РГПУ им. А.И. Герцена

**Отчет по лабораторной работе №2**

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

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:**](#_3yrqr1w8x6vh) **3**

[Используемое оборудование:](#_30j0zll) **3**

[Постановка задачи:](#_1fob9te) **3**

[Задание 1](#_3znysh7) 3

[Задание 2](#_2et92p0) 3

[Задание 3](#_tyjcwt) 3

[Результат выполненной работы:](#_3dy6vkm) **3**

[Отчет Ивановой Марии:](#_jvkqr83o5oqp) 3

[**Заключение:**](#_6ecejgkkq48w) **9**

# 

# 

# **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

Модифицировать программу по решению СЛУ методом Гаусса для реализации метода Гаусса-Жордана. В качестве образца используйте Программу из файла Gauss\_Jordana.jpg (Папка Код)

## Задание 2

Модифицировать программу по решению СЛУ методом Гаусса для реализации алгоритма c выбором главного элемента (по строкам, по столбцам, по всей матрице)

## Задание 3

Модифицировать программу по решению СЛУ методом Гаусса для реализации алгоритма обратного хода с распараллеливанием вычислений. (алгоритм на стр. 4 Лекции) В качестве образца используйте Программу из файла Gauss\_Jordana.jpg

# **Результат выполненной работы:**

## Отчет Ивановой Марии:

Для начала реализуем ввод и вывод матрицы. А после меню выбора метода решения.

По итогу будет получен следующий главный файл:

import MainElement

import GaussJordan

import Parallel

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 menu(n,myA):

print("Выбирите способ решения:")

print("1. Метода Гаусса-Жордана")

print("2. Классический метод Гаусса с выбором главного элемента по столбцам")

print("3. Классический метод Гаусса с распараллеливанием вычислений")

user\_input = int(input())

if user\_input == 1:

GaussJordan.main(n,myA)

elif user\_input == 2:

MainElement.main(n,myA)

elif user\_input == 3:

Parallel.main(n,myA)

elif user\_input == 4:

pass

else:

print("Введите число от 1 до 3.\n")

menu()

def main():

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

myA = matrixInput(n)

matrixOutput(n,myA)

menu(n,myA)

main()

## 

Задание 1

Рассмотрим код программы из файла Gauss\_Jordana.jpg и реализуем его.

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,myA):

resA = forwardStroke(n,myA)

res2A = JordanStroke(n,resA)

print('Получим результат:')

print(reverseStroke(n,res2A))

Задание 2

Модифицируем программу для реализации алгоритма с выбором главного элемента по столбцам.

def matrixOutput(n,a):

for i in range(n):

print(a[i])

print()

def forwardStroke(n,a):

print("Алгоритм с выбором главного элемента по столбцам\nХод решения:\n")

q = 0

for p in range(q+1, n):

if a[q]<a[p]:

a[q], a[p] = a[p], a[q]

q+=1

for i in range(n):

p = a[i][i]

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

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

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

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

matrixOutput(n,a)

return a

def reverseStroke(n,a):

res = [0] \* n

res[-1] = a[-1][-1]

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

s = 0

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

s += a[i][j] \* res[j]

res[i] = round(a[i][-1] - s, 2)

return res

def main(n,myA):

resA = forwardStroke(n,myA)

print('Получим результат:')

print(reverseStroke(n,resA))

Задание 3

Модифицируем программу для реализации алгоритма обратного хода с распараллеливанием вычислений.

Для этого реализуем алгоритм на странице 4 лекции.

def matrixOutput(n,a):

for i in range(n):

print(a[i])

print()

def forwardStroke(n,a):

print('Алгоритм с распараллеливанием вычислений\nХод решения:\n')

for i in range(n):

p = a[i][i]

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

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

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

a[k][j] = a[k][j] - a[i][j]\*a[k][i]

matrixOutput(n,a)

return a

def reverseStroke(n,a):

res = [0] \* n

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

res[i] = a[i][-1]

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

a[k][-1] = round(a[k][-1] - a[k][i] \* res[i], 10)

a[k][i] = 0

return res

def main(n,myA):

resA = forwardStroke(n,myA)

print('Получим результат:')

print(reverseStroke(n,resA))

## Отчет Чернышевой Виктории:

**Предварительная подготовка**: Для начала работы необходимо реализовать ввод и вывод матрицы. Далее меню выбора метода решения.  
В итоге будет получен главный файл:

import MainElement

import GaussJordan

import Parallel

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 menu(n,myA):

print("Выбирите способ решения:")

print("1. Метода Гаусса-Жордана")

print("2. Классический метод Гаусса с выбором главного элемента по столбцам")

print("3. Классический метод Гаусса с распараллеливанием вычислений")

user\_input = int(input())

if user\_input == 1:

GaussJordan.main(n,myA)

elif user\_input == 2:

MainElement.main(n,myA)

elif user\_input == 3:

Parallel.main(n,myA)

elif user\_input == 4:

pass

else:

print("Введите число от 1 до 3.\n")

menu()

def main():

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

myA = matrixInput(n)

matrixOutput(n,myA)

menu(n,myA)

main()

**Задание 1**

Рассмотрим код программы:  
1) Введенные исходные данные - строки 1-4  
2) Организация этапа прямого хода - строки 6-32  
3) Организация этапа обратного хода и его вывод - строки 34-46

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,myA):

resA = forwardStroke(n,myA)

res2A = JordanStroke(n,resA)

print('Получим результат:')

print(reverseStroke(n,res2A))

Этот способ решения образует ряд единиц на главной диагонали, а все остальное преобразуется в нули.

**Задание 2**

Рассмотрим код программы:  
1) Введенные исходные данные - строки 1-4  
2) Организация этапа прямого хода - строки 6-22  
3) Организация этапа обратного хода и его вывод - строки 24-39

def matrixOutput(n,a):

for i in range(n):

print(a[i])

print()

def forwardStroke(n,a):

print("Алгоритм с выбором главного элемента по столбцам\nХод решения:\n")

q = 0

for p in range(q+1, n):

if a[q]<a[p]:

a[q], a[p] = a[p], a[q]

q+=1

for i in range(n):

p = a[i][i]

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

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

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

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

matrixOutput(n,a)

return a

def reverseStroke(n,a):

res = [0] \* n

res[-1] = a[-1][-1]

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

s = 0

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

s += a[i][j] \* res[j]

res[i] = round(a[i][-1] - s, 2)

return res

def main(n,myA):

resA = forwardStroke(n,myA)

print('Получим результат:')

print(reverseStroke(n,resA))

Мы модифицировали программу для реализации алгоритма с выбором главного элемента по столбцам.  
В данном алгоритме преобразование идет по столбцу. Вычислив первый раз значение корня и далее двигаясь вверх по столбцу, в котором он расположен, с его помощью преобразуем одновременно элементы столбца свободных членов.

**Задание 3**

Рассмотрим код программы, представленный на слайде:  
1) Введенные исходные данные - строки 1-4  
2) Организация этапа прямого хода - строки 6-17  
3) Организация этапа обратного хода и его вывод - строки 19-35

def matrixOutput(n,a):

for i in range(n):

print(a[i])

print()

def forwardStroke(n,a):

print('Алгоритм с распараллеливанием вычислений\nХод решения:\n')

for i in range(n):

p = a[i][i]

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

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

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

a[k][j] = a[k][j] - a[i][j]\*a[k][i]

matrixOutput(n,a)

return a

def reverseStroke(n,a):

res = [0] \* n

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

res[i] = a[i][-1]

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

a[k][-1] = round(a[k][-1] - a[k][i] \* res[i], 10)

a[k][i] = 0

return res

def main(n,myA):

resA = forwardStroke(n,myA)

print('Получим результат:')

print(reverseStroke(n,resA))

Мы модифицируем программу для реализации алгоритма обратного хода с распараллеливанием вычислений.  
  
Если провести анализ эффективности данного метода и рассмотреть общую оценку показателей ускорения и эффективности, мы увидим что балансировка вычислительной нагрузки между процессорами является достаточно равномерной. Таким образом, можно говорить о эффективности программы.

* 1. Отчет Собинина Егора

Для удобства работы, разделим программу на несколько файлов: файл ввода данных и 3 файла с вычислениями.

import MainElement

import GaussJordan

import Parallel

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 menu(n,myA):

print("Выбирите способ решения:")

print("1. Метода Гаусса-Жордана")

print("2. Классический метод Гаусса с выбором главного элемента по столбцам")

print("3. Классический метод Гаусса с распараллеливанием вычислений")

user\_input = int(input())

if user\_input == 1:

GaussJordan.main(n,myA)

elif user\_input == 2:

MainElement.main(n,myA)

elif user\_input == 3:

Parallel.main(n,myA)

elif user\_input == 4:

pass

else:

print("Введите число от 1 до 3.\n")

menu()

def main():

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

myA = matrixInput(n)

matrixOutput(n,myA)

menu(n,myA)

main()

**Задание 1**

Пример кода программы метода Жордана-Гаусса

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,myA):

resA = forwardStroke(n,myA)

res2A = JordanStroke(n,resA)

print('Получим результат:')

print(reverseStroke(n,res2A))

Этот способ решения образует ряд единиц на главной диагонали, а все остальное преобразуется в нули.

**Задание 2**

Пример кода программы с использованием метода выбора главного элемента:

def matrixOutput(n,a):

for i in range(n):

print(a[i])

print()

def forwardStroke(n,a):

print("Алгоритм с выбором главного элемента по столбцам\nХод решения:\n")

q = 0

for p in range(q+1, n):

if a[q]<a[p]:

a[q], a[p] = a[p], a[q]

q+=1

for i in range(n):

p = a[i][i]

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

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

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

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

matrixOutput(n,a)

return a

def reverseStroke(n,a):

res = [0] \* n

res[-1] = a[-1][-1]

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

s = 0

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

s += a[i][j] \* res[j]

res[i] = round(a[i][-1] - s, 2)

return res

def main(n,myA):

resA = forwardStroke(n,myA)

print('Получим результат:')

print(reverseStroke(n,resA))

Мы модифицировали программу для реализации алгоритма с выбором главного элемента по столбцам.  
В этом алгоритме преобразование идет по столбцу. Вычислив первое значение корня и далее двигаясь вверх по столбцу, в котором он расположен, таким образом преобразуем одновременно элементы столбца свободных членов.

**Задание 3**

Пример кода программы с использованием метода обратного хода с параллельными вычислениями:

def matrixOutput(n,a):

for i in range(n):

print(a[i])

print()

def forwardStroke(n,a):

print('Алгоритм с распараллеливанием вычислений\nХод решения:\n')

for i in range(n):

p = a[i][i]

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

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

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

a[k][j] = a[k][j] - a[i][j]\*a[k][i]

matrixOutput(n,a)

return a

def reverseStroke(n,a):

res = [0] \* n

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

res[i] = a[i][-1]

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

a[k][-1] = round(a[k][-1] - a[k][i] \* res[i], 10)

a[k][i] = 0

return res

def main(n,myA):

resA = forwardStroke(n,myA)

print('Получим результат:')

print(reverseStroke(n,resA))

Мы модифицируем программу для реализации алгоритма обратного хода с распараллеливанием вычислений.  
  
Проведя анализ эффективности данного метода, мы увидим что он является одним из самых эффективных, благодаря распараллеливанию процессов. Таким образом, можно говорить о эффективности программы.

# **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-Жордана | Метод Гаусса для реализации алгоритма с выбором главного элемента | Методом Гаусса для реализации алгоритма обратного хода с распараллеливанием вычислений |
| --- | --- | --- | --- |
| [[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] |