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 Зейделя, релаксации»

Выполнил(а):
Студент(ка) группы 09-813
Махмутов Ринат -

Проверил:
Макаров М.В.

КАЗАНЬ – 2021 г.

Содержание

Постановка задачи

Решить систему линейных алгебраических уравнений

$$(a_1 + a_2 + h^2 g_1) y_1 - a_2 y_2 = f_1 h^2,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..,$$

$$-a_i y_{i-1} + (a_i + a_{i+1} + h^2 g_i) y_i - a_{i+1} y_{i+1} = f_i h^2,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..,$$

$$(a_{n-1} + a_n + h^2 g_{i-1}) y_{n-1} - a_{n-1} y_{n-2} = f_{n-1} h^2 .$$

Здесь: $a_i = p(i * h), g_i = q(i * h), f_i = f(i * h), h = \frac{1}{n},$

$f(x) = -(p(x) * u'(x))' + q(x) * u(x), p, q, u -$ заданные функции.

Цель задания: Данную систему решить методом прогонки и итерационными методами:

- Зейделя
- Релаксации

Во всех итерационных методах вычисления продолжать до выполнения условия:

$$\max_{1 \le i \le n-1} |r_i^k| \le \varepsilon, \text{ где } r - \text{вектор невязки, } \varepsilon - \text{заданное число.}$$

Исходные данные:

$n = 40, \varepsilon = h^3, u(x) = x^\alpha * (1 - x)^\beta, p(x) = 1 + x^\gamma, g(x) = x + 1,$

$\alpha = 3, \beta = 1, \gamma = 1$

Метод решения

Для решения задачи приведем нашу систему линейных алгебраических уравнений к удобному формату:

$$B_1 y_1 + C_1 y_2 = D_1,$$

……..,

$$A_i y_{i-1} + B_i y_i + C_i y_{i+1} = D_i,$$

……,

$$A_{n-1} y_{n-2} + B_{n-1} y_{n-1} = D_{n-1}.$$

Здесь: $A_i = -a_i, B_i = a_i + a_{i+1} + h^2 g_i, C_i = -a_{i+1}$

Метод прогонки

Метод прогонки - метод для решения систем линейных уравнений вида

$$Ay = b,$$

где A - трехдиагональная матрица. Представляет собой вариант метод последовтельного исключения неизвестных.

Выведем формулы для решения методом прогонки.

Система $Ay = b$ равносильна соотношению:

$$A_i y_{i-1} + B_i y_i + C_i y_{i+1} = D_i.$$

Метод прогонки основывается на предположения, что искомые неизвестные связаны рекуррентными соотношением:

$$y_i = \alpha_{i+1} y_{i+1} + \beta_{i+1}, \text{ где } i = n-1, n-2, \ldots, 1$$

Преобразуем уравнение к виду:

$$y_1 = \alpha_1 y_2 + \beta_1, \text{ где } \alpha_1 = -\frac{C_1}{B_1}, \beta_1 = \frac{D_1}{B_1}.$$

Подставим полученное во второе уравнение системы и получим:

$$\alpha_2(\alpha_1 y_2 + \beta_1) + B_2 y_2 + C_2 y_3 = D_2.$$

Вновь преобразуем уравнение к виду:

$$y_2 = \alpha_2 y_3 + \beta_2, \text{ где } \alpha_2 = -\frac{C_2}{A_2\alpha_1 + B_2}, \beta_2 = \frac{D_2 - A_2\beta_1}{B_2 + A_2\alpha_1}$$

Аналогично доходим i-ом шаге ($n < i < n - 1$) придём к преобразованию системы к виду:

$$y_i = \alpha_i y_{i+1} + \beta_i, \text{ где } \alpha_i = -\frac{C_i}{A_i\alpha_{i-1} + B_i}, \beta_i = \frac{D_i - A_i\beta_{i-1}}{B_i + A_i\alpha_{i-1}}$$

На $(n - 1)$ шаге подстановка в последнее уравнение дает:

$$y_{n-2} = \alpha_{n-2} y_{n-1} + \beta_{n-2}\,.$$

Так же мы имеем: $A_{n-1}(\alpha_{n-2} y_{n-1} + \beta_{n-2}) + B_{n-1} y_{n-1} = D_{n-1}\,.$

Отсюда находим значение y_{n-1}:

Значение остальных неизвестных, теперь легко вычисляется во формуле:

$$y_i = \alpha_i y_{i+1} + \beta_i$$

Процесс вычисления значений α_i и β_i является прямым ходом метода прогонки, а вычисление y_i – обратный ход. Для реализации метода прогонки необходимо, чтобы параметры α_i и β_i были отличны от нуля и выполнялось условие устойчивости:

$$|B_i| > |A_i| + |C_i|, i = 1,\ldots, n - 1$$

Из системы уравнений заметим, что элементы главной диагонали больше элементов побочной на $h^2 g_i, i = 2,\ldots, n - 2$. Для $i = 1$ разность между главной и побочной диагоналями равна $a_1 + h^2 g_i$ и $i = n - 1$ разность будет равна $a_n + h^2 g_i$. Следовательно, система выполняет условие устойчивости метода прогонки.

Решив систему линейных алгебраических уравнений методом прогонки, получим следующие результаты:

Итерация i	Метод Прогонка
1	8.679346268144755e-06
2	0.00010516887542011337
3	0.00036830152006807766
4	0.0008676990035768542
5	0.0016637568035098441
6	0.0028076308028910694
7	0.004341225415542548
8	0.006297183002785994
9	0.008698874424781278
10	0.01156039059161715

Итерация i	Метод Прогонка
11	0.014886534897701106
12	0.018672816438603853
13	0.022905443922777387
14	0.027561320201874004
15	0.03260803735306547
16	0.03800387225505925
17	0.04369778260664837
18	0.04962940334279201
19	0.05572904340855484
20	0.06191768285585683
21	0.06810697023200787
22	0.07419922023250809
23	0.08008741159365926
24	0.08565518520321797
25	0.09077684240967775
26	0.09531734351284128
27	0.09913230642017261
28	0.10206800545503424
29	0.10396137030434482
30	0.10463998509446101
31	0.10392208758521475
32	0.10161656847304071
33	0.09752297079502366
34	0.09143148942649554
35	0.08312297066552748
36	0.07236891189830398
37	0.05893146133994272
38	0.04256341784584168
39	0.023008230789102493

Метод Зейделя

Метод Зейделя можно рассматривать, как модификацию метода Якоби.

Основная идея модификации состоит в том, что при вычислении $y^{(k+1)}$ будем использовать уже найденные компоненты вектора y^{k+1}, то есть $y_1^{k+1}, y_2^{k+1}, \ldots, y_{i-1}^{k+1}$. В результате приходим в итерационному методу Зейделя.

Матричная запись для системы $A\,x = b$:

$$y_i^{(k+1)} = -\sum_{j=1}^{i=1} \frac{a_{ij}}{a_{ii}} y_j^{k+1} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^k + \frac{b_i}{a_{ii}} i = 1,2,\ldots,n, k = 0,1,\ldots,$$ где k – погрешность приближения.

Метод Зейделя позволяет более экономно расходовать память компьютера, поскольку в данном случае вновь получаемые компоненты вектора $y_i^{(n+1)}$ можно размещать на месте советующих компонент вектора y^k, в то время как при реализации метода Якоби все компоненты векторов y^k, y^{k+1} должны одновременно быть в памяти компьютера.

Достаточное условие сходимости методов Зейделя и Якоби аналогичные, и они сходятся для матрицы с диагональным преобладанием по строкам, то есть должно выполняться:

$$q = \max_{1 \le i \le n} \sum_{j=1, j \ne i}^{n} \frac{|a_{ij}|}{|a_{ii}|} < 1$$

Наша матрица является матрицей с диагональным преобладанием. Это условие мы выяснили при проверке устойчивости метода прогонки.

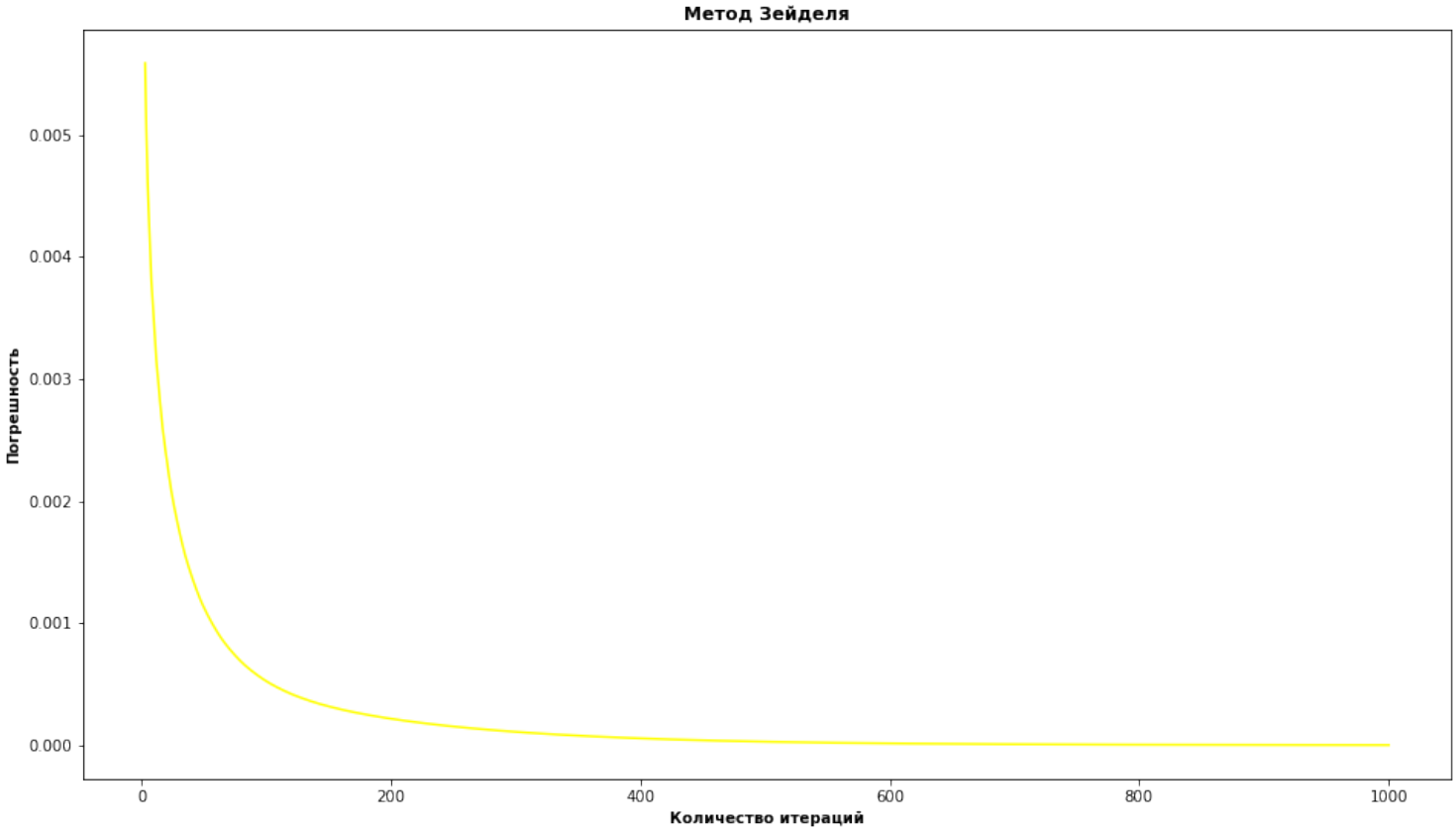
Решив систему линейных алгебраических уравнений, получим следующие результаты:

Метод Зейделя (y_i)	**Погрешность метода Зейделя**
-0.00022271734749306068	3.217440598559976e-06
-0.00034919201740167907	4.842294902149594e-06
-0.00029961126898383675	6.462625926190863e-06
-3.4450731353642037e-06	8.06689296889364e-06
0.0006005358956774652	9.643589561075783e-06
0.0015642443101457132	1.1181325900886763e-05
0.002930262088081704	1.2668910435618681e-05
0.004731827969867914	1.4095430039262114e-05
0.006992826630406782	1.5450328243584626e-05
0.009727779199336178	1.672348099341595e-05
0.012941835078722173	1.7905269413246295e-05
0.016630764960167306	1.8986649092252628e-05

Метод Зейделя	Погрешность метода Зейделя
0.025369401749858394	2.081526451696528e-05
0.030363708741984338	2.154784938488384e-05
0.03572208304452786	2.2150830831738555e-05
0.04139333334421674	2.2618922889854347e-05
0.04731686852781396	2.294773237941877e-05
0.053422697031889343	2.3133792362471314e-05
0.059631426863063	2.3174589256930148e-05
0.06585426623926073	2.3068583427354724e-05
0.0719930248042806	2.2815223112299094e-05
0.07794011536942966	2.2414951592936454e-05
0.08357855613721893	2.1869207556269904e-05
0.08878197336316129	2.1180418648991868e-05
0.09341460441264776	2.0351988267775824e-05
0.09733130117072748	1.9388275675521214e-05
0.10037753376341767	1.8294569580934146e-05
0.10238939454995559	1.7077055359938313e-05
0.1031936023461957	1.574277614515944e-05
0.10260750684017823	1.429958804608205e-05
0.1004390931617615	1.2756109808222896e-05
0.09648698656913962	1.1121667250832933e-05
0.09054045721606521	9.406232863847126e-06
0.08237942496467578	7.6203609721348445e-06
0.0717744642099859	5.775118908756503e-06
0.05848680868336116	3.882014662752974e-06
0.042268356203633534	1.952921490170137e-06
0.02286167334595351	8.673617379884035e-19

Максимальная погрешность метода Зейделя равна 2.3174589256930148e-05 и данная точность достигается на 530-ой итерации

График зависимости погрешностей от количества итераций:



Метод релаксации

Метод релаксации – итерационный метод решения систем линейных алгебраических уравнений. Является обобщенным методом Зейделя, который, благодаря наличию итерационного параметра w, позволяет в некоторых случаях ускоряет итерационный процесс. Рассмотрим итерационный процесс:

$$y_i^{k+1} = (1-w)x_i^k + w(-\sum_{j=1}^{i-1}\frac{a_{ij}}{a_{jj}}x_j^{k+1} - \sum_{j=i+1}^{n}\frac{a_{ij}}{a_{ii}}x_j^k + \frac{f_i}{a_{ii}}),$$

где $i = \overline{1,n}, k = 0,1,\ldots$

Число w называют релаксационным параметром. При $w = 1$ метод переходит в метод Зейделя.

Проверка условий устойчивости сходимости метода. Наша матрица симметрична и положительно определена, параметр релаксации удовлетворяет условию: $0 < w < 2$, а это и есть условие сходимости метода релаксации.

Решив систему линейных алгебраических уравнений методом релаксации, получим следующие результаты:

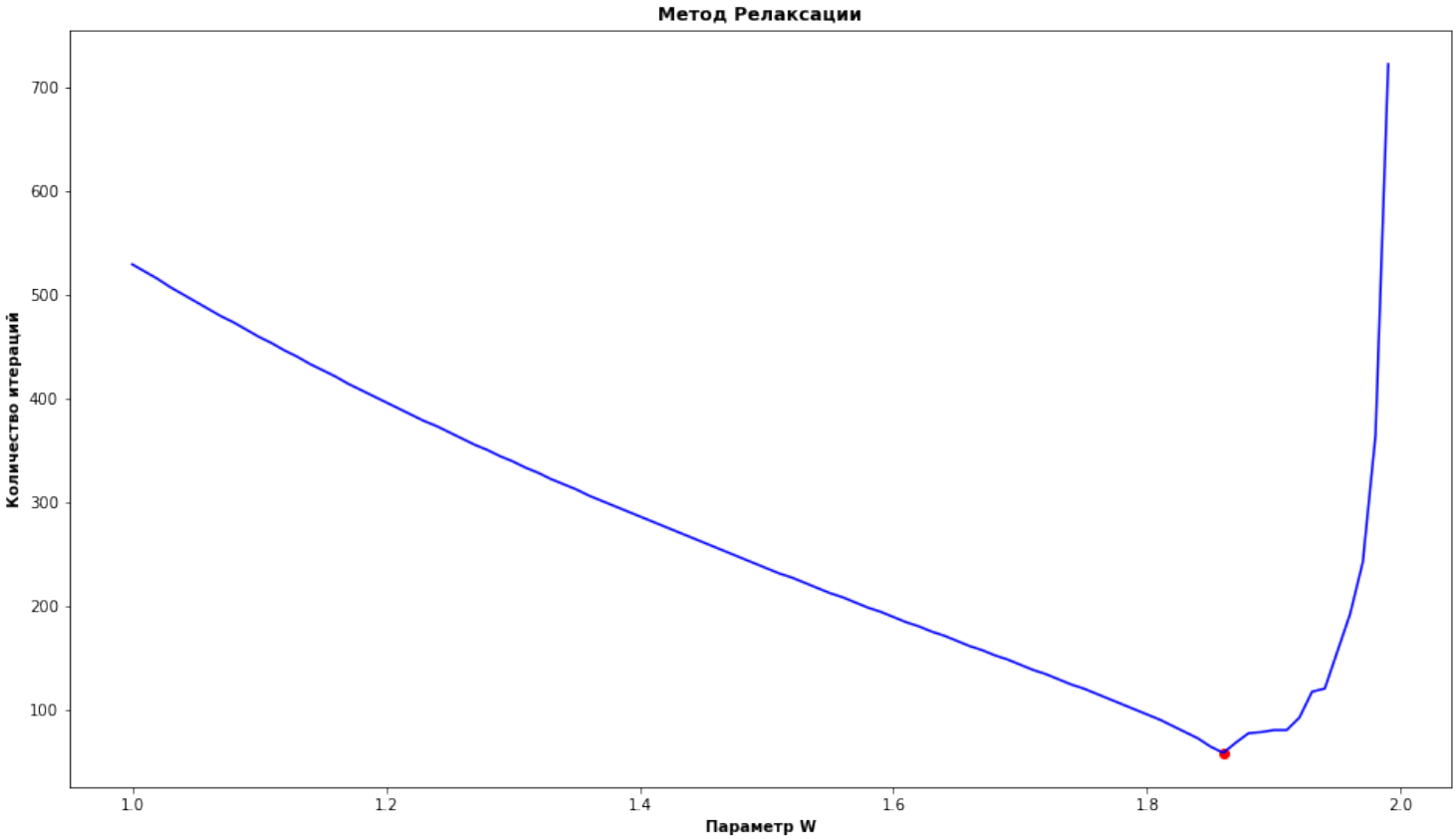
Метод Релаксации	Погрешность метода Релаксации
9.917174446992537e-06	4.05239158944225e-06
0.00011147523546223573	3.1650414280212386e-06
0.00038250661353646714	2.2469992036660614e-06
0.0008916747167580411	1.318422975881469e-06
0.0016984725948801607	4.015502314520065e-07
0.0028532235922918473	4.792211645475658e-07
0.004397083818890445	1.2972518911668176e-06
0.0063620462940602715	2.023580326233009e-06
0.008770946640670968	2.626842289673052e-06
0.011637470224030437	3.073191716280034e-06
0.01496616064572305	3.3262225988896265e-06
0.018752429514756645	3.346892523446959e-06
0.02298256742887058	3.093448114505591e-06
0.027633756107567197	2.521352695300928e-06
0.03267408162571149	1.5832164528637588e-06
0.038062548702636756	2.2872939307535568e-07

Метод Релаксации	Погрешность метода Релаксации
0.04374909600678912	1.595402654573923e-06
0.049674612440211534	3.945518716507888e-06
0.05577095437073048	6.881059813538756e-06
0.06196096378269717	1.0464622316860831e-05
0.06815848731962386	1.5288604949531563e-05
0.0742490087023547	6.503728967072349e-07
0.08013511657280167	6.128109736100299e-07
0.08570048558908296	5.62538257713424e-07
0.09081945689577922	5.011901976712798e-07
0.09535703527350284	4.305041075560942e-07
0.09916888649787582	3.522997540619712e-07
0.10210133490100776	2.6845899076563366e-07
0.10399136112979956	1.8090442727635922e-07
0.10466660009650804	9.157712527359252e-08
0.10394533911800526	2.4133113537064266e-09
0.10163651624106873	8.467990672772832e-08
0.09753971875185896	1.6784981506697688e-07
0.09144518186848036	2.4532438499137027e-07
0.08313378761620438	3.154395683069647e-07
0.07237706388554932	3.7666760810319694e-07
0.05893718367397923	4.276463631538649e-07
0.042566964512503476	4.6720964233006035e-07
0.023009868078934726	4.94418544287091e-07

Максимальная погрешность метода релаксации равна 1.5288604949531563e-05 и данная точность достигается на 59-ой итерации. Можно заметить, что метод релаксаци и сходится быстрее метода Зейделя.

Оптимальный параметр для метода релаксации $w = 1.86$.

График зависимости числа итераций от параметра w:



Код программы

```
# подключаем библиотеки
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Исходные Данные и функции
alpha = 3
beta = 1
gamma = 1
n = 40
h = 1/n
eps = h**3
print("eps =", eps)


def q(x):
    '''Заданная функция q:
    q(x) = 1+x
    '''
    return 1 + x


def p(x, _gamma=gamma):
    '''Заданная функция p:
    p(x) = 1+x^gamma
    '''
    return 1 + x ** _gamma
```

```
def p_(x, _gamma=gamma):
    '''Производная функции p
    d/dx(1 + x^ gamma ) = gamma x^( gamma - 1)
    '''
    return _gamma * x ** (_gamma - 1)


def g(i, _h=h):
    '''Заданная функция g:
    g_i = q(i*h)
    '''
    return q(i * _h)


def a(i, _h=h):
    '''Заданная функиция a:
    a_i = p(i*h)
    '''
    return p(i * _h)


def u(x, _alpha=alpha, _beta=beta):
    '''Заданная функция u:
    u(x) = (x^alpha)(1-x)^beta
    '''
    return (x ** _alpha) * (1 - x) ** _beta
def u_(x, _alpha=alpha, _beta=beta):
    '''Производная функции u:
    d/dx(x^α (1 - x)^β) = α x^(α - 1) (1 - x)^β - β x^α (1 - x)^(β - 1)
    '''
    return (_alpha * x ** (_alpha - 1) * (1 - x) ** _beta) - x ** _alpha * _beta * (1 - x) ** (_beta - 1)
```

```
def u__(x, _alpha=alpha, _beta=beta):
    '''Производная второго порядка функции u:
    d/dx(α x^(α - 1) (1 - x)^β - β x^α (1 - x)^(β - 1)) =
    (α - 1) α x^(α - 2) (1 - x)^β - 2 α β x^(α - 1) (1 - x)^(β - 1)
    + (β - 1) β x^α (1 - x)^(β - 2)'''
    first = (_alpha - 1) * _alpha * x**(_alpha - 2) * (1 - x)**_beta \
    - 2 * _alpha * _beta * x**(_alpha - 1) * (1 - x)**(_beta - 1)
    second = (_beta - 1) * _beta * x**_alpha * (1 - x)**(_beta - 2)
    return first + second


def f(x, _h=h):
    '''Заданная функйия f:
    f_i = f(i*h), где f(x) = -(p(x)u'(x))' + q(x)u(x)
    '''
    x = x * _h
    return -(p_(x) * u_(x) + p(x) * u__(x)) + q(x) * u(x)


# Преобразуем нашу систему линейных алгебраических уравнений к удобному формату
def A(i):
    return -a(i)
def B(i, _h=h):
    return a(i) + a(i + 1) + _h ** 2 * g(i)
def C(i):
    return -a(i + 1)
def D(i):
    return f(i) * h ** 2
```

```
# Функция проверки решений
def checkSolution(solution):
    '''функция нахождения погрешности
    solution - список, результат работы метода
    '''
    error = list()
    for i in range(len(solution)):
        if i == 0:
            left = B(1) * solution[i] + C(1) * solution[i + 1]
        elif i == len(solution) - 1:
            left = A(i + 1) * solution[i - 1] + B(i + 1) * solution[i]
        else:
            left = A(i + 1) * solution[i - 1] + B(i + 1) * solution[i] + C(i + 1) * solution[i + 1]

        right = D(i + 1)
        error.append(abs(left - right))
    return max(error), error

# Метод Прогонки
def progonka():
    """Метод прогонки
    применяют к случаю трехдиагональной системы уравнений
    y_{i} = alpha_{i} * y_{i+1} + beta_{i}, где
```

```
    alpha_{i} = -(c_{i})/(A_{i}*alpha_{i-1} + B_{i})

    beta_{i} = (D_{i} - A_{i}*beta_{i-1})/(B_{i} + A_{i}*alpha_{i-1})

    """

    y = [0 for i in range(n)]

    alpha_c = [0 for i in range(n - 1)]

    beta_c = [0 for i in range(n - 1)]

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

        alpha_c[i] = -C(i) / (B(i) + A(i) * alpha_c[i - 1])

        beta_c[i] = (D(i) - A(i) * beta_c[i - 1]) / (B(i) + A(i) * alpha_c[i - 1])

    y[-1] = (D(n - 1) - A(n - 1) * beta_c[n - 2]) / (B(n - 1) + A(n - 1) * alpha_c[n - 2])

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

        y[i] = alpha_c[i] * y[i + 1] + beta_c[i]

    return checkSolution(y[1:]), y[1:]


# Метод Релаксации

def sor(w,iters=0):

    '''Метод Релаксации

    Обобщенй метод Зейделя, с параметром w

    y_{i}^{k+1} = (1-w)*y_{i}^{k} +

    w * (-\SUM_{j=1}^{i-1} (a_{ij}/a_{ii}) * y_{i}^{k+1}

    - \Sum_{j=i+1}^{n} (a_{ij}/a_{ii}} * y_{j}^{k}

    + f_{i}/a_{ii})

    '''
```

```
y = [0 for i in range(n)]
prev = y.copy()
iterations = 0
while True:
    iterations += 1
    y[1] = (1-w) * y[1] + w * ( D(1) - C(1) * y[2]) / B(1)
    for i in range(2, n - 1):
        y[i] =(1-w) * y[i] + w* (D(i) - A(i) * y[i - 1] - C(i) * y[i + 1]) / B(i)

    y[n - 1] = (1-w) * y[n-1] + w *(D(n - 1) - A(n - 1) * y[n - 2]) / B(n - 1)
    error = [abs(y[i] - prev[i]) for i in range(len(y))]
    max_error = max(error)
    if iters != 0 and iters == iterations:
        break
    elif iters == 0:
        if max_error < eps:
            break
    prev = y.copy()
return iterations, checkSolution(y[1:]), y[1:]
```

```
# Метод Зейделя

def seidel(iters=0):

    '''Метод Зейделя

    y_{i}^{k+1} = -\SUM_{j=1}^{i-1} (a_{ij}/a_{ii}) * y_{i}^{k+1}

    - \Sum_{j=i+1}^{n} (a_{ij}/a_{ii}} * y_{j}^{k}

    + f_{i}/a_{ii}

    '''

    y = [0 for i in range(n)]

    prev = y.copy()

    iterations = 0

    while True:

        iterations += 1

        y[1] = (D(1) - C(1) * y[2]) / B(1)

        for i in range(2, n - 1):

            y[i] = (D(i) - A(i) * y[i - 1] - C(i) * y[i + 1]) / B(i)

        y[n - 1] = (D(n - 1) - A(n - 1) * y[n - 2]) / B(n - 1)

        error = [abs(y[i] - prev[i]) for i in range(len(y))]

        max_error = max(error)

        if iters != 0 and iters == iterations:

            break

        elif iters == 0:

            if max_error < eps:

                break
```

```
        prev = y.copy()
    return iterations, checkSolution(y[1:]), y[1:]


# Графики
# График Релаксакции
W = np.linspace(1,2,100,endpoint=False)
errors = np.empty(100)
iterations = np.empty(100)
for i in range (W.shape[0]):
    iterations[i] = sor(W[i])[0]
W_best = W[np.argmin(iterations)] # лучший параметр ω
fig, ax = plt.subplots(figsize=(16, 9))
ax.plot(W,iterations, color='blue')
ax.scatter(W[np.argmin(iterations)], np.min(iterations),c="r")
ax.set_title('Метод Релаксации', fontweight='bold')
plt.xlabel('Параметр W', fontweight='bold')
plt.ylabel('Количество итераций', fontweight='bold')
plt.savefig('sor.png')
# График Зейделя
errors = np.empty(1000)
iterations = np.linspace(1,1000,1000)
for i in range (iterations.shape[0]):
    errors[i] = seidel(iterations[i])[1][0]
```

```
fig, ax = plt.subplots(figsize=(16, 9))

ax.plot(iterations[2:], errors[2:], color='yellow')

ax.set_title('Метод Зейделя',fontweight='bold')

plt.xlabel('Количество итераций', fontweight='bold')

plt.ylabel('Погрешность', fontweight='bold')

plt.savefig(‘seidel.png')


# Вывод результатов

y_r = sor(W_best)

y_z = seidel()

y_p = progonka()

print(f'Итераций в методе Релаксации: {y_r[0]}')

print(f'Максимальная погрешность метода Релаксации: {y_r[1][0]}')

print(f'Оптимальный W: {W_best}')

print("------------------------")

print(f'Итераций в методе Зейделя: {y_z[0]}')

print(f'Максимальная погрешность метода Зейделя: {y_z[1][0]}')

print("------------------------")

print(f'Максимальная погрешность метода прогонки: {y_p[0][0]}')

columns=["Итераций","Максимальная погрешность"]

index=["Релаксация","Зейделя","Прогонка"]

df = pd.DataFrame([[y_r[0], y_r[1][0]],  [y_z[0],  y_z[1][0]],  [None,y_p[0][0]]],
columns=columns, index=index)

print(df)
```

```
# Таблица Релаксации

err = [x for x in y_r[1][1]]

y = [x for x in y_r[2]]

data = {'Метод Релаксации': y, 'Погрешность метода Релаксации': err}

df = pd.DataFrame(data)

df.to_csv('sor.csv', encoding='utf-8', index=False)

print(df)


# Таблица Зейдель

err = [x for x in y_z[1][1]]

y = [x for x in y_z[2]]

data = {'Метод Зейделя': y, 'Погрешность метода Зейделя': err}

df = pd.DataFrame(data)

df.to_csv('seidel.csv', encoding='utf-8', index=False)

print(df)

# Таблица Прогонка

i = [x for x in range(1, n)]

y = [x for x in y_p[1]]

data = {'Итерация i': i,  'Метод Прогонка': y }

df = pd.DataFrame(data)

df.to_csv('progonka.csv', encoding='utf-8', index=False)

print(df)
```