МIНIСТЕРСТВО ОСВIТИ I НАУКИ УКРАЇНИ

НАЦIОНАЛЬНИЙ ТЕХНIЧНИЙ УНIВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛIТЕХНIЧНИЙ IНСТИТУТ»

Кафедра прикладної математики

Звіт

із лабораторної роботи

з дисципліни «Алгоритми і системи комп’ютерної математики.

Математичні алгоритми»

на тему:

«Метод квадратних коренів»

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-63 | *Старший викладач Бай Ю.П.* |
| *Артеменко Я.К.* |  |

Київ — 2019

# **ЗМІСТ**

[**1 ВСТУП** 2](#_Toc20063355)

[**2 ОСНОВНА ЧАСТИНА** 3](#_Toc20063356)

[**2.1 Завдання** 3](#_Toc20063357)

[**2.2 Описання методу** 3](#_Toc20063358)

[**2.3 Описання алгоритму** 4](#_Toc20063359)

[**2.4 Використані тест кейси** 4](#_Toc20063360)

[**3 ВИСНОВКИ** 5](#_Toc20063361)

[**ПЕРЕЛІК ПОСИЛАНЬ** 6](#_Toc20063362)

[**ДОДАТКИ** 7](#_Toc20063363)

[**Додаток А (код програми для алгоритму)** 7](#_Toc20063364)

[**Додаток Б (код програми для тестів)** 8](#_Toc20063365)

[**Додаток В (знімки екрану результату)** 10](#_Toc20063366)

# **1 ВСТУП**

Мета лабораторної роботи:

Розробити програмне забезпечення для методу квадратних коренів вирішення системи лінійних алгебраїчних рівнянь та провести аналіз задачі, що розв’язується, та методу її розв’язання на предмет виключних ситуацій, які можуть виникати під час застосування заданого методу до розв’язання поставленої задачі.

# **2 ОСНОВНА ЧАСТИНА**

# **2.1 Завдання**

Вирішити систему

# **2.2 Описання методу**

Метод квадратного кореня включає в себе знаходження трикутної матриці U і транспонованої до неї для того, щоб представити матрицю А і щоб зменшити об’єм обчислень в два рази.

Обчислення елементів матриці U відбуваються за певними формулами:

u11 = , u1j = a1j/u11, j = 2,3,…,n

u22 = , u2j = (a2j – u12\*u1j)/u22 , j = 3,4,…,n

unn =

Таким чином знаходяться елементи матриці U із співвідношень:

Uii = 1/uii( ), i = 1,2,3,…,n

Uij = 1/uii( ), j = 2,3,…,n; j>I; uij = 0(j<i)

Якщо матриця A подана в формі Ut\*U , То система Ax = b  має вигляд Ut \*U\*x = b . Вирішення цієї системи зводиться до послідовного розв'язування двох систем з трикутними матрицями. В результаті процедура вирішення складається з двох етапів.

1. Прямий хід. Добуток U\*x позначається через y. В результаті рішення системи Ut \*y = b знаходиться стовпець y.

2. Зворотний хід. В результаті рішення системи U\*x = y знаходиться рішення задачі - стовпець x.

# **2.3 Описання алгоритму**

1. Представити матрицю A в формі **A = Ut⋅U**, Використовуючи формули з розділу 2.2.

2. Скласти систему рівнянь **Ut⋅y = b** і знайти y.

3. Скласти систему рівнянь **U⋅x = y** і знайти x.

# **2.4 Використані тест кейси**

Для заданого алгоритму було розроблено 4 перевірки на вірність вводу даних. Для перевірки подається 10 згенерованих матриць за допомогою генератора випадкових чисел. Також додаються до масиву матриці з певними помилками та матриця, яка задана в завданні.

Перевірки:

1. Виконується перевірка на те, чи має матриця 4 рядка.
2. Виконується перевірка на те, чи має матриця 4 стовпчика.
3. Виконується перевірка на те, чи введені підходящі дані.
4. Виконується перевірка на те, чи немає ділення на 0.

Якщо в якомусь з випадків вирішення системи рівнянь за заданим алгоритмом виникає помилка, то матриця, при підрахунку якої виникла якась з помилок, заноситься до іншого масиву. Це зроблено для того, щоб після виконання програми було зручно розпізнавати матриці, в яких виникла помилка і матриці, які повністю нормальні.

# **3 ВИСНОВКИ**

В даній лабораторній роботі було розроблено програмне забезпечення для розв’язання систем лінійних алгебраїчних рівнянь методом квадратних коренів. Також було розроблено ряд тестів, які перевіряють наявність різних помилок і виконують певні дії з помилками, які можуть виникати при розв1язання рівнянь.

# **ПЕРЕЛІК ПОСИЛАНЬ**

1. <http://mathhelpplanet.com/static.php?p=chislennyye-metody-resheniya-slau>

# **ДОДАТКИ**

# **Додаток А (код програми для алгоритму)**

import math  
import numpy as np  
import pprint  
import random  
  
  
def SS(A, B):  
 U = [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]  
 Ut = [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]  
  
 U[0][0] = (A[0][0]) \*\* (1 / 2)  
 U[0][1] = A[0][1] / U[0][0]  
 U[0][2] = A[0][2] / U[0][0]  
 U[0][3] = A[0][3] / U[0][0]  
  
 U[1][1] = ((A[1][1]) - U[0][1] \*\* 2) \*\* (1 / 2)  
 for j in range(2, 4):  
 U[1][j] = (A[1][j] - U[0][1] \* U[0][j]) / U[1][1]  
  
 U[2][2] = (A[2][2] - (U[0][2] \*\* 2 + U[1][2] \*\* 2)) \*\* (1 / 2)  
 U[2][3] = ((A[2][3] - (U[0][2] \* U[0][3]) + (U[1][2] \* U[1][3])) / U[2][2])  
  
 U[3][3] = (A[3][3] - (U[0][3] \*\* 2 + U[1][3] \*\* 2 + U[2][3] \*\* 2)) \*\* (1 / 2)  
  
  
 U = np.array(U)  
 Ut = U.transpose()  
 Ut.tolist()  
 U.tolist()  
  
  
  
 y = []  
  
 y.append(B[0] / Ut[0][0])  
 y.append((B[1] - Ut[1][0] \* y[0]) / Ut[1][1])  
 y.append((B[2] - Ut[2][0] \* y[0] - Ut[2][1] \* y[1]) / Ut[2][2])  
 y.append((B[3] - Ut[3][0] \* y[0] - Ut[3][1] \* y[1] - Ut[3][2] \* y[2]) / Ut[3][3])  
  
 x = []  
  
 x.append(y[3] / U[3][3])  
 x.append((y[2] - U[2][3] \* x[0]) / U[2][2])  
 x.append((y[1] - U[1][3] \* x[0] - U[1][2] \* x[1]) / U[1][1])  
 x.append((y[0] - U[0][3] \* x[0] - U[0][2] \* x[1] - U[0][1] \* x[2]) / U[0][0])  
  
 x.reverse()  
  
 return x

# **Додаток Б (код програми для тестів)**

import unittest  
from squaresuicide import SS  
import random  
  
minus = []  
a = [random.randint(-10, 10) for i in range(20)]  
a.append("A")  
matrix = [[[random.choice(a) for i in range(4)] for j in range(4)] for k in range(9)]  
matrix.append([[4.31, 0.26, 0.61, 0.27], # adekvat  
 [0.26, 2.32, 0.18, 0.34],  
 [0.61, 0.18, 3.2, 0.31],  
 [0.27, 0.34, 0.31, 5.17]])  
matrix.append([[2, -5, 2, -2],  
 [2, 3, -1, 2],  
 [3, -3, 2, -2],  
 [1, 1, -1, 2]])  
matrix.append([[8.7, -2.2, -1.1, -0.7],  
 [-2.2, 10, 2.3, -0.7],  
 [-1.1, 2.3, -5.1, 2.8],  
 [-0.7, -0.7, 2.8, 7.9, 20]])  
matrix.append([[1,1,1,1],  
 [1,-1,-1,-1],  
 [2,-1,1,0],  
 [3,5,4,10]])  
  
class SS\_test(unittest.TestCase):  
  
 def test\_SS(self):  
 for A in matrix:  
 B = [6,-4,1,26]  
 try:  
 self.assertEqual(len(A), 4)  
 except AssertionError:  
 print("{}: Не 4 строчки".format(A))  
 minus.append(A)  
 for row in A:  
 try:  
 self.assertEqual(len(row), 4)  
 except AssertionError:  
 print("{}: Не 4 cтолбца".format(A))  
 minus.append(A)  
 break  
 try: x = SS(A,B)  
 except TypeError:  
 minus.append(A)  
 print("{}: Не подходящие данные".format(A))  
  
 except ZeroDivisionError:  
 minus.append(A)  
 print("{}: Деление на 0".format(A))  
 for A in minus:  
 try:  
 matrix.remove(A)  
 except: pass  
 for A in matrix:  
 try:  
 x = SS(A, B)  
 print("\n\n{} \n{}".format(A,x))  
 except: pass  
  
  
  
if \_\_name\_\_=="\_\_main\_\_":  
 unittest.main()

# **Додаток В (знімки екрану результату)**

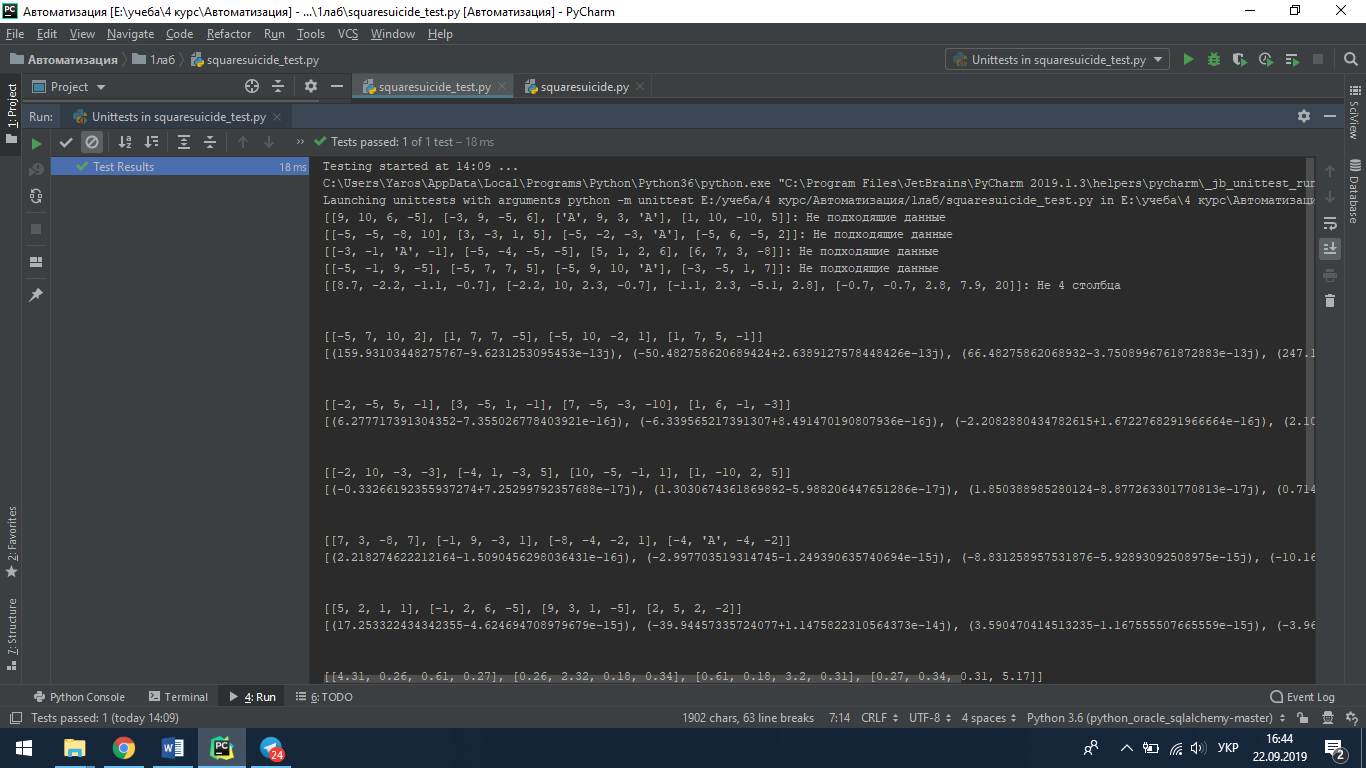


Рисунок 1. Повідомлення про помилки та частина результатів обчислень

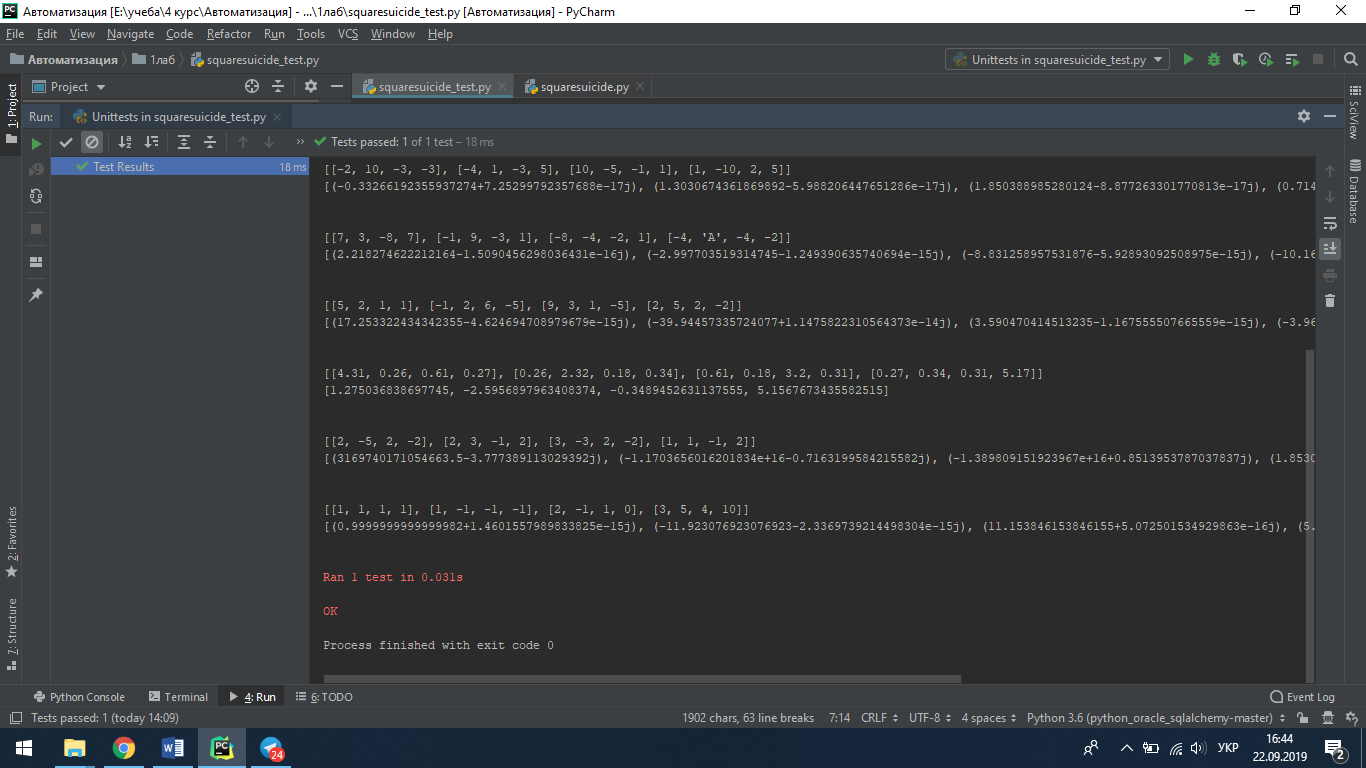


Рисунок 2. Частина результатів обчислень та повідомлення про те, що все працює