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 №2.2

з дисципліни «Математичне моделювання та оптимізація процесів і систем»

тема «Чисельні методи розв'язування СЛАР»

Варіант № _12_

Виконала:

Студентка 3 курсу, групи _*TI-01*_

___*Крутъ Катерина*____

(прізвище ім'я)

Дата здачі ____09.04.2023_____

Київ – 2023

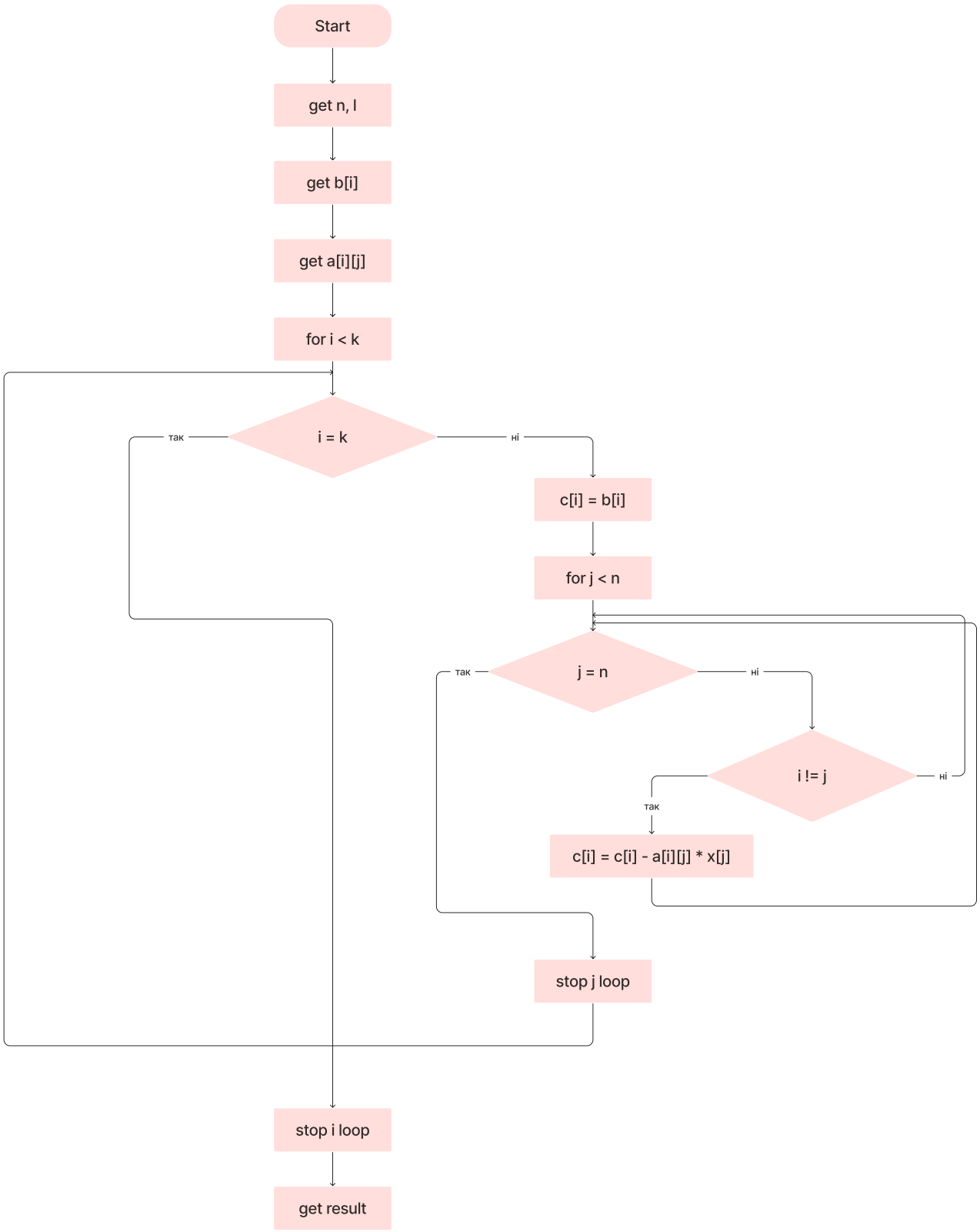
Завдання:

Знайти з точністю 0,001 розв’язок СЛАР методом Гауса-Зейделя: Для виконання завдання розробити програму на одній з мов програмування.

Варіант 12:

12	2	20	-3	0	39
	4	-2	0	24	0
	0	2	16	-1	-25
	12	0	0	3	18

Блок-схема алгоритму:



Код програми:

```
import numpy as np
import pandas as pd


class Seidel:
    def __call__(self, a, b, x):
        old_x = np.copy(x)
        for j in range(b.size):
            temp = b[j] - sum([a[j][i] * x[i] if j != i else 0 for i in range(b.size)])
            x[j] = temp / a[j][j]
        return x, old_x - x


class Jakobi:
    def __call__(self, a, b, x):
        old_x = np.copy(x)
        for j in range(b.size):
            temp = b[j] - sum([a[j][i] * old_x[i] if j != i else 0 for i in range(b.size)])
            x[j] = temp / a[j][j]
        return x, old_x - x


def get_equation(a, b):
    sole = ""
    for i in range(b.size):
        for j in range(b.size):
            sole += f" {str(a[i][j])}" + (f"x{str(j)} +" if j > 1 else "x + " if j == 1 else " +")
        sole = sole[:-1] + "= " + str(b[i]) + ";\n"
    return sole


def get_errors(a, b, x, algorythm, iter, error_):
    errors = np.array([x])
    for i in range(iter):
        x, error = algorythm(a, b, x)
        errors = np.concatenate((errors, np.array([error])), axis=0)
        if max(abs(error)) < error_:
            break

    return errors
```

```
class Sole:
    def __init__(self, algorythm=Seidel(), error=1e-3, iter=100):
        self.algorythm = algorythm
        self.error = error
        self.iter = iter

    def __call__(self, a, b):
        x = np.random.random(b.size) * 100
        errors = get_errors(a, b, x, self.algorythm, self.iter, self.error)
        errors_table = pd.DataFrame(errors, columns=["error x" + str(i) for i in
range(b.size)])
        return f"Equation:\n{get_equation(a, b)}\n", f"Errors:\n{errors_table}\n\n",
f"Result:\n{str(x)}"


A = np.array([
    [12, 0, 0, 3],
    [2, 20, -3, 0],
    [0, 2, 16, -1],
    [4, -2, 0, 24]
])
B = np.array([18, 39, -25, 0])


result = Sole()
# result = Sole(Jakobi())

for data in result(A, B):
    print(data)
```

Результат виконання програми:

```
>>> runfile('/Users/katiakrut/PycharmProjects/mathModeling/F
Equation:
 12 + 0x +  0x2 + 3x3 = 18;
 2 + 20x +  -3x2 + 0x3 = 39;
 0 + 2x +  16x2 + -1x3 = -25;
 4 + -2x +  0x2 + 24x3 = 0;


Errors:
    error x0   error x1   error x2   error x3
0  88.226276  22.184712  92.720573  26.652398
1  93.389375   5.810316  94.664098  24.427348
2  -6.106837  14.810298  -0.324578   2.251998
3  -0.562999   0.007613   0.139798   0.094468
4  -0.023617   0.023331   0.002988   0.005880
5  -0.001470   0.000595   0.000293   0.000295
6  -0.000074   0.000051   0.000012   0.000017


Result:
[ 1.53189772  1.53250637 -1.76203773 -0.12760742]

>>>
```

Перевірка збіжності:

Початкові дані:

X0	X1	X2	X3			
2	20	-3	0		ΣC1j=	8,5
4	-2	0	24		ΣC2j=	14
0	2	16	-1		ΣC3j=	0,0625
12	0	0	3		ΣC4j=	4
					Max=	14

Як видно, умові збіжності — незадовільні

Змінений порядок рівнянь:

X0	X1	X2	X3			
12	0	0	3		ΣC1j=	0,25
2	20	-3	0		ΣC2j=	0,05
0	2	16	-1		ΣC3j=	0,0625
4	-2	0	24		ΣC4j=	0,08333333
					Max=	0,25

Умові збіжності — задовільні

Онлайн калькулятор:

Решение СЛАУ методом Зейделя

Прежде чем применять метод, необходимо переставить строки исходной системы таким образом, чтобы на диагонали стояли наибольшие по модулю коэффициенты матрицы.

12	0	0	3
2	20	-3	0
0	2	16	-1
4	-2	0	24

Приведем к виду:

$x_1=1.5 - (0.25x_4)$

$x_2=1.95 - (0.1x_1-0.15x_3)$

$x_3=-1.5625 - (0.13x_2-0.0625x_4)$

$x_4=0 - (0.17x_1-0.0833x_2)$

Покажем вычисления на примере нескольких итераций.

N=1

$x_1=1.5 - 0*0 - 0*0 - 0*0.25=1.5$

$x_2=1.95 - 1.5*0.1 - 0*(-0.15) - 0*0=1.8$

$x_3=-1.5625 - 1.5*0 - 1.8*0.125 - 0*(-0.0625)=-1.7875$

$x_4=0 - 1.5*0.1667 - 1.8*(-0.0833) - (-1.7875)*0=-0.1$

N=2

$x_1=1.5 - 1.8*0 - (-1.7875)*0 - (-0.1)*0.25=1.525$

$x_2=1.95 - 1.525*0.1 - (-1.7875)*(-0.15) - (-0.1)*0=1.5294$

$x_3=-1.5625 - 1.525*0 - 1.5294*0.125 - (-0.1)*(-0.0625)=-1.7599$

$x_4=0 - 1.525*0.1667 - 1.5294*(-0.0833) - (-1.7599)*0=-0.1267$

N=3

$x_1=1.5 - 1.5294*0 - (-1.7599)*0 - (-0.1267)*0.25=1.5317$

$x_2=1.95 - 1.5317*0.1 - (-1.7599)*(-0.15) - (-0.1267)*0=1.5328$

$x_3=-1.5625 - 1.5317*0 - 1.5328*0.125 - (-0.1267)*(-0.0625)=-1.762$

$x_4=0 - 1.5317*0.1667 - 1.5328*(-0.0833) - (-1.762)*0=-0.1275$

Висновки:

В результаті виконання завдання було запрограмовано алгоритми для розв’язку СЛАР, знайдено з точністю 0,001 розв’язок СЛАР методом Гауса-Зейделя. Було проведено перевірку збіжності та тестування запрограмованих алгоритмів методом порівняння з онлайн-калькулятором, результат тестування показав, що програма працює правильно та видає задовільні результати.