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 № 5

по курсу «Численные методы линейной алгебры»

«Изучение сходимости метода Якоби»

Студент группы ИУ9-71Б Яровикова А. С.

Преподаватель Посевин Д. П.

Москва 2023

1 Цель работы

Реализовать метод Якоби.

2 Задание

1. Реализовать метод Якоби.

2. Ввести критерий остановки итерационного процесс используя равномерную норму.

3. Проверить решение путем сравнения с решением любым методом Гаусса.

4. Проверить выполнение условия диагонального преобладания.

5. Используя согласованную векторную и матричную нормы проверить выполнение условия:

$$||P|| \leq q < 1$$

3 Реализация

Исходный код программы представлен в листингах 1– 4.

Листинг 1 — Вспомогательные функции

```
import numpy as np
from copy import deepcopy
import matplotlib.pyplot as plt
import sys

# matrix norm
def matrix_norm(matrix):
  sum = 0
  for i in range(len(matrix)):
    sum += abs(matrix[i])
  return max(sum)

# vector norm
def vec_norm(v):
  return max(map(abs, v))

# generate vector
def generate_vec(l, r, n):
    while True:
      vec = np.random.uniform(l, r, n)
      if vec_norm(vec) < 1:
        break
    return vec

# generate matrix
def generate_matrix(l, r, n):
    a = np.random.uniform(l, r, (n, n))
    return a

def increase_diag_elems(a, diag):
    n = len(a)
    for i in range(0, len(a)):
        a[i][i] = diag * sum(abs(a[i][j]) if j != i else 0 for j in
   range(n))
    return a

# check diagonal dominance
def calc_diagonal_dominance(a):
  degree = max(abs(a[i][i]) - sum(abs(a[i][j]) if j != i else 0 for j in
    range(len(a))) for i in range(len(a)))
  return degree > 0
```

Листинг 2 — Метод Гаусса

```
# Gauss
def gauss(matrix, vec):
    n = len(matrix)
    A = deepcopy(matrix)
    b = deepcopy(vec)
    x = np.zeros(shape=(n, ))
    # towards
    for i in range(n - 1):
        if A[i][i] == 0:
            for j in range(i + 1, n):
                if A[j][i] != 0:
                    A[i], A[j] = A[j], A[i]
                    break

        for j in range(i + 1, n):
            c = - A[j][i] / A[i][i]
            A[j] += c * A[i]
            b[j] += c * b[i]

    # backwards
    for i in range(n - 1, -1, -1):
        x[i] = b[i] / A[i][i]
        for j in range(i - 1, -1, -1):
            b[j] -= A[j][i] * x[i]

    return np.array(x)
```

Листинг 3 — Метод Якоби

```
def jacobi(A, f):
    eps = eps = 1e-6
    n = len(A)
    D = np.array([[0 if i != j else A[i][j] for j in range(n)] for i in
        range(n)])
    D_inv = np.linalg.inv(D)
    P = np.dot(-D_inv, A - D)

    norm_P = matrix_norm(deepcopy(P))
    q = np.max(np.abs(np.linalg.eigvals(P)))
    if not (norm_P <= q < 1):
        print('||P|| <= q < 1 not working')
        print(f'\nnorma_P: {norm_P}, q: {q}')
    else:
        print('||P|| <= q < 1 correct')
        print(f'\nnorma_P: {norm_P}')

    g = np.dot(D_inv, f)
    x_k = g
    iters = 1
    while True:
        x_ki = np.dot(P, x_k) + g
        # iterative process stop criterion using uniform norm
        if vec_norm(x_ki - x_k) < eps:
            break
        iters += 1
        x_k = x_ki
    return x_ki, iters
```

Листинг 4 — Запус программы

```python
n = 10
a = generate_matrix(0, 10, n)
a = increase_diag_elems(a, 3)
print('matrix a:')
print(a)
f = generate_vec(0, 0.5, n)
print(f'\nvector f: {f}')

diag_cond = calc_diagonal_dominance(a)
print(f'\ncheck for diagonal dominance condition: {diag_cond}')

# correct res
x = np.dot(np.linalg.inv(a), f)

# jacobi res
x_j, iters = jacobi(a, f)

print(f'jacobi iterations: {iters}')
# gauss res
x_g = gauss(a, f)

print(f'\ncorrect res: {x}')
print(f'\njacobi res: {x_j}')
print(f'\ngauss res: {x_g}')

print('\n\nrelative errors:')
from tabulate import tabulate
g = vec_norm(x_g - x) / vec_norm(x_g)
j = vec_norm(x_j - x) / vec_norm(x_j)
mydata = [[g, j]]

# create header
head = ["Jacobi", "Gauss"]

# display table
print(tabulate(mydata, headers=head, tablefmt="grid"))
```

4 Результаты

Результат запуска методов представлены на рисунках 1 - 4.

```
matrix a:
[[144.97561466   9.16381466   9.25421316   6.21727221   5.57038817
    8.07527481   2.59785917   1.88257564   3.75902006   1.80478701]
 [  5.39570649 115.22388431   2.39397764   9.14577052   2.71531561
    0.86663494   2.73075289   6.05541836   7.8783051    1.22607989]
 [  6.43069531   3.10528649 101.49360379   0.47120823   2.86130315
    1.95455403   7.91862718   2.69908653   6.20846721   2.18197314]
 [  7.56262108   6.44263741   3.61166505 171.34205967   2.34143891
    8.61527615   7.2185058    7.86560749   6.0716131    7.38465489]
 [  5.20924534   5.56477755   3.03002561   1.78521576 135.7496333
    5.34928475   1.16241279   8.76045733   5.64235047   8.74610816]
 [  3.83321992   3.25366125   9.10035653   1.27739288   8.55390764
  149.88976614   4.18063237   4.67936269   9.19982741   5.88489468]
 [  6.22738329   5.71663493   1.49596839   1.44603041   9.67829332
    2.06802323 131.42636921   6.29471654   2.05879028   8.82294933]
 [  8.34591933   3.19210406   3.74815506   4.43715844   1.36456584
    9.91554293   0.98672866 125.67730453   7.60428288   2.29797766]
 [  5.28848423   8.5658438    7.72679639   6.13306822   9.59081818
    0.77930074   2.96092994   2.86916051 147.37453118   5.21044172]
 [  8.16093946   9.0596119    7.47871716   8.41469976   3.62226561
    0.23017551   1.29233291   1.36376113   6.38705916 138.02868782]]

vector f: [0.2454429  0.23154752 0.19474097 0.24836864 0.17592054 0.03501035
 0.30801967 0.04694681 0.35541206 0.20858502]
```

Рис. 1 — Тестовые данные СЛАУ: матрица A размером 10x10, вектор f

```
check for diagonal dominance condition: True
||P|| ≤ q < 1 correct
norma_P: 0.42707388767437165
jacobi iterations: 7
```

Рис. 2 — Проверка условия диагонального преобладания и выполрнения условия $||P|| \leq q < 1$

```
correct res: [ 1.32428559e-03  1.61225084e-03  1.45444821e-03  1.09555069e-03
  9.71587346e-04 -2.09168980e-04  2.01176301e-03  1.21156906e-05
  2.00884407e-03  1.04439811e-03]

jacobi res: [ 1.32410641e-03  1.61207166e-03  1.45426903e-03  1.09537151e-03
  9.71408164e-04 -2.09348162e-04  2.01158383e-03  1.19365083e-05
  2.00866489e-03  1.04421893e-03]

gauss res: [ 1.32428559e-03  1.61225084e-03  1.45444821e-03  1.09555069e-03
  9.71587346e-04 -2.09168980e-04  2.01176301e-03  1.21156906e-05
  2.00884407e-03  1.04439811e-03]
```

Рис. 3 — Решения СЛАУ: точное, поулченное методом Гаусса, полученное методом Якоби

```
relative errors:
+-------------+-------------+
|      Jacobi |       Gauss |
+=============+=============+
| 3.23359e-16 | 8.90755e-05 |
+-------------+-------------+
```

Рис. 4 — Относительные погрешности методов

5 Выводы

В результте выполнения данной лабораторной работы был реализован реализован метод Якоби для решения СЛАУ. Реализация была выполнена на языке программирования Python. Также выполнено сравнение относительной погрешности методов Якоби и стандартного метода Гаусса на матрице с диагональным преобладанием.