ECEN 3021
Experimental Methods II
Fall 2004

Laboratory Session Using MATLAB®

Lab #2

Introduction/MATLAB Environment
Introduction to Mathematical Computation Tools

• This software category includes packages such as Mathematica, Mathcad, Maple, Macsyma and MATLAB

• Allow symbolic calculations and the manipulation of complex mathematical formulas

• Contain extensive capabilities for generating graphs

• Useful tools for engineers because of their combination of computational and visualization power
An Engineering Problem-Solving Methodology:

Can be used with any of the mathematics packages, including MATLAB

- State clearly the problem which is to be solved
- Input/Output Description
  - What information is given (inputs)?
  - What quantities must be found (outputs)?
  - What mathematical relations link the inputs to the outputs?
- Hand Example
  - Using a simple set of data, work the problem by hand or with a calculator
  - This is the step which allows the solution sequence to be developed in detail
- MATLAB Solution
  - Develop an algorithm, which is a step-by-step mathematical outline of the your proposed solution
  - Translate the algorithm into MATLAB code
- Testing: Ensure that your MATLAB routine works properly by testing it using a variety of data
MATLAB Environment

MATLAB Windows

- The *command window* is active when you first enter MATLAB
  - Interactive commands can be entered at the prompt
  - Results (output) will automatically be displayed

  ![MATLAB Command Window]

  
  ```
  >> sqrt(9)  % command (typed at prompt)
  ans =
  3
  >>  % MATLAB output
  >>  % MATLAB prompt (>>) and cursor (|)
  
  ```

- The *graphics window* is used to display plots and graphs. To see the graphics window
  - Type the following at the prompt: `>> plot([1,2,4,9,16],[1,2,3,4,5])`
  - MATLAB plots the vectors as shown below:

  ![MATLAB Plot]
MATLAB Windows (continued)

- The *demo window*
  - Activate by typing `demo` at the command window prompt
  - Choose from among the topics listed in the left window

- The *edit window*
  - Used to create and modify *M-files* (MATLAB scripts)
  - Type `edit` at the command window prompt
Using M-files

- M-files allow you to save and execute multiple commands or entire programs with a single command line entry
- Creating an m-file
  - Open the MATLAB editor
  - Type in the commands you want to execute
  - Save the file in a location accessible to MATLAB (usually the MATLAB work directory or current working directory)
  - In the MATLAB command window, type in the name of the file to execute the commands
- Executing an m-file of this type has the same effect as copying and pasting the commands into the command window
- MATLAB also supports functions, which execute in a separate workspace and do not have access to all user workspace variables
- Writing functions
  - Functions are also contained in m-files, so the creation process is similar
  - A function must begin with a line of the following format:
    ```matlab
    function <outputs>=functionname(<inputs>)
    ```
  - The commands following this line are standard MATLAB commands that may use the inputs and must assign values to the outputs
MATLAB Interactive Help Window

- Access via the pull down Help menu - click on Help Window
- Double-click on a topic of interest
- A non-interactive version of help is available by typing help at the command window prompt
- An HTML version of help is available by choosing Help Desk from the pull down Help menu
Managing the MATLAB Environment

Access the following by typing into the command window:

<table>
<thead>
<tr>
<th>Task</th>
<th>MATLAB Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short description of runtime environment (assigned variables)</td>
<td><code>who</code></td>
</tr>
<tr>
<td>Detailed description of runtime environment</td>
<td><code>whos</code></td>
</tr>
<tr>
<td>Clearing the environment (removing all variables from memory)</td>
<td><code>clear</code></td>
</tr>
<tr>
<td>Clear command window</td>
<td><code>clc</code></td>
</tr>
<tr>
<td>Clear current figure (graphics window)</td>
<td><code>clf</code></td>
</tr>
<tr>
<td>Save your environment (defined variables)</td>
<td><code>save filename</code></td>
</tr>
<tr>
<td>Load previously saved environment (.mat extension will be automatically added)</td>
<td><code>load filename</code></td>
</tr>
<tr>
<td>List files in the current directory</td>
<td><code>dir</code></td>
</tr>
<tr>
<td>Delete a file from the current directory</td>
<td><code>delete</code></td>
</tr>
<tr>
<td>Move to another directory</td>
<td><code>cd</code></td>
</tr>
<tr>
<td>Show current path (directory)</td>
<td><code>path</code></td>
</tr>
</tbody>
</table>

Some tasks can be accessed via the *File* pull down menu:
MATLAB Environment

The Matrix Data Structure

- All variables in MATLAB are represented as matrices
  - ** Scalars**: 1 by 1 matrices
  - ** Vectors **: n by 1 or 1 by n matrices

- **Anatomy of a matrix**
  - Elements (entries) arranged in rows and columns
  - Individual elements can be referenced by their row and column location; e.g., \( a_{4,2} = 7 \)

\[
a = \begin{bmatrix}
2 & 0.5 \\
-4 & 1 \\
3 & 2 \\
1 & 7 \\
\end{bmatrix}
\]

- **Square matrix**: A matrix whose number of rows and columns are equal

- **Rules for variables**
  - Variable names must start with a letter
  - Variable names can contain letters, digits and the underscore character (_)
  - Variable names can be any length, but they must be unique within the first 19 characters
  - MATLAB is case sensitive, so \( A \) and \( a \) represent different variables
Initializing Variables: Explicit Lists

- Enclose values within brackets: \( \text{a} = [3.5]; \)
- Values are typically entered by row, with rows separated by semicolons: \( \text{c} = [-1, 0, 0; 1, -1, 0; 0, 0, 2]; \)
- Omitting the final semicolon causes MATLAB to automatically print the matrix value:
  \[ \text{c} = [-1, 0, 0; 1, -1, 0; 0, 0, 2] \]
  \[
  \begin{pmatrix}
  -1 & 0 & 0 \\
  1 & -1 & 0 \\
  0 & 0 & 2 \\
  \end{pmatrix}
  \]

- Each row can be listed on a separate line:
  \[ \text{b} = [-1, 0, 1 \\
  1, 2, 1 \\
  3, 1, 2 \\
  4, 0, 4]; \]

- Long rows can be continued on the next line through the use of a comma and three periods (an ellipsis):
  \[ \text{f} = [1, 52, 64, 197, 42, -42, \ldots \\
  55, 82, 22, 109] \]

- Elements of a matrix can be changed individually by referring to a specific location:
  - If \( \text{S} = [5, 6, 4] \)...
  - ...we can change the second element of \( \text{S} \) from 6 to 8 by issuing the command \( \text{S}(2) = 8 \)

- We can define a matrix using previously defined matrices. For example, if \( \text{S} = [5, 6, 4] \), we can do the following:
  \[ \text{b} = [3, \text{S}, 2] \]
Saving and Loading Individual Variables

- `.mat` files are the default format used when issuing the `save` command
  - Compact format which conserves disk space
  - Cannot be easily exported to other application software
- General form of the `save` command
  - `save <fname> <vlist> -option1 -option2..., etc.`
  - *Examples:*

<table>
<thead>
<tr>
<th>Operation</th>
<th>MATLAB Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save variable ( m ) in MATLAB file named <code>file.mat</code></td>
<td><code>save file m</code></td>
</tr>
<tr>
<td>Save variable ( m ) in file named <code>file.dat</code> using 8 digit precision/text format</td>
<td><code>save file.dat m -ascii</code></td>
</tr>
<tr>
<td>Save variable ( m ) in file named <code>file.dat</code> using 16 digit precision/text format</td>
<td><code>save file.dat m -ascii -double</code></td>
</tr>
<tr>
<td>Save variable ( m ) in file named <code>file.dat</code> using 16 digit precision/text format with individual elements delimited by tabs</td>
<td><code>save file.dat m -ascii -double -tabs</code></td>
</tr>
</tbody>
</table>

- ASCII (text) files can be viewed, modified, or prepared using programs like *WordPad* or *NotePad* in the *Windows* environment, or *vi* in the UNIX environment
- ASCII files are formatted such that each row of a matrix is contained on a separate line
The Colon (:) Operator

• Use in place of an index to represent all elements in a row or column of a previously defined matrix

\[
S = \begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
10 & 11 & 12 \\
\end{bmatrix}
\]

\[
R = S(4,:) \quad \text{all elements in fourth row of } S
\]

• Use to generate vectors containing increasing or decreasing sequences of numbers

\[
A = 0:2:8
\]

• Use to select a submatrix from a previously defined matrix

Assume \( C = \begin{bmatrix}
-1 & 0 & 0 \\
1 & 1 & 0 \\
1 & -1 & 0 \\
0 & 0 & 2 \\
\end{bmatrix} \)

Issuing the commands

\[
C1 = C(:,2:3)
\]

\[
C2 = C(3:4,1:2)
\]

results in the following matrices:

\[
C1 = \begin{bmatrix}
0 & 0 \\
1 & 0 \\
-1 & 0 \\
0 & 2 \\
\end{bmatrix}
\]

\[
C2 = \begin{bmatrix}
1 & -1 \\
0 & 0 \\
\end{bmatrix}
\]
• **Transpose Operator**: The *transpose* of $A = A'$ and represents a new matrix in which the rows of $A$ are transformed into the columns of $A'$

```matlab
>> a=[4,2,3;2,1,5]   a =
    4   2   3
    2   1   5
>> a'
ans =
    4   2
    2   1
    3   5
```

• **Empty Matrix**: A matrix which does not contain any elements, e.g.

```matlab
>> a=[]  a =
        []
```

• **User Input**:
  • The *input* command displays a text string, and waits for a typed response
  • Value entered is stored in the specified variable
  • Matrices must be entered from the keyboard using the correct syntax
  • Note that this command is most useful when running MATLAB *scripts* (a sequence of MATLAB commands which can be run over and over)

```matlab
>> z=input('Enter a value for the matrix z:');
  Enter a value for the matrix z:[4,4]
  user response
  MATLAB response
    z =
    4   4
```
MATLAB Environment

Printing Matrices

• Simplest way: enter the name of the matrix
  • Name of the matrix will be repeated
  • Contents of the matrix will be printed starting on the next line
    
    \[
    \text{» } a
    \]

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

    \text{MATLAB response}

• Format commands
  • Changes how numbers are displayed
  • Your chosen format mode “sticks” until another format command is issued

<table>
<thead>
<tr>
<th>MATLAB Command</th>
<th>Display Mode</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>format short</td>
<td>default</td>
<td>15.2345</td>
</tr>
<tr>
<td>format long</td>
<td>14 decimals</td>
<td>15.234533333333333333333333333333</td>
</tr>
<tr>
<td>format short e</td>
<td>4 decimals</td>
<td>1.5235e+01</td>
</tr>
<tr>
<td>format long e</td>
<td>15 decimals</td>
<td>1.523453333333333333333333333333e+01</td>
</tr>
<tr>
<td>format bank</td>
<td>2 decimals</td>
<td>15.23</td>
</tr>
<tr>
<td>format +</td>
<td>Prints the sign only (not the value)</td>
<td>+</td>
</tr>
<tr>
<td>format compact</td>
<td>Suppresses line feeds</td>
<td></td>
</tr>
<tr>
<td>format loose</td>
<td>Turns off format compact mode</td>
<td></td>
</tr>
</tbody>
</table>
MATLAB Environment

Printing Matrices (continued)

• The `disp` command
  • Command argument is enclosed in parentheses
    • Matrix: `disp(A)`
    • Character string: `disp('A')`
  • Prints the command argument (matrix value or text) on the screen:

```
» disp(a)
   4   5   6  -1
   2   4   5   1

» disp('hi')
   hi
```

• The `fprintf` command
  • Similar to the `fprintf()` function in ANSI C
  • Allows precise specification of the print format and line spacing when printing both text and matrix values
Simple XY Plots

- Allows the generation of scatter (x vs. y) plots
- Column matrices are used to hold each set of values
- The plot can be enhanced by adding a grid, titles and axis labels
- General format: `plot(x,y)` where x and y are each \( m \)-element vectors
- Line plots (y versus index) can be generated by including only one argument in the plot command
- Example:

```matlab
a=[1;2;3;4;5;6;7];
b=[1;4;9;16;25;36;49];
plot(a,b),title('Squares'),xlabel('number'),ylabel('number squared'),grid
```
MATLAB Environment

Simple XY Plots (continued)

- MATLAB plot commands

<table>
<thead>
<tr>
<th>Plot Command</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot(x,y)</td>
<td>Generates a scatter plot of x vs. y on linear axes</td>
</tr>
<tr>
<td>semilogx(x,y)</td>
<td>Generates a scatter plot of x vs. y using a logarithmic scale for x and a linear scale for y</td>
</tr>
<tr>
<td>semilogy(x,y)</td>
<td>Generates a scatter plot of x vs. y using a linear scale for x and a logarithmic scale for y</td>
</tr>
<tr>
<td>loglog(x,y)</td>
<td>Generates a scatter plot of x vs. y using a logarithmic scale for both x and y</td>
</tr>
</tbody>
</table>

- Multiple plots on one axis (three methods)
  - *hold* allows a second curve to be plotted on existing axes
  - Include multiple sets of arguments in a plot command, e.g. *plot(x,y,w,z)*. Here, x vs. y and w vs. z curves will be generated on the same plot
  - Use *plot(A)*, where A is a matrix. A separate curve will be plotted for each column

- Plot Style
  - *plot(x,y,'o')* plots x-y points using the circle (o) mark.
    Other line and point options include the point(.), plus(+), star(*), x-mark(x), dashed(--), and dotted(:)
  - The *axis* command allows the current axis scaling to be frozen for subsequent plots.
  - *axis(v)* allows user-specified plot ranges. v is a four element vector containing scaling values [xmin,xmax,ymin,ymax]
Scalar and Array Operations

- MATLAB scalar calculations obey standard algebraic precedence (order of operations)
- Arithmetic operations between two scalars $a$ and $b$:

<table>
<thead>
<tr>
<th>Operation</th>
<th>MATLAB Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>addition</td>
<td>$a + b$</td>
</tr>
<tr>
<td>subtraction</td>
<td>$a - b$</td>
</tr>
<tr>
<td>multiplication</td>
<td>$a \times b$</td>
</tr>
<tr>
<td>division</td>
<td>$a / b$</td>
</tr>
<tr>
<td>exponentiation</td>
<td>$a ^ b$</td>
</tr>
</tbody>
</table>

- Array operations: Element-by-element operations between two matrices of the same size
- Note that array operations and matrix operations are not equivalent!

<table>
<thead>
<tr>
<th>Operation</th>
<th>MATLAB Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>addition</td>
<td>$a + b$</td>
</tr>
<tr>
<td>subtraction</td>
<td>$a - b$</td>
</tr>
<tr>
<td>multiplication</td>
<td>$a \times b$</td>
</tr>
<tr>
<td>division</td>
<td>$a / b$</td>
</tr>
<tr>
<td>exponentiation</td>
<td>$a ^ b$</td>
</tr>
</tbody>
</table>

- Example array operation:

<table>
<thead>
<tr>
<th>$A =$</th>
<th>$B =$</th>
<th>$A \times B =$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 4 2</td>
<td>0.3333 0.2000 3.0000</td>
<td>1.0000 0.8000 6.0000</td>
</tr>
<tr>
<td>2 1 5</td>
<td>5.0000 2.0000 1.0000</td>
<td>10.0000 2.0000 5.0000</td>
</tr>
</tbody>
</table>
**Special Scalar Values**

- Predefined values which are available for use by MATLAB
- Redefining these values in MATLAB could cause unexpected results

<table>
<thead>
<tr>
<th>Special Scalar</th>
<th>What it Represents</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi$</td>
<td>$\Pi$</td>
</tr>
<tr>
<td>$i,j$</td>
<td>imaginary operator (square root of minus one)</td>
</tr>
<tr>
<td>$\text{Inf}$</td>
<td>infinity</td>
</tr>
<tr>
<td>$\text{NaN}$</td>
<td>Not a number. Occurs when the results of a calculation are undefined</td>
</tr>
<tr>
<td>$\text{clock}$</td>
<td>Current time</td>
</tr>
<tr>
<td>$\text{date}$</td>
<td>Current date</td>
</tr>
<tr>
<td>$\text{eps}$</td>
<td>The smallest amount by which two values can differ in the computer</td>
</tr>
<tr>
<td>$\text{ans}$</td>
<td>A computed value not assigned to a particular variable</td>
</tr>
</tbody>
</table>

**Special Matrices**

<table>
<thead>
<tr>
<th>MATLAB Matrix Command</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{zeros}(m,n)$</td>
<td>Generates an m by n matrix of all zeros</td>
</tr>
<tr>
<td>$\text{ones}(m,n)$</td>
<td>Generates an m by n matrix of all ones</td>
</tr>
<tr>
<td>$\text{zeros}(m)$</td>
<td>Generates an m by m square matrix of zeros</td>
</tr>
<tr>
<td>$\text{ones}(m)$</td>
<td>Generates an m by m square matrix of ones</td>
</tr>
<tr>
<td>$\text{eye}(m)$</td>
<td>Generates an m by m identity matrix</td>
</tr>
<tr>
<td>$\text{diag}(A)$</td>
<td>Puts the diagonal elements of matrix A into a column vector</td>
</tr>
<tr>
<td>$\text{diag}(V,0)$</td>
<td>Creates a matrix with the elements of vector V on the diagonals</td>
</tr>
</tbody>
</table>
Control System Toolbox

- Toolboxes are available for MATLAB to simplify specific tasks. We will use the Control System Toolbox in this class.
- Useful functions in the toolbox:

<table>
<thead>
<tr>
<th>Function call</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tf(num,den)</code></td>
<td>Creates a system model with the specified transfer function</td>
</tr>
<tr>
<td><code>impulse(sys)</code></td>
<td>Calculates the impulse response of the system model <code>sys</code></td>
</tr>
<tr>
<td><code>step(sys)</code></td>
<td>Calculates the step response of the system model <code>sys</code></td>
</tr>
<tr>
<td><code>lsim(sys,u,t)</code></td>
<td>Calculates the response of the system model <code>sys</code> to an arbitrary input signal</td>
</tr>
<tr>
<td><code>bode(sys)</code></td>
<td>Bode plot for the system model <code>sys</code></td>
</tr>
</tbody>
</table>

Other Useful Functions

<table>
<thead>
<tr>
<th>Function call</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>residue(num,den)</code></td>
<td>Calculates the partial fraction expansion of the specified ratio of polynomials</td>
</tr>
<tr>
<td><code>conv(a,b)</code></td>
<td>Polynomial multiplication</td>
</tr>
<tr>
<td><code>roots(a)</code></td>
<td>Calculates the roots of a polynomial</td>
</tr>
</tbody>
</table>