MATLAB WORKSHOP

Presented by:
Institute of Electrical and Electronics Engineers (IEEE)
Eta Kappa Nu (HKN)

Prepared by: Jack Sun
Revised by: Alex Rusin
Overview

- **Background**
- Basic syntax and commands
- Loops
- Graphing
- Advanced Graphing
- Notebook
Background

- MATLAB = Matrix Laboratory (developed by The MathWorks).
- Strengths:
  - Vector/Array: 1-D, such as 1xN or Nx1.*
  - Matrix: 2-D, such as MxN or NxM.*
  - Multidimensional arrays: 3-D or more.
  - Logic and algorithm are intuitive to learn.
  - Graphing.

*Most courses at CSUN tend to focus on vectors, matrices, and any derivatives of those two.
Background cont’d

- Weaknesses:
  - Learning curve can be quite high.
  - Index is different than most programming languages (will be discussed shortly).
  - Syntax help does not always describe source of the issue (will be discussed in dot operator).

- Courses that utilize MATLAB:
  - ECE 350, 351, 410, 411, 412, 450, 451, 455, 460, 561, and more!
Overview

- Background
- **Basic syntax and commands**
- Loops
- Graphing
- Advanced Graphing
- Notebook
Basic Commands

- Variables
- Operators
- Arrays
- Matrices
- Dot Operator
- Index
- Max/Min
- Matrix Operations
- M-file
Variables

- Can be anything, but first character must be a letter:
  - Valid:
    - x1 = 5
    - var_name = ‘some string’
  - Invalid:
    - 1x = 5

- Variables are case sensitive:
  - i.e., a1 does NOT equal A1.
Operators

- Ex. Using “Command Window”
  ```
  >> x = 5;   % this is used as a comment
  >> y = 3;
  >> z = x + y
  z = 8
  ```

- Semicolon (;): Suppresses output.
  - In [ ], semicolon generates new rows.

- Percentage (%): Commenting. Only good for that line, anything after % will be in green color (default color scheme).
Operators cont’d

- Multi-line comments:

```%
%

Comment here
%
%

The % and % must be on their own lines.
Operators cont’d

- Logic operators:
  - `<` less than
  - `<=` less than or equal to
  - `>` greater than
  - `>=` greater than or equal to
  - `==` equal to
  - `~=` not equal to

Note: logic equal to is NOT the same as arithmetic equal to!
Arrays

- Ex.
  
  ```
  >> a = [1, 3, 5, 7, 9]  %comma optional
  >> b = 1:9
  >> c = 1:2:9
  ```

- Array a: 5 elements, but can be tedious if need to define many elements.

- Array b: Sets min and max values for the array. Formula is min:step:max. If no step size is defined, MATLAB uses default of 1.
Matrices

- How to express this matrix in MATLAB:

\[
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6
\end{bmatrix}
\]

2 by 3 matrix

- \( \gg \gg a = [1 \ 2 \ 3; \ 4 \ 5 \ 6] \)
Matrices cont’d

- Matrix elements can be expanded in the same way as an array.
  - Ex. \( b = \begin{bmatrix} 1:5; 5:9 \end{bmatrix} \)

- Note: They need to have the same number of elements!
  - Ex. Is \( a = \begin{bmatrix} 1:5; 5:8 \end{bmatrix} \) valid?

\[
a = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 6 & 7 & 8 \end{bmatrix}
\]

No, it’s not!
Dot Operator

• For scalar operations, nothing new is needed.
  • Ex.
    
    \begin{align*}
    a &= 5; \\
    b &= 3; \\
    c &= a \times b & \text{%}c = 15
    \end{align*}

• For element operations, a dot must be used before the operator.

• Note: Dot operator not the same as dot product!
Example

\[ a = [1 \ 2 \ 3 \ 4]; \]
\[ b = [5 \ 6 \ 7 \ 8]; \]
\[ c = a \times b \]

\[
\begin{align*}
\text{>> } a &= [1 \ 2 \ 3 \ 4]; \\
\text{>> } b &= [5 \ 6 \ 7 \ 8]; \\
\text{>> } c &= a \times b \\
\text{??? Error using } \Rightarrow \text{ mtimes} \\
\text{Inner matrix dimensions must agree.}
\end{align*}
\]
Example

\[a = [1 \ 2 \ 3 \ 4];\]
\[b = [5 \ 6 \ 7 \ 8];\]
\[c = a.*b \quad \%\text{notice the dot!}\]

\[\begin{array}{c}
\text{c} = \[5 \ 12 \ 21 \ 32\]
\end{array}\]

Notice what it is doing: \(a(1)*b(1), a(2)*b(2),\) etc.
Index

- MATLAB index starts with 1, NOT 0!
  - Side note: many programming languages start the index at 0.
- Vector indexing works in a similar way as matrix indexing, but NOT the same!
Vector Index

- $a = [22 \ 17 \ 7 \ 4 \ 42]$
  - $a(1) = 22$
  - $a(3) = 7$
  - Etc.
Matrix Index

- \( \mathbf{a} = \begin{bmatrix} 7 & 12 & 42 \\ 5 & 1 & 23 \\ 4 & 9 & 10 \end{bmatrix} \)

- \( \mathbf{a}(1, 3) = 42 \)
- \( \mathbf{a}(3, 2) = 9 \)
- Etc.
Max/Min

- To find max/min in a vector, use:
  - max(a) or min(a) where a = the vector
- Ex.

```matlab
>> a = [12 7 15 1 3];
   max_a = max(a)
   min_a = min(a)

max_a =

15

min_a =

1
```
Matrix Operations

- **Transpose:**
  - Use the single quote, or apostrophe, to take the transpose of a matrix.
  - Ex. 
    ```latex
    \begin{bmatrix}
    1 & 3 & 5 \\
    7 & 9 & 11
    \end{bmatrix}
    
    \text{a =}
    
    \begin{bmatrix}
    1 & 3 & 5 \\
    7 & 9 & 11
    \end{bmatrix}
    
    \text{b = a'} 
    ```
    ```latex
    \begin{bmatrix}
    1 & 7 \\
    3 & 9 \\
    5 & 11
    \end{bmatrix}
    ```
Matrix Operations cont’d

- **Inverse:**
  - \( \text{inv}(a) \) where \( a \) = the matrix

- **Ones:**
  - Fills the matrix with all 1’s.
  - For square matrix:
    - \( \text{ones}(n) \) where \( n = \text{nxn matrix} \)
  - For non-square matrix:
    - \( \text{ones}(n, m) \) where \( n, m = \text{nxm matrix} \)

- Examples next page.
Examples

```plaintext
>> ones(5)
ans =

    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1

>> ones(4, 5)
ans =

    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1

>> ones(1, 2)
ans =

    1     1
```
Matrix Operations cont’d

- Zeros:
  - Similar to ones(n) command, but with 0’s instead.
  - Square matrix:
    - zeros(n) where \( n = nxn \) matrix
  - Non-square matrix:
    - zeros(n, m) where \( n, m = nxm \) matrix
  - Examples next page.
Examples

```plaintext
>> zeros(4)

ans =

    0     0     0     0
    0     0     0     0
    0     0     0     0
    0     0     0     0

>> zeros(2, 3)

ans =

    0     0     0
    0     0     0
```
Polynomials

- We can use an array to represent a polynomial. To do so we use list the coefficients in decreasing order of the powers. For example $x^3+4x+15$ will look like $[1 0 4 15]$
- To find roots of this polynomial we use roots command. \[ \text{roots } ([1 0 4 15]) \]
- To create a polynomial from its roots poly command is used. \[ \text{poly([1 2 3]) where r1=1, r2=2, r3=3} \]
- To evaluate the new polynomial at $x = 5$ we can use polyval command. \[ \text{polyval } ([1 -6 11 -6], 5) \]
- To multiply two polynomials we use conv command. If we want to multiply our polynomial $x^3+4x+15$ with $x^3-6x^2+11x-6$ we type \[ \text{conv([1 0 4 15],[1 -6 11 -6])} \]
Consider the following system of equations

\[ x + 5y + 15z = 7 \]
\[ x - 3y + 13z = 3 \]
\[ 3x - 4y - 15 = 11 \]

One way to solve this system of equations is to use matrices. First, define matrix A:

\[ A = \begin{bmatrix} 1 & 5 & 15 \\ 1 & -3 & 13 \\ 3 & -4 & 15 \end{bmatrix} \]

Second, matrix b:

\[ b = [7; 3; 11] \]

Third, we solve the equation \( Ax = b \) for \( x \), taking the inverse of \( A \) and multiply it by \( b \):

\[ x = \text{inv}(A) \times b \]

Note that we cannot solve equation \( Ax = b \) by dividing \( b \) by \( A \) because vectors \( A \) and \( b \) have different dimensions!
MATLAB Editor: M-file

• Why use M-file:
  • Easier to write a block of code, then execute as opposed to line-by-line execution in command window.

• To access the editor:
  • 1) File -> new -> blank M-file.
  • 2) Paper icon right below the File tab.

• Try to use M-file for all examples hereon out!
Overview

- Background
- Basic syntax and commands
- **Loops**
- Graphing
- Advanced Graphing
- Notebook
Loops

- Loop statements do not need parenthesis. The statements are recognized via tabs.
- 3 general types of loops:
  - If/else/elseif loops
  - For loops
  - While loops
- There is a 4\textsuperscript{th} type, called nested loop, that can be any of the above 3 (and any combinations of them).
If/else/elseif Loops

- The condition must be previously defined!

```plaintext
if condition
    statement
end

if condition
    statement
else
    statement
end
```
If/else/elseif Loops

if condition
    statement
elseif
    statement
else
    statement
end
Example

x = 5;
if x < 5
    answer = 'x is less than 5!'
elseif x < 10
    answer = 'x is less than 10, but greater than 4!'
else
    answer = 'x is greater than 10!'
end

Which loop statement will be triggered?
Example

```plaintext
x = 5;
if x <= 5
    answer = 'x is less than or equal to 5!'
elseif x < 10
    answer = 'x is less than 10, but greater than 5!'
else
    answer = 'x is greater than 10!'
end

Which loop statement will be triggered?
```
For Loops

- Counter variable does not have to be i; it can be any variable.
- Iterations can be tightly controlled with min:step size:max.
- No need to pre-define the counter because you are declaring in the for loop itself!

```
for i = 1:10
    statement
end
```
Example

\[ x = 100; \]
\[ \text{for } i = 1:3 \]
\[ \quad x = x + 1 \]
\[ \text{end} \]

\[
\begin{align*}
\text{>> clear} \\
\text{>> } x &= 100; \\
\text{for } i &= 1:3 \\
\quad x &= x + 1 \\
\text{end} \\
\end{align*}
\]

\[ x = \]
\[ 101 \]
\[ x = \]
\[ 102 \]
\[ x = \]
\[ 103 \]
While Loops

- As long as output of condition is a logic true, it will continue looping until the condition becomes false.
- BE CAREFUL OF INFINITE LOOPS.

while condition
    statement
end
Example

\[ x = 10; \]
while \( x < 15 \)
\[ x = x + 1 \]
end
Nested Loops

- Can use a mix of the different types of loops.
- Very useful for performing algorithm/operations on vectors and matrices.
Example

%Example nested loops
x = zeros(10, 5);
for i = 1:10
    for j = 1:5
        x(i, j) = j;
    end
end
Example Result

```matlab
>> x = zeros(10, 5);
for i = 1:10
    for j = 1:5
        x(i, j) = j;
    end
end
>> x

x =

1     2     3     4     5
1     2     3     4     5
1     2     3     4     5
1     2     3     4     5
1     2     3     4     5
1     2     3     4     5
1     2     3     4     5
1     2     3     4     5
1     2     3     4     5
1     2     3     4     5
```
Overview

- Background
- Basic syntax and commands
- Loops
- **Graphing**
- Advanced Graphing
- Notebook
Common Graphing Commands

- **plot** – plot(x,y)
  - x and y vectors must be same length!
- **subplot** – subplot(m,n,p) where mxn matrix, p = current graph
- **figure** – creates new window for graph
- **title** – creates text label at top of graph
- **xlabel/ylabel** – horizontal and vertical labels for graph
Example

%Example: Graphing commands
x = 1:5;
y = [1 2.3 2.7 3 5];
z = [2 4 6 8 10];

figure;
subplot(2, 1, 1)
plot(x, y)
title('X vs Y plot')
xlabel('Values of x')
ylabel('Values of y')

subplot(2, 1, 2)
plot(x, z)
title('X vs Z plot')
xlabel('Values of x')
ylabel('Values of z')

Note: Spacing is very important because it makes the code readable!
Example Result

X vs Y plot

Subplot(2, 1, 1)

X vs Z plot

Subplot(2, 1, 2)
Another Example

%Example: More Graphing Commands
x = 1:5;
y = [1 2.3 2.7 3 5];
z = [2 4 6 8 10];

figure(2);
plot(x, y)
title('X vs Y plot')
xlabel('Values of x')
ylabel('Values of y')

figure(5);
plot(x, z)
title('X vs Z plot')
xlabel('Values of x')
ylabel('Values of z')
Example Result
Axis-specific

- Helps focus on what is important on the graph.
- Change only the x or y axis limits:
  - `xlim([xmin xmax])` or `ylim([ymin ymax])`
  - min and max can be positive or negative.
- Change both axes:
  - `axis([xmin xmax ymin ymax])`
Example

%Example: Graphing commands with specific axes
x = 1:5;
y = [1 2.3 2.7 3 5];
z = [2 4 6 8 10];

figure;
subplot(2, 1, 1)
plot(x, y)
title('X vs Y plot')
xlabel('Values of x')
ylabel('Values of y')
xlim([1 4]);
ylim([-5 5]);

subplot(2, 1, 2)
plot(x, z)
title('X vs Z plot')
xlabel('Values of x')
ylabel('Values of z')
axis([1 4 -5 5]);

Added code
Example

Using `xlim/ylim`:
```matlab
xlim([1 4]);
ylim([-5 5]);
```

Using `axis`:
```matlab
axis([1 4 -5 5]);
```

Achieves the same result!
Copy Figure saves to clipboard so user can paste into document file or image editing software.
Data Cursor

Good for determining a specific data point
Plot Line

- Inside the plot command, can change the plot line and the color.
- Colors are useful for multi-plots in 1 graph.
- `plot(x, y, ‘<insert>’)`
- `<insert>` can be:
  - * ← Changes the line to stars/asterisks.
  - r ← Changes the color of the line to red.
    - Colors are case insensitive, i.e., r is same as R.
  - There are many others, type help plot for more!
Example

%Example: Graphing - Plot Lines
x = 1:5;
y = [1 2.3 2.7 3 5];
z = [2 4 6 8 10];

figure;
subplot(2, 1, 1)
plot(x, y, '*')
title('X vs Y plot')
xlabel('Values of x')
ylabel('Values of y')

subplot(2, 1, 2)
plot(x, z, 'r')
title('X vs Z plot')
xlabel('Values of x')
ylabel('Values of z')

Added code
Example Result

X vs Y plot

Values of x
Values of y

X vs Z plot

Values of x
Values of z
Overview

- Background
- Basic syntax and commands
- Loops
- Graphing
- Advanced Graphing
- Notebook
Advanced Graphing

- Stem plot – stem(x, y)
- Hold – holds current plot so another set of data points can be plotted onto same graph
  - Command: hold on, release via hold off.
- FFT – fft(x, n)
- Inverse FFT – ifft(x, n)
  
  Note: n in both cases = how many points
- FFT Shift – fftshift(x)
  - Shifts zero-freq. component to center of spectrum.
Advanced Graphing cont’d

• Linspace: Generates a 1xN vector (row vector) with automatic step size to create equal space points.
  • linspace(a, b) generates 100 points.
  • To specify your own points:
    • linspace(a, b, N) where N = number of points.

• Angle: Phase angle of the matrix (in radians).
  • angle(a)

• Absolute Value: Turns all elements inside matrix to positive value.
  • abs(a)
%Example: Sine wave generation

```matlab
%Example: Sine wave generation
frequency = 1000;
fs = 16000;
t = 0:1/fs:10;
xn = zeros(length(t),1);

%generates 5 frequency components and sum them into xn
for k = 1:5
    xn = xn + cos(2*pi*k*frequency*t)';
end

xn = fft(xn,1024);
xnfs = fftshift(xn);
f = linspace(-fs/2, fs/2, 1024);
figure;
subplot(2,1,1)
plot(f, abs(xnfs))
title('Magnitude of xn');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(2,1,2)
plot(f,angle(xnfs))
title('Phase of xn');
xlabel('Frequency (Hz)');
ylabel('Phase');
```
Example Result

Magnitude of $x_n$

Phase of $x_n$
Overview

- Background
- Basic syntax and commands
- Loops
- Graphing
- Advanced Graphing
- Notebook
Notebook

- Typing notebook in command prompt opens up MS Word (in Windows).
- Allows user to type like a word document, but also execute MATLAB code in MS Word.
- Good for MATLAB assignments/projects!
Notebook Commands

- Type code as normally would in the editor. To execute a block of code, highlight the code then press ctrl+enter.
Example

• Live Example!
Summary

- Background
- Basic syntax and commands
- Loops
- Graphing
- Advanced Graphing
- Notebook
One Thing to Remember...

- If there is only one thing to take away from this workshop, please let it be this:

  help <insert command>
THE END