Часть 1

1. Решите систему уравнений методом Гаусса:

$$\begin{cases} x_1+x_2-x_3-2x_4=0, \\ 2x_1+x_2-x_3+x_4=-2, \\ x_1+x_2-3x_3+x_4=4. \end{cases}$$

Решение

Из трьетего уравнения вычтем первое, и из второго первое:

$$\begin{cases} x_1+x_2-x_3-2x_4=0 \\ x_1+3x_4=-2 \\ -2x_3+3x_4=4 \end{cases}$$

из второго выразим $x_1=-2-3x_4$, из третьего выразим $x_3=2-1.5x_4$ подставим в первое, найдем x_2:

$$-2-3x_4+x_2-2+1.5x_4-2x_4=0$$

$$x_2=4-3.5x_4$$

Общее решение выглядит так:

$$\begin{cases} x_1=-2-3x_4 \\ x_2=4-3.5x_4 \\ x_3=2-1.5x_4 \end{cases}$$

2. Проверьте на совместность и выясните, сколько решений будет иметь система линейных уравнений:

а) $\begin{cases} 3x_1-x_2+x_3=4, \\ 2x_1-5x_2-3x_3=-17, \\ x_1+x_2-x_3=0; \end{cases}$

б) $\begin{cases} 2x_1-4x_2+6x_3=1, \\ x_1-2x_2+3x_3=-2, \\ 3x_1-6x_2+9x_3=5; \end{cases}$

в) $\begin{cases} x_1+2x_2+5x_3=4, \\ 3x_1+x_2-8x_3=-2. \end{cases}$

```
# а)
import numpy as np

A = np.array([[3, -1, 1], [2, -5, -3], [1, 1, -1]])
b = np.array([4, -17, 0])[:, np.newaxis]
A_extended = np.hstack((A, b))
print(f'rank A = {np.linalg.matrix_rank(A)}')
print(f'rank A_extended = {np.linalg.matrix_rank(A_extended)}')
```

```
rank A = 3
rank A_extended = 3
```

число неизвестных 3, система совместна и имеет единственное решение

```
# б)
A = np.array([[2, -4, 6], [1, -2, 3], [3, -6, 9]])
b = np.array([1, -2, 5])[:, np.newaxis]
A_extended = np.hstack((A, b))
print(f'rank A = {np.linalg.matrix_rank(A)}')
print(f'rank A_extended = {np.linalg.matrix_rank(A_extended)}')
```

```
rank A = 1
rank A_extended = 2
```

Система не совместна и не имеет решений

```
# в)
A = np.array([[1, 2, 5], [3, 1, -8]])
b = np.array([4, -2])[:, np.newaxis]
A_extended = np.hstack((A, b))
print(f'rank A = {np.linalg.matrix_rank(A)}')
print(f'rank A_extended = {np.linalg.matrix_rank(A_extended)}')
```

```
rank A = 2
rank A_extended = 2
```

ранги матриц равны 2, но число неизвестных 3, система имеет бесконечное количество решений

3. Проверьте на совместность и выясните, сколько решений будет иметь система линейных уравнений, заданная расширенной матрицей:

$$\widetilde{A}=\left(\begin{array}{cccc|c} 1 & 3 & -2 & 4 & 3 \\ 0 & 5 & 0 & 1 & 2 \\ 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 0 & 2 & 1 \end{array}\right).$$

```
A = np.array([[1, 3, -2, 4], [0, 5, 0, 1], [0, 0, 3, 0], [0, 0, 0,
2]])
b = np.array([[3], [2], [4], [1]])
A_ext = np.hstack((A, b))
```

```
print(f'Система совместна: {np.linalg.matrix_rank(A) ==
np.linalg.matrix_rank(A_ext)}')
print(f'Имеет единственное решение: {np.linalg.matrix_rank(A) ==
np.linalg.matrix_rank(A_ext) == 4}')
```

```
Система совместна: True
Имеет единственное решение: True
```

4. Дана система линейных уравнений, заданная расширенной матрицей:

$$\widetilde{A}=\left(\begin{array}{ccc|c} 1 & 2 & 3 & a \\ 4 & 5 & 6 & b \\ 7 & 8 & 9 & c \end{array}\right).$$

Найдите соотношение между параметрами a, b и c, при которых система считается несовместной.

Решение

Система будет несовместной когда $rank\, A < rank\, \widetilde{A}$. $rank\, A = 2$ т.к. третья строка является линейной комбинацией первых двух $A_3 = 2A_2 - A_1$ преобразуем матрицу. Из третье строки вычтем удвоенную вторую и затем сложим с первой

```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
A[2] = A[2] - 2*A[1] + A[0]
A
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [0, 0, 0]])
```

Чтобы система была несовместной нужно чтобы после преобразований в расширенной матрице в третьей строке свободных членов был не 0, то есть $c - 2b + a \neq 0$

Часть 2

1. Решите систему уравнений методом Крамера:

а) $\begin{cases} x_1 - 2x_2 = 1 \\ 3x_1 - 4x_2 = 7 \end{cases}$

б) $\begin{cases} 2x_1 - x_2 + 5x_3 = 10 \\ x_1 + x_2 - 3x_3 = -2 \\ 2x_1 + 4x_2 + x_3 = 1 \end{cases}$

```
def kramer(A, b):
    assert A.shape[0] == A.shape[1], 'Wrong matrix size!'
```

```
    det_A = np.linalg.det(A)
    result = []
    if np.abs(det_A) > 0.000001: #может быть ошибка округления
        for i in range(A.shape[0]):
            A_i = np.hstack((A[:, :i], b, A[:, i+1:]))
            result.append(np.linalg.det(A_i) / det_A)
        return np.round(result, 2)
    else:
        return 'det A is 0'

# а)
A = np.array([[1, -2], [3, -4]])
b = np.array([[1, 7]]).T
print(f'а) {kramer(A, b)}')
# б)
A = np.array([[2, -1, 5], [1, 1, -3], [2, 4, 1]])
b = np.array([[10, -2, 1]]).T
print(f'б) {kramer(A, b)}')
```

```
а) [5. 2.]
б) [ 2. -1.  1.]
```

_2*._ Найдите L-матрицу LU-разложения для матрицы коэффициентов:

а)

$$\begin{pmatrix} 1 & 2 & 4 \\ 2 & 9 & 12 \\ 3 & 26 & 30 \end{pmatrix}$$

б)

$$\begin{pmatrix} 1 & 1 & 2 & 4 \\ 2 & 5 & 8 & 9 \\ 3 & 18 & 29 & 18 \\ 4 & 22 & 53 & 33 \end{pmatrix}$$

```
def LU(A):
    assert A.shape[0] == A.shape[1] and A[0][0] !=0 and
np.abs(np.linalg.det(A)) > 0.00001, 'Невозможно разложить матрицу'
    L = np.zeros_like(A)
    U = np.zeros_like(A)
    # первая строка U и первый столбец L
    U[0] = A[0]
    L[:, 0] = A[:, 0] / U[0, 0]
    for i in range(A.shape[1]):
        for j in range(i, A.shape[1]):
            U[0, i] = A[0, i]
            L[i, 0] = A[i, 0] / U[0, 0]
            u_sum, l_sum = 0, 0
```

```
            for k in range(i):
                u_sum += L[i, k] * U[k, j]
                l_sum += L[j, k] * U[k, i]
            U[i, j] = A[i, j] - u_sum
            L[j, i] = (A[j, i] - l_sum) / U[i, i]
    return U, L

# а)
A = np.array([[1, 2, 4], [2, 9, 12], [3, 26, 30]])
U, L = LU(A)
print("а) ")
print("Матрица U:")
print(U)
print("Матрица L:")
print(L)
print("Матрица LU == A:")
print(L.dot(U))
print(A)
# б)
A = np.array([[1, 1, 2, 4], [2, 5, 8, 9], [3, 18, 29, 18], [4, 22, 53,
33]])
U, L = LU(A)
print("б) ")
print("Матрица U:")
print(U)
print("Матрица L:")
print(L)
print("Матрица LU == A:")
print(L.dot(U))
print(A)
```

```
а)
Матрица U:
[[1 2 4]
 [0 5 4]
 [0 0 2]]
Матрица L:
[[1 0 0]
 [2 1 0]
 [3 4 1]]
Матрица LU == A:
[[ 1  2  4]
 [ 2  9 12]
 [ 3 26 30]]
[[ 1  2  4]
 [ 2  9 12]
 [ 3 26 30]]
б)
Матрица U:
[[1 1 2 4]
```

```
 [0 3 4 1]
 [0 0 3 1]
 [0 0 0 4]]
Матрица L:
[[1 0 0 0]
 [2 1 0 0]
 [3 5 1 0]
 [4 6 7 1]]
Матрица LU == A:
[[ 1  1  2  4]
 [ 2  5  8  9]
 [ 3 18 29 18]
 [ 4 22 53 33]]
[[ 1  1  2  4]
 [ 2  5  8  9]
 [ 3 18 29 18]
 [ 4 22 53 33]]
```

PS

в ноутбуке в общей формуле расчета матрицы U ошибка: вместо

$$u_{ij}=a_{ij}-\sum_{k=1}^{i-1} l_{ik}u_{ki},$$

нужно:

$$u_{ij}=a_{ij}-\sum_{k=1}^{i-1} l_{ik}u_{kj},$$

Всю голову себе сломал, пока нашел почему не получается)))

__3*.__ Решите систему линейных уравнений методом LU-разложения:

$$\begin{cases} 2x_1+x_2+3x_3=1 \\ 11x_1+7x_2+5x_3=-6 \\ 9x_1+8x_2+4x_3=-5 \end{cases}$$

```python
def LU_slau(A, b):
    U, L = LU(A)
    y = np.zeros(A.shape[0])
    x = np.zeros(A.shape[0])
    for i in range(A.shape[0]):
        y[i] = b[i] - (L[i, :i] * y[:i]).sum()
    for i in range(A.shape[0]-1, -1, -1):
        x[i] = (y[i] - (U[i, i+1:]*x[i+1:]).sum())/U[i, i]
    return x

A = np.array([[2, 1, 3], [11, 7, 5], [9, 8, 4]])
b = np.array([1, -6, 5])
```

```
x = LU_slau(A, b)
print(f'x = {x}')
print('Проверка')
for i in range(len(A)):
    print((x * A[0]).sum() == b[0])
```

```
x = [-4.41666667  4.08333333  1.91666667]
Проверка
True
True
True
```

__5*.__ Напишите на Python программу с реализацией одного из изученных алгоритмов решения СЛАУ.

функции kramer, LU_slau