Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

ДОМАШНЯ РОБОТА № 8
з дисципліни «Математичні моделі процесів і систем»
тема «Дослідження процесів однопараметричної оптимізації»

Варіант № 13

Виконав:

Студент 3 курсу, групи *ТІ-01*

Круть Катерина

Дата здачі 03.06.2023

Київ – 2023

Розв'язання задачі з оптимізації

Завдання:

Варіант: №13 (варіант №3 за вихідними даними)

№ варіанту	*Вихідні дані*
7	$d_1 = 1$, $d_2 = -5$, $d_3 = -10$, $d_4 = 70$

x є [0;10] з точністю *δ=0,01*.

Розв'язання:

d1	d2	d3	d4
1	-5	-10	70

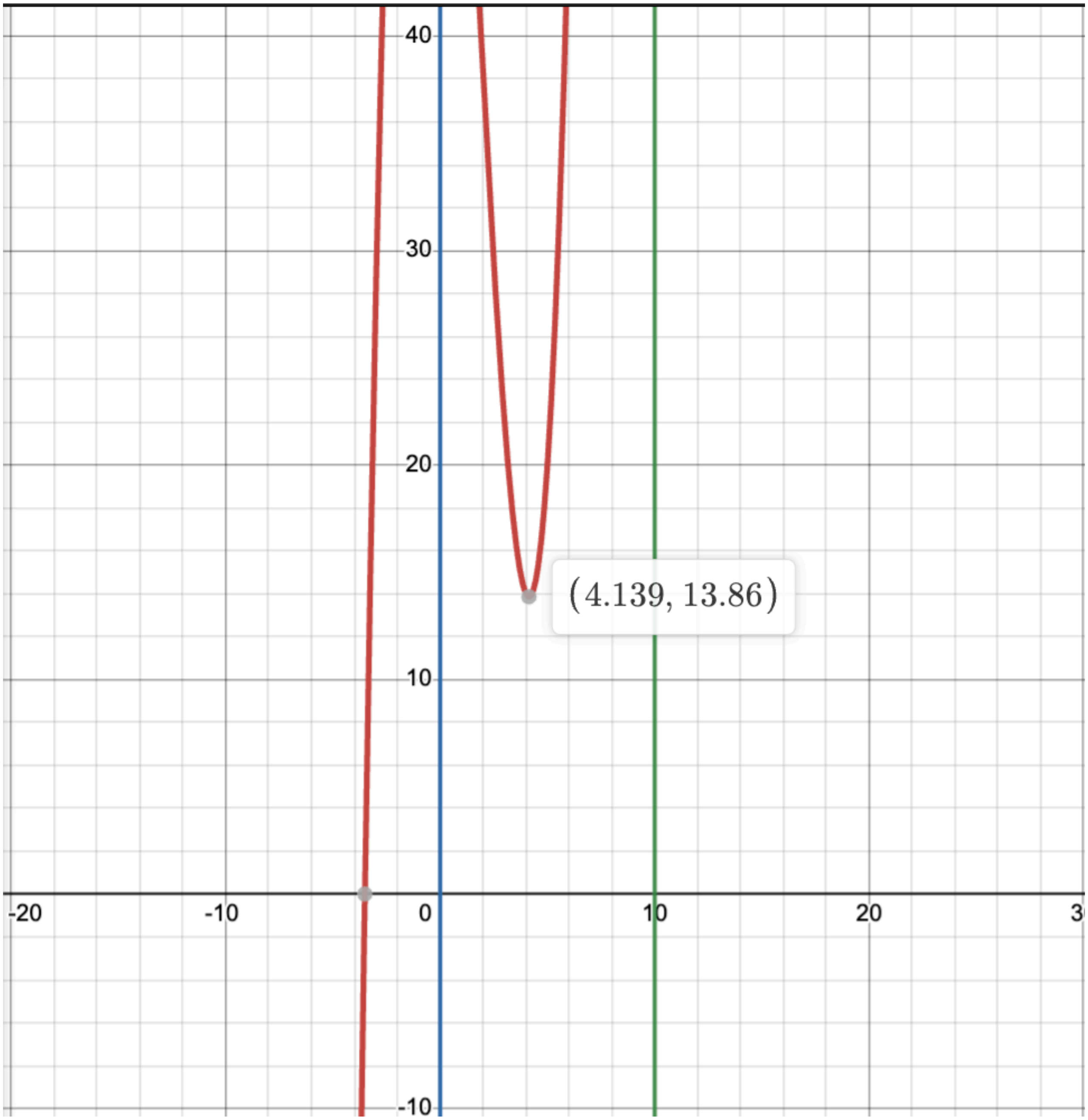
a	b
0	10

ε
0.01

k	a_k	b_k	$x_k(c)$	$L_{2k} = [a_k, b_k]$	$\|L_{2k}\|$
0	0	10	5.0	[0; 10]	5.0
1	2.5	7.5	5.0	[2.5; 7.5]	2.5
2	2.5	5.0	3.75	[2.5; 5.0]	1.25
3	3.75	5.0	4.375	[3.75; 5.0]	0.625
4	3.75	4.375	4.062	[3.75; 4.375]	0.312
5	3.906	4.219	4.062	[3.906; 4.219]	0.156
6	4.062	4.219	4.141	[4.062; 4.219]	0.078
7	4.102	4.18	4.141	[4.102; 4.18]	0.039
8	4.121	4.16	4.141	[4.121; 4.16]	0.02
9	4.131	4.15	4.141	[4.131; 4.15]	0.01
10	4.136	4.146	4.141	[4.136; 4.146]	0.005

Графік

Графік функції на відрізку x є [0; 10] з вказаною точкою локального мінімуму:

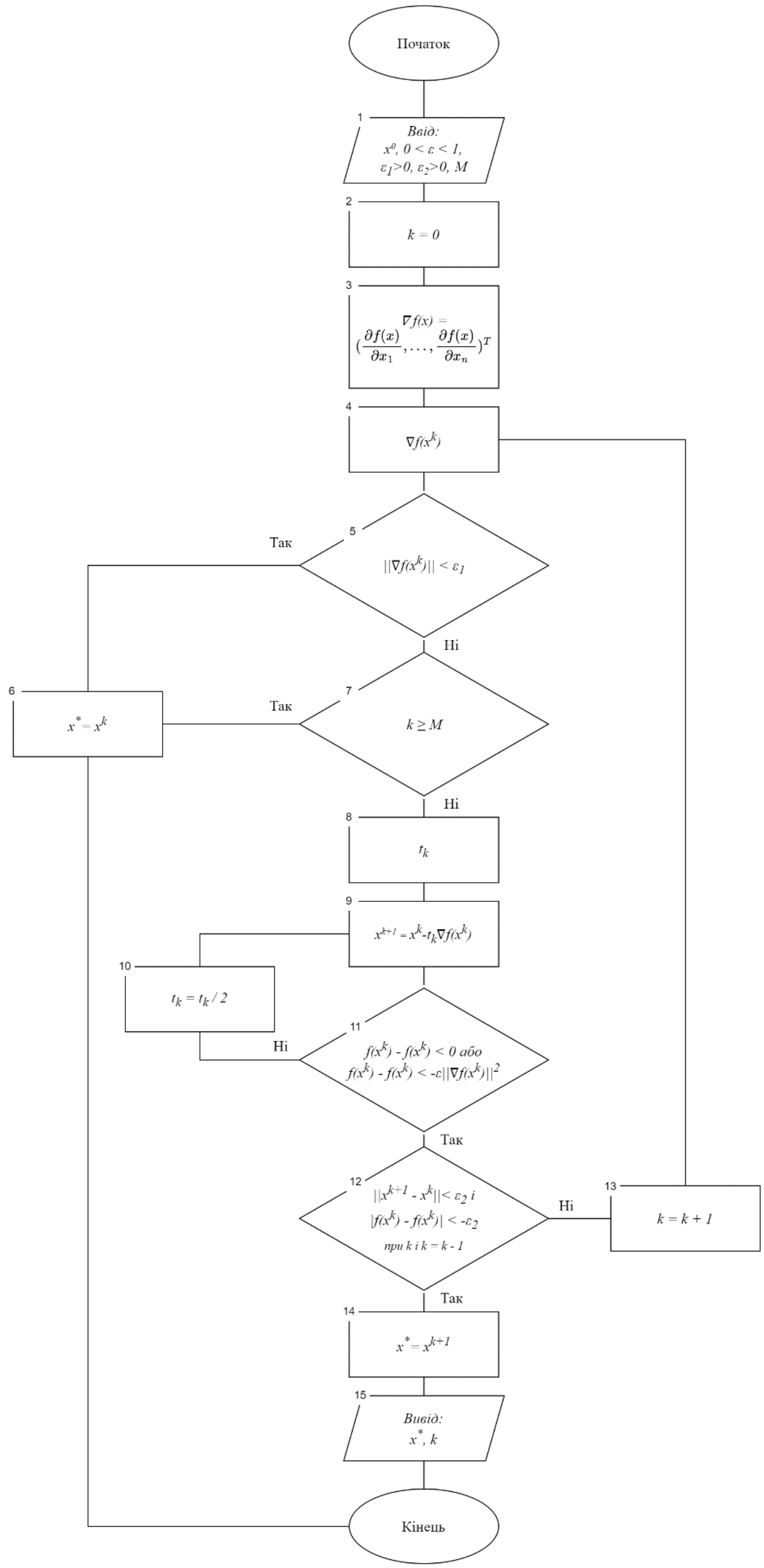


Відповідь:

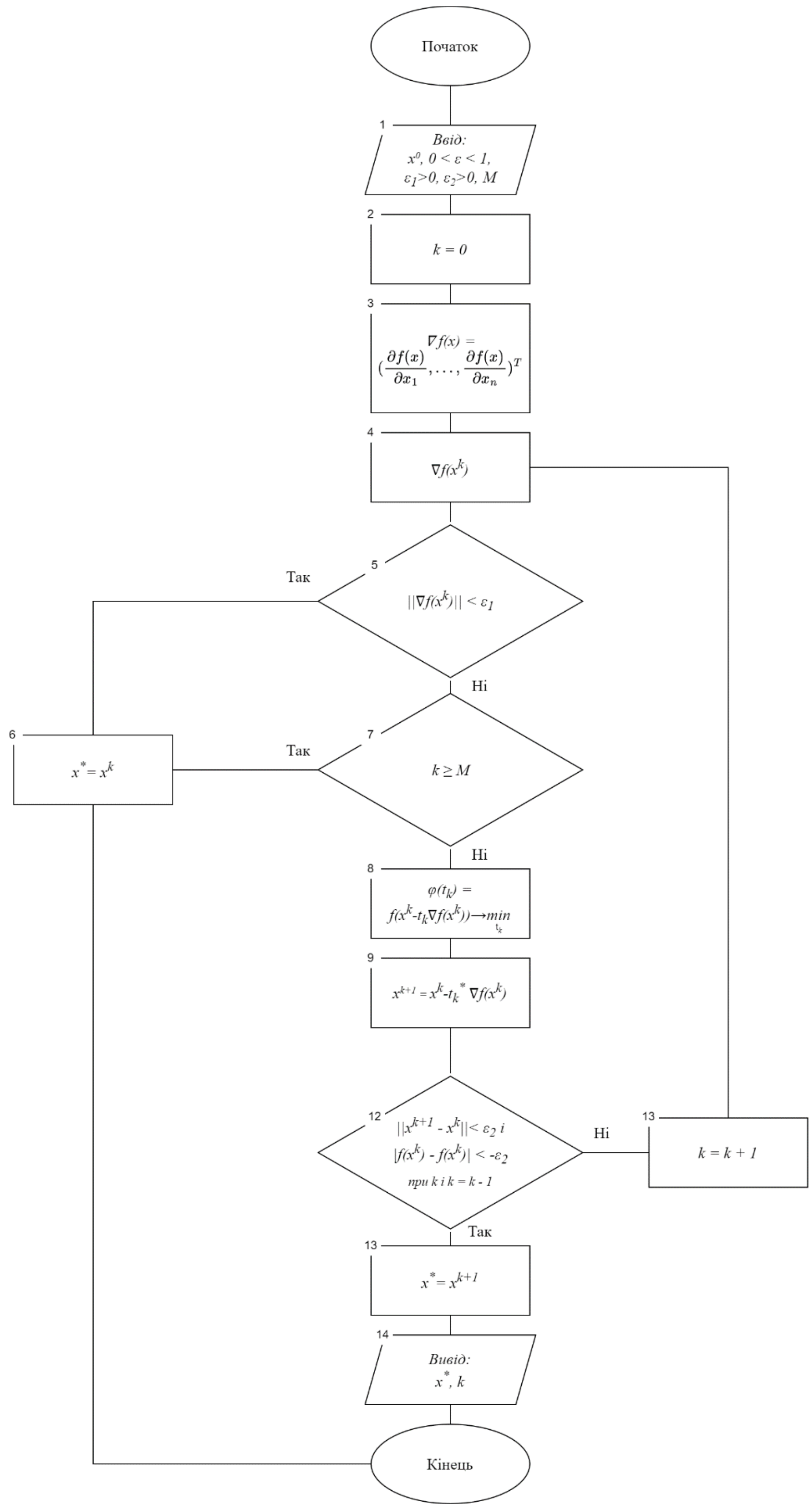
Оптимальне значення *x* = 4.1406 (через приблизно 6 кроків, стає рівним 4.140625). Мінімальне значення функції *f(x) = 13.86.*

Блок схеми

Метод градієнтного спуску із постійним кроком



Метод найшвидшого градієнтного спуску



Результат роботи

```
>>> runfile('/Users/katiakrut/PycharmProjects/mathModeling/Project_8/project_8.py', wdir='/Users/katiakrut/PycharmProjects/mathModeli
Оптимальне значення x = 4.1406
Мінімальне значення функції f(x) = 13.86
```

k	Ak	Bk	Xk(c)	L2k = [Ak; Bk]	|L2k|
0	0	10	5.0	[0; 10]	5.0
1	2.5	7.5	5.0	[2.5; 7.5]	2.5
2	2.5	5.0	3.75	[2.5; 5.0]	1.25
3	3.75	5.0	4.375	[3.75; 5.0]	0.625
4	3.75	4.375	4.0625	[3.75; 4.375]	0.3125
5	3.90625	4.21875	4.0625	[3.90625; 4.21875]	0.15625
6	4.0625	4.21875	4.140625	[4.0625; 4.21875]	0.078125
7	4.1015625	4.1796875	4.140625	[4.1015625; 4.1796875]	0.0390625
8	4.12109375	4.16015625	4.140625	[4.12109375; 4.16015625]	0.01953125
9	4.130859375	4.150390625	4.140625	[4.130859375; 4.150390625]	0.009765625
10	4.1357421875	4.1455078125	4.140625	[4.1357421875; 4.1455078125]	0.0048828125

Усі дані були занесені до таблиці із заокругленням до 3 числа після коми.

Код програми

```
def polynomial(coeffs, x):
    return sum([coeffs[len(coeffs) - i - 1] * pow(x, i) for i in
range(len(coeffs))])


def intervaling(arg_a, arg_b, arg_eps, coeff):
    all_a, all_b = [arg_a], [arg_b]
    a, b = arg_a, arg_b
    k = 0
    len_internal = abs(a - b)
    all_x = [(a + b) / 2]

    while len_internal > arg_eps:
        mid = polynomial(coeff, all_x[-1])
        left_mid = a + len_internal / 4
        right_mid = b - len_internal / 4
        left_mid_f = polynomial(coeff, left_mid)
        right_mid_f = polynomial(coeff, right_mid)
        if left_mid_f < mid:
            all_a.append(all_a[-1])
            all_b.append(all_x[-1])
            b = all_x[-1]
            all_x.append(left_mid)
        elif right_mid_f < mid:
            all_a.append(all_x[-1])
            all_b.append(all_b[-1])
            a = all_x[-1]
            all_x.append(right_mid)
        else:
            all_a.append(left_mid)
            all_b.append(right_mid)
            a = left_mid
            b = right_mid
            all_x.append(all_x[-1])
        len_internal = abs(a - b)
        k += 1
    return all_x[-1], polynomial(coeff, all_x[-1]), k, all_x, all_a, all_b


def draw(k, all_x, all_a, all_b):
    result = "|" + f"{'-' * 123}" + "\n" + "|  {:2}  ".format(" k") + "|  {:^15}
".format("Ak") + "|  {:^15}  ".format("Bk")
    result += "|  {:^15}  ".format("Xk(c)") + "|  {:^30}  ".format("L2k = [Ak;
Bk]")
    result += "|  {:^15}  ".format("|L2k|") + "|\n|" + f"{'-' * 123}" + "\n"

    for i in range(k + 1):
        result += "|  {:2}  ".format(i) + "|  {:^15}  ".format(all_a[i]) + "|
{:^15}  ".format(all_b[i])
        result += "|  {:^15}  ".format(all_x[i]) + "|  {:^30}
".format(f"[{all_a[i]}; {all_b[i]}]")
        result += "|  {:^15}  ".format((all_b[i] - all_a[i]) / 2) + "|\n" +
f"{'-' * 123}" + "\n"
    return result
```

```
interval = [0, 10]
eps = 0.01
coeff = [2, -3, -60, 170]
best_x, best_f, k_r, all_x_r, all_a_r, all_b_r = intervaling(interval[0],
interval[1], eps, coeff)

print(f"Оптимальне значення x = {round(best_x, 2)}")
print(f"Мінімальне значення функції f(x) = {round(best_f, 2)}\n\n")
print(draw(k_r, all_x_r, all_a_r, all_b_r))
```

Висновки

У даній роботі №8, була розглянута тема про процеси одно параметричної оптимізації. Були розглянуті такі методи оптимізації, як метод половинного ділення інтервалу, метод градієнтного спуску із постійним кроком та метод найшвидшого градієнтного спуску. Була створена програма, яка розраховує необхідні характеристики і виводить на екран у вигляді таблиці як ці параметри змінювалися, під час проходження по кожному кроку методу половинного ділення інтервалу. Також на основі алгоритмів двох методів, зв’язаним із градієнтом, були побудовані блок схеми, які наочно допомагають зрозуміти суть цих алгоритмів.