Introduction to MATLAB

Introduction to MATLAB

A Tutorial for the Course *Computational Intelligence*

http://www.igi.tugraz.at/lehre/CI

Stefan Häusler
Institute for Theoretical Computer Science
Inffeldgasse 16b/I

Abstract

This tutorial gives a brief introduction to the mathematical software package Matlab. It is basically a condensation of the Matlab User's Guide, as applied to the Unix or Windows operating system. Please consult this guide for more detailed information, although Matlab's on-line help facility should answer the great majority of questions.

Usage

To make full use of this tutorial you have

1. to download the file matlab_intro.zip[1] which contains this tutorial and the accompanying Matlab programs.
2. Unzip matlab_intro.zip which will generate a subdirectory named matlab_intro where you can find all the Matlab programs.
3. Add the path matlab_intro to the matlab search path with a command like
   ```
   addpath('C:\Work\matlab_intro')
   ```
   if you are using a Windows machine or
   ```
   addpath('/home/jack/matlab_intro')
   ```
   if you are using a Unix/Linux machine.

1 Getting Started

Matlab is a high-performance language for technical computing. Although MATLAB stands for MATrix LABoratory, it is well suited to handle most mathematical needs, not just matrix manipulation. It is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

Furthermore Matlab features a family of add-on application-specific solutions called toolboxes. Toolboxes are comprehensive collections of Matlab functions that extend the Matlab environment to solve particular classes of problems like signal processing, control systems, neural networks, and many others.

[1] http://www.igi.tugraz.at/lehre/CI/tutorials/matlab_intro.zip

Starting and Quiting Matlab

On Windows platforms double-click the Matlab shortcut icon on your Windows desktop. On UNIX platforms type matlab at the operating system prompt. Once running, Matlab will present a pair of greater-than sybols, >> , as its command-line prompt.

To end your Matlab session, type quit or exit in this prompt. To execute specified functions each time Matlab quits, such as saving the workspace, you can create and run a finish.m script.

2 Manipulating Matrices

Matlab works with essentially one kind of object: a matrix of numbers (which could include complex elements). *Scalars* are 1-by-1 matrices, while vectores are 1-by-n or n-by-1 matrices.

When entering a matrix, seperate columns by spaces or commas and rows by semicolons, e.g.:

```
>> P = [3 5 -6 1; -2 5 4 9]

P =
    3     5    -6     1
   -2     5     4     9
```

Matlab stores the above 2-by-4 matrix in its workspace for later use. To retrieve a variable, simply type its name (e.g., P). Matlab is case-sensitive, so p and P are not the same.

When you do not specify an output variable, Matlab uses the variable ans, short for answer, to store the results of a calculation.

Including a semicolon (;) at the end of a command suppresses Matlab's echoing to the teminal (this is useful when dealing with large sets of numbers.)

To take the sum along the columns of P type

```
>> sum(P)

ans =
     1    10    -2    10
```

Matlab has a preference for working with the columns of a matrix, so the easiest way to get the row sums is to transpose the matrix with the transpose operator .', compute the column sums of the transpose, and then transpose the result:

```
>> sum(P.').';
```

In contrast single quote operator ' conjugates and transposes matrices (*Hermitian* operator).

Subscripts

The element in row i and column j of P is denoted by P(i,j). If you try to use the value of an element outside of the matrix, it is an error.

```
>> t = P(4,5)
Index exceeds matrix dimensions.
```

On the other hand, if you store a value in an element outside of the matrix, the size increases to accommodate the newcomer.

```
>> X = P; X(1,5) = 17
```

```
X =
     3     5    -6     1    17
    -2     5     4     9     0
```

It is also possible to refer to the elements of a matrix with a single subscript, P(5). In this case the array is regarded as one long column vector formed from the columns of the original matrix.

```
>> P(5)

ans =
    -6
```

Expressions

Matlab provides mathematical expressions that involve entire matrices. The building blocks of expressions are *variables*, *numbers*, *operators* and *functions*.

Matlab does not require any type declarations or dimension statements for *variables*. Variable names consist of a letter, followed by any number of letters, digits, or underscores. Matlab uses only the first 31 characters of a variable name.

Expressions use familiar arithmetic *operators* and precedence rules: + addition, - subtraction, * multiplication, / division, \ left division (described in Matrices and Linear Algebra in the MATLAB documentation), ^ power, ' complex conjugate transpose and ( ) specify evaluation order. The mathematical operations defined on matrices are the subject of linear algebra. The multiplication symbol, *, denotes the matrix multiplication involving inner products between rows and columns:

```
>> w = [0.5 1]; h = w*P

h =
   -0.5000    7.5000    1.0000    9.5000
```

Matlab uses conventional decimal notation, with an optional decimal point and leading plus or minus sign, for *numbers*. Scientific notation uses the letter e to specify a power-of-ten scale factor. Imaginary numbers use either i or j as a suffix. All numbers are stored internally using the long format specified by the IEEE floating-point standard.

Matlab provides a large number of standard elementary mathematical *functions* as well as more advanced mathematical functions, including Bessel and gamma functions. Some of the functions, like sqrt and sin or

pi	3.14159265...
i	Imaginary unit, $\sqrt{-1}$
j	Same as i
Inf	Infinity
NaN	Not-a-number

are built in. Other functions, like gamma, sinh or hardlim, are implemented in M-files. You can see the code, e.g. by typing

```
>> type hardlim
```

and even modify it if you want.

3 Programming with Matlab

Flow Control

if

As usual the if statement evaluates a logical expression and executes a group of statements when the expression is true. The optional elseif and else keywords provide for the execution of alternate groups of statements. An end keyword terminates the last group of statements. The groups of statements are delineated by the four keywords no braces or brackets are involved.

```
>> T = [1 1 0 0];
>> if T == 1
     disp('Vector of ones.')
   elseif T == 0
     disp('Vector of zeros.')
   else
     disp('Vector of ones and zeros.')
   end

Vector of ones and zeros.
```

Note that for two matrices A and B , A == B does not test if they are equal, it tests where they are equal; the result is another matrix of 0's and 1's showing element-by-element equality. Only if T == 1 is true for all elements the if clause is executed.

switch and case

The switch statement executes groups of statements based on the value of a variable or expression. The keywords case and otherwise delineate the groups. Only the first matching case is executed.

```
>> method = 'Bilinear';
>> switch method
     case {'linear','bilinear'}
       disp('Method is linear')
     case 'cubic'
       disp('Method is cubic')
     otherwise
       disp('Unknown method.')
   end

Method is linear
```

for and while

The for loop repeats a group of statements a fixed, predetermined number of times. A matching end delineates the statements.

```
>> b = 0; y = hardlim(w*P+b);
>> for i=1:4
     if y(i) == 1
        disp(h(i));
     end
   end
```

The while loop repeats a group of statements an indefinite number of times under control of a logical condition. A matching end delineates the statements.

continue and break

The continue statement passes control to the next iteration of the for or while loop in which it appears, skipping any remaining statements in the body of the loop. In nested loops, continue passes control to the next iteration of the for or while loop enclosing it.

The break statement lets you exit early from a for or while loop. In nested loops, break exits from the innermost loop only.

Scripts and Functions

Files that contain code in the Matlab language are called M-files. You create M-files using a text editor, then use them as you would any other Matlab function or command. You can organize them into other directories and personal toolboxes that you can add to your Matlab search path. There are two kinds of M-files:

scripts

Scripts do not accept input arguments or return output arguments. They can operate on existing data in the workspace, or they can create new data on which to operate. Although scripts do not return output arguments, any variables that they create remain in the workspace, to be used in subsequent computations. An example is the M-file matlab_intro.m.

functions

Functions, which can accept input arguments and return output arguments. Functions operate on variables within their own workspace, separate from the workspace you access at the Matlab command prompt. As example see the file perc.m.

```
function w=perc(P,T)
% PERC(P,T) Train a threshold unit using the
%    Delta-Rule until a separating Hyperplane is found.

figure(1); clf;
...
```

The first line of a function M-file starts with the keyword function. It gives the function name and order of arguments. In this case, there are up to two input arguments and one output argument. The next several lines, up to the first blank or executable line, are comment lines that provide the help text. These lines are printed when you type

```
>>  help perc
```

The first line of the help text is the H1 line, which Matlab displays when you use the lookfor command or request help on a directory.

An aspect of Matlab functions that is not ordinarily found in other programming languages is a variable number of arguments. Not all listed input arguments have to specified during a function call, e.g.

```
>> r = rank(A);
>> r = rank(A,1.e-6);
```

are both valid commands. Within the body of the function, two quantities named nargin and nargout are available which tell you the number of input and output arguments involved in each particular use of the function.

If you want more than one function to share a single copy of a variable, simply declare the variable as global in all the functions, e.g. by typing

```
>> global P T
```

Do the same thing at the command line if you want the base workspace to access the variable. The global declaration must occur before the variable is actually used in a function.

4 On-line Help

The command help, by itself, lists all primary help topics. Each primary topic corresponds to a directory name on the Matlab path.

```
>> help topic
```

gives help on the specified topic. The topic can be a command name, a directory name, or a Matlab path relative partial pathname. If it is a command name, help displays information on that command. If it is a directory name, help displays the Table-Of-Contents for the specified directory.

```
>> helpwin topic
```

does the same as help but displays the help text for the specified topic inside a desktop help window.

```
>> helpdesk
```

displays the start page of a comprehensive hypertext documentation and troubleshooting in the Matlab Help browser

5 More About Matrices and Arrays

The Colon Operator

The colon, :, is one of the most important Matlab operators. It occurs in several different forms. The expression

```
>> 1:10

ans =
     1     2     3     4     5     6     7     8     9    10
```

is a row vector containing the integers from 1 to 10. To obtain nonunit spacing, specify an increment. For example,

```
>> 20:-2:0

ans =
    20    18    16    14    12    10     8     6     4     2     0
```

Subscript expressions involving colons refer to portions of a matrix.

```
>> P(1,3:4)

ans =
    -6     1
```

or

```
>> P(1,:)

ans =
     3     5    -6     1
```

The colon by itself refers to all the elements in a row or column of a matrix and the keyword end refers to the last row or column. As index also other matrices could be used, e.g.

```
>> P(:,[1 3 4])

ans =
     3    -6     1
    -2     4     9
```

or

```
>> P(:,randperm(4))

ans =
     1    -6     3     5
     9     4    -2     5
```

The logical vectors created from logical and relational operations can be used to reference subarrays. For example

```
>> P(:,y)

ans =
     5    -6     1
     5     4     9
```

specifies the elements of P where the elements of y are nonzero and

```
>> P(P>1)'

ans =
     3     5     5     4     9
```

displays all positive elements of P as a row vector.

Concatenation and Deletion

Concatenation is the process of joining small matrices to make bigger ones. In fact, you made your first matrix by concatenating its individual elements. The pair of square brackets, [], is the concatenation operator.

```
>> P1 = [3 5 1 0.1; 8 4 2.2 0];
>> [P1 P; P P1]
```

To delete for instance a column of a matrix, use

```
>> X = P;
>> X(:,2) = []

X =
     3    -6     1
    -2     4     9
```

and the remaining elements are shaped into a smaler matrix.

Array Operations

Arithmetic operations on arrays are done element-by-element. Matlab uses a decimal point as part of the notation for multiplicative array operations. The list of operators includes multiplication

```
>> P.*P1

ans =
    9.0000   25.0000   -6.0000    0.1000
  -16.0000   20.0000    8.8000         0
```

division

```
>> P./P1

ans =
    1.0000    1.0000   -6.0000   10.0000
   -0.2500    1.2500    1.8182       Inf
```

or array power

```
>> P.^P1

ans =
   1.0e+03 *
    0.0270    3.1250   -0.0060    0.0010
    0.2560    0.6250    0.0211    0.0010
```

as well as left division .\ and unconjugated array transpose .'.

Special Matrices

Matlab provides five functions that generate basic matrices.

zeros	All zeros
ones	All ones
rand	Uniformly distributed random elements
randn	Normally distributed random elements
eye	Identity matrix

Workspace

The commands, who and whos, show the current contents of the workspace. The who command gives a short list, while whos also gives size and storage information.

To delete all the existing variables form the workspace, enter

```
>> clear all
```

The save command preserve the contents of the workspace in a MAT-file that can be read withe the load command in a later Matlab session. For example

```
>> save August17th
```

save the entire workspace contents in the file August17th.mat. If desired, you can save only certain variables by specifying the variable names after the filename.

6 Creating Graphs

To open a new figure with handle 1 type

```
>> figure(1); clf;
```

whereas 1 clears the current figure. For this demonstration please load the variables C and X form the file winedata.mat

```
>> load winedata X C
```

X is a 2 dimensional matrix, where each column specifies a point in the 2 dimensional plane. The corresponding element of the vector C defines to which class a point belongs. To plot the point sets of all three classes in different styles type

```
hold on
plot(X(C==1,1),X(C==1,2),'ro');
plot(X(C==2,1),X(C==2,2),'bs');
plot(X(C==3,1),X(C==3,2),'g*');
```

The command hold on holds the current plot and all axis properties so that subsequent graphing commands add to the existing graph. To label the axis use

```
xlabel('Markmal 1');
ylabel('Markmal 2');
```

Additional

```
legend('Klasse 1','Klasse 2','Klasse 3');
```

puts a legend on the current plot using the specified strings as labels. To see further styles of 2-D and 3-D plots start the Matlab demos

```
>> graf2d
>> graf3d
```

7 Other Data Structures

Mulitdimendional Arrays

Multidimensional arrays in Matlab are arrays with more than two subscripts. They can be created by calling zeros, ones, rand, or randn with more than two arguments, e.g.

```
>> R = rand(2,3,2);
```

or directly by

```
>> R(2,4,1) = 2

R(:,:,1) =
    0.7843    0.0310    0.5586         0
    0.3879    0.5855    0.2007    2.0000

R(:,:,2) =
    0.0874    0.2594    0.0492         0
    0.9332    0.2042    0.6062         0
```

Indexing works straight forward

```
>> R(2,1,1)

ans =
    0.3879
```

Character and Text

Enter text into Matlab using single quotes. For example,

```
>> S = 'Hello World!';
```

It is a 1-by-12 character array. For the conversion between character and numeric array use

```
>> a = double(S);
```

and

```
>> S = char(a);
```

For the conversion from numbers to strings use

```
>> S2 = int2str(1024);

S2 =
1024
```

Further

```
>> S2 = char(S,S2)

S2 =
Hello World!
1024
```

produces a 2-by-12 character array, where blanks are added to each line to make them all the same length.

Cell Arrays

Cell arrays in Matlab are multidimensional arrays whose elements are copies of other arrays. Cell arrays are created by enclosing a miscellaneous collection of things in curly braces, {}:

```
>> C = {R -1 S rand(2,2)}

C =
    [2x4x2 double]    [-1]    'Hello World!'    [2x2 double]
```

To retrieve the contents of one of the cells, use subscripts in curly braces.

```
>> C{3}

ans =
Hello World!
```

Like numeric arrays also cell arrays can be mulitdimensional

```
>> CM{2,2} = S;
>> CM{1,1} = 'hallo'

CM =
    'hallo'                []
         []    'Hello World!'
```

Structures

Structures are multidimensional Matlab arrays with elements accessed by textual field designators. For example,

```
>> A.name = 'Rudi';
>> A.score = [ 2 3 4];
>> A.fancy = rand;
>> A(1)

A =
     name: 'Rudi'
    score: [2 3 4]
    fancy: 0.0664
```

Like everything else in Matlab, structures are arrays, so you can insert additional elements.

```
>> A(2) = A(1);
>> A(2).name = 'Fredi';
```