CECC421: Numerical Analysis
Lecture 6: Function Interpolation

Eng. Aya Kherbek
Eng. Baher Kherbek
Faculty of Engineering
Department of Informatics
Manara University

Purpose of the Session:

Study of Interpolation Algorithms Using Numerical Methods and Converting Them into Python Code

NEWTON'S DIVIDED-DIFFERENCE INTERPOLATING

This method requires extracting the divided differences table and then using it to compute Newton's polynomial.

In Python:

1.Use a function to find the forward differences table, returning the coefficients of Newton's interpolation polynomial.

2.Use a function to find Newton's interpolation for the value 1.5.

3.Apply the above to find Newton's interpolation for the following table.

x	0	1	2	3
y	0	1	8	27

4.Plot the resulting polynomial curve using matplotlib.pyplot, showing how it passes through the points in the table.

To find the coefficients:

a_list = [a0, a1, a2, a3]

The values of a[i] must be computed in the columns shown in the table.

Each element of each column is calculated using the formula:

$$\frac{y[i+1]-y[i]}{x[i+1+k]-x[i]}$$

- We observe that the number of elements decreases from one column to the next.
- The calculations stop when the number of elements in a rank reaches one. That means we need an **outer loop** to represent the ranks and an **inner loop** for the values of x and y, which are found at the core of the function.

X	y	الفروق الأولى	الفروق الثانية	الفروق الثالثة
1.0	2.0			
1.3	3.1	$f[x1,x2]=\frac{3.1-2.0}{1.3-1.0}$		
1.6	3.9	$f[x2,x3]=\frac{3.9-3.1}{1.6-1.3}$	$f[x1,x2,x3]=\frac{f[x2,x3]-f[x1,x2]}{1.6-1.0}$	
1.9	4.5	$f[x3,x4]=\frac{4.5-3.9}{1.9-1.6}$	$f[x2,x3,x4]=\frac{f[x3,x4]-f[x2,x3]}{1.9-1.3}$	$f[x1,x2,x3,x4]=\frac{f[x2,x3,x4]-f[x1,x2,x3]}{1.9-1.0}$

```
x=[0,1,2,3]
y=[0,1,8,27]
def diff(x,y):
  a_list=[y[0]]
  for j in range(len(x)):
    Temp=[]
    for i in range(len(y)-1):
      temp=(y[i+1]-y[i])/(x[i+1+j]-x[i])
      Temp.append(temp)
    y=Temp
    a_list.append(y[0])
    if (len(y)==1):
      break
  return a_list
print(diff(x,y))
```

The solution code using the diff function to compute the differences table.

OUTPUT

```
[0, 1.0, 3.0, 1.0]
```

$P_n(value)=$

```
a[0]
+
a[1]*(value-x[0])
+
a[2]*(value-x[0])* (value-x[1])
+
a[3]*(value-x[0])* (value-x[1])* (value-x[2])
+
.
.
+
a[n]* (value-x[0])* (value-x[1])* (value-
x[2])*..*(value-x[n-1])
```

Note: The new value is the result of multiplying the old value by the next value.

```
a_value=diff(x,y)
def newton(a_value,value):
 sum=a_value[0]
 temp=1
 for i in range(len(a_value)-1):
  temp=temp*(value-x[i])
  sum=sum+a_value[i+1]*temp
 return sum
print(newton(a_value,1.5))
```

OUTPUT

```
3.375
```

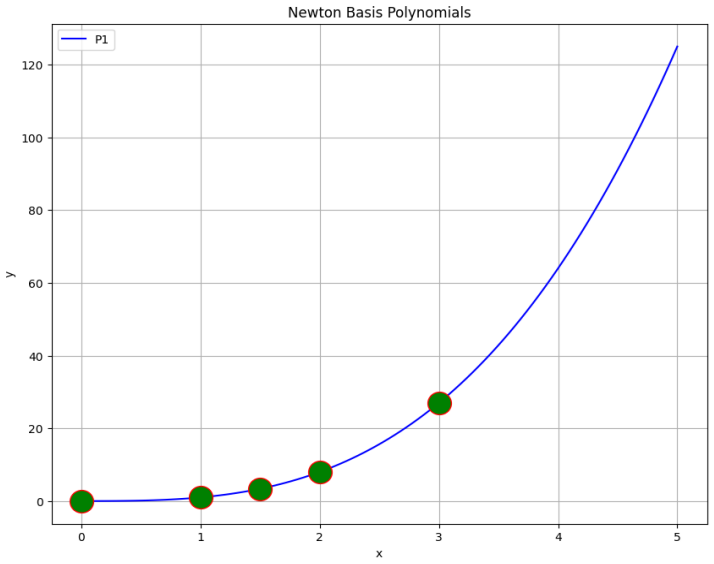
```
import numpy as np
import numpy.polynomial.polynomial as
poly
import matplotlib.pyplot as plt

x_new = np.linspace(0, 5, 100)

fig = plt.figure(figsize = (10,8))
plt.plot(x_new, newton(a_value,x_new),
'b', label = 'P1')

f=[0,1,1.5,2,3]
for i in range(len(f)):
  plt.plot(f[i], newton(a_value,f[i]),
marker="o", markersize=20,
markeredgecolor="red",
markerfacecolor="green")
```

```
plt.title('Newton Basis
Polynomials')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.legend()
plt.show()
```



LAGRANGE INTERPOLATING POLYNOMIAL

This method does not require extracting a table of divisors as in calculating Newton's polynomials.

طريقة لاغرانج:

ليكن لدينا النقاط $y_0 = f(x_0), y_1 = f(x_1), \ldots\ldots, y_n = f(x_n)$

فإنّ كثير حدود الاستيفاء لاغرانج يعطى بالعلاقة:

$$f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i) = \\ = L_0(x) f(x_0) + L_1(x) f(x_1) + \ldots\ldots L_n(x) f(x_n)$$

حيث

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

1. Start
2. Read Data of x and y
3. Read value
4. Define function takes "value" as an argument
    4.1.Initialize list Y ,
        4.2. For i = 0 to n-1
            4.2.1.Initialize temp to 1
            4.2.2.For j =0 to n-1
                4.2.2.1.If i ≠ j
                        temp = temp * (xp - Xj)/(Xi - Xj)
            4.2.3.Add temp to list Y
    4.3. For i = 0 to n-1
            Y = Y + temp * yi
    4.4. Return Y
5. Print y
6. Stop

Use Python:

1.Write a function that returns the Lagrange polynomial corresponding to the function y = f(x).

2.Apply the above to find Newton's interpolation for the given table and for the value **1.5**.

x	0	1	2	3
f(x)	1	2	9	28

3.Plot the resulting polynomial curve using matplotlib.pyplot, showing how it passes through the points in the table.

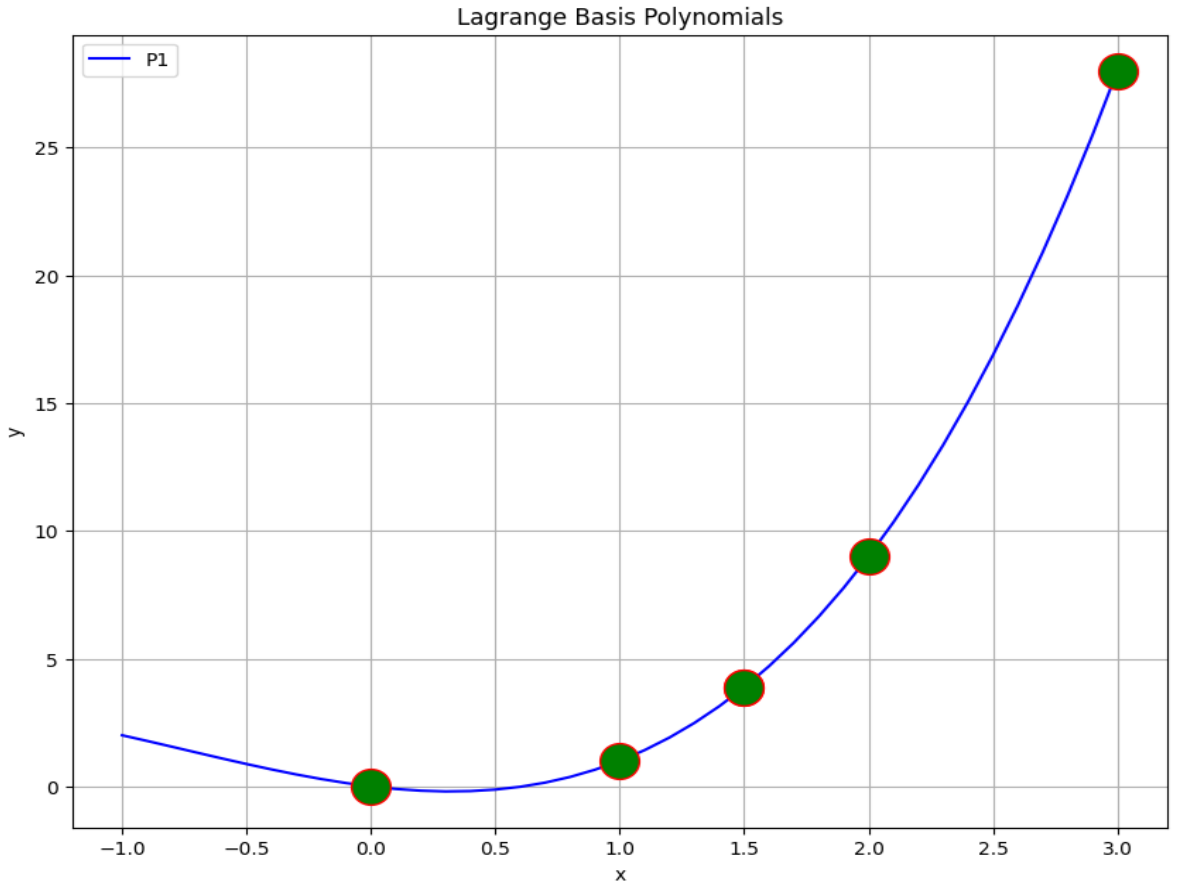
```
import numpy as np
import
numpy.polynomial.polynomi
al as poly
import matplotlib.pyplot
as plt


x=[0,1,2,3]
y=[0,1,9,28]
```

```
def Lagrange_Interpolating(value):
  L=[]
  for i in range(len(x)):
    temp=1
    for j in range(len(x)):
      if i!=j :
        temp=temp*(value-x[j])/(x[i]-x[j])
    L.append(temp)
  Y=0
  for i in range(len(y)):
    Y=Y+L[i]*y[i]
  return Y
print(Lagrange_Interpolating(1.5))
```

```
x_new = np.arange(-1.0, 3.1, 0.1)
fig = plt.figure(figsize = (10,8))
plt.plot(x_new, Lagrange_Interpolating(x_new), 'b', label = 'P1')
f=[0,1,1.5,2,3]
for i in range(len(f)):
  plt.plot(f[i], Lagrange_Interpolating(f[i]), marker="o", markersize=20,
markeredgecolor="red", markerfacecolor="green")
plt.plot(1.5, Lagrange_Interpolating(1.5), marker="o", markersize=20,
markeredgecolor="red", markerfacecolor="green")
plt.title('Lagrange Basis Polynomials')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.legend()
plt.show()
```

output


Lagrange Basis Polynomials
P1
25
20
15
10
5
0
y
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

جَامعة
المَنارة
MANARA UNIVERSITY

Thanks for Listening