Министерство образования Республики Беларусь
Учреждение образования
“БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ”

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Методы численного анализа

ОТЧЁТ
к лабораторной работе 2
на тему

Численное решение систем линейных алгебраических уравнений (СЛАУ) методом простых итераций и методом Зейделя

Выполнил: студент группы 053502
Герчик Артём Вадимович

Проверил: Анисимов Владимир Яковлевич

Минск 2022

Оглавление

Цели выполнения задания 3
Краткие теоретические сведения 4
Задание 8
Программная реализация 9
Полученные результаты 15
Выводы 18

Цели выполнения задания

1) Изучить итерационные методы решения СЛАУ (метод простых итераций, метод Зейделя).
2) Составить алгоритм решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ.
3) Составить программу решения СЛАУ по разработанному алгоритму.
4) Численно решить тестовые примеры и проверить правильность работы программы. Сравнить трудоемкость решения методом простых итераций и методом Зейделя.

Краткие теоретические сведения

Прямые методы применяют главным образом для решения задач малой размерности, когда нет ограничений в доступной оперативной памяти ЭВМ или необходимости выполнения чрезмерно большого числа арифметических операций. Большие системы уравнений, возникающие в основном в приложениях, как правило, являются разреженными. Методы исключения для систем с разреженным и матрицами неудобны, например, тем, что при их использовании большое число нулевых элементов превращается в ненулевые, и матрица теряет свойство разреженности. В противоположность им при использовании итерационных методов в ходе итерационного процесса матрица не меняется, и она, естественно, остается разреженной. Большая эффективность итерационных методов по сравнению с прямыми методами тесно связанна с возможностью существенного использования разреженности матриц.

Итерационные методы основаны на построении сходящейся к точному решению x рекуррентной последовательности.

Для решения СЛАУ методом простых итераций преобразуем систему от первоначальной формы Ax = b или

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2, \\ &\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned} \qquad (2.1)$$

к виду

$$x = Bx + c \qquad (2.2)$$

Здесь B – квадратная матрица с элементами b_{ij} (i, j = 1, 2, ..., n), c – вектор-столбец с элементами c_i (i = 1, 2, ..., n). В развернутой форме записи система (2.2) имеет следующий вид:

$$\begin{aligned} x_1 &= b_{11}x_1 + b_{12}x_2 + \ldots + b_{1n}x_n + c_1, \\ x_2 &= b_{21}x_1 + b_{22}x_2 + \ldots + b_{2n}x_n + c_2, \\ &\ldots \\ x_n &= b_{n1}x_1 + b_{n2}x_2 + \ldots + b_{nn}x_n + c_n \end{aligned}$$

Вообще говоря, операция приведения системы к виду, удобному для итераций, не является простой и требует специальных знаний, а также существенного использования специфики системы.

Можно, например, преобразовать систему (2.1) следующим образом

$$x_1 = (b_1 - a_{11}x_1 - a_{12}x_2 - \ldots - a_{1n}x_n)/a_{11} + x_1$$

,

$$x_2 = (b_2 - a_{21}x_1 - a_{22}x_2 - \ldots - a_{2n}x_n)/a_{22} + x_2$$

,

если диагональные элементы матрицы A отличны от нуля.

Можно преобразовать систему (2.1) в эквивалентную ей систему

$$x = (\boldsymbol{E} - \boldsymbol{A})\boldsymbol{x} + \boldsymbol{b}$$

Задав произвольным образом столбец начальных приближений $x^0 = (x_1^0, x_2^0, ..., x_n^0)^T$, подставим их в правые части системы (2.2) и вычислим новые приближения $x^1 = (x_1^1, x_2^1, ..., x_n^1)^T$, которые опять подставим в систему (2.2) и т.д. Таким образом, организуется итерационный процесс

$\boldsymbol{x^k} = \boldsymbol{B}\boldsymbol{x^{k-1}} + \boldsymbol{c}$, k = 1, 2, …

Известно, что система (2.1) имеет единственное решение x* и последовательность $\{x^k\}$ сходится к этому решению со скоростью геометрической прогрессии, если $\| B \| < 1$ в любой матричной норме.

Т.е. для того, чтобы последовательность простых итераций сходилась к единственному решению достаточно, чтобы выполнялось одно из следующих условий:

1) $\max\limits_{1 \le i \le n} \left(\sum\limits_{j=1}^{n} |b_{ij}| \right) < 1$

2) $\sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} b_{ij}^{\,2} < 1$

3) $\max\limits_{1 \le j \le n} \left(\sum\limits_{i=1}^{n} |b_{ij}| \right) < 1$

Метод Зейделя является модификацией метода простых итераций. Суть его состоит в том, что при вычислении следующего $\boldsymbol{x_i^k}$ в формуле $\boldsymbol{x^k} = \boldsymbol{B}\boldsymbol{x^{k-1}} + \boldsymbol{c}$, k = 1, 2, … вместо $x_1^{k-1}, x_2^{k-1}, \ldots, x_{i-1}^{k-1}$ используются уже вычисленные $x_1^k, x_2^k, \ldots, x_{i-1}^k$, т.е.

$$x_i^k = \sum_{j=1}^{i-1} g_{ij}x_j^k + \sum_{j=i+1}^{n} g_{ij}x_j^{k-1} + c_i \qquad (2.3)$$

Такое усовершенствование позволяет ускорить сходимость итерации почти в два раза. Кроме того, данный метод может быть реализован на ЭВМ без привлечения дополнительного массива, т.к. полученное новое x_i^k сразу

засылается на место старого.

Схема алгоритма аналогична схеме метода простых итераций. Самый простой способ приведения системы к виду, удобному для итераций, состоит в следующем. Из первого уравнения системы (2.1) выразим неизвестное x_1:

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3 - \ldots - a_{1n}x_n)/a_{11}$$

из второго уравнения выразим неизвестное x_2:

$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3 - \ldots - a_{2n}x_n)/a_{22}$$

и т.д. В результате получим систему

$$x_1 = b_{12}x_2 + b_{13}x_3 + \ldots + b_{1n}x_n + c_1$$

$$x_2 = b_{21}x_1 + b_{23}x_3 + \ldots + b_{2n}x_n + c_2$$

в которой на главной диагонали матрицы B находятся нули, а остальные элементы выражаются по формулам

$$b_{ij} = -a_{ij}/a_{ii},\ c_i = b_i/a_{ii},\ (i, j = 1,2,\ldots,n,\ i \neq j)$$

Конечно, для возможности выполнения указанного преобразования необходимо, чтобы диагональные элементы матрицы A были ненулевыми.

Введем нижнюю B_1 (получается из B заменой нулями элементов, стоявших на главной диагонали и выше ее) и верхнюю B_2 (получается из B заменой нулями элементов, стоявших на главной диагонали и ниже ее) треугольные матрицы.

Заметим, что $B = B_1 + B_2$ и поэтому решение x исходной системы удовлетворяет равенству

$$\boldsymbol{x} = \boldsymbol{B}_1\boldsymbol{x} + \boldsymbol{B}_2\boldsymbol{x} + \boldsymbol{c} \quad (2.5)$$

Выберем начальное приближение $x^{(0)} = [x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}]^T$. Подставляя его в правую часть равенства, находим первое приближение

$$\boldsymbol{x}^{(1)} = \boldsymbol{B}_1\boldsymbol{x}^{(1)} + \boldsymbol{B}_2\boldsymbol{x}^{(0)} + \boldsymbol{c} \quad (2.6)$$

Подставляя приближение $x^{(1)}$, получим

$$\boldsymbol{x}^{(2)} = \boldsymbol{B}_1\boldsymbol{x}^{(2)} + \boldsymbol{B}_2\boldsymbol{x}^{(1)} + \boldsymbol{c} \quad (2.7)$$

Продолжая этот процесс далее, получим последовательность x^0), x^, ., x^n), ... приближений к вычисляемых по формуле

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{B}_1\boldsymbol{x}^{(k+1)} + \boldsymbol{B}_2\boldsymbol{x}^{(k)} + \boldsymbol{c} \quad (2.8)$$

Или же

$$x_i^{(k+1)} = \sum_{j=1}^{i-1} b_{ij} x_j^{(k+1)} + \sum_{j=i+1}^{n} b_{ij} x_j^{(k)} + c_i$$

Объединив приведение системы к виду, удобному для итераций и метод Зейделя в одну формулу, получим

$$x_i^{(k+1)} = x_i^{(k)} - (\sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} + \sum_{j=i}^{n} a_{ij} x_j^{(k)} - b_i)/a_{ii} \quad (2.9)$$

Тогда достаточным условием сходимости метода Зейделя будет условие доминирования диагональных элементов в строках или столбцах матрицы A, т.е.

$a_{ii} > a_{i1} + \ldots + a_{in}$ *для всех* $i = 1, 2, \ldots, n$
или
$a_{jj} > a_{1j} + \ldots + a_{nj}$ *для всех* $j = 1, 2, \ldots, n$

Методы простой итерации и Зейделя сходятся примерно так же, как геометрическая прогрессия со знаменателем || B ||.

Задание

Методом простых итераций и методом Зейделя найти с точностью 0,0001 численное решение системы $\boldsymbol{Ax=b}$, где A = kC +D, A - исходная матрица для расчета, k - номер варианта (7), матрицы C, D и вектор свободных членов b задаются ниже.

$$C = \begin{bmatrix} 0{,}01 & 0 & -0{,}02 & 0 & 0 \\ 0{,}01 & 0{,}01 & -0{,}02 & 0 & 0 \\ 0 & 0{,}01 & 0{,}01 & 0 & -0{,}02 \\ 0 & 0 & 0{,}01 & 0{,}01 & 0 \\ 0 & 0 & 0 & 0{,}01 & 0{,}01 \end{bmatrix}, \quad D = \begin{bmatrix} 1{,}33 & 0{,}21 & 0{,}17 & 0{,}12 & -0{,}13 \\ -0{,}13 & -1{,}33 & 0{,}11 & 0{,}17 & 0{,}12 \\ 0{,}12 & -0{,}13 & -1{,}33 & 0{,}11 & 0{,}17 \\ 0{,}17 & 0{,}12 & -0{,}13 & -1{,}33 & 0{,}11 \\ 0{,}11 & 0{,}67 & 0{,}12 & -0{,}13 & -1{,}33 \end{bmatrix}.$$

Вектор $\boldsymbol{b}= (1{,}2;\ 2{,}2;\ 4{,}0;\ 0{,}0;\ -1{,}2)^{T}$.

Программная реализация

```
import numpy

epsilon = 10.0 ** -4


def initial():
    numpy.set_printoptions(suppress=True,
precision=4, floatmode="fixed")
    b = numpy.array([[1.2], [2.2], [4.0], [0.0],
[-1.2]])
    C = numpy.array([
        [0.01, 0.0, -0.02, 0.0, 0.0],
        [0.01, 0.01, -0.02, 0.0, 0.0],
        [0.0, 0.01, 0.01, 0.0, -0.02],
        [0.0, 0.0, 0.01, 0.01, 0.0],
        [0.0, 0.0, 0.0, 0.01, 0.01]
    ])
    D = numpy.array([
        [1.33, 0.21, 0.17, 0.12, -0.13],
        [-0.13, -1.33, 0.11, 0.17, 0.12],
        [0.12, -0.13, -1.33, 0.11, 0.17],
        [0.17, 0.12, -0.13, -1.33, 0.11],
        [0.11, 0.67, 0.12, -0.13, -1.33]
    ])
    A = 7 * C + D

    # # test example 1
    # A = numpy.array([
    #     [2.0, 1.0],
    #     [1.0, -2.0]
    # ])
    # b = numpy.array([[3.0], [1.0]])
    # # test example 1
```

```
    # # test example 2 with error
    # A = numpy.array([
    #     [-1.0, 0.5, 0.6],
    #     [0.0, -1.0, 0.5],
    #     [0.5, 0.0, -1.0]
    # ])
    # b = numpy.array([[1.0], [1.0], [1.0]])
    # # test example 2 with error

    return A, b


def output_initial(a_to_work, b_to_work):
    print(f'Входные данные:\n\nA:\n {a_to_work}')
    print(f'b:\n {b_to_work}\n\n')


def element_is_zero():
    print("Element == 0!")
    quit()


def get_matrix_b(a_to_work):
    n = a_to_work.shape[0]
    alpha = numpy.zeros((n, n))
    for i in range(n):
        for j in range(n):
            alpha[i, j] = - a_to_work[i, j] /
a_to_work[i, i]
        alpha[i, i] = 0
    return alpha


def get_all_norms(a_to_work):
```

```
    n = a_to_work.shape[0]
    first_norm =
max(numpy.absolute(a_to_work[i]).sum() for i in
range(n))
    second_norm =
max(numpy.absolute(a_to_work.T[j]).sum() for j in
range(n))
    third_norm = ((a_to_work ** 2).sum()) ** (1 / 2)
    return first_norm, second_norm, third_norm


def method_of_simple_iterations(a_to_work,
b_to_work):
    n = a_to_work.shape[0]

    for i in range(n):
        if a_to_work[i, i] == 0.0:
            element_is_zero()

    matrix_b = get_matrix_b(a_to_work)
    if not (min(get_all_norms(matrix_b)) < 1):
        print("Одно из достаточных условий сходимости
не выполняется!")
        # quit(0)

    c_column = numpy.zeros((n, 1))
    for i in range(n):
        c_column[i] = b_to_work[i] / a_to_work[i, i]

    print(f'B:\n {matrix_b}')

    current_iteration_x = numpy.zeros((n, 1))

    count_of_iterations = 0
    delta_one = epsilon
```

```
    delta_second = epsilon

    while delta_one + delta_second > epsilon:
        previous_iteration_x =
current_iteration_x.copy()
        current_iteration_x = c_column +
matrix_b.dot(current_iteration_x)  # .dot - скалярное
произведение

        delta_one =
numpy.absolute((current_iteration_x -
previous_iteration_x)).max()
        delta_second =
numpy.absolute((a_to_work.dot(current_iteration_x) -
b_to_work)).max()

        count_of_iterations += 1

    print(f"\nКоличество итераций, при расчете
методом простых итераций: {count_of_iterations}")
    print(f'x: {current_iteration_x.T}\n\n')


def
check_is_sufficient_condition_is_satisfied(a_to_work)
:  # ???
    n = a_to_work.shape[0]

    sum_of_row = 0.0
    for i in range(n):
        for j in range(n):
            if i != j:
                sum_of_row += a_to_work[i][j]
        if abs(a_to_work[i][i]) < sum_of_row:
```

```
            print('Достаточное условие сходимости
метода Зейделя не выполняется!')
            quit(0)
        else:
            sum_of_row = 0


def method_of_seidel(a_to_work, b_to_work):
    n = a_to_work.shape[0]

    #
check_is_sufficient_condition_is_satisfied(a_to_work.
copy())  # ???

    for i in range(n):
        if a_to_work[i, i] == 0.0:
            element_is_zero()

    matrix_b = get_matrix_b(a_to_work)

    print(f'B:\n {matrix_b}')

    if not (min(get_all_norms(matrix_b)[:2]) < 1):
        print("Одно из достаточных условий сходимости
не выполняется!")
        # quit(0)

    current_iteration_x = numpy.zeros((n, 1))
    count_of_iterations = 0
    delta_one = epsilon
    delta_second = epsilon

    while delta_one + delta_second > epsilon:
        previous_iteration_x =
current_iteration_x.copy()
```

```
        for i in range(n):
            s = 0
            for j in range(n):
                s += a_to_work[i, j] *
current_iteration_x[j]
            s -= b_to_work[i]
            current_iteration_x[i] =
current_iteration_x[i] - s / a_to_work[i, i]

        delta_one =
numpy.absolute((current_iteration_x -
previous_iteration_x)).max()
        delta_second =
numpy.absolute((a_to_work.dot(current_iteration_x) -
b_to_work)).max()

        count_of_iterations += 1

    print(f"\nКоличество итераций, при расчете
методом Зейделя: {count_of_iterations}")
    print(f'x: {current_iteration_x.T}\n\n')


def main():
    a_to_work, b_to_work = initial()
    output_initial(a_to_work.copy(),
b_to_work.copy())
    method_of_simple_iterations(a_to_work.copy(),
b_to_work.copy())
    method_of_seidel(a_to_work.copy(),
b_to_work.copy())


if __name__ == '__main__':
    main()
```

Полученные результаты

Тестовый пример 1

Методом простых итераций и методом Зейделя найти с точностью 0,0001 численное решение СЛАУ:

$$\begin{cases} 2x_1 + x_2 = 3 \\ x_1 - 2x_2 = 1 \end{cases}$$

Ответ:

Метод простых итераций	Метод Зейделя
$x_1 = 1.400$ $x_2 = 0.200$	$x_1 = 1.400$ $x_2 = 0.200$
Количество итераций	
16	8

Тестовый пример 2

Методом простых итераций и методом Зейделя найти с точностью 0,0001 численное решение СЛАУ:

$$\begin{cases} -x_1 + 0.5*x_2 + 0.6*x_3 = 1 \\ -x_2 + 0.5*x_3 = 1 \\ 0.5*x_1 - x_3 = 1 \end{cases}$$

Ответ:

Метод простых итераций	Метод Зейделя
Предупреждение о невыполнении достаточных условий сходимости	
$x_1 = -4.0869$ $x_2 = -2.5217$ $x_3 = -3.0434$	$x_1 = -4.0870$ $x_2 = -2.5217$ $x_3 = -3.0434$
Количество итераций	
29	18

ЗАДАНИЕ

Вариант 7

Методом простых итераций и методом Зейделя найти с точностью 0,0001 численное решение системы ***Ax=b***:

$$\begin{cases} 1.40x_1 + 0.21x_2 + 0.03x_3 + 0.12x_4 - 0.13x_5 = 1.2 \\ -0.06x_1 - 1.26x_2 - 0.03x_3 + 0.17x_4 + 0.12x_5 = 2.2 \\ 0.12x_1 - 0.06x_2 - 1.26x_3 + 0.11x_4 + 0.03x_5 = 4.0 \\ 0.17x_1 + 0.12x_2 - 0.06x_3 - 1.26x_4 + 0.11x_5 = 0 \\ 0.11x_1 + 0.67x_2 + 0.12x_3 - 0.06x_4 - 1.26x_5 = -1.2 \end{cases}$$

Ответ:

Метод простых итераций	Метод Зейделя
$x_1 = 1.1556$ $x_2 = -1.7289$ $x_3 = -2.9755$ $x_4 = 0.1194$ $x_5 = -0.1551$	$x_1 = 1.1556$ $x_2 = -1.7289$ $x_3 = -2.9755$ $x_4 = 0.1194$ $x_5 = -0.1551$
Количество итераций	
10	6

Консоль:

```
/Users/lnxd/Desktop/MCHA/LAB2/venv/bin/python /Users/lnxd/Desktop/MCHA/LAB2/main.py
Входные данные:

A:
 [[ 1.4000  0.2100  0.0300  0.1200 -0.1300]
 [-0.0600 -1.2600 -0.0300  0.1700  0.1200]
 [ 0.1200 -0.0600 -1.2600  0.1100  0.0300]
 [ 0.1700  0.1200 -0.0600 -1.2600  0.1100]
 [ 0.1100  0.6700  0.1200 -0.0600 -1.2600]]
```

```
b:
 [[ 1.2000]
 [ 2.2000]
 [ 4.0000]
 [ 0.0000]
 [-1.2000]]


B:
 [[ 0.0000 -0.1500 -0.0214 -0.0857  0.0929]
 [-0.0476  0.0000 -0.0238  0.1349  0.0952]
 [ 0.0952 -0.0476  0.0000  0.0873  0.0238]
 [ 0.1349  0.0952 -0.0476  0.0000  0.0873]
 [ 0.0873  0.5317  0.0952 -0.0476  0.0000]]
```

Количество итераций, при расчете методом простых итераций: 10

```
x: [[ 1.1556 -1.7289 -2.9755  0.1194 -0.1551]]


B:
 [[ 0.0000 -0.1500 -0.0214 -0.0857  0.0929]
 [-0.0476  0.0000 -0.0238  0.1349  0.0952]
 [ 0.0952 -0.0476  0.0000  0.0873  0.0238]
 [ 0.1349  0.0952 -0.0476  0.0000  0.0873]
 [ 0.0873  0.5317  0.0952 -0.0476  0.0000]]
```

Количество итераций, при расчете методом Зейделя: 6

```
x: [[ 1.1556 -1.7289 -2.9755  0.1194 -0.1551]]


Process finished with exit code 0
```

Выводы

Таким образом, в ходе выполнения лабораторной работы мы изучили итерационные методы решения СЛАУ (метод простых итераций, метод Зейделя), составили алгоритм решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ, составили программу решения СЛАУ по разработанному алгоритму, численно решили тестовые примеры и проверили правильность работы программы, сравнили трудоемкость решения методом простых итераций и методом Зейделя. Результаты показали, что метод Зейделя справляется с поставленной задачей быстрее. По трудоемкости метод Зейделя немного уступает методу простых итераций, поскольку требует дополнительной проверки на достаточное условие сходимости, т. к. выполнение условия $||B|| < 1$ не гарантирует сходимость метода Зейделя.