Fmincon Command:

Minimizing Constrained Nonlinear Multivariable Functions

Spencer Ashley, Julie DiNitto, Jennifer Huling, Nichole Smith

Wake Forest University

Department of Mathematics

Nonlinear Optimization Theory

April 9, 2013

Constrained Optimization Set Up:

In order to use *fmincon*, first define an objective function $f(x)$ and constraints:

$$\min_x f(x) \text{ such that } \begin{cases} c(x) \leq 0 & \text{nonlinear inequality} \\ ceq(x) = 0 & \text{nonlinear equality} \\ A \cdot x \leq b & \text{linear inequality} \\ Aeq \cdot x = beq & \text{linear equality} \\ lb \leq x \leq ub & \text{lower and upper bounds} \\ & \quad \text{on variables.} \end{cases}$$

fmincon is a gradient-based method that is designed to work on problems where the objective and constraint functions are both continuous and have continuous first derivatives.

Algorithms

fmincon is capable of using four different algorithms:

1. Interior Point Method
2. Active Set Method
3. Sequential Quadratic Programming (SQP) Method
4. Trust Region Reflective Method

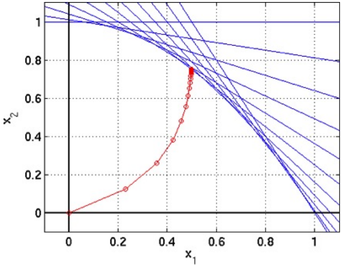
Algorithms Continued

	Interior Point	SQP	Active Set	Trust-Region
Solves non-linear constraints*	X	X	X	
Satisfies bounds at iterations	X	X		
Recovers from NaN or Inf results	X	X		
Fast Speeds			X	
Allows for non-smooth constraints			X	
Requires user-supplied gradient				X
Cheap on memory		X		X

Interior Point Algorithm

Idea behind the method:

- Instead of moving along the function until finding a minimum, the interior point method starts inside the feasible set of the function and moves toward the boundary of the function as defined by the barrier.
- This method is used for minimizing linear and non-linear convex functions.



Interior Point Continued

Process:

- Replaces inequality constraints with easier to solve equality constraints.
- Solves new minimization problem subject to $h(x) = 0$ and $g(x) + s = 0$:

$$\min_{x,s} f_\mu(x,s) = \min_{x,s} f(x) - \mu \sum_i \ln(s_i).$$

  - $\mu \sum_i \ln(s_i)$, is known as the barrier function
  - μ is a positive scaling factor
  - s_i is the slack variable
  - As $\mu \to 0$, the minimum of $f_\mu(x,s)$ approaches the minimum of $f(x)$
- Each iteration tries to take a direct step based on the Newton method. If that is not possible, it uses the conjugate gradient method.

Active Set Algorithm

Idea behind the method:

- Uses Sequential Quadratic Programming (SQP) methods to guarantee super-linear convergence by accumulating second-order information regarding the KKT equations.
- Feasible initial point is necessary.
- Medium-scale algorithm which internally create full matrices and uses dense linear algebra. This may take longer to execute.
- This method will be used as the default method in future versions of MATLAB.

Active Set Continued

Process:

- **Updating the Hessian Matrix:**
  A positive definite quasi-Newton approximation of the Hessian of the Lagrangian function is calculated using the BFGS method at each iteration.
- **Quadratic Programming Solution:**
  The inequality constraints are written as equality constraints and this sub-problem is solved using a quadratic programming method. The solution is used to form a search direction for a line search procedure. The first phase calculates a feasible point (if one exists). The second phase generates an iterative sequence of feasible points that converge to the solution.
- **Line Search and Merit Function:**
  The step length parameter defined by the steps above is determined so that a sufficient decrease in a merit function is obtained.

SQP Algorithm

Idea behind the method:

The Sequential Quadratic Programming algorithm uses the same SQP iterations used above in the Active Set algorithm, except SQP has certain restrictions at each iteration which slows it down comparatively.

Differences:

1. Each iterative step is taken within the region constrained by the bounds. Used if the function behaves poorly, is complex-valued, or is undefined outside the boundary.
2. The SQP can recover and take a smaller step in the case that the value initially returned is undefined (NaN) or infinite (Inf).
3. SQP algorithm tries to minimize the merit function in order to find the most feasible solution, which allows the command to converge in fewer iterations.

Trust Region Reflective Algorithm:

Idea behind the method:

- The Trust Region Reflective method is currently the default algorithm MATLAB implements when running *fmincon*.
- Best used for solving non-convex problems with variable bounds or linear equality constraints, but not both.
- Not optimal for large scale problems.

Trust Region Method Continued

Process:

1. **Subspace Trust Region Method: Formulates Trust Region Subproblem**
   - Defines a neighborhood, N around an initial point, x_0.
   - Approximates the objective function f, with a more simple function q, which behaves similarly to f within N.
   - Solves trust region subproblem:

$$\min_s \{\frac{1}{2} s^T H s + s^T g\} \text{ such that } ||Ds|| \leq \Delta.$$

     - H is the hessian
     - D is a diagonal scaling matrix
     - Δ is a positive scalar
     - g is the user-supplied gradient of f evaluated at the current iterate point
   - Restricts the trust region sub-problem to a two dimensional subspace.

Trust Region Method Continued

Method:

1. Formulate the two-dimensional trust region sub-problem
2. Solve $\min_s\{\frac{1}{2}s^T Hs + s^T g\}$ such that $||Ds|| \leq \Delta$ to determine the trial step s
3. If $f(x+s) < f(x)$, then $x = x + s$.
4. Adjust Δ.

Trust Region Method Continued

Constraints:

1. Linear Equality constraints
   - Initial point x_0 is computed using sparse least-squares step so $Aeq * x_0 = beq$
   - PCG is replaced with Reduced PCG to compute approx Newton Step
2. Box constraints: Adding lower and upper bounds using Scalar modified Newton step and reflections are used to increase the step size
3. Does not do inequality constraints (only linear equality constraints or bounds but not both)
4. Requires the gradient to be supplied

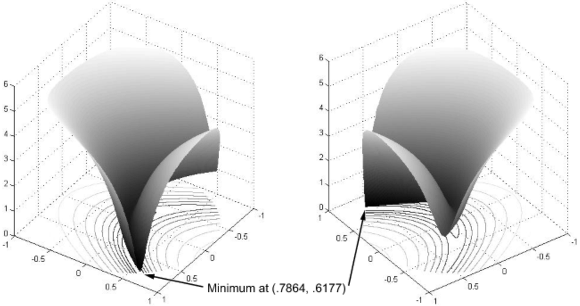
Rosenbrock Example

Consider minimizing the standard Rosenbrock function

$$f(x) = 100(y - x^2)^2 + (1 - x)^2$$

subject to the unit disk, $x^2 + y^2 \leq 1$.

Note that because this is an inequality constraint, the Trust Region Reflective Method cannot be utilized.



Rosenbrock Example continued

Implementing *fmincon*:

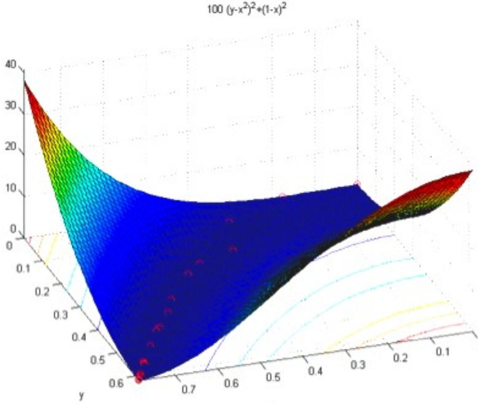
- *fmincon* can be implemented in either the Command Window or through the Optimization Toolbox. The next slide includes the appropriate code for the Rosebrock minimization problem within the unit disk in each format.
- To change the desired algorithm in the Command Window, simply replace the given algorithm with 'active-set', 'sqp', or 'interior-point'. The GUI makes this user friendly by giving the user a drop-down box to choose an algorithm. The constraints which are unallowed for a given algorithm are then unaccessible.

Rosenbrock Example Code

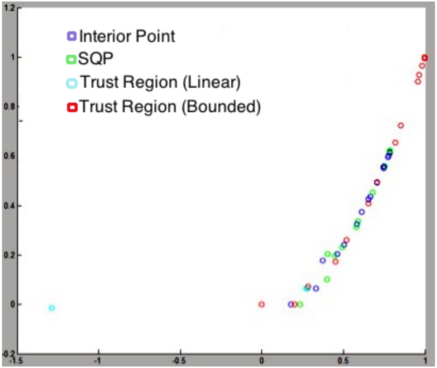
Command Window Code:

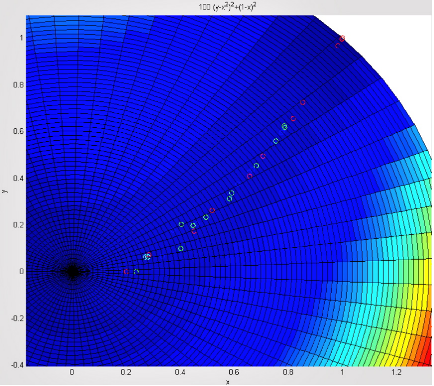
```
function [c, ceq] = unitdisk(x)
c = x(1)^2 + x(2)^2 - 1;
ceq = [ ];
options = optimset('Display',
'iter','Algorithm','interior-
point','PlotFcns',@optimplotx);
[x,fval] = fmincon(@rosenbrock,
[0 0],[],[],[],[],[],[],
@unitdisk,options)
Output:
x =     0.7864    0.6177
fval =     0.0457
```

IPM Iterates Shown:



Comparison:





These graphs plot each iterative step for the various algorithms. The SQP Algorithm converges in the fewest iterations (58 vs 79 and 84 with Active Set and Interior Point respectively), but with less "nice" functions, the SQP iterates can be more expensive in terms of time.

Rosenbrock Example: Trust Region Method with Equality Constraint

To solve a constrained Rosenbrock function using the Trust Region Method, it is necessary to define a linear equality constraint. Consider the following equality constraint:

$$h(x) = -x + 20y - 1.$$

To write this constraint in terms MATLAB can configure, we define $A = [-1, 20]$ and $b = 1$. And run this in the Command Window with the following code:

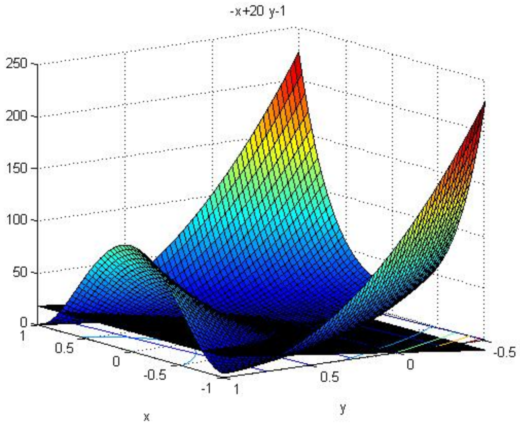
```
f = @(x,y)100*(y - x^2)^2 + (1-x)^2;
Aeq = [-1,20];
beq = 1;
Options = optimset('Display','iter','GradObj','on','Algorithm',
'trust-region-reflective','TolFun',1.00E-19);
[x,fval] = fmincon(@rosenbrock,[-1.3,0.2],[],[],Aeq,beq,[],[],[],
Options)
```

Rosenbrock: TRM with Equality Constraint Continued

Output after 7 iterations:

```
x = 0.2795    0.0640
fval = 0.5391
```

Plot of Rosenbrock function and equality constraint:



Rosenbrock: TRM with Variable Bounds

Additional Code:

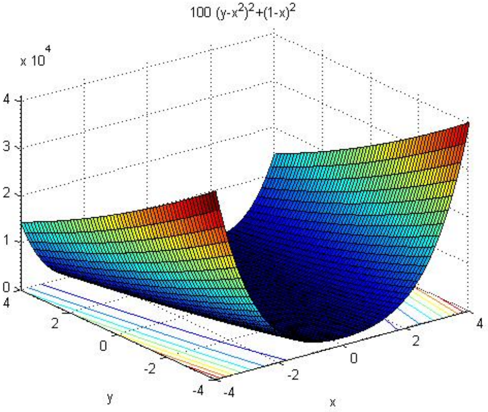
```
lb = [-4,-4];
ub = [4,4];
figure(2)
ezsurfc(f,[lb(1),ub(1)]
,[lb(2),ub(2)])

[x,fval] = fmincon
(@rosenbrock,[0,0],
[],[],[],[],lb,ub,[],
Options)
```

Output after 16 iterations:

```
x =1.0000    1.0000
fval =4.8482e-23
```

Plot of Rosenbrock function with variable bounds:



Practical Application:

Problem:

In designing structural columns which are subjected to exceptionally large loads (as on a jib crane), it is optimum (cheaper and accessible) to build such a structure with minimum mass that can still support heavy loads. In one particular example we are given the following objective function and constraints:

Objective function:

$$f(x) = \text{(Mass Density)(Length)(Cross-sectional Area)} = (7850)(5)(2\pi Rt)$$

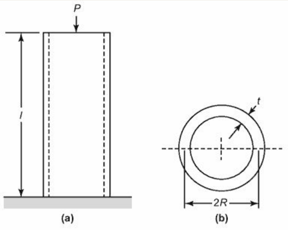
where R is the mean radius of the tube, and t is the wall thickness.

Application Continued

Problem Continued:

Constraints:

- Stress Constraint: $\sigma \leq \sigma_a$ where σ is the equation describing the normal stress and σ_a is the allowable stress.
- Buckling load constraint: $P \leq P_{cr}$ where P is the actual load and P_{cr} is the equation describing the buckling load.
- Deflection (bending) constraint: $\delta \leq \Delta$ where δ is the equation describing the point of deflection and Δ is the allowable amount of deflection.
- Radius/thickness constraint: $\frac{R}{t} \leq 50$.
- Bounds on the variables: $0.01 \leq R \leq 1$ and $0.005 \leq t \leq 0.2$.



Application Continued

Solution

After coding m-files for invoking *fmincon* (SQP Algorithm), defining the objective function, and defining the constraints, the output gives

$$R = 0.0537 \text{ and } t = 0.005,$$

suggesting the minimum wall width and an appropriate width for the inner tube in order to minimize the mass of the column, but still satisfy and constraints dictated by the mass the column will be bearing.

(Arora)

References:

Arora, J.S. "Introduction to Optimum Design 3rd Ed.," Waltham, MA: Academic Press, 2012.

Coleman, T.F. and Li, Y. "A Reflecting Newton Method for Minimizaing a Quadratic Function Subject to Bounds on Some of the Variables," Advanced Computing Research Institute, Cornell University, 1992.

MATLAB R2013a Help Documents: fmincon, Choosing a Solver, Constrained Nonlinear Optimization Algorithms

Mesgari, F.C. "Application of MAG Index for Optimal Grasp Planning," 2011 IEEE International Conference on Mechatronics and Automation, Aug 2011.