ECE 204 Simulation Assignment 4

Phoebe Luo
University of Waterloo

Output Results

Develop your Derivative / Integration code that can do the following:

a) Upload the data which will be in the form of x, f(x) (one file will be uploaded, test_1).
b) Once you run the program, it will ask you what you want to perform Derivative or Integration (you select one of them)

- The code is:

```
clc;

% Obtaining Data
A = readmatrix("test_1.txt");
x = A(:,1);
y = A(:,2);
n = size(x,1);
prompt1 = "Choose to perform 1. Derivative or 2. Integration: ";
command = input(prompt1);
```

c) If you selected Derivative, then the program will do the following:
   i. Ask you to decide at what point, p you want to perform the derivative
   ii. If the point is from the data set and the spacing between the points is even, then the program will calculate the derivative using the CDD method.
   iii. If the point is not from the data set and/or the spacing between the points is not even, then the program first will use polynomial regression to estimate the function and then perform Derivative using CDD method with h = minimum of Δx from the data points.
   iv. show the solution for following cases of test1.txt
      1) p = 7
      2) p = 18.5

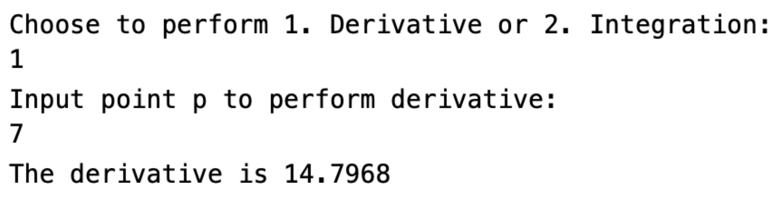
- The code is:

```
% Derivative
if command == 1
prompt2 = "Input point p to perform derivative: ";
p = input(prompt2);
% check for invalid p
if p < x(1,1)
disp("Invalid data point, lower than lowest bound.");
return;
elseif p == x(1,1)
disp("Invalid data point, at lowest bound.");
return;
elseif p > x(n,1)
disp("Invalid data point, higher than highest bound.");
return;
elseif p == x(n,1)
disp("Invalid data point, at highest bound.");
return;
```

```
end
check = 0; % set check to 1 if p found in x with even spacing
% p from data set with even spacing
for a = 1:n
if x(a,1) == p && abs(x(a-1,1)-p) == abs(x(a+1,1)-p)
check = 1;
d = (y(a+1,1)-y(a-1,1)) / (2*abs(x(a-1,1)-p));
end
end
% p from data set without even spacing & p not from data set
if check == 0
% polynomial regression to get 3rd order polynomial
e_1 = x.^2; % matrix of x^2
e_2 = x.^3; % matrix of x^3
        e_3 = x.^4; % matrix of x^4
        e_4 = x.^5; % matrix of x^5
        e_5 = x.^6; % matrix of x^6
        e_6 = x.*y; % matrix of xy
        e_7 = e_1.*y; % matrix of x^2y
        e_8 = e_2.*y; % matrix of x^3y
        % sum of the data
        sum_x = sum(x);
        sum_y = sum(y);
        sum_e_1 = sum(e_1);
        sum_e_2 = sum(e_2);
        sum_e_3 = sum(e_3);
        sum_e_4 = sum(e_4);
        sum_e_5 = sum(e_5);
        sum_e_6 = sum(e_6);
        sum_e_7 = sum(e_7);
        sum_e_8 = sum(e_8);
        % do matrix manipulation to get coefficients, y =
a0+a1*x+a2*x^2+a3*x^3
        A = [n sum_x sum_e_1 sum_e_2; sum_x sum_e_1 sum_e_2 sum_e_3; sum_e_1
sum_e_2 sum_e_3 sum_e_4; sum_e_2 sum_e_3 sum_e_4 sum_e_5];
        B = [sum_y; sum_e_6; sum_e_7; sum_e_8];
        C = inv(A)*B;
        a0 = C(1,1);
        a1 = C(2,1);
        a2 = C(3,1);
        a3 = C(4,1);
        % calculate R^2
        St = 0;
        Sr = 0;
        for c = 1:n
            % for St
            cal_1 = (y(c,1) - (sum(y)/n))^2; % (yi-avg(y))^2
            St = St + cal_1;
            % for Sr
            cal_2 = (y(c,1) - (a0 + a1*x(c,1) + a2*x(c,1)^2 +
a3*x(c,1)^3))^2; % (yi-y)^2
            Sr = Sr + cal_2;
        end
        error = (St - Sr)/St;
        % find h as the smallest Δx
        h = x(1,1)-x(2,1);
        for b = 1:(n-1)
            if (x(b,1)-x(b+1,1)) < h
                h = x(b,1)-x(b+1,1);
            end
        end
        % find derivative
        x1 = p+h;
        x2 = p-h;
        d = ((a0+a1*x1+a2*x1^2+a3*x1^3)-(a0+a1*x2+a2*x2^2+a3*x2^3))/(2*h);
        % plot
        plot(x,y,'.');
        xlabel('x');
        ylabel('y');
        hold on;
        plot(x, a0+a1*x+a2*(x.^2)+a3*(x.^3));
        gravstr = 'Polynomial, y = %0.4f + %0.4fx + %0.4fx^{2} + %0.4fx^{3}, 
R^{2} = %0.4f';
        gravstr=sprintf(gravstr, a0, a1, a2, a3, error);
        legend("Actual Data", gravstr);
    end
    disp("The derivative is " + d);
end
```

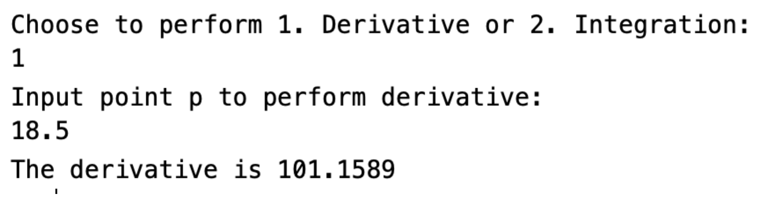
- The output is:

1. **p = 7:**



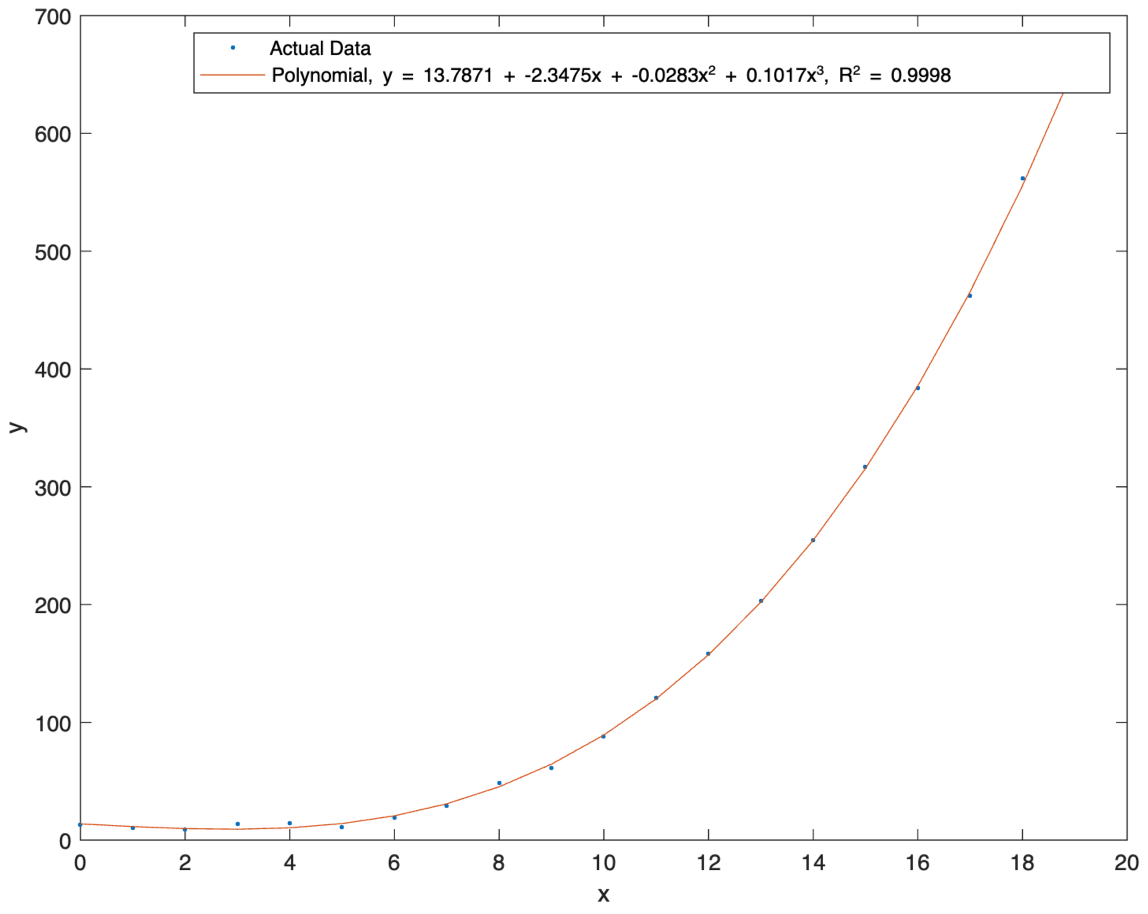
```
Choose to perform 1. Derivative or 2. Integration:
1
Input point p to perform derivative:
7
The derivative is 14.7968
```

2. **p = 18.5:**



```
Choose to perform 1. Derivative or 2. Integration:
1
Input point p to perform derivative:
18.5
The derivative is 101.1589
```

Polynomial regression is used and show the plot containing actual data point, fitted curve, R2, and the coefficients of the polynomial:



d) If you selected Integration, then the program will do the following:
   i. Ask for the integration limit; p1, p2 where p2 > p1
   ii. Ask for the number of segments; n
   iii. Generate new data set with the limits p1, p2 and n evenly spaced segments. If all the values of the new data set belong to the original data set, then the program will calculate the Integration simply using the discrete Trapezoidal method.
   iv. If the limits (p1 and p2) are within the range of the original data set, but any of the values of the new data set do not belong to the original dataset, then the program first will use polynomial regression to estimate the function and then perform Integration using the Trapezoidal method. Select appropriate Δx.
   v. If at least one of the limits (p1 and p2) is out of the range of the original data set, display a message specifying the error (without calculating the integral).
   vi. show the solution for following cases of test1.txt
      1) p1 = 3, p2 = 7, n = 4
      2) p1 = 3, p2 = 7, n = 10

- The code is:

```
% Integration
```

```
if command == 2
prompt3 = "Input lower limit point p1 to perform integration(p1 < p2): ";
p1 = input(prompt3);
prompt4 = "Input upper limit point p2 to perform integration(p1 < p2): ";
p2 = input(prompt4);
prompt5 = "Input number of segments: ";
n_seg = input(prompt5);
% check for invalid p1, p2
if p1 > p2
disp("Invalid data point, lower limit greater than upper limit(p1 > p2).");
return;
elseif p1 < x(1,1)
disp("Invalid data point, p1 lower than lowest bound.");
return;
elseif p2 > x(n,1)
disp("Invalid data point, p2 higher than highest bound.");
return;
elseif p1 == x(1,1)
disp("Invalid data point, p1 at lowest bound.");
return;
elseif p2 == x(n,1)
disp("Invalid data point, p2 at highest bound.");
return;
end
% generate new data set x_new
h = (p2 - p1)/n_seg; % length of each segment
x_new = zeros(n_seg+1,1); % new data set with n_seg+1 new x's
x_new(1,1) = p1; % set first value to be lower bound p1
x_new(n_seg+1,1) = p2; % set last value to be upper bound p2
for b = 1:(n_seg-1)
x_new(b+1,1) = p1 + b*h;
end
check = 1; % set check to 0 if all the values of the new data set belong to
the original
% check if any of the values of the new data set do not belong to the
original
count = 0;
for d = 1:n
for e = 1:(n_seg+1)
if x(d,1) == x_new(e,1) % find data that belongs
count = count + 1;
end
end
end
if count == (n_seg+1)
check = 0;
end
% Discrete Trapezoidal Method, all values of new data set belong to the
original data set
if check == 0
% generate new data set y_new
y_new = zeros(n_seg+1,1); % new data set with n_seg+1 new y's
for g = 1:n
for h = 1:(n_seg+1)
if x(g,1) == x_new(h,1)
```

```
y_new(h,1) = y(g,1);
end
end
end
% calculate summation of y's
y_total = 0;
for f = 2:n_seg
y_total = y_total + y_new(f,1);
end
i = ((p2 - p1)/ (2*n_seg)) * (y_new(1,1) + 2*y_total + y_new(n_seg+1,1));
end

    % any of the values of the new data set do not belong to the original
    if check == 1
        % polynomial regression to get 3rd order polynomial
        e_1 = x.^2; % matrix of x^2
        e_2 = x.^3; % matrix of x^3
        e_3 = x.^4; % matrix of x^4
        e_4 = x.^5; % matrix of x^5
        e_5 = x.^6; % matrix of x^6
        e_6 = x.*y; % matrix of xy
        e_7 = e_1.*y; % matrix of x^2y
        e_8 = e_2.*y; % matrix of x^3y
        % sum of the data
        sum_x = sum(x);
        sum_y = sum(y);
        sum_e_1 = sum(e_1);
        sum_e_2 = sum(e_2);
        sum_e_3 = sum(e_3);
        sum_e_4 = sum(e_4);
        sum_e_5 = sum(e_5);
        sum_e_6 = sum(e_6);
        sum_e_7 = sum(e_7);
        sum_e_8 = sum(e_8);
        % do matrix manipulation to get coefficients, y =
a0+a1*x+a2*x^2+a3*x^3
        A = [n sum_x sum_e_1 sum_e_2; sum_x sum_e_1 sum_e_2 sum_e_3; sum_e_1
sum_e_2 sum_e_3 sum_e_4; sum_e_2 sum_e_3 sum_e_4 sum_e_5];
        B = [sum_y; sum_e_6; sum_e_7; sum_e_8];
        C = inv(A)*B;
        a0 = C(1,1);
        a1 = C(2,1);
        a2 = C(3,1);
        a3 = C(4,1);
        % calculate R^2
        St = 0;
        Sr = 0;
        for c = 1:n
            % for St
            cal_1 = (y(c,1) - (sum(y)/n))^2; % (yi-avg(y))^2
            St = St + cal_1;
            % for Sr
            cal_2 = (y(c,1) - (a0 + a1*x(c,1) + a2*x(c,1)^2 +
a3*x(c,1)^3))^2; % (yi-y)^2
```

```
            Sr = Sr + cal_2;
        end
        error = (St - Sr)/St;

        % generate new data set y_new
        y_new = zeros(n_seg+1,1); % new data set with n_seg+1 new y's
        for g = 1:(n_seg+1)
            x_g = x_new(g,1);
            y_new(g,1) = a0 + a1 * x_g + a2 * x_g^2 + a3 * x_g^3;
        end

        % calculate summation of y's
        y_total = 0;
        for f = 2:n_seg
            y_total = y_total + y_new(f,1);
        end
        i = ((p2 - p1)/ (2*n_seg)) * (y_new(1,1) + 2*y_total +
y_new(n_seg+1,1));
        % plot
        plot(x,y,'.');
        xlabel('x');
        ylabel('y');
        hold on;
        plot(x, a0+a1*x+a2*(x.^2)+a3*(x.^3));
        gravstr = 'Polynomial, y = %0.4f + %0.4fx + %0.4fx^{2} + %0.4fx^{3},
R^{2} = %0.4f';
        gravstr=sprintf(gravstr, a0, a1, a2, a3, error);
        legend("Actual Data", gravstr);
    end
    disp("The integral is " + i);
end
```

- The output is:

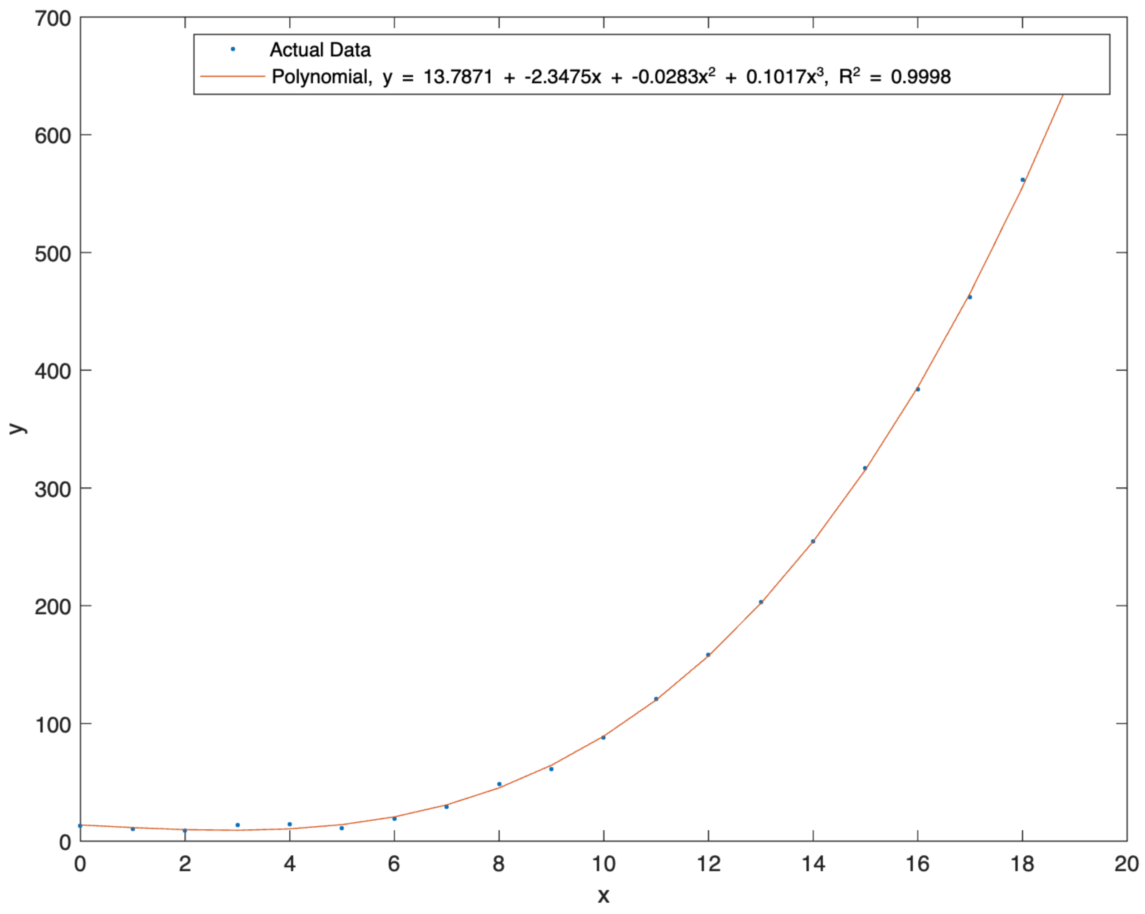
1. p1 = 3, p2 = 7, n = 4:

```
Choose to perform 1. Derivative or 2. Integration:
2
Input lower limit point p1 to perform integration(p1 < p2):
3
Input upper limit point p2 to perform integration(p1 < p2):
7
Input number of segments:
4
The integral is 65.3392
```

2. p1 = 3, p2 = 7, n = 10:

```
Choose to perform 1. Derivative or 2. Integration:
2
Input lower limit point p1 to perform integration(p1 < p2):
3
Input upper limit point p2 to perform integration(p1 < p2):
7
Input number of segments:
10
The integral is 64.3792
```

Polynomial regression is used and show the plot containing actual data point, fitted curve, R2, and the coefficients of the polynomial:


Actual Data
Polynomial, $y = 13.7871 + -2.3475x + -0.0283x^2 + 0.1017x^3$, $R^2 = 0.9998$
y
x