Практическое задание 4.2

In [1]:

```
import sympy as sym
import numpy as np
np.set_printoptions(precision=2, suppress=True)
import warnings
warnings.filterwarnings("ignore")
```

1. Решить систему уравнений методом Крамера:

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

In [2]:

```
A = np.array([[1., -2.], [3., -4.]])
A_1 = np.array([[1., -2.], [7., -4.]])
A_2 = np.array([[1., 1.], [3., 7.]])
B = np.array([1., 7.])
```

In [3]:

```
det_A = np.linalg.det(A)
det_A
```

Out[3]:

```
2.0000000000000004
```

In [4]:

```
X_1 = np.linalg.det(A_1)/det_A
X_1
```

Out[4]:

```
4.999999999999998
```

In [5]:

```
X_2 = np.linalg.det(A_2)/det_A
X_2
```

Out[5]:

```
1.9999999999999991
```

Видим, что корни данной СЛАУ это 5 и 2. Проврим

In [6]:

```
np.linalg.solve(A, B)
```

Out[6]:

```
array([5., 2.])
```

Видим, что решение найдено верно

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

In [7]:

```
A = np.array([[2., -1., 5.], [1., 1., -3.], [2., 4., 1.]])
A_1 = np.array([[10., -1., 5.], [-2., 1., -3.], [1., 4., 1.]])
A_2 = np.array([[2., 10., 5.], [1., -2., -3.], [2., 1., 1.]])
A_3 = np.array([[2., -1., 10.], [1., 1., -2.], [2., 4., 1.]])
B = np.array([10., -2., 1.])
```

In [8]:

```
det_A = np.linalg.det(A)
det_A
```

Out[8]:

```
42.99999999999998
```

In [9]:

```
A
```

Out[9]:

```
array([[ 2., -1.,  5.],
       [ 1.,  1., -3.],
       [ 2.,  4.,  1.]])
```

In [10]:

```
A_3
```

Out[10]:

```
array([[ 2., -1., 10.],
       [ 1.,  1., -2.],
       [ 2.,  4.,  1.]])
```

In [11]:

```
X_1 = np.linalg.det(A_1)/det_A
X_1
```

Out[11]:

```
2.0000000000000018
```

In [12]:

```
X_2 = np.linalg.det(A_2)/det_A
X_2
```

Out[12]:

```
-1.0000000000000009
```

In [13]:

```
X_3 = np.linalg.det(A_3)/det_A
X_3
```

Out[13]:

```
1.0
```

Видим, что корни данной СЛАУ это 2, -1 и 1. Проврим

In [14]:

```
np.linalg.solve(A, B)
```

Out[14]:

```
array([ 2., -1.,  1.])
```

Видим, что решение найдено верно

2*. Найти L-матрицу LU-разложения для матрицы коэффициентов:

а)

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

In [15]:

```
A = np.array([[1., 2., 4.], [2., 9., 12.], [3., 26., 30.]])
A
```

Out[15]:

```
array([[ 1.,  2.,  4.],
       [ 2.,  9., 12.],
       [ 3., 26., 30.]])
```

Напишем функцию для поиска L и U матриц

In [16]:

```
def LU_decomposition(A):
    global U, L
    U = A
    L = np.array(np.eye(len(A)))
    for nrow in range(len(A)):
        for i in range(nrow + 1):
            if nrow != len(A) - 1:
                L[nrow + 1][i] = U[nrow + 1][i] / U[i][i]
                U[nrow + 1] = U[nrow + 1] - U[i] * (U[nrow + 1][i] / U[i][i])

            else:
                break
    print("Матрица U:")
    print(U)
    print("Матрица L:")
    print(L)
```

In [17]:

```
LU_decomposition(A)
```

```
Матрица U:
[[1. 2. 4.]
 [0. 5. 4.]
 [0. 0. 2.]]
Матрица L:
[[1. 0. 0.]
 [2. 1. 0.]
 [3. 4. 1.]]
```

Проверим правильность расчетов перемножением L на U

In [18]:

```
L @ U
```

Out[18]:

```
array([[ 1.,  2.,  4.],
       [ 2.,  9., 12.],
       [ 3., 26., 30.]])
```

Видим, что матрицы L и U найдены правильно

б)

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

In [19]:

```
A = np.array([[1., 1., 2., 4], [2., 5., 8., 9], [3., 18., 29., 18], [4., 22., 53., 33]]
A
```

Out[19]:

```
array([[ 1.,  1.,  2.,  4.],
       [ 2.,  5.,  8.,  9.],
       [ 3., 18., 29., 18.],
       [ 4., 22., 53., 33.]])
```

In [20]:

```
LU_decomposition(A)
```

```
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.]]
```

Проверим правильность расчетов перемножением L на U

In [21]:

```
L @ U
```

Out[21]:

```
array([[ 1.,  1.,  2.,  4.],
       [ 2.,  5.,  8.,  9.],
       [ 3., 18., 29., 18.],
       [ 4., 22., 53., 33.]])
```

Видим, что матрицы L и U найдены правильно

3*. Решить систему линейных уравнений методом LU-разложения

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

In [22]:

```
np.set_printoptions(precision=10, floatmode='fixed')
```

In [23]:

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

Out[23]:

```
array([[ 2.0000000000,  1.0000000000,  3.0000000000],
       [11.0000000000,  7.0000000000,  5.0000000000],
       [ 9.0000000000,  8.0000000000,  4.0000000000]])
```

In [24]:

```
LU_decomposition(A)
```

```
Матрица U:
[[  2.0000000000   1.0000000000   3.0000000000]
 [  0.0000000000   1.5000000000 -11.5000000000]
 [  0.0000000000   0.0000000000  17.3333333333]]
Матрица L:
[[1.0000000000 0.0000000000 0.0000000000]
 [5.5000000000 1.0000000000 0.0000000000]
 [4.5000000000 2.3333333333 1.0000000000]]
```

Проверим правильность расчетов перемножением L на U

In [25]:

```
L @ U
```

Out[25]:

```
array([[ 2.0000000000,  1.0000000000,  3.0000000000],
       [11.0000000000,  7.0000000000,  5.0000000000],
       [ 9.0000000000,  8.0000000000,  4.0000000000]])
```

Видим, что матрицы L и U найдены правильно

Решим теперь систему

$$Ly = b:$$

$$\begin{cases} y_1 = 1, \\ 5.5y_1 + y_2 = -6, \\ 4.5y_1 + 2.33y_2 + y_3 = -5. \end{cases}$$

$$\begin{gathered} y_1 = 1, \\ y_2 = -11.5, \\ y_3 = 17.33333333295 \end{gathered}$$

И затем систему

$$Ux = y:$$

$$\begin{cases} 2x_1 + x_2 + 3x_3 = 1, \\ 1.5x_2 - 11.5x_3 = -11.5, \\ 17.33x_3 = 17.33333333295. \end{cases}$$

$$x_3 = 1,$$
$$x_2 = 0,$$
$$x_1 = -1.$$

Произведем проверку, подставив полученные значения переменных в исходную систему:

In [26]:

```
2 * (-1) - 0 + 3*1
```

Out[26]:

```
1
```

In [27]:

```
11 * (-1) - 7 * 0 + 5*1
```

Out[27]:

```
-6
```

In [28]:

```
9 * (-1) - 8 * 0 + 4*1
```

Out[28]:

```
-5
```

Таким образом, видим, что найденное решение верно.

Проверим с помощью библиотеки питона

In [29]:

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

In [30]:

```
np.linalg.solve(A, B)
```

Out[30]:

```
array([-1.0000000000,  0.0000000000,  1.0000000000])
```

Видим,что корни найдены верно

4*. Решить систему линейных уравнений методом Холецкого

$$\begin{cases} 81x_1 - 45x_2 + 45x_3 = 531 \\ -45x_1 + 50x_2 - 15x_3 = -460 \\ 45x_1 - 15x_2 + 38x_3 = 193 \end{cases}$$

In [31]:

```
np.set_printoptions(precision=2, floatmode='fixed')
A = np.array([[81., -45., 45.], [-45., 50., -15.], [45., -15., 38.]])
B = np.array([531., -460., 193.])
A
```

Out[31]:

```
array([[ 81.00, -45.00,  45.00],
       [-45.00,  50.00, -15.00],
       [ 45.00, -15.00,  38.00]])
```

Произведем разложение на LL^T:

$$l_{11} = \sqrt{a_{11}} = \sqrt{81} = 9,$$
$$l_{21} = \frac{a_{21}}{l_{11}} = -5,$$
$$l_{31} = \frac{a_{31}}{l_{11}} = 5,$$

$$l_{22} = \sqrt{a_{22} - l_{21}^2} = 5,$$
$$l_{32} = \frac{1}{l_{22}}(a_{32} - l_{21}l_{31}) = 2,$$

$$l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2} = \sqrt{14 - 8 - 2} = 3.$$

Получили матрицу

In [32]:

```
L = np.array([[9., 0., 0.], [-5., 5., 0.], [5., 2., 3.]])
L
```

Out[32]:

```
array([[ 9.00,  0.00,  0.00],
       [-5.00,  5.00,  0.00],
       [ 5.00,  2.00,  3.00]])
```

Получим L^T

In [33]:

```
L.T
```

Out[33]:

```
array([[ 9.00, -5.00,  5.00],
       [ 0.00,  5.00,  2.00],
       [ 0.00,  0.00,  3.00]])
```

Решим теперь систему

$$Ly = b:$$

$$\begin{cases} 9y_1 = 531, \\ -5y_1 + 5y_2 = -460, \\ 5y_1 + 2y_2 + 3y_3 = 193. \end{cases}$$

$$\begin{aligned} y_1 &= 59, \\ y_2 &= -33, \\ y_3 &= -12. \end{aligned}$$

И затем систему

$$L^T x = y :$$

$$\begin{cases} 9x_1 - 5x_2 + 5x_3 = 59, \\ 5x_2 + 2x_3 = -33, \\ 3x_3 = -12. \end{cases}$$

$$\begin{aligned} x_3 &= -4, \\ x_2 &= -5, \\ x_1 &= 6. \end{aligned}$$

Проверим

In [34]:

```
1 np.linalg.solve(A, B)
```

Out[34]:

```
array([ 6.00, -5.00, -4.00])
```

Видим,что корни найдены верно

5*. Написать на Python программу с реализацией одного из изученных алгоритмов решения СЛАУ.

In [35]:

```
def solv_equations(A):
    for nrow in range(len(A)):
        for i in range(nrow + 1):
            if nrow != len(A) - 1:
                A[nrow + 1] = A[nrow + 1] - A[i] * (A[nrow + 1][i] / A[i][i])
            else:
                break
    print("Диагональная матрица:")
    print(A)
    roots = []
    if np.count_nonzero(A[len(A)-1])==2:
        for i in reversed(range(1, len(A) + 1)):
            x = (A[i - 1][len(A)] / A[i - 1][i - 1])
            roots.append(x)
            A[:, i - 1] = A[:, i - 1] * x
            A[:, len(A)] = A[:, len(A)] - A[:, i - 1]
        print("Данная СЛАУ является совместной со следующими корнями: ", roots[::-1])

    elif np.count_nonzero(A[len(A)-1]) < 2 or np.count_nonzero(A[len(A)-2]) < 2:
        print("Диагональная матрица:")
        print(A)
        print("Данная СЛАУ является несовместной и решений не имеет")

    elif np.count_nonzero(A[len(A)-1]) > 2:
        print("Диагональная матрица:")
        print(A)
        print("Данная СЛАУ является совместной и имеет бесконечное множество решений")
```

Проверим функцию

In [36]:

```
A = np.array([[3., -1., 1., 4.], [2., -5., -3., -17.], [1., 1., -1., 0.]])
A
```

Out[36]:

```
array([[  3.00,  -1.00,   1.00,   4.00],
       [  2.00,  -5.00,  -3.00, -17.00],
       [  1.00,   1.00,  -1.00,   0.00]])
```

In [37]:

```
solv_equations(A)
```

```
Диагональная матрица:
[[  3.00  -1.00   1.00   4.00]
 [  0.00  -4.33  -3.67 -19.67]
 [  0.00   0.00  -2.46  -7.38]]
Данная СЛАУ является совместной со следующими корнями:  [1.0000000000000002,
2.0000000000000004, 3.0]
```

Проверим

In [38]:

```
A = np.array([[3.,  -1.,   1.], [2.,  -5.,  -3.], [1.,   1.,  -1.]])
B = np.array([4., -17., 0.])
```

In [39]:

```
np.linalg.solve(A, B)
```

Out[39]:

```
array([1.00, 2.00, 3.00])
```

Видим,что корни найдены верно