Практическое задание 2.2

In [1]:

```
import numpy as np
from numpy import delete
```

Все задания рекомендуется выполнять вручную, затем проверяя полученные результаты с использованием Numpy.

1. Вычислить определитель:

а)

$$\begin{vmatrix} sinx & -cosx \\ cosx & sinx \end{vmatrix} = \sin x \cdot \sin x + \cos x \cdot \cos x = \sin^2 x + \cos^2 x = 1$$

Проверим

In [2]:

```
b = np.array([[np.sin(1), -np.cos(1)], [np.cos(1), np.sin(1)]])
print(f'Матрица:\n{b}')
print(f'Определитель: {np.linalg.det(b):.0f}')
```

```
Матрица:
[[ 0.84147098 -0.54030231]
 [ 0.54030231  0.84147098]]
Определитель: 1
```

б)

$$\begin{vmatrix} 4 & 2 & 3 \\ 0 & 5 & 1 \\ 0 & 0 & 9 \end{vmatrix};$$

Данная марица является диагональной, поэтому можно воспользоваться следующим свойством: Определитель матрицы треугольного вида равен произведению элементов, стоящих на ее главной диагонали. используя данное свойство получаем^ $\begin{vmatrix} 4 & 2 & 3 \\ 0 & 5 & 1 \\ 0 & 0 & 9 \end{vmatrix} = 4 \cdot 5 \cdot 9 = 180$

Проверим

In [3]:

```
b = np.array([[4, 2, 3], [0, 5, 1], [0, 0, 9]])
print(f'Матрица:\n{b}')
print(f'Определитель: {np.linalg.det(b):.0f}')
```

```
Матрица:
[[4 2 3]
 [0 5 1]
 [0 0 9]]
Определитель: 180
```

Теперь проверим мою функцию из 5 задания

In [4]:

```
def determinant(B):
    global b, k, d, h, p, size, det_2_2, det_1, det_3
    b = B
    k = 0
    d = 1
    h = 0
    p = 0
    det_2_2 = 0
    det_1 = 0
    det_3 = []
    size = len(b[0])
    def det(A):
        global A_2
        global det_2_2
        global det_1, det_3
        global det_3
        global k, d, c, h, p, size
        A_string_1 = A[0]

        for i in range(A.shape[0]):
            if A.shape[0] > 2:
                c = A_string_1[i]
                A_2 = delete(A,[0],0)
                A_2 = delete(A_2,[i],1)
                det(A_2)

            if A.shape[0] == 2:
                if len(b[0]) == 2:
                    c = A_string_1[i]
                    det_2_2 = det_2_2 + ((-1)**(k+1) * c * (A[0][0] * A[1][1] - A[1][0]
                    det_1 =  det_2_2
                    det_3.append(det_1)
                    print("Определитель матрицы =", det_3[-1])

                elif len(b[0]) > 2:
                    h +=1
                    det_2_2 = det_2_2 + ((-1)**(k+1) * c * (A[0][0] * A[1][1] - A[1][0]
                    k += 1
                    if h%3 == 0:
                        if len(b[0]) > 3:
                            det_1 = det_1 + ((-1)**(d+1)*b[0][p])*det_2_2
                            det_3.append(det_1)
                            d += 1
                            p += 1
                        elif len(b[0]) == 3:
                            det_1 = - det_2_2
                            det_3.append(det_1)
                            print("Определитель матрицы =", det_3[-1])
                        if p == len(b[0]):
                            print("Определитель матрицы =", det_3[-1])
                    break

    return det(b)
```

In [5]:

```
1  determinant(b)
```

```
Определитель матрицы = 180
```

Видим, что все работает и посчитано правильно

в)

$$\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}.$$

Поскольку я написал функцию сам, то хорошо разобрался с вычислением определителя. Поэтому для экономии времени воспользуюсь здесь своей функцией.

In [6]:

```
1  b = np.array([[1, 2, 3], [3, 5, 6], [7, 8, 9]])
2  print(f'Матрица:\n{b}')
```

```
Матрица:
[[1 2 3]
 [3 5 6]
 [7 8 9]]
```

In [7]:

```
1  determinant(b)
```

```
Определитель матрицы = -6
```

Проверим

In [8]:

```
1  print(f'Определитель: {np.linalg.det(b):.0f}')
```

```
Определитель: -6
```

Видим, что все работает и посчитано правильно

2. Определитель матрицы A равен 4. Найти:

а) $det(A^2)$;

б) $det(A^T)$;

в) $det(2A)$.

Для выпонения данного упражнения воспользуемя свойствами детермината.

а) $det(A^2)$

Помним, что для двух квадратных матриц одинакового размера

$$det(AB) = detA \cdot detB.$$

Поэтому $det(A^2) = det(A \cdot A) = detA \cdot detA = 4 \cdot 4 = 16.$

б) $det(A^T)$

Помним, что определитель транспонированной матрицы равен определителю исходной:

$$detA^T = detA.$$

.

Поэтому

$detA^T = detA = 4$

в) $det(2A)$
помним, что если квадратная матрица n-того порядка умножается на некоторое ненулевое число, то определитель полученной матрицы равен произведению определителя исходной матрицы на это число в n-той степени.

$$B = k \cdot A => det(B) = k^n \cdot det(A)$$

Поэтому, $det(2A) = 2^n \cdot 2^2 = 2^{n+2}$, где n - порядок матрицы.

3. Доказать, что матрица

$$\begin{pmatrix} -2 & 7 & -3 \\ 4 & -14 & 6 \\ -3 & 7 & 13 \end{pmatrix}$$

вырожденная.

Помним, что матрица являтся вырожденной если ее определитель равен 0. Найдем с помощью моей функции определитель данной матрицы.

In [9]:

```
b = np.array([[-2, 7, -3], [4, -14, 6], [-3, 7, 13]])
print(f'Матрица:\n{b}')
```

```
Матрица:
[[ -2   7  -3]
 [  4 -14   6]
 [ -3   7  13]]
```

In [10]:

```
determinant(b)
```

```
Определитель матрицы = 0
```

Проверим

In [11]:

```
print(f'Определитель: {np.linalg.det(b):.0f}')
```

```
Определитель: 0
```

Видим,что определитель посчитан правильно и равен 0. Следовательно, данная матрица является вырожденной.

4. Найти ранг матрицы:

а) $\begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \\ 2 & 3 & 4 \end{pmatrix}$;

Третья строка является суммой первой и второй строк, а значит, ее можно отбросить:

$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix}$

Вычтем из второй сроки первую:

$\begin{pmatrix} 1 & 2 & 3 \\ 0 & -1 & -2 \end{pmatrix}$

Видим, что ненулевых строк 2, поэтому ранг матрицы равен 2. Проверим:

In [12]:

```
a = np.array([[1, 2, 3], [1, 1, 1], [2, 3, 4]])
print("Ранг матрицы =", np.linalg.matrix_rank(a))
```

```
Ранг матрицы = 2
```

Видим, что все посчитано правильно

б) $\begin{pmatrix} 0 & 0 & 2 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 4 & 3 \\ 2 & 3 & 5 & 6 \end{pmatrix}$.

Третья строка является суммой первой и второй строк, а значит, ее можно отбросить:

$\begin{pmatrix} 0 & 0 & 2 & 1 \\ 0 & 0 & 2 & 2 \\ 2 & 3 & 5 & 6 \end{pmatrix}$

Меняем первую и третью строку местами

$\begin{pmatrix} 2 & 3 & 5 & 6 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 1 \end{pmatrix}$

Полученная матрица является треугольной. Видим, что ненулевых строк 3, поэтому ранг матрицы равен 3. Проверим

In [13]:

```
a = np.array([[0, 0, 2, 1], [0, 0, 2, 2], [0, 0, 4, 3], [2, 3, 5, 6]])
print("Ранг матрицы =", np.linalg.matrix_rank(a))
```

```
Ранг матрицы = 3
```

Видим, что все посчитано правильно

5*. Написать функцию для поиска определителя путем разложения по строкам или столбцам

Функцию привел выше. Проверим ее на матрице размером 4 x 4.

In [14]:

```
b = np.array([[-1, -4, 0, -2], [0, 1, 5, 4], [3, 1, 1, 0], [-1, 0, 2, 2]])
print(f'Матрица:\n{b}')
```

```
Матрица:
[[-1 -4  0 -2]
 [ 0  1  5  4]
 [ 3  1  1  0]
 [-1  0  2  2]]
```

In [15]:

```
determinant(b)
```

```
Определитель матрицы = -12
```

Проверим

In [16]:

```
print(f'Определитель:\n{np.linalg.det(b):.0f}')
```

```
Определитель:
-12
```

Видим, что все работает и посчитано правильно

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
1
```

In [ ]:

```
1
```

Это я оставил для себя с комментариями

In [17]:

```
import numpy as np
from numpy import delete
k = 0
d = 1
h = 0
p = 0
det_2_2 = 0
det_1 = 0
size = len(b[0])
def det(A):
    global A_2
    global det_2_2
    global det_1
    global k, d, c, h, p, size
    A_string_1 = A[0]

    for i in range(A.shape[0]):
        if A.shape[0] > 2:
            c = A_string_1[i]
            print("Работает функция > 2")
            print(A_string_1)
            A_2 = delete(A,[0],0)
            A_2 = delete(A_2,[i],1)
            print(A_2)
            print(f"a_1_{i} = {A_string_1[i]}")
            det(A_2)

        if A.shape[0] == 2:
            print("Работает функция det_1")
            if len(b[0]) == 2:
                c = A_string_1[i]
                print(f"Определитель матрицы 3 x 3 равен {det_2_2} + ((-1)**({k}+1) * 
                det_2_2 = det_2_2 + ((-1)**(k+1) * c * (A[0][0] * A[1][1] - A[1][0] * A
                det_1 =  - det_2_2
                print("Определитель матрицы =", det_1)

            elif len(b[0]) > 2:
                h +=1
                print(A_2)
                print(f"Определитель матрицы 3 x 3 равен {det_2_2} + ((-1)**({k}+1) * 
                det_2_2 = det_2_2 + ((-1)**(k+1) * c * (A[0][0] * A[1][1] - A[1][0] * A
                print(det_2_2)
                k += 1
                print("A_string_1[i] = ", c)
                if h%3 == 0:
                    if len(b[0]) > 3:
                        print(f"Определитель матрицы 4 x 4 равен {det_1} +++-1**({d}+1)
                        det_1 = det_1 + ((-1)**(d+1)*b[0][p])*det_2_2
                        print("p = ", p)
                        print(det_1)
                        d += 1
                        p += 1
                        if p == 5:
                            p = 0


                    elif len(b[0]) == 3:
                        det_1 = - det_2_2
                        print("Определитель матрицы =", det_1)
```

```
            break

```

In [18]:

```
def determinant(B):
    global b, k, d, h, p, size, det_2_2, det_1
    b = B
    k = 0
    d = 1
    h = 0
    p = 0
    det_2_2 = 0
    det_1 = 0
    size = len(b[0])
    def det(A):
        global A_2
        global det_2_2
        global det_1
        global k, d, c, h, p, size
        A_string_1 = A[0]

        for i in range(A.shape[0]):
            if A.shape[0] > 2:
                c = A_string_1[i]
                A_2 = delete(A,[0],0)
                A_2 = delete(A_2,[i],1)
                det(A_2)

            if A.shape[0] == 2:
                if len(b[0]) == 2:
                    c = A_string_1[i]
                    det_2_2 = det_2_2 + ((-1)**(k+1) * c * (A[0][0] * A[1][1] - A[1][0]
                    det_1 =  - det_2_2
                    print("Определитель матрицы =", det_1)

                elif len(b[0]) > 2:
                    h +=1
                    det_2_2 = det_2_2 + ((-1)**(k+1) * c * (A[0][0] * A[1][1] - A[1][0]
                    k += 1
                    if h%3 == 0:
                        if len(b[0]) > 3:
                            det_1 = det_1 + ((-1)**(d+1)*b[0][p])*det_2_2
                            d += 1
                            p += 1
                        elif len(b[0]) == 3:
                            det_1 = - det_2_2
                        print("Определитель матрицы =", det_1)
                    break
    return det(b)
```