МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В. Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа № 1**

по дисциплине: Вычислительная математика

тема: «Решение систем линейных алгебраических уравнений (СЛАУ)»

Выполнил: ст. группы ПВ-223

Игнатьев Артур Олегович

Проверил:

асс. Четвертухин Виктор Романович

Белгород 2024г.

**Лабораторная работа №1**

**«Решение систем линейных алгебраических уравнений (СЛАУ)»**

**Цель работы:** Изучить методы решения СЛАУ и особенности их алгоритмизации в современных программных библиотеках NumPy, SciPy языка Python.

Вариант 3

Изображение выглядит как текст, Шрифт, белый, чек

Автоматически созданное описание

**Ход выполнения лабораторной работы:**

1) Решение СЛАУ вручную по классическому методу Гаусса.

Изображение выглядит как текст, рукописный текст, блокнот, бумага

Автоматически созданное описание

2) Написание и выполнение коротких программ на языке Python для решения той же СЛАУ (своего индивидуального задания) с использованием разных алгоритмических техник в интерактивном блокноте Jupyter.

Код программы:

'''

Использование библиотеки NumPy

Основным методом, используемым в numpy.linalg.solve

является метод LU-разложения (LU decomposition) и его вариации

'''

import numpy as np

# Коэффициенты системы уравнений

A = np.array([[58, -63, 42],

              [84, -23, 32],

              [39, -48, 65]], dtype=float)

# Вектор свободных членов

b = np.array([-15, 10, -70], dtype=float)

# Решение системы

x = np.linalg.solve(A, b)

print("Решение системы с использованием numpy.linalg.solve:", x)

'''

Использование библиотеки SciPy

SciPy предлагает scipy.linalg.lu\_solve

для решения системы с помощью LU-разложения

'''

from scipy.linalg import lu\_factor, lu\_solve

lu, piv = lu\_factor(A)

x\_lu = lu\_solve((lu, piv), b)

print("Решение системы с использованием scipy.linalg:", x\_lu)

# Программы, демонстрирующие алгоритмическую реализацию:

# а) рассмотренного на лекции метода Гаусса, основанного на преобразовании

# исходной системы к верхнетреугольной форме с последующим обратным ходом для нахождения решений;

# б) улучшенного метода Гаусса с частичным выбором ведущего элемента

# (метод улучшает точность вычислений за счет минимизации ошибок округления, выбирая в качестве ведущего

# элемента максимальный по модулю в текущем столбце);

# в) решения СЛАУ с помощью LU- разложения матрицы.

import numpy as np

# Коэффициенты системы уравнений

A = np.array([[ 4.2, 1.5, -2.1],

              [ 7.1,-4.8, 3.2],

              [-8.1, 0.3, -3.8]], dtype=float)

# Вектор свободных членов

b = np.array([2.0, -3.2, 0.1], dtype=float)

def gauss(A, b):

    """

    Решение СЛАУ методом Гаусса.

    Метод Гаусса — классический алгоритм решения СЛАУ,

    основанный на преобразовании исходной системы к

    верхнетреугольной форме с последующим обратным ходом

    для нахождения решений.

    Параметры:

    - A (numpy.ndarray): Матрица коэффициентов СЛАУ.

    Должна быть квадратной (n x n) и невырожденной.

    - b (numpy.ndarray): Вектор свободных членов СЛАУ.

    Должен быть длины n, где n - количество уравнений в системе.

    Возвращает:

    - numpy.ndarray: Вектор решения x системы Ax = b.

    Имеет размерность n, соответствующую числу уравнений в системе.

    """

    numEquations = len(b)

    # Прямой ход

    for pivotRow in range(numEquations):

        for currentRow in range(pivotRow + 1, numEquations):

            factor = A[currentRow, pivotRow] / A[pivotRow, pivotRow]

            for currentCol in range(pivotRow, numEquations):

                A[currentRow, currentCol] -= factor \* A[pivotRow, currentCol]

            b[currentRow] -= factor \* b[pivotRow]

    # Обратный ход

    solutionVector = np.zeros(numEquations)

    for currentRow in range(numEquations - 1, -1, -1):

        sum\_ax = 0

        for currentCol in range(currentRow + 1, numEquations):

            sum\_ax += A[currentRow, currentCol] \* solutionVector[currentCol]

        solutionVector[currentRow] = (b[currentRow] - sum\_ax) / A[currentRow, currentRow]

    return solutionVector

print(gauss(A.copy(), b.copy()))

def gauss\_elimination\_with\_partial\_pivoting(matrix, vector):

    """

    Решает СЛАУ методом Гаусса с частичным выбором

    ведущего элемента. Этот метод улучшает точность вычислений

    за счет минимизации ошибок округления, выбирая в качестве

    ведущего элемента максимальный по модулю в текущем столбце.

    Параметры:

    - matrix (numpy.ndarray): квадратная матрица коэффициентов СЛАУ.

    - vector (numpy.ndarray): вектор свободных членов СЛАУ.

    Возвращает:

    - numpy.ndarray: вектор решения СЛАУ.

    """

    matrix\_size = len(matrix)

    # Прямой ход

    for current\_column in range(matrix\_size):

        # Поиск максимального элемента в текущем столбце

        max\_index = np.argmax(np.abs(matrix[current\_column:, current\_column])) + current\_column

        # Обмен строк в матрице и векторе свободных членов

        matrix[[current\_column, max\_index]], vector[[current\_column, max\_index]] = \

            (matrix[[max\_index, current\_column]], vector[[max\_index, current\_column]])

        for i in range(current\_column + 1, matrix\_size):

            factor = matrix[i][current\_column] / matrix[current\_column][current\_column]

            matrix[i, current\_column:] -= factor \* matrix[current\_column, current\_column:]

            vector[i] -= factor \* vector[current\_column]

    # Обратный ход

    solution = np.zeros(matrix\_size)

    for i in range(matrix\_size - 1, -1, -1):

        solution[i] = (vector[i] - np.dot(matrix[i, i + 1:], solution[i + 1:])) / matrix[i][i]

    return solution

print(gauss\_elimination\_with\_partial\_pivoting(A, b))

def lu\_decomposition(matrix, vector):

    """

    Выполняет LU-разложение матрицы и решает СЛАУ с помощью

    этого разложения.

    LU-разложение преобразует матрицу A в произведение

    двух матриц: L (нижнетреугольная) и U (верхнетреугольная),

    что облегчает решение СЛАУ.

    Параметры:

    - matrix (numpy.ndarray): квадратная матрица коэффициентов СЛАУ.

    - vector (numpy.ndarray): вектор свободных членов СЛАУ.

    Возвращает:

    - numpy.ndarray: вектор решения СЛАУ.

    """

    matrix\_size = len(matrix)

    L = np.zeros((matrix\_size, matrix\_size))

    U = np.zeros((matrix\_size, matrix\_size))

    # LU разложение

    for row in range(matrix\_size):

        L[row, row] = 1

        for col in range(row, matrix\_size):

            sum\_upper = sum(L[row, sum\_index] \* U[sum\_index, col] for sum\_index in range(row))

            U[row, col] = matrix[row, col] - sum\_upper

        for col in range(row + 1, matrix\_size):

            sum\_lower = sum(L[col, sum\_index] \* U[sum\_index, row] for sum\_index in range(row))

            L[col, row] = (matrix[col, row] - sum\_lower) / U[row, row]

    # Решение Ly = b для y

    y = np.zeros(matrix\_size)

    for row in range(matrix\_size):

        y[row] = vector[row] - np.dot(L[row, :row], y[:row])

    # Решение Ux = y для x

    x = np.zeros(matrix\_size)

    for row in range(matrix\_size - 1, -1, -1):

        x[row] = (y[row] - np.dot(U[row, row + 1:], x[row + 1:])) / U[row, row]

    return x

print(lu\_decomposition(A, b))

Результат выполнения:

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Таким образом, решение системы методом Гаусса даёт нам значения , , , которые соответствуют результатам, полученным решением вручную и с использованием библиотек NumPy и SciPy.

3) Изменить параметры своего индивидуального задания так, чтобы представленные программы не справлялись с решением (полученное решение являлось бы неточным, либо программа не смогла бы получить решение и завершилась бы с ошибкой). Изучить типы полученных ошибок и проинтерпретировать численные ситуации.

Код программы:

# Коэффициенты системы уравнений

A = np.array([[58, -63, 42],

              [840, -230, 320],

              [3900, -4800, 6500]], dtype=float)

# Вектор свободных членов

b = np.array([-15, 100, -70000], dtype=float)

Результат выполнения:

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Проанализировав результаты, предоставленные библиотеками NumPy и SciPy, и учитывая рассмотренные аспекты, можем сделать выводы о том, что полученное решение не является точным и, вероятно, связано с численными ошибками, возникшими из-за сингулярности матрицы, плохой обусловленности или других факторов.

**Вывод:** в ходе выполнения лабораторной работы были изучены методы решения СЛАУ и особенности их алгоритмизации в современных программных библиотеках NumPy, SciPy языка Python.