Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Лабораторна робота №1

З курсу «Чисельні методи»

З теми «Методи розв’язання нелінійних рівнянь»

Варіант №5

Виконав студент 2 курсу групи КА-01

Вагін Олександр Вікторович

Перевірила старший викладач

Хоменко Ольга Володимирівна

Київ-2022

Завдання №1

$$-74x^7 - 789x^6 - 840x^5 + 907x^4 + 730x^3 - 348x^2 - 50x + 19 = 0$$

Спочатку виокремимо корені заданого нелінійного рівняння:

$-74x^7 - 789x^6 - 840x^5 + 907x^4 + 730x^3 - 348x^2 - 50x + 19 = 0$

$A = \max\{|a_{n-1}| \dots |a_0|\} = 907$

$B = \max\{|a_n| \dots |a_1|\} = 907$

$a_0 = 19$

$$\frac{1}{1+\frac{B}{|a_0|}} \le |x_i| \le 1 + \frac{A}{|a_n|}$$

$$\frac{1}{1+\frac{907}{19}} \le |x_i| \le 1 + \frac{907}{74}$$ (за т. про оцінку модулів коренів)

$0,02 \le |x_i| \le 13,25$

S_1 - число додатніх коренів

$S_1 = 3$ або 1

$P(-x) = 74x^7 - 789x^6 + 840x^5 + 907x^4 - 730x^3 - 348x^2 +$

$+50x + 19$

$S_2 = 4$ або 2 за т. Декарта

S_2 - кількість відємних коренів

За т. про кількість коренів алгебраїчного р-ня n-ого степенення р-ня 7-ого степеня повинно мати 7 коренів, отже $S_1 = 3$ $S_2 = 4$

Застосуємо т. Штурма, щоб виділити корені рівня: ~~Ко~~ скористуюсь онлайн калькулятором:

	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1
f_0	+	−	−	−	−	−	−	−	−	+	+	−
f_1	−	−	+	+	+	+	+	+	+	−	−	−
f_2	+	+	+	+	+	+	+	+	+	−	−	−
f_3	−	−	−	−	−	−	−	−	−	+	−	−
f_4	+	+	+	+	+	+	+	+	+	+	−	−
f_5	−	−	−	−	−	−	−	−	−	−	+	−
f_6	+	+	+	+	+	+	+	+	+	+	+	−
f_7	−	−	−	−	−	−	−	−	−	−	−	−
	7	6	6	6	6	6	6	6	6	5	3	0

Можемо зробити висновок, що на:

[-10; -9] - 1 корінь

[-2; 1] - 1 корінь

[-1; 0] - 2 корня

[0; 1] - 3 корня

Відокремимо корені на відрізку [-1; 0]

$f(-1) = -23$

$f(-0{,}95) = 13{,}03$

...

$f(-0{,}3) = 8{,}2$

$f(-0{,}25) = -2{,}5$

~~Ото~~

Відокремимо корені на відрізку [0; 1]

$f(0{,}2) = -2$

$f(0{,}25) = 1{,}3$

...

$f(0{,}35) = 2{,}1$

$f(0{,}4) = -1{,}3$

...

$f(0{,}8) = -3{,}94$

$f(0{,}85) = 47{,}1$

Використовуючи метод табуляції та т. Больцана-Коші маємо, ~~во~~ відрізки на яких по одному кореню: [-10; -9], [-2; -1], ~~[-1; 0]~~, ~~[0~~

[-1; -0,95] [-0,3; -0,25] і [0,2; 0,25]

$[0,35; 0,4]$, $[0,8; 0,85]$

Звузимо проміжки, так, щоб перша та друга похідна зберігали знаки

$f'(x) = -518x^6 - 4734x^5 - 4200x^4 + 3628x^3 + 2190x^2 - 696x - 50$

$f''(x) = -3108x^5 - 23670x^4 - 16800x^3 + 10884x^2 + 4380x - 696$

Застосувавши метод табуляції маємо:

$[-9,35; -9,30]$, $[-1,55; -1,5]$, $[-0,97; -0,96]$,

$[-0,27; -0,26]$, $[0,22; 0,23]$, $[0,38; 0,39]$

$[0,8; 0,81]$

Завдання №2

$$\lg x - \frac{7}{2x+6} = 0$$

В графічному калькуляторі побудуємо функції:

$f(x) = \lg x$

$g(x) = \dfrac{7}{2x+6}$

Обидва графіки зображені на рисунку 1.

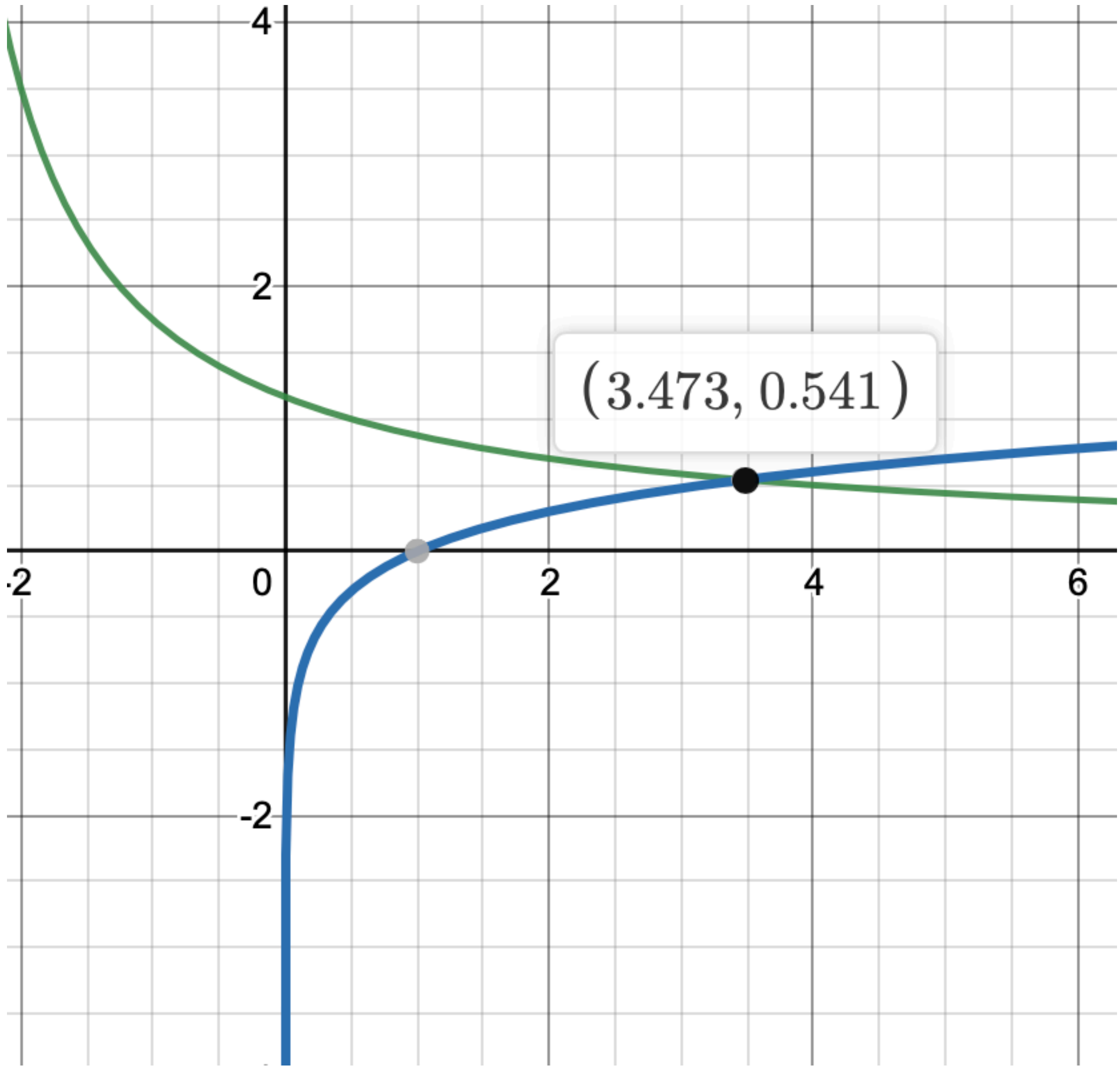

Рис. 1

Отже корінь знаходиться на відрізку:

[3; 4]

Код програм:

```
from functools import reduce

from decimal import Decimal

from decimal import *

from math import log10, log

import numpy as np

from scipy.optimize import fsolve
```

```
def generate_polynom(*args):
    """"Takes coefficients, returns polynomial function"""

    def function(x):
        return reduce(lambda t, y: t + x ** y[0] * y[1], enumerate(args[::-1]), 0)

    return function


def derivative_of_polynomial(*args):
    """Takes coefficients of polynomial function, returns coefficients of derivative
as tuple"""
    return tuple(map(lambda x: x[0] * x[1], enumerate(args[::-1])))[-1:-len(args): -1]


def tabulation(start, end, step, function):
    """Takes segment [start;end], step and function, returns values of function and
her 2 first derivatives"""
    p = start
    while p <= end + step:
```

```
        value = [Decimal(function[j](p)).quantize(Decimal('1.00000')) for j in range(3)]
        print(
            f"x = {Decimal(p).quantize(Decimal('1.00'))}\tf(x) = {value[0]} \tf'(x) = {value[1]}\tf''(x) = {value[2]}")
        p += step



def binary_search(start, end, epsilon, function):
    """Takes [start;end], precision and function, returns root on this segment using binary search"""
    middle = (start + end) / 2
    i = 1
    while abs(start - end) >= epsilon or abs(function[0](middle)) >= epsilon:
        print(
            f'Iteration {i} (a,f(a)) = ({r5(start)};{r5(function[0](start))})     (b,f(b)) = ({r5(end)};{r5(function[0](end))})')
        i += 1
        if abs(function[0](middle)) < epsilon:
            return middle
        elif (function[0](start) * function[0](middle)) / abs(function[0](start)) < 0:
            end = middle
        else:
            start = middle
```

```
        middle = (start + end) / 2
    return r5(middle)



def chord_search(start, end, epsilon, function):
    """Takes [start;end], precision and function, returns root on this segment using
chord method"""
    if function[2](start) * function[0](start) > 0:
        x1, n = end, start
    else:
        x1, n = start, end
    x2 = x1 - (function[0](x1) / (function[0](x1) - function[0](n))) * (x1 - n)
    i = 1
    while abs(x2 - x1) >= epsilon or abs(function[0](x2)) >= epsilon:
        if n == start:
            print(
                f'Iteration {i} (a,f(a)) = ({r5(start)};{r5(function[0](start))}) (x,f(x)) =
({r5(x2)};{r5(function[0](x2))})')
        else:
            print(
                f'Iteration {i} (x,f(x)) = ({r5(x2)};{r5(function[0](x2))})   (b,f(b)) =
({r5(end)};{r5(function[0](end))})')
        i += 1
```

```
        x1 = x2
        x2 = x1 - (function[0](x1) / (function[0](x1) - function[0](n))) * (x1 - n)
    return r5(x2)


def newton_search(start, end, epsilon, function):
    """Takes [start;end], precision and function, returns root on this segment using
Newton's method"""
    if function[0](start) * function[2](start) > 0:
        x1 = start
    else:
        x1 = end
    x2 = x1 - function[0](x1) / function[1](x1)
    i = 1
    while abs(x2 - x1) >= epsilon or abs(function[0](x2)) >= epsilon:
        print(f'Iteration {i} (x,f(x)) = ({r5(x2)};{r5(function[0](x2))})')
        i += 1
        x1 = x2
        x2 = x1 - function[0](x1) / function[1](x1)
    return r5(x2)


def print_roots(intervals, method, precision, function):
```

```
    for i in range(len(intervals)):
        print(f'{i}) Root on the {intervals[i]} is {method(*intervals[i], precision, function)}')



r5 = (lambda x: round(x, 5))


coefficients = (74, 789, 840, -907, -730, 348, 50, -19)
coefficients_d1 = derivative_of_polynomial(*coefficients)
coefficients_d2 = derivative_of_polynomial(*coefficients_d1)


polynom = {0: generate_polynom(*coefficients),
           1: generate_polynom(*coefficients_d1),
           2: generate_polynom(*coefficients_d2)}


transcendent = {0: lambda x: log10(x) - 7 / (2 * x + 6),
                1: lambda x: 1 / (log(10) * x) + 14 / ((2 * x + 6) ** 2),
                2: lambda x: -1 / (log(10) * (x ** 2)) - 56 / ((2 * x + 6) ** 3)}


# Табуляція поліноміальної функції для визначення відрізків на, яких єдиний корінь


# tabulation(-1, 0, 0.05, polynom)
```

```
# tabulation(0, 1, 0.05, polynom)

unique_interval = ((-10, -9), (-2, -1), (-1, -0.95), (-0.3, -0.25), (0.2, 0.25), (0.35, 0.4), (0.8, 0.85))

# Табуляція поліноміальної функції для визначення відрізків, на яких похідні є знакосталими

# tabulation(-10, -9, 0.05, polynom)
# tabulation(-2, -1, 0.05, polynom)
# tabulation(-1, -0.95, 0.01, polynom)
# tabulation(-0.3, -0.25, 0.01, polynom)
# tabulation(0.2, 0.25, 0.01, polynom)
# tabulation(0.35, 0.4, 0.01, polynom)
# tabulation(0.8, 0.85, 0.01, polynom)

const_sign_interval = (
    (-9.35, -9.30), (-1.55, -1.5), (-0.97, -0.96), (-0.27, -0.26), (0.22, 0.23), (0.38, 0.39), (0.8, 0.81))

# Пошук коренів методом бісекцій

# print_roots(const_sign_interval, binary_search, 0.0001, polynom)
```

```
# Пошук коренів методом хорд

# print_roots(unique_interval, chord_search, 0.0001, polynom)



# Пошук коренів методом Ньютона

# print_roots(unique_interval, newton_search, 0.0001, polynom)



# Пошук коренів за допомогою бібліотечних функцій

print(*sorted(np.roots(coefficients)))

print(binary_search(-10, -9, 0.0001, polynom))
print(chord_search(-9.35, -9.30, 0.0001, polynom))
print(newton_search(-9.35, -9.30, 0.0001, polynom))

# Пошук коренів методом бісекцій

print("Root of the equation is %.5f" % binary_search(3, 4, 0.0001, transcendent))
```

```
# Пошук коренів методом хорд

print("Root of the equation is %.5f" % chord_search(3, 4, 0.0001, transcendent))

# Пошук коренів методом Ньютона

print("Root of the equation is %.5f" % newton_search(3, 4, 0.0001, transcendent))

# Пошук коренів за допомогою бібліотечних функцій

print(fsolve(transcendent[0], 3))
```

Результат роботи програм:

Завдання №1

Метод бісекції:

Iteration 1 (a,f(a)) = (-10;-43305719) (b,f(b)) = (-9;10375145)

Iteration 2 (a,f(a)) = (-9.5;-8510471.09375) (b,f(b)) = (-9;10375145)

Iteration 3 (a,f(a)) = (-9.5;-8510471.09375) (b,f(b)) = (-9.25;2541842.57727)

Iteration 4 (a,f(a)) = (-9.375;-2538781.46154) (b,f(b)) = (-9.25;2541842.57727)

Iteration 5 (a,f(a)) = (-9.375;-2538781.46154) (b,f(b)) = (-9.3125;107268.75955)

Iteration 6 (a,f(a)) = (-9.34375;-1188632.27715) (b,f(b)) = (-9.3125;107268.75955)

Iteration 7 (a,f(a)) = (-9.32812;-533987.92288) (b,f(b)) = (-9.3125;107268.75955)

Iteration 8 (a,f(a)) = (-9.32031;-211696.93319) (b,f(b)) = (-9.3125;107268.75955)

Iteration 9 (a,f(a)) = (-9.31641;-51799.77059) (b,f(b)) = (-9.3125;107268.75955)

Iteration 10 (a,f(a)) = (-9.31641;-51799.77059) (b,f(b)) = (-9.31445;27837.90564)

Iteration 11 (a,f(a)) = (-9.31543;-11955.05871) (b,f(b)) = (-9.31445;27837.90564)

Iteration 12 (a,f(a)) = (-9.31543;-11955.05871) (b,f(b)) = (-9.31494;7947.88928)

Iteration 13 (a,f(a)) = (-9.31519;-2001.96793) (b,f(b)) = (-9.31494;7947.88928)

Iteration 14 (a,f(a)) = (-9.31519;-2001.96793) (b,f(b)) = (-9.31506;2973.36483)

Iteration 15 (a,f(a)) = (-9.31519;-2001.96793) (b,f(b)) = (-9.31512;485.79949)

Iteration 16 (a,f(a)) = (-9.31516;-758.05896) (b,f(b)) = (-9.31512;485.79949)

Iteration 17 (a,f(a)) = (-9.31514;-136.12342) (b,f(b)) = (-9.31512;485.79949)

Iteration 18 (a,f(a)) = (-9.31514;-136.12342) (b,f(b)) = (-9.31513;174.83962)

Iteration 19 (a,f(a)) = (-9.31514;-136.12342) (b,f(b)) = (-9.31514;19.3585)

Iteration 20 (a,f(a)) = (-9.31514;-58.38236) (b,f(b)) = (-9.31514;19.3585)

Iteration 21 (a,f(a)) = (-9.31514;-19.51191) (b,f(b)) = (-9.31514;19.3585)

Iteration 22 (a,f(a)) = (-9.31514;-0.0767) (b,f(b)) = (-9.31514;19.3585)

Iteration 23 (a,f(a)) = (-9.31514;-0.0767) (b,f(b)) = (-9.31514;9.6409)

Iteration 24 (a,f(a)) = (-9.31514;-0.0767) (b,f(b)) = (-9.31514;4.7821)

Iteration 25 (a,f(a)) = (-9.31514;-0.0767) (b,f(b)) = (-9.31514;2.3527)

Iteration 26 (a,f(a)) = (-9.31514;-0.0767) (b,f(b)) = (-9.31514;1.138)

Iteration 27 (a,f(a)) = (-9.31514;-0.0767) (b,f(b)) = (-9.31514;0.53065)

Iteration 28 (a,f(a)) = (-9.31514;-0.0767) (b,f(b)) = (-9.31514;0.22698)

Iteration 29 (a,f(a)) = (-9.31514;-0.0767) (b,f(b)) = (-9.31514;0.07514)

Iteration 30 (a,f(a)) = (-9.31514;-0.00078) (b,f(b)) = (-9.31514;0.07514)

Iteration 31 (a,f(a)) = (-9.31514;-0.00078) (b,f(b)) = (-9.31514;0.03718)

Iteration 32 (a,f(a)) = (-9.31514;-0.00078) (b,f(b)) = (-9.31514;0.0182)

Iteration 33 (a,f(a)) = (-9.31514;-0.00078) (b,f(b)) = (-9.31514;0.00871)

Iteration 34 (a,f(a)) = (-9.31514;-0.00078) (b,f(b)) = (-9.31514;0.00396)

Iteration 35 (a,f(a)) = (-9.31514;-0.00078) (b,f(b)) = (-9.31514;0.00159)

Iteration 36 (a,f(a)) = (-9.31514;-0.00078) (b,f(b)) = (-9.31514;0.00041)

Iteration 37 (a,f(a)) = (-9.31514;-0.00019) (b,f(b)) = (-9.31514;0.00041)

-9.31514

Метод хорд:

Iteration 1 (a,f(a)) = (-9.35;-1454277.48134) (x,f(x)) = (-9.31479;14200.13735)

Iteration 2 (a,f(a)) = (-9.35;-1454277.48134) (x,f(x)) = (-9.31513;325.15557)

Iteration 3 (a,f(a)) = (-9.35;-1454277.48134) (x,f(x)) = (-9.31514;7.44282)

Iteration 4 (a,f(a)) = (-9.35;-1454277.48134) (x,f(x)) = (-9.31514;0.17036)

Iteration 5 (a,f(a)) = (-9.35;-1454277.48134) (x,f(x)) = (-9.31514;0.0039)

-9.31514

Метод Ньютона:

Iteration 1 (x,f(x)) = (-9.31592;-32115.29412)

Iteration 2 (x,f(x)) = (-9.31514;-16.81198)

-9.31514

Завдання №2

Метод бісекції:

Iteration 1 (a,f(a)) = (3;-0.10621) (b,f(b)) = (4;0.10206)

Iteration 2 (a,f(a)) = (3;-0.10621) (b,f(b)) = (3.5;0.00561)

Iteration 3 (a,f(a)) = (3.25;-0.04812) (b,f(b)) = (3.5;0.00561)

Iteration 4 (a,f(a)) = (3.375;-0.02075) (b,f(b)) = (3.5;0.00561)

Iteration 5 (a,f(a)) = (3.4375;-0.00745) (b,f(b)) = (3.5;0.00561)

Iteration 6 (a,f(a)) = (3.46875;-0.00089) (b,f(b)) = (3.5;0.00561)

Iteration 7 (a,f(a)) = (3.46875;-0.00089) (b,f(b)) = (3.48438;0.00237)

Iteration 8 (a,f(a)) = (3.46875;-0.00089) (b,f(b)) = (3.47656;0.00074)

Root of the equation is 3.47266

Метод хорд:

Iteration 1 (a,f(a)) = (3;-0.10621) (x,f(x)) = (3.50997;0.00767)

Iteration 2 (a,f(a)) = (3;-0.10621) (x,f(x)) = (3.47564;0.00055)

Iteration 3 (a,f(a)) = (3;-0.10621) (x,f(x)) = (3.4732;4e-05)

Iteration 4 (a,f(a)) = (3;-0.10621) (x,f(x)) = (3.47303;0.0)

Root of the equation is 3.47301

Метод Ньютона:

Iteration 1 (x,f(x)) = (3.43892;-0.00715)

Iteration 2 (x,f(x)) = (3.47284;-4e-05)

Iteration 3 (x,f(x)) = (3.47301;-0.0)

Root of the equation is 3.47301

Висновок:

Завдання 1:

Метод бісекції:

x = -9.31514, Ітерацій = 32

x = -1.53325, Ітерацій = 18

x = -0.96903, Ітерацій = 14

x = -0.26271, Ітерацій = 12

x = 0.22773, Ітерацій = 10

x = 0.38517, Ітерацій = 11

x = 0.80506, Ітерацій = 14

Метод хорд:

x = -9.31514, Ітерацій = 24

x = -1.53325, Ітерацій = 89

x = -0.96903, Ітерацій = 3

x = -0.26271, Ітерацій = 3

x = 0.22773, Ітерацій = 4

x = 0.38517, Ітерацій = 4

x = 0.80506, Ітерацій = 7

Метод Ньютона:

x = -9.31514, Ітерацій = 4

x = -1.53325, Ітерацій = 5

x = -0.96903, Ітерацій = 2

x = -0.26271, Ітерацій = 2

x = 0.22773, Ітерацій = 2

x = 0.38517, Ітерацій = 2

x = 0.80506, Ітерацій = 3

Завдання 2:

Метод бісекції:

x = 3.47301, Ітерацій = 8

Метод хорд:

x = 3.47301, Ітерацій = 4

Метод Ньютона:

x = 3.47301, Ітерацій = 3

З отриманих результатів можемо зробити висновок, шо метод бісекціїї працює повільніше за метод хорд та метод Ньютона. Метод хорд, в більшості випадків, працює швидше за метод половинного ділення, але в деяких випадках він може працювати значно довше. Метод Ньютона працює швидше за інші методи.