

Simulation Lab 3 Report

ECE 204 - Group 5 - 205

November 12, 2023

Punit Shah (20951260 - p73shah)

Dylan Nogueira (20951078 - dvnoguei)

Table of Contents:

General Code:	3
Polynomial Regression	3
Exponential Regression	4
Saturation Regression	4
 Code Snippets / Results	 5
Builder Code	5
Polynomial Degree 1 Code	6
Polynomial Degree 2 Code	7
Polynomial Degree 3 Code	8
Polynomial Degree 1 Results	9
Polynomial Degree 2 Results	10
Polynomial Degree 3 Results	11
Exponential Code	12
Exponential Results	13
Saturation Code	14
Saturation Results	15
 Table / Analysis:	
Table of Results	16
Best Case Analysis	17
Polynomial Analysis	17
Exponential Analysis	17
Saturation Analysis	17
 Appendix:	
All Code Snippets	18

General Code

The following code performs regression analysis on the 2 provided datasets and with user input, it will determine what method to use. When prompted, the user will be prompted to select which method to choose:

- 1) Polynomial Regression
 - a) If selected, user will choose either a polynomial from degree 1, 2 or 3 to perform the regression on
- 2) Exponential Regression
- 3) Saturation

After Calculations are made, the solutions will be given along with the graphed results.

Polynomial Regression:

The following program defines the function builder, that takes three arguments: number, xval, and yval. The purpose of the function is to perform polynomial regression analysis of different degrees (1, 2, or 3) based on the user's input. The function fits a polynomial curve to the input data points (xval, yval) and plots the resulting curve along with the raw data points.

For each degree,

- It calculates the coefficients of the polynomial using the method of least squares.
- Computes the sum of x values (sumx), sum of y values (sumy), sum of xy values (xy), sum of x^2 values (xsq), and additional terms for higher degrees.
- Constructs coefficient and constant matrices (matA and matB).
- Solves the system of linear equations using matrix division (matA\matB).
- Extracts the coefficients (a0, a1, a2, a3) from the solution matrix.
- Computes the total sum of squares (St), sum of squares of residuals (Sr), and the coefficient of determination (r2).
- Defines a symbolic polynomial function f(x) based on the obtained coefficients.
- Plots the fitted polynomial curve using fplot.

Exponential Regression:

The following program executes exponential regression on the test files.

This is done by:

- Taking the input and calculating the sum of the x values (sumx) and y values (sumy).
- Calculates the sum of natural logarithm of y values (logyval), the sum of x times natural logarithm of y values (xlogyval), and the sum of x squared values (squareval).
- Calculates the number of elements in the x column (n).
- Calculates the exponential regression coefficients a0 and a1 using the least squares method for the exponential model using the provided formula
- Plots the result of the exponential function.

Saturation Regression:

The following program executes saturation regression on the test files.

This is done by:

- Computes the sums of reciprocal x values (suminversex) and reciprocal y values (suminversey).
- Calculates the sum of squared reciprocal x values (squareval) and the sum of the product of reciprocal x and reciprocal y values (xsys)
- Calculates the number of elements in the x column (n).
- Calculates the saturation regression coefficients a0 and a1 using the formula provided
- Find A and B
- Computes the total sum of squares (St) and the sum of squares of residuals (Sr) using a temporary function (tempfunc).
- Calculates the coefficient of determination (r2) using the function "rsaturation" that compares the original yval and the values obtained from tempfunc.
- Plots the result of the saturation regression.

Matlab Code - Building Code

```
Editor - NAVECE 204 Sim Assignment 3 regression.m
1  regression.m  builder.m  test1.txt  test2.txt  test3.txt  test4.txt  +
2  % prompts user input for desired function representation
3  regressionmodel = input("Select the function to fit your data: \n 1.Polynomial: y = a0 + a1x + .. +amx^m \n 2.Exponential: y = ae^bx \n 3.Saturation: y = ax/b+x \n");
4
5  % loads text file, comment and uncomment as necessary
6  A = load('test1.txt');
7  % A = load('test2.txt');
8
9  % x values are clubbed together
10 xval = A(:,1);
11 % y values are clubbed together
12 yval = A(:,2);
13
14 % If 1 is entered, display Polynomial function
15 if regressionmodel == 1
16     builder(1, xval, yval);
17 % If 2 is entered, display Exponential function
18 elseif regressionmodel == 2
19     builder(2, xval, yval);
20
21 % If 3 is entered, display Saturation function
22 elseif regressionmodel == 3
23     builder(3, xval, yval);
24
25 else
26     disp("Please enter a number between 1 and 3");
27 end
```

Matlab Code - Polynomial Regression

Degree 1:

```

Editor - N:\ECE 204\Sim_Assignment_3\builder.m
regression.m x builder.m x test1.txt x test2.txt x test3.txt x test4.txt x +

1  function builder(number, xval, yval)
2
3  % POLYNOMIAL
4  if (number == 1)
5      degree = input("Please input the degree of polynomial you would like ");
6      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7      % DEGREE 1
8      if (degree == 1)
9          % summing all x values
10         sumx = double(sum(xval));
11         % summing all y values
12         sumy = double(sum(yval));
13         % summing all x * y values
14         xy = double(sum(xval.* yval));
15         % summing all x squared values
16         xsq = double(sum(xval.^2));
17
18         % finding number of elements in 1 column
19         n = size(xval, 1);
20
21         % finding a1 value
22         a1 = (n * xy - sumx * sumy) / (n * xsq - (sumx)^2);
23         % finding a0 value
24         a0 = sumy / n - a1 * (sumx / n);
25
26         % finding St and Sr values
27         St = sum((yval - sumy / n).^2);
28         Sr = sum((yval - a0 - a1.* xval).^2);
29
30         % calculating R^2
31         r2 = (St-Sr)/St;
32
33         % define symbolic function
34         syms f(x)
35         % allocate function to symbol using a1 and a0
36         f(x) = a0 + a1 * x;
37         % plot function
38         fplot(f)
39         hold on
40         title("Polynomial Degree 1: y = " + a0 + "+" + a1 + "x" + ' ' + "R^2 = " + r2);
41         xlabel("X values")
42         ylabel("Y values")
43         % scatter plot of raw data
44         scatter(xval, yval);
45         legend("Estimated Function", "Raw Data");
46         hold off
47
48         % desired values
49         disp("R^2 = " + r2);
50         disp("y = " + a0 + "+" + a1 + "x");

```

Degree 2:

```

52 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53 % DEGREE 2
54 - elseif(degree == 2)
55
56     % sum x values
57 -     sumx = double(sum(xval));
58     % sum x squared values
59 -     xsq = double(sum(xval.^2));
60     % sum x cubed values
61 -     xcu = double(sum(xval.^3));
62     % sum x^4 values
63 -     xfo = double(sum(xval.^4));
64
65     % sum y values
66 -     sumy = double(sum(yval));
67     % sum x*y values
68 -     xy = double(sum(xval.*yval));
69     % sum y*x^2 values
70 -     x2y = double(sum(yval.*(xval.^2)));
71
72     % finding number of elements in 1 column
73 -     n = size(xval,1);
74
75     % make coefficient matrix
76 -     matA = [n sumx xsq;
77             sumx xsq, xcu;
78             xsq xcu xfo];
79
80     % make constant matrix
81 -     matB = [sumy;
82             xy;
83             x2y];
84
85     % use matrix division and solve
86 -     sol = matA\matB;
87
88     % transpose and obtain column vector
89 -     solmat = (sol.');
```

```

90
91     % attribute each variable to elements in solution matrix
92 -     a0 = solmat(1);
93 -     a1 = solmat(2);
94 -     a2 = solmat(3);
95
96     % find St and Sr
97 -     St = sum((yval - sumy / n).^2);
98 -     Sr = sum((yval - a0 - a1.* xval - a2.*xval.^2).^2);
99
100     % find R^2 value
101 -     r2 = (St-Sr)/St;
102
103     % define symbolic function
104 -     syms f(x)
105     % allocate function to symbol
106 -     f(x) = a0 + a1*x + a2*x^2;
107     % plot function
108 -     fplot(f)
109 -     hold on
110 -     title("Polynomial Degree 1: y = " + a0 + "+" + a1 + "x" + a2 + 'x^2' + ' ' + "R^2 = " + r2);
111 -     xlabel("X values")
112 -     ylabel("Y values")
113     % scatter plot of raw data
114 -     scatter(xval, yval);
115 -     legend("Estimated Function", "Raw Data");
116
117     hold off
118
119     % desired values
120 -     disp("R^2 = " + r2);
121 -     disp("y = " + a0 + " + " + a1 + "x + " + a2 + 'x^2');
```

Degree 3:

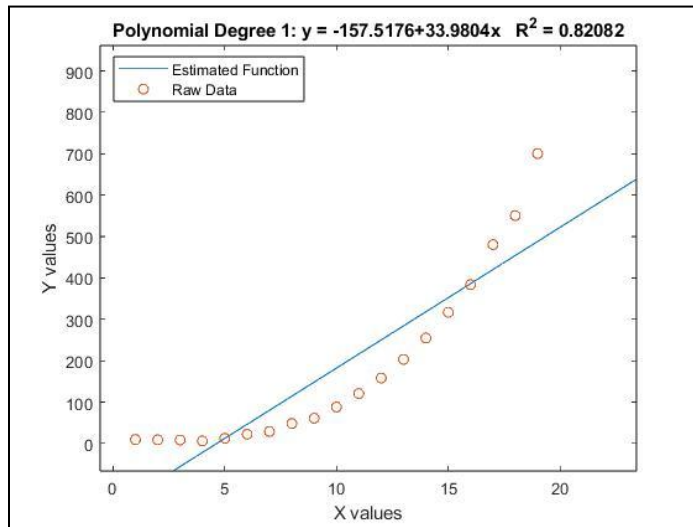
```

123 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
124 % DEGREE 3
125 elseif(degree == 3)
126     % sum x values
127     sumx = double(sum(xval));
128     % sum x squared values
129     xsq = double(sum(xval.^2));
130     % sum x cubed values
131     xcu = double(sum(xval.^3));
132     % sum x^4 values
133     xfo = double(sum(xval.^4));
134     % sum x^5 values
135     xqu = double(sum(xval.^5));
136     % sum x^6 values
137     xsi = double(sum(xval.^6));
138
139     % sum y values
140     sumy = double(sum(yval));
141     % sum x*y values
142     xy = double(sum(xval.*yval));
143     % sum y*x^2 values
144     x2y = double(sum(yval.*(xval.^2)));
145     % sum y*x^3 values
146     x3y = double(sum(yval.*(xval.^3)));
147
148     % finding number of elements in 1 column
149     n = size(xval,1);
150
151     % create coefficient matrix
152     matA = [n sumx xsq xcu;
153             sumx xsq xcu xfo;
154             xsq xcu xfo xqu;
155             xcu xfo xqu xsi];
156
157     % create solution matrix
158     matB = [sumy;
159             xy;
160             x2y;
161             x3y];
162
163     % solve using matrix division
164     sol = matA\matB;
165
166     % transpose and obtain column vector
167     solmat = (sol. ');
168
169     % attribute variables to elements in solution matrix
170     a0 = solmat(1);
171     a1 = solmat(2);
172     a2 = solmat(3);
173     a3 = solmat(4);
174
175     % find St and Sr values
176     St = sum((yval - sumy / n).^2);
177     Sr = sum((yval - a0 - a1.* xval - a2.*xval.^2 - a3.*xval.^3).^2);
178     r2 = (St-Sr)/St;
179
180     % define symbolic function
181     syms f(x)
182     % attribute function to symbol
183     f(x) = a0 + a1*x + a2*x^2 + a3*x^3;
184     % plot function
185     fplot(f)
186     hold on
187     title('Polynomial Degree 1: y = ' + a0 + '+' + a1 + "x" + a2 + 'x^2' + a3 + "x^3" + ' ' + "R^2 = " + r2);
188     xlabel("X values")
189     ylabel("Y values")
190     % scatter plot of raw data
191     scatter(xval, yval);
192     legend("Simulated Function", "Points");
193     hold off
194
195     % desired values
196     disp("R^2 = " + r2);
197     disp("y = " + a0 + " + " + a1 + "x" + " + a2 + 'x^2' + ' + a3 + 'x^3');
198
199 end
200 end

```


Results: Degree 1

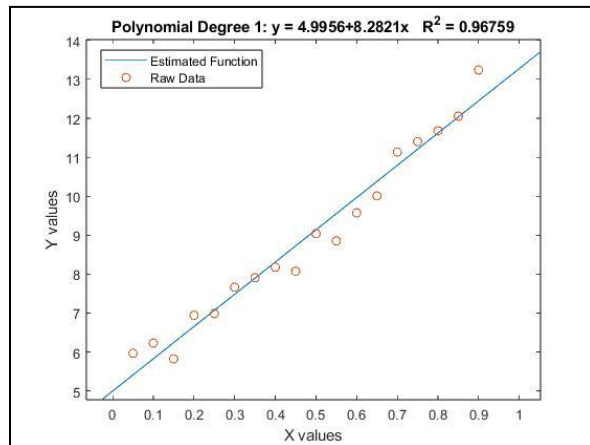
Test 1:



```

Command Window
>> regression
Select the function to fit your data:
1.Polynomial: y = a0 + alx + .. +amx^m
2.Exponential: y = ae^bx
3.Saturation: y = ax/b+x
1
Please input the degree of polynomial you would like 1
R^2 = 0.82082
y = -157.5176+33.9804x
  
```

Test 2:

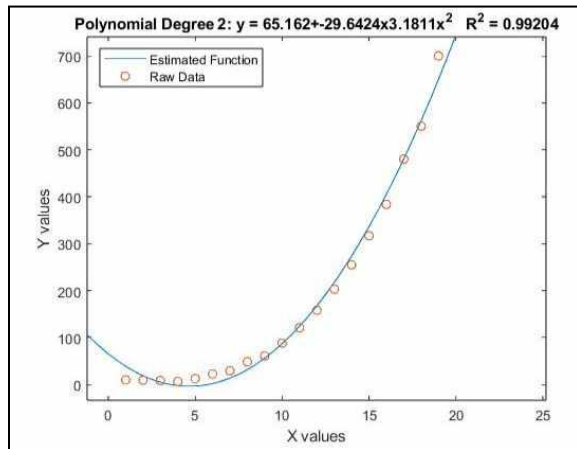


```

Command Window
>> regression
Select the function to fit your data:
1.Polynomial: y = a0 + alx + .. +amx^m
2.Exponential: y = ae^bx
3.Saturation: y = ax/b+x
1
Please input the degree of polynomial you would like 1
R^2 = 0.96759
y = 4.9956+8.2821x
  
```

Results: Degree 2

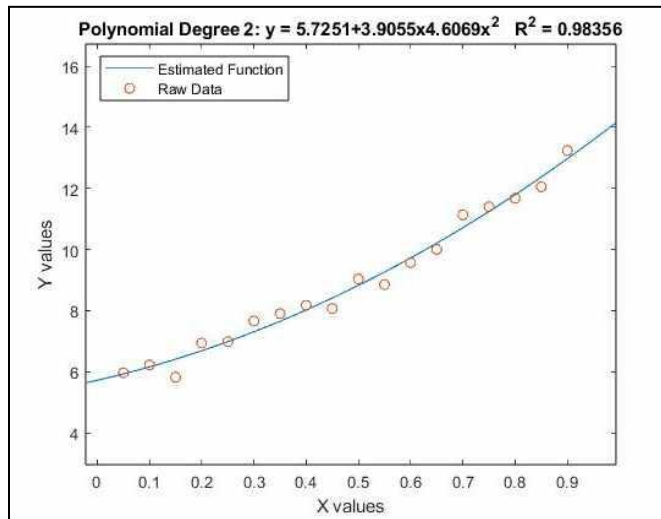
Test 1:



```

Command Window
>> regression
Select the function to fit your data:
1.Polynomial: y = a0 + alx + .. +amx^m
2.Exponential: y = ae^bx
3.Saturation: y = ax/b+x
1
Please input the degree of polynomial you would like 2
R^2 = 0.99204
y = 65.162 + -29.6424x + 3.1811x^2
  
```

Test 2:

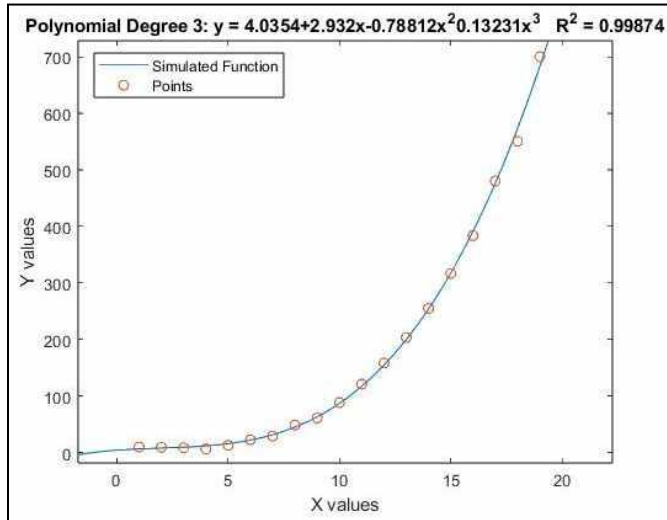


```

Command Window
>> regression
Select the function to fit your data:
1.Polynomial: y = a0 + alx + .. +amx^m
2.Exponential: y = ae^bx
3.Saturation: y = ax/b+x
1
Please input the degree of polynomial you would like 2
R^2 = 0.98356
y = 5.7251 + 3.9055x + 4.6069x^2
  
```

Results: Degree 3

Test 1:

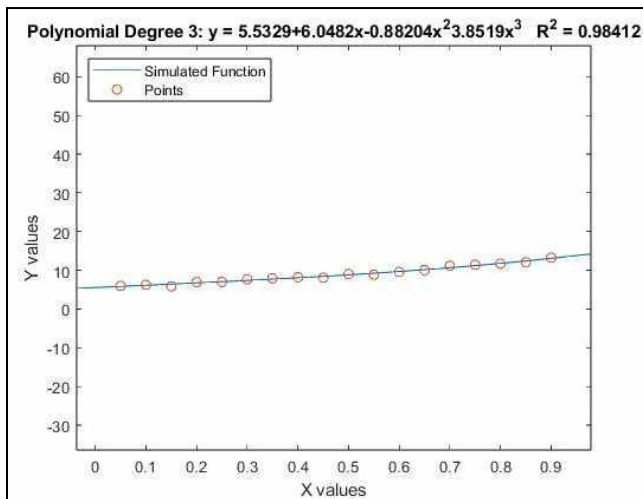


```

Command Window

>> regression
Select the function to fit your data:
1.Polynomial: y = a0 + alx + .. +amx^m
2.Exponential: y = ae^bx
3.Saturation: y = ax/b+x
1
Please input the degree of polynomial you would like 3
R^2 = 0.99874
y = 4.0354 + 2.932x + -0.78812x^2 + 0.13231x^3
  
```

Test 2:



```

Command Window

>> regression
Select the function to fit your data:
1.Polynomial: y = a0 + alx + .. +amx^m
2.Exponential: y = ae^bx
3.Saturation: y = ax/b+x
1
Please input the degree of polynomial you would like 3
R^2 = 0.98412
y = 5.5329 + 6.0482x + -0.88204x^2 + 3.8519x^3
  
```

Matlab Code - Exponential Regression

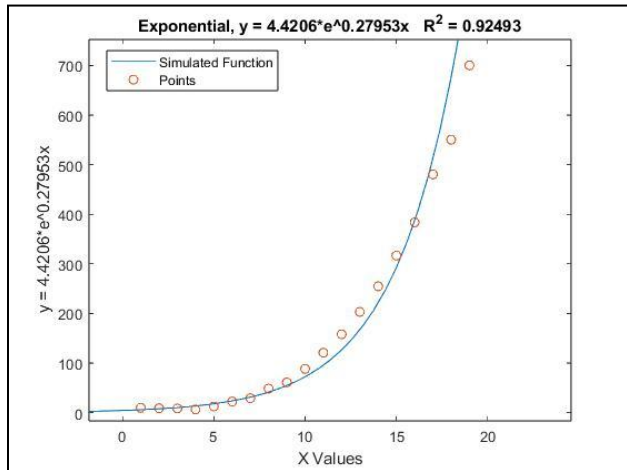
```

257 % EXPONENTIAL
258 - if(number == 2)
259     % sum x values
260     sumx = sum(xval);
261     % sum y values
262     sumy = sum(yval);
263
264     % sum logy values
265     logyval = sum(log(yval));
266     % sum x*logy values
267     xlogyval = sum(xval.* log(yval));
268     % sum x^2 values
269     squareval = sum(xval.^2);
270
271     % find number of elements in 1 column
272     n = size(xval,1);
273
274     % find a1
275     a1 = (n * xlogyval - logyval * sumx) / (n * squareval - sumx^2);
276     % find a0
277     a0 = logyval / n - a1 * (sumx / n);
278
279     % find St and Sr
280     St = sum((yval - sumy / n).^2);
281     Sr = sum((yval - (exp(a0)) * exp((a1.* xval))).^2);
282     % find R^2 value
283     r2 = (St - Sr) / St;
284
285     % define symbolic function
286     syms f(x)
287     % attribute function to symbol
288     f(x) = exp(a0) *exp(a1*x);
289     % plot function
290     fplot(f)
291     hold on
292     title("Exponential, y = " + exp(a0) + "*" + "e" + "\^" + a1 + "x    R^2 = " + r2);
293     % scatter plot of raw data
294     scatter(xval,yval);
295     hold off
296     legend('Simulated Function', 'Points');
297     xlabel('X Values');
298     ylabel("y = " + exp(a0) + "*" + "e" + "\^" + a1 + "x");
299     disp("R^2 = " + r2);
300     disp("Y = " + exp(a0) + " * " + "e" + "\^" + a1 + "x");
301
302 - end

```

Results:

Test 1:

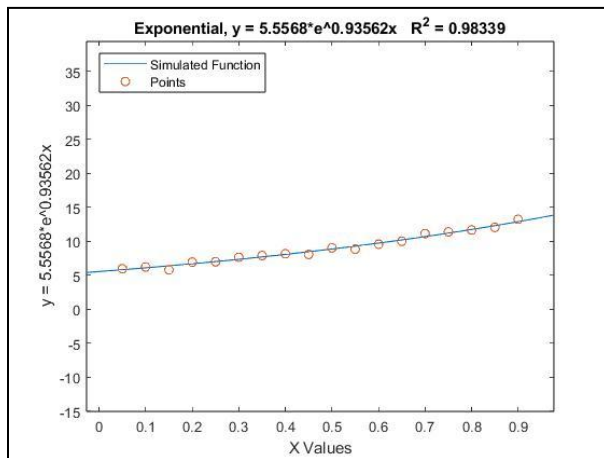


```

Command Window
>> regression
Select the function to fit your data:
1. Polynomial: y = a0 + alx + .. +amx^m
2. Exponential: y = ae^bx
3. Saturation: y = ax/b+x
2
R^2 = 0.92493
Y = 4.4206 * e^0.27953x

```

Test 2:



```

Command Window
>> regression
Select the function to fit your data:
1. Polynomial: y = a0 + alx + .. +amx^m
2. Exponential: y = ae^bx
3. Saturation: y = ax/b+x
2
R^2 = 0.98339
Y = 5.5568 * e^0.93562x

```

Matlab Code - Saturation Regression

```

202 % SATURATION
203 - if(number == 3)
204
205     % sum 1/x values
206     suminversex = double(sum(1./xval));
207     % sum 1/y values
208     suminversey = double(sum(1./yval));
209     % sum x^2 values
210     squareval = double(sum((1./xval).^2));
211     % sum 1/x * 1/y values
212     xsys = double(sum((1./xval).*(1./yval)));
213
214     % find number of elemtns in 1 column
215     n = size(xval,1);
216
217     % find a1
218     a1 = ((n * xsys) - (suminversex*suminversey))/(n * squareval - (suminversex^2));
219     % find a0
220     a0 = (suminversey / n) - a1*(suminversex / n);
221
222     % rearrange coefficients
223     A = 1/a0;
224     B = A*a1;
225
226     % find St and Sr
227     % St = sum((yval - sumy/n).^2);
228     % Sr = sum((yval - A.*xval/(B+sumx/n)).^2);
229
230     % temporary function
231     tempfunc = A.*xval ./ (B+xval);
232     % r2 value obtained from function
233     r2 = rsaturation(yval, tempfunc);
234
235     % define symbolic function
236     syms f(x)
237     % attribute function to symbol
238     f(x) = (A*x)/(B+x);
239     % plot function
240     fplot(f)
241     hold on
242     title("Saturation, y = (" + A + " * " + "x")/" + B + " + x" + ' ' + "R^2 = " + r2);
243     % scatter plot of raw data
244     scatter(xval,yval);
245     legend('Simulated Function', 'Points');
246     xlabel("X Values")
247     ylabel("Y Values")
248     hold off
249
250     % desired values
251     disp("R^2 = " + r2);
252     disp("y = (" + A + " * " + "x")/" + B + " + x");
253 - end

```

Rsaturation:

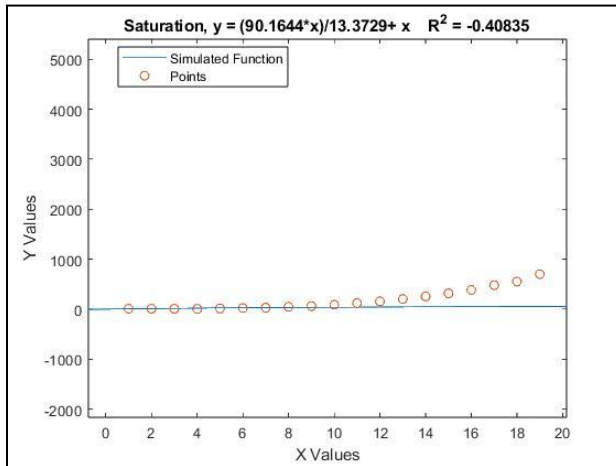
```

1  function r2 = rsaturation(yacc, ypred)
2  -     meany = mean(yacc);
3  -     ST = sum((yacc - meany).^2);
4  -     SR = sum((yacc - ypred).^2);
5  -     r2 = (ST-SR)/ST;
6  - end

```

Results:

Test 1:

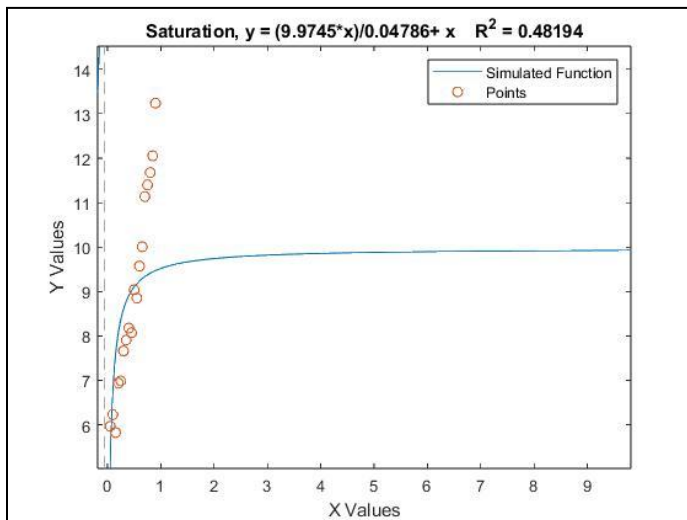


```

Command Window
>> regression
Select the function to fit your data:
1.Polynomial: y = a0 + alx + .. +amx^m
2.Exponential: y = ae^bx
3.Saturation: y = ax/b+x
3
R^2 = -0.40835
y = (90.1644 * x) / (13.3729 + x)

```

Test 2:



```

Command Window
>> regression
Select the function to fit your data:
1.Polynomial: y = a0 + alx + .. +amx^m
2.Exponential: y = ae^bx
3.Saturation: y = ax/b+x
3
R^2 = 0.48194
y = (9.9745 * x) / (0.04786 + x)

```

Results Summary

Tests	Test 1	R ² T1	Test 2	R ² T2
Polynomial Deg 1	$-157.5176 + 33.9804x$	0.82082	$4.9956 + 8.2821x$	0.96759
Polynomial Deg 2	$65.162 - 29.6424x + 3.1811x^2$	0.99204	$5.7251 + 3.9055x + 4.6069x^2$	0.98356
Polynomial Deg 3	$4.0354 + 2.932x - 0.78812x^2 + 0.13231x^3$	0.99874	$5.5329 + 6.0482x - 0.88204x^2 + 3.8519x^3$	0.98412
Exponential	$4.4206 * e^{0.27953x}$	0.92493	$5.5568 * e^{0.93562x}$	0.98339
Saturation	$(90.1644 * x)/(13.3729 + x)$	-0.40835	$(9.9745 * x)/(0.04786 + x)$	0.48194

Analysis:

In both test cases, the **Polynomial Degree 3** is the best regression analysis method. This is because the higher-degree polynomial can take into consideration more complex relationships between the data points. It also is more flexible in comparison to a linear or quadratic model which is beneficial to the dataset provided.

Polynomial Analysis:

Since the polynomial degree is **one**, we have a linear function. With this, linear regression will be used. The formula for both datasets is derived using two equations, where a_1 represents the slope, and a_0 denotes the y-intercept. Since both datasets do not have a y-intercept of 0, the special case formula is not required. To determine a_1 , we calculate the number of data points, denoted by n , and find the sums of x values, y values, product of x and y , and the sum of each x value squared.

Upon obtaining a_1 , we proceed to find a_0 , where \bar{y} signifies the mean value of the y values and \bar{x} denotes the mean value of the x values. Subsequently, we aim to compute the coefficient of determination, R^2 . If R^2 equals 1, it indicates a perfect fit. The objective is to calculate R^2 to assess its proximity to 1, signifying a perfect fit. To compute R^2 , we calculate S_t (the sum of squared differences

between each y value and the mean of all y values) and S_r (the sum of squared residuals, i.e., the difference between y_{actual} and the estimated formula).

The Calculated R^2 of data set 1 came to be 0.82082 which is close but not quite 1 indicating it is a near perfect fit. The Calculated R^2 of data set 2 came to be 0.96759 which is much closer to 1 indicating a better fit.

Since the next polynomial is of degree **two**, we have a quadratic function. With this, matrices will be used with the constants a_1 , a_2 and a_3 . The process involves setting up a matrix to solve for these constants and computing the coefficient of determination R^2 . Two matrices will be created one containing the coefficients and the other the output.

To determine a_0 , a_1 , and a_2 , the inverse matrix is taken and using the same methods as before, S_t and S_r are found and result in a quadratic.

The Calculated R^2 of data set 1 came to be 0.99204 which is almost 1 indicating it is a very close to perfect. The Calculated R^2 of data set 2 came to be 0.9836 which is slightly less closer to 1 resulting in this technique being even more optimal.

With the a new polynomial of degree **three**, we need to set up a 4x4 matrix and determine the coefficients once again. After applying previous techniques mentioned, the Calculated R^2 of data set 1 came to be 0.99874 which is even closer to 1 indicating it is very close to perfect. The Calculated R^2 of data set 2 came to be 0.98412 which is slightly less closer to 1 resulting in this technique being even more beneficial than the previous degrees.

Exponential Analysis:

For exponential model, the formula $y = A_1 \cdot e^{(B_1 \cdot x)}$ is used resulting in an exponential curve. Since it is non linear, it will be more difficult to determine the constants. This will mean that we need to convert the equation into a linear model and apply linear regression techniques. The natural logaritm of the new set is taken and linear regression is applied to determine a_0 and a_1 . With this, we have the results of test 1 with 0.92493 and test 2 of 0.98339 making it very good but not quite as good as degree 3.

Saturation Analysis:

For the Saturation model, we obtain a nonlinear function that needs to be linearized for analysis. For this, we find the sum of the inverse x values (suminverse_x), the sum of the inverse y values (suminverse_y), the sum of the x squared values (squareval) and the sum of the x inverse times y inverse values (xsys). We then find the coefficients by utilizing the functions taught to us in class, that of a_1 and a_0 . We then obtain our A and B values used for the final function we want to plot. We do this by rearranging the a_1 and a_0 coefficients and finally obtain a function of $(A \cdot x)/(B + x)$. When we plot the function we find that for test1 we get an R^2 value of -0.40835 and for test2, we get an R^2 value of 0.48194. For both test cases, we see that both perform very poorly in the regression analysis, and as a result are not the best case for modeling either set of data.

Appendix:

regression.m

```
Editor - N:\ECE 204\Sim_Assignment_3\regression.m
regression.m x builder.m x test1.txt x test2.txt x test3.txt x test4.txt x +
1 % prompts user input for desired function representation
2 regressionmodel = input("Select the function to fit your data: \n 1.Polynomial: y = a0 + a1x + .. +amx^m \n 2.Exponential: y = ae^bx \n 3.Saturation: y = ax/b+x \n");
3
4 % loads text file, comment and uncomment as necessary
5 A = load('test1.txt');
6 % A = load('test2.txt');
7
8 % x values are clubbed together
9 xval = A(:,1);
10 % y values are clubbed together
11 yval = A(:,2);
12
13 % If 1 is entered, display Polynomial function
14 if regressionmodel == 1
15     builder(1, xval, yval);
16 % If 2 is entered, display Exponential function
17 elseif regressionmodel == 2
18     builder(2, xval, yval);
19
20 % If 3 is entered, display Saturation function
21 elseif regressionmodel == 3
22     builder(3, xval, yval);
23
24 else
25     disp("Please enter a number between 1 and 3");
26
27 end
```

builder.m

```

Editor - N:\ECE 204\Sim_Assignment_3\builder.m
regression.m  builder.m  test1.txt  test2.txt  test3.txt  test4.txt  +

1  function builder(number, xval, yval)
2
3  % POLYNOMIAL
4  if (number == 1)
5      degree = input("Please input the degree of polynomial you would like ");
6      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7      % DEGREE 1
8      if (degree == 1)
9          % summing all x values
10         sumx = double(sum(xval));
11         % summing all y values
12         sumy = double(sum(yval));
13         % summing all x * y values
14         xy = double(sum(xval.* yval));
15         % summing all x squared values
16         xsq = double(sum(xval.^2));
17
18         % finding number of elements in 1 column
19         n = size(xval, 1);
20
21         % finding a1 value
22         a1 = (n * xy - sumx * sumy) / (n * xsq - (sumx)^2);
23         % finding a0 value
24         a0 = sumy / n - a1 * (sumx / n);
25
26         % finding St and Sr values
27         St = sum((yval - sumy / n).^2);
28         Sr = sum((yval - a0 - a1.* xval).^2);
29
30         % calculating R^2
31         r2 = (St-Sr)/St;
32
33         % define symbolic function
34         syms f(x)
35         % allocate function to symbol using a1 and a0
36         f(x) = a0 + a1 * x;
37         % plot function
38         fplot(f)
39         hold on
40         title("Polynomial Degree 1: y = " + a0 + "+" + a1 + "x" + ' ' + "R^2 = " + r2);
41         xlabel("X values")
42         ylabel("Y values")
43         % scatter plot of raw data
44         scatter(xval, yval);
45         legend("Estimated Function", "Raw Data");
46         hold off
47
48         % desired values
49         disp("R^2 = " + r2);
50         disp("y = " + a0 + "+" + a1 + "x");

```

```

52 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53 % DEGREE 2
54 - elseif(degree == 2)
55
56     % sum x values
57 -     sumx = double(sum(xval));
58     % sum x squared values
59 -     xsq = double(sum(xval.^2));
60     % sum x cubed values
61 -     xcu = double(sum(xval.^3));
62     % sum x^4 values
63 -     xfo = double(sum(xval.^4));
64
65     % sum y values
66 -     sumy = double(sum(yval));
67     % sum x*y values
68 -     xy = double(sum(xval.*yval));
69     % sum y*x^2 values
70 -     x2y = double(sum(yval.*(xval.^2)));
71
72     % finding number of elements in 1 column
73 -     n = size(xval,1);
74
75     % make coefficient matrix
76 -     matA = [n sumx xsq;
77             sumx xsq, xcu;
78             xsq xcu xfo];
79
80     % make constant matrix
81 -     matB = [sumy;
82             xy;
83             x2y];
84
85     % use matrix division and solve
86 -     sol = matA\matB;
87
88     % transpose and obtain column vector
89 -     solmat = (sol.');
```

```

90
91     % attribute each variable to elements in solution matrix
92 -     a0 = solmat(1);
93     a1 = solmat(2);
94 -     a2 = solmat(3);
95
96     % find St and Sr
97 -     St = sum((yval - sumy / n).^2);
98     Sr = sum((yval - a0 - a1.* xval - a2.*xval.^2).^2);
99
100     % find R^2 value
101 -     r2 = (St-Sr)/St;
102
103     % define symbolic function
104 -     syms f(x)
105     % allocate function to symbol
106 -     f(x) = a0 + a1*x + a2*x^2;
107     % plot function
108 -     fplot(f)
109     hold on
110 -     title("Polynomial Degree 1: y = " + a0 + "+" + a1 + "x" + a2 + 'x^2' + ' ' + "R^2 = " + r2);
111     xlabel("X values")
112 -     ylabel("Y values")
113     % scatter plot of raw data
114 -     scatter(xval, yval);
115     legend("Estimated Function", "Raw Data");
116
117 -     hold off
118
119     % desired values
120 -     disp("R^2 = " + r2);
121 -     disp("y = " + a0 + " + " + a1 + "x + " + a2 + 'x^2');
```

```

123 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
124 % DEGREE 3
125 elseif(degree == 3)
126     % sum x values
127     sumx = double(sum(xval));
128     % sum x squared values
129     xsq = double(sum(xval.^2));
130     % sum x cubed values
131     xcu = double(sum(xval.^3));
132     % sum x^4 values
133     xfo = double(sum(xval.^4));
134     % sum x^5 values
135     xqu = double(sum(xval.^5));
136     % sum x^6 values
137     xsi = double(sum(xval.^6));
138
139     % sum y values
140     sumy = double(sum(yval));
141     % sum x*y values
142     xy = double(sum(xval.*yval));
143     % sum y*x^2 values
144     x2y = double(sum(yval.*(xval.^2)));
145     % sum y*x^3 values
146     x3y = double(sum(yval.*(xval.^3)));
147
148     % finding number of elements in 1 column
149     n = size(xval,1);
150
151     % create coefficient matrix
152     matA = [n sumx xsq xcu;
153             sumx xsq xcu xfo;
154             xsq xcu xfo xqu;
155             xcu xfo xqu xsi];
156
157     % create solution matrix
158     matB = [sumy;
159             xy;
160             x2y;
161             x3y];

```

```

162
163 % solve using matrix division
164 sol = matA\matB;
165
166 % transpose and obtain column vector
167 solmat = (sol. ');
168
169 % attribute variables to elements in solution matrix
170 a0 = solmat(1);
171 a1 = solmat(2);
172 a2 = solmat(3);
173 a3 = solmat(4);
174
175 % find St and Sr values
176 St = sum((yval - sumy / n).^2);
177 Sr = sum((yval - a0 - a1.* xval - a2.*xval.^2 - a3.*xval.^3).^2);
178 r2 = (St-Sr)/St;
179
180 % define symbolic function
181 syms f(x)
182 % attribute function to symbol
183 f(x) = a0 + a1*x + a2*x^2 + a3*x^3;
184 % plot function
185 fplot(f)
186 hold on
187 title("Polynomial Degree 1: y = " + a0 + "+" + a1 + "x" + a2 + 'x^2' + a3 + "x^3" + ' ' + "R^2 = " + r2);
188 xlabel("X values")
189 ylabel("Y values")
190 % scatter plot of raw data
191 scatter(xval, yval);
192 legend("Simulated Function", "Points");
193 hold off
194
195 % desired values
196 disp("R^2 = " + r2);
197 disp("y = " + a0 + " + " + a1 + "x + " + a2 + 'x^2 + ' + a3 + 'x^3');
198
199 end
200 end

```

```

257 % EXPONENTIAL
258 - if(number == 2)
259     % sum x values
260 -     sumx = sum(xval);
261     % sum y values
262 -     sumy = sum(yval);
263
264     % sum logy values
265 -     logyval = sum(log(yval));
266     % sum x*logy values
267 -     xlogyval = sum(xval.* log(yval));
268     % sum x^2 values
269 -     squareval = sum(xval.^2);
270
271     % find number of elements in 1 column
272 -     n = size(xval,1);
273
274     % find a1
275 -     a1 = (n * xlogyval - logyval * sumx) / (n * squareval - sumx^2);
276     % find a0
277 -     a0 = logyval / n - a1 * (sumx / n);
278
279     % find St and Sr
280 -     St = sum((yval - sumy / n).^2);
281 -     Sr = sum((yval - (exp(a0)) * exp((a1.* xval))).^2);
282     % find R^2 value
283 -     r2 = (St - Sr) / St;
284
285     % define symbolic function
286 -     syms f(x)
287     % attribute function to symbol
288 -     f(x) = exp(a0) *exp(a1*x);
289     % plot function
290 -     fplot(f)
291     hold on
292 -     title("Exponential, y = " + exp(a0) + "*" + "e" + "\^" + a1 + "x    R^2 = " + r2);
293     % scatter plot of raw data
294 -     scatter(xval,yval);
295     hold off
296 -     legend('Simulated Function', 'Points');
297     xlabel('X Values');
298 -     ylabel("y = " + exp(a0) + "*" + "e" + "\^" + a1 + "x");
299     disp("R^2 = " + r2);
300 -     disp("Y = " + exp(a0) + " * " + "e" + "^" + a1 + "x");
301
302 - end

```

```

202 % SATURATION
203 - if(number == 3)
204
205     % sum 1/x values
206     suminversex = double(sum(1./xval));
207     % sum 1/y values
208     suminversey = double(sum(1./yval));
209     % sum x^2 values
210     squareval = double(sum((1./xval).^2));
211     % sum 1/x * 1/y values
212     xsys = double(sum((1./xval).*(1./yval)));
213
214     % find number of elemtns in 1 column
215     n = size(xval,1);
216
217     % find a1
218     a1 = ((n * xsys) - (suminversex*suminversey))/(n * squareval - (suminversex^2));
219     % find a0
220     a0 = (suminversey / n) - a1*(suminversex / n);
221
222     % rearrange coefficients
223     A = 1/a0;
224     B = A*a1;
225
226     % find St and Sr
227     St = sum((yval - sumy/n).^2);
228     Sr = sum(((yval - A.*xval/(B+sumx/n))).^2);
229
230     % temporary function
231     tempfunc = A.*xval ./ (B+xval);
232     % r2 value obtained from function
233     r2 = rsaturation(yval, tempfunc);
234
235     % define symbolic function
236     syms f(x)
237     % attribute function to symbol
238     f(x) = (A*x)/(B+x);
239     % plot function
240     fplot(f)
241     hold on
242     title("Saturation, y = (" + A + "*" + "x)/" + B + "+ x" + ' ' + "R^2 = " + r2);
243     % scatter plot of raw data
244     scatter(xval,yval);
245     legend('Simulated Function', 'Points');
246     xlabel("X Values")
247     ylabel("Y Values")
248     hold off
249
250     % desired values
251     disp("R^2 = " + r2);
252     disp("y = (" + A + "*" + "x)/(" + B + "+ x)");
253 - end

```

rsaturation.m

```
1  function r2 = rsaturation(yacc, ypred)
2  -     meany = mean(yacc);
3  -     ST = sum((yacc - meany).^2);
4  -     SR = sum((yacc - ypred).^2);
5  -     r2 = (ST-SR)/ST;
6  - end
```