РГПУ им. А.И. Герцена

**Отчет по лабораторной работе №5**

“Решение систем линейных уравнений. Метод вращения”

Работу выполнили: Иванова Мария Алексеевна

Буряков Иван Олегович

Чернышева Виктория Викторовна

Егор Собинин Яковлевич

Факультет 2об-ИВТ

Группа 1 гр. 2 п.гр

# **Цель лабораторной работы:**

Научиться реализовывать решение систем линейных алгебраических уравнений методом вращения посредством языка программирования.

# **Используемое оборудование:**

ПК, язык программирования Python.

# **Постановка задачи:**

разработать программу для решения систем линейных уравнений методом вращения.

# **Результат выполненной работы:**

## Отчет Ивановой Марии:

def matrixInput(n):

a = []

print('Вводите элементы ряда через пробел ')

for i in range(n):

ai = list(map(float, input('Элементы ряда: ').split()))

a.append(ai)

print()

return a

def rounding(n, a):

for j in range(n):

a[j] = round(a[j], 2)

print(a)

print()

def matrixOutput(n, a):

a2 = [0] \* n

for i in range(n):

for j in range(n):

a2[j] = round(a[i][j], 2)

print(a2)

print()

def forwardStroke(n, A):

for i in range(n - 1):

for k in range(i + 1, n):

sumOfSquares = (A[i][i]\*\*2 + A[k][i]\*\*2)\*\*0.5

cos = A[i][i] / sumOfSquares

sin = A[k][i] / sumOfSquares

for j in range(i, n + 1):

p = A[i][j]

A[i][j] = A[i][j] \* cos + A[k][j] \* sin

A[k][j] = A[k][j] \* cos - p \* sin

A[k][i] = 0

matrixOutput(n, A)

return A

def reverseStroke(n, A):

X = [0] \* n

X[n - 1] = A[n - 1][n] / A[n - 1][n - 1]

for i in range(n - 1, -1, -1):

s = 0

for j in range(i + 1, n):

s += A[i][j] \* X[j]

X[i] = (A[i][-1] - s) / A[i][i]

return X

def main():

#n = int(input('Введите кол-во строк: '))

#myA = matrixInput(n)

n = 4

myA = [[5, 7, 6, 5, 23], [7, 10, 8, 7, 32], [6, 8, 10, 9, 33],

[5, 7, 9, 10, 31]]

matrixOutput(n, myA)

print('Метод вращения')

resA = forwardStroke(n, myA)

x = reverseStroke(n, resA)

print('Получим результат:')

rounding(n, x)

if \_\_name\_\_ == "\_\_main\_\_":

main()

## Отчет Чернышевой Виктории:

Рассмотрим код программы:  
1) Введенные исходные данные - строки 1-25 (функции ввода и вывода матрицы)  
2) Организация этапа прямого хода - строки 28-42  
3) Организация этапа обратного хода и его вывод - строки 45-76

def matrixInput(n):

a = []

print('Вводите элементы ряда через пробел ')

for i in range(n):

ai = list(map(float, input('Элементы ряда: ').split()))

a.append(ai)

print()

return a

def rounding(n, a):

for j in range(n):

a[j] = round(a[j], 2)

print(a)

print()

def matrixOutput(n, a):

a2 = [0] \* n

for i in range(n):

for j in range(n):

a2[j] = round(a[i][j], 2)

print(a2)

print()

def forwardStroke(n, A):

for i in range(n - 1):

for k in range(i + 1, n):

sumOfSquares = (A[i][i]\*\*2 + A[k][i]\*\*2)\*\*0.5

cos = A[i][i] / sumOfSquares

sin = A[k][i] / sumOfSquares

for j in range(i, n + 1):

p = A[i][j]

A[i][j] = A[i][j] \* cos + A[k][j] \* sin

A[k][j] = A[k][j] \* cos - p \* sin

A[k][i] = 0

matrixOutput(n, A)

return A

def reverseStroke(n, A):

X = [0] \* n

X[n - 1] = A[n - 1][n] / A[n - 1][n - 1]

for i in range(n - 1, -1, -1):

s = 0

for j in range(i + 1, n):

s += A[i][j] \* X[j]

X[i] = (A[i][-1] - s) / A[i][i]

return X

def main():

#n = int(input('Введите кол-во строк: '))

#myA = matrixInput(n)

n = 4

myA = [[5, 7, 6, 5, 23], [7, 10, 8, 7, 32], [6, 8, 10, 9, 33],

[5, 7, 9, 10, 31]]

matrixOutput(n, myA)

print('Метод вращения')

resA = forwardStroke(n, myA)

x = reverseStroke(n, resA)

print('Получим результат:')

rounding(n, x)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Метод вращения является одним из ортогональных методов решения СЛАУ.   
  
Важная черта данного метода - хорошая устойчивость:

* При преобразовании исходной матрицы A используем матрицу вращения и при этом умножаем элементы матрицы A на sin и cos, значения которых не превышают 1. Поэтому преобразованная матрица не слишком отличается от исходной.

## Отчет Собинина Егора:

def matrixInput(n):

a = []

print("Вводите элементы ряда через пробел ")

for i in range(n):

ai = list(map(float, input("Элементы ряда: ").split()))

a.append(ai)

print()

return a

def rounding(n, a):

for j in range(n):

a[j] = round(a[j], 2)

print(a)

print()

def matrixOutput(n, a):

a2 = [0] \* n

for i in range(n):

for j in range(n):

a2[j] = round(a[i][j], 2)

print(a2)

print()

def forwardStroke(n, A):

for i in range(n - 1):

for k in range(i + 1, n):

sumOfSquares = (A[i][i]\*\*2 + A[k][i]\*\*2)\*\*0.5

cos = A[i][i] / sumOfSquares

sin = A[k][i] / sumOfSquares

for j in range(i, n + 1):

p = A[i][j]

A[i][j] = A[i][j] \* cos + A[k][j] \* sin

A[k][j] = A[k][j] \* cos - p \* sin

A[k][i] = 0

matrixOutput(n, A)

return A

def reverseStroke(n, A):

X = [0] \* n

X[n - 1] = A[n - 1][n] / A[n - 1][n - 1]

for i in range(n - 1, -1, -1):

s = 0

for j in range(i + 1, n):

s += A[i][j] \* X[j]

X[i] = (A[i][-1] - s) / A[i][i]

return X

def main():

n = 4

myA = [[5, 7, 6, 5, 23], [7, 10, 8, 7, 32], [6, 8, 10, 9, 33],

[5, 7, 9, 10, 31]]

matrixOutput(n, myA)

print('Метод вращения')

resA = forwardStroke(n, myA)

x = reverseStroke(n, resA)

print('Получим результат:')

rounding(n, x)

if \_\_name\_\_ == "\_\_main\_\_":

main()

# **Заключение**

Мы проверили правильность работы программ из задач на конкретных примерах вводимых данных. Программа выводит ожидаемый нами результат, соответствующий предыдущим методам.

| Вводимые данные | Метод вращения |
| --- | --- |
| [[5, 7, 6, 5, 23],  [7, 10, 8, 7, 32],  [6, 8, 10, 9, 33],  [5, 7, 9, 10, 31]] | [1.0, 1.0, 1.0, 1.0] |