



MFM Orientation Seminar

MATLAB and Programming Intro

Chris Prouty

Orientation Seminar
University of Minnesota

- **Agenda**

- Introduction to MATLAB

- Defining matrices
- Simple matrix operations
- Math operators applied to matrices

- Introduction to MATLAB m-files

- Functions versus scripts
- Math operators in programming
- Flow of control
- The If statement
- The For statement

- Discussion of 5091 and 5092 class projects and materials

- **Programming in MATLAB**

- MATLAB is short for “Matrix Laboratory”
- MATLAB provides a superb environment for declaring and manipulating matrices
- Today we’ll start by declaring some simple matrices and performing some simple manipulations
- The commands used at the MATLAB prompt can be aggregated into a structure called an “m-file” to automate large tasks in MATLAB
- m-files can also be used to create functions and rich programs with data input/output, flow control, and visual elements

- **Programming in MATLAB**

- At the MATLAB command prompt, matrices are defined by setting an arbitrary matrix name equal to a matrix value
- Matrices are defined using the following characters: **[]**; and **,**

- Example:

MyMatrix=[1,2,3;4,5,6;7,8,9] \longrightarrow

1	2	3
4	5	6
7	8	9

- Brackets indicate the beginning and end of matrix data
 - Commas advance a column (spaces can be used alternatively)
 - Semicolons advance a row
- Note that variable names in MATLAB are case-sensitive, so **MyMatrix!=MYMATRIX**

- **Programming in MATLAB**

- Specific scalars in a matrix can be referenced by calling the appropriate row and column, with the upper-left scalar being (1,1)

- Example:

MyMatrix(2,3)

- The command above returns the value **6**, since it references the 2nd column and 3rd row

- Scalars can be set equal to values, too

- Example:

MyMatrix(2,3)=50,000

- The command above sets scalar in the 2nd column and 3rd row equal to the value 50,000

• Programming in MATLAB

- Math operators in MATLAB function intuitively
 - + Add two scalars/vectors/matrices
 - - Subtract one scalar/vector/matrix from another
 - * Multiply two scalars/vectors/matrices
 - / Divide one scalar/vector/matrix by another
- Math operators in MATLAB adhere to all rules applying to matrix operations
- For example, multiplying the two highlighted vectors would raise an error unless one vector was transposed

1	2	3
4	5	6
7	8	9

- **Programming in MATLAB**

- Referencing entire vectors of a matrix can be accomplished by using the **:** operator

- Example:

- MyMatrix(:,1)**

- The command above returns every row value in the first column

- A matrix or vector can be transposed using the **'** operator

- Example:

- MyMatrix(:,1)'**

- The command above returns every row value in the first column transposed

- **Programming in MATLAB**

- Therefore, a new 3x3 matrix can be generated from MyMatrix by referencing the same vector and transposing

- Example:

$$\mathbf{MyMatrix(:,1)*MyMatrix(:,1)'} \longrightarrow \begin{array}{ccc} 1 & 4 & 7 \\ 4 & 16 & 28 \\ 7 & 28 & 49 \end{array}$$

- There are many occasions when a series of commands may be needed to generate a custom matrix and automating those commands would greatly improve efficiency

- MATLAB m-files come in two varieties and extend richness enormously

- Scripts
- Functions

- **Programming in MATLAB**

- We'll begin exploring m-files by creating a **script**

- To defined a new script, use the **edit** keyword at the MATLAB prompt

- Example:

- edit makematrix**

- The command above creates an m-file called “makematrix.m” and opens the editor

- All the commands explored so far today can be used verbatim in an m-file

- Once an m-file is created and saved, it can be run from the MATLAB prompt by entering the name of the file and pressing Enter

- **Programming in MATLAB**

- 5 Minute Challenge

Create a MATLAB script that starting by defining a matrix like so:

1	2	3
4	5	6
7	8	9

And using matrix operations turning it into a matrix like so:

10	40	70
40	160	280
70	280	490

- **Programming in MATLAB**

- Assuming that the m-file is saved after the code is entered, running it will define a new matrix in the MATLAB workspace called “A”



**Variables declared in an m-file script
appear in the MATLAB workspace**



- Scripts are useful when automating a number of simple commands that perform a repetitive operation
- Another breed of m-file, called a function, provides a more flexible venue for creating complicated code

- **Programming in MATLAB**

- Function m-files are defined in the same way as scripts by using the **edit** keyword
- However, function m-files require a special header
- Syntax:

function [o₁, o₂, o₃...] = name(i₁, i₂, i₃...)

- A function may contain any number of input and output parameters, as indicated by o_n and i_n in the syntax shown above
- Example:

function [result] = MultiplyTwo(a, b)

- In this example there are two input parameters and one output parameter

- **Programming in MATLAB**

- Once the header is declared, code may be added
- Values are returned by the function by setting the variables defined inside the brackets equal to a given value

- Example:

```
function [result]=MultiplyTwo(a,b)  
result=a*b;
```

- Once saved, the m-file function shown here could be run from the command prompt like so

- Example:

```
MultiplyTwo(5,5)
```

- The value 25 would be returned from the function

- Programming in MATLAB

- In computer science, a principle known as **flow of control** governs the manner in which a program runs
- Consider the two pseudo-code programs below

```
1: [a,b]=GetFromUser  
2: a=a+1  
3: b=b+1  
4: c=a*b  
5: Show c
```

```
1: [a,b]=GetFromUser  
2: a=a+1  
3: b=b+1  
4: c=a*b  
5: If c > 10 Then Goto 1  
6: Else Show c
```

- The program on the left starts at the top and runs line-by-line to the bottom
- The program on the right has a logical statement that modifies flow if a condition is met

- **Programming in MATLAB**

- By adding the ability to make logical tests, code can behave in much richer ways
- MATLAB has a handful of standard **flow of control** statements to test conditions and loop code
- Today we'll introduce two of these statements
 - If
 - For
- Some incarnation of the **If** statement is virtually ubiquitous in the world of computer programming
- The If statement compares two or more values and executes different code based on the outcome of the comparison

- **Programming in MATLAB**

- Four keywords comprise the If statement

- If
- Else
- Elself
- End

- Example:

```
if (A==5)  
    A=A*10;  
end
```

- When this code is encountered, code within the statement (`A=A*10;`) is only executed if the condition (`A==5`) is true, otherwise the block of code is skipped

- **Programming in MATLAB**

- The If statement requires a number of logical operators to perform value comparisons

- == Equal to
- > Greater than
- < Less than
- >= Greater than or equal to
- <= Less than or equal to
- ~= Not equal to

- Any number of logical statements can also be joined

- & And
- | Or

• Programming in MATLAB

- Consider the m-file shown below

```
function [result]=ifthen(v1, v2)

if(v1==10)|(v2==5)
    disp("v1 is 10 OR v2 is 5")
elseif(v1*v2==10)
    disp("v1 times v2 is 10")
else
    disp("No other conditions met")
end
```

The **disp** command is used to display a string of text.

- In the code example above, at least one of the code blocks will be executed
- The first condition test is expressed in the first If statement and the Elseif statement provides the next conditional test
- If both tests fail then Else code is executed

- **Programming in MATLAB**

- 5 Minute Challenge

Write a function m-file that accepts two user inputs. If the two inputs multiplied together are greater than or equal to 1000, then output the text “Condition Satisfied”. If the two inputs multiplied together are less than 1000, then output the text “Condition Not Satisfied”.

- **Programming in MATLAB**

- The second flow of control statement we'll introduce today is the **For** statement
- The For statement is one variety of a structure known as a **loop**
- The purpose of a loop is to execute a block of code repeatedly until some condition is met
- In the case of a For loop, the condition is a counter that terminates the loop once it has executed a specific number of times

- Syntax:

```
for counter_variable=start_value:step:end_value  
    code block  
end
```

- Programming in MATLAB

- Example:

- for a=0:5:50**

- disp(a)**

- end**

- The code above uses the variable “a” as a counter and displays the value of “a” each time the loop executes

- **What would this code display?**

- In a For loop, the step value (in this example 5) can be omitted; the default step value is 1

- The start and end values are arbitrary; if the step value is negative then the loop can count down

- **Programming in MATLAB**

- Flow of control statements can be **nested** within each other

- Using two For loops, code can be created that counts through a matrix of any given size

- Example (assume MyMatrix is 10x10):

```
for a=1:10
```

```
    for b=1:10
```

```
        Total=Total+MyMatrix(a,b);
```

```
    end
```

```
end
```

- In this code the variable “Total” is aggregated my the scalar values each time through the loops

- **Programming in MATLAB**

- Loops are powerful, but MATLAB's innate functions can usually process data *much* faster
- The code in the previous slide sums all the scalars in a given matrix
- The same thing can be accomplished using the MATLAB function **sum(...)**
- Example:
sum(sum(MyMatrix))
- This code sums the matrix into one row with ten columns (sum of the rows) and then sums the row into a single scalar (sum of the columns)
- **Use MATLAB functions wherever possible!**

- Programming in MATLAB

- 5 Minute Challenge

Determine the problem with this code:

```
for a=1:50
    if(a==49)
        a=a-1;
    end
end
```

- **Programming in MATLAB**

- In the coming year in 5091 and 5092 we will explore programming in MATLAB and C#
- Projects will mostly involve models and numerical techniques popular in finance
- As time permits, I encourage students who do not have a strong programming background to experiment with MATLAB and C# to build a strong intuition of their use
- See you in class!