Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования
"Национальный исследовательский университет
"Высшая школа экономики"

Московский институт электроники и математики Национального исследовательского университета "Высшая школа экономики"

Департамент прикладной математики

ОТЧЕТ
По лабораторной работе №2
на тему
«РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ ПРЯМЫМИ МЕТОДАМИ. ТЕОРИЯ ВОЗМУЩЕНИЙ»

ФИО студента	Номер группы	Дата	Вариант
Ткаченко Никита Андреевич	БПМ211	04.02.2024	3.1.21, 3.5.6, 3.6.5

Москва – 2024 г.

1. Зависимость погрешности решения СЛАУ

Задача 3.1. Дана система уравнений $Ax=b$ порядка n. Исследовать зависимость погрешности решения x от погрешностей правой части системы b.

3.1.21	6	$\frac{1000}{3c^2+c^3}$

Матрица A и вектор b:

	0	1	2	3	4	5
0	44.46222...	7.873519...	2.709168...	1.243187...	0.671873...	0.403770...
1	7.873519...	1.243187...	0.403770...	0.178943...	0.094482...	0.055840...
2	2.709168...	0.403770...	0.127829...	0.055840...	0.029211...	0.017153...
3	1.243187...	0.178943...	0.055840...	0.024201...	0.012597...	0.007372...
4	0.671873...	0.094482...	0.029211...	0.012597...	0.006537...	0.003817...
5	0.403770...	0.055840...	0.017153...	0.007372...	0.003817...	0.002226...

	0	1	2	3	4	5
0	21.0	21.0	21.0	21.0	21.0	21.0

Решение системы x=*[5.83328395e+05, -1.28347866e+08, 2.87812060e+09, -1.71784680e+10, 3.55160291e+10, -2.30797857e+10],*
Число обусловленности = *3749057141917.2344*

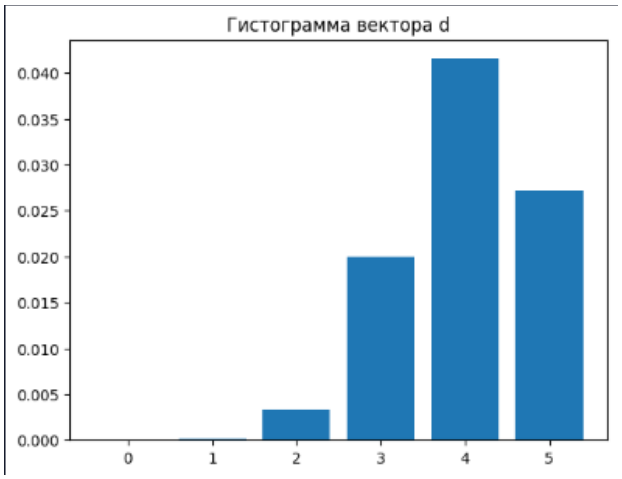
```
N, n = 21, 6
A, b = np.zeros((n, n), dtype=float), np.full(n, fill_value=N, dtype=float)

for i in range(n):
    for j in range(n):
        c = 0.1 * N * (i + 1) * (j + 1)
        A[i, j] = 1000 / (3*c**2 + c**3)

x = np.linalg.solve(A, b)
cond_value = np.linalg.cond(A, p=np.inf)
print(f"Решение системы {x=}, Число обусловленности = {cond_value}")
```

Вычислим вектор относительных погрешностей решения d:

```
delta = 0.05
d = np.empty(n)
for i in range(n):
    b_i = b.copy()
    b_i[i] += delta
    x_i = np.linalg.solve(A, b_i)
    d[i] = np.linalg.norm(x - x_i, ord=np.inf) / np.linalg.norm(x, ord=np.inf)
```

Вектор относительных погрешностей d=*[6.07655693e-07, 1.43070054e-04, 3.29189476e-03, 1.99160852e-02, 4.15215031e-02, 2.71358027e-02]*

Гистограмма вектора d:



Наиболее значимая компонента вектора b имеет индекс 4

Оценим теоретически погрешность решений xi по формуле, используя число обусловленности матриц:

$$\delta(x^m) \le cond(A) \cdot \delta(b^m)$$

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

    estimation = cond_value * (np.linalg.norm(b - b_i, ord=np.inf) / np.linalg.norm(b, ord=np.inf))
```

x0: Погрешность detla(xi): 6.076556930291947e-07, Оцененная погрешность delta(xi): 8926326528.374495
x1: Погрешность detla(xi): 0.00014307005358855097, Оцененная погрешность delta(xi): 8926326528.374495
x2: Погрешность detla(xi): 0.0032918947563169328, Оцененная погрешность delta(xi): 8926326528.374495
x3: Погрешность detla(xi): 0.01991608515744848, Оцененная погрешность delta(xi): 8926326528.374495
x4: Погрешность detla(xi): 0.04152150313758436, Оцененная погрешность delta(xi): 8926326528.374495
x5: Погрешность detla(xi): 0.027135802719507645, Оцененная погрешность delta(xi): 8926326528.374495

Вывод: теоретическая оценка погрешности верная, но сильно больше полученного на практике решения из-за точности готовых методов решения СЛАУ.

2. Метод Холецкого

Задача 3.5. Дана система уравнений $Ax=b$ порядка n с симметричной положительно определенной матрицей A. Решить систему методом Холецкого.

ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ:

1. Написать функцию **cholesky**(A), и с ее помощью получить LL^T - разложение матрицы A.
2. Решить последовательно системы $Ly=b$ и $L^T x = y$ с треугольными матрицами.

УКАЗАНИЕ. Функция **cholesky**(A) должна возвращать нижнетреугольную матрицу L.

Реализуем метод Холецкого разложения матрицы по виду $A=L.L^T$

```
import pandas as pd

n, m = 25, 10

def cholesky(A: np.ndarray, n: int) -> np.ndarray:
    L = np.zeros((n,n))

    for j in range(n):
        for i in range(n):
            if j < i:
                sum = np.dot(L[i,:], L[j,:])
                multiplier = 1/L[j,j]
                sum = 0 if np.isnan(sum) or np.isinf(sum) else sum
                multiplier = 1 if np.isnan(multiplier) or np.isinf(multiplier) else multiplier

                L[i,j] = multiplier * (A[i,j] - sum)
            if i == j:
                sum = np.sum(L[i,:i]**2)
                sum = 0 if np.isnan(sum) or np.isinf(sum) else sum

                L[i,j] = np.sqrt(A[i,i] - sum)
    return L
```

Найдем с помощью него разложение заданной матрицы A и решение СЛАУ:

```
A = np.empty((n,n))
b = np.arange(n)**2 - n

for i in range(n):
    for j in range(n):
        if i == j:
            A[i,j] = n + m**2 + j/m + i/n
        else:
            A[i,j] = (i+j)/(m+n)

L = cholesky(A,n)
A_tmp = L.dot(L.transpose())   # Проверка решения
print(np.allclose(A, A_tmp))
pd.DataFrame(L).to_csv('lmatrix.csv')
pd.DataFrame(L.transpose()).to_csv('lmatrix.csv')
```

Матрица L:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	11.18034	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
1	0.00256	11.18660	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	0.00511	0.00766	11.19285	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
3	0.00767	0.01021	0.01275	11.19909	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
4	0.01022	0.01277	0.01530	0.01782	11.20532	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
5	0.01278	0.01532	0.01785	0.02037	0.02286	11.21153	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
6	0.01533	0.01788	0.02040	0.02291	0.02540	0.02786	11.21771	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
7	0.01789	0.02043	0.02295	0.02546	0.02794	0.03039	0.03282	11.22387	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
8	0.02044	0.02298	0.02550	0.02800	0.03047	0.03292	0.03534	0.03773	11.23000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
9	0.02300	0.02554	0.02805	0.03054	0.03301	0.03545	0.03786	0.04023	0.04257	11.23610	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
10	0.02556	0.02809	0.03060	0.03309	0.03555	0.03798	0.04037	0.04274	0.04506	0.04735	11.24216	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
11	0.02811	0.03064	0.03315	0.03563	0.03808	0.04051	0.04289	0.04524	0.04755	0.04982	0.05205	11.24818	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
12	0.03067	0.03320	0.03570	0.03818	0.04062	0.04303	0.04541	0.04775	0.05004	0.05230	0.05451	0.05668	11.25415	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
13	0.03322	0.03575	0.03825	0.04072	0.04316	0.04556	0.04793	0.05025	0.05254	0.05478	0.05697	0.05912	0.06121	11.26008	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
14	0.03578	0.03830	0.04080	0.04326	0.04570	0.04809	0.05044	0.05276	0.05503	0.05725	0.05943	0.06156	0.06364	0.06566	11.26597	0.00000	0.00000	0.00000	0.00000	0.00000
15	0.03833	0.04086	0.04335	0.04581	0.04823	0.05062	0.05296	0.05526	0.05752	0.05973	0.06189	0.06400	0.06606	0.06806	0.07001	11.27180	0.00000	0.00000	0.00000	0.00000
16	0.04089	0.04341	0.04590	0.04835	0.05077	0.05315	0.05548	0.05777	0.06001	0.06220	0.06435	0.06644	0.06848	0.07046	0.07239	0.07427	11.27758	0.00000	0.00000	0.00000
17	0.04344	0.04596	0.04845	0.05090	0.05331	0.05567	0.05800	0.06027	0.06250	0.06468	0.06681	0.06888	0.07090	0.07286	0.07477	0.07662	0.07841	11.28330	0.00000	0.00000
18	0.04600	0.04852	0.05100	0.05344	0.05584	0.05820	0.06051	0.06278	0.06499	0.06716	0.06927	0.07132	0.07332	0.07526	0.07715	0.07897	0.08074	0.08245	11.28897	0.00000
19	0.04855	0.05107	0.05355	0.05599	0.05838	0.06073	0.06303	0.06528	0.06749	0.06963	0.07173	0.07376	0.07574	0.07766	0.07952	0.08133	0.08307	0.08475	0.08637	11.29458
20	0.05111	0.05362	0.05610	0.05853	0.06092	0.06326	0.06555	0.06779	0.06998	0.07211	0.07418	0.07620	0.07816	0.08006	0.08190	0.08368	0.08540	0.08705	0.08865	0.09018
21	0.05367	0.05618	0.05865	0.06107	0.06346	0.06579	0.06807	0.07030	0.07247	0.07459	0.07664	0.07864	0.08058	0.08246	0.08428	0.08603	0.08773	0.08936	0.09092	0.09243
22	0.05622	0.05873	0.06120	0.06362	0.06599	0.06831	0.07059	0.07280	0.07496	0.07706	0.07910	0.08109	0.08300	0.08486	0.08666	0.08839	0.09006	0.09166	0.09320	0.09468
23	0.05878	0.06128	0.06375	0.06616	0.06853	0.07084	0.07310	0.07531	0.07745	0.07954	0.08156	0.08353	0.08543	0.08726	0.08903	0.09074	0.09238	0.09396	0.09547	0.09692
24	0.06133	0.06384	0.06630	0.06871	0.07107	0.07337	0.07562	0.07781	0.07994	0.08201	0.08402	0.08597	0.08785	0.08966	0.09141	0.09310	0.09471	0.09626	0.09775	0.09917

Решение системы:

```
y = np.linalg.solve(L, b)
x = np.linalg.solve(L.transpose(), y)
print(x)
```

[-0.3326762 -0.33098052 -0.31330826 -0.27969114 -0.23016078 -0.16474871
-0.08348641 0.01359477 0.12646352 0.25508864 0.39943901 0.55948358
0.73519138 0.92653154 1.13347326 1.35598581 1.59403856 1.84760094
2.11664249 2.4011328   2.70104155 3.0163385   3.3469935   3.69297646
4.05425737]

3. Метод Гаусса с выбором главной строки по столбцу (частичным выбором)

Задача 3.6.* Дана система уравнений $Ax=b$ порядка n, где $A=A(t)$, t - параметр. Исследовать зависимость решения системы $Ax=b$ от вычислительной погрешности при заданных значениях параметра t.

ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ:

1. Составить программу, реализующую метод Гаусса (схема частичного выбора) для произвольной системы $Ax=b$. Используя составленную программу, найти решение заданной системы $Ax=b$.
2. Составить программу округления числа до m знаков после запятой. Вычислить элементы матрицы A и вектора b по формулам индивидуального варианта, производя округление до m- знаков после запятой (в результате будут получены матрица $A1$ и вектор $b1$).
3. Решить систему уравнений $A1x1=b1$ методом, указанным в п.1, обращаясь каждый раз к программе округления. Оценить практически полученную погрешность решения.
4. Сравнить результаты, полученные при разных значениях параметра t.

Реализация метода Гаусса со схемой частичного выбора:

```
def gauss_partial_sle(A, b):
    n = len(b)
    M = np.hstack((A, b.reshape(-1, 1)))
    # Прямой ход
    for k in range(n):
        # Выбор главной строки по столбцу
        max_index = k + np.argmax(np.abs(M[k:,k]))
        M[k], M[max_index] = M[max_index].copy(), M[k].copy()

        for i in range(k+1, n):
            modifier = M[i, k] / M[k, k]
            M[i, :] -= M[k, :] * modifier

    # Обратный ход
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        multiplier = 1 / M[i,i]
        sum = np.dot(M[i,i+1:n], x[i+1:])
        x[i] = multiplier * (M[i, -1] - sum)
    return x
```

Функция округления числа до m знаков после запятой:

```
def value_to_decimal(value, m):
    return int(value * 10 ** m + 0.5) / 10 ** m
```

Сгенерируем нужные матрицы и решим системы уравнений:

```
def gauss_partial_sle_rounded(A, b):
    n = len(b)
    M = np.hstack((A, b.reshape(-1, 1)))
    for k in range(n):
        max_index = k + np.argmax(np.abs(M[k:,k]))
        M[k], M[max_index] = M[max_index].copy(), M[k].copy()

        for i in range(k+1, n):
            modifier = value_to_decimal(M[i, k] / M[k, k], m)
            M[i, :] -= M[k, :] * modifier
            M[i, :] = np.array([value_to_decimal(v, m) for v in M[i, :]])

    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        multiplier = value_to_decimal(1 / M[i,i], m)
        sum = np.dot(M[i,i+1:n], x[i+1:])
        x[i] = value_to_decimal(multiplier * (M[i, -1] - sum), m)
    return x
```

```
def generate_sle(t):
    qm = 0.993 + (-1)**M * M * 10 ** -4

    A = np.empty((n,n))
    b = qm**(n+1-np.arange(n))
    for i in range(n):
        for j in range(n):
            q_j = qm ** j
            if i == j:
                A[i,j] = q_j + t
            else:
                A[i,j] = q_j
    return A, b
```

```
def generate_sle_rounding(t):
    qm = np.round(0.993 + np.round((-1)**M * M * 10 ** -4, m),m)

    A = np.empty((n,n))
    b = np.round(qm**(n+1-np.arange(n)),m)
    for i in range(n):
        for j in range(n):
            q_j = np.round(qm ** j, m)
            if i == j:
                A[i,j] = np.round(q_j + np.round(t,m))
            else:
                A[i,j] = q_j
    return A, b
```

```
M, N, n, m = 5, 5, 45, 4
T = [0.0001, 1, 10000]

for t in T:
    A1, b1 = generate_sle(t)
    A2, b2 = generate_sle_rounding(t)
    x1 = gauss_partial_sle(A1, b1)
    x2 = gauss_partial_sle_rounded(A2,b2)
    discrepancy1 = b2 - A2.dot(x2)
    print(f"t: {t}; Решение системы {x1=}")
    print(f"t: {t}; Решение системы {x2=}")
    print(f"t: {t}; Норма разности решений: {np.linalg.norm(x1 - x2)}, Норма неувязки второго решения:
{np.linalg.norm(discrepancy1)}")
    print()
```

Матрицы и решения 1 — стандарные матрицы и решения, Матрицы и решения 2 — со включенным округлением до 4-х знаков после запятой.

Оценка полученной погрешности для округленного решения проводится благодаря вычислению значения Неувязки по формуле: $r = b - Ax^*$

(Но на экран выводится ее норма для более визуального представления погрешности) Также проводится оценка нормы разности решений, для сравнения результатов между разными значенями t:

t = 0.0001:

Решение системы x1=*[-1234.25290944, -1180.80454033, -1126.95227926, -1072.69307416, -1018.02384987, -962.94150801, -907.4429268 , -851.52496084,*
-795.18444098, -738.41817412, -681.22294303, -623.59550616,
-565.53259748, -507.03092626, -448.08717693, -388.69800883,
-328.86005608, -268.56992737, -207.82420575, -146.61944845,
-84.95218669, -22.81892546, 39.78385662, 102.8597076 ,
166.41220228, 230.44494251, 294.96155736, 359.9657033 ,
425.46106444, 491.45135275, 557.94030823, 624.93169914,
692.42932222, 760.4370029 , 828.95859554, 897.99798358,
967.55907984, 1037.6458267 , 1108.26219634, 1179.41219094,
1251.09984292, 1323.3292152 , 1396.10440138, 1469.42952599,
1543.30874474]

Решение системы x2=*[-30.4121, 0.7067, 0.7181, 0.7265, 0.7273, 0.7344,*
0.7398, 0.7466, 0.7517, 0.7573, 0.7628, 0.7686,
0.7743, 0.78 , 0.786 , 0.7921, 0.7974, 0.8043,
0.8098, 0.8162, 0.8219, 0.8283, 0.8349, 0.8409,
0.8475, 0.8537, 0.86 , 0.8668, 0.8732, 0.8802,
0.8863, 0.8933, 0.9 , 0.9068, 0.9136, 0.9206,
0.9275, 0.9346, 0.9417, 0.9489, 0.9558, 0.9631,
0.9701, 0.9779, 0.9851]

Норма разности решений: *5513.487819214705*, Норма неувязки второго решения: *0.0007951569484329792*

t = 1:

Решение системы x1=*[-0.10229689, -0.09695205, -0.09156682, -0.0861409 , -0.08067398,*
-0.07516575, -0.06961589, -0.06402409, -0.05839004, -0.05271341,
-0.04699389, -0.04123115, -0.03542486, -0.02957469, -0.02368031,
-0.0177414 , -0.0117576 , -0.00572859, 0.00034598, 0.00646646,
0.01263319, 0.01884651, 0.02510679, 0.03141437, 0.03776962,
0.0441729 , 0.05062456, 0.05712497, 0.06367451, 0.07027354,
0.07692244, 0.08362157, 0.09037134, 0.0971721 , 0.10402426,
0.1109282 , 0.11788431, 0.12489299, 0.13195462, 0.13906962,
0.14623839, 0.15346133, 0.16073884, 0.16807136, 0.17545928]

Решение системы x2=*[-0.094 , -0.0879, -0.0821, -0.076 , -0.0702, -0.0643, -0.0586,*
-0.0527, -0.0472, -0.0414, -0.0359, -0.0303, -0.0247, -0.0192,
-0.0137, -0.0082, -0.0029, 0.0024, 0.0077, 0.0132, 0.0185,
0.0238, 0.0292, 0.0343, 0.0397, 0.0449, 0.0501, 0.0554,
0.0606, 0.0659, 0.071 , 0.0763, 0.0815, 0.0867, 0.0918,
0.097 , 0.1022, 0.1074, 0.1127, 0.1178, 0.1229, 0.1281,
0.1333, 0.1385, 0.1437]

Норма разности решений: *0.08895640095170222*, Норма неувязки второго решения: *0.0031698397197645277*

t = 10000:

Решение системы x1=*[7.04129382e-05, 7.09474219e-05, 7.14859445e-05, 7.20285366e-05, 7.25752288e-05,*
7.31260523e-05, 7.36810381e-05, 7.42402177e-05,
7.48036229e-05, 7.53712856e-05, 7.59432379e-05, 7.65195123e-05,
7.71001414e-05, 7.76851581e-05, 7.82745956e-05, 7.88684872e-05,
7.94668668e-05, 8.00697681e-05, 8.06772253e-05, 8.12892729e-05,
8.19059455e-05, 8.25272781e-05, 8.31533059e-05, 8.37840644e-05,
8.44195894e-05, 8.50599168e-05, 8.57050829e-05, 8.63551244e-05,
8.70100780e-05, 8.76699809e-05, 8.83348704e-05, 8.90047843e-05,
8.96797606e-05, 9.03598374e-05, 9.10450533e-05, 9.17354472e-05,
9.24310581e-05, 9.31319256e-05, 9.38380893e-05, 9.45495892e-05,
9.52664658e-05, 9.59887595e-05, 9.67165114e-05, 9.74497626e-05,
9.81885548e-05]

Решение системы x2=*[0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001,*
0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001,
0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001,
0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001,
0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001,
0.0001, 0.0001, 0.0001, 0.0001, 0.0001]

Норма разности решений: *0.0001232597219967743*, Норма неувязки второго решения: *1.2366289073704584*

Графики относительной, абсолютной погрешности и невязки в зависимости от t: