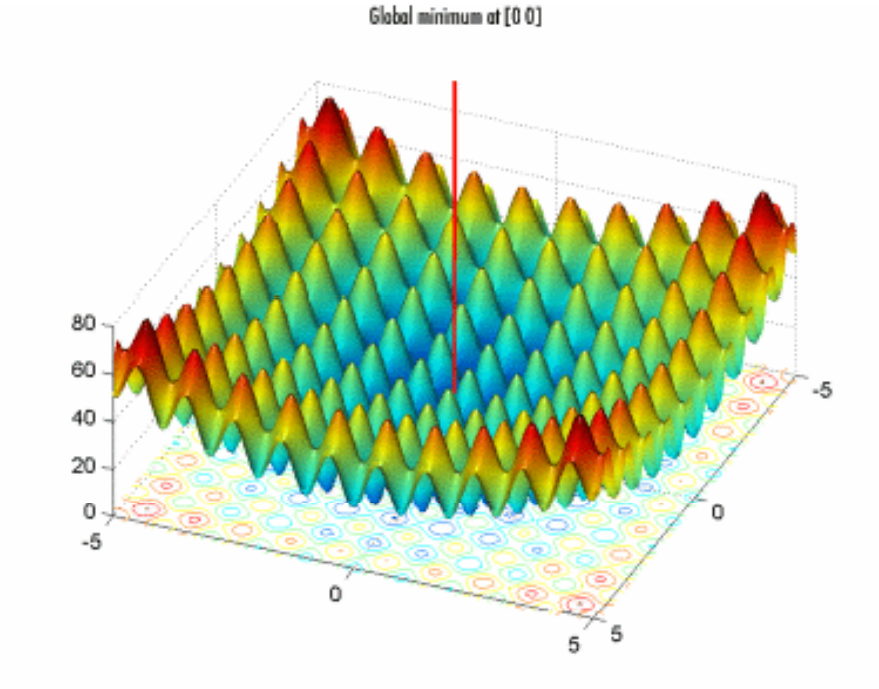
subject to

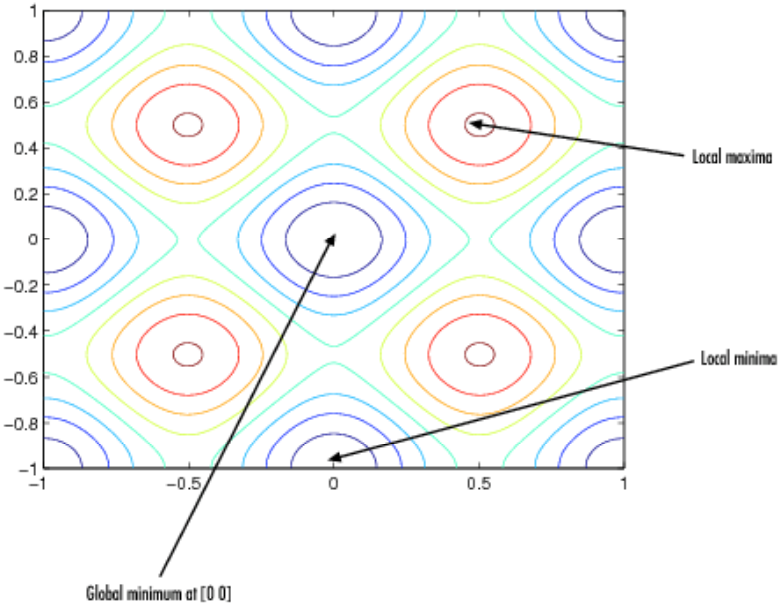
$$G_i(x) = 0 \quad i = 1,\ldots,m_e,$$
$$G_i(x) \le 0 \quad i = m_e + 1,\ldots,m,$$

where x is the vector of length n design parameters, $f(x)$ is the objective function, which returns a scalar value, and the vector function $G(x)$ returns a vector of length m containing the values of the equality and inequality constraints evaluated at x.

Rastrigin's function

$$f_6(x) = 10 \cdot n + \sum_{i=1}^{n} \left(x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i) \right) \qquad -5.12 \le x_i \le 5.12$$





[Global Optimization Toolbox Guide]

General Search Algorithm

Move from point $\mathbf{x}^{(k)}$ to $\mathbf{x}^{(k+1)}=\mathbf{x}^{(k)}+\Delta\mathbf{x}^{(k)}$, where $\Delta\mathbf{x}^{(k)}=\alpha_k d^{(k)}$, such that $f(\mathbf{x}^{(k+1)})<f(\mathbf{x}^{(k)})$.

$d^{(k)}$ is a "desirable" search direction and,

α_k is called the step size.

To find $\Delta\mathbf{x}^{(k)}$ we need to solve to subproblems, one to find $d^{(k)}$ and one for α_k.

These iterative procedures (techniques) are often called "direction methods".

General Steps

1. Estimate a reasonable initial point $\mathbf{x}^{(0)}$. Set k=0 (iteration counter).
2. Compute a direction search $d^{(k)}$
3. Check convergence.
4. Calculate step size α_k in the direction $d^{(k)}$.
5. Update $\mathbf{x}^{(k+1)}=\mathbf{x}^{(k)}+\alpha_k d^{(k)}$
6. Go to step 2.

Algorithms to compute Search Direction (**d**)

- **Steepest Descent Method** (Gradient method)
- **Conjugate Gradient Method**
- **Newton's Method** (Uses second order partial derivative information)
- **Quasi-Newton Methods** (Approximates Hessian matrix and its inverse using first order derivative)
  - **DFP Method** (Approximates the inverse of the Hessian)
  - **BFGS Method** (Approximates Hessian matrix)

Steepest Descent Method

1. Estimate starting point $\mathbf{x}^{(0)}$. Set k=0
2. Calculate the gradient of $f(\mathbf{x})$ at $\mathbf{x}^{(k)}$ as $\mathbf{c}^{(k)}=\nabla f(\mathbf{x}^{(k)})$
3. Check convergence.
4. Let $d^{(k)} = -\mathbf{c}^{(k)}$. Use another algorithm to find α_k .
5. Update $\mathbf{x}^{(k+1)}=\mathbf{x}^{(k)}+\alpha_k d^{(k)}$. Set k=k+1
6. Go to step 2.

Drawbacks: Poor rate of convergence

Newton's Method

1. Estimate starting point $\mathbf{x}^{(0)}$. Set k=0
2. Calculate $\mathbf{c}^{(k)}$(gradient of $f(\mathbf{x})$ at $\mathbf{x}^{(k)}$)
3. Check convergence (if $||\mathbf{c}^{(k)}||<\varepsilon$, stop)
4. Calculate the Hessian matrix at $\mathbf{x}^{(k)}$, $H^{(k)}$.
5. Calculate the direction search as $d^{(k)} = -[H^{(k)}]^{-1}\mathbf{c}^{(k)}$. Use another algorithm to compute α_k .
6. Update $\mathbf{x}^{(k+1)}=\mathbf{x}^{(k)}+\alpha_k d^{(k)}$.
7. Set k=k+1 and go to step 2.

Drawbacks:

- Needs to calculate second order derivatives.
- H needs to be positive definite to assure a decent direction
- H may be singular at some point.

Quasi-Newton's Method
BFGS Method (Used in MALTAB)

Step 1. Estimate an initial design $\mathbf{x}^{(0)}$. Choose a symmetric positive definite $n \times n$ matrix $\mathbf{H}^{(0)}$ as an estimate for the Hessian of the cost function. In the absence of more information, let $\mathbf{H}^{(0)} = \mathbf{I}$. Choose a convergence parameter ε. Set $k = 0$, and compute the gradient vector as $\mathbf{c}^{(0)} = \nabla f(\mathbf{x}^{(0)})$.

Step 2. Calculate the norm of the gradient vector as $\|\mathbf{c}^{(k)}\|$. If $\|\mathbf{c}^{(k)}\| < \varepsilon$ then stop the iterative process; otherwise continue.

Step 3. Solve the linear system of equations $\mathbf{H}^{(k)}\mathbf{d}^{(k)} = -\mathbf{c}^{(k)}$ to obtain the search direction.

Step 4. Compute optimum step size $\alpha_k = \alpha$ to minimize $f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)})$.

Step 5. Update the design as $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{d}^{(k)}$

Step 6. Update the Hessian approximation for the cost function as

$$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \mathbf{D}^{(k)} + \mathbf{E}^{(k)} \tag{a}$$

where the correction matrices $\mathbf{D}^{(k)}$ and $\mathbf{E}^{(k)}$ are given as

$$\mathbf{D}^{(k)} = \frac{\mathbf{y}^{(k)}\mathbf{y}^{(k)^T}}{(\mathbf{y}^{(k)} \cdot \mathbf{s}^{(k)})}; \qquad \mathbf{E}^{(k)} = \frac{\mathbf{c}^{(k)}\mathbf{c}^{(k)^T}}{(\mathbf{c}^{(k)} \cdot \mathbf{d}^{(k)})} \tag{b}$$

$$\mathbf{s}^{(k)} = \alpha_k\mathbf{d}^{(k)} \text{ (change in design)}; \qquad \mathbf{y}^{(k)} = \mathbf{c}^{(k+1)} - \mathbf{c}^{(k)} \text{ (change in gradient)};$$
$$\mathbf{c}^{(k+1)} = \nabla f(\mathbf{x}^{(k+1)}) \tag{c}$$

Step 7. Set $k = k + 1$ and go to Step 2.

[Arora, J. (2004), Introduction to Optimum Design, 2nd Ed, page. 327]

Method to calculate step size (assuming **d** is known)

- Equal interval search
- Golden Search
- Polynomial Interpolation
- Inaccurate Line Search

Optimization toolbox for Non Linear Optimization

- **Solvers:**
  - **fmincon** (constrained nonlinear minimization)
    - **Trust-region-reflective (default)**
      - Allows only bounds or linear equality constraints, but not both.
    - **Active-set** (solve Karush-Kuhn-Tucker (KKT) equations and used quasi-Netwon method to approximate the hessian matrix)
    - **Interior-point**
    - **Sequential Quadratic Programming** (SQP)
  - **fminunc** (unconstrained nonlinear minimization)
    - Large-Scale Problem: **Trust-region method** based on the interior-reflective Newton method
    - Medium–Scale: **BFGS Quasi-Newton method** with a cubic line search procedure.
  - **fminsearch** (unconstrained multivariable optimization, nonsmooth functions)
    - **Nelder-Mead simplex** (derivative-free method)

Multiobjective Optimization

- **Solvers:**
  - **fminmax** (minimize the maximum value of a set of functions).
  - **fgoalattain** (find a location where a collection of functions are below some prespecified values).

Choosing a Solver (Optimization Decision Table)

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	*
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				

[Optimization Toolbox Guide]