Spring 2018

LAB 3 - MATRICES AND 3-D PLOTTING

Objectives: To learn how to use MATLAB to plot data in a matrix array, in particular:

- How to enter data into a matrix
- How to define matrices
- How to plot the data
- How to use different plot commands to plot the same data set

1. **Boot It Up**

   (a) Start up MATLAB

   - Before you begin all lab assignments, start by typing *diary lab*03 to begin recording your session. This is critical since you will be required to show this file to get credit for your lab.

2. **Matrix Basics**

   (a) To create a matrix in MATLAB, use square brackets and enter your row values separated by a space (or comma) then start a new row by inserting a semicolon (;)

```
S = [5 2 4 2 4;6 3 9 8 7;8 5 5 4 7]
T = [-3 2 0;1 1 1;-4 -5 -1;9 -4 9;3 8 -2]
U = [3 9 3 9;3 9 3 9;3 9 3 9;3 9 3 9;3 9 3 9]
```

   Create a 4×5 matrix X, a 5×4 matrix Y, and a 4×4 matrix Z. Recall a matrix is described as follows: number of rows(m) by number of columns(n) ($m \times n$).

   (b) Operations on matrices are simple, but can also be a source of frustration if the inner dimensions do not match. Evaluate these in the command window (note that you may receive errors :-):

```
X+X
X+Y
X*X
X*X'
X*Y
X*transpose(Y)
X+Z
Z^2
```

   (c) Element-wise operations do not necessarily depend on matrix dimensions to work properly. Evaluate these in the command window:

```
Z.^2
X1 = cos(X)
Z1 = exp(Z)
```

   (d) We can use vectorization to generate matrices in much the same way we created vectors. For example,

```
W = [0:20;20:40;40:60;60:80;80:100]
U = [35:-2:12;12:2:35]
X2 = [linspace(0,2*pi);linspace(-3*pi,1);linspace(0,12)]
Z2 = [linspace(0,2*pi,10);linspace(-3*pi,1,10);linspace(0,12,10)]
A = zeros(15,30)
B = ones(30,15)
C = rand(7)
D = nan(3)
```

   Create the following matrices: a 4×4 ones matrix, a 45×30 zeros matrix, a 13×13 rand matrix, and a 4×8 nan matrix.

(e) Inspecting elements in your vector can be done by using parenthesis and the address of said element. You can use the same notation to add elements to your array if necessary.

```
X(2,4)
X(1,2) = 35
X
```

Set the value in the 3rd row and 2nd column of matrix Y to 33. View matrix Y in the command window to confirm the change.

(f) There are several helpful functions built-in to MATLAB that you can use to get information about your data. These include:

max(X)	row vector containing the maximum value of each column
min(X)	row vector containing the minimum value of each column
sort(X)	treats the columns of X as vectors and sorts each column
sum(X)	sum of all values in X
prod(X)	product of all values in X
median(X)	median value of X
mean(X)	mean value of X
std(X)	standard deviation of X
size(X)	dimensions of X

Find the following: the sum of all values in Z, the dimensions of Y, and the mean value of X.

(g) There are many types of 3-d plots in MATLAB. You must first define your variable space and, as we saw in class, *linspace* will allow us to generate a grid on which to plot. Suppose we wanted to plot $z = e^x + 4sin(y)$ from $0 \rightarrow 2\pi$ in both the x and y direction:

```
X = linspace(0,2*pi);
[x,y] = meshgrid(X);
z = exp(x)+4*sin(y);
plot3(x,y,z)
figure,surf(x,y,z)
figure,surfc(x,y,z)
figure,contour(x,y,z)
```

Plot the following function using the method outlined above: $z = cos(x) + sin(y)$ for $0 \leq x \leq 3\pi$ and $0 \leq y \leq 3\pi$

(h) We can also use the subplot feature to view multiple plots in one figure. The notation for subplot defines the image grid and the plot location. As an example:

```
subplot(2,2,1),plot3(x,y,z)
subplot(2,2,2),surf(x,y,z)
subplot(2,2,3),surfc(x,y,z)
subplot(2,2,4),contour(x,y,z)
```

This will create a 2×2 grid layout with *plot*3 in the upper left, $surf$ in the upper right, $surfc$ in the lower left, and *contour* in the lower right (note: the position numbering is counted along the top row of the Figure window, then the second row, etc). Plot all 4 of these images vertically on one figure.

(i) One last note: we aren't restricted to only $m \times n$ matrices. MATLAB can handle n-dimensional matrices with ease. Enter these into the command window and notice the output:

```
a = ones(2,2,2)
b = rand(3,3,5)
c = round(10*rand(2,2,2,2,2,2))
```

Now create a $4 \times 3 \times 5$ zeros matrix.

3. **Putting it Together**

(a) Create a new M-file, *lab*0301.*m*. In the window, type the following exactly:

```
% Author: Mr T (or your name)
% 3-d plot of a function entered by user

% reset all session variables
clear all

% Enter 4 numbers
x1 = input('Enter your minimum x value: ');
x2 = input('Enter your maximum x value: ');
y1 = input('Enter your minimum y value: ');
y2 = input('Enter your maximum y value: ');

% build 100x100 matrix
X = linspace(x1,x2);
Y = linspace(y1,y2);
[x,y]=meshgrid(X,Y); % build the matrix
z = input('Enter some function z(x,y). (e.g., >>x.^2+sin(y)): ');

% plot the function using 4 different methods
figure(1);
subplot(2,2,1)
plot3(x,y,z)
title('Plot Data Using plot3')

subplot(2,2,2)
surf(x,y,z)
title('Plot Data Using surf')

subplot(2,2,3)
surfc(x,y,z)
title('Plot Data Using surfc')

subplot(2,2,4)
mesh(x,y,z)
title('Plot Data Using mesh')
```

4. **And...**

   (a) Create a script, *lab*0302.*m*, that solves the following problems. Put your name in the script and don't forget to comment.

   - Solve the system of equations:

$$\begin{aligned} -2x + 7y - 8z &= 4 \\ x - 3y + 21z &= -3 \\ -3x + 5z &= 1 \end{aligned}$$

   Define A as the matrix of coefficients and b as a column vector representing the right-hand side of these equations.

$$\begin{bmatrix} -2 & 7 & -8 \\ 1 & -3 & 21 \\ -3 & 0 & 5 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ -3 \\ 1 \end{bmatrix}$$

   Recall the matrix form of this equation is $AX = b$ with solution $X = A^{-1}b$.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -2 & 7 & -8 \\ 1 & -3 & 21 \\ -3 & 0 & 5 \end{bmatrix}^{-1} \begin{bmatrix} 4 \\ -3 \\ 1 \end{bmatrix}$$

Use both $X = A\backslash b$ and $X = inv(A) * b$ to find x, y, z. Use *tic toc* to time each method.

- Create a general solver to solve any linear system of n equations and n unknowns of the form $AX = b$. Allow the user to input the $n \times n$ matrix A and $n \times 1$ column vector b. Use $X = A\backslash b$ to solve the system in your script and display the solution to the user. Try your solver with $A = randi(50, 10)$, $b = randi(20, 10, 1)$ and $A = randi(3, 500)$, $b = randi(7, 500, 1)$.

(b) Open a new M-file and name it *lab*0303.*m*. Draw a small cityscape that is built from an 11×11 matrix where the values in the matrix are the building's heights. Assume a city block is 2x2 with a building in each cell (so each block has 4 buildings). Be sure each block has a street on each interior side. You are welcome to use the default colormap, or import a colormap of your choosing (you can find examples of other colormaps in the help section *help colormap*). You will need to utilize the *bar*3 function to get your buildings (consult the MATLAB documentation or Google *matlab bar*3). *Hint:* You can specify each cell manually using $A(1, 1) = 50$, $A(1, 2) = 75$, etc, or you can get creative by using a scaled *randi* function to set the height...just remember the streets will need to eventually be set to zero. You can set rows/columns to a certain value by using colon notation: B(1,:) = 0 (this sets all row 1 values to zero). Be sure to comment your code!

(c) Create your own planet! Write a script entitled *lab*0304.*m* that creates a rough sketch of your planet in 3-d. Create a 50×50 matrix of default values (maybe $A = zeros(50)$) then change the values where you want a land mass ($A(10 : 20, 20 : 30) = 1$). *You must create 5 independent land masses.* You will need to obtain your x, y, z coordinates from the *sphere* function, then use the $surface$ function to overlay your 50×50 matrix of values. Note: the indexing on the sphere will start at the south pole...of course if you end up with your earth upside-down just rotate it using the tool in the figure window. The *colormap* function will allow you set the colors of our planet however you choose. The input will be RGB values in rows scaled as a percentage (both methods below are valid):

```
colormap([1 0 0; 0 1 0; 0 0 1])
colormap([255/255 0 0; 0 255/255 0; 0 0 255/255])
```

The code above will set your RGB values as red $(1, 0, 0)$, green $(0, 1, 0)$, blue $(0, 0, 1)$. If you are wanting a challenge, feel free to model our earth. In order to replicate the earth you will likely need the grid on the next page to "draw" the continents. *Note:* The grid is 25 blocks high by 50 blocks wide so you will need to add 12 rows above and 13 below to obtain your 50×50 matrix. The $sphere(50)$ function will produce your planet with an $n \times n$ grid overlay. You will need *colormap*([0 1 0;0 0 1]) to get your green $(0, 1, 0)$ and blue $(0, 0, 1)$ colors to appear on the globe. Interact with your planet by using the 'rotate 3d' tool in the figure window (it is a circular arrow). Comment, comment, comment!

(d) You should submit 5 files to your TA for review: your diary file, *lab*0301.*m*, *lab*0302.*m*, *lab*0303.*m* and *lab*0304.*m*.

(e) Once the TA has checked your files you are free to go. Log off of your machine before you leave the lab.