The MATLAB Environment

Characteristics

• In the ECE PC lab we have the full professional version of MATLAB installed (version 5.x)
• The version you buy at the bookstore is the student edition (version 5.x), marketed by Prentice Hall
• Student version 5 contains the following:
  – Complete toolboxes for signal processing, controls, and symbolic math
  – Tools for user interface building
  – Integrated debugger, editor, and profiler
  – The MATLAB notebook interface for Microsoft Word MATLAB document generation
  – The student tutorial guide
  – Complete on-line documentation and help system
  – Full functional compatibility with the professional version except matrix sizes are limited 16,384 elements (128x128) square matrix; the number of these elements is limited by system RAM
  – In the professional version each element can be as large as
desired, limited only by system RAM

– On Windows 95 16 MB or greater RAM is recommended, and the disk space varies from 25 MB to 50 MB with optional help files, and the disk’s cluster size

Scalars, Vectors, and Matrices

• MATLAB was originally developed to be a matrix laboratory

• In its present form it has been greatly expanded to include many other features, but still maintains inherent matrix and vector based arithmetic at its core

• The basic data structure of MATLAB is the matrix

\[
A = \begin{bmatrix}
  a_{11} & a_{12} & a_{13} \\
  a_{21} & a_{22} & a_{23} \\
  a_{31} & a_{32} & a_{33}
\end{bmatrix},
B = \begin{bmatrix}
  b_{11} & b_{12}
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
  c_{11} \\
  c_{21}
\end{bmatrix},
D = \begin{bmatrix}
  d_{11}
\end{bmatrix}
\]  \hspace{1cm} (2.1)

– A matrix with a single row is also known as a row vector

– A matrix with a single column is also known as a column vector

– A matrix with just one row and one column (a single element) is simply a scalar
Variable Initialization

- Variable names in MATLAB
  - Must be begin with a letter
  - May contain digits and the underscore character
  - May be any length, but must be unique in the first 19 characters

- A common way to create a matrix instance is by simply assigning a list of numbers to it, e.g.
  
  ```
  A = [3.1415]; % A 1x1 matrix
  B = [23, 35.3]; % A 1x2 matrix
  C = [1, 2, 3; 4, 5, 6; 7, 8, 9]; % a 3x3 matrix
  ```
  
  - Note: a comma or a blank is used to separate elements in the same row, while a semicolon is used to separate rows
  - The assignment statements listed above are terminated with a semicolon to suppress MATLAB from echoing the variables value

- To continue one line into another we can break a line where a comma occurs, and then follow the comma with an ellipsis (three periods in a row), e.g.,
  
  ```
  M = [.1, .2, .3, .4, .5, .6, .7, .8, .9, 1.0]; % or break the line
  M = [.1, .2, .3, .4, .5, ...
       .6, .7, .8, .9, 1.0];
  ```

- One matrix may be used to define another matrix, e.g.,
  
  ```
  A = [4, 5, 6];
  B = [1, 2, 3, A]; % creates
  B = [1, 2, 3, 4, 5, 6];
  ```
The Colon Operator

- To make matrix generation and addressing easier we use the colon operator

- The colon operator is indeed powerful, and mastering it is essential to becoming a MATLAB expert

  - Here we use it to generate row vectors

    \[ k = 0:6; \% \text{ creates} \]
    \[ k = [0, 1, 2, 3, 4, 5, 6]; \]
    \[ t = 0:.25:2; \% \text{ creates} \]
    \[ t = [0, .25, .5, .75, 1, 1.25, 1.5, 1.75, 2]; \]
    \[ s = -10:2:0; \% \text{ creates} \]
    \[ s = [-10, -8, -6, -4, -2, 0]; \]

  - Here we use it to generate row and column slices of a matrix

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

    \[
    A = \begin{bmatrix}
    1 & 2 & 3 \\
    4 & 5 & 6 \\
    7 & 8 & 9
    \end{bmatrix}
    \]

    \[ A_{\text{col2}} = A(:,2); \% \text{ Span all rows with the column index fixed at 2.} \]
    \[ A_{\text{row1}} = A(1,:); \% \text{ With row index fixed at 1, span all columns.} \]
    \[ A_{11} = A(2,2); \% \text{ Produces the scalar } A_{11} = 5 \]

- We can combine the techniques to extract a submatrix of \( A \)

    \[ A_{\text{sub}} = A(2:3,2:3) \% \text{ Extract a sub matrix} \]
    \[ \% \text{ consisting of rows 2-3 and columns 2-3} \]
\[ A_{\text{sub}} = \begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix} \]

- We can swap the rows and columns of a matrix using the **transpose operator**, e.g.,
  
  \[
  A = [1, 2, 3]; \\
  A_{\text{transpose}} = A'; \quad \% \text{ produces} \\
  A_{\text{transpose}} = [1; 2; 3];
  \]

- A simple formatting scheme to have MATLAB display the values of several equal length vectors side-by-side is the following (actual MATLAB command line interaction)

  » A = [0, 1, 2, 3, 4];
  » B = [0, 1, 4, 9, 16];
  » C = [0, 1, 8, 27, 64];
  » \% \text{ Display in side-by-side columns:}
  » [A' B' C'] \% \text{ commas may be included but not needed}

  \[
  \begin{array}{ccc}
  0 & 0 & 0 \\
  1 & 1 & 1 \\
  2 & 4 & 8 \\
  3 & 9 & 27 \\
  4 & 16 & 64 \\
  \end{array}
  \]

**Special Values and Matrices**

- To make life easier, MATLAB comes with several predefined values and matrix generators
  
  – \texttt{pi} represents \(\pi\) in MATLAB floating point precision, e.g.,
  
  » pi

  \[
  \begin{array}{c}
  \text{ans} = 3.1416 \\
  \end{array}
  \]
– \( i, j \) represents the value \( \sqrt{-1} \)

– \( \text{Inf} \) is the MATLAB notation for infinity, i.e., \( 1/0 \)

– \( \text{Nan} \) is the MATLAB representation for \( \text{not-a-number} \); often a result of a \( 0/0 \) operation

– \( \text{clock} \) displays the current time, e.g.,

```matlab
» clock
ans =
1.0e+003 *
1.9970   0.0080   0.0270   0.0230   0.0160   0.0508
```

– \( \text{date} \) is the current date in a string format

```matlab
» date
ans = 27-Aug-1997
```

– \( \text{eps} \) is the smallest floating-point number by which two numbers can differ, e.g.,

```matlab
» eps
ans = 2.2204e-016
```

• A matrix of zeros can be generated with

```matlab
A_3x3 = zeros(3,3); % or
A_3x3 = zeros(3);
B_3x2 = ones(3,2);
C_3x3 = ones(size(A_3x3));
```
In linear algebra the **identity matrix** is often needed

\[
A_{3\times3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B_{3\times2} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}
\]

\[
C_{3\times3} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}
\]

**User Prompting**

- In the future we will write MATLAB programs or **script files** (saved as *.m files)
- Short of making a full graphical user interface (GUI) MATLAB program, which MATLAB can also do, we may simply wish to prompt the user to supply some input data
- When the following is encountered in a MATLAB program, the variable on the left is assigned the value the user enters at the MATLAB prompt
  \[
x = \text{input}('\text{Please enter your height and weight: '});
\]
– We create a simple script file (more on this in the next chapter) `user_prompt`

```matlab
% the *.m file user_prompt.m
x = input('Please enter your height and weight: ');
```

– To run the script we type the file name at the MATLAB prompt (making sure the file is in the MATLAB path)

```matlab
» clear % clear all variables from the workspace
» user_prompt % run the script file user_prompt.m
Please enter your height and weight: [100 200]
» x
x =
 100   200
```

### Output Display/Print Formatting

- Globally the display format for the command window can be changed by:

- Typing commands directly, e.g.,

  ```matlab
  format short % four decimal digits
  format long % 14 decimal digits
  format short e % short with scientific notation
  format long e % long with scientific notation
  % others available
  ```
– Selecting displays formats from the command window pull down menus, e.g.,

```
MATLAB Command Window
```

- Formatted output from script files is also possible, that is the format of variables printed to the screen can be made unique for each variable you want the program to print, and custom text be distributed around the numeric values

- The command `disp()` is used to display text and print the contents of a matrix

```matlab
>> disp('A Text String to Display.'),
A Text String to Display.
>> M = [1, 2; 3, 4];
>> disp(M)
    1     2
    3     4
```
• A fully customizable way of printing both text and matrices is the `fprintf()` command

\[
\text{fprintf(format\_string,matrices)}
\]

• The format string contains text you wish to display along with `format specifiers`, `%e`, `%f`, and `%g`, which tell `fprintf` how to display respective values in the matrices list

  – `%e`, `%f`, and `%g` select exponential, fixed point, or global (either `%f` or `%e` which ever is smaller)

  – `\n` tells MATLAB to start a new line

\[
\begin{align*}
\text{» A = 24.567;} \\
\text{» fprintf('Hello, we have A = %f and pi = %6.4f. \n',...} \\
\text{A,pi)} \\
\text{Hello, we have A = 24.567000 and pi = 3.1416.}
\end{align*}
\]

**The x–y Plot Command**

• A simple `y` versus `x` plot is obtained with the command

\[
\text{plot(x,y)}
\]

• Plots without a title and axis labels are uninformative, so we must also add labels

\[
\begin{align*}
\text{» x = [0, 1, 2, 3, 4, 5, 6 ,7 ,8];} \\
\text{» y = [0, 1, 4, 9, 16, 25 ,36 ,49 ,64];} \\
\text{» plot(x,y)} \\
\text{» grid} \\
\text{» title('The Plot of y = x^2')} \\
\text{» ylabel('The y axis')} \\
\text{» xlabel('The x axis')}
\end{align*}
\]
Data File Commands

- Data file commands are used to save and load files in either standard ASCII text or the more compact MATLAB binary format which uses the *.mat extension.

- The binary .mat format is useful for files which will be used with MATLAB, while the ASCII format is useful when working with other programs, or sharing data with others who may not have MATLAB.
» x = 0:5;
» y = rand(size(x));
» [x' y']
ans =
   0   0.3529
   1.0000   0.8132
   2.0000   0.0099
   3.0000   0.1389
   4.0000   0.2028
   5.0000   0.1987
» save mat_data x y; % creates the file mat_data.mat
» save mat_data.dat x y /ascii; % creates the text file % mat_data.dat

• To verify that these files are valid we first clear the MATLAB work space using the command clear
  » clear
  » whos

• Next we load the .mat file back into the work space
» load mat_data
» whos

Name      Size         Bytes  Class
x         1x6             48  double array
y         1x6             48  double array
Grand total is 12 elements using 96 bytes

• To see how the ASCII file is loaded we again clear the work space
» load mat_data.dat
» whos

Name      Size         Bytes  Class
mat_data   2x6            96  double array
Grand total is 12 elements using 96 bytes
• When loading an ASCII file we simply place each row of the text file into a corresponding matrix row
• The variable name that holds the loaded text file is in this case mat_data
• To recover the data vectors \( x \) and \( y \) from mat_data we must parse them out using the colon operator; left as an exercise