General Instructions

Due Date: Sunday, October 9th by 11:59pm (submit via Zybooks)

Assignment Summary Instructions:
This assignment has one problem, summarized below. You will use MATLAB as a tool to solve the problem for the given test cases, ensuring that your code is flexible for any additional test cases that might be used to evaluate it.

- Finding Zeros of a Polynomial (Application of Numerical Methods)

zyBooks Submission Instructions:
After completing this assignment in MATLAB, to receive credit, you must submit your code in Zybooks. The following components must be submitted under the specified chapter of the course Zybooks before the deadline to receive credit.

- Chapter 33.1 MA4: FindZero Function
- Chapter 33.2 MA4: Main Script

To submit your script, copy and paste your code into the submission window, making sure to remove any housekeeping commands. You may submit to Zybooks as many times as you want before the deadline, without any penalty. The highest score attained before the deadline will be graded. All components are due before the due date. No credit will be given if it is not submitted through the Zybooks platform before the deadline. Credit for each component will be awarded based upon the percentage of successfully completed assessments.

Explanation of P-Code:
Under the Additional Resources folder accompanying this prompt, you will find a file named **FindZero_Solution.p**. This is a working solution to the FindZero function required to complete this problem. This file is unopenable and the contents can't be read in a text editor, but it can be called like a typical .m function file in MATLAB. If you get stuck and are unable to successfully develop your own FindZero function, you are encouraged to instead copy FindZero_Solution.p to your working folder and rename it **FindZero.p**. Now, when your main script calls the FindZero function, it will automatically call the .p file, and you will now have an opportunity to successfully finish developing your main script.

Proficiency Time: Times are included with the Background and Task sections. These times are the estimated amount of time it should take you to **redo** an assignment once you are fully proficient in material that it covers. To practice, reread the background in the given Comprehension Time and attempt to complete the problem in the given Proficiency Time.

Academic Honesty Reminder

The work you submit for this assignment should be your work alone. You are encouraged to support one another through collaboration in brainstorming approaches to the problem and troubleshooting. In this capacity, you are permitted to view other students' solutions, however, copying of another student's work is strongly discouraged.

This assignment will be checked for similarity using a MATLAB code. The similarity code will check each submission for likeness between other student submissions, past student submissions, the solution manual, and online resources and postings. If your submission is flagged for an unreasonably high level of similarity, it will be reviewed by the ENGI 1331 faculty, and action will be taken by faculty if deemed appropriate.

NOTE: Since this is an automated system for all sections, if any of your work is not your own, you will be caught. Changing variable names, adding comments, or spacing will not trick the similarity algorithm.

Background: Comprehension Time: 10 – 15 min

Your coworker has taken several volume measurements of a gas at various temperatures. The measurements all correlate to odd order polynomials (i.e. 1st, 3rd, 5th…), and your coworker has already fit these polynomials to the data. She now needs your help creating a program to find the volume of the gas when the temperature is at 0°C. You will be provided with the coefficients for the polynomial fit and need to solve for the volume at 0°C. In other words, you must find the root (zero) of the polynomial.

Knowing that all odd order polynomials have at least 1 real root (roots are values where the equation is equal to zero), you have decided to use the Newton-Raphson method to find the root of the polynomial. The Newton-Raphson method is an iterative method of finding roots which utilizes the derivative of a given polynomial to approach the root of that polynomial with each successful iteration. The Newton-Raphson method is defined as

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})} \quad \text{for } n \geq 1 \qquad \text{Eq. 1 (General)}$$

where n is any whole number greater than or equal to 1. To approximate the root r of a function, one can take an initial guess x_0 and use Equation 1 to find x_1, then plug x_1 into Equation 1 to find x_2, etc., repeatedly, until reaching an x_n where $x_n = r$ to some predefined precision.

A simple tutorial of the Newton-Raphson method can be found at http://www.sosmath.com/calculus/diff/der07/der07.html

Below are some examples of the temperature polynomials your coworker has provided, where each *C* is a coefficient of a polynomial (you could be given higher order polynomials as well). Your code should be flexible to any odd order polynomials. Assume that each given polynomial will only have one real root.

1st order polynomial	$T(V) = C_1V + C_2$
3rd order polynomial	$T(V) = C_1V^3 + C_2V^2 + C_3V + C_4$
5th order polynomial	$T(V) = C_1V^5 + C_2V^4 + C_3V^3 + C_4V^2 + C_5V + C_6$

Using the variables shown in the temperature functions above, the Newton-Raphson method can be given as

$$V_n = V_{n-1} - \frac{T(V_{n-1})}{T'(V_{n-1})}, \quad \text{for } n \geq 1 \qquad \text{Eq. 1 (Applied)}$$

This form is equivalent to Equation 1 and simply replaces x and f(x) with V and T(V), respectively.

NOTE: Two helpful functions when working with polynomials and derivatives are polyval() and polyder().

Tasks: Proficiency Time: 35 – 60 min

TASK 1: (8 – 12 min)
Prompt the user to enter a vector of coefficients for the polynomial model. Verify that the entry has an even number of elements (an odd number of elements would mean an even order polynomial). If an invalid vector is entered, prompt the user to re-enter the vector until an acceptable vector is entered. If the user does not enter an acceptable vector after 5 attempts (including the first prompt), display a warning and remove the last element of the last vector entered. (For example, if the last user input is [1 2 3 4 5], the vector becomes [1 2 3 4]). Save the validated vector of coefficients to **MA4_Polynomial.csv**.

TASK 2: (5 – 10 min)
Prompt the user to enter a starting guess V_0 to use as the first value for the Newton-Raphson method. If this guess causes the derivative of the temperature function to be zero (i.e. if $T'(V_0) = 0$), the method will immediately fail due to a division by zero. Check if the initial guess will cause the method to fail. If so, prompt the user to enter a new initial guess until the user enters a valid value. If the user does not enter an acceptable guess after 3 attempts, produce an error and terminate the program. Finally, prompt the user to enter a percent error that will be used to determine the precision of the Newton-Raphson method.

TASK 3: (15 – 25 min)
In order to approximate the root of the polynomial entered in Task 1, create a user-defined function called **FindZero.m** that applies the Newton-Raphson method to the polynomial. This function should have three inputs: the vector of coefficients entered in Task 1, the starting guess entered in Task 2, and the percent error that was entered in Task 2. Your function should repeatedly apply the Newton-Raphson method until the percent error between the two most recent iterations is less than the percent error entered by the user in Task 2. Determine the percent error between the two most recent iterations using the formula

$$Percent\ Error\ [\%] = \left|\frac{V_n - V_{n-1}}{V_{n-1}}\right| * 100\ [\%] \qquad \text{Eq. 2}$$

Your function will return two outputs: the final value of the root of the polynomial and the number of iterations the method took to converge to that value. Additionally, if during any iteration the derivative of the temperature function would equal zero, produce an error and terminate the program. From the main script, use your function to output the root of the polynomial and number of iterations to the command window.

TASK 4: (7 – 13 min)
Prompt the user to select whether to repeat the Newton-Raphson calculation (starting at Task 2) or end the program. If the user exits out of the menu, repeat the request until the user makes a selection. If the program is repeated, store the new final value of the root and new number of iterations in the same variables without overwriting the previous results. Once the user chooses to end the program, save the variables containing the root values and numbers of iterations as **MA4_Results.mat**.

HINT: The length of the final value of the root and number of iterations variables should equal the number of times the root is computed (the number of times Tasks 3 and 4 are run).

Sample Output

Sample of Task 1 and 2 data validations, given the 3rd order polynomial: $T(V) = V^3 - V^2 - V - 1$

```
Command Window
Enter a vector of coefficients for an odd-order polynomial: [1 -1 -1 -1 5]
Enter a vector of coefficients for an odd-order polynomial: [1 -1 -1 -1 5]
Enter a vector of coefficients for an odd-order polynomial: [1 -1 -1 -1 5]
Enter a vector of coefficients for an odd-order polynomial: [1 -1 -1 -1 5]
Enter a vector of coefficients for an odd-order polynomial: [1 -1 -1 -1 5]
Warning: Odd number of coefficients entered. Last element of user input removed.
> In MA5_cougarnetID (line 10)

Enter an initial guess for the Newton-Raphson method: 1
Enter an initial guess for the Newton-Raphson method: 1
Enter an initial guess for the Newton-Raphson method: -1/3
Error using MA5_cougarnetID (line 29)
Initial guess causes a divide by 0 error. Exiting program.
```

Note that $T'(1) = T'\left(-\frac{1}{3}\right) = 0$

Given the 1st order polynomial: $T(V) = 5.4\,V - 37$

```
Command Window
Enter a vector of coefficients for an odd-order polynomial: [5.4 -37]

Enter an initial guess for the Newton-Raphson method: 19
Enter the percent difference to compute to [0.1% - 10%]: 2
The volume at T=0 for the given equation is 6.85185 [m^3].
The solution converged in 2 iterations.
```

Sample of repeated calculations, given the 3rd order polynomial: $T(V) = V^3 + 2.6\,V^2 - 17.8$

```
Command Window
Enter a vector of coefficients for an odd-order polynomial: [1, 2.6, 0, -17.8]

Enter an initial guess for the Newton-Raphson method: 0.2
Enter the percent difference to compute to [0.1% - 10%]: 1
The volume at T=0 for the given equation is 1.97301 [m^3].
The solution converged in 8 iterations.

Enter an initial guess for the Newton-Raphson method: 1
Enter the percent difference to compute to [0.1% - 10%]: 0.1
The volume at T=0 for the given equation is 1.97293 [m^3].
The solution converged in 5 iterations.
fx >>
```

Repeat the Newton-Raphson calculation?
Yes
No

Repeat the Newton-Raphson calculation?
Yes
No

Variables stored in **MA4_Results.mat**

	1	2
1	1.9730	1.9729

	1	2
1	8	5

Given the 5th order polynomial: $T(V) = 0.2\,V^5 + 0.5\,V^4 + 3.1\,V^2 + 2.15\,V - 67.3$

```
Command Window
  Enter a vector of coefficients for an odd-order polynomial: [0.2 0.5 0 3.1 2.15 -67.3]

  Enter an initial guess for the Newton-Raphson method: 0.4
  Enter the percent difference to compute to [0.1% - 10%]: 1
  The volume at T=0 for the given equation is 2.53873 [m^3].
  The solution converged in 11 iterations.
```