РГПУ им. А.И. Герцена

**Отчет по лабораторной работе №4**

“Решение систем линейных уравнений методом квадратного корня”

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 квадратного корня посредством языка программирования.

# **Используемое оборудование:**

ПК, язык программирования Python.

# **Постановка задачи:**

разработать программу для решения систем линейных уравнений методом квадратного корня.

# **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 metod(n, A):

print('Метод квадратного корня')

print('Мatrix A')

matrixOutput(n, A)

R = [([0] \* n) for i in range(n)]

R[0][0] = A[0][0]\*\*0.5

for i in range(1, n):

R[0][i] = A[0][i] / R[0][0]

for i in range(1, n):

sum = 0

for k in range(i):

sum += R[k][i]\*\*2

R[i][i] = (A[i][i] - sum)\*\*0.5

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

sum = 0

for k in range(i):

sum += R[k][i] \* R[k][j]

R[i][j] = (A[i][j] - sum) / R[i][i]

print('Мatrix R')

matrixOutput(n, R)

Z = [0] \* n

X = [0] \* n

Z[0] = A[0][-1] / R[0][0]

for i in range(1, n):

sum = 0

for k in range(i):

sum += R[k][i] \* Z[k]

Z[i] = (A[i][-1] - sum) / R[i][i]

print('Мatrix Z')

rounding(n, Z)

X[-1] = Z[-1] / R[-1][-1]

for i in range(n - 2, -1, -1):

sum = 0

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

sum += R[i][j] \* X[j]

X[i] = (Z[i] - sum) / R[i][i]

print('Получим результат:')

rounding(n, X)

def main():

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

myA = matrixInput(n)

metod(n, myA)

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 30-60  
3) Организация этапа обратного хода и его вывод - строки 62-89

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 metod(n, A):

print('Метод квадратного корня')

print('Мatrix A')

matrixOutput(n, A)

R = [([0] \* n) for i in range(n)]

R[0][0] = A[0][0]\*\*0.5

for i in range(1, n):

R[0][i] = A[0][i] / R[0][0]

for i in range(1, n):

sum = 0

for k in range(i):

sum += R[k][i]\*\*2

R[i][i] = (A[i][i] - sum)\*\*0.5

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

sum = 0

for k in range(i):

sum += R[k][i] \* R[k][j]

R[i][j] = (A[i][j] - sum) / R[i][i]

print('Мatrix R')

matrixOutput(n, R)

Z = [0] \* n

X = [0] \* n

Z[0] = A[0][-1] / R[0][0]

for i in range(1, n):

sum = 0

for k in range(i):

sum += R[k][i] \* Z[k]

Z[i] = (A[i][-1] - sum) / R[i][i]

print('Мatrix Z')

rounding(n, Z)

X[-1] = Z[-1] / R[-1][-1]

for i in range(n - 2, -1, -1):

sum = 0

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

sum += R[i][j] \* X[j]

X[i] = (Z[i] - sum) / R[i][i]

print('Получим результат:')

rounding(n, X)

def main():

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

myA = matrixInput(n)

metod(n, myA)

main()

Метод квадратного корня является частным случаем факторизации матриц с использованием операции «квадратный корень».  
  
Важные черты данного метода:

* Есть дополнительные ограничения: A-симметричная матрица, и две треугольные матрицы (на которые раскладывается матрица A) имеют одинаковые диагональные элементы
* Значительно сокращено время на факторизацию матрицы, т.к. исходная матрица является симметричной.
* Характеризуется удобством обработки информации, т. к. при вычислении i-ой строки верхней правой матрицы требуется только i—ая строка матрицы А и все (i -1) -е верхние строки правой матрицы. Это особенно удобно при вычислениях на ЭВМ.

## 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 metod(n, A):

print("Метод квадратного корня")

print("Мatrix A")

matrixOutput(n, A)

R = [([0] \* n) for i in range(n)]

R[0][0] = A[0][0]\*\*0.5

for i in range(1, n):

R[0][i] = A[0][i] / R[0][0]

for i in range(1, n):

sum = 0

for k in range(i):

sum += R[k][i]\*\*2

R[i][i] = (A[i][i] - sum)\*\*0.5

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

sum = 0

for k in range(i):

sum += R[k][i] \* R[k][j]

R[i][j] = (A[i][j] - sum) / R[i][i]

print("Мatrix R")

matrixOutput(n, R)

Z = [0] \* n

X = [0] \* n

Z[0] = A[0][-1] / R[0][0]

for i in range(1, n):

sum = 0

for k in range(i):

sum += R[k][i] \* Z[k]

Z[i] = (A[i][-1] - sum) / R[i][i]

print("Мatrix Z")

rounding(n, Z)

X[-1] = Z[-1] / R[-1][-1]

for i in range(n - 2, -1, -1):

sum = 0

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

sum += R[i][j] \* X[j]

X[i] = (Z[i] - sum) / R[i][i]

print("Получим результат:")

rounding(n, X)

def main():

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

myA = matrixInput(n)

metod(n, myA)

main()

Особые черты данного метода:

* Есть дополнительные ограничения: A-симметричная матрица, и две треугольные матрицы имеют одинаковые диагональные элементы
* Характеризуется удобством обработки информации.

# **Заключение**

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

| Вводимые данные | Метод квадратного корня |
| --- | --- |
| [[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] |