به نام خدا

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

شبیه سازی و مدلسازی

گزارش تمرین شماره 1

[دینا مقیمی]

[40009473]

استاد : آقای دکتر مهدی علیاری

بهمن 1402

فهرست مطالب

بخش۱: سوالات تحلیلی

سوال اول

$$\text{Input:}\quad u = \begin{bmatrix}1\\2\\3\end{bmatrix} \qquad \text{Output:}\quad y = \begin{bmatrix}4\\5\\7\end{bmatrix}$$

$$\text{Design matrix:}\quad U = \begin{bmatrix}1 & 1\\1 & 2\\1 & 3\end{bmatrix}$$

$$\theta = (U^T U)^{-1} U^T y = \left(\begin{bmatrix}1 & 1 & 1\\1 & 2 & 3\end{bmatrix} \begin{bmatrix}1 & 1\\1 & 2\\1 & 3\end{bmatrix} \right)^{-1} \begin{bmatrix}1 & 1 & 1\\1 & 2 & 3\end{bmatrix} \begin{bmatrix}4\\5\\7\end{bmatrix}$$

$$= \begin{bmatrix}3 & 6\\6 & 14\end{bmatrix}^{-1} \begin{bmatrix}16\\35\end{bmatrix} = \frac{1}{6} \begin{bmatrix}14 & -6\\-6 & 3\end{bmatrix} \begin{bmatrix}16\\35\end{bmatrix} = \begin{bmatrix}\frac{14}{6}\\ \frac{9}{6}\end{bmatrix} = \begin{bmatrix}2.34\\1.5\end{bmatrix}$$

$$\theta_1 = 2.34\ ,\ \ \theta_2 = 1.5$$

```
% data points
x = [1; 2 ;3]; % input
y = [4; 5 ;7]; % output

X = [ones(size(x)),x]; %  design matrix

% method 1
coefficients = X \ y;
m = coefficients(2);
b = coefficients(1);
% method 2
theta = (inv(X'*X))*X'*y;
theta_1 = theta(1);
theta_2 = theta(2);
fprintf('theta_{1} = %.2f\n',theta_1)
fprintf('theta_{2} = %.2f\n\n',theta_2)
```

code 1:

1. **Data Points and Design Matrix**:
   - I’ve defined two vectors, x and y, representing input and output data points.
   - The design matrix X is created by adding a column of ones to x.
2. **Method 1: Solving Using Backslash Operator (\)**:
   - You’ve used the backslash operator to solve the linear system X * coefficients = y.
   - The coefficients m and b represent the slope and intercept of the regression line, respectively.
3. **Method 2: Using Matrix Inversion**:
   - Alternatively, you’ve computed the coefficients using matrix inversion.
   - theta_1 and theta_2 correspond to the intercept and slope, respectively.

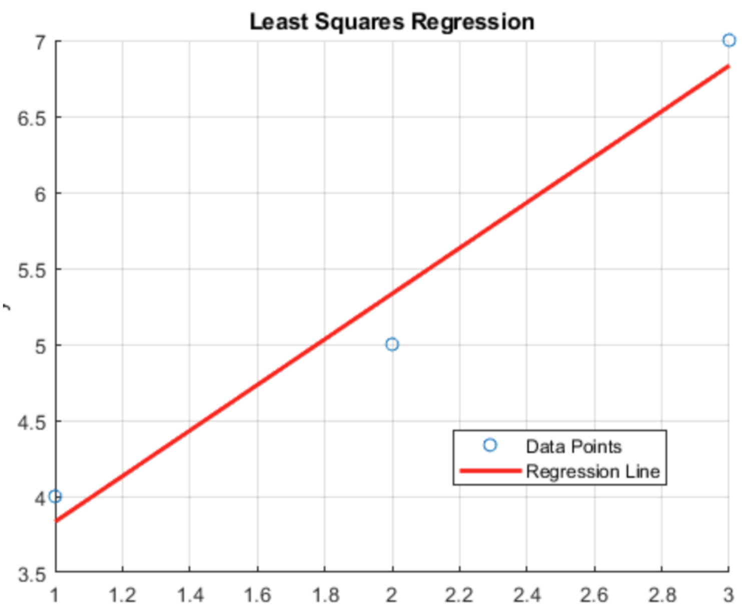
Predict y for a specific x (e.g., x = 4)

```
x_new = 4 ;

y_predicted = m * x_new + b;

fprintf('Predicted y for x = %.2f is y = %.2f\n', x_new, y_predicted);

% Plot the data points and the regression line
figure;
scatter(x, y, 'o', 'DisplayName', 'Data Points');
hold on;
x_fit = linspace(min(x), max(x), 100);
y_fit = m * x_fit + b;
plot(x_fit, y_fit, 'r-', 'LineWidth', 2, 'DisplayName', 'Regression Line');
xlabel('x');
ylabel('y');
title('Least Squares Regression');
legend('Location', 'best');
grid on;
```

Predicted y for x = 4.00 is y = 8.33

Code 2: **Prediction for a Specific** x:

- I've predicted the value of y for a new x (in this case, x_new = 4).
- The predicted y value is calculated as y_predicted = m * x_new + b.



1 Figure

Plotting the Regression Line:

I've visualized the data points and the regression line.

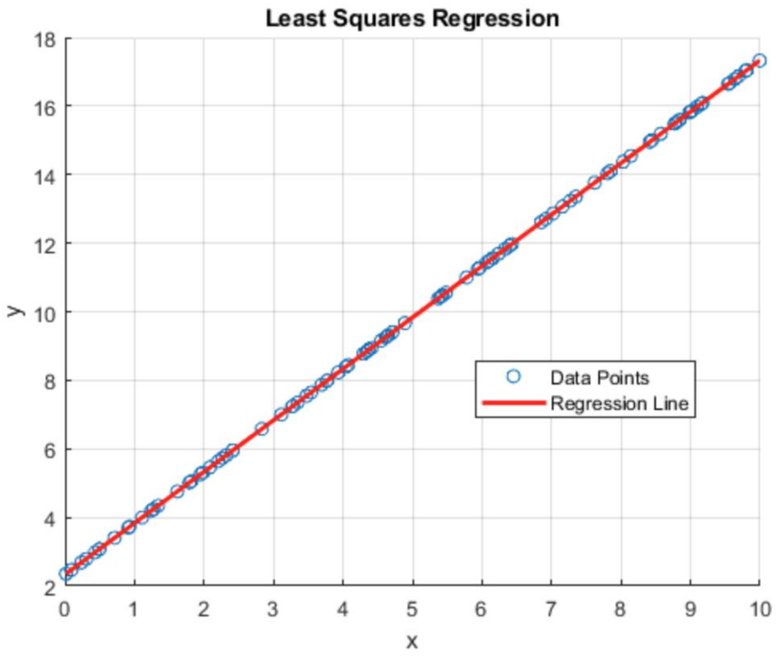
The red line represents the least squares regression fit.

Predict y-values using the model

```
mdl = fitlm(x, y);

% Predict y-values using the model
x_new = 10 * rand(1,100)'; % New x-values for prediction
y_pred = predict(mdl, x_new);

% Plot
figure()
scatter(x_new,y_pred,'DisplayName', 'Data Points');
hold on;
plot(x_new, y_pred, 'r-', 'LineWidth', 2, 'DisplayName', 'Regression Line');
xlabel('x');
ylabel('y');
title('Least Squares Regression');
legend('Location', 'best');
grid on;
```

code 3:

- I’ve also used the fitlm function to create a linear regression model (mdl).
- Next, I’ve predicted y values for new x values.



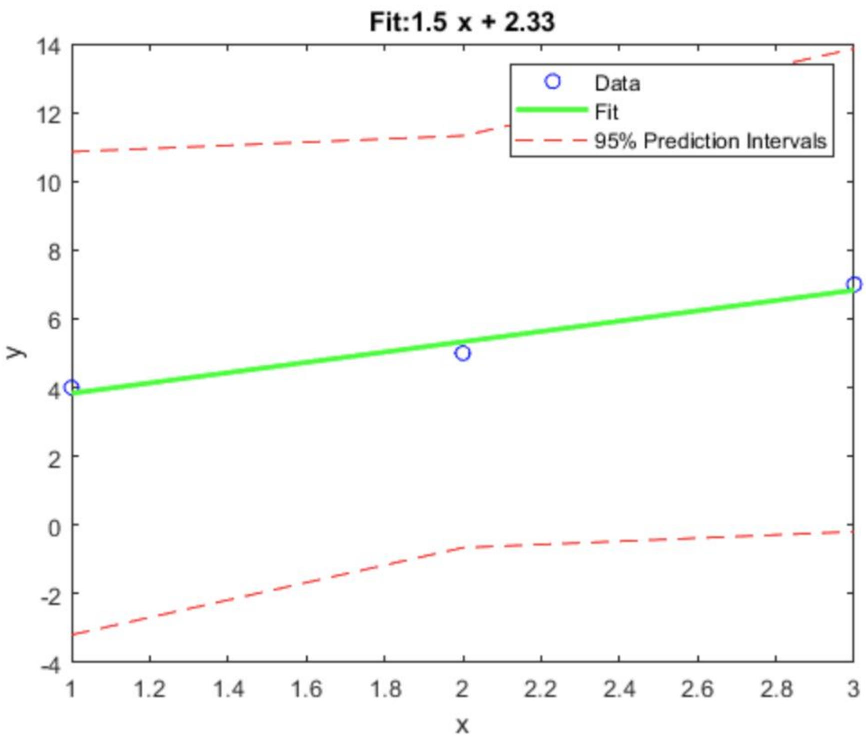
2 Figure

95% Prediction Intervals

```
degree = 1; % Degree of the fit
[p,S] = polyfit(x,y,degree);

alpha = 0.05; % Significance level
[yfit,delta] = polyconf(p,x,S,'alpha',alpha);

figure()
plot(x,y,'bo')
hold on
plot(x,yfit,'g-','LineWidth',2)
plot(x,yfit-delta,'r--',x,yfit+delta,'r--')
legend('Data','Fit','95% Prediction Intervals')
xlabel('x');
ylabel('y');
title(['Fit:',texlabel(polystr(round(p,2)))])
hold off
```

code 4 : Confidence band around linear least-squares line

I've found the the band by using p 'polytool' and defined a polystr function



3 Figure

بخش۲: سوالات شبیه سازی

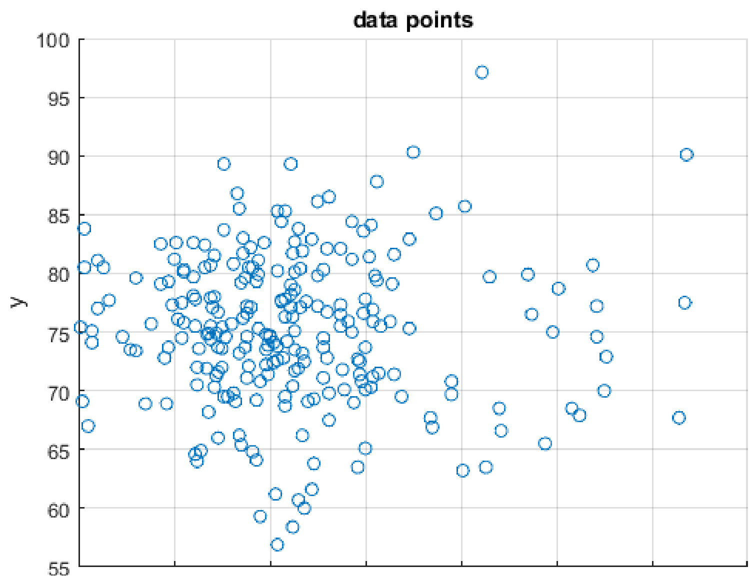
سوال اول

import data

```
A=readmatrix('batch-yield-and-purity.csv');

x=A(:,2);
y=A(:,1);
X=[ones(size(x)),x];

figure()
scatter(x,y)
title('data points')
xlabel('x');
ylabel('y');
grid on
hold on
```

Data Import and Visualization:

- I've imported data from a CSV file named "batch-yield-and-purity.csv."
- The data consists of two columns: x (input) and y (output).
- I've plotted the data points using a scatter plot.



4 **Figure**

```
% without command
Theta_ncmd1=(inv(X'*X))*X'*y

Theta_ncmd2=pinv(X)*y

% with command
Theta_cmd=lsqr(X,y)

% curve fitting
ytilda = X*Theta_cmd ;
plot(x,ytilda,'r',LineWidth=3)
xlabel('x');
ylabel('y');
title('curve fitting')

% error
error = y - ytilda;
J = error' * error;
```

```
Theta_ncmd1 =
   76.7595
   -0.0251

Theta_ncmd2 =
   76.7595
   -0.0251

Theta_cmd =
   76.7595
   -0.0251
```

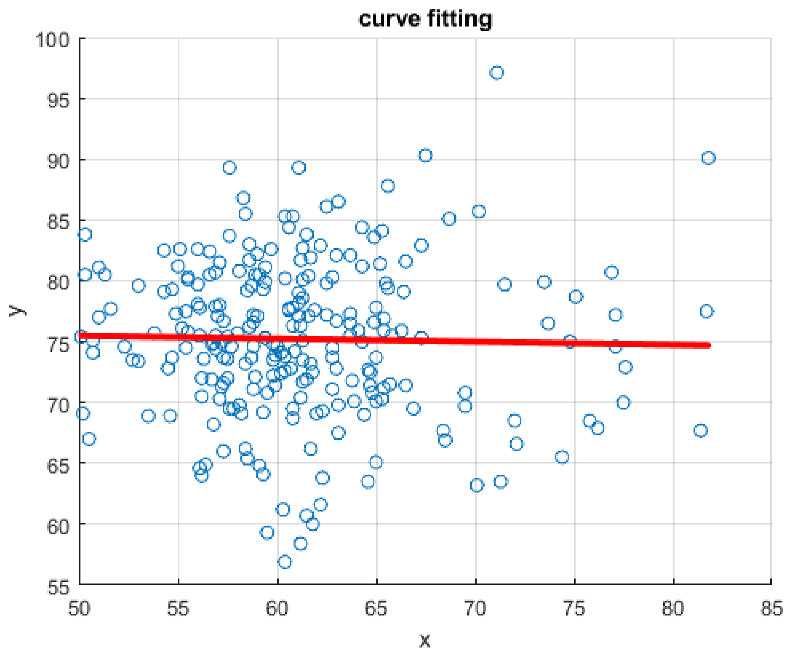
Estimating Coefficients:

I've calculated the coefficients using different methods:

- **Without Commands**: Theta_ncmd1 and Theta_ncmd2 represent the coefficients obtained without specific MATLAB commands.
- **With Command**: Theta_cmd represents the coefficients obtained using the lsqr command.

Error Calculation:

- You've computed the error between the actual y values and the predicted y values. The objective function J quantifies the error



5 Figure

Curve Fitting:

I’ve fitted a curve to the data using the estimated coefficients.

The red line represents the fitted curve.

BLUE

```
theta = X \ y;

theta_0 = theta(1);

theta_1 = theta(2)

% Display the estimated linear model
fprintf('Estimated linear model: y = %.4f + %.4fx\n', theta_0, theta_1);

% Plot the data points and the fitted line
figure;
scatter(x, y, 'o', 'DisplayName', 'Data Points');
hold on;
x_fit = linspace(min(x), max(x), 100);
y_fit = theta_0 + theta_1 * x_fit;
plot(x_fit, y_fit, 'r-', 'LineWidth',2, 'DisplayName', 'Fitted Line');
xlabel('x');
ylabel('y');
title('Best Linear Unbiased Estimation (BLUE)');
legend('Location', 'best');
grid on;
```

```
theta_1 = -0.0251

Estimated linear model: y = 76.7595 + -0.0251x
```

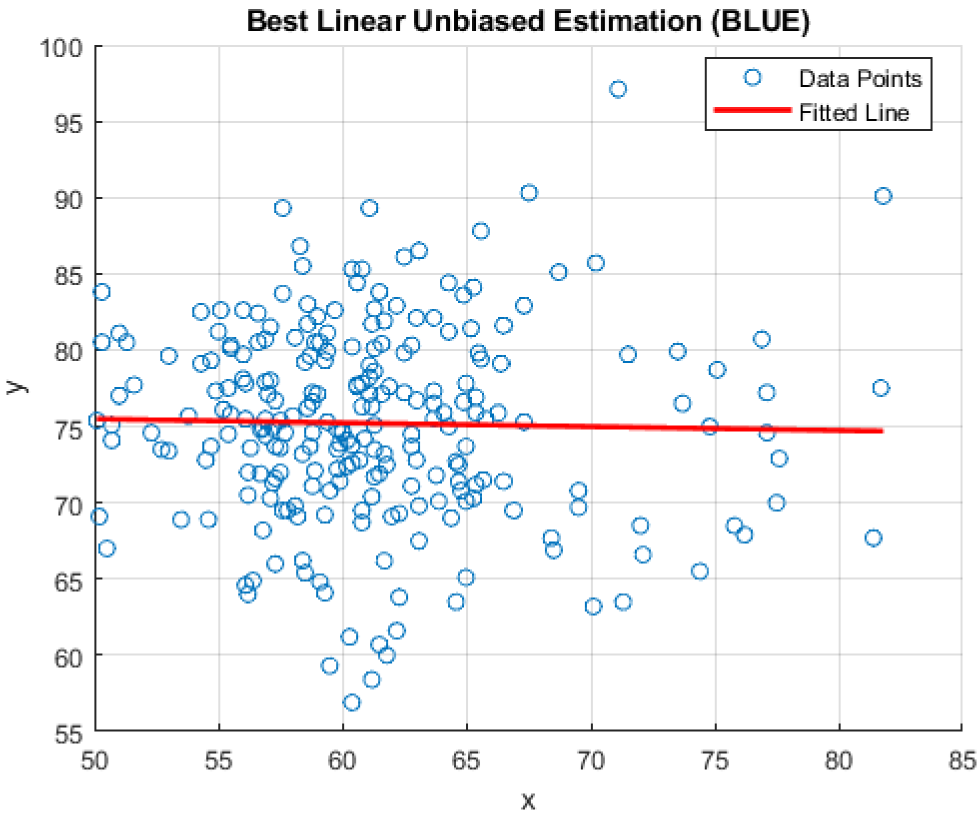


Figure 6.

Best Linear Unbiased Estimation (BLUE):

- You've also estimated the linear model using the backslash operator (\).
- The estimated model is given by y = theta_0 + theta_1 * x.

Plotting the Fitted Line:

- The scatter plot shows the data points, and the red line represents the fitted regression line.

setup

```
A=importdata("pHdata.dat");
%data Columns
timeSteps=A(:,1);
u1=A(:,2);  %input
u2=A(:,3);  %input
y=A(:,4);   %output

figure()
scatter(u1,y)
xlabel('u1')
ylabel('y')
grid on
figure()
scatter(u2,y)
xlabel('u2')
ylabel('y')
grid on
figure()
scatter3(u1,u2,y)
xlabel('u1')
ylabel('u2')
zlabel('y')
% normalizing data
u1 = (u1 - min(u1)) / (max(u1) - min(u1));
u2 = (u2 - min(u2)) / (max(u2) - min(u2));
y = (y - min(y)) / (max(y) - min(y));

U=[ones(size(y)),u1,u2];
```

Data Import and Normalization:

- The code starts by importing data from a file named “pHdata.dat.”
- The data includes columns for timeSteps, u1 (input 1), u2 (input 2), and y (output).
- The u1, u2, and y values are normalized to the range [0, 1]

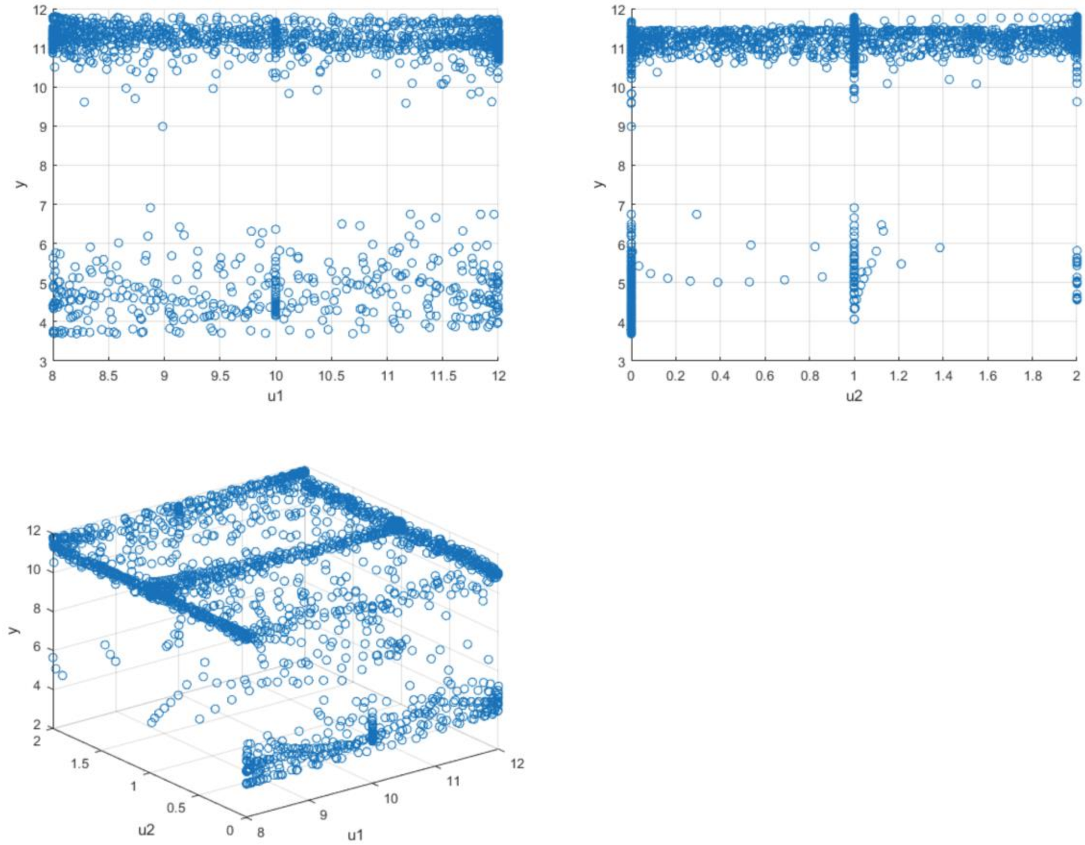


Figure1,2,3

Least square

```
x = [u1 , u2];

X=[ones(size(y))];

theta=zeros(9,4);

for n=1:4
    X=[X,x.^n];

    for i=1:2*n+1
        LS=lsqr(X,y);
        theta(i,n)=(LS(i,1));
    end
    intercept=theta(1,n);
    slope11=theta(2,n);
    slope21=theta(3,n);
    slope12=theta(4,n);
    slope22=theta(5,n);
    slope13=theta(6,n);
    slope23=theta(7,n);
    slope14=theta(8,n);
    slope24=theta(9,n);
    yFit = intercept + slope11*u1 + slope21*u2 + slope12*u1.^2 + slope22*u2.^2 ...
        + slope13*u1.^3 + slope23*u2.^3 + slope14*u1.^4 + slope24*u2.^4;
    e_LS = y - yFit;
    %plot
    figure(4)
    subplot(2,2,n)
    scatter3(u1,u2,yFit)
    hold on
    scatter3(u1,u2,y,'filled')
    title( num2str(n),'order')
    xlabel('u1')
    ylabel('u2')
    zlabel('y')
    legend('Fitted', 'Data')

    figure(5)
    subplot(2,2,n)
    plot(e_LS)
    title( num2str(n),'order')
    xlabel('Sample Index')
    ylabel('Residual')
    grid on

end
```

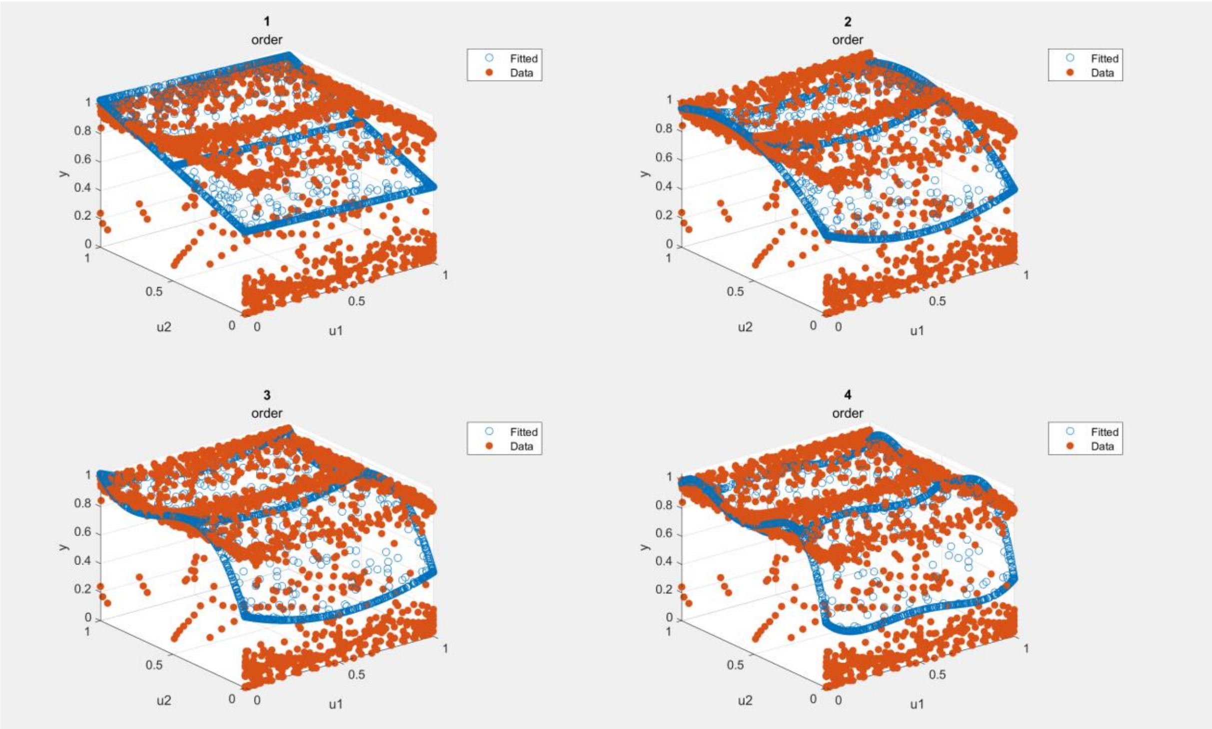


Figure 4

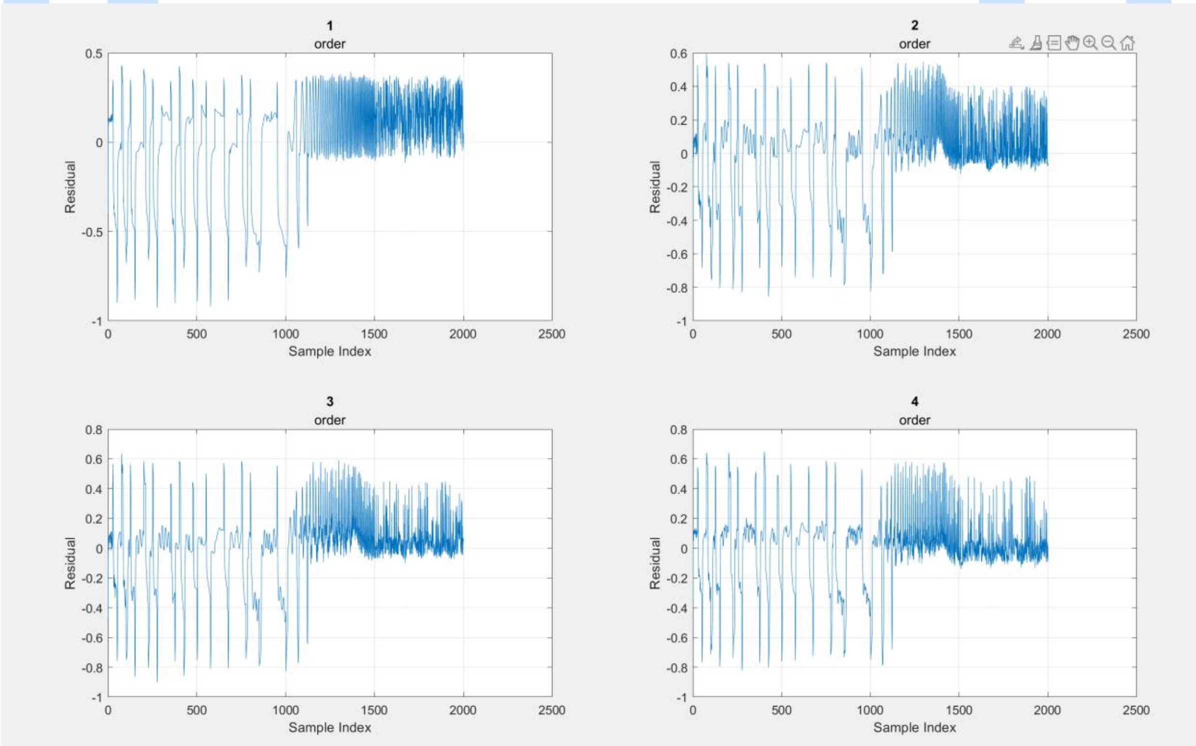


Figure 5

Least Squares Polynomial Fitting:

- The code performs polynomial fitting using least squares.
- For each polynomial order (from 1 to 4), it constructs a design matrix X with powers of u1 and u2.
- The coefficients (theta) are estimated using the lsqr function.
- The fitted values (yFit) are calculated based on the polynomial model.
- Residuals (e_LS) are obtained by subtracting the fitted values from the actual y values.
- The results are plotted in Figure 4 (scatter plot of fitted vs. actual y) and Figure 5 (residuals).

forgetting factor

```
n=0;

for lambda = 0.09:0.05:0.99
    n=n+1;

    theta = zeros(size(U,2),1);
    P = eye(size(U,2)) / lambda;

    for i=1:length(y)
        u_i = U(i,:)';
        y_predict = u_i'*theta;
        e = y(i) - y_predict;
        K = P*u_i/(lambda + u_i'*P*u_i);
        theta = theta + K*e;
        P = (P - K*u_i'*P)/lambda;
    end

% error
intercept=theta(1);
slope_u1=theta(2);
slope_u2=theta(3);
yFit_ff = intercept + slope_u1*u1 + slope_u2*u2;
e_ff = y - yFit_ff;
figure(6)
subplot(10,2,n)
plot(e_ff)
grid on
title('\lambda=',num2str(lambda))
end
```

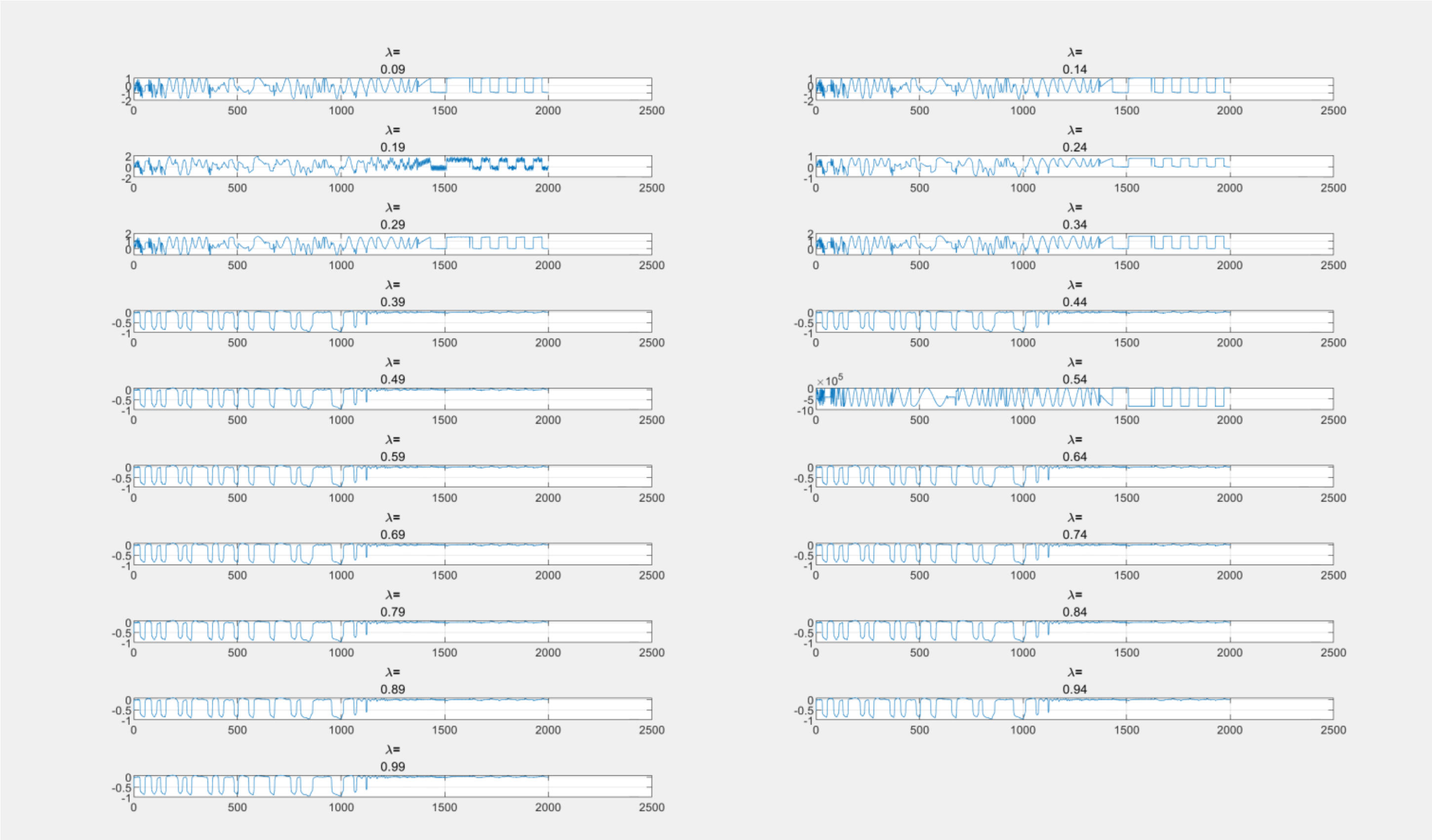


Figure 6

Forgetting Factor Recursive Estimation:

- The code then performs recursive estimation with a forgetting factor (lambda).
- The parameters (theta) are updated iteratively using the recursive least squares (RLS) algorithm.
- The fitted values with the forgetting factor (yFit_ff) are calculated.
- Residuals (e_ff) are obtained by subtracting the fitted values from the actual y values.
- The results are plotted in Figure 6 (residuals for different forgetting factors).

sliding window

```
window_size=1000;

step_size=1;

num_points=length(y);

num_windows = floor((num_points - window_size) / step_size) + 1;

intercept_sw = zeros(num_windows,1);

slope1_sw = zeros(num_windows,1);

slope2_sw = zeros(num_windows,1);

e_sw = zeros(num_windows,window_size);

for i=1:num_windows
    Start = (i-1) * step_size + 1;
    End = Start + window_size - 1;
    u1_inWindow = u1(Start:End);
    u2_inWindow = u2(Start:End);
    y_inWindow = y(Start:End);
   U_window = [ones(size(y_inWindow)),u1_inWindow,u2_inWindow];
    theta_window = pinv(U_window) * y_inWindow;
    intercept_sw(i)=theta_window(1);
    slope1_sw(i)=theta_window(2);
    slope2_sw(i)=theta_window(3);
    yFit_window = U_window * theta_window;
    e_sw(i,:) = y_inWindow - yFit_window ;
end
%plot
figure(7)
for i=1:num_windows
    Start = (i-1) * step_size + 1;
    End = (i-1) * step_size + window_size ;
    u1_inWindow = u1(Start:End);
    u2_inWindow = u2(Start:End);
   yFit_sw = intercept_sw(i) + slope1_sw(i)*u1_inWindow + slope2_sw(i)*u2_inWindow;
    scatter3(u1,u2,y,'filled','k')
    hold on
    plot3(u1_inWindow,u2_inWindow,yFit_sw)
end
figure(8)
plot(e_sw)
grid on:
```

1. **Sliding Window Parameters**:
   - window_size is set to 1000 data points.
   - step_size is set to 1, meaning the window moves one data point at a time.
2. **Window Initialization**:
   - The total number of data points (num_points) is determined from the length of the y vector.
   - The number of windows (num_windows) is calculated based on the window size and step size.
3. **Initialization of Variables**:

   Arrays are initialized to store results for each window:

   - intercept_sw: Intercept of the linear regression model for each window.
   - slope1_sw: Slope coefficient for u1 in the linear regression model for each window.
   - slope2_sw: Slope coefficient for u2 in the linear regression model for each window.
   - e_sw: Residuals (errors) for each window.
4. **Sliding Window Loop**:

   For each window:

   - Determine the start and end indices of the current window.
   - Extract the corresponding data points (u1_inWindow, u2_inWindow, y_inWindow).
   - Create a design matrix U_window by adding a column of ones to u1_inWindow and u2_inWindow.
   - Calculate the coefficients (theta_window) using the pseudo-inverse (pinv) method.
   - Store the intercept and slopes in the respective arrays.
   - Compute the fitted values (yFit_window) using the estimated coefficients.
   - Calculate the residuals (e_sw) by subtracting the fitted values from the actual y values.

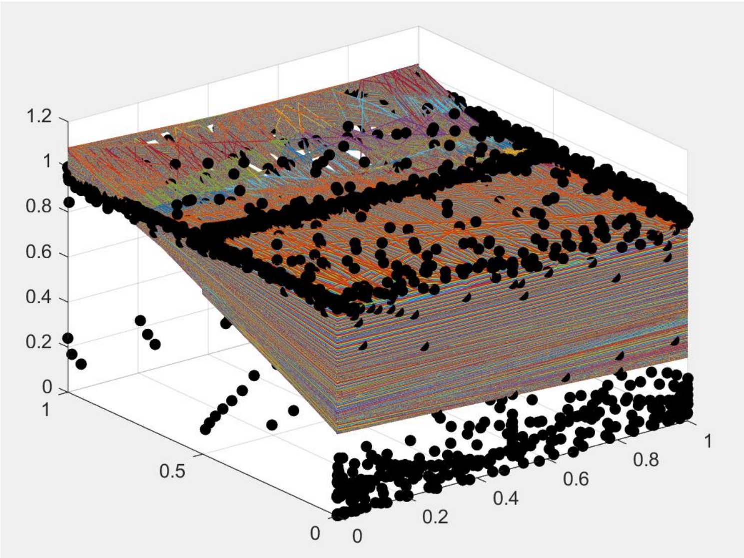


Figure 7

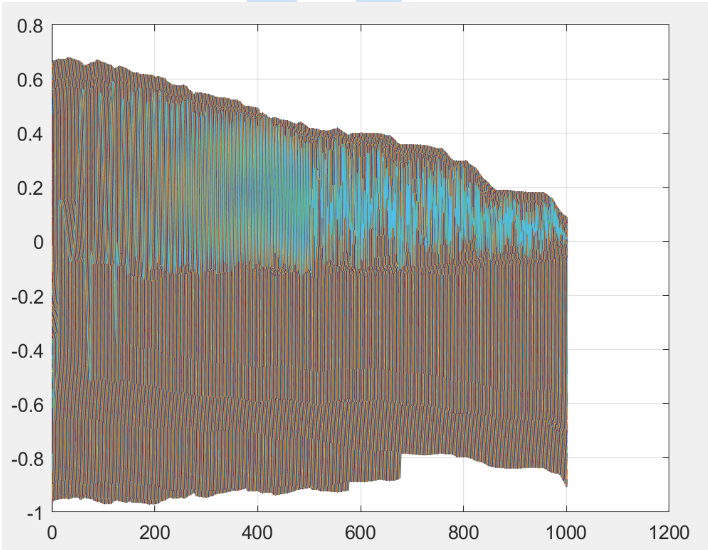


Figure 8

5. **Plotting Results**:
   - Figure 7:
     - For each window, scatter plot the actual data points (u1, u2, y) in black.
     - Overlay the regression line (yFit_sw) for the current window.
   - Figure 8:
     - Plot the residuals (e_sw) for all windows.
     - The grid is turned on for clarity.
6. **Interpretation**:
   - The sliding window analysis allows you to perform local linear regression on subsets of the data.
   - It estimates regression coefficients within each window and visualizes the results.

سوال سوم

RLS sliding window

```
window_size=1000;  step_size=1;

num_points=length(y);

intercept_swR = zeros(num_windows,1);

slope1_swR = zeros(num_windows,1);

slope2_swR = zeros(num_windows,1);

e_swR = zeros(num_windows,window_size);

for i = 1 : num_points - window_size + 1

   Start = (i-1) * step_size + 1;
   End = Start + window_size - 1;
   u1_inWindow = u1(Start:End);
   u2_inWindow = u2(Start:End);
   y_inWindow = y(Start:End);
   P = eye(3);
   theta_window = zeros (3,1);
   window_errors = zeros(1,window_size);
   for j =1:window_size
      u=[1; u1_inWindow(j); u2_inWindow(j)];
      e = y_inWindow(j) - u' * theta_window;
      K = (P*u) / (1 + u'*P*u);
      theta_window= theta_window +  K*e ;
      P = P - K*u'*P;
      window_errors(j) = e;
   end
   e_swR(i,:) = window_errors;
end
%plot
figure(9)
for i=1:num_windows
   Start = (i-1) * step_size + 1;
   End = (i-1) * step_size + window_size ;
   u1_Window = u1(Start:End);
   u2_Window = u2(Start:End);
   yFit_swR = intercept_swR(i) + slope1_swR(i)*u1_Window + slope2_swR(i)*u2_Window;
   plot3(u1_Window,u2_Window,yFit_swR)
   hold on
end
scatter3(u1,u2,y,'filled','b')
figure(10)
plot(e_swR)
grid on
```

1. **Sliding Window Parameters**:
   - window_size is set to 1000 data points.
   - step_size is set to 1, meaning the window moves one data point at a time.
2. **Initialization of Variables**:
   - num_points represents the total number of data points in the vector y.
   - Arrays are initialized to store results for each window:
     - intercept_swR: Intercept of the linear regression model for each window.
     - slope1_swR: Slope coefficient for u1 in the linear regression model for each window.
     - slope2_swR: Slope coefficient for u2 in the linear regression model for each window.
     - e_swR: Residuals (errors) for each window.
3. **Sliding Window Loop**:

   For each window:

   - Determine the start and end indices of the current window.
   - Extract the corresponding data points (u1_inWindow, u2_inWindow, y_inWindow).
   - Initialize the covariance matrix P and the parameter vector theta_window.
   - Iterate over each data point within the window:
     - Compute the Kalman gain K.
     - Update the parameter vector theta_window.
     - Update the covariance matrix P.
     - Calculate the error (e) and store it in window_errors.
   - Store the window errors in e_swR.
4. **Interpretation**:
   - The sliding window analysis with recursive estimation allows you to adaptively estimate regression coefficients within each window.
   - The Kalman gain (K) adjusts the parameter estimates based on the observed errors.
   - The results are useful for tracking changes in the regression model over time.

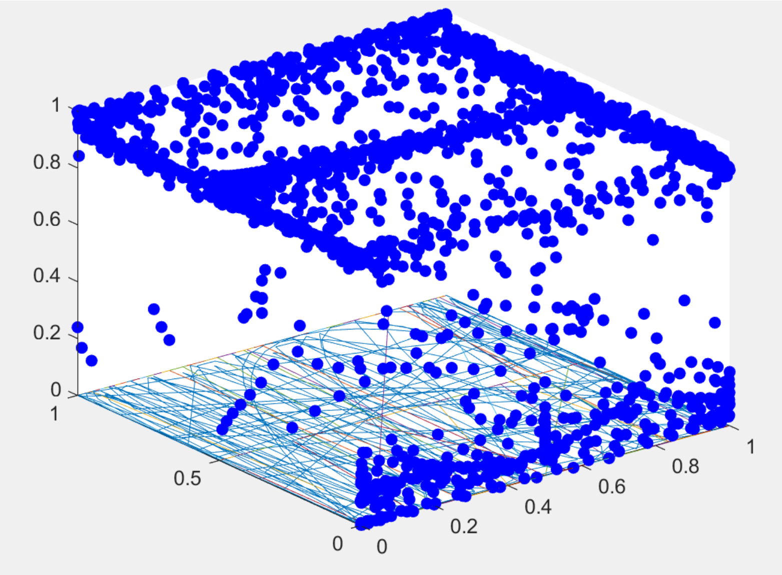


Figure 9

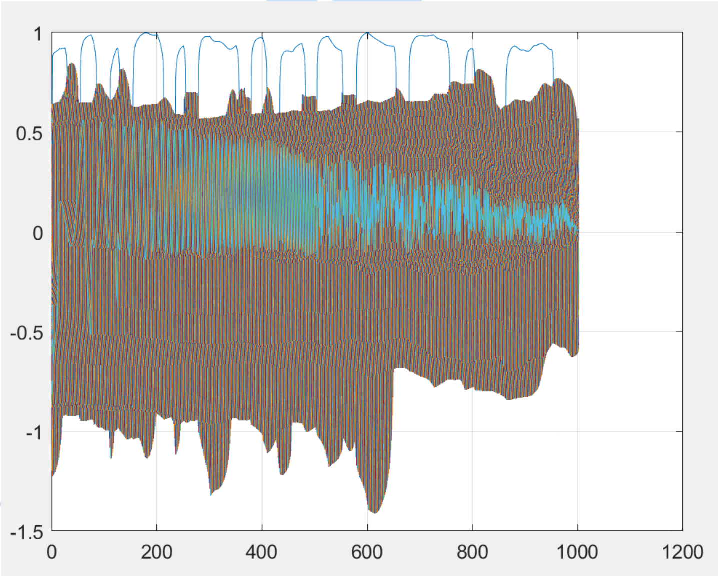


Figure 10

5. **Plotting Results**:
   - Figure 9:
     - For each window, compute the fitted values (yFit_swR) using the estimated coefficients.
     - Plot the regression line (yFit_swR) for the current window.
     - Overlay the actual data points (u1, u2, y) in blue.
   - Figure 10:
     - Plot the residuals (e_swR) for all windows.
     - The grid is turned on for clarity.

مراجع

[1] https://www.mathworks.com/matlabcentral/answers/392636-confidence-band-around-linear-least-squares-line