**ENED 1090: Engineering Models I**

**Homework Assignment #2**

**Due: Week of September 14th at the beginning of your Recitation Section**

**Problem 1: 1-d Arrays (Vectors)**

1. What MATLAB command will create a vector, t, that starts at 0, increments by 0.25, and ends at 2? Execute the command and display results.

MATLAB Command & Results:

**Command: t = 0:.25:2**

**Results: t =**

**0 0.2500 0.5000 0.7500 1.0000 1.2500 1.5000 1.7500 2.0000**

1. What MATLAB command will determine how many entries are in vector t? Execute the command and display results.

MATLAB Command & Results

**Command: length(t)**

**Results: ans =**

**9**

*Remember, t refers to all of the entries in the vector t. To get to specific entries, we need* ***to index into the vector****. For example, t(3) is the 3rd entry in the vector t.* ***Use indexing for the following problems****.*

1. What MATLAB command will show the 5th entry in the vector t? Execute the command and display results.

MATLAB Command & Results:

**Command: t(5)**

**Results: ans =**

**1**

1. What MATLAB command will show entries 5 through 8 in the vector t? Execute the command and display results.

MATLAB Command & Results:

**Command: t(5:8)**

**Results: ans =**

**1.0000 1.2500 1.5000 1.7500**

1. What MATLAB command will change entry 2 in vector t to 100? Execute the command and display results.

MATLAB Command & Results:

**Command: t(2) = 100**

**Results: t =**

**0 100.0000 0.5000 0.7500 1.0000 1.2500 1.5000 1.7500 2.0000**

1. What ***single*** MATLAB command will change entries 2 through 4 in vector t to 1000? ? Execute the command and display results.

MATLAB Command & Results:

**Command: t(2:4) = 1000**

**Results: t =**

**1.0e+03 \***

**0 1.0000 1.0000 1.0000 0.0010 0.0013 0.0015 0.0018 0.0020**

**t =**

**0 1000 1000 1000 1.0 1.25 1.5 1.75 2.0**

**Results Explained: When t(2:4) = 1,000 Matlab took out 1\*10^3. When the smaller numbers, everything except 1,000, had this 1\*10^3 taken out they were taken down to four decimal places and then rounded so multiplied back up they would not be exactly the previous. However fprintf command with a %0.5f will get the original number when displaying output as there will then be five decimal places and thus the 5th will be rounded from the 6th and because the 6th is non-existent in this case, the numbers would be as they were originally, except for 1,000.**

**Problem 2: Curve Fitting**

The purpose of this question is to appreciate how a curve fit can be used as a MODEL to predict future values. You will also see that fitting a higher-order polynomial to data will not necessarily do a better job at MODELING future values. You are provided with an Excel file (HW2\_WorldPopulation.xlsx) which provides the world population from 1950 to 2000. Similar to Lab 3, import this data and go through the first 3 steps for fitting a polynomial to the data: **Step 1:** Import Raw Data and adjust the Year vector to start at 0 (Year = Year – 1950), **Step 2:** Plot Raw Data to make sure you have imported the correct data, **Step 3**: Find the curve of best fit. You will do Step 3 for a 1st, 3rd, and 5th order polynomial.

*Note: For the 3rd and 5th order polynomials, some of the coefficients may be displayed as 0.0000 because they are so much smaller than the other coefficients in the polynomial. To actually see these values, use the MATLAB command:* ***format shorte*** *which will display the coefficients in scientific notation. Simply put this command in your script file before displaying the coefficients. To revert back to the standard display in MATLAB, simply use* ***format short****.*

**Paste the equation for the fitted polynomial below:**

**1st Order Polynomial fitted equation:**

**Poly1st = 72.8x + 2336.3**

**3rd Order Polynomial fitted equation:**

**Poly3rd = (-9.2572e-03)x^3 + (1.0954e+00)x^2 + (3.8974e+01)x + (2.5546e+03)**

**5th Order Polynomial fitted equation:**

**Poly5th = (-3.9595e-06)x^5 + (5.6365e-04)x^4 - (3.8022e-02)x^3 + (1.7195e+00)x^2 + (3.3811e+01)x + (2.5640e+03)**

Once the coefficients are found, use the *polyval* command to estimate the population in 2013. This is 63 years after 1950. In Lab 3 we substituted the Year vector into the *polyval* command which provided us a set of values that could be plotted as the trend line. For this problem, you are just substituting in a single value (63) to estimate a future value. Fill out the following table for estimated values based on fitting a 1st, 3rd and 5th order polynomial to the raw data. The actual population for 2013 is known and is provided in the table.

The percent error can be found using the following equation:

Error = *abs*((Estimated – Actual)/Actual)\*100

where *abs* is the absolute value command in MATLAB®, type **help abs** in the command window if you need more information about what this command is doing.

|  |  |  |  |
| --- | --- | --- | --- |
| **Order of Fitted Polynomial** | **Estimated Population**  **(millions)** | **Actual Population in 2013**  **(millions)** | **Percent Error (%)** |
| **1st** | **6921** | 7089 | **2.37%** |
| **3rd** | **7043** | 7089 | **0.649%** |
| **5th** | **6961** | 7089 | **1.81** |

Finally, extrapolation of data (going outside the range of provided data) is a step that must be done very carefully. There is no guarantee that the estimate will be accurate. This can be related to things like weather models which try to predict the weather for the next day, 5 days or 10 days. Usually the model is pretty accurate for the next day and becomes less accurate the farther away you get from the data. The same would be true for the world population data.

**Paste your script file for this problem below:**

year = xlsread('HW2\_WorldPopulation.xlsx', 'A2:A52');

population = xlsread('HW2\_WorldPopulation.xlsx', 'B2:B52');

year = year - 1950;

%1st Order

poly1st = polyfit(year,population, 1);

yfit1st = polyval(poly1st,year);

pop\_1\_2013 = polyval(poly1st, 63);

%3nd Order

poly3rd = polyfit(year, population, 3);

yfit3rd = polyval(poly3rd,year);

pop\_3\_2013 = polyval(poly3rd, 63);

%5th Order

poly5th = polyfit(year, population, 5);

yfit5th = polyval(poly5th,year);

pop\_5\_2013 = polyval(poly5th, 63);

%Plotting

subplot(3,1,1);

plot(year,population,'b--');hold on

plot( year, yfit1st,'r');

subplot(3,1,2)

plot(year,population,'b--');hold on

plot(year, yfit3rd, 'r');

subplot(3,1,3)

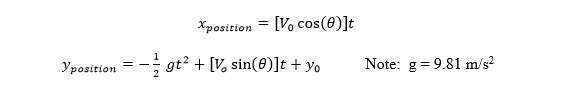
plot(year,population,'b--');hold on

plot(year, yfit5th, 'r');

**Problem 3: Curve Fitting – Trajectory of Projectile**

The excel file, ProjectileData, has three columns of data: time, range, and height. Import each of these columns into MATLAB using either xlsread or the Import Data tool. The variable distance represents measurements of the x-position (horizontal position) of the projectile over time and the variable height represents measurements of the y-position (vertical position) of the projectile over time.

The equations for the x and y position (in meters) of a projectile launched at an angle of θ (rad or degrees) with an initial velocity of V0 (m/s) are:



1. Plot **time** on the x-axis and **range** on the y-axis. Add axis labels (with units) and a title to your plot. We know that the x-position of the projectile increases linearly with time. Therefore, **use the curve fitting** tool to fit a 1st order polynomial (line) to the distance data. Display the equation for the fitted polynomial on your graph with 5 significant digits. Then copy and paste your plot in the space below.

**MATLAB PLOT (with curve fit equation):**

****

**The “Best Fit” line and the “linear” lines, are the same lines. However the line named “Best Fit” was created through code in the script. The other, “linear”, was created via tools tab in the graph window, this allowed for the equation to be shown.**

1. Plot **time** on the x-axis and **height** on the y-axis. Add axis labels (with units) and a title to your plot. We know the height of the projectile follows a parabolic (2nd order) curve. So **use the curve fitting tool** to fit a 2nd order polynomial (quadratic) to the height data. Display the equation for the fitted polynomial on your graph with 5 significant digits. Then copy and paste your plot in the space below.

**MATLAB PLOT (with curve fit equation):**

****

**The “Best Fit” line and the “quadratic” lines, are the same lines. However the line named “Best Fit” was created through code in the script. The other, “quadratic”, was created via tools tab in the graph window, this allowed for the equation to be shown.**

1. Look at the numerical coefficient for the squared term in the fitted polynomial for the **height** data. Theoretically, this coefficient should be equal to 1/2\*g. How close is it? Calculate a percent error using the following formula with 1/2\*g as actual value:

Error = *abs*((Estimated – Actual)/Actual)\*100

**Show Calculations:**

coeff = -4.8842;

g = 9.81;

%theoretical = -0.5\*(g);

theoretical = -4.9050;

Error = abs((4.8842 - 4.9050)./ 4.9050)\*100

% Error = \_\_\_\_\_\_\_0.4241%\_\_\_\_\_\_\_\_\_\_\_

1. The numerical coefficient for the linear term in the fitted polynomial for **height** should be approximately V0 sin(θ) and the numerical coefficient for the linear term in the fitted polynomial for **distance** should be approximately V0 cos(θ). Enter the values below.

V0 sin(θ) \_\_\_\_\_\_\_\_49.162\_\_\_\_\_\_\_\_\_\_\_\_

V0 cos(θ) \_\_\_\_\_\_\_\_36.31\_\_\_\_\_\_\_\_\_\_\_\_

(e) Now you have two equations and two unknowns. Solve for launch angle and initial velocity. *Hint: Solve one equation for V0 and plug into 2nd equation to eliminate V0. Then use trig. to solve for theta.*

θ \_\_\_\_\_\_\_\_\_\_\_53.55°\_\_\_\_\_\_\_\_\_\_\_ (include units)

V0 \_\_\_\_\_\_\_61.12m/s\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (include units)

**Show Calculations:**

**49.162/36.31= tan^-1(1.354) = 53.55°**

**Sin(53.55) = 0.8044**

**= 49.162/0.8044**

**=61.12**

**Turn In:**

**The word (or pdf) document with your plots, MATLAB commands, and answers to questions.**