Introduction to MATLAB

Eric Rodger and Elliot Cameron

McMaster University

September 16, 2009

Introduction

- MATLAB (MATrix LABoratory) has become the academic and industry standard for numerical computation
- Excels at performing matrix operations
  - Therefore it can handle large data sets (stored as matrices) very easily
- There are three main MATLAB windows
  - Command Window - you can enter code here that you want processed immediately
  - Editor - where you can write functions and scripts (to be discussed in detail later)
  - Graphics Window - where plots are displayed
- MATLAB is case-sensitive! (very important when working with variable and function names)

Useful Functions and Commands

- doc *function_name* - brings up information on any built-in MATLAB function
- *edit* - opens the text editor
- *clc* - clears the screen
- *clear* - clears the variables from the workspace (and from memory)
- Commonly used built in MATLAB functions (most of these are self-explanatory)
  - sin(), cos(), sqrt(), abs(), round(), min(), max(), mean(), var(), exp()
- Placing a semicolon at the end of a line of code will suppress any output produced by that line
- To display output on the screen use disp() or for more organized display use fprintf()
- Predefined constants include:
  - pi, Inf, i

Entering a Matrix

- The core strength of MATLAB is in its handling of matrices
  - It is proficient at handling scalars too (since these are just 1-by-1 matrices anyways)

```
>> A = [1 2 3; 4 5 6; 7 8 9]
```

$$A = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$$

- The above code creates a 3-by-3 matrix with the elements shown and assigns it to the variable A
- Note: Normally when you enter data or perform calculations, you assign the results to a variable
  - Otherwise it gets put into the *ans* variable which MATLAB creates automatically when you specify no output

Matrix Operations

- All operations in MATLAB follow matrix arithmetic rules
  - Be careful that you have matrices of the correct dimensions for matrix multiplication!

$$>> B = A * A \quad A = \begin{matrix} 30 & 36 & 42 \\ 66 & 81 & 96 \\ 102 & 126 & 150 \end{matrix}$$

- Other common matrix operations include:
  - A + B - adds A to B
  - A - B - subtracts B from A
  - A^ 2 - multiplies A by itself
  - inv(A) - takes the inverse of the matrix A
  - A' - takes the transpose of the matrix A

Matrix Operations Continued...

- Placing a dot before the multiplication operator allows you to do element-wise multiplication of two matrices
  - Note: Matrices must be the same size to do this!

$>> C = A.^*A$	$C = \begin{matrix} 1 & 4 & 9 \\ 16 & 25 & 36 \\ 49 & 64 & 81 \end{matrix}$

- This can also be done to raise the elements of a matrix to different powers contained in another matrix (.^ operator)
- Custom vectors of numbers can be created automatically by MATLAB

Vector Creation Example

$>> D = 1 : 4 : 20$ | $D = 1 \quad 5 \quad 9 \quad 13 \quad 17$

- Notation: *starting_point* : *increment* : *end_point*

Retrieving Values from Matrices

- MATLAB uses the standard (*row*,*column*) notation to assign locations to elements of a matrix
- Note that indexing starts at one (i.e. 0 is not a valid index in MATLAB)
- You can retrieve a sub-matrix from a larger matrix by specifying ranges using the colon operator

Retrieving Sub-Matrices Example

$>> E = A(2, 2:3) \mid E = 5 \quad 6$

- The colon (:), when used on its own, is a special character which allows you to grab all of the rows or columns of a matrix

Colon Example

$>> F = A(2, :) \mid F = 4 \quad 5 \quad 6$

- When you see the colon, just think *everything* in your head

Joining Matrices

- Rows and columns can be added to a preexisting matrix by horizontal and vertical concatenation

$>> G = [A; F]$	$G = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 4 & 5 & 6 \end{matrix}$
$>> H = [A\ F']$	$H = \begin{matrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 5 \\ 7 & 8 & 9 & 6 \end{matrix}$

- Creation of common matrices is automated by MATLAB
  - zeros(2,1) - matrix of zeros with 2 rows and 1 column
  - ones(1,2) - matrix of ones with 1 row and 2 columns
  - eye(2,2) - an identity matrix with 2 rows and 2 columns

Scripts

- Up until now we have only discussed entering statements into the command line (sometimes called interactive mode)
- Scripts are a series of statements, saved in a .m file, that are all executed at once
- Useful if you want to execute a set of commands multiple times
- Executed by typing the file name of the script into the command line
  - Note that the script file must be saved in the working directory!
- Any variable values resulting from calculations performed in the script are available to the user after the script is finished
- A simple example of a two-line script:

$$z = x^2 + y^2;$$
$$d = sqrt(z);$$

Functions

- Functions are more versatile than scripts because they can accept input arguments and return output values
- An example of a simple function:

```
function d = distance(x,y)
  % calculates distance between the point (x,y) and the origin
  z = x^2 + y^2;
  d = sqrt(z);
end
```

- The output variable here is d and the input variables are x and y
- The name of the function is *distance*
  - Save the function by this name as well to avoid confusion
- The % sign signifies the start of a comment (text that is ignored by MATLAB)

Notes on Functions

- The function on the preceding slide can be called with the following command:

```
>> D = distance(2.0,3.5)
```

- The function will calculate the distance between (2.0,3.5) and the origin and put the result in the variable D
- Note that the function allows us to perform the distance calculation without previously assigning values to the variables x and y in the workspace
- A function can assign multiple output variables (consult MATLAB help to find out how)
- Functions can call other functions
  - This is generally how complex programs are created in MATLAB

Conditionals

- Conditionals provide a means of controlling the flow of a program
- Conditionals accomplish this by using logic operators as arguments within the main conditional structure
- The primary conditional loop statements used in MATLAB are:
  - if
  - elseif
  - else
- While "elseif" and "else" are optional, the conditional "if" must always be used to begin this conditional structure
- MATLAB also supports the less used "Switch/Case" conditional (this will not be discussed here)

if, elseif, else

```
Syntax:
if(conditional argument)
   conditional expression
elseif(conditional argument)
   conditional expression
else
   conditional expression
end
```

```
Example:
a=1;b=2
if(a<b)
   fprintf('a is less than b')
elseif(a>b)
   fprintf('a is greater than b')
else
   fprintf('a is equal to b')
end

The above would print the string
'a is less than b' in the command window
```

- MATLAB supports the 6 following conditional operators:
  - A == B - Checks if A is equal to B
  - A =~ B - Checks if A is not equal to B
  - A > B - Checks if A is greater than B
  - A >= B - Checks if A is greater than or equal to B
  - A < B - Checks if A is less than B
  - A <= B - Checks if A is less than or equal to B

Control Structures - for loop

Syntax:	Definitions:
`for i=m:n:o`	m - starting iteration position
`  executed expression`	n - iteration increment
`end`	o - final iteration position

- "for loops" are ideal for performing tasks a predetermined number of times
- Nested "for loops" are ideal for manipulating multidemensional arrays
- Example:

```
m=0;
for i=1:2:7
     m=m+1;
end
```

The above code will result in 'm' having a final value of *4*

Control Structures - while loop

Syntax:

```
while(conditional argument)
    executed expression
end
```

while loops use the same conditional arguments as if,elseif,else control structures

- "While loops" allow the programmer to repeat a given task until a predetermined criteria is met (perfect for repeating numerical methods to a given tolerance)
- Example:

```
m=0;
while(m<4)
      m=m+1;
end
```

The above code will continue looping (each time adding an additional value to 'm') until 'm' reaches a value of *4*, at which point it will fail the next condition test and exit the loop

Plotting Data

- Matlab is able to represent data graphically using the "plot" function

Syntax:

```
plot(X1,Y1,LineSpec,'PropertyName',PropertyValue,...,
    Xn,Yn,LineSpec,'PropertyName',PropertyValue)
title{String}
xlabel{String},ylabel{String}
legend{String1,...,Stringn}
```

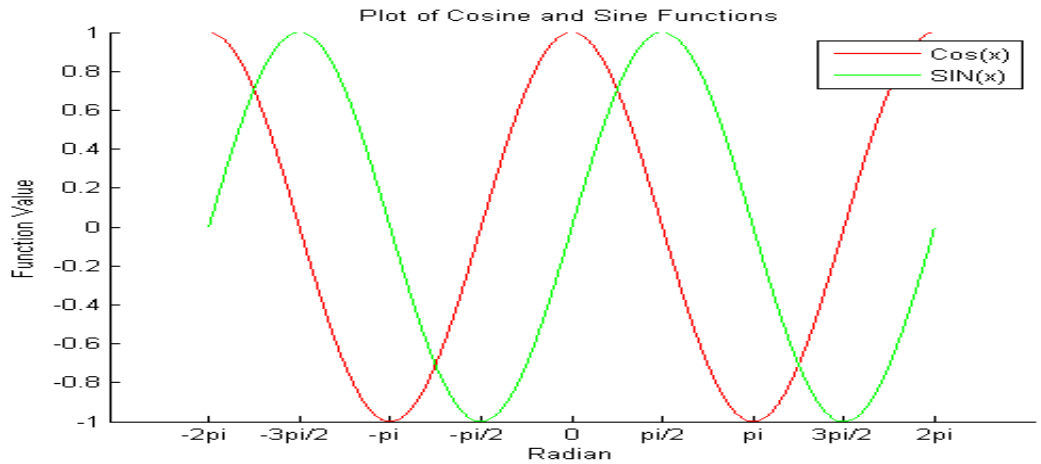
- MATLAB is able to plot multiple data sets by providing each X,Y set sequentially in the argument section
- 'Linespec' is a special property set that allows one to specify the line type, marker type, and color for the data set it postcedes
- Similarly, the "'Property Name', PropertyValue" couple allows further modification of individual data sets (additional information can be found under "lineseries properties" in the help menu)

Example Plot

- The following code:

```
x = [-2*pi:0.01:2*pi];
plot(x,cos(x),'r',x,sin(x),'g')
title('Plot of Cosine and Sine Functions');
xlabel('Radian'); ylabel('Function Value');
legend('Cos(x)','SIN(x)');
set(gca,'XTick',-2*pi:pi/2:2*pi)
set(gca,'XTickLabel','-2pi','-3pi/2','-pi',...
'-pi/2','0','pi/2','pi','3pi/2','2pi')
```

- Results in the following output in the plot window



Additional Packages and Functions

- Simulink
  - Good for modelling open and closed loop systems with and without control action
  - This package will likely be of use in CHE 4E3
- Symbolic Math Toolbox
  - This package allows theoretical calculations to be conducted using symbols instead of numbers
  - Additional built in functions within this package allow direct derivation, factorization, etc. of symbolic functions (useful for performing exact calculation of things like Jacobian and Hessian matrices)
- ODE and FSOLVE function sets
  - Very useful for numerically integrating ODEs and DAEs and solving sets of algebraic equations
- There are many other packages available. Documentation for these is available in the MATLAB help menu

Practice Example Time!

For additional MATLAB information/help please consult the following resources:

- Getting Started with MATLAB 7: link to pdf document
- MathWorks Documentation Listing: link to listing
- Online MATLAB Tutorial: link to U of F MATLAB tutorial
- MATLAB Cental File Exchange: link to central file exchange

Remember! MATLAB is your friend!