EX.3.1.2.a, Sauer3

Use Newton's divided differences to find the interpolating polynomials of the points in EX.3.1.1.a, and verify agreement with the Lagrange interpolating polynomial.

a. $(0, 1)$, $(2, 3)$, $(3, 0)$.

EX.3.1.2.a, Sauer3, solution, Langou

Colab: `https://colab.research.google.com/drive/1BTq_0IcW6zc7cmCuzNj3DkOWypQbsS6W`

This method is NOT allowed. But it helps understand.

We seek the polynomial of degree 0, $p_0(x)$, that interpolates the point $(0, 1)$. We have

$$p_0(x) = c_0$$

And so $p_0(0) = 1$ leads to

$$c_0 = 1.$$

And we get

$$p_0(x) = 1.$$

We seek the polynomial of degree 1, $p_1(x)$, that interpolates the points $(0, 1)$ and $(2, 3)$. We take p_1 of the form

$$p_1(x) = p_0(x) + c_1 x.$$

That way, since $p_0(0) = 1$, we have $p_1(0) = 1$. Since we also want $p_1(2) = 3$, we get

$$p_1(2) = 3 \quad \Rightarrow \quad 1 + 2c_1 = 3 \quad \Rightarrow \quad c_1 = 1.$$

And we get

$$p_1(x) = 1 + x.$$

We seek the polynomial of degree 2, $p_2(x)$, that interpolates the points $(0, 1)$, $(2, 3)$, and $(3, 0)$. We take p_2 of the form

$$p_2(x) = p_1(x) + c_2 x(x - 2).$$

That way, since $p_1(0) = 1$ and $p_1(2) = 3$, we have $p_2(0) = 1$ and $p_2(2) = 3$. Since we also want $p_2(3) = 0$, we get

$$p_2(3) = 0 \quad \Rightarrow \quad (1 + 3) + c_2(3)(3 - 2) = 0 \quad \Rightarrow \quad c_1 = -\frac{4}{3}.$$

And we get

$$\boxed{p_2(x) = 1 + x - \tfrac{4}{3}x(x - 2).}$$

This is the answer in nested form. If we want the answer in "natural" form, we need to develop

$$p_2(x) = 1 + x - \frac{4}{3}x(x - 2) = 1 + x - \frac{4}{3}(x^2 - 2x) = -\frac{4}{3}x^2 + \frac{11}{3}x + 1.$$

We can check that this indeed the same answer as EX.3.1.1.a.

We use Newton's divided difference.
First step, we set up the table with $x_0 = 0$, $x_1 = 2$, $x_2 = 3$ and $f[x_0] = y_0 = 1$, $f[x_1] = y_2 = 3$, $f[x_2] = y_2 = 0$.
We get

$$\begin{array}{c|c} 0 & 1 \\ 2 & 3 \\ 3 & 0 \end{array}$$

Second step, we compute

$$\begin{aligned} f[x_0, x_1] &= \frac{f[x_1] - f[x_0]}{x_1 - x_0} = \frac{3-1}{2-0} = 1 \\ f[x_1, x_2] &= \frac{f[x_2] - f[x_1]}{x_2 - x_1} = \frac{0-3}{3-2} = -3 \end{aligned}$$

We get

$$\begin{array}{c|ccc} 0 & 1 & & \\ 2 & 3 & 1 & \\ 3 & 0 & -3 & \end{array}$$

Third step, we compute

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{-3-1}{3-0} = -\frac{4}{3}$$

We get

$$\begin{array}{c|ccc} 0 & 1 & & \\ 2 & 3 & 1 & \\ 3 & 0 & -3 & -\frac{4}{3} \end{array}$$

We get the coefficients c_0, c_1 and c_2 by reading the diagonal of the table:

$$\begin{array}{c|ccc} 0 & 1 & & \\ 2 & 3 & 1 & \\ 3 & 0 & -3 & -\frac{4}{3} \end{array}$$

And so the nested form representation

$$p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1)$$

gives

$$p(x) = 1 + 1(x - 0) - \frac{4}{3}(x - 0)(x - 2)$$

Finally:

$$\boxed{p_2(x) = 1 + x - \tfrac{4}{3}x(x-2).}$$

This is the answer in nested form. If we want the answer in "natural" form, we need to develop

$$p_2(x) = 1 + x - \frac{4}{3}x(x-2) = 1 + x - \frac{4}{3}(x^2 - 2x) = -\frac{4}{3}x^2 + \frac{11}{3}x + 1.$$

We can check that this indeed the same answer as EX.3.1.1.a.

What is below is not needed for full credit

We can check that

$$\begin{aligned} p(0) &= -\tfrac{4}{3}(0)^2 + \tfrac{11}{3}(0) + 1 = 1 & \checkmark \\ p(2) &= -\tfrac{4}{3}(2)^2 + \tfrac{11}{3}(2) + 1 = -\tfrac{16}{3} + \tfrac{22}{3} + 1 = 3 & \checkmark \\ p(3) &= -\tfrac{4}{3}(3)^2 + \tfrac{11}{3}(3) + 1 = -12 + 11 + 1 = 0 & \checkmark \end{aligned}$$

What is below is not needed for full credit

```python
import copy
import numpy as np

def newtdd_inplace( x, y ):
    n = len( x )
    for i in range(1,n):
        for j in range(n-i-1,-1,-1):
            y[j+i] = ( y[j+i] - y[j+i-1] ) / ( x[j+i] - x[j] )
    return y

def newtdd( x, y ):
    c = copy.deepcopy( y )
    newtdd_inplace( x, c )
    return c

def polyval_nested_w_base_points( c, b, x ):
  d = np.size(c)
  px = c[ d-1 ] * np.ones( np.shape(x) )
  for i in range( d-2, -1, -1 ):
    px = px * ( x - b[i] ) + c[i]
  return px
```

```python
x = np.array([0., 2., 3.])
y = np.array([1., 3., 0.])
```

```python
c = newtdd( x, y )
print("coefficients of interpolating polynomial in nested form")
print("from degree 0 to higest degree:\n",c)

yy = polyval_nested_w_base_points( c, x, x )
err = abs( yy - y )
print("------------------------------------------------------------------ -----")
print("|    x     |            y             |            p(x)          |  error
|")
print("------------------------------------------------------------------------")
for i in range(0,len(x)):
    print("|",f"{x[i]: 8.1f}","|", f"{y[i]: 20.16f}","|", f"{yy[i]: 20.16f}","|"
print("------------------------------------------------------------------------")

print( "absolute error = ",  f"{np.linalg.norm( y - yy, np.infty):6.1e}" )
```

```
coefficients of interpolating polynomial in nested form
from degree 0 to higest degree:
 [ 1.          1.          -1.33333333]
------------------------------------------------------------------ -----
|    x     |          y            |          p(x)          |  error   |
------------------------------------------------------------------------
|      0.0 |   1.0000000000000000 |   1.0000000000000000 |  0.0e+00 |
|      2.0 |   3.0000000000000000 |   3.0000000000000000 |  0.0e+00 |
|      3.0 |   0.0000000000000000 |   0.0000000000000002 |  2.2e-16 |
------------------------------------------------------------------------
absolute error =  2.2e-16
```