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

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 wich method to choose:

1) Polynomial Regression
    a) If selected, user will chose 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 - N:\ECE 204\Sim_Assignment_3\regression.m

regression.m | builder.m | test1.txt | test2.txt | test3.txt | test4.txt

```matlab
% prompts user inpt for desired function representation
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");

% loads text file, comment and uncomment as necessary
A = load('test1.txt');
% A = load('test2.txt');

% x values are clubbed together
xval = A(:,1);
% y values are clubbed together
yval = A(:,2);

% If 1 is entered, display Polynomial function
if regressionmodel == 1
    builder(1, xval, yval);
% If 2 is entered, display Exponential function
elseif regressionmodel == 2
    builder(2, xval, yval);

% If 3 is entered, display Saturation function
elseif regressionmodel == 3
    builder(3, xval, yval);

else
    disp("Please enter a number between 1 and 3");

end
```

Matlab Code - Polynomial Regression

Degree 1:

Editor - N:\ECE 204\Sim_Assignment_3\builder.m

regression.m | builder.m | test1.txt | test2.txt | test3.txt | test4.txt | +

```
function builder(number, xval, yval)

% POLYNOMIAL
if (number == 1)
    degree = input("Please input the degree of polynomial you would like ");
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % DEGREE 1
    if (degree == 1)
        % summing all x values
        sumx = double(sum(xval));
        % summing all y values
        sumy = double(sum(yval));
        % summing all x * y values
        xy = double(sum(xval.* yval));
        % summing all x squared values
        xsq = double(sum(xval.^2));

        % finding number of elements in 1 column
        n = size(xval, 1);

        % finding a1 value
        a1 = (n * xy - sumx * sumy) / (n * xsq - (sumx)^2);
        % finding a0 value
        a0 = sumy / n - a1 * (sumx / n);

        % finding St and Sr values
        St = sum((yval - sumy / n).^2);
        Sr = sum((yval - a0 - a1.* xval).^2);

        % calculating R^2
        r2 = (St-Sr)/St;

        % define symbolic function
        syms f(x)
        % allocate function to symbol using a1 and a0
        f(x) = a0 + a1 * x;
        % plot function
        fplot(f)
        hold on
        title("Polynomial Degree 1: y = " + a0 + "+" + a1 + "x" + '    ' + "R^2 = " + r2);
        xlabel("X values")
        ylabel("Y values")
        % scatter plot of raw data
        scatter(xval, yval);
        legend("Estimated Function", "Raw Data");
        hold off

        % desired values
        disp("R^2 = " + r2);
        disp("y = " + a0 + "+" + a1 + "x");
```

Degree 2:

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % DEGREE 2
    elseif(degree == 2)

        % sum x values
        sumx = double(sum(xval));
        % sum x squared values
        xsq = double(sum(xval.^2));
        % sum x cubed values
        xcu = double(sum(xval.^3));
        % sum x^4 values
        xfo = double(sum(xval.^4));

        % sum y values
        sumy = double(sum(yval));
        % sum x*y values
        xy = double(sum(xval.*yval));
        % sum y*x^2 values
        x2y = double(sum(yval.*(xval.^2)));

        % finding number of elements in 1 column
        n = size(xval,1);

        % make coefficient matrix
        matA = [n sumx xsq;
                sumx xsq, xcu;
                xsq xcu xfo];

        % make constant matrix
        matB = [sumy;
                xy;
                x2y];

        % use matrix division and solve
        sol = matA\matB;

        % transpose and obtain column vector
        solmat = (sol.');
```

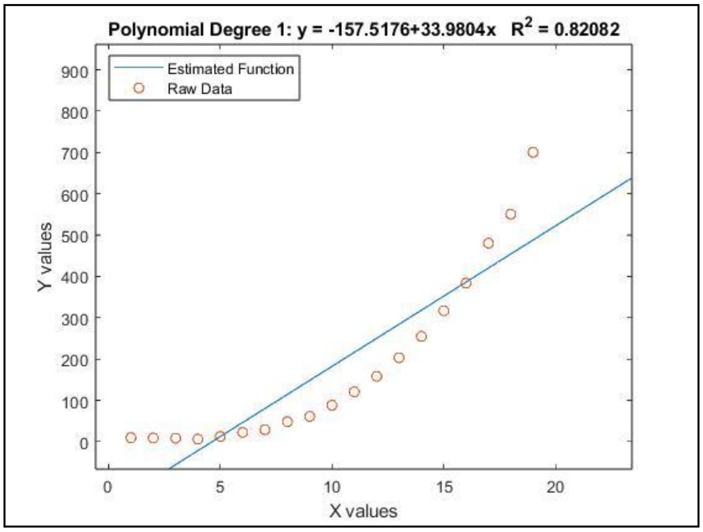
```

        % attribute each variable to elements in solution matrix
        a0 = solmat(1);
        a1 = solmat(2);
        a2 = solmat(3);

        % find St and Sr
        St = sum((yval - sumy / n).^2);
        Sr = sum((yval - a0 - a1.* xval - a2.*xval.^2).^2);

        % find R^2 value
        r2 = (St-Sr)/St;

        % define symbolic function
        syms f(x)
        % allocate function to symbol
        f(x) = a0 + a1*x + a2*x^2;
        % plot function
        fplot(f)
        hold on
        title("Polynomial Degree 1: y = " + a0 + "+" + a1 + "x" + a2 + 'x^2' + '    ' + "R^2 = " + r2);
        xlabel("X values")
        ylabel("Y values")
        % scatter plot of raw data
        scatter(xval, yval);
        legend("Estimated Function", "Raw Data");

        hold off

        % desired values
        disp("R^2 = " + r2);
        disp("y = " + a0 + " + " + a1 + "x + " + a2 + 'x^2');
```

Degree 3:

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % DEGREE 3
    elseif(degree == 3)
       % sum x values
        sumx = double(sum(xval));
        % sum x squared values
        xsq = double(sum(xval.^2));
        % sum x cubed values
        xcu = double(sum(xval.^3));
        % sum x^4 values
        xfo = double(sum(xval.^4));
        % sum x^5 values
        xqu = double(sum(xval.^5));
        % sum x^6 values
        xsi = double(sum(xval.^6));

        % sum y values
        sumy = double(sum(yval));
        % sum x*y values
        xy = double(sum(xval.*yval));
        % sum y*x^2 values
        x2y = double(sum(yval.*(xval.^2)));
        % sum y*x^3 values
        x3y = double(sum(yval.*(xval.^3)));

        % finding number of elements in 1 column
        n = size(xval,1);

        % create coefficient matrix
        matA = [n sumx xsq xcu;
                sumx xsq xcu xfo;
                xsq xcu xfo xqu;
                xcu xfo xqu xsi];

        % create solution matrix
        matB = [sumy;
                xy;
                x2y;
                x3y];

        % solve using matrix division
        sol = matA\matB;

        % transpose and obtain column vector
        solmat = (sol.');

        % attribute variables to elements in solution matrix
        a0 = solmat(1);
        a1 = solmat(2);
        a2 = solmat(3);
        a3 = solmat(4);

        % find St and Sr values
        St = sum((yval - sumy / n).^2);
        Sr = sum((yval - a0 - a1.* xval - a2.*xval.^2 - a3.*xval.^3).^2);
        r2 = (St-Sr)/St;

        % define symbolic function
        syms f(x)
        % attribute function to symbol
        f(x) = a0 + a1*x + a2*x^2 + a3*x^3;
        % plot function
        fplot(f)
        hold on
        title("Polynomial Degree 1: y = " + a0 + "+" + a1 + "x" + a2 + 'x^2' + a3 + "x^3" + '    ' + "R^2 = " + r2);
        xlabel("X values")
        ylabel("Y values")
        % scatter plot of raw data
        scatter(xval, yval);
        legend("Simulated Function", "Points");
        hold off

        % desired values
        disp("R^2 = " + r2);
        disp("y = " + a0 + " + " + a1 + "x + " + a2 + 'x^2 + ' + a3 + 'x^3');

    end
end
```

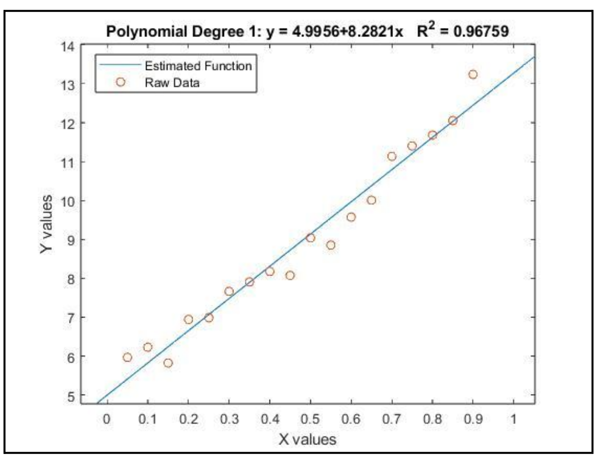
Results: Degree 1

Test 1:




```
Command Window
>> regression
Select the function to fit your data:
 1.Polynomial: y = a0 + a1x + .. +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
```

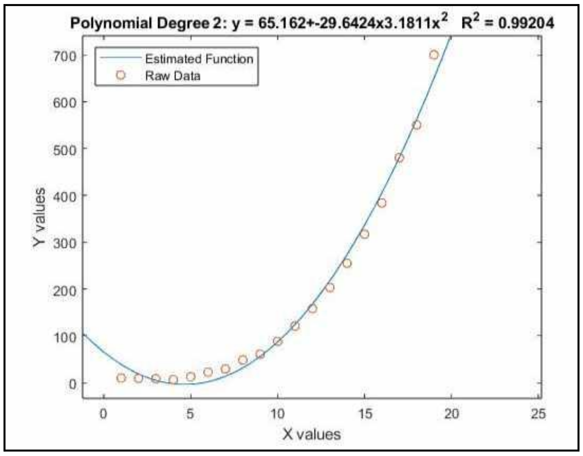
Test 2:




```
Command Window
>> regression
Select the function to fit your data:
 1.Polynomial: y = a0 + a1x + .. +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
```

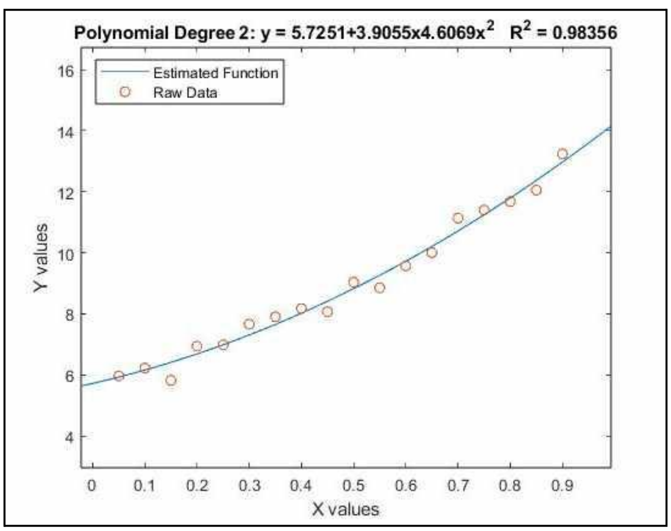
Results: Degree 2

Test 1:




```
Command Window
>> regression
Select the function to fit your data:
 1.Polynomial: y = a0 + a1x + .. +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
```

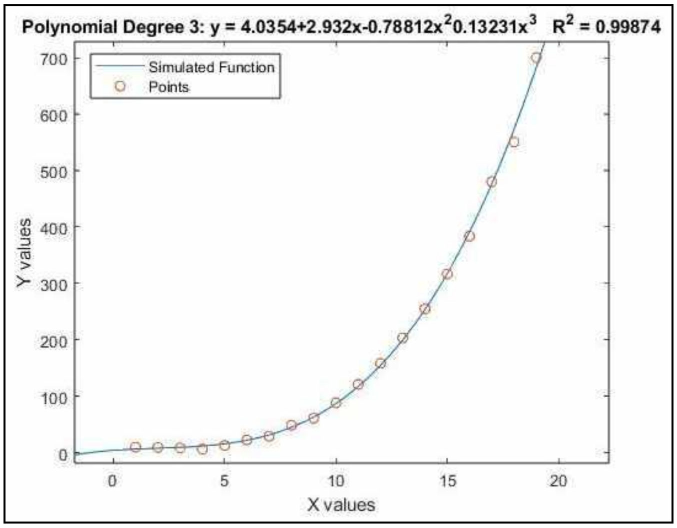
Test 2:




```
Command Window
>> regression
Select the function to fit your data:
 1.Polynomial: y = a0 + a1x + .. +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
```

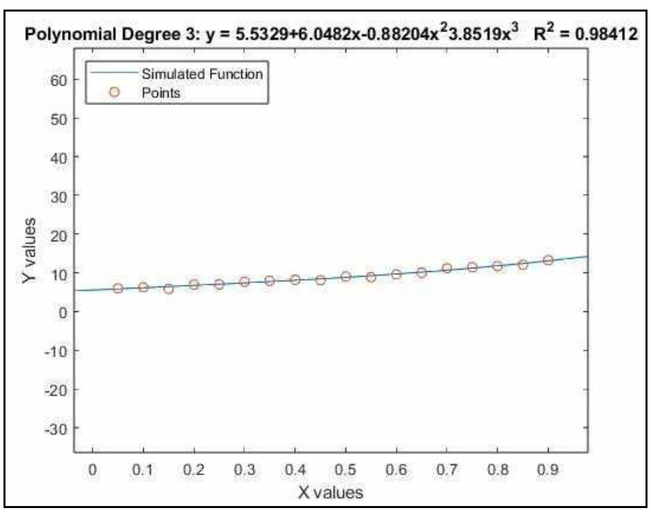
Results: Degree 3

Test 1:




```
Command Window
>> regression
Select the function to fit your data:
 1.Polynomial: y = a0 + a1x + .. +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:



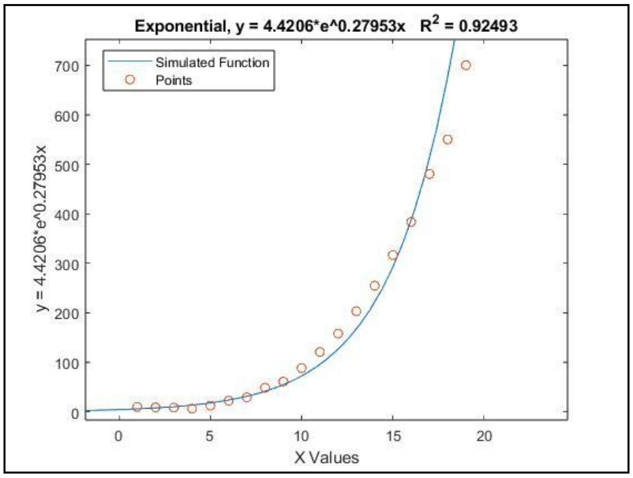

```
Command Window
>> regression
Select the function to fit your data:
 1.Polynomial: y = a0 + a1x + .. +amx^m
 2.Exponential: y = ae^bx
 3.Saturation: y = ax/b+x
1
Please input the degree of polynomial you would like 3
R^2 = 0.98412
y = 5.5329 + 6.0482x + -0.88204x^2 + 3.8519x^3
```

Matlab Code - Exponential Regression

```
% EXPONENTIAL
if(number == 2)
    % sum x values
    sumx = sum(xval);
    % sum y values
    sumy = sum(yval);

    % sum logy values
    logyval = sum(log(yval));
    % sum x*logy values
    xlogyval = sum(xval.* log(yval));
    % sum x^2 values
    squareval = sum(xval.^2);

    % find number of elements in 1 column
    n = size(xval,1);

    % find a1
    a1 = (n * xlogyval - logyval * sumx) / (n * squareval - sumx^2);
    % find a0
    a0 = logyval / n - a1 * (sumx / n);

    % find St and Sr
    St = sum((yval - sumy / n).^2);
    Sr = sum((yval - (exp(a0)) * exp((a1.* xval))).^2);
    % find R^2 value
    r2 = (St - Sr) / St;

    % define symbolic function
    syms f(x)
    % attribute function to symbol
    f(x) = exp(a0) *exp(a1*x);
    % plot function
    fplot(f)
    hold on
    title("Exponential, y = " + exp(a0) + "*" + "e" + "\^" + a1 + "x   R^2 = " + r2);
    % scatter plot of raw data
    scatter(xval,yval);
    hold off
    legend('Simulated Function', 'Points');
    xlabel('X Values');
    ylabel("y = " + exp(a0) + "*" + "e" + "\^" + a1 + "x");
    disp("R^2 = " + r2);
    disp("Y = " + exp(a0) + " * " + "e" + "^" + a1 + "x");

end
```

Results:

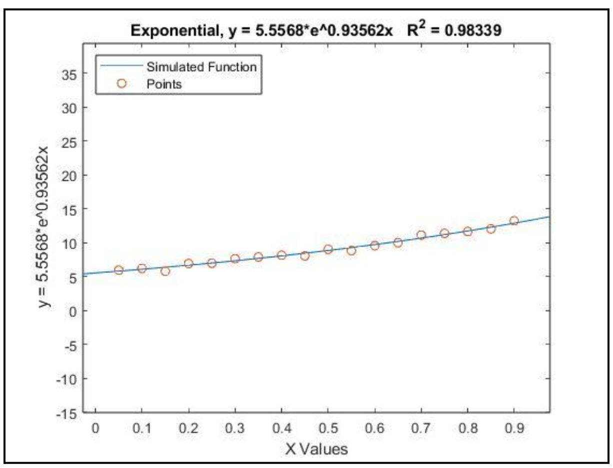
Test 1:



Command Window

```
>> regression
Select the function to fit your data:
 1.Polynomial: y = a0 + a1x + .. +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 + a1x + .. +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

```
% SATURATION
if(number == 3)

    % sum 1/x values
    suminversex = double(sum(1./xval));
    % sum 1/y values
    suminversey = double(sum(1./yval));
    % sum x^2 values
    squareval = double(sum((1./xval).^2));
    % sum 1/x * 1/y values
    xsys = double(sum((1./xval).*(1./yval)));

    % find number of elemtns in 1 column
    n = size(xval,1);

    % find a1
    a1 = ((n * xsys) - (suminversex*suminversey))/(n * squareval - (suminversex^2));
    % find a0
    a0 = (suminversey / n) - a1*(suminversex / n);

    % rearrage coefficients
    A = 1/a0;
    B = A*a1;

    % find St and Sr
    % St = sum((yval - sumy/n).^2);
    % Sr = sum(((yval - A.*xval/(B+sumx/n))).^2);

    % temporary function
    tempfunc = A.*xval ./ (B+xval);
    % r2 value obtained from function
    r2 = rsaturation(yval, tempfunc);
```

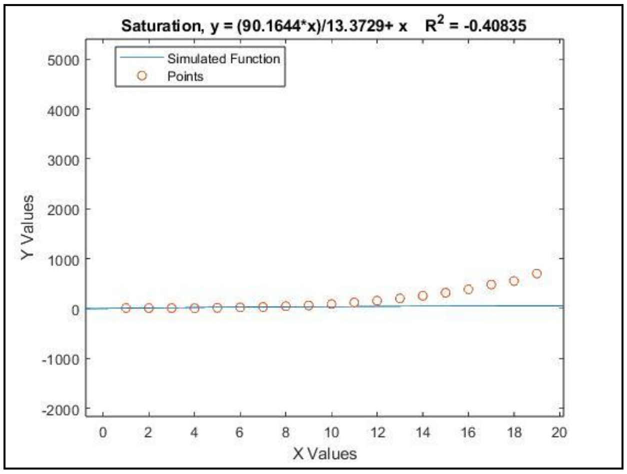
```

    % define symbolic function
    syms f(x)
    % attribute function to symbol
    f(x) = (A*x)/(B+x);
    % plot function
    fplot(f)
    hold on
    title("Saturation, y = (" + A + "*" + "x)/" + B + "+ x" + '      ' + "R^2 = " + r2);
    % scatter plot of raw data
    scatter(xval,yval);
    legend('Simulated Function', 'Points');
    xlabel("X Values")
    ylabel("Y Values")
    hold off

    % desired values
    disp("R^2 = " + r2);
    disp("y = (" + A + " * " + "x)/(" + B + " + x)");
end
```

Rsaturation:

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

Results:

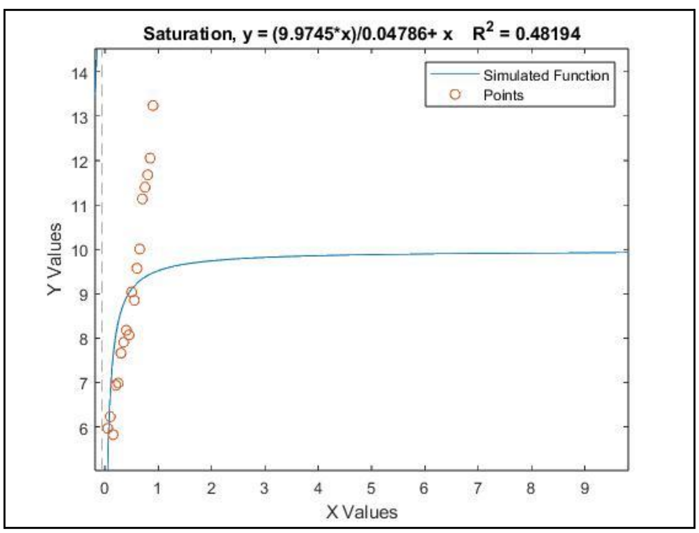
Test 1:



Command Window

```
>> regression
Select the function to fit your data:
 1.Polynomial: y = a0 + a1x + .. +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 + a1x + .. +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 wich 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 a1 represents the slope, and a0 denotes the y-intercept. Since both datasets do not have a y-intercept of 0, the special case formula is not required. To determine a1, 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 a1, we proceed to find a0, where y signifies the mean value of the y values and 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 St (the sum of squared differences between each y value and the mean of all y values) and Sr (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 a1, a2 and a3. 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 containg the coefficients and the other the output.

To determine a0, a1, and a2, the inverse matrix is taken and using the same methods as before, St and Sr 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 = A1*e^(B1*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 lograitm of the new set is taken and linear regression is applied to determine a0 and a1. 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 (suminversex), the sum of the inverse y values (suminversey), 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 a1 and a0. We then obtain our A and B values used for the final function we want to plot. We do this by rearranging the a1 and a0 coefficients and finally obtain a function of (A*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 | builder.m | test1.txt | test2.txt | test3.txt | test4.txt

```matlab
% prompts user inpt for desired function representation
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");

% loads text file, comment and uncomment as necessary
A = load('test1.txt');
% A = load('test2.txt');

% x values are clubbed together
xval = A(:,1);
% y values are clubbed together
yval = A(:,2);

% If 1 is entered, display Polynomial function
if regressionmodel == 1
    builder(1, xval, yval);
% If 2 is entered, display Exponential function
elseif regressionmodel == 2
    builder(2, xval, yval);

% If 3 is entered, display Saturation function
elseif regressionmodel == 3
    builder(3, xval, yval);

else
    disp("Please enter a number between 1 and 3");

end
```

builder.m

Editor - N:\ECE 204\Sim_Assignment_3\builder.m

regression.m | builder.m | test1.txt | test2.txt | test3.txt | test4.txt | +

```matlab
function builder(number, xval, yval)

% POLYNOMIAL
if (number == 1)
    degree = input("Please input the degree of polynomial you would like ");
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % DEGREE 1
    if (degree == 1)
        % summing all x values
        sumx = double(sum(xval));
        % summing all y values
        sumy = double(sum(yval));
        % summing all x * y values
        xy = double(sum(xval.* yval));
        % summing all x squared values
        xsq = double(sum(xval.^2));

        % finding number of elements in 1 column
        n = size(xval, 1);

        % finding a1 value
        a1 = (n * xy - sumx * sumy) / (n * xsq - (sumx)^2);
        % finding a0 value
        a0 = sumy / n - a1 * (sumx / n);

        % finding St and Sr values
        St = sum((yval - sumy / n).^2);
        Sr = sum((yval - a0 - a1.* xval).^2);

        % calculating R^2
        r2 = (St-Sr)/St;

        % define symbolic function
        syms f(x)
        % allocate function to symbol using a1 and a0
        f(x) = a0 + a1 * x;
        % plot function
        fplot(f)
        hold on
        title("Polynomial Degree 1: y = " + a0 + "+" + a1 + "x" + '    ' + "R^2 = " + r2);
        xlabel("X values")
        ylabel("Y values")
        % scatter plot of raw data
        scatter(xval, yval);
        legend("Estimated Function", "Raw Data");
        hold off

        % desired values
        disp("R^2 = " + r2);
        disp("y = " + a0 + "+" + a1 + "x");
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % DEGREE 2
    elseif(degree == 2)

        % sum x values
        sumx = double(sum(xval));
        % sum x squared values
        xsq = double(sum(xval.^2));
        % sum x cubed values
        xcu = double(sum(xval.^3));
        % sum x^4 values
        xfo = double(sum(xval.^4));

        % sum y values
        sumy = double(sum(yval));
        % sum x*y values
        xy = double(sum(xval.*yval));
        % sum y*x^2 values
        x2y = double(sum(yval.*(xval.^2)));

        % finding number of elements in 1 column
        n = size(xval,1);

        % make coefficient matrix
        matA = [n sumx xsq;
                sumx xsq, xcu;
                xsq xcu xfo];

        % make constant matrix
        matB = [sumy;
                xy;
                x2y];

        % use matrix division and solve
        sol = matA\matB;

        % transpose and obtain column vector
        solmat = (sol.');
```

```

        % attribute each variable to elements in solution matrix
        a0 = solmat(1);
        a1 = solmat(2);
        a2 = solmat(3);

        % find St and Sr
        St = sum((yval - sumy / n).^2);
        Sr = sum((yval - a0 - a1.* xval - a2.*xval.^2).^2);

        % find R^2 value
        r2 = (St-Sr)/St;

        % define symbolic function
        syms f(x)
        % allocate function to symbol
        f(x) = a0 + a1*x + a2*x^2;
        % plot function
        fplot(f)
        hold on
        title("Polynomial Degree 1: y = " + a0 + "+" + a1 + "x" + a2 + 'x^2' + '    ' + "R^2 = " + r2);
        xlabel("X values")
        ylabel("Y values")
        % scatter plot of raw data
        scatter(xval, yval);
        legend("Estimated Function", "Raw Data");

        hold off

        % desired values
        disp("R^2 = " + r2);
        disp("y = " + a0 + " + " + a1 + "x + " + a2 + 'x^2');
```

```
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % DEGREE 3
        elseif(degree == 3)
           % sum x values
            sumx = double(sum(xval));
            % sum x squared values
            xsq = double(sum(xval.^2));
            % sum x cubed values
            xcu = double(sum(xval.^3));
            % sum x^4 values
            xfo = double(sum(xval.^4));
            % sum x^5 values
            xqu = double(sum(xval.^5));
            % sum x^6 values
            xsi = double(sum(xval.^6));

            % sum y values
            sumy = double(sum(yval));
            % sum x*y values
            xy = double(sum(xval.*yval));
            % sum y*x^2 values
            x2y = double(sum(yval.*(xval.^2)));
            % sum y*x^3 values
            x3y = double(sum(yval.*(xval.^3)));

            % finding number of elements in 1 column
            n = size(xval,1);

            % create coefficient matrix
            matA = [n sumx xsq xcu;
                    sumx xsq xcu xfo;
                    xsq xcu xfo xqu;
                    xcu xfo xqu xsi];

            % create solution matrix
            matB = [sumy;
                    xy;
                    x2y;
                    x3y];

            % solve using matrix division
            sol = matA\matB;

            % transpose and obtain column vector
            solmat = (sol.');

            % attribute variables to elements in solution matrix
            a0 = solmat(1);
            a1 = solmat(2);
            a2 = solmat(3);
            a3 = solmat(4);

            % find St and Sr values
            St = sum((yval - sumy / n).^2);
            Sr = sum((yval - a0 - a1.* xval - a2.*xval.^2 - a3.*xval.^3).^2);
            r2 = (St-Sr)/St;

            % define symbolic function
            syms f(x)
            % attribute function to symbol
            f(x) = a0 + a1*x + a2*x^2 + a3*x^3;
            % plot function
            fplot(f)
            hold on
            title("Polynomial Degree 1: y = " + a0 + "+" + a1 + "x" + a2 + 'x^2' + a3 + "x^3" + '    ' + "R^2 = " + r2);
            xlabel("X values")
            ylabel("Y values")
            % scatter plot of raw data
            scatter(xval, yval);
            legend("Simulated Function", "Points");
            hold off

            % desired values
            disp("R^2 = " + r2);
            disp("y = " + a0 + " + " + a1 + "x + " + a2 + 'x^2 + ' + a3 + 'x^3');

        end
    end
```

```
% EXPONENTIAL
if(number == 2)
    % sum x values
    sumx = sum(xval);
    % sum y values
    sumy = sum(yval);

    % sum logy values
    logyval = sum(log(yval));
    % sum x*logy values
    xlogyval = sum(xval.* log(yval));
    % sum x^2 values
    squareval = sum(xval.^2);

    % find number of elements in 1 column
    n = size(xval,1);

    % find a1
    a1 = (n * xlogyval - logyval * sumx) / (n * squareval - sumx^2);
    % find a0
    a0 = logyval / n - a1 * (sumx / n);

    % find St and Sr
    St = sum((yval - sumy / n).^2);
    Sr = sum((yval - (exp(a0)) * exp((a1.* xval))).^2);
    % find R^2 value
    r2 = (St - Sr) / St;

    % define symbolic function
    syms f(x)
    % attribute function to symbol
    f(x) = exp(a0) *exp(a1*x);
    % plot function
    fplot(f)
    hold on
    title("Exponential, y = " + exp(a0) + "*" + "e" + "\^" + a1 + "x    R^2 = " + r2);
    % scatter plot of raw data
    scatter(xval,yval);
    hold off
    legend('Simulated Function', 'Points');
    xlabel('X Values');
    ylabel("y = " + exp(a0) + "*" + "e" + "\^" + a1 + "x");
    disp("R^2 = " + r2);
    disp("Y = " + exp(a0) + " * " + "e" + "^" + a1 + "x");

end
```

```
% SATURATION
if(number == 3)

    % sum 1/x values
    suminversex = double(sum(1./xval));
    % sum 1/y values
    suminversey = double(sum(1./yval));
    % sum x^2 values
    squareval = double(sum((1./xval).^2));
    % sum 1/x * 1/y values
    xsys = double(sum((1./xval).*(1./yval)));

    % find number of elemtns in 1 column
    n = size(xval,1);

    % find a1
    a1 = ((n * xsys) - (suminversex*suminversey))/(n * squareval - (suminversex^2));
    % find a0
    a0 = (suminversey / n) - a1*(suminversex / n);

    % rearrage coefficients
    A = 1/a0;
    B = A*a1;

    % find St and Sr
    % St = sum((yval - sumy/n).^2);
    % Sr = sum(((yval - A.*xval/(B+sumx/n))).^2);

    % temporary function
    tempfunc = A.*xval ./ (B+xval);
    % r2 value obtained from function
    r2 = rsaturation(yval, tempfunc);

    % define symbolic function
    syms f(x)
    % attribute function to symbol
    f(x) = (A*x)/(B+x);
    % plot function
    fplot(f)
    hold on
    title("Saturation, y = (" + A + "*" + "x)/" + B + "+ x" + '     ' + "R^2 = " + r2);
    % scatter plot of raw data
    scatter(xval,yval);
    legend('Simulated Function', 'Points');
    xlabel("X Values")
    ylabel("Y Values")
    hold off

    % desired values
    disp("R^2 = " + r2);
    disp("y = (" + A + " * " + "x)/(" + B + " + x)");
end
```

rsaturation.m

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