oppgave3_oving1

September 8, 2021

[7]:
```
# importing useful packages
import numpy as np
import sympy as sp
import re

# setting x equal to the symbol x
x = sp.symbols("x")
```

Given a set of $n+1$ datapoints (x_i, y_i), the interpolate-function returns a polynom p of $deg(p) = n$ that interpolates all of the $n+1$ datapoints. This function uses Newtons idea in which the polynom p is defined as $p(x) = \sum_{i=0}^{n} c_i w_i(x)$ where w_i is the omega function and c_i is a constant and equal to the divided difference of $i+1$ values $[y_0, y_1, ..., y_{i+1}]$.

[8]:
```
def interpolate(data_x, data_y):
    constants = np.array([])
    c_0 = data_y[0]
    c_1 = (data_y[1] - c_0)/(data_x[1] - data_x[0])
    constants = np.append(constants, [c_0, c_1])
    omegas = np.array([1])
    omega_1 = x - data_x[0]
    omegas = np.append(omegas, omega_1)

    if (len(data_x) <= 2):
        return;

    for i in range(2, len(data_x)):
        if (i == len(data_x) - 1):
            c_i = divided_diff(data_x[0:], data_y[0:])
        else:
            c_i = divided_diff(data_x[0:i+1], data_y[0:i+1])

        constants = np.append(constants, c_i)
        omega_i = omegas[i-1] * (x-data_x[i-1])
        omegas = np.append(omegas, omega_i)

    return sp.expand(np.dot(constants, omegas))
```

[9]:
```
def divided_diff(data_x, values):
    if (len(values) == 1):
        return values[0]
    else:
        return ((divided_diff(data_x[1:], values[1:])
                 - divided_diff(data_x[0:-1],values[0:-1]))/(data_x[-1] -␣
 ↪data_x[0]))
```

[15]:
```
# main program

if __name__ == "__main__":
    data_x = [1976, 1981, 1986, 1991, 1996, 2001]
    data_y = [4017101, 4092340, 4159187, 4249830, 4369957, 4503436]

    """
    num_inp = int(input("Number of data points (int): "))
    for i in range(1, num_inp+1):
        inp = input(f"Data point {i} (format: (x,y)): ")
        inp = re.sub("[(|)]", "", inp)
        data_point = inp.split(",")
        for i in range(2):
            data_point[i] = data_point[i].strip()
        data_x.append(float(data_point[0]))
        data_y.append(float(data_point[1]))
    """
    print(interpolate(data_x, data_y))
```

```
0.0124799999999998*x**5 - 125.693066666665*x**4 + 506293.75146666*x**3 -
1019529128.85412*x**2 + 1026369657797.55*x - 413243740768195.0
```

[21]:
```
def f(x):
    return (0.0124799999999998*x**5
            - 125.693066666665*x**4
            + 506293.75146666*x**3
            - 1019529128.85412*x**2
            + 1026369657797.55*x
            - 413243740768195.0)

print("Population in 1983:", f(1983), "\nActual population: 4122511\n")
print("Population in 1999:", f(1999), "\nActual population: 4445329\n")
print("Population in 2010:", f(2010), "\nActual population: 4858199\n")
print("Population in 2020:", f(2020), "\nActual population: 5367580\n")
```

```
Population in 1983: 4117633.75
Actual population: 4122511

Population in 1999: 4450240.5
```

```
Actual population: 4445329

Population in 2010: 4663065.0
Actual population: 4858199

Population in 2020: 4412780.5
Actual population: 5367580
```