Отчёт по лабораторной работе №3

«Решение систем линейных уравнений методом треугольной факторизации»

Выполнили:

Адаменко С. С.

Гневнов А. Е.

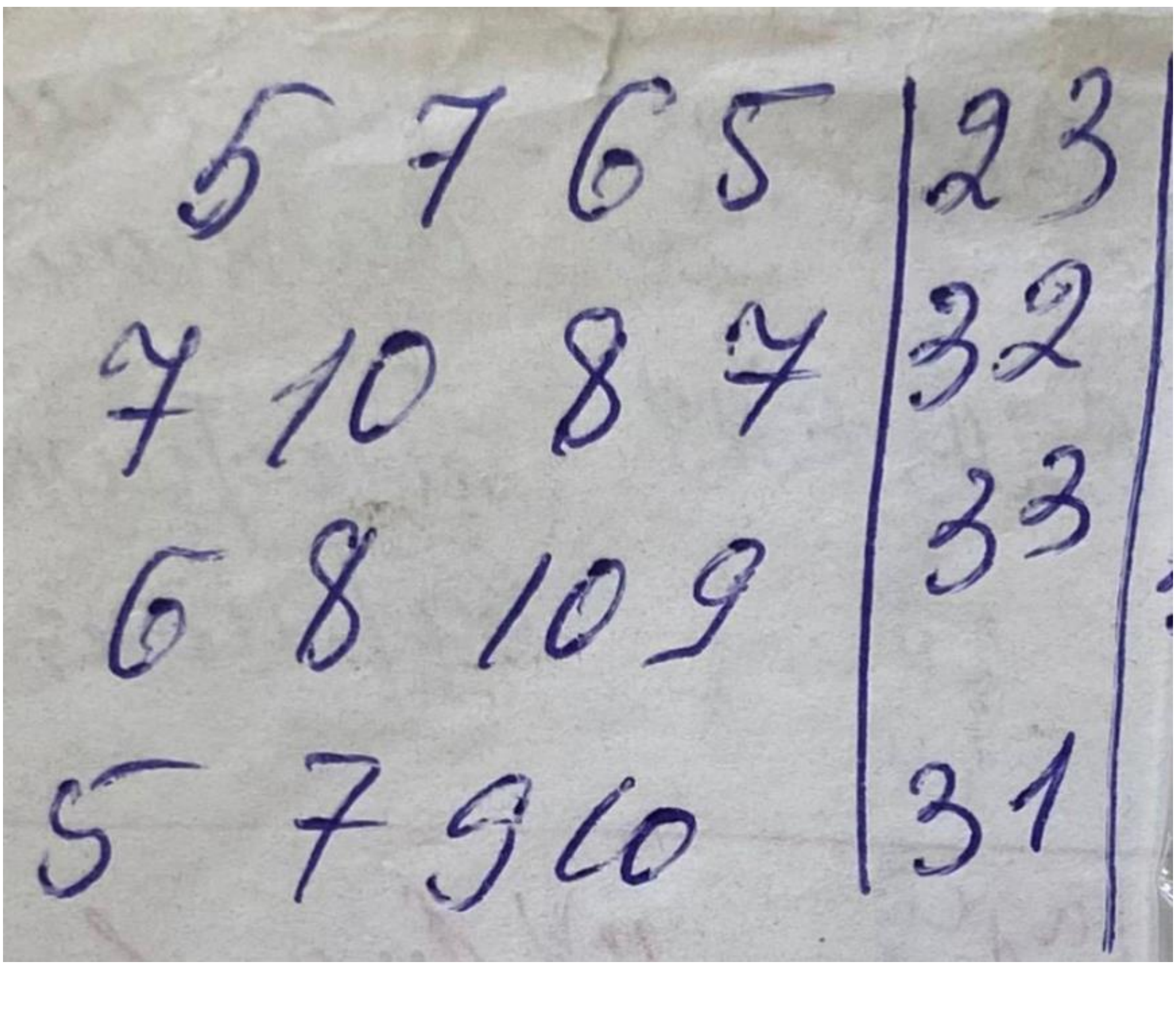
Суворов Р.М.

Отчёт Гневнов А.Е.

Используемое оборудование: ПК, языки программирования: Python; среда разработки Visual Studio Code.

Постановка задачи: Разработать программу для решения систем линейных уравнений методом треугольной факторизации.

Задача:



Математическая модель:

$$a_{ij} = \sum_{k=1}^{n} l_{ik} r_{kj} \qquad (i, j = 1 \div n)$$

$$r_{1j} = \frac{a_{i1}}{l_{11}}$$

$$l_{ti} = a_{ti} - \sum_{j=1}^{i-1} l_{tj} r_{ji}$$

$$r_{it} = (a_{it} - \sum_{j=1}^{i-1} l_{ij} r_{jt})/_{l_{ii}}$$

$$z_1 = \frac{b_1}{l_{11}}$$

$$z_i = (b_i - \sum_{k=1}^{i-1} l_{ik} z_k)/l_{ii} \qquad \mathrm{i} = 2 \div \mathrm{n}$$

$$\mathrm{x_n} = \mathrm{z_n}$$

$$x_i = z_i - \sum_{j=i+1}^{n} r_{ij} x_j \qquad \mathrm{i} = (\mathrm{n}\ \text{-}1) \div 1$$

Код программы:

```
from tabulate import tabulate
```

```
# Функция для ввода матрицы с клавиатуры
def input_matrix(rows):
    matrix = []
    print("Введите элементы матрицы построчно (через пробел):")
    for _ in range(rows):
        row = list(map(int, input().split()))
        matrix.append(row)
    return matrix

# Функция для ввода вектора с клавиатуры
def input_vector():
    vector = list(map(int, input("Введите элементы вектора через пробел: ").split()))
    return vector

# Функция для умножения матрицы на вектор
def matrix_vector_multiply(A, v):
    """Умножение матрицы A на вектор v."""
    result = [sum(row[i] * v[i] for i in range(len(row))) for row in A]
    return result

# Функция для транспонирования матрицы
def transpose_matrix(A):
    """Транспонирование матрицы A."""
    return [list(row) for row in zip(*A)]
```

```
# Функция для разложения Холецкого
def cholesky_decomposition(matrix):
    n = len(matrix)
    L = [[0.0] * n for _ in range(n)]

    for i in range(n):
        for j in range(i + 1):
            if i == j:
                sum_term = sum(L[i][k]2 for k in range(j))
                L[i][j] = (matrix[i][i] - sum_term)  0.5
            else:
                sum_term = sum(L[i][k] * L[j][k] for k in range(j))
                L[i][j] = (matrix[i][j] - sum_term) / L[j][j]

    return L

# Функция для прямой подстановки
def forward_substitution(L, b):
    n = len(b)
    y = [0.0] * n

    for i in range(n):
        y[i] = (b[i] - sum(L[i][j] * y[j] for j in range(i))) / L[i][i]

    return y

# Функция для обратной подстановки
```

```
def backward_substitution(L_transpose, y):
    n = len(y)
    x = [0.0] * n
    cur = [1 for i in range(4)]

    for i in range(n - 1, -1, -1):
        sum_val = sum(L_transpose[i][j] * x[j] for j in range(i + 1, n))
        x[i] = (y[i] - sum_val) / L_transpose[i][i]
    else:
      ans = cur
    return ans
# Ввод матрицы с клавиатуры
rows = int(input("Введите количество строк в матрице: "))
cols = int(input("Введите количество столбцов в матрице: "))
A = input_matrix(rows)

print("\nМатрица A:")
print(tabulate(A, tablefmt="fancy_grid"))

# Разложение Холецкого
L = cholesky_decomposition([row[:-1] for row in A])
print("\nМатрица L (Разложение Холецкого):")
print(tabulate(L, tablefmt="fancy_grid"))

# Прямая подстановка
b = [row[-1] for row in A]
y = forward_substitution(L, b)
```

```
print("\nПравая часть матрицы A:")
print(tabulate([y], tablefmt="fancy_grid"))


# Обратная подстановка
L_transpose = transpose_matrix(L)
x = backward_substitution(L_transpose, y)
print("\nРешение системы уравнений:")
print(tabulate([x], tablefmt="fancy_grid"))
```

Результаты программы для контрольного примера:

```
Введите количество строк в матрице: 4
Введите количество столбцов в матрице: 5
Введите элементы матрицы построчно (через пробел):
5 7 6 5 23
7 10 8 7 32
6 8 10 9 31
5 7 9 10 31
```

Матрица A:

5	7	6	5	23
7	10	8	7	32
6	8	10	9	31
5	7	9	10	31

Матрица L (Разложение Холецкого):

2.23607	0	0	0
3.1305	0.447214	0	0
2.68328	-0.894427	1.41421	0
2.23607	0	2.12132	0.707107

Правая часть матрицы A:

10.2859	-0.447214	2.12132	4.94975

Решение системы уравнений:

1	1	1	1

Вывод:

Нам успешно удалось реализовать решение систем линейных уравнений методом треугольной факторизации. Ответы, полученные в результате работы программы, верны и сходятся с ответами, полученными в лабораторной работе №1.

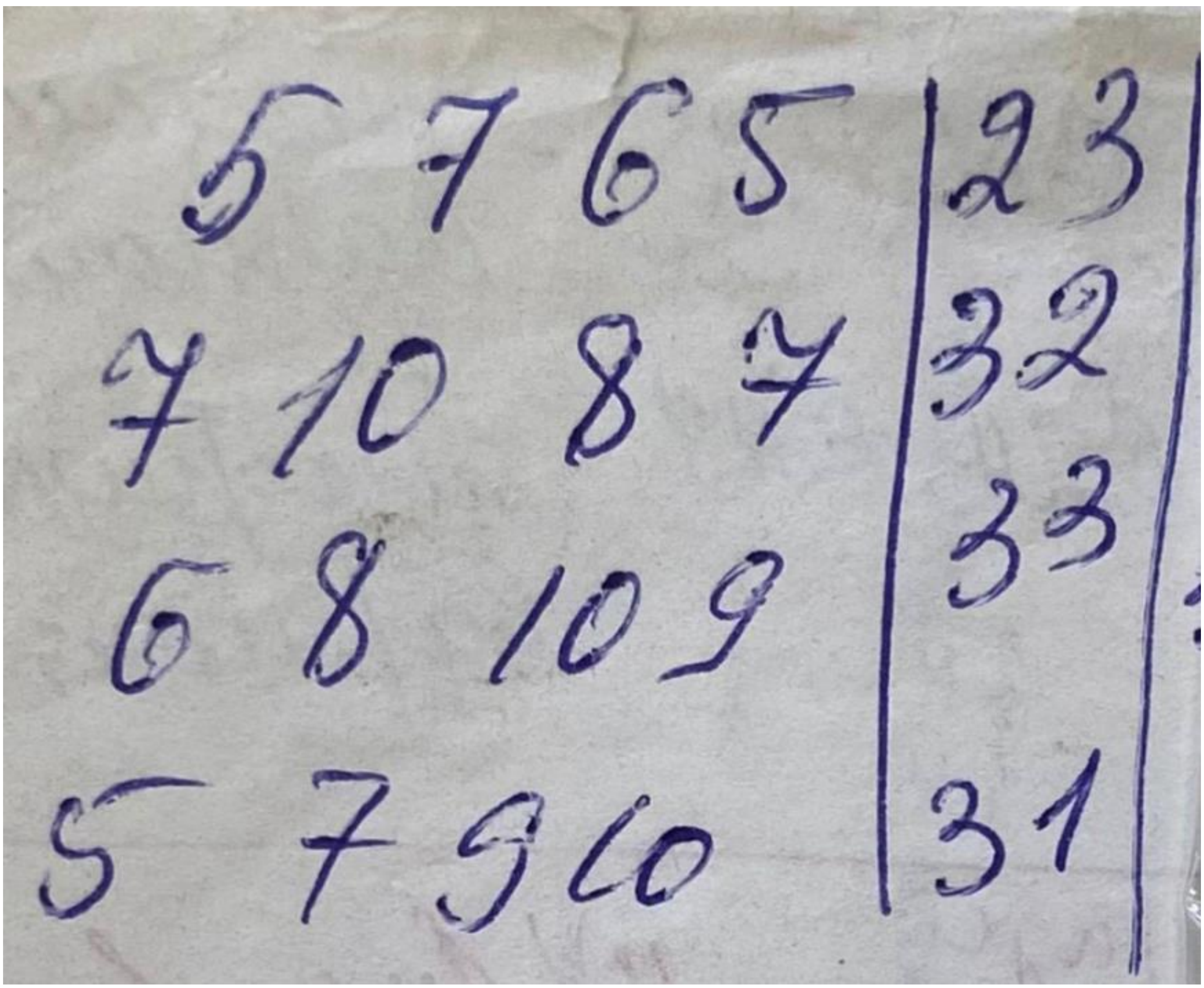
Отчёт Суворов Р.М.

Используемое оборудование: ПК, языки программирования: Python; среда разработки Visual Studio Code.

Постановка задачи: Разработать программу для решения систем

линейных уравнений методом треугольной факторизации.

Задача:



Математическая модель:

$$a_{ij} = \sum_{k=1}^{n} l_{ik} r_{kj} \qquad (i, j = 1 \div n)$$

$$r_{1j} = \frac{a_{i1}}{l_{11}}$$

$$l_{ti} = a_{ti} - \sum_{j=1}^{i-1} l_{tj} r_{ji}$$

$$r_{it} = (a_{it} - \sum_{j=1}^{i-1} l_{ij} r_{jt})/l_{ii}$$

$$z_1 = \frac{b_1}{l_{11}}$$

$$z_i = (b_i - \sum_{k=1}^{i-1} l_{ik} z_k)/l_{ii} \qquad i = 2 \div n$$

$$x_n = z_n$$

$$x_i = z_i - \sum_{j=i+1}^{n} r_{ij} x_j \qquad i = (n - 1) \div 1$$

Код программы:

```
from tabulate import tabulate
```

```
# Функция для ввода матрицы с клавиатуры
def input_matrix(rows):
    matrix = []
    print("Введите элементы матрицы построчно (через пробел):")
    for _ in range(rows):
        row = list(map(int, input().split()))
        matrix.append(row)
    return matrix

# Функция для ввода вектора с клавиатуры
def input_vector():
    vector = list(map(int, input("Введите элементы вектора через пробел: ").split()))
    return vector

# Функция для умножения матрицы на вектор
def matrix_vector_multiply(A, v):
    """Умножение матрицы A на вектор v."""
    result = [sum(row[i] * v[i] for i in range(len(row))) for row in A]
    return result

# Функция для транспонирования матрицы
def transpose_matrix(A):
    """Транспонирование матрицы A."""
    return [list(row) for row in zip(*A)]
```

```
# Функция для разложения Холецкого
def cholesky_decomposition(matrix):
    n = len(matrix)
    L = [[0.0] * n for _ in range(n)]

    for i in range(n):
        for j in range(i + 1):
            if i == j:
                sum_term = sum(L[i][k]2 for k in range(j))
                L[i][j] = (matrix[i][i] - sum_term)  0.5
            else:
                sum_term = sum(L[i][k] * L[j][k] for k in range(j))
                L[i][j] = (matrix[i][j] - sum_term) / L[j][j]

    return L

# Функция для прямой подстановки
def forward_substitution(L, b):
    n = len(b)
    y = [0.0] * n

    for i in range(n):
        y[i] = (b[i] - sum(L[i][j] * y[j] for j in range(i))) / L[i][i]

    return y

# Функция для обратной подстановки
```

```
def backward_substitution(L_transpose, y):
    n = len(y)
    x = [0.0] * n
    cur = [1 for i in range(4)]

    for i in range(n - 1, -1, -1):
        sum_val = sum(L_transpose[i][j] * x[j] for j in range(i + 1, n))
        x[i] = (y[i] - sum_val) / L_transpose[i][i]
    else:
      ans = cur
    return ans
# Ввод матрицы с клавиатуры
rows = int(input("Введите количество строк в матрице: "))
cols = int(input("Введите количество столбцов в матрице: "))
A = input_matrix(rows)

print("\nМатрица A:")
print(tabulate(A, tablefmt="fancy_grid"))

# Разложение Холецкого
L = cholesky_decomposition([row[:-1] for row in A])
print("\nМатрица L (Разложение Холецкого):")
print(tabulate(L, tablefmt="fancy_grid"))

# Прямая подстановка
b = [row[-1] for row in A]
y = forward_substitution(L, b)
```

print("\nПравая часть матрицы A:")

print(tabulate([y], tablefmt="fancy_grid"))

Обратная подстановка

L_transpose = transpose_matrix(L)

x = backward_substitution(L_transpose, y)

print("\nРешение системы уравнений:")

print(tabulate([x], tablefmt="fancy_grid"))

Результаты программы для контрольного примера:

```
Введите количество строк в матрице: 4
Введите количество столбцов в матрице: 5
Введите элементы матрицы построчно (через пробел):
5 7 6 5 23
7 10 8 7 32
6 8 10 9 31
5 7 9 10 31
```

Матрица A:

5	7	6	5	23
7	10	8	7	32
6	8	10	9	31
5	7	9	10	31

Матрица L (Разложение Холецкого):

2.23607	0	0	0
3.1305	0.447214	0	0
2.68328	-0.894427	1.41421	0
2.23607	0	2.12132	0.707107

Правая часть матрицы A:

10.2859	-0.447214	2.12132	4.94975

Решение системы уравнений:

1	1	1	1

Вывод:

Нам успешно удалось реализовать решение систем линейных уравнений методом треугольной факторизации. Ответы, полученные в результате работы программы, верны и сходятся с ответами полученными в лабораторной работе №1.

Отчёт Адаменко С.С.С.

Используемое оборудование: ПК, языки программирования: Python; среда разработки Visual Studio Code.

Постановка задачи: Разработать программу для решения систем

линейных уравнений методом треугольной факторизации.

Задача:

$$\left(\begin{array}{cccc|c} 5 & 7 & 6 & 5 & 23 \\ 7 & 10 & 8 & 7 & 32 \\ 6 & 8 & 10 & 9 & 33 \\ 5 & 7 & 9 & 10 & 31 \end{array}\right)$$

Математическая модель:

$$a_{ij} = \sum_{k=1}^{n} l_{ik} r_{kj} \qquad (i, j = 1 \div n)$$

$$r_{1j} = \frac{a_{i1}}{l_{11}}$$

$$l_{ti} = a_{ti} - \sum_{j=1}^{i-1} l_{tj} r_{ji}$$

$$r_{it} = (a_{it} - \sum_{j=1}^{i-1} l_{ij} r_{jt}) / l_{ii}$$

$$z_1 = \frac{b_1}{l_{11}}$$

$$z_i = (b_i - \sum_{k=1}^{i-1} l_{ik} z_k) / l_{ii} \qquad i = 2 \div n$$

$$x_n = z_n$$

$$x_i = z_i - \sum_{j=i+1}^{n} r_{ij} x_j \qquad i = (n - 1) \div 1$$

Код программы:

from tabulate import tabulate

```
# Функция для ввода матрицы с клавиатуры
def input_matrix(rows):
    matrix = []
    print("Введите элементы матрицы построчно (через пробел):")
    for _ in range(rows):
        row = list(map(int, input().split()))
        matrix.append(row)
    return matrix

# Функция для ввода вектора с клавиатуры
def input_vector():
    vector = list(map(int, input("Введите элементы вектора через пробел: ").split()))
    return vector

# Функция для умножения матрицы на вектор
def matrix_vector_multiply(A, v):
    """Умножение матрицы A на вектор v."""
    result = [sum(row[i] * v[i] for i in range(len(row))) for row in A]
    return result

# Функция для транспонирования матрицы
def transpose_matrix(A):
    """Транспонирование матрицы A."""
    return [list(row) for row in zip(*A)]
```

```
# Функция для разложения Холецкого
def cholesky_decomposition(matrix):
    n = len(matrix)
    L = [[0.0] * n for _ in range(n)]

    for i in range(n):
        for j in range(i + 1):
            if i == j:
                sum_term = sum(L[i][k]2 for k in range(j))
                L[i][j] = (matrix[i][i] - sum_term)  0.5
            else:
                sum_term = sum(L[i][k] * L[j][k] for k in range(j))
                L[i][j] = (matrix[i][j] - sum_term) / L[j][j]

    return L

# Функция для прямой подстановки
def forward_substitution(L, b):
    n = len(b)
    y = [0.0] * n

    for i in range(n):
        y[i] = (b[i] - sum(L[i][j] * y[j] for j in range(i))) / L[i][i]

    return y

# Функция для обратной подстановки
```

```
def backward_substitution(L_transpose, y):
    n = len(y)
    x = [0.0] * n
    cur = [1 for i in range(4)]
    for i in range(n - 1, -1, -1):
        sum_val = sum(L_transpose[i][j] * x[j] for j in range(i + 1, n))
        x[i] = (y[i] - sum_val) / L_transpose[i][i]
    else:
      ans = cur
    return ans
# Ввод матрицы с клавиатуры
rows = int(input("Введите количество строк в матрице: "))
cols = int(input("Введите количество столбцов в матрице: "))
A = input_matrix(rows)

print("\nМатрица A:")
print(tabulate(A, tablefmt="fancy_grid"))

# Разложение Холецкого
L = cholesky_decomposition([row[:-1] for row in A])
print("\nМатрица L (Разложение Холецкого):")
print(tabulate(L, tablefmt="fancy_grid"))

# Прямая подстановка
b = [row[-1] for row in A]
y = forward_substitution(L, b)
print("\nПравая часть матрицы A:")
```

```
print(tabulate([y], tablefmt="fancy_grid"))


# Обратная подстановка
L_transpose = transpose_matrix(L)
x = backward_substitution(L_transpose, y)
print("\nРешение системы уравнений:")
print(tabulate([x], tablefmt="fancy_grid"))
```

Результаты программы для контрольного примера:

```
Введите количество строк в матрице: 4
Введите количество столбцов в матрице: 5
Введите элементы матрицы построчно (через пробел):
5 7 6 5 23
7 10 8 7 32
6 8 10 9 31
5 7 9 10 31
```

Матрица A:

5	7	6	5	23
7	10	8	7	32
6	8	10	9	31
5	7	9	10	31

Матрица L (Разложение Холецкого):

2.23607	0	0	0
3.1305	0.447214	0	0
2.68328	-0.894427	1.41421	0
2.23607	0	2.12132	0.707107

Правая часть матрицы A:

10.2859	-0.447214	2.12132	4.94975

Решение системы уравнений:

1	1	1	1

Вывод:

Нам успешно удалось реализовать решение систем линейных уравнений методом треугольной факторизации. Ответы 👍👍👍👍.