

Алгебра матриц. Системы линейных уравнений

Матрицы в библиотеке numpy

```
In [5]: import numpy as np
        from sympy import *
```

```
In [6]: def Minor_elem(A,i, j):
        ''' Вычисляет минор элемента a_ij '''
        m,n = A.shape
        if m != n:
            raise ValueError('Матрица должна быть квадратной')
        if (0 < i <= n) & (0 < j <= n):
            A.row_del(i-1) # нумерация элементов массива с 0
            A.col_del(j-1)
        else:
            raise ValueError('индекс элемента больше размера матрицы')
        return(det(A))
```

```
In [7]: def Algebr_compl(A,i,j):
        m = Minor_elem(A,i,j)
        return (-1)**(i+j)*m
```

```
In [8]: def Algebr_compl(A,i,j):
        m = Minor_elem(A,i,j)
        return (-1)**(i+j)*m
```

```
In [9]: def Minor_Matrix(A,Row,Col):
        n = len(Row)
        m = len(Col)
        if n != m:
            raise ValueError('The quantities of the given \
            rows and columns must be equal')
        if (n < 1) or (n > A.shape[0]):
            raise ValueError('Invalid number of minor rows')
```

```

M_Row = A.row(Row[0]-1)
for i in range(1,n):
    M_Row = M_Row.row_insert(i,A.row(Row[i]-1))
M_Col = M_Row.col(Col[0]-1)
for j in range(1,m):
    M_Col = M_Col.col_insert(j,M_Row.col(Col[j]-1))
return det(M_Col)

```

In [10]:

```

def silvestr(A):

    m,n = A.shape
    if m!=n:
        raise ValueError('Матрица должна быть квадратной')
    M1 = A[0,0]
    if M1 == 0:
        return('Не является знакоопределенной')
    elif M1 > 0: # проверка на положительную определенность
        for k in range(2,n+1):
            Mk = det(A[0:k,0:k])
            if Mk <=0:
                return('Не является знакоопределенной')
            return('Положительно определена')
    else: # проверка на отрицательную определенность
        for k in range(2,n+1):
            Mk = det(A[0:k,0:k])
            if Mk == 0:
                return('Не является знакоопределенной')
            else:
                s1 = M1/abs(M1)
                s2 = Mk/abs(Mk)
                if s1*s2 > 0:
                    return('Не является знакоопределенной')
            M1 = Mk
        return('Отрицательно определена')

```

In [11]:

```

A = np.array([[ -7,4,0],
              [ 0,-1,0],
              [-1,5,7]])

A

```

```
Out[11]: array([[ -7,  4,  0],
               [  0, -1,  0],
               [-1,  5,  7]])
```

```
In [12]: '''Единичная матрица третьего порядка'''
E = np.eye(3)
E
```

```
Out[12]: array([[1., 0., 0.],
               [0., 1., 0.],
               [0., 0., 1.]])
```

```
In [13]: ''' Матрица-строка '''
A = np.array([1,2,3])
A
```

```
Out[13]: array([1, 2, 3])
```

```
In [14]: A = np.array([[-7,4,0],
                       [0,-1,0],
                       [-1,5,7]])
''' Элемент a12 (нумерация с 0) '''
A[0,1]
```

```
Out[14]: 4
```

```
In [15]: ''' Первая строка '''
A[0]
```

```
Out[15]: array([-7,  4,  0])
```

```
In [16]: ''' Столбцы, начиная со второго
и заканчивая третьим '''
A[:,1:3]
```

```
Out[16]: array([[ 4,  0],
                [-1,  0],
                [ 5,  7]])
```

```
In [17]: A = np.array([[1,2,3],
                        [4,5,6] ,
                        [7,8,9] ])
E = np.eye(3)
A-E
```

```
Out[17]: array([[0.,  2.,  3.],
                [4.,  4.,  6.],
                [7.,  8.,  8.]])
```

```
In [18]: 3*A
```

```
Out[18]: array([[ 3,  6,  9],
                [12, 15, 18],
                [21, 24, 27]])
```

Поэлементное умножение - операция *

```
In [19]: A = np.array([[1,2,3],
                        [4,5,6] ]),
B = np.array([[0,0,0],
               [2,2,2]])
A * B
```

```
Out[19]: array([[ 0,  0,  0],
                [ 8, 10, 12]])
```

Транспонирование - метод .T

Пример 1.

$$B = \begin{pmatrix} 1 & 0 \\ 0 & -2 \\ 1 & 1 \end{pmatrix}, \quad B^T = \begin{pmatrix} 1 & 0 & 1 \\ 0 & -2 & 1 \end{pmatrix}.$$

```
In [20]: B = np.array([[1,0],
                        [0,-2],
                        [1,1] ])
B1 = B.T
B1
```

```
Out[20]: array([[ 1,  0,  1],
                [ 0, -2,  1]])
```

Пример 2.

$$A = \begin{pmatrix} -7 & 4 & 0 \\ 0 & -1 & 0 \\ -1 & 5 & 7 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 \\ 0 & -2 \\ 1 & 1 \end{pmatrix}, \quad F = AB,$$

$$f_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} = -7 \cdot 1 + 4 \cdot 0 + 0 \cdot 1 = -7, \dots,$$

$$f_{32} = a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33} = -1 \cdot 5 + 5 \cdot (-2) + 7 \cdot 1 = -3.$$

```
In [21]: A = np.array([[ -7, 4, 0],
                        [ 0, -1, 0],
                        [-1, 5, 7]])
B = np.array([[1,0],
               [0,-2],
               [1,1] ])
F = A@B
F
```

```
Out[21]: array([[ -7, -8],
                [  0,  2],
                [  6, -3]])
```

Определитель матрицы

Пример 3.

$$A = \begin{pmatrix} 7 & -3 \\ 1 & 1 \end{pmatrix}, \quad \det A = 7 \cdot 1 - (-3) \cdot 1 = 10.$$

```
In [22]: A = np.array([[7,-3],
                      [1,1]])
detA = np.linalg.det(A)
detA
```

Out[22]: 9.999999999999998

Пример 4.

$$A^{-1} = \frac{1}{\det A} \begin{pmatrix} 1 & 3 \\ -1 & 7 \end{pmatrix} = \begin{pmatrix} 0,1 & 0,3 \\ -0,1 & 0,7 \end{pmatrix}.$$

```
In [23]: A = np.array([[7,-3],
                      [1,1] ])
A1 = np.linalg.inv(A)
A1
```

Out[23]: array([[0.1, 0.3],
 [-0.1, 0.7]])

Матрицы в библиотеке sympy

Создание матрицы

```
In [24]: a = Matrix([[1,2,3], [0,-1, 1]])
a
```

Out[24]: $\begin{bmatrix} 1 & 2 & 3 \\ 0 & -1 & 1 \end{bmatrix}$

```
In [25]: x,y,z = symbols('x y z')
v = Matrix([[1,x],[y,z]])
v
```

Out[25]: $\begin{bmatrix} 1 & x \\ y & z \end{bmatrix}$

In [26]: `''' Создание единичной матрицы '''
eye(3)`

Out[26]: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

In [27]: `''' Создание нулевой матрицы '''
zeros(2,3)`

Out[27]: $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

In [28]: `'''Создание матрицы, все элементы которой равны 1'''
ones(3,2)`

Out[28]: $\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$

In [29]: `''' Создание диагональной матрицы '''
diag(1,5,-2)`

Out[29]: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & -2 \end{bmatrix}$

In [30]: `''' Элементами диагонали могут быть матрицы '''
diag(-1, ones(2, 2), Matrix([5, 7, 5]))`

```
Out[30]: 
$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 5 \end{bmatrix}$$

```

```
In [31]: ''' Матрица 1x3 (вектор-строка) '''  
A = Matrix([[1,2,3]])  
A
```

```
Out[31]: 
$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

```

```
In [32]: ''' Матрица 3x1 (вектор-столбец) '''  
A = Matrix([[1], [2], [3]])  
A
```

```
Out[32]: 
$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

```

```
In [33]: ''' Отличие от правила в модуле numpy:  
одна пара квадратных скобок приводит к созданию вектор-столбца '''  
A = Matrix([1,2,3])  
A
```

```
Out[33]: 
$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

```

Пример 5. Матрица размера 2×3 (2 строки, 3 столбца):

```
In [34]: A = Matrix([[1,2,3], [0,-1, 1]])  
A.shape
```


Out[34]: (2, 3)

Пример 6. Матрица третьего порядка.

```
In [35]: all, a12, a13, a21, a22, a23, a31, a32, a33 = \
symbols('a11 a12 a13 a21 a22 a23 a31 a32 a33')
A = Matrix([[all, a12, a13],
            [a21, a22, a23],
            [a31, a32, a33]])
A
```

Out[35]:
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

```
In [36]: ''' Элемент в третьей строке, в первом столбце (нумерация с 0)'''
A[2,0]
```

Out[36]: a_{31}

```
In [37]: ''' Второй столбец '''
A[:, 1:2]
```

Out[37]:
$$\begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \end{bmatrix}$$

```
In [38]: A = Matrix([[1,2,3],
[4,5,6] ,
[7,8,9] ])
''' Первая строка '''
A.row(0)
```

Out[38]: $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$

```
In [39]: ''' Второй столбец '''  
A.col(1)
```

Out[39]: $\begin{bmatrix} 2 \\ 5 \\ 8 \end{bmatrix}$

Пример 7.

```
In [40]: A = Matrix([[1,2,3],  
                     [4,5,6] ,  
                     [7,8,9] ])  
A
```

Out[40]: $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

```
In [41]: A.row_del(0)  
A
```

Out[41]: $\begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

```
In [42]: A.col_del(1)  
A
```

Out[42]: $\begin{bmatrix} 4 & 6 \\ 7 & 9 \end{bmatrix}$

Пример 8.

```
In [43]: B = Matrix([[1,2,3],
                    [7,8,9] ])
A = B.row_insert(1, Matrix([[4,5,6]]))
A
```

```
Out[43]: 
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```

```
In [44]: ''' Матрица B не изменилась '''
B
```

```
Out[44]: 
$$\begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \end{bmatrix}$$

```

```
In [45]: D = B.col_insert(3, Matrix([4,10]))
D
```

```
Out[45]: 
$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 7 & 8 & 9 & 10 \end{bmatrix}$$

```

Пример 9.

```
In [46]: V = Matrix([[1,x],[y,z]])
V*V
```

```
Out[46]: 
$$\begin{bmatrix} xy + 1 & xz + x \\ zy + y & xy + z^2 \end{bmatrix}$$

```

Пример 10.

Результат умножения матрицы $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ на матрицу $\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$:

```
In [47]: a11, a12, a21, a22, a31, a32, b11, b12, b21, b22 = \
symbols('a11 a12 a21 a22 a31 a32 b11 b12 b21 b22')
A = Matrix([[a11, a12], [a21, a22], [a31, a32]])
B = Matrix([[b11, b12], [b21, b22]])
A*B
```

```
Out[47]: 
$$\begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \\ a_{31}b_{11} + a_{32}b_{21} & a_{31}b_{12} + a_{32}b_{22} \end{bmatrix}$$

```

Пример 11.

```
In [48]: B = Matrix( [ [b11, b12], [b21, b22]])
B
```

```
Out[48]: 
$$\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

```

```
In [49]: B.T
```

```
Out[49]: 
$$\begin{bmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \end{bmatrix}$$

```

Пример 12.

```
In [50]: D = Matrix([[0,1], [1,0]])  
det(D)
```

Out[50]: -1

Пример 13.
Результат вычисления определителя матриц второго и третьего порядка:

```
In [51]: x11, x12, x21, x22 = symbols('x11 x12 x21 x22')  
X = Matrix([[x11, x12], [x21, x22]])  
det(X)
```

Out[51]: $-x_{12}x_{21} + x_{11}x_{22}$

```
In [52]: y11, y12, y13, y21, y22, y23, y31, y32, y33 = \  
symbols('y11 y12 y13 y21 y22 y23 y31 y32 y33')  
Y = Matrix([[y11, y12, y13], [y21, y22, y23], [y31, y32, y33]])  
det(Y)
```

Out[52]: $y_{11}y_{22}y_{33} - y_{11}y_{23}y_{32} - y_{12}y_{21}y_{33} + y_{12}y_{23}y_{31} + y_{13}y_{21}y_{32} - y_{13}y_{22}y_{31}$

Пример 14.

```
In [53]: A = Matrix( [[ 2,-3,-8],  
                     [-2,-1, 2],  
                     [ 1, 0,-3]] )  
A.inv()
```

Out[53]:

$$\begin{bmatrix} \frac{3}{10} & -\frac{9}{10} & -\frac{7}{5} \\ -\frac{2}{5} & \frac{1}{5} & \frac{6}{5} \\ \frac{1}{10} & -\frac{3}{10} & -\frac{4}{5} \end{bmatrix}$$

In [54]:

```
x11, x12, x21, x22 = symbols('x11 x12 x21 x22')
X = Matrix([[x11, x12], [x21, x22]])
X.inv()
```

Out[54]:

$$\begin{bmatrix} \frac{x_{22}}{x_{11}x_{22}-x_{12}x_{21}} & -\frac{x_{12}}{x_{11}x_{22}-x_{12}x_{21}} \\ -\frac{x_{21}}{x_{11}x_{22}-x_{12}x_{21}} & \frac{x_{11}}{x_{11}x_{22}-x_{12}x_{21}} \end{bmatrix}$$

In [55]:

```
A = Matrix( [[ 2,-3,-8],
               [-2,-1, 2],
               [ 1, 0,-3]] )
A**-1
```

Out[55]:

$$\begin{bmatrix} \frac{3}{10} & -\frac{9}{10} & -\frac{7}{5} \\ -\frac{2}{5} & \frac{1}{5} & \frac{6}{5} \\ \frac{1}{10} & -\frac{3}{10} & -\frac{4}{5} \end{bmatrix}$$

Пример 15.

In [56]:

```
A = Matrix([[2,4,5,6,0,4],
             [8,-2,0,2,4,-2],
             [6,-6,-5,-4,4,-6],
             [-4,0,2,-2,2,0],
             [-2,-4,-5,-6,0,-4],
             [0,1,0,1,0,1]])
A.rank()
```

Out[56]: 4

Пример 16. Найти максимальное число линейно независимых векторов, входящих в систему: $x_1 = (1, 3, 4, 5, 0)$, $x_2 = (4, -1, 0, 1, 2)$, $x_3 = (3, 2, 5, 5, 3)$, $x_4 = (-2, 0, 1, -1, 1)$, $x_5 = (4, 6, 7, 11, -1)$.

Решение. Указанное значение совпадает с рангом матрицы, сформированной из векторов.

In [57]:

```
a = Matrix([[1,3,4,5,0],
            [4,-1,0,1,2],
            [3,2,5,5,3],
            [-2,0,1,-1,1],
            [4,6,7,11,-1]])
a.rank()
```

Out[57]: 3

Пример 17. Найти базис системы векторов $x_1 = (1, 3, 4)$, $x_2 = (4, -1, 0)$, $x_3 = (3, 2, 5)$, $x_4 = (-2, 0, 1)$, $x_5 = (4, 6, 7)$.

Решение. Наберем векторы по строкам матрицы (так легче), а затем матрицу транспонируем.

In [58]:

```
A = Matrix([[1,3,4],
            [4,-1,0],
            [3,2,5],
            [-2,0,1],
            [4,6,7]])
A
```

Out[58]:

$$\begin{bmatrix} 1 & 3 & 4 \\ 4 & -1 & 0 \\ 3 & 2 & 5 \\ -2 & 0 & 11 \\ 4 & 6 & 7 \end{bmatrix}$$

In [59]: `A.T.columnspace()`

Out[59]: [Matrix([
[1],
[3],
[4]]),
Matrix([
[4],
[-1],
[0]]),
Matrix([
[3],
[2],
[5]])]

In [60]: `A.T.rref()[1]`

Out[60]: (0, 1, 2)

Пример 18. Найти минор элемента a_{32} матрицы

$$\begin{pmatrix} 1 & 0 & -3 & 9 \\ 2 & -7 & 11 & 5 \\ -9 & 4 & 25 & 84 \\ 3 & 12 & -5 & 58 \end{pmatrix}.$$

In [61]:

```
A = Matrix([[1,0,-3,9],  
[2,-7,11,5],  
[-9,4,25,84],  
[3,12,-5,58]])  
''' Матрица после удаления 3-й строки и 2-го столбца'''
```



```
M = Matrix([[1,-3,9], [2,11,5], [3,-5,58]])
''' Определитель '''
det(M)
```

Out[61]: 579

Пример 19. Вычислить алгебраическое дополнение A_{32} для матрицы предыдущего примера.

```
In [62]: A = Matrix([[1,0,-3,9],
                    [2,-7,11,5],
                    [-9,4,25,84],
                    [3,12,-5,58]])
Algebr_compl(A, 3,2)
```

Out[62]: -579

Пример 20. Для матрицы из предыдущего примера найти минор, образованный 1-й и 3-й строками и 3-м и 4-м столбцами матрицы A .

```
In [63]: A = Matrix([[1,0,-3,9],
                    [2,-7,11,5],
                    [-9,4,25,84],
                    [3,12,-5,58]])
Row = [1,3]
Col = [3,4]
Minor_Matrix(A, Row, Col)
```

Out[63]: -477

Пример 21. Дана матрица A :

$$\begin{pmatrix} 1 & 3 & 2 & 4 & 5 \\ 0 & 0 & -1 & 2 & 7 \\ 3 & 9 & 6 & 12 & 15 \\ 5 & 15 & 9 & 26 & 22 \\ 1 & 3 & 1 & 10 & 2 \end{pmatrix}.$$

```
In [64]: A = Matrix([[1,3,2,4,5],  
[0,0,-1,2,7],  
[3,9,6,12,15],  
[5,15,9,26,22],  
[1,3,1,10,2]])  
A
```

```
Out[64]:  $\begin{bmatrix} 1 & 3 & 2 & 4 & 5 \\ 0 & 0 & -1 & 2 & 7 \\ 3 & 9 & 6 & 12 & 15 \\ 5 & 15 & 9 & 26 & 22 \\ 1 & 3 & 1 & 10 & 2 \end{bmatrix}$ 
```

```
In [65]: A.rank()
```

```
Out[65]: 3
```

Системы линейных уравнений

Пример 22. Решить систему уравнений

$$\begin{cases} 3x + 2y = 2, \\ x - y = 4, \\ 5y + z = -1. \end{cases}$$

```
In [66]: A = np.array([[3, 2, 0],
[1, -1, 0],
[0, 5, 1]])
b = np.array([2, 4, -1])
u = np.linalg.solve(A,b)
u
```

```
Out[66]: array([ 2., -2.,  9.])
```

```
In [67]: np.dot(A, u) == b
```

```
Out[67]: array([ True,  True,  True])
```

Пример 23. Чтобы решить систему уравнений

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

запишем ее в матричном виде:

$$Ax = b, \text{ где } A = \begin{pmatrix} 3 & 2 & 0 \\ 1 & -1 & 0 \\ 0 & 5 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ 4 \\ -1 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

Тогда решение системы находится из матричного уравнения:

$$x = A^{-1}b.$$

```
In [68]: A = Matrix([[3, 2, 0],
[1, -1, 0],
[0, 5, 1]])
b = Matrix([2, 4, -1])
```

```
x = A.inv()*b
x
```

Out[68]: $\begin{bmatrix} 2 \\ -2 \\ 9 \end{bmatrix}$

Пример 24. Разложить вектор $a = (0,5)$ по системе векторов $f_1 = (2,6), f_2 = (-1,2)$.

```
In [69]: F = Matrix([[2,-1],
                    [6,2]])
a = Matrix([0,5])
x = F.inv()*a
x
```

Out[69]: $\begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix}$

Пример 25. Найти общее решение системы

$$\begin{cases} x_1 + x_2 + 3x_3 = 18, \\ 2x_1 - x_2 + 9x_3 = 30. \end{cases}$$

Решение. Сначала непосредственно выполним задание методом Гаусса.

В этой системе число переменных $m = 3$, число уравнений $n = 2$. Система или несовместная, или неопределенная.

```
In [70]: A = Matrix([[1,1,3],
                    [2,-1,9]])
A.rank()
```

Out[70]: 2

```
In [71]: x1,x2,x3 = symbols('x1 x2 x3')

y1 = x1 + x2 + 3*x3 - 18
y2 = 2*x1 - x2 + 9*x3 - 30

linsolve([y1,y2], [x1,x2])
```

Out[71]: $\{(16 - 4x_3, x_3 + 2)\}$

```
In [72]: A = Matrix([[1, 1, 3, 18],
[2, -1, 9, 30]])
A.rref()
```

Out[72]: $\begin{pmatrix} \text{Matrix}([\\ [1, 0, 4, 16], \\ [0, 1, -1, 2]]), \\ (0, 1) \end{pmatrix}$

```
In [73]: rref_matrix, rref_pivots = A.rref()
rref_matrix
```

Out[73]: $\begin{bmatrix} 1 & 0 & 4 & 16 \\ 0 & 1 & -1 & 2 \end{bmatrix}$

```
In [74]: rref_pivots
```

Out[74]: (0, 1)

Пример 26. Найти общее решение системы

$$\begin{cases} x_1 - 2x_2 + 4x_3 = 6 \\ x_1 - 2x_2 + x_3 = 4 \\ -3x_1 + 6x_2 - 12x_3 = -18. \end{cases}$$

Найдем ранги матриц A и $(A|b)$.

```
In [75]: A = Matrix([[1,-2,4],  
[1,-2,1],  
[-3,6,-12]])  
A.rank()
```

Out[75]: 2

```
In [76]: Ab = Matrix([[1,-2,4,6],  
[1,-2,1,4],  
[-3,6,-12,-18]])  
A.rank()
```

Out[76]: 2

```
In [77]: A = Matrix([[1,-2,4,6],  
[1,-2,1,4],  
[-3,6,-12,-18]])  
rref_matrix, rref_pivots = A.rref()  
rref_matrix
```

Out[77]:
$$\begin{bmatrix} 1 & -2 & 0 & \frac{10}{3} \\ 0 & 0 & 1 & \frac{2}{3} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

```
In [78]: rref_pivots
```

Out[78]: (0, 2)

Пример 27. Решить систему

$$\begin{cases} x + 2y + 3z = 3, \\ 4x + 5y + 6z = 6, \\ 7x + 8y + 10z = 9. \end{cases}$$

Решение. Формируем отдельно основную матрицу системы и вектор свободных членов.

```
In [79]: x, y, z = symbols('x, y, z')
A = Matrix([[1, 2, 3], [4, 5, 6], [7, 8, 10]])
b = Matrix([3, 6, 9])
A
```

Out[79]: $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 10 \end{bmatrix}$

```
In [80]: b
```

Out[80]: $\begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix}$

```
In [81]: linsolve((A, b), [x, y, z])
```

Out[81]: $\{(-1, 2, 0)\}$

Пример 28. Решить систему

$$\begin{cases} x + 2y + 3z = 3, \\ 4x + 5y + 6z = 6, \\ 7x + 8y + 9z = 9. \end{cases}$$

Решение. При использовании функции `linsolve()`, переменные можно задать простым перечислением.

```
In [82]: A = Matrix([[1, 2, 3],  
[4, 5, 6],  
[7, 8, 9]])  
b = Matrix([3, 6, 9])  
linsolve((A, b), x, y, z)
```

```
Out[82]: {(z - 1, 2 - 2z, z)}
```

```
In [83]: linsolve((A, b))
```

```
Out[83]: {(\tau_0 - 1, 2 - 2\tau_0, \tau_0)}
```

Пример 29. В качестве параметров функции можно указывать не матрицы, а уравнения системы (приведенные к нулевой правой части) в виде кортежа. Решение системы

$$\begin{cases} 3x + 2y - z = 1, \\ 2x - 2y + 4z = -2, \\ -x + 0,5y - z = 0 \end{cases}$$


```
In [84]: Eqns = [3*x + 2*y - z - 1, 2*x - 2*y + 4*z + 2, -x + y/2 - z]
linsolve(Eqns, x, y, z)
```

```
Out[84]: {(1, -2, -2)}
```

Пример 30. Вместо указания матриц A и b , можно указать только расширенную матрицы системы. Решение системы:

$$\begin{cases} 2x + y + 3z = 1, \\ 2x + 6y + 8z = 3, \\ 6x + 8y + 18z = 5. \end{cases}$$

```
In [85]: A = Matrix([[2, 1, 3, 1],
[2, 6, 8, 3],
[6, 8, 18, 5]])
linsolve(A, x, y, z)
```

```
Out[85]: {(3/10, 2/5, 0)}
```

Пример 31. Пример решения системы, записанной с произвольными коэффициентами.

$$\begin{cases} ax + by = c, \\ dx + ey = f. \end{cases}$$

```
In [86]: a, b, c, d, e, f = symbols('a b c d e f')
eqns = [a*x + b*y - c, d*x + e*y - f]
linsolve(eqns, x, y)
```

Out[86]: $\left\{ \left(\frac{-bf + ce}{ae - bd}, \frac{af - cd}{ae - bd} \right) \right\}$

Однородные системы уравнений

Пример 32. Решить однородную систему уравнений

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

```
In [87]: A = Matrix([[1,-1,2],
[2,1,-3],
[3,0,2]])
''' Ранг матрицы системы '''
A.rank()
```

Out[87]: 3

Пример 33. Решить однородную систему уравнений

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 0, \\ 4x_1 + 5x_2 + 6x_3 = 0, \\ 7x_1 + 8x_2 + 9x_3 = 0. \end{cases}$$

Записать фундаментальную систему решений.

```
In [88]: A = Matrix([[1,2,3],
[4,5,6],
[7,8,9]])

A.rank()
```

Out[88]: 2

```
In [89]: A = Matrix( [[1,2,3],  
[4,5,6],  
[7,8,9]])  
A.nullspace()
```

```
Out[89]: [Matrix([  
[ 1],  
[-2],  
[ 1]])]
```

Пример 34. Решить однородную систему уравнений

$$\begin{cases} x_1 + 3x_2 + 4x_3 - 2x_4 = 0, \\ 5x_2 + 7x_3 - 4x_4 = 0, \\ x_1 + 8x_2 + 11x_3 - 6x_4 = 0, \\ -x_1 + 2x_2 + 3x_3 - 2x_4 = 0. \end{cases}$$

Записать фундаментальную систему решений.

```
In [90]: A = Matrix([[1,3,4, -2],  
[0,5,7,-4],  
[1,8,11,-6],  
[-1,2,3,-2]])  
  
A.rank()
```

Out[90]: 2

Преобразование координат вектора при переходе к новому базису

Пример 35. Пусть вектор x и базис векторов e'_1, e'_2, e'_3 заданы своими координатами в некотором базисе e_1, e_2, e_3 :

$$x = \begin{pmatrix} -2 \\ 3 \\ 1 \end{pmatrix}, \quad e'_1 = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}, \quad e'_2 = \begin{pmatrix} -2 \\ 0 \\ 3 \end{pmatrix}, \quad e'_3 = \begin{pmatrix} -1 \\ 1 \\ -1 \end{pmatrix}.$$

Разложить вектор x по базису e'_1, e'_2, e'_3 .

Решение. Разложить – означает найти координаты вектора x в базисе e'_1, e'_2, e'_3 , то есть нужно решить уравнение $x' = T^{-1}x$.

Сформировать матрицу T из векторов e'_1, e'_2, e'_3 можно следующим образом: составить матрицу, строками которой являются вектор-строки e'_1, e'_2, e'_3 , а затем ее транспонировать:

In [91]:

```
x = Matrix([-2,3,1])
e1 = Matrix([1,2,-1])
e2 = Matrix([-2,0,3])
e3 = Matrix([-1,1,-1])
T = Matrix([e1.T,e2.T,e3.T]).T
T
```

Out[91]:

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

In [92]:

```
xn = T.inv()*x
xn
```

Out[92]: $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

Пример 36. В базисе e_1, e_2 оператор \hat{A} задается матрицей $A = \begin{pmatrix} 2 & -3 \\ 1 & -4 \end{pmatrix}$. Найти матрицу этого оператора в базисе $e'_1 = 3e_1 + 2e_2$, $e'_2 = -e_1 + 5e_2$.

Решение. Столбцами матрицы перехода T от старого базиса к новому являются коэффициенты разложения векторов старого базиса по новому базису:

$$T = \begin{pmatrix} 3 & -1 \\ 2 & 5 \end{pmatrix}.$$

```
In [93]: T = Matrix([[3,-1],
[2,5]])
T1 = T.inv()
T1
```

Out[93]: $\begin{bmatrix} \frac{5}{17} & \frac{1}{17} \\ -\frac{2}{17} & \frac{3}{17} \end{bmatrix}$

```
In [94]: A = Matrix([[2,-3],
[1,-4]])
T1*A*T
```

Out[94]: $\begin{bmatrix} -\frac{5}{17} & -\frac{106}{17} \\ -\frac{15}{17} & -\frac{29}{17} \end{bmatrix}$

Собственные векторы

Пример 37.

```
In [95]: A = np.array([[3,6],  
[1, 4]])  
np.linalg.eig(A)
```

```
Out[95]: (array([1., 6.]),  
array([[-0.9486833 , -0.89442719],  
[ 0.31622777, -0.4472136 ]]))
```

```
In [96]: L,V = np.linalg.eig(A)  
L
```

```
Out[96]: array([1., 6.])
```

```
In [97]: V[:,0]
```

```
Out[97]: array([-0.9486833 ,  0.31622777])
```

```
In [98]: V[:,1]
```

```
Out[98]: array([-0.89442719, -0.4472136 ])
```

Собственные векторы в библиотеке sympy.

```
In [99]: A = Matrix([[3,6],  
[1, 4]])  
A.eigenvals()
```

```
Out[99]: {6: 1, 1: 1}
```

```
In [100]: list(A.eigenvals().keys())
```

Out[100... [6, 1]

Пример 39.

```
In [101... A = Matrix([[3,6],  
[1, 4]])  
A.eigenvects()
```

```
Out[101... [(1,  
1,  
[Matrix(  
[-3],  
[ 1]])]),  
(6,  
1,  
[Matrix(  
[2],  
[1]])])] ]
```

```
In [102... A.eigenvects()[0][2]
```

```
Out[102... [Matrix(  
[-3],  
[ 1]])]
```

```
In [103... [list(t[2][0]) for t in A.eigenvects()]
```

```
Out[103... [[-3, 1], [2, 1]]
```

Характеристический многочлен

Пример 40. Найти характеристический многочлен матрицы

$$A = \begin{pmatrix} 3 & -2 & 4 & -2 \\ 5 & 3 & -3 & -2 \\ 5 & -2 & 2 & -2 \\ 5 & -2 & -3 & 3 \end{pmatrix}.$$

In [104...

```
A = Matrix([[3, -2, 4, -2],
            [5, 3, -3, -2],
            [5, -2, 2, -2],
            [5, -2, -3, 3]])
lamda = symbols('lamda')
p = A.charpoly(lamda)
p
```

Out[104... PurePoly ($\lambda^4 - 11\lambda^3 + 29\lambda^2 + 35\lambda - 150, \lambda, domain = \mathbb{Z}$)

In [105...

```
factor(p)
```

Out[105... PurePoly ($\lambda^4 - 11\lambda^3 + 29\lambda^2 + 35\lambda - 150, \lambda, domain = \mathbb{Z}$)

In [106...

```
A.eigenvals()
```

Out[106... {3: 1, -2: 1, 5: 2}

Приведение матрицы линейного оператора к диагональному виду

Пример 41. Привести матрицу A к диагональному виду.

$$A = \begin{pmatrix} 3 & -2 & 4 & -2 \\ 5 & 3 & -3 & -2 \\ 5 & -2 & 2 & -2 \\ 5 & -2 & -3 & 3 \end{pmatrix}.$$

```
In [107... A = Matrix([[3, -2, 4, -2],  
[5, 3, -3, -2],  
[5, -2, 2, -2],  
[5, -2, -3, 3]])  
T, D = A.diagonalize()
```

```
In [108... T
```

```
Out[108...  $\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$ 
```

```
In [109... D
```

```
Out[109...  $\begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}$ 
```

```
In [110... T*D*T**-1
```

Out[110...
$$\begin{bmatrix} 3 & -2 & 4 & -2 \\ 5 & 3 & -3 & -2 \\ 5 & -2 & 2 & -2 \\ 5 & -2 & -3 & 3 \end{bmatrix}$$

Квадратичные формы

Пример 43. По матрице A записать квадратичную форму

$$A = \begin{pmatrix} -1 & 1 & 2 \\ 1 & -3 & 5 \\ 2 & 5 & -2 \end{pmatrix}$$

```
In [111... A = Matrix([[ -1,1,2],  
              [1,-3,5],  
              [2,5,-2]])  
x1,x2,x3 = symbols('x1 x2 x3')
```

```
In [112... X = Matrix([[x1,x2,x3]])  
Q = X*A*X.T  
Q.simplify()  
Q
```

Out[112...
$$[-x_1^2 + 2x_1x_2 + 4x_1x_3 - 3x_2^2 + 10x_2x_3 - 2x_3^2]$$

Пример 44. Исследовать на знакоопределенность квадратичную форму

```
In [113... A = Matrix( [[ -1,1,2],  
                [1,-3,5],  
                [2,5,-2]])  
silvestr(A)
```

Out[113... 'Положительно определена'

Приведение квадратичной формы к каноническому виду

Пример 45. Привести квадратичную форму Q к каноническому виду методом собственных значений и найти соответствующую матрицу перехода.

$$Q(x_1, x_2) = -2x_1^2 + 4x_1x_2 + x_2^2.$$

```
In [114... A = Matrix([[ -2, 2],  
                [ 2, 1]])  
  
T,D = A.diagonalize()  
T
```

Out[114... $\begin{bmatrix} -2 & 1 \\ 1 & 2 \end{bmatrix}$

```
In [115... D
```

Out[115... $\begin{bmatrix} -3 & 0 \\ 0 & 2 \end{bmatrix}$

Пример 46. Предприятие выпускает ежедневно пять видов продукции. Показатели процесса производства приведены в таблице:

Вид изделия	Число изделий	Расход сырья, (кг/изд.)	Норма времени изготовления (ч/изд.)	Цена изделия, ден.ед/изд.
1	30	5	7	45
2	60	3	10	20
3	40	7	8	50
4	80	2	15	25
5	50	4	8	30

```
In [116... q = Matrix([30,60,40,80,50])
r = Matrix([5,3,7,2,4])
t = Matrix([7,10,8,15,8])
p = Matrix([45,20,50,25,30])
```

```
In [117... R = q.T*r
R
```

```
Out[117... [970]
```

```
In [118... T = q.T*t
T
```

```
Out[118... [2730]
```

```
In [119... P = q.T*p
P
```

Out[119... [8050]

Пример 47. В таблице приведены данные о производительности 5 предприятий, которые выпускают 4 вида продукции с потреблением трех видов сырья, а также продолжительность работы всех предприятий и цена каждого вида сырья.

Вид изделия	Производительность предприятий, изд./день					Затраты сырья, ед.веса/изд		
	1	2	3	4	5	1	2	3
1	3	5	4	4	6	4	3	2
2	4	2	3	5	2	2	1	5
3	2	3	5	2	4	6	4	4
4	7	4	2	8	3	3	5	2
	Число рабочих дней					Цена сырья		
	120	200	150	170	220	60	80	50

In [120...

```
Q = Matrix([[3,5,4,4,6],
[4,2,3,5,2] ,
[2,3,5,2,4],
[7,4,2,8,3] ])
N = Matrix([120,200,150,170,220]).T
B = Matrix([[4,2,6,3],
[3,1,4,5],
[2,5,4,2]])
p = Matrix([60,80,50]).T
```

In [121...

```
Qy = zeros(4,5)
for j in range(0,5):
    for i in range(0,4):
```

```
Qy[i, j] = Q[i, j] * N[j]  
Qy
```

```
Out[121...  $\begin{bmatrix} 360 & 1000 & 600 & 680 & 1320 \\ 480 & 400 & 450 & 850 & 440 \\ 240 & 600 & 750 & 340 & 880 \\ 840 & 800 & 300 & 1360 & 660 \end{bmatrix}$ 
```

```
In [122... BQ = B*Q  
BQ
```

```
Out[122...  $\begin{bmatrix} 53 & 54 & 58 & 62 & 61 \\ 56 & 49 & 45 & 65 & 51 \\ 48 & 40 & 47 & 57 & 44 \end{bmatrix}$ 
```

```
In [123... BQy = zeros(3,5)  
for j in range(0,5):  
    for i in range(0,3):  
        BQy[i,j] = BQ[i,j]*N[j]  
BQy
```

```
Out[123...  $\begin{bmatrix} 6360 & 10800 & 8700 & 10540 & 13420 \\ 6720 & 9800 & 6750 & 11050 & 11220 \\ 5760 & 8000 & 7050 & 9690 & 9680 \end{bmatrix}$ 
```

```
In [124... P = p*BQy  
P
```

```
Out[124... [1207200 1832000 1414500 2000900 2186800]
```

Примеры решения задач

Найти $A - A^T$, если $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$.

In [125...

```
A = Matrix([[1,2], [3,4]])  
A - A.T
```

Out[125...

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Найти AB и BA , если $A = \begin{pmatrix} 1 & 2 \\ 4 & -1 \end{pmatrix}$, $B = \begin{pmatrix} 2 & -3 \\ -4 & 1 \end{pmatrix}$.

In [126...

```
A = Matrix([[1,2], [4,-1]])  
B = Matrix([[2,-3], [-4,1]])  
A * B
```

Out[126...

$$\begin{bmatrix} -6 & -1 \\ 12 & -13 \end{bmatrix}$$

In [127...

```
B * A
```

Out[127...

$$\begin{bmatrix} -10 & 7 \\ 0 & -9 \end{bmatrix}$$

Квадрат ненулевой матрицы, в отличие от чисел, может быть нулевым. Проверить равенство:

$$\begin{pmatrix} 2 & 1 \\ -4 & -2 \end{pmatrix}^2 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

In [128...

```
A = Matrix([[2,1], [-4,-2]])  
A**2
```

Out[128... $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

Пусть $f(x) = x^3 - 5x^2 + 3x$, $A = \begin{pmatrix} 2 & -1 \\ -3 & 3 \end{pmatrix}$. Найти $f(A)$.

```
In [129...  
x = symbols('x')  
y = x**3 - 5*x**2 + 3*x  
A = Matrix([[2,-1], [-3,3]])  
y.subs(x,A)
```

Out[129... $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

Вычислить $\begin{pmatrix} 2 & -1 \\ 3 & -2 \end{pmatrix}^n$.

```
In [130...  
A = Matrix([[2,-1],  
[3,-2]])  
n = symbols('n')  
A**n
```

Out[130... $\begin{bmatrix} \frac{3}{2} - \frac{(-1)^n}{2} & \frac{(-1)^n}{2} - \frac{1}{2} \\ \frac{3}{2} - \frac{3(-1)^n}{2} & \frac{3(-1)^n}{2} - \frac{1}{2} \end{bmatrix}$

Вычислить \sqrt{A} , где $A = \begin{pmatrix} 20 & -4 \\ 4 & 12 \end{pmatrix}$.

```
In [131...  
A = Matrix([[20,-4],  
[4,12]])  
A**(1/2)
```


Out[131... $\begin{bmatrix} 4.5 & -0.5 \\ 0.5 & 3.5 \end{bmatrix}$

Вычислить определитель $\begin{vmatrix} 1 & -2 & 1 \\ 2 & 1 & 4 \\ 3 & 5 & 1 \end{vmatrix}$.

```
In [132... A = Matrix([[1,-2,1], [2,1,4], [3,5,1]])  
det(A)
```

Out[132... -32

Решить уравнение $\begin{vmatrix} x^2 - 4 & 4 \\ x - 2 & x + 2 \end{vmatrix} = 0$.

Решение. Для решения уравнения используем функцию solve().

```
In [133... x = symbols('x')  
A = Matrix([[x**2-4, 4], [x-2, x+2]])  
solve(det(A),x)
```

Out[133... [-4, 0, 2]

Найти $(A^{-1})^T$ и $(A^T)^{-1}$, если $A = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 1 & 0 \\ 2 & 2 & 1 \end{pmatrix}$.

```
In [134... A = Matrix([[1,0,-1], [2,1,0], [2,2,1]])  
A.inv().T
```

Out[134...
$$\begin{bmatrix} -1 & 2 & -2 \\ 2 & -3 & 2 \\ -1 & 2 & -1 \end{bmatrix}$$

In [135... `A.T.inv()`

Out[135...
$$\begin{bmatrix} -1 & 2 & -2 \\ 2 & -3 & 2 \\ -1 & 2 & -1 \end{bmatrix}$$

Найти ранг матрицы $\begin{pmatrix} 1 & 4 & 4 & 3 \\ 2 & 6 & 4 & 0 \\ 2 & -8 & 1 & 10 \\ 5 & 10 & 5 & 5 \end{pmatrix}$.

In [136... `A = Matrix([[1,4,4,3], [2,6,4,0], [2,-5,-3,2], [5,5,5,5]])`
`A.rank()`

Out[136... 3

Являются ли векторы $\mathbf{a} = (-1, 0, 1)$, $\mathbf{b} = (2, -1, 0)$ и $\mathbf{c} = (3, 2, -1)$ линейно независимыми?

In [137... `Matrix([[-1,0,1], [2,-1,0], [3,2,-1]]).rank()`

Out[137... 3

Решить систему уравнений

$$\begin{cases} 2x + 3y - z = 3, \\ 3x + 4y - 2z = 5, \\ x + 2y - 3z = 6. \end{cases}$$

In [138...

```
A = np.array([[2, 3, -1],
[3, 4, -2],
[1, 2, -3]])
b = np.array([3, 5, 6])
w = np.linalg.solve(A, b)
w
```

Out[138... array([-0.33333333, 0.66666667, -1.66666667])

Найти общее и базисное решения системы

$$\begin{cases} x_1 + 3x_2 + 4x_3 - 2x_4 = -2, \\ 5x_2 + 7x_3 - 4x_4 = 4, \\ x_1 + 8x_2 + 11x_3 - 6x_4 = 2, \\ -x_1 + 2x_2 + 3x_3 - 2x_4 = 6. \end{cases}$$

In [139...

```
A = Matrix([[1,3,4,-2],
[0,5,7,-4],
[1,8,11,-6],
[-1,2,3,-2]])
b = Matrix([-2,4,2,6])
Ab = Matrix([[1,3,4,-2,-2],
[0,5,7,-4,4],
[1,8,11,-6,2],
[-1,2,3,-2,6]])
```

In [140...

```
x1,x2,x3,x4, = symbols('x1 x2 x3 x4')
gensolve = linsolve((A,b), [x1,x2,x3,x4])
gensolve
```

Out[140...

$$\left\{ \left(\frac{x_3}{5} - \frac{2x_4}{5} - \frac{22}{5}, -\frac{7x_3}{5} + \frac{4x_4}{5} + \frac{4}{5}, x_3, x_4 \right) \right\}$$

Предприятие производит продукцию четырех видов P_1, P_2, P_3, P_4 , и использует сырье пяти типов S_1, S_2, S_3, S_4, S_5 . Нормы затрат сырья (по строкам) на единицу продукции каждого вида (по столбцам) заданы матрицей

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

Стоимость единицы сырья каждого типа S_i задана матрицей

$$B = (25 \quad 10 \quad 18 \quad 20 \quad 15).$$

Каковы общие затраты предприятия на производство 200, 300, 250 и 350 единиц продукции вида P_1, P_2, P_3, P_4 соответственно?

In [141]...

```
A = Matrix([[3,2,5,2],
[1,6,3,0],
[5,0,4,5],
[2,4,1,3] ,
[4,1,0,4]])
B = Matrix([25,10,18,20,15])
Q = Matrix([200,300,250,350])
```

In [142]...

B

Out[142... $\begin{bmatrix} 25 \\ 10 \\ 18 \\ 20 \\ 15 \end{bmatrix}$

In [143...
`P = B.T*A`
`P`

Out[143... $[275 \quad 205 \quad 247 \quad 260]$

In [144...
`P*Q`

Out[144... $[269250]$

Индивидуальное задание

Решения СЛАУ из линейной алгебры

Попробуем решить систему одним из уже известных методов:

Методы, известные из курса линейной алгебры:

- Метод Гаусса - сведение матрицы к верхнетреугольной матрице. Сложность $O(n^3)$
- Правило Крамера - $x_i = \frac{\Delta_i}{\Delta}$, где Δ - определитель. Сложность вычисления определителя $n!$, а посчитать необходимо $n + 1$ определитель.
- Нахождение обратной матрицы - $\mathbf{Ax} = \mathbf{b}$, откуда $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. Сложность $O(n^3)$ (при нахождении методом Гаусса) или $O(n^2)O_{det}$ (при использовании алгебраических дополнений). Здесь O_{det} - сложность вычисления определителя матрицы.

Погрешность вычислений

Зачастую данные для задачи получают в ходе эксперимента. В таком случае данные никогда не будут точными, в них всегда будет некоторая случайная ошибка.

$$\mathbf{Ax} = \mathbf{b} + \delta\mathbf{b}$$

Пусть x - точное значение величины, а x^* - измеренное.

Введем несколько определений :

- Абсолютная погрешность величины x^*

$$\Delta(x^*) \geq |x^* - x|$$

- Относительная погрешность величины x^*

$$\delta(x^*) \geq \frac{|x^* - x|}{x^*}$$

Рассмотрим различные возможные погрешности:

- Возмущение входных данных
- Погрешность метода. Рассмотреть на примере ряда Тейлора.
- Ошибки вычислений

Приближенное представление чисел в компьютере один из основных источников вычислительных погрешностей. Число x , не представимое в компьютере заменяется близким числом x^* , представимым в компьютере точно. Вычислим эту погрешность. Рассмотрим простейшее округление – отбрасывание всех разрядов числа, выходящих за пределы разрядной сетки. Система счисления – двоичная. Надо записать бесконечную двоичную дробь : $x = \pm 2^p(0, \alpha_1\alpha_2\alpha_3\dots)$, где α_k либо 1 либо 0.

Это число можно представить в виде суммы $x = \pm 2^p(\sum_{i=1}^{\infty} \frac{\alpha_i}{2^i})$

Записываются только t двоичных разрядов, тогда

$$|x - x^*| \leq 2^p \frac{1}{2^{t+1}} (1 + \frac{1}{2} + \dots) = 2^{p-t}$$

И относительная погрешность:

$$\delta(x^*) \leq 2^{1-t}.$$

На практике используются более продвинутые методы, для которых в среднем верно:

$$\delta(x^*) \leq 2^{-t}$$

Эти методы основаны на том, чтобы округлять не вниз всегда, а случайным образом. Например, последнюю цифру записывать как операцию XOR над всеми остальными двоичными цифрами (то есть 1, если цифр 1 нечётное количество, 0 в противоположном случае).

В итоге приближенное представление числа можно записать в виде $x^* = x(1 + \varepsilon)$, где ε - машинный эпсилон.

Погрешности арифметических операций:

- при сложении и вычитании чисел необходимо складывать их абсолютные погрешности;
- при сложении большого количества чисел, погрешности которых имеют независимый и случайный характер, целесообразнее пользоваться статистической оценкой относительной погрешности суммы: $S = x_1 + x_2 + \dots + x_n; \Rightarrow \Delta_S = \sqrt{\Delta_1^2 + \Delta_2^2 + \dots + \Delta_n^2}$
- при делении и умножении чисел требуется сложить относительные погрешности;
- при возведении в степень относительную погрешность умножают на показатель степени.
- при перемножении большого количества чисел, погрешности которых имеют независимый и случайный характер, целесообразнее пользоваться статистической оценкой относительной погрешности произведения: $\Pi = x_1 x_2 \dots x_n; \Rightarrow \delta_\Pi = \sqrt{\delta_1^2 + \delta_2^2 + \dots + \delta_n^2}$

Норма вектора

Обобщим длину вектора (расстояние между двумя точками). Такое обобщение называется нормой.

Норма - функционал $\mu(\mathbf{x})$ заданный на векторном пространстве \mathbb{R}^n , для которого выполняются утверждения:

- $\mu(\mathbf{x}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$
- $\mu(\alpha\mathbf{x}) = |\alpha|\mu(\mathbf{x})$
- $\mu(\mathbf{x} + \mathbf{x}) \leq \mu(\mathbf{x}) + \mu(\mathbf{x})$

Норма обозначается $\|\mathbf{x}\| = \mu(\mathbf{x})$

Рассмотрим семейство норм $\|\mathbf{x}\|_p = \left(\sum_{i=1}^n |x_i|^p\right)^{\frac{1}{p}}$

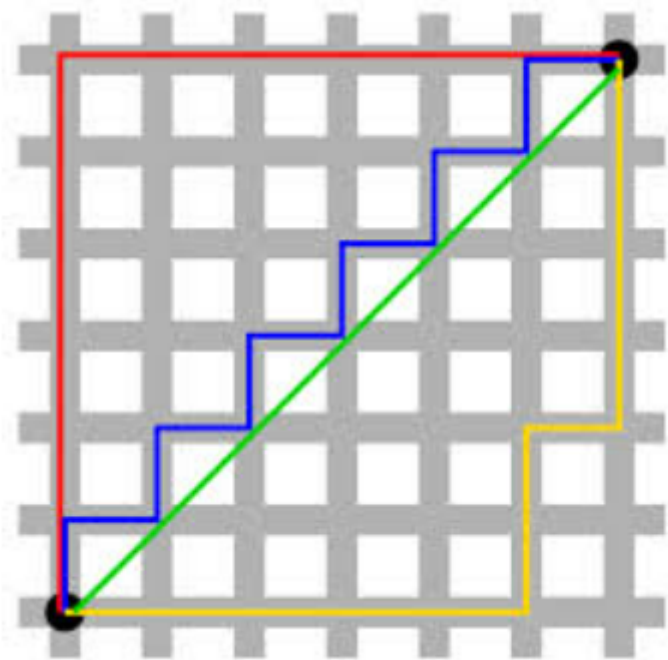
- $\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$ - L1 норма, манхэттенское расстояние, октаэдрическая .
- $\|\mathbf{x}\|_2 = \left(\sum_{i=1}^n x_i^2\right)^{\frac{1}{2}}$ - L2 норма, евклидова норма.
- $\|\mathbf{x}\|_\infty = \max_i |x_i|$ - кубическая норма.

Теорема о константе эквивалентности : Рассмотрим векторное пространство \mathbb{R}^n . Тогда :

$$\forall p, q \exists C_1, C_2 > 0 \forall x \in \mathbb{R}^n : C_1 \|\mathbf{x}\|_p \leq \|\mathbf{x}\|_q \leq C_2 \|\mathbf{x}\|_p$$

$$\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$$

Пример визуализации нормы.



In [24]:

```
import numpy as np
n = 100
a = np.ones(n)
```



```

b = a + 1e-3 * np.random.randn(n)
print('Relative error in L1 norm:', np.linalg.norm(a - b, 1) / np.linalg.norm(b, 1))
print('Relative error in L2 norm:', np.linalg.norm(a - b) / np.linalg.norm(b))
print('Relative error in Chebyshev norm:', np.linalg.norm(a - b, np.inf) / np.linalg.norm(b, np.inf))

```

Relative error in L1 norm: 0.0007643233990701784
 Relative error in L2 norm: 0.0009661431571276279
 Relative error in Chebyshev norm: 0.003218462784199595

Единичный диск – это множество точек такое, что

$$\|\mathbf{x}\| \leq 1$$

Для евклидовой нормы единичный диск совпадает с обычным диском.

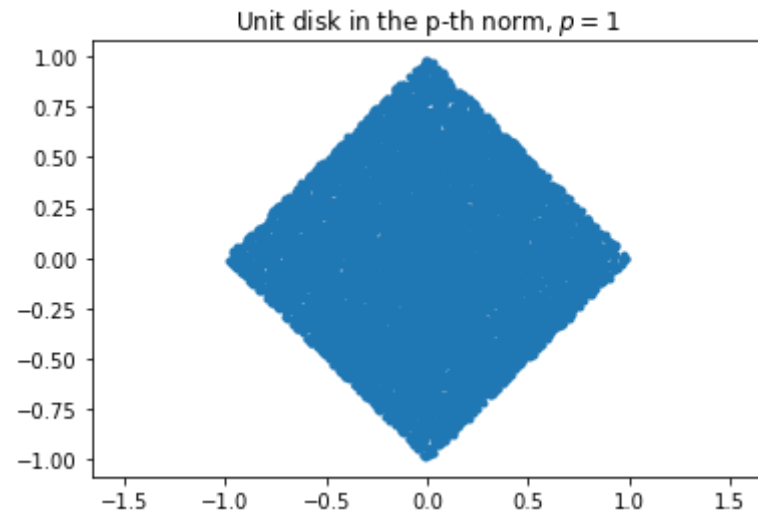
Для других норм единичный диск сильно отличается от привычного нам диска.

```

In [23]: %matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
p = 1 # Which norm do we use
M = 40000 # Number of sampling points
a = np.random.randn(M, 2)
b = []
for i in range(M):
    if np.linalg.norm(a[i, :], p) <= 1:
        b.append(a[i, :])
b = np.array(b)
plt.plot(b[:, 0], b[:, 1], '.')
plt.axis('equal')
plt.title('Unit disk in the p-th norm, $p={0:$}'.format(p))

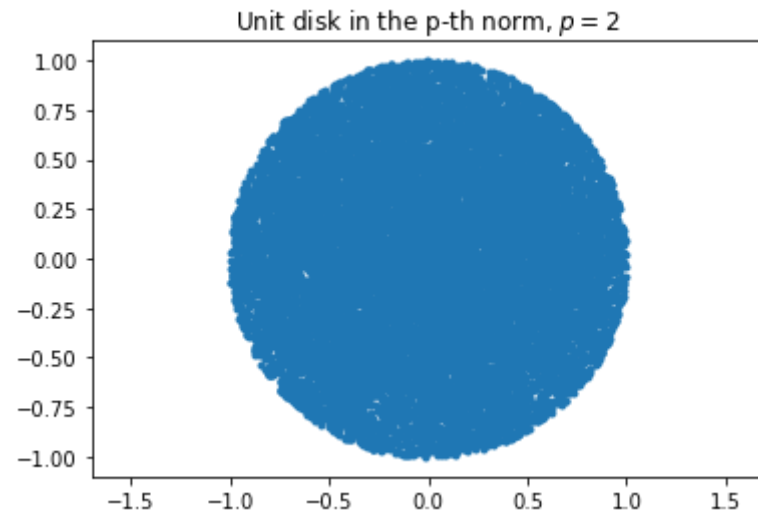
```

Out[23]: Text(0.5, 1.0, 'Unit disk in the p-th norm, \$p=1\$')



```
In [25]: %matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
p = 2 # Which norm do we use
M = 40000 # Number of sampling points
a = np.random.randn(M, 2)
b = []
for i in range(M):
    if np.linalg.norm(a[i, :], p) <= 1:
        b.append(a[i, :])
b = np.array(b)
plt.plot(b[:, 0], b[:, 1], '.')
plt.axis('equal')
plt.title('Unit disk in the p-th norm, $p=2$')
```

Out[25]: Text(0.5, 1.0, 'Unit disk in the p-th norm, \$p=2\$')



```
In [26]: %matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
p = 3 # Which norm do we use
M = 40000 # Number of sampling points
a = np.random.randn(M, 2)
b = []
for i in range(M):
    if np.linalg.norm(a[i, :], p) <= 1:
        b.append(a[i, :])
b = np.array(b)
plt.plot(b[:, 0], b[:, 1], '.')
plt.axis('equal')
plt.title('Unit disk in the p-th norm, $p={0}$'.format(p))
```

Out[26]: Text(0.5, 1.0, 'Unit disk in the p-th norm, \$p=3\$')

