MATH 2059 – Project #1 Report

1st question:

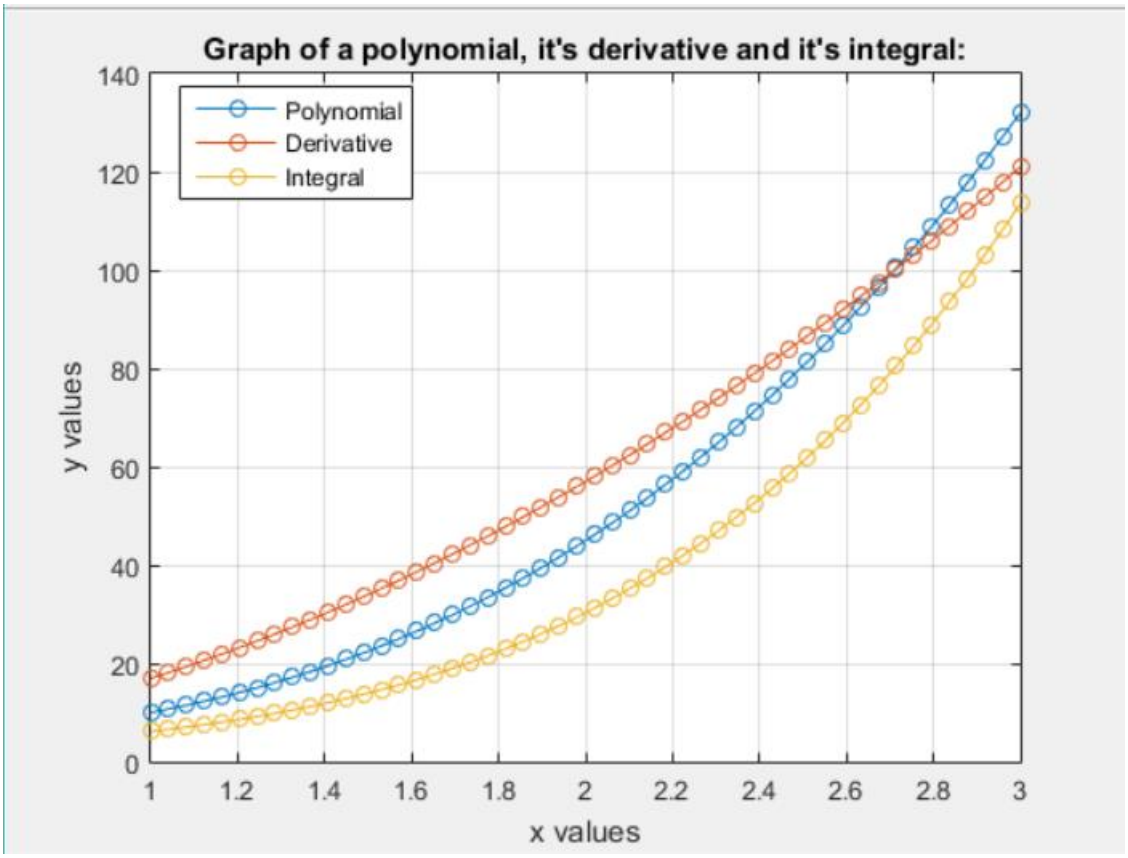
```
%Mert KELKİT - 150115013
%Rümeysa ELİÖZ - 150114016
%Question 1

function plotpint( coef, x1, x2, numpoints )

%creating a vector from x1 to x2 for x values.
xValues = linspace(x1, x2, numpoints);
%we need a flipped matrix in order to use polyval function because polyval
%function takes first coefficient from the last index.
flippedCoef = fliplr(coef);
%evaluating polynomial value.
polynomial = polyval(flippedCoef, xValues);

%taking derivative and creating a new coefficient matrix.
%this operation decreases 1 the exponent and multiplies by the old coefficient.
derivativeCoef = coef(2:end).*(linspace(1, size(coef,2)-1, size(coef,2)-1)); % Derivative coefficients
%we need a flipped vector in order to use polyval function
flippedDerivativeCoef = fliplr(derivativeCoef);
%taking derivatived values with polyval function
derivativePolynomial = polyval(flippedDerivativeCoef, xValues);

%taking integral and creating a new coefficient matrix.
integrationCoef = linspace(0, 0, size(coef,2)+1);
%setting c = 1.
integrationCoef(1) = 1;
%integration operation
%increasing the exponent by one and divide coefficient by it.
integrationCoef(2:end) = coef./(linspace(1, size(coef,2), size(coef,2)));
%flipping vector to use polyval function
flippedIntegrationCoef = fliplr(integrationCoef);
%taking integrated values of the polynomial
integratedPolynomial = polyval(flippedIntegrationCoef, xValues);

%plotting all of them in the same graph
plot(xValues, polynomial, '-o', xValues, derivativePolynomial, '-o', xValues, integratedPolynomial, '-o');
grid on
xlabel('x values');
ylabel('y values');
%setting title
title('Graph of a polynomial, it''s derivative and it''s integral:');
%setting legend
legend('Polynomial', 'Derivative', 'Integral', 'location', 'best')
end
```

In this function, inputs are coefficients as a row vector, x1 and x2 bound values for plotting, numpoints are number of points to be plotted. At first, we evaluate x values x1 to x2 with ‘numpoints’ points as a row vector. Then we evaluate flip this vector because polyval function assumes that last element of this vector is the coefficient of the first term of the polynomial. Then we evalute polynomial’s values with polyval function and it returns a vector.

For taking derivative, inputted coefficient’s first element will be zero, because degree of x is 0. So, coefficients start with our inputted coefficient’s second element. Then we multiply each coefficient with their x’s degrees and decrease their degrees by one. Size of the row vector will be old size – 1. After evaluating derivatived coefficients, we flip it in order to use polyval function. Then use polyval function in order to evaluate derivatived polynomial’s values as a row vector.

For integral, constant c = 1, first element of our coefficient will be 1. Then we start with the second element to the last element. We increase the degree of corresponding x by one, and divide the corresponding coefficient by x's new degree. After that we evaluate the integrated coefficients as a row vector. Then we flip this vector by fliplr function to use polyval function. After evaluating polynomial values, we start plotting. We will use same x values in order to plot. "-o" parameter means that mark the datapoints. We set legend, x axis labels, y axis labels and title. plot(xValues, polynomial, "-o", xValues, derivativePolynomial, "-o", …) means that plot polynomial with x values, mark the datapoints with a circle; plot derivatived polynomial with x values, mark the datapoints with a circle(same for integral).

If we input coef = [3, 1, 2, 4], x1 = 1, x2 = 3, numpoints = 50, sample plot will be like this:



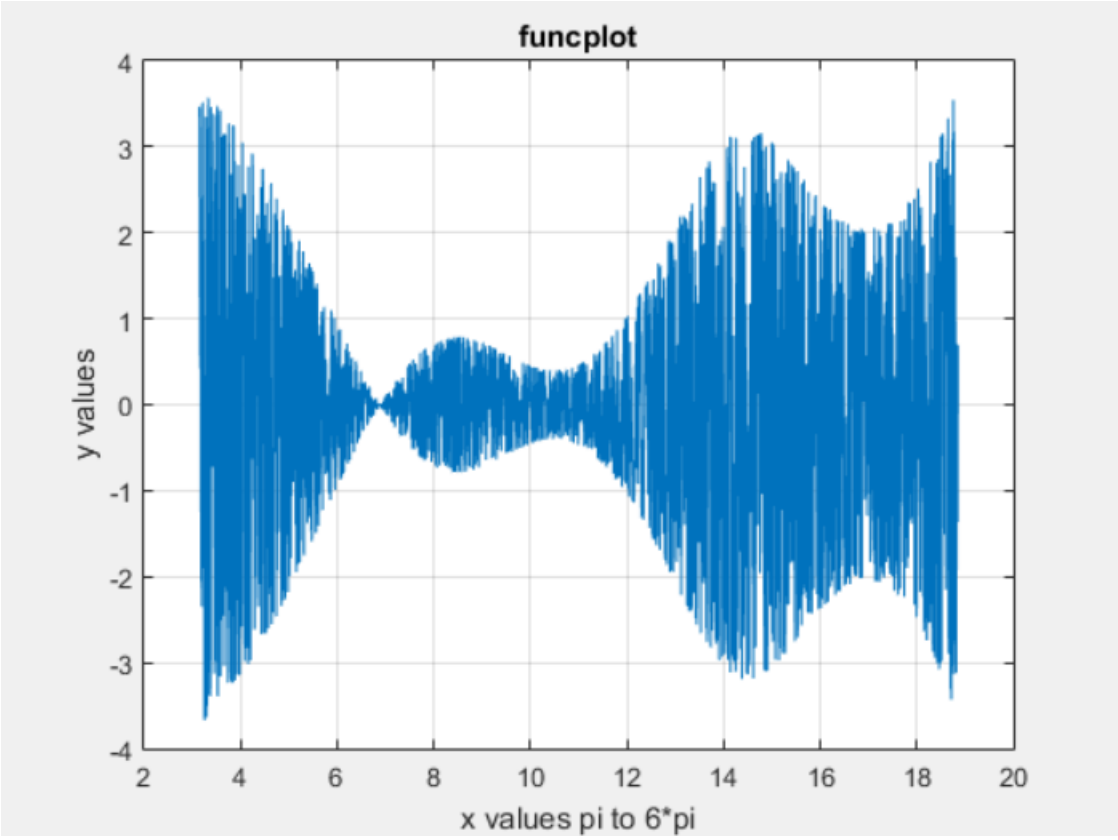
2nd question:

Second question asks that plot a function "f(x) = 3*sin (x(3*x)/x)*tan(ln(3*e(0.2*sin(x)) *x))"

```
1       %Mert KELKİT - 150115013
2       %Rümeysa ELİÖZ - 150114016
3       %Question 2
4
5       %evaluating x values
6 -     x = linspace(pi, 6*pi, 2000);
7       %making vector multiplication with the function
8 -     y = 3.*sin((x.^(3.*x))./x).*tan(log(3.*(exp(1).^(0.2.*sin(x)))).*x));
9       %plotting x values with the corresponding y values.
10 -    plot(x, y)
11 -    grid on;
12 -    xlabel('x values pi to 6*pi');
13 -    ylabel('y values');
14 -    title('funcplot');
```

We evaluate x values pi to 6*pi with 2000 points as a row vector. Then we declare a y variable with our given function with x named unknown because we named x values as “x”. We use array operators like “./, .*” because our x is an array, y will be array too. After evaluating y values, use plot(x,y) function. It means that plot y values with corresponding x values. After that, we set x label , y label and function with the functions between lines 12-14 respectively. “grid on;” means put a grid behind the graph.

Sample plot is:



3rd question:

a) This question asks that find the approximate pi value with Monte Carlo Simulation.

```
%Mert KELKİT - 150115013
%Rümeysa ELİÖZ - 150114016
%Question 3 - a

%Using Monte Carlo Simulation to approximate the pi value
function zpi = mypi( numpoints )
%success counter
t = 0;

%loop starts with one to numpoints(input)
for k = 1 : numpoints
    %generating random x and y coordinates between 0 and 1
    %because of unit circle !!!
    %random because of Monte Carlo Simulation
    x = rand(1);
    y = rand(1);

    %if randomly generated x and y points are in the unit circle
    if (x^2 + y^2)^(1/2) < 1
        %increment success
        t = t + 1;
    end
end
%it gives us approximate pi value
zpi = 4 * t / numpoints;
end
```

We declare a t variable which counts the successful trials. For loop starts with one to numpoints. It means that we have numpoints trials. We generate random x and y points between 0 and 1. Because we evaluate a approximated pi value with a unit circle. We test these generated x and y values with a general circle equation if these x and y points are in this unit circle or not. If (x,y) is in unit circle, increment t value. After end of the loop, we evalute approximate pi value with 4 multiplied by success counter divided by number of points.

Sample outputs:

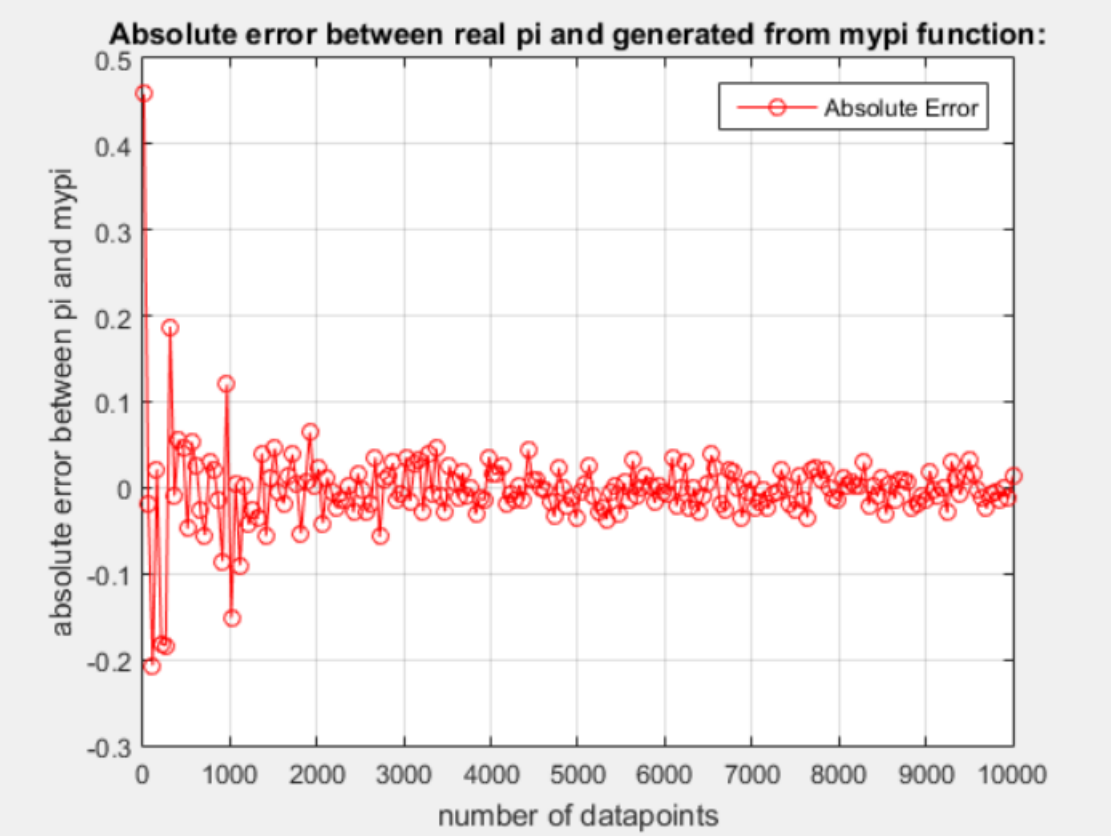
```
>> zpi = mypi(10000)

zpi =

    3.1572

>> zpi = mypi(100)

zpi =

    3.4000

>> zpi = mypi(7382)

zpi =

    3.1309
```

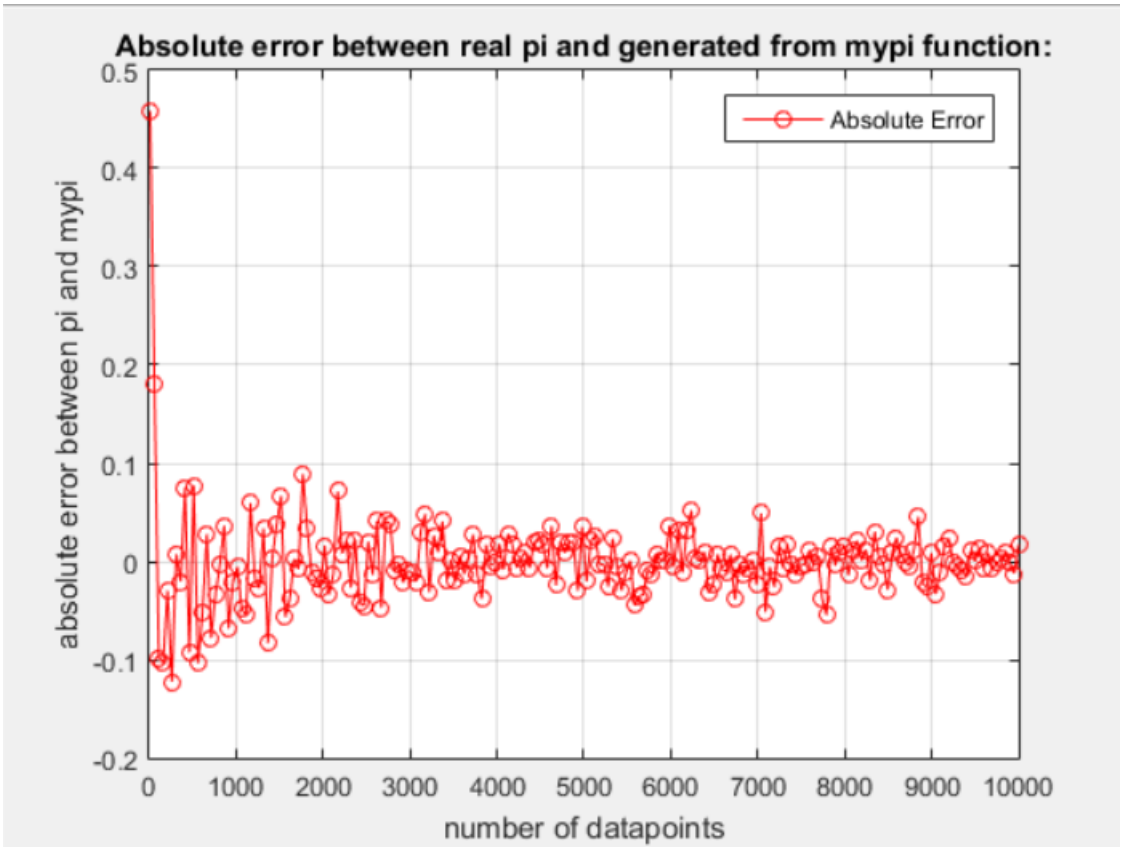
b) This question asks that find the absolute error between approximate pi value and normal pi value and plot it.

```
%Mert KELKİT - 150115013
%Rümeysa ELİÖZ - 150114016
%Question 3 - b

%creating a vector 10 to 10000 with 200 elements
x = linspace(10, 10000, 200);
%creating a pi vector which is full of 200 pi numbers.
piVector = repmat(pi, 1, 200);
%creating a y vector with 200 zeros.
y = zeros(1, 200);
%1 to 200...
for i = 1 : 200
    %calculate absolute error and add it to y vector's corresponding index.
    y(i) = mypi(x(i)) - piVector(i);
end
%plot and plot settings
plot(x, y, '-ro')
grid on;
xlabel('number of datapoints');
ylabel('absolute error between pi and mypi');
title('Absolute error between real pi and generated from mypi function:');
legend('Absolute Error');
```

We create a row vector which is 10 to 10000 with 200 points. After that we create a pi vector which is full of 200 pis and create a y vector which is full of zeros initially. Then fill this y vector with generated pi value minus normal pi value in a for loop. Then plot y values with corresponding x values. “-ro” means mark datapoints with a red circle. Plots may be different because Monte Carlo Simulation generates pi values randomly.

Sample plots:





We can see that absolute error decreases as numpoints increases.

4th question:

a) In this question, we use 2-point method, 3-point method and 5-point method in order to differentiate a given function. We write a function named "functionforq4"

```
%Mert KELKİT - 150115013
%Rümeysa ELİÖZ - 150114016
%Question 4-a

% we wrote a function named functionforq4 for this script to make
% computations easily.
x = pi;
h1 = 0.01;
h2 = 0.1;
h3 = 1;
% h = 0.01 , 2-point method
h1d1 = (functionforq4(x+h1)-functionforq4(x))/h1;
% h = 0.1 , 2-point method
h2d1 = (functionforq4(x+h2)-functionforq4(x))/h2;
% h = 1 , 2-point method
h3d1 = (functionforq4(x+h3)-functionforq4(x))/h3;
% h = 0.01 , 3-point method
h1d2 = ((-3*functionforq4(x))+(4*functionforq4(x+h1))-(functionforq4(x+(2*h1))))/2*h1;
% h = 0.1 , 3-point method
h2d2 = ((-3*functionforq4(x))+(4*functionforq4(x+h2))-(functionforq4(x+(2*h2))))/2*h2;
% h = 1 , 3-point method
h3d2 = ((-3*functionforq4(x))+(4*functionforq4(x+h3))-(functionforq4(x+(2*h3))))/2*h3;
% h = 0.01 , 5-point method
h1d3 = (functionforq4(x-(2*h1))-8*(functionforq4(x-h1))+8*(functionforq4(x+h1))-functionforq4(x+(2*h1)))/(12*h1);
% h = 0.1 , 5-point method
h2d3 = (functionforq4(x-(2*h2))-8*(functionforq4(x-h2))+8*(functionforq4(x+h2))-functionforq4(x+(2*h2)))/(12*h2);
% h = 1 , 5-point method
h3d3 = (functionforq4(x-(2*h3))-8*(functionforq4(x-h3))+8*(functionforq4(x+h3))-functionforq4(x+(2*h3)))/(12*h3);

```

2-point method is this :

$$f'(x_0) \approx \frac{f(x_0) - f(x_0 - h)}{h}.$$

3-point method is this:

$$f'(x_0) = \frac{-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)}{2h}$$

5-point method is this:

$$f'(x_0) = \frac{f(x_0 - 2h) - 8f(x_0 - h) + 8f(x_0 + h) - f(x_0 + 2h)}{12h}$$

f(x) is our functionforq4(x).

functionforq4 :

```
%Mert KELKİT - 150115013
%Rümeysa ELİÖZ - 150114016
%function for question 4.

function output = functionforq4( x )
output = 3*sin((x^(3*x))/x)*tan(log(3*(exp(1)^(0.2*sin(x)))*x));
end
```

We compute derivatives with every h and every method and make a table with these lines (in same script) :

```
%Setting up table...
rowNames = {'h = 0.01'; 'h = 0.1'; 'h = 1'};
Two_Point_Method = [h1d1; h2d1; h3d1];
Three_Point_Method = [h1d2; h2d2; h3d2];
Five_Point_Method = [h1d3; h2d3; h3d3];
table(Two_Point_Method, Three_Point_Method, Five_Point_Method, 'RowNames', rowNames)
```

Two_Point_Method is the column which has differentiates with h=0.01, h=0.1 and h=1 with 2-point method.

Three_Point_Method is the column which has differentiates with h=0.01, h=0.1 and h=1 with 3-point method.

Five_Point_Method is the column which has differentiates with h=0.01, h=0.1 and h=1 with 5-point method.

Also we created a rowNames array to set row names in table function.

Table is this:

```
ans =

                Two_Point_Method    Three_Point_Method    Five_Point_Method
                ________________    __________________    _________________

    h = 0.01    -26.164             0.0051653              455.1
    h = 0.1     -2.8566               0.01735              35.45
    h = 1       -1.0088               0.33155             1.0032
```

b) This question asks that integrate same functionforq4 with Trapezoidal Rule, Simpson's Rule and Simpson's 3/8 Rule.

We only made Trapezoidal Rule:

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}\left[f(x_0)+2f(x_1)+2f(x_2)+\cdots+2f(x_{n-1})+f(x_n)\right]$$

We created row vectors for x values which are a to b (boundaries)(a = 1.8, b=3.2) which increases by h(h1 = 0.01, h2 = 0.1, h3 = 1).
We calculate functionforq4 values with every x values as a row vector in a for loop. After that we sum all of the elements of this vector but first element and last element must be divided by 2 as in the formula. We made this computation by ./2 for y(1) and y(end) and extract them from y vector's sum. After that we multiply it by h value.

We create a table like this for functionforq4 with h1, h2, h3:

```
ans =

                Trapezoidal_Rule
                ________________

    h = 0.01    -0.14652
    h = 0.1      -2.8763
    h = 1        -4.9585
```

Corresponding Source Codes:

Q1 -> **plotpint.m**

Q2 -> **funcplot.m**

Q3 -> **mypi.m, ploterr.m**

Q4 -> **functionforq4.m, derivative.m, integration.m**

References:

1- https://www.mathworks.com/
2- https://stackoverflow.com/
3- http://tutorial.math.lamar.edu/
4- http://www.math.usm.edu/

Mert KELKİT – 150115013

Rümeysa ELİÖZ – 150114016