Numerical analysis

Assignment no 2

Find a root of the equation

$$X^3 - 7x^2 + 14x - 6 = 0 \text{ correct to 4 dp in } [0,1]$$

Using the Newton Raphson method, Also perform 10 iterations by using any programming language.

$P_0 = 0$

$P_1 = P_0 - f(P_0)/f'(P_0)$

$P_1 = 0 - -6/14 = 0.4286$

$P_2 = P_1 - f(P_1)/f'(P_1)$

$P_2 = 0.4286 + 1.2070/8.5510 = 0.5697$

$P_3 = P_2 - f(P_2)/f'(P_2)$

$P_3 = 0.5697 + 0.1110/6.9976 = 0.5856$

$P_4 = P_3 - f(P_3)/f'(P_3)$

$P_4 = 0.5856 + 0.0013/6.8305 = 0.5858$

$P_5 = P_4 - f(P_4)/f'(P_4)$

$P_5 = 0.5858 + 1.9825/6.8284 = 0.5858$

$P_6 = P_5 - f(P_5)/f'(P_5)$

$P_6 = 0.5858 + 5.3290/6.8284 = 0.5858$

$P_7 = P_6 - f(P_6)/f'(P_6)$

$P_7 = 0.5858 - 0/6.8284 = 0.5858$

$P_8 = P_7 - f(P_7)/f'(P_7)$

$P_8 = 0.5858 - 0/6.8284 = 0.5858$

$P_9 = P_8 - f(P_8)/f'(P_8)$

$P_9 = 0.5858 - 0/6.8284 = 0.5858$

$P_{10} = P_9 - f(P_9)/f'(P_9)$

$P_{10} = 0.5858 - 0/6.8284 = 0.5858$

CODE: [Github]

Figure 1: Polynomial.py

```python
from math import *


class Polynomial:

    def __init__(self, coef, const):
        self.coef = coef
        self.const = const

    def __str__(self):
        result = ""
        for item in self.coef:
            term = f"{self.coef[item]}x^{item}"
            result += f"({term}) + "
        result += f" ({self.const})"
        return result

    def evaluate(self, value):
        ans = self.const
        for var in self.coef:
            ans += self.coef[var] * (value ** var)
        return ans

    def derivative(self):
        # keeps derivative form of the equation
        der = Polynomial({}, 0)

        for var in self.coef:
            der.coef[var-1] = self.coef[var] * var

        #return the derivative
        return der

    def formula(self, P0):
        f_P0 = self.evaluate(P0)
        f_der_P0 = (self.derivative()).evaluate(P0)
        result = P0 - (f_P0/f_der_P0)
        return result
```

Figure 2: app.py

```python
from Polynomial import *

def newton_raphson_iterations(fx, p0, n):
    pn = [p0]
    for i in range(1, n+1):
        ans = fx.formula(pn[i-1])
        pn.append(ans)
    return pn

fx = Polynomial(
    {
        3: 1,
        2: -7,
        1: 14,
    }, -6
)
print(f"f(x) = {fx}")

p0 = [0,1]
n = 10

pn = newton_raphson_iterations(fx, p0[0], n)

print(f"\nP\tNewton Raphson")
for i in range(len(pn)):
    print(f"{i} \t {'%.4f' % pn[i]}")
```

Figure 3: Output

```
f(x) = (1x^3) + (-7x^2) + (14x^1) +  (-6)

P   Newton Raphson
0    0.0000
1    0.4286
2    0.5697
3    0.5856
4    0.5858
5    0.5858
6    0.5858
7    0.5858
8    0.5858
9    0.5858
10   0.5858
```