НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
"МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ"

Кафедра математического и компьютерного моделирования

Численные методы
Отчет по лабораторной работе №3 "Решение систем линейных алгебраических уравнений прямыми методами. Теория возмущений." Вариант 33

Студент:	Волков Павел Евгеньевич
Преподаватель:	Амосова Ольга Алексеевна
Группа:	А-14-19

Москва
2021

Задача 3.1

Постановка задачи

Реализовать решение СЛАУ с помощью LU-разложения и LU-разложения по схеме частичного выбора. Решить систему небольшой размерности с возмущенной матрицей обоими методами, оценить погрешность и сравнить с теоретической оценкой. Проанализировать поведение метода с ростом числа уравнений.

$(33+3)mod2 = 0$ - решение с помощью LU-разложения реализовано в виде 2-х функций, одна из которых возвращает две матрицы - L и U, не модифицируя A, а вторая функция решает систему, решение с помощью LU по схеме частичного выбора модифицирует исходную матрицу A.

$$(33+3)mod4 = 0 \rightarrow A_{i,j} = \mathrm{tg}^{17-j}(i+1)$$

Решение

Приведем код простого LU-разложения:

```
def LU_dec(A: np.ndarray) -> tuple:
    L = np.empty(A.shape)
    U = np.empty(A.shape)
    n = A.shape[0]
    for i in range(n):
        for j in range(n):
            U[i, j] = A[i, j]
            L[i, j] = int(i == j)
    for i in range(n):
        for j in range(i+1, n):
            L[j, i] = U[j, i] / U[i, i]
            U[j] = U[j] - L[j, i] * U[i]
    return (L, U)
```

Метод LU-разложения по схеме частичного выбора:

```
def swap(A: np.ndarray, L: np.ndarray, permutations: list, i: int):
    mx, line = np.abs(A[i, i]), i
    for j in range(i+1, A.shape[0]):
        if np.abs(A[j, i]) > mx:
            mx = np.abs(A[j, i])
            line = j
    for j in range(A.shape[0]):
        A[line, j], A[i, j] = A[i, j], A[line, j]
        L[line, j], L[i, j] = L[i, j], L[line, j]
    permutations[i], permutations[line] = permutations[line],
        permutations[i]

def LU_part(A: np.ndarray) -> tuple:
    L = np.empty(A.shape)
    n = A.shape[0]
    permutations = list(range(n))
    for i in range(n):
        for j in range(n):
            L[i, j] = int(i == j)
    for i in range(n):
        swap(A, L, permutations, i)
        for j in range(i+1, n):
            L[j, i] = A[j, i] / A[i, i]
            A[j] = A[j] - L[j, i] * A[i]
    return L, permutations
```

Точное решение системы 5 на 5 $Ax = b$, где $b = A(N, N, N, N, N)^T$:

$$x = (33., 33., 33., 33., 33.)^T$$

Теперь прибавим 0.001 к первому элементу матрицы, и найдем решения с помощью обоих методов:

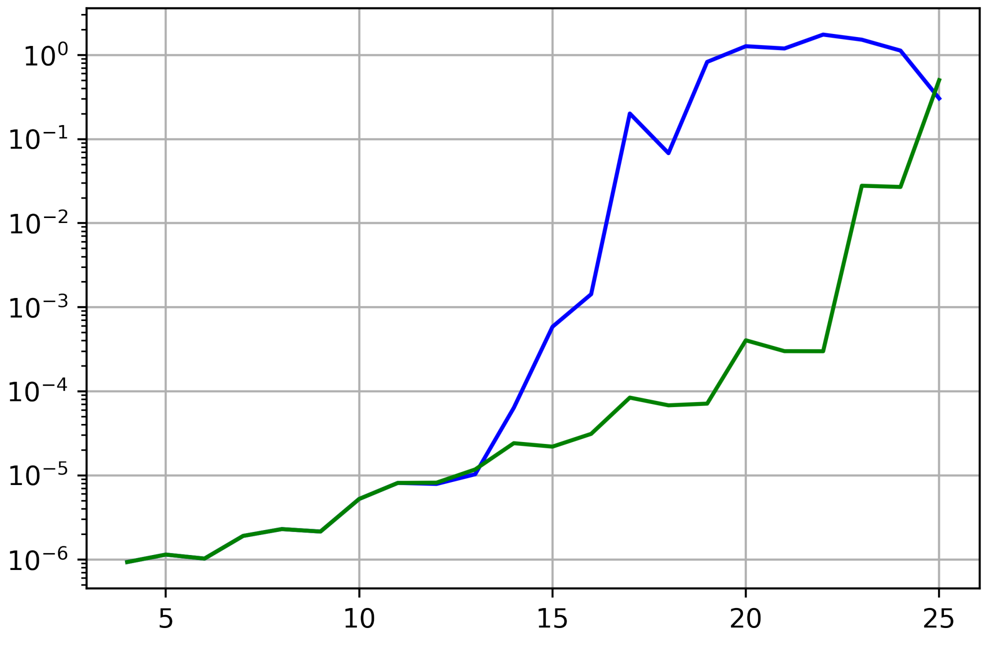
$$x1 = (32.99999171, 32.99996228, 32.99998698, 33.00006978, 33.00001011)^T$$
$$x2 = (32.99999171, 32.99996228, 32.99998698, 33.00006978, 33.00001011)^T$$

Вычислим относительную погрешность решения, и сравним ее с теоретической оценкой:

$$\delta(x^*) = \frac{||\bar{x} - x^*||}{||x^*||} = 1.143 \cdot 10^{-6}$$
$$\delta(b^*) = 0$$
$$\delta(A^*) = \frac{||A - A^*||}{||A||} = 7.177 \cdot 10^{-13}$$
$$\nu_\delta = cond(A) = ||A|| \cdot ||A^{-1}|| = 1.265 \cdot 10^{20}$$

Таким образом, неравенство $\delta(x^*) \leq \nu_\delta \cdot (\delta(b^*) + \delta(A^*)) = 1.143 \cdot 10^{-6} \leq 90.805552 \cdot 10^6$ выполняется.

Ниже представлены графики зависимости относительной погрешности решения от числа уравнений для обоих методов:



На графике видно, что благодаря перестановкам строк, и выбору ведущими элементами наибольших по модулю в столбце, метод частичного выбора сохраняет относительную погрешность менее 1% для числа уравнений < 23, а метод без перестановок < 17.

Задача 3.3

Постановка задачи

Решить задачу итерационным методом, указанным в индивидуальном варианте. Вектор правой части задается как $b = Ax$, где $x_i = 33$

Элементы матрицы A задаются формулами
$a_{i,j} = \dfrac{\cos i + j}{0.1 \cdot \beta} + 0.1\beta \cdot e^{-(i-j)^2}$.

Параметр β задается формулой $\beta = (|66 - 33| + 5) \cdot m$, здесь N - номер варианта, m - размерность матрицы, указанная в варианте. Вектор b задается по вектору решения.

m = 26, метод минимальных невязок.

Решение

В качестве отчета по данной задаче приведем код метода минимальных невязок и функцию, вычисляющую бесконечную норму вектора(матрицы).

Метод минимальных невязок:

```
def minimum_deviation(matr: np.ndarray, b: np.ndarray, eps:
    np.float64) -> np.ndarray:
    x = np.zeros_like(b)
    r = matr.dot(x) - b
    r0 = r
    tau = r.dot(r) / ((matr.dot(r)).dot(r))
    x = x - tau * r
    while np.abs(Norm(r) / Norm(r0)) > eps:
        r = matr.dot(x) - b
        tau = r.dot(r) / ((matr.dot(r)).dot(r))
        x = x - tau * r
    return x
```

Бесконечая норма вектора(матрицы):

```
def Norm(m: np.ndarray) -> np.float64:
    if len(m.shape) == 1:
        tmp, mx = 0, np.abs(m[0])
        for i in range(m.shape[0]):
            if np.abs(m[i]) > mx:
                mx = np.abs(m[i])
        return mx
```

```
    mx = 0
    for i in range(m.shape[0]):
        tmp = 0
        for j in range(m.shape[0]):
            tmp += np.abs(m[i][j])
        if tmp > mx:
            mx = tmp
    return mx
```

Код основной программы:

```
def matrix(n):
    beta = 38 * n
    f = lambda i, j: (np.cos(i) + j) / (0.1 * beta) + 0.1 * beta *
        np.exp(-(i - j)**2)
    return np.array([[f(i, j) for j in range(n)] for i in range(n)],
        np.float64)

n = 26
A = matrix(n)
b = A.dot(np.array([33 for _ in range(n)], np.float64))
x = minimum_deviation(A, b, np.float64(1e-5))
print(x)
```

Результат работы программы:

$$[32.99977437, 33.0004763, 32.99945582, 33.00066142, 32.99947897,\\ 33.00053767, 32.99971532, 33.00031782, 32.99993953, 33.0001698,\\ 33.00005689, 33.00011429, 33.00009631, 33.00009634, 33.00011434,\\ 33.00005693, 33.00016975, 32.9999395, 33.0003177, 32.99971541,\\ 33.00053757, 32.99947915, 33.0006612, 32.9994559, 33.00047608,\\ 32.99977439]$$