FA1_PART2_AitkenHorner_KHAFAJI

February 21, 2025

0.1 1. Use Newton's Method and Aitken's Method to approximate the zero of f(x) = cos(x) with $p_0 = 0.5$.

[1]:
```
import numpy as np
```

[2]:
```
def newton_cos(p0):

    ps = [p0]

    while len(ps) < 3:
        new_p = ps[len(ps)-1] + (np.cos(ps[len(ps)-1])/np.sin(ps[len(ps)-1]))
        ps.append(new_p)

    return aitken(ps)


def aitken(p_list):

    pn = p_list[0]
    numerator = ((p_list[1] - p_list[0])**2)
    denominator = (p_list[2]-(2*p_list[1]) +p_list[0])
    p_hat = pn - numerator / denominator

    return p_hat

def approximate(p0):

    p_list = [p0]

    if len(p_list) == 1:
        p_list.append(newton_cos(p_list[len(p_list)-1]))

    while abs( p_list[len(p_list) -1] - p_list[len(p_list) -2] ) >= 1e-5:
        p_list.append(newton_cos(p_list[len(p_list)-1]))

    return p_list[len(p_list)-1]
```

[3]:
```python
approx = approximate(0.5)

print("The approximation of the zero of cosx using newton's method and aitken's␣
↪method, accurate to the 10^-5, yields x = ", approx)
```

```
The approximation of the zero of cosx using newton's method and aitken's method,
accurate to the 10^-5, yields x =  1.5707963267948966
```

0.2 2. Find approximations to within 10^{-5} to all zeroes of $f(x) = x^4 + 5x^3 - 9x^2 - 85x - 136$ by finding real zeroes using Newton's with Horner's method, then reducing to a polynomial of lower degree to determine the complex zeroes.

[4]:
```python
from numpy.polynomial import polynomial as npoly
```

[5]:
```python
def newton_w_horner(p0, polynomial):

    p = [p0]

    horn_coef_f, horn_quot_f = horner(p0, polynomial)
    horn_coef_fp, horn_quot_fp = horner(p0, horn_coef_f)

    newton_p = p0 - horn_quot_f/horn_quot_fp

    p.append(newton_p)

    while abs(p[-1] - p[-2]) >= 1e-5:

        horn_coef_f, horn_quot_f = horner(p[-1], polynomial)
        horn_coef_fp, horn_quot_fp = horner(p[-1], horn_coef_f)

        newton_p = p[-1] - horn_quot_f/horn_quot_fp

        p.append(newton_p)

    return p, horn_coef_f


def horner(p0, polynomial):

    polynomial = np.array(polynomial)
    divisor = np.array([-p0, 1])
    quotient, remainder = npoly.polydiv(polynomial, divisor)
    return quotient, remainder[-1]
```

[6]:

```python
poly_list = np.array([-136, -85, -9, 5, 1])

p_list, new_poly = newton_w_horner(1, poly_list)

roots = npoly.Polynomial(coef=new_poly).roots()

print("root found with p0 = 1, using newton and Horner's method:\n",␣
 ↪p_list[-1], end="\n\n")
print("Remaining coefficients of polynomials after reduction:\n", new_poly,␣
 ↪end="\n\n")
print("remaining real roots:\n", [root for root in roots if np.
 ↪isreal(root)],end="\n\n")
print("remaining complex roots:\n", [root for root in roots if not np.
 ↪isreal(root)], end="\n\n")
```

```
root found with p0 = 1, using newton and Horner's method:
 -4.123105625617661

Remaining coefficients of polynomials after reduction:
 [-32.98484501 -12.61552811   0.87689437   1.        ]

remaining real roots:
 [(4.123105626216223+0j)]

remaining complex roots:
 [(-2.4999999971856055-1.3228756608510046j),
(-2.4999999971856055+1.3228756608510046j)]
```

[6]:

```python

```