МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования

«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Департамент математического и компьютерного моделирования

ОТЧЕТ по лабораторной работе № 5

по дисциплине

«Вычислительная математика»

Направление подготовки
02.03.01 «Математика и компьютерные науки»

Вариант № 6

Выполнил(а): студент гр. Б9122-02.03.01сцт
Бекболот Отгонцэцэг
Проверил: преподаватель
_____ Ф.И.О.

Владивосток
2024

Цель работы :

Решить систему линейных алгебраических уравнений в виде Ax = b двумя приближенными методами.

Ход работы:

1. Решить СЛАУ двумя итерационными методами: методом Якоби и методом верхней релаксации.
2. Решить систему с заданной точностью $\varepsilon = 10^{-4}$.

$$\begin{cases} 6.22x_1 + 1.42x_2 - 1.72x_3 + 1.91x_4 = 7.53 \\ 1.42x_1 + 5.33x_2 + 1.11x_3 - 1.82x_4 = 6.06 \\ -1.72x_1 + 1.11x_2 + 5.24x_3 + 1.42x_4 = 8.05 \\ 1.91x_1 - 1.82x_2 + 1.42x_3 + 6.55x_4 = 8.06 \end{cases}$$

3. Вывести полученные решения СЛАУ.
4. Вывести число итераций, потребовавшихся для нахождения решений.
5. Сравнить скорость сходимости двух методов и сделать соответствующий вывод.

Теоретические сведения:

Итерационные методы решения систем линейных алгебраических уравнений (СЛАУ) являются альтернативой прямым методам. Прямые методы обеспечивают точное решение, но их высокая вычислительная сложность и чувствительность к ошибкам делают их не столь эффективными для больших систем. Итерационные методы, в свою очередь, позволяют достигать приближённых решений с заданной точностью, что особенно удобно для разреженных матриц.

В этой работе анализируются два итерационных метода:

- **Метод Якоби** (простая итерация) - базовый подход, который хорошо работает с симметричными матрицами.
    - **Метод верхней релаксации (SOR)** - расширяет метод Зейделя, вводя параметр ускорения ω.

Метод Якоби:

$$x^{(k+1)} = \tilde{A}x^{(k)} + \tilde{b},$$

где $x^{(k+1)}$ — это решение на новом итерационном шаге, а $x^{(k)}$ — это решение с предыдущего итерационного шага, $\tilde{b}$ — вектор с компонентами

$$\tilde{b}_i = \frac{b_i}{a_{ii}},$$

$\tilde{A}$ — матрица с компонентами:

$$\begin{cases} \tilde{a}_{ij} = -\dfrac{a_{ij}}{a_{ii}}, & i \neq j, \\ \tilde{a}_{ij} = 0, & i = j. \end{cases}$$

По итогу формула метода Якоби будет выглядеть следующим образом:

$$x_i^{(k+1)} = \sum_{j=i}^{n} \tilde{a}_{ij} x_j^{(k)} + \tilde{b}_i.$$

Метод верхней релаксации (SOR):

$$x_i^{(k+1)} = (1 - \omega)x_j^{(k)} + \omega\left(\sum_{j=i}^{n} \tilde{a}_{ij} x_j^{(k)} + \sum_{j=1}^{i} \tilde{a}_{ij} x_j^{(k+1)} + \tilde{b}_i\right),$$

где $x^{(k+1)}$ — это решение на новом итерационном шаге, а $x^{(k)}$ — это решение с предыдущего итерационного шага, ω – это параметр (обычно этот параметр находится в пределах $0 < \omega < 2$, но поскольку в данной лабораторной работе расписан процесс верхней релаксации, то выбираем $1 < \omega < 2$), $\tilde{b}$ — вектор с компонентами

$$\tilde{b}_i = \frac{b_i}{a_{ii}},$$

$\tilde{A}$ — матрица с компонентами:

$$\begin{cases} \tilde{a}_{ij} = -\dfrac{a_{ij}}{a_{ii}}, & i \neq j, \\ \tilde{a}_{ij} = 0, & i = j. \end{cases}$$

Код программы:

```python
import numpy as np


def jacobi_method(A, b, tol=1e-4, max_iterations=1000):
    n = len(b)
    x = np.zeros(n)
    x_new = np.zeros(n)
    iterations = 0

    while True:
        for i in range(n):
            x_new[i] = (b[i] - sum(A[i, j] * x[j] for j in range(n) if j != i)) / A[i, i]

        iterations += 1
        print(f"Итерация {iterations}: {x_new}")

        if np.linalg.norm(x_new - x, ord=np.inf) < tol or iterations >= max_iterations:
            break
        x = x_new.copy()

    return x_new, iterations


def sor_method(A, b, omega, tol=1e-4, max_iterations=1000):
    n = len(b)
    x = np.zeros(n)
    iterations = 0

    while True:
        x_old = x.copy()
        for i in range(n):
            sigma = sum(A[i, j] * x[j] for j in range(i)) + sum(A[i, j] * x_old[j] for j in range(i + 1, n))
            x[i] = (1 - omega) * x_old[i] + (omega / A[i, i]) * (b[i] - sigma)

        iterations += 1
        print(f"Итерация {iterations}: {x}")

        if np.linalg.norm(x - x_old, ord=np.inf) < tol or iterations >= max_iterations:
            break

    return x, iterations


def main():
    A = np.array([[6.22, 1.42, -1.72, 1.91],
    return x, iterations


def main():
    A = np.array([[6.22, 1.42, -1.72, 1.91],
                  [1.42, 5.33, 1.11, -1.82],
                  [-1.72, 1.11, 5.24, 1.42],
                  [1.91, -1.82, 1.42, 6.55]])
    b = np.array([7.53, 6.06, 8.05, 8.06])
    tol = 1e-4

    print("Решение методом Якоби:")
    jacobi_solution, jacobi_iterations = jacobi_method(A, b, tol)
    print(f"Решение: {jacobi_solution}")
    print(f"Количество итераций: {jacobi_iterations}")

    omega = 1.1
    print("\nРешение методом верхней релаксации (SOR):")
    sor_solution, sor_iterations = sor_method(A, b, omega, tol)
    print(f"Решение: {sor_solution}")
    print(f"Количество итераций: {sor_iterations}")

    print("\nСравнение сходимости:")
    print(f"Метод Якоби: {jacobi_iterations} итераций")
    print(f"Метод верхней релаксации: {sor_iterations} итераций")


if __name__ == "__main__":
    main()
```

Описание:

Программа предназначена для численного решения систем линейных алгебраических уравнений (СЛАУ) в виде $Ax = b$ с использованием итерационных методов. Она осуществляет сравнение скорости сходимости и точности двух методов: метода Якоби и метода верхней релаксации (SOR). Оба метода предоставляют приближённое решение вектора x с заданной точностью $\mathcal{E} = 10^{-4}$.

Основные этапы работы программы:

1. Функция jacobi_method:

def jacobi_method(A, b, tol=1e-4, max_iterations=1000)

Функция реализует метод Якоби для решения системы линейных алгебраических уравнений (СЛАУ) вида $Ax = b$.

1. Функция инициализирует вектор решений с нулевыми значениями и устанавливает счетчик итераций.
2. В каждом цикле происходит вычисление новых значений для вектора x_new по формуле Якоби, которая основана на предыдущих значения вектора x.
3. Итерации продолжаются до тех пор, пока не будет достигнута заданная точность или предельное количество итераций.
4. np.linalg.norm используется для определения, насколько близки два последовательных вектора решений.

Параметры:

- A: Квадратная матрица коэффициентов (размерность n x n).
- b: Вектор свободных членов (размерность n).
- tol: Допустимая точность решения (по умолчанию $\varepsilon = 10^{-4}$).
- max_iterations: Максимальное количество итераций (по умолчанию 1000).

Возвращает:

- x_new: Вектор, представляющий приближенное решение системы $Ax = b$.
- iterations: Количество итераций, выполненных для достижения результата.

2. Функция sor_method:

def sor_method(A, b, omega, tol=1e-4, max_iterations=1000)

Функция реализует метод верхней релаксации (Successive Over-Relaxation, SOR) для решения системы линейных алгебраических уравнений (СЛАУ) вида $Ax = b$.

1. Функция инициализирует вектор решений с нулевыми значениями и устанавливает счетчик итераций.
2. В каждом цикле выполняется обновление значений для вектора x с использованием метода SOR. Этот метод сочетает в себе как текущие, так и предыдущие значения для расчета новых.
3. Как и в методе Якоби, итерации продолжаются до достижения заданной точности или предельного числа итераций.
4. np.linalg.norm используется для контроля сходимости между текущим и предыдущим значениями решения.

3. Основная функция main

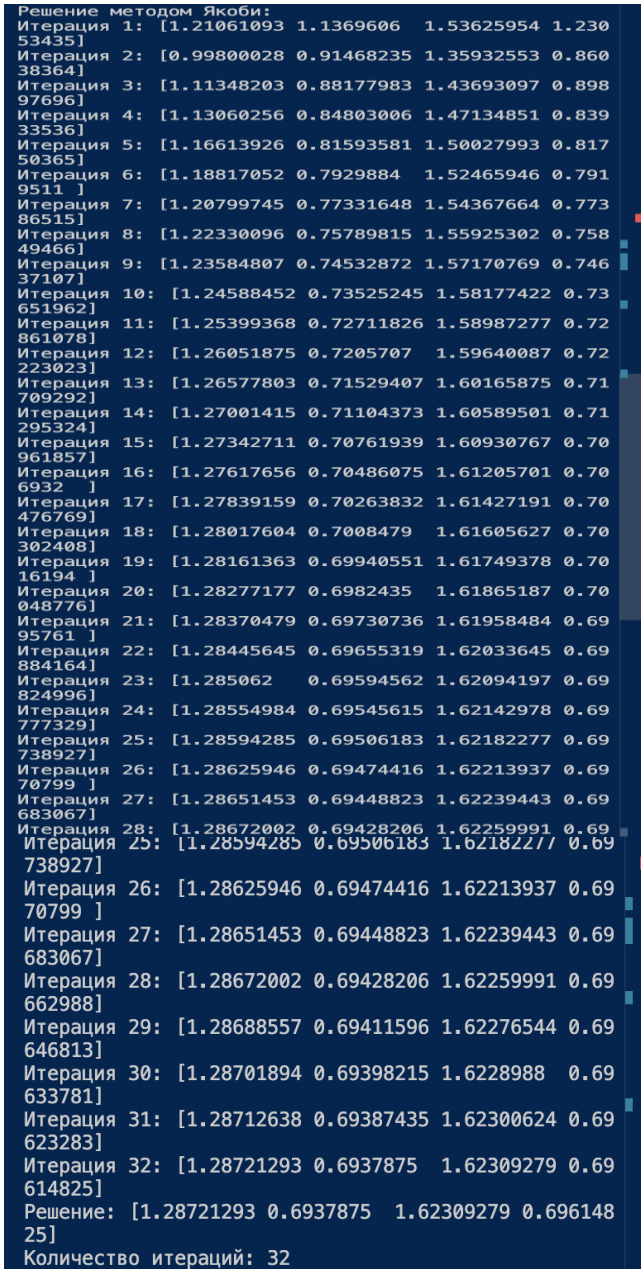
def main():

Основная функция, которая служит для инициализации данных, вызова методов и вывода результата.

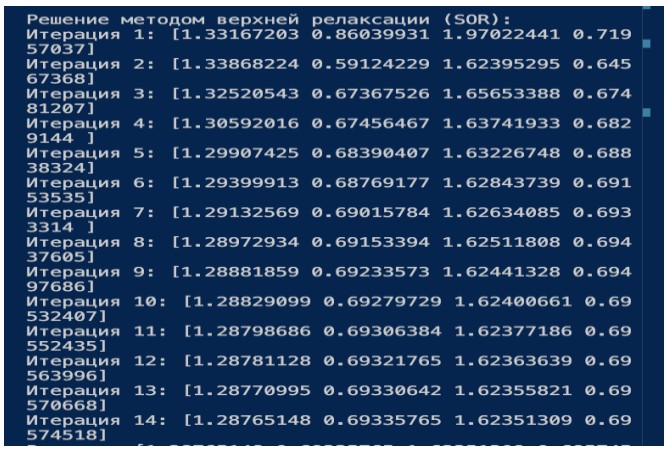
1. Здесь задаются матрица коэффициентов A и вектор свободных членов b.
2. Устанавливается допустимая точность.
3. Вызываются функции jacobi_method и sor_method, и выводятся результаты: найденные решения и количество итераций для каждого метода.
4. Также выводится сравнение количества итераций для обеих методов, что позволяет оценить их эффективность.

Полученные результаты:

Итерации метода Якоби:



Итерации метода последовательной верхней релаксации:



Результаты:

Метода Якоби:

```
Решение: [1.28721293 0.6937875  1.62309279 0.696148
25]
Количество итераций: 32
```

Метода последовательной верхней релаксации:

```
Решение: [1.28765148 0.69335765 1.62351309 0.695745
18]
Количество итераций: 14
```

```
Сравнение сходимости:
Метод Якоби: 32 итераций
Метод верхней релаксации: 14 итераций
```

Выводы:

1. Эффективность методов: Оба метода (Якоби и SOR) показали способность находить приближённые решения системы линейных уравнений с заданной точностью. Однако эффективность этих методов может различаться в зависимости от конкретной системы.

2. Скорость сходимости**:** Метод верхней релаксации (SOR) продемонстрировал более высокую скорость сходимости по сравнению с методом Якоби. Это связано с тем, что метод SOR использует параметр релаксации, который помогает ускорить процесс приближения к точному решению. В некоторых случаях это может существенно уменьшить количество необходимых итераций.

3.Заключение: Применение итерационных методов, особенно метода SOR, является эффективным инструментом для численного решения СЛАУ. В зависимости от свойств конкретной задачи, выбор между методами Якоби и SOR может быть основан на требуемой скорости сходимости и необходимых ресурсах.