Introduction to MATLAB

William F. Moss*

Files containing MATLAB commands and statements are called M-files and have the (.m) extension. They are plain text files and can be created, edited, and viewed with any text editor. By convention, M-files contain documenting comments at the top. There are two types of M-files: *script* files and *function* files. A script file could also be called a MATLAB program. It contains a sequence of MATLAB commands and statements. If pole.m is a script file, then the MATLAB command `pole` will cause the commands and statements contained in pole.m to be executed. A function file begins with the word function and has inputs and outputs. If rod.m is a function file that takes a matrix input and produces a matrix output, then the MATLAB command `B = rod(A)` will evaluate the function "rod". In this course you will use built-in M-files, M-files downloaded from the course homepage, and your own M-files.

The MATLAB path is a list of directories that MATLAB searches for M-files. You can add directories to the path using the MATLAB "path" function (`help path`). The MATLAB path always contains the current directory, that is, the directory you were in when you started MATLAB. MATLAB function documentation is available using the MATLAB "help" function. For example, to get information about the built-in MATLAB function "plot", use the MATLAB command `help plot`. Note that MATLAB is case sensitive. All built-in MATLAB functions should be typed lower case, contrary to what you see in the help documentation. The MATLAB command `help name` displays the comments that appear at the top of the M-file name.m provided that name.m is in the MATLAB path. If your current directory contains M-files that you downloaded from the course homepage, then the "help" function can be used to obtain information about the downloaded files. As an alternative, you can look at the downloaded files with a text editor, and simply read the comments at the top of the file.

1. MATLAB is an object oriented programming language with essentially only one kind of object, a rectangular numerical matrix. In some situations, special meaning is attached to 1×1 matrices, which are scalars, and to matrices with only one row or column, which are called vectors. MATLAB's built-in functions are generally designed to accept vector or matrix arguments and to return vector or matrix values. For example, using the colon notation (`help colon`) introduced in the Primer, it is easy to generate a row vector of 10 equispaced values from 0 to π as follows. Note that in MATLAB π is written as `pi`.

```
>> n = 10;
>> h = (pi - 0)/(n-1);
>> x = 0:h:pi;
```

*Department of Mathematical Sciences, Clemson University, Clemson, SC 29634–1907, U.S.A. (bmoss@math.clemson.edu).

Here `h` is called the increment used to generate `x`. As an alternative, MATLAB provides the function "linspace" (`help linspace`) which allows `x` to be defined in one line by `x = linspace(0,pi,10)`. Now `y = sin(x)` will generate a row vector of sine values. A plot of the sine function over the interval $[0, \pi]$, can be obtained using the MATLAB "plot" function (`help plot`). Create this plot. From Calculus we know what this plot should look like. What you see doesn't look right. The plot has corners. The problem is that we have not sampled the sine function enough times. Produce a second plot using 100 equispaced x values. Use the MATLAB commands "xlabel" (`help xlabel`), "ylabel" (`help ylabel`), and "title" (`help title`) to label the x and y axes and to add explanatory text at the top of the plot. Print this second plot using the "print" command (`help print`). **Turn in this second plot and the commands you used to create it.**

> **Readme:** The MATLAB command "diary" (`help diary`) can be used to record terminal input and output in a text file. When making plots for this course, be sure to sample the function enough times. This may require some experimentation. Do not print out the vectors used to create your plots except for debugging purposes. If you turn in a plot with corners, you'll lose some points.

2. Users are often required to write their own MATLAB functions. A knowledge of MATLAB arithmetic operations is needed here. Look at the help file on arithmetic (`help arith`). In addition to the matrix operations of addition, subtraction, multiplication, and power (`+, -, *, ^`), note that MATLAB also has array operations `.*` and `.^`. It is important to understand the distinction between the matrix and array operations. I prefer to call the array operations, element-wise operations. Also, note the special role of scalars in matrix multiplication. Using a text editor, create a file poly.m containing the following lines.

```
function y = poly(x)
y = x.^3 - 2*x - 1;
```

The name of the file and the name of the function must coincide. To see happens when function "poly" is given a vector input, use the following MATLAB commands.

```
>> x = linspace(0,1,5)
>> y1 = x.^3
>> y2 = 2*x
>> y = y1 - y2
>> y = y - 1
```

Make sure that you understand the output from each of these 5 lines. **Turn in these command and the output generated.**

Recall that MATLAB will only find your file poly.m if it is in the MATLAB path. If poly.m is in the directory you were in when you started MATLAB, it will be in the MATLAB path. Function "poly" has a zero at -1 and another one close to -0.6. Verify this with a plot or plots. Use the MATLAB "grid" command (`help grid`), to help you see where the plot crosses the x-axis. **Turn in your plot and the commands you used to create it.**

Now look at the help file on the MATLAB function "fzero" (`help fzero`). Use "fzero" to find an approximation to the zero near -0.6. Notice that MATLAB gives the answer to five decimal places. MATLAB's default format is "short". Look at the help page for format (`help format`) to see what other formats are available. Look at your approximate zero in "long" and "long e" formats. **Turn in your commands and the output generated.**

3. Download all the M-files from the course homepage. Use the MATLAB function "rand" (`help rand`) to define a 3×4 matrix. Find a row echelon form for this matrix using the MATLAB functions "ref" and "refmovie" which you downloaded. If you use `format rat` with "refmovie", you'll get rational approximations to all the matrix entries. For problems with entries that are simple fractions (such as the homework problems), this gives you a way to check your hand calculations. Note that "refmovie" takes you through the algorithm one step at a time. **Turn in the output from "ref" and "refmovie" for your matrices.**

4. Construct a linear system of 4 equations in 4 unknowns. This is equivalent to constructing a 4×4 coefficient matrix and a 4×1 right hand side column vector. Solve this system using MATLAB slash notation (`help slash`). If you generate your coefficient matrix using the MATLAB function "rand", it will most likely be nonsingular. If MATLAB reports that the coefficient matrix is singular, modify it until you get a nonsingular matrix. **Turn in your solution and the commands used to generate it.**

   Compute the inverse of your coefficient matrix using the MATLAB function "inv" (`help inv`). Find the the eigenvalues of your coefficient matrix using the MATLAB function "eig". Find the LU factorization of your coefficient matrix using the MATLAB function "lu" (`help lu`). Find the SVD of your coefficient matrix using the MATLAB function "svd" (`help svd`). **Turn in all MATLAB commands and the output generated.**

5. The M-file temp.m which you downloaded from the course homepage defines the temperature in a rod of length L whose left end has been raised to temperature T at the initial time 0. The rod has diffusivity D and its initial temperature is 0. Obtain a surface plot of temperature versus x and t using the function "temp" which you downloaded and the MATLAB function "surf" (`help surf`). Try the following values: $D = 1$, $L = 10$, $T = 100$, $N = 100$. The function "temp" defines the temperature as the sum of a series. The assignment $N = 100$, tells "temp" to use the first 101 terms in the series. Define x to be a row vector from 0 to L with an increment of 0.1. Define t to be a row vector from 0 to 80 with an increment of 1. Obtain a matrix of temperature values from function "temp" and then plot using "surf". The comments in temp.m explain how to do this. Experiment with the MATLAB command "view" (`help view`) and find the best angle from which to view the 3-D plot. Label your axes and give the plot a title. **Turn in your plot and all the commands used to generate it.**

6. In control theory we encounter the linear system of ordinary differential equations

$$\mathbf{x}' = A\mathbf{x} + B\mathbf{u} \tag{1}$$

   where $\mathbf{x}$ is an n-vector function of time, $\mathbf{u}$ is an m-vector function of time, A is a real $n \times n$ matrix, and B is a real $n \times m$ matrix, with $1 \leq m \leq n$ and $\text{rank}(B) = m$. The variable $\mathbf{x}$ is called the *state*, while $\mathbf{u}$ is called the *control*. The matrix A is called the open loop matrix and

the matrix B is called the control matrix. The system is said to be *completely controllable* if the matrix

$$L = \left[B, AB, \ldots, A^{n-m}B\right] \tag{2}$$

has rank n.

Write a MATLAB program called control.m that uses the MATLAB function rand to define a 4×4 matrix A and a 4×2 matrix B, and then computes the matrix L and its rank. Use the MATLAB zeros command to allocate storage for L. This step is not necessary but MATLAB will run faster if you use it. Then use a MATLAB for loop and colon notation to compute L, m columns at a time. Finally, use the MATLAB "rank" (`help rank`) function to compute the rank of L. **Turn in your program control.m and its output.**