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

3.

$$\begin{cases} 58x_1 - 63x_2 + 42x_3 = -15 \\ 84x_1 - 23x_2 + 32x_3 = 10 \\ 39x_1 - 48x_2 + 65x_3 = -70 \end{cases}$$

Ход выполнения лабораторной работы:

1) Решение СЛАУ вручную по классическому методу Гаусса.

$$\begin{cases} 58x_1 - 63x_2 + 42x_3 = -15 \\ 84x_1 - 23x_2 + 32x_3 = 10 \\ 39x_1 - 48x_2 + 65x_3 = -70 \end{cases}$$

$$\left[\begin{array}{ccc|c} 58 & -63 & 42 & -15 \\ 84 & -23 & 32 & 10 \\ 39 & -48 & 65 & -70 \end{array}\right]$$

$$\left[\begin{array}{ccc|c} 1 & -\frac{63}{58} & \frac{42}{58} & -\frac{15}{58} \\ 84 & -23 & 32 & 10 \\ 39 & -48 & 65 & -70 \end{array}\right]$$

$$\left[\begin{array}{ccc|c} 1 & -\frac{63}{58} & \frac{42}{58} & -\frac{15}{58} \\ 0 & 58\cdot\left(-\frac{63}{58}\right)+84 & 58\cdot\left(\frac{42}{58}\right)+(-23) & 58\cdot\left(-\frac{15}{58}\right)+10 \\ 0 & 39\cdot\left(-\frac{63}{58}\right)+39 & 39\cdot\left(\frac{42}{58}\right)+(-48) & 39\cdot\left(-\frac{15}{58}\right)+(-70) \end{array}\right]$$

$x_1 = 0,11766805$

$x_2 = 0,69240325$

$x_3 = -0,22244111$

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 округле-
ния, 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, cur-
rentCol]
            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[cur-
rentCol]
        solutionVector[currentRow] = (b[currentRow] - sum_ax) / A[curren-
tRow, 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_col-
umn])) + 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, cur-
rent_column]])
        for i in range(current_column + 1, matrix_size):
            factor = matrix[i][current_column] / matrix[current_col-
umn][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:

```
Решение системы с использованием numpy.linalg.solve: [ 0.68846278 -0.23915216 -1.66660542]
Решение системы с использованием scipy.linalg: [ 0.68846278 -0.23915216 -1.66660542]
[ 0.11766805  0.69240325 -0.22247111]
[ 0.11766805  0.69240325 -0.22247111]
[ 0.11766805  0.69240325 -0.22247111]
```

Таким образом, решение системы методом Гаусса даёт нам значения $x_1 = 0.11766805$, $x_2 = 0.69240325$, $x_3 = -0.22247111$, 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:

```
Решение системы с использованием numpy.linalg.solve: [  5.55014552  -7.98080315 -19.99283425]
Решение системы с использованием scipy.linalg: [  5.55014552  -7.98080315 -19.99283425]
[ 0.11766805  0.69240325 -0.22247111]
[ 0.11766805  0.69240325 -0.22247111]
[ 0.11766805  0.69240325 -0.22247111]
```

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.