Matlab Optimization

Arnab Sarkar, Sonal Varshney

The MATLAB Optimization Toolbox [1] is a collection of functions that extend the capability of the MATLAB numeric computing environment. The toolbox includes routines for many types of optimization including:

- Unconstrained nonlinear minimization
- Quadratic and linear programming
- Nonlinear least squares and curve-fitting
- Nonlinear system of equation solving
- Constrained linear least squares
- Sparse and structured large-scale problems
- Constrained nonlinear minimization, including goal attainment problems, minimax problems, and semi-infinite minimization problems

1 Optimization functions

The following table lists the types of problems in order of increasing complexity and the associated functions that invoke such an optimization process in MATLAB.

Type	**Function**
Scalar Minimization	`fminbnd`
Unconstrained Minimization	`fminunc` or `fminsearch`
Linear Programming	`linprog`
Quadratic Programming	`quadprog`
Constrained Minimization	`fmincon`
Goal Attainment	`fgoalattain`
Minimax	`fminimax`
Semi-Infinite Minimization	`fseminf`

1.1 Unconstrained Minimization

Consider the problem of finding a set of values $[x_1, x_2]$ that solves

$$\text{minimize x} f(x) = e^{x_1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1) \tag{1}$$

To solve this problem we write an M-file that returns the function value. We then invoke the unconstrained minimization routine `fminunc`.

objfun.m

```
function f = objfun(x)
f = exp(x(1))*(4*x(1)^2+2*x(2)^+4*x(1)*x(2)+2*x(2)+1);
```

Now we invoke the optimization routine

```
x0 = [-1,1]
options = optimset('LargeScale','off');
```

[1] To know more about the MATLAB Optimization Toolbox, visit http://www.mathworks.com/products/optimization/

`[x,fval,exitflag,output] = fminunc('objfun', x0, options);` And we get the result as:
`x = 0.5000 -1.0000`

The functions at the solution `x` is returned in `fval`
`fval = 1.3030e-10`

The exiting flag tells if the algorithm converged. An `exitflag > 0` means a local minimum was found:
`exitflag =1`

1.2 Nonlinear Inequality constrained Minimization

If inequality constraints are added to Equation(1), the resulting problem may be solved by the `fmincon` function. For example, find `x` that solves:

$$\begin{aligned} \text{minimize}_x \ f(x) &= e^{x_1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1) \\ \text{subject to } x_1x_2 - x_1 - x_2 &\leq -1.5 \\ x_1x_2 &\geq -10 \end{aligned}$$

We write confun.m for the constraints

```
function [c, ceq] = confun(x)
c = [1.5 + x(1)*x(2) - x(1) - x(2); - x(1)*x(2) - 10];
ceq = [];
```

We now invoke constrained optimization routine:

```
x0 = [-1, 1]
options = optimset('LargeScale', 'off');
[x,fval] = fmincon('objfun', x0, [],[],[],[],[],[], 'confun',options)
```

The solution `x` produced with function value `fval` is:
`x = -9.5474 1.0474`
`fval = 0.0236`

2 Portfolio Optimization

We will consider the example given by Prof. Shabbir Ahmed (ISyE 6673) [2] and solve it using MATLAB. In short, we will minimize a *quadratic function* subject to some *constraints*. Observe that the quadratic function quantifies the notion of *risk* in the investment problem.

The Mathematical Problem

Indices:
i = Stocks(IBM,WMT,SEHI)
Parameters:
B = Investment budget.
R = Desired expected return.
$\bar{r}_i$ =Expected return of stock i.
σ_{ij} = Covariance of the return of stock i to that of stock j.
Variables: x_i = Amount of money to invest in stock i.

Formulation:

[2] For course details, visit http://www.isye.gatech.edu/∼sahmed/isye6673

$$\min \sum_{i=1}^{3}\sum_{j=1}^{3} x_i x_j \sigma_{ij}$$
$$\text{s.t.} \sum_{i=1}^{3} x_i \leq B,$$
$$\sum_{i=1}^{3} \bar{r}_i x_i \geq R,$$
$$x_i \geq 0, i = 1, 2, 3.$$

Data:
B=1000.00, R = 50.00.

σ_{ij}	IBM	WMT	SEHI
IBM	0.017087987	0.003298885	0.001224849
WMT	0.003298885	0.005900944	0.004488271
SEHI	0.001224849	0.004488271	0.063000818
$\bar{r}_i$	0.026	0.008	0.074

2.1 Solving by Matlab

We solve the quadratic programming problem

$$\min_x \frac{1}{2} x^T H x + f^T x$$
$$\text{s.t.} Ax \leq b$$
$$Aeqx = beq$$
$$lb \leq x \leq ub$$

where H, A, and Aeq are matrices, and f, b, beq, lb, ub, and x are vectors by"

2.2 Running the program

`x = quadprog(H,f,A,b)` returns a vector x that minimizes `1/2*x'*H*x + f'*x` subject to `A*x <= b`. Before running `quadprog` create the matrices H, A and the vectors f, b as shown in the next page.

```
>> x = quadprog(H,f,A,b)
Warning:  Large-scale method does not currently solve this problem formulation,
switching to medium-scale method.

Optimization terminated successfully.

x =

500.0000
0
500.0000
```

From the optimization problem solution, our optimal decision is to invest 500 Dollars in IBM stocks, and 500 Dollars is SEHI stocks. **Observe the warning message** [3] **when quadprog is used. Also, observe the values of x which is a bit different from what is obtained from GAMS (Refer to Appendix for solution of the problem by GAMS).**

[3]The large-scale algorithm is a subspace trust-region method based on the interior-reflective Newton method. Each iteration involves the approximate solution of a large linear system using the method of preconditioned conjugate gradients (PCG).

example.m

Store this program in a file `example.m` and do `>> example` before you do `>> x = quadprog(H,f,A,b)`.

```
for i=1:3
for j=1:3
H(i,j) = 0;
end
end

H(1,1) = 0.017087987;
H(1,2) = 0.003298885;
H(1,3) = 0.001224849;
H(2,1) = 0.003298885;
H(2,2) = 0.005900944;
H(2,3) = 0.004488271;
H(3,1) = 0.001224849;
H(3,2) = 0.004488271;
H(3,3) = 0.063000818;

for i=1:5
for j=1:3
A(i,j) = 0;
end
end

A(1,1) = 1;
A(1,2) = 1;
A(1,3) = 1;
A(2,1) = -0.026;
A(2,2) = -0.008;
A(2,3) = -0.074;
A(3,1) = -1;
A(3,2) = 0;
A(3,3) = 0;
A(4,1) = 0;
A(4,2) = -1;
A(4,3) = 0;
A(5,1) = 0;
A(5,2) = 0;
A(5,3) = -1;

f = [0 0 0];
b = [1000 -50 0 0 0];
b = b';
```

2.3 Functions for different constraint types

- `x = quadprog(H,f,A,b)` returns a vector x that minimizes `1/2*x'*H*x + f'*x` subject to `A*x <= b`.
- `x = quadprog(H,f,A,b,Aeq,beq)` solves the preceding problem while additionally satisfying the equality constraints `Aeq*x = beq`.
- `x = quadprog(H,f,A,b,Aeq,beq,lb,ub)` defines a set of lower and upper bounds on the design variables, x, so that the solution is in the range `lb <= x <= ub`.
- `x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)` sets the starting point to `x0`.

- `x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)` minimizes with the optimization options specified in the structure options. Use optimset to set these options.

3 Large-scale and Medium scale problems

It would be appropriate to mention something more about Large-scale and Medium scale problems at this time since we encountered it in the *WARNING* message in the given example.

The MATLAB toolbox separates "*medium-scale*" algorithms from "*large-scale* algorithms. Medium-scale is not a standard term and is used here only to differentiate these algorithms from large-scale algorithms, which are designed to handle large-scale problems efficiently.

3.1 Medium-scale problems (Trivia!)

The optimization toolbox routines offer a choice of algorithms and line search strategies.

✔ The principal algorithms for unconstrained minimization are the *Nelder-Mead simplex search method* and the *BFGS quasi-Newton method*.
✔ For constrained minimization, *minmax, goal attainment* and *semi-infinite optimization*, variations of *Sequential Quadratic Programming* are used.
✔ Nonlinear least square problems use the *Guass-Newton* and *Levenberg-Marquardt* methods.
✔ A choice of *line* search (or 1-D search) strategy is given for unconstrained minimization and nonlinear least squares problems. The line search strategies use safeguard *cubic interpolation* (with analytical gradient functions) and *quadratic interpolation and extrapolation* (without analytical gradient functions) methods.

3.2 Large-scale problems (An insider's view)

All the large-scale algorithms, except linear programming, are TRUST-REGION methods.

✗ Bound constrained problems are solved using *reflective Newton methods*.
✗ Equality constrained problems are solved using *projective preconditioned conjugate gradient iteration*.
✗ One can use sparse iterative solvers or sparse direct solvers in solving the linear systems to determine the current steps. Some choice of preconditioning in the iterative solvers is also available.

Finally,
The *linear programming* method is a variant of *Mehrotra's predictor-corrector* algorithm, a *primal dual interior point method*.

APPENDIX

Solving by GAMS

We will now solve the Financial Optimization problem by GAMS. *Note the differences in the results.* The basic structure of a *mathematical program* in GAMS requires the programmer to declare the:

- `SETS` of indices over which the problem is defined and assign its members.
- parameters and assign their values using `PARAMETER, TABLE` or `SCALAR`.
- `VARIABLES` and their types.
- `EQUATIONS`.
- `MODEL`.
- provide `SOLVE` statements.
- instruction for `DISPLAY` of solutions.

Input	Output
opti.gms	opti.lst
Sets	Echo print
Data	Reference Maps
Variables	Equation Listings
Equations	Status Reports
Model Specification	Results
Solve Statement	

Input File

We will solve the earlier example by GAMS. *Note that GAMS is case sensitive.* The program is written in the sequence as discussed above.

File opti.gms

```
sets i/1*3/;
alias (i,j);
parameter rbar(i)
/
1 0.0260023
2 0.008100891
3 0.073774971
/;
table sigma(i,i)
1 2 3
1 0.017087987 0.003298885 0.001224849
2 0.003298885 0.005900944 0.004488271
3 0.001224849 0.004488271 0.063000818
;
scalar B /1000.00/
R /50.00/;
positive variables x(i);
free variable z;
equations obj, c1, b1;
obj..  z =e= sum(i, sum(j, sigma(i,j)*x(i)*x(j)));
b1..  sum(i, x(i)) =l= B;
c1..  sum(i, rbar(i)*x(i) ) =g= R;
```

```
model example /all/;
solve example minimizing z using nlp;
display x.l;
```

Output File

Here is the GAMS output file for **opti.gms**. Note that the output file has a `*.lst` extension.

File example.lst:

```
GAMS Rev 120 Windows NT/95/98 01/14/02 07:07:54 PAGE 1
General Algebraic Modeling System
Compilation
1 sets i/1*3/;
2 alias (i,j);
3
4 parameter rbar(i)
5 /
6 1 0.0260023
7 2 0.008100891
8 3 0.073774971
9 /;
10
11 table sigma(i,i)
12 1 2 3
13 1 0.017087987 0.003298885 0.001224849
14 2 0.003298885 0.005900944 0.004488271
15 3 0.001224849 0.004488271 0.063000818
16 ;
17
18 scalar B /1000.00/
19 R /50.00/;
20
21 positive variables x(i)
22 free variable z
23
24 equations obj, c1, b1;
25
26 obj..  z =e= sum(i, sum(j, sigma(i,j)*x(i)*x(j)));
27 b1..  sum(i, x(i)) =l= B;
28 c1..  sum(i, rbar(i)*x(i) ) =g= R;
29
30 model example /all/;
31
32 solve example minimizing z using nlp;
33
34 display x.l;
COMPILATION TIME = 0.000 SECONDS 0.7 Mb WIN198-120
GAMS Rev 120 Windows NT/95/98 01/14/02 07:07:54 PAGE 2
General Algebraic Modeling System
Equation Listing SOLVE example USING NLP FROM LINE 32
---- obj =E=
obj..  (0)*x(1) + (0)*x(2) + (0)*x(3) + z =E= 0 ; (LHS = 0)
---- c1 =G=
c1..  0.026*x(1) + 0.0081*x(2) + 0.0738*x(3) =G= 50 ; (LHS = 0, INFES = 50 ***)
---- b1 =L=
9
b1..  x(1) + x(2) + x(3) =L= 1000 ; (LHS = 0)
```

```
GAMS Rev 120 Windows NT/95/98 01/14/02 07:07:54 PAGE 3
General Algebraic Modeling System
Column Listing SOLVE example USING NLP FROM LINE 32
---- x
x(1)
(.LO, .L, .UP = 0, 0, +INF)
(0) obj
0.026 c1
1 b1
x(2)
(.LO, .L, .UP = 0, 0, +INF)
(0) obj
0.0081 c1
1 b1
x(3)
(.LO, .L, .UP = 0, 0, +INF)
(0) obj
0.0738 c1
1 b1
---- z
z
(.LO, .L, .UP = -INF, 0, +INF)
1 obj
GAMS Rev 120 Windows NT/95/98 01/14/02 07:07:54 PAGE 4
General Algebraic Modeling System
Model Statistics SOLVE example USING NLP FROM LINE 32
MODEL STATISTICS
BLOCKS OF EQUATIONS 3 SINGLE EQUATIONS 3
BLOCKS OF VARIABLES 2 SINGLE VARIABLES 4
NON ZERO ELEMENTS 10 NON LINEAR N-Z 3
DERIVATIVE POOL 11 CONSTANT POOL 14
CODE LENGTH 153
GENERATION TIME = 0.130 SECONDS 1.9 Mb WIN198-120
EXECUTION TIME = 0.130 SECONDS 1.9 Mb WIN198-120
GAMS Rev 120 Windows NT/95/98 01/14/02 07:07:54 PAGE 5
General Algebraic Modeling System
S O L V E S U M M A R Y
MODEL example OBJECTIVE z
TYPE NLP DIRECTION MINIMIZE
SOLVER CONOPT FROM LINE 32
**** SOLVER STATUS 1 NORMAL COMPLETION
**** MODEL STATUS 2 LOCALLY OPTIMAL
**** OBJECTIVE VALUE 20742.0766
10
RESOURCE USAGE, LIMIT 0.170 1000.000
ITERATION COUNT, LIMIT 4 10000
EVALUATION ERRORS 0 0
C O N O P T Windows NT/95/98 version 2.043F-007-043
Copyright (C) ARKI Consulting and Development A/S
Bagsvaerdvej 246 A
DK-2880 Bagsvaerd, Denmark
Using default control program.
** Optimal solution.  There are no superbasic variables.
CONOPT time Total 0.070 seconds
of which:  Function evaluations 0.000 = 0.0
Derivative evaluations 0.000 = 0.0
Work length = 0.05 Mbytes
```

```
Estimate = 0.05 Mbytes
Max used = 0.04 Mbytes
LOWER LEVEL UPPER MARGINAL
---- EQU obj .  .  .  1.000
---- EQU c1 50.000 50.000 +INF 968.646
---- EQU b1 -INF 1000.000 1000.000 -6.948
---- VAR x
LOWER LEVEL UPPER MARGINAL
1 .  497.669 +INF .
2 .  .  +INF 6.894
3 .  502.331 +INF .
LOWER LEVEL UPPER MARGINAL
---- VAR z -INF 20742.077 +INF .
GAMS Rev 120 Windows NT/95/98 01/14/02 07:07:54 PAGE 6
General Algebraic Modeling System
**** REPORT SUMMARY : 0 NONOPT
0 INFEASIBLE
0 UNBOUNDED
0 ERRORS
GAMS Rev 120 Windows NT/95/98 01/14/02 07:07:54 PAGE 7
General Algebraic Modeling System
E x e c u t i o n
---- 34 VARIABLE x.L
1 497.669, 3 502.331
EXECUTION TIME = 0.040 SECONDS 1.4 Mb WIN198-120
```

From the optimization problem solution, our optimal decision is to invest 497.669 Dollars in IBM stocks, and 502.331 Dollars is SEHI stocks. Observe that this result is different from the one obtained by MATLAB as discussed earlier.

References

1. Notes from in ISYE-6673.
2. http://www.isye.gatech.edu/∼sahmed.
3. http://www.me.uvic.ca/∼zdong/courses/mech620/matlab_tutorials.
4. http://www.mathworks.com/products/optimization/