Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Алгоритми та методи обчислень

Лабораторна робота №5

«Розв’язання систем лінійних алгебраїчних рівнянь»

Виконав:

Студент групи ІВ-82

Троценко Д.А.

Номер залікової книжки 8227

Номер у списку 25

Перевірив:

ст.вик. Порєв В. М.

Київ

2020 р.

**Мета:** Вивчити алгоритми методів розв'язання систем лінійних алгебраїчних

рівнянь на ЕОМ.

**Завдання:** Відповідно до варіанту завдання скласти схему алгоритму

розв’язання систем лінійних алгебраїчних рівнянь зазначеним у варіанті

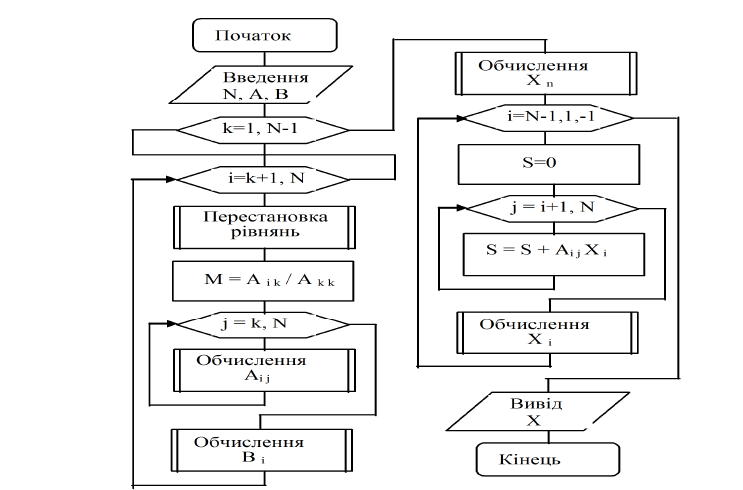
методом. Відповідно до блок-схеми скласти програму розв'язання систем

лінійних алгебраїчних рівнянь алгоритмічною мовою, узгодженою з

викладачем. Розв’язати СЛАР на ЕОМ відповідно до варіанту.

|  |  |
| --- | --- |
| № Варіанта | Метод |
| 25 | Метод Гауса з послідовним виключенням змінних |

Блок-схема:



Текст програми:

main.py

import numpy as np

import matplotlib.pyplot as plt

import sys

from PyQt5.QtWidgets import (QInputDialog, QApplication, QLineEdit, QGridLayout, QLabel, QWidget, QToolTip, QTextEdit, QPushButton, QApplication, QMessageBox, QDesktopWidget, QMainWindow)

from algo import gauss

class MainWindow(QWidget):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.initUI()

    def initUI(self):

        self.setGeometry(300, 300, 600,600)

        self.matrix = [[2,5,4,1,20],[1,3,2,1,11],[2,10,9,7,40],[3,8,9,2,37]]

        self.matrix\_edit = [[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0]]

        lbl1 = QLabel('Лабораторна робота 5 - \nМетод Гауса\nЗ послідовним виключенням елементів - \n25 варіант')

        lbl2 = QLabel('Студент групи ІВ-82')

        lbl3 = QLabel('Троценко\nДанііл\nАнатолійович')

        btn1 = QPushButton('Вірішити')

        btn1.clicked.connect(self.solve)

        grid = QGridLayout()

        grid.setSpacing(1)

        for i in range(len(self.matrix\_edit)):

            for j in range(len(self.matrix\_edit[i])):

                self.matrix\_edit[i][j] = QLineEdit(str( self.matrix[i][j] ))

        grid.addWidget(lbl1,0,0)

        grid.addWidget(lbl2,0,1)

        grid.addWidget(lbl3,0,2)

        for i in range(len(self.matrix\_edit)):

            for j in range(len(self.matrix\_edit[i])):

                print(self.matrix\_edit[i][j])

                grid.addWidget(self.matrix\_edit[i][j], i+1, j)

        self.l1 = QLabel()

        self.l2 = QLabel()

        self.l3 = QLabel()

        self.l4 = QLabel()

        grid.addWidget(btn1,8,1)

        grid.addWidget(self.l1,7,0)

        grid.addWidget(self.l2,7,1)

        grid.addWidget(self.l3,7,2)

        grid.addWidget(self.l4,7,3)

        self.setLayout(grid)

        self.setWindowTitle('Lab-5')

        self.show()

    def solve(self):

        for i in range(len(self.matrix)):

            for j in range(len(self.matrix[i])):

                self.matrix[i][j] = float(str(self.matrix\_edit[i][j].text()))

        solves = gauss(self.matrix)

        self.l1.setText("x1 = " + str(solves[0]))

        self.l2.setText("x2 = " + str(solves[1]))

        self.l3.setText("x3 = " + str(solves[2]))

        self.l4.setText("x4 = " + str(solves[3]))

if \_\_name\_\_ == "\_\_main\_\_":

    app = QApplication(sys.argv)

    ex = MainWindow()

    sys.exit(app.exec\_())

algo.py

from functools import reduce

from operator import mul, sub

def column(row):

    i = 0

    n = len(row)

    while not row[i] and i < n:

        i += 1

    return i if i < n else -1

def sign(indexes):

    s = sum((0, 1)[x < y] for k, y in enumerate(indexes) for x in indexes[k+1:])

    return ((s + 1) % 2 - s % 2)

def det(matrix):

    M = matrix[:]

    n = len(M)

    indexes = [0 for \_ in range(n)]

    for i in range(n):

        k = column(M[i])

        for j in (x for x in range(n) if x != i):

            M[j] = tuple(map(sub, M[j],

                             map(mul, M[i], [M[j][k] / M[i][k]] \* (n+1))))

        indexes[i] = k

    return reduce(mul, (M[i][indexes[i]] for i in range(n)), 1) \* sign(indexes)

def gauss(les):

    n = len(les)

    M = les[:]

    indexes = [0 for \_ in range(n)]

    for i in range(n):

        k = column(M[i])

        M[i] = [M[i][j] / M[i][k] for j in range(n+1)]

        for j in (x for x in range(n) if x != i):

            M[j] = tuple(map(sub, M[j], map(mul, M[i], [M[j][k]] \* (n + 1))))

        indexes[i] = k

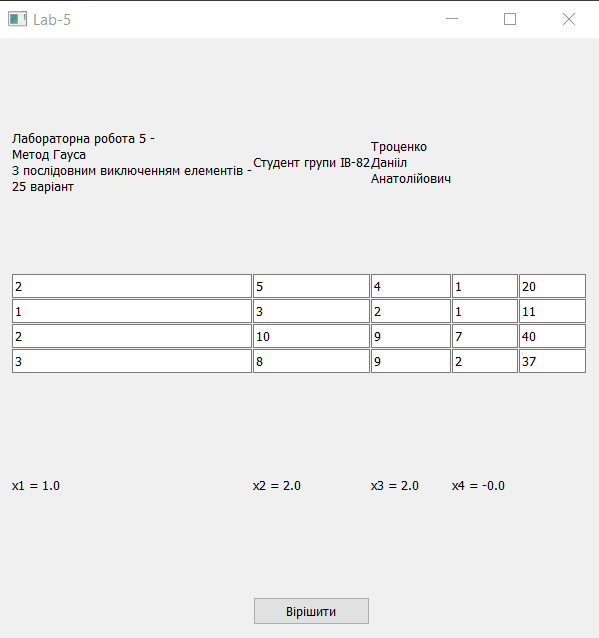
    return [M[indexes[i]][n] for i in range(n)]

if "\_\_main\_\_" == \_\_name\_\_:

    matrix = [(2,5,4,1,20),(1,3,2,1,11),(2,10,9,7,40),(3,8,9,2,37)]

    print(gauss(matrix))

Результат роботи програми:



Висновок:

У ході виконання лабораторної роботи я ознайомився з методиками різних алгоритмів розв’язання нелінійних рівнянь на комп’ютері, реалізував метод Гауса з послідонвним виключенням елемента на мові Python.