BE.420 MATLAB Tutorial

Originally written by Nate Tedford

Since modified by generations of TAs!

Part I

Syntax and Basic Use

How to Start and Run MATLAB

- On a Mac or PC, run as you would any other program..just point and click

- All 12 PC's in the building 26 computer lab have MATLAB 6.5 (and Solver!!) now installed

- Common commands include:
  - -h|-help - display usage message
  - -n - print environment variables only
  - -display Xserver - set display variable
  - -Ddebugger [options] - Startup matlab with debugger
  - -tty - start Matlab in current window
  - -msg - force redisplay of current start message
  - directory_list - Adds each directory in list to MATLAB search path if it is an accessible directory
  - xterm args - all valid xterm arguments can be used to control appearance of Matlab command window

MATLAB Helpdesk

- At the MATLAB prompt, type:
  - helpdesk

➢ This will gives you a searchable command help index which is toolbox specific and more similar to the help resources that you will see in the PC version 6.xx

MATLAB Basics

Functions:

- Matrix formulation: fast calculation
- Strong in numerical, but weak in analytical
- General purpose math solvers: nonlinear equations, ODEs, PDEs, optimization

Basic mode of usage:

- Interactive mode
- Permanent MATLAB files (M-files)
  - M-script
  - Functions

➢ M-script and Functions must be written in separate files

- Note: M-files are saved in “Work” folder in the MATLAB program files subdirectory

Basic Syntax

- Case sensitive variable name
- Library of Reserved Words
  - These will appear in blue if you are writing your code as an M-file
- Put a semi-colon at the end of each line to prevent output from displaying in command window
- Assigning variables
  - Simple Variable
    - A = 4;
  - Vector
    - A = [1 2 3 4]; or A = [1, 2, 3, 4];
  - Matrix
    - A = [1 2; 3 4]; or A = [1, 2; 3, 4];

Basic Syntax Continued

- Shortcuts to creating “regular” vectors
  - Z = (1:5) means Z = [1 2 3 4 5]
  - Z = (1:3:10) means Z = [1 4 7 10]
  - Z = linspace(a, b, n) creates a vector with n evenly spaced points between a and b
  - Z = logspace(a, b, n) creates a vector with n logarithmically spaced points between 10^a and 10^b

Matrix Manipulation

- Use the ’ symbol to transpose vectors or matrices
  - A = [1 2 3 4]’ makes A a column vector
- Let’s say A = [1 2 3]; B = [4 5 6];
  - C = [A B] means C = [1 2 3 4 5 6]
  - D = [A; B] means $D = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
  - E = D(1,2) means E = 2
  - F = D(2, 2:3) means F = [5 6]
  - G = D(:, 3) means the 3rd column of D, i.e. $G = \begin{bmatrix} 3 \\ 6 \end{bmatrix}$

Operators

- Matlab uses matrix-oriented operators
  - “*” and “/” are for matrix multiplication and division and the dimensions of the matrices must be compatible
  - A^2 means the cross product of matrix A with itself, and A must be a square matrix
  - The above also work with 1x1 matrices i.e. numbers
- To do element-by-element operations, you need to add a “.” before the operator
  - [1 2 3].*[4 5 6] gives [4 10 18]
  - [1 2 3].^3 gives [1 8 27] but [1 2 3]^3 gives an error
  - Forgetting the dot is one of the most common bugs!
  - Don’t use the dot for addition or subtraction

Two Important Points

- If you do not put a semi-colon at the end of the line, the result of the operation for that line will be displayed when your program is run. This can slow down your program and/or upset your TA.
- Assignment vs. Equals: Important in Loops!
  - Assignment: a = b
  - Equals: a == b (results in 1 or yes if they are equal, 0 or no if they are not)

Looping in MATLAB

- For loops are used to repeat statements a known number of times

```
for I = 1:N
        for J = 1:N
                A(I,J) = 1/(I+J-1)
        end
end
```

- While loops are useful when you want to repeat statements until a condition is met

```
I = 1; N = 1;
while I <= 100
        I = I + rand;
        N = N + 1;
end
```

Looping Continued…

- If and else statements

```
if I == J
        A(I,J) = 2;
elseif abs(I-J) == 1
        A(I,J) = -1;
else
        A(I,J) = 0;
end
```

- Keep careful track of the number of “end”s in nested loops
- The usual relational and logical operators can be used
  - > greater than, < less than, >= greater than or equal, <= less than or equal, == equal, ~= not equal
  - & AND, | OR, ~ NOT

Basic MATLAB Commands

Matlab commands	Functions and descriptions
help *functionname*	Matlab on-line help for functions
lookfor *searchphrase*	To find matlab function with descriptions containing the search phrase
who	To list all variables currently used
size(*matrix*)	To identify the dimensionality of the *matrix* (use length(*vector*) for a vector)
ones(*m*,*n*)	To create a unit matrix of size m x n
print –depsc *filename.ps*	To print an active plot

Importing Data

- TEXTREAD, DLMREAD
  - DLMREAD read ASCII delimited file.
    - M = dlmread(FILENAME,DLM)
  - TEXTREAD read text file in a certain format
    - [A B] = textread(‘datafile','%f %f');

Comments

- You can write comments between and after lines of code by typing “%” in front of your message
- You shold write your name and assignment info on top of each program
- Lastly, use comments throughout the code to show me that you know what you’re doing

Part II

Solving Ordinary Differential Equations

Numerical Solution of ODE's

- There are a number of preprogrammed functions that you can use to solve your ODE or system of ODE's in MATLAB:
  - ODE23, ODE45, ODE113, ODE15S, ODE23S
- ODE45: Runge Kutta (4,5) formula, best first try function
- Many biological systems are "stiff" and ODE45 will take a long time or give obviously wrong answers. Try ODE15S or ODE23S instead (S is for stiff).

Solving the ODE (single ODE)

- First you need to write a function of the form:
  - function dxdt = functionname(t,x,parameter 1,parameter 2,etc.)
- Then write your differential equation:
  - dxdt = some function of: x, parameters; (end with a semicolon)
- Save this function as a separate M-file
  - File must have the same name as the functionname above

Solving the ODE, Continued…

- Now you must call your function in your main MATLAB program, type:
  - [t,X]=ode45(@functionname,t,IC,options,parameter1, parameter 2, etc.);
    - IC is the initial condition, you must assign IC, or whatever name you choose for it, a value in your program file before you run the above function call
    - For options, normally type in empty brackets: []
    - t is your time vector that you need to define previously
    - X is the solution vector for your unknown

Solving a System of ODE's

- Now the function is set up a bit differently and you will need to call the function and get separate solutions for each unknown from a solution matrix (i.e. use the semicolon in one of your matrix indices, such as X2=X(:,2))

Solving a System of ODE's, Cont.

- Now the function will take the form:
  - function fun = ligand(t,Y,k1,k2,k3,k4,….)

  - fun(1,:)=krs*Y(2)-kfs*Y(1)*Y(3)-kfp*Y(1); %dCls/dt
  - fun(2,:)=kfs*Y(1)*Y(3)-krs*Y(2)+krec*Y(5)-kec*Y(2); %dCcs/dt
  - fun(3,:)=krs*Y(2)-kfs*Y(1)*Y(3); %dCrs/dt
  - fun(4,:)=kfp*Y(1)+kri*Y(5)-kfi*Y(4)*Y(6)-kdeg*Y(4); %dCli/dt
  - fun(5,:)=kec*Y(2)+kfi*Y(4)*Y(6)-kri*Y(5)-krec*Y(5); %dCci/dt
  - fun(6,:)=kri*Y(5)-kfi*Y(4)*Y(6); %dCri/dt
  - fun(7,:)=-kdeg*Y(4); %dCLtot/dt

Plotting Your Data

- After you have called your function (and assigned a variable name to the sol'n)
  - Type: figure (don't need a semicolon here)
  - Type: plot(t,X)
    - t is your time vector and X is the sol'n vector that you named in your function call or part of your sol'n matrix (i.e. X(:,1), first column of matrix)
  - Note: You can type semilogx(t,X) or semilogy(t,X) to get a semilog plot of your choosing

Plotting Your Data Continued..

- Typing “hold on” after introducing a second figure will allow you to plot multiple curves on the same set of axes
- Using the “subplot(x,y,z), plot(t,X)” sequence will allow you to plot a matrix of graphs of size (x,y) on the same page, with z being the location of the graph in the matrix
- Typing “plot(t,X,’*letter*’)” will allow you to control the color of the line for that plot, type ‘help plot” in prompt to see the color key

Labeling Axes,Making Legends

- For the plot title, x-axis, and y-axis, type:
  - title(‘*title*’)
  - xlabel(‘*axis name*’)
  - ylabel*(‘axis name’)*
- To make a legend, type:
  - legend*(‘name of curve 1,’name of curve 2’, etc.*)
- You can include variable values in these labels
  - title([‘Binding isotherm with k = ’ num2str(k) ‘ s^-1’])
- If you are out of time, you can also use the “Insert text” button (with an “A” on it) to label your plots just before you print them. However, these labels will not appear when you rerun the m-file.

Saving Your Work

- In the MATLAB prompt, type:
  - save *filename*
- In Windows, just use the save icon or the save option in the drop down menu under file
- Make sure that your file is saved in the proper directory so that it can run from the MATLAB prompt
  - In Windows, it is normally the “Work” folder

Running Your Saved Work

- Type the name of the M-file in the Matlab prompt and hit enter
  - Also make sure that any functions that your program uses are in the same directory as this main M-file
- If there are any errors in your code, they will show up as messages in red text in the prompt window

Other Useful Functions in MATLAB

You will mainly use the ODE solving functions, but the following functions may come in useful for some of the implementations:

- NLINFIT
  - Allows you to do nonlinear curve-fitting
- FSOLVE
  - Allows you to solve for unknowns in an algebraic equation or in systems of algebraic equations
- Use the “help functionname” command to see the proper syntax for setting up these types of problems

Part III

HELP!!!

Some advice on getting help…

- USE THE HELP SEARCH TOOL
  - In MATLAB 6.xx, type:
    - help *functionname*
- Debug carefully
  - Write your code a little at a time
  - Use flags to see where errors are
- If debugging is going nowhere, ask a friend to check things out
- If things are still stuck, come see the TAs

MIT Help

- Go to:

http://web.mit.edu/answers/www/matlab/

- The Copy Tech also has printouts of basic MATLAB commands and operations, you can pick up a copy for free there

If you have a Pentium 4 and you have MATLAB Version 6.0……

- Go to:

http://www.mathworks.com/

- Search for Pentium 4, Matlab version 6.0, and you'll be directed to a link that gives you instructions to fix everything.