APSC 1001 Introduction to Matlab #2

Based on "Getting Your Hands Dirty With MATLAB by Dr. Kartik Bulusu

September 23, 2015

1 Matrices in MATLAB

MATLAB assumes every variable to be potentially, a matrix. Recalling the deïňĄnition of an array we can start creating matrices in MATLAB.

An array is a list of numbers or expressions arranged in horizontal rows and vertical columns. When an array has one row or column, it is called a vector. An array with m rows and n columns is a called a matrix of size $m \times n$. (Pratap, 1999)

A matrix is entered row-wise, the rows separated by semicolons or carriage returns. Elements of the matrix may be real or complex numbers, or valid MATLAB expressions.

Matrix	**MATLAB input command**
$A = \begin{bmatrix} 1 & 2 & 7 \\ 4 & 9 & 0 \end{bmatrix}$	$A = [1\ 2\ 7; 4\ 9\ 0]$
$B = \begin{bmatrix} 8x & ln(x) + sin(y) \\ 3i & 7 + 2i \end{bmatrix}$	$B = [8 * x\ log(x) + sin(y); 3i\ 7 + 2i]$

1.1 Vectors and Scalars as special cases of matrices

- A *vector* is a special case of a matrix, with just one row or one coulmn. It is entered the same way as a matrix.

  $>> x = [1\ 5\ 9];$ % produces a row vector

  $>> y = [1; 5; 9];$ % produces a column vector

- A *scalar* can be thought of as a matrix with just one row and one column. A scalar does not require brackets when entered.

  >> $g = 9.81$; % notice that the semicolon suppresses the output

- A *null* or empty matrix has no elements in it. Create a null matrix with square brackets. This can be useful for initializing matrices, although it is not necessary every time.

  >> $u = [\,]$;

1.2 Indexing Matrices

Once a matrix is created in MATLAB, it's elements can be accessed by specifying the index of their row and column.

A(i,j) in MATLAB refers to the element in the ith row and jth column of a matrix A. While this notation is fairly common in many programming languages, MATLAB provides a few more options in index specification. These options make matrix manipulation much easier than in other programming languages.

- A(m:n, k:l) specifies rows m to n and columns k to l of matrix A.
- A(:, 3:6) refers to columns 3 through 6 of all rows (:) in matrix A.

Walk through these examples on your own. You should type each example and make sure you understand what is happening and why it happens. These are the basics of MATLAB; learning them now is the only way we can cover more advanced topics this semester.

```
>> A = [1 2 3; 4 5 6; 7 8 8] %Matrices are entered row-wise. Rows are
%separated by semicolons and columns are separated by spaces or commas.

A =
 1 2 3
 4 5 6
 7 8 8

>> A(3,2) %Element A(3,2) is accessed
ans =
 8

>> A(3,3) = 9 %Changing an element in matrix A
A =
 1 2 3
 4 5 6
```

```
 7 8 9

>> B = A(2:3, 1:3) %Any submatrix of A is generated by specifying the range
%of rows and columns
B =
 4 5 6
 7 8 9

>> B = A(2:3, :) %All rows or all columns of a matrix
%can be accessed by specifying ':' as the index
B =
 4 5 6
 7 8 9

>>B(:,2) = [ ] %A row or column of a matrix is
%deleted by setting it to a null vector
B =
 4 6
 7 9
```

2 Matrix Manipulation and Operations

We have already seen that it is easy fairly easy to correct wrong entries, add or delete rows or columns or extract a submatrix from a matrix because of the unique indexing feature in MATLAB. Matrix dimensions are determined automatically by MATLAB. In order for MATLAB to be useful, we need to perform matrix operations on the matrices we have created.

- **Matrix Algebra** is different from element-by-element arithmetic.
  - Matrix addition and subtraction only work if the matrices are of identical size.
  - Matrix multiplication is valid if the number of columns in A equals the number of rows in B.
  - Scalar multiplication takes a single value and multiplies it by each entry in a matrix. This is written the same way as matrix multiplication, $\alpha \cdot A$, but has a very different result.
  - Matrix division can be a tricky topic. In MATLAB A/B is equivalent to $A \cdot B^{-1}$, and $A\backslash B$ is equivalent to $A^{-1} \cdot B$.
  - Neither matrix multiplication nor division is commutative.
  - Matrix powers are only valid for square matrices. A^2 is equivalent to $A \cdot A$. In a linear algebra class you may discuss these topics further.

- **Element by element operations** use the operator '.' in MATLAB. Using this operator, you can multiply each term in a vector by one term in another vector. See the examples for further information.
- **Other common matrix operations**
  - The *transpose* operator, A', switches the rows and columns of A
  - The *exponent* operator, .^ , raises each element in an array to an exponent
  - **size(A)** returns the number of rows and columns in a matrix
  - **det(A)** takes the *determinant* of matrix A
  - **max(v)** and **min(v)** give the maximum and minimum entries in a vector, respectively
- **Utility Matrices**

  To further aid matrix manipulation and creation, MATLAB provides some useful utility matrices.
  - **eye(m,n)** returns an m by n matrix with 1's on the main diagonal, also called the *identity* matrix
  - **zeros(m,n)** returns an m by n matrix with all zeros
  - **ones(m,n)** returns an m by n matrix with all ones
  - **rand(m,n)** returns an m by n matrix of random numbers between 0 and 1
  - **diag(v)** creates a matrix with vector **v** on the main diagonal
  - **diag(A)** creates a vector of the main diagonal of matrix A

2.1 Examples

```
>>A = [1 2 3; 2 3 4; 6 4 8];
>>B = [2 3 4; 2 5 7; 9 3 2];
>>C = A*B %Notice the different outputs of A*B vs A.*B
C =
 33 22 24
 46 33 37
 92 62 68

>>D = A.*B
D =
 2  6 12
 4 15 28
 54 12 16

>>E = A\B
E =
```

```
  0.1667 -0.1667 -0.3333
 -2.3333  3.3333  5.6667
  2.1667 -1.1667 -2.3333

>>F = inv(A)*B  %The result of this is identical to the previous example
F =
  0.1667 -0.1667 -0.3333
 -2.3333  3.3333  5.6667
  2.1667 -1.1667 -2.3333

>>A = [1 5 3; 4 2 6];
>>size(A)
ans =
 2 3

>>[m, n] = size(A)  %answers returned by a function can be saved as variables.
m =
 2
n =
 3

>>A'  %transpose operator
ans =
 1 4
 5 2
 3 6

>>eye(3)
ans =
 1 0 0
 0 1 0
 0 0 1

>>[max_value, max_index] = max(A)
max_value =
 4 5 6
max_index =
 2 1 2
```

3 Plotting

3.1 Useful Functions

There are a few functions that are in MATLAB which are not specific to plotting, but will be both helpful and frequently used.

- **help *function*** In the MATLAB command window, you can type *help* then the name of any function. This will bring up documentation on the different uses for that function, the arguments it takes, what it returns, and often example uses and related functions. Further help and documentation is available online, or by typing *doc* plus the name of the function.
- **linspace(m,n,N)** The linspace function creates an array of size $m \times n$ with N linearly spaced entries.

```
>>t = linspace(0, 1, 11) %notice that 11 numbers are required to create 10 steps betw
t =
  Columns 1 through 7
         0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000

  Columns 8 through 11
    0.7000    0.8000    0.9000    1.0000
```

- **colon operator** The colon, :, is used in Matlab to denote ranges of numbers. It can also be used to create arrays. The command:

```
>>t = 0:0.1:1
```

  will create an array with entries ranging from 0 to 1 in step sizes of 0.1. This will have the same output as the previous *linspace(m, n, N)* example

3.2 Plot Command

The plot command in MATLAB looks like the following: `plot(x, y, optional_arguments)` In order to plot arrays in Matlab, they must be of the same length. The optional arguments cover settings on the figure such as labels, color, or style. Type 'help plot' for more information.

The best way to learn how the plot functionality in MATLAB works is with an example.

```
T = 10;      %final time
N = 101;     %number of steps to take
t = linspace(0, T, N); %create time array
x = sin(t);
y = cos(t);
plot(t, x, 'b:', 'LineWidth', 2)    %plot x vs. t in blue with dotted ...
    line and linewidth of 2
title('My Example Plot') %create title
xlabel('x axis label')   %label axes
ylabel('y axis label')
hold on      %turn on hold so that the plot is not overwritten
plot(t, y, 'k--', 'LineWidth', 2)   %plot y vs. t in bloack with ...
    dashed line and linewidth of 2
legend('x', 'y') %create legend
```

The code above will produce the following output: The figure will open in a new window.

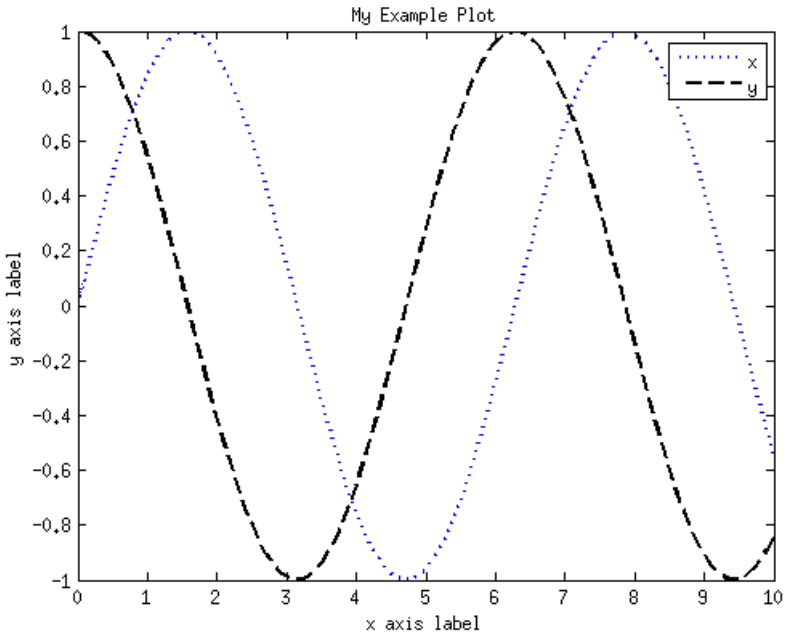

Figure 3.1: It is good practice (and required for this class!) to always include a title and axis labels for each plot.

Multiple figure windows can be open at a time, and any plotting commands will affect the current figure, which is the most recently selected figure window. Unless the *hold* option is on, a plot command will overwrite an existing plot.

3.3 Additional Commands

Some useful commands for plotting can be found below. For more information, use the help function.

- The `figure` command will open a new figure window. With an argument, `figure(1)` will open a specific existing figure window.
- The command `close` will close the current figure window. `close all` will close all open figures.
- A useful property of a figure window is its hold state, which determines whether an additional `plot` command will overwrite the current figure or add to it. The command `hold on` or `hold off` will change this state.
- MATLAB will automatically scale the axes of a plot. To set them manually, use the `min xmax ymin ymax]axis(x` command.
- To create a legend for your figure, use the `legend('string one', 'string ... two'` command.

References

[1] Pratap, R., *Getting Started with MATLAB 5 - A Quick Introduction for Scientists and Engi- neers*, Oxford University Press, 1999.

[2] Kreyszig, E., *Advanced Engineering Mathematics*, 8th ed., John Wiley & Sons, Inc., 1999.