Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №1
по дисциплине «Вычислительная математика»

Вариант: **1**

Преподаватель:
Малышева Татьяна Алексеевна
Машина Екатерина Алексеевна

Выполнил: Бондарев Алексей Михайлович
Группа: P3212

Санкт-Петербург, 2025 г

Цель работы

Изучить численные методы решения систем линейных алгебраических уравнений и реализовать один из них средствами программирования.

Задание

Для итерационных методов должно быть реализовано:

1.Точность задается с клавиатуры/файла,

2. Проверка диагонального преобладания (в случае, если диагональное преобладание в исходной матрице отсутствует, сделать перестановку строк/столбцов до тех пор, пока преобладание не будет достигнуто). В случае невозможности достижения диагонального преобладания - выводить соответствующее сообщение.

3. Вывод нормы матрицы (любой, на Ваш выбор),

4. Вывод вектора неизвестных.

5. Вывод количества итераций, за которое было найдено решение,

6. Вывод вектора погрешностей.

Описание метода

Итерационные методы дают возможность для системы (1) построить последовательность векторов $x^{(0)}, x^{(1)}, \dots, x^{(k)}$, пределом которой должно быть точное решение $x^{(*)}$: $x^{(*)} = \lim_{k\to\infty} x^{(k)}$ Построение последовательности заканчивается, как только достигается желаемая точность.

Приведем систему уравнений, выразив неизвестные $x1, x2, \dots, xn$ соответственно из первого, второго и т.д. уравнений системы.

Листинг программы

```python
import numpy as np


1 usage
def read_data_from_keyboard(): #Считывает данные с клавиатуры.
    print("Введите размерность матрицы n (1-20): ")
    n = int(input().strip())
    if n > 20 or n <= 0:
        raise ValueError("Недопустимое значение n.")

    A = []
    print(f"Введите матрицу A ({n}x{n} построчно, числа через пробел):")
    for _ in range(n):
        row = list(map(float, input().split()))
        if len(row) != n:
            raise ValueError("Неверное количество элементов в строке.")
        A.append(row)

    print(f"Введите вектор b ({n} чисел через пробел):")
    b = list(map(float, input().split()))
    if len(b) != n:
        raise ValueError("Неверное количество элементов в векторе b.")

    return n, A, b
```

```python
def read_data_from_file(filename="input.txt"):  #Считывает данные из файла.
    with open(filename, 'r') as f:
        lines = [line.strip() for line in f if line.strip()]

    idx = 0
    n = int(lines[idx])
    idx += 1
    A = []
    for _ in range(n):
        A.append(list(map(float, lines[idx].split())))
        idx += 1

    b = list(map(float, lines[idx].split()))
    return n, A, b
```

```python
def check_and_make_diagonally_dominant(A, b): #беспечивает диагональное преобладание.
    n = len(A)
    for i in range(n):
        max_row = i
        max_val = abs(A[i][i])
        for j in range(i, n):
            current = abs(A[j][i])
            if current > max_val:
                max_val = current
                max_row = j

        if max_val == 0:
            return False  # Нулевой диагональный элемент

        A[i], A[max_row] = A[max_row], A[i]
        b[i], b[max_row] = b[max_row], b[i]

    return True
```

```python
def simple_iterations(A, b, eps=1e-6, max_iter=1000): #Реализация метода простых итераций.
    n = len(A)
    D = np.diag(A)
    if any(d == 0 for d in D):
        raise ValueError("На диагонали есть нулевые элементы после перестановки.")

    B = -np.array(A) / D[:, None]
    np.fill_diagonal(B,  val: 0)
    c = np.array(b) / D

    x = np.zeros(n)
    for it in range(1, max_iter + 1):
        x_new = B @ x + c
        if np.linalg.norm(x_new - x, np.inf) < eps:
            return x_new, it, (x_new - x)
        x = x_new

    return x, max_iter, (x - x_new)
```

```python
def print_results(A, b, x, iterations, diff, eps): #Выводит результаты работы.
    print("\nРезультаты:")
    print(f"Количество итераций: {iterations}")
    print("Вектор решения:")
    print(np.array2string(x, precision=9, suppress_small=True))

    print("\nВектор погрешностей (последняя итерация):")
    print(np.array2string(diff, precision=2, suppress_small=True))

    residual = A @ x - b
    print("\nВектор невязки (A*x - b):")
    print(np.array2string(residual, precision=2, suppress_small=True))

    print(f"\nПроверка точности (||Δx|| < {eps}): {np.linalg.norm(diff, np.inf):.2e}")
```

```python
def compare_with_numpy(A, b):  #Сравнение с библиотечным решением.
    try:
        x_lib = np.linalg.solve(A, b)
        print("\nРешение с использованием numpy.linalg.solve:")
        print(np.array2string(x_lib, precision=9, suppress_small=True))
        return x_lib
    except np.linalg.LinAlgError:
        print("\nМатрица вырождена, библиотечное решение невозможно.")
        return None
```

```python
def main():
    print("Лабораторная работа: Метод простых итераций")
    print("=" * 50)

    # Выбор источника данных
    source = input("Ввод данных:\n1 - Клавиатура\n2 - Файл\nВыбор: ").strip()
    if source == '1':
        n, A, b = read_data_from_keyboard()
    elif source == '2':
        n, A, b = read_data_from_file()
    else:
        print("Неверный ввод!")
        return

    A = np.array(A, dtype=float)
    b = np.array(b, dtype=float)

    # Проверка диагонального преобладания
    if not check_and_make_diagonally_dominant(A, b):
        print("Невозможно достичь диагонального преобладания!")
        return

    # Ввод точности
    eps = float(input("Введите точность (например, 1e-6): ").strip() or 1e-6)

    # Решение методом простых итераций
    try:
        x, iterations, diff = simple_iterations(A, b, eps)
    except ValueError as e:
        print(f"Ошибка: {str(e)}")
        return

    # Вывод результатов
    print_results(A, b, x, iterations, diff, eps)

    # Сравнение с numpy
    x_lib = compare_with_numpy(A, b)

    # Анализ различий
    if x_lib is not None:
        diff = x - x_lib
        print("\nРазница с библиотечным решением:")
        print(f"Максимальная: {np.abs(diff).max():.2e}")
        print(f"Средняя: {np.abs(diff).mean():.2e}")
        print("\nОбъяснение: Различия вызваны:")
        print("- Ограниченным числом итераций метода")
        print("- Накоплением ошибок округления")
        print("- Особенностями метода (диагональное преобладание)")


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

Примеры работы программы

Ввод с клавиатуры:

```
Лабораторная работа: Метод простых итераций
==================================================
Ввод данных:
1 - Клавиатура
2 - Файл
Выбор: 1
Введите размерность матрицы n (1-20):
3
Введите матрицу A (3x3 построчно, числа через пробел
1 2 3
4 5 6
7 8 9
Введите вектор b (3 чисел через пробел):
1 2 3
Введите точность (например, 1e-6): 1

Результаты:
Количество итераций: 1
Вектор решения:
[0.428571429 0.125       0.222222222]

Вектор погрешностей (последняя итерация):
[0.43 0.12 0.22]

Вектор невязки (A*x - b):
[3. 5. 4.]

Проверка точности (||Δx|| < 1.0): 4.29e-01

Матрица вырождена, библиотечное решение невозможно.
```

Ввод из файла:
Содержание текстового файла:

```
3 # размер матрицы
4 1 1 # 1 строка
1 5 1 # 2 строка
1 1 6 # 3 строка
6 7 8 # вектор
```

Пример вывода:

```
==================================================
Ввод данных:
1 - Клавиатура
2 - Файл
Выбор: 2
Введите точность (например, 1e-6): 1

Результаты:
Количество итераций: 2
Вектор решения:
[0.816666667 0.833333333 0.85      ]

Вектор погрешностей (последняя итерация):
[-0.68 -0.57 -0.48]

Вектор невязки (A*x - b):
[-1.05 -1.17 -1.25]

Проверка точности (||Δx|| < 1.0): 6.83e-01

Решение с использованием numpy.linalg.solve:
[1. 1. 1.]

Разница с библиотечным решением:
Максимальная: 1.83e-01
Средняя: 1.67e-01

Объяснение: Различия вызваны:
- Ограниченным числом итераций метода
- Накоплением ошибок округления
- Особенностями метода (диагональное преобладание)

Process finished with exit code 0
```

Вывод:
В результате выполнения данной лабораторной работой я познакомился с численными методами решения математических задач на примере систем алгебраических уравнений, реализовав на языке программирования python метод простых итераций.