**Министерство образования и науки Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования «КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ**

Специальность: «01.03.02 Прикладная математика и информатика»

**Семестровая работа по курсу «Численные методы»**

**на тему:**

**«Итерационные методы решения систем линейных уравнений. Метод Якоби, Зейделя»**

Выполнил:

Студент группы 09-812

Садыков И. Н.

Проверил:

Макаров М.В.

КАЗАНЬ – 2021 г.

Оглавление

[Постановка задачи 3](#_Toc90817793)

[Описание методов решения 4](#_Toc90817794)

[Условия сходимости итерационных методов 6](#_Toc90817795)

[Таблицы и графики значений и с указанием числа итераций, потребовавшихся для достижения заданной точности 7](#_Toc90817796)

[Вывод 13](#_Toc90817797)

[Листинг программы 14](#_Toc90817798)

# Постановка задачи

Решить систему линейных алгебраических уравнений

Здесь

*–* заданные функции.

Данную систему решить методом прогонки и итерационным методами:

1. Якоби
2. Зейделя

Во всех итерационных методах продолжать вычисления до выполнения условия:

*r* – вектор невязки, – заданное число.

**Исходные данные**: Расчеты провести при n = 40, ,

*,* ,

# Описание методов решения

Система линейных алгебраических уравнений в матричном виде может быть представлена как:

или

Здесь A — это матрица системы, x — столбец неизвестных, а b — столбец свободных членов.

Для метода Якоби и Зейделя будем считать, что все диагональные элементы матрицы A отличны от нуля. Выберем некоторое начальное приближение .

**Метод Якоби**:

Перепишем (1), разрешая каждое уравнение относительно переменной, стоящей на диагонали:

Выберем некоторое начальное приближение и построим последовательность векторов определяя вектор по уже найденному вектору при помощи соотношений:

Формула определяет итерационный **метод Якоби** для решения системы (1).

**Метод Зейделя:**

Добавим далее естественную модификацию в (2) и получим другой метод, называемый методом **Зейделя.**

При вычислении будем использовать уже найденные компоненты вектора , то есть . В результате:

(3) определяет итерационный метод Зейделя для решения (1).

**Метод прогонки**:

Метод прогонки состоит из двух этапов: прямой прогонки и обратной прогонки. На первом этапе определяются прогоночные коэффициенты, а на втором – находят неизвестные .

Формула для определения прогоноччных коэффициентов в прямом ходе:

с условиями:

Обратный ход:

# Условия сходимости итерационных методов

Методы Якоби и Зейделя сходятся от любого начального приближения, если матрица системы уравнений обладает свойством диагонального преобладания.

# Таблицы и графики значений и с указанием числа итераций, потребовавшихся для достижения заданной точности

Верное решение системы:

u(ih) = [0.00059414 0.00225625 0.00481289 0.0081 0.01196289 0.01625625

0.02084414 0.0256 0.03040664 0.03515625 0.03975039 0.0441

0.04812539 0.05175625 0.05493164 0.0576 0.05971914 0.06125625

0.06218789 0.0625 0.06218789 0.06125625 0.05971914 0.0576

0.05493164 0.05175625 0.04812539 0.0441 0.03975039 0.03515625

0.03040664 0.0256 0.02084414 0.01625625 0.01196289 0.0081

0.00481289 0.00225625 0.00059414]

Решение методом Якоби:

jacobi y = [-1.34479730e-04 -1.78772155e-04 -5.18817496e-05 3.17996049e-04

9.93230805e-04 2.02700242e-03 3.46290075e-03 5.33532546e-03

7.66894287e-03 1.04792294e-02 1.37718031e-02 1.75430948e-02

2.17795748e-02 2.64585322e-02 3.15472190e-02 3.70037149e-02

4.27760126e-02 4.88029424e-02 5.50132244e-02 6.13264256e-02

6.76520032e-02 7.38902691e-02 7.99314468e-02 8.56566176e-02

9.09368151e-02 9.56339274e-02 9.95998506e-02 1.02677325e-01

1.04699168e-01 1.05489023e-01 1.04860684e-01 1.02618751e-01

9.85580569e-02 9.24642059e-02 8.41131220e-02 7.32714660e-02

5.96963029e-02 4.31353948e-02 2.33269922e-02]

Погрешности методом Якоби:

error jacobi = [0.00072862 0.00243502 0.00486477 0.007782 0.01096966 0.01422925

0.01738124 0.02026467 0.0227377 0.02467702 0.02597859 0.02655691

0.02634582 0.02529772 0.02338442 0.02059629 0.01694313 0.01245331

0.00717467 0.00117357 0.00546411 0.01263402 0.02021231 0.02805662

0.03600517 0.04387768 0.05147446 0.05857733 0.06494878 0.07033277

0.07445404 0.07701875 0.07771392 0.07620796 0.07215023 0.06517147

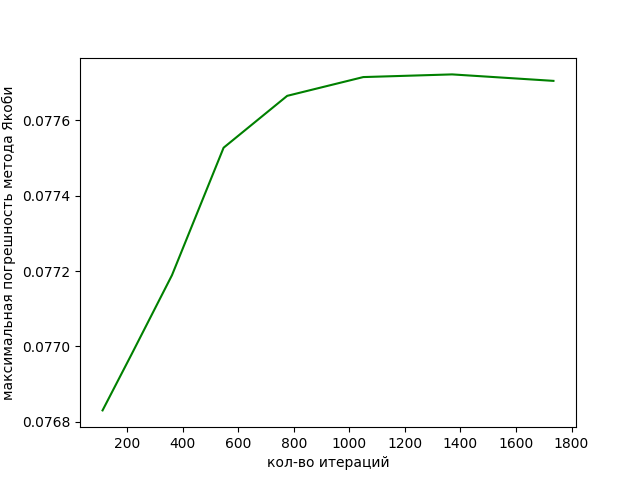
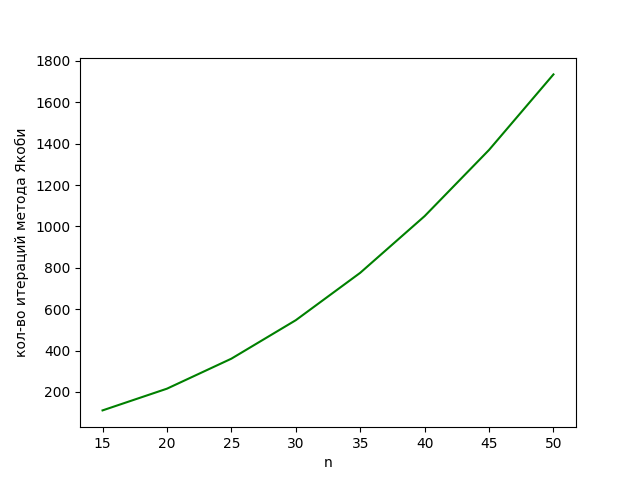
0.05488341 0.04087914 0.02273285]

Максимальная погрешность методом Якоби:

max error jacobi = 0.07771391631677127

Максимальное количество итераций методом Якоби:

max iterations jacobi = 1050



Верное решение системы:

u(ih) = [0.00059414 0.00225625 0.00481289 0.0081 0.01196289 0.01625625

0.02084414 0.0256 0.03040664 0.03515625 0.03975039 0.0441

0.04812539 0.05175625 0.05493164 0.0576 0.05971914 0.06125625

0.06218789 0.0625 0.06218789 0.06125625 0.05971914 0.0576

0.05493164 0.05175625 0.04812539 0.0441 0.03975039 0.03515625

0.03040664 0.0256 0.02084414 0.01625625 0.01196289 0.0081

0.00481289 0.00225625 0.00059414]

Решение методом Зейделя:

seidel y = [-1.46980901e-04 -2.02539033e-04 -8.55782228e-05 2.75704666e-04

9.43780355e-04 1.97174820e-03 3.40329832e-03 5.27267544e-03

7.60464504e-03 1.04144621e-02 1.37078434e-02 1.74809424e-02

2.17203287e-02 2.64029707e-02 3.14962221e-02 3.69578116e-02

4.27358372e-02 4.87687626e-02 5.49854171e-02 6.13049990e-02

6.76370800e-02 7.38816131e-02 7.99289413e-02 8.56598090e-02

9.09453731e-02 9.56472164e-02 9.96173607e-02 1.02698281e-01

1.04722921e-01 1.05514703e-01 1.04887547e-01 1.02645883e-01

9.85846619e-02 9.24893720e-02 8.41360490e-02 7.32912890e-02

5.97122589e-02 4.31467074e-02 2.33329744e-02]

Погрешности методом Зейделя:

error seidel = [0.00074112 0.00245879 0.00489847 0.0078243 0.01101911 0.0142845

0.01744084 0.02032732 0.022802 0.02474179 0.02604255 0.02661906

0.02640506 0.02535328 0.02343542 0.02064219 0.0169833 0.01248749

0.00720247 0.001195 0.00544919 0.01262536 0.0202098 0.02805981

0.03601373 0.04389097 0.05149197 0.05859828 0.06497253 0.07035845

0.07448091 0.07704588 0.07774052 0.07623312 0.07217316 0.06519129

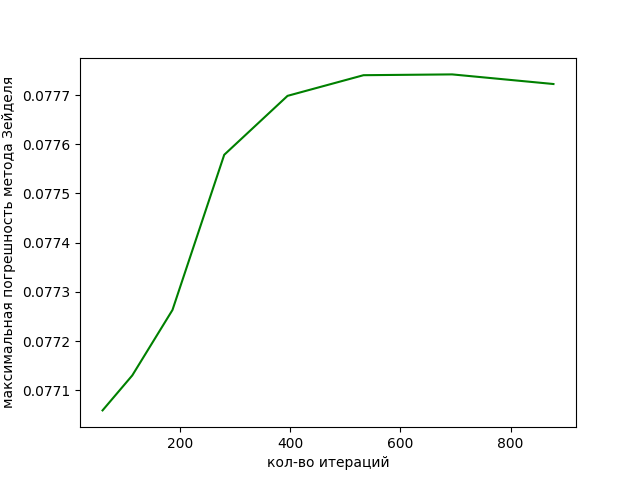
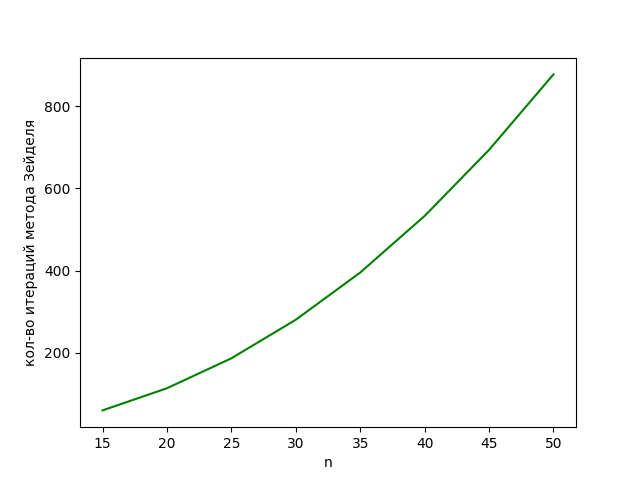
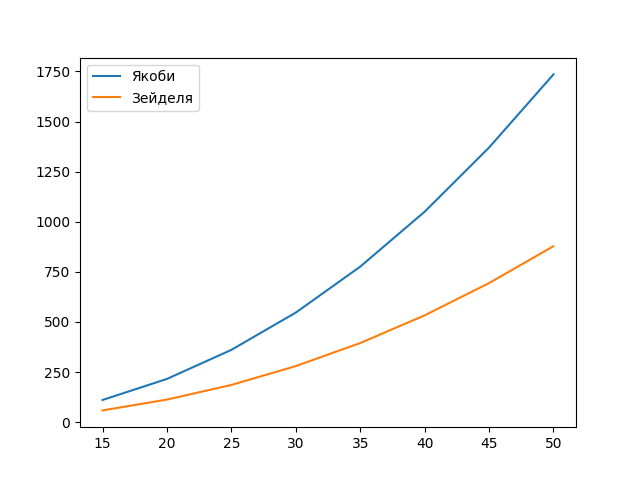
0.05489937 0.04089046 0.02273883]

Максимальная погрешность методом Зейделя:

max error seidel = 0.07774052126541595

Максимальное количество итераций методом Зейделя:

max iterations seidel = 533

Количество итераций в методах в зависимости от размера системы

# Вывод

Рассмотренные методы позволяют эффективно находить решения СЛАУ.

Метод Зейделя превосходит по скорости нахождения решения метод Якоби. Так же исходя из построения метода Зейделя он эффективнее тем, что требует меньше памяти.

Все методы требуют большое количество итераций для нахождения решения. Количество итераций увеличивается с уменьшением заданной ошибки вычисления и увеличением размера матрицы. Метод прогонки позволяет находить решение систем с матрицами с n=40 с точностью до 12-го знака после запятой и требует лишь одну итерацию, что существенно выгоднее итерационных методов. То есть при решении трёхдиагональных предпочтительнее использовать метод прогонки, однако он не применим к системам с матрицами, не являющимися трёхдиагональными.

# Листинг программы

import numpy as np

from matplotlib import pyplot as plt

alpha = 2

beta = 2

gamma = 2

n = 40

h = 1 / n

eps = h \*\* 3

def u(x):

return (x \*\* alpha) \* ((1 - x) \*\* beta)

def p(x):

return 1 + x \*\* gamma

def g(x):

return x + 1

def f(x):

return 20 \* x \*\* 4 - x \*\* 5 - 11 \* x \*\* 3 + 12 \* x \*\* 2 - 6 \* x

def generate\_A():

A = np.zeros((n - 1, n - 1))

for i in range(n - 1):

if i > 0:

A[i, i - 1] = -p(i \* h)

A[i, i] = p(i \* h) + p((i + 1) \* h) + h \* h \* g(i \* h)

if i < n - 2:

A[i, i + 1] = -p((i + 1) \* h)

return A

def generate\_b():

b = np.zeros(n - 1)

for i in range(n - 1):

b[i] = f((i + 1) \* h) \* h \* h

return b

def compute\_abs\_error\_vector(sol):

exact\_sol = np.array([u((i + 1) \* h) for i in range(n - 1)])

return np.absolute(sol - exact\_sol)

def get\_u():

exact\_sol = np.array([u((i + 1) \* h) for i in range(n - 1)])

return exact\_sol

def max\_abs\_error(sol):

return np.max(compute\_abs\_error\_vector(sol))

def method\_jacobi(A, b, x=None):

if x is None:

x = np.zeros(len(A[0]))

D = np.diag(A)

R = A - np.diagflat(D)

k = 0

while not np.max(np.absolute(np.dot(A, x) - b)) <= eps:

x = (b - np.dot(R, x)) / D

k += 1

return x, k

def method\_seidel(A, b, x=None):

if x is None:

x = np.zeros(len(A[0]))

converge = False

k = 0

while not converge:

x\_new = np.copy(x)

for i in range(n - 1):

s1 = np.dot(A[i, :i], x\_new[:i])

s2 = np.dot(A[i, i + 1:], x[i + 1:])

x\_new[i] = (b[i] - s1 - s2) / A[i][i]

converge = np.max(np.absolute(np.dot(A, x) - b)) <= eps

x = x\_new

k += 1

return x, k

def main():

global n, h, eps

A = generate\_A()

b = generate\_b()

y\_jacobi, k\_jacobi = method\_jacobi(A, b)

abs\_err\_jacobi = compute\_abs\_error\_vector(y\_jacobi)

exact\_solution = get\_u()

max\_abs\_err\_jacobi = np.max(abs\_err\_jacobi)

print('u(ih) = ', exact\_solution)

print('jacobi y = ', y\_jacobi)

print('error jacobi = ', abs\_err\_jacobi)

print('max error jacobi = ', max\_abs\_err\_jacobi)

print('max iterations jacobi = ', k\_jacobi)

y\_seidel, k\_seidel = method\_seidel(A, b)

abs\_err\_seidel = compute\_abs\_error\_vector(y\_seidel)

max\_abs\_err\_seidel = np.max(abs\_err\_seidel)

print('seidel y = ', y\_seidel)

print('error seidel = ', abs\_err\_seidel)

print('max error seidel = ', max\_abs\_err\_seidel)

print('max iterations seidel = ', k\_seidel)

n\_arr = [15, 20, 25, 30, 35, 40, 45, 50]

h\_arr = [1 / x for x in n\_arr]

eps\_arr = [x \*\* 3 for x in h\_arr]

iterations\_arr\_jacobi = []

iterations\_arr\_seidel = []

max\_abs\_err\_jacobi = []

max\_abs\_err\_seidel = []

for i in range(len(n\_arr)):

n, h, eps = n\_arr[i], h\_arr[i], eps\_arr[i]

A = generate\_A()

b = generate\_b()

sol, k = method\_jacobi(A, b)

iterations\_arr\_jacobi.append(k)

max\_abs\_err\_jacobi.append(max\_abs\_error(sol))

sol, k = method\_seidel(A, b)

iterations\_arr\_seidel.append(k)

max\_abs\_err\_seidel.append(max\_abs\_error(sol))

print('jacobi iterations array = ', iterations\_arr\_jacobi)

print('max abs err jacobi', max\_abs\_err\_jacobi)

print('seidel iterations array = ', iterations\_arr\_seidel)

print('max abs err seidel', max\_abs\_err\_seidel)

plt.subplots()

plt.plot(n\_arr, iterations\_arr\_jacobi, '-g')

plt.xlabel('n')

plt.ylabel('кол-во итераций метода Якоби')

plt.show()

plt.subplots()

plt.plot(iterations\_arr\_jacobi, max\_abs\_err\_jacobi, '-g')

plt.xlabel('кол-во итераций')

plt.ylabel('максимальная погрешность метода Якоби')

plt.show()

plt.subplots()

plt.plot(n\_arr, iterations\_arr\_seidel, '-g')

plt.xlabel('n')

plt.ylabel('кол-во итераций метода Зейделя')

plt.show()

plt.subplots()

plt.plot(iterations\_arr\_seidel, max\_abs\_err\_seidel, '-g')

plt.xlabel('кол-во итераций')

plt.ylabel('максимальная погрешность метода Зейделя')

plt.show()

plt.plot(n\_arr, iterations\_arr\_jacobi, label='Якоби')

plt.plot(n\_arr, iterations\_arr\_seidel, label='Зейделя')

plt.legend(loc='best')

plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

main()