Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
Московский государственный технический университет имени Н.Э. Баумана

Лабораторная работа №7
по курсу «Численные методы»
«Решение системы линейных алгебраических уравнений методом простой итерации и методом Зейделя»

Выполнил:
студент группы ИУ9-62Б
Головкин Дмитрий

Проверила:
Домрачева А.Б.

Москва, 2023

Цель:

Анализ метода простой итерации и метода Зейделя для решения системы алгебраических уравнений.

Постановка задачи:

Дано: СЛАУ с квадратичной невырожденной матрицей $Ax = b$, где $x = (x_i), \quad b = (b_i)$ - столбцы, а $A = (a_{ij})$ - квадратная матрица порядка n; $i, j = \overline{1, n}$.

Найти: Вектор решений СЛАУ $x = (x_i)$.

Тестовый пример:

В качестве тестового примера были предложены следующие входные данные:

$$A_{4\times4} = \begin{pmatrix} 10.0+\alpha & -1.0 & 0.2 & 2.0 \\ 1.0 & 12.0-\alpha & -2.0 & 0.1 \\ 0.3 & -4.0 & 12.0-\alpha & 1.0 \\ 0.2 & -0.3 & -0.5 & 8.0-\alpha \end{pmatrix}$$

$$b_{1\times4} = \begin{pmatrix} 1.0+\beta \\ 2.0-\beta \\ 3.0 \\ 1.0 \end{pmatrix}$$

, где $\alpha = 0.1 * 4, \quad \beta = 0.1 * 4$.

Описание методов:

В начале необходимо привести исходную СЛАУ к виду $x = Fx + c$, где $x = (x_i), \quad c = (c_i)$ - столбцы, а $F = (f_{ij})$ - квадратная матрица порядка n; $i, j = \overline{1, n}$. Если для всех строк $|a_{ii}| > \sum\limits_{j\neq i} |a_{ij}, \quad i = \overline{1, n}$ (диагональные элементы превалируют) , то для приведения к требуемому виду достаточно разрешить каждое уравнение системы относительно ведущей неизвестной. Тогда

$$f_{ij}() = \begin{cases} 0 & \text{, i=j} \\ -\frac{a_{ij}}{a_{ii}} & \text{, } i \neq j \end{cases}$$

и $c_i = \frac{b_i}{a_{ii}}, \quad i, j = \overline{1, n}; \quad ||F|| < 1.$

Теперь возьмем некоторое начальное приближение к решению $x^0 = (x_1^0, ..., x_n^0)^T$. Используя метод простой итерации каждое последующее приближение (итерацию) находим по предыдущему:

$$x^k = Fx^{k-1} + c, \quad k = 1, 2, ... \quad .$$

Последовательность приближений сходится к точному решению

$$\lim_{k\to\inf} x^k = x \quad ,$$

если какая-либо норма матрицы F меньше единицы, например,

$$||F|| = \max_{1 \le i \le n} \sum_{j=1}^{n} |f_{ij}| < 1 \quad .$$

При этом абсолютная погрешность очередной итерации

$$\Delta_k \le \frac{||F||}{1 - ||F||} ||x^k - x^{k-1}|| \quad ,$$

а относительная погрешность -

$$\delta_k = \frac{\Delta_k}{||x^k||} \quad ,$$

где $||x^k|| = \max_{1 \le i \le n} |x_i^k|$.

Метод Зейделя является модификацией метода простой итерации и сходится, как правило, быстрее. Для того, чтобы решить СЛАУ методом Зейделя неободимо представить матрицу F как сумму нижнетреугольной матрицы F_d и верхнетреугольной матрицы F_u.

Тогда k-ая итерация метода будет удовлетворять рекуррентному соотношению

$$x^k = F_d x^k F_u x^{k-1} + c, \quad k = 1, 2, ... \quad .$$

Практическая реализация: Листинг 1. Решение СЛАУ методом простой итерации и методом Зейделя

```
#!python
# -*- coding: utf-8 -*-
import numpy as np

alpha = 0.1 * 4
beta = 0.1 * 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]
              ])
b = np.array([1 + beta, 2 - beta, 3, 1])
# A = np.ones((4,4)) - A
# alpha = 0.01 * 4
# beta = 0.01 * 4
# A = np.array([[2 + alpha, 1, -0.1, 1],
#               [0.4, 0.5 - alpha, 4, -8.5],
#               [0.3, -1, 1 + 2*alpha, 5.2],
```

```
#                 [1, 0.2, 2.5, -1 + alpha]
#                 ])
# b = np.array([2.7 - beta, 21.9, -3.9, 9.9])

F = []
for i in range(len(A)):
    buf = A[i]
    buf = buf * -1
    buf = buf / (A[i][i] * -1)
    buf[i] = 0
    F.append(buf)
F = np.asarray(F)
F

c = []
for i in range(len(b)):
    buf = b[i] / A[i][i]
    c.append(buf)
c = np.asarray(c)
c

F_norm = max([x.sum() for x in np.array([[abs(x) for x in elem] for elem in F]
F_norm

def calculate_abs_error(x_k, x_k_min_1):
    m = x_k - x_k_min_1
    norm = max([x.sum() for x in np.array([abs(elem) for elem in m])])
    return (F_norm * norm) / (1 - F_norm)

def calculate_rel_error(abs_err, x_k):
    return abs_err / max([x.sum() for x in np.array([abs(elem) for elem in x_k

x_begin = c
rel_err = 1
k = 1
while rel_err > 0.01:
    x_new = F @ x_begin + c
    abs_err = calculate_abs_error(x_new, x_begin)
    print('----------------------')
```

```
    print('Abs error for', k, 'iter - ', abs_err)
    rel_err = calculate_rel_error(abs_err, x_new)
    print('Rel error for', k, 'iter - ', rel_err)
    k += 1
    x_begin = x_new

F_d = np.tril(F)
F_u = np.triu(F)

x_begin = c
rel_err = 1
k = 1
while rel_err > 1e-4:
    x_new = F_u @ x_begin
    x_new = F_d @ x_new + c
    abs_err = calculate_abs_error(x_new, x_begin)
    print('-----------------------')
    print('Abs error for', k, 'iter - ', abs_err)
    rel_err = calculate_rel_error(abs_err, x_new)
    print('Rel error for', k, 'iter - ', rel_err)
    k += 1
    x_begin = x_new
```

Результаты:

В результате работы программы были получены следующие результаты:

$$F_{4\times4} = \begin{pmatrix} 0. & -0.09615385 & 0.01923077 & 0.19230769 \\ 0.0862069 & 0. & -0.17241379 & 0.00862069 \\ 0.02586207 & -0.34482759 & 0. & 0.0862069 \\ 0.02631579 & -0.03947368 & -0.06578947 & 0. \end{pmatrix}$$

$$c_{1\times4} = \begin{pmatrix} 0.13461538 \\ 0.13793103 \\ 0.25862069 \\ 0.13157895 \end{pmatrix}$$

$||F|| = 0.45689655172413796$.

Результат выполнение программы (Листинг 1), метод простой итерации, относительная погрешность 0.01:

Номер итерации k	Абсолютная погрешность Δ_k	Относительная погрешность δ_k
1	0.02754147575866707	0.12192821609126081
2	0.008237953160882702	0.03495471902485724
3	0.0017619754007710238	0.007543330360705073 4

Метод с заданными параметрами сошелся за 3 итерации.

Результат выполнение программы (Листинг 1), метод Зейделя, относительная погрешность $10^{-}4$:

Номер итерации k	Абсолютная погрешность Δ_k	Относительная погрешность δ_k
1	0.01297631511809442	0.04735097440978927
2	0.0007772513942147164	0.0028266845885775767
3	4.670064163651293e-05	0.00016980521355419676
4	2.8044621287576496e-06	1.0197001622219773e-05

Метод с заданными параметрами сошелся за 4 итерации.

Выводы:

В ходе выполнения лабораторной работы был рассмотрен метод простой итерации и метод Зейделя для решения СЛАУ. Для этих методов была написана реализация на языке программирования Python. Анализируя результаты полученной программы, можно заметить, что метод Зейделя сходится быстрее, чем метод простой итерации. При относительной погрешности $\delta = 0.01$ метод простой итерации сходится за 3 итерации, при этом метод Зейделя при значительно меньшей погрешности ($\delta = 10^{-1}$) сходится за 4 итерации. Оба метода требуют преобразования СЛАУ к определенному виду, а также наличие нормы матрицы, которая будет меньше единицы. Последнее условие может не всегда выполняться, поэтому может потребоваться приведение СЛАУ к виду, обеспечивающему сходимость методов.