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 \_*ТІ-01\_***

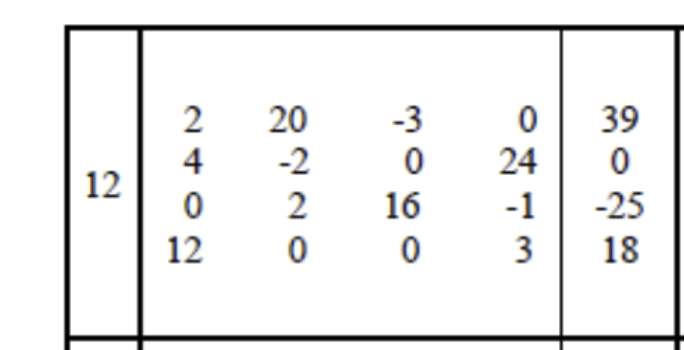
***\_\_Круть Катерина\_\_\_***

**(прізвище ім’я)**

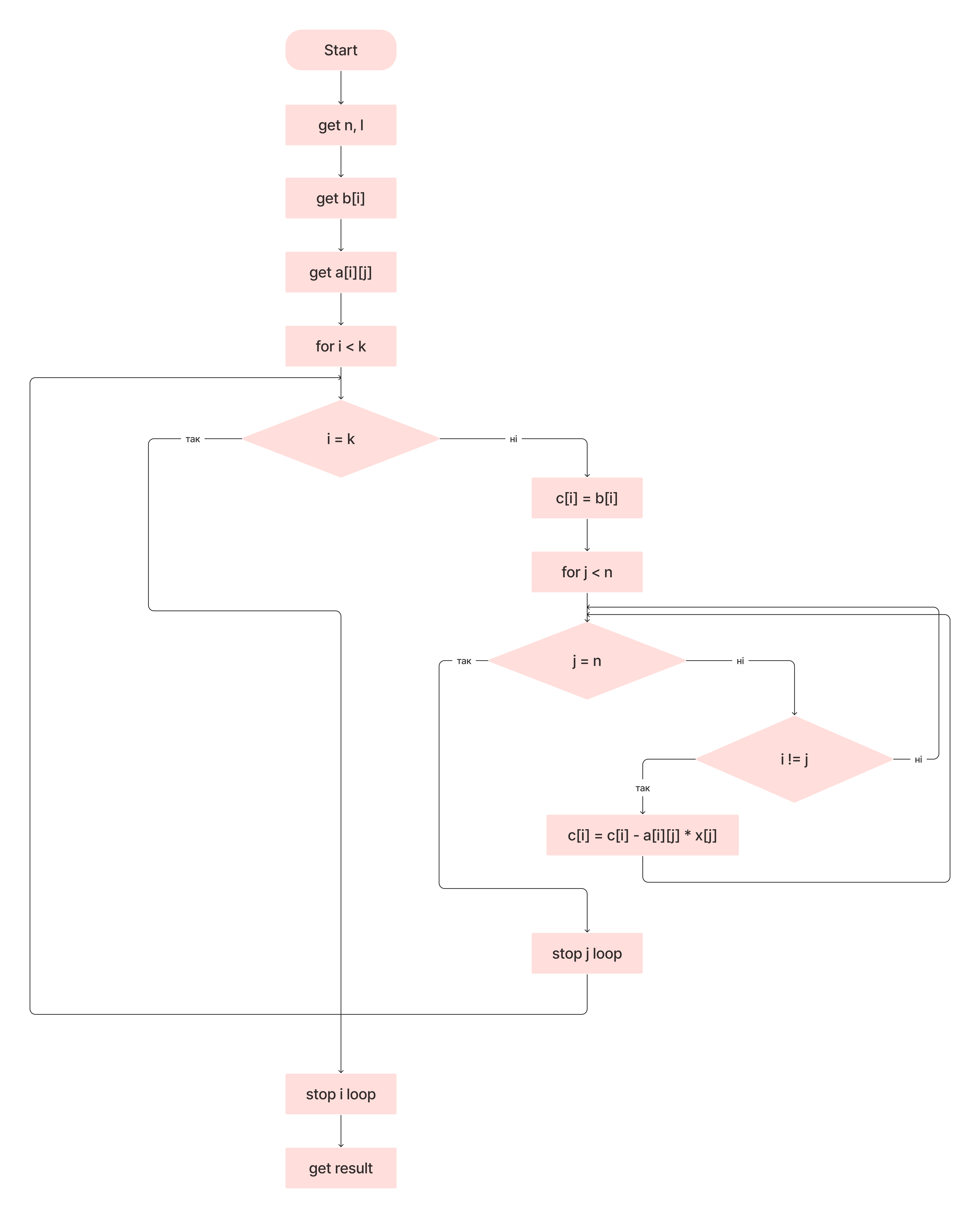
**Дата здачі \_\_\_\_09.04.2023\_\_\_\_\_**

**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:

**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]

])

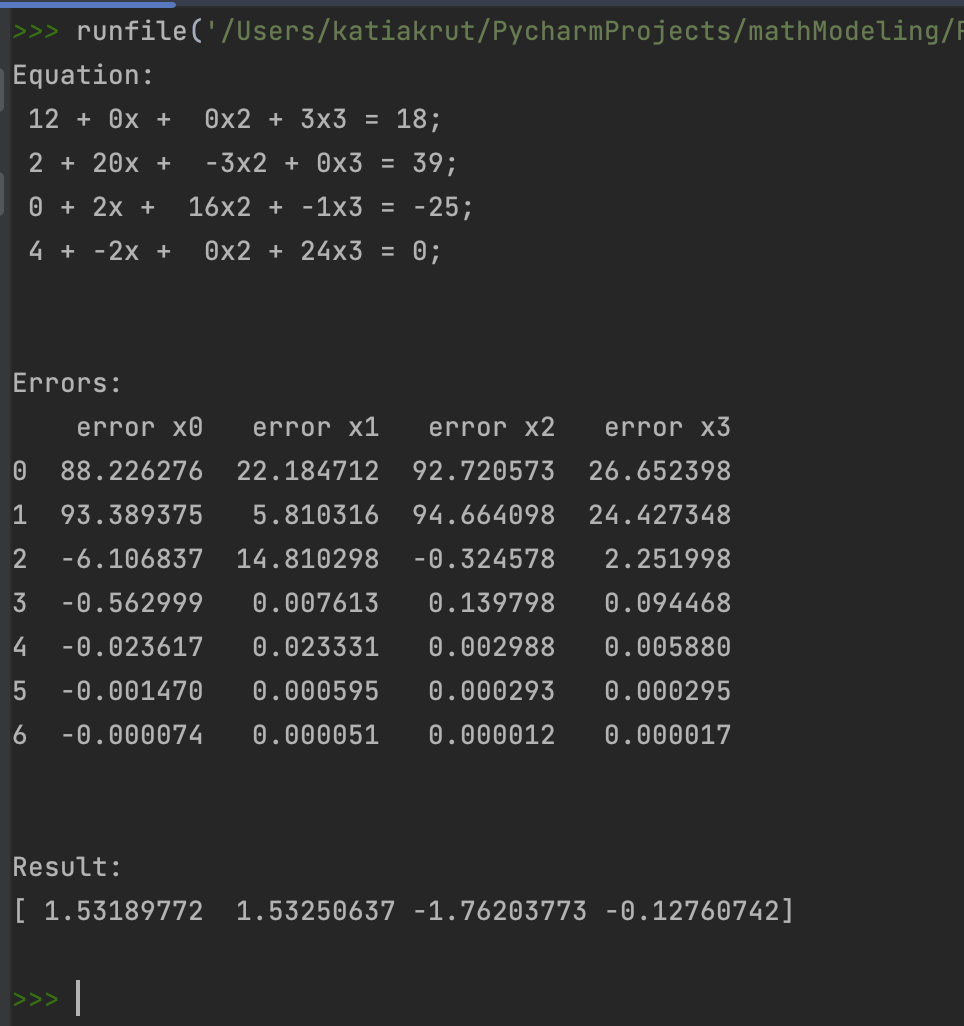
B = np.array([18, 39, -25, 0])

result = Sole()

# result = Sole(Jakobi())

for data in result(A, B):

print(data)

**Результат виконання програми:**

**Перевірка збіжності:**

Початкові дані:

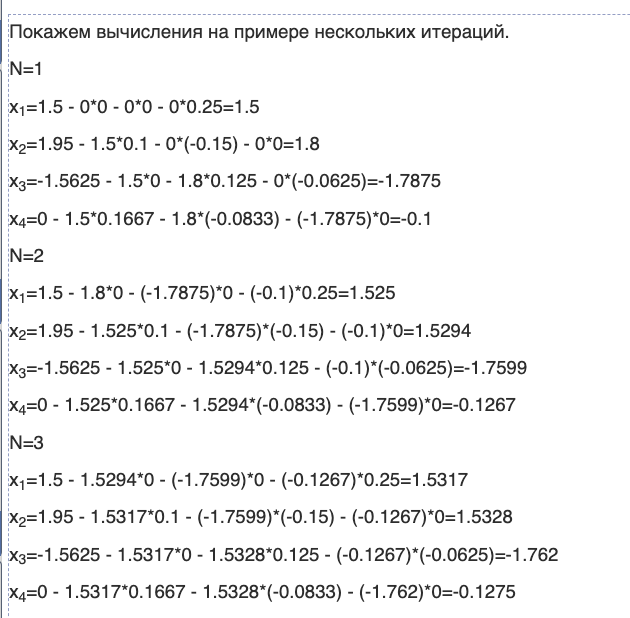
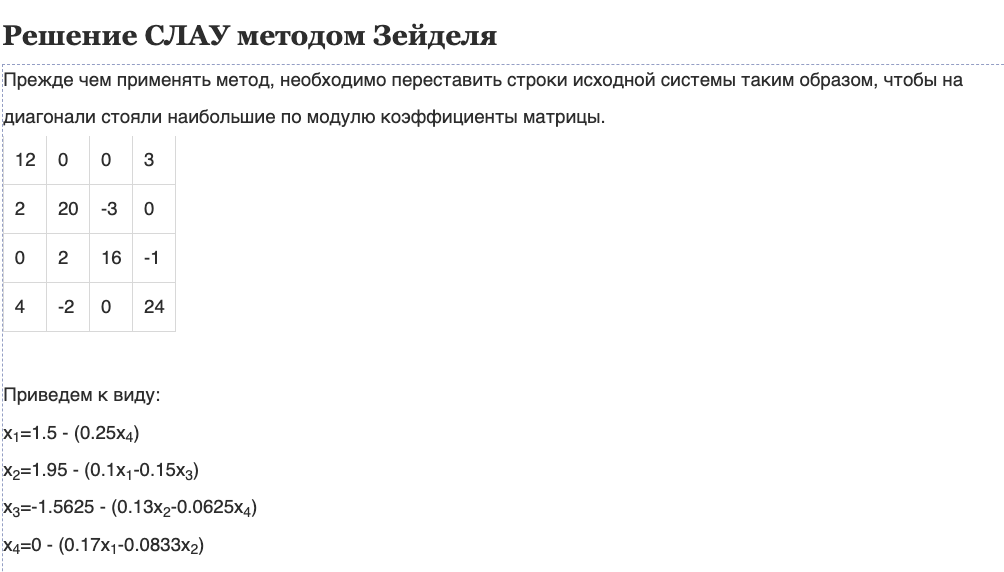
Як видно, умові збіжності — незадовільні

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X0 | X1 | X2 | X3 |  |  |  |
| 2 | 20 | -3 | 0 |  | ΣС1j= | 8,5 |
| 4 | -2 | 0 | 24 |  | ΣС2j= | 14 |
| 0 | 2 | 16 | -1 |  | ΣС3j= | 0,0625 |
| 12 | 0 | 0 | 3 |  | ΣС4j= | 4 |
|  |  |  |  |  | Max= | 14 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X0 | X1 | X2 | X3 |  |  |  |
| 12 | 0 | 0 | 3 |  | ΣС1j= | 0,25 |
| 2 | 20 | -3 | 0 |  | ΣС2j= | 0,05 |
| 0 | 2 | 16 | -1 |  | ΣС3j= | 0,0625 |
| 4 | -2 | 0 | 24 |  | ΣС4j= | 0,0833333333333333 |
|  |  |  |  |  | Max= | 0,25 |

Змінений порядок рівнянь:

Умові збіжності — задовільні

**Онлайн калькулятор:**

**Висновки:**

В результаті виконання завдання було запрограмовано алгоритми для розвʼязку 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.