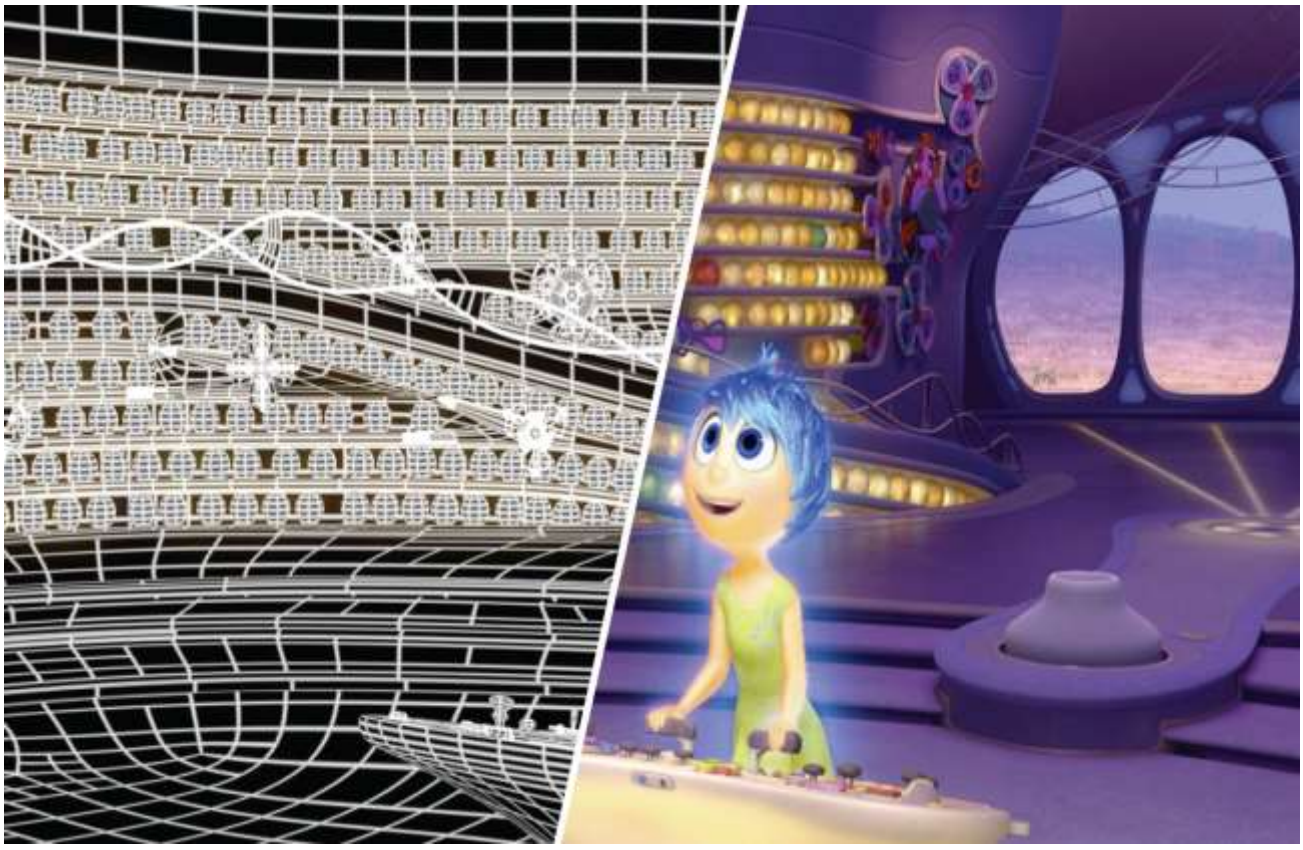
NEWTON DIVIDED DIFFERENCE FORMULA

In our previous lectures, we explored the applications of Lagrange interpolation and examined various software, such as Autodesk Maya, Gazebo, and ROS, that utilize this method for smooth transitions and path planning. However, it's important to note that Newton's divided differences can also serve as an effective interpolation technique in similar contexts. Besides its use in robotics for trajectory generation,

Rendering – Science behind PIXAR



In animated movie rendering, both Lagrange interpolation and Newton's divided differences can be used to generate smooth transitions and curves essential for realistic motion and surface design. For instance, when animators need to create smooth movements of characters or objects, interpolation is applied to generate frames between key positions. Lagrange interpolation helps create smooth motion paths by passing through these keyframes exactly, ensuring fluid transitions. Similarly, Newton's divided differences can be used to generate curves for lighting effects, camera motion, or the shape of surfaces, allowing efficient computation of smooth, continuous changes. These interpolation methods help in rendering realistic animations, blending movements and surfaces seamlessly without sharp, abrupt transitions, thus improving the visual quality of the film.

ADVANTAGE OF NDD OVER LAGRANGE INTERPOLATION

- A practical challenge with Lagrange interpolation is that the effort required to calculate the approximation using a lower-degree polynomial does not reduce the amount of work needed to compute a higher-degree polynomial. Each time the degree increases, the entire interpolation process must be recalculated from scratch, making it computationally inefficient as the degree of the polynomial rises.

MATHEMATICAL REPRESENTATION

x	f(x)	First divided differences	Second divided differences	Third divided differences	Fourth divided difference
x_0	$f[x_0]$				
		$f[x_0, x_1] = A = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$			
x_1	$f[x_1]$		$f[x_0, x_1, x_2] = \frac{B - A}{x_2 - x_0} = E$		
		$f[x_1, x_2] = B = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$		$f[x_0, x_1, x_2, x_3] = \frac{F - E}{x_3 - x_0} = H$	
x_2	$f[x_2]$		$f[x_1, x_2, x_3] = \frac{C - B}{x_3 - x_1} = F$		$f[x_0, x_1, \dots, x_4] = \frac{I - H}{x_4 - x_0} = J$
		$f[x_2, x_3] = C = \frac{f(x_3) - f(x_2)}{x_3 - x_2}$		$f[x_1, x_2, x_3, x_4] = \frac{G - F}{x_4 - x_1} = I$	
x_3	$f[x_3]$		$f[x_2, x_3, x_4] = \frac{D - C}{x_4 - x_2} = G$		
		$f[x_3, x_4] = D = \frac{f(x_4) - f(x_3)}{x_4 - x_3}$			
x_4	$f[x_4]$				

Newton interpolating polynomials

$$P_1(x) = f(x_0) + (x - x_0)A,$$

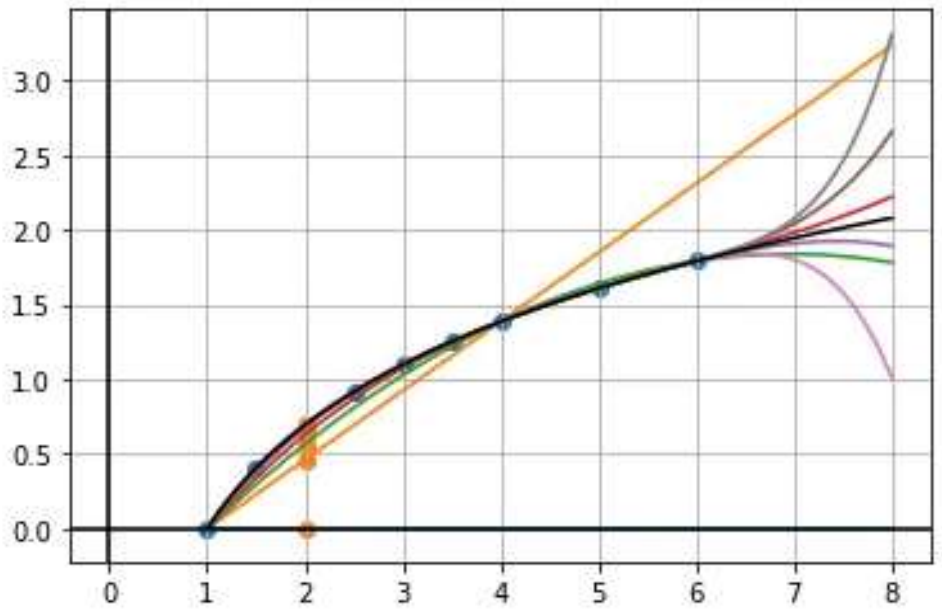
$$P_2(x) = f(x_0) + (x - x_0)A + (x - x_0)(x - x_1)E$$

And $P_3(x) = f(x_0) + (x - x_0)A + (x - x_0)(x - x_1)E + (x - x_0)(x - x_1)(x - x_2)H$

Next, Newton interpolating polynomial of degree four is?

$$P_4(x) = f(x_0) + (x - x_0)A + (x - x_0)(x - x_1)E + (x - x_0)(x - x_1)(x - x_2)H$$
$$+ (x - x_0)(x - x_1)(x - x_2)(x - x_3)J$$

GRAPHICAL REPRESENTATION



Estimated value of ln(2) using Newton's interpolating polynomial

Example

For the given functions $f(x) = \sqrt{1+x}$ and $x_0 = 0, x_1 = 0.6$ and $x_2 = 0.9$. Construct interpolation polynomials of degree one and two to approximate $f(0.45)$, and find the absolute error.

x	f(x)	First divided differences
$x_0 = 0$	$f[x_0] = 1$	
		$f[x_0, x_1] = \frac{\sqrt{1.6} - 1}{0.6 - 0} = 0.44152$
$x_1 = 0.6$	$f[x_1] = \sqrt{1.6}$	

$$P_1(x) = f(x_0) + (x - x_0) f[x_0, x_1] = 1 + 0.44152\, x$$

$$P_1(0.45) = 1.198684$$

x	f(x)	First divided differences	Second divided differences
$x_0 = 0$	$f[x_0] = 1$		
		$f[x_0, x_1] = \frac{\sqrt{1.6} - 1}{0.6 - 0} = 0.44152$	
$x_1 = 0.6$	$f[x_1] = \sqrt{1.6}$		$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = -0.07023$
		$f[x_1, x_2] = \frac{\sqrt{1.9} - \sqrt{1.6}}{0.9 - 0.6} = 0.37831$	
$x_2 = 0.9$	$f[x_2] = \sqrt{1.9}$		

$$P_2(x) = f(x_0) + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2]$$
$$= 1 + 0.44152\,x - 0.07023x(x - 0.6)$$
$$P_2(0.45) = 1.203425$$

We can also use Newton backward divided difference formula as

$$P_2(x) = f(x_2) + (x - x_2)f[x_1, x_2] + (x - x_2)(x - x_1)f[x_0, x_1, x_2]$$
$$= \sqrt{1.9} + 0.37831\,(x - 0.9) - 0.07023(x - 0.9)(x - 0.6)$$
$$P_2(0.45) = 1.203425$$

PROBLEM STATEMENT

In robotics, effective path planning is essential for guiding a robot's movement through various points in its environment. For instance, in autonomous navigation systems, a robot may need to travel through several target positions, such as specific locations in a warehouse or along a route in a complex space. To ensure smooth and precise movement, it is crucial to create a continuous trajectory that minimizes abrupt changes in direction, which can be important for maintaining speed, conserving energy, and performing delicate operations efficiently.

Consider designing a path-planning algorithm for a robot that must follow four target coordinates:

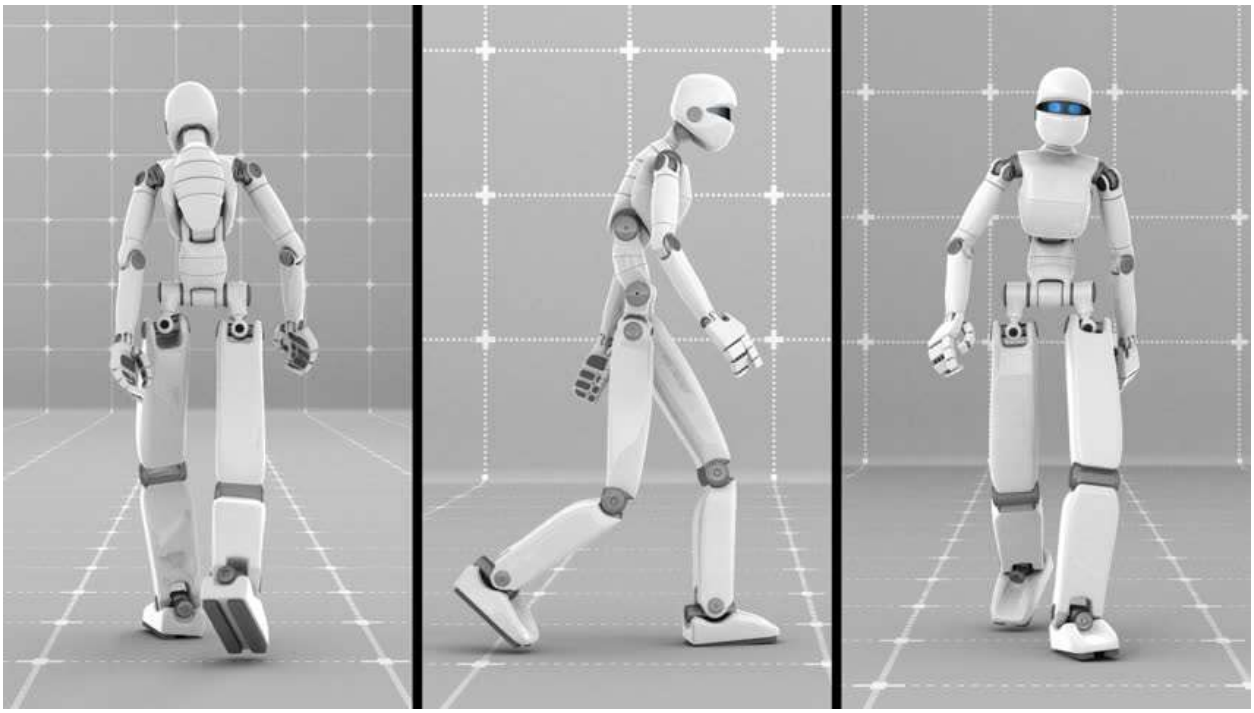
Target 1: (2, 4)

Target 2: (5, 1)

Target 3: (6, 4)

Target 4: (7, 10)

Using Lagrange interpolation, determine the cubic polynomial (degree 3) that interpolates through these points. Also, solve the same interpolation problem using Newton's divided difference formula and compare the results of both methods. Pay particular attention to their accuracy and error analysis to assess which method provides a more reliable trajectory for the robot.



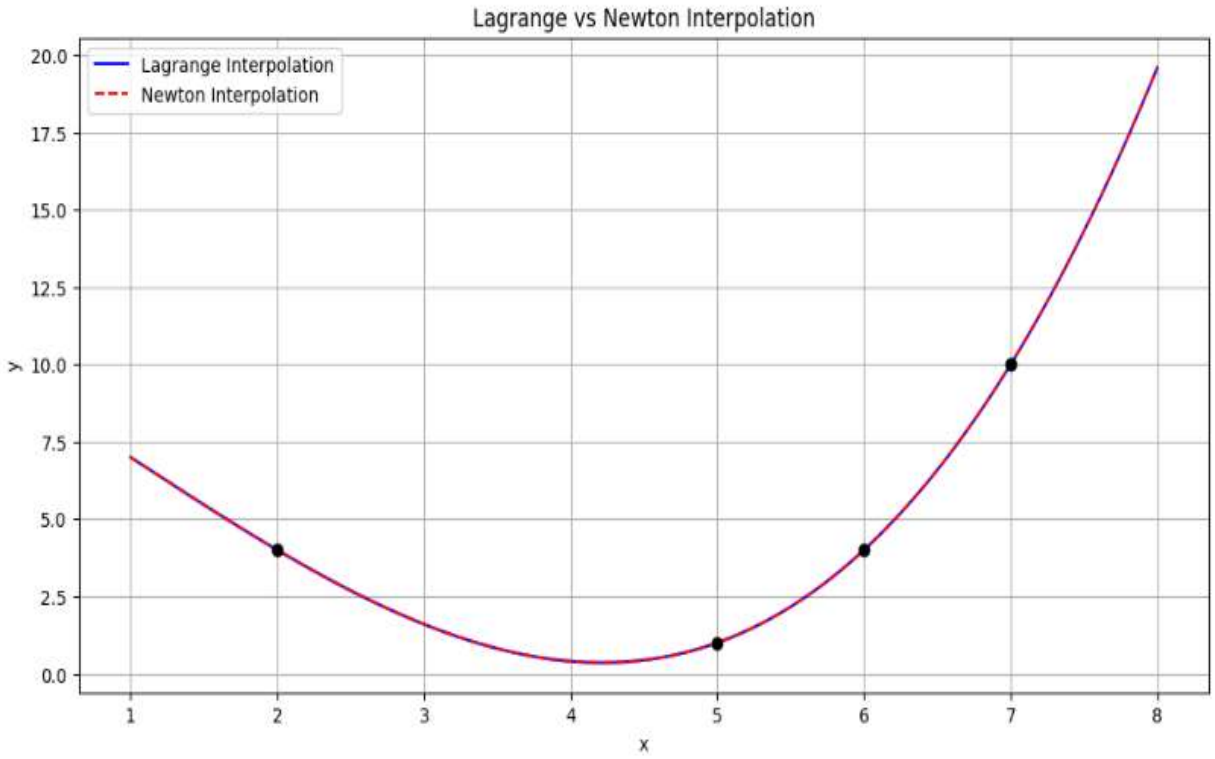
PYTHON CODE AND RESULTS

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange
from scipy.interpolate import BarycentricInterpolator

# Define the data points
x_points = np.array([2, 5, 6, 7])
y_points = np.array([4, 1, 4, 10])

# Lagrange Interpolation
lagrange_poly = lagrange(x_points, y_points)
```

```python
# Newton's Divided Difference Interpolation
def newton_divided_difference(x, y):
    n = len(x)
    coefficients = np.zeros(n)
    divided_diff = np.zeros([n, n])

    divided_diff[:, 0] = y

    for j in range(1, n):
        for i in range(n - j):
            divided_diff[i, j] = (divided_diff[i + 1, j - 1] - divided_diff[i, j - 1]) / (x[i + j] - x[i])

    coefficients = divided_diff[0, :]

    def newton_poly(t):
        result = coefficients[0]
        term = 1
        for i in range(1, n):
            term *= (t - x[i - 1])
            result += coefficients[i] * term
        return result

    return newton_poly
```

```python
# Create the Newton polynomial
newton_poly = newton_divided_difference(x_points, y_points)

# Define a range of x values for plotting
x_range = np.linspace(min(x_points) - 1, max(x_points) + 1, 400)
lagrange_values = lagrange_poly(x_range)
newton_values = [newton_poly(x) for x in x_range]

# Evaluate polynomials at x = 3
x_eval = 3
lagrange_at_3 = lagrange_poly(x_eval)
newton_at_3 = newton_poly(x_eval)

# Print the results
print(f"Lagrange polynomial evaluated at x = 3: {lagrange_at_3:.2f}")
print(f"Newton polynomial evaluated at x = 3: {newton_at_3:.2f}")

# Plotting
plt.figure(figsize=(12, 6))
plt.plot(x_range, lagrange_values, label='Lagrange Interpolation', color='blue')
plt.plot(x_range, newton_values, label='Newton Interpolation', color='red', linestyle='--')
plt.scatter(x_points, y_points, color='black', zorder=5)
plt.title('Lagrange vs Newton Interpolation')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```

```
Lagrange polynomial evaluated at x = 3: 1.60
Newton polynomial evaluated at x = 3: 1.60
```



PRACTICE

Problem 1

A software engineer is developing an application to approximate sensor readings based on given data points. The sensor collects values of a function $f(x)$ at several positions: at $x = 0.6$, $f(x) = -0.17694460$; at $x = 0.7$, $f(x) = 0.01375227$; at $x = 0.8$, $f(x) = 0.22363362$; and at $x = 1.0$, $f(x) = 0.65809197$.

To predict the sensor value at $x = 0.9$, the engineer decides to use Newton's divided difference interpolation.

a. Use Newton's divided difference interpolating polynomials of degrees 1, 2, and 3 to approximate $f(0.9)$.
b. Calculate the actual value of $f(0.9)$ using the function $f(x) = sin(e^x - 2)$.
c. Compare the values obtained from the interpolating polynomials with the actual value.

Problem 2

For a function f, the forward divided differences are given by

$$
\begin{array}{lllll}
x_0 & = & 0.0 & f[x_0] & \\
 & & & & f[x_0, x_1] \\
x_1 & = & 0.4 & f[x_1] & f[x_0, x_1, x_2] = \frac{50}{7} \\
 & & & & f[x_1, x_2] = 10 \\
x_2 & = & 0.7 & f[x_2] = 6 &
\end{array}
$$

Determine the missing entries.

Problem 3

Use Newton divided difference interpolating polynomials of degrees 1, 2, and 3 to approximate $f(8.4)$, if $f(8.1) = 16.94410$, $f(8.3) = 17.56492$, $f(8.6) = 18.50515$, $f(8.7) = 18.82091$.

Solution: First, let us make divided difference table:

x	f(x)	First divided differences	Second divided differences	Third divided differences
$x_0 = 8.1$	$f[x_0] = 16.94410$			
		$f[x_0, x_1] = 3.1041$		
$x_1 = 8.3$	$f[x_1] = 17.56492$		$\frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = 0.06$	
		$f[x_1, x_2] = 3.1341$		$\frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0} = -0.002083$
$x_2 = 8.6$	$f[x_2] = 18.50515$		$\frac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1} = 0.05875$	
		$f[x_2, x_3] = 3.1576$		
$x_3 = 8.7$	$f[x_3] = 18.82091$			

$$P_1(x) = f(x_0) + (x - x_0) f[x_0, x_1] = 16.94410 + 3.1041\,(x - 8.1)$$

$$P_1(8.4) = 17.87533$$

$$P_2(x) = f(x_0) + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2]$$

$$= 16.94410 + 3.1041\ (x - 8.1) + 0.06(x - 8.1)(x - 8.3)$$

$$P_2(8.4) = 17.87713$$

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

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

$$= 16.94410 + 3.1041\ (x - 8.1) + 0.06(x - 8.1)(x - 8.3) - 0.002083(x - 8.1)(x - 8.3)(x - 8.6)$$

$$P_3(8.4) = 17.87714$$