

WORKING IN TEAMS of TWO (optional) – WORK TOGETHER ON EACH PROBLEM.

1. Write *m-files* – *script files* and *user-defined functions* – to satisfy the requirements below.
2. Demonstrate each m-file to your instructor.
3. Print out each completed m-file.
4. When you are finished with all questions and m-files, etc., turn them in as one packet.
(If you work in teams, one packet for the team is sufficient).

Minimum Requirements:

- Each m-file should have **comment lines** at the top with your name(s) and the date; e.g.: % Joe Student (see model m-files in Workbook)
- **Use comment lines** to annotate the m-files – so that another person can figure out what each section and/or line does; e.g.: % Plotting the polynomial, etc.
- Make sure that anyone who uses the m-file (besides a member of your team), knows *what to do* when the m-file is run. **Include input prompts and other instructions that guide the user**; i.e., use the **disp** command, etc., to print on-screen instructions. The exception is, of course, a *function*; it is assumed the user knows a *function* exists (e.g., the cosine function).
- Any plot asked for below should be automatically generated by (within) the m-file. In general, the m-file should create the plot, generate a title, and label the axes (including units). You may annotate the plot as necessary with the text tools, either within the m-file or after the fact using the interface. When two or more curves are graphed on the same plot, include a legend or use the text tools to directly label each plot.

Hint: Breathe. You **can** program. *Plan* out what the m-file will need to do – write a flow chart or algorithm – before you type in the lines of MatLab code. Think about what input needs to be obtained from the user (and when). What calculations must MatLab do? What decisions must be made depending on various inputs and the results of various calculations? What output is necessary? Then, consider what MatLab commands are needed to get MatLab to do what you want it to do – whether than is to follow a straight-forward list of steps, or to *interact* with the user.

1. Script m-file: Grade

- a. Write a script m-file (not a function) that gives the letter grade for a student when any numerical grade between 0 and 100 (inclusive) is entered.

The user will enter a grade, and the output should look like:

```
The grade is a B.
```

or something to that effect. Use the grade scale at right.

The grade scale is:

A	$90 \leq \text{score} \leq 100$
B	$80 \leq \text{score} < 90$
C	$70 \leq \text{score} < 80$
D	$60 \leq \text{score} < 70$
F	$\text{score} < 60$

- b. Show your electronic file to the instructor for testing.
c. Print your completed m-file.

2. Script m-file: Quadratic Formula

The solution to the quadratic equation:

$$ax^2 + bx + c = 0$$

is:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- a. Create an m-file that will find the roots of a quadratic equation, and print them out.

Your m-file should:

- ask the user for the values of coefficients a , b and c .
- check if $a = 0$, and solve for the single root.
- print out the roots, real and imaginary.

Your file should also tell the user, *in words*, if:

- the roots are real and distinct (2 real roots).
- the roots are real and the same (1 double root).
- the roots are complex and distinct (2 complex roots).
- there is only one root (e.g., when $a = 0$).

Bonus: Warn the user that there is no solution if a and b are both zero.

- b. Show your electronic file to the instructor for testing.
- c. Print the completed m-file. A complete m-file includes your name and date in comment lines at the top of the m-file. Do not forget to use comment lines throughout the file to explain what key parts of the m-file are supposed to do.

3. Script m-file: Plotting a Polynomial

- a. Create an m-file to plot a polynomial of any degree.

- Ask the user to enter the coefficients of the polynomial as a row vector. For example, if the polynomial is:

$$3x^3 - 7x + 2$$

The user would enter:

$$[3, 0, -7, 2]$$

Make sure the instructions to the user on how to enter the polynomial coefficients are clear.

Hint: In the past, some students have used a loop to enter the polynomial coefficients. This is not using MatLab to the greatest advantage; entering the coefficients as a row vector is doing exactly the same thing.

- Ask the user to enter the endpoints of the interval over which the polynomial is to be plotted, e.g.: x_{min} and x_{max} .

- Use the **polyval** command to evaluate the polynomial at 101 points (minimum) between and including x_{min} and x_{max} . The user should not have to enter the number of points (or increments).

- Plot the polynomial. Make sure the plot has x - and y -labels, and a title.

- b. Show your electronic file to the instructor for testing.
- c. Print your completed m-file.

4. Function: Height of a Projectile

- a. Write a **function** called **height** that gives the height of a ball at time t after it has been thrown straight upward with initial velocity v_o ($v_o > 0$) and with initial height $h_o = 0$.

The equation for the height h (m) of the ball at t seconds is:

$$h = v_o t - 0.5gt^2$$

where $g = 9.81 \text{ m/s}^2$, and v_o is in m/s.

The input arguments of the function should be **vo** and **t**. The first line of the m-file should be:

```
function [h] = height(vo,t)
```

To find the height of a ball with $v_o = 20 \text{ m/s}$, 3 seconds after it is released, the user should type into the *Command Window*:

```
>>height(20,3)
```

- b. Show your electronic file to the instructor for testing.
c. Print your completed m-file.

Optional: write a function that includes **g** as a third input.

5. Script m-file: Compare the Elements of Two Vectors of the Same Length

- a. Write an m-file that will compare two vectors, **u** and **v**, that are entered by the user. The vectors should have the same length.

The m-file should:

- ask the user to enter the two vectors, **u** and **v** (of any length).
- check that the vectors are the same length.
 - If the vectors are not the same length, the m-file should output that fact, and then end.
 - If the two vectors are the same length, output:
 - the length of each vector.
 - the indices (element numbers) and values of the elements of **u** that are *greater* than the corresponding elements in **v**.
The output should be a 2D array, with the index numbers in the first column, and the values of **u** in the second column.

Example: If **u**=[45, 23, 12] and **v**=[30,80,-15], the output should look something like:

The length of the vectors is 3.

The indices and values of elements in vector **u** that are greater than those of vector **v** are:

1	45
3	12

- b. Show your electronic file to the instructor for testing.
c. Print your completed m-file.

6. Script m-file: Resistors in Series and In-Parallel

When n electrical resistors are connected *in series* (one after another), the equivalent resistance is:

$$R_s = R_1 + R_2 + R_3 + \dots + R_n = \sum_{k=1}^n R_k$$

When electric resistors are connected *in parallel*, the equivalent resistance is:

$$R_p = \left[\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \dots + \frac{1}{R_n} \right]^{-1} = \left[\sum_{k=1}^n \frac{1}{R_k} \right]^{-1}$$

- a. Write an m-file that computes and outputs equivalent resistance for n resistors *in series*, and for the same n resistors *in parallel*.

The m-file should:

- Ask the user to enter the values of the resistors. You may either:
 - have the user enter the number of resistors, n , and then have the user enter values for each resistor

individually. You will need to create a loop structure to cycle through n resistors.

OR, more efficiently:

- have the user enter the values for all the resistors in a vector.
- calculate R_s and R_p . You may do this using loop structures or using array operations (which is more efficient?). If you use a loop structure, do not forget to clear R_s and R_p at the start of the m-file, or you may data from before.
- output R_s and R_p .

Make sure the user knows how to enter the information by displaying instructions and input prompts.

- b. Show your electronic file to the instructor for testing.
c. Print your completed m-file.

7. Friction

The horizontal force F required to get a stationary block on a rough horizontal surface to move is:

$$F = \mu mg$$

where μ (Greek “mu”) is the coefficient of static friction, m is mass, and g is the acceleration of gravity.

- a. Write an m-file that computes and prints the force F (in newtons, N).

The file should ask the user for:

- the mass of the object (in kg).
- the *materials* that the block and rough surface are made of (to determine the coefficient of friction) – see the table at right for the FOUR options that you have.

Gravity g can be defined in the m-file ($g = 9.81 \text{ m/s}^2$), or you can include g as a user input.

Note that the user does not need to enter μ , only the material pair option (1, 2, 3 or 4).

Coefficient of Static Friction μ

Option #	Materials	μ
1	Rubber/Concrete:	0.70
2	Metal/Wood	0.40
3	Wood/Wood	0.35
4	Metal/Metal	0.20

Use the **switch** command for selecting μ (“mu”).

Make sure the user knows how to use the program.

- b. Show your electronic file to the instructor for testing.
c. Print your completed m-file.

End.