In [1]:
```python
import sympy
from sympy import  Matrix, symbols
from sympy.vector import Vector, matrix_to_vector, AxisOrienter, express
```

Занятие 11

Алгебра

Векторы

Выражение вектора через его координаты неразрывно связано с координатной системой, которой эти координаты определяются, поэтому в sympy для работы с векторами необходимо прежде всего ввести систему координат.

Система координат вводится так:

In [2]:
```python
from sympy.vector import CoordSys3D
N = CoordSys3D('N')
N
```

Out[2]:

$$CoordSys3D\left(N, \left(\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}, \hat{\mathbf{0}}\right)\right)$$

Имя 'N' используется для в основном для выведения на печать, математический смысл ему не придается.

Введя систему кооринат, мы получаем доступ к ее ортам (ортонормированным базисным векторам):

In [3]:
```python
display(N.i, N.j, N.k, 2*N.i + 3*N.j - 5*N.k)
```

$\hat{\mathbf{i}}_{\mathbf{N}}$

$\hat{\mathbf{j}}_{\mathbf{N}}$

$\hat{\mathbf{k}}_{\mathbf{N}}$

$(2)\hat{\mathbf{i}}_{\mathbf{N}} + (3)\hat{\mathbf{j}}_{\mathbf{N}} + (-5)\hat{\mathbf{k}}_{\mathbf{N}}$

Нулевой вектор Vector.zero:

In [4]:
```python
Vector.zero
```

Out[4]: $\hat{\mathbf{0}}$

Пример 1.

Найти скалярное и векторное произведение векторов $a(-1, 3, 7)$ и $b(9, -2, 2)$.

Зададим векторы в линейных комбинаций ортов:

In [5]:
```python
a = -N.i + 3*N.j + 7*N.k
b = 9*N.i - 2*N.j + 2*N.k
display(a.dot(b), a.cross(b), b.cross(a))
```

-1

$(20)\hat{\mathbf{i}}_{\mathbf{N}} + (65)\hat{\mathbf{j}}_{\mathbf{N}} + (-25)\hat{\mathbf{k}}_{\mathbf{N}}$

$(-20)\hat{\mathbf{i}}_{\mathbf{N}} + (-65)\hat{\mathbf{j}}_{\mathbf{N}} + (25)\hat{\mathbf{k}}_{\mathbf{N}}$

Для того, чтобы имя системы координат не отражалось индексами ортов, введем безымянную систему координат:

In [6]:
```python
M = CoordSys3D('')
display(M.i, M.j, M.k, 2*M.i + 3*M.j - 5*M.k)
```

$\hat{\mathbf{i}}$

$\hat{\mathbf{j}}$

$\hat{\mathbf{k}}$

$(2)\hat{\mathbf{i}} + (3)\hat{\mathbf{j}} + (-5)\hat{\mathbf{k}}$

Для более компактной записи скалярного и векторного произведения в sympy использована перегрузка операторов & и ^. Эти операторы удобно использовать в громоздких выражениях, а для небольших выражений рекомендуется использовать более понятные средства Примера 1.

Пример 2.

Найти скалярное и векторное произведение векторов $v + u$ и $3v - 2u$, $v(0, -3, 2)$, $u(-9, 2, 1)$. Использовать & и ^.

In [7]:
```python
u = -9*M.i + 2*M.j + M.k
v = - 3*M.j + 2*M.k
display((v + u) & (3*v - 2*u), (v + u) ^ (3*v - 2*u))
```

-137

$(35)\hat{\mathbf{i}} + (90)\hat{\mathbf{j}} + (135)\hat{\mathbf{k}}$

Действия с векторами

Разложить на множители координаты вектора можно с помощью factor, упростить выражения координат, содержащие тригонометрические функции, можно с помощью trigsimp

Пример 3.

Упростить вектор $g(a^3 - 3a^2 + 3a - 1, a^2 - b^2, \sin^2(a) + \cos^2(a)$.

In [8]:
```python
from sympy.abc import a, b
g = (a**3 - 3*a**2 + 3*a - 1)*M.i + (a**2 - b**2)*M.j +(sympy.sin(a)**2 + sympy.cos(a)**2)*M.k
display(g.factor().trigsimp())
```

$((a-1)^3)\hat{\mathbf{i}} + ((a-b)(a+b))\hat{\mathbf{j}} + \hat{\mathbf{k}}$

Преобразование матрицы в вектор

Матрицу-столбец из трех элементов можно преобразовать в вектор с помощью matrix_to_vector, параметры этой функции - матрица и система координат.

Пример 4.

Преобразовать матрицу $\begin{pmatrix}-1\\2\\5\end{pmatrix}$ в вектор в системе координат N и M из Примера 1.

In [9]:
```python
b = Matrix([-1, 2, 5])
display(matrix_to_vector(b, N), matrix_to_vector(b, M))
```

$-\hat{\mathbf{i}}_{\mathbf{N}} + (2)\hat{\mathbf{j}}_{\mathbf{N}} + (5)\hat{\mathbf{k}}_{\mathbf{N}}$

$-\hat{\mathbf{i}} + (2)\hat{\mathbf{j}} + (5)\hat{\mathbf{k}}$

Преобразование системы координат, преобразование вектора в матрицу

С помощью orient_new_axis получим новую систему координат, которая получается поворотом системы координат на некоторый угол. Для преобразования вектора в матрицу используем метод to_matrix.

Пример 5.

Введем новую систему координат Sys5_new, которая получается поворотом системы координат Sys5 на угол $\pi/3$ относительно оси i против часовой стрелки. Определим вектор b5_Sys5 на основе матрицы b Примера 4 в соответствии с системой координат Sys5, затем получим представление в матричном виде вектора b5_Sys5 в системе координат Sys5_new. Получить координаты вектора b5_Sys5 в системе координат Sys5_new можно с помощью матрицы поворота на угол $\pi/3$ относительно оси i против часовой стрелки. Роль матрицы поворота играет Sys5.rotation_matrix(Sys5_new), умножая матрицу поворота на b5_Sys5_new получаем матрицу b, так что для получения b5_Sys5_new из b можно было бы умножить обратную матрицу к Sys5.rotation_matrix(Sys5_new) на b.

In [10]:
```python
Sys5 = CoordSys3D('S5')
Sys5_new = Sys5.orient_new_axis('S5new', sympy.pi/3, Sys5.i)
b5_Sys5 = matrix_to_vector(b, Sys5)
b5_Sys5_new = b5_Sys5.to_matrix(Sys5_new)
R_matr = Sys5.rotation_matrix(Sys5_new)
display(R_matr, b5_Sys5, b5_Sys5_new, sympy.simplify(R_matr*b5_Sys5_new))
```

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

$-\hat{\mathbf{i}}_{\mathbf{S5}} + (2)\hat{\mathbf{j}}_{\mathbf{S5}} + (5)\hat{\mathbf{k}}_{\mathbf{S5}}$

$$\begin{bmatrix}-1\\1+\frac{5\sqrt{3}}{2}\\\frac{5}{2}-\sqrt{3}\end{bmatrix}$$

$$\begin{bmatrix}-1\\2\\5\end{bmatrix}$$

Поворот системы координат на угол относительно произвольной оси

осуществляется в помощью метода orient_new_axis, параметры - название новой системы координат, угол и вектор, определяющий ось вращения, выраженный в старой системе координат.

Пример 6.

Повернем систему координат Sys5 на $\pi/6$ по часовой стрелке ($-\pi/6$ против часовой стрелки) относительно оси, определяемой вектором $(1, -2, 0)$ (по умолчанию ось проходит через начало координат). Выразим вектор b5_Sys5 в новой системе координат.

In [11]:
```python
Sys6 = Sys5.orient_new_axis('S6', -sympy.pi/6, Sys5.i - 2*Sys5.j)
b5_Sys5.to_matrix(Sys6)
```

Out[11]:

$$\begin{bmatrix}-\sqrt{5}-1\\2-\frac{\sqrt{5}}{2}\\\frac{5\sqrt{3}}{2}\end{bmatrix}$$

Пример 7.

Повернем систему координат M на $\pi/4$ против часовой стрелки относительно оси, определяемой вектором $(1, 0, 1)$ (по умолчанию ось проходит через начало координат). Выразим вектор v Примера 2 в новой системе координат, обозначим его v7. Выведем на экран v7 в векторной и матричной записи.

In [12]:
```python
Sys7 = M.orient_new_axis('7', sympy.pi/4, M.i + M.k)
v7 = express(v, Sys7)
display(v7, v.to_matrix(Sys7))
```

$$\left(-\frac{\sqrt{2}}{2}-\frac{1}{2}\right)\hat{\mathbf{i}}_{\mathbf{7}} + \left(1-\frac{3\sqrt{2}}{2}\right)\hat{\mathbf{j}}_{\mathbf{7}} + \left(\frac{\sqrt{2}}{2}+\frac{5}{2}\right)\hat{\mathbf{k}}_{\mathbf{7}}$$

$$\begin{bmatrix}-\frac{\sqrt{2}}{2}-\frac{1}{2}\\1-\frac{3\sqrt{2}}{2}\\\frac{\sqrt{2}}{2}+\frac{5}{2}\end{bmatrix}$$

components, magnitude, normalize, projection

components - координаты вектора в виде словаря с ключами - ортами, значениями - координатами, это свойство (@property)

magnitude - длина вектора, метод

normalize возвращает нормированный вектор, метод

a.projection(b) возвращает проекцию вектора b на вектор a.

Пример 8.

Выведем на экран координаты вектора v Примера 2, его длину, нормированный v и проекции v на координатные оси и на вектор с координатами (1, -1, 2) в той же системе координат.

In [13]:
```python
display(v, v.components, v.magnitude(), v.normalize(),
        M.i.projection(v), M.j.projection(v), M.k.projection(v),
        (M.i - M.j + 2*M.k).projection(v))
```

$(-3)\hat{\mathbf{j}} + (2)\hat{\mathbf{k}}$

```
{.j: -3, .k: 2}
```

$\sqrt{13}$

$\left(-\frac{3\sqrt{13}}{13}\right)\hat{\mathbf{j}} + \left(\frac{2\sqrt{13}}{13}\right)\hat{\mathbf{k}}$

$\hat{\mathbf{0}}$

$(-3)\hat{\mathbf{j}}$

$(2)\hat{\mathbf{k}}$

$\left(\frac{7}{6}\right)\hat{\mathbf{i}} + \left(-\frac{7}{6}\right)\hat{\mathbf{j}} + \left(\frac{7}{3}\right)\hat{\mathbf{k}}$