МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.Ломоносова

Факультет вычислительной математики и кибернетики

Компьютерный практикум по учебному курсу

«ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»

ЗАДАНИЕ № 1

Подвариант № 2

ИТЕРАЦИОННЫЕ МЕТОДЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

(на примере метода верхней релаксации)

ОТЧЕТ

о выполненном задании

студента 202 учебной группы факультета ВМК МГУ

Плеханова Антона Дмитриевича

гор. Москва

2020 год

Содержание

Цель работы

Изучить итерационный метод верхней релаксации, используемый для численного решения систем линейных алгебраических уравнений; изучить скорость сходимости этого метода в зависимости от выбора итерационного параметра ω

Постановка задачи

Дана система уравнений $Ax = f$ порядка $n \times n$ с невырожденной матрицей A. Написать программу численного решения данной системы линейных алгебраических уравнений (n - параметр программы), использующую численный алгоритм итерационного метода верхней релаксации.

Задачи практической работы

1. Решить заданную СЛАУ итерационным методом верхней релаксации;
2. Разработать критерий остановки итерационного процесса, гарантирующий получение приближенного решения исходной системы СЛАУ с заданной точностью;
3. Изучить скорость сходимости итераций к точному решению задачи (при использовании итерационного метода верхней релаксации провести эксперименты с различными значениями итерационного параметра;
4. Правильность решения СЛАУ подтвердить системой тестов.

Алгоритм

Рассматриваем СЛАУ $Ax = f$ с матрицей $(A)_{ij} = a_{ij}$
Метод верхней релаксации - линейный одношаговый итерационный метод, имеющий следующую форму:

$$(D + \omega A^{(-)}) \frac{x^{k+1} - x^k}{\omega} + Ax^k = f,$$

где D - диагональная часть матрицы A, $A^{(-)}$ - нижняя треугольная часть матрицы A (элементы выше либо на диагонали - нулевые) , ω - итерационный параметр.
Придадим записанному выше соотношению следующий вид:

$$\left(\frac{1}{\omega} D + A^{(-)}\right)(x_{k+1} - x_k) + Ax_k = f$$

,

$$\left(\frac{1}{\omega} D + A^{(-)}\right) x_{k+1} + \left[\left(1 - \frac{1}{\omega}\right) D + A^{(+)}\right] x_k = f,$$

где $A^{(+)}$ - верхняя треугольная часть матрицы A (элементы ниже либо на диагонали - нулевые). Перейдем к векторной записи в последнем соотношении и выразим компоненту x_i^{k+1}:

$$x_i^{k+1} = x_i^k + \frac{\omega}{a_{ii}} \left(f_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i}^{n} a_{ij} x_i^k \right),\ i = 1, ..., n$$

Именно в таком виде происходит вычисление компонент вектора x_{k+1} на $k-$ой итерации метода.
В качестве критерия остановки было выбрано следующее условие: $||\psi_k|| <= \varepsilon$, где $\psi_k = Ax_k - f$ - невязка матричного уравнения.

Описание программы

Программа написана на языке Python3. Для ее работы необходимо наличие установленной библиотеки numpy. При запуске программы пользователь должен выбрать способ задания матрицы (во всех случаях подразумевается расширенная матрица СЛАУ):

1. Выбрать матрицу из варианта задания (всего 4 варианта, для варианта №4 необходимо дополнительно ввести число x - параметр правой части системы)

2. Задать матрицу из файла

3. Ввести матрицу через стандартный поток ввода

Затем программа попросит ввести точность вычисления решения системы (параметр ε в критерии остановки итерационного процесса) и итерационный параметр ω. После этого программа выведет на стандартный поток вывода решение системы и предложит проверить его корректность с помощью функции $numpy.linalg.solve$ (для этого надо ввести 1, иначе - 0). В программе реализованы следующие функции:

- $create\_LAES(x)$ - функция генерирует расширенную матрицу СЛАУ, заданной вариантом задания (приложение 2, пример 2, вариант 4)

- $UpperRealxMethod(A,\ f,\ omega,\ eps,\ print\_iters = False)$ - функция реализует метод итерационный процесс метода верхней релаксации для системы $Ax = f$ с итерационным параметром $omega$ и точностью вычисления eps. Если аргумент $print\_iters == True$, то функция каждую 10000-ую итерацию будет выводить номер итерации и норму невязки. В данной функции на самом деле решается эквивалентная система $Bx = g$, где $B = A^T A$, $g = A^T f$. Преимущество этой системы заключается в положительно определенной симметричной матрице коэффициентов, то есть это уравнение удовлетворяет теореме Самарского, что обеспечивает сходимость метода.

Код программы

```
# coding: utf-8

import numpy as np
from numpy import linalg as LA
import sys


# ## Функция создания матрицы
# приложение( 2, пример 2, вариант 4)

# Функция генерирует расширенную матрице СЛАУ, заданной вариантом задания
def create_LAES(x):
    n = 100
    M = 4
    q = 1.001 - 2 * M * 10**(-3)

    # матрица системы
    A = np.zeros((n, n + 1))

    for i in range(n):
        for j in range(n):
            if (i == j):
                A[i][j] = (q - 1)**(i + j)
            else:
                A[i][j] = q**(i + j) + 0.1 * (j - i)

    # вычисляем столбец правой части системы
    vec_x = np.full(n, x)
    vec_x_div_i = np.array([x / i for i in range(1, n+1)])

    A[:, n] = vec_x * np.exp(vec_x_div_i) * np.cos(vec_x_div_i)

    return A


# ## Реализация метода верхней релаксации

def UpperRelaxMethod(A, f, omega, eps, print_iters=False):
    # Поскольку все матрицы исходных уравнений не удовлетворяли условиям теоремы Самарского,
    # вместо исходной системы Ax=f рассматривается система (A^T)Ax = (A^T)f, имеющая то же
    # решение, но уже с симметричной и положительно определенной матрицей коэффициентов

    n = A.shape[0]
    B = np.dot(A.transpose(), A) # B = (A^T)A
    g = np.dot(A.transpose(), f) # g = (A^T)f

    # вектор решений
    x = np.zeros(n)

    # количество итераций, понадобившихся для решения
```

```python
    iters = 0

    # пока норма невязки больше заданной точности:
    while LA.norm(g - np.dot(B, x)) > eps:
        iters += 1

        # вычисляем компоненты решения в соответствии с рекуррентной формулой
        for i in range(n):
            x[i] = x[i] + omega / B[i][i] * (g[i] - np.dot(B[i], x))

        if print_iters == True and iters % 10000 == 0:
            print(f"{iters} итерация; норма невязки: {LA.norm(g - np.dot(B, x))}")

    print(f"Решение найдено за {iters} итераций")

    return x

def main():
    print("Выберите способ задания матрицы:\n"
        "1 - выбор матрицы из варианта задания\n"
        "2 - ввод матрицы из файла\n"
        "3 - ввод матрицы через стандартный поток ввода")

    mode = int(input())

    if mode == 1:
        print("Выберите номер матрицы от( 1 до 4): ")
        num = input()
        if num == '4':
            x = float(input("Введите x: "))
            extend_A = create_LAES(x)
        else:
            file_name = 'matrix_' + num + '.txt'
            extend_A = np.loadtxt(file_name, delimiter=' ')
    elif mode == 2:
        file_name = input('Введите имя файла: ')
        extend_A = np.loadtxt(file_name, delimiter=' ')
    elif mode == 3:
        print("Введите расширенную матрицу СЛАУ:")
        extend_A = np.loadtxt(sys.stdin, delimiter=' ')
    else:
        print("Invalid input")
        return 1

    A = extend_A[:, : -1]
    f = extend_A[:, -1]

    eps = float(input('Введите точность: '))
    omega = float(input('Введите параметр омега: '))

    x = UpperRelaxMethod(A, f, omega=omega, eps=eps, print_iters=(A.shape[0] >= 10))
    print(f"Решение:\n{x}")

```

```python
    print("Для проверки корректности решения с помощью библиотеки numpy
   введите 1, иначе введите 0")
    check = int(input())

    if check == 1:
        print("Решение системы функцией numpy.linalg.solve:")
        print(LA.solve(A, f))


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

Тестирование программы

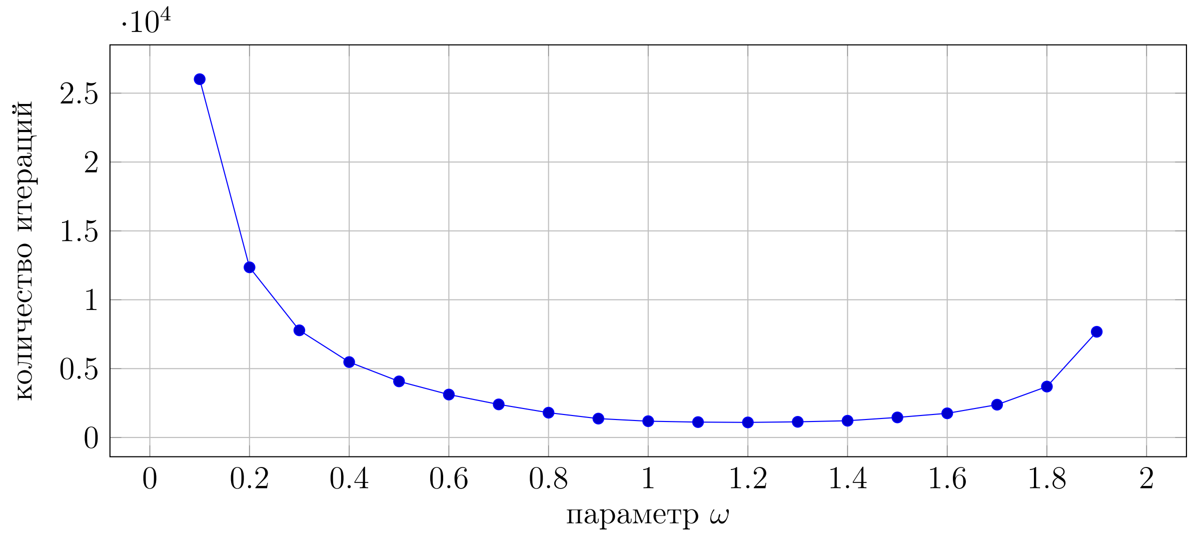
Приложение 1. Вариант 1.

На графиках представлена зависимость количества итераций от значения параметра ω, пробегающем отрезок $[0.1,\ 1.9]$ с шагом 0.1.

Первая система

$$A = \begin{pmatrix} 2 & 2 & -1 & 1 \\ 4 & 3 & -1 & 2 \\ 8 & 5 & -3 & 4 \\ 3 & 3 & -2 & 4 \end{pmatrix}, f = \begin{pmatrix} 4 \\ 6 \\ 12 \\ 6 \end{pmatrix}$$

Точность: $\varepsilon = 10^{-10}$
Решение: $x = (0.6,\ 1.0,\ -1.0,\ -0.2)^T$



Наилучшая сходимость наблюдается при $\omega = 1.2$

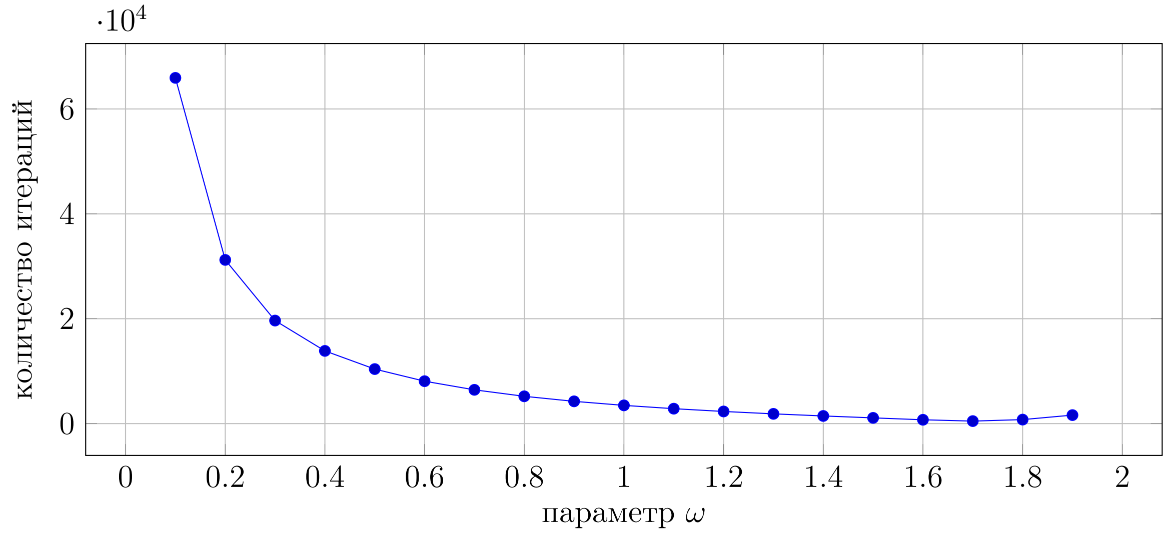
Вторая система

Внимание! Поскольку матрица, заданная вариантом, была вырождена, в матрице коэффициентов был изменен элемент $a_{32} = 4$ (прежнее значение: 3)

$$A = \begin{pmatrix} 1 & 1 & 3 & -2 \\ 2 & 2 & 4 & -1 \\ 3 & 4 & 5 & -2 \\ 2 & 2 & 8 & -3 \end{pmatrix}, f = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 2 \end{pmatrix}$$

Точность: $\varepsilon = 10^{-10}$
Решение: $x = (3.00000000e{+}00, -2.00000000e{+}00, 2.02239287e{-}12, -1.07528979e-$

$11)^T$

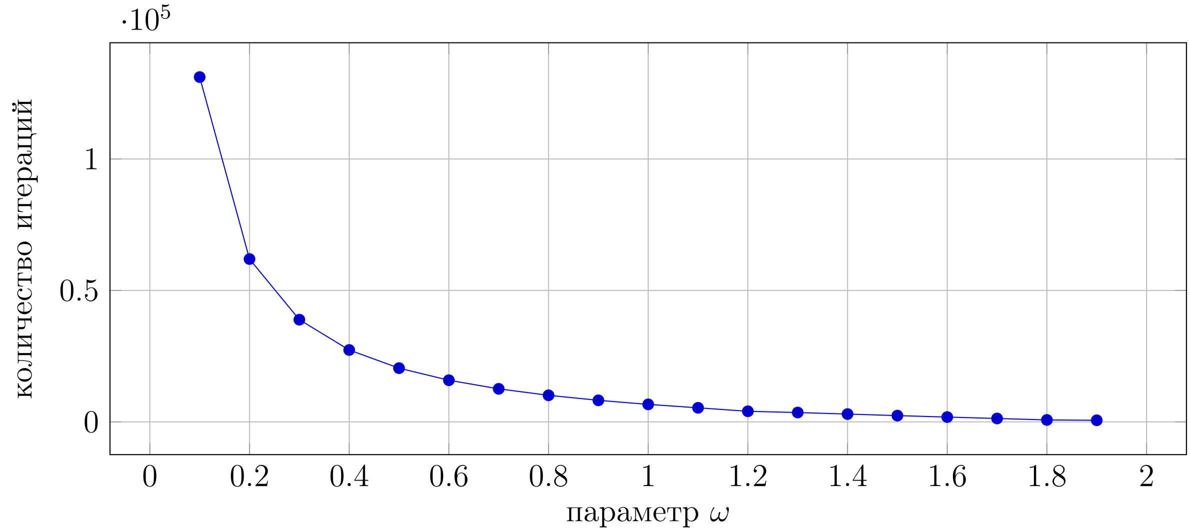


Наилучшая сходимость наблюдается при $\omega = 1.7$

Третья система

$$A = \begin{pmatrix} 2 & 5 & -8 & 3 \\ 4 & 3 & -9 & 1 \\ 2 & 3 & -5 & -6 \\ 1 & 8 & -7 & 0 \end{pmatrix}, f = \begin{pmatrix} 8 \\ 9 \\ 7 \\ 12 \end{pmatrix}$$

Точность: $\varepsilon = 10^{-10}$
Решение: $x = (3.00000000e{+}00, 2.00000000e{+}00, 1.00000000e{+}00, -3.94845977e{-}11)^T$



Наилучшая сходимость наблюдается при $\omega = 1.9$

Приложение 2. Вариант 2-4

Матрица задана следующим образом:

$$A_{ij} = \begin{cases} \mathrm{q}_M^{i+j} + 0.1 \cdot (j - i), & i \neq j \\ (\mathrm{q}_M - 1)^{i+j}, & i = j \end{cases},$$

где $M = 4, n = 100, \quad q_M = 1.001 - 2 \cdot M \cdot 10^{-3}, \quad f = \begin{pmatrix} f_1 \\ f_2 \\ ... \\ f_n \end{pmatrix}$, где $f_i = n \cdot e^{\frac{x}{i}} \cdot \cos x$

Вычисление решения проводилось при $x = 1$

В связи с высоким порядком матрицы (100) вычисление приблизительного решения СЛАУ на ПК с удовлетворительной точностью для нескольких параметров ω не представилось возможным. Были построены графики зависимости количества итераций от параметра ω для точности порядков > 0.01, и, исходя из них, оптимальное значение итерационного параметра выбрано следующим: $\omega = 0.17$

Точность: $\varepsilon = 10^{-6}$

Решение:

x = (-14.789163, 0.016967, 0.14373, 0.218334, 0.265753, 0.298357, 0.322124, 0.340211, 0.354427, 0.365875, 0.375265, 0.383074, 0.389633, 0.395177, 0.39988, 0.40387, 0.407246, 0.410083, 0.41244, 0.414362, 0.415886, 0.41704, 0.417849, 0.418329, 0.418497, 0.418363, 0.417935, 0.417223, 0.416229, 0.414958, 0.413413, 0.411594, 0.409502, 0.407137, 0.404497, 0.40158, 0.398385, 0.394907, 0.391144, 0.387092, 0.382746, 0.378102, 0.373156, 0.3679, 0.362331, 0.356442, 0.350227, 0.343679, 0.336792, 0.329559, 0.321972, 0.314024, 0.305708, 0.297015, 0.287938, 0.278467, 0.268595, 0.258311, 0.247608, 0.236476, 0.224905, 0.212886, 0.200408, 0.187461, 0.174035, 0.160118, 0.145701, 0.130771, 0.115318, 0.099329, 0.082792, 0.065695, 0.048026, 0.029772, 0.01092, -0.008544, -0.028633, -0.049361, -0.070742, -0.09279, -0.115521, -0.138949, -0.16309, -0.18796, -0.213573, -0.239948, -0.267099, -0.295045, -0.323803, -0.353389, -0.383823, -0.415122, -0.447305, -0.480391, -0.514399, -0.54935, -0.585264, -0.62216, -0.660061, -0.698987)^T

Количество итераций: 216443661

Выводы

В ходе работы был изучен итерационный метод верхней релаксации решения СЛАУ. Разработан критерий остановки итерационного процесса, гарантирующий получение приближенного решения исходной СЛАУ с заданной точнстью. На вариантах задания изучена зависимость скорости сходимости метода от итерационного параметра ω, найдены оптимальные значения ω.
Среди достоинств метода верхней релаксации можно выделить простоту реализации на ЭВМ, а также возможность задать итерационный параметр (во второй системе максимальное количество итераций - 65927, минимальное - 474, отличаются в более чем 100 раз). Так, можно найти оптимальный параметр для одной системы и использовать его для родственных систем.