# MathExam2

April 26, 2024

# 1 1

- Given points:
  - $x_0 = -1, y_0 = -1$
  - $x_1 = 0, y_1 = 2$
  - $x_2 = 1, y_2 = -2$
  - $x_3 = 2, y_3 = 0$

## 1.1 a

- Using Lagrange method

### 1.1.1 Construct Each Lagrange Basis Polynomial:

1. **$L_0(x)$:**

$$L_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} = \frac{(x-0)(x-1)(x-2)}{(-1-0)(-1-1)(-1-2)} = \frac{x(x-1)(x-2)}{-6}$$

2. **$L_1(x)$:**

$$L_1(x) = \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} = \frac{(x+1)(x-1)(x-2)}{(0+1)(0-1)(0-2)} = \frac{(x+1)(x-1)(x-2)}{2}$$

3. **$L_2(x)$:**

$$L_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} = \frac{(x+1)x(x-2)}{(1+1)(1-0)(1-2)} = \frac{(x+1)x(x-2)}{-2}$$

4. **$L_3(x)$:**

$$L_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} = \frac{(x+1)x(x-1)}{(2+1)(2-0)(2-1)} = \frac{(x+1)x(x-1)}{6}$$

### 1.1.2 Construct the Interpolating Polynomial $P_3(x)$:

$$P_3(x) = y_0L_0(x) + y_1L_1(x) + y_2L_2(x) + y_3L_3(x)$$

$$P_3(x) = -1 \cdot \frac{x(x-1)(x-2)}{6} + 2 \cdot \frac{(x+1)(x-1)(x-2)}{2} - 2 \cdot \frac{(x+1)x(x-2)}{-2} + 0 \cdot \frac{(x+1)x(x-1)}{6}$$

### 1.1.3 Simplify $P_3(x)$:

$$P_3(x) = \frac{13x^3}{6} - \frac{7x^2}{2} - \frac{8x}{3} + 2$$

## 1.2 b

- Using Newton's divided difference method

### 1.2.1 Given Data Points

- $x_0 = -1, y_0 = -1$
- $x_1 = 0, y_1 = 2$
- $x_2 = 1, y_2 = -2$
- $x_3 = 2, y_3 = 0$

### 1.2.2 Step 1: Calculate First-Order Divided Differences

$$f[x_0, x_1] = \frac{y_1 - y_0}{x_1 - x_0} = \frac{2 - (-1)}{0 - (-1)} = \frac{3}{1} = 3$$

$$f[x_1, x_2] = \frac{y_2 - y_1}{x_2 - x_1} = \frac{-2 - 2}{1 - 0} = \frac{-4}{1} = -4$$

$$f[x_2, x_3] = \frac{y_3 - y_2}{x_3 - x_2} = \frac{0 - (-2)}{2 - 1} = \frac{2}{1} = 2$$

### 1.2.3 Step 2: Calculate Second-Order Divided Differences

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{-4 - 3}{1 - (-1)} = \frac{-7}{2} = -3.5$$

$$f[x_1, x_2, x_3] = \frac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1} = \frac{2 - (-4)}{2 - 0} = \frac{6}{2} = 3$$

### 1.2.4 Step 3: Calculate Third-Order Divided Difference

$$f[x_0, x_1, x_2, x_3] = \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0} = \frac{3 - (-3.5)}{2 - (-1)} = \frac{6.5}{3} = \frac{13}{6}$$

### 1.2.5 Construct the Polynomial

Using these divided differences, the Newton form of the interpolating polynomial $P_3(x)$ is:

$$P_3(x) = f[x_0] + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] + (x - x_0)(x - x_1)(x - x_2) f[x_0, x_1, x_2, x_3]$$

$$P_3(x) = -1 + 3(x + 1) - \frac{7}{2}(x + 1)x + \frac{13}{6}(x + 1)x(x - 1)$$

### 1.2.6 Simplified Polynomial

$$P_3(x) = \frac{13x^3}{6} - \frac{7x^2}{2} - \frac{8x}{3} + 2$$

## 1.3 c

- Using a system of linear equations and solving matrix style using Row Echelon Form

```
-1 1 -1 1 -1
0 0 0 1 2
1 1 1 1 -2
8 4 2 1 0

-1 1 -1 1 -1
1 1 1 1 -2
8 4 2 1 0
0 0 0 1 2

-1 1 -1 1 -1
1 1 1 1 -2
0 -4 -6 -7 16
0 0 0 1 2


1 1 1 1 -2
0 2 0 2 -3
0 -4 -6 -7 16
0 0 0 1 2

1 1 1 0 -4
0 2 0 0 -7
0 0 -6 0 16
0 0 0 1 2

1 0 0 0 (-4 + (16/6) + (7/2))
0 1 0 0 (-7/2)
0 0 1 0 (-8/3)
0 0 0 1 2


1 0 0 0 (13/6)
0 1 0 0 (-7/2)
0 0 1 0 (-8/3)
0 0 0 1 2
```

### 1.3.1 Simplified Polynomial

$$P_3(x) = \frac{13x^3}{6} - \frac{7x^2}{2} - \frac{8x}{3} + 2$$

## 1.4 discussion

- The interpolating polynomial $P_3(x)$ is the same for all three methods.
- The polynomial is:

- $P_3(x) = \frac{13x^3}{6} - \frac{7x^2}{2} - \frac{8x}{3} + 2$
- I find it hard to say that one of these methods is better than the others. They all have interesting properties that could enable optimization or access to important properties in different scenarios.
- The divided difference method is very useful for calculating the polynomial in a recursive manner and if we cache the divided differences we can significantly improve the performance when calculating higher order polynomials.
- The Lagrange method is very useful for understanding the polynomial and its properties, and it is very easy to implement.
- The matrix method is very useful for solving the system of linear equations and it is very easy to implement as well. Gaussian elimination as well as other methods are very well studied and optimized in computer science.
- In conclusion, all three methods are useful and have their own advantages and disadvantages.

# 2 2

- Given x and cos(x)
  - $x = 0, cos(x) = 1$
  - $x = \frac{\pi}{6}, cos(x) = \frac{\sqrt{3}}{2}$
  - $x = \frac{\pi}{4}, cos(x) = \frac{1}{\sqrt{2}}$
  - $x = \frac{\pi}{3}, cos(x) = \frac{1}{2}$
  - $x = \frac{\pi}{2}, cos(x) = 0$

## 2.1 Construct the Interpolating Polynomial $P_4(x)$ using Newton Divided Differences Interpolation Method

### 2.1.1 Step 1:

We will calculate divided differences up to fourth order using python sympy

### 2.1.2 Step 2:

Using the divided differences, we will construct the interpolating polynomial $P_4(x)$ then use it to approximate $cos(\frac{\pi}{7})$. We can also do this in python.

### 2.1.3 Step 3:

We will caculate the error bound

$$f(x) - P_n(x) = \frac{(x - x_0)(x - x_1)...(x - x_n)}{(n+1)!} f^{(n+1)}(c_x)$$

- We know that when $n + 1 = 5$ in our case then $f^{(5)}(x) = sin(x)$ and $|sin(x)| \leq 1$ for all $x$ in the interval $[0, \frac{\pi}{2}]$
- Therefore we can derive the error bound based on that information.

### 2.1.4 Calculating the Divided Differences

- We will use the following divided differences to construct the interpolating polynomial $P_4(x)$

**First-Order Divided Differences**

$$\frac{6\left(-1+\frac{\sqrt{3}}{2}\right)}{\pi}$$

**Simplified First-Order Divided Differences**

$$\frac{3\left(-2+\sqrt{3}\right)}{\pi}$$

**Second-Order Divided Differences**

$$\frac{4\cdot\left(\frac{12\left(-\frac{\sqrt{3}}{2}+\frac{\sqrt{2}}{2}\right)}{\pi}-\frac{6\left(-1+\frac{\sqrt{3}}{2}\right)}{\pi}\right)}{\pi}$$

**Simplified Second-Order Divided Differences**

$$\frac{12\left(-3\sqrt{3}+2+2\sqrt{2}\right)}{\pi^2}$$

**Third-Order Divided Differences**

$$\frac{3\cdot\left(\frac{6\cdot\left(\frac{12\cdot\left(\frac{1}{2}-\frac{\sqrt{2}}{2}\right)}{\pi}-\frac{12\left(-\frac{\sqrt{3}}{2}+\frac{\sqrt{2}}{2}\right)}{\pi}\right)}{\pi}-\frac{4\cdot\left(\frac{12\left(-\frac{\sqrt{3}}{2}+\frac{\sqrt{2}}{2}\right)}{\pi}-\frac{6\left(-1+\frac{\sqrt{3}}{2}\right)}{\pi}\right)}{\pi}\right)}{\pi}$$

**Simplified Third-Order Divided Differences**

$$\frac{36\left(-8\sqrt{2}+1+6\sqrt{3}\right)}{\pi^3}$$

**Fourth-Order Divided Differences**

$$\frac{2\left(-\frac{3\cdot\left(\frac{6\cdot\left(\frac{12\cdot\left(\frac{1}{2}-\frac{\sqrt{2}}{2}\right)}{\pi}-\frac{12\left(-\frac{\sqrt{3}}{2}+\frac{\sqrt{2}}{2}\right)}{\pi}\right)}{\pi}-\frac{4\cdot\left(\frac{12\left(-\frac{\sqrt{3}}{2}+\frac{\sqrt{2}}{2}\right)}{\pi}-\frac{6\left(-1+\frac{\sqrt{3}}{2}\right)}{\pi}\right)}{\pi}\right)}{\pi}+\frac{3\cdot\left(\frac{4\left(-\frac{3}{\pi}-\frac{12\cdot\left(\frac{1}{2}-\frac{\sqrt{2}}{2}\right)}{\pi}\right)}{\pi}-\frac{6\cdot\left(\frac{12\cdot\left(\frac{1}{2}-\frac{\sqrt{2}}{2}\right)}{\pi}-\frac{12\left(-\frac{\sqrt{3}}{2}+\frac{\sqrt{2}}{2}\right)}{\pi}\right)}{\pi}\right)}{\pi}\right)}{\pi}$$

**Simplified Fourth-Order Divided Differences**

$$\frac{72\left(-9\sqrt{3}-7+16\sqrt{2}\right)}{\pi^4}$$

### 2.1.5 Constructing $P_4(x)$

- Sorry, I had to break it into terms because it would not fit on my page without going off screen.
- I originally constructed it as one equation but it was too long.

1. The first term:

$$T_1(x) = x\left(x - \frac{\pi}{3}\right)\left(x - \frac{\pi}{4}\right)\left(x - \frac{\pi}{6}\right)\left(-\sqrt{3} + \frac{\sqrt{2}}{2} + 1\right)$$

2. The second term:

$$T_2(x) = \frac{6x\left(x - \frac{\pi}{4}\right)\left(x - \frac{\pi}{6}\right)\left(-\frac{3\sqrt{2}}{2} - \frac{1}{2} + \frac{3\sqrt{3}}{2}\right)}{\pi}$$

3. The third term:

$$T_3(x) = \frac{12x\left(x - \frac{\pi}{6}\right)\left(\frac{4\left(-\frac{\sqrt{3}}{2} + \frac{\sqrt{2}}{2}\right)}{\pi} + \frac{12\left(-\frac{3\sqrt{2}}{2} - \frac{1}{2} + \frac{3\sqrt{3}}{2}\right)}{\pi}\right)}{\pi}$$

4. The fourth term and constant:

$$T_4(x) = \frac{6x\left(-\sqrt{3} + \frac{\sqrt{2}}{2} + 1\right)}{\pi} + 1$$

Then, combine these to form $P_4(x)$:

$$P_4(x) = T_1(x) + T_2(x) + T_3(x) + T_4(x)$$

I suspect I completely messed this up in my code hence the extremely large equation.

# 3 3

## 3.1 trapezoidal rule

$$Ei(2) = \int_1^2 \frac{e^{-t}}{t} dt$$

$$h = \frac{1}{8}$$

- $x = 1$ to $x = 2$
  - $x_0 = 1$
  - $x_1 = 1.125$
  - $x_2 = 1.25$
  - $x_3 = 1.375$
  - $x_4 = 1.5$
  - $x_5 = 1.625$
  - $x_6 = 1.75$

- $x_7 = 1.875$
- $x_8 = 2$

$$f(x_i) = \frac{e^{-x_i}}{x_i}$$

$$\int_1^2 \frac{e^{-t}}{t} dt \approx \frac{1/8}{2} \left[ f(x_0) + 2f(x_1) + 2f(x_2) + 2f(x_3) + 2f(x_4) + 2f(x_5) + 2f(x_6) + 2f(x_7) + f(x_8) \right]$$

We can calculate this using Python for precise decimal values of $f(x_i)$.

$$= 0.171307407473616 25$$

### 3.2 simpson's rule

- $x_0 = 1$, $x_1 = 1.125$, ..., $x_8 = 2$ with $h = \frac{1}{8}$
- Function values calculated as $f(x_i) = \frac{e^{-x_i}}{x_i}$

Simpson's sum is:

$$\int_1^2 \frac{e^{-t}}{t} dt \approx \frac{1/8}{3} \left[ f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + 4f(x_5) + 2f(x_6) + 4f(x_7) + f(x_8) \right]$$

And in python...

$$= 0.170490692028207$$

[ ]:
```python
import numpy as np
from sympy import pi, sqrt, cos, symbols, simplify, factorial, sin, Abs,
 ↪Rational, latex, prod, Sum



def recursive_divided_difference(x_vals, y_vals):
    if len(y_vals) == 1:
        return y_vals[0]
    else:
        new_y_vals = []
        for i in range(1, len(y_vals)):
            # Calculate each divided difference level
            new_y = (y_vals[i] - y_vals[i-1]) / (x_vals[i] -
 ↪x_vals[i-len(y_vals)+1])
            new_y_vals.append(new_y)
        return recursive_divided_difference(x_vals[1:], new_y_vals)

def newton_interpolation(x_vals, y_vals, x_sym):
```

```python
    coefficients = []
    temp_y_vals = y_vals.copy()

    for i in range(len(x_vals)):
        coeff = recursive_divided_difference(x_vals[:i+1], temp_y_vals[:i+1])
        coefficients.append(coeff)
        # Prepare the next level of base y-values for divided differences
        for j in range(1, len(temp_y_vals)-i):
            temp_y_vals[j-1] = temp_y_vals[j] - temp_y_vals[j-1]
        temp_y_vals.pop()

    # Building the polynomial from the coefficients
    polynomial = 0
    for i, coeff in enumerate(coefficients):
        term = coeff
        for j in range(i):
            term *= (x_sym - x_vals[j])
        polynomial += term

    return polynomial




# Example usage
x = symbols('x')
points_example_degree_4 = {
        0: 1,
        pi/ 6: sqrt(3)/2,
        pi/4: 1/sqrt(2),
        pi/3: Rational(1,2),
        pi/2: 0
}

points_example_degree_3 = {
        0: 1,
        pi/ 6: sqrt(3)/2,
        pi/4: 1/sqrt(2),
        pi/3: Rational(1,2)
}

points_example_degree_2 = {
        0: 1,
        pi/ 6: sqrt(3)/2,
        pi/4: 1/sqrt(2)
}
```

```python
points_example_degree_1 = {
        0: 1,
        pi/ 6: sqrt(3)/2
}

DD_1 = recursive_divided_difference(list(points_example_degree_1.keys()),␣
 ↪list(points_example_degree_1.values()))
DD_2 = recursive_divided_difference(list(points_example_degree_2.keys()),␣
 ↪list(points_example_degree_2.values()))
DD_3 = recursive_divided_difference(list(points_example_degree_3.keys()),␣
 ↪list(points_example_degree_3.values()))
DD_4 = recursive_divided_difference(list(points_example_degree_4.keys()),␣
 ↪list(points_example_degree_4.values()))


DDs = [DD_1, DD_2, DD_3, DD_4]

for i in DDs:
        print(latex(i))
        print(latex(simplify(i)))
```

```
\frac{6 \left(-1 + \frac{\sqrt{3}}{2}\right)}{\pi}
\frac{3 \left(-2 + \sqrt{3}\right)}{\pi}
\frac{12 \cdot \left(\frac{12 \left(-1 + \frac{\sqrt{3}}{2}\right)}{\pi} +
\frac{4 \left(- \frac{\sqrt{3}}{2} +
\frac{\sqrt{2}}{2}\right)}{\pi}\right)}{\pi}
\frac{24 \left(-6 + \sqrt{2} + 2 \sqrt{3}\right)}{\pi^{2}}
\frac{12 \cdot \left(\frac{6 \cdot \left(\frac{12 \left(- \frac{\sqrt{3}}{2} +
\frac{\sqrt{2}}{2}\right)}{\pi} + \frac{3 \cdot \left(\frac{1}{2} -
\frac{\sqrt{2}}{2}\right)}{\pi}\right)}{\pi} + \frac{12 \cdot \left(\frac{12
\left(-1 + \frac{\sqrt{3}}{2}\right)}{\pi} - \frac{12 \left(- \frac{\sqrt{3}}{2}
+ \frac{\sqrt{2}}{2}\right)}{\pi}\right)}{\pi}\right)}{\pi}
\frac{108 \left(-15 - 5 \sqrt{2} + 12 \sqrt{3}\right)}{\pi^{3}}
\frac{6 \cdot \left(\frac{4 \cdot \left(\frac{3 \cdot \left(\frac{6 \cdot
\left(\frac{1}{2} - \frac{\sqrt{2}}{2}\right)}{\pi} - \frac{1}{\pi}\right)}{\pi}
+ \frac{6 \cdot \left(\frac{12 \left(- \frac{\sqrt{3}}{2} +
\frac{\sqrt{2}}{2}\right)}{\pi} - \frac{6 \cdot \left(\frac{1}{2} -
\frac{\sqrt{2}}{2}\right)}{\pi}\right)}{\pi}\right)}{\pi} + \frac{6 \cdot
\left(\frac{12 \cdot \left(\frac{12 \left(-1 + \frac{\sqrt{3}}{2}\right)}{\pi} -
\frac{12 \left(- \frac{\sqrt{3}}{2} +
\frac{\sqrt{2}}{2}\right)}{\pi}\right)}{\pi} - \frac{6 \cdot \left(\frac{12
\left(- \frac{\sqrt{3}}{2} + \frac{\sqrt{2}}{2}\right)}{\pi} - \frac{6 \cdot
\left(\frac{1}{2} -
\frac{\sqrt{2}}{2}\right)}{\pi}\right)}{\pi}\right)}{\pi}\right)}{\pi}
\frac{72 \left(- 48 \sqrt{2} - 67 + 78 \sqrt{3}\right)}{\pi^{4}}
```

[ ]:
```python
P_4 = newton_interpolation(list(points_example_degree_4.keys()),
    list(points_example_degree_4.values()), x)

print(latex(P_4))
```

```
x \left(x - \frac{\pi}{3}\right) \left(x - \frac{\pi}{4}\right) \left(x -
\frac{\pi}{6}\right) \left(- \sqrt{3} + \frac{\sqrt{2}}{2} + 1\right) + \frac{6
x \left(x - \frac{\pi}{4}\right) \left(x - \frac{\pi}{6}\right) \left(- \frac{3
\sqrt{2}}{2} - \frac{1}{2} + \frac{3 \sqrt{3}}{2}\right)}{\pi} + \frac{12 x
\left(x - \frac{\pi}{6}\right) \left(\frac{4 \left(- \frac{\sqrt{3}}{2} +
\frac{\sqrt{2}}{2}\right)}{\pi} + \frac{12 \left(- \frac{3 \sqrt{2}}{2} -
\frac{1}{2} + \frac{3 \sqrt{3}}{2}\right)}{\pi}\right)}{\pi} + \frac{6 x \left(-
\sqrt{3} + \frac{\sqrt{2}}{2} + 1\right)}{\pi} + 1
```

[ ]:
```python
print(latex(simplify(P_4)))
```

```
\frac{- 576 x \left(6 x - \pi\right) \left(- 8 \sqrt{3} + 3 + 8 \sqrt{2}\right)
+ 18 \pi x \left(- \left(4 x - \pi\right) \left(6 x - \pi\right) \left(- 3
\sqrt{3} + 1 + 3 \sqrt{2}\right) - 48 \sqrt{3} + 24 \sqrt{2} + 48\right) +
\pi^{2} \left(x \left(3 x - \pi\right) \left(4 x - \pi\right) \left(6 x -
\pi\right) \left(- 2 \sqrt{3} + \sqrt{2} + 2\right) + 144\right)}{144 \pi^{2}}
```

[ ]:
```python
import numpy as np

# Define the function
def f(t):
    return np.exp(-t) / t

# Trapezoidal rule parameters
a = 1
b = 2
n = 8
h = (b - a) / n

# Calculate the x points
x_points = np.linspace(a, b, n+1)

# Calculate the function values at each point
f_values = f(x_points)

# Apply the trapezoidal rule
trapezoidal_approximation = (h/2) * (f_values[0] + 2*sum(f_values[1:-1]) +
    f_values[-1])
trapezoidal_approximation
```

[ ]: 0.17130740747361625

[ ]:
```python
import numpy as np

# Redefine the function and variables for Simpson's rule calculation
def f(t):
    return np.exp(-t) / t

# Simpson's rule parameters
a = 1
b = 2
n = 8
h = (b - a) / n

# Calculate the x points
x_points = np.linspace(a, b, n+1)

# Calculate the function values at each point
f_values = f(x_points)

# Apply Simpson's rule
simpson_approximation = (h/3) * (f_values[0] + 2*sum(f_values[2:-1:2]) +
 ↪4*sum(f_values[1:-1:2]) + f_values[-1])
simpson_approximation
```

[ ]: 0.170490692028207

[ ]: