Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота № 4

з дисципліни «Спеціальні розділи математики-2.
Чисельні методи»

Виконав:

студент гр. ІС-34

Колосов Ігор

Викладач:

доц. Рибачук Л.В.

Київ – 2024

Зміст

1. **Постановка задачі у вигляді вихідного рівняння**

$$a0 := 0 \quad a1 := -2 \quad a2 := 1 \quad a3 := 5 \quad a4 := -2 \quad a5 := 1 \quad k := 0$$

$$f(x) := a5\cdot(1+1)\cdot x^5 + a4\cdot x^4 + a3\cdot x^3 + a2\cdot x^2 + a1\cdot x + a0\cdot k$$

$$f\_prime(x) := 10\cdot x^4 - 8\cdot x^3 + 15\cdot x^2 + 2\cdot x - 2$$

$$f\_prime2(x) := 40\cdot x^3 - 24\cdot x^2 + 30\cdot x + 2$$

2. **Виконання допрограмового етапу, результатом якого повинні бути проміжки, щодо яких проводиться уточнення**

$f(x) = 2x^5 - 2x^4 + 5x^3 + x^2 - 2x$ кількість коренів - 5

$A = \max(2; 5; 1; +2) = 5$

$B = \max(2; 2; 5; 1) = 5$

$r = \frac{1}{1 + \frac{B}{|a_0|}} = \frac{1}{1 + \frac{5}{|a|}} = 1$

$R = 1 + \frac{1}{|a_n|} = 1 + \frac{5}{2} = \frac{7}{2}$

$1 < x^+ < \frac{7}{2}$ $-\frac{7}{2} < x^- < -1$

Теорема Гюа:

$a_k^2 \le a_k + a_{k+2}$

$4 < 2 \cdot 5$

$25 > (-2) \cdot 1$

$1 > 5 \cdot (-2)$

існує хоча б одна пара коренів

Метод Штурма

$f_0(x) = f(x) = 2x^5 - 2x^4 - 5x^3 + x^2 - 2x$

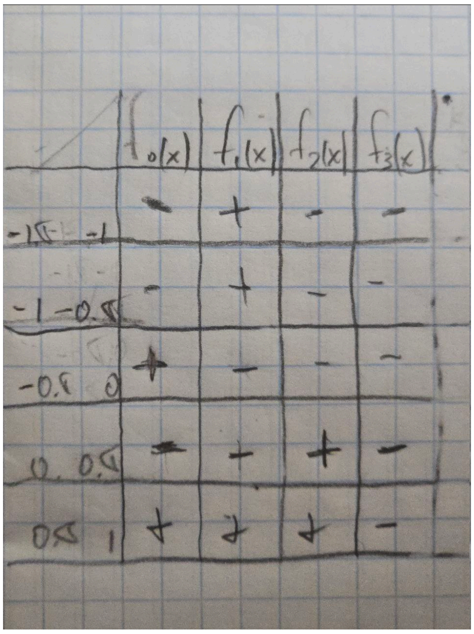
$f_1(x) = f(x) = 10x^4 - 8x^3 + 15x^2 + 2x - 2$

$2x^5 - 2x^4 + 5x^3 + x^2 - 2x = \left(\frac{1}{5}x - \frac{1}{25}\right) \cdot \left(10x^4 - 8x^3 + 5x^2 + 2x - 2\right) + \left(+\frac{42}{25}x^3 + \frac{6}{5}x^2 - \frac{38}{25}x - \frac{2}{25}\right)$

$f_2(x) = -4x^3 - 30x^2 + 38x + 2$

$10x^4 - 8x^3 + 15x^2 + 2x - 2 = \left(-\frac{5}{2}x + \frac{83}{4}\right) \cdot \left(-4x^3 - 30x^2 + 38x + 2\right) + \left(\frac{1465}{2}x^2 - \frac{563}{2} - \frac{87}{2}\right)$

$f_3(x) = -1465$



	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$
-1.5 -1	-	+	-	-
-1 -0.5	-	+	-	-
-0.5 0	+	-	-	-
0 0.5	-	-	+	-
0.5 1	+	+	+	-

3. Розв'язок уточнення коренів за методами бісекції, хорд, дотичних у Mathcad

```
bisection(f, a, b, e) := | n ← 0
                         | while |b − a| > e
                         |   | n ← n + 1
                         |   | c ← (a + b)/2
                         |   | a ← c  if f(a)·f(c) > 0
                         |   | b ← c  otherwise
                         | return (n, c)

chord(f, a, b, e) := | if f((a + b)/2)·f_prime2((a + b)/2) > 0
                     |   | x0 ← a
                     |   | x1 ← b
                     | otherwise
                     |   | x0 ← b
                     |   | x1 ← a
                     | x_prev ← x1
                     | n ← 0
                     | while 1
                     |   | x_curr ← x_prev − f(x_prev)/(f(x_prev) − f(x0))·(x_prev − x0)
                     |   | n ← n + 1
                     |   | break  if |x_prev − x_curr| < e
                     |   | x_prev ← x_curr
                     | return (n, x_curr)

newton(f, a, b, e) := | x_prev ← a  if f((a + b)/2)·f_prime2((a + b)/2) < 0
                      | x_prev ← b  otherwise
                      | n ← 0
                      | while 1
                      |   | x_curr ← x_prev − f(x_prev)/f_prime(x_prev)
                      |   | n ← n + 1
                      |   | break  if |x_curr − x_prev| < e
                      |   | x_prev ← x_curr
                      | return (n, x_curr)

ne := newton(f, start, end, epsilon) = (5, −0.612)
ch := chord(f, start, end, epsilon) = (20, −0.612)
bi := bisection(f, start, end, epsilon) = (16, −0.612)
```

4. Порівняння власного результату з розв'язком у Mathcad

```
Range: -1, -0.5
Bisection method root: -0.6120262145996094
Chord method root:     -0.6120285295141431
Newton method root:    -0.6120295311034961
Numpy roots:           [np.float64(-0.6286010114308493), np.float64(0.5827028146617159), np.float64(0.0)]
```

$$ch0 := -0.6120285295141431 \qquad ch_1 - ch0 = 1.628 \times 10^{-5}$$

$$ne0 := -0.6120295311034961 \qquad ne_1 - ne0 = -4.83 \times 10^{-12}$$

$$bi0 := -0.6120262145996094 \qquad bi_1 - bi0 = 3.815 \times 10^{-6}$$

5. Висновки

У ході виконання лабораторної роботи було помічено, що метод хорд та метод бісекції схожі, але метод хорд дає швидше сходження до точки і тому у загальному випадку в нього менше ітерацій.

Кількість ітерацій при використанні методу Ньютона напряму залежить від вказаного початкового наближення. У кожному з випадків було перевірено чи однаковий знак у даного рівняння і другої похідної. Якщо знак співпадає, то початкове наближення дорівнює початку проміжку уточненого кореня, якщо ж ні, то його кінцю. При такому наближенні метод Ньютона виявився найшвидшим.

6. Лістинг програми.

```
import numpy as np

def f(x, k=0, alpha=1):
    a0, a1, a2, a3, a4, a5 = 0, -2, 1, 5, -2, 1
    return a5*(1+alpha)*x**5 + a4*x**4 + a3*x**3 + a2*x**2 + a1*x +
a0*k

def bisection(f, a=-10, b=+10, eps=1e-5):
    while abs(b - a) > eps:
        c = (a + b) / 2
        if f(c) == 0:
            return c
        elif f(c) * f(a) < 0:
            b = c
        else:
            a = c
    return (a + b) / 2

def chord(f, a=-10, b=+10, eps=1e-5):
    c = 1.0
    while  abs(f(c)) > eps:
        c = (a * f(b) - b * f(a)) / (f(b) - f(a))
        if f(c) * f(b) > 0:
            a = c
        else: b = c
    return c

def derivative(f, x, eps=1e-5):
    return (f(x + eps) - f(x)) / eps

def newton_method(f, x0, eps=1e-5):
    x = x0
    x_old = x - 1
```

```python
    while abs(x_old - x) > eps:
        x_old = x
        x = x - f(x) / derivative(f, x, eps=eps)
    return x


a, b = -1, -0.5
bisection_root = bisection(f, a, b)
chord_root = chord(f, b,a)
newton_root = newton_method(f, a)

coeffs = [1, -2, 5, 1, -2, 0]
np_roots = np.roots(coeffs)
np_roots = [root.real for root in np_roots if root.imag == 0]

x_vals = np.linspace(a, b, 1000)
y_vals = f(x_vals)

print(f"""
Range: {a}, {b}
Bisection method root: {bisection_root}
Chord method root:     {chord_root}
Newton method root:    {newton_root}
Numpy roots:           {np_roots}
""")
```