Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина «Методы численного анализа»

ОТЧЕТ

к лабораторной работе №2

на тему:

«ЧИСЛЕННОЕ РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ (СЛАУ) МЕТОДОМ ПРОСТЫХ ИТЕРАЦИЙ И МЕТОДОМ ЗЕЙДЕЛЯ»

БГУИР 1-40 04 01

Выполнил студент группы 253505
БЕКАРЕВ Станислав Сергеевич

(дата, подпись студента)

Проверил доцент кафедры информатики
АНИСИМОВ Владимир Яковлевич

(дата, подпись преподавателя)

Минск 2023

Содержание

Цель работы

- изучить метод простых итераций и метод Зейделя, получить численное решение заданной СЛАУ;
- составить программу решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ;
- выполнить тестовые примеры и проверить правильность работы программы

Задание:

Методом простых итераций и методом Зейделя найти с точностью 0,0001 численное решение системы **Ax=b**, где **A = kC + D**, **A** - исходная матрица для расчёта, **k** - номер варианта (0–15), матрицы **C, D** и вектор свободных членов **b** задаются ниже.

Исходные данные:

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

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

Вариант 3

Программная реализация

Для проверки решения умножим исходную матрицу на полученный вектор решений и сравним с изначальным вектором свободных членов.

Исходные данные:

Матрица А, полученная в результате вычисления А = 3С + D:

```
Исходная матрица:
[ 1.36  0.21  0.11  0.12 -0.13] = 1.2
[-0.1  -1.3   0.05  0.17  0.12] = 2.2
[ 0.12 -0.1  -1.3   0.11  0.23] = 4.0
[ 0.17  0.12 -0.1  -1.3   0.11] = 0.0
[ 0.11  0.67  0.12 -0.1  -1.3 ] = -1.2
```

Код простых итераций:

```
def iteration_method():
    global X, X_next
    X = numpy.zeros(len(A))
    X_next = numpy.zeros(len(A))
    check = True
    amount = 0
    while check:
        for i in range(0, len(A)):
            summ = 0.
            summ += b[i]
            for j in range(0, len(A)):
                if i != j:
                    summ -= A[i][j] * X[j]
            X_next[i] = summ / A[i][i]
        eps = X_next - X
        for i in range(0, len(eps)):
            eps[i] = abs(eps[i])
        max_eps = max(eps)
        if max_eps < 10 ** (-4):
            check = False
        X = X_next.copy()
        amount += 1
    print("Кол-во итераций: ", amount)
```

Код метода Зейделя:

```
def zeidel_method():
    global X, X_next
    X = numpy.zeros(len(A))
    check = True
    amount = 0
    while check:
        X_next = X.copy()
        for i in range(0, len(A)):
            summ = 0.
            summ += b[i]
            for j in range(0, len(A)):
                if i != j:
                    summ -= A[i][j] * X_next[j]
            X_next[i] = summ / A[i][i]
        eps = X_next - X
        for i in range(0, len(eps)):
            eps[i] = abs(eps[i])
        max_eps = max(eps)
        if max_eps < 10 ** (-4):
            check = False
        X = X_next.copy()
        amount += 1
    print("Кол-во итераций: ", amount)
```

Код проверок для возможности применения данных методов:

```
def all_check():
    return check_on_row() or check_on_column() or check_norm()


def check_on_row():
    matrix = A.copy()
    for i in range(0, len(matrix)):
        summ = 0.
        for j in range(0, len(matrix)):
            if i != j:
                summ += abs(matrix[i][j])
        if summ > abs(matrix[i][i]):
            print(f"Сумма модулей по строке {i} ({summ}) больше модуля 
диагонального элемента {A[i][i]}")
            return False
    return True


def check_on_column():
    matrix = A.transpose()
    for i in range(0, len(matrix)):
        summ = 0.
        for j in range(0, len(matrix)):
            if i != j:
                summ += abs(matrix[i][j])
        if summ > abs(matrix[i][i]):
            print(f"Сумма модулей по столбцу {i} ({summ}) больше 
модуля диагонального элемента {A[i][i]}")
            return False
    return True


def check_norm():
    matrix = A.copy()
    for i in range(0, len(matrix)):
        summ = 0.
        for j in range(0, len(matrix)):
            if i != j:
                summ += (matrix[i][j]/matrix[i][i]) ** 2
        if summ > 1:
            print(f"||B|| больше 1")
            return False
    return True
```

Полученные результаты

```
Исходная матрица:
[ 1.36  0.21  0.11  0.12 -0.13] = 1.2
[-0.1  -1.3   0.05  0.17  0.12] = 2.2
[ 0.12 -0.1  -1.3   0.11  0.23] = 4.0
[ 0.17  0.12 -0.1  -1.3   0.11] = 0.0
[ 0.11  0.67  0.12 -0.1  -1.3 ] = -1.2
Решение методом простых итераций:
Кол-во итераций:  9
Вектор решений:
[ 1.36565272 -1.89960952 -2.82598094  0.20228337 -0.21676252]
Получившийся вектор свободных значений:
1.19996493 2.20000473 4.00001030 -0.00000635 -1.20007135
Отклонение от изначального вектора свободных членов
[-3.50656580e-05  4.72611968e-06  1.02968741e-05 -6.34801997e-06
 -7.13492944e-05]
Решение методом Зейделя:
Кол-во итераций:  5
Вектор решений:
[ 1.36567931 -1.89961588 -2.8259832   0.20227558 -0.21681805]
Получившийся вектор свободных значений:
1.20000579 2.20000224 4.00000343 0.00000165 -1.20000000
Отклонение от изначального вектора свободных членов
[ 5.78803193e-06  2.24020312e-06  3.42692938e-06  1.65108390e-06
 -2.22044605e-16]
```

Результаты вычислений (вектор решений):

Метод простых итераций	*Метод Зейделя*	*Кол-во итераций*
1.36565272	1.36567931	Метод простых итераций: 9
-1.89960952	-1.89961588	Метод Зейделя: 5
-2.82598094	-2.82598320	
0.20228337	0.20227558	
-0.21676252	-0.21681805	

Результаты вычислений (получившийся вектор свободных членов):

Метод простых итераций	*Метод Зейделя*
1.19996493	1.20000579
2.20000473	2.20000224
4.00001030	4.00000343
-0.00000635	0.00000165
-1.20007135	-1.20000000

Тестовый пример 1.

С помощью пакета numpy создадим матрицу и вектор свободных членов и заполним их случайными числами:

```
Исходная матрица:
[8.33060289 2.89682981 0.40123842 2.60653485 1.58848179] = 1.99662355
[2.23101953 8.45617941 0.34926343 1.52959562 0.58708367] = 4.69119701
[0.276433   1.19903174 5.54678509 1.27876262 0.39383994] = 4.52123997
[2.21418937 1.10934538 2.37985173 7.72672466 1.94227667] = 4.84621620
[2.66539241 2.92417303 1.31808922 1.65837782 5.28908716] = 4.76331142
Решение методом простых итераций:
Кол-во итераций:  80
Вектор решений:
[-0.13575644  0.48049017  0.61825529  0.29156537  0.45778314]
Получившийся вектор свободных значений:
1.99618849 4.69090388 4.52106078 4.84578185 4.76288830
Отклонение от изначального вектора свободных членов
[-0.00043506 -0.00029314 -0.00017919 -0.00043435 -0.00042313]
Решение методом Зейделя:
Кол-во итераций:  8
Вектор решений:
[-0.13572219  0.48051233  0.61827963  0.29160075  0.45781647]
Получившийся вектор свободных значений:
1.99669297 4.69124992 4.52129021 4.84627828 4.76331143
Отклонение от изначального вектора свободных членов
[ 6.94167244e-05  5.29060881e-05  5.02281960e-05  6.20787742e-05
 -8.88178420e-16]
```

Тестовый пример 2.

В данном примере мы видим матрицу, в которой диагональ не является преимущественной.

```
Исходная матрица:
[1. 2. 3.] = 1.0
[2. 1. 3.] = 2.0
[2. 3. 1.] = 3.0
Сумма модулей по строке 0 (5.0) больше модуля диагонального элемента 1.0
Сумма модулей по столбцу 0 (4.0) больше модуля диагонального элемента 1.0
||B|| больше 1
Нельзя решить методом Зейдаля или простых итераций
```

Тестовый пример 3.

Так как проверки являются достаточным, но не необходимыми, то может существовать матрица с не преимущественной диагональю, решаемая с помощью метода простых итераций или метода Зейделя.
Вывод с отключенными проверками:

```
Исходная матрица:
[3. 0. 2.] = 1.0
[2. 3. 2.] = 2.0
[2. 0. 3.] = 3.0
Решение методом простых итераций:
Кол-во итераций:  25
Вектор решений:
[-0.59994456  0.13336502  1.39997624]
Получившийся вектор свободных значений:
1.00011881 2.00015841 3.00003960
Отклонение от изначального вектора свободных членов
[1.18806384e-04 1.58408512e-04 3.96021280e-05]
Решение методом Зейделя:
Кол-во итераций:  13
Вектор решений:
[-0.59994456  0.13335181  1.39996304]
Получившийся вектор свободных значений:
1.00009240 2.00009240 3.00000000
Отклонение от изначального вектора свободных членов
[9.24049654e-05 9.24049654e-05 0.00000000e+00]
```

:

Обычный вывод программы:

```
Исходная матрица:
[3. 0. 2.] = 1.0
[2. 3. 2.] = 2.0
[2. 0. 3.] = 3.0
Сумма модулей по строке 1 (4.0) больше модуля диагонального элемента 3.0
Сумма модулей по столбцу 0 (4.0) больше модуля диагонального элемента 3.0
||B|| больше 1
Нельзя решить методом Зейдаля или простых итераций
```

Вывод

В ходе выполнения лабораторной работы я изучил метод простых итераций и метод Зейделя, написал программу их реализации на языке Python для решения СЛАУ, правильность работы программы проверил на тестовых примерах.

На основании тестов можно сделать следующие выводы:

- Программа позволяет получить решения системы с заданной точностью (заданная точность в условиях лабораторной работы 10^-4);
- Метод Зейделя эффективнее по сравнению с методом простых итераций, так как затрачивает меньшее число итераций;
- Имеет ограничение в использовании (главная диагональ должна быть преимущественной), однако существуют матрицы, которые имеют не преимущественную диагональ и решаются с помощью метода простых итераций или метода Зейделя.
- Метод Зейделя более точен, так как использует уже найденные значения вектора решения на данной итерации, в отличие метода простых итераций.