Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №5:

Решение систем линейных алгебраических уравнений методом простой итерации и методом Зейделя.

Студент ИУ9-61Б (Группа) Е.А. Матвеев (И.О. Фамилия)

Проверила А. Б. Домрачева (И.О. Фамилия)

2023 г.

Содержание

1 Цель

Целью данной работы является изучение двух методов решения систем линейных алгебраических уравнений (далее СЛАУ): метода простой итерации и метода Зейделя, а также сравнение этих методов по скорости сходимости и особенностям результатов.

2 Постановка задачи

Дано: Матрица $A \in \mathbb{R}^{n \times n}$, вектор $b \in \mathbb{R}^n$

Требуется: Найти вектор x такой, что $Ax = b$ с помощью двух методов: метода простой итерации и метода Зейделя, выявить разницу между методами.

Для достижения цели работы были поставлены следующие задачи:

1. Изучить теоретическую основу обоих методов.
2. Реализовать алгоритмы методов на языке программирования python3.
3. Рассчитать значение абсолютной и относительной ошибок, сравнить результаты и объяснить причины различий.

3 Теоретические сведения

И метод простой итерации и метод Зейделя являются итерационными методами и почти не различаются.

Метод простой итерации работает следующим образом. Изначальное уравнение $Ax = b$ приводится к виду $x = Fx + c$. За изначальное приближение x_0 берется какой-нибудь вектор. Подставляя его в правую часть получается вектор $x_1 = Fx_0 + c$. Процесс повторяется, пока $||x_{i+1} - x_i||$ не будет меньше заранее заданного ϵ.

Метод Зейделя работает по той же схеме, но вычисление x_{i+1} происходит поэлементно. Первый элемент вектора x_{i+1} - это первый элемент вектора $Fx_i + c$, второй элемент - это второй элемент вектора $Fx_i^1 + c$, где x_i^k - это вектор x_i с замененными первыми k элементами, которые уже вычислены.

Исходя из предположения о том, что на каждой итерации мы получаем более точное решение задачи, можно сделать вывод о том, что метод Зейделя должен сходиться быстрее, так как каждый раз вектор x_{i+1} вычисляется на основе более точного приближения, чем в методе простой итерации. Также, можно предположить, что значения старших коэффициентов вектора x_i в методе Зейделя будут, в общем случае, лучше приближены к аналитическому решению, чем младшие коэффициенты того же вектора.

Для определения условий сходимости, а также для получения априорной и апостериорной погрешности, вводится норма матрицы F: $||F|| = \max_{1 \le i \le n} \sum_{j=1}^{n} |f_{ij}|$.

Условие сходимости для обоих методов: $||F|| < 1$.

Абсолютная погрешность на i-ом шаге итерации высчитывается по формуле $\Delta = \frac{||F||}{1-||F||} ||x_i - x_{i-1}||$.

Относительная погрешность на i-ом шаге итерации высчитывается по формуле $\delta = \frac{\Delta}{||x_i||}$, где $||x_i|| = max_{1 \le j \le n} x_i^j$.

4 Реализация

На листинге 1 представлена реализация методов простой итерации и Зейделя решения СЛАУ.

```
import numpy as np
import pandas as pd

eps = 0.001
alpha, beta = 0.1*13, 0.1*13
N = 4
A = np.array([
    [10 + alpha, -1, 0.2, 2],
    [1, 12-alpha, -2, 0.1],
    [0.3, -4, 12-alpha, 1],
    [0.2, -0.3, -0.5, 8-alpha]
])
Adiag = np.diagonal(A)
B = np.array([1+beta, 2-beta, 3, 1])

F = np.copy(A)
for i in range(N):
    F[i,i] = 0
for i in range(N):
    for j in range(N):
        F[i,j] = F[i,j]/Adiag[i]
Fup = np.copy(F)
Fdown = np.copy(F)
for i in range(N):
    for j in range(N):
        if j > i:
            Fdown[i,j] = 0
        else:
            Fup[i,j] = 0
C = np.copy(B)
for i in range(N):
    C[i] = C[i]/Adiag[i]


def matrixMultByVector(A, x):
    return np.array([
        sum([ A[i,j]*x[j] for j in range(N) ]) for i in range(N)
    ])

def simpleNextIter(x):
    return C - matrixMultByVector(F, x)

def zeidelNextIter(x):
    res = np.zeros(N)
    for i in range(N):
        res[i] = C[i] - matrixMultByVector(Fdown, res)[i] - matrixMultByVector(Fup, x)[i]
    return res

def norm(x):
    return np.max(np.abs(x))

def countDelta(xOld, xNew):
    deltaBig = findFNorm()/(1-findFNorm())*norm(xOld-xNew)
    deltaSmall = deltaBig/norm(xNew)
    return deltaBig, deltaSmall

def checkStop(xOld, xNew):
    return norm(xOld-xNew) < eps
```

```python

def findFNorm():
    return max( np.sum(np.abs(F[i])) for i in range(N) )

def run(method):
    nextIter = simpleNextIter if method == 'simple' else zeidelNextIter
    df = pd.DataFrame({
        'DELTA': [],
        'delta': [],
    })
    xOld = C
    xNew = nextIter(xOld)
    i = 0
    DELTA, delta = countDelta(xOld = xOld, xNew = xNew)
    app = {
        'delta': delta,
        'DELTA': DELTA
    }
    df = df.append(app, ignore_index=True)

    while not checkStop(xOld,xNew):
        i += 1
        xNew, xOld = nextIter(xNew), xNew
        DELTA, delta = countDelta(xOld = xOld, xNew = xNew)
        app = {
            'delta': delta,
            'DELTA': DELTA
        }
        df = df.append(app, ignore_index=True)

    print(df)
    return xNew, delta

def showMethod(method):
    xNew, delta = run(method)
    check = matrixMultByVector(A, xNew)
    print()
    print("CHECK:", check)
    print("WANT TO GET", B)
    print("DIFFERENCE", check - B)
    print("NORM F", findFNorm())

showMethod('simple')
showMethod('zeidel')
```

Листинг 1: Реализация методов решения СЛАУ

5 Тестирование

Для тестирования была взята следующая СЛАУ:

$$A = \begin{matrix} 11.30 & -1.00 & 0.20 & 2.00 \\ 1.00 & 10.70 & -2.00 & 0.10 \\ 0.30 & -4.00 & 10.70 & 1.00 \\ 0.20 & -0.30 & -0.50 & 6.70 \end{matrix}$$

$$b = (2.3, 0.7, 3, 1)$$

Норма A = 0.4953271028037384.

5.1 Метод простой итерации

Для тестирования метода простой итерации выбран $\epsilon = 0.001$

На рисунке 1 представлены Δ и δ для каждой итерации отработки алгоритма.

	DELTA	delta
0	0.031396626467000568	0.11009621298156506
1	0.010810609187886264	0.03649901973675971
2	0.0020339736712574347	0.006845425448688896
3	0.0006812451847897599	0.002287416378897702

Рисунок 1 — Таблица вычислений методом простой итерации.

Получен результат: $x_3 = 0.17717355 \quad 0.10280586 \quad 0.29782298 \quad 0.17073283$

$|Ax_3 - b| = 0.00028559253 \quad 0.0013763858 \quad 0.00063256479 \quad 0.00040853341$.

5.2 Метод Зейделя

Для тестирования метода простой итерации выбран $\epsilon = 0.0001$

На рисунке 2 представлены Δ и δ для каждой итерации отработки алгоритма.

Получен результат: $x_3 = 0.17715646 \quad 0.10297131 \quad 0.29793517 \quad 0.17081010$

$|x_2 - b| = 00.0001038826492 \quad 0.0001601705369 \quad 2.192025484 \times 10^{-05} \quad 6.730274$

	DELTA	delta
0	0.03374388329096478	0.1129523335683491
1	0.0032724067957183587	0.010980628455552227
2	0.00028015035586208895	0.0009403064383075029

Рисунок 2 — Таблица вычислений методом простой итерации.

5.3 Сравнение

Как видно из результатов тестирования, метод Зейделя обеспечивает более быструю сходимость - за меньшее количество итераций можно получить значение с точность более, чем на порядок. Подтвердилась гипотеза о более точных значениях старших элементов вектора, полученного методом Зейделя: у вектора $|x_2 - b|$ старший элемент гораздо ближе к нулю, чем остальные.

6 Вывод

При выполнении лабораторной работы был изучен и реализован в программном коде два метода решения СЛАУ: метод простой итерации и метод Зейделя.

Из результатов тестирования следует превосходство метода Зейделя над методом простой итерации - сходимость быстрее, а за одинаковое количество итераций можно получить значение с меньшими погрешностями. При этом никаких ограничений, мотивирующих применять метод простой итерации вместо метода Зейделя не выявлено.