Jake Mavrides

1] a) $P_1(0,2): y_1(0) = \boxed{2 = a_1}$

$P_2(\frac{1}{2},0): y_1(\frac{1}{2}) = \boxed{0 = a_1 + \frac{1}{2}a_2 + \frac{1}{4}a_3 + \frac{1}{8}a_4}$

$P_3(1,1): y_1(1) = \boxed{1 = a_1 + a_2 + a_3 + a_4}$

Matrix Form: $A\bar{x} = \bar{b}$ $\quad A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1/2 & 1/4 & 1/8 \\ 1 & 1 & 1 & 1 \end{bmatrix}$ $\quad \bar{x} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}$ $\quad \bar{b} = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$

---

b) $P_4(\frac{3}{2},2): y_2(\frac{3}{2}) = \boxed{2 = a_5 + \frac{3}{2}a_6 + \frac{9}{4}a_7 + \frac{27}{8}a_8}$

Sorry order

$P_3(1,1): y_2(1) = \boxed{1 = a_5 + a_6 + a_7 + a_8}$

$P_5(2,0): y_2(2) = \boxed{0 = a_5 + 2a_6 + 4a_7 + 8a_8}$

Matrix: $A\bar{x} = \bar{b}$ $\quad A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 3/2 & 9/4 & 27/8 \\ 1 & 2 & 4 & 8 \end{bmatrix}$ $\quad \bar{x} = \begin{bmatrix} a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix}$ $\quad \bar{b} = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$

---

c) First check $(y_1'(1) = y_2'(1))$ for slope continuity:

$$y_1'(x) = a_2 + 2a_3x + 3a_4x^2$$
$$y_2'(x) = a_6 + 2a_7x + 3a_8x^2$$
$$\Rightarrow \boxed{y_1'(1) = a_2 + 2a_3 + 3a_4 = a_6 + 2a_7 + 3a_8 = y_2'(1)}$$
$$\boxed{\text{or: } a_2 + 2a_3 + 3a_4 - a_6 - 2a_7 - 3a_8 = 0}$$

Curvature: $(y_1''(1) = y_2''(1))$

$$y_1''(x) = 2a_3 + 6a_4x$$
$$y_2''(x) = 2a_7 + 6a_8x$$

@ x=1 $\Rightarrow \boxed{y_1''(1) = 2a_3 + 6a_4 = 2a_7 + 6a_8 = y_2''(1)}$

$$\boxed{\text{or: } 2a_3 + 6a_4 - 2a_7 - 6a_8 = 0}$$

1] d) $A\bar{a} = \bar{b}$

$\bar{a} = [a_1\ a_2\ \dots\ a_8]^T$ $\quad \bar{b} = \begin{bmatrix} 2 \\ 0 \\ 1 \\ 1 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ $\quad A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{8} & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & \frac{3}{2} & \frac{9}{4} & \frac{27}{8} \\ 0 & 0 & 0 & 0 & 1 & 2 & 4 & 8 \\ 0 & 1 & 2 & 3 & 0 & -1 & -2 & -3 \\ 0 & 0 & 2 & 6 & 0 & 0 & -2 & -6 \end{bmatrix}$

e) $\bar{a} = \begin{bmatrix} 2 \\ -9 \\ 12 \\ -4 \\ 2 \\ -9 \\ 12 \\ -4 \end{bmatrix}$ $\leftarrow$ in Python

$\Rightarrow y_1(x) = 2 - 9x + 12x^2 - 4x^3$

$\rightarrow y_2(x) = 2 - 9x + 12x^2 - 4x^3$

f) in Python

2) a) $\bar{x}_1 = [1,1]$, $y_1 = 0.1$
$\bar{x}_2 = [-1,-1]$, $y_2 = 5.95$
$\bar{x}_3 = [0,1]$, $y_3 = .8$
$\bar{x}_4 = [1,0]$, $y_4 = 2.1$
$\bar{x}_5 = [1,2]$, $y_5 = -1.8$
$\bar{x}_6 = [2,1]$, $y_6 = -1.05$

$(\hat{y} = X\beta)$

- $J(\beta) = \sum_{i=1}^{n} |y_i - \hat{y}_i|^2 = (y-\hat{y})^T(y-\hat{y})$
$= (y - X\beta)^T(y - X\beta)$

- Let $f(\beta) = y - X\beta$
and $g(u) = u^T u$ where $u = f(\beta)$

- Then $\nabla J = \frac{\partial(g(u))}{\partial u} \cdot \frac{\partial(f(\beta))}{\partial \beta}$

- $\frac{\partial(g(u))}{\partial u} = \frac{u^T u}{\partial u} = 2u^T$

- $\frac{\partial f(\beta)}{\partial \beta} = \frac{\partial(y - X\beta)}{\partial \beta} = -X$

$\rightarrow \nabla J = 2u^T(-X) = -2(y-X\beta)^T X$

- want $\nabla J = 0 \Rightarrow -2(y - X\beta)^T X = 0 \Rightarrow (y-X\beta)^T X = 0$

- transpose both sides $\Rightarrow X^T(y - X\beta) = 0 \Rightarrow X^T y - X^T X\beta = 0$

$\Rightarrow X^T y = X^T X \beta$

$$\boxed{\Rightarrow (X^T X)^{-1} X^T y = \beta}$$

normal eqn

w/ $\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} \in \mathbb{R}^{3\times 1}$

and $X = \begin{bmatrix} 1 & x_{11} & x_{21} \\ 1 & x_{12} & x_{22} \\ \vdots & \vdots & \vdots \\ 1 & x_{1n} & x_{2n} \end{bmatrix} \in \mathbb{R}^{n \times 3}$

n=6 in our ex.

and

$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^{n\times 1}$ ($n=6$ in our ex.)

2) b) In notebook: $f(x) = 2.9769 + -.9577x_1 + -1.9827x_2$

c) In notebook

d) In notebook: ~ .0873

3) a) $J(\beta) = \|X\beta - y\|^2 + \lambda^2 \|\beta\|^2$

Question 2

gradient: $\nabla J(\beta) = 2X^T(X\beta - y) + 2\lambda^2 \beta$

Set to 0: $0 = X^T X\beta - X^T y + \lambda^2 \beta \Rightarrow X^T y = (X^T X + \lambda^2)\beta$

$$\Rightarrow \boxed{\beta = (X^T X + \lambda^2)^{-1} X^T y}$$

Hessian: $\nabla^2 J(\beta) = 2X^T X + 2\lambda^2 I$

Must be positive definite for normal eqn to be valid

1) $\cdot$ $X^T X$ and I are symmetric ✓

2) $\cdot$ Eigenvalues are positive b/c $\lambda > 0$ and $X^T X$ is pos. semi-definite

So our normal eqn. is valid ✓

b) In notebook: $\beta \cong \begin{bmatrix} 2.204 \\ -.824 \\ -1.508 \end{bmatrix}$

c) In notebook: $\beta \approx \begin{bmatrix} 1.211 \\ -.580 \\ -.922 \end{bmatrix}$

d) $\lambda = 1$: RSS = 2.6166

$\lambda = 2$: RSS = 13.4289

```
import numpy as np
from numpy import linalg as LA
import matplotlib.pyplot as plt
```

Question 1:

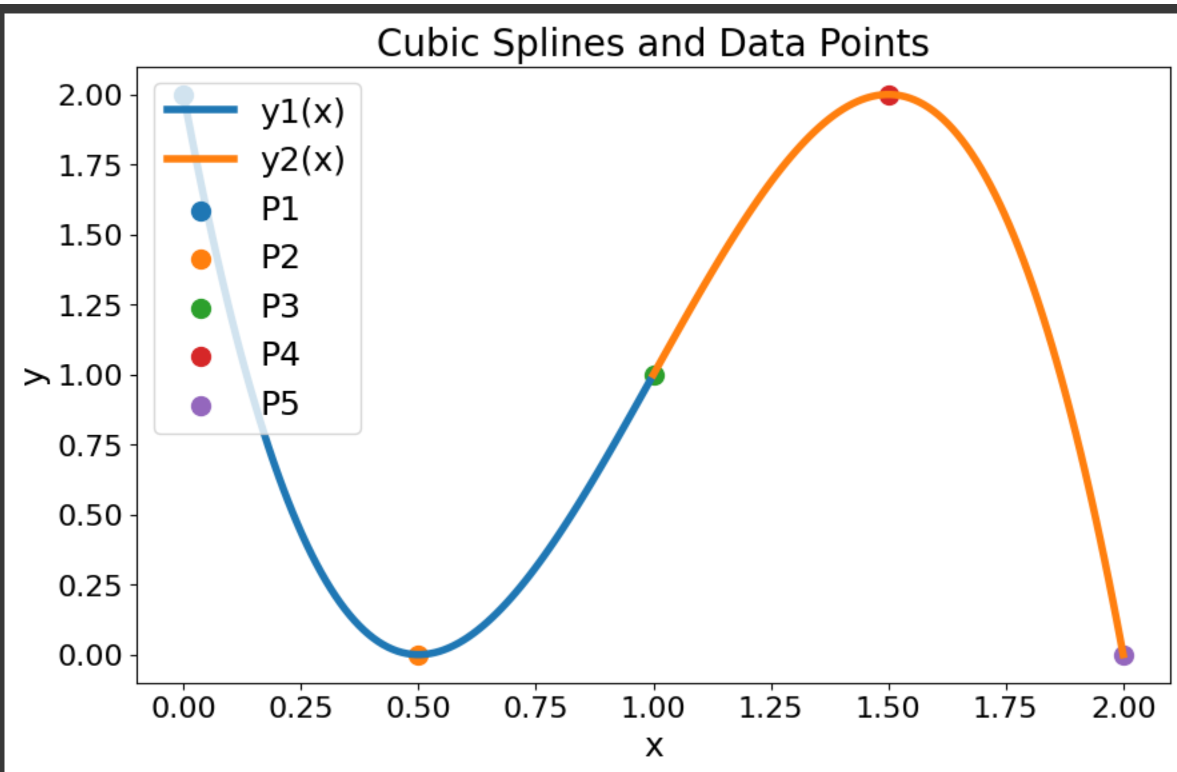
```
A = np.array([
    [1, 0, 0, 0, 0, 0, 0, 0],
    [1, .5, .25, .125, 0, 0, 0, 0],
    [1, 1, 1, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 1, 1, 1, 1],
    [0, 0, 0, 0, 1, (3/2), (9/4), (27/8)],
    [0, 0, 0, 0, 1, 2, 4, 8],
    [0, 1, 2, 3, 0, -1, -2, -3],
    [0, 0, 2, 6, 0, 0, -2, -6]
])

b = np.array([2, 0, 1, 1, 2, 0, 0, 0])
```

```
a = LA.solve(A, b)
a
```

```
array([ 2., -9., 12., -4.,  2., -9., 12., -4.])
```

```
# Generate x values for each interval
x1 = np.linspace(0, 1, 100)
x2 = np.linspace(1, 2, 100)

def y1(x):
    return a[0] + a[1]*x + a[2]*x**2 + a[3]*x**3

def y2(x):
    return a[4] + a[5]*x + a[6]*x**2 + a[7]*x**3
```

```
# Define the data points
data_points = {
    'P1': (0, 2),
    'P2': (0.5, 0),
    'P3': (1, 1),
    'P4': (1.5, 2),
    'P5': (2, 0)
}

# Generate y values using y1 and y2
y_interval_1 = y1(x1)
y_interval_2 = y2(x2)

# Plot the cubics, linewidth of 4 and markersize of 10
plt.figure(figsize=(10, 6))
plt.plot(x1, y_interval_1, label='y1(x)', linewidth=4, markersize=10)
plt.plot(x2, y_interval_2, label='y2(x)', linewidth=4, markersize=10)

# Plot points
for point, coordinates in data_points.items():
    plt.scatter(*coordinates, label=point, s=100, marker='o')

# Create labels and title with 18pt font
plt.xlabel('x', fontsize=18)
plt.ylabel('y', fontsize=18)
plt.title('Cubic Splines and Data Points', fontsize=20)

# Create legend with 18pt font
plt.legend(fontsize=18)

# Set tick font size
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)

plt.show()
```



Question 2:

```
X_data = np.array([[1, 1], [-1, -1], [0, 1], [1, 0], [1, 2], [2, 1]])
y_data = np.array([0.1, 5.95, 0.8, 2.1, -1.8, -1.05])
```

```
# add column of ones for intercept term
X = np.c_[np.ones(X_data.shape[0]), X_data]
X
```

```
array([[ 1.,  1.,  1.],
       [ 1., -1., -1.],
       [ 1.,  0.,  1.],
       [ 1.,  1.,  0.],
       [ 1.,  1.,  2.],
       [ 1.,  2.,  1.]])
```

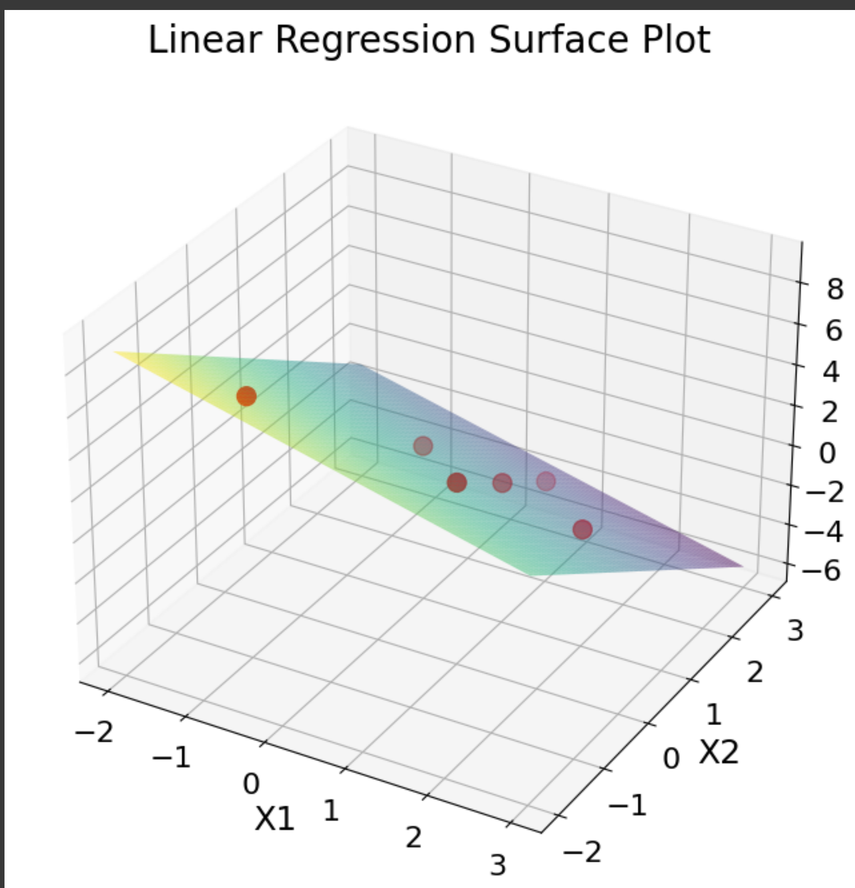
```
XTX = X.T @ X
XTy = X.T @ y_data
beta = LA.solve(XTX, XTy)
print("coeffs: ", beta)
```

```
coeffs:  [ 2.97692308 -0.95769231 -1.98269231]
```

```
print("Regression model: f(x) = {:.4f} + {:.4f}*x_1 + {:.4f}*x_2".format(beta[0], beta[1], beta[2]))
```

```
Regression model: f(x) = 2.9769 + -0.9577*x_1 + -1.9827*x_2
```

```
# Creating meshgrid for plot
x1_range = np.linspace(-2, 3, 100)
x2_range = np.linspace(-2, 3, 100)
x1_mesh, x2_mesh = np.meshgrid(x1_range, x2_range)

# Our regression model
f_hat = beta[0] + beta[1] * x1_mesh + beta[2] * x2_mesh

# Plotting
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Surface plot of model
ax.plot_surface(x1_mesh, x2_mesh, f_hat, alpha=0.5, cmap='viridis', label=r'$\hat{f}$', linewidth=0)

# Adding our data points:
ax.scatter(X_data[:, 0], X_data[:, 1], y_data, color='red', s=100, label='Data Points')

# Labeling
ax.set_xlabel('X1', fontsize=18)
ax.set_ylabel('X2', fontsize=18)
ax.set_zlabel('Y', fontsize=18)
ax.set_title('Linear Regression Surface Plot', fontsize=20)
#ax.legend(fontsize=18)
ax.tick_params(labelsize=16)

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




```
# Finding residual SS:
y_hat = X @ beta
RSS = np.sum((y_data - y_hat)**2)
print("RSS: ", RSS)
```

```
RSS:  0.08730769230769221
```

Question 3:

```
# regularization param
lambda_val_1 = 1

# design matrix
X_data = np.array([[1, 1, 1],
              [1, -1, -1],
              [1, 0, 1],
              [1, 1, 0],
              [1, 1, 2],
              [1, 2, 1]])

y_data = np.array([0.1, 5.95, 0.8, 2.1, -1.8, -1.05])
```

```
# Normal equation matrix A = X^T*X + lambda^2*Identity
X = X_data
y = y_data
A = X.T@X + (lambda_val_1**2 * np.identity(3))
beta = LA.solve(A, X.T@y)
```

```
print("beta vals (lambda=1): ", beta)
```

```
beta vals (lambda = 1):  [ 2.20410959 -0.82442922 -1.50776256]
```

```
# Part c:
lambda_val_2 = 2

A2 = X.T@X + (lambda_val_2**2 * np.identity(3))
beta2 = LA.solve(A2, X.T@y_data)
```

```
print("beta vals (lambda=2): ", beta2)
```

```
beta vals (lambda = 2):  [ 1.21081081 -0.58018018 -0.92184685]
```

```
# Part d:

# Finding RSS with lambda=1:
y_pred = X @ beta
rss1 = np.sum((y - y_pred)**2)
print("rss with lambda=1: ", rss1)
```

```
rss with lambda=1:  2.6166205875607265
```

```
# Part d:
# Finding RSS with lambda=2:
y_pred2 = X @ beta2
rss2 = np.sum((y - y_pred2)**2)
print("rss with lambda=2: ", rss2)
```

```
rss with lambda=2:  13.428867380894411
```