**MINISTERUL EDUCAŢIEI, CULTURII ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**UNIVERSITATEA TEHNICĂ A MOLDOVEI**

**Facultatea Calculatoare, Informatică și Microelectronică**

**Departamentul Ingineria Software și Automatică**

**RAPORT**

**la lucrarea de laborator nr.2**

**Tema: „Rezolvarea numerică a sistemelor de ecuații liniare”**

**Disciplina: „Metode Numerice”**

**A elaborat *st. gr. SI-211, Vozian Vladimir***

**A verificat *lect.univ V.Struna***

**Chișinău 2022**

**Scopul lucrării :**

* Să se separe toate rădăcinile reale ale ecuaţiei .
* Să se determine o rădăcină reală a ecuaţiei aplicând metodele : înjumătăţirii intervalului , aproximațiilor succesive , tangentelor ( Newton ) .
* Să se aprecieze rădăcina obţinută cu exactitatea .
* Să se compare rezultatele luând în consideraţie numărul de iteraţii , evaluările pentru funcţii şi derivată .

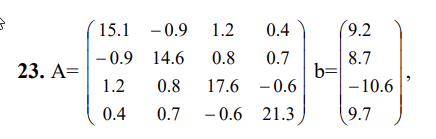
**Scopul lucrărilor:**

1) Să se rezolve sistemul de ecuaţii lineare Ax=b, utilizând

* Metoda eliminării lui Gauss;
* Metoda lui Cholesky (metoda rădăcinii pătrate);
* Metoda iterativă a lui Jacobi cu o eroare ε=10-3 ;
* Metoda iterativă a lui Gauss-Seidel cu o eroare ε=10-3 şi ε=10-5 .

2) Să se determine numărul de iteraţii necesare pentru aproximarea soluţiei sistemului cu eroarea dată ε. Să se compare rezultatele

***Varianta 23***



1. **Codul**

from numpy import array, diag, diagflat, dot, append, zeros

import sys

def seidel(a, b, x):

    for j in range(0, len(a)):

        d = b[j]

        for i in range(0, len(a)):

            if j != i:

                d -= a[j][i] \* x[i]

        x[j] = d / a[j][j]

    return x

def cholesky(matrix\_a, matrix\_b, solution\_array, n=4):

    matrix\_a = append(matrix\_a, matrix\_b, axis=1)

    for i in range(n):

        if matrix\_a[i][i] == 0.0:

            sys.exit('Eroare! Divizare cu 0')

        for j in range(i + 1, n):

            ratio = matrix\_a[j][i] / matrix\_a[i][i]

            for k in range(n + 1):

                matrix\_a[j][k] = matrix\_a[j][k] - ratio \* matrix\_a[i][k]

    solution\_array[n - 1] = matrix\_a[n - 1][n] / matrix\_a[n - 1][n - 1]

    for i in range(n - 2, -1, -1):

        solution\_array[i] = matrix\_a[i][n]

        for j in range(i + 1, n):

            solution\_array[i] = solution\_array[i] - matrix\_a[i][j] \* solution\_array[j]

        solution\_array[i] = solution\_array[i] / matrix\_a[i][i]

    return solution\_array

def jacobi(a, b, x, n=3):

    d = diag(a)

    r = a - diagflat(d)

    for i in range(n):

        x = (b - dot(r, x)) / d

    return x

def menu(array\_a, array\_b, solution):

    print("\*\*\*\*\*\*\*\*\* Alegeti metoda \*\*\*\*\*\*\*\*\*")

    print('matricea A \n', array\_a)

    print('matricea B \n', array\_b)

    while True:

        print('\n1.Gauss-Seidel\n'

              '2.Jacobi\n'

              '3.Cholesky\n')

        method = int(input('Numarul metodei: '))

        match method:

            case 1:

                for i in range(0, 5):

                    solution = seidel(array\_a, array\_b, solution)

                print('metoda Gauss-Seidel')

                print(solution, '\n')

            case 2:

                sol = jacobi(array\_a, array\_b, solution)

                print('Metoda Jacobi')

                print(sol, '\n')

            case 3:

                cahnged\_matrix\_b = array([[9.2], [8.7], [-10.6], [9.7]])

                sol = cholesky(array\_a, cahnged\_matrix\_b, initial\_solutions)

                print('Metoda Cholesky')

                print(sol, '\n')

            case \_\_:

                break

initial\_solutions = [0, 0, 0, 0]

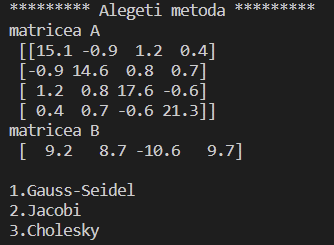
matrix\_A = array([[15.1, -0.9, 1.2, 0.4],[-0.9, 14.6, 0.8, 0.7],[1.2, 0.8, 17.6, -0.6],[0.4, 0.7, -0.6, 21.3]])

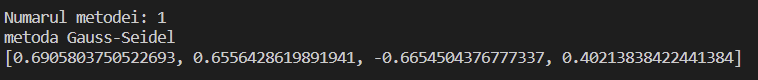
matrix\_B= array([9.2, 8.7, -10.6, 9.7])

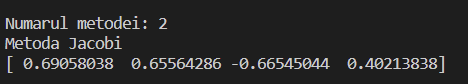
menu(matrix\_A, matrix\_B, initial\_solutions)

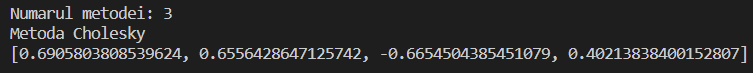
**Rezultatele**

Meniul









**Concluzie :**

În urma efectuării acestei lucrări de laborator am realizat calculul sistemului de ecuații liniare prin diferite metode: Gauss-Seidel, Jacobi și Cholesky. Metoda Gauss-Seidel este mai rapida și efectivă la un număr mai mic de necunoscute, dar la un număr mai mare eroarea de calcul crește și in acest caz mai efectiv de folosit Jacobi si Cholesky. Din rezultatele obținute vedem că rezultatele sunt aproximativ egale între ele deci metodele au fost create corect.