Examples of Running Constrained Optimization Codes

CEE 201L. Uncertainty, Design, and Optimization
Department of Civil and Environmental Engineering
Duke University
Henri P. Gavin
Spring, 2016

Optimized Step-Size Random Search (ORSopt)  
Nelder-Mead with inequality constraints (NMAopt)  
Sequential Quadratic Programming (SQPopt)  
Symmetric Perturbation Stochastic Approximation (SPSA1opt)

www.duke.edu/~hpgavin/cee201/ORSopt.m
www.duke.edu/~hpgavin/cee201/NMAopt.m
www.duke.edu/~hpgavin/cee201/SQPopt.m
www.duke.edu/~hpgavin/cee201/SPSA1opt.m

Objective function

\[
\text{minimize } J = f(x_1, x_2) = 1 + (x_1 - c_1)^2 + (x_2 - c_2)^2 + c_3 N,
\]

where \(c_1, c_2,\) and \(c_3\) are constants, and \(N\) is a random value with a standard-normal distribution. The coefficient \(c_3\) determines the level of randomness in the objective.

Constraint Inequalities

such that:
\[
g_1(x_1, x_2) = ((x_1 - 0.2)^2 + (x_2 - 0.5)^2) - 0.3 \leq 0,
\]
\[
g_2(x_1, x_2) = -((x_1 + 0.5)^2 + (x_2 - 0.5)^2) + 2.0 + 1.5 \leq 0,
\]
and
\[
0 \leq x_i \leq 1
\]

Code

```matlab
function [ objective , constraints ] = optim_example_analysis( x , c )
% Deb, K. , "An e f f i c i e n t constraint handling method for genetic algorithms ,"
% H.P. Gavin , Dept . Civil & Environ . Eng'g , Duke Univ . , 2011−2015

objective = 1.0 + ( x (1) - c (1) )ˆ2 + ( x (2) - c (2) )ˆ2 + c (3)* randn ;
constraints = [ (( x (1) -0.2)ˆ2 + (x (2) -0.5)ˆ2) *1 - 0.3 ;
-(( x (1)+0.5)ˆ2 + (x (2) -0.5)ˆ2) *2.0 + 1.5 ];

x_min = [ 0 ; 0 ]; % minimum allowable parameter values
x_max = [ 1 ; 1 ]; % maximum allowable parameter values
x_init = x_min + rand(2,1).* (x_max - x_min ); % random initial guess , or ...  
% a specified initial guess

c = [ 0.8 ; 0.8 ]; % other constants used in the cost function

% display tolX tolP tolG MaxEvals Penalty Exponent nMax errJ
options = [ 3 0.02 0.02 0.001 5e2 1.0 0.5 1 0.10 ];

% ORSopt('optim_example_analysis', x_init, x_min, x_max, options, c )
% NMAopt('optim_example_analysis', x_init, x_min, x_max, options, c )
% SQPopt('optim_example_analysis', x_init, x_min, x_max, options, c )
% SPSA1opt('optim_example_analysis', x_init, x_min, x_max, options, c )
plot_cvg_hst ( cvg_hst , x_opt , 100); % plot the convergence history
```
Figure 1. ORSopt convergence on penalized objective function surface.

\[ f_{\text{opt}} = 1.0166 \times 10^0 \quad \max(g_{\text{opt}}) = -5.8112 \times 10^{-3} \]

Figure 2. ORSopt convergence histories.
Examples of Running Constrained Optimization Codes

Figure 3. NMAopt convergence on penalized objective function surface with no noise in \( f(x) \).
\[ f_{\text{opt}} = 1.0155e+00 \quad \text{max}(g_{\text{opt}}) = -1.1384e-03 \]

Figure 4. NMAopt convergence histories.
Figure 5. SQPopt convergence on constrained objective function surface.

\[ f_{\text{opt}} = 1.0151 \times 10^0 \quad \text{max}(g_{\text{opt}}) = 2.3186 \times 10^{-4} \]

Figure 6. SQPopt convergence histories.
The Nelder-Mead method for noisy objective functions

In problems with noisy objective functions numerical gradients are very sensitive to the noise level. For these kinds of problems, search methods (such as the Nelder-Mead method) can be less computationally-intensive than gradient-based methods (such as SQP).

The implementation of the Nelder-Mead method in NMAopt.m can handle problems with noisy objective functions by changing three values in the vector of options:

- If the objective function is noisy then repeated evaluation of $J = f(x)$ for the same $x$ will give randomly different numbers. In this setting, it is not reasonable to expect convergence with respect to the objective function. So we should assess convergence only with respect to the parameter values, $x$, (via $\text{tolX}$). To do so, set $\text{tolF}$ (the third value in options) to be large (the value 10.0 in the code below).

- Noisy objective functions can be assessed in terms of their statistical properties (for example, their mean (average) and their standard deviation (coefficient of variation)). Estimating the mean $m_J(x)$ and coefficient of variation $c_J(x)$ of $f(x)$ requires a sample of values of $f$ for the same value of $x$. Given a sample $[J_1, J_2, ..., J_n]$, from $n$ repeated evaluations of the objective function, the estimates of the mean and coefficient of variation can be computed as follows:

$$m_J = \frac{1}{n} \sum_{i=1}^{n} J_i$$

and

$$c_J = \frac{1}{m_J} \left[ \frac{1}{n-1} \sum_{i=1}^{n} (J_i - m_J)^2 \right]^{1/2}$$

Because the estimate of the mean and c.o.v. of $J$ requires a sample of $n$ evaluations, the total number of evaluations for the optimization will be larger than what is used in non-random (deterministic) optimization. To allow for this the maximum total number of function evaluations, $\text{MaxEvals}$ (the fifth value in options) should be set larger.

- How many evaluations should be used in computing $m_J$ and $c_J$? That is, what value for $n$ should be specified? The answer to this question depends on the level of inherent variability in $f(x)$, on the desired statistical error $e_J$ in the estimate of $m_J$, and the level of confidence we require of our estimate. The inherent variability of $f(x)$ is estimated as $c_J$.

  The greater this inherent variability, the larger the sample size should be.

  The smaller the desired error, the larger the sample size should be.

  The larger the level of confidence in the estimate, the larger the sample size should be.

  The equation for $n$ is:

$$n = \left[ \frac{z_{\alpha/2} \cdot c_J}{e_J} \right]^2$$

where $z_{\alpha/2} = 1.645$ for a 90% confidence level on the estimate of $m_J$.

The ninth value in options, $\text{nMax}$, sets a limit on the sample size $n$ in order to restrict $n$ from becoming too large, (in cases with very large $c_J$ or very small desired $e_J$).

The tenth value in options, $\text{errJ}$, sets the value of $e_J$, the desired estimation error for the mean, $m_J$.

Note that optimizing with small values of $\text{errJ}$ and large values of $\text{nMax}$ could require many (many) function evaluations, but will represent the statistics of the objective function very well.

- For optimization problems with noisy objective functions, it is sometimes desirable to recognize the randomness of the objective in the cost function. Optimization cost functions for random objective functions are called risk measures.
The .m-file `avg_cov_func.m` takes care of computing \( m_J, c_J \), and the risk measure to be optimized. A number of risk measures for stochastic optimization problems can be selected within `avg_cov_func.m`: the sample max, the sample average, the sample average plus the sample standard deviation (the 84th percentile of the objective function), or the sample average plus the sample standard deviation divided by \( \sqrt{n} \) (the 84th percentile of the mean estimate).

In the example script below, the noise-level in this example problem is set by the coefficient \( c_3 \), so in this example we know in advance that the standard deviation of \( J \) is equal to the value we use for \( c_3 \) (0.15).

Setting \( \text{errJ} \) to 0.10 means that we desire an estimate for the mean of \( J \) that is accurate to within \( \pm 10\% \), with a 90\% confidence level. Using this information along with the equation for \( n \), above, we will need a sample size of \( n = (1.645 \times 0.15/0.10)^2 \approx 6 \). We have set the maximum sample size, \( n_{\text{Max}} \), equal to 10.

Overall, the goal in setting values of \( n_{\text{Max}} \) and \( \text{errJ} \) is to use values that get the overall optimization to consistently converge to good solutions with the smallest number of function evaluations.

Note that:

- The risk measure used in this example is \( m_J(1 + c_J) \).
- The values of the optimized objective functions \( f_{\text{opt}} \) shown in the figures are all very close to one another, even for the problem with added noise.
- The noisy objective function takes about ten times as many function evaluations to converge as does the problem without noise. This is related to the value of \( n_{\text{Max}} \) ... 10 in the noisy case and 1 in the case without noise.
- For noisy objective functions, the metric for convergence in \( f \) does not decrease with improved solutions, but hovers around the value of noise in the problem, (the value of \( c_3 \), in this example).
Figure 7. NMAopt convergence on penalized objective function surface with 15% noise in $f(x)$.

$\mathbf{f}_{\text{opt}} = 1.0682e+00$  \hspace{1cm} $\max(g_{\text{opt}}) = -2.8882e-02$

Figure 8. NMAopt convergence histories.
ORSopt.m

```matlab
function [x_opt, f_opt, g_opt, cvg_hst] = ORSopt(func, x_init, x_min, x_max, options, consts)
% ORSopt: Optimized Step Size Randomized Search
% Nonlinear optimization with inequality constraints using Random Search
% minimizes \( f(x) \) such that \( g(x) < 0 \) and \( x_{\text{min}} \leq x_{\text{opt}} \leq x_{\text{max}} \).
% \( f \) is a scalar objective function, \( x \) is a vector of parameters, and \( g \) is a vector of constraints.
% INPUT
% func: the name of the matlab function to be optimized in the form
% \[ \text{[objective, constraints]} = \text{func}(x, consts) \]
% x_init: the vector of initial parameter values ... a column vector
% x_min: minimum permissible values of the parameters, \( x \)
% x_max: maximum permissible values of the parameters, \( x \)
% options: options (1) = 1 means display intermediate results
% options(2) = tol_x tolerance on convergence of parameters
% options(3) = tol_f tolerance on convergence of objective
% options(4) = tol_g tolerance on convergence of constraints
% options(5) = max_evals limit on number of function evaluations
% options(6) = penalty on constraint violations
% options(7) = exponent on constraint violations
% options(8) = max number of function evals in est. of mean \( f(x) \)
% options(9) = desired accuracy of mean \( f \) (as a c.o.v.)
% options(10) = 1 means stop when solution is feasible
% consts: an optional vector of constants to be passed to func(x, consts)
% OUTPUT
% x_opt: a set of parameters at or near the optimal value
% f_opt: the objective associated with the near-optimal parameters
% g_opt: the constraints associated with the near-optimal parameters
% cvg_hst: record of \( x, f, g, \text{function_count}, \text{and convergence criteria} \)
```

NMAopt.m

```matlab
function [x_opt, f_opt, g_opt, cvg_hst] = NMAopt(func, x_init, x_min, x_max, options, consts)
% NMAopt: Nelder-Mead method for the nonlinear optimization with inequality constraints
% minimizes \( f(x) \) such that \( g(x) < 0 \) and \( x_{\text{min}} \leq x_{\text{opt}} \leq x_{\text{max}} \).
% \( f \) is a scalar objective function, \( x \) is a vector of parameters, and \( g \) is a vector of constraints.
% INPUT
% func: the name of the matlab function to be optimized in the form
% \[ \text{[objective, constraints]} = \text{func}(x, consts) \]
% x_init: the vector of initial parameter values ... a column vector
% x_min: minimum permissible values of the parameters, \( x \)
% x_max: maximum permissible values of the parameters, \( x \)
% options: options (1) = 1 means display intermediate results
% options(2) = tol_x tolerance on convergence of parameters
% options(3) = tol_f tolerance on convergence of objective
% options(4) = tol_g tolerance on convergence of constraints
% options(5) = max_evals limit on number of function evaluations
% options(6) = penalty on constraint violations
% options(7) = exponent on constraint violations
% options(8) = max number of function evals in est. of mean \( f(x) \)
% options(9) = desired accuracy of mean \( f \) (as a c.o.v.)
% options(10) = 1 means stop when solution is feasible
% consts: an optional vector of constants that are not design variables
% OUTPUT
% x_opt: a set of parameters at or near the optimal value
% f_opt: the objective associated with the near-optimal parameters
% g_opt: the constraints associated with the near-optimal parameters
% cvg_hst: record of \( x, f, g, \text{function_count}, \text{and convergence criteria} \)
```
Examples of Running Constrained Optimization Codes

SQPopt.m

```matlab
% [x_opt, f_opt, g_opt, cvg_hst, Hess, lambda] = SQPopt(func, x_init, x_min, x_max, options, c)
% SQPopt: Nonlinear optimization with inequality constraints using Sequential Quadratic Programming
% minimizes \( f(x) \) such that \( g(x) \leq 0 \) and \( x_{\text{min}} \leq x_{\text{opt}} \leq x_{\text{max}} \).
% \( f \) is a scalar objective function, \( p \) is a vector of parameters, and \( g \) is a vector of constraints.
% INPUT
% func : the name of the matlab function to be optimized in the form [objective, constraints] = func(x, c)
% x_init : the vector of initial parameter values ... a column vector
% x_min : minimum permissible values of the parameters, \( x \)
% x_max : maximum permissible values of the parameters, \( x \)
% options : options(1) = 1 means display intermediate results
% options(2) = tol_x tolerance on convergence of parameters
% options(3) = tol_f tolerance on convergence of objective
% options(4) = tol_g tolerance on constraint functions
% options(5) = max_evals limit on number of function evaluations
% options(6) = penalty on constraint violations
% options(7) = exponent on constraint violations
% options(8) = number of function averages at each evaluation point
% options(9) = 1 means stop when solution is feasible
% c : an optional vector of constants used by func(x, c)
% OUTPUT
% x_opt : a set of parameters at or near the optimal value
% f_opt : the objective associated with the near-optimal parameters
% g_opt : the constraints associated with the near-optimal parameters
% cvg_hst : record of \( x, f, g, \text{function count}, \text{and convergence criteria} \)
% Hess : the Hessian of the objective function at the optimal point
% lambda : the set of Lagrange multipliers at the active constraints
```

SPSA1opt.m

```matlab
% [x_opt, f_opt, g_opt, cvg_hst] = SPSA1opt(func, x_init, x_min, x_max, options, consts)
% SPSA1opt: Symmetric Perturbation Stochastic Approximation optimization for nonlinear optimization with inequality constraints (first order)
% minimizes \( f(x) \) such that \( g(x) \leq 0 \) and \( x_{\text{min}} \leq x_{\text{opt}} \leq x_{\text{max}} \).
% \( f \) is a scalar objective function, \( x \) is a vector of parameters, and \( g \) is a vector of constraints.
% INPUT
% func : the name of the matlab function to be optimized in the form [objective, constraints] = func(x, consts)
% x_init : the vector of initial parameter values ... a column vector
% x_min : minimum permissible values of the parameters, \( x \)
% x_max : maximum permissible values of the parameters, \( x \)
% options : options(1) = 1 means display intermediate results
% options(2) = tol_x tolerance on convergence of parameters
% options(3) = tol_f tolerance on convergence of objective
% options(4) = tol_g tolerance on constraint functions
% options(5) = max_evals limit on number of function evaluations
% options(6) = penalty on constraint violations
% options(7) = exponent on constraint violations
% options(8) = number of function averages at each evaluation point
% options(9) = 1 means stop when solution is feasible
% consts : an optional vector of constants to be passed to func(x, consts)
% OUTPUT
% x_opt : a set of parameters at or near the optimal value
% f_opt : the objective associated with the near-optimal parameters
% g_opt : the constraints associated with the near-optimal parameters
% cvg_hst : record of \( x, f, g, \text{function count}, \text{and convergence criteria} \)
```