МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.Ломоносова

Факультет вычислительной математики и кибернетики

Компьютерный практикум по учебному курсу «ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»

ЗАДАНИЕ № 1.

ОТЧЕТ

о выполненном задании

студента 204 учебной группы факультета ВМК МГУ

Скрябина Глеба Денисовича

(фамилия, имя, отчество студента)

гор. Москва

2020 год

Оглавление

Подвариант 1

Постановка задачи и ее цели

Целью работы является изучение классического и модифицированного методов Гаусса, применяемых для решения системы линейных алгебраических уравнений.

Постановка задачи. Дана система уравнений $Ax=f$ порядка $n \times n$ с невырожденной матрицей A. Необходимо написать программу, решающую систему линейных алгебраических уравнений заданного пользователем размера (n – параметр программы) методом Гаусса и методом Гаусса с выбором главного элемента. Предусмотреть возможность задания элементов матрицы системы и ее правой части как во входном файле данных, так и путем задания специальных формул.

Необходимо:

1. Решить указанные ниже СЛАУ методом Гаусса и методом Гаусса с выбором главного элемента;
2. Вычислить определитель матрицы det(A);
3. Вычислить обратную матрицу A^{-1};
4. Определить число обусловленности $M_A=\|A\|\times\|A^{-1}\|$;
5. Исследовать вопрос вычислительной устойчивости метода Гаусса (при больших значениях параметра n);
6. Правильность решения СЛАУ подтвердить системой тестов.

Задание 1

$$\begin{cases} 3x_1 - 2x_2 + 2x_3 - 2x_4 = 8, \\ 2x_1 - x_2 + 2x_3 = 4, \\ 2x_1 + x_2 + 4x_3 + 8x_4 = -1, \\ x_1 + 3x_2 - 6x_3 + 2x_4 = 3. \end{cases} \quad \begin{cases} 2x_1 + 3x_2 + x_3 + 2x_4 = 4, \\ 4x_1 + 3x_2 + x_3 + x_4 = 5, \\ x_1 - 7x_2 - x_3 - 2x_4 = 7, \\ 2x_1 + 5x_2 + x_3 + x_4 = 1. \end{cases} \quad \begin{cases} x_1 - x_2 + x_3 - x_4 = 0, \\ 4x_1 - x_2 - x_4 = 0, \\ 2x_1 + x_2 - 2x_3 + x_4 = 0, \\ 5x_1 + x_2 - 4x_4 = 0. \end{cases}$$

Задание 2.

Элементы матрицы A вычисляются по формулам:

$$A_{ij} = \begin{cases} q_M^{i+j} + 0.1 \cdot (j - i), & i \neq j, \\ (q_M - 1)^{i+j}, & i = j, \end{cases}$$

где

- $q_M = 1.001 - 2 \cdot M \cdot 10^{-3}$,
- $i, j = 1, \ldots n$,
- M=4,
- n=100.

Элементы вектора f (вектор правой части системы) задаются формулами:

$$b_i = n \cdot \exp\left(\frac{x}{i}\right) \cdot \cos(x)$$

.

Метод решения

Рассмотрим метод решения систем линейных алгебраических уравнений, в которых число уравнений равно числу неизвестных методом Гаусса:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= f_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= f_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= f_n \end{aligned} \qquad (1)$$

Этот метод можно условно разделить на два этапа. На первом этапе (прямой ход) система (1) приводится к треугольному виду. Затем на втором этапе (обратный ход) осуществляется последовательное отыскание неизвестных x_1 , … , x_n из этой треугольной системы.

Не ограничивая общности, будем считать, что коэффициент a_{11}, который называют ведущим элементом первого шага, отличен от нуля (в случае $a_{11} = 0$ поменяем местами уравнения с номерами 1 и i, при котором $a_{i1} \neq 0$; поскольку система предполагается невырожденной, то такой номер i заведомо найдется).

Разделим все члены первого уравнения на a_{11} и введем в качестве новых коэффициентов c_{1i}, i=2,…,n и правой части y_1 отношения

$$c_{12} = \frac{a_{12}}{a_{11}}, \quad c_{13} = \frac{a_{13}}{a_{11}}, \quad \ldots \quad c_{1n} = \frac{a_{1n}}{a_{11}}, \; y_1 = \frac{f_1}{a_{11}}. \qquad (2)$$

Вычтем из каждого i -го уравнения системы ($i = 2, \ldots, n$) первое уравнение умноженное на a_{i1}. Проделав это, мы исключим неизвестное x_1 из всех уравнений, кроме первого.

Преобразованная таким образом система (1) примет эквивалентный вид:

$$\begin{aligned} x_1 + c_{12}x_2 + c_{13}x_3 + \ldots + c_{1n}x_n &= y_1 \\ a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2n}^{(1)}x_n &= f_2^{(1)} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\ a_{n2}^{(1)}x_2 + a_{n3}^{(1)}x_3 + \ldots + a_{nn}^{(1)}x_n &= f_n^{(1)} \end{aligned} \qquad (3)$$

Значения новых коэффициентов и правых частей системы (3) вычисляются по формулам:

$$a_{ij}^{(1)} = a_{ij} - a_{i1}\frac{a_{1j}}{a_{11}}, \; f_i^{(1)} = f_i - a_{i1}\frac{f_1}{a_{11}} \qquad (4)$$

Выделим из (3) «укороченную» систему, содержащую n −1 уравнение

$$\begin{aligned} a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2n}^{(1)}x_n &= f_2^{(1)} \\ a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + \ldots + a_{3n}^{(1)}x_n &= f_3^{(1)} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\ a_{n2}^{(1)}x_2 + a_{n3}^{(1)}x_3 + \ldots + a_{nn}^{(1)}x_n &= f_n^{(1)} \end{aligned}$$

Продолжая далее процесс исключения после (n−1) шага сведем исходную систему к виду:

$$
\begin{aligned}
x_1 + c_{12}x_2 + c_{13}x_3 + \ldots + c_{1n}x_n &= y_1 \\
x_2 + c_{23}x_3 + \ldots + c_{2n}x_n &= y_2 \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
x_{n-1} + c_{n-1,n}x_n &= y_{n-1} \\
x_n &= y_n
\end{aligned} \qquad (5)
$$

Или в матричной форме $C\mathbf{x} = \mathbf{y}$, где матрица C является верхней треугольной матрицей с единицами на главной диагонали

$$
C = \begin{bmatrix} 1 & c_{12} & c_{13} & \ldots & c_{1n} \\ 0 & 1 & c_{23} & \ldots & c_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix} \qquad (6)
$$

Построение системы (5) завершает прямой ход метода Гаусса.
Обратный ход состоит в последовательном определении неизвестных из системы (5) в обратном порядке:

$$
\begin{aligned}
x_n &= y_n \\
x_{n-1} &= y_{n-1} - c_{n-1,n}x_n \\
x_{n-2} &= y_{n-2} - c_{n-2,n-1}x_{n-1} - c_{n-2,n}x_n \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
x_1 &= y_1 - c_{12}x_2 - c_{13}x_3 - \ldots - c_{1n}x_n
\end{aligned} \qquad (7)
$$

Описанная выше процедура решения системы (1) методом Гаусса может оказаться неустойчивой по отношению к случайным ошибкам, которые неизбежны при компьютерных расчетах в результате округления чисел из-за конечной длины машинного слова. Действительно, предположим, что в процессе приведения системы (1) к треугольному виду (5) у матрицы C (6) образовались большие по модулю элементы: $|c_{i,j}| > 1$. Тогда при вычислении неизвестных по формулам (7) во время обратного хода умножение найденных с ошибками округления чисел x_i на большие по модулю элементы матрицы C приведет к увеличению этих ошибок. Наоборот, если матрица C оказалась такой, что все ее элементы удовлетворяют условию

$$
|c_{i,j}| \le 1, \qquad (8)
$$

то роль ошибок округления в процессе вычислений будет нивелироваться.
Опишем, как можно добиться выполнения условия (8). Приступая к первому шагу прямого хода метода Гаусса, рассмотрим элементы $a_{1,j}$ первой строки матрицы A и найдем среди них элемент наибольший по модулю. Пусть он имеет номер j_1. Поменяем в системе (1) первый столбец и столбец с номером j_1 местами, изменив соответствующим образом нумерацию неизвестных. В результате такой процедуры наибольший по модулю элемент первой строки станет ведущим элементом первого шага $a_{1,1}$. Благодаря этому элементы $c_{1,j}$ первой строки матрицы C, которые рассчитываются по формулам (2), будут удовлетворять неравенству (8).
Процедуру выделения наибольшего по модулю элемента в очередной строке и превращения его в ведущий элемент нужно затем повторять во время каждого шага

прямого хода метода Гаусса. В этом случае все элементы $c_{i,j}$ треугольной матрицы C (6) будут удовлетворять неравенствам (8), обеспечивая устойчивость метода по отношению к ошибкам округления. Такой способ коррекции называется выбором ведущего элемента по строке.
Дополнительно отметим, что первый этап метода Гаусса может быть использован для вычисления определителя матрицы A. Прямой ход метода Гаусса основан на многократном выполнении операции сложения одной из строк матрицы с другой строкой, взятой с некоторым множителем, что не меняет определителя. Следует лишь учесть, что при делении ведущей строки на ее диагональный элемент определитель также делится на этот элемент. Кроме того, иногда приходится переставлять столбцы при выборе главного элемента по строке. Поскольку определитель приведенной треугольной системы (матрицы C) всегда равен единице, то определитель Δ исходной системы равен

$$\Delta = \det A = (-1)^k a_{11} a_{22}^{(1)} \ldots a_{nn}^{(n-1)}$$

где k – число перестановок столбцов в процессе редукции матрицы A к треугольной матрице C.

Описание программы

Метод Гаусса. Если параметр mainElement = True, то используется метод Гаусса с выбором главного элемента, иначе обычный метод Гаусса.

```
def gauss(sle, mainElement = True):
    N = sle.N
    A = [[sle.A[i][j] for j in range(N)] for i in range(N)]
    B = [sle.B[i] for i in range(N)]
    col = 0
    while (col < N):
        if mainElement:
            cr = getCurRow(A, col)
            swap(A, B, cr, col)
        if A[col][col] == 0:
            return None
        triangleStep(A, B, col)
        col += 1
    X = [0 for b in B]
    for i in range(N - 1, -1, -1):
        X[i] = B[i] - sum(x * a for x, a in zip(X[(i + 1):], A[i][(i + 1):]))
    return X
```

Вспомогательная функция, используемая при приведении матрицы к треугольному виду:

```
def triangleStep(A, B, col, BAct=True):
    div = A[col][col]
    A[col] = [a / div for a in A[col]]  #строка
    if BAct: B[col] /= div
    # сложение строки системы с другой строкой, умноженной на число
    for r in range(col + 1, len(A)):
        mul = -A[r][col]
        A[r] = [(a + k * mul) for a, k in zip(A[r], A[col])]
        if BAct: B[r] += B[col] * mul
    return div
```

Вычисление детерминанта:

```
def getCurRow(A, col):
    cr = None
    for r in range(col, len(A)):
        cr = r if cr is None or abs(A[r][col]) > abs(A[cr][col]) else cr
    return cr

def det(inpA):
    N = len(inpA)
    A = [[inpA[i][j] for j in range(N)] for i in range(N)]
    col = 0
    determinant = 1
    while (col < N):
        cr = getCurRow(A, col)
        if cr != col:
            A[cr], A[col] = A[col], A[cr]
            determinant *= -1
        if A[col][col] == 0:
            return 0
        determinant *= triangleStep(A, None, col, BAct=False)
        col += 1
    return determinant
```

Вычисление обратной матрицы:

```
def minor(A, ix, jx):
    N = len(A)
    M = [[A[i][j] for j in range(N)] for i in range(N)]
    del M[ix]
    for i in range(N - 1):
        del M[i][jx]
    return M

def inverse(A):
    N = len(A)
    AT = [[A[i][j] for i in range(N)] for j in range(N)]
    result = [[0 for j in range(N)] for i in range(N)]
    for i in range(N):
        for j in range(N):
            tmp = minor(AT, i, j)
            result[i][j] = (-1 if (i + j) % 2 else 1) * det(tmp) / det(AT)
    #result = [[(-1 if (i + j) % 2 else 1) * det(minor(AT, i, j)) /
    #          det(AT) for j in range(N)] for i in range(N)]
    return result
```

Вычисление числа обусловленности:

```
norm = lambda A : max([sum([abs(c) for c in row]) for row in A])
cond = lambda A : norm(A) * norm(inverse(A))
```

Класс, описывающий СЛАУ:

```
class SLE:
    # SLE - System of linear equations
    def __init__(self, A, B):
        self.N = len(B)
        self.A = [[A[i][j] for j in range(self.N)] for i in range(self.N)]
        self.B = [B[i] for i in range(self.N)]


    def deepcopy(self, A, B):
        for i in range(self.N):
            for j in range(self.N):
                A[i][j] = self.A[i][j]
            B[i] = self.B[i]


    def show(self, shift="", b=True):
        for i in range(self.N):
            print(end = shift)
            for j in range(self.N):
                print("{0:6.2f}x{1}".format(self.A[i][j], j + 1), end = " ")
            if b:
                print(" = {0:6.2f}".format(self.B[i]))
            else:
                print()
        print()
```

Считывание СЛАУ из файла:

```
class DataSet:
    def __init__(self):
        self.root = './Data/'
        # var 13
        self.sle = [self.getSLEfile(f'test{i + 1}.txt') for i in range(3)]
        inpx = float(input("Введите x для вычисления матрицы из примера 2: "))
        self.sle.append(self.getSLEcustom(inpx))
        self.len = len(self.sle)


    def getSLEfile(self, filename):
        with open(self.root + filename) as f:
            read_data = f.read()
        rd1 = " ".join(read_data.split())
        rd2 = " ".join(rd1.split(sep=','))
        data = list(map(float, rd2.split()))

        n = int(len(data)**(1 / 2))
        A = [[data[i * (n + 1) + j] for j in range(n)] for i in range(n)]
        B = [data[i * (n + 1) + n] for i in range(n)]
        return SLE(A, B)


    def getSLEcustom(self, x):
        # ex 2 var 4
        M = 4
        n = 10 # 100
        e = 2.718281828459045
        # using Euler's formula: e^ix = cosx + isinx
        cos = lambda a: (e ** (a * 1j)).real
        bi = lambda i: n * e ** (x / i) * cos(x)
        qm = 1.001 - 2 * M * 0.001
        Aij = lambda i, j: qm**(i+j) + 0.1 * (j-i) if i!=j else (qm-1)**(i+j)
        A = [[Aij(i, j) for j in range(1, n + 1)] for i in range(1, n + 1)]
        B = [bi(i) for i in range(1, n + 1)]
        return SLE(A, B)
```

Тестирование

Проверим результаты работы с использованием онлайн системы WolframAlpha.

1. $$\begin{cases} 3x_1 - 2x_2 + 2x_3 - 2x_4 = 8, \\ 2x_1 - x_2 + 2x_3 = 4, \\ 2x_1 + x_2 + 4x_3 + 8x_4 = -1, \\ x_1 + 3x_2 - 6x_3 + 2x_4 = 3. \end{cases}$$

	Программа	**WolframAlpha**
Решение методом Гаусса	x1 = 2.00 x2 = -3.00 x3 = -1.50 x4 = 0.50	x1 = 2 x2 = -3 x3 = -1.5 x4 = 0.5
Решение заданной СЛАУ методом Гаусса с выбором главного элемента	x1 = 2.00 x2 = -3.00 x3 = -1.50 x4 = 0.50	
Определитель матрицы A	24.00	24
Обратная матрица A^-1	-0.50 1.33 -0.17 0.17 -5.00 8.67 -1.33 0.33 -2.00 3.50 -0.50 0.00 1.75 -3.17 0.58 -0.08	-0.50 1.33 -0.17 0.17 -5.00 8.67 -1.33 0.33 -2.00 3.50 -0.50 0.00 1.75 -3.17 0.58 -0.08
Число обусловленности MA = ||A||×||A^-1||	230.00	108,076

2. $$\begin{cases} 2x_1 + 3x_2 + x_3 + 2x_4 = 4, \\ 4x_1 + 3x_2 + x_3 + x_4 = 5, \\ x_1 - 7x_2 - x_3 - 2x_4 = 7, \\ 2x_1 + 5x_2 + x_3 + x_4 = 1. \end{cases}$$

	Программа	**WolframAlpha**
Решение методом Гаусса	x1 = -3.00 x2 = -5.00 x3 = 39.00 x4 = -7.00	x1 = -3.00 x2 = -5.00 x3 = 39.00 x4 = -7.00
Решение заданной СЛАУ методом Гаусса с выбором главного элемента	x1 = -3.00 x2 = -5.00 x3 = 39.00 x4 = -7.00	
Определитель матрицы A	2.00	2
Обратная матрица A^-1	-1.00 2.00 -1.00 -2.00 -1.00 1.50 -1.00 -1.50 8.00 -14.50 9.00 16.50 -1.00 3.00 -2.00 -4.00	-1.00 2.00 -1.00 -2.00 -1.00 1.50 -1.00 -1.50 8.00 -14.50 9.00 16.50 -1.00 3.00 -2.00 -4.00
Число обусловленности MA = ||A||×||A^-1||	528.00	272.865

3. $$\begin{cases} x_1 - x_2 + x_3 - x_4 = 0, \\ 4x_1 - x_2 - x_4 = 0, \\ 2x_1 + x_2 - 2x_3 + x_4 = 0, \\ 5x_1 + x_2 - 4x_4 = 0. \end{cases}$$

	Программа	**WolframAlpha**
Решение методом Гаусса	Матрица A вырождена	-
Решение заданной СЛАУ методом Гаусса с выбором главного элемента		
Определитель матрицы A	0	0
Обратная матрица A^-1	-	-
Число обусловленности MA = ||A||×||A^-1||	-	-

4. $$A_{ij} = \begin{cases} q_M^{i+j} + 0.1 \cdot (j - i), & i \neq j, \\ (q_M - 1)^{i+j}, & i = j, \end{cases}$$

где $q_M = 1.001 - 2 \cdot M \cdot 10^{-3}$, $i, j = 1,...n$, M=4, n=100.

Элементы вектора f (вектор правой части системы) задаются формулами:

$$b_i = n \cdot \exp\left(\frac{x}{i}\right) \cdot \cos(x)$$

.

Проверим работу программы для n=10 (для удобства проверки в WolframAlpha) и x=4:

	Программа	**WolframAlpha**
Решение методом Гаусса	x1 = -1054.19 x2 = -1174.30 x3 = -953.50 x4 = -706.17 x5 = -447.04 x6 = -178.79 x7 = 97.85 x8 = 382.69 x9 = 675.73 x10 = 977.04	x1 = -1054.19 x2 = -1174.30 x3 = -953.50 x4 = -706.17 x5 = -447.04 x6 = -178.79 x7 = 97.85 x8 = 382.69 x9 = 675.73 x10 = 977.04
Решение заданной СЛАУ методом Гаусса с выбором главного элемента	x1 = -1054.19 x2 = -1174.30 x3 = -953.50 x4 = -706.17 x5 = -447.04 x6 = -178.79 x7 = 97.85 x8 = 382.69 x9 = 675.73 x10 = 977.04	
Определитель матрицы A	0.29	0.29
Обратная матрица A^-1	2.56 2.66 1.71 0.73 -0.27 -1.31 -2.37 -3.46 -4.59 -5.74 2.98 1.19 1.43 0.63 -0.21 -1.06 -1.95 -2.85 -3.78 -4.74 2.37 1.77 0.11 0.52 -0.14 -0.81 -1.51 -2.22 -2.96 -3.71 1.74 1.31 0.86 -0.66 -0.07 -0.56 -1.06 -1.57 -2.10 -2.65 1.09 0.83 0.56 0.29 -1.07 -0.29 -0.60 -0.91 -1.23 -1.56 0.43 0.34 0.26 0.17 0.07 -1.11 -0.12 -0.22 -0.33 -0.43 -0.26 -0.16 -0.06 0.04 0.15 0.26 -0.74 0.48 0.60 0.72 -0.96 -0.68 -0.38 -0.08 0.23 0.54 0.87 0.09 1.55 1.91 -1.68 -1.21 -0.72 -0.21 0.30 0.84 1.39 1.95 1.40 3.13 -2.43 -1.75 -1.06 -0.35 0.39 1.14 1.92 2.72 3.54 3.23	2.56 2.66 1.71 0.73 -0.27 -1.31 -2.37 -3.46 -4.59 -5.74 2.98 1.19 1.43 0.63 -0.21 -1.06 -1.95 -2.85 -3.78 -4.74 2.37 1.77 0.11 0.52 -0.14 -0.81 -1.51 -2.22 -2.96 -3.71 1.74 1.31 0.86 -0.66 -0.07 -0.56 -1.06 -1.57 -2.10 -2.65 1.09 0.83 0.56 0.29 -1.07 -0.29 -0.60 -0.91 -1.23 -1.56 0.43 0.34 0.26 0.17 0.07 -1.11 -0.12 -0.22 -0.33 -0.43 -0.26 -0.16 -0.06 0.04 0.15 0.26 -0.74 0.48 0.60 0.72 -0.96 -0.68 -0.38 -0.08 0.23 0.54 0.87 0.09 1.55 1.91 -1.68 -1.21 -0.72 -0.21 0.30 0.84 1.39 1.95 1.40 3.13 -2.43 -1.75 -1.06 -0.35 0.39 1.14 1.92 2.72 3.54 3.23
Число обусловленности MA = ||A||×||A^-1||	332.06	159.90

Выводы

В ходе выполнения поставленной задачи были реализованы метод Гаусса и метод Гаусса с выбором главного элемента. Тестирование показало удобство и высокую точность данных методов особенно для систем с небольшим порядком и хорошей обусловленностью. Метод Гаусса с выбором главного элемента является более устойчивым по отношению к случайным ошибкам, возникающих при компьютерных расчетах в результате округления чисел из-за конечной длины машинного слова.

Подвариант 2

Постановка задачи и ее цели

Целью работы является изучение метода верхней релаксации, используемого для численного решения систем линейных алгебраических уравнений, а также изучить скорость сходимости этого метода в зависимости от выбора итерационного параметра.
Постановка задачи. Дана система уравнений $Ax=f$ порядка $n \times n$ с невырожденной матрицей A. Необходимо написать программу численного решения данной системы линейных алгебраических уравнений (n – параметр программы), использующую численный алгоритм итерационного метода верхней релаксации:

$$(D + \omega A^{(-)})\frac{x^{k+1} - x^k}{\omega} + Ax^k = f,$$

где $D, A^{(-)}$ - соответственно диагональная и нижняя треугольные матрицы, k - номер текущей итерации, ω - итерационный параметр.
Необходимо предусмотреть возможность задания элементов матрицы системы и ее правой части как во входном файле данных, так и путем задания специальных формул.
Необходимо:

1. Решить заданную СЛАУ итерационным методом верхней релаксации;
2. Разработать критерий остановки итерационного процесса, гарантирующий получение приближенного решения исходной системы СЛАУ с заданной точностью;
3. Провести эксперименты с различными значениями итерационного параметра ω (в случае симметрической положительно определенной матрицы системы известно, что для сходимости итераций следует выбирать $0 < \omega < 2$; при $\omega = 1$ метод верхней релаксации совпадает с методом Зейделя);
4. Правильность решения СЛАУ подтвердить системой тестов (например, можно использовать ресурсы on-line системы http://www.wolframalpha.com, пакета Maple и т.п.).

Задание 1.

$$\begin{cases} 3x_1 - 2x_2 + 2x_3 - 2x_4 = 8, \\ 2x_1 - x_2 + 2x_3 = 4, \\ 2x_1 + x_2 + 4x_3 + 8x_4 = -1, \\ x_1 + 3x_2 - 6x_3 + 2x_4 = 3. \end{cases} \quad \begin{cases} 2x_1 + 3x_2 + x_3 + 2x_4 = 4, \\ 4x_1 + 3x_2 + x_3 + x_4 = 5, \\ x_1 - 7x_2 - x_3 - 2x_4 = 7, \\ 2x_1 + 5x_2 + x_3 + x_4 = 1. \end{cases} \quad \begin{cases} x_1 - x_2 + x_3 - x_4 = 0, \\ 4x_1 - x_2 - x_4 = 0, \\ 2x_1 + x_2 - 2x_3 + x_4 = 0, \\ 5x_1 + x_2 - 4x_4 = 0. \end{cases}$$

Задание 2.
Элементы матрицы A вычисляются по формулам:

$$A_{ij} = \begin{cases} q_M^{i+j} + 0.1 \cdot (j - i), & i \neq j, \\ (q_M - 1)^{i+j}, & i = j, \end{cases}$$

где $q_M = 1.001 - 2 \cdot M \cdot 10^{-3}$,
$i, j = 1, \ldots n$, M=4, n=100.

Элементы вектора f (вектор правой части системы) задаются формулами:

$$b_i = n \cdot \exp\left(\frac{x}{i}\right) \cdot \cos(x)$$

.

Метод решения

Рассмотрим СЛАУ вида $A\mathbf{x} = \mathbf{f}$. Запишем линейный одношаговый алгоритм решения СЛАУ итерационным методом в стандартной канонической форме:

$$B_{k+1}\frac{\mathbf{x}_{k+1}-\mathbf{x}_k}{\tau_{k+1}} + A\mathbf{x}_k = \mathbf{f},\ \det B_{k+1} \neq 0,\ \tau_{k+1} > 0 \quad (1)$$

Где последовательность матриц B_{k+1} и числовых параметров τ_{k+1} являются итерационными параметрами.

Пусть матрица A имеет вид

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mm} \end{bmatrix}$$

Разложим матрицу A на сумму трех матриц

$$A=D+T_H+T_B,$$

где D - диагональная часть матрицы A, которая содержит элементы a_{ii}, стоящие на главной диагонали:

$$D_{ij} = a_{ij}\delta_{ij} = \begin{cases} 0, & i \neq j \\ a_{ii}, & i = j \end{cases},$$

T_H - нижняя треугольная матрица

$$\left(T_H\right)_{ij} = \begin{cases} a_{ij}, & i > j \\ 0, & i \leq j \end{cases},$$

T_B - верхняя треугольная матрица.

$$\left(T_B\right)_{ij} = \begin{cases} 0, & i \geq j \\ a_{ij}, & i < j \end{cases}.$$

Введем параметр ω и запишем рекуррентное соотношение (1) в виде

$$\left(D + \omega T_H\right)\frac{\left(\mathbf{x}_{k+1}-\mathbf{x}_k\right)}{\omega} + A\mathbf{x}_k = \mathbf{f} \quad (2)$$

Где

$$B = D + \omega T_H,\ \tau = \omega > 0$$

Соотношению (2) можно придать вид

$$\left(\frac{1}{\omega}D + T_H\right)\left(\mathbf{x}_{k+1}-\mathbf{x}_k\right) + A\mathbf{x}_k = \mathbf{f}. \quad (3)$$

Для построения алгоритма вычисления очередной итерации нужно разделить в левой части рекуррентной формулы (3) члены, содержащие $\mathbf{x}_k$ и $\mathbf{x}_{k+1}$:

$$\left(\frac{1}{\omega}D + T_H\right)\mathbf{x}_{k+1} + \left[\left(1-\frac{1}{\omega}\right)D + T_B\right]\mathbf{x}_k = \mathbf{f}.$$

Если перейти от векторной записи к записи в виде отдельных уравнений, то можно получить для компонент x_i^{k+1} очередной итерации следующие формулы:

$$x_i^{k+1} = x_i^k + \frac{\omega}{a_{ii}}\left(f_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i}^{n} a_{ij} x_j^k \right),\ i = 1,\ldots,n\,.$$

Условие окончания итерационного процесса при достижении точности ε в упрощённой форме имеет вид:

$$\| x^{(k+1)} - x^{(k)} \| \leq \varepsilon$$

Более точное условие окончания итерационного процесса имеет вид

$$\| Ax^{(k)} - b \| \leq \varepsilon$$

и требует больше вычислений.

Описание программы

Метод верхней релаксации:

```
def relaxationMethod(sle, w, eps):
    '''
    Принимает СЛАУ, итерационный параметр и точность
    Возвращает решение системы и колличество итераций
    '''
    done = False
    iteration = 0
    x = [0 for i in range(sle.N)]
    xnext = [0 for i in range(sle.N)]
    while (not done):
        norm = 0
        x = [a for a in xnext]
        xnext = [0 for i in range(sle.N)]
        for i in range(sle.N):
            delta = sle.B[i]
            for j in range(0, i - 1):
                delta -= sle.A[i][j] * xnext[j]
            for j in range(i - 1, sle.N):
                delta -= sle.A[i][j] * x[j]
            delta *= w / sle.A[i][i]
            try:  # Рассчитываем норму и выходим при переполнении
                norm += delta ** 2
            except:
                return None, -1
            xnext[i] = x[i] + delta
        norm **= (1/2)
        done = norm < eps
        iteration += 1
    return xnext, iteration
```

Классы СЛАУ и датасета использовались из подварианта 1.

Тестирование

Достаточным условием сходимости метода верхней релаксации является положительная определенность и симметричность матрицы A. Ни одна из матриц в задании не удовлетворяет этим условиям, метод верхней релаксации не позволяет найти решение для них. Для проведения анализа изучаемого метода рассмотрим поведение алгоритма на указанных ниже СЛАУ:

Проанализируем скорость сходимости алгоритма для различных параметров:

СЛАУ	**Итерационный параметр**	**Точность**	**Кол-во итераций**	**Решение СЛАУ (WolframAlpha)**
6 -2 2 -5 2 3 2 7 3 -2 3 15	0.10	0.01	182	(-12.2692, -0.6923, 16.8077)
	0.50		44	
	1.00		**17**	
	1.70		529	
	0.10	0.0001	405	
	0.50		79	
	1.00		**29**	
	1.70		877	
16 -2 2 -5 20 5 20 -112 3 -2 11 15	0.10	0.01	40	(-1.8167, -13.0667, -0.5167)
	0.50		**21**	
	0.80		45	
	0.90		141	
	0.10	0.0001	90	
	0.50		**32**	
	0.80		72	
	0.90		232	

Выводы

В ходе выполнении поставленной задачи был реализован метод верней релаксации. Сходимость данного метода зависит от исходных данных. Тестирование показало, что скорость сходимости метода зависит от подбора итерационного параметра. Подбор оптимального итерационного параметра зависит от решаемой СЛАУ. Так для описанных в предыдущем разделе тестах для первого примера оптимальное значение итерационного параметра близко к 1, а для второго к 0,5.