Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 3

по курсу «Численные методы линейной алгебры»

«Реализация метода Гаусса с перестановками»

Студентка группы ИУ9-72Б Самохвалова П. С.

Преподаватель Посевин Д. П.

Москва 2023

1 Цель работы

Реализовать три варианта метода Гаусса с перестановками и научиться оценивать погрешность решения системы линейных уравнений для матриц произвольной размерности.

2 Задание

- Реализовать метод Гаусса с перестановками по столбцам, по строкам, по столбцам и строкам одновременно для действительных квадратных матриц произвольной размерности n.

- Для проверки работоспособности алгоритмов необходимо использовать алгоритм тестирования задачи написанный в лабораторной работе №2 «Реализация метода Гаусса», который заключался в том, что мы заведомо определяем значения координат вектора x, данный вектор заведомо является решением уравнения A·x = b, вычисляем b путем прямого перемножения матрицы A на вектор x и далее производим поиск решения уравнения A·x = b тем или иным методом Гаусса, получая x_{num}, после чего производим сравнение полученного x_{num} с заданным x, а также решением x_{lib}, полученным с использованием сторонней библиотеки выбранной студеном. При этом сравнение производится по Евклидовой норме разности вектора x - x_{num} и x - x_{lib}.

- На защите лабораторной работы студент должен показать умение оценивать погрешность вычислений в зависимости от выполнения условия диагонального преобладания матрицы, умение сравнивать погрешности вычислений полученных методом Гаусса с перестановками по столбцам, по строкам, по столбцам и строкам одновременно. Понимать связь теории с практикой.

- Результат работы должен быть представлен в виде графиков зависимости абсолютной погрешности вычислений классическим методом Гаусса, методом Гаусса с перестановками по строкам, методом Гаусса с перестановками по столбцам, методом Гаусса с перестановками по столбцам и

строкам, библиотечным методом от степени диагонального преобладания. Все графики должны быть построены на одной координатной плоскости. Напомним, что погрешность вычисления вектора x системы линейных алгебраических уравнений A·x=b тем или иным способом рассчитывается по Евклидовой норме разности точного решения и решения полученного соответствующим методом. Степень диагонального преобладания вычисляется, как максимальная разность по i между модулем диагонального элемента и суммы модулей вне диагональных элементов. Очевидно, что если значение степени диагонального преобладания положительна, то условие диагонального преобладания выполняется, в противном случае — не выполняется. Поэтому график должен быть построен как для отрицательных значений степени диагонального преобладания, так и для положительных.

3 Практическая реализация

Исходный код программы представлен в листинге 1.

Листинг 1: Метод Гаусса

```
from num_methods import *
import random
import numpy as np
import copy
from matplotlib import pyplot as plt


def gauss_method(a, b):
    n = len(a)
    for i in range(n):
        for j in range(i + 1, n):
            c = - a[j][i] / a[i][i]
            for k in range(i, n):
                if k == i:
                    a[j][k] = 0
                else:
                    a[j][k] += c * a[i][k]
            b[j] += c * b[i]
    x = [0] * n
    for i in range(n - 1, -1, -1):
        x[i] = b[i]
        for j in range(n - 1, i, -1):
```

```
23              x[i] -= x[j] * a[i][j]
24          x[i] /= a[i][i]
25      return x
26
27
28  def gauss_method_permutation_strings(a, b):
29      n = len(a)
30      for i in range(n):
31          j_max = i
32          for j in range(i + 1, n):
33              if abs(a[j][i]) > abs(a[j_max][i]):
34                  j_max = j
35          if j_max != i:
36              for k in range(n):
37                  a[i][k], a[j_max][k] = a[j_max][k], a[i][k]
38              b[i], b[j_max] = b[j_max], b[i]
39          for j in range(i + 1, n):
40              c = - a[j][i] / a[i][i]
41              for k in range(i, n):
42                  if k == i:
43                      a[j][k] = 0
44                  else:
45                      a[j][k] += c * a[i][k]
46              b[j] += c * b[i]
47      x = [0] * n
48      for i in range(n - 1, -1, -1):
49          x[i] = b[i]
50          for j in range(n - 1, i, -1):
51              x[i] -= x[j] * a[i][j]
52          x[i] /= a[i][i]
53      return x
54
55
56  def gauss_method_permutation_columns(a, b):
57      n = len(a)
58      perm = [i for i in range(n)]
59      for i in range(n):
60          j_max = i
61          for j in range(i + 1, n):
62              if abs(a[i][j]) > abs(a[i][j_max]):
63                  j_max = j
64          if j_max != i:
65              for k in range(n):
66                  a[k][i], a[k][j_max] = a[k][j_max], a[k][i]
67              perm[i], perm[j_max] = perm[j_max], perm[i]
68          for j in range(i + 1, n):
```

```
69              c = - a[j][i] / a[i][i]
70              for k in range(i, n):
71                  if k == i:
72                      a[j][k] = 0
73                  else:
74                      a[j][k] += c * a[i][k]
75              b[j] += c * b[i]
76      x1 = [0] * n
77      for i in range(n - 1, -1, -1):
78          x1[i] = b[i]
79          for j in range(n - 1, i, -1):
80              x1[i] -= x1[j] * a[i][j]
81          x1[i] /= a[i][i]
82      x = [0] * n
83      for i in range(n):
84          x[perm[i]] = x1[i]
85      return x
86
87
88  def gauss_method_permutation_strings_and_columns(a, b):
89      n = len(a)
90      perm = [i for i in range(n)]
91      for i in range(n):
92          j_max_str = i
93          for j in range(i + 1, n):
94              if abs(a[j][i]) > abs(a[j_max_str][i]):
95                  j_max_str = j
96          j_max_col = i
97          for j in range(i + 1, n):
98              if abs(a[i][j]) > abs(a[i][j_max_col]):
99                  j_max_col = j
100         if j_max_str != i or j_max_col != i:
101             if j_max_str > j_max_col:
102                 for k in range(n):
103                     a[i][k], a[j_max_str][k] = a[j_max_str][k], a[i][k]
104                 b[i], b[j_max_str] = b[j_max_str], b[i]
105             else:
106                 for k in range(n):
107                     a[k][i], a[k][j_max_col] = a[k][j_max_col], a[k][i]
108                 perm[i], perm[j_max_col] = perm[j_max_col], perm[i]
109         for j in range(i + 1, n):
110             c = - a[j][i] / a[i][i]
111             for k in range(i, n):
112                 if k == i:
113                     a[j][k] = 0
114                 else:
```

```
                    a[j][k] += c * a[i][k]
            b[j] += c * b[i]
    x1 = [0] * n
    for i in range(n - 1, -1, -1):
        x1[i] = b[i]
        for j in range(n - 1, i, -1):
            x1[i] -= x1[j] * a[i][j]
        x1[i] /= a[i][i]
    x = [0] * n
    for i in range(n):
        x[perm[i]] = x1[i]
    return x


def generate_matrix(n, v1, v2):
    return [[random.uniform(v1, v2) for i in range(n)] for j in range(n)
    ]


def calculate_diagonal_predominance_degree(a):
    n = len(a)
    d = 0
    for i in range(n):
        s = 0
        for j in range(n):
            if i != j:
                s += abs(a[i][j])
        r = abs(a[i][i]) - s
        if i == 0:
            d = r
        elif r > d:
            d = r
    return d


def increase_diagonal_predominance_degree(a, k):
    n = len(a)
    a_new = copy.deepcopy(a)
    for i in range(n):
        s = 0
        for j in range(n):
            if i != j:
                s += abs(a[i][j])
        a_new[i][i] += s * k
    return a_new

```

```
160
161 def input_matrix():
162     n = int(input())
163     a = []
164     b = []
165     for i in range(n):
166         l = list(map(int, input().split()))
167         a.append(l)
168     for i in range(n):
169         b.append(int(input()))
170     return a, b
171
172
173 plt.xlabel('Diagonal predominance degree')
174 plt.ylabel('Errors')
175
176
177 for n in [10, 50, 100]:
178
179     a1 = generate_matrix(n, -10, 10)
180     x = [i for i in range(1, n + 1)]
181
182     k = [0, 0.5, 1, 1.5, 2]
183
184     l = len(k)
185
186     x_degrees = [0] * l
187     y_gauss = [0] * l
188     y_gauss_perm_str = [0] * l
189     y_gauss_perm_col = [0] * l
190     y_gauss_perm_str_and_col = [0] * l
191     y_gauss_lib = [0] * l
192
193     for i in range(l):
194
195         a = increase_diagonal_predominance_degree(a1, k[i])
196         d = calculate_diagonal_predominance_degree(a)
197
198         b = mult_matr_vec(a, x)
199
200         x_num = gauss_method(copy.deepcopy(a), copy.copy(b))
201         x_num_perm_str = gauss_method_permutation_strings(copy.deepcopy(
    a), copy.copy(b))
202         x_num_perm_col = gauss_method_permutation_columns(copy.deepcopy(
    a), copy.copy(b))
```

```
203         x_num_perm_str_and_col =
    gauss_method_permutation_strings_and_columns(copy.deepcopy(a), copy.
    copy(b))
204         x_lib = np.linalg.solve(a, b)
205
206         norm_x_num = norm_vec(sub_vec(x, x_num))
207         norm_x_num_perm_str = norm_vec(sub_vec(x, x_num_perm_str))
208         norm_x_num_perm_col = norm_vec(sub_vec(x, x_num_perm_col))
209         norm_x_num_perm_str_and_col = norm_vec(sub_vec(x,
    x_num_perm_str_and_col))
210         norm_x_lib = norm_vec(sub_vec(x, x_lib))
211
212         x_degrees[i] = d
213
214         y_gauss[i] = norm_x_num
215         y_gauss_perm_str[i] = norm_x_num_perm_str
216         y_gauss_perm_col[i] = norm_x_num_perm_col
217         y_gauss_perm_str_and_col[i] = norm_x_num_perm_str_and_col
218         y_gauss_lib[i] = norm_x_lib
219
220     plt.plot(x_degrees, y_gauss, color = "blue", label = "Gauss method
    classic")
221     plt.plot(x_degrees, y_gauss_perm_str, color = "green", label = "
    Gauss method permutation strings")
222     plt.plot(x_degrees, y_gauss_perm_col, color = "yellow", label = "
    Gauss method permutation columns")
223     plt.plot(x_degrees, y_gauss_perm_str_and_col, color = "red", label =
     "Gauss method permutation strings and columns")
224     plt.plot(x_degrees, y_gauss_lib, color = "black", label = "Gauss
    method library")
225
226     plt.legend()
227
228     plt.show()
```

4 Результаты

Результаты работы программы представлены на рисунках 1 – 3.

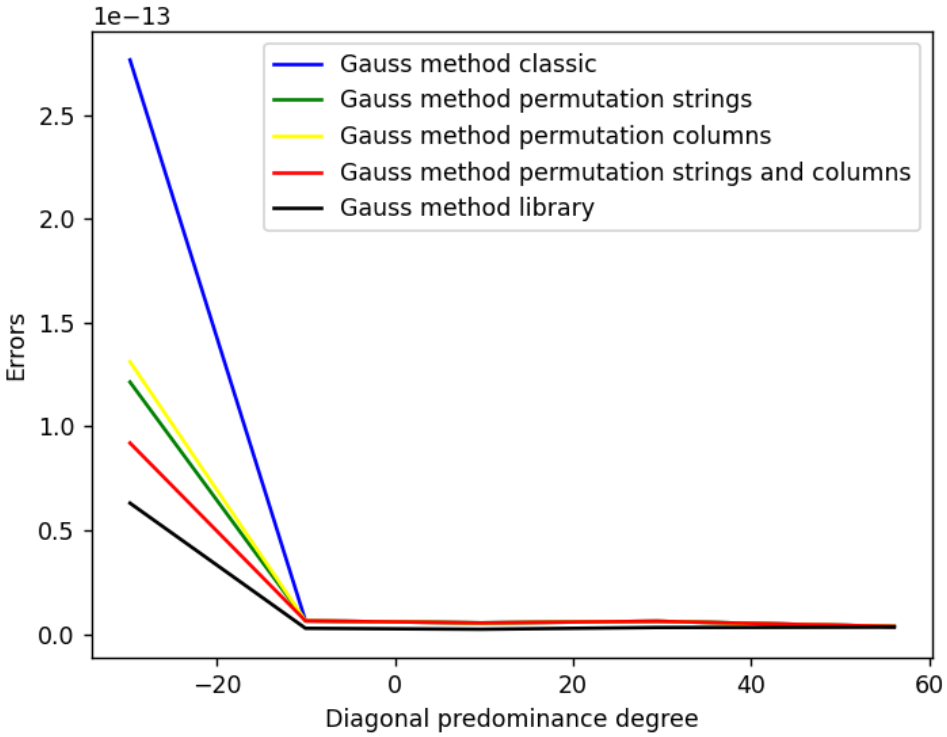

Рис. 1 — Результат работы программы при n = 10

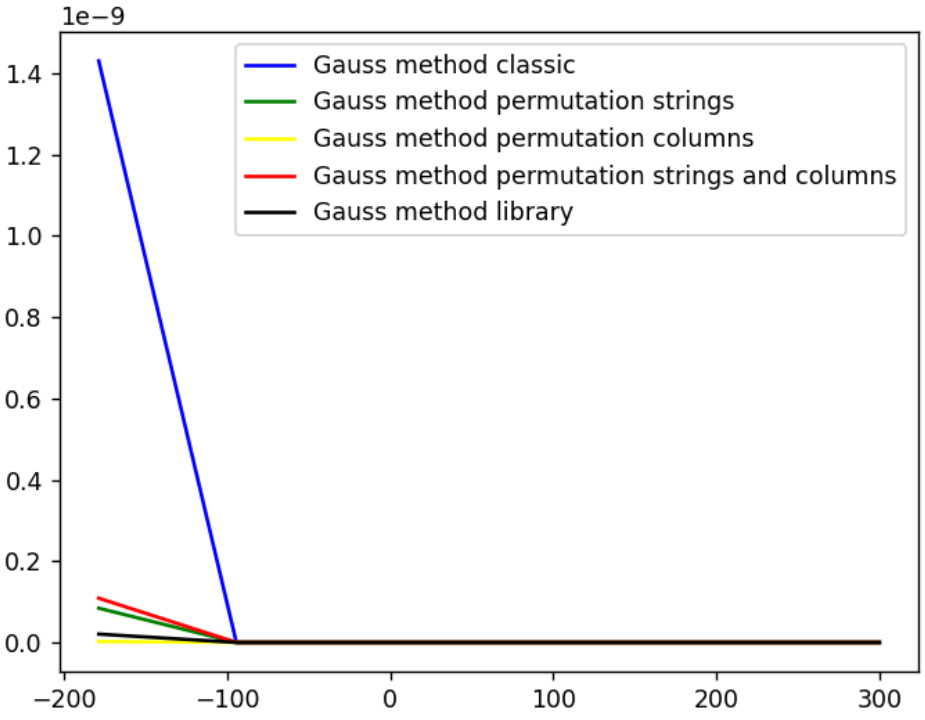

Рис. 2 — Результат работы программы при n = 50

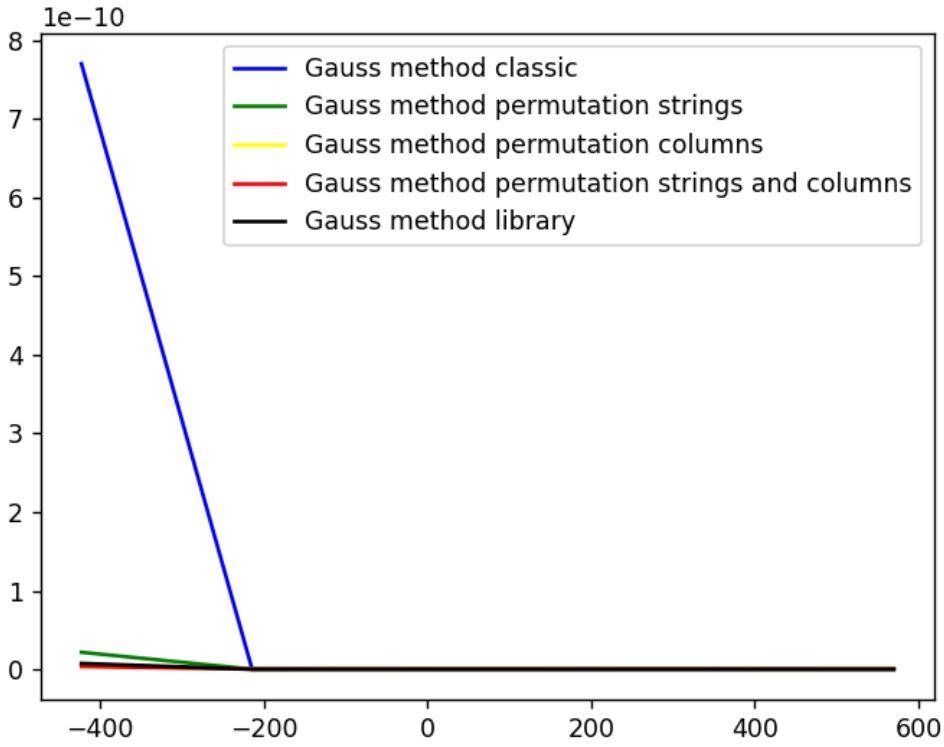

Рис. 3 — Результат работы программы при n = 100

5 Выводы

В результате выполнения лабораторной работы были реализованы три варианта метода Гаусса с перестановками, была оценена погрешность решения системы линейных уравнений для матриц произвольной размерности.