**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Лабораторная работа № 1**

Тема: Построение изображений 2D-кривых

Студент: Манташев Асадулла Уллубиевич

Группа: 08-305

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2022

1. Постановка задачи

Написать и отладить программу, строящее изображение заданной кривой.

Вариант 3: , , где – полярные координаты, константа, значение которой выбирается пользователем (вводится в окне программы)

1. Описание программы

Программа написана на Python. Для работы с основным окном и динамическим изменением переменных использовалась библиотека Tkinter, для отрисовки графика - Matplotlib. Программа состоит из одного файла. Вся основная логика программы (создание окна, размещение слайдеров и отрисовка графика) описана в классе App.

Основные методы класса App:

* initialize(self) - инициализация переменных и кнопок для работы с графиком
* configure\_plot() - инициализация специальных объектов для работы с графикой, создание дополнительных осей с нужным стилем
* data(a) - генерация точек (x, y) графика для заданного параметра a.
* calculate\_borders(x, y) - возвращает пару чисел (xBorder, yBorder) - границы координатных осей для отображения графика
* on\_change(self, value) - метод, вызывающийся при изменении какого-либо параметра у графика. Получает набор точек (x, y) из метода data(), после чего устанавливает границы системы координат для отрисовки при помощи метода calculate\_borders

1. Набор тестов

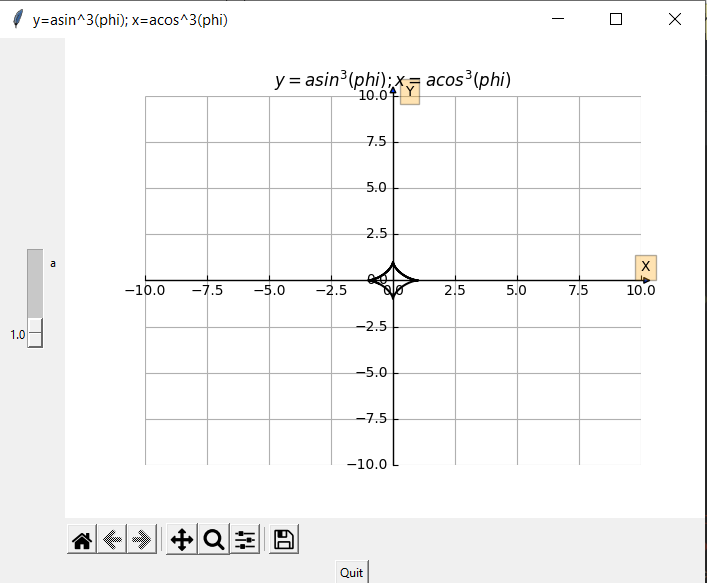
Тест 1: a=1

Тест 2: a=5

Тест 3: a=10

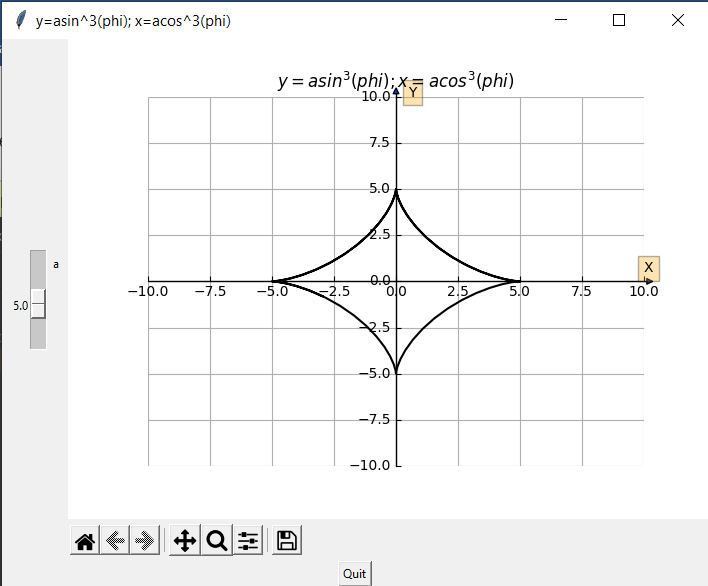
1. Результаты выполнения тестов

Тест 1:



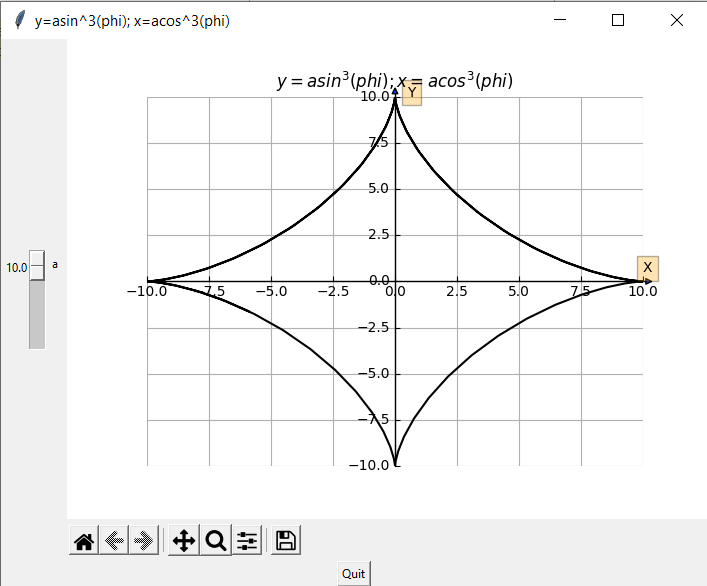
a=1

Тест 2:



a=5

Тест 3:



a=10

1. Листинг программы

'''  
Манташев А. М8О-305Б-20  
Лабораторная работа №1.  
Задание: Написать и отладить программу, строящую изображение заданной замечательной кривой.  
Вариант №3:  
x = a \* cos^3(phi)  
y = a \* sin^3(phi)  
'''

import numpy as np  
import tkinter as tk  
import matplotlib.pyplot as plt  
from matplotlib.backends.\_backend\_tk import NavigationToolbar2Tk  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
from mpl\_toolkits.axisartist import SubplotZero  
import random  
class App(tk.Tk):  
 def \_\_init\_\_(self):  
 tk.Tk.\_\_init\_\_(self)  
 self.initialize()  
 def initialize(self):  
 self.title('Some title name')  
 button = tk.Button(self, text='Quit', command=self.on\_click)  
 button.pack(side=tk.BOTTOM)  
 self.a = tk.DoubleVar()  
 self.a.set(5.0)  
 slider\_a = tk.Scale(self, from\_=10, to=1, resolution=0.1,  
 label='a', variable=self.a,  
 command=self.on\_change)  
 slider\_a.pack(side=tk.LEFT, padx = 0, fill=None, expand=False)  
  
  
 x, y = self.data(self.a.get())  
 self.fig, self.ax = self.configure\_plot()  
 xBorder, yBorder = self.calculate\_borders(x, y)  
 xlim = (-xBorder, xBorder)  
 ylim = (-yBorder, yBorder)  
 self.change\_limits(xlim, ylim)  
 self.draw\_text(xBorder, yBorder)  
 self.line1, = self.ax.plot(x, y, color=random.choice(['red', 'green', 'blue', 'black', 'orange']))  
 self.canvas = FigureCanvasTkAgg(self.fig, master=self)  
 self.canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)  
 toolbar = NavigationToolbar2Tk(self.canvas, self)  
 toolbar.update()  
 self.canvas.draw()  
 def on\_click(self):  
 self.quit()  
 def on\_change(self, value):  
 x, y = self.data(self.a.get())  
 xBorder, yBorder = self.calculate\_borders(x, y)  
 xlim = (-xBorder, xBorder)  
 ylim = (-yBorder, yBorder)  
 self.change\_limits(xlim, ylim)  
 self.line1.set\_data(x, y)  
 for i in range(len(self.ax.texts)):  
 self.ax.texts[i].set\_visible(False)  
 self.draw\_text(xBorder, yBorder)  
 self.canvas.draw()  
 @staticmethod  
 def change\_limits(xlim, ylim):  
 plt.xlim(xlim)  
 plt.ylim(ylim)  
 @staticmethod  
 def calculate\_borders(x, y):  
 xmin = -10  
 xmax = 10  
 ymin = 0  
 ymax = 10  
 xBorder = max(abs(min(min(x), xmin)), abs(max(max(x), xmax)))  
 yBorder = max(abs(min(min(y), ymin)), abs(max(max(y), ymax)))  
 return xBorder, yBorder  
 def draw\_text(self, xBorder, yBorder):  
 self.ax.text(xBorder, yBorder / 20, 'X', size=10, bbox=dict(facecolor='orange', alpha=0.3))  
 self.ax.text(xBorder / 20, yBorder, 'Y', size=10, bbox=dict(facecolor='orange', alpha=0.3))  
 @staticmethod  
 def configure\_plot():  
 fig = plt.figure(1)  
 plt.style.use('seaborn-bright')  
 ax = SubplotZero(fig, 111)  
 fig.add\_subplot(ax)  
 for direction in ['xzero', 'yzero']:  
 ax.axis[direction].set\_axisline\_style('-|>')  
 ax.axis[direction].set\_visible(True)  
 for direction in ['left', 'right', 'bottom', 'top']:  
 ax.axis[direction].set\_visible(False)  
 plt.grid()  
 plt.title('$y=asin^3(phi); x=acos^3(phi)$')  
 return fig, ax  
 @staticmethod  
 def data(a):  
 Phi = np.linspace(0, 10, 100)  
 x = a \* np.power(np.cos(Phi), 3)  
 y = a \* np.power(np.sin(Phi), 3)  
 return x, y  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = App()  
 app.mainloop()

ЛИТЕРАТУРА

1. Статься ‘Введение в Tkinter’ - <https://habr.com/ru/post/133337/>
2. Документация Matplotlib - <https://matplotlib.org/>