РГПУ им. А.И. Герцена

**Отчет по лабораторной работе №1**

“Численные методы решения систем линейных алгебраических уравнений. Метод Гаусса”

Работу выполнили: Иванова Мария Алексеевна

Буряков Иван Олегович

Чернышева Виктория Викторовна

Волосатова Катя

Егор Собинин Яковлевич

Факультет 2об-ИВТ

Группа 1 гр. 2 п.гр

Оглавление

[Цель лабораторной работы:](#_heading=h.gjdgxs) **3**

[Используемое оборудование:](#_heading=h.30j0zll) **3**

[Постановка задачи:](#_heading=h.1fob9te) **3**

[Задание 1](#_heading=h.3znysh7) 3

[Задание 2](#_heading=h.2et92p0) 3

[Задание 3](#_heading=h.tyjcwt) 3

[Результат выполненной работы:](#_heading=h.3dy6vkm) **3**

[Отчет Ивановой Марии:](#_heading=h.1t3h5sf) 3

[Отчет Бурякова Ивана:](#_heading=h.exn2qj14yq7b) 7

[Отчет Чернышевой Виктории:](#_heading=h.9dc9riy5dyw6) 11

[Отчет Собинина Егора](#_heading=h.62j5jqm903ou) 16

[**Заключение:**](#_heading=h.robq1ehhqrib) **22**

# Цель лабораторной работы:

Научиться реализовывать решение систем линейных алгебраических уравнений методом Гаусса посредствам языка программирования.

# Используемое оборудование:

ПК, язык программирования Python.

# Постановка задачи:

## Задание 1

Разработать программу по решению СЛУ методом Гаусса (алгоритм исключения неизвестных по столбцам). В качестве образца используйте Программу из файла Gauss\_Klass.jpg 4.

## Задание 2

Модифицировать программу по решению СЛУ методом Гаусса для реализации алгоритма оптимального исключения неизвестных. В качестве образца используйте Программу из файла Gauss\_Optim\_Iskluch.jpg 4.

## Задание 3

Модифицировать программу по решению СЛУ методом Гаусса для реализации метода Гаусса-Жордана.

# Результат выполненной работы:

## Отчет Ивановой Марии:

Задание 1.

Код программы:

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 matrixOutput(n, a):

    for i in range(n):

        print(a[i])

    print()

def forwardStroke(n, a):

    print('Ход решения:')

    for i in range(n):

        p = a[i][i]

        a[i][i] = 1

        for j in range(i+1, n+1):

            a[i][j] = round(a[i][j]/p, 2)

        for k in range(i+1, n):

            for j in range(i+1, n+1):

                a[k][j] = round(a[k][j] - a[i][j]\*a[k][i], 2)

            a[k][i] = 0

        matrixOutput(n, a)

    return a

def reverseStroke(n, a):

    x = []

    for i in range(n):

        x.append(a[i][n])

    x[n-1] = round(a[n-1][n]/a[n-1][n-1], 2)

    for i in reversed(range(n-1)):

        sum = 0

        for j in range(i+1, n):

            sum = sum + a[i][j] \* x[j]

        x[i] = a[i][n] - sum

    return x

def main():

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

    myA = matrixInput(n)

    matrixOutput(n, myA)

    resA = forwardStroke(n, myA)

    x = reverseStroke(n, resA)

    print('Получим результат:')

    print(x)

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

    main()

Задание 2.

Код программы:

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 matrixOutput(n, a):

    for i in range(n):

        print(a[i])

    print()

def forwardStroke(n, a):

    print('Ход решения:')

    for i in range(n):

        p = a[i][i]

        a[i][i] = 1

        for j in range(i+1, n+1):

            a[i][j] = round(a[i][j]/p, 2)

        for k in range(i+1, n):

            for j in range(i+1, n+1):

                a[k][j] = round(a[k][j] - a[i][j]\*a[k][i], 2)

            a[k][i] = 0

        matrixOutput(n, a)

    return a

def reverseStroke(n, a):

    x = []

    for i in range(n):

        x.append(a[i][n])

    x[n-1] = round(a[n-1][n]/a[n-1][n-1], 2)

    for i in reversed(range(n-1)):

        sum = 0

        for j in range(i+1, n):

            sum = sum + a[i][j] \* x[j]

        x[i] = round((a[i][n] - sum)/a[i][i], 2)

    return x

def main():

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

    myA = matrixInput(n)

    matrixOutput(n, myA)

    resA = forwardStroke(n, myA)

    x = reverseStroke(n, resA)

    print('Получим результат:')

    print(x)

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

    main()

Задание 3.

Код программы:

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 matrixOutput(n, a):

    for i in range(n):

        print(a[i])

    print()

def forwardStroke(n, a):

    print('Ход решения:')

    for i in range(n):

        p = a[i][i]

        a[i][i] = 1

        for j in range(i+1, n+1):

            a[i][j] = round(a[i][j]/p, 2)

        for k in range(i+1, n):

            for j in range(i+1, n+1):

                a[k][j] = round(a[k][j] - a[i][j]\*a[k][i], 2)

            a[k][i] = 0

        matrixOutput(n, a)

    return a

def JordanStroke(n, a):

    print('Метода Гаусса-Жордана:')

    for i in reversed(range(1, n)):

        p = a[i][i]

        a[i][i] = 1

        for j in range(i+1, n+1):

            a[i][j] = round(a[i][j]/p, 2)

        for k in reversed(range(i)):

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

                a[k][j] = round(a[k][j] - a[i][j]\*a[k][i], 2)

            a[k][i] = 0

        matrixOutput(n, a)

    return a

def reverseStroke(n, a):

    x = []

    for i in range(n):

        x.append(a[i][n])

    return x

def main():

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

    myA = matrixInput(n)

    matrixOutput(n, myA)

    resA = forwardStroke(n, myA)

    res2A = JordanStroke(n, resA)

    print('Получим результат:')

    print(reverseStroke(n, res2A))

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

    main()

## Отчет Бурякова Ивана:

Задание 1.

Напишем код программы по решению СЛАУ методом Гаусса (алгоритм исключения неизвестных по столбцам):

Код программы:

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 matrixOutput(n, a):

    for i in range(n):

        print(a[i])

    print()

def forwardStroke(n, a):

    print('Ход решения:')

    for i in range(n):

        p = a[i][i]

        a[i][i] = 1

        for j in range(i+1, n+1):

            a[i][j] = round(a[i][j]/p, 2)

        for k in range(i+1, n):

            for j in range(i+1, n+1):

                a[k][j] = round(a[k][j] - a[i][j]\*a[k][i], 2)

            a[k][i] = 0

        matrixOutput(n, a)

    return a

def reverseStroke(n, a):

    x = []

    for i in range(n):

        x.append(a[i][n])

    x[n-1] = round(a[n-1][n]/a[n-1][n-1], 2)

    for i in reversed(range(n-1)):

        sum = 0

        for j in range(i+1, n):

            sum = sum + a[i][j] \* x[j]

        x[i] = a[i][n] - sum

    return x

def main():

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

    myA = matrixInput(n)

    matrixOutput(n, myA)

    resA = forwardStroke(n, myA)

    x = reverseStroke(n, resA)

    print('Получим результат:')

    print(x)

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

    main()

Задание 2.

Внесем немного изменений в код программы, а именно модифицируем программу по решению СЛАУ методом Гаусса для реализации алгоритма оптимального исключения неизвестных:

Код программы:

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 matrixOutput(n, a):

    for i in range(n):

        print(a[i])

    print()

def forwardStroke(n, a):

    print('Ход решения:')

    for i in range(n):

        p = a[i][i]

        a[i][i] = 1

        for j in range(i+1, n+1):

            a[i][j] = round(a[i][j]/p, 2)

        for k in range(i+1, n):

            for j in range(i+1, n+1):

                a[k][j] = round(a[k][j] - a[i][j]\*a[k][i], 2)

            a[k][i] = 0

        matrixOutput(n, a)

    return a

def reverseStroke(n, a):

    x = []

    for i in range(n):

        x.append(a[i][n])

    x[n-1] = round(a[n-1][n]/a[n-1][n-1], 2)

    for i in reversed(range(n-1)):

        sum = 0

        for j in range(i+1, n):

            sum = sum + a[i][j] \* x[j]

        x[i] = round((a[i][n] - sum)/a[i][i], 2)

    return x

def main():

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

    myA = matrixInput(n)

    matrixOutput(n, myA)

    resA = forwardStroke(n, myA)

    x = reverseStroke(n, resA)

    print('Получим результат:')

    print(x)

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

    main()

Задание 3.

Внесем последние изменения в код, а именно модифицируем программу по решению СЛАУ методом Гаусса для реализации метода Гаусса-Жордана:

Код программы:

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 matrixOutput(n, a):

    for i in range(n):

        print(a[i])

    print()

def forwardStroke(n, a):

    print('Ход решения:')

    for i in range(n):

        p = a[i][i]

        a[i][i] = 1

        for j in range(i+1, n+1):

            a[i][j] = round(a[i][j]/p, 2)

        for k in range(i+1, n):

            for j in range(i+1, n+1):

                a[k][j] = round(a[k][j] - a[i][j]\*a[k][i], 2)

            a[k][i] = 0

        matrixOutput(n, a)

    return a

def JordanStroke(n, a):

    print('Метода Гаусса-Жордана:')

    for i in reversed(range(1, n)):

        p = a[i][i]

        a[i][i] = 1

        for j in range(i+1, n+1):

            a[i][j] = round(a[i][j]/p, 2)

        for k in reversed(range(i)):

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

                a[k][j] = round(a[k][j] - a[i][j]\*a[k][i], 2)

            a[k][i] = 0

        matrixOutput(n, a)

    return a

def reverseStroke(n, a):

    x = []

    for i in range(n):

        x.append(a[i][n])

    return x

def main():

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

    myA = matrixInput(n)

    matrixOutput(n, myA)

    resA = forwardStroke(n, myA)

    res2A = JordanStroke(n, resA)

    print('Получим результат:')

    print(reverseStroke(n, res2A))

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

    main()

## Отчет Чернышевой Виктории:

Задача 1:

Рассмотрим код программы:  
1) Введенные исходные данные - строки 1-13  
2) Организация этапа прямого хода - строки 15-29  
3) Организация этапа обратного хода и его вывод -. строки 31-53

    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 matrixOutput(n, a):

    for i in range(n):

        print(a[i])

    print()

def forwardStroke(n, a):

    print('Ход решения:')

    for i in range(n):

        p = a[i][i]

        a[i][i] = 1

        for j in range(i+1, n+1):

            a[i][j] = round(a[i][j]/p, 2)

        for k in range(i+1, n):

            for j in range(i+1, n+1):

                a[k][j] = round(a[k][j] - a[i][j]\*a[k][i], 2)

            a[k][i] = 0

        matrixOutput(n, a)

    return a

def reverseStroke(n, a):

    x = []

    for i in range(n):

        x.append(a[i][n])

    x[n-1] = round(a[n-1][n]/a[n-1][n-1], 2)

    for i in reversed(range(n-1)):

        sum = 0

        for j in range(i+1, n):

            sum = sum + a[i][j] \* x[j]

        x[i] = a[i][n] - sum

    return x

def main():

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

    myA = matrixInput(n)

    matrixOutput(n, myA)

    resA = forwardStroke(n, myA)

    x = reverseStroke(n, resA)

    print('Получим результат:')

    print(x)

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

    main()

Метод Гаусса последовательного исключения неизвестных наиболее распространенный из прямых численных методов решения СЛАУ.  
Преимуществами метода являются:

1. возможность решать неопределенные системы линейных уравнений, то есть, имеющие общее решение;
2. возможность решать системы линейных уравнений, в которых число неизвестных не равно числу уравнений;
3. метод основан на элементарных методах - методе подстановки неизвестных и методе сложения уравнений.

Также у метода есть существенные недостатки:

1. Невозможность сформулировать условия совместности и определенности системы в зависимости от значений коэффициентов и свободных членов.
2. Метод Гаусса требует, чтобы диагональные элементы в процессе исключения переменных не были равны нулю (т.к. строки делятся на них)
3. Метод Гаусса требует большего объема памяти ЭВМ по сравнению с итерационными методами.

Задача 2:  
Рассмотрим код программы:  
1) Введенные исходные данные - строки 1-13  
2) Организация этапа прямого хода - строки 15-29  
3) Организация этапа обратного хода и его вывод -. строки 31-53

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 matrixOutput(n, a):

    for i in range(n):

        print(a[i])

    print()

def forwardStroke(n, a):

    print('Ход решения:')

    for i in range(n):

        p = a[i][i]

        a[i][i] = 1

        for j in range(i+1, n+1):

            a[i][j] = round(a[i][j]/p, 2)

        for k in range(i+1, n):

            for j in range(i+1, n+1):

                a[k][j] = round(a[k][j] - a[i][j]\*a[k][i], 2)

            a[k][i] = 0

        matrixOutput(n, a)

    return a

def reverseStroke(n, a):

    x = []

    for i in range(n):

        x.append(a[i][n])

    x[n-1] = round(a[n-1][n]/a[n-1][n-1], 2)

    for i in reversed(range(n-1)):

        sum = 0

        for j in range(i+1, n):

            sum = sum + a[i][j] \* x[j]

        x[i] = round((a[i][n] - sum)/a[i][i], 2)

    return x

def main():

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

    myA = matrixInput(n)

    matrixOutput(n, myA)

    resA = forwardStroke(n, myA)

    x = reverseStroke(n, resA)

    print('Получим результат:')

    print(x)

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

    main()

В данном методе на этапе прямого хода выполняется на n операций деления меньше, чем в методе последовательного исключения, однако этот выигрыш является кажущимся, т.к. на втором этапе обратного хода требуется ровно на n операций деления больше.

Таким образом, метод оптимального исключения по временным затратам решения СЛАУ эквивалентен классическому методу последовательного исключения неизвестных, но по организации вычислений он проще.

Задача 3:

Рассмотрим код программы, представленный на слайде:  
1) Введенные исходные данные - строки 1-13  
2) Организация этапа прямого хода - строки 31-46  
3) Организация этапа обратного хода и его вывод -. строки 48-63

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 matrixOutput(n, a):

    for i in range(n):

        print(a[i])

    print()

def forwardStroke(n, a):

    print('Ход решения:')

    for i in range(n):

        p = a[i][i]

        a[i][i] = 1

        for j in range(i+1, n+1):

            a[i][j] = round(a[i][j]/p, 2)

        for k in range(i+1, n):

            for j in range(i+1, n+1):

                a[k][j] = round(a[k][j] - a[i][j]\*a[k][i], 2)

            a[k][i] = 0

        matrixOutput(n, a)

    return a

def JordanStroke(n, a):

    print('Метода Гаусса-Жордана:')

    for i in reversed(range(1, n)):

        p = a[i][i]

        a[i][i] = 1

        for j in range(i+1, n+1):

            a[i][j] = round(a[i][j]/p, 2)

        for k in reversed(range(i)):

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

                a[k][j] = round(a[k][j] - a[i][j]\*a[k][i], 2)

            a[k][i] = 0

        matrixOutput(n, a)

    return a

def reverseStroke(n, a):

    x = []

    for i in range(n):

        x.append(a[i][n])

    return x

def main():

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

    myA = matrixInput(n)

    matrixOutput(n, myA)

    resA = forwardStroke(n, myA)

    res2A = JordanStroke(n, resA)

    print('Получим результат:')

    print(reverseStroke(n, res2A))

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

    main()

Данная модификация характеризуется тем, что в каждом цикле преобразования i-го столбца преобразуются элементы всех ведомых строк, расположенных над и под диагональю по единому алгоритму прямого хода. Не преобразуются элементы только ведущей строки.

Сравнивая данный метод с предыдущими задачами, можно заметить следующее:  
1. матрица приводится не к треугольному, а к диагональному виду.

2. Преобразование элементов матрицы и элементов столбца свободных членов производится по аналогичному Гаусса методу оптимального исключения неизвестных по столбцам.

## Отчет Собинина Егора

Задача 1:

Разработаем программу по решению СЛАУ методом Гаусса, а именно с помощью алгоритма исключения неизвестных по столбцам:

#ввод значений

def matrixInput(n):

m = []

print("Вводите элементы ряда через пробел")

for i in range(n):

mi = list(map(float, input("\nЭлементы ряда:").split()))

m.append(mi)

return m

#вывод готовой матрицы

def matrixOutput(n, m):

for i in range(n):

print()

print(m[i])

#прямой ход

def forwardMatrix(n, m):

print("\nХод решения:")

for i in range(n):

p = m[i][i]

m[i][i] = 1

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

m[i][j] = round(m[i][j]/p, 2)

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

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

m[k][j] = round(m[k][j] - m[i][j]\*m[k][i], 2)

m[k][i] = 0

matrixOutput(n, m)

return m

#обратный ход

def reverseMatrix(n, m):

x = []

for i in range(n):

x.append(m[i][n])

x[n-1] = round(m[n-1][n]/m[n-1][n-1], 2)

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

sum = 0

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

sum = sum + m[i][j] \* x[j]

x[i] = m[i][n] - sum

return x

#вызов условия

def main():

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

M = matrixInput(n)

matrixOutput(n, M)

Mresult = forwardMatrix(n, M)

x = reverseMatrix(n, Mresult)

print("\nПолучим результат:\n")

print(x)

#инициализация

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

main()

Этот метод - один из самых распространенных. Он обладает рядом преимуществ и недостатков.

Преимущетсва:

1. Возможность решать системы уравнений имеющих общее решение
2. Возможность решать системы уравнений, где число неизвестных ≠ число уравнений
3. Метод прост в освоении

Недостатки:

1. Диагональные элементы в процессе исключения ≠ 0
2. Метод требует больших ресурсов при реализации на компьютере

Задача 2:

Для выполнения этой задачи нам необходимо немного переработать код из прошлой:

#ввод значений

def matrixInput(n):

m = []

print("Вводите элементы ряда через пробел")

for i in range(n):

mi = list(map(float, input("\nЭлементы ряда:").split()))

m.append(mi)

return m

#вывод готовой матрицы

def matrixOutput(n, m):

for i in range(n):

print()

print(m[i])

#прямой ход

def forwardMatrix(n, m):

print("\nХод решения:")

for i in range(n):

p = m[i][i]

m[i][i] = 1

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

m[i][j] = round(m[i][j]/p, 2)

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

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

m[k][j] = round(m[k][j] - m[i][j]\*m[k][i], 2)

m[k][i] = 0

matrixOutput(n, m)

return m

#обратный ход

def reverseMatrix(n, m):

x = []

for i in range(n):

x.append(m[i][n])

x[n-1] = round(m[n-1][n]/m[n-1][n-1], 2)

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

sum = 0

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

sum = sum + m[i][j] \* x[j]

x[i] = round((m[i][n] - sum)/m[i][i], 2)

return x

#вызов условия

def main():

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

M = matrixInput(n)

matrixOutput(n, M)

Mresult = forwardMatrix(n, M)

x = reverseMatrix(n, Mresult)

print("\nПолучим результат:\n")

print(x)

#инициализация

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

main()

Данный метод требует меньших ресурсов на прямом ходе, но больше на обратном. Поэтому с точки зрения использования ресурсов компьютера абсолютно не важно, какой из методов мы будем использовать.

Задача 3:

Для решения этой задачи требуются несколько более заметные изменения. Однако для решения мы будем продолжать модифицировать программу по решению СЛАУ методом Гаусса:

#ввод значений

def matrixInput(n):

m = []

print("Вводите элементы ряда через пробел")

for i in range(n):

mi = list(map(float, input("\nЭлементы ряда:").split()))

m.append(mi)

print()

return m

#вывод готовой матрицы

def matrixOutput(n, m):

for i in range(n):

print(m[i])

print()

#прямой ход

def forwardMatrix(n, m):

print("\nХод решения:")

for i in range(n):

p = m[i][i]

m[i][i] = 1

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

m[i][j] = round(m[i][j]/p, 2)

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

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

m[k][j] = round(m[k][j] - m[i][j]\*m[k][i], 2)

m[k][i] = 0

matrixOutput(n, m)

return m

#Метод Гаусса-Жордана

def JordanStroke(n, m):

print('Метода Гаусса-Жордана:')

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

p = m[i][i]

m[i][i] = 1

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

m[i][j] = round(m[i][j]/p, 2)

for k in reversed(range(i)):

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

m[k][j] = round(m[k][j] - m[i][j]\*m[k][i], 2)

m[k][i] = 0

matrixOutput(n, m)

return m

#обратный ход

def reverseMatrix(n, m):

x = []

for i in range(n):

x.append(m[i][n])

return x

#вызов условия

def main():

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

M = matrixInput(n)

matrixOutput(n, M)

Mresult = forwardMatrix(n, M)

M2result = JordanStroke(n, Mresult)

print("Получим результат:\n")

print(reverseMatrix(n, M2result))

#инициализация

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

main()

Этот способ решения преобразует матрицу не к виду “треугольника”, а образует ряд единиц на главной диагонали, а все остальное преобразуется в нули. Однако само преобразование происходит аналогичным методом.

# Заключение:

Мы проверили правильность работы программ из задач на конкретных примерах вводимых данных. Результаты представлены на слайде в таблице. С каждым прогоном программы мы изменяли значение элементов столбца свободных членов, придавая им более высокую точность. Данные махинации не придали методам разные результаты.

| Вводимые данные | Метод Гаусса классический | Метод Гаусса алгоритма оптимального исключения неизвестных | Метода Гаусса-Жордана |
| --- | --- | --- | --- |
| [[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] | [1.0, 1.0, 1.0, 1.0] | [1.0, 1.0, 1.0, 1.0] |
| [[5, 7, 6, 5, 23.1],  [7, 10, 8, 7, 32.1],  [6, 8, 10, 9, 33.1],  [5, 7, 9, 10, 31.1]] | [3.0, -0.2, 0.5, 1.3] | [3.0, -0.2, 0.5, 1.3] | [3.0, -0.2, 0.5, 1.3] |
| [[5, 7, 6, 5, 23.01],  [7, 10, 8, 7, 32.01],  [6, 8, 10, 9, 33.01],  [5, 7, 9, 10, 31.01]] | [0.35, 1.39, 1.17, 0.9] | [0.35, 1.39, 1.17, 0.9] | [0.35, 1.39, 1.17, 0.9] |
| [[5, 7, 6, 5, 23.001],  [7, 10, 8, 7, 32.001],  [6, 8, 10, 9, 33.001],  [5, 7, 9, 10, 31.001]] | [1.0, 1.0, 1.0, 1.0] | [1.0, 1.0, 1.0, 1.0] | [1.0, 1.0, 1.0, 1.0] |