Описание алгоритмов

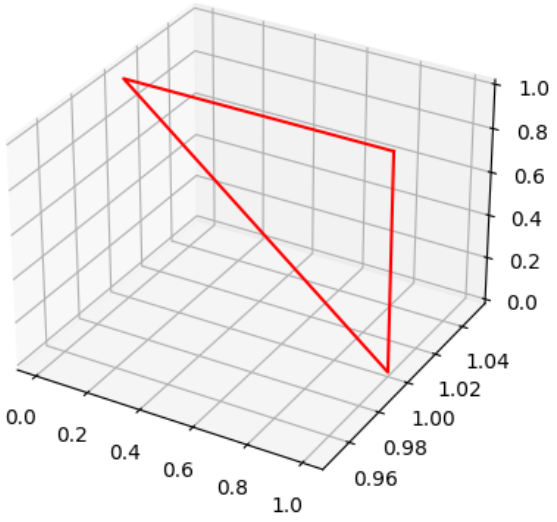
Данное описание в процессе написания

Приложение основано на использовании классических определений из аналитической геометрии и состоит из нескольких классов, позволяющих создавать различные примитивы, по типу плоскостей, линий и изменять их.

STL файл

Структура формата STL ASCII

Данный формат используется для описания моделей с помощью треугольников. У каждого треугольника есть нормаль и три координаты вершин треугольника. Нормаль обычно смотрит во внешнюю часть фигуры. При этом порядок координат говорит нам сторону, в которую смотрит вектор нормали (для этого используется правило правого винта).




```
solid ASCII
facet normal 0.000000e+00 1.000000e+00 0.000000e+00
    outer loop
```

```
      vertex   1.000000e+00 1.000000e+00 1.000000e+00
      vertex   1.000000e+00 1.000000e+00 0.000000e+00
      vertex   0.000000e+00 1.000000e+00 1.000000e+00
    endloop
  endfacet
endsolid
```

solid - название файла.

facet normal - координаты нормали треугольника (она всегда исходит из нуля).

vertex - координаты вершины треугольника.

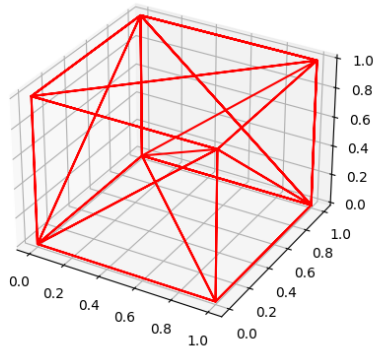
Класс Parser_stl

Функция parse_stl(file) принимает в себя stl файл asc2 формата (текстовый) и возвращает `triangles_array и name` .

```
triangles_array = [[n1, n2, n3],
                                    x1, y1, z1],
                                      x2, y2, z2],
                                      x3, y3, z3]]
```

name - имя, которое стоит после solid

Функция show(`triangles` **)** принимает в себя `triangles_array` , описанный выше и рисует фигуру из треугольников.



Класс линии Line

Этот класс содержит шесть коэффициентов из канонического уравнения линии:

$$\frac{x-a}{p1} = \frac{y-b}{p2} = \frac{z-c}{p3}$$

a, b и c - это координаты точки, через которую проходит прямая.

p1, p2, p3 - координаты вектора, который указывает направление прямой. Начало вектора исходит всегда из нуля. В моей программе используется единичный вектор.

Создание линии по двум точкам

Функция `line_create_from_points(self, point1, point2) -> None:` принимает две точки списком или объектом класса numpy.ndarray:

$$[x_1,\ y_1,\ z_1],\ [x_2,\ y_2,\ z_2]$$

В качестве коэффициентов a, b, c возьмем координаты первой точки (нет разницы, брать первую или вторую точку).

В качестве направляющего вектора возьмем следующий вектор, получаемый из двух точек:

$$p_1 = x_2 - x_1,\ p_2 = y_2 - y_1,\ p_3 = z_2 - z_1$$

$$\vec{P}\{p_1\ , p_2,\ p_3\}$$

Далее необходимо нормировать вектор:

$$|\vec{P}| = \sqrt{p_1^2 + p_2^2 + p_3^2}$$

И разделить каждую координату на норму вектора:

$$p_1 = \frac{x_2 - x_1}{\sqrt{p_1^2 + p_2^2 + p_3^2}} = \frac{x_2 - x_1}{\sqrt{|\vec{P}|}}$$

$$p_2 = \frac{y_2 - y_1}{\sqrt{p_1^2 + p_2^2 + p_3^2}} = \frac{y_2 - y_1}{\sqrt{|\vec{P}|}}$$

$$p_3 = \frac{z_2 - z_1}{\sqrt{p_1^2 + p_2^2 + p_3^2}} = \frac{z_2 - z_1}{\sqrt{|\vec{P}|}}$$

Создание линии по двум плоскостям

Как упоминалось выше, класс линии состоит из координат точки, через которую она проходит и из координат вектора, начало которого исходит из нуля системы координат и конец в координатах , указанных в нижней части канонического уравнения.

Для начала нам необходимо найти вектор, который будет задавать направление прямой, а потом найти любую точку, через которую проходит прямая, и вписать ее коэффициенты в наши параметры прямой.

Для начала нам необходимо убедиться, что две плоскости не параллельны друг другу или не совпадают, поэтому проверяем векторы нормали на коллинеарность векторным произведением, если при перемножении получается нулевой вектор - плоскости параллельны и смысла искать пересечение этих плоскостей - нет. При этом это же векторное произведение мы будем использовать для поиска координат вектора направления прямой.

Пусть имеется две плоскости:

$$A_1x + B_1y + C_1z + D_1 = 0$$

$$A_2x + B_2y + C_2z + D_2 = 0$$

Эти плоскости имеют такие вектора нормалей:

$$\vec{N}_1\{A_1,\ B_1,\ C_1\}$$

$$\vec{N}_2\{A_2,\ B_2,\ C_2\}$$

Их векторное произведение будет равно:

$$\vec{P} = \vec{N}_1 \times \vec{N}_2 = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \end{vmatrix} =$$
$$= (B_1C_2 - C_1B_2)\vec{i} + (C_1A_2 - A_1C_2)\vec{j} + (A_1B_2 - A_2B_1)\vec{k}$$

Если $B_1C_2 - C_1B_2 = 0,\ C_1A_2 - A_1C_2 = 0,\ A_1B_2 - A_2B_1 = 0$, то $\vec{N}_1 || \vec{N}_2$ и плоскости параллельны, либо совпадают.

Далее мы должны нормировать вектор:

$\vec{P}_n = \frac{\vec{P}}{|\vec{P}|} = \{\frac{B_1C_2-C_1B_2}{\sqrt{(B_1C_2-C_1B_2)^2+(C_1A_2-A_1C_2)^2+(A_1B_2-A_2B_1)^2}},$
$\frac{C_1A_2-A_1C_2}{\sqrt{(B_1C_2-C_1B_2)^2+(C_1A_2-A_1C_2)^2+(A_1B_2-A_2B_1)^2}},$
$\frac{A_1B_2-A_2B_1}{\sqrt{(B_1C_2-C_1B_2)^2+(C_1A_2-A_1C_2)^2+(A_1B_2-A_2B_1)^2}}\}$

Если вектор $\vec{P} \neq \vec{0}$ (равносильно $\vec{P} \neq \{0,\ 0,\ 0\}$), то координаты этого вектора идут в коэффициенты $p_1,\ p_2,\ p_3$ канонического уравнения прямой.

Далее, необходимо вычислить координаты точки, которая будет принадлежать пересечению двух плоскостей. Поместим уравнения плоскостей в систему:

$$\begin{cases} A_1x + B_1y + C_1z + D_1 = 0 \\ A_2x + B_2y + C_2z + D_2 = 0 \end{cases}$$

Можно заметить, что у нас два уравнения и три неизвестных. Для того, чтобы выйти из ситуации мы будем приравнивать каждую из координат нулю и искать, получится ли найти другие две.

Пускай z = 0, тогда:

$$\begin{cases} A_1x + B_1y + C_1 \cdot 0 + D_1 = 0 \\ A_2x + B_2y + C_2 \cdot 0 + D_2 = 0 \end{cases}$$

$$\begin{cases} A_1x + B_1y + D_1 = 0 \\ A_2x + B_2y + D_2 = 0 \end{cases}$$

Далее разделим оба уравнения на A_1 и A_2 соответственно:

$$\begin{cases} \frac{A_1x + B_1y + D_1}{A_1} = 0 \\ \frac{A_2x + B_2y + D_2}{A_2} = 0 \end{cases}$$

$$\begin{cases} x + \frac{B_1}{A_1}y + \frac{D_1}{A_1} = 0 \\ x + \frac{B_2}{A_2}y + \frac{D_2}{A_2} = 0 \end{cases}$$

$$\begin{cases} x + \frac{B_1y + D_1}{A_1} = 0 \\ x + \frac{B_2y + D_2}{A_2} = 0 \end{cases}$$

Вычтем из первого уравнения второе, тогда мы получим следующие уравнения:

$$\frac{B_1y + D_1}{A_1} = \frac{B_2y + D_2}{A_2}$$

$$\begin{gathered} B_1A_2y + D_1A_2 = B_2A_1y + D_2A_1 \\ B_1A_2y - B_2A_1y = D_2A_1 - D_1A_2 \\ y(B_1A_2 - B_2A_1) = D_2A_1 - D_1A_2 \\ y = \frac{D_2A_1 - D_1A_2}{B_1A_2 - B_2A_1} \end{gathered}$$

Тогда координата x вычисляется следующим образом:

$$x = -\frac{B_1 y + D_1}{A_1} = -\frac{B_2 y + D_2}{A_2}$$

Класс плоскости Plane

Данный класс содержит четыре коэффициента a, b, c, d из канонического уравнения плоскости:

$$Ax + By + Cz + D = 0$$

Создание плоскости из массива треугольника

Функция `create_plane_from_triangle(triangle, point=1) -> None:`

Данная функция принимает массив 4x3. Строка 1 - координаты вектора нормали (пишутся координаты только второй точки, первая исходит из нуля). Строки 2, 3, 4 - координаты вершин треугольника формата [x, y, z]
На основе четырех точек создается плоскость и коэффициенты a, b, c, d записываются в поля объекта класса Plane.

Математика

На входе мы имеем:

$\vec{N}\{A,\ B,\ C\},\ P_1(x_1,\ y_1,\ z_1),\ P_2(x_2,\ y_2,\ z_2),\ P_3(x_3,\ y_3,\ z_3)$, где N - вектор нормали, а $P_1,\ P_2,\ P_3$ - координаты вершин треугольников.

Координаты с вектора $\vec{N}$ идут в коэффициенты плоскости Plane: $Ax + By + Cz + D = 0$

Для вычисления коэффициента D выразим его:

$$D = -Ax - By - Cz$$

В качестве $x,\ y,\ z$ возьмем любую точку из треугольника, например P_1. Это сработает, так как $P_1 \in Plane$. Тогда получается:

$$D = -Ax_1 - By_1 - Cz_1$$

Функции проекций

Данные функции используются для различных алгоритмов, когда мы знаем значения двух координат из трех в рамках плоскости (если бы это был класс криволинейной поверхности, то в рамках нее) и нам нужно найти координату треьей координаты, зная все коэффициенты канонического уравнения плоскости.

Общее описание на примере нахождения точки z на плоскости по координатам x и y

Данные функции ищут по двум координатам третью, на основе заданной плоскости.

Логика работает следующим образом, напишем снова уравнение плоскости:

$$Ax + By + Cz + D = 0$$

Выражаем одну из координат через остальные члены уравнения, например z в функции `projection_z(self, point_x, point_y):`

$$z = \frac{-A \cdot x - B \cdot y - D}{C}$$

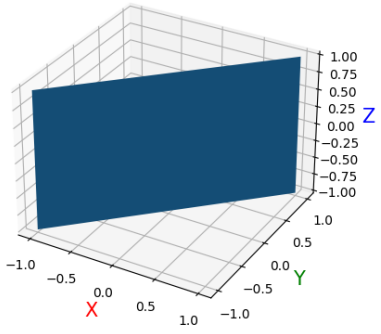
Но здесь можно заметить, что происходит деление на C. Если C = 0, то возникает неопределенность в определении Z. Такое происходит при параллельности плоскости оси Z

Только C = 0.

Примем вектора $\vec{x}, \vec{y}, \vec{z}$, где $\vec{x}\{1,\ 0,\ 0\},\ \vec{y}\{0,\ 1,\ 0\},\ \vec{z}\{0,\ 0,\ 1\}$ являются единичными векторами и рассмотрим различные случаи расположения плоскости относительно осей.

Пускай $Plane||\vec{z}$. В таком случае $\vec{N} \cdot \vec{z} = 0$. Надо заметить, что под знаком “·” подразумевается скалярное умножение векторов.

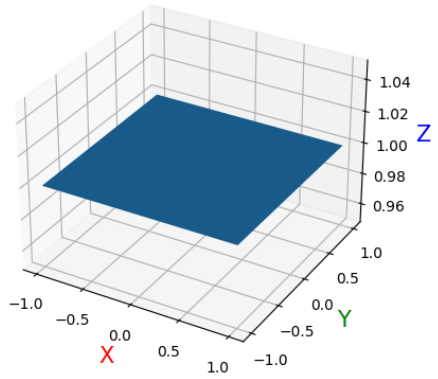
$$\vec{N} \cdot \vec{z} = A \cdot 0 + B \cdot 0 + C \cdot 1 = C$$



Из вышеприведенного уравнения видно, что в случае, если коэффициент C станет равен нулю, то плоскость станет параллельной оси z или будет с ней совпадать, а определение координаты z по точкам x, y станет неопределенным из-за деления на нуль. В этом случае функция возвращает `"Uncertainty z".` Так происходит, потому что в этом случае задаются ограничения на точки x и y. Если в функцию поданы точки, которые не лежат на плоскости, то проекцию по оси z будет невозможно найти, так как эту проекцию мы проводим параллельно оси z и плоскости, мы можем либо никогда не попасть в плоскость, либо проводить ее внутри плоскости. В данном случае мы можем выбирать любой z при получении `"Uncertainty z"` на выходе функции, при это также надо проверить, правильно подавались точки x и y, может быть в этом случае проекцию будет не найти.

B = 0 и A = 0.

В данном случае вектор $\vec{N}$ будет иметь одно фиксированное положение и иметь координаты $\vec{N}\{0,\ 0,\ 1\}$ (В моем приложении в качестве нормалей используются только единичные вектора).



Тогда z можно будет найти следующим образом:

$$z = \frac{-A \cdot x - B \cdot y - D}{C} = \frac{-0 \cdot x - 0 \cdot y - D}{C} = -\frac{D}{C}$$

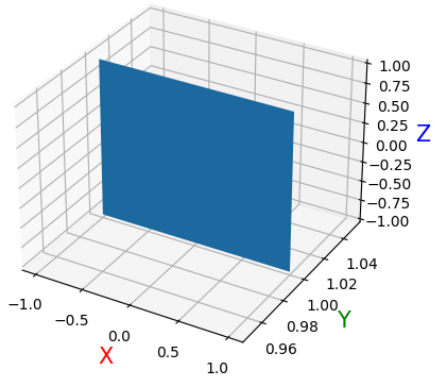
Нахождение точки y по координатам x и z

Соответственно в `projection_y(self, point_x, point_z)` y находится также:

$$y = \frac{-A \cdot x - C \cdot z - D}{B}, \; B \neq 0$$

В случае A = 0 и C = 0:

$$y = -\frac{D}{B}$$



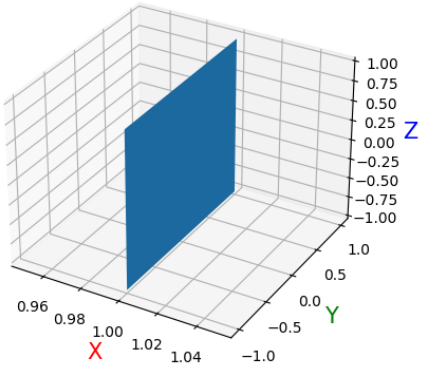
Нахождение точки x по координатам x и z

`projection_x(self, point_y, point_z):`

$$x = \frac{-B \cdot y - C \cdot z - D}{A}$$

В случае Z = 0 и C = 0:

$$x = -\frac{D}{A}$$



Модуль threeDTool

Нахождение точки пересечения двух прямых

Пусть на входе даны следующие уравнения прямых:

$$\frac{x - a_1}{P_1} = \frac{y - b_1}{P_2} = \frac{z - c_1}{P_3} = t_1 \ (1)$$

$$\frac{x - a_2}{R_1} = \frac{y - b_2}{R_2} = \frac{z - c_2}{R_3} = t_2 \ (2)$$

Выразим x, y, z координаты из этих уравнений. С учетом, что при условии пересечения прямых их координаты равны между собой, то:

$$x = t_1 p_1 + a_1 = t_2 r_1 + a_2 \ (3)$$

$$y = t_1 p_2 + b_1 = t_2 r_2 + b_2 \ (4)$$

$$z = t_1 p_3 + c_1 = t_2 r_3 + c_2 \ (5)$$

Выразим из уравнения (3) t_1:

$$t_1 p_1 = t_2 r_1 + a_2 - a_1$$

$$t_1 = \frac{t_2 r_1 + a_2 - a_1}{p_1}$$

Подставим t_1 в уравнение (4):

$$t_2 \frac{r_1 p_2}{p_1} + a_2 \frac{p_2}{p_1} - a_1 \frac{p_2}{p_1} + b_1 = t_2 r_2 + b_2$$

$$t_2 (\frac{r_1 p_2}{p_1} - r_2) = a_1 \frac{p_2}{p_1} - a_2 \frac{p_2}{p_1} + b_2 - b_1$$

$$t_2 \frac{r_1 p_2}{p_1} = a_1 \frac{p_2}{p_1} - a_2 \frac{p_2}{p_1} + b_2 - b_1$$

Умножим уравнение на p_1, чтобы избавиться от знаменателя:

$$t_2(r_1p_2 - r_2p_1) = a_1p_2 - a_2p_2 + b_2p_1 - b_1p_1$$

Таким образом, t_2, полученный путем выделения t_1 из (3) и подставлением его в (4) будет равен:

$$t_2 = \frac{a_1p_2 - a_2p_2 + b_2p_1 - b_1p_1}{r_1p_2 - r_2p_1} \quad (6)$$

Из формулы (6) можно заметить, что в случае, если $r_1p_2 = r_2p_1$, то появляется неопределенность. Возьмем вектора P{1, 2, 0} и R{2, 1, 0}:

$$r_1p_2 - r_2p_1 = 2 \cdot 2 - 1 \cdot 1 = 4 - 1 = 3$$

В данном случае все заработает, но если поменять первые две координаты местами у первого вектора P2{2, 1, 0} и R2{2, 1, 0}:

$$r_1p_2 - r_2p_1 = 2 \cdot 1 - 1 \cdot 1 = 2 - 2 = 0$$

Если присмотреться, то можно заметить, что мы взяли по сути два одинаковых вектора. Но если взять любой линейно-зависимый вектор в качестве второго, например {2, 4, 0}, то получится то же самое:

$$r_1p_2 - r_2p_1 = 2 \cdot 2 - 4 \cdot 1 = 0$$

Эту проблему легко выявить, если поместить два вектора в матрицу и найти ее определитель. Так как два вектора имеют нулевой z, то матрица не имеет третий строки и третьего столбца: 2

$$\begin{vmatrix} p_1 & p_2 & p_3 \\ r_1 & r_2 & r_3 \\ 0 & 0 & 0 \end{vmatrix} = \begin{vmatrix} p_1 & p_2 \\ r_1 & r_2 \end{vmatrix} = p_1r_2 - p_2r_1$$

или наоборот $p_2r_1 - p_1r_2$, если вектора поменять местами, сути это не изменит.

Таким образом можно определить, коллинеарны ли вектора, что будет нам говорить о параллельности прямых, либо о их совпадении, тогда поиск точки пересечения становится бессмысленным.

Далее, если взять векторы в плоскостях xz {1, 0, 1} и {2, 0, 1} или yz {0, 1, 1} и {0, 2, 1} то получим следующее:

$$p_2 r_1 - p_1 r_2 = 0 \cdot 2 - 1 \cdot 0 = 0$$

$$p_2 r_1 - p_1 r_2 = 1 \cdot 0 - 0 \cdot 2 = 0$$

Из этого следует, что выражение (6) не работает для линейно-зависимых векторов, а также при попарно нулевых координатах x или y.

Решение для первой проблемы искать нет смысла, так как прямые в случае линейной зависимости векторов параллельны, либо совпадают. Для второй проблемы решение нужно найти. Для выхода из этой ситуации найдем другие решения для t_2. Подставим t_1 в (5), тогда:

$$t_2 = \frac{a_1 p_3 - a_2 p_3 + c_2 p_1 - c_1 p_1}{r_1 p_3 - r_3 p_1} \quad (7)$$

В данном случае неопределенность получается, если $r_1 p_3 = r_3 p_1$. Это случается, если у векторов попарно зануляются координаты x или z. То есть, если вектора находятся в плоскостях yz и xy. Возьмем вектора P2{0, 1, 1} и R2{0, 1, 40}:

$$r_1 p_3 - r_3 p_1 = 0 \cdot 1 - 40 \cdot 0 = 0$$

Снова получается нулевой случай. Давайте на этот раз выразим t_1 из уравнения (4), получится:

$$t_1 = \frac{t_2 r_2 + b_2 - b_1}{p_2}$$

Тогда найдем t_2:

$$t_2 = \frac{b_1 p_3 - b_2 p_3 + p_2 c_2 - p_2 c_1}{r_2 p_3 - r_3 p_2} \quad (8)$$

Давайте посчитаем, чему будет равен знаменатель для предыдущих тестировочных векторов P2 и R2:

$$r_2 p_3 - r_3 p_2 = 1 \cdot 1 - 40 \cdot 1 = -39$$

Таким образом, мы получили выражения (6), (7), (8) для случаев, когда у векторов попарно координаты x или y или z равны нулю.

Давайте распишем эти t_2 более удобно. В степени буквы напишем координаты, которые занулять можно:

$$t_2^z = \frac{a_1 p_2 - a_2 p_2 + b_2 p_1 - b_1 p_1}{r_1 p_2 - r_2 p_1} \quad (6)$$

$$t_2^y = \frac{a_1 p_3 - a_2 p_3 + c_2 p_1 - c_1 p_1}{r_1 p_3 - r_3 p_1} \quad (7)$$

$$t_2^x = \frac{b_1 p_3 - b_2 p_3 + p_2 c_2 - p_2 c_1}{r_2 p_3 - r_3 p_2} \quad (8)$$

Теперь, когда мы нашли t_2, для нахождения точки пересечения мы можем подставить ее в каноническое уравнение прямой (2) и выразить оттуда координаты:

$$x = t_2 r_1 + a_2$$

$$y = t_2 r_2 + b_2$$

$$z = t_2 r_3 + c_2$$

где $t_2 = \{t_2^x, \text{ если } p_2 = r_2 = 0 \text{ или } p_3 = r_3 = 0;\ t_2^y, \text{ если } p_1 = r_1 = 0 \text{ или } p_3 = r_3 = 0;\ t_2^z, \text{ если } p_1 = r_1 = 0 \text{ или } p_2 = r_2 = 0\}$

Реализация в коде

Функция check_position_lines() будет рассмотрена в следующем пункте

```
def point_from_line_line_intersection(line1, line2, log=False):
    """
    Функция возвращает точку пересечения двух линий. В случае, е
    параллельны или не компланарны, то вернет
    None
    :param line1:
    :param line2:
    :return: ndarray([x, y, z])
    """
    # Проверка на принадлежность одной плоскости
    var = check_position_lines(line1, line2)
    if var == 2:
        if line1.coeffs()[3] == 0 and line1.coeffs()[4] == 0 and
            return None
        elif line2.coeffs()[3] == 0 and line2.coeffs()[4] == 0 a
            return None
        if line2.p1 * line1.p3 != line2.p3 * line1.p1:
            # t_2^x
            t = ((line1.a * line1.p3 - line2.a * line1.p3 +
            line2.c * line1.p1 - line1.c * line1.p1) /
                 (line2.p1 * line1.p3 - line2.p3 * line1.p1))
        elif line2.p2 * line1.p3 != line2.p3 * line1.p2:
            # t_2^y
            t = ((line1.b * line1.p3 - line2.b * line1.p3 +
            line1.p2 * line2.c - line1.p2 * line1.c) /
                 (line2.p2 * line1.p3 - line2.p3 * line1.p2))
        else:
            # t_2^z
            t = ((line1.a * line1.p2 - line2.a * line1.p2 + line
            line1.b * line1.p1) /
                 (line2.p1 * line1.p2 - line2.p2 * line1.p1))
        x = t * line2.p1 + line2.a
```

```
        y = t * line2.p2 + line2.b
        z = t * line2.p3 + line2.c
        return np.array([x, y, z])
    else:
        if log:
            logger.error("Прямые не пересекаются, либо совпадают
```

Функция для определения положения прямых между собой

Допустим на входе у нас есть уравнения двух прямых:

$$\frac{x - a_1}{p_1} = \frac{y - b_1}{p_2} = \frac{z - c_1}{p_3} \ (1)$$

$$\frac{x - a_2}{r_1} = \frac{y - b_2}{r_2} = \frac{z - c_2}{r_3} \ (2)$$

Какие вообще могут быть взаимные положения прямых в 3D пространстве? Прямые могут скрещиваться, быть параллельными, пересекаться, совпадать.

Для начала, необходимо определить, компланарны ли данные две прямые 1 и 2. Для этого создадим вектор из точек прохождения прямых по формуле:

$$\vec{H} = \{a_2 - a_1,\ b_2 - b_1,\ c_2 - c_1\} = \{h_1,\ h_2,\ h_3\}$$

Далее впишем данные вектора в матрицу и найдем определитель этой матрицы:

$$\vec{P} = \vec{N}_1 \times \vec{N}_2 = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \end{vmatrix} =$$

$$= (B_1C_2 - C_1B_2)\vec{i} + (C_1A_2 - A_1C_2)\vec{j} + (A_1B_2 - A_2B_1)\vec{k}$$

$$\begin{vmatrix} p_1 & p_2 & p_3 \\ r_1 & r_2 & r_3 \\ h_1 & h_2 & h_3 \end{vmatrix} = p_1r_2h_3 + p_2r_3h_1 + h_2r_1p_3 - p_3r_2h_1 - p_2r_1h_3 - p_1r_3h_2$$

Если определитель будет равен нулю, то матрица вырожденная и это значит, что вектора между собой линейно-зависимы и прямые лежат в одной плоскости. Далее нужно просто определить, параллельны ли эти прямые. Найдя определитель, мы узнаем, равен ли ранг матрицы размерности матрицы, если равен, то прямые 1 и 2 не лежат в одной плоскости, так как у определителя с размерностью n = 3 есть геометрический смысл - это объем параллелепипеда и если определитель становится равен нулю, то размерность параллелепипеда складывается до n = 2.

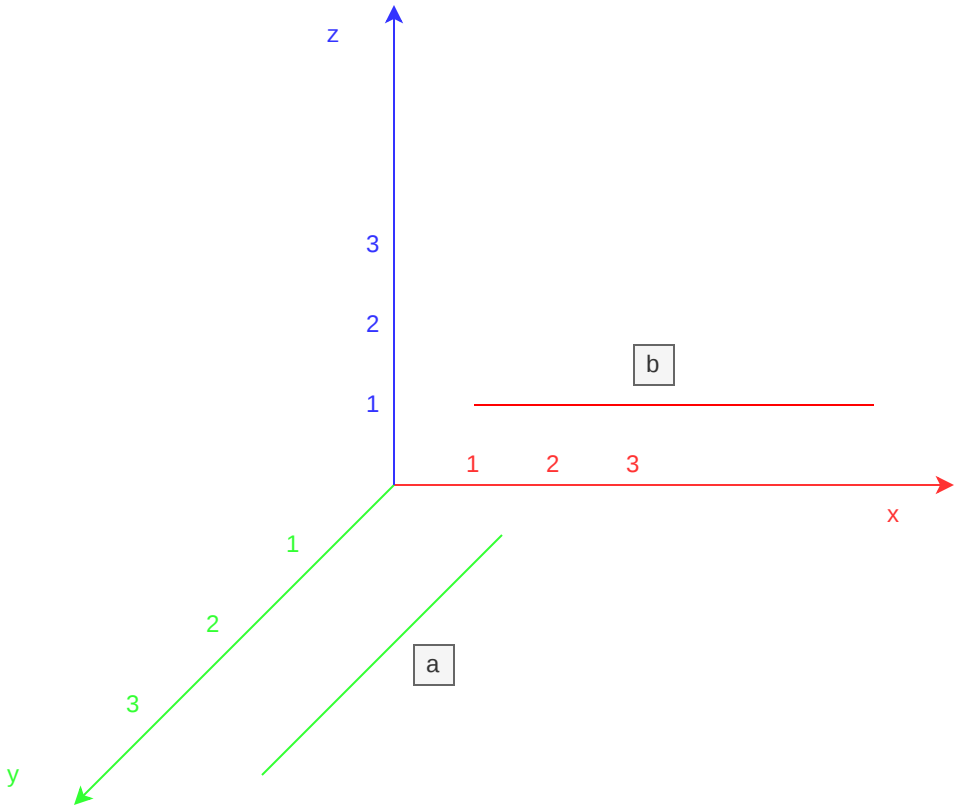
Далее можно с помощью векторного произведения векторов определить их параллельность:

$$|\vec{P} \times \vec{R}| = |\vec{P}| \cdot |\vec{R}| \cdot sin(\vec{P}, \vec{R})$$

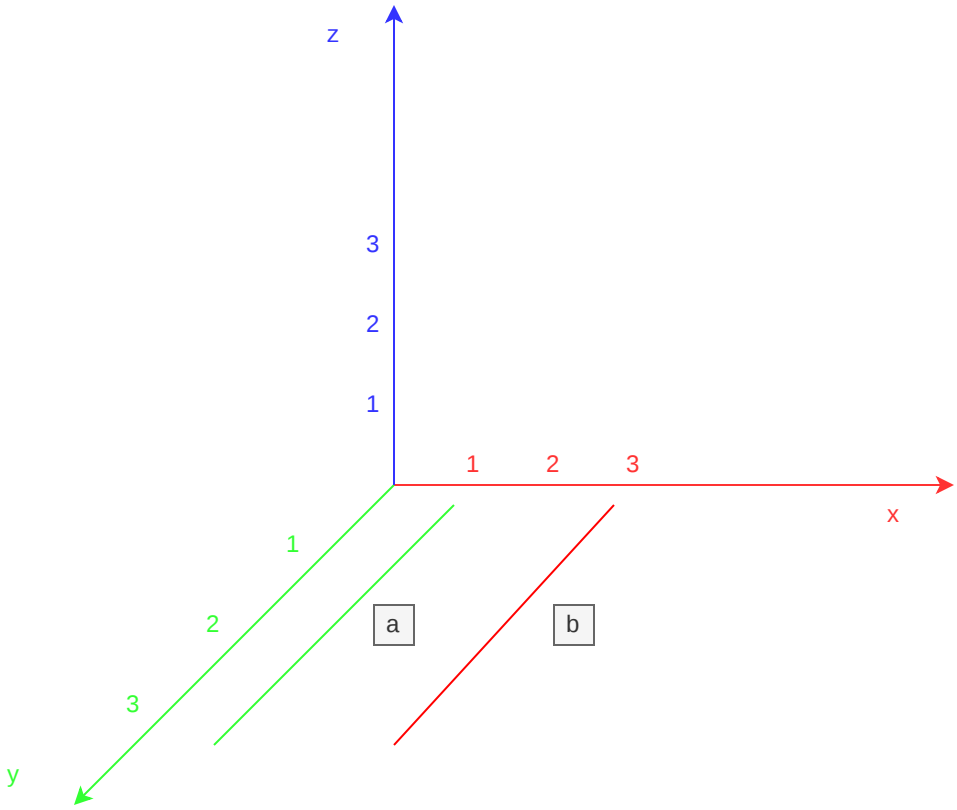
Если модуль вектора будет равен 0, то, то вектора будут коллинеарны, а прямые параллельны, так как синус 0 и 180 градусов равен нулю.

Также можно пойти другим путем и найти ранг матрицы с помощью встроенных пакетов numpy.linalg. Если он равен n-1, то вектора компланарны, если ранг равен n-2, то они еще и параллельны, либо совпадают.

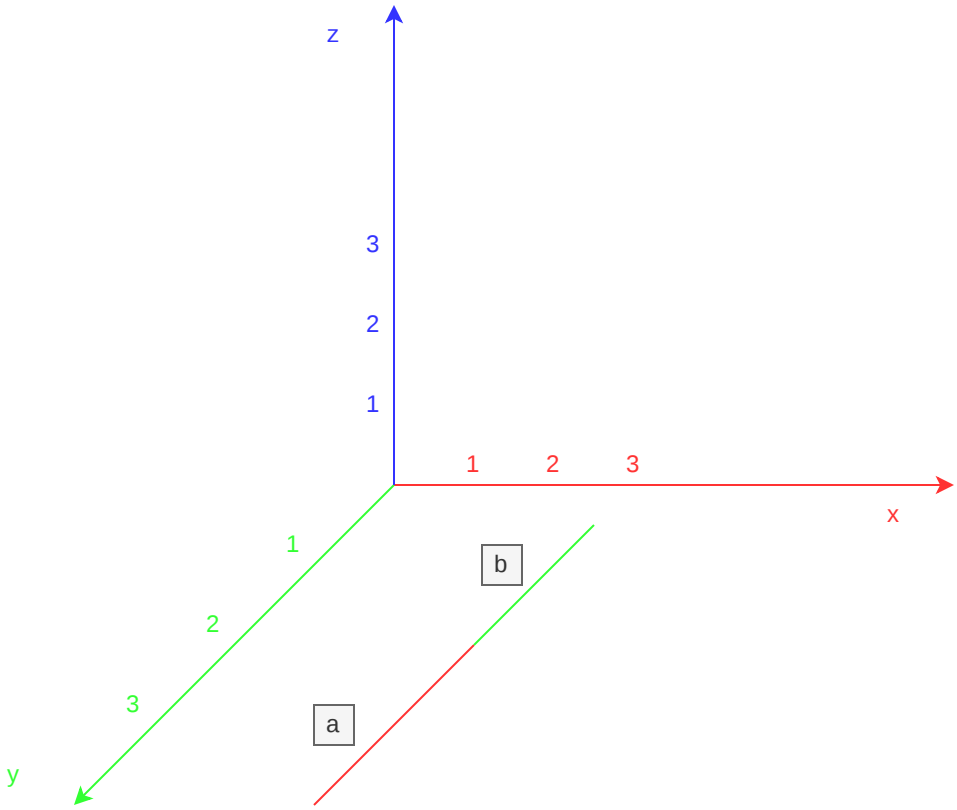
Скрещивающиеся прямые



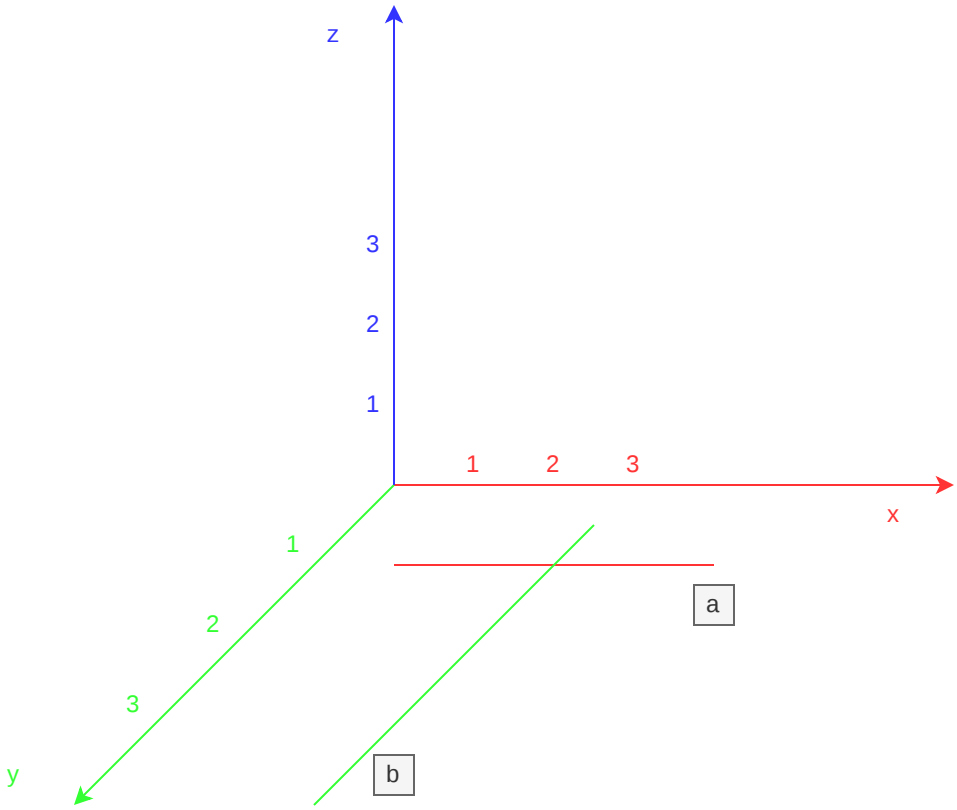
Две параллельные прямые



Совпадающие прямые



Пересекающиеся прямые



Реализация функции для определения позиции двух линий относительно друг друга

```
def check_position_lines(line1: Line, line2: Line) -> int:
    """
    :param line1:
    :param line2:
    :return: 0 - если линии не компланарны,
     1 - если прямые компланарны параллельны,
     2 - если прямые компланарны и не параллельны
    """
    if (np.array_equal(line1.coeffs()[0:3], line2.coeffs()[0:3])
            np.linalg.matrix_rank(np.array([line1.coeffs()[3:6],
            line2.coeffs()[3:6], line2.coeffs()[3:6]])) == 2):
            np.linalg.det(np.array([line1.coeffs()[3:6],
```

```
            line2.coeffs()[3:6], line2.coeffs()[3:6]]))
        return 2
    else:
        line3 = Line()
        line3.line_create_from_points(line1.coeffs()[0:3], line2
        """Проверка на компланарность. Если определитель трех ве
        то они находятся в одной плоскости"""
        arr = np.array([line3.coeffs()[3:6],
                        line1.coeffs()[3:6],
                        line2.coeffs()[3:6]])
        var = np.linalg.det(arr)
        if var == 0:
            cross = np.linalg.norm(np.cross(line1.coeffs()[3:6],
            if cross == 0:
                # прямые параллельны
                return 1
            else:
                # прямые не параллельны
                return 2
        else:
            # прямые не компланарны
            return 0
```