МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий математики и механики**

Направление подготовки: «Фундаментальная информатика и информационные технологии»

**ОТЧЕТ**

По курсу «Вычислительные методы»

На тему:

**Метод Рунге-Кутта четвертого порядка.**

**Выполнил**:студент группы 381806-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Напылов Е.И.

Подпись

**Научный руководитель**:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Эгамов А.И.

Подпись

Нижний Новгород

2021

# Содержание

[Содержание 2](#_Toc65252870)

[Введение 3](#_Toc65252871)

[Постановка задачи 4](#_Toc65252872)

[Теоретическая часть 5](#_Toc65252873)

[Реализация 6](#_Toc65252874)

[Руководство к программе 8](#_Toc65252875)

[Тестирование 10](#_Toc65252876)

[Заключение 12](#_Toc65252877)

[Список литературы 13](#_Toc65252878)

[Приложение 14](#_Toc65252879)

# Введение

Для решения многих практических и теоретических задач требуется решить задачу Коши, т.е. найти частное решение по начальным условиям. Чаще всего динамические системы получаются достаточно сложными, из-за чего их проблематично решить точными теоретическими способами. Следовательно, возникает потребность в численных методах решения задачи Коши. Одним из способов решения является метод Рунге Кутта. Данный метод является более точным по сравнению с известным методом Эйлера. Самым распространенным является метод четвертого порядка. Однако, если требуется высокая точность, то используют методы пятого, шестого и седьмого порядков. Построение схем более высоких порядков проблематично из-за большой вычислительной нагрузки.

# Постановка задачи

Требуется разработать приложение для исследования систем дифференциальных уравнений с использованием задачи Коши. Программа должна численно решать задачу Коши с помощью метода Рунге-Кутта четвертого порядка. Программа должна обладать следующими функциями:

1. Ввод системы ДУ, состоящей из двух или трех уравнений.
2. Ввод начальных условии (задача Коши)
3. Построение фазовой траектории на заданном отрезке
4. Построение фазового портрета системы на заданном отрезке.

# Теоретическая часть

Методы Рунге-Кутты - большой класс численных методов решения задачи Коши для обыкновенных дифференциальных уравнений и их систем. Идея методов Эйлера и Рунге-Кутты состоит в том, чтобы заменить фрагмент графика ломаной линией.

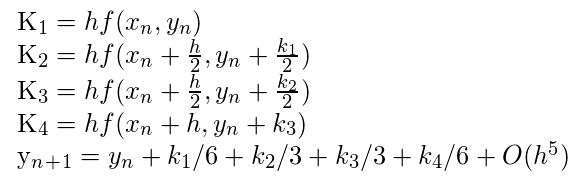
Наиболее распространённым является метод Рунге-Кутты четвёртого порядка.

Рассмотрим обыкновенное дифференциальное уравнение и задачу Коши.



Задача состоит в том, чтобы найти значение неизвестной функции y в заданной точке.

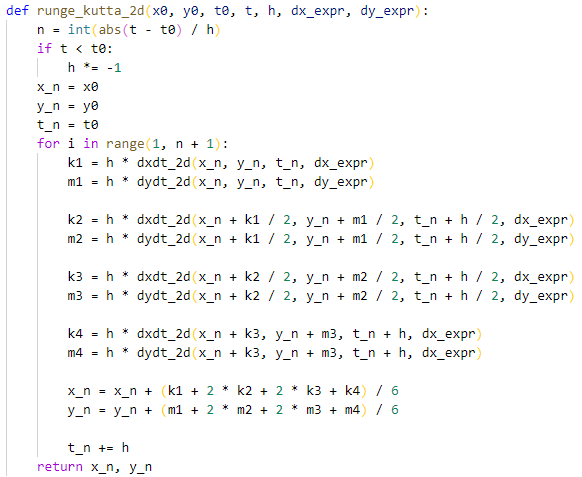
Расчет происходит по рекуррентной формуле.

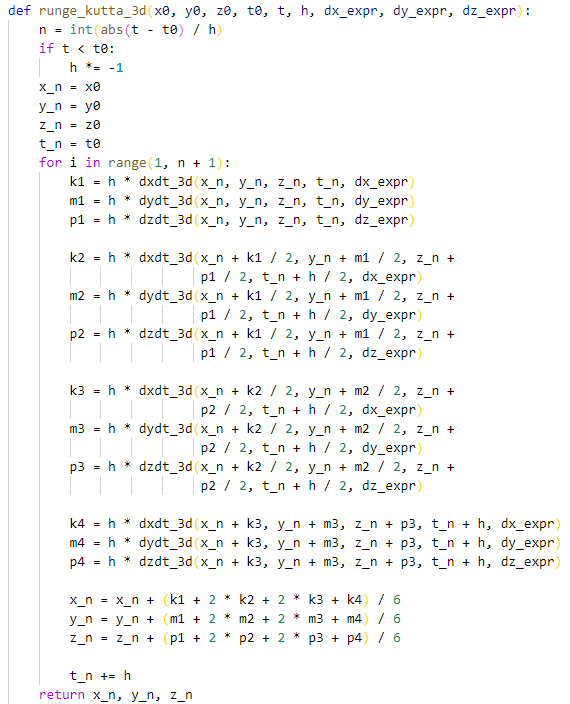


Здесь h – размер шага метода. Итерации производятся (x-x0)/h раз.

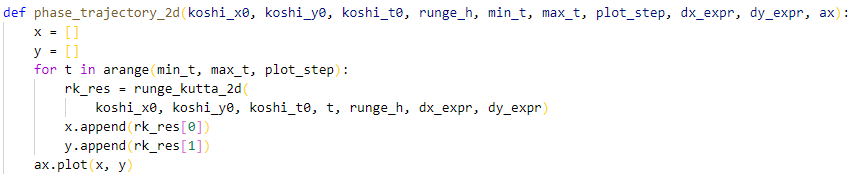
# Реализация

Метод Рунге-Кутта четвертого порядка был реализован в двух функциях для двух и трех уравнений соответственно. Функции принимают начальные условия, новую точку, шаг метода и скомпилированные из пользовательского ввода выражения для производных.

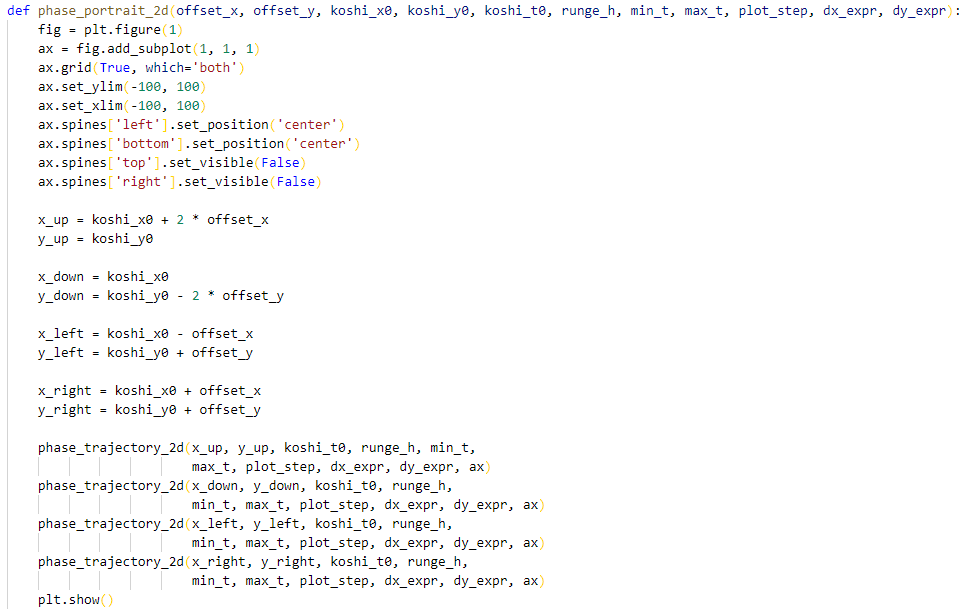




Фазовый портрет состоит из фазовых траекторий. Для построения траекторий используется библиотека визуализации данных matplotlib. Фазовая траектория строится на заданном пользователем отрезке.



Для построения фазового портрета необходимо построить четыре фазовые траектории вокруг заданной точки.

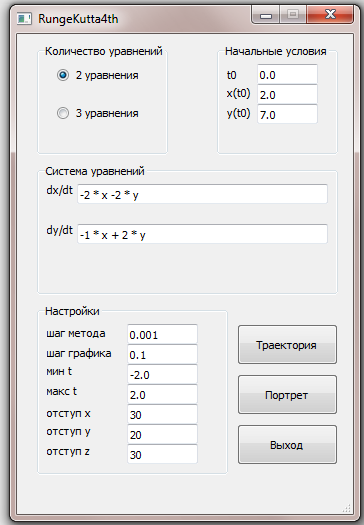


Для трех уравнений аналогично.

Для разработки пользовательского интерфейса была использована библиотека PyQt5. Эта библиотека вместе с Qt Designer позволяет быстро и удобно разрабатывать GUI приложения. Язык Python является интерпретируемым и позволяет преобразовать введенную пользователем строку в исполняемую строку кода (функции compile(…) и eval(…)). Эта особенность была использована для ввода системы уравнений.

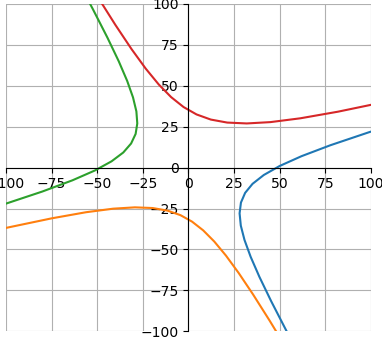
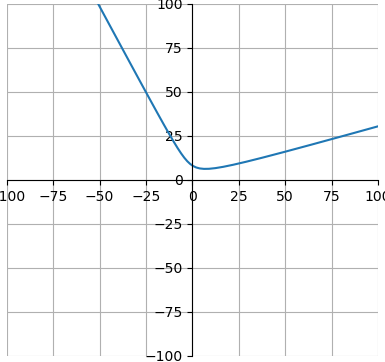
# Руководство к программе

Окно программы позволяет настроить большое количество параметров.

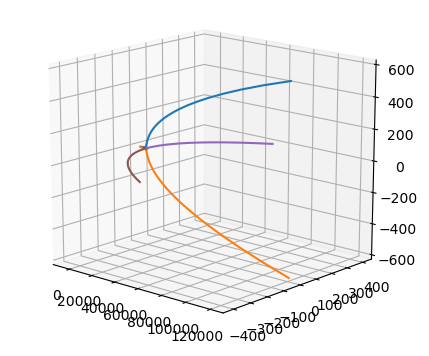
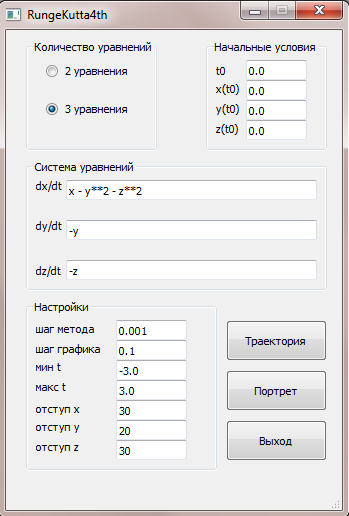


Отступы в настройках необходимы для корректного отображения фазового портрета.

Для построения необходимо ввести данные. Функции вводятся в соответствии с синтаксисом языка Python! и нажать кнопки «Траектория» или «Портет».



Чтобы ввести систему из трех уравнений необходимо переключить Check-Box.

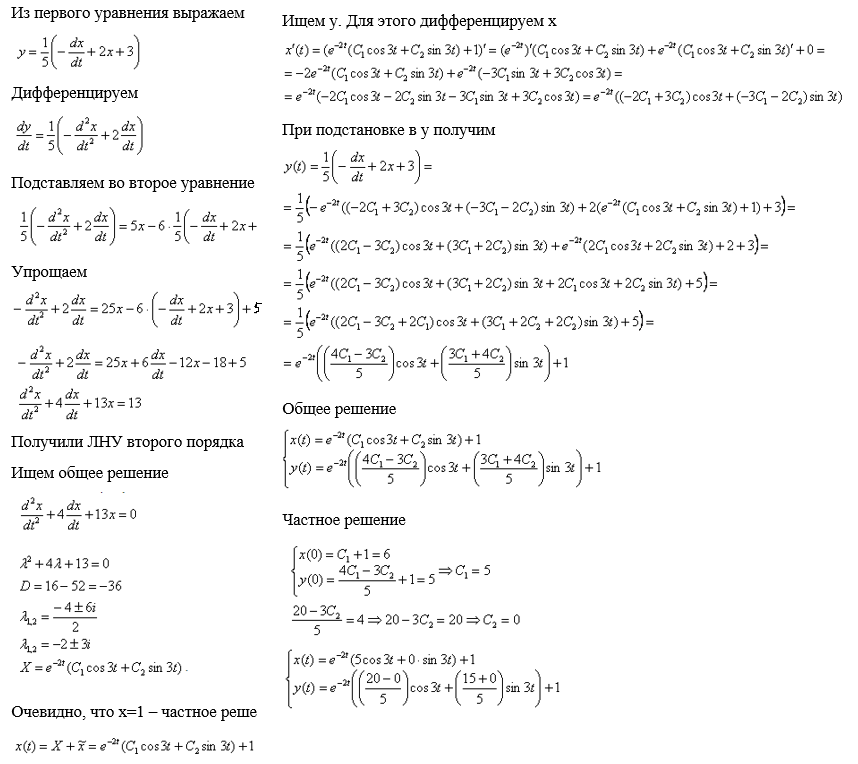


# Тестирование

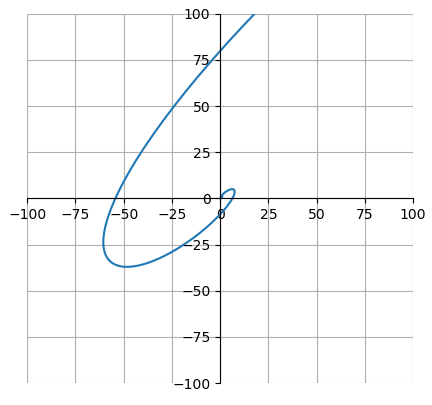
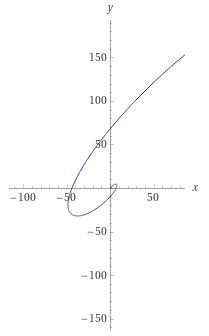
Тестирование программы проводилось со следующей системой:



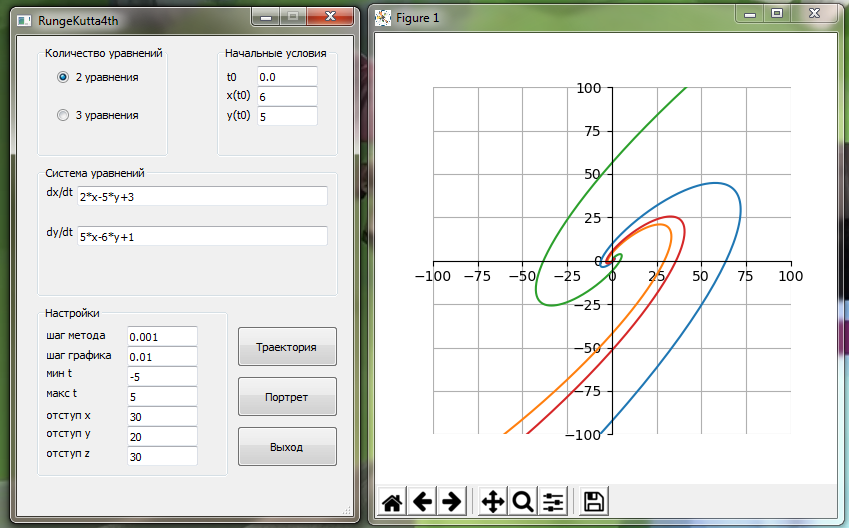
Теоретическое решение:



Сравним полученное частное решение с программой. Для построения графика теоретического решения использован сервис Wolfram Alpha.



Фазовый портрет системы:



# Заключение

В ходе работы я изучил метод Рунге-Кутта в теории. После теории я реализовал данный метод на языке программирования Python. После консольного тестирования был разработан графический пользовательский интерфейс, а также возможность построения фазовой траектории и фазового портрета. Для проверки корректности программы я сравнил результаты с аналитическим решением. Ответ совпал, следовательно, программа работает правильно.

# Список **литературы**

1. Волков Е.А., «Численные методы», 2008
2. Жидков Е.Н., «Вычислительная математика», 2013
3. Лутц М., «Изучаем Python», том 1, 2019

# Приложение

# -\*- coding: utf-8 -\*-

# Form implementation generated from reading ui file 'ui.ui'

#

# Created by: PyQt5 UI code generator 5.15.2

#

# WARNING: Any manual changes made to this file will be lost when pyuic5 is

# run again.  Do not edit this file unless you know what you are doing.

from PyQt5 import QtCore, QtGui, QtWidgets

from math import \*

import matplotlib.pyplot as plt

from numpy import arange

from mpl\_toolkits.mplot3d import Axes3D

def runge\_kutta\_2d(x0, y0, t0, t, h, dx\_expr, dy\_expr):

    n = int(abs(t - t0) / h)

    if t < t0:

        h \*= -1  # !!!

    x\_n = x0

    y\_n = y0

    t\_n = t0

    for i in range(1, n + 1):

        k1 = h \* dxdt\_2d(x\_n, y\_n, t\_n, dx\_expr)

        m1 = h \* dydt\_2d(x\_n, y\_n, t\_n, dy\_expr)

        k2 = h \* dxdt\_2d(x\_n + k1 / 2, y\_n + m1 / 2, t\_n + h / 2, dx\_expr)

        m2 = h \* dydt\_2d(x\_n + k1 / 2, y\_n + m1 / 2, t\_n + h / 2, dy\_expr)

        k3 = h \* dxdt\_2d(x\_n + k2 / 2, y\_n + m2 / 2, t\_n + h / 2, dx\_expr)

        m3 = h \* dydt\_2d(x\_n + k2 / 2, y\_n + m2 / 2, t\_n + h / 2, dy\_expr)

        k4 = h \* dxdt\_2d(x\_n + k3, y\_n + m3, t\_n + h, dx\_expr)

        m4 = h \* dydt\_2d(x\_n + k3, y\_n + m3, t\_n + h, dx\_expr)

        x\_n = x\_n + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6

        y\_n = y\_n + (m1 + 2 \* m2 + 2 \* m3 + m4) / 6

        t\_n += h

    return x\_n, y\_n

def phase\_trajectory\_2d(koshi\_x0, koshi\_y0, koshi\_t0, runge\_h, min\_t, max\_t, plot\_step, dx\_expr, dy\_expr, ax):

    x = []

    y = []

    for t in arange(min\_t, max\_t, plot\_step):

        rk\_res = runge\_kutta\_2d(

            koshi\_x0, koshi\_y0, koshi\_t0, t, runge\_h, dx\_expr, dy\_expr)

        x.append(rk\_res[0])

        y.append(rk\_res[1])

    ax.plot(x, y)

def phase\_portrait\_2d(offset\_x, offset\_y, koshi\_x0, koshi\_y0, koshi\_t0, runge\_h, min\_t, max\_t, plot\_step, dx\_expr, dy\_expr):

    fig = plt.figure(1)

    ax = fig.add\_subplot(1, 1, 1)

    ax.grid(True, which='both')

    ax.set\_ylim(-100, 100)

    ax.set\_xlim(-100, 100)

    ax.spines['left'].set\_position('center')

    ax.spines['bottom'].set\_position('center')

    ax.spines['top'].set\_visible(False)

    ax.spines['right'].set\_visible(False)

    x\_up = koshi\_x0 + 2 \* offset\_x

    y\_up = koshi\_y0

    x\_down = koshi\_x0

    y\_down = koshi\_y0 - 2 \* offset\_y

    x\_left = koshi\_x0 - offset\_x

    y\_left = koshi\_y0 + offset\_y

    x\_right = koshi\_x0 + offset\_x

    y\_right = koshi\_y0 + offset\_y

    phase\_trajectory\_2d(x\_up, y\_up, koshi\_t0, runge\_h, min\_t,

                        max\_t, plot\_step, dx\_expr, dy\_expr, ax)

    phase\_trajectory\_2d(x\_down, y\_down, koshi\_t0, runge\_h,

                        min\_t, max\_t, plot\_step, dx\_expr, dy\_expr, ax)

    phase\_trajectory\_2d(x\_left, y\_left, koshi\_t0, runge\_h,

                        min\_t, max\_t, plot\_step, dx\_expr, dy\_expr, ax)

    phase\_trajectory\_2d(x\_right, y\_right, koshi\_t0, runge\_h,

                        min\_t, max\_t, plot\_step, dx\_expr, dy\_expr, ax)

    plt.show()

def runge\_kutta\_3d(x0, y0, z0, t0, t, h, dx\_expr, dy\_expr, dz\_expr):

    n = int(abs(t - t0) / h)

    if t < t0:

        h \*= -1  # !!!

    x\_n = x0

    y\_n = y0

    z\_n = z0

    t\_n = t0

    for i in range(1, n + 1):

        k1 = h \* dxdt\_3d(x\_n, y\_n, z\_n, t\_n, dx\_expr)

        m1 = h \* dydt\_3d(x\_n, y\_n, z\_n, t\_n, dy\_expr)

        p1 = h \* dzdt\_3d(x\_n, y\_n, z\_n, t\_n, dz\_expr)

        k2 = h \* dxdt\_3d(x\_n + k1 / 2, y\_n + m1 / 2, z\_n +

                         p1 / 2, t\_n + h / 2, dx\_expr)

        m2 = h \* dydt\_3d(x\_n + k1 / 2, y\_n + m1 / 2, z\_n +

                         p1 / 2, t\_n + h / 2, dy\_expr)

        p2 = h \* dzdt\_3d(x\_n + k1 / 2, y\_n + m1 / 2, z\_n +

                         p1 / 2, t\_n + h / 2, dz\_expr)

        k3 = h \* dxdt\_3d(x\_n + k2 / 2, y\_n + m2 / 2, z\_n +

                         p2 / 2, t\_n + h / 2, dx\_expr)

        m3 = h \* dydt\_3d(x\_n + k2 / 2, y\_n + m2 / 2, z\_n +

                         p2 / 2, t\_n + h / 2, dy\_expr)

        p3 = h \* dzdt\_3d(x\_n + k2 / 2, y\_n + m2 / 2, z\_n +

                         p2 / 2, t\_n + h / 2, dz\_expr)

        k4 = h \* dxdt\_3d(x\_n + k3, y\_n + m3, z\_n + p3, t\_n + h, dx\_expr)

        m4 = h \* dydt\_3d(x\_n + k3, y\_n + m3, z\_n + p3, t\_n + h, dy\_expr)

        p4 = h \* dzdt\_3d(x\_n + k3, y\_n + m3, z\_n + p3, t\_n + h, dz\_expr)

        x\_n = x\_n + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6

        y\_n = y\_n + (m1 + 2 \* m2 + 2 \* m3 + m4) / 6

        z\_n = z\_n + (p1 + 2 \* p2 + 2 \* p3 + p4) / 6

        t\_n += h

    return x\_n, y\_n, z\_n

def phase\_trajectory\_3d(koshi\_x0, koshi\_y0, koshi\_z0, koshi\_t0, runge\_h, min\_t, max\_t, plot\_step, dx\_expr, dy\_expr, dz\_expr, ax):

    x = []

    y = []

    z = []

    for t in arange(min\_t, max\_t, plot\_step):

        rk\_res = runge\_kutta\_3d(

            koshi\_x0, koshi\_y0, koshi\_z0, koshi\_t0, t, runge\_h, dx\_expr, dy\_expr, dz\_expr)

        x.append(rk\_res[0])

        y.append(rk\_res[1])

        z.append(rk\_res[2])

    ax.plot(x, y, z)

def phase\_portrait\_3d(offset\_x, offset\_y, offset\_z, koshi\_x0, koshi\_y0, koshi\_z0, koshi\_t0, runge\_h, min\_t, max\_t, plot\_step, dx\_expr, dy\_expr, dz\_expr):

    fig = plt.figure()

    ax = fig.add\_subplot(111, projection='3d')

    x1 = koshi\_x0

    y1 = koshi\_y0

    z1 = koshi\_z0 + offset\_z

    x2 = koshi\_x0

    y2 = koshi\_y0

    z2 = koshi\_z0 - offset\_z

    x3 = koshi\_x0 + offset\_x

    y3 = koshi\_y0

    z3 = koshi\_z0

    x4 = koshi\_x0 - offset\_x

    y4 = koshi\_y0

    z4 = koshi\_z0

    x5 = koshi\_x0

    y5 = koshi\_y0 + offset\_y

    z5 = koshi\_z0

    x6 = koshi\_x0

    y6 = koshi\_y0 - offset\_y

    z6 = koshi\_z0

    phase\_trajectory\_3d(x1, y1, z1, koshi\_t0, runge\_h, min\_t,

                        max\_t, plot\_step, dx\_expr, dy\_expr, dz\_expr, ax)

    phase\_trajectory\_3d(x2, y2, z2, koshi\_t0, runge\_h, min\_t,

                        max\_t, plot\_step, dx\_expr, dy\_expr, dz\_expr, ax)

    phase\_trajectory\_3d(x3, y3, z3, koshi\_t0, runge\_h, min\_t,

                        max\_t, plot\_step, dx\_expr, dy\_expr, dz\_expr, ax)

    phase\_trajectory\_3d(x4, y4, z4, koshi\_t0, runge\_h, min\_t,

                        max\_t, plot\_step, dx\_expr, dy\_expr, dz\_expr, ax)

    phase\_trajectory\_3d(x5, y5, z5, koshi\_t0, runge\_h, min\_t,

                        max\_t, plot\_step, dx\_expr, dy\_expr, dz\_expr, ax)

    phase\_trajectory\_3d(x6, y6, z6, koshi\_t0, runge\_h, min\_t,

                        max\_t, plot\_step, dx\_expr, dy\_expr, dz\_expr, ax)

    plt.show()

def dxdt\_2d(x, y, t, dx\_expr):

    return eval(dx\_expr)

def dydt\_2d(x, y, t, dy\_expr):

    return eval(dy\_expr)

def dxdt\_3d(x, y, z, t, dx\_expr):

    return eval(dx\_expr)

def dydt\_3d(x, y, z, t, dy\_expr):

    return eval(dy\_expr)

def dzdt\_3d(x, y, z, t, dz\_expr):

    return eval(dz\_expr)

class Ui\_MainWindow(object):

    def setupUi(self, MainWindow):

        MainWindow.setObjectName("MainWindow")

        MainWindow.resize(336, 480)

        MainWindow.setMinimumSize(QtCore.QSize(336, 480))

        MainWindow.setMaximumSize(QtCore.QSize(336, 480))

        icon = QtGui.QIcon()

        icon.addPixmap(QtGui.QPixmap("favicon.ico"),

                       QtGui.QIcon.Normal, QtGui.QIcon.Off)

        MainWindow.setWindowIcon(icon)

        self.centralwidget = QtWidgets.QWidget(MainWindow)

        self.centralwidget.setObjectName("centralwidget")

        self.btn\_start = QtWidgets.QPushButton(self.centralwidget)

        self.btn\_start.setGeometry(QtCore.QRect(220, 340, 101, 41))

        self.btn\_start.setObjectName("btn\_start")

        self.groupBox\_eq\_count = QtWidgets.QGroupBox(self.centralwidget)

        self.groupBox\_eq\_count.setGeometry(QtCore.QRect(20, 10, 131, 111))

        self.groupBox\_eq\_count.setObjectName("groupBox\_eq\_count")

        self.radioButton\_two\_eq = QtWidgets.QRadioButton(

            self.groupBox\_eq\_count)

        self.radioButton\_two\_eq.setGeometry(QtCore.QRect(20, 20, 81, 21))

        self.radioButton\_two\_eq.setObjectName("radioButton\_two\_eq")

        self.radioButton\_three\_eq = QtWidgets.QRadioButton(

            self.groupBox\_eq\_count)

        self.radioButton\_three\_eq.setGeometry(QtCore.QRect(20, 60, 82, 17))

        self.radioButton\_three\_eq.setObjectName("radioButton\_three\_eq")

        self.groupBox = QtWidgets.QGroupBox(self.centralwidget)

        self.groupBox.setGeometry(QtCore.QRect(20, 130, 301, 131))

        self.groupBox.setObjectName("groupBox")

        self.inp\_dxdt = QtWidgets.QLineEdit(self.groupBox)

        self.inp\_dxdt.setGeometry(QtCore.QRect(40, 20, 251, 20))

        self.inp\_dxdt.setObjectName("inp\_dxdt")

        self.label = QtWidgets.QLabel(self.groupBox)

        self.label.setGeometry(QtCore.QRect(10, 10, 31, 31))

        self.label.setObjectName("label")

        self.label\_2 = QtWidgets.QLabel(self.groupBox)

        self.label\_2.setGeometry(QtCore.QRect(10, 50, 31, 31))

        self.label\_2.setObjectName("label\_2")

        self.inp\_dydt = QtWidgets.QLineEdit(self.groupBox)

        self.inp\_dydt.setGeometry(QtCore.QRect(40, 60, 251, 20))

        self.inp\_dydt.setObjectName("inp\_dydt")

        self.label\_3 = QtWidgets.QLabel(self.groupBox)

        self.label\_3.setGeometry(QtCore.QRect(10, 100, 31, 21))

        self.label\_3.setObjectName("label\_3")

        self.inp\_dzdt = QtWidgets.QLineEdit(self.groupBox)

        self.inp\_dzdt.setGeometry(QtCore.QRect(40, 100, 251, 20))

        self.inp\_dzdt.setObjectName("inp\_dzdt")

        self.groupBox\_start\_cond = QtWidgets.QGroupBox(self.centralwidget)

        self.groupBox\_start\_cond.setGeometry(QtCore.QRect(200, 10, 121, 111))

        self.groupBox\_start\_cond.setObjectName("groupBox\_start\_cond")

        self.label\_4 = QtWidgets.QLabel(self.groupBox\_start\_cond)

        self.label\_4.setGeometry(QtCore.QRect(10, 20, 21, 21))

        self.label\_4.setObjectName("label\_4")

        self.inp\_t0 = QtWidgets.QLineEdit(self.groupBox\_start\_cond)

        self.inp\_t0.setGeometry(QtCore.QRect(40, 20, 61, 20))

        self.inp\_t0.setObjectName("inp\_t0")

        self.label\_5 = QtWidgets.QLabel(self.groupBox\_start\_cond)

        self.label\_5.setGeometry(QtCore.QRect(10, 40, 31, 16))

        self.label\_5.setObjectName("label\_5")

        self.inp\_x0 = QtWidgets.QLineEdit(self.groupBox\_start\_cond)

        self.inp\_x0.setGeometry(QtCore.QRect(40, 40, 61, 20))

        self.inp\_x0.setObjectName("inp\_x0")

        self.label\_6 = QtWidgets.QLabel(self.groupBox\_start\_cond)

        self.label\_6.setGeometry(QtCore.QRect(10, 60, 31, 16))

        self.label\_6.setObjectName("label\_6")

        self.inp\_y0 = QtWidgets.QLineEdit(self.groupBox\_start\_cond)

        self.inp\_y0.setGeometry(QtCore.QRect(40, 60, 61, 20))

        self.inp\_y0.setObjectName("inp\_y0")

        self.inp\_z0 = QtWidgets.QLineEdit(self.groupBox\_start\_cond)

        self.inp\_z0.setGeometry(QtCore.QRect(40, 80, 61, 20))

        self.inp\_z0.setObjectName("inp\_z0")

        self.label\_7 = QtWidgets.QLabel(self.groupBox\_start\_cond)

        self.label\_7.setGeometry(QtCore.QRect(10, 80, 31, 16))

        self.label\_7.setObjectName("label\_7")

        self.groupBox\_2 = QtWidgets.QGroupBox(self.centralwidget)

        self.groupBox\_2.setGeometry(QtCore.QRect(20, 270, 191, 171))

        self.groupBox\_2.setObjectName("groupBox\_2")

        self.label\_8 = QtWidgets.QLabel(self.groupBox\_2)

        self.label\_8.setGeometry(QtCore.QRect(10, 20, 71, 16))

        self.label\_8.setObjectName("label\_8")

        self.label\_9 = QtWidgets.QLabel(self.groupBox\_2)

        self.label\_9.setGeometry(QtCore.QRect(10, 40, 71, 16))

        self.label\_9.setObjectName("label\_9")

        self.label\_10 = QtWidgets.QLabel(self.groupBox\_2)

        self.label\_10.setGeometry(QtCore.QRect(10, 60, 47, 13))

        self.label\_10.setObjectName("label\_10")

        self.label\_11 = QtWidgets.QLabel(self.groupBox\_2)

        self.label\_11.setGeometry(QtCore.QRect(10, 80, 47, 13))

        self.label\_11.setObjectName("label\_11")

        self.inp\_ruge\_step = QtWidgets.QLineEdit(self.groupBox\_2)

        self.inp\_ruge\_step.setGeometry(QtCore.QRect(90, 20, 71, 20))

        self.inp\_ruge\_step.setObjectName("inp\_ruge\_step")

        self.inp\_plot\_step = QtWidgets.QLineEdit(self.groupBox\_2)

        self.inp\_plot\_step.setGeometry(QtCore.QRect(90, 40, 71, 20))

        self.inp\_plot\_step.setObjectName("inp\_plot\_step")

        self.inp\_min\_t = QtWidgets.QLineEdit(self.groupBox\_2)

        self.inp\_min\_t.setGeometry(QtCore.QRect(90, 60, 71, 20))

        self.inp\_min\_t.setObjectName("inp\_min\_t")

        self.inp\_max\_t = QtWidgets.QLineEdit(self.groupBox\_2)

        self.inp\_max\_t.setGeometry(QtCore.QRect(90, 80, 71, 20))

        self.inp\_max\_t.setObjectName("inp\_max\_t")

        self.label\_12 = QtWidgets.QLabel(self.groupBox\_2)

        self.label\_12.setGeometry(QtCore.QRect(10, 100, 51, 16))

        self.label\_12.setObjectName("label\_12")

        self.label\_13 = QtWidgets.QLabel(self.groupBox\_2)

        self.label\_13.setGeometry(QtCore.QRect(10, 120, 47, 13))

        self.label\_13.setObjectName("label\_13")

        self.label\_14 = QtWidgets.QLabel(self.groupBox\_2)

        self.label\_14.setGeometry(QtCore.QRect(10, 140, 47, 13))

        self.label\_14.setObjectName("label\_14")

        self.inp\_offset\_x = QtWidgets.QLineEdit(self.groupBox\_2)

        self.inp\_offset\_x.setGeometry(QtCore.QRect(90, 100, 71, 20))

        self.inp\_offset\_x.setObjectName("inp\_offset\_x")

        self.inp\_offset\_y = QtWidgets.QLineEdit(self.groupBox\_2)

        self.inp\_offset\_y.setGeometry(QtCore.QRect(90, 120, 71, 20))

        self.inp\_offset\_y.setObjectName("inp\_offset\_y")

        self.inp\_offset\_z = QtWidgets.QLineEdit(self.groupBox\_2)

        self.inp\_offset\_z.setGeometry(QtCore.QRect(90, 140, 71, 20))

        self.inp\_offset\_z.setObjectName("inp\_offset\_z")

        self.btn\_traject = QtWidgets.QPushButton(self.centralwidget)

        self.btn\_traject.setGeometry(QtCore.QRect(220, 290, 101, 41))

        self.btn\_traject.setObjectName("btn\_traject")

        self.btn\_exit = QtWidgets.QPushButton(self.centralwidget)

        self.btn\_exit.setGeometry(QtCore.QRect(220, 390, 101, 41))

        self.btn\_exit.setObjectName("btn\_exit")

        MainWindow.setCentralWidget(self.centralwidget)

        self.menubar = QtWidgets.QMenuBar(MainWindow)

        self.menubar.setGeometry(QtCore.QRect(0, 0, 336, 21))

        self.menubar.setObjectName("menubar")

        MainWindow.setMenuBar(self.menubar)

        self.statusbar = QtWidgets.QStatusBar(MainWindow)

        self.statusbar.setObjectName("statusbar")

        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)

        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):

        \_translate = QtCore.QCoreApplication.translate

        MainWindow.setWindowTitle(\_translate("MainWindow", "RungeKutta4th"))

        self.btn\_start.setText(\_translate("MainWindow", "Портрет"))

        self.groupBox\_eq\_count.setTitle(

            \_translate("MainWindow", "Количество уравнений"))

        self.radioButton\_two\_eq.setText(

            \_translate("MainWindow", "2 уравнения"))

        self.radioButton\_three\_eq.setText(

            \_translate("MainWindow", "3 уравнения"))

        self.groupBox.setTitle(\_translate("MainWindow", "Система уравнений"))

        self.label.setText(\_translate("MainWindow", "dx/dt"))

        self.label\_2.setText(\_translate("MainWindow", "dy/dt"))

        self.label\_3.setText(\_translate("MainWindow", "dz/dt"))

        self.groupBox\_start\_cond.setTitle(

            \_translate("MainWindow", "Начальные условия"))

        self.label\_4.setText(\_translate("MainWindow", "t0"))

        self.label\_5.setText(\_translate("MainWindow", "x(t0)"))

        self.label\_6.setText(\_translate("MainWindow", "y(t0)"))

        self.label\_7.setText(\_translate("MainWindow", "z(t0)"))

        self.groupBox\_2.setTitle(\_translate("MainWindow", "Настройки"))

        self.label\_8.setText(\_translate("MainWindow", "шаг метода"))

        self.label\_9.setText(\_translate("MainWindow", "шаг графика"))

        self.label\_10.setText(\_translate("MainWindow", "мин t"))

        self.label\_11.setText(\_translate("MainWindow", "макс t"))

        self.btn\_traject.setText(\_translate("MainWindow", "Траектория"))

        self.btn\_exit.setText(\_translate("MainWindow", "Выход"))

        self.label\_12.setText(\_translate("MainWindow", "отступ x"))

        self.label\_13.setText(\_translate("MainWindow", "отступ y"))

        self.label\_14.setText(\_translate("MainWindow", "отступ z"))

        self.inp\_offset\_x.setText(\_translate("MainWindow", "30"))

        self.inp\_offset\_y.setText(\_translate("MainWindow", "20"))

        self.inp\_offset\_z.setText(\_translate("MainWindow", "30"))

        self.inp\_ruge\_step.setText(\_translate("MainWindow", "0.001"))

        self.inp\_plot\_step.setText(\_translate("MainWindow", "0.1"))

        self.inp\_min\_t.setText(\_translate("MainWindow", "-2.0"))

        self.inp\_max\_t.setText(\_translate("MainWindow", "2.0"))

        self.inp\_dxdt.setText(\_translate("MainWindow", "-2 \* x -2 \* y"))

        self.inp\_dydt.setText(\_translate("MainWindow", "-1 \* x + 2 \* y"))

        self.inp\_dzdt.setText(\_translate("MainWindow", "5 \* y + cos(z)"))

        self.inp\_t0.setText(\_translate("MainWindow", "0.0"))

        self.inp\_x0.setText(\_translate("MainWindow", "2.0"))

        self.inp\_y0.setText(\_translate("MainWindow", "7.0"))

        self.inp\_z0.setText(\_translate("MainWindow", "4.0"))

    def my\_setup\_ui(self):

        self.add\_functions()

        self.radioButton\_two\_eq.setChecked(True)

        self.hide\_z()

    def add\_functions(self):

        self.btn\_start.clicked.connect(lambda: self.start\_portrait())

        self.btn\_traject.clicked.connect(lambda: self.start\_traject())

        self.radioButton\_two\_eq.toggled.connect(lambda: self.hide\_z())

        self.radioButton\_three\_eq.toggled.connect(lambda: self.show\_z())

        self.btn\_exit.clicked.connect(lambda: exit())

    def example\_2d(self):

        self.inp\_ruge\_step.setText("0.001")

        self.inp\_plot\_step.setText("0.1")

        self.inp\_min\_t.setText("-2.0")

        self.inp\_max\_t.setText("2.0")

        self.inp\_dxdt.setText("-2 \* x -2 \* y")

        self.inp\_dydt.setText("-1 \* x + 2 \* y")

        self.inp\_t0.setText("0.0")

        self.inp\_x0.setText("2.0")

        self.inp\_y0.setText("7.0")

    def example\_3d(self):

        self.inp\_ruge\_step.setText("0.001")

        self.inp\_plot\_step.setText("0.1")

        self.inp\_min\_t.setText("-3.0")

        self.inp\_max\_t.setText("3.0")

        self.inp\_dxdt.setText("x - y\*\*2 - z\*\*2")

        self.inp\_dydt.setText("-y")

        self.inp\_dzdt.setText("-z")

        self.inp\_t0.setText("0.0")

        self.inp\_x0.setText("0.0")

        self.inp\_y0.setText("0.0")

        self.inp\_z0.setText("0.0")

    def hide\_z(self):

        if self.radioButton\_two\_eq.isChecked():

            self.label\_3.hide()

            self.inp\_dzdt.hide()

            self.label\_7.hide()

            self.inp\_z0.hide()

            self.example\_2d()

    def show\_z(self):

        if self.radioButton\_three\_eq.isChecked():

            self.label\_3.show()

            self.inp\_dzdt.show()

            self.label\_7.show()

            self.inp\_z0.show()

            self.example\_3d()

    def start\_portrait(self):

        if self.radioButton\_two\_eq.isChecked():

            dxdt\_str = self.inp\_dxdt.text()

            dydt\_str = self.inp\_dydt.text()

            t0\_str = self.inp\_t0.text()

            x0\_str = self.inp\_x0.text()

            y0\_str = self.inp\_y0.text()

            runge\_step\_str = self.inp\_ruge\_step.text()

            plot\_step\_str = self.inp\_plot\_step.text()

            min\_t\_str = self.inp\_min\_t.text()

            max\_t\_str = self.inp\_max\_t.text()

            offset\_x\_str = self.inp\_offset\_x.text()

            offset\_y\_str = self.inp\_offset\_y.text()

            try:

                dx\_expr = compile(dxdt\_str, '', 'eval')

                dy\_expr = compile(dydt\_str, '', 'eval')

            except:

                QtWidgets.QMessageBox.about(

                    None, 'Ошибка', 'В системе уравнений')

                return

            try:

                t0 = float(t0\_str)

                x0 = float(x0\_str)

                y0 = float(y0\_str)

                runge\_step = float(runge\_step\_str)

                plot\_step = float(plot\_step\_str)

                min\_t = float(min\_t\_str)

                max\_t = float(max\_t\_str)

                offset\_x = float(offset\_x\_str)

                offset\_y = float(offset\_y\_str)

            except:

                QtWidgets.QMessageBox.about(None, 'Ошибка', 'В числовых полях')

            phase\_portrait\_2d(offset\_x, offset\_y, x0, y0, t0,

                              runge\_step, min\_t, max\_t, plot\_step, dx\_expr, dy\_expr)

        if self.radioButton\_three\_eq.isChecked():

            dxdt\_str = self.inp\_dxdt.text()

            dydt\_str = self.inp\_dydt.text()

            dzdt\_str = self.inp\_dzdt.text()

            t0\_str = self.inp\_t0.text()

            x0\_str = self.inp\_x0.text()

            y0\_str = self.inp\_y0.text()

            z0\_str = self.inp\_z0.text()

            runge\_step\_str = self.inp\_ruge\_step.text()

            plot\_step\_str = self.inp\_plot\_step.text()

            min\_t\_str = self.inp\_min\_t.text()

            max\_t\_str = self.inp\_max\_t.text()

            offset\_x\_str = self.inp\_offset\_x.text()

            offset\_y\_str = self.inp\_offset\_y.text()

            offset\_z\_str = self.inp\_offset\_z.text()

            try:

                dx\_expr = compile(dxdt\_str, '', 'eval')

                dy\_expr = compile(dydt\_str, '', 'eval')

                dz\_expr = compile(dzdt\_str, '', 'eval')

            except:

                QtWidgets.QMessageBox.about(

                    None, 'Ошибка', 'В системе уравнений')

                return

            try:

                t0 = float(t0\_str)

                x0 = float(x0\_str)

                y0 = float(y0\_str)

                z0 = float(z0\_str)

                runge\_step = float(runge\_step\_str)

                plot\_step = float(plot\_step\_str)

                min\_t = float(min\_t\_str)

                max\_t = float(max\_t\_str)

                offset\_x = float(offset\_x\_str)

                offset\_y = float(offset\_y\_str)

                offset\_z = float(offset\_z\_str)

            except:

                QtWidgets.QMessageBox.about(None, 'Ошибка', 'В числовых полях')

            phase\_portrait\_3d(offset\_x, offset\_y, offset\_z, x0, y0, z0, t0,

                              runge\_step, min\_t, max\_t, plot\_step, dx\_expr, dy\_expr, dz\_expr)

    def start\_traject(self):

        if self.radioButton\_two\_eq.isChecked():

            dxdt\_str = self.inp\_dxdt.text()

            dydt\_str = self.inp\_dydt.text()

            t0\_str = self.inp\_t0.text()

            x0\_str = self.inp\_x0.text()

            y0\_str = self.inp\_y0.text()

            runge\_step\_str = self.inp\_ruge\_step.text()

            plot\_step\_str = self.inp\_plot\_step.text()

            min\_t\_str = self.inp\_min\_t.text()

            max\_t\_str = self.inp\_max\_t.text()

            try:

                dx\_expr = compile(dxdt\_str, '', 'eval')

                dy\_expr = compile(dydt\_str, '', 'eval')

            except:

                QtWidgets.QMessageBox.about(

                    None, 'Ошибка', 'В системе уравнений')

                return

            try:

                t0 = float(t0\_str)

                x0 = float(x0\_str)

                y0 = float(y0\_str)

                runge\_step = float(runge\_step\_str)

                plot\_step = float(plot\_step\_str)

                min\_t = float(min\_t\_str)

                max\_t = float(max\_t\_str)

            except:

                QtWidgets.QMessageBox.about(None, 'Ошибка', 'В числовых полях')

            fig = plt.figure(1)

            ax = fig.add\_subplot(1, 1, 1)

            ax.grid(True, which='both')

            ax.set\_ylim(-100, 100)

            ax.set\_xlim(-100, 100)

            ax.spines['left'].set\_position('center')

            ax.spines['bottom'].set\_position('center')

            ax.spines['top'].set\_visible(False)

            ax.spines['right'].set\_visible(False)

            phase\_trajectory\_2d(x0, y0, t0, runge\_step, min\_t,

                                max\_t, plot\_step, dx\_expr, dy\_expr, ax)

            plt.show()

        if self.radioButton\_three\_eq.isChecked():

            dxdt\_str = self.inp\_dxdt.text()

            dydt\_str = self.inp\_dydt.text()

            dzdt\_str = self.inp\_dzdt.text()

            t0\_str = self.inp\_t0.text()

            x0\_str = self.inp\_x0.text()

            y0\_str = self.inp\_y0.text()

            z0\_str = self.inp\_z0.text()

            runge\_step\_str = self.inp\_ruge\_step.text()

            plot\_step\_str = self.inp\_plot\_step.text()

            min\_t\_str = self.inp\_min\_t.text()

            max\_t\_str = self.inp\_max\_t.text()

            try:

                dx\_expr = compile(dxdt\_str, '', 'eval')

                dy\_expr = compile(dydt\_str, '', 'eval')

                dz\_expr = compile(dzdt\_str, '', 'eval')

            except:

                QtWidgets.QMessageBox.about(

                    None, 'Ошибка', 'В системе уравнений')

                return

            try:

                t0 = float(t0\_str)

                x0 = float(x0\_str)

                y0 = float(y0\_str)

                z0 = float(z0\_str)

                runge\_step = float(runge\_step\_str)

                plot\_step = float(plot\_step\_str)

                min\_t = float(min\_t\_str)

                max\_t = float(max\_t\_str)

            except:

                QtWidgets.QMessageBox.about(None, 'Ошибка', 'В числовых полях')

            fig = plt.figure()

            ax = fig.add\_subplot(111, projection='3d')

            phase\_trajectory\_3d(x0, y0, z0, t0, runge\_step, min\_t,

                                max\_t, plot\_step, dx\_expr, dy\_expr, dz\_expr, ax)

            plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

    import sys

    app = QtWidgets.QApplication(sys.argv)

    MainWindow = QtWidgets.QMainWindow()

    ui = Ui\_MainWindow()

    ui.setupUi(MainWindow)

    ui.my\_setup\_ui()

    MainWindow.show()

    sys.exit(app.exec\_())