TUTORIAL

MATLAB OPTIMIZATION TOOLBOX

INTRODUCTION

MATLAB is a technical computing environment for high performance numeric computation and visualization. MATLAB integrates numerical analysis, matrix computation, signal processing, and graphics in an easy-to-use environment.

MATLAB also features a family of application-specific solutions -toolboxes-. Toolboxes are collections of MATLAB functions (M-files) that extend the MATLAB environment in order to solve particular classes of problems. Areas in which toolboxes are available include optimization, signal processing, control design, dynamic systems simulation, and so on.

Following is a very succinct description of the basic commands for introduction. Optimization Toolbox, a number of examples, and hands-on information on how to write your own optimization programs are briefed as well.

Since MATLAB uses specific structures and organizes data differently than most common programming languages, it is important that users understand a few fundamentals of the package. The next sections are intended for those who have no or little experience working in a MATLAB environment. If when reading this tutorial intention is only to get specific information about the Optimization Toolbox you may skip the following sections.

MATLAB FUNDAMENTALS.

a. Matrices.

MATLAB works with essentially only one kind of object, a rectangular numerical matrix with possibly complex elements. Special meaning is attached to 1 by 1 matrices -scalars- and to matrices with only one row or column -vectors.

Let’s see now how matrices can be introduced into MATLAB environment. There are several different ways, each one being more suitable for certain applications than the others. So matrices can be entered by an explicit list of elements, or can be generated by built-in statements and functions, or can be created in M-files and finally can be loaded from external data files.

The easiest method of entering small matrices is to use an explicit list. The list of elements is separated by blanks or commas, is surrounded by square brackets ([and]) and uses the semicolon (;) to indicate the ends of rows except the last row.

```
A=[1 2 3; 4 5 6; 7 8 9]
```

output

```
A=
  1 2 3
  4 5 6
  7 8 9
```

A is designed as a 3 by 3 matrix. Without a semicolon at the end of a statement, the results of this statement will be shown right after the statement, otherwise, the results will not be displayed.

On the other hand, matrices can be constructed using smaller-sized matrices as its enties.

```
r=[10 11 12];
A=[A;r]
```

output

```
A=
  1  2  3
  4  5  6
  7  8  9
  10 11 12
```

or, submatrix can be extracted from the matrices using colon character indicating the rows and columns.

```
A=A(1:3,:)
```

will take a submatrix of the first three rows and all columns out of the current A matrix, and then assign the submatrix to A.

```
A=
  1 2 3
  4 5 6
  7 8 9
```

For more information on entering matrices consult the MATLAB User's Guide.

b. Workspace Information.

A list of all variables in use at a certain moment can be obtained by typing

who

Also executing

what

will result in a listing of the M-files and MAT-files in the current directory.

c. Arithmetic Expressions.

Relative accuracy is approximately 16 significant decimal digits and the range is roughly 10 to the power -308 to 10 to the power 308.

Expressions are built up using the common arithmetic operators and precedence rules (+, -, *, ^, /, \). Note that a new operation is defined -left division-. More about this is in a later section.

Special handling is attributed to the operation

s=1/0

output

s=NaN
Warning: Divide by zero.

where NaN stands for Not A Number. The execution will not be stopped, as expected, and the NaNs will propagate throughout the program.

d. Scripts and Functions.

MATLAB works in a command-driven mode; when single-line commands are entered MATLAB processes them immediately and/or displays the results. One use of the M-files is to automate long sequences of commands. Such files are called SCRIPT FILES. A second type of M-file provides extensibility to MATLAB. These are the FUNCTION FILES that allow new functions to be added to the existing functions. Scripts and functions are ordinary ASCII text files, and are created using an editor or word processor of your choice. A script file is invoked by typing its name. Let’s consider that there is a script file whose name is EXAMPLE.m, Then

example

will load the M-file example.m into memory and will execute the commands contained.

A function differs from a script in that arguments may be passed, and variables defined inside the function are local to the function and not global variables in the workspace. A function should begin with the keyword `function' defining the function name.

```
function y=mean(x)
```

e. Matrix Operations.

Apostrophe denotes the transpose of a matrix.

```
B=A';
```

Addition, subtraction and multiplication follow the well-known rules. A special meaning is attached to the matrix division operation. In general

```
x=A\B is solution to A*x=B and
x=B/A is solution to x*A=B
```

More about other matrix operations can be found in the MATLAB User's Guide.

f. Array Operations.

The term array operation is referring to the element-by-element arithmetic operations. Preceding an operator with a period indicates an array operation.

g. Vectors and Matrix Manipulation.

MATLAB allows manipulation of rows, columns, individual elements, and portions of matrices.

Vectors can be generated using the colon character

```
x=1:4
```

output

```
x=
  1 2 3 4
```

When using the format

```
x=1:.5:2
```

output will be

```
x=
  1. 1.5 2.
```

Matrix elements may be referred to by enclosing their subscripts in parentheses. A subscript can be a vector.

```
A(1:5,3)
```

specifies a column vector that consists of the first five elements of the third column of matrix A.

The colon character by itself denotes all of the corresponding rows or columns.

Large-sized matrices may be formed from small matrices by enclosing the small matrices into square brackets ([and]).

```
C=[A zeros(size(A)) ones(size(A))]
```

For more about matrix manipulation see respective section in the MATLAB User's Guide.

OPTIMIZATION TOOLBOX.

The Optimization Toolbox consists of functions that perform minimization/maximization on linear/nonlinear constrained/non-constrained objective functions.

These routines usually require the definitions of the objective functions in M-files. Alternatively, a string variable containing a MATLAB expression, with x representing the independent variables, can be used. Optional arguments to the routines change optimization parameters and place bounds on the variables.

It is recommended that all files specific to the MATLAB environment be kept in a separate directory.

Lets get started with the actual use of the Optimization Toolbox. After creating the new folder that will contain M-files and MAT-files and switching to that directory, MATLAB can be executed by the way

```
start/program/matlab/matlab.exe
```

MATLAB will respond with MATLAB command window,

```
>>
```

and will be waiting for commands.

Of interest is the HELP command that is self-explanatory.

```
>>help help
```

is going to list the uses of help. If you want to get a list of help topics simply type

```
>>help
```

For detailed information about the MATLAB commands and the toolboxes including optimization toolbox, click Help/Help (HTML) in the MATLAB command window, and a detailed user guide can be accessed.

Since tutorial's primary interest lays in the Optimization Toolbox, get more information about the subject by entering

```
>>help optim
```

It can be seen that the help topics are: nonlinear minimization of functions, minimization of matrix problems, controlling defaults and options, and demonstrations. Users are encouraged to go through all these help topics. After getting a good grasp of the contents of the Toolbox, it is of benefit if some time is spent in running the demonstrations provided. To do that one has to type in the name of demonstrations to be viewed:

```
>>tutdemo
```

and afterwards follow the instructions on the screen.

Analyzing `tutdemo' demonstration script one can learn a few things about writing his own optimization programs. Following is a review of some of the important features showed in this demo (observations were made in the order of appearance).

1. Ctrl-C can be used to abort a program;
2. When take a guess at the solution, the more informed is the guess the more accurate the solution;
3. In order to introduce the objective function, a string variable has been used instead of a M-file

   ```
   [x,options]=fminunc(`exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+1)',x0,options);
   ```

   Observe the uses of the variable x. In the LHS x will contain the solution and in the RHS x represent the independent variables.
4. OPTIONS is a vector that has the length 18 elements. For possible values of OPTIONS see the Optimization Toolbox User's Guide.

   options(8) contains the value of the objective function at the solution;
   options(10) contains the number of functions evaluations.

Note that when entering

```
options=[];
```

the optimizer will use default settings. But, for example, if it's found to be necessary that the termination criterion for the independent variables x should be modified then

```
options(2)=precision;
```

will do the trick. Here, precision will actually be a measure of the worst case. Default setting for precision is 1e-4.

5. The following table shows the functions provided in Optimization toolbox for different optimization problems.

Function	**Purpose**
`fgoalattain`	Multiobjective goal attainment
`fminbnd`	Scalar nonlinear minimization with bounds
`fmincon`	Constrained nonlinear minimization
`fminimax`	Minimax optimization
`fminsearch, fminunc`	Unconstrained nonlinear minimization
`fseminf`	Semi-infinite minimization
`linprog`	Linear programming
`quadprog`	Quadratic programming

As mentioned earlier users are strongly recommended to investigate all demos and whenever necessary, reference should be made to the Optimization Toolbox User's Guide and to the MATLAB User's Guide. It can be accessed by Help/Help(HTML)/Optimization Toolbox Ref.

References:

1. MATLAB Reference Guide.
2. Optimization Toolbox for use with MATLAB, User's Guide, November 1990.