 БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Лабораторная работа №2

**Решение CЛАУ с трехдиагональной матрицей методом прогонки**

**Выполнилa:**

Осипчик Анна

2 курс 8 группа

**Преподаватель:**

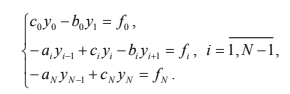
Горбачева Ю.Н.

**Постановка задачи**

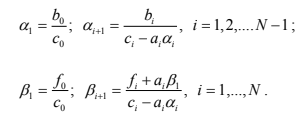
Написать и отладить программу, реализующую метод прогонки для численного решения систем линейных алгебраических уравнений с трехдиагональной матрицей порядка .

**Теоретические сведения**

Исходная система имеет следующий вид



Получение прогоночных коэффициентов (прямая прогонка)



Получение решения СЛАУ (обратная прогонка)



Относительная погрешность вычисляется по формуле

, где .

**Листинг программы**

import numpy as np  
from random import randint  
  
dim = 10  
  
def generation():  
 k = 2  
 a = np.array([-randint(-100, 100) for \_ in range(dim)])  
 b = np.array([-randint(-100, 100) for \_ in range(dim)])  
 c = np.array([randint(abs(a\_el) + abs(b\_el) + k, abs(a\_el) + abs(b\_el) + 2 \* k) for a\_el, b\_el in zip(a, b)])  
 c[0], c[dim - 1] = randint(abs(b[0]) + k, abs(b[0]) + 2 \* k), randint(abs(a[dim - 1]) + k, abs(a[dim - 1]) + 2 \* k)  
 a[0], b[dim - 1] = False, False  
 y, f = np.array(list(range(1, dim+1))), np.array([0 for \_ in range(dim)])  
 for i in range(1, dim - 1):  
 f[i] = a[i] \* y[i - 1] + c[i] \* y[i] + b[i] \* y[i + 1]  
 f[0], f[dim - 1] = c[0] \* y[0] + b[0] \* y[1], a[dim - 1] \* y[dim - 2] + c[dim - 1] \* y[dim - 1]  
 print("a: " , a)  
 print("b: ", b)  
 print("c: ", c)  
 print("y: ", y)  
 print("f: ", f)  
 return a, b, c, y, f  
def straight\_stroke(a,b,c,f):  
 alpha, betta = np.zeros((1,dim+1))[0], np.zeros((1,dim+1))[0]  
 alpha[1], betta[1] = -(b[0])/c[0], f[0]/c[0]  
 for i in range(1, dim):  
 denominator = c[i] - (-1)\*a[i]\*alpha[i]  
 alpha[i+1], betta[i+1] = -b[i]/denominator, (f[i] + (-1)\*a[i]\*betta[i])/denominator  
  
 #betta[dim] = (f[dim-1] + abs(a[dim-1])\*betta[dim-1])/(c[dim-1] - abs(a[dim-1])\*alpha[dim-1])  
 return alpha, betta  
def reverse\_stroke(alpha, betta):  
 #x = np.zeros((1,dim+1))[0]  
 x = [0 for \_ in range(dim+1)]  
 x[dim-1] = betta[dim]  
  
 for i in range(dim-1, -1, - 1):  
 x[i] = alpha[i+1]\*x[i+1] + betta[i+1]  
 return x[:len(x)-1]  
  
a, b, c, y, f = generation()  
alpha, betta = straight\_stroke(a,b,c,f)  
alpha, betta = np.asarray(alpha), np.asarray(betta)  
x = reverse\_stroke(alpha, betta)  
eps = max(a - b for a, b in zip(x, y))/max(y)  
print("x: ", x)  
print("Погрешность: ", eps)

**ВЫВОД:**

a: [ 0 32 -46 17 -32 -54 51 55 -71 -16]

b: [ 73 45 36 93 -36 -36 78 99 -2 0]

c: [ 77 81 85 113 72 93 133 158 75 18]

y: [ 1 2 3 4 5 6 7 8 9 10]

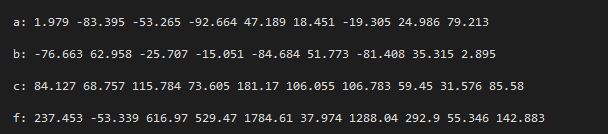
f: [ 223 329 307 968 16 36 1861 2540 87 36]

x: [0.9999999999999996, 2.0000000000000004, 2.999999999999999, 4.0, 5.0, 5.999999999999999, 6.999999999999998, 8.0, 9.000000000000002, 10.000000000000002]

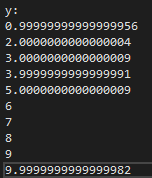
Погрешность: 1.7763568394002506e-16

**Результаты**

*Векторы и вектор свободных членов :*



*Полученное приблизительное решение:*



*Относительная погрешность, высчитанная по формуле :*



**Выводы**

Для решения СЛАУ с трехдиагональной матрицей был использован метод прогонки, причем полученное численное решение оказалось близко к точному. Относительная погрешность оказалась порядка .