

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ     
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

**ДАЛЬНЕВОСТОЧНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**

**Институт математики и компьютерных технологий**

**Департамент математического и компьютерного моделирования**

**ОТЧЕТ**

**по лабораторной работе** **№ 6**

«Приближенные методы решения СЛАУ. Метод последовательной верхней релаксации, метод Ричардсона»

Выполнил: студент гр. Б9120-02.03.01сцт

Пограничный К. О.

Проверил: преподаватель

Кузнецов К. С.

Владивосток 2023

**Постановка задачи**

Дана система линейных алгебраических уравнений в виде Ax=b. Требуется решить ее приближенными методами с точностью и сравнить число итераций, потребовавшихся для решения.

**Метод решения ПВР**

Метод последовательной верхней релаксации — это численный метод решения систем линейных уравнений. Он является модификацией метода Зейделя, использующей параметр релаксации для ускорения сходимости. Метод выполняет итерации, обновляя значения переменных на каждом шаге. Выбор оптимального значения параметра релаксации может существенно влиять на скорость сходимости метода.

**Алгоритм решения ПВР**

Для каждого метода можно выделить общий алгоритм:

1. Выбирается начальное приближение
2. Выбирается итерационный параметр
3. Вычисляется новое приближенное значение
4. Проверяется условие выхода из цикла
5. Повтор 3–4

Метод последовательной верхней релаксации:

где решение на новом итерационном шаге, решение со старого итерационного шага,– итерационный параметр, предпочтительно выбираемый из интервала (0, 2), матрица с компонентами:

вектор с компонентами

**Метод решения Ричардсона**

Метод Ричардсона — это итерационный численный метод для решения систем линейных уравнений. Он основан на последовательном обновлении значений переменных с использованием простой формулы.

**Алгоритм решения Ричардсона**

где решение на новом итерационном шаге, решение со старого итерационного шага, матрица с компонентами:

вектор с компонентами , – адаптивный шаг, и минимальные и максимальные собственные значения матрицы А.

**Результаты тестов**

После проведения численных экспериментов с матрицей

A = , векторами b = , ,

были получены следующие результаты:

1. Метод последовательной верхней релаксации:

Решение: [0.42752475 0.36574258 0.17574257 0.20029703]

Количество шагов: 51

1. Метод Ричардсона:

Решение: [0.43855807 0.35764046 0.18689612 0.2120633]

Количество шагов: 7

**Вывод**

В ходе выполнения лабораторной работы были изучены и реализованы два итерационных численных метода для решения систем линейных уравнений: метод последовательной верхней релаксации (ПВР) и метод Ричардсона.

Метод последовательной верхней релаксации позволяет ускорить сходимость метода Зейделя путем введения параметра релаксации. Этот параметр может быть выбран оптимальным образом для достижения наилучшей скорости сходимости.

Метод Ричардсона может быть эффективен для хорошо обусловленных систем линейных уравнений или систем с большим числом уравнений. В частности, для нашей положительно определенной матрицы с диагональным преобладанием, метод Ричардсона был эффективней.

**Приложение**

Листинг программы:

import math  
import numpy as np  
import warnings  
  
  
warnings.simplefilter(action="ignore", category=RuntimeWarning)  
  
  
def relaxation(\_A, \_b, \_x\_start):  
 omega = 0.7  
 \_x\_next = np.copy(\_x\_start)  
 for i in range(len(\_A)):  
 \_x\_next[i] = (1 - omega) \* \_x\_start[i] +\  
 omega \* (- sum((\_A[i][j] \* \_x\_start[j]) for j in range(i + 1, len(\_A)))  
 - sum((\_A[i][j] \* \_x\_next[j]) for j in range(0, i)) + \_b[i]) / \_A[i][i]  
 k = 1  
 while np.linalg.norm(\_x\_next - \_x\_start) >= 10e-10:  
 \_x\_start = np.copy(\_x\_next)  
 for i in range(len(\_A)):  
 \_x\_next[i] = (1 - omega) \* \_x\_start[i] +\  
 omega \* (- sum((\_A[i][j] \* \_x\_start[j]) for j in range(i + 1, len(\_A)))  
 - sum((\_A[i][j] \* \_x\_next[j]) for j in range(0, i)) + \_b[i]) / \_A[i][i]  
  
 k += 1  
  
 return \_x\_next, k  
  
  
def nu(k, n):  
 return math.cos((2 \* k - 1) \* math.pi / (2 \* n))  
  
  
def richardson(\_A, \_b, \_x\_start):  
 lambda\_max = max(np.linalg.eigvals(\_A))  
 lambda\_min = min(np.linalg.eigvals(\_A))  
  
 eta = lambda\_min / lambda\_max  
 ro0 = (1 - eta) / (1 + eta)  
 tau0 = 2 / (lambda\_max + lambda\_min)  
 n = 7  
  
 \_x\_next = \_x\_start + (tau0 / (1 + ro0 \* nu(1, n))) \* (- \_A.dot(\_x\_start) + \_b)  
 k = 2  
 while k < n and np.linalg.norm(\_x\_next - \_x\_start) >= 10e-10:  
 \_x\_start = np.copy(\_x\_next)  
 \_x\_next = \_x\_start + (tau0 / (1 + ro0 \* nu(k, n))) \* (- \_A.dot(\_x\_start) + \_b)  
 k += 1  
  
 return \_x\_next, k  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 A = np.array([  
 [8, -1, -6, 0],  
 [-5, 24, 10, 8],  
 [0, 9, 12, -2],  
 [1, 0, -7, 9]  
 ])  
 b = np.array([2, 10, 5, 1])  
  
 print(np.linalg.eigvals(A))  
 x\_start = np.array([0.00, 0.00, 0.00, 0.00])  
  
 x1, k1 = relaxation(A, b, x\_start)  
 print(f"Метод последовательной верхней релаксации:\nx = {x1}\nk = {k1}\n")  
 x2, k2 = richardson(A, b, x\_start)  
 print(f"Метод Ричардсона:\nx = {x2}\nk = {k2}\n")  
  
 print(f"Действительные значения x = {np.linalg.inv(A).dot(b)}")