В математике можно встретиться с изменениями объектов в пространстве: прямолинейным движением, вращением, деформацией. Например, нужно описать на плоскости уравнением, относительно какой-то системы отсчёта и известно уравнение относительно другой системы отсчёта. Подобные задачи можно решить, используя методы линейной алгебры. Однако, сложно использовать матрицы, не имея визуального представления об их природе.

Проблемы: визуализация геометрического смысла операций с матрицами; построение графиков функций и их линейное преобразование.

Цель: Автоматизировать сложные вычислительные процессы при построении, преобразовании графиков функции с помощью программы.

План реализации:

1. Изучение и подключение необходимых библиотек.
2. Реализация классов основных объектов на плоскости.
3. Создание плоскости для изображения графиков и векторов.
4. Создание окна для параметров.
5. Добавление векторов и матрицы преобразований .
6. Математическое решение проблемы поворота графиков функций.
7. Автоматизация решения, создание интерфейса для ввода функций.
8. Тестирование.

1.0

Библиотеки

Для написания программы в первую очередь требуется знание библиотек: для написания оконного приложения – PyQt, для моделирования плоскости, на которой можно изображать векторы уравнения графика – PyGame, для упрощения вычислений и применения операций с матрицами и векторами – NumPy.

Все три библиотеки, а также программирование на Python в общем я изучил на курсах дополнительного образования Яндекс.Лицей.

2.0

Реализация классов основных объектов на плоскости

В качестве графического движка я выбрал PyGame, для упрощения работы. Основные объекты на плоскости в данной реализации – точка (вершина) и вектор.

2.1

Точка

Создадим класс:

class Vertex:  
 def \_\_init\_\_(self, coords, unit, center):  
 # coordinates  
 self.coords = coords  
 # coords after applying transformation  
 self.transformed = coords  
 # information to get real coords if vertex  
 self.unit = unit  
 self.center = center

В инициализатор класс принимает координаты, количество пикселей в единичном отрезке и координаты центра плоскости, в *transformed* сохраняются координаты после перемещения.

def real\_cords(self):  
 return [int(self.transformed[0] \* self.unit + self.center[0]),  
 int(-self.transformed[1] \* self.unit + self.center[1])]

Данный метод возвращает “настоящие” координаты (т.е. относительно плоскости PyGame): координата точки \* количество пикселей + координата центра. Минус во второй координате стоит с учётом, что ось Oy в плоскости окна PyGame направлена вниз (Начало координат в PyGame в верхнем левом углу)

3.0

Создание плоскости для изображения графиков и векторов

Для реализации