|  |  |
| --- | --- |
|  | **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 Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

**Лабораторная работа № 7**

«Решение системы линейных алгебраических уравнений методом простой итерации и методом Зейделя»

по курсу «Численные методы»

Выполнила:

студент группы ИУ9-61Б

Яровикова Анастасия

Проверила:

Домрачева А. Б.

Москва, 2023

1. **Цель**

Целью данной работы является изучение и сравнительный анализ методов решения систем линейных алгебраических уравнений (СЛАУ):

- метод простой итерации,

- метод Зейделя.

**Постановка задачи**

**Дано:** СЛАУ квадртаная матрица порядка

**Задание:**

* Решить систему методом простой итерации с относительной погрешностью 0.01:

а) преобразовать систему к виду , распечатать матрицу и столбец ;

б) найти норму ;

в) в качестве начального приближения выбрать столбец ;

г) для каждой итерации распечатать абсолютную и относительную ошибки .

* Решить систему методом Зейделя, обеспечив .

**Индивидуальный вариант:**

1. **Основные теоретические сведения**

Преобразуем СЛАУ с квадратной невырожденной матрицей к виду , квадртаная матрица порядка

Зададим некоторое начальное приближение к решению .

**Метод простой итерации**

Используя метод простой итерации, каждое последующее приближение находим по предыдущему:

или в скалярной форме: .

Последовательность приближений сходится к точному решению: , если какая-либо норма матрицы меньше единицы, например, . При этом абсолютная погрешность очередной итерации:

а относительная погрешность:

Сходимость гарантирована при любом начальном приближении, однако требуемая точность будет достигнута тем скорее, чем ближе начальное приближение к точному решению. Алгоритма назначения начального приближения не существует.

В отличие от точных методов (метода Гаусса и т.д.) итерационные методы дают заведомо приближенное решение, зато методическую погрешность на каждом шаге легко оценить по формуле (1).

Погрешность вычислений не влияет на окончательный результат и может лишь увеличивать число требуемых итераций (метод является самоисправляющимся).

Всякую невырожденную систему можно привести к виду , для которого и метод простой итерации сходится. Для этого умножим обе части уравнения на матрицу слева. Получим:

т.е. . Остается подобрать матрицу так, чтобы выполнялось условие тогда

Если для всех строк (диагональные элементы превалируют), то для приведения к требуемому виду достаточно разрешить каждое уравнение системы относительно ведущей неизвестной. Тогда:

**Метод Зейделя**

Представим матрицу как сумму нижнетреугольной матрицы и верхнетреугольной матрицы

Теперь k-ая итерация метода удовлетворяет рекуррентному соотношению

или в скалярной форме: .

Метод Зейделя является модификацией и сходится, как правило, быстрее.

1. **Реализация**

Листинг 1. Метод простой итерации

def simple\_iter\_method(x, a, b):

x1 = [0] \* len(b)

for i in range(len(x)):

s = 0

for j in range(len(x)):

if i != j:

s += a[i][j] \* x[j]

x1[i] = (b[i] - s) / a[i][i]

return x1

Листинг 2. Метод Зейделя

def calc\_X\_norm(xk\_new, xk\_old):

D = 0

for i in range(len(xk\_new)):

D = max(abs(xk\_new[i] - xk\_old[i]), D)

return D

def zeidel(A, b, eps):

N = len(A)

F\_v = [[0] \* N for i in range(N)]

F\_n = [[0] \* N for i in range(N)]

c = [0] \* N

for i in range(N):

for j in range(N):

if i < j:

F\_v[i][j] = -A[i][j] / A[i][i]

elif i > j:

F\_n[i][j] = -A[i][j] / A[i][i]

c[i] = b[i] / A[i][i]

Xk1 = c.copy()

Xk2 = c.copy()

x\_norm = 1

iters = 0

while x\_norm > eps:

t = Xk2

Xk2 = [0] \* N

for i in range(N):

for j in range(N):

Xk2[i] += F\_n[i][j] \* t[j] + F\_v[i][j] \* Xk1[j]

Xk2[i] += c[i]

x\_norm = calc\_X\_norm(Xk2, t)

Xk1 = t

iters += 1

print("iters:", iters)

return Xk2

Листинг 3. Исходный код реализации

import numpy as np

import pandas as pd

from tabulate import tabulate

def simple\_iter\_method(x, a, b):

x1 = [0] \* len(b)

for i in range(len(x)):

s = 0

for j in range(len(x)):

if i != j:

s += a[i][j] \* x[j]

x1[i] = (b[i] - s) / a[i][i]

return x1

def calc\_X\_norm(xk\_new, xk\_old):

D = 0

for i in range(len(xk\_new)):

D = max(abs(xk\_new[i] - xk\_old[i]), D)

return D

def zeidel(A, b, eps):

N = len(A)

F\_v = [[0] \* N for i in range(N)]

F\_n = [[0] \* N for i in range(N)]

c = [0] \* N

for i in range(N):

for j in range(N):

if i < j:

F\_v[i][j] = -A[i][j] / A[i][i]

elif i > j:

F\_n[i][j] = -A[i][j] / A[i][i]

c[i] = b[i] / A[i][i]

Xk1 = c.copy()

Xk2 = c.copy()

x\_norm = 1

iters = 0

while x\_norm > eps:

t = Xk2

Xk2 = [0] \* N

for i in range(N):

for j in range(N):

Xk2[i] += F\_n[i][j] \* t[j] + F\_v[i][j] \* Xk1[j]

Xk2[i] += c[i]

x\_norm = calc\_X\_norm(Xk2, t)

Xk1 = t

iters += 1

print("iters:", iters)

return Xk2

def true\_solution():

return [1626087/6243449, -444093/6243449, 1666859/6243449, 1297732/6243449]

if \_\_name\_\_ == "\_\_main\_\_":

k = 29

aa = 0.1 \* k

bb = 0.1 \* k

a = np.array([[10.0 + aa, -1.0, 0.2, 2.0], [1.0, 12.0 - aa, -2.0, 0.1], [0.3, -4.0, 12.0 - aa, 1.0], [0.2, -0.3, -0.5, 8.0 - aa]])

b = np.array([1.0 + bb, 2.0 - bb, 3.0, 1.0])

print(a)

print(b)

blen = len(b)

x = [0] \* blen

c = [0] \* blen

for i in range(blen):

x[i] = b[i] / a[i][i]

c[i] = x[i]

f = []

for i in range(blen):

f.append([0] \* blen)

for j in range(blen):

if i == j:

f[i][j] = 0

else:

f[i][j] = -a[i][j] / a[i][i]

print("\n--Simple iteration method--")

print('\nF:')

for i in f:

print(i)

print('\nC:', c)

f\_norm = -1

for i in range(blen):

s = 0

for j in range(blen):

s += abs(f[i][j])

if s < 1:

f\_norm = max(s, f\_norm)

print('\n||F||:', f\_norm)

iters = 0

while True:

iters += 1

x1 = simple\_iter\_method(x, a, b)

D = 0

for i in range(len(x)):

D = max(abs(x1[i] - x[i]), D)

Delta = 0

for i in range(len(x)):

Delta = max(abs(x1[i]), Delta)

delta = D / Delta

print()

print("iter:", iters)

print("Δ\_k:", Delta)

print("δ\_k:", delta)

if delta < 0.01:

print("\nstop\n")

break

else:

x = x1

print("X:", x)

new\_b = [0] \* blen

for i in range(blen):

for j in range(blen):

new\_b[i] += a[i][j] \* x1[j]

print("B:", new\_b)

print("\n--Zeidel's method--\n")

new\_x = zeidel(a, b, 0.0001)

print("X", new\_x)

new\_b = [0] \* blen

for i in range(blen):

for j in range(blen):

new\_b[i] += a[i][j] \* new\_x[j]

print("B:", new\_b)

rr = true\_solution()

print()

diff1 = [np.abs(rr[i] - x[i]) for i in range(len(rr))]

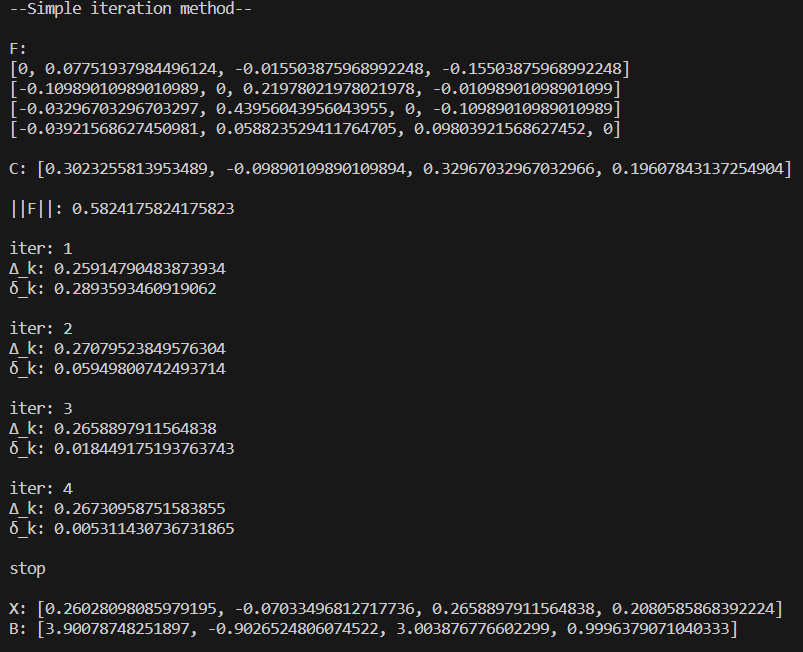
diff2 = [np.abs(rr[i] - new\_x[i]) for i in range(len(rr))]

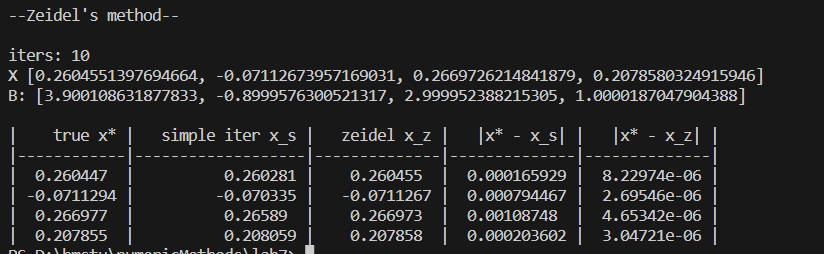
dd = {"true x\*": rr, "simple iter x\_s": x, "zeidel x\_z": new\_x, "|x\* - x\_s|": diff1, "|x\* - x\_z|": diff2}

res = pd.DataFrame(dd)

print(tabulate(res, headers='keys', tablefmt='github', showindex=False))

1. **Результаты**



**фотка**

1. **Вывод**

В ходе выполнения лабораторной работы