Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет «Высшая школа экономики»

Образовательная программа
«Прикладная математика»

ОТЧЕТ
По лабораторной работе № 1

По предмету
«Численные методы»

По теме
«Теория погрешностей и машинная арифметика»

Выполнил
студент группы БПМ211
Кудряшов Максим Дмитриевич

Проверил
Брандышев Петр Евгеньевич

Москва - 2024

Содержание

1 3.1.8

1.1 Постановка задачи

Дана система уравнений $Ax = b$ порядка $n = 6$. Исследовать зависимость погрешности решения x от погрешностей правой части системы b.

$$A_{ij} = \frac{1}{\sqrt{c_{ij}^2 + 0.58c_{ij}}}$$

$$b_i = N$$

$$c_{ij} = 0.1Nij$$

$$N = 8$$

1.2 Теория

1.3 аналитическое решение тестового примера и результат вычислительного эксперимента по тесту

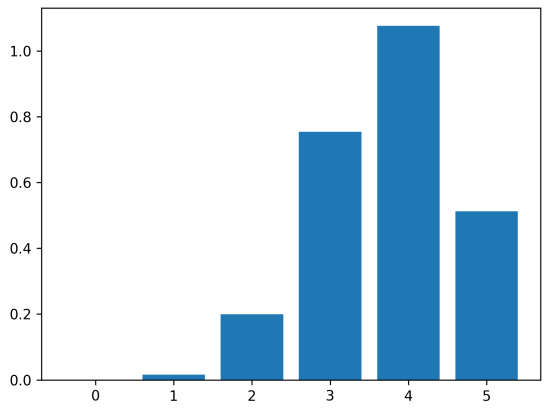
1.4 Код на Python

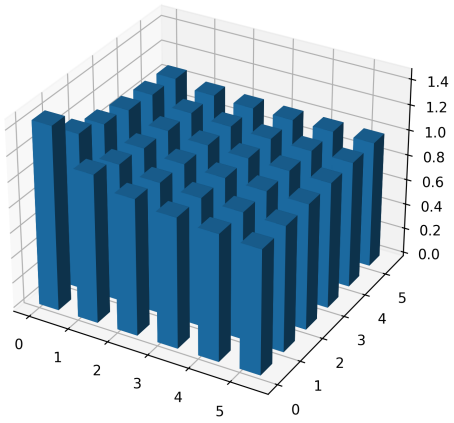
```python
import numpy as np
import matplotlib.pyplot as plt
import os
import pandas as pd

output_file_path = "3.1.8.txt"

n = 6
N = 8
delta = 0.05

os.remove(output_file_path)
with open(output_file_path, "a") as f:
    # заполним матрицу A
    A = np.empty((n, n))
    for i in range(n):
        for j in range(n):
            c = 0.1 * N * (i + 1) * (j + 1)
            A[i][j] = 1 / np.sqrt(c ** 2 + 0.58 * c)
    # print(f'A = {A}', file=f)

    # заполним вектор b
    b = np.full(n, N, dtype=float)

    # найдем лист из измененных векторов b
```

```
    b_error_list = []
    for i in range(n):
        b_error = b.copy()
        b_error[i] += delta
        b_error_list.append(b_error)

    # найдем решение через встроенную функцию
    x = np.linalg.solve(A, b)
    print(f"{x = }", file=f)

    # найдем число обусловленности через встроенную функцию
    A_cond = np.linalg.cond(A, np.inf)
    print(f"{A_cond = }", file=f)

    # найдем вектор d
    d = np.empty(n)
    for i in range(n):
        b_error = b_error_list[i]
        x_error = np.linalg.solve(A, b_error)
        d[i] = np.linalg.norm(x - x_error, ord=np.inf) / np.linalg.norm(x, ord=np.inf)
    print(f"{d = }", file=f)

    # найдем что оказывает наибольшее влияние на погрешность
    d_max = d.max()
    d_max_index = list(d).index(d_max)
    print(f'{d_max = } with index = {d_max_index}', file=f)

    # построим гистограмму
    plt.bar(range(n), d)
    plt.savefig('3.1.8.png', dpi=300)
    plt.show()

    # print(f"Сравнение погрешностей:", file=f)
    delta_rel_x_error_list = np.empty(n, dtype=float)
    for i in range(n):
        b_error = b_error_list[i]
        # delta_rel_b_old = np.linalg.norm(b - b_error, ord=np.inf) / np.linalg.norm(b, ord=np.inf)
        delta_rel_b = np.abs(delta) / np.linalg.norm(b, ord=np.inf)
        delta_rel_x_error = A_cond * delta_rel_b
        delta_rel_x_error_list[i] = delta_rel_x_error
        # print(f"практически: {d[i]},\tтеоритически: {delta_rel_x_error}", file=f)

        df = pd.DataFrame({
            'Практические': d,
            "Теоретические": delta_rel_x_error_list
        })
        df.to_latex("3.1.8.tex")

# with open(output_file_path, "r") as f:
#     print(f.read())
```

1.5 Результаты

```
x = array([ 2.28625963e+06, -1.97930833e+08,  2.39587644e+09, -8.96828858e+09,
        1.27233076e+10, -6.02721331e+09])
A_cond = 1678868976234.5142
d_for_b = array([1.75867282e-04, 1.61316244e-02, 1.99311489e-01, 7.53942029e-01,
       1.07646727e+00, 5.12130976e-01])
d_for_A = array([[1.45164313, 1.1693928 , 1.07560493, 1.02840626, 1.00000045,
        0.98102369],
       [1.23898367, 1.09095131, 1.04062713, 1.0152746 , 1.        ,
        0.98979009],
       [1.16321821, 1.06218071, 1.02778666, 1.01044896, 1.        ,
        0.99301434],
       [1.12391529, 1.04724248, 1.02111544, 1.00794103, 1.        ,
        0.99469056],
       [1.09987034, 1.03809273, 1.01702794, 1.00640416, 1.        ,
        0.99571792],
       [1.08363861, 1.03191264, 1.0142665 , 1.00536578, 1.        ,
        0.99641211]])
d_for_b_max = 1.0764672729236027 with index = 4
d_for_A_max = 1.4516431254007967 with index = [0, 0]
```





Индекс	Практические	Теоретические
0	0.000176	10492931101.465714
1	0.016132	10492931101.465714
2	0.199311	10492931101.465714
3	0.753942	10492931101.465714
4	1.076467	10492931101.465714
5	0.512131	10492931101.465714

i	Практические	Теоретические
(0, 0)	1.451643	32548247136.209515
(0, 1)	1.169393	32548247136.209515
(0, 2)	1.075605	32548247136.209515
(0, 3)	1.028406	32548247136.209515
(0, 4)	1.000000	32548247136.209515
(0, 5)	0.981024	32548247136.209515
(1, 0)	1.238984	32548247136.209515
(1, 1)	1.090951	32548247136.209515
(1, 2)	1.040627	32548247136.209515
(1, 3)	1.015275	32548247136.209515
(1, 4)	1.000000	32548247136.209515
(1, 5)	0.989790	32548247136.209515
(2, 0)	1.163218	32548247136.209515
(2, 1)	1.062181	32548247136.209515
(2, 2)	1.027787	32548247136.209515
(2, 3)	1.010449	32548247136.209515
(2, 4)	1.000000	32548247136.209515
(2, 5)	0.993014	32548247136.209515
(3, 0)	1.123915	32548247136.209515
(3, 1)	1.047242	32548247136.209515
(3, 2)	1.021115	32548247136.209515
(3, 3)	1.007941	32548247136.209515
(3, 4)	1.000000	32548247136.209515
(3, 5)	0.994691	32548247136.209515
(4, 0)	1.099870	32548247136.209515
(4, 1)	1.038093	32548247136.209515
(4, 2)	1.017028	32548247136.209515
(4, 3)	1.006404	32548247136.209515
(4, 4)	1.000000	32548247136.209515
(4, 5)	0.995718	32548247136.209515
(5, 0)	1.083639	32548247136.209515
(5, 1)	1.031913	32548247136.209515
(5, 2)	1.014267	32548247136.209515
(5, 3)	1.005366	32548247136.209515
(5, 4)	1.000000	32548247136.209515
(5, 5)	0.996412	32548247136.209515

2 3.2

3 3.8.2

3.1 Постановка задачи

Дана система уравнений $Az(x) = b(x)$ порядка n. Построить график функции $y(x) = \sum_{i=1}^{n} z_i(x)$ на отрезке $[a, b]$. Для решения системы написать и использовать метод Гаусса со схемой полного перебора.

3.2 Код на Python

Я реализовал два метода Гаусса. Сначала метод Гаусса для численно заданных матрицы и вектора, а потом для численно заданой матрицы, но при этом символьго заданного вектора b.

3.2.1 Метод Гаусса для числел

```
import numpy as np


def gauss_elimination_number(A: np.ndarray, b: np.array) -> np.array:
    n = A.shape[0]
    x = np.empty(n, dtype=float)

    # прямой ход
    for i in range(n):

        # ищем максимальный коэффициент среди уравнений с неизвестными x и берем его индекс
        A_slice = A[i:][:]
        max_element_row_index = np.unravel_index(np.argmax(A_slice), A_slice.shape)[0] + i

        # меняем местами i-ое и max_element_row_index-ое уравнения
        if max_element_row_index != i:
            A[[i, max_element_row_index]] = A[[max_element_row_index, i]]
            b[[i, max_element_row_index]] = b[[max_element_row_index, i]]

        # убираем коэффициенты при x
        for j in range(i + 1, n):
            mu = A[j][i] / A[i][i]
            A[j] -= mu * A[i]
            b[j] -= mu * b[i]

    # обратный ход
    for m in range(n - 1, -1, -1):
        numerator = b[m]  # числитель
        for l in range(m + 1, n):
            numerator -= A[m][l] * x[l]
        denominator = A[m][m]  # знаменатель
        x[m] = numerator / denominator

    return x

```

```
36
37 A = np.array([[2.0, 1.0, -1.0],
38                [-3.0, -1.0, 2.0],
39                [-2.0, 1.0, 2.0]])
40 b = np.array([8.0, -11.0, -3.0])
41
42 my_x = gauss_elimination_number(A, b)
43 np_x = np.linalg.solve(A, b)
44
45 print("Решение системы уравнений Ax=b методом Гаусса:", my_x)
46 print("Решение системы уравнений Ax=b методом Гаусса:", np_x)
47 print(f"Решения через np.solve и мое решения {'совпадают' if np.allclose(my_x, np_x) else 'не совпадают'}")
```

3.2.2 Метод Гаусса для числел

```
1  import numpy as np
2  import sympy as sp
3  import os
4
5  output_file = "3.8.2"
6
7
8  def gauss_elimination_sympy(A: np.array, b: sp.Matrix):
9      n = A.shape[0]
10     x = sp.Matrix([0] * n)
11
12     # прямой ход
13     for i in range(n):
14
15         # ищем максимальный коэффициент среди уравнений с неизвестными x и берем его индекс
16         A_slice = A[i:][:]
17         max_element_row_index = np.unravel_index(np.argmax(A_slice), A_slice.shape)[0] + i
18
19         # меняем местами i-ое и max_element_row_index-ое уравнения
20         if max_element_row_index != i:
21             A[[i, max_element_row_index]] = A[[max_element_row_index, i]]  # синтаксис питру
22             b.elementary_row_op('n<->m', row1=i, row2=max_element_row_index)  # синтаксис sympy
23
24         # избавляемся от коэффициентов при x
25         for j in range(i + 1, n):
26             mu = A[j][i] / A[i][i]
27             A[j] -= mu * A[i]
28             b[j] -= mu * b[i]
29
30     # обратный ход
31     for m in range(n - 1, -1, -1):
32         numerator = b[m]  # числитель
33         for l in range(m + 1, n):
```

```
            numerator -= A[m][l] * x[l]
        denominator = A[m][m]  # знаменатель
        x[m] = numerator / denominator

    return sp.sympify(x)


# os.remove(output_file + '.txt')
with open(output_file + '.txt', "w") as f:
    # зададим константы
    n = 40
    M = 2
    q = 1.001 - 2 * M / 1000
    X = sp.Symbol('x')

    # заполним матрицу A и вектор b
    A = np.ndarray((n, n))
    for i in range(1, n + 1):
        for j in range(1, n + 1):
            if i != j:
                A[i - 1][j - 1] = q ** (i + j) + 0.1 * (j - i)
            else:
                A[i - 1][j - 1] = (q - 1) ** (i + j)
    b = sp.Matrix([sp.Abs(X - n / 10) * i * sp.sin(X) for i in range(1, n + 1)])

    # решим систему
    my_z = gauss_elimination_sympy(A, b)
    sp_z = sp.simplify(sp.Matrix(A.tolist()).solve(b))
    print("Решение системы посчитанное через мою функцию:\n", my_z, file=f)
    print("Решение системы посчитанное через sympy:\n", sp_z, file=f)
    print(file=f)

    # вычислим y
    my_y = 0
    sp_y = 0
    for i in range(len(my_z)):
        my_y += my_z[i]
        sp_y += sp_z[i]
    print("Значение 'y' посчитанное через мою функцию:", my_y, file=f)
    print("Значение 'y' посчитанное через sympy:", sp_y, file=f)

    # построим график
    graph = sp.plot(my_y, sp_y, (X, -5, 5), line_color='red', dpi=300)
    backend = graph.backend(graph)
    backend.process_series()
    backend.fig.savefig(output_file + '.png', dpi=300)
    backend.show()
```

3.3 Результаты

```
Решение системы посчитанное через мою функцию:
 Matrix([[-158335.979769201*sin(x)*Abs(x - 4.0)], [-166233.537343039*sin(x)*Abs(x - 4.0)], [-254109.815812915*sin(
Решение системы посчитанное через sympy:
 Matrix([[-158335.979459671*sin(x)*Abs(x - 4.0)], [-166233.537340764*sin(x)*Abs(x - 4.0)], [-254109.815813078*sin(

Значение 'y' посчитанное через мою функцию: -77224.4853234382*sin(x)*Abs(x - 4.0)
Значение 'y' посчитанное через sympy: -77224.4850137634*sin(x)*Abs(x - 4.0)
```

Видно, что результаты, полученные через написанную мной функцию и высичленные через готовую функцию из sympy совпадают.