МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра вычислительной математики

Отчёт

Лабораторная работа

“ Метод исключения Гаусса без выбора главного элемента”

Вариант №5

Благодарного Артёма Андреевича
студента 3 курса, 3 группы
специальности «Информатика»
дисциплина «Численные методы»

Минск, 2024

Постановка задачи

Найти решение системы линейных алгебраических уравнений $Ax = b$, используя Метод исключения Гаусса без выбора главного элемента, где:

Matrix A

0.8894	0.0000	−0.2323	0.1634	0.2723
−0.0545	0.5808	0.0000	−0.1107	0.0363
0.0182	−0.1634	1.0527	0.0200	0.0635
0.0545	0.0000	−0.1325	1.0527	0.0000
0.0363	−0.0545	0.2632	−0.0218	0.7623

Matrix b

4.2326
−4.1037
−2.6935
1.6916
3.1908

Необходимо:

1. Применить метод исключения Гаусса без выбора главного элемента для решения системы.
2. Найти определитель матрицы А
3. Найти обратную матрицу A^{-1}
4. Найти число обусловленности ν(А).
5. Найти матрицу невязки $R = E - A^{-1}A$
6. Найти вектор невязки $r = Ax - b$

Алгоритм решения

1. Метод Гаусса с выбором главного элемента по всей матрице:

Метод исключения Гаусса без выбора главного элемента, называемый просто методом Гаусса, состоит в том, что сначала система уравнений приводится к треугольному виду элементарными строчными преобразованиями без использования перестановки строк и/или столбцов (прямой ход), а затем неизвестные выражаются из полученной системы (обратный ход).

Прямой ход (переход к треугольной системе):

1. **Исключение неизвестных на каждом шаге**:
   - На шаге k, начиная с первого уравнения, исключается неизвестное x_k из всех последующих уравнений $x_{k+1}, x_{k+2}, \dots, x_n$.

   Основные формулы для исключения неизвестных:
   - $l_{ik} = \frac{a_{ik}}{a_{kk}}, \quad i = k+1, k+2, \dots, n$ (формула для коэффициента исключения l_{ik})
   - $a_{ij}^{(k)} = a_{ij}^{(k-1)} - l_{ik} a_{kj}^{(k-1)}, \quad j = k+1, k+2, \dots, n$ (обновление коэффициентов матрицы на шаге k)
   - $b_i^{(k)} = b_i^{(k-1)} - l_{ik} b_k^{(k-1)}$ (обновление столбца свободных членов на шаге k)
2. **Преобразованная система** на каждом шаге превращается в систему с треугольной матрицей, где все элементы ниже главной диагонали равны нулю.

Обратный ход (нахождение неизвестных):

1. Вычисление неизвестных из треугольной системы:
   - Из последнего уравнения системы вычисляем x_n:

$$x_n = \frac{b_n}{a_{nn}}$$

   - Подставляя x_n в предпоследнее уравнение, находим x_{n-1}, затем x_{n-2} и так далее:

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij} x_j}{a_{ii}}, \quad i = n-1, n-2, \dots, 1$$

2. Определитель матрицы А

Так как мы сводим матрицу А к треугольному виду, то:

$$detA = a_{11} a_{12} \cdot \ldots \cdot a_{nn}$$

3. Обратная матрица A^{-1}:

Чтобы найти матрицу A^{-1}, обратную к А, надо решить матричное уравнение AX = E.

4. Число обусловленности $v(A)$:

$$v(A) = ||A|| \cdot ||A^{-1}||$$

где норма определяется как максимум суммы абсолютных значений элементов в каждом столбце.

5. Матрица невязки:

$$R = E - A^{-1}A$$

6. Вычисление вектора невязки:

$$r = Ax - b$$

Листинг кода

```
import numpy as np

def gauss_elimination(A, b):
    """
    Метод Гаусса для решения системы уравнений Ax = b.
    Параметры:
    A (ndarray): Матрица коэффициентов.
    b (ndarray): Вектор свободных членов.

    Возвращает:
    x (ndarray): Вектор решения.
    """
    n = len(b)

    # Преобразуем систему к верхнетреугольному виду
    for i in range(n):
        for j in range(i + 1, n):
            if A[j, i] != 0:  # Проверяем, что элемент не равен нулю
                ratio = A[j, i] / A[i, i]
                A[j, i:] -= ratio * A[i, i:]
                b[j] -= ratio * b[i]

    # Обратная подстановка для нахождения решения
    x = np.zeros(n)
    for i in range(n - 1, -1, -1):
        x[i] = (b[i] - np.dot(A[i, i + 1:], x[i + 1:])) / A[i, i]

    return x

def compute_determinant(A):
    """
    Вычисляет определитель матрицы A.

    Параметры:
    A (ndarray): Матрица коэффициентов.

    Возвращает:
    det (float): Определитель матрицы A.
    """
    det = 1
    # Определитель равен произведению диагональных элементов
    for i in range(len(A)):
        det *= A[i, i]
    return det


def compute_inverse(A):
    n = len(A)
```

```python
    A = A.astype(float)  # Преобразуем A к типу float для корректного
деления
    b = np.eye(n)  # Единичная матрица для хранения обратной матрицы

    # Преобразование системы к верхнетреугольному виду
    for i in range(n):
        # Нормализуем строку, деля на диагональный элемент
        b[i] /= A[i, i]
        A[i] /= A[i, i]

        for j in range(i + 1, n):
            if A[j, i] != 0:  # Проверяем, что элемент не равен нулю
                ratio = A[j, i] / A[i, i]
                A[j, i:] -= ratio * A[i, i:]
                b[j] -= ratio * b[i]

    # Обратный ход для получения диагональной матрицы
    for i in range(n - 1, -1, -1):
        for j in range(i - 1, -1, -1):
            ratio = A[j, i]
            A[j, i:] -= ratio * A[i, i:]
            b[j] -= ratio * b[i]

    return b
def compute_condition_number(A):
    """
    Вычисляет число обусловленности матрицы A.

    Параметры:
    A (ndarray): Матрица коэффициентов.

    Возвращает:
    condition_number (float): Число обусловленности матрицы A.
    """
    A_inv = compute_inverse(A)
    norm_A = np.max(np.sum(np.abs(A), axis=0))  # Норма 1
    norm_A_inv = np.max(np.sum(np.abs(A_inv), axis=0))
    return norm_A * norm_A_inv

def compute_residual_matrix(A):
    """
    Вычисляет матрицу невязки R = E - A^(-1) * A.

    Параметры:
    A (ndarray): Матрица коэффициентов.

    Возвращает:
    R (ndarray): Матрица невязки.
    """
    A_inv = compute_inverse(A)
```

```
    return np.eye(A.shape[0]) - A_inv @ A

def compute_residual_vector(A, x, b):
    """
    Вычисляет вектор невязки r = Ax - b.

    Параметры:
    A (ndarray): Матрица коэффициентов.
    x (ndarray): Вектор решения.
    b (ndarray): Вектор свободных членов.

    Возвращает:
    r (ndarray): Вектор невязки.
    """
    return A @ x - b
```

Результаты

Решение системы $Ax = b$:

$$x = \begin{pmatrix} 1.99963133 \\ -6.99987764 \\ -4.00034508 \\ 0.99988066 \\ 4.99987967 \end{pmatrix}$$

Определитель матрицы A:

```
det(A)= 0.4177139291529895
```

Обратная матрица A^{-1}:

```
[[ 1.1503  0.0544  0.3406 -0.1885 -0.4419]
 [ 0.0997  1.7308  0.0735  0.1626 -0.1241]
 [-0.0003  0.2653  0.9742  0.0075 -0.0937]
 [-0.0596  0.0306  0.1050  0.9606  0.0111]
 [-0.0492  0.0304 -0.3443  0.0455  1.3566]]
```

Число обусловленности ν(A):

```
2.260275166964685
```

Матрица невязки R:

$$R = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 & 2.22 \times 10^{-16} & -4.44 \times 10^{-16} \\ 0.0000 & 0.0000 & 0.0000 & 2.22 \times 10^{-16} & -4.44 \times 10^{-16} \\ 0.0000 & 0.0000 & 0.0000 & 2.22 \times 10^{-16} & -4.44 \times 10^{-16} \\ 0.0000 & 0.0000 & 0.0000 & 2.22 \times 10^{-16} & -4.44 \times 10^{-16} \\ 0.0000 & 0.0000 & 0.0000 & 2.22 \times 10^{-16} & -4.44 \times 10^{-16} \end{bmatrix}$$

Вектор невязки r:

$$r = \begin{bmatrix} 0.0000 \\ 0.0000 \\ 0.0000 \\ 2.22 \times 10^{-16} \\ -4.44 \times 10^{-16} \end{bmatrix}$$

Анализ

Число обусловленности невелико ($<10^2$), что свидетельствует о хорошей устойчивости системы. Вектор и матрица невязки имеют значения, близкие к нулю ($<10^{16}$), указывая на высокую точность решения. В целом, малые значения этих невязок подтверждают, что полученное решение очень близко к точному. Однако из-за погрешностей, возникающих при вычислениях с плавающей точкой, некоторые значения могут незначительно отличаться от нуля. Это естественная особенность таких вычислений и должна учитываться при интерпретации результатов.