Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных

систем и технологий

**КУРСОВАЯ РАБОТА**

по дисциплине «**Разработка приложений в визуальных средах**»

на тему: «**Разработка приложения решения обыкновенных дифференциальных уравнений высших порядков**» 1

Выполнил

студент гр.10701119 Маканов Д.В.

Принял

доц. Гурский Н.Н.

Минск 2021

Белорусский национальный технический университет

Кафедра программного обеспечения информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту (работе)**

**по дисциплине «**Разработка приложений в визуальных средах»

Тема: «Разработка приложения решения обыкновенных дифференциальных уравнений высших порядков»

**Исполнитель**: Маканов Д.В. (фамилия, инициалы)

**Студент 2 курса 10701119 группы**

**Руководитель**: доц. Гурский Н.Н. (фамилия, инициалы)

Минск 2021

**Содержание**

[ВВЕДЕНИЕ 4](#_Toc71488340)

[1 МАТЕМАТИЧЕСКАЯ ФОРМУЛИРОВКА ЗАДАЧИ 5](#_Toc71488341)

[2 ОПИСАНИЕ ПРОГРАММЫ 7](#_Toc71488342)

[2.1 Структурная схема программы 7](#_Toc71488343)

[2.2 Описание разработанного класса 7](#_Toc71488344)

[2.3 Основные возможности программы 8](#_Toc71488345)

[2.4 Средства использования сервисов, предоставляемых MicrosoftOffice 8](#_Toc71488346)

[3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 9](#_Toc71488347)

[4 МЕТОДИКА ИСПЫТАНИЙ 13](#_Toc71488348)

[ЗАКЛЮЧЕНИЕ 15](#_Toc71488349)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 16](#_Toc71488350)

[ПРИЛОЖЕНИЕ A 17](#_Toc71488351)

# ВВЕДЕНИЕ

Вычислительная математика — [раздел математики](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D1%8B_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B8), включающий круг вопросов, связанных с производством разнообразных вычислений. В более узком понимании вычислительная математика — теория численных методов решения типовых математических задач. Современная вычислительная математика включает в круг своих проблем изучение особенностей вычисления с применением [компьютеров](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80). Вычислительная математика обладает широким кругом прикладных применений для проведения научных и инженерных расчётов. На её основе в последнее десятилетие образовались такие новые области естественных наук, как [вычислительная физика](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%B8%D0%B7%D0%B8%D0%BA%D0%B0), [вычислительная химия](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%85%D0%B8%D0%BC%D0%B8%D1%8F), [вычислительная биология](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F) и так далее.

Дифференциальные уравнения - раздел математики, изучающий теорию и способы решения уравнений, содержащих искомую функцию и ее производные различных порядков одного аргумента (обыкновенные дифференциальные) или нескольких аргументов (дифференциальные уравнения в частных производных). В самом уравнении участвует не только неизвестная функция, но и различные ее производные. Дифференциальным уравнением описывается связь между неизвестной функцией и ее производными. Такие связи отыскиваются в различных областях знаний: в механике, физике, химии, биологии, экономике и др. Дифференциальные уравнения применяются для математического описания природных явлений. Так, например, в биологии дифференциальные уравнения применяются для описания популяции; в физике многие законы можно описать с помощью дифференциальных уравнений. Широкое применение находят дифференциальные уравнения и в моделях экономической динамики. В данных моделях отражается не только зависимость переменных от времени, но и их взаимосвязь во времени.

Целью курсовой работы является разработка приложения численного решения обыкновенных дифференциальных уравнений высших порядков. Для работы был выбран язык программирования Python, который даёт возможность быстрого создания приложений с графическим интерфейсом, а также позволяет реализовать концепции ООП.

# 1 МАТЕМАТИЧЕСКАЯ ФОРМУЛИРОВКА ЗАДАЧИ

Методы численного решения обыкновенных дифференциальных уравнений (ОДУ) применяются, когда отсутствует или затруднено аналитическое, или приближенное решение. Также, как преимущество перед аналитическими и приближенными методами можно выделить большую универсальность и удобство программирования численных методов. Совокупность методов численного решения ОДУ, использующих сеточные функции можно условно разделить по типу сетки: с постоянным шагом сетки; с переменным шагом сетки (с контролем ошибок). По типу решаемых задач: для решения начальной задачи (задачи Коши); для решения краевой задачи. По типу вычисления значений в узлах сетки: явные (значение вычисляется явно, в один шаг); неявные (для вычисления значений требуется решать уравнения либо системы уравнений).

Большинство методов численного решения ОДУ разработаны для решения одиночных уравнений первого порядка и их систем, уравнения и системы высших порядком приводятся к системам уравнений первого порядка путем замены переменных. Например, уравнение второго порядка *y'' = f (x, y, y’)* решается путем введения замены *t = y’* и приводится к системе уравнений ОДУ первого порядка:

В данном приложении применяются методы Эйлера, Рунге-Кутты третьего и четвертого порядков. Рассмотрим метод Эйлера. Метод Эйлера является самым простым численным методом решения ОДУ, однако его применение довольно ограничено из-за его невысокого порядка сходимости и плохой устойчивости. Данный подход основан на том свойстве производной, что она равна тангенсу угла наклона касательной в данной точке . C другой стороны, имеет место равенство Отсюда, выражая неизвестное значение *y1* через известные, имеем . Аналогично для следующего узла сетки . Общая формула имеет вид:

.

Рассмотрим метод Рунге-Кутты. Распишем функцию *y = y(x)* в ряд Тейлора по шагу сетки *h* до степени, равной требуемому порядку аппроксимации включительно ***k***:

В данном случае δ определяется как результат вычисления всех производных, входящих в ряд Тейлора, записанном до нужной степени. Такое вычисление неудобно тем, что необходимо находить значения производных функции *f (x, y)* высших порядков, что создает дополнительные вычислительные трудности. В семействе методов Рунге-Кутты данное затруднение обходиться аппроксимацией зависимости, содержащей производные, выражением, зависящим только от координат узлов сетки, значений функции в узлах и размера шага:

Функция Ф (*x, y, h*) имеет очень сложный вид, который в зависимости от порядка аппроксимации ***k*** имеет *k* коэффициентов, выражающиеся через координаты точек и шаг. Приведем формулы для метода Рунге-Кутты третьего порядка:

*k1 = h \* f(xi , yi)  
k2 = h \* f(xi + h / 2, yi + k1 / 2)  
k3 = h \* f(xi + h, yi + k2)  
yi+1 = yi + 1 / 4 \* (k1 + 2 \* k2 + k3)*

Приведем формулы для метода Рунге-Кутты четвертого порядка:

*k1 = h \* f(xi , yi),  
k2 = h \* f(xi + h / 2, yi + k1 / 2),*

*k3 = h \* f(xi + h / 2, yi + k2 / 2),  
k4 = h \* f(xi + h, yi + k3),  
yi+1 = yi + 1 / 6 \* (k1 + 2 \* k2 + 2 \* k3 + k4).*

# 2 ОПИСАНИЕ ПРОГРАММЫ

## **2.1 Структурная схема программы**

Представленное в данной работе приложение является клиент-серверным. Серверная часть программы реализует численное решение ОДУ, а в клиентской части программы пользователь самостоятельно вводит исходные данные: тип уравнения, метод решения, само уравнение, начальные значения для задачи Коши, интервал и шаг. Структурная схема программы изображена на рисунке 2.1.

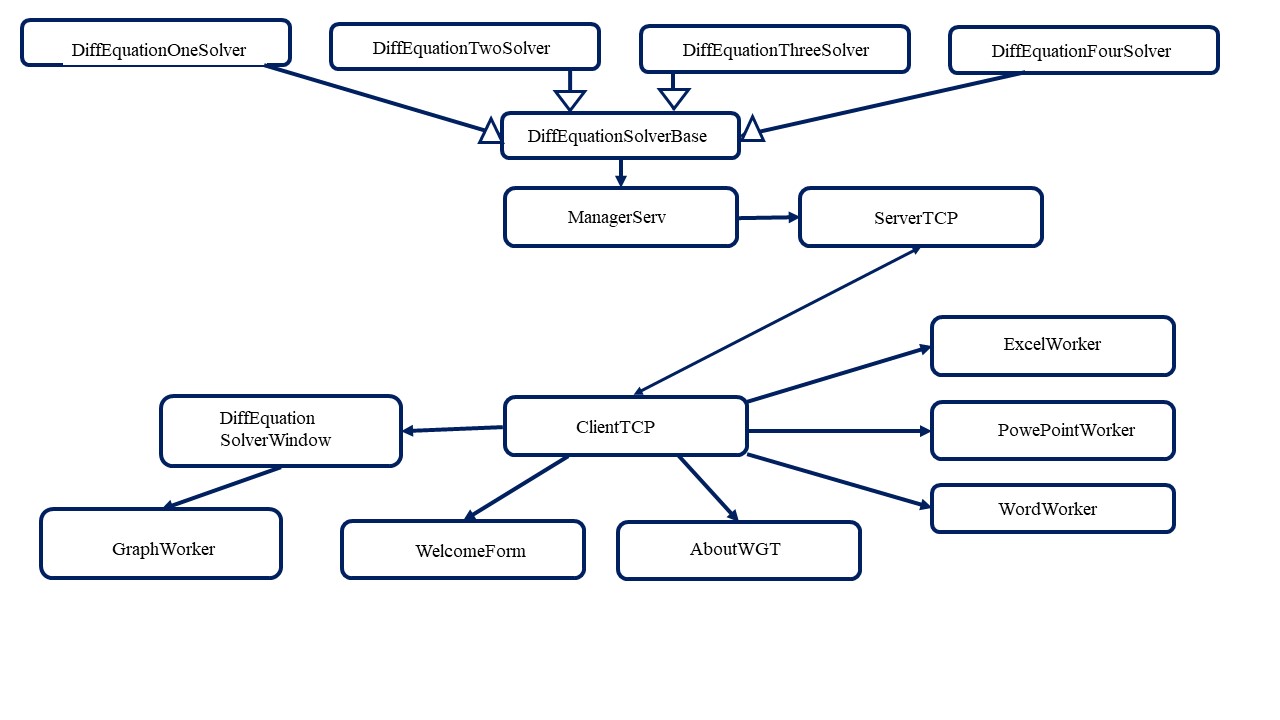
****

Рисунок 2.1 – Структурная схема связей программы.

Стрелочкой с белым треугольником показано наследование классов DiffEquationOneSolver, DiffEquationTwoSolver, DiffEquationThreeSolver, DiffEquationFourSolver от DiffEquationSolverBase.

Клиент-серверное приложение реализовано на базе протокола TCP и потоковых сокетов. Для сервера использовалась многопоточная модель, в которой создается пул потоков, каждым из которых обслуживается подключенный клиент. Все передаваемые данные формируются в строку и разделяются символом “^” и потом преобразуются в списки исходных данных.

## **2.2 Описание разработанного класса**

В результате анализа предметной области для её описания и моделирования был разработан класс DiffEquationSolverBase, инкапсулирующий поля, методы и свойства, применимые к рассматриваемому объекту.

protected float interval\_beg; # начальное значение интервала  
protected float interval\_end; # конечное значение интервала  
protected float step;# шаг  
protected str equation; # уравнение

protected str reformat\_expression(str expression): # перевод уравнения в удобный для обработки вид

protected str is\_right\_expression(str expression): # проверка уравнения на корректность

protected float equation\_generator(float x, float 0, float y2, float y3, float y4): # вычисление значения поставленной функции

Каждый класс наследник имеет методы:

public float euler\_method(args):

public float runge\_third\_method(args):

public float runge\_forth\_method(args):

Где args – список аргументов для каждого такого метода у соответствующего класса наследника.

## **2.3 Основные возможности программы**

Для работы с программой пользователю необходимо ввести DNS сервера. После этого ему открывается главное окно (DesignEquationSolverWindow). Функционал, который предоставляет форма:

* Поля для ввода типа уравнения, метод решения, само уравнение, начальные значения для задачи Коши, интервал и шаг
* Вывод данных в:
  + MS Word
  + MS Excel
  + MS PowerPoint
* Вызов помощи окна помощи.
* Вызов окна “О программе”
* Вывод графика-решение ОДУ
* Вывод результатов вычислений в виде таблицы

Пользователь выбирает тип уравнения (первого, второго, третьего, четвертого порядка), вводит само уравнение, интервал и шаг, выбирает метод решения.При выборе кнопки «Экспорт» у пользователя есть выбор для экспорта в MS Word, MS Excel, MS PowerPoint.При использовании кнопки «Справка» можно открыть форму “О программе” или систему помощи.

## **2.4 Средства использования сервисов, предоставляемых Microsoft Office**

Программа обеспечивает возможность вывода данных не только на экран внутри самого приложения, но и с помощью средств MSOffice. Программа обеспечивает взаимодействие с серверами автоматизации приложений MSWord, MS Excel, MS PowerPoint. Для этого используются классы WordWorker, EcelWorker, PowerPointWorker. Использованные библиотеки:

* docx – библиотека работы с Word;
* openpyxl – библиотека работы с Excel;
* pptx – библиотека работы с PowerPoint.

Данные классы создают документ с соответствующем расширением в директории, выбранной пользователем. В файлы MS Office сохраняются данные, введенные в программу, а также скриншот программы.

# 3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для запуска приложения необходимо сначала запустить сервер, расположеный в модуле server\_tcp.py, потом вызвать клиента из файла main\_client.py. После вызова приложения появится оригинальная заставка, изображённая на рисунке 3.1.

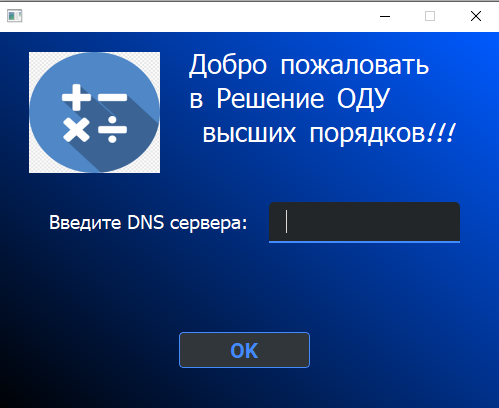


Рисунок 3.1 – Приветственное окно

После заставки в случае подключения к серверу появляется главное окно программы (см. рисунок 3.2).

Ввод уравнений осуществляется в поле для ввода уравнений. Правила ввода:

* Переменная в уравнении обозначается x, функция y, первая производная y', вторая производная y'', третья производная y''.
* Разрешенные математические функции: sin, cos, tan, log, exp, sqrt(корень квадратный), fabs(модуль)
* Оператор степени - \*\*: 2\*\*5 = 32.

Минимальная точность шага 0,001. В интервале могут вводиться вещественные числа с двумя знаками после запятой. При изменении интервала график автоматически подстраивается под интервал по оси абсцисс.

При выборе типа уравнения автоматически появляются поля для задания начальных условий и в том количестве, которое соответствует типу уравнения.

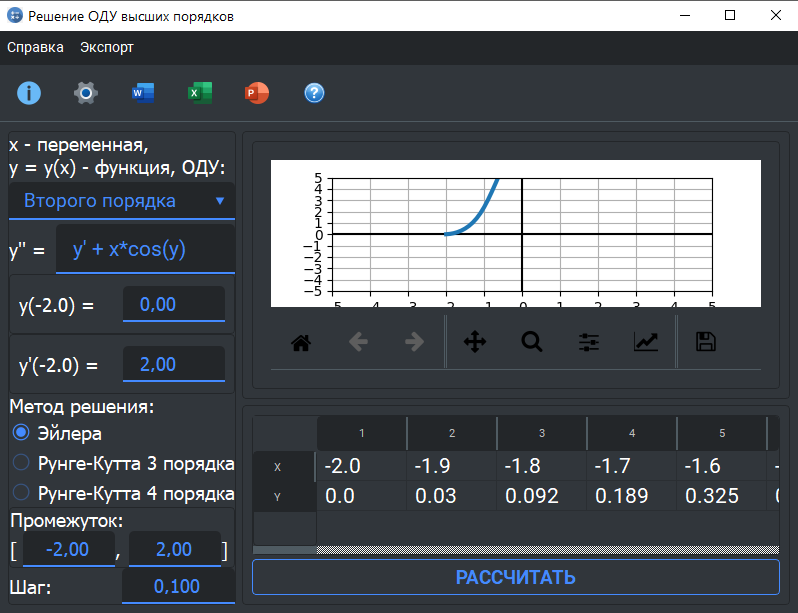


Рисунок 3.2 - Главное окно и пример работы программы

Главная форма имеет:

* Главное меню (см. рисунок 3.3)
* Панель с быстрыми действиями (см. рисунок 3.5)
* Области ввода данных
* Управляющие кнопки (см. рисунок 3.8)

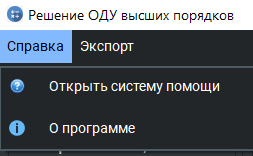


Рисунок 3.3 – Пункты главного меню и подпункта Справка

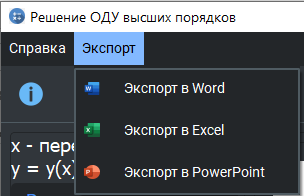


Рисунок 3.4 – Подпункты элемента меню «Экспорт»

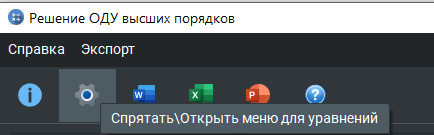


Рисунок 3.5 – Панель с быстрыми действиями и подсказками по ним

Подпункты меню «Экспорт» экспортируют данные в документы формата MS Office.Пункт меню «О программе» выводит информацию о создателе программы. Пункт меню «Система помощи» вызывает help-файл с небольшим руководством пользователя и описанием математической модели (см. рисунок 3.6)

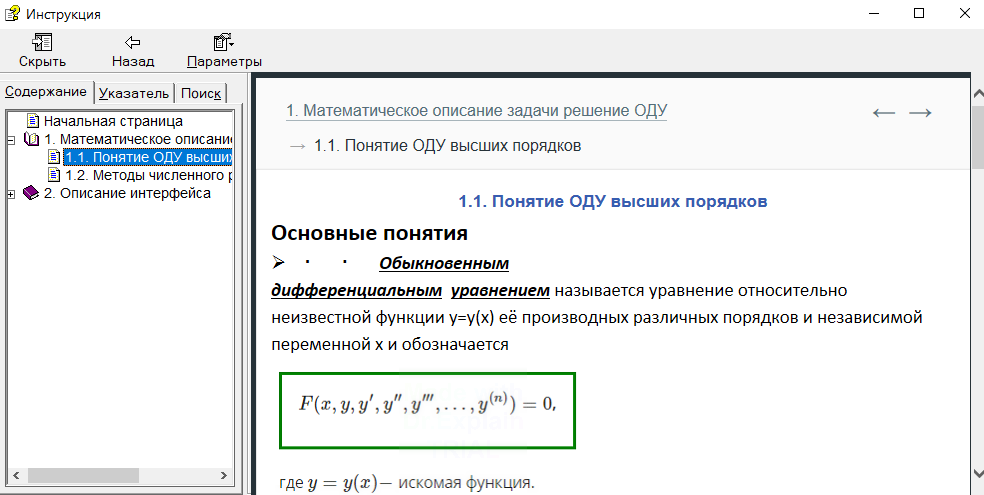


Рисунок 3.6 – Руководство пользователя, вызываемое клавишей «Система помощи»

При выборе кнопки «Рассчитать» отображается график и таблица с расчетами по решению заданного уравнения.

Для расширенной работы с графиком, есть панель (см. рисунок 3.7) с кнопками, позволяющими расширить область просмотра графика и перемещаться по декартовой системе координат.

Рисунок 3.7 – Панель для работы с графиком

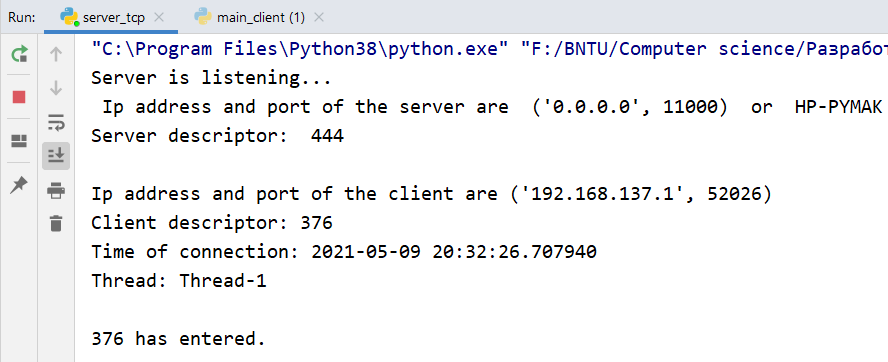
При закрытии клиентского окна приложение автоматически отключается от сервера. Наглядно это можно увидеть на рисунке 3.8, на котором отображено консольное окно серверной части приложения.

Рисунок 3.8 – Консольное окно серверной части приложения

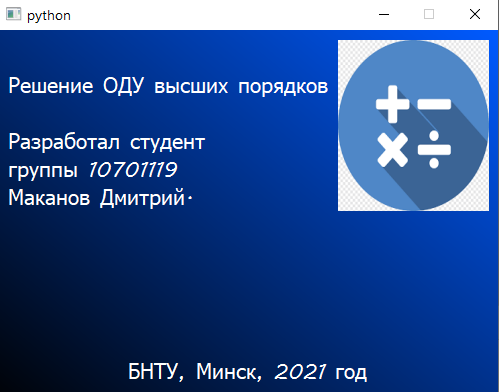
Через пункт меню “Справка” можно вызвать окно “О программе”, представленное на рисунке 3.9.

Рисунок 3.9 – Окно “О программе”

# 4 МЕТОДИКА ИСПЫТАНИЙ

Целью проведения испытаний является проверка работоспособности (надежности) программы при различных условиях её функционирования.

Программа должна обеспечивать корректность ввода исходных данных (путем осуществления соответствующих проверок и информирования пользователя о возникших неточностях в работе), а также получение непротиворечивого результата.

Для обеспечения нормальной работы программы требуется нормальная работа сервера, а также приложений Word и Excel.

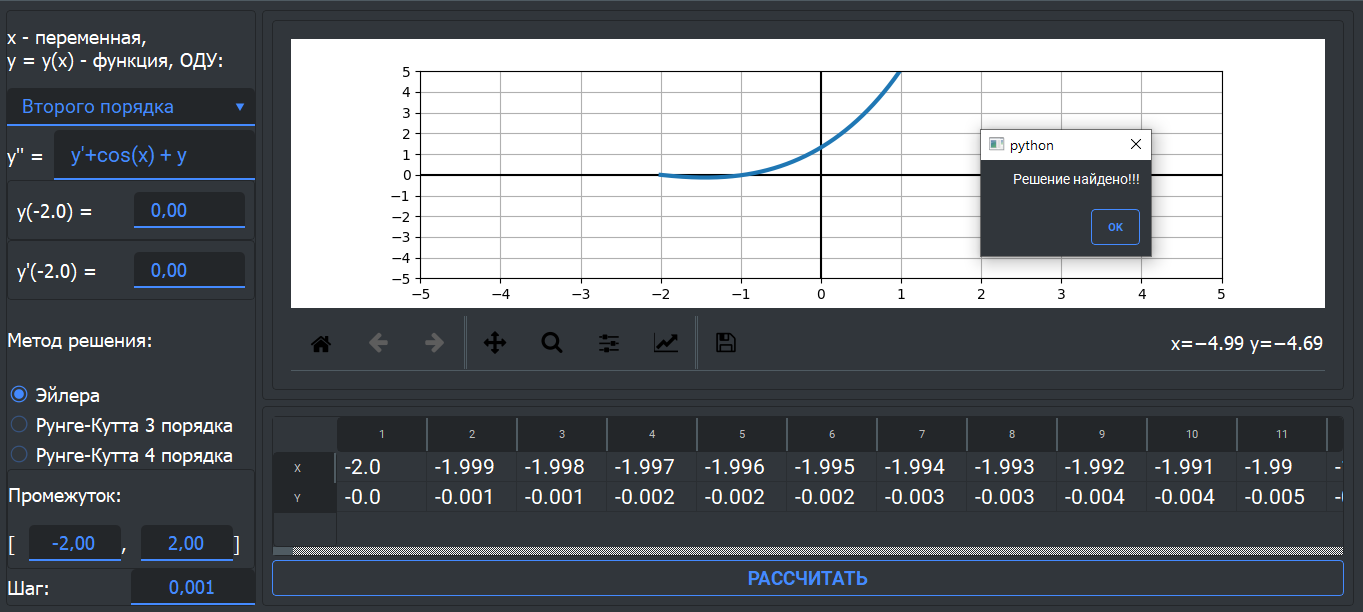
Первое испытание будет проведено при корректных данных (см. рисунок 4.1, 4.2).

Рисунок 4.1 – Ввод корректных данных и вывод сообщения о найденном решении

Второе испытание будет проведено при неверном вводе уравнения, его результат можно увидеть на рисунке 4.2.

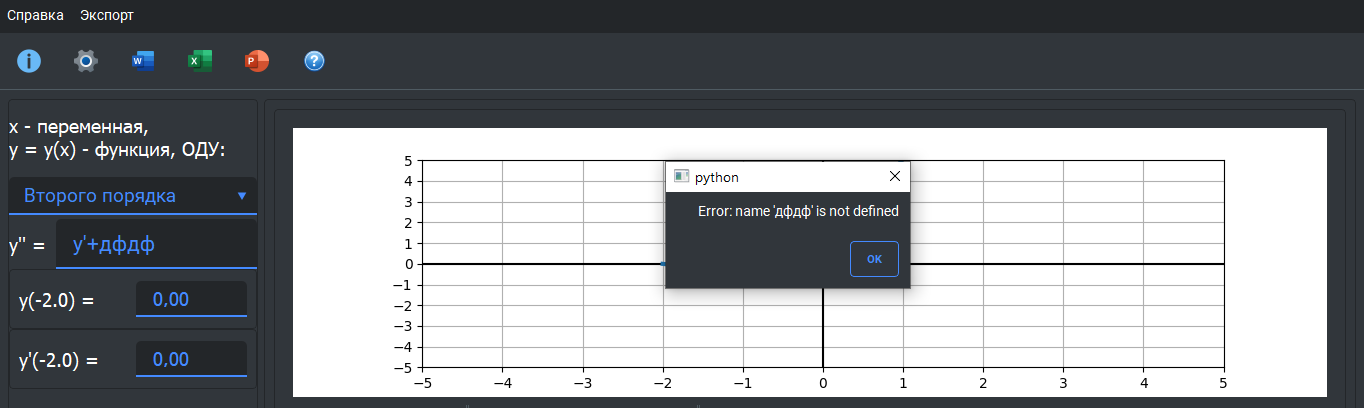
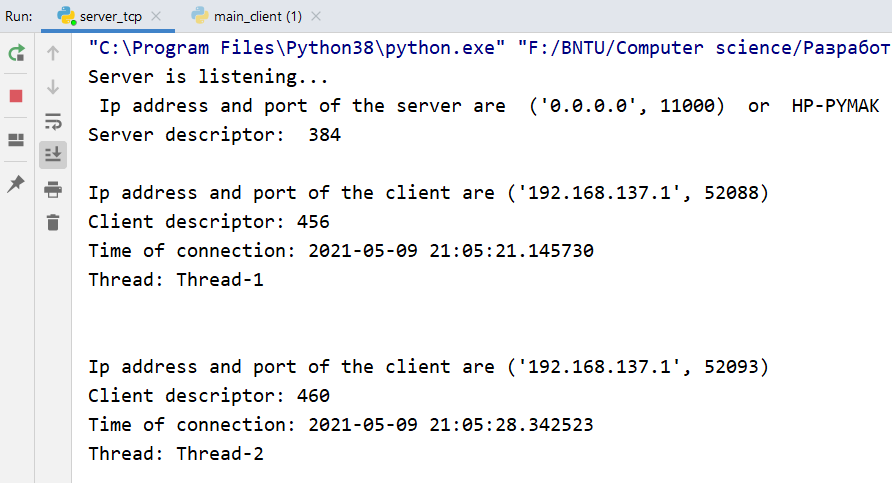


Рисунок 4.2 – Ввод некорректных данных и вывод сообщения об ошибке

Третье испытание на работу нескольких клиентских программ.



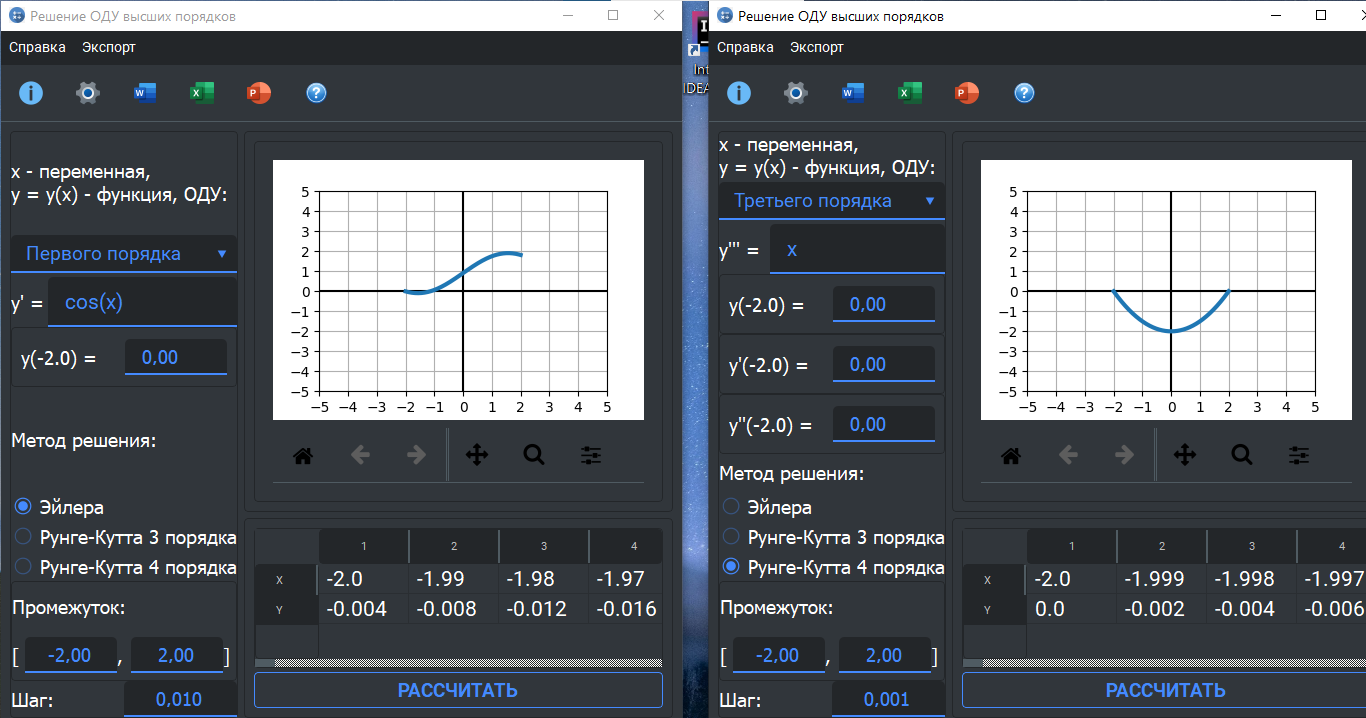
Рисунок 4.3 – Вывод серверной части приложения

Рисунок 4.4 – Работа нескольких клиентов

В результате тестирования приложения ошибок обнаружено не было. Следует считать, что программа протестирована, отвечает поставленным требованием и вполне работоспособна.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы был приобретен опыт при разработке объектно-ориентированных программ, закреплен опыт, полученный при прохождении курса «Разработка приложений в визуальных средах», было изучено взаимодействие с серверами автоматизации приложений MicrosoftOffice.

Программа имеет удобный для пользователя интерфейс, различные формы вывода информации. Программа может использоваться для получения изображений из подобных треугольников. В дальнейшем можно доработать программу, чтобы она давала аналитическое решение ОДУ.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Гринчишин Я.Т. и др. Алгоритмы и программы на Бейсике: Учеб. Пособие для студентов пед. Ин-тов по физ.-мат. Спец./Я. Т. Гринчишин, В. И. Ефимов, А. Н. Ломакович. –М.: Просвещение. 1988.-160с.
2. Доусон М. Программируем на Python. – СПб.: Питер, 2014. – 416 с.
3. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ - Плюс, 2011. – 1280 с.

# ПРИЛОЖЕНИЕ A

**Листинг программы**

**Файл main\_client.py**

import sys  
  
from PyQt5.QtWidgets import QApplication  
  
from client.class\_client import Client  
from qt\_material import apply\_stylesheet  
  
if \_\_name\_\_ == **"\_\_main\_\_"**:  
 app = QApplication(sys.argv)  
 client = Client()  
 apply\_stylesheet(app, theme=**'dark\_blue.xml'**)  
 app.exec\_()  
 client.exit\_action()

**Файл class\_client.py**

import os  
import socket  
  
from client.view.DE\_Solver\_window import DiffEquationSolverWindow  
  
from client.view.welcomeWin import WelcomeWin  
  
from util import \*  
from util.NumberArrayWorker import NumberArrayWorker  
from util.StrArrayWorker import StrArrayWorker  
  
SERVER\_ADDRESS = **"127.0.0.1"**PORT = 11000  
BYTES\_PER\_PACKAGE = 100000  
  
  
class Client:  
 def \_\_init\_\_(self):  
 self.client = socket.socket(  
 socket.AF\_INET,  
 socket.SOCK\_STREAM  
 )  
 self.app = DiffEquationSolverWindow()  
 self.app.action.triggered.connect(lambda : os.system(**"help.chm"**))  
  
 self.server\_dns\_name = **""** *# HP-PYMAK* self.welcome\_win = WelcomeWin()  
 self.welcome\_win.show()  
 self.welcome\_win.pushButton.clicked.connect(self.welcome\_commands)  
 self.welcome\_win.lineEdit.returnPressed.connect(self.welcome\_commands)  
  
 self.app.result\_btn.clicked.connect(self.solve\_equation\_action)  
  
 def welcome\_commands(self):  
 try:  
 self.server\_dns\_name = self.welcome\_win.lineEdit.text()  
 self.connect()  
 self.welcome\_win.close()  
 self.app.show()  
 except Exception:  
 self.welcome\_win.warningMsg.show()  
 self.welcome\_win.lineEdit.clear()  
  
 def connect(self):  
 self.client.connect(  
 (self.server\_dns\_name, PORT)  
 )  
  
 def send(self, msg):  
 self.client.send(msg.encode(**"utf-8"**))  
  
 def receive(self):  
 return self.client.recv(BYTES\_PER\_PACKAGE).decode(**"utf-8"**)  
  
 def solve\_equation\_action(self):  
 self.send(SOLVE\_DIFF)  
 self.send(StrArrayWorker.convert\_list\_str\_to\_string(  
 self.app.get\_general\_info()))  
  
 info = self.receive()  
 if **"Error"** not in info:  
 x\_array = NumberArrayWorker.convert\_string\_to\_list\_number(info)  
 y\_array = NumberArrayWorker.convert\_string\_to\_list\_number(self.receive())  
  
 self.app.show\_solution(x\_array, y\_array)  
 else:  
 self.app.msgBox.setText(info)  
 self.app.msgBox.show()  
  
 def exit\_action(self):  
 self.send(EXIT)

**Файл de\_solver\_window.py**  
from PyQt5.QtGui import QIcon  
from PyQt5.QtWidgets import QMainWindow, QTableWidgetItem, QAction, QFileDialog, QMessageBox  
  
from client.view.design\_DE\_Solver import Ui\_MainWindow  
from client.view.images.aboutWGT import AboutWGT  
from client.view.plotWidget import PlotWidget  
  
from util import \*  
from util.OfficeWork.excel import ExcelWorker  
from util.OfficeWork.powerpoint\_ex import PowerPointWorker  
from util.OfficeWork.word import WordWorker  
  
  
class DiffEquationSolverWindow(Ui\_MainWindow, QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setupUi(self)  
 self.\_\_graph = PlotWidget()  
 self.graph\_lay.addWidget(self.\_\_graph)  
  
 self.msgBox = QMessageBox()  
  
 self.\_\_aboutWGT = AboutWGT()  
 self.toolbar = self.addToolBar(**""**)  
 self.toolbar.setMovable(False)  
  
 self.d\_y\_frame.hide()  
 self.dd\_y\_frame.hide()  
 self.ddd\_y\_frame.hide()  
 self.y\_label.setText(**"y("** + str(self.low\_val.value()) + **") = "**)  
  
 self.comboBox.currentIndexChanged.connect(self.\_\_comboBox\_actions)  
  
 self.low\_val.valueChanged.connect(self.make\_low\_box\_actions)  
 self.high\_val.valueChanged.connect(self.make\_high\_box\_actions)  
  
 self.menu\_actions()  
  
 def menu\_actions(self):  
 self.action\_2.triggered.connect(self.\_\_aboutWGT.show)  
 self.action\_2.setText(**"О программе"**)  
 self.action\_2.setIcon(QIcon(**r"client\view\images\info.png"**))  
 self.toolbar.addAction(self.action\_2)  
  
 setts\_hide\_action = QAction(  
 QIcon(**r"client\view\images\setts.png"**),  
 **"Спрятать\Открыть меню для уравнений"**, self)  
  
 setts\_hide\_action.triggered.connect(self.hide\_open\_setts\_menu)  
  
 self.toolbar.addAction(setts\_hide\_action)  
  
 self.action\_Word.setIcon(QIcon(**r"client\view\images\word.png"**))  
 self.action\_Word.setText(**"Экспорт в Word"**)  
 self.action\_Word.triggered.connect(self.to\_word)  
  
 self.action\_Excel.setIcon(QIcon(**r"client\view\images\Xcel.png"**))  
 self.action\_Excel.setText(**"Экспорт в Excel"**)  
 self.action\_Excel.triggered.connect(self.to\_excel)  
  
 self.action\_PowerPoint.setIcon(QIcon(**r"client\view\images\pptx.png"**))  
 self.action\_PowerPoint.setText(**"Экспорт в PowerPoint"**)  
 self.action\_PowerPoint.triggered.connect(self.to\_pptx)  
  
 self.action.setText(**"Открыть систему помощи"**)  
 self.action.setIcon(QIcon(**r"client\view\images\help.png"**))  
  
 self.toolbar.addAction(self.action\_Word)  
 self.toolbar.addAction(self.action\_Excel)  
 self.toolbar.addAction(self.action\_PowerPoint)  
 self.toolbar.addAction(self.action)  
  
 def hide\_open\_setts\_menu(self):  
 if self.setts\_frame.maximumWidth() == 250:  
 self.setts\_frame.setMaximumWidth(0)  
 elif self.setts\_frame.maximumWidth() == 0:  
 self.setts\_frame.setMaximumWidth(250)  
  
 def make\_low\_box\_actions(self):  
 self.\_\_graph.change\_Ox(self.low\_val.value(), self.high\_val.value())  
 self.y\_label.setText(**"y("** + str(self.y\_spin.value()) + **")="**)  
  
 def make\_high\_box\_actions(self):  
 self.\_\_graph.change\_Ox(self.low\_val.value(), self.high\_val.value())  
  
 def show\_solution(self, x, y):  
 self.\_\_graph.clear()  
 self.\_\_graph.plot\_from\_lists(x, y)  
 self.\_\_table\_actions(x, y)  
  
 self.msgBox.setText(**"Решение найдено!!!"**)  
 self.msgBox.show()  
  
 def \_\_comboBox\_actions(self):  
 self.d\_y\_frame.hide()  
 self.dd\_y\_frame.hide()  
 self.ddd\_y\_frame.hide()  
  
 self.y\_spin.clear()  
 self.d\_y\_spin.clear()  
 self.dd\_y\_spin.clear()  
 self.ddd\_y\_spin.clear()  
 if self.comboBox.currentIndex() == 0:  
 self.diffLabel.setText(**"y' = "**)  
  
 self.y\_label.setText(**"y("** + str(self.low\_val.value()) + **") = "**)  
  
 elif self.comboBox.currentIndex() == 1:  
 self.diffLabel.setText(**"y'' = "**)  
 self.d\_y\_frame.show()  
  
 self.y\_label.setText(**"y("** + str(self.low\_val.value()) + **") = "**)  
 self.d\_y\_label.setText(**"y'("** + str(self.low\_val.value()) + **") = "**)  
  
 elif self.comboBox.currentIndex() == 2:  
 self.diffLabel.setText(**"y''' = "**)  
 self.d\_y\_frame.show()  
 self.dd\_y\_frame.show()  
  
 self.y\_label.setText(**"y("** + str(self.low\_val.value()) + **") = "**)  
 self.d\_y\_label.setText(**"y'("** + str(self.low\_val.value()) + **") = "**)  
 self.dd\_y\_label.setText(**"y''("** + str(self.low\_val.value()) + **") = "**)  
  
 elif self.comboBox.currentIndex() == 3:  
 self.diffLabel.setText(**"y'''' = "**)  
 self.d\_y\_frame.show()  
 self.dd\_y\_frame.show()  
 self.ddd\_y\_frame.show()  
  
 self.y\_label.setText(**"y("** + str(self.low\_val.value()) + **") = "**)  
 self.d\_y\_label.setText(**"y'("** + str(self.low\_val.value()) + **") = "**)  
 self.dd\_y\_label.setText(**"y''("** + str(self.low\_val.value()) + **") = "**)  
 self.ddd\_y\_label.setText(**"y'''("** + str(self.low\_val.value()) + **") = "**)  
  
 def \_\_table\_actions(self, x, y):  
 self.tableWidget.setColumnCount(len(x))  
 self.tableWidget.resizeColumnsToContents()  
  
 if len(x) != len(y):  
 return  
  
 for i in range(len(x)):  
 self.tableWidget.setColumnWidth(i, 90)  
 self.tableWidget.setItem(0, i, QTableWidgetItem(str(round(x[i], 3))))  
 if isinstance(y[i], complex):  
 self.tableWidget.setItem(1, i, QTableWidgetItem(str(round(y[i].real, 3))))  
 elif isinstance(y[i], float):  
 self.tableWidget.setItem(1, i, QTableWidgetItem(str(round(y[i], 3))))  
  
 def get\_general\_info(self):  
 *"""  
 Info structure:  
 [DIFF\_EQUATION\_TYPE, EQUATION, METHOD, INTERVAL\_BEG, INTERVAL\_END,  
 STEP, Y, Y', Y'', Y''']  
 """* info = []  
  
 if self.comboBox.currentIndex() == 0:  
 info.append(DIFF\_EQUATION\_TYPE\_ONE)  
 elif self.comboBox.currentIndex() == 1:  
 info.append(DIFF\_EQUATION\_TYPE\_TWO)  
 elif self.comboBox.currentIndex() == 2:  
 info.append(DIFF\_EQUATION\_TYPE\_THREE)  
 elif self.comboBox.currentIndex() == 3:  
 info.append(DIFF\_EQUATION\_TYPE\_FOUR)  
  
 info.append(self.equationLine.text())  
  
 info.append(EULER\_METHOD)  
  
 if self.radioButton\_3.isChecked():  
 info[2] = RUNGE\_THIRD\_METHOD  
  
 elif self.radioButton\_4.isChecked():  
 info[2] = RUNGE\_FOURTH\_METHOD  
  
 info.append(str(self.low\_val.value()))  
 info.append(str(self.high\_val.value()))  
 info.append(str(self.stepSpin.value()))  
  
 info.append(str(self.y\_spin.value()))  
 info.append(str(self.d\_y\_spin.value()))  
 info.append(str(self.dd\_y\_spin.value()))  
 info.append(str(self.ddd\_y\_spin.value()))  
  
 return info  
  
 def get\_general\_info\_for\_presentation\_in\_office(self):  
 result = (**"Уравнение: "** + self.diffLabel.text() +  
 self.equationLine.text())  
  
 result += **"**\n**Тип уравнения: "** if self.comboBox.currentIndex() == 0:  
 result += **"первого порядка "** result += (**"**\n**Начальные условия: "** + self.y\_label.text()  
 + str(self.y\_spin.value()) + **"**\n**"**)  
 elif self.comboBox.currentIndex() == 1:  
 result += **"второго порядка "** result += (**"**\n**Начальные условия: "** + self.y\_label.text()  
 + str(self.y\_spin.value()) + **" "** +  
 self.d\_y\_label.text()  
 + str(self.d\_y\_spin.value()) +  
 **"**\n**"**)  
 elif self.comboBox.currentIndex() == 2:  
 result += **"третьего порядка "** result += (**"**\n**Начальные условия: "** + self.y\_label.text()  
 + str(self.y\_spin.value()) + **" "** +  
 self.d\_y\_label.text()  
 + str(self.d\_y\_spin.value()) + **" "** +  
 self.dd\_y\_label.text()  
 + str(self.dd\_y\_spin.value()) +  
 **"**\n**"**)  
 elif self.comboBox.currentIndex() == 3:  
 result += **"четвертого порядка "** result += (**"**\n**Начальные условия: "** + self.y\_label.text()  
 + str(self.y\_spin.value()) + **" "** +  
 self.d\_y\_label.text()  
 + str(self.d\_y\_spin.value()) + **" "** +  
 self.dd\_y\_label.text()  
 + str(self.dd\_y\_spin.value()) + **" "** +  
 self.ddd\_y\_label.text()  
 + str(self.ddd\_y\_spin.value()) +  
 **"**\n**"**)  
  
 result += **"**\n**Метод решения: "** method = **"Эйлера"** if self.radioButton\_3.isChecked():  
 method = **"Рунге-Кутта 3-го порядка"** elif self.radioButton\_4.isChecked():  
 method = **"Рунге-Кутта 4-го порядка"** result += method  
  
 result += (**"**\n**Промежуток: ["** + str(self.low\_val.value()) +  
 **", "** + str(self.high\_val.value()) + **"]"**)  
  
 result += **"**\n**Шаг: "** + str(self.stepSpin.value()) + **"**\n**"** return result  
  
 def to\_word(self):  
 file\_way = QFileDialog.getExistingDirectory(self, **"Выбрать папку"**, **"."**)  
 word\_worker = WordWorker(file\_way)  
 p = self.grab()  
 p.save(**"test.jpg"**)  
  
 word\_worker.insert\_info(self.get\_general\_info\_for\_presentation\_in\_office(),  
 **"test.jpg"**)  
  
 word\_worker.save()  
  
 self.msgBox.setText(**"Экспортировано в Word по пути**\n**"** + file\_way)  
 self.msgBox.show()  
  
 def to\_excel(self):  
 file\_way = QFileDialog.getExistingDirectory(self, **"Выбрать папку"**, **"."**)  
 excel\_worker = ExcelWorker(file\_way)  
 excel\_worker.append\_info(self.get\_general\_info\_for\_presentation\_in\_office())  
 p = self.grab()  
 p.save(**"test.jpg"**)  
 excel\_worker.insert\_image(**"test.jpg"**, position=**"E1"**)  
  
 excel\_worker.save()  
  
 self.msgBox.setText(**"Экспортировано в Excel по пути**\n**"** + file\_way)  
 self.msgBox.show()  
  
 def to\_pptx(self):  
 file\_way = QFileDialog.getExistingDirectory(self, **"Выбрать папку"**, **"."**)  
 pptx\_worker = PowerPointWorker(file\_way)  
 p = self.grab()  
 p.save(**"test.jpg"**)  
  
 pptx\_worker.insert\_info(self.get\_general\_info\_for\_presentation\_in\_office(),  
 **"test.jpg"**)  
  
 pptx\_worker.save()  
  
 self.msgBox.setText(**"Экспортировано в PowerPoint по пути**\n**"** + file\_way)  
 self.msgBox.show()

**Файл servet\_tcp.py**

import socket  
import threading  
  
from server.logic.exceptions.ExpressionMistakeException import ExpressionMistakeException  
from server.logic.exceptions.ImpossibleExpressionException import ImpossibleExpressionException  
from util.TimeTask import Time  
  
from server.logic.ManagerServ import ManagerServ  
from util.NumberArrayWorker import NumberArrayWorker  
from util.StrArrayWorker import StrArrayWorker  
from util import EXIT, SOLVE\_DIFF  
  
THREADS\_AMOUNT = 5  
  
DEFAULT\_GATEWAY = **"0.0.0.0"**PORT = 11000  
LENGTH\_QUEUE = 10  
BYTES\_PER\_PACKAGE = 100000  
  
  
def serve\_client(server):  
 while True:  
 user\_socket, address = server.accept()  
 print(**'**\n**Ip address and port of the client are '** + str(address) +  
 **"**\n**Client descriptor: "** + str(user\_socket.fileno()) +  
 **"**\n**Time of connection: "** + str(Time.get\_data\_and\_time()) +  
 **"**\n**Thread: "** + threading.current\_thread().name + **"**\n**"**)  
  
 *# получение сообщения клиенту в виде пакета в байтах* mode = user\_socket.recv(BYTES\_PER\_PACKAGE).decode(**"utf-8"**)  
  
 while mode != EXIT:  
  
 if mode == SOLVE\_DIFF:  
 try:  
 info = StrArrayWorker.convert\_string\_to\_list\_str(  
 user\_socket.recv(BYTES\_PER\_PACKAGE).decode(**"utf-8"**))  
  
 data = ManagerServ.solve\_equation(info)  
  
 x\_array = NumberArrayWorker.convert\_list\_number\_to\_string(data[0]).encode(**"utf-8"**)  
 user\_socket.send(x\_array)  
  
 y\_array = NumberArrayWorker.convert\_list\_number\_to\_string(data[1]).encode(**"utf-8"**)  
  
 user\_socket.send(y\_array)  
 except ExpressionMistakeException:  
 user\_socket.send(**"Error: Ошибка в уравнении!!"**.encode(**"utf-8"**))  
 except ImpossibleExpressionException:  
 user\_socket.send(**"Error: Невозможное выражение!!!"**.encode(**"utf-8"**))  
 except Exception as ex:  
 user\_socket.send((**"Error: "** + str(ex)).encode(**"utf-8"**))  
  
 mode = user\_socket.recv(BYTES\_PER\_PACKAGE).decode(**"utf-8"**)  
  
 print(str(user\_socket.fileno()) + **" has entered.**\n**"**)  
 user\_socket.shutdown(socket.SHUT\_RDWR)  
 user\_socket.close()  
  
  
def main():  
 server = socket.socket(  
  
 socket.AF\_INET,  
 socket.SOCK\_STREAM  
 )  
  
 server.bind(  
 (DEFAULT\_GATEWAY, PORT) *# localhost* )  
  
 threads = []  
  
 for i in range(THREADS\_AMOUNT):  
 threads.append(threading.Thread(target=serve\_client, args=(server,)))  
  
 server.listen(LENGTH\_QUEUE)  
 print(**"Server is listening...**\n**"**,  
 **'Ip address and port of the server are '**, server.getsockname(),  
 **" or "**, socket.gethostname(),  
 **"**\n**Server descriptor: "**, server.fileno())  
  
 for item in threads:  
 item.start()  
  
 for item in threads:  
 item.join()  
  
  
if \_\_name\_\_ == **"\_\_main\_\_"**:  
 try:  
 main()  
 except Exception as ex:  
 print(ex)

**Файл diff\_equation\_solver\_base.py**

import keyword  
import re  
  
from server.logic.exceptions.ExpressionMistakeException import ExpressionMistakeException  
from server.logic.exceptions.ImpossibleExpressionException import ImpossibleExpressionException  
  
  
*# def controller(expression, a, b, step, mode, method, y, y2=0, y3=0, y4=0):*class DiffEquationSolverBase:  
 def \_\_init\_\_(self, equation, interval\_beg, interval\_end, step):  
  
 self.\_interval\_beg = interval\_beg  
 self.\_interval\_end = interval\_end  
 self.\_step = step  
  
 self.\_equation = self.\_reformat\_expression(equation)  
  
 if not self.\_is\_right\_expression(self.\_equation):  
 raise ImpossibleExpressionException()  
  
 @staticmethod  
 def \_reformat\_expression(expression):  
 expression = re.sub(**"y'''"**, **"y4"**, expression)  
 expression = re.sub(**"y''"**, **"y3"**, expression)  
  
 return re.sub(**"y'"**, **"y2"**, expression)  
  
 @staticmethod  
 def \_is\_right\_expression(expression):  
 if expression == **""**:  
 return False  
 impossible\_words = keyword.kwlist  
  
 impossible\_words += [**","**, **"print"**, **"exec"**, **"eval"**]  
 for item in impossible\_words:  
 temp = re.findall(item, expression)  
 if len(temp) != 0:  
 return False  
  
 return True  
  
 def \_equation\_generator(self, x=0, y=0, y2=0, y3=0, y4=0):  
 fns = **'''def function(x, y, y2, y3, y4):  
 from math import sin, cos, tan, log, exp, sqrt, fabs  
 return '''** + self.\_equation  
 try:  
 exec(fns)  
 result = eval(**'function('** + str(x) + **','** + str(y) + **','** +  
 str(y2) + **','** + str(y3) + **','** + str(y4) + **')'**)  
 except SyntaxError:  
 raise ExpressionMistakeException()  
 return result  
  
 def get\_equation(self):  
 return self.\_equation  
  
 def set\_equation(self, equation):  
 self.\_equation = self.\_reformat\_expression(equation)  
 if not self.\_is\_right\_expression(self.\_equation):  
 raise ImpossibleExpressionException  
  
 def get\_interval\_beg(self):  
 return self.\_interval\_beg  
  
 def set\_interval\_beg(self, interval\_beg):  
 self.\_interval\_beg = interval\_beg  
  
 def get\_interval\_end(self):  
 return self.\_interval\_end  
  
 def set\_interval\_end(self, interval\_end):  
 self.\_interval\_end = interval\_end  
  
 def get\_step(self):  
 return self.\_step  
  
 def set\_step(self, step):  
 self.\_step = step

**Файл diff\_equation\_one\_solver.py**

from server.logic.diffSolvers.DiffEquationSolverBase import DiffEquationSolverBase  
  
  
class DiffEquationOneSolver(DiffEquationSolverBase):  
 def \_\_init\_\_(self, equation, interval\_beg, interval\_end, step):  
 super().\_\_init\_\_(equation, interval\_beg, interval\_end, step)  
  
 def F(self, x, y):  
  
 return self.\_equation\_generator(x, y)  
  
 def euler\_method(self, y):  
 x\_s = []  
 answers\_euler = []  
 x = self.\_interval\_beg  
 h = self.\_step  
 while x <= self.\_interval\_end + h:  
 y1\_1 = y + h \* self.F(x, y)  
 y = y1\_1  
  
 x\_s.append(x)  
 answers\_euler.append(y)  
  
 x += h  
 return x\_s, answers\_euler  
  
 def runge\_third\_method(self, y):  
 x\_s = []  
 answers\_runge = []  
 x = self.\_interval\_beg  
 h = self.\_step  
 while x <= self.\_interval\_end + h:  
 k1\_1 = h \* self.F(x, y)  
  
 k2\_1 = h \* self.F(x + h / 2, y + k1\_1 / 2)  
  
 k3\_1 = h \* self.F(x + h, y + k2\_1)  
  
 answers\_runge.append(y)  
 x\_s.append(x)  
 y = y + 1 / 4 \* (k1\_1 + 2 \* k2\_1 + k3\_1)  
 x += h  
  
 return x\_s, answers\_runge  
  
 def runge\_forth\_method(self, y):  
 x\_s = []  
 answers\_runge = []  
 x = self.\_interval\_beg  
 h = self.\_step  
 while x <= self.\_interval\_end + h:  
 k1\_1 = h \* self.F(x, y)  
 k2\_1 = h \* self.F(x + h / 2, y + k1\_1 / 2)  
 k3\_1 = h \* self.F(x + h / 2, y + k2\_1 / 2)  
 k4\_1 = h \* self.F(x + h, y + k3\_1)  
  
 answers\_runge.append(y)  
 x\_s.append(x)  
 y = y + 1 / 6 \* (k1\_1 + 2 \* k2\_1 + 2 \* k3\_1 + k4\_1)  
 x += h  
  
 return x\_s, answers\_runge

**Файл diff\_equation\_two\_solver.py**

from server.logic.diffSolvers.DiffEquationSolverBase import DiffEquationSolverBase  
  
  
class DiffEquationTwoSolver(DiffEquationSolverBase):  
 def \_\_init\_\_(self, equation, interval\_beg, interval\_end, step):  
 super().\_\_init\_\_(equation, interval\_beg, interval\_end, step)  
  
 def F(self, x, y, y2, n):  
 f = 0  
 if n == 1:  
 f = self.\_equation\_generator(x, y, y2)  
 elif n == 2:  
 f = y2  
  
 return f  
  
 def euler\_method(self, y, y2):  
 x\_s = []  
 answers\_euler = []  
 x = self.\_interval\_beg  
 h = self.\_step  
 while x <= self.\_interval\_end + h:  
 y1\_1 = y + h \* self.F(x, y, y2, 1)  
 y1\_2 = y2 + h \* self.F(x, y, y2, 2)  
 y = y1\_1  
 y2 = y1\_2  
  
 x\_s.append(x)  
 answers\_euler.append(y)  
  
 x += h  
 return x\_s, answers\_euler  
  
 def runge\_third\_method(self, y, y2):  
 x\_s = []  
 answers\_runge = []  
 x = self.\_interval\_beg  
 h = self.\_step  
 while x <= self.\_interval\_end + h:  
 k1\_1 = h \* self.F(x, y, y2, 1)  
 k1\_2 = h \* self.F(x, y, y2, 2)  
  
 k2\_1 = h \* self.F(x + h / 2, y + k1\_1 / 2, y2 + k1\_2 / 2, 1)  
 k2\_2 = h \* self.F(x + h / 2, y + k1\_1 / 2, y2 + k1\_2 / 2, 2)  
  
 k3\_1 = h \* self.F(x + h, y + k2\_1, y2 + k2\_2, 1)  
 k3\_2 = h \* self.F(x + h, y + k2\_1, y2 + k2\_2, 2)  
  
 answers\_runge.append(y)  
 x\_s.append(x)  
 y = y + 1 / 4 \* (k1\_1 + 2 \* k2\_1 + k3\_1)  
 y2 = y2 + 1 / 4 \* (k1\_2 + 2 \* k2\_2 + k3\_2)  
 x += h  
  
 return x\_s, answers\_runge  
  
 def runge\_forth\_method(self, y, y2):  
 x\_s = []  
 answers\_runge = []  
 x = self.\_interval\_beg  
 h = self.\_step  
 while x <= self.\_interval\_end + h:  
 k1\_1 = h \* self.F(x, y, y2, 1)  
 k1\_2 = h \* self.F(x, y, y2, 2)  
  
 k2\_1 = h \* self.F(x + h / 2, y + k1\_1 / 2, y2 + k1\_2 / 2, 1)  
 k2\_2 = h \* self.F(x + h / 2, y + k1\_1 / 2, y2 + k1\_2 / 2, 2)  
  
 k3\_1 = h \* self.F(x + h / 2, y + k2\_1 / 2, y2 + k2\_2 / 2, 1)  
 k3\_2 = h \* self.F(x + h / 2, y + k2\_1 / 2, y2 + k2\_2 / 2, 2)  
 k4\_1 = h \* self.F(x + h, y + k3\_1, y2 + k3\_2, 1)  
 k4\_2 = h \* self.F(x + h, y + k3\_1, y2 + k3\_2, 2)  
  
 answers\_runge.append(y)  
 x\_s.append(x)  
 y = y + 1 / 6 \* (k1\_1 + 2 \* k2\_1 + 2 \* k3\_1 + k4\_1)  
 y2 = y2 + 1 / 6 \* (k1\_2 + 2 \* k2\_2 + 2 \* k3\_2 + k4\_2)  
 x += h  
  
 return x\_s, answers\_runge

**Файл diff\_equation\_three\_solver.py**

from server.logic.diffSolvers.DiffEquationSolverBase import DiffEquationSolverBase  
class DiffEquationThreeSolver(DiffEquationSolverBase):  
 def \_\_init\_\_(self, equation, interval\_beg, interval\_end, step):  
 super().\_\_init\_\_(equation, interval\_beg, interval\_end, step)  
  
 def F(self, x, y, y2, y3, n):  
 f = 0  
 if n == 1:  
 f = self.\_equation\_generator(x, y, y2, y3)  
 elif n == 2:  
 f = y2  
 elif n == 3:  
 f = y3  
  
 return f  
  
 def euler\_method(self, y, y2, y3):  
 x\_s = []  
 answers\_euler = []  
 x = self.\_interval\_beg  
 h = self.\_step  
 while x <= self.\_interval\_end + h:  
 y1\_1 = y + h \* self.F(x, y, y2, y3, 1)  
 y1\_2 = y2 + h \* self.F(x, y, y2, y3, 2)  
 y1\_3 = y3 + h \* self.F(x, y, y2, y3, 3)  
  
 y = y1\_1  
 y2 = y1\_2  
 y3 = y1\_3  
 x\_s.append(x)  
 answers\_euler.append(y)  
  
 x += h  
 return x\_s, answers\_euler  
  
 def runge\_third\_method(self, y, y2, y3):  
 x\_s = []  
 answers\_runge = []  
 x = self.\_interval\_beg  
 h = self.\_step  
 while x <= self.\_interval\_end + h:  
 k1\_1 = h \* self.F(x, y, y2, y3, 1)  
 k1\_2 = h \* self.F(x, y, y2, y3, 2)  
 k1\_3 = h \* self.F(x, y, y2, y3, 3)  
  
 k2\_1 = h \* self.F(x + h / 2, y + k1\_1 / 2,  
 y2 + k1\_2 / 2, y3 + k1\_3 / 2, 1)  
 k2\_2 = h \* self.F(x + h / 2, y + k1\_1 / 2, y2 + k1\_2 / 2,  
 y3 + k1\_3 / 2, 2)  
  
 k2\_3 = h \* self.F(x + h / 2, y + k1\_1 / 2, y2 + k1\_2 / 2,  
 y3 + k1\_3 / 2, 3)  
  
 k3\_1 = h \* self.F(x + h, y + k2\_1, y2 + k2\_2,  
 y3 + k2\_3, 1)  
 k3\_2 = h \* self.F(x + h, y + k2\_1, y2 + k2\_2,  
 y3 + k2\_3, 2)  
  
 k3\_3 = h \* self.F(x + h, y + k2\_1, y2 + k2\_2,  
 y3 + k2\_3, 3)  
  
 answers\_runge.append(y)  
 x\_s.append(x)  
 y = y + 1 / 4 \* (k1\_1 + 2 \* k2\_1 + k3\_1)  
 y2 = y2 + 1 / 4 \* (k1\_2 + 2 \* k2\_2 + k3\_2)  
 y3 = y3 + 1 / 4 \* (k1\_3 + 2 \* k2\_3 + k3\_3)  
 x += h  
  
 return x\_s, answers\_runge  
  
 def runge\_forth\_method(self, y, y2, y3):  
 x\_s = []  
 answers\_runge = []  
 x = self.\_interval\_beg  
 h = self.\_step  
 while x <= self.\_interval\_end + h:  
 k1\_1 = h \* self.F(x, y, y2, y3, 1)  
 k1\_2 = h \* self.F(x, y, y2, y3, 2)  
 k1\_3 = h \* self.F(x, y, y2, y3, 3)  
  
 k2\_1 = h \* self.F(x + h / 2, y + k1\_1 / 2, y2 + k1\_2 / 2,  
 y3 + k1\_3 / 2, 1)  
 k2\_2 = h \* self.F(x + h / 2, y + k1\_1 / 2, y2 + k1\_2 / 2,  
 y3 + k1\_3 / 2, 2)  
  
 k2\_3 = h \* self.F(x + h / 2, y + k1\_1 / 2, y2 + k1\_2 / 2,  
 y3 + k1\_3 / 2, 3)  
  
 k3\_1 = h \* self.F(x + h / 2, y + k2\_1 / 2, y2 + k2\_2 / 2,  
 y3 + k2\_3 / 2, 1)  
 k3\_2 = h \* self.F(x + h / 2, y + k2\_1 / 2, y2 + k2\_2 / 2,  
 y3 + k2\_3 / 2, 2)  
 k3\_3 = h \* self.F(x + h / 2, y + k2\_1 / 2, y2 + k2\_2 / 2,  
 y3 + k2\_3 / 2, 3)  
  
 k4\_1 = h \* self.F(x + h, y + k3\_1, y2 + k3\_2,  
 y3 + k3\_3, 1)  
 k4\_2 = h \* self.F(x + h, y + k3\_1, y2 + k3\_2,  
 y3 + k3\_3, 2)  
 k4\_3 = h \* self.F(x + h, y + k3\_1, y2 + k3\_2,  
 y3 + k3\_3, 3)  
  
 answers\_runge.append(y)  
 x\_s.append(x)  
 y = y + 1 / 6 \* (k1\_1 + 2 \* k2\_1 + 2 \* k3\_1 + k4\_1)  
 y2 = y2 + 1 / 6 \* (k1\_2 + 2 \* k2\_2 + 2 \* k3\_2 + k4\_2)  
 y3 = y3 + 1 / 6 \* (k1\_3 + 2 \* k2\_3 + 2 \* k3\_3 + k4\_3)  
 x += h  
  
 return x\_s, answers\_runge

**Файл diff\_equation\_four\_solver.py**

from server.logic.diffSolvers.DiffEquationSolverBase import DiffEquationSolverBase  
  
  
class DiffEquationFourSolver(DiffEquationSolverBase):  
 def \_\_init\_\_(self, equation, interval\_beg, interval\_end, step):  
 super().\_\_init\_\_(equation, interval\_beg, interval\_end, step)  
  
 def F(self, x, y, y2, y3, y4, n):  
 f = 0  
 if n == 1:  
 f = self.\_equation\_generator(x, y, y2, y3, y4)  
 elif n == 2:  
 f = y2  
 elif n == 3:  
 f = y3  
 elif n == 4:  
 f = y4  
  
 return f  
  
 def euler\_method(self, y, y2, y3, y4):  
 x\_s = []  
 answers\_euler = []  
 x = self.\_interval\_beg  
 h = self.\_step  
 while x <= self.\_interval\_end + h:  
 y1\_1 = y + h \* self.F(x, y, y2, y3, y4, 1)  
 y1\_2 = y2 + h \* self.F(x, y, y2, y3, y4, 2)  
 y1\_3 = y3 + h \* self.F(x, y, y2, y3, y4, 3)  
 y1\_4 = y4 + h \* self.F(x, y, y2, y3, y4, 4)  
  
 y = y1\_1  
 y2 = y1\_2  
 y3 = y1\_3  
 y4 = y1\_4  
 x\_s.append(x)  
 answers\_euler.append(y)  
  
 x += h  
 return x\_s, answers\_euler  
  
 def runge\_third\_method(self, y1, y2, y3, y4):  
 x\_s = []  
 answers\_runge = []  
 x = self.\_interval\_beg  
 h = self.\_step  
 while x <= self.\_interval\_end + h:  
 k1\_1 = h \* self.F(x, y1, y2, y3, y4, 1)  
 k1\_2 = h \* self.F(x, y1, y2, y3, y4, 2)  
 k1\_3 = h \* self.F(x, y1, y2, y3, y4, 3)  
 k1\_4 = h \* self.F(x, y1, y2, y3, y4, 4)  
  
 k2\_1 = h \* self.F(x + h / 2, y1 + k1\_1 / 2,  
 y2 + k1\_2 / 2, y3 + k1\_3 / 2,  
 y4 + k1\_4 / 2, 1)  
 k2\_2 = h \* self.F(x + h / 2, y1 + k1\_1 / 2,  
 y2 + k1\_2 / 2, y3 + k1\_3 / 2,  
 y4 + k1\_4 / 2, 2)  
  
 k2\_3 = h \* self.F(x + h / 2, y1 + k1\_1 / 2,  
 y2 + k1\_2 / 2, y3 + k1\_3 / 2,  
 y4 + k1\_4 / 2, 3)  
 k2\_4 = h \* self.F(x + h / 2, y1 + k1\_1 / 2,  
 y2 + k1\_2 / 2, y3 + k1\_3 / 2,  
 y4 + k1\_4 / 2, 4)  
  
 k3\_1 = h \* self.F(x + h, y1 + k2\_1, y2 + k2\_2,  
 y3 + k2\_3, y4 + k2\_4, 1)  
 k3\_2 = h \* self.F(x + h, y1 + k2\_1, y2 + k2\_2,  
 y3 + k2\_3, y4 + k2\_4, 2)  
  
 k3\_3 = h \* self.F(x + h, y1 + k2\_1, y2 + k2\_2,  
 y3 + k2\_3, y4 + k2\_4, 3)  
  
 k3\_4 = h \* self.F(x + h, y1 + k2\_1, y2 + k2\_2,  
 y3 + k2\_3, y4 + k2\_4, 4)  
  
 answers\_runge.append(y1)  
 x\_s.append(x)  
 y1 = y1 + 1 / 4 \* (k1\_1 + 2 \* k2\_1 + k3\_1)  
 y2 = y2 + 1 / 4 \* (k1\_2 + 2 \* k2\_2 + k3\_2)  
 y3 = y3 + 1 / 4 \* (k1\_3 + 2 \* k2\_3 + k3\_3)  
 y4 = y4 + 1 / 4 \* (k1\_4 + 2 \* k2\_4 + k3\_4)  
 x += h  
  
 return x\_s, answers\_runge  
  
 def runge\_forth\_method(self, y1, y2, y3, y4):  
 x\_s = []  
 answers\_runge = []  
 x = self.\_interval\_beg  
 h = self.\_step  
 while x <= self.\_interval\_end + h:  
 k1\_1 = h \* self.F(x, y1, y2, y3, y4, 1)  
 k1\_2 = h \* self.F(x, y1, y2, y3, y4, 2)  
 k1\_3 = h \* self.F(x, y1, y2, y3, y4, 3)  
 k1\_4 = h \* self.F(x, y1, y2, y3, y4, 4)  
  
 k2\_1 = h \* self.F(x + h / 2, y1 + k1\_1 / 2, y2 + k1\_2 / 2,  
 y3 + k1\_3 / 2, y3 + k1\_4, 1)  
 k2\_2 = h \* self.F(x + h / 2, y1 + k1\_1 / 2, y2 + k1\_2 / 2,  
 y3 + k1\_3 / 2, y3 + k1\_4, 2)  
 k2\_3 = h \* self.F(x + h / 2, y1 + k1\_1 / 2, y2 + k1\_2 / 2,  
 y3 + k1\_3 / 2, y3 + k1\_4, 3)  
 k2\_4 = h \* self.F(x + h / 2, y1 + k1\_1 / 2, y2 + k1\_2 / 2,  
 y3 + k1\_3 / 2, y3 + k1\_4, 4)  
  
 k3\_1 = h \* self.F(x + h / 2, y1 + k2\_1 / 2, y2 + k2\_2 / 2,  
 y3 + k2\_3 / 2, y4 + k2\_4 / 2, 1)  
 k3\_2 = h \* self.F(x + h / 2, y1 + k2\_1 / 2, y2 + k2\_2 / 2,  
 y3 + k2\_3 / 2, y4 + k2\_4 / 2, 2)  
 k3\_3 = h \* self.F(x + h / 2, y1 + k2\_1 / 2, y2 + k2\_2 / 2,  
 y3 + k2\_3 / 2, y4 + k2\_4 / 2, 3)  
 k3\_4 = h \* self.F(x + h / 2, y1 + k2\_1 / 2, y2 + k2\_2 / 2,  
 y3 + k2\_3 / 2, y4 + k2\_4 / 2, 4)  
  
 k4\_1 = h \* self.F(x + h, y1 + k3\_1, y2 + k3\_2,  
 y3 + k3\_3, y4 + k3\_4, 1)  
 k4\_2 = h \* self.F(x + h, y1 + k3\_1, y2 + k3\_2,  
 y3 + k3\_3, y4 + k3\_4, 2)  
 k4\_3 = h \* self.F(x + h, y1 + k3\_1, y2 + k3\_2,  
 y3 + k3\_3, y4 + k3\_4, 3)  
 k4\_4 = h \* self.F(x + h, y1 + k3\_1, y2 + k3\_2,  
 y3 + k3\_3, y4 + k3\_4, 4)  
  
 answers\_runge.append(y1)  
 x\_s.append(x)  
 y1 = y1 + 1 / 6 \* (k1\_1 + 2 \* k2\_1 + 2 \* k3\_1 + k4\_1)  
 y2 = y2 + 1 / 6 \* (k1\_2 + 2 \* k2\_2 + 2 \* k3\_2 + k4\_2)  
 y3 = y3 + 1 / 6 \* (k1\_3 + 2 \* k2\_3 + 2 \* k3\_3 + k4\_3)  
 y4 = y4 + 1 / 6 \* (k1\_4 + 2 \* k2\_4 + 2 \* k3\_4 + k4\_4)  
 x += h  
  
 return x\_s, answers\_runge