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»

Летучка № 1

по курсу «Численные методы линейной алгебры»

«Метод прогонки для трехдиагональной матрицы.»

Студент группы ИУ9-71Б Яровикова А. С.

Преподаватель Посевин Д. П.

Москва 2023

1 Задание

Реализовать метод прогонки для уравнения $A\overrightarrow{x} = \overrightarrow{f}$, где $A \in R^{100\times100}$, $\overrightarrow{f} \in R^{100}$, $\overrightarrow{x} \in R^{100}$, A - трехдиагональная матрица.

Решить:

- методом Гаусса: $\|\Delta x\|_2 = \|\overrightarrow{x}_G - \overrightarrow{x}_{exact}\|$
- методом прогонки: $\|\Delta x\|_2 = \|\overrightarrow{x}_P - \overrightarrow{x}_{exact}\|$
- библиотечным методом: $\|\Delta x\|_2 = \|\overrightarrow{x}_B - \overrightarrow{x}_{exact}\|$

2 Реализация

Исходный код программы представлен в листингах 1– 5.

Листинг 1 — 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 2 — Метод прогонки

```
def progonka(a, b, c, d):
    n = len(b)
    x = [0 for i in range(n)]
    # forward
    alp = [-c[0] / b[0]]
    bet = [d[0] / b[0]]
    for i in range(1, n):
        if i != n - 1:
            y = a[i-1] * alp[i-1] + b[i]
            alp.append(-c[i] / y)
            bet.append((d[i]-a[i-1] * bet[i-1]) / y)
        else:
            y = a[n-2] * alp[n-2] + b[n-1]
            bet.append((d[n-1] - a[n-2] * bet[n-2]) / y)

    # backwards
    for i in reversed(range(n)):
      if i == n - 1:
        x[n - 1] = bet[n - 1]
      else:
        x[i] = alp[i] * x[i + 1] + bet[i]
    return x
```

Листинг 3 — Функции для создания трехдиагональной матрицы

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

def generate_3d_matrix(n, a, b, c):
    m = [[0 for i in range(n)] for j in range(n)]
    for i in range(n):
      for j in range(n):
        if (i == j):
          m[i][i] = b[i]
          if (i != n - 1):
            m[i][i+1] = c[i]
            m[i+1][i] = a[i]
    return m
```

Листинг 4 — Функции для сравнения векторов по Евклидовой норме

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

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 5 — Тестирование

```python
N = 100
x = [1] * N
bb = generate_diag(N, 1, 5)
aa = generate_diag(N - 1, 1, 5)
cc = generate_diag(N - 1, 1, 5)

a = generate_3d_matrix(N, aa, bb, cc)
d = mul_on_vector(a, x)
# print(f'\nf: {d}')

A = np.column_stack((a, copy.deepcopy(d)))
x_calc = gauss(A)
x_prog = progonka(aa, bb, cc, copy.deepcopy(d))
x_lib = np.linalg.solve(a, copy.deepcopy(d))

print(f'\ngauss: {get_diff(x, x_calc)*100} %')
print(f'\nprogonka: {get_diff(x, x_prog)*100} %')
print(f'\nlib: {get_diff(x, x_lib)*100} %')
```

3 Результаты

Результат запуска методов представлены на рисунке 1.

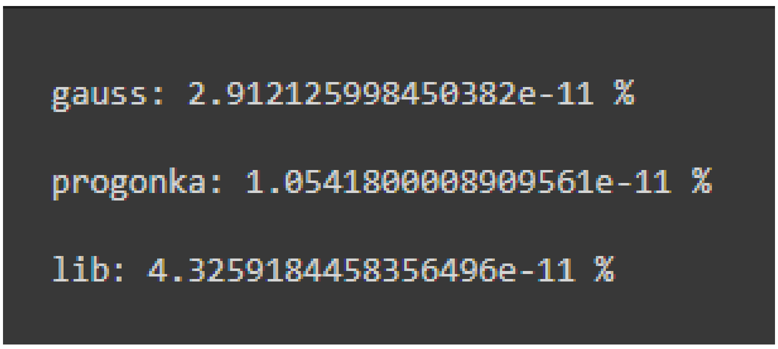

Рис. 1 — Сравнение методов

4 Выводы

В ходе лабораторной работы были исследованы три метода решения системы линейных уравнений с трехдиагональной матрицей A и вектором правой части b: классический метод Гаусса, метод прогонки для трехдиагональных матриц и использование библиотечной функции из NumPy. Были вычислены относительные ошибки каждого метода относительно точного решения x_{exact}.