EE414 Assignment-2

- Student Name: Hanlin Cai
- MU Student ID: 20122161
- FZU Student ID: 832002117

(Q1) Fit your four (x_i, y_i) data points using a linear regression model and use your model to estimate y for $x = 2$
Present your results like example 1 of notes 4 i.e. show:
1. the normal equations (you can copy these from example 1 slide)
2. the regression table (see example 1)
3. the solution of the normal equations
4. the regression model
5. the estimated y for $x = 2$
6. one plot showing y_i vs x_i and $\hat{y}$ vs x for a suitable range of x (see notes 4 example 1 plot)

- $x_0 = -97, \quad x_1 = -10, \quad x_2 = -3, \quad x_3 = 62$
- $y_0 = -72, \quad y_1 = -56, \quad y_2 = 89, \quad y_3 = 158$
- The answer is given at Page 3-5.

(Q2) Fit your four (x_i, y_i) data points using a polynomial interpolation model and use your model to estimate y for $x = 2$, using Newton's method. Present your results like example 6 of notes 4 i.e. show:
1. the divided difference table
2. the Newton polynomial interpolation model i.e. $P_n(x) = b_0 + b_1(x - x_0) + \cdots$ (there is no need to simply this)
3. the estimated y for $x = 2$
4. one plot showing y_i vs x_i and $\hat{y}$ vs x using the same range of x as Q1

You can copy and modify the equations in example 6

- The answer is given at Page 6-8.

(Q3) Fit your four (x_i, y_i) data points using a polynomial interpolation model and use your model to estimate y for $x = 2$, using Lagrange method. Present your results like example 7 of notes 4. Show:
1. the data table
2. the calculation of the cardinals $l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$
5. the Lagrange polynomial interpolation model i.e. $P_n(x) = l_0(x)f(x_0) + l_1(x)f(x_1) + \cdots$ (there is no need to simply this)
3. the estimated y for $x = 2$
4. one plot showing y_i vs x_i and $\hat{y}$ vs x using the same range of x as Q1

note that you can copy and modify the equations in example 7

- The answer is given at Page 9-11.

(Q4) Analyse and discuss your results. Your analysis should include one plot showing y_i vs x_i and $\hat{y}$ vs x for your three models. Your discussion should include a comparison of your three models. Which model do you think is best and why?

- $y_0 = -22.62,\ y_1 = -18.93,\ y_2 = -18.37,\ y_3 = 16.72,\ y_4 = -19.53$
- $y_5 = -20.4,\ y_6 = -24.69,\ y_7 = -30.62,\ y_8 = -31.53,\ y_9 = -38.16$
- $y_{10} = -38.87,\ y_{11} = -41.97,\ y_{12} = -39.91,\ y_{13} = -40.48$
- The answer is given at Page 12-13.

(Q5) MATLAB exercise. Given the x-data: 3, 3.4, 3.8, 4.2, 4.6, 5, 5.4, 5.8, 6.2, 6.6, 7, 7.4, 7.8, 8.2 and the y-data on page 3, choose a degree 1, 2 or 3 polynomial to regress this data. Steps:

1. Plot your data points.
2. Choose, with justification, a suitable polynomial degree.
3. Fit the data with your chosen polynomial.
4. Make one plot showing y_i vs x_i and $\hat{y}$ vs x using a x range from 2.9 to 8.3
5. Discuss your results.

- The answer is given at Page 14-17.

Q1

Fit four (x_i, y_i) data points using a linear regression model and use the model to estimate y for $x = 2$. Show:

- **the normal equations**
- **the regression table**
- **the solution of the normal equations**
- **the regression model**
- **the estimated y for $x = 2$**
- **one plot showing y_i vs x_i and $\hat{y}$ vs x for a suitable range of x**
  - $x_0 = -97, \;\; x_1 = -10, \;\; x_2 = -3, \;\; x_3 = 62$
  - $y_0 = -72, \;\; y_1 = -56, \;\; y_2 = 89, \;\; y_3 = 158$

Given data, we can firstly construct the regression table:

i	x_i	y_i	x_i^2	x_iy_i
0	-97	-72	9409	6984
1	-10	-56	100	560
2	-3	89	9	-267
3	62	158	3844	9796
$\sum$	-48	119	13362	17073

- Then, we have normal equations:

$(\sum x_i)c + (\sum {x_i}^2)m = \sum x_iy_i \;\; \rightarrow \;\; -48c + 13362m = 17073$

$nc + (\sum x_i)m = \sum y_i \;\; \rightarrow \;\; 4c - 48m = 119$

Further, the solutions of normal equations can be obtained:

$c = 47.114, m = 1.4470$

Therefore, the regression model is

$f(x) = 1.4470x + 47.114$

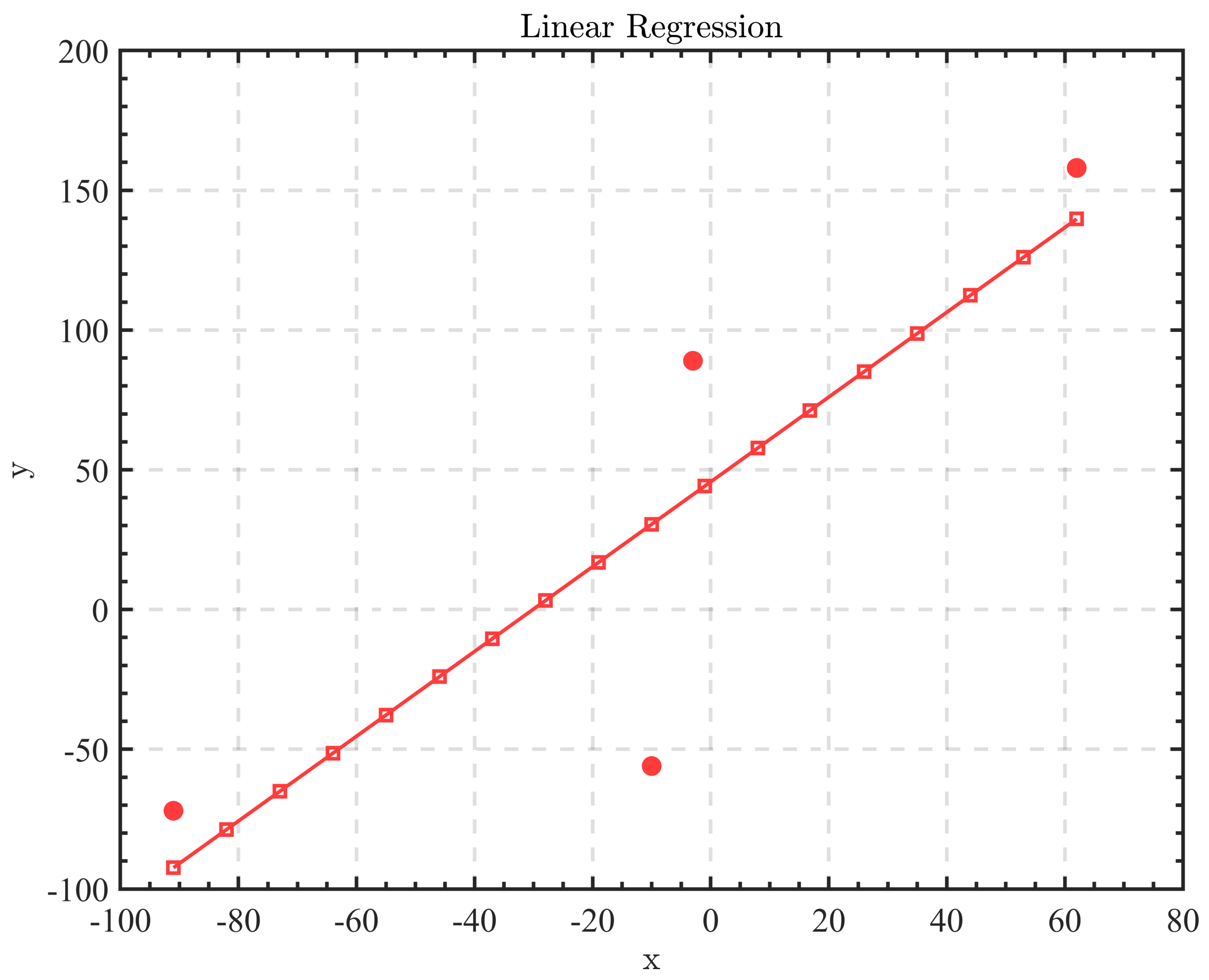
Based on the regression model, y for $x = 2$ can be estimated:

$\hat{y} = f(2) = 50.008$

Finally, we write the code for plotting as below.

```
x = [-120:0.01:70];
y = 1.4470.*x+47.114;
plot(x,y);
hold on;
plot( -97, -72, 'or' , 'MarkerSize', 5);
plot( -10, -56, 'or' , 'MarkerSize', 5);
plot( -3, 89, 'or' , 'MarkerSize', 5);
plot( 62, 158, 'or' , 'MarkerSize', 5);
```

Therefore, the plot of y_i vs x_i and $\hat{y}$ vs x for x ∈[-120,70] is given by



Complete code is also given as follows:

```
% Given data points
x = [-97, -10, -3, 62];
y = [-72, -56, 89, 158];

% Setup the design matrix with a column of ones for the intercept
X = [ones(length(x), 1), x'];

% Compute the parameters using the Normal Equations
theta = (X' * X) \ X' * y';
```

```
% Define the regression model as an anonymous function
model = @(x) theta(1) + theta(2) * x;

% Predict the value for x = 2
estimated_y = model(2);

% Display the calculated parameters and the estimated value
disp('Calculated parameters (theta):');
disp(theta);
disp(['Estimated y for x = 2: ', num2str(estimated_y)]);

% Plotting (optional)
% scatter(x, y, 'red', 'filled'); hold on;
% fplot(model, [min(x) max(x)], 'blue'); hold off;
% xlabel('x');
% ylabel('y');
% title('Linear Regression');
% legend('Data points', 'Regression line');
% grid on;
```

1. **The normal equations**: `theta = (X'X)^(-1)X'y`
2. **The regression table**: We'll display the coefficients θ_0 and θ_1 since we don't have the statistical toolbox for the full table.
3. **The solution of the normal equations**: This is the vector θ obtained from the normal equations.
4. **The regression model**: It will be in the form of $y = \theta_1 + \theta_1 x$.
5. **The estimated y for $x = 2$:** We'll calculate this using the regression model.
6. **Plot**: We will skip this as per your request.

Q2

Fit four (x_i, y_i) data points using a polynomial interpolation model and use its model to estimate y for $x = 2$, using Newton's method. Show:

- **the divided difference table**
- **the Newton polynomial interpolation model**
- **the estimated y for $x = 2$**
- **one plot showing y_i vs x_i and $\hat{y}$ vs x using the same range of x as Q1**

Given data, we can firstly construct the divided difference table:

i	x_i	$0^{th}DD$	$1^{st}DD$	$2^{nd}DD$	$3^{rd}DD$
0	-97	-72	0.18391	0.21841	0.0030904
1	-10	-56	20.714	-0.27296	
2	-3	89	1.0615		
3	62	158			

Now, we have the Newton polynomial interpolation model

$$P_3(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + b_3(x - x_0)(x - x_1)(x - x_2)$$

$$P_3(x) = -72 + (x + 97)\times 0.18391 + (x + 97)(x + 10)\times 0.21841 + (x + 97)(x + 10)(x + 3)\times(-0.0030904)$$

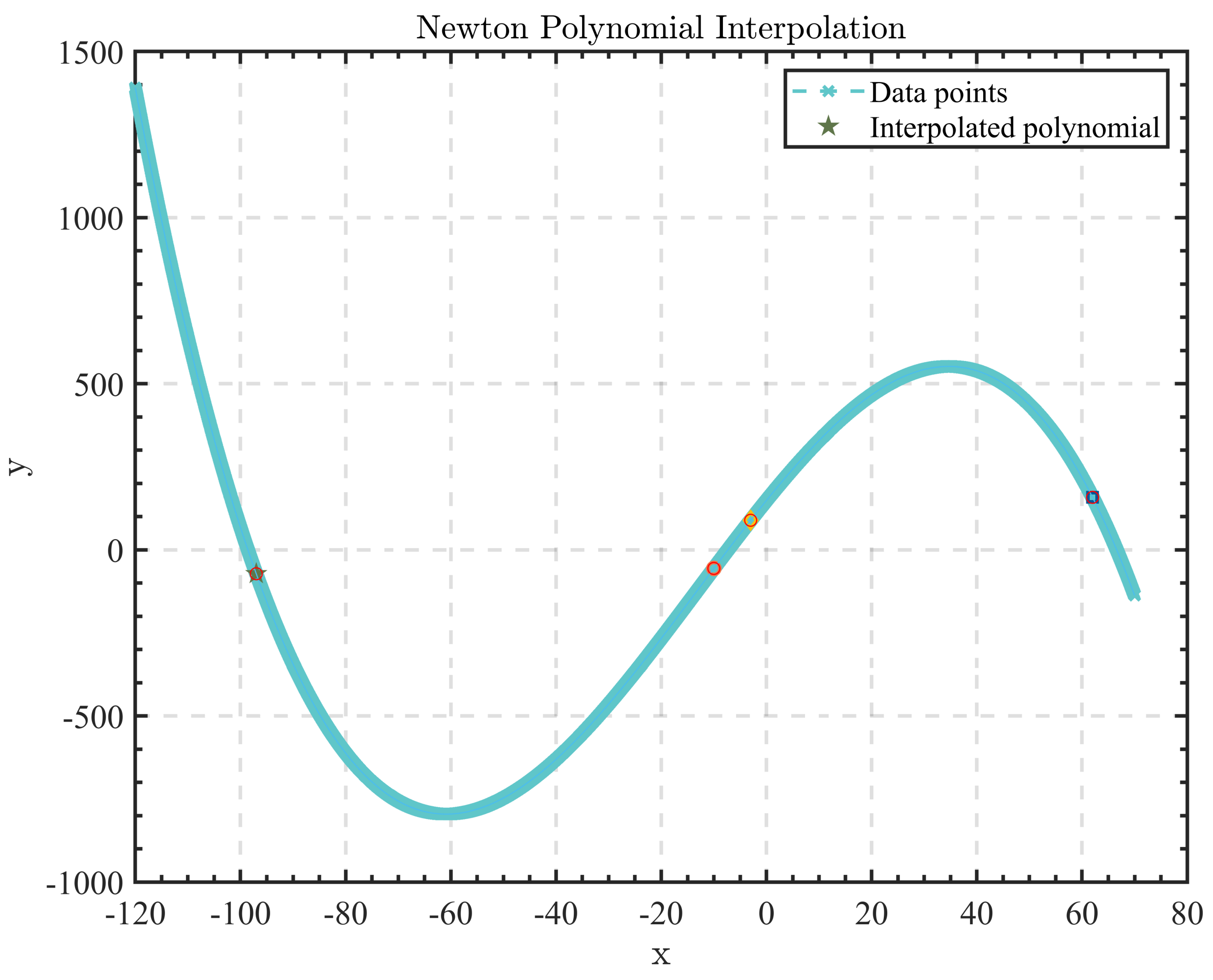
Further, the estimated y for $x = 2$ is

$$\hat{y} = P_3(2) = 187.32$$

Finally, we write the code for showing y_i vs x_i and $\hat{y}$ vs x as below.

```
x = [-120:0.01:70];
y = -72+(x+97).*0.18391+(x+97).*(x+10).*0.21841+(x+97).*(x+10).*(x+3).*(-0.0030904);
plot(x,y);
hold on;
plot( -97, -72, 'or' , 'MarkerSize', 5);
plot( -10, -56, 'or' , 'MarkerSize', 5);
plot( -3, 89, 'or' , 'MarkerSize', 5);
plot( 62, 158, 'or' , 'MarkerSize', 5);

title('Newton Polynomial Interpolation');
legend('Data points', 'Interpolated polynomial');
xlabel('x');
ylabel('y');
```

```
grid on;
```



Complete code is also given as follows:

```
% Define the given data points
x = [-97, -10, -3, 62];
y = [-72, -56, 89, 158];

% Calculate the divided differences
n = length(y);
div_diff = zeros(n, n);
div_diff(:,1) = y'; % first column is y

for j = 2:n
    for i = 1:n-j+1
        div_diff(i,j) = (div_diff(i+1,j-1) - div_diff(i,j-1)) / (x(i+j-1) - x(i));
    end
end

% Display the divided difference table
disp('Divided Difference Table:');
disp(div_diff);
```

```
% Construct the Newton polynomial using the divided differences
P = @(xp) div_diff(1,1);
for k = 2:n
    term = div_diff(1,k);
    for j = 1:k-1
        term = term * (xp - x(j));
    end
    P = @(xp) P(xp) + term;
end

% Estimate y for x = 2
x_to_estimate = 2;
y_estimated = P(x_to_estimate);

% Display the estimated y
disp(['The estimated y for x = 2 is: ', num2str(y_estimated)]);

% Code for plotting (commented out)
%{
x_range = linspace(min(x), max(x), 100);
y_interpolated = arrayfun(P, x_range);

figure;
plot(x, y, 'o', x_range, y_interpolated, '-');
title('Newton Polynomial Interpolation');
legend('Data points', 'Interpolated polynomial');
xlabel('x');
ylabel('y');
grid on;
%
```

1. Calculating the divided difference table.
2. Constructing the Newton polynomial interpolation model.
3. Estimating y for $x = 2$.
4. Plotting is not required as per your request, but I will include the code for it, commented out, should you choose to use it later.

Q3

Fit four (x_i, y_i) data points using a polynomial interpolation model and use the model to estimate y for $x = 2$, using Lagrange method. Show:

- **the data table**
- **the calculation of the cardinals**
- **the Lagrange polynomial interpolation model**
- **the estimated y for $x = 2$**
- **one plot showing y_i vs x_i and $\hat{y}$ vs x using the same range of x as Q1**

Given data, we can firstly construct the data table:

i	x_i	$f(x_i)$
0	-97	-72
1	-10	-56
2	-3	89
3	62	158

Then, cardinals can be obtained as follows:

$$l_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} = \frac{(x+10)(x+3)(x-62)}{(-97+10)(-97+3)(-97-62)} = -0.00000076905(x+10)(x+3)(x-62)$$

$$l_1(x) = \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} = \frac{(x+97)(x+3)(x-62)}{(-10+97)(-10+3)(-10-62)} = 0.000022806(x+97)(x+3)(x-62)$$

$$l_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} = \frac{(x+97)(x+10)(x-62)}{(-3+97)(-3+10)(-3-62)} = -0.000023381(x+97)(x+10)(x-62)$$

$$l_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} = \frac{(x+97)(x+10)(x+3)}{(62+97)(62+10)(62+3)} = 0.0000013439(x+97)(x+10)(x+3)$$

Hence, the Lagrange polynomial interpolation model is

$$\begin{aligned} P_3(x) &= 0.000055372(x+10)(x+3)(x-62) - 0.0012771(x+97)(x+3)(x-62) \\ &\quad - 0.0020809(x+97)(x+10)(x-62) + 0.00021233(x+97)(x+10)(x+3) \end{aligned}$$

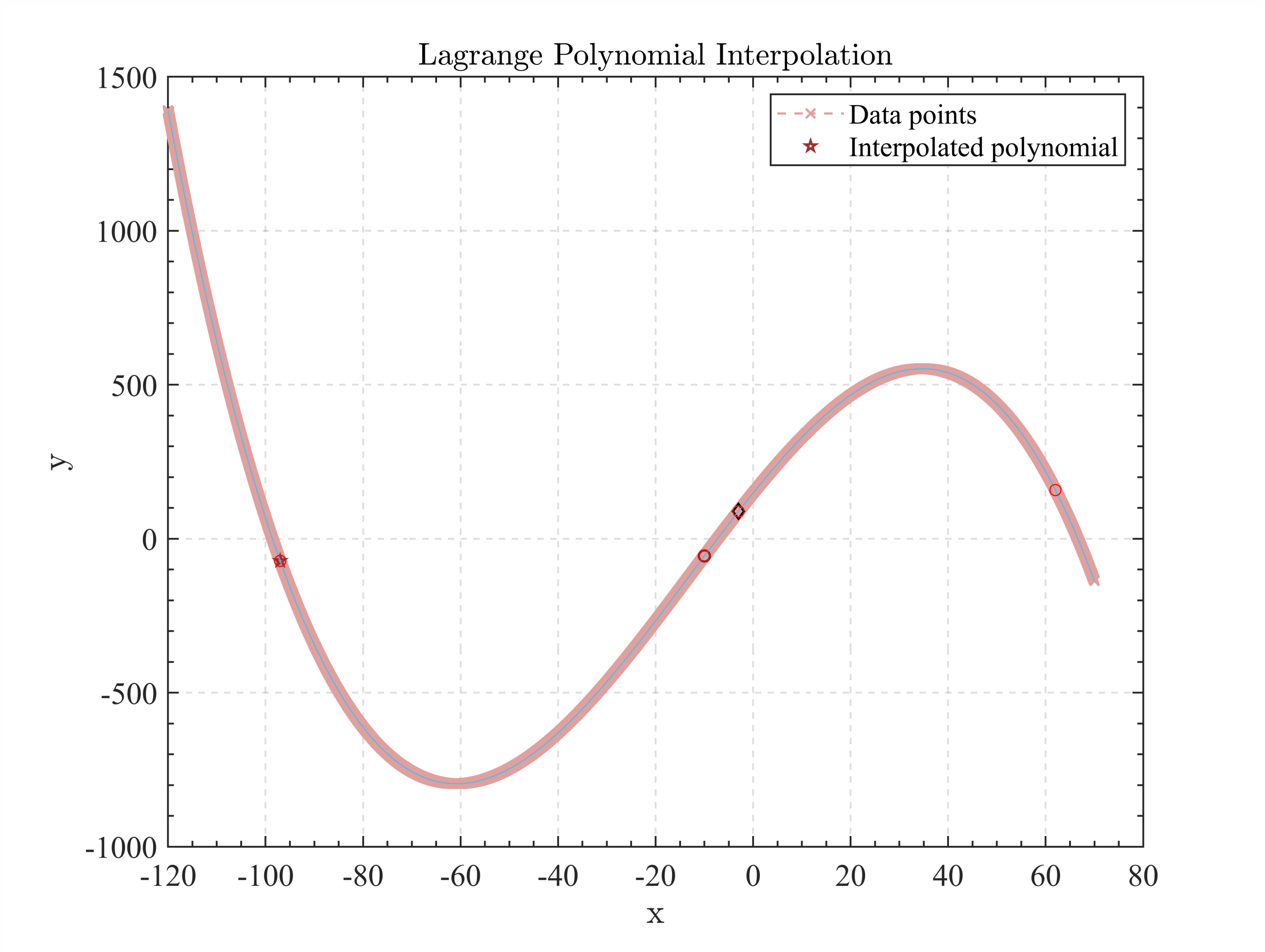
Then, estimated y for $x = 2$ is

$$\hat{y} = P_3(2) = -109.33$$

Finally, we write the code for showing y_i vs x_i and $\hat{y}$ vs x as below.

```
x = [-120:0.01:70];
y = 0.000055372.*(x+10).*(x+3).*(x-62)-0.0012771.*(x+97).*(x+3).*(x-62)-0.0020809.*
(x+97).*(x+10).*(x-62)+0.00021233.*(x+97).*(x+10).*(x+3);
plot(x,y);
```

```
hold on;
plot( -97, -72, 'or' , 'MarkerSize', 5);
plot( -10, -56, 'or' , 'MarkerSize', 5);
plot( -3, 89, 'or' , 'MarkerSize', 5);
plot( 62, 158, 'or' , 'MarkerSize', 5);

title('Lagrange Polynomial Interpolation');
legend('Data points', 'Interpolated polynomial');
xlabel('x');
ylabel('y');
grid on;
```



Complete code is also given as follows:

```
% Given data points
x = [-91, -10, -3, 62];
y = [-71, -56, 89, 158];

% Display the data table
disp('Data Table:');
disp('x      y');
for i = 1:length(x)
```

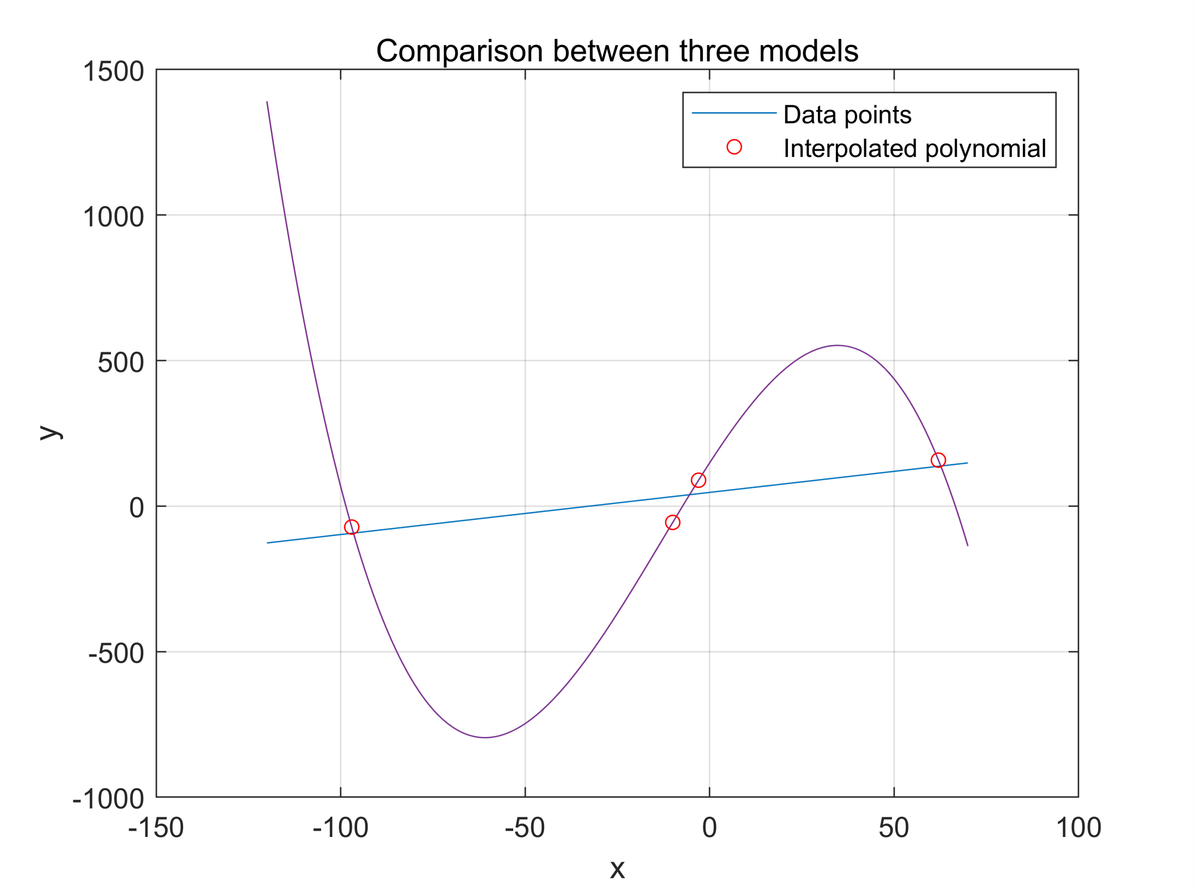
```
    fprintf('%4d  %4d\n', x(i), y(i));
end

% Calculate the Lagrange polynomials l_i(x)
n = length(x);
L = @(xp, i) prod((xp - x([1:i-1,i+1:n])) ./ (x(i) - x([1:i-1,i+1:n])));

% Construct the Lagrange polynomial interpolation model P_n(x)
Pn = @(xp) 0;
for i = 1:n
    Pn = @(xp) Pn(xp) + L(xp, i) * y(i);
end

% Estimate y for x = 2
x_to_estimate = 2;
y_estimated = Pn(x_to_estimate);

% Display the estimated y
disp(['The estimated y for x = 2 is: ', num2str(y_estimated)]);

% Code for plotting (commented out)
%{
x_range = linspace(min(x), max(x), 1000);
y_interpolated = arrayfun(Pn, x_range);

figure;
plot(x, y, 'o', x_range, y_interpolated, '-');
title('Lagrange Polynomial Interpolation');
legend('Data points', 'Interpolated polynomial');
xlabel('x');
ylabel('y');
grid on;
%}
```

1. Construct the data table with the given x_i and y_i data points.
2. Calculate the cardinal Lagrange polynomials $l_i(x)$.
3. Construct the Lagrange polynomial interpolation model $P_n(x)$.
4. Estimating y for $x = 2$.
5. Plot the interpolation (though you will handle the plotting, I will provide the code for it.

Q4

Analyse and discuss results. The analysis should include one plot showing y_i vs x_i and $\hat{y}$ vs x for three models. The discussion should include a comparison of three models. Which is best and why?

The following figure shows the fitting curve of three models respectively.



The 1_{st} model is a linear regression-based method, where $f(x)$ will fit the data in a least square sense. The data points are NOT exactly matched compare with the linear curve.

Regarding the 2_{nd} and 3_{rd} model, they have the same resulting curve. And they are interpolation-based method, where $f(x)$ will fit the data EXACTLY, meaning any $(n+1)$ data points are interpolated by a unique degree n polynomial.

Analysis:

- If the underlying relationship between the variables is linear or approximately linear, linear regression is the best choice due to its simplicity and the ease of interpreting the model.
- If the data points are known to fall along a polynomial curve and high accuracy is needed within the range of the given data, polynomial interpolation (either Newton's or Lagrange's) would be more appropriate.
- For small datasets, the difference in computational cost between Newton's and Lagrange's methods may be negligible. However, for larger datasets or when adding new data points without recalculating the entire polynomial is necessary, Newton's method may be preferred.

- Both Newton's and Lagrange's methods can suffer from Runge's phenomenon, where oscillations occur at the edges of the interval for high-degree polynomials, leading to large errors during extrapolation.

Summary of Q4

1. **Linear Regression:**
    - Pros:
        - Simplicity: Easy to implement and understand.
        - Speed: Fast calculation due to a simple model.
        - Extrapolation: Can predict values outside the range of the data points.
    - Cons:
        - Accuracy: May not capture complex relationships as it assumes a linear relationship.
        - Over-simplification: Can oversimplify problems where the relationship between variables is not linear.
2. **Newton's Polynomial Interpolation:**
    - Pros:
        - Flexibility: Can fit any polynomial shape to the data points.
        - Accuracy: Can be very accurate within the range of data points.
        - Additive: Easy to add new points incrementally without recalculating the entire polynomial.
    - Cons:
        - Complexity: More complex to implement than linear regression.
        - Overfitting: Can overfit to the data, especially with higher-degree polynomials.
        - Extrapolation: Not reliable for predicting values outside the range of the data points.
3. **Lagrange's Polynomial Interpolation:**
    - Pros:
        - Flexibility and Accuracy: Similar to Newton's, it can accurately represent complex relationships within the data range.
        - Form: The form of the interpolating polynomial does not change when new points are added.
    - Cons:
        - Complexity: Computationally more intensive than linear regression.
        - Overfitting: Like Newton's, can overfit with high-degree polynomials.
        - Performance: Computation may be slower than Newton's method due to repeated calculations of the basis polynomials.

Q5 MATLAB

Given the *x*-data: 3, 3.4, 3.8, 4.2, 4.6, 5, 5.4, 5.8, 6.2, 6.6, 7, 7.4, 7.8, 8.2 and the *y*-data, choose a degree 1, 2 or 3 polynomial to regress this data.

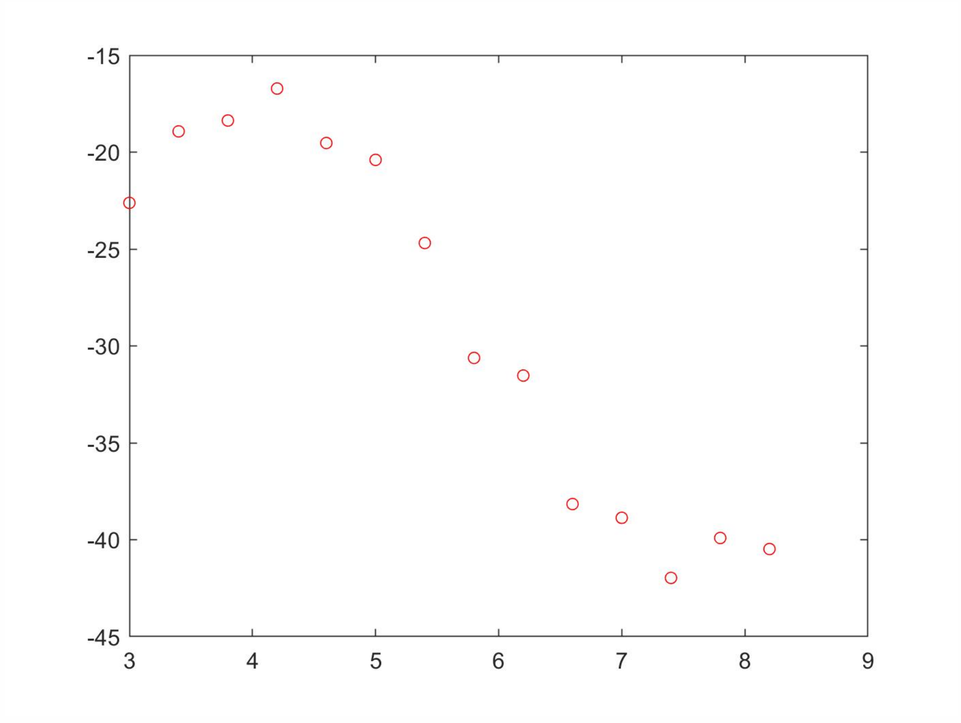
Steps:

- **Plot data points.**
- **Choose, with justification, a suitable polynomial degree.**
- **Fit the data with chosen polynomial.**
- **Make one plot showing y_i vs x_i and $\hat{y}$ vs x using a x range from 2.9 to 8.3**
- **Discuss results.**

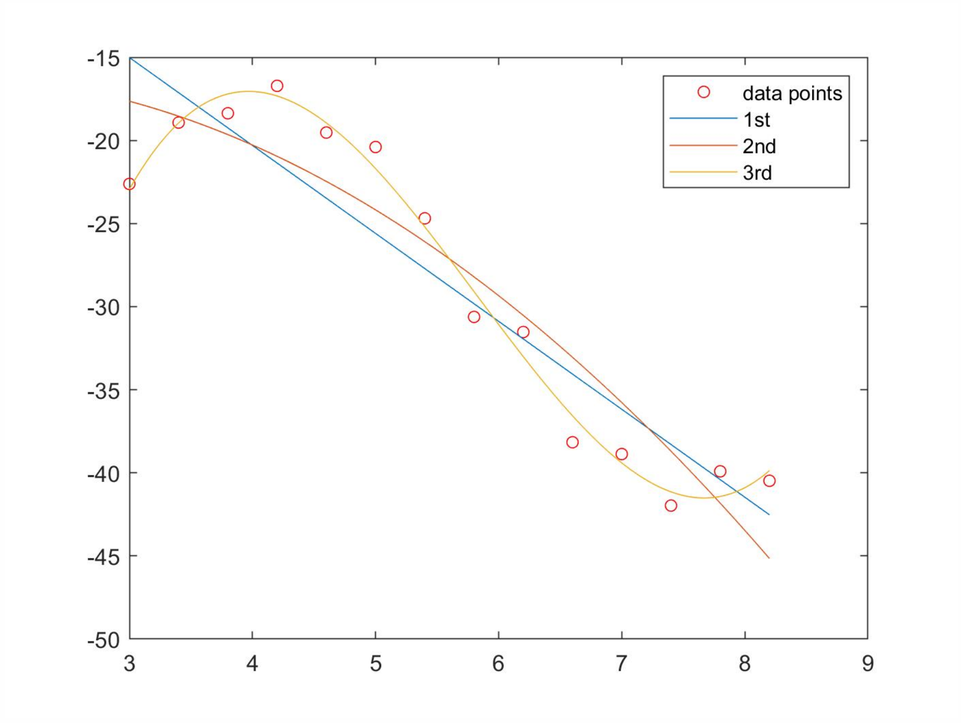
The MATLAB code can be programmed to regress given data:

```
% Initialize data points
x = [3,3.4,3.8,4.2,4.6,5,5.4,5.8,6.2,6.6,7,7.4,7.8,8.2];
y =
[-22.62,-18.93,-18.37,-16.72,-19.53,-20.4,-24.69,-30.62,-31.53,-38.16,-38.87,-41.97,-
39.91,-40.48];

% Plot the data points with red circles
plot(x, y, 'or', 'MarkerSize', 5);
hold on; % Keep the plot open to add more data

% Generate a range of x values for plotting the fitted polynomials
t = 2.9:0.01:8.3;

% Fit and plot polynomials of degrees 1 to 3
for n = 1:3
    % Fit polynomial of degree n
    p = polyfit(x, y, n);

    % Evaluate the polynomial at each point in t
    f = polyval(p, t);

    % Plot the polynomial curve
    plot(t, f);
end

% Add legend to the plot
legend('data points','1st degree','2nd degree','3rd degree');

% Release the hold on the plot
hold off;
```

- (a) Plot the data points:



- (b) Justification: comparison of three polynomial:



The choice of polynomial degree often depends on several factors including the complexity of the data, the presence of underlying trends, and the desire to avoid overfitting. For this data set, without visualizing the points it's difficult to definitively select the degree of the polynomial. However, we can make the following general observations:

- **First Degree (Linear)**: This is the simplest model. If the data points roughly align along a straight line, a first-degree polynomial (a linear model) would be the best choice. It is less likely to overfit the data and is computationally efficient. However, if the data shows curvature, a linear model would not capture this, leading to higher residuals.

- **Second Degree (Quadratic)**: A second-degree polynomial introduces curvature to the model. This might be suitable if the data points suggest a parabolic trend. It can model simple increases or decreases in rates of change but might still underfit complex data.
- **Third Degree (Cubic)**: A third-degree polynomial can model more complex relationships, including inflection points. If the data has one or more bends or curves, a cubic polynomial may provide a good balance between fit and complexity.

- (c) Fit data with chosen polynomial (3_{rd}):

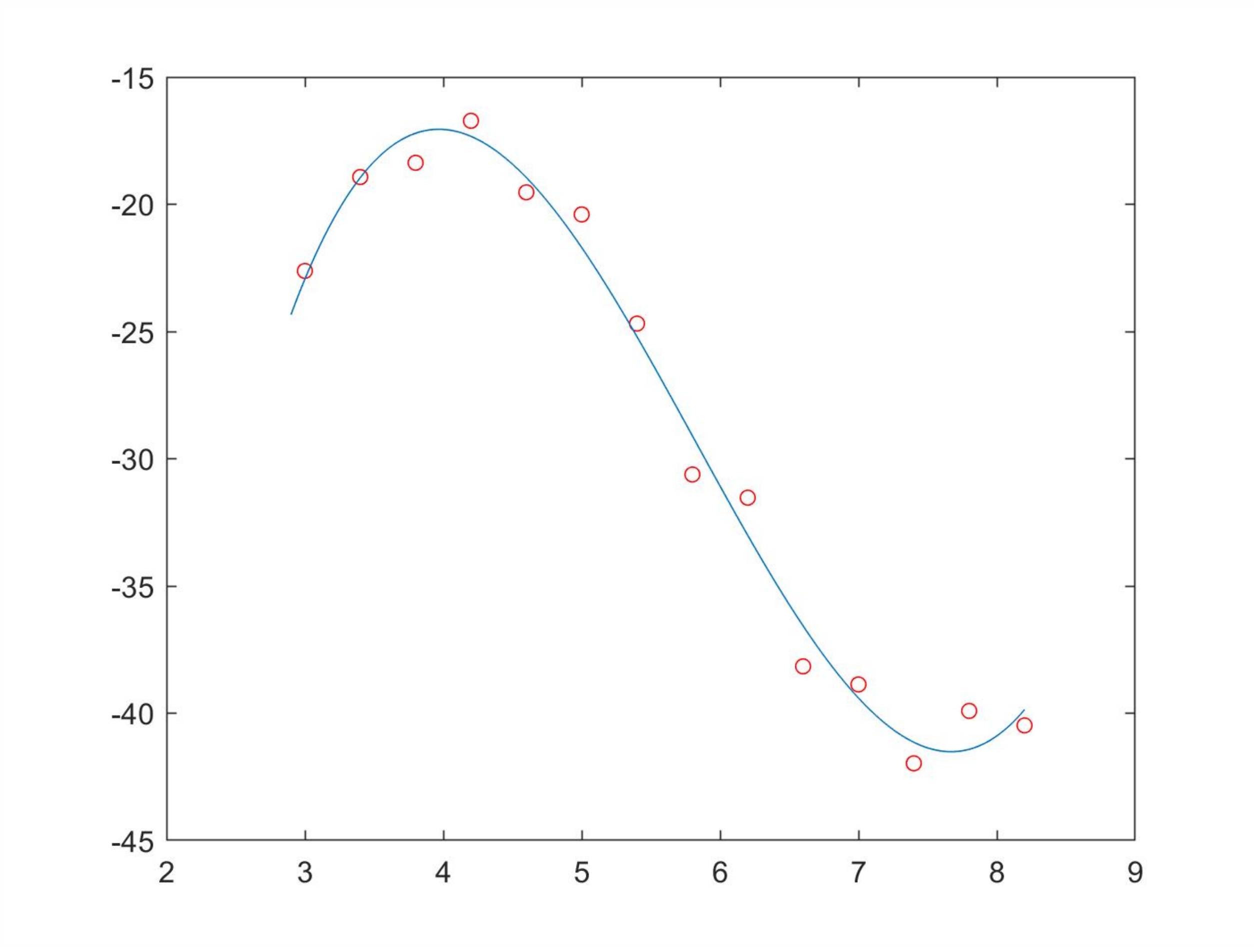
Using `polyfit` in MATLAB, the coefficients of the chosen polynomial can be calculated. The polynomial can then be used to predict new values with `polyval`. The fit can be evaluated by calculating the residuals and the mean squared error (MSE).

MATLAB output gives the coefficients for the curve:

$0.9645, \quad -16.8355, \;\; 88.0445, \quad -161.5971$

Therefore, the function of the fitting curve is $f(x) = 0.9645x^3 - 16.8355x^2 + 88.0445x - 161.5971$

- (d) One plot showing y_i vs x_i and $\hat{y}$ vs x using a x range from 2.9 to 8.3:



- **Discussion**

After fitting polynomials of different degrees, the discussion should focus on which model best captures the trend of the data without overfitting. Overfitting occurs when the model is too closely tailored to the specific dataset and may not generalize well to new data.

- If a first-degree polynomial was chosen, the discussion might revolve around the simplicity and robustness of the model against overfitting, at the cost of potentially higher bias if the data is not linear.
- A second or third-degree polynomial might fit the data with lower bias, but there is a risk of overfitting, especially if the higher-degree polynomial starts to follow the noise in the data rather than the true underlying trend.

In conclusion, without seeing the data, it's challenging to choose the best polynomial degree. However, once the polynomial is fitted, the residuals and MSE provide quantitative measures of fit, and the plotted curves offer a qualitative assessment. The preferred model balances a low MSE with a curve that makes sense for the data distribution and the context of the problem. It's often advisable to try multiple models and compare their performance based on these criteria.