Homework 5

May 9, 2025

Question 1 : Part 1 : minimum value of $S(a_0, a_1, a_2)$

$$z(x,y) = a_0 + a_1 x + a_2 y$$

$$S(a_0, a_1, a_2) = \sum_{i=1}^{N} \left[z_i - (a_0 + a_1 x_1 + a_2 y_i) \right]^2$$

1.

$$\frac{\partial S}{\partial a_0} = \sum_{i=1}^{N} 2 \left[z_i - (a_0 + a_1 x + a_2 y) \right] \cdot \frac{\partial}{\partial a_0} \left[-(a_0 + a_1 x_i + a_2 y_i) \right]$$

$$= -2 \sum_{i=1}^{N} \left[z_i - (a_0 + a_1 x_i + a_2 y_i) \right]$$

$$\sum_{i=1}^{N} \left[z_i - (a_0 + a_1 x_i + a_2 y_i) \right] = 0$$

$$\sum_i z_i - \sum_i a_0 - \sum_i a_1 x_i - \sum_i a_2 y_i = 0$$

$$\Rightarrow N a_0 + \left(\sum_i x_i \right) a_1 + \left(\sum_i y_i \right) a_2 = \sum_i z_i$$

2.

$$\frac{\partial S}{\partial a_1} = -2 \sum_{i=1}^{N} x_i \left[z_i - (a_0 + a_1 x_i + a_2 y_i) \right] = 0$$

$$\sum_i x_i z_i - a_0 \sum_i x_i - a_1 \sum_i x_i^2 - a_2 \sum_i x_i y_i = 0$$

$$\left(\sum_i x_i \right) a_0 + \left(\sum_i x_i^2 \right) a_1 + \left(\sum_i x_i y_i \right) a_2 = \sum_i x_i z_i$$

3. $\sum_i y_i z_i - a_0 \sum_i y_i - a_1 \sum x_i y_i - a_2 \sum_i y_i^2 = 0$

$$\left(\sum_i y_i\right)a_0 + \left(\sum_i x_i y_i\right)a_1 + \left(\sum_i y_i^2\right)a_2 = \sum_i y_i z_i$$

abbrv: $S_0 = \sum_i 1 = N \quad S_x = \sum_i x_i, \; S_y = \sum_i y_i,$

$S_{xx} = \sum_i x_i^2, \quad S_{xy} = \sum_i x_i y_i, \; S_{yy} = \sum_i y_i^2,$

$S_z = \sum_i z_i \quad S_{xz} = \sum_i x_i z_i, \; S_{yz} = \sum_i y_i z_i$

three equations become

$$\begin{cases} S_0 a_0 + S_x a_1 + S_y a_2 = S_z \\ S_x a_0 + S_{xx} a_1 + S_{xy} a_2 = S_{xz} \\ S_y a_0 + S_{xy} a_1 + S_{yy} a_2 = S_{yz} \end{cases}$$

Compact Matrix form:

$$\begin{bmatrix} S_0 & S_x & S_y \\ S_x & S_{xx} & S_{xy} \\ S_y & S_{xy} & S_{yy} \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} S_z \\ S_{xz} \\ S_{yz} \end{bmatrix}$$

Homework 5

May 9, 2025

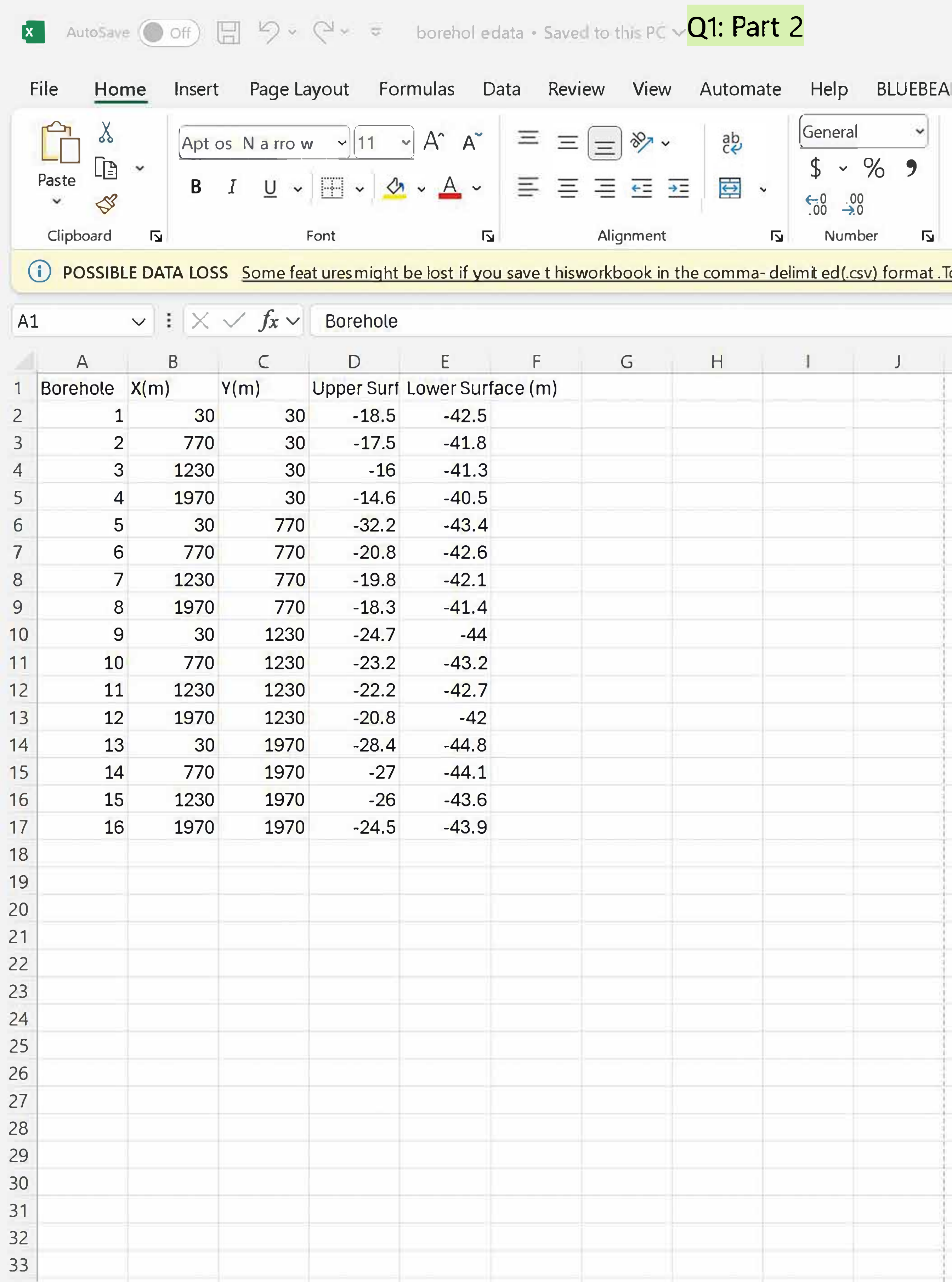
Question 1: Part 2, 3, and 4

[13]:
```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import pandas as pd

# Read the borehole data
borehole_data = pd.read_csv('borehole-data.csv')
print("Borehole Data:")
print(borehole_data)

# Print column names to debug
print("\nColumn names in the CSV file:")
for col in borehole_data.columns:
    print(f"'{col}'")

# Extract the data
upper_col = [col for col in borehole_data.columns if 'Upper Surface' in col][0]
lower_col = [col for col in borehole_data.columns if 'Lower Surface' in col][0]

x = borehole_data['X(m)'].values
y = borehole_data['Y(m)'].values
upper_z = borehole_data[upper_col].values
lower_z = borehole_data[lower_col].values
N = len(x)

print(f"\nUsing columns: X(m), Y(m), {upper_col}, {lower_col}")

# Create a 3D plot
fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111, projection='3d')

# Plot the upper and lower surfaces
ax.scatter(x, y, upper_z, c='b', marker='o', label='Upper Surface')
ax.scatter(x, y, lower_z, c='r', marker='^', label='Lower Surface')

# Setting labels and title
```

```
ax.set_xlabel('X (m)')
ax.set_ylabel('Y (m)')
ax.set_zlabel('Depth (m)')
ax.set_title('Mineral Deposits: Upper and Lower Surfaces')
ax.legend()

# Set up matrix equations for upper surface: z(x, y) = a0 + a1*x + a2*y

A = np.zeros((3, 3))
A[0, 0] = N
A[0, 1] = np.sum(x)
A[0, 2] = np.sum(y)
A[1, 0] = np.sum(x)
A[1, 1] = np.sum(x**2)
A[1, 2] = np.sum(x * y)
A[2, 0] = np.sum(y)
A[2, 1] = np.sum(x * y)
A[2, 2] = np.sum(y**2)

# Right hand side for upper surface
b_upper = np.zeros(3)
b_upper[0] = np.sum(upper_z)
b_upper[1] = np.sum(x * upper_z)
b_upper[2] = np.sum(y * upper_z)

# Right hand side for lower surface
b_lower = np.zeros(3)
b_lower[0] = np.sum(lower_z)
b_lower[1] = np.sum(x * lower_z)
b_lower[2] = np.sum(y * lower_z)

# Solve for coefficients
coeffs_upper = np.linalg.solve(A, b_upper)
coeffs_lower = np.linalg.solve(A, b_lower)

print("\nUpper Surface Equation: z(x,y) = {:.4f} + {:.4f}*x + {:.4f}*y".format(
    coeffs_upper[0], coeffs_upper[1], coeffs_upper[2]))
print("Lower Surface Equation: z(x,y) = {:.4f} + {:.4f}*x + {:.4f}*y".format(
    coeffs_lower[0], coeffs_lower[1], coeffs_lower[2]))

# Create a grid for the plane surfaces
grid_x, grid_y = np.meshgrid(np.linspace(min(x), max(x), 50),
                             np.linspace(min(y), max(y), 50))
grid_z_upper = coeffs_upper[0] + coeffs_upper[1] * grid_x + coeffs_upper[2] *␣
↪grid_y
grid_z_lower = coeffs_lower[0] + coeffs_lower[1] * grid_x + coeffs_lower[2] *␣
↪grid_y
```

```
# Plot the planes
ax.plot_surface(grid_x, grid_y, grid_z_upper, alpha=0.3, color='blue')
ax.plot_surface(grid_x, grid_y, grid_z_lower, alpha=0.3, color='red')

# Calculate average depth and volume
# Site dimensions are explicitly given as 2 km × 2 km
site_length = 2000  # 2 km = 2000 m
site_width = 2000   # 2 km = 2000 m
site_area = site_length * site_width  # 4,000,000 m²

# Sample the planes at a grid of points and calculate average thickness
num_samples = 100
sample_x = np.linspace(min(x), max(x), num_samples)
sample_y = np.linspace(min(y), max(y), num_samples)
xx, yy = np.meshgrid(sample_x, sample_y)
upper_surface = coeffs_upper[0] + coeffs_upper[1] * xx + coeffs_upper[2] * yy
lower_surface = coeffs_lower[0] + coeffs_lower[1] * xx + coeffs_lower[2] * yy



# For raw data points
thickness_raw = np.abs(upper_z - lower_z)
avg_thickness_raw = np.mean(thickness_raw)

# For the fitted planes
# Both should be negative values, with lower surface more negative than upper
thickness_planes = np.abs(lower_surface - upper_surface) # Should be positive␣
  ↪values
avg_thickness_planes = np.mean(thickness_planes)

# Calculate volume (area × average thickness) - using the fitted planes value
volume = site_area * avg_thickness_planes

print(f"Average thickness from raw data points: {avg_thickness_raw:.2f} m")
print(f"Average thickness from fitted planes: {avg_thickness_planes:.2f} m")

# Print results
print("\nCalculations:")
print(f"Site dimensions: {site_length} m × {site_width} m")
print(f"Site area: {site_area} m²")
print(f"Average mineral deposit thickness: {avg_thickness_planes:.2f} m")
```

Q1: Part 2


AutoSave Off
borehol edata • Saved to this PC
File Home Insert Page Layout Formulas Data Review View Automate Help BLUEBEA
Paste
Clipboard
Apt os N a rro w 11
Font
Alignment
General
Number

POSSIBLE DATA LOSS Some feat ures might be lost if you save t his workbook in the comma- delimit ed (.csv) format . To

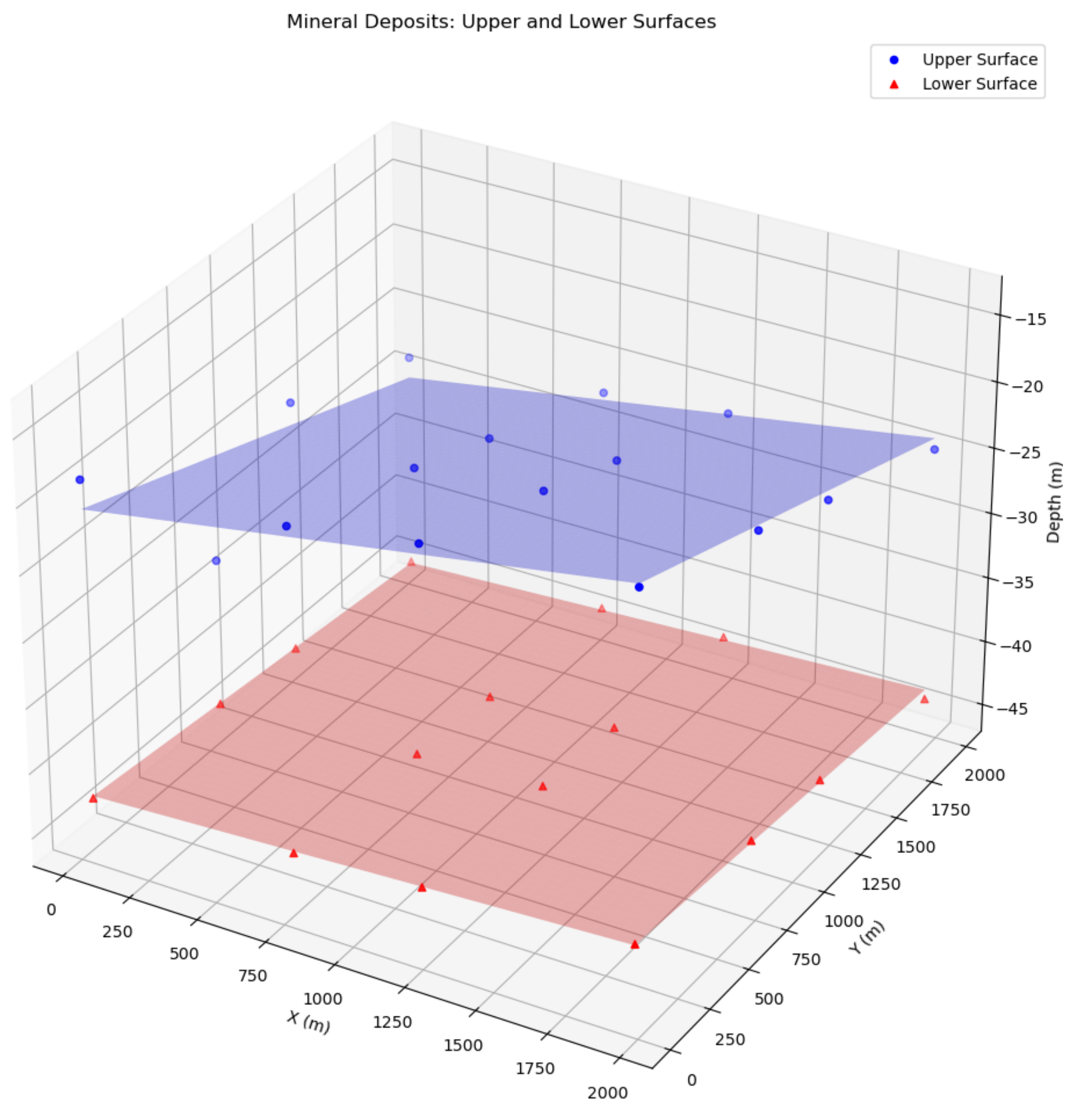
A1 Borehole

Borehole	X(m)	Y(m)	Upper Surf	Lower Surface (m)
1	30	30	-18.5	-42.5
2	770	30	-17.5	-41.8
3	1230	30	-16	-41.3
4	1970	30	-14.6	-40.5
5	30	770	-32.2	-43.4
6	770	770	-20.8	-42.6
7	1230	770	-19.8	-42.1
8	1970	770	-18.3	-41.4
9	30	1230	-24.7	-44
10	770	1230	-23.2	-43.2
11	1230	1230	-22.2	-42.7
12	1970	1230	-20.8	-42
13	30	1970	-28.4	-44.8
14	770	1970	-27	-44.1
15	1230	1970	-26	-43.6
16	1970	1970	-24.5	-43.9

```python
print(f"Total mineral deposit volume: {volume:.2f} m³")

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

```
Borehole Data:
    Borehole  X(m)  Y(m)  Upper Surface (m)   Lower Surface (m)
0          1    30    30              -18.5               -42.5
1          2   770    30              -17.5               -41.8
2          3  1230    30              -16.0               -41.3
3          4  1970    30              -14.6               -40.5
4          5    30   770              -32.2               -43.4
5          6   770   770              -20.8               -42.6
6          7  1230   770              -19.8               -42.1
7          8  1970   770              -18.3               -41.4
8          9    30  1230              -24.7               -44.0
9         10   770  1230              -23.2               -43.2
10        11  1230  1230              -22.2               -42.7
11        12  1970  1230              -20.8               -42.0
12        13    30  1970              -28.4               -44.8
13        14   770  1970              -27.0               -44.1
14        15  1230  1970              -26.0               -43.6
15        16  1970  1970              -24.5               -43.9

Column names in the CSV file:
'Borehole'
'X(m)'
'Y(m)'
'Upper Surface (m)'
' Lower Surface (m)'

Using columns: X(m), Y(m), Upper Surface (m),  Lower Surface (m)

Upper Surface Equation: z(x,y) = -20.6207 + 0.0033*x + -0.0048*y
Lower Surface Equation: z(x,y) = -42.3174 + 0.0009*x + -0.0013*y
Average thickness from raw data points: 20.59 m
Average thickness from fitted planes: 20.59 m

Calculations:
Site dimensions: 2000 m × 2000 m
Site area: 4000000 m²
Average mineral deposit thickness: 20.59 m
Total mineral deposit volume: 82350000.00 m³
```

Q2: Part 3



Question 4: beginning

[7]:
```
import numpy as np

# Define the function to be integrated
def f(x):
    return 3*x**2 + 4*x**3 + 5*x**4

# Define the exact solution (given)
exact_solution = 56

# Integration limits
a = 0
b = 2
```

Question 2 : Part 1 : Divided differences

i	x_i	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$	$f[x_i, x_{i+1}, x_{i+2}, x_{i+3}]$
0	0.0	36.0	$\frac{32-36}{2-0} = -2.0$	$\frac{-5-(-2)}{3-0} = -1.$	$\frac{-1-(-1)}{5-0} = 0$
1	2.0	32.0	$\frac{27-32}{3-2} = -5.0$	$\frac{-8-(-5)}{5-2} = -1.$	
2	3.0	27.0	$\frac{11-27}{5-3} = -8.0$		
3	5.0	11.0			

first differences	Second differences	Third differences
$f[x_0, x_1] = -2$	$f[x_0, x_1, x_2] = -1$	$f[x_0, x_1, x_2, x_3] = 0$
$f[x_1, x_2] = -5$	$f[x_1, x_2, x_3] = -1$	
$f[x_2, x_3] = -8$		

$$P(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)$$

$\longrightarrow$ substituting :

$$P(x) = 36 + (-2)x + (-1)x(x-2) = 36 - 2x - x^2 + 2x = \boxed{36 - x^2}$$

$P(0) = 36$, $P(2) = 32$, $P(3) = 27$, $P(5) = 11$

Part 2: Lagrange Interpolation

$$P(x) = \sum_{i=0}^{3} f(x_i) L_i(x), \quad L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{3} \frac{x - x_j}{x_i - x_j}$$

$$L_0(x) = \frac{(x-2)(x-3)(x-5)}{(0-2)(0-3)(0-5)} = \frac{(x-2)(x-3)(x-5)}{-30}$$

$$L_1(x) = \frac{(x-0)(x-3)(x-5)}{(2-0)(2-3)(2-5)} = \frac{(x)(x-3)(x-5)}{6}$$

$$L_2(x) = \frac{(x-0)(x-2)(x-5)}{(3-0)(3-2)(3-5)} = \frac{(x)(x-2)(x-5)}{-6}$$

$$L_3(x) = \frac{(x-0)(x-2)(x-3)}{(5-0)(5-2)(5-3)} = \frac{(x)(x-2)(x-3)}{30}$$

$$P(x) = 36 L_0(x) + 32 L_1(x) + 27 L_2(x) + 11 L_3(x)$$

$$P(x) = 36 \frac{(x-2)(x-3)(x-5)}{-30} + 32 \frac{x(x-3)(x-5)}{6} + 27 \frac{x(x-2)(x-5)}{-6}$$

$$- 11 \frac{x(x-2)(x-3)}{30}$$

Numerators:

1. $(x-2)(x-3)(x-5) = (x^2-5x+6)(x-5) = x^3 - 10x^2 + 31x - 30$

2. $\cancel{x}(x-3)(\cancel{x}-5) = (x^2-8x+15)x = x^3-8x^2+15x$

3. $x(x-2)(x-5) = (x^2-7x+10)x = x^3-7x^2+10x$

4. $x(x-2)(x-3) = (x^2-5x+6)x = x^3-5x^2+6x$

Term 1

$$36 \cdot \frac{x^3-10x^2+31x-30}{-30} = -\frac{36}{30}(x^3-10x^2+31x-30) = -\frac{6}{5}x^3 + 12x^2 - \frac{186}{5}x + 36$$

$+$

Term 2

$$32 \cdot \frac{x^3-8x^2+15x}{6} = \frac{32}{6}(x^3-8x^2+15x) = \frac{16}{3}x^3 - \frac{128}{3}x^2 + 80x$$

$+$

Term 3

$$27 \cdot \frac{-(x^3-7x^2+10}{6} = \frac{-27}{6}(x^3-7x^2+10x) = -\frac{9}{2}x^3 + \frac{63}{2}x^2 - 45x$$

$+$

Term 4:

$$11 \cdot \frac{x^3-5x^2+6x}{30} = \frac{11}{30}x^3 - \frac{11}{6}x^2 + \frac{11}{5}x$$

$$\rightarrow \frac{-36+160-136+11}{30} = 0(x^3)$$

$$\rightarrow 12 - \frac{128}{3} + \frac{63}{2} - \frac{11}{6} = -1(x^2)$$

$$\rightarrow -\frac{186}{5} + 80 - 45 + \frac{11}{5} = 0(x)$$

$$\rightarrow +36$$

$$\Longrightarrow P(x) = 36 + (-1)x^2 = \boxed{36 - x^2}$$

Question 3

Part 1: Intersection equation

$f(x) = x^2 - 16$

$g(x) = \frac{x^3}{8} - 2x + 5$

$> \ f(x) = g(x)$

$$\Rightarrow \quad x^2 - 16 = \frac{x^3}{8} - 2x + 5$$

$$8 \cdot (x^2 - 16) = \cancel{8}\left(\frac{x^3}{\cancel{8}} - 2x^{\cdot 8} + 5^{\cdot 8}\right)$$

$$8x^2 - 128 = x^3 - 16x + 40$$

$$\rightarrow \quad 8x^2 - 16x + 168 = 0$$

long division: $(x-6)(x^2 - 2x - 28) = 0$

$x = 6$

$$x_{1,2} = \frac{-(-2) \pm \sqrt{(-2)^2 - 4 \cdot 1 \cdot (-28)}}{2 \cdot 1}$$

$$x_{1,2} = \frac{2 \pm \sqrt{4 + 112}}{2} = \frac{2 \pm 2\sqrt{29}}{2}$$

$$x_{1,2} = 1 \pm \sqrt{29}$$

The two intersection labeled A and E in the figure lie at :

$A: x = 1 - \sqrt{29} \qquad E: x = 1 + \sqrt{29}$

Part 2: Area

$x = A$ and $x = E$, since $g(x) \geq f(x)$ for x from A up to the center crossing at $x = 6$, and $f(x) \geq g(x)$ for $6 \leq x \leq E$

$$\text{Area} = \int_A^6 [g(x) - f(x)]\,dx + \int_6^E [f(x) - g(x)]\,dx$$

$$h(x) = g(x) - f(x) = \frac{x^3}{8} - x^2 - 2x + 21$$

$$H(x) = \int h(x)\,dx = \frac{x^4}{32} - \frac{x^3}{3} - x^2 + 21x$$

$$\text{Area} = [H(6) - H(A)] + [H(6) - H(E)]$$

$$= 2H(6) - H(A) - H(E)$$

$$\left[2\left(\frac{(6)^4}{32} - \frac{(6)^3}{3} - (6)^2 + 21(6)\right)\right] - \left[\left(\frac{(1-\sqrt{29})^4}{32} - \frac{(1-\sqrt{29})^3}{3} - (1-\sqrt{29})^2 + 21(1-\sqrt{29})\right)\right]$$

$$- \left[\left(\frac{(1+\sqrt{29})^4}{32} - \frac{(1+\sqrt{29})^3}{3} - (1+\sqrt{29})^2 + 21(1+\sqrt{29})\right)\right]$$

$$= \frac{781 + 145\sqrt{29}}{12} = 130.1540748$$

Part 3: Simpson's Rule

Single Simpson's step ove $[A, E]$ midpoint $M = \frac{A+E}{2} = 1$ then:

$$\text{Area}_s \approx \frac{E-A}{6}\left[h(A) + 4h(1) + h(E)\right]$$

But $hA = g(A) - fA) = 0$ and $h(E) = 0$, while

$$h(1) = \frac{1^3}{8} - 1^2 - 2(1) + 21 = 0.125 - 1 - 2 + 21$$
$$= 18.125$$

$\therefore$

$$\text{Area}_s = \frac{(1+\sqrt{29}) - (1-\sqrt{29})}{6}(0 + 4(18.125) + 0)$$

$$= \frac{2\sqrt{29}}{6} \times 72.5 = \frac{145\sqrt{29}}{6} \approx 130.141$$

Part 4: Error estimation and Comparsion.

Theoretical Error bound for Simpson's Rule over $[A, E]$ is

$$E_s = -\frac{(E-A)h^4}{180} h^{(4)}(\xi)$$

Since $h(x) = g(x) - f(x)$ is a cubic polynomial, $h^{(4)} \equiv 0$

so Simpson's Rule is exact for the net integral $g - f$.

Comparsion: Calculus area = 130.154
Simpson approx. = 130.141

|Error| : difference = $\underline{130.154 - 130.141} = 0.013$

$$\frac{|\text{Error}|}{\text{actual}} \times 100 = \%\text{ error} \longrightarrow \frac{0.013}{130.154} \times 100 = \underbrace{0.01\ \%}_{\text{excellent agreement}}$$

Because the true area requires taking the absolute value of $g-f$ (which breaks the integrand into two cubic pieces), Simpson's rule no longer intergrates that exactly, but even so the single-step estimate errors by ~ 0.013 units.

Question 4: continued

```
# --- 1. Trapezoidal Rule ---
# Using 3 data ordinates means 2 intervals
n_trapezoid = 2
x_trapezoid = np.linspace(a, b, n_trapezoid + 1)
y_trapezoid = f(x_trapezoid)
h_trapezoid = (b - a) / n_trapezoid

# Trapezoidal rule formula
trapezoidal_approximation = (h_trapezoid / 2) * (y_trapezoid[0] + 2*np.
 ↪sum(y_trapezoid[1:-1]) + y_trapezoid[-1])

# --- 2. Simpson's Rule ---
# Using 3 data ordinates means 2 intervals
n_simpson = 2
x_simpson = np.linspace(a, b, n_simpson + 1)
y_simpson = f(x_simpson)
h_simpson = (b - a) / n_simpson

# Simpson's rule formula
simpson_approximation = (h_simpson / 3) * (y_simpson[0] + 4*np.sum(y_simpson[1:
 ↪-1:2]) + 2*np.sum(y_simpson[2:-2:2]) + y_simpson[-1])


# --- 3. Two-Point Gauss Quadrature ---

gauss_points = np.array([-1/np.sqrt(3), 1/np.sqrt(3)])
gauss_weights = np.array([1, 1])

# Transformation from [-1, 1] to [a, b]
x_gauss = ((b - a) / 2) * gauss_points + ((a + b) / 2)
y_gauss = f(x_gauss)

# Gauss Quadrature formula
gauss_approximation = ((b - a) / 2) * np.sum(gauss_weights * y_gauss)

# --- Error Calculations ---

# Absolute Error = |Approximate Value - Exact Value|
abs_error_trapezoid = np.abs(trapezoidal_approximation - exact_solution)
abs_error_simpson = np.abs(simpson_approximation - exact_solution)
abs_error_gauss = np.abs(gauss_approximation - exact_solution)

# Relative Error = |Absolute Error / Exact Value|
rel_error_trapezoid = abs_error_trapezoid / np.abs(exact_solution)
rel_error_simpson = abs_error_simpson / np.abs(exact_solution)
rel_error_gauss = abs_error_gauss / np.abs(exact_solution)
```

```python

# --- Output ---

print("Numerical Integration Results:\n")
print(f"Trapezoidal Rule Approximation: {trapezoidal_approximation}")
print(f"Simpson's Rule Approximation: {simpson_approximation}")
print(f"Gauss Quadrature Approximation: {gauss_approximation}\n")

print("Error Analysis:\n")
print(f"Trapezoidal Rule Absolute Error: {abs_error_trapezoid}")
print(f"Trapezoidal Rule Relative Error: {rel_error_trapezoid}")
print(f"Simpson's Rule Absolute Error: {abs_error_simpson}")
print(f"Simpson's Rule Relative Error: {rel_error_simpson}")
print(f"Gauss Quadrature Absolute Error: {abs_error_gauss}")
print(f"Gauss Quadrature Relative Error: {rel_error_gauss}")
```

```
Numerical Integration Results:

Trapezoidal Rule Approximation: 74.0
Simpson's Rule Approximation: 57.33333333333333
Gauss Quadrature Approximation: 55.11111111111111

Error Analysis:

Trapezoidal Rule Absolute Error: 18.0
Trapezoidal Rule Relative Error: 0.32142857142857145
Simpson's Rule Absolute Error: 1.3333333333333286
Simpson's Rule Relative Error: 0.023809523809523725
Gauss Quadrature Absolute Error: 0.8888888888888928
Gauss Quadrature Relative Error: 0.015873015873015945
```

Question 5

[5]:
```python
import numpy as np
import pandas as pd

def f(x):
    """The function to integrate: 4/(1+x^2)"""
    return 4 / (1 + x**2)

def trapezoid(f, a, b, n):
    """
    Computes the trapezoid rule approximation with n subintervals
    """
    h = (b - a) / n
    x = np.linspace(a, b, n+1)
    y = [f(xi) for xi in x]

    # Trapezoid rule formula: h/2 * (f(a) + 2f(x ) + 2f(x ) + ... + 2f(x  ) +␣
 ↪f(b))
    return h/2 * (y[0] + 2*sum(y[1:-1]) + y[-1])

def romberg(f, a, b, max_level=4):
    """
    Performs Romberg integration
    Returns R matrix and errors
    """
    R = np.zeros((max_level+1, max_level+1))
    h = b - a

    # First column (R(i,0)) - trapezoid rule with h, h/2, h/4, ...
    for i in range(max_level+1):
        n = 2**i
        R[i, 0] = trapezoid(f, a, b, n)

    # Fill the Romberg table using the extrapolation formula
    for j in range(1, max_level+1):
        for i in range(j, max_level+1):
            # Romberg extrapolation formula
            R[i, j] = R[i, j-1] + (R[i, j-1] - R[i-1, j-1]) / (4**j - 1)

    # True value of the integral is
    true_value = np.pi
    abs_errors = np.abs(R - true_value)
    rel_errors = abs_errors / true_value

    return R, abs_errors, rel_errors

def print_results(R, abs_errors, rel_errors, max_level):
    """Prints Romberg table and errors in nicely formatted tables"""
    print("Romberg Integration Table:")
    rows = []

    for i in range(max_level+1):
        row = [f"R({i},{j})" for j in range(i+1)]
        rows.append(row)

    df_labels = pd.DataFrame(rows)

    # Create table of values
    rows_values = []
    for i in range(max_level+1):
        row = [f"{R[i,j]:.10f}" for j in range(i+1)]
        rows_values.append(row)
    df_values = pd.DataFrame(rows_values)

    # Print label and value
    for i in range(max_level+1):
        line = ""
        for j in range(i+1):
            line += f"{df_labels.iloc[i,j]}: {df_values.iloc[i,j]}  "
        print(line)

    print("\nAbsolute Errors:")
    rows_abs_err = []
    for i in range(max_level+1):
        row = [f"{abs_errors[i,j]:.10f}" for j in range(i+1)]
        rows_abs_err.append(row)

    df_abs_err = pd.DataFrame(rows_abs_err)

    for i in range(max_level+1):
        line = ""
        for j in range(i+1):
            line += f"E({i},{j}): {df_abs_err.iloc[i,j]}  "
        print(line)

    print("\nRelative Errors (%):")
    rows_rel_err = []
    for i in range(max_level+1):
        row = [f"{rel_errors[i,j]*100:.8f}%" for j in range(i+1)]
        rows_rel_err.append(row)

    df_rel_err = pd.DataFrame(rows_rel_err)

    for i in range(max_level+1):
        line = ""
        for j in range(i+1):
            line += f"RE({i},{j}): {df_rel_err.iloc[i,j]}  "
        print(line)

# Main execution
max_level = 4  # Number of levels in Romberg table
a, b = 0, 1    # Integration limits
R, abs_errors, rel_errors = romberg(f, a, b, max_level)

print(f"True value =  = {np.pi:.10f}")
print_results(R, abs_errors, rel_errors, max_level)

# Specifically evaluate at the points mentioned in the problem
print("\nEvaluating f(x) at specified points:")
```

```
points = [0, 1/4, 1/2, 3/4, 1]
for x in points:
    print(f"f({x}) = {f(x):.10f}")

# To show how starting with those specific points works in Romberg integration:
print("\nRomberg integration starting with points [0, 1/4, 1/2, 3/4, 1]:")
# This means we start with n=4 intervals
h = 1/4
x_values = np.array([0, 1/4, 1/2, 3/4, 1])
y_values = f(x_values)

# Calculate initial trapezoid rule approximation with these points
R0 = h/2 * (y_values[0] + 2*sum(y_values[1:-1]) + y_values[-1])
print(f"Initial trapezoid approximation (h=1/4): {R0:.10f}")
print(f"Absolute error: {abs(R0 - np.pi):.10f}")
print(f"Relative error: {abs(R0 - np.pi)/np.pi*100:.8f}%")
```

```
True value =  = 3.1415926536
Romberg Integration Table:
R(0,0): 3.0000000000
R(1,0): 3.1000000000  R(1,1): 3.1333333333
R(2,0): 3.1311764706  R(2,1): 3.1415686275  R(2,2): 3.1421176471
R(3,0): 3.1389884945  R(3,1): 3.1415925025  R(3,2): 3.1415940941  R(3,3):
3.1415857838
R(4,0): 3.1409416120  R(4,1): 3.1415926512  R(4,2): 3.1415926611  R(4,3):
3.1415926384  R(4,4): 3.1415926653

Absolute Errors:
E(0,0): 0.1415926536
E(1,0): 0.0415926536  E(1,1): 0.0082593203
E(2,0): 0.0104161830  E(2,1): 0.0000240261  E(2,2): 0.0005249935
E(3,0): 0.0026041591  E(3,1): 0.0000001511  E(3,2): 0.0000014405  E(3,3):
0.0000068698
E(4,0): 0.0006510415  E(4,1): 0.0000000024  E(4,2): 0.0000000076  E(4,3):
0.0000000152  E(4,4): 0.0000000117

Relative Errors (%):
RE(0,0): 4.50703414%
RE(1,0): 1.32393528%  RE(1,1): 0.26290233%
RE(2,0): 0.33155740%  RE(2,1): 0.00076478%  RE(2,2): 0.01671106%
RE(3,0): 0.08289296%  RE(3,1): 0.00000481%  RE(3,2): 0.00004585%  RE(3,3):
0.00021867%
RE(4,0): 0.02072330%  RE(4,1): 0.00000008%  RE(4,2): 0.00000024%  RE(4,3):
0.00000048%  RE(4,4): 0.00000037%

Evaluating f(x) at specified points:
f(0) = 4.0000000000
f(0.25) = 3.7647058824
```

```
f(0.5) = 3.2000000000
f(0.75) = 2.5600000000
f(1) = 2.0000000000

Romberg integration starting with points [0, 1/4, 1/2, 3/4, 1]:
Initial trapezoid approximation (h=1/4): 3.1311764706
Absolute error: 0.0104161830
Relative error: 0.33155740%
```

```
[ ]:
```

Question 6 : Part 1 : Romberg integration to $O(h^6)$

$R_{i,0} = T_{2^i}$ (composite trapezoid with 2^i panels)

$$R_{ij} = R_{i,j} + \frac{R_{i,j-1} - R_{i-1,j-1}}{4^j - 1}$$

Let $f(x) = x^3(16 - x^2)$.

$f(0) = 0$, $f(1) = 1^3(16-1) = 15$, $f(2) = 8(16-4) = 96$, $f(3) = 27(16-9) = 189$

$f(4) = 64(16-16) = 0$

a) $R_{0,0} = T_1$ (one panel, $h=4$)

$$T_1 = \frac{h}{2}\left[f(0) + f(4)\right] = \frac{4}{2}(0+0) = 0$$

b) $R\,1,0 = T_2$ (two panels, $h=2$) $x = 0, 2, 4$:

$$T_2 = \frac{2}{2}\left[f(0) + 2f(2) + f(4)\right] = 1 \cdot (0 + 2 \cdot 96 + 0) = 192$$

c) $R_{2,0} = T_4$ (four panels, $h=1$) $x = 0, 1, 2, 3, 4$:

$$T_4 = \frac{1}{2}\left[f(0) + 2\left[f(1) + f(2) + f(3) + f(4)\right]\right]$$

$$= \frac{1}{2}\left(0 + 2(15 + 96 + 189) + 0\right) = \frac{1}{2} \cdot 600 = 300$$

Richardson extrapolation

1. Level 1 , column 1

$$R_{1,1} = R_{1,0} + \frac{R_{1,0} - R_{0,0}}{4^1 - 1} = 192 + \frac{192 - 0}{3} = 192 + 64 = 256$$

2. Level 2, column 1 :

$$R_{2,1} = 300 + \frac{300 - 192}{3} = 300 + 36 = 336$$

3. Level 2, column2 ($O(h^6)$):

$$R_{2,2} = R_{2,1} + \frac{R_{2,1} - R_{1,1}}{4^2 - 1} = 336 + \frac{336 - 256}{15} = 336 + \frac{18}{15}$$

$$= 336 + 5.\overline{33}$$

$$= 341.\overline{33}$$

Romberg $O(h^6)$ estimate :

$$R_{2,2} = 341.33333 = \frac{1024}{3}$$

Part 2: Three-point Gauss quadrature.

a) Change of variable

Let

$$x = 2t + 2 \quad , \quad dx = 2dt$$

so

$$\int_0^4 f(x)\,dx = \int_{-1}^{1} f(2t+2)\,2dt$$

b) nodes and Wheights

i	t_i	w_i	$x_i = 2t_i + 2$
1	0	8/9	2
2	$-\sqrt{\frac{3}{5}}$	5/9	$2 - 2\sqrt{\frac{3}{5}}$
3	$+\sqrt{\frac{3}{5}}$	5/9	$2 + 2\sqrt{\frac{3}{5}}$

Numerically $\sqrt{3/5} \approx 0.7746$ so

$$x_2 \approx 2 - 1.54919 = 0.4508067$$

$$x_3 \approx 2 + 1.54919 = 3.5491933$$

c) Evaluat f at x_i:

$$f(2) = 2^3(16-4) = 8 \cdot 12 = 96$$

$$f(0.4508) \approx (0.4508)^3(16-(0.4508)^2) \approx 1.44723$$

$$f(3.5492) \approx (3.5492)^3(16-(3.5492)^2) \approx 152.152$$

d)

$$I \approx 2\sum_{i=1}^{3} w_i f(x_i)$$

$$= 2\left(\frac{8}{9}\cdot 96 + \frac{5}{9}\left(f(0.4508) + f(3.5492)\right)\right)$$

$$= 2\left(\frac{768}{9} + \frac{5}{9}(1.447236 + 152.152764)\right)$$

$$= 2\left(85.3333 + \frac{5}{9}\cdot 153.60000\right)$$

$$= 2(85.3333 + 85.33333)$$

$$= 341.\overline{33333}$$

$$= \frac{1024}{3}$$

Both methods we estimate:

- Romberg $(O(h^6))$: $R_{2,2} = 341.3333333$
- Three-Point Gauss : 341.3333333