3rd Homework

Theofanis Nitsos - p3352325

Exercise 1

The correct answer is 2. 3 would be also correct assuming we know the true mean temperature at noon.

Exercise 2

The correct answer is 3, 4

Exercise 3

The correct answer is 1, 3

Exercise 4

The correct answer is 2

Exercise 5

The correct answer is 4

Exercise 6

From the slides we know that:
$MSE = E\left[(\hat{\theta} - \theta_0)^2\right] = E\left[(\hat{\theta} - E(\hat{\theta}))^2\right] + \left(E(\hat{\theta}) - \theta_0^2\right)$

Thus we can calculate

- Case (A) $MSE = a$, unbiased
- Case (B) $MSE = a + \frac{a}{3} = \frac{4a}{3}$, biased
- Case (C) $MSE = 3a$, unbiased
- Case (D) $MSE = 3a + \frac{a}{3} = \frac{10a}{3}$, biased
- Case (E) $MSE = a + 3a = 4a$, biased

Exercise 7

1 and 3 (what 3 is concerned the estimators differ because they are applied to different datasets)

Exercise 8

1 and 4

Exercise 9

2

Exercise 10

2

Exercise 11

4

Exercise 12

To minimise the Lagrangia function of the ridge regression problem we will calculate its derivative:

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial\left(\sum_{n=1}^{N}(y_n-\boldsymbol{\theta}^T\boldsymbol{x}_n)^2+\lambda\|\boldsymbol{\theta}\|^2\right)}{\partial \boldsymbol{\theta}} = \frac{\partial\left(\sum_{n=1}^{N}(y_n-\boldsymbol{\theta}^T\boldsymbol{x}_n)^2\right)}{\partial \boldsymbol{\theta}} + \frac{\partial(\lambda\|\boldsymbol{\theta}\|^2)}{\partial \boldsymbol{\theta}} = -2\sum_{n=1}^{N}\big((y_n - \boldsymbol{\theta}^T\boldsymbol{x}$$

To find a solution to this equation we will equate it to 0

$$\Rightarrow -\sum_{n=1}^{N}\big((y_n - \boldsymbol{\theta}^T\boldsymbol{x}_n)(-\boldsymbol{x}_n)\big) + \lambda\boldsymbol{\theta} = 0 \Rightarrow \sum_{n=1}^{N}\left(y_n\boldsymbol{x}_n - (\boldsymbol{\theta}^T\boldsymbol{x}_n)\boldsymbol{x}_n\right) + \lambda\boldsymbol{\theta} = 0 \Rightarrow$$

it stands that $(\boldsymbol{\theta}^T)\boldsymbol{x} = (\boldsymbol{x}\boldsymbol{x}^T)\boldsymbol{\theta}$ Thus:
$\sum_{n=1}^{N}\left(y_n\boldsymbol{x}_n\right) = \left(\sum_{n=1}^{N}\left(\boldsymbol{x}_n\boldsymbol{x}_n^T\right) + \lambda I\right)\hat{\boldsymbol{\theta}}$

(b)

From the 1st Homework we proved that:
$X^TX = \sum_{n=1}^{N}(\boldsymbol{x}_n\boldsymbol{x}_n^T)$ and
$X^T\boldsymbol{y} = \sum_{n=1}^{N}(y_n\boldsymbol{x}_n)$

From the 1st Homework:

$X^T X = \sum_{n=1}^N (\boldsymbol{x}_n \boldsymbol{x}_n^T)$

$$X^T = \begin{bmatrix} x_{11} & x_{21} & \dots & x_{N1} \\ x_{12} & x_{22} & \dots & x_{N2} \\ \vdots & \vdots & \ddots & \\ x_{1l} & x_{2l} & \dots & x_{Nl} \end{bmatrix} = [\, \boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \dots \quad \boldsymbol{x}_N \,]$$

$$X^T X = [\, \boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \dots \quad \boldsymbol{x}_N \,] \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix} = \boldsymbol{x}_1 \boldsymbol{x}_1^T + \boldsymbol{x}_2 \boldsymbol{x}_2^T + \dots + \boldsymbol{x}_N \boldsymbol{x}_N^T = \sum_{n=1}^N (\boldsymbol{x}_n \boldsymbol{x}_n^T)$$

and

$X^T \boldsymbol{y} = [\, \boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \dots \quad \boldsymbol{x}_N \,] \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \boldsymbol{x}_1 y_1 + \boldsymbol{x}_2 + y_2 + \dots + \boldsymbol{x}_N y_n =$ (since y_n is a scalar we can use the commutative property)

$= y_1 \boldsymbol{x}_1 + y_2 \boldsymbol{x}_2 + \dots + y_N \boldsymbol{x}_N = \sum_{n=1}^N (y_n \boldsymbol{x}_n)$

Thus using the equation from (a)

$X^T \boldsymbol{y} = \left(X^T X + \lambda I\right) \hat{\boldsymbol{\theta}} \Rightarrow \left(X^T X + \lambda I\right)^{-1} X^T \boldsymbol{y} = \left(X^T X + \lambda I\right)^{-1} \left(X^T X + \lambda I\right) \hat{\boldsymbol{\theta}} \Rightarrow$

Exercise 13

(a)

$E(\hat{\theta}_{MVU}) = \theta_0$

(b)

$E(\hat{\theta}_{MVU}) = \theta_0 \Rightarrow (a+1)E(\hat{\theta}_{MVU}) = (a+1)\theta_0 \Rightarrow E\left((a+1)\hat{\theta}_{MVU}\right) = (a+1)\theta_0 \neq$ for $a \neq 0$

(c)

$MSE(\hat{\theta}_{MVU}) = E\left[(\hat{\theta}_{MVU} - E(\hat{\theta}_{MVU}))^2\right] + 0 = Var\left[\hat{\theta}_{MVU}\right] = Cov\left(\hat{\theta}_{MVU}, \hat{\theta}_{MVU}\right)$

$= E[\hat{\theta}^2_{MVU}] - 2E[\hat{\theta}_{MVU}]E[\hat{\theta}_{MVU}] + E[\hat{\theta}_{MVU}]^2 = E[\hat{\theta}^2_{MVU}] - E[\hat{\theta}_{MVU}]^2 = E[\hat{\theta}^2_{MVU}]$ -

For the quantity $E[\hat{\theta}^2_{MVU}] - \theta_0^2$ to equal 0, the variance of the estimator equals the variance of the model. But that is highly unlikely because of:

- Finite Sample Variability: In a finite sample, the observed data exhibit variability due to random sampling. Even if you have an unbiased estimator, the observed values will differ from one sample to another. This inherent variability contributes to the variance of the estimator.
- Cramér-Rao Lower Bound (CRLB): The Cramér-Rao Inequality states that, for an unbiased estimator, the variance of the estimator is lower-bounded by the reciprocal of the Fisher Information. The CRLB provides a theoretical lower limit on the achievable variance for any unbiased estimator. However, this lower bound is not always attainable, and it does not guarantee zero variance.
- As the sample size increases, the variability due to random sampling tends to decrease, and the estimator becomes more precise. However, even in large samples, achieving zero variance is not practically feasible.

(d)

$MSE(\hat{\theta}_b) = E\left[(\hat{\theta}_b - E[\hat{\theta}_b])^2\right] + \left(E[\hat{\theta}_b] - \theta_0\right)^2 =$

using $\hat{\theta}_b = (1+a)\hat{\theta}_{MVU}$ and $E\left[\hat{\theta}_b\right] = (1+a)E[\hat{\theta}_{MVU}] = (1+a)\theta_0 \Rightarrow$

$E\left[(\hat{\theta}_b - (1+a)\theta_0)^2\right] + ((1+a)\theta_0 - \theta_0)^2 =$

$= E\left[\hat{\theta}_b^2 + ((1+a)\theta_0)^2 - 2(1+a)\hat{\theta}_{MVU}(1+a)\theta_0\right] + (a\theta_0)^2 =$

$= E\left[\hat{\theta}_b^2\right] + ((1+a)\theta_0)^2 - 2(1+a)\theta_0(1+a)\theta_0 + (a\theta_0)^2 =$

$= E\left[((1+a)\hat{\theta}_{MVU})^2\right] + ((1+a)\theta_0)^2 - 2(1+a)^2\theta_0^2 + (a\theta_0)^2 =$

$= (1+a)^2 E\left[\hat{\theta}^2_{MVU}\right] + \theta_0^2\left(a^2 - 1 - 2a - a^2\right) =$

$= (1+a)^2 E\left[\hat{\theta}^2_{MVU}\right] - \theta_0^2\,(1+2a)$

(e)

$MSE(\hat{\theta}_b) < MSE(\hat{\theta}_{MVU}) \Rightarrow$

$(1+a)^2 E\left[\hat{\theta}^2_{MVU}\right] - \theta_0^2\,(1+2a) < E[\hat{\theta}^2_{MVU}] - \theta_0^2 \Rightarrow$

$E\left[\hat{\theta}^2_{MVU}\right](1+2a+a^2) - \theta_0^2\,(1+2a) < E[\hat{\theta}^2_{MVU}] - \theta_0^2 \Rightarrow$

$a^2 E\left[\hat{\theta}^2_{MVU}\right] + 2a\left(E\left[\hat{\theta}^2_{MVU}\right] - \theta_0^2\right) < 0$

Solving the polynomial of a

$a\left(aE\left[\hat{\theta}^2_{MVU}\right] + 2\left(E\left[\hat{\theta}^2_{MVU}\right] - \theta_0^2\right)\right) = 0$

has 2 roots

$a = 0$ and $a = 2\frac{\theta_0^2 - E\left[\hat{\theta}_{MVU}^2\right]}{E\left[\hat{\theta}_{MVU}^2\right]} = 2\frac{\theta_0^2}{E[\hat{\theta}_{MVU}^2]} - 2$

Since the coefficient of a^2 is greater than 0 the parabola opens upwards, meaning the negative values of the polynomial lie between the 2 roots

We know that $MSE(\hat{\theta}_{MVU}) \geq 0 \Rightarrow$ from (c) $E[\hat{\theta}_{MVU}^2] - \theta_0^2 \geq 0 \Rightarrow \frac{\theta_0^2}{E[\hat{\theta}_{MVU}^2]} \leq 1$ thus

$a = 2\frac{\theta_0^2}{E[\hat{\theta}_{MVU}^2]} - 2 \leq 2 - 2 = 0$

meaning one of the roots is 0 and the other is less than 0 or the polynomial will be negative for:

$2\frac{\theta_0^2}{E[\hat{\theta}_{MVU}^2]} - 2 < a < 0$

(f)

From (e) we proved that

$0 \leq \frac{\theta_0^2}{E[\hat{\theta}_{MVU}^2]} \leq 1 \Rightarrow 0 \leq 2\frac{\theta_0^2}{E[\hat{\theta}_{MVU}^2]} \leq 2 \Rightarrow -2 \leq 2\frac{\theta_0^2}{E[\hat{\theta}_{MVU}^2]} - 2 \leq 0 \Rightarrow$ using the roots of the polynomial from (e)

$-2 < a < 0 \Rightarrow -1 < a + 1 < 1 \Rightarrow |a + 1| < 1$

multiplying by $|\hat{\theta}_{MVU}|$ which is by default a positive quantity

$|\hat{\theta}_{MVU}||a + 1| < |\hat{\theta}_{MVU}| \Rightarrow |\hat{\theta}_{MVU}(a + 1)| < |\hat{\theta}_{MVU}| \Rightarrow |\hat{\theta}_b| < |\hat{\theta}_{MVU}|$

(g)

$\frac{\partial\left((1+a)^2 E\left[\hat{\theta}_{MVU}^2\right] - \theta_0^2(1+2a)\right)}{\partial(a)} = 2(1 + a)E\left[\hat{\theta}_{MVU}^2\right] - 2\theta_0^2 = 2aE\left[\hat{\theta}_{MVU}^2\right] + 2E\left[\hat{\theta}_{MVU}^2\right] -$

Solving the above equation we get

$a^* E\left[\hat{\theta}_{MVU}^2\right] + E\left[\hat{\theta}_{MVU}^2\right] - \theta_0^2 = 0 \Rightarrow a^* = \frac{\theta_0^2}{E\left[\hat{\theta}_{MVU}^2\right]} - 1$

(h)

In practice we don't really know θ_0. If we did then there is no need to try and estimate/calculate it. We already know the model that produced (x_i, y_i).

Exercise 14

(a)

$\theta_0 = \frac{\sum_{n=1}^{N}(y_n)}{N}$

(b)

$E[y_n] = E[\theta_0 + \eta_n] = E[\theta_0] + E[\eta_n] = E[\theta_0] + 0 = E[\theta_0] = \theta_0$

(c)

$E[E[\bar{y}]] = E[E[\frac{1}{N}\sum_{n=1}^{N} y_n]] = E[\frac{1}{N}\sum_{n=1}^{N} E[y_n]] = E[\frac{1}{N}\sum_{n=1}^{N} \theta_0] = \theta_0$

(e)

$\sum_{n=1}^{N}(y_n \boldsymbol{x}_n) = \left(\sum_{n=1}^{N}\left(\boldsymbol{x}_n \boldsymbol{x}_n^T\right) + \lambda I\right) \acute{\boldsymbol{\theta}}$

since we have the 1-dim case this equation becomes:

$(N+\lambda)\acute{\theta} = \sum_{n=1}^{N}(y_n) \Rightarrow \acute{\theta} = \frac{\sum_{n=1}^{N}(y_n)}{N+\lambda}$

(f)

$\acute{\theta} = \frac{\sum_{n=1}^{N}(y_n)}{N+\lambda} = \frac{1}{N+\lambda}\frac{N}{N}\sum_{n=1}^{N}(y_n)$
$= \frac{N}{N+\lambda}\frac{\sum_{n=1}^{N}(y_n)}{N} = \frac{N}{N+\lambda}\acute{\theta}_{MVU}$

(g)

$E\left[\acute{\theta}\right] = E\left[\frac{N}{N+\lambda}\acute{\theta}_{MVU}\right] = \frac{N}{N+\lambda}E[\hat{\theta}_{MVU}] = \frac{N}{N+\lambda}\theta_0 \neq \theta_0$, for $N \neq 0$

(h)

For $\lambda \in R$ and $\lambda > 0$ it stands
$|\frac{N}{N+\lambda}| < 1 \Rightarrow |\acute{\theta}_{MVU}||\frac{N}{N+\lambda}| < |\acute{\theta}_{MVU}| \Rightarrow |\acute{\theta}_{MVU}\frac{N}{N+\lambda}| < |\acute{\theta}_{MVU}| \Rightarrow$
$|\acute{\theta}| < |\acute{\theta}_{MVU}|$

(i)

$(1+a) = \frac{N}{N+\lambda} \Rightarrow a = \frac{N}{N+\lambda} - 1 \Rightarrow a = -\frac{\lambda}{N+\lambda}$

From exercise 13 we know that $-2 < a < 0$, thus

$-2 < -\frac{\lambda}{N+\lambda} < 0 \Rightarrow \frac{\lambda}{N+\lambda} > 0$ and $\frac{\lambda}{N+\lambda} < 2$

Exercise 15

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np

# Set seed for reproducibility
np.random.seed(42)

# Number of data sets
d = 50

# Number of data points in each set
N = 30

# Standard deviation for Gaussian noise
sigma = 8  # sqrt(64)

# Initialize a list to store the data sets
data_sets = []

# Generate d data sets
for _ in range(d):
    # Generate random x values
    x_values = np.random.rand(N) * 10  # Generating random x values between

    # Calculate y' without noise using the formula y' = 2 * x
    y_prime_values = 2 * x_values

    # Add Gaussian noise to y' to get y
    noise = np.random.normal(0, sigma, N)
    y_values = y_prime_values + noise

    # Create the data set
    data_set = list(zip(y_values, x_values))

    # Append the data set to the list
    data_sets.append(data_set)

# Print the first few data points of the first data set
print("First few data points of the first data set:")
for i, (y, x) in enumerate(data_sets[0][:5], 1):
    print(f"Data Point {i}: (y = {y:.2f}, x = {x:.2f})")
```

```
First few data points of the first data set:
Data Point 1: (y = -1.72, x = 3.75)
Data Point 2: (y = 22.02, x = 9.51)
Data Point 3: (y = 9.83, x = 7.32)
Data Point 4: (y = 9.64, x = 5.99)
Data Point 5: (y = -1.69, x = 1.56)
```

In [53]:
```
# Lists to store linear regression coefficients
slope_estimates = []
intercept_estimates = []
```

```
ones = np.ones((30,1))

# Iterate through each dataset
for i in range(d):
    # Extract x and y values from the dataset
    data_set = data_sets[i]
    x_values = np.array([x for _, x in data_set])
    y_values = np.array([y for y, _ in data_set])

    # Calculate the least squares estimates
    x_mean = np.mean(x_values)
    y_mean = np.mean(y_values)

    X=np.column_stack((ones, x_values))
    theta = np.dot(np.linalg.inv(np.dot(X.T,X)), np.dot(X.T,y_values))

    slope_estimates.append(theta[1])
    intercept_estimates.append(theta[0])

# Print the estimates for each dataset
for i in range(d):
    print(f"Dataset {i+1}: Slope = {slope_estimates[i]:.4f}, Intercept = {ir
```

```
Dataset 1: Slope = 0.8855, Intercept = 3.4318
Dataset 2: Slope = 1.4171, Intercept = 1.8500
Dataset 3: Slope = 1.7396, Intercept = 0.9223
Dataset 4: Slope = 1.6237, Intercept = 3.6451
Dataset 5: Slope = 2.2530, Intercept = -1.9541
Dataset 6: Slope = 1.6480, Intercept = 4.2255
Dataset 7: Slope = 1.9549, Intercept = 2.1584
Dataset 8: Slope = 1.8966, Intercept = 1.9867
Dataset 9: Slope = 1.6802, Intercept = 0.0127
Dataset 10: Slope = 1.1818, Intercept = 1.3484
Dataset 11: Slope = 1.0920, Intercept = 4.9542
Dataset 12: Slope = 1.8273, Intercept = 1.6187
Dataset 13: Slope = 2.5953, Intercept = -3.0751
Dataset 14: Slope = 1.9520, Intercept = -1.4153
Dataset 15: Slope = 1.8675, Intercept = 0.6933
Dataset 16: Slope = 1.9708, Intercept = 0.2143
Dataset 17: Slope = 2.3091, Intercept = 2.9377
Dataset 18: Slope = 0.9847, Intercept = 5.0254
Dataset 19: Slope = 2.6572, Intercept = -1.5957
Dataset 20: Slope = 1.5884, Intercept = 2.7151
Dataset 21: Slope = 1.8936, Intercept = 1.9667
Dataset 22: Slope = 1.3569, Intercept = 4.4995
Dataset 23: Slope = 2.5161, Intercept = -2.0560
Dataset 24: Slope = 2.3645, Intercept = -1.0333
Dataset 25: Slope = 1.7929, Intercept = 1.6249
Dataset 26: Slope = 1.5936, Intercept = 1.1014
Dataset 27: Slope = 2.2815, Intercept = 0.4415
Dataset 28: Slope = 2.6304, Intercept = -3.8267
Dataset 29: Slope = 1.2089, Intercept = 4.1263
Dataset 30: Slope = 2.2882, Intercept = -3.9193
Dataset 31: Slope = 2.1615, Intercept = 0.8326
Dataset 32: Slope = 2.2613, Intercept = 0.4185
Dataset 33: Slope = 2.5877, Intercept = -2.4649
Dataset 34: Slope = 2.5581, Intercept = -3.3282
Dataset 35: Slope = 2.8432, Intercept = -2.9272
Dataset 36: Slope = 1.7880, Intercept = 0.4692
Dataset 37: Slope = 1.6530, Intercept = 2.9326
Dataset 38: Slope = 1.5975, Intercept = 1.1174
Dataset 39: Slope = 2.3985, Intercept = -1.9690
Dataset 40: Slope = 2.2225, Intercept = -0.7466
Dataset 41: Slope = 2.3612, Intercept = -1.6221
Dataset 42: Slope = 1.8923, Intercept = -0.0808
Dataset 43: Slope = 2.5107, Intercept = -0.1338
Dataset 44: Slope = 2.5728, Intercept = -4.1990
Dataset 45: Slope = 1.6530, Intercept = 0.8269
Dataset 46: Slope = 2.6328, Intercept = -2.5984
Dataset 47: Slope = 2.0464, Intercept = 1.2018
Dataset 48: Slope = 2.4789, Intercept = -1.7700
Dataset 49: Slope = 1.9910, Intercept = 0.3192
Dataset 50: Slope = 2.6925, Intercept = -5.3301
```

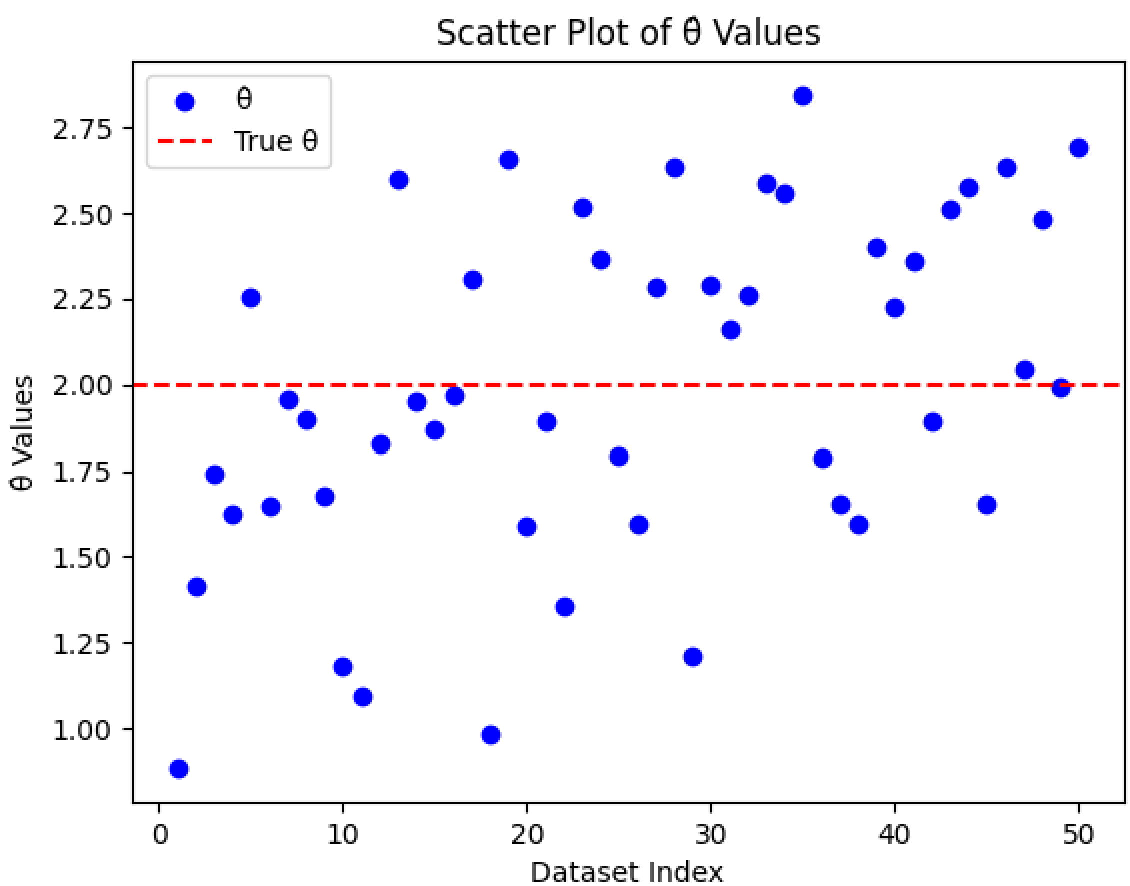
In [54]:

```
# True parameter value
true_theta = 2

# Number of datasets
d = len(slope_estimates)
```

```
# Calculate MSE
mse_theta_hat = np.mean((np.array(slope_estimates) - true_theta) ** 2)

print(f"Estimated MSE of thetâ: {mse_theta_hat:.4f}")
```

```
Estimated MSE of thetâ: 0.2380
```

In [55]:
```
# Scatter plot of estimated slopes
plt.scatter(range(1, d + 1), slope_estimates, label='θ̂', color='blue', marke
plt.axhline(y=true_theta, color='red', linestyle='--', label='True θ')

# Add labels and title
plt.xlabel('Dataset Index')
plt.ylabel('θ̂ Values')
plt.title('Scatter Plot of θ̂ Values')
plt.legend()

# Show the plot
plt.show()
```



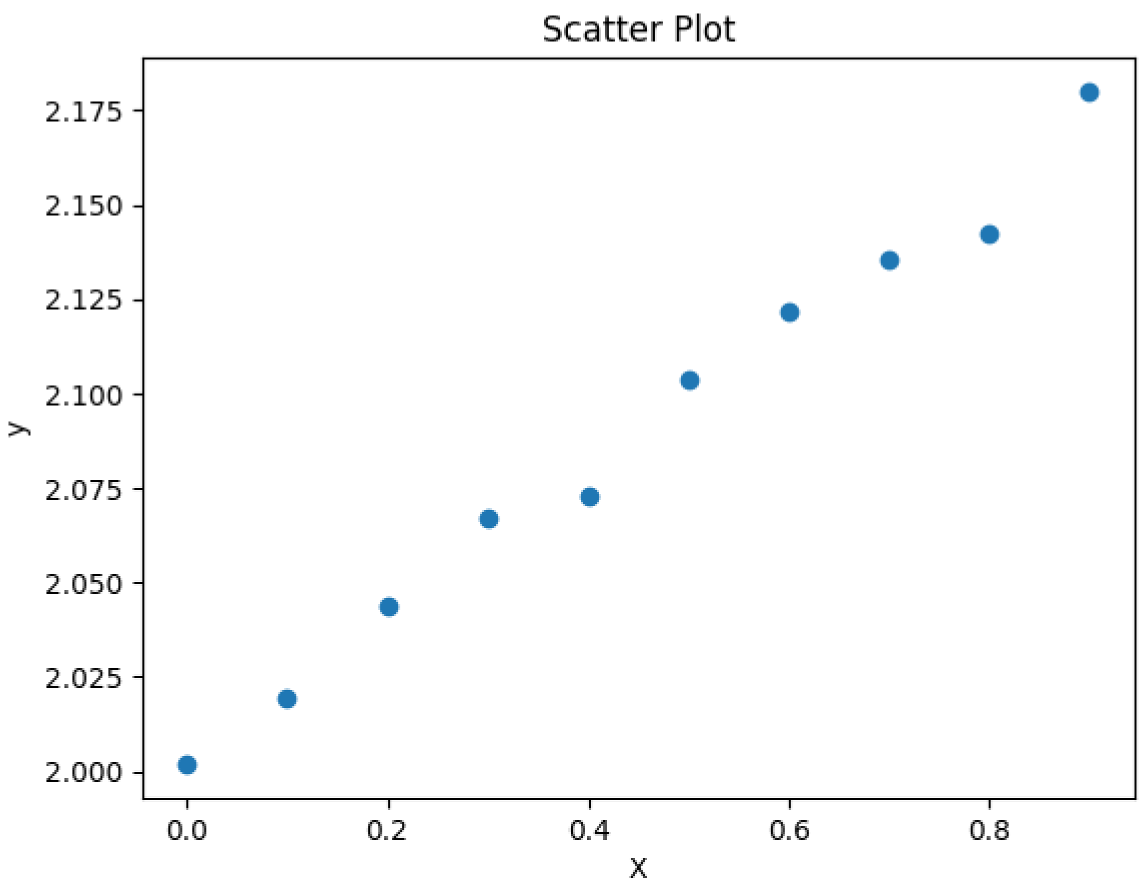
Exercise 16

In [56]:
```
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt
```

```
Training_Set = sio.loadmat('Training_Set.mat')
X = Training_Set['X']
y = Training_Set['y']
```

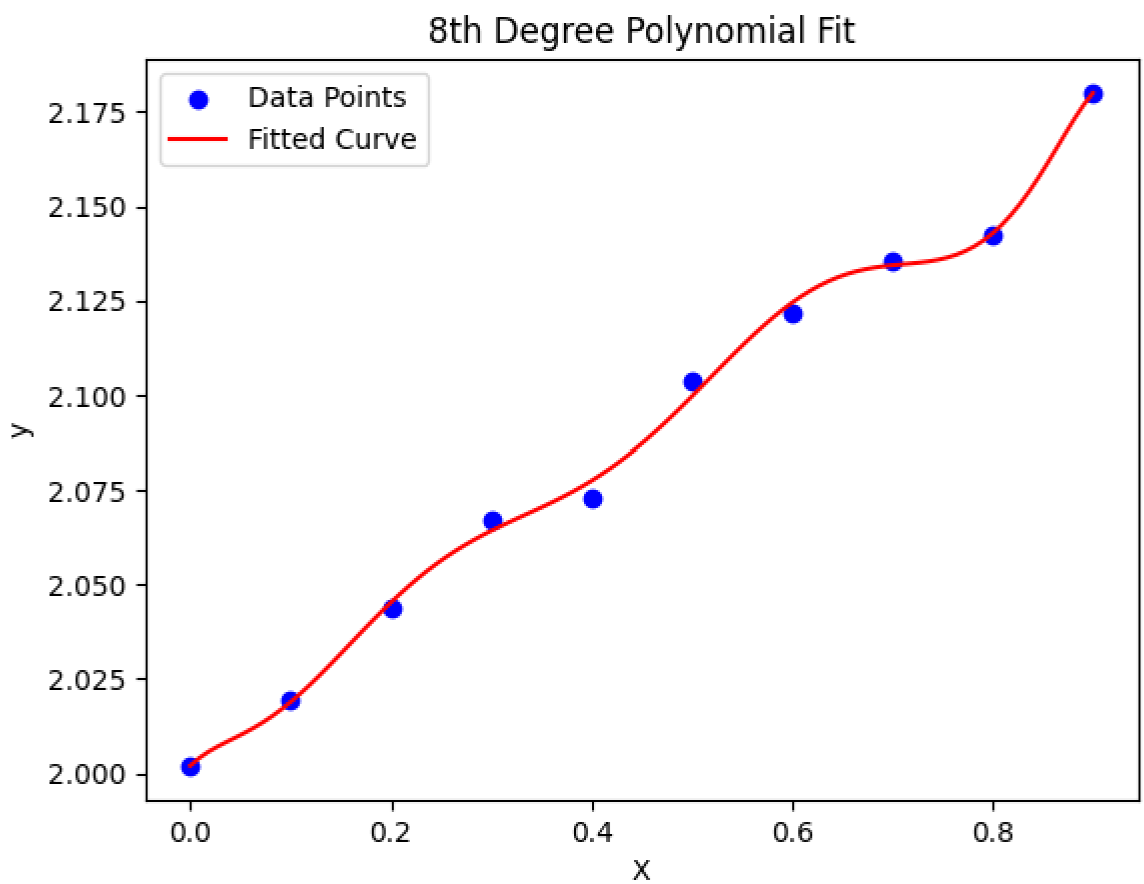
In [57]:
```
# (a)
plt.scatter(X, y, marker='o')
plt.xlabel('X')
plt.ylabel('y')
plt.title('Scatter Plot')
```

Out[57]: Text(0.5, 1.0, 'Scatter Plot')



In [58]:
```
# Fit an 8th degree polynomial using Least Squares
coefficients = np.polyfit(X.flatten(), y.flatten(), 8)

# Create a polynomial function based on the coefficients
poly_function = np.poly1d(coefficients)

# Generate x values for smooth curve plotting
x_values = np.linspace(min(X), max(X), 100)

# Calculate corresponding y values using the polynomial function
y_fit = poly_function(x_values)

# Plot the data points
plt.scatter(X, y, label='Data Points', color='blue')
```

```python
# Plot the fitted curve
plt.plot(x_values, y_fit, label='Fitted Curve', color='red')

# Add labels and title
plt.xlabel('X')
plt.ylabel('y')
plt.title('8th Degree Polynomial Fit')
plt.legend()

# Show the plot
plt.show()

# Output the estimated coefficients
print("Estimated Coefficients:")
print(coefficients)
```

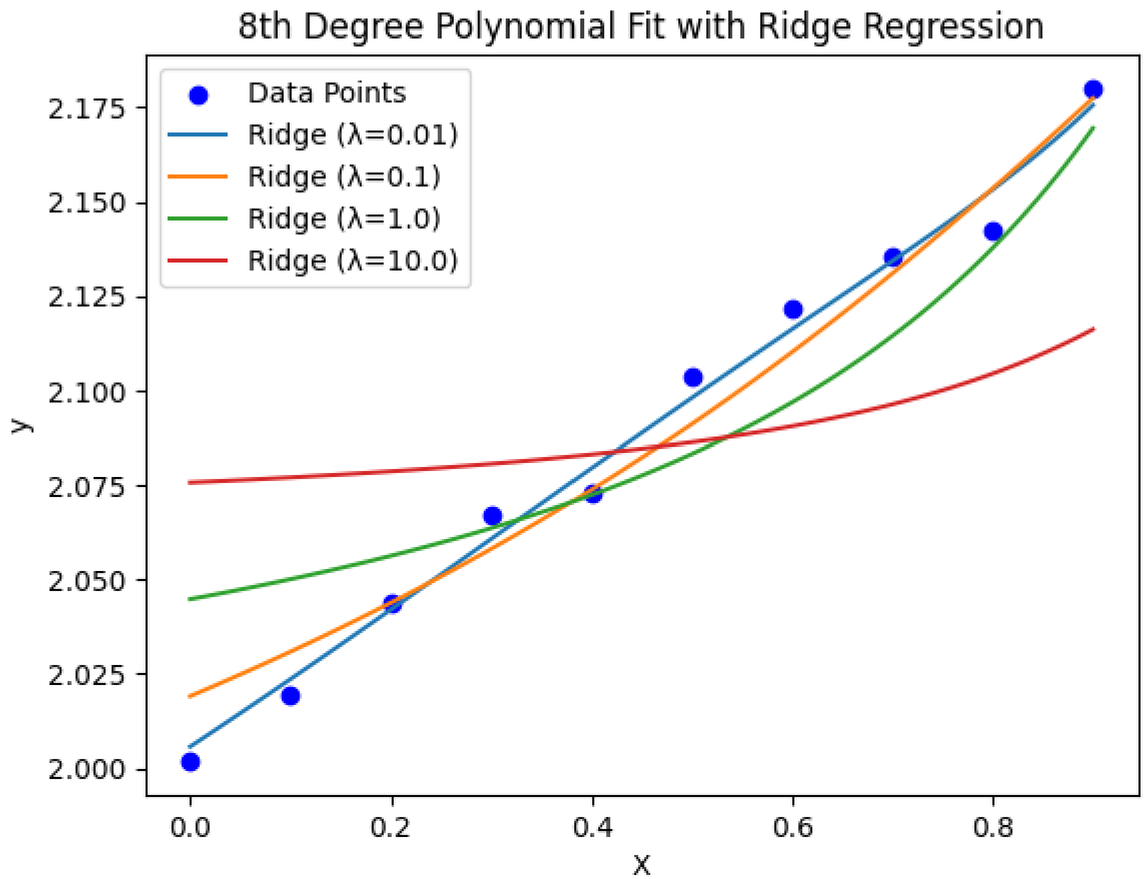



```
Estimated Coefficients:
[-2.01450791e+02  7.06143362e+02 -9.86590551e+02  6.98304163e+02
 -2.62206857e+02  4.94205642e+01 -3.83248362e+00  2.59408163e-01
  2.00199165e+00]
```

In [71]:

```python
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline


# Reshape X into a 1D array
X_flat = X.flatten()
y_flat = y.flatten()
```

```python
# Set up lambda values (ridge regularization parameter)
lambda_values = [0.01, 0.1, 1.0, 10.0]

# Plot the data points
plt.scatter(X_flat, y_flat, label='Data Points', color='blue')

# Fit and plot for each lambda value
for alpha in lambda_values:
    model = make_pipeline(PolynomialFeatures(degree=8), Ridge(alpha=alpha))
    model.fit(X_flat[:, np.newaxis], y)

    # Generate x values for smooth curve plotting
    x_values = np.linspace(min(X_flat), max(X_flat), 100)

    # Predict corresponding y values using the fitted model
    y_fit = model.predict(x_values[:, np.newaxis])

    # Plot the fitted curve
    plt.plot(x_values, y_fit, label=f'Ridge (λ={alpha})')

# Add labels and title
plt.xlabel('X')
plt.ylabel('y')
plt.title('8th Degree Polynomial Fit with Ridge Regression')
plt.legend()

# Show the plot
plt.show()

# Output the estimated coefficients for the last lambda value
coefficients = model.named_steps['ridge'].coef_
print("Estimated Coefficients:")
print(coefficients)
```




```
Estimated Coefficients:
[[0.         0.01220436 0.01061126 0.00870739 0.00716156 0.0059602
  0.00502111 0.00427527 0.00367277]]
```

(d)

The line created using the LS estimator follows the data points more closely (overfitting) than the lines created using ridge regression. This is to be expected since the $\lambda \| \boldsymbol{\theta}\|^2$ term biases the solution away from the one obtained for the unregularized case, thus the line does not follow the data points as closely. For smaller values of λ the line follows the data points but for $\lambda > 0.1$ the produced polynomial coefficients lead to lines that do not follow the data points (underfitting).

```
In [ ]:
```