Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 2

по курсу «Численные методы линейной алгебры»

«Реализация метода Гаусса и оценка погрешностей вычислений»

Студент группы ИУ9-71Б Яровикова А. С.

Преподаватель Посевин Д. П.

Москва 2023

1 Цель работы

Реализовать простейший метод Гаусса и научиться оценивать погрешности решения системы уравнения для матриц произвольной размерности.

2 Задание

1. Реализовать метод Гаусса для действительных квадратных матриц произвольной размерности n. Возможноть быстро менять размерность матрицы n в дальнейшем потребуется для проведения численных экспериментов по оценке скорости выполнения алгоритма и его точности.

2. Реализовать возможность ручного ввода элементов матрицы произвольной размерности.

3. Реализовать возможность генерации матриц со случайными элементами в заданном диапазоне $[-a, b]$, где a и b принадлежат R. При этом необходимо уметь регулировать условие диагонального преобладания, другими словами реализовать возможность принудительного увеличения на заданный порядок среднее значение генерируемых диагональных элементов a_{ii} матрицы A системы уравнений $A * x = b$.

4. Реализовать алгоритм тестирования задачи, который заключается в том, что мы заведомо определяем значения координат вектора x, данный вектор заведомо является решением уравнения $A * x = b$, вычисляем b путем прямого перемножения матрицы A на вектор x и далее производим поиск решения уравнения $A * x = b$ методом Гаусса, получая x_{chisl}. После этого производим сравнение полученного x_{chisl} с заданным x, а также решением хбибл , полученным с использованием сторонней библиотеки выбранной студеном. При этом сравнение производится по евклидовой норме разности вектора $(x - x_{chisl})$ и $(x - x_{bibl}))$.

3 Реализация

Исходный код программы представлен в листингах 1– 8.

Листинг 1 — Вспомогательные функции Евклидовой нормы и умножения матрицы на вектор

```
import math

def euclidean_norm(vec):
    res = 0
    for el in vec:
        res += el**2
    return math.sqrt(res)

def mul_on_vector(matrix, vector):
    res = []
    for i in range(len(matrix)):
        el = 0
        for j in range(len(vector)):
            el += matrix[i][j] * vector[j]
        res.append(el)
    return res
```

Листинг 2 — метод Гаусса

```
def gauss(matrix):
    n = len(matrix)
    x = [0 for i in range(n)]
    A = copy.deepcopy(matrix)
    # towards
    for i in range(n):
        for j in range(i + 1, n):
            c = A[j][i] / A[i][i]
            for k in range(n + 1):
                A[j][k] = A[j][k] - c * A[i][k]
    x[n - 1] = A[n - 1][n] / A[n - 1][n - 1]
    # backwards
    for i in range(n - 2, -1, -1):
        x[i] = A[i][n]
        for j in range(i + 1, n):
            x[i] = x[i] - A[i][j] * x[j]
        x[i] = x[i] / A[i][i]
    return x
```

Листинг 3 — Метод ручного ввода элементов матрицы произвольной размерности

```
def manual_fill_matrix(n):
    matrix = [[0 for _ in range(n)] for _ in range(n)]
    for i in range(n):
        for j in range(n):
            matrix[i][j] = float(input( 'a['+str(i)+']['+ str(j)+'] ='))
    return matrix
```

Листинг 4 — Метод генерации матриц со случайными элементами в заданном диапазоне

```
def generate_matrix(n, a, b):
    matrix = [[random.uniform(a, b) for i in range(n)] for j in range(n)
    ]
    return matrix
```

Листинг 5 — Метод принудительного увеличения на заданный порядок значение диагональных элементов матрицы

```
def increase_diag_elems(a, p, pow):
    for i in range(0, len(a)):
        a[i][i] = abs(a[i][i] + (p**pow))
    return a
```

Листинг 6 — Метод сравнения двух векторов по Евклидовой норме

```
def get_diff(x1, x2):
    x = []
    for i in range(0, len(x1)):
        x.append(abs(x1[i] - x2[i]))
    return euclidean_norm(x)
```

Листинг 7 — Метод печати матрицы

```
def print_matrix(a):
    for i in range(len(a)):
        print(a[i])
```

Листинг 8 — Метод тестирования алгоритма

```
def testing(x):
    n = 100
    l = 1
    r = 5

    a = generate_matrix(n, l, r)
    print("matrix A:")
    print_matrix(a)
    b = mul_on_vector(a, x)
    print("\nvector b:", b)
    A = np.column_stack((a, b))
    x_calc = gauss(A)
    print("\nx_calc:", x_calc)
    x_lib = np.linalg.solve(a, b)
    print("\nx_lib:", x_lib)
    print(f'\nEuclid ||x - x_chisl||: {get_diff(x, x_calc)}')
    print(f'Euclid ||x - x_bibl||: {get_diff(x, x_lib)}')

    print("\n\n\nmatrix A with increased diagonal elements:")
    a = increase_diag_elems(a, 2, 5)
    print_matrix(a)
    b = mul_on_vector(a, x)
    print("\nvector b:", b)
    A = np.column_stack((a, b))
    x_calc = gauss(A)
    print("\nx_calc:", x_calc)
    x_lib = np.linalg.solve(a, b)
    print("\nx_lib:", x_lib)
    print(f'\nEuclid ||x - x_chisl||: {get_diff(x, x_calc)}')
    print(f'Euclid ||x - x_bibl||: {get_diff(x, x_lib)}')

# test
x = [1] * 100
testing(x)
```

4 Результаты

Результат запуска методов представлены на рисунках 1 - 3.

```
mat = generate_matrix(5, -5, 5)
print('\ngenerated matrix:\n')
print_matrix(mat)

print('\ngenerated matrix with increased diag elems:\n')
print_matrix(increase_diag_elems(mat,10, 2))
# mat = manual_fill_matrix(3)
# print(mat)
```

```
generated matrix:

[-3.845630178966659, -3.260805306799477, -4.707629283463547, 0.29875805492088325, -3.5604476158114604]
[2.940237721266027, 2.3902518222198124, 0.8214335151499084, -2.9811186604903126, -3.60472606286401]
[-3.23018430780229, -1.1962314884330771, -4.533903338955697, 0.6319948872407597, 1.9144062301269837]
[2.047675426019727, -1.9701390410199213, -1.3264686719186658, 3.9236463414991167, -4.166272140182258]
[-4.8612206574883405, -4.705713989653032, 2.4682764599620324, -3.725327668915522, 3.56945704758083]

generated matrix with increased diag elems:

[96.15436982103334, -3.260805306799477, -4.707629283463547, 0.29875805492088325, -3.5604476158114604]
[2.940237721266027, 102.39025182221981, 0.8214335151499084, -2.9811186604903126, -3.60472606286401]
[-3.23018430780229, -1.1962314884330771, 95.4660966610443, 0.6319948872407597, 1.9144062301269837]
[2.047675426019727, -1.9701390410199213, -1.3264686719186658, 103.92364634149912, -4.166272140182258]
[-4.8612206574883405, -4.705713989653032, 2.4682764599620324, -3.725327668915522, 103.56945704758083]
```

Рис. 1 — Сгенерированные матрицы размерностью 5x5 без диагонального преобладания и с диагональным преобладанием

```
Euclid ||x - x_числ||: 1.0570127614280218e-11
Euclid ||x - x_библ||: 9.450456136253299e-12
```

Рис. 2 — Евклидовы нормы разностей векторов $(x - x_{chisl})$ и $(x - x_{bibl})$ для матрицы размерности 100x100

```
Euclid ||x - x_числ||: 3.715201732147977e-14
Euclid ||x - x_библ||: 3.5317309504984386e-14
```

Рис. 3 — Евклидовы нормы разностей векторов $(x - x_{chisl})$ и $(x - x_{bibl})$ для матрицы размерности 100x100 с диагональным преобладанием

5 Выводы

В результте выполнения данной лабораторной работы был реализован метод Гаусса на языке программирования Python, была оценена по- грешность решения метода Гаусса и решения, полученного с помощью биб- лиотеки NumPy.