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 № 5.2

по курсу «Численные методы линейной алгебры»

«Изучение сходимости метода Зейделя»

Студент группы ИУ9-71Б Яровикова А. С.

Преподаватель Посевин Д. П.

Москва 2023

1 Цель работы

Реализовать метод Зейделя.

2 Задание

1. Реализовать метод Зейделя.

2. Сравнить число итераций, необходимых для сходимости метода Якоби и метода Зейделя.

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

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

def vec_norm(v):
  return max(map(abs, v))

def generate_vec(l, r, n):
  vec = np.random.uniform(l, r, n)
  return vec

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

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 Якоби

```
def jacobi(A, f):
  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 < 1):
    print('||P||    q < 1 not working')
    print(f'norma_P: {norm_P}')
  else:
    print('||P||    q < 1 correct')
    print(f'norma_P: {norm_P}')

  g = np.dot(D_inv, f)
  x_k = g
  iters = 1
  while True:
    x_ki = np.dot(P, x_k) + g
    if vec_norm(x_ki - x_k) < eps:
      break
    iters += 1
    x_k = x_ki
  return x_ki, iters
```

Листинг 3 — Метод Зейделя

```
def zeidel(A, f):
  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)])
  A_n = np.array([[0 if i <= j else - A[i][j] / A[i][i] for j in range(n)] for i in range(n)])
  A_v = np.array([[0 if i >= j else - A[i][j] / A[i][i] for j in range(n)] for i in range(n)])
  D_inv = np.linalg.inv(D)

  x_k = np.dot(D_inv, f)
  iters = 1
  while True:
    x_ki = np.zeros(len(x_k))
    for i in range(n):
      for j in range(n):
        x_ki += A_n[i][j] * x_ki[j] + A_v[i][j] * x_k[j]
      x_ki[i] += f[i] / A[i][i]
    iters += 1
    if vec_norm(x_ki - x_k) < eps:
      break
    x_k = x_ki
  return x_ki, iters
```

Листинг 4 — Запуск программы

```
n = 100
a = generate_matrix(1, 10, n)
a = increase_diag_elems(a, 3)
print('matrix a:')
print(a)
f = generate_vec(1, 10, 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)

# zeidel res
x_z, itrs = zeidel(a, f)

print(f'\ncorrect res: {x}')
print(f'\njacobi res: {x_j}')
print(f'\nzeidel res: {x_z}')

print('\n\nmethods iterations:')
from tabulate import tabulate
mydata = [[iters, itrs]]

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

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

print(f'Jacobi iters / Zeidels iters: {iters/itrs}')
```

4 Результаты

Результат запуска методов представлены на рисунках 1 - **??**.

```
matrix a:
[[1673.18664163    3.07723018    6.03498596 ...    2.57782977
     6.46011845    9.68410145]
 [   5.1816453  1480.56248826    6.58798962 ...    9.89730148
     2.49833286    7.97657581]
 [   9.34244779    3.28564968 1699.18516292 ...    4.44822118
     5.67437181    7.40581673]
 ...
 [   8.00333499    4.22485228    6.14148132 ... 1661.03524786
     7.27109037    2.5497818 ]
 [   5.81438184    2.54261082    7.77094685 ...    2.33122965
  1555.34222955    5.48337809]
 [   4.93183909    8.17853633    7.3228769  ...    9.86657969
     8.47476648 1642.47757091]]

vector f: [2.47373341 8.91390561 2.83005682 7.7974442  7.50794229 6.79914262
 6.4019982  3.97645116 9.38201179 8.45999941 7.0516114  6.63535218
 1.72923426 1.79583406 3.33875492 6.57751383 8.98793205 1.34318547
 4.53886833 5.34536071 7.28619954 5.94489864 5.38877705 1.93635144
 4.28145559 6.00721705 7.43897371 8.7611063  4.7559474  7.86204664
 5.7123703  2.4554749  9.52722514 4.39443217 3.92712822 9.24566901
 2.82605378 7.12581855 7.02475578 9.26088559 4.18636999 6.70361144
 3.43625556 4.37243639 9.66204365 1.45148732 1.14988687 5.7547513
 6.4012498  8.08955787 8.89568162 5.99157312 5.44904508 6.45003672
 1.81455185 6.36580773 1.72968596 6.97665781 3.82549129 4.17938118
 5.20758391 9.30428321 7.1283251  2.3589263  1.23346145 4.90647736
 1.87492256 9.69468417 6.52794485 3.53007335 9.36148713 9.67490485
 4.78094606 6.35671813 9.14512596 2.18943377 2.3160168  5.86967873
 5.88128623 2.58652636 6.58046891 4.70905479 4.97828147 5.96728306
 8.1043555  1.42082247 3.51802547 8.46853177 3.52572105 8.43473159
 3.12457293 3.57381525 7.362514   9.19499789 3.44823213 7.75950239
 8.0033232  4.25900972 6.77756027 4.52707562]
```

Рис. 1 — Тестовые данные СЛАУ: матрица А размером 100x100, вектор f

```
methods iterations:
+----------+----------+
|   Jacobi |   Zeidel |
+==========+==========+
|        8 |        5 |
+----------+----------+
Jacobi iters / Zeidels iters: 1.6
```

Рис. 2 — Сравнение методов Якоби и Зейделя

5 Выводы

В результте выполнения данной лабораторной работы был реализован реализован метод Зейделя для решения СЛАУ. Реализация была выполнена на языке программирования Python. Также выполнено сравнение количества операций методов Якоби и Зейделя на матрице с диагональным преобладанием.