Midterms_Khafaji_1

March 24, 2025

[1]:
```
import numpy as np
import polars as pl
from sympy import *
import scipy.interpolate as spi
import matplotlib.pyplot as plt
```

Using bisection method to find a solution in [0.1, 1] accurate to within 10^{-4} for $600x^4 - 550x^3 + 200x^2 - 20x - 1 = 0$

let $a_n = 0.1, b_n = 1$

from the theorem:

$$|p - p_n| \leq \frac{b_n - a_n}{2^n}$$

We can calculate n from:

$$\frac{1 - 0.1}{2^n} < 10^{-4}$$
$$\Rightarrow 2^{-n} < \frac{10^{-4}}{0.9}$$
$$\Rightarrow -n = \frac{-4 - \log 0.9}{\log 2}$$

Solving for how many iterations, or n, is needed to achieve an approximation with an error within 10^{-4} yields:

[2]:
```
n_approx = -(-4 - np.log10(0.9))/np.log10(2)
print(n_approx)
```

```
13.1357092861044
```

Which says that we'll need at least 14 iterations to get the accuracy we want from our approximation.

[3]:
```
iterations = 14
an = 0.1
bn = 1
x = symbols('x')

# $600x^4 -550x^3 + 200x^2 -20x -1 = 0$
```

```
f = (600*(x**4))-(550*(x**3)) + (200*(x**2)) - (20*x) - 1
```

[4]:
```
def bisecting_iter_num1(n_iter, an, bn, func):

    # create dictionary to store iteration data
    val_table_dict = {'n': [], 'an': [], 'bn': [], 'pn': [],
                      'f(pn)': []}

    # starting loop
    i = 0

    while i <= n_iter:


        # setting n and pn for each iteration
        n = i+1
        pn = (an+bn)/2

        fpn = func.subs(x, pn)

        val_table_dict['n'].append(n)
        val_table_dict['an'].append(an)
        val_table_dict['bn'].append(bn)
        val_table_dict['pn'].append(pn)
        val_table_dict['f(pn)'].append(fpn)

        # break the loop if solution found earlier than expected:

        if fpn == 0:
            break

        # compare signs

        fan = func.subs(x, an)
        fbn = func.subs(x, bn)

        if (fan>0 and fpn>0) or (fan<0 and fpn<0):
            an = pn
        elif (fbn>0 and fpn>0) or (fbn<0 and fpn<0):
            bn = pn

        i=i+1

    return val_table_dict
```

[5]:
```
df_val_table = pl.DataFrame(bisecting_iter_num1(iterations, an, bn, f),␣
 ↪strict=False)
```

```python
with pl.Config(tbl_rows = 15):
    print(df_val_table)
```

```
shape: (15, 5)

 n     an         bn         pn         f(pn)
 ---   ---        ---        ---        ---
 i64   f64        f64        f64        f64

 1     0.1        1.0        0.55       11.8975
 2     0.1        0.55       0.325      1.438516
 3     0.1        0.325      0.2125     -0.272935
 4     0.2125     0.325      0.26875    0.52433
 5     0.2125     0.26875    0.240625   0.116296
 6     0.2125     0.240625   0.226563   -0.08051
 7     0.226563   0.240625   0.233594   0.017347
 8     0.226563   0.233594   0.230078   -0.031717
 9     0.230078   0.233594   0.231836   -0.007219
 10    0.231836   0.233594   0.232715   0.005056
 11    0.231836   0.232715   0.232275   -0.001083
 12    0.232275   0.232715   0.232495   0.001986
 13    0.232275   0.232495   0.232385   0.000451
 14    0.232275   0.232385   0.23233    -0.000316
 15    0.23233    0.232385   0.232358   0.000067
```

the Approximation $x = 0.232358$ yields the value 0.000067, which is within the desired accuracy.