AMS 27L **LAB #2** **Winter 2009**

Plots and Matrix Algebra in MATLAB

Objectives:

1. To practice basic display methods
2. To learn how to program loops
3. To learn how to write *m*-files

---

1 Vectors

Matlab handles vectors and matrices. The notation for a row vector, x =(1,2,3,4) is:

```
>>  x=[1,2,3,4]
x=    1   2   3   4
```

(so the commas separate elements in the row, and Matlab writes the row vector with spaces between elements). For the column vector

$$y = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

the notation is

```
>>  y=[1;2;3;4]
```

and Matlab puts the rows on separate lines.
Amazingly, Matlab can handle arithmetic defined on the elements of the vector. For example, let's take the row vector $x = (1, 2, 3, 4)$ and multiply every element by 5:

```
>>  5 * x
ans = 5   10   15   20
```

The output is, of course, another row vector with the appropriate elements. Now suppose we wanted to square each element of x:

```
>>  x.^2
ans =  1   4   9   16
```

All you need to do is *a*dd the period before the arithmetic operation, so that Matlab knows that it must accomplish the operation element-by-element. With this artifice, we can also multiply the elements of two vectors together if they have the same length:

```
>> z=[4,3,2,1]; x.* z
ans =  4  6  6  4
```

Remember the extra period. Many times, you will not get the right answer or Matlab will complain about your program because you forgot a period somewhere. The period is not necessary when the algebraic operation on the corresponding elements is identical to the vector algebra; that is, when adding or subtracting vectors, we need not add the period:

```
>> x + z, x - z
ans = 5  5  5  5
ans = -3  -1  1  3
```

We can even formulate functions of each element by simple operations on the vector as a whole: the exponentiation of each element is obtained by:

```
>> exp(x)
ans = 5  5  5  5
ans                                =
  2.7183  7.3891  20.0855  54.5982
```

A shorthand way to generate the vector x is:

```
>> x=[1 : 1 : 4]
ans =  1  2  3  4
```

The notation $p : n : q$ means proceed from p to q in steps of n. The row vector whose elements are the even integers from 2 to 1000 is:

```
>> x =[2 : 2 : 1000]
```

At this stage, it is wise to add the semi-colon. We may rescale this vector so that the largest element is 1 by writing:

```
>> x=x/1000;
```

This divides each element by 1000.

2 Plots

We are ready for some simple plotting. Let $y = \sin(2\pi x)$, where x is the long row vector constructed above. We may plot x against y with the simple plot command:

```
>> y=sin(2 * pi * x);
>> plot(x,y)
```

We can plot more than one curve at the same time:

```
>> plot(x,y,x,cos(2 * pi * x))
```

Note how this plots a function directly, without first creating a new vector like y. We can even change the style of the curves:

```
>> plot(x,sin(2 * pi * x. * x),'--',x,cos(2 * pi * x. * x),':')
```

which plots the sine curve with a dashed line and the cosine curve with a dotted one. More plotting styles can be found by executing "help plot".

Problem #1: Try plotting the graphs above and check the results.

Problem #2: Graph the function, $f(x) = x\, e^{-x^2}$ over the interval $[-1, 2]$. Use the command "grid on" to show the graph.

Problem #3: Type in the following commands and press `<Enter>` to see the result:

(a) x=linspace(0,2 ∗ pi, 30)

(b) y= sin(x)

(c) plot(x,y) *Create a graph.*

(d) plot(x, y, '.') *Create graph with data points.*

(e) plot(x, y, 'ro') *Create graph with red data circles.*

(f) plot(x, y, 'g-') *Create graph with a green line.*

(g) plot(x, y, 'b:') *Create graph with a blue dots.*

(h) plot(x, y, 'k--') *Create graph with dashes.*

(i) plot(x, y, 'r+','markersize',16) *Make the data points larger.*

(j) plot(x,y,'r–','linewidth',5) *Increase the line width.*

(k) hold on *You can overlap graphs.*

(l) plot (x, y, 'bx','markersize',14) *You can change the data size.*

(m) hold off *Reset the plotting area.*

(n) close

Now let's get a little fancy by adding these lines:

(o) plot(x, y, 'o','markersize',12, 'markerfacecolor','g','markeredgecolor','r')

(p) title('My Graph') *Adds title.*

(q) title('My Graph','fontsize',16,'fontname','times') *Controls appearance.*

(r) axis([0, $2 * \pi$, $-1.5, 1.5$]) *Rescale axes.*

(s) xlabel('\theta','fontsize',16,'color','g')

(t) ylabel('sin(\theta)','fontsize',16,'color',[.8, .6, .3]) *Colors can be fine-tuned with 3-vector codes.*

(u) text(3.2, .1, \'leftarrow sin(\pi)+=0!', 'fontsize', 14, 'color','b','fontname','times')
You can add comments

Problem #4: While doing some physics homework, you find that a capacitor discharges exponentially with $Q = Q_0 \exp(-t/\tau)$, with $Q_o = 3$ coulombs and $\tau = 2$ seconds. Create a graph demonstrating this relationship from $t = 0$ to $t = 3\tau$.

Problem #5: Plot the function $3x^3 - 2x$ against x for $-1 < x < 1$. Estimate the position of any extremal points from the picture using the data cursor. Compute these exactly by differentiation.
Note : Graphs are built upon the axes. If you need to rescale the axes, you have to do this before the `plot` command.
Note : To access the data cursor, select "Data Cursor" in the "Tools" scroll down menu. Find the data values of any point on a graph by clicking the mouse on the graph location. Click and hold to move the data cursor along the graph. Right-click to add additional data cursors to the graph.

Problem #6 Find the value of x that satisfies the equation $sinh\{20\ ln[tan(\sqrt{x})\} = 2$.
Hint : To begin, set a variable, say "y", to the *sinh* function. Then plot the corresponding xy-graph to find the value of x when the function is 2. Be sure to discretize the graph finely so that your estimate of x is good! Next, work with the function inside the *sinh* function to work your way to the solution.

3 Introduction to Script Files

Matlab scripts are called M-files and they are the programs you make to run in Matlab. They are simple text files that can also be read by editors like MSWord. Matlab has a built in editor that is very useful for debugging.

Problem #7: *Type the following commands into the command line and press* `<Enter>` *to see the result.*

(a) Use a file manager to create a directory called something like *'matlab_prog'* on your thumb drive or your hard drive.

(b) Change the Matlab path to the directory (look for [...] button or use "`cd`")

(c) Open a new M-file with "`file > new > M-file`", or use "*edit filename*"

(d) Save this (initially blank) file as "*intro_script.m*"

(e) Program the following script line-by-line into your M-file:

```
close all;
clear all;
clc;
format short;
x = linspace(-7,7,100000);
y=sin(x)./x
plot(x,y);
return;
```

Now hit 'F5'. This will simultaneously run the script and resave the script. Note that the semicolons are necessary because otherwise the output prints (painfully slowly) to the screen. Try removing the semicolon from `x=linspace(-7,7,100000)` and re-run the script. Way too many print-outs! Yikes! The 'return' statement is not necessary here, but can be very handy to know when you want to terminate the script manually for some reason.

Example: Plotting and m-files. In one figure, plot the solutions to the differential equation $y' = y + t$ with initial conditions $y(0) = -2, -1, 0, 1, 2$ over the interval $0 \leq t \leq 2$.

The general solution to the differential equation is $y(t) = Ce^t - t - 1$. The initial condition is $y(0) = C - 1$, so the constant satisfies $C = y(0) + 1$. The graphs can be drawn using the techniques we have already discussed, but it is easy to make mistakes in executing all of the commands needed. Instead we will create a script M-file. To see how this is done, choose the menu item **File** → **New** → **M-file**. The built-in MATLAB editor will open at a blank page. You can also call up the editor by executing the command `edit`. We will let MATLAB perform the simple arithmetic to compute the constant in addition to plotting the solutions. Enter the following list of commands into the blank editor page:

```
t=0:0.05:2;
C = -2+1; plot(t,C*exp(t) - t - 1 ‘,’-’)
hold on
C = -1 + 1; plot(t,C*exp(t) - t - 1,’-.’)
C = 0 + 1; plot(t,C*exp(t) - t - 1,’--’)
C = 1 + 1; plot(t,C*exp(t) - t - 1,’.’)
C = 2 + 1; plot(t,C*exp(t) - t - 1,’:’)
grid on
xlabel(’t’)
ylabel(’y’)
title(’Solutions to y”=y+t.’)
legend(’y(0) = -2’,’y(0) = -1’,’y(0) = 0’,’y(0) = 1’,’y(0) = 2’)
shg, hold off
```

Save the file with a meaningful filename. Now whenever you execute the command `filename` at the command line, all of these commands are executed and the figure appears.

Problem #8 Write and execute the m-file lines above. **Save your *m*-file for further reference!** We will use it again in the next lab.

---

4 Programming Loops

In some calculations it is useful to do things recursively. We may repeat commands by placing them in a loop; the construction is as follows. We choose a loop variable n and then increment n from 1 to N:

```
>> for n=1:N
.... “blah, blah” ....
end
```

“blah, blah” denotes the commands we want to repeat N times, and “end” terminates the loop. The “blah, blah” commands can be just about anything you want. For example, let us say we want to plot the function $f(x) = 1 - ax^2$ a number of times, each time with a different value for the parameter a. Using ‘for’ we can do it as follows:

```
>> x=[0:100]/100;
>> z=[1.1,1.2,1.4,1.6,1.9,2,2.5,3.4,4]/4;
>> plot(x,x)
>> for n=1:9
fx=1-z(n)*x.^2;
hold on, plot(x,fx)
end
>> hold off
```

The initial plot command "plot(x,x)" gets everything going here. The instruction "hold on" makes sure that previous plots are not erased when plotting the current one.
If you already have previous knowledge of a programming language, the idea of the loop should be familiar, what's different is the MATLAB format and the fact that it can deal with plotting within the loop.

Problem #9 Enter the lines of Matlab code above, to illustrate use of the FOR loop.

Problem #10 Write a program to sum the numbers from 1 to 100 using a FOR loop. Display the final sum using the "`disp`" command. (Do not display the intermediate sums). Use "`help disp`" to get information on how to use the "`disp`" command.

Quit MATLAB by clicking on the `File` menu in the upper left corner and choosing `Exit`. Please remember to **Log Off** (from the "Start" menu in the lower left of the screen).