Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра АОИ

**Численные методы**

**ОТЧЕТ**

ПО РЕЗУЛЬТАТАМ

учебной практики: **ознакомительной практики**

|  |  |
| --- | --- |
|  | Обучающийся гр. \_423-3\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_ М.Д. Щилко  (подпись) (ФИО)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата) |
| **\_\_\_\_\_\_\_\_\_**  (оценка)  Томск 2024 | Руководитель практики от университета:  ст. преподаватель каф. АОИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.А. Петкун  (подпись)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата) |

**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное   
учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра АОИ

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Зав. кафедрой АОИ  Сидоров А.А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на учебную практику: **ознакомительную практику**

студенту гр. 423-3 факультета систем управления

*Щилко Максиму Денисовичу*

1. **Тема практики:** Численные методы.
2. **Цель практики:** написать программу, визуализирующую метод дихотомии для нахождения локального экстремума функции.
3. **Задачи практики:** изучить средства для изображения графических объектов, изучить алгоритм метода дихотомии, реализовать алгоритм и визуализацию алгоритма.
4. **Сроки прохождения практики**: 02.09.2024 – 30.12.2024.

**Совместный рабочий график (план) проведения практики**

|  |  |  |
| --- | --- | --- |
| № п/п | Перечень заданий | Сроки выполнения |
| 1. | Подбор источников информации, изучение источников информации. Изучение численного метода дихотомии | 07.09.2024 – 21.09.2024 |
| 2. | Написание фрагментов отчета: содержание, введение, описание изучаемого метода. Реализация метода. Тестирование метода | 21.09.2024– 17.10.2024 |
| 3. | Изучение графических примитивов языка программирования Python и реализация графического представления метода. | 19.10.2024 – 31.10.2024 |
| 4. | Разработка интерфейса программы. Отладка программы. Демонстрация работы программы. | 02.11.2024 – 05.12.2024 |
| 5. | Написание отчета | 07.12.2024 – 19.12.2024 |

Дата выдачи: «06» сентября 2024 г.

Руководитель практики от университета

|  |  |  |
| --- | --- | --- |
| Ст. преподаватель каф. АОИ |  | Петкун Т.А. |

Согласовано:

Задание принял к исполнению «06» сентября 2024 г.

|  |  |  |
| --- | --- | --- |
| Студент гр. 423-3 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Подпись) | Щилко М. Д.  (Ф.И.О.) |

# ОГЛАВЛЕНИЕ

[**ВВЕДЕНИЕ**](#_3znysh7)**..................................................................................................................... 4**

[**1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**](#_tyjcwt)**............................................................ 5**

[1.1 Описание решаемой задачи](#_3dy6vkm)...................................................................................... 5

[1.2 Теоретические основы](#_1t3h5sf).............................................................................................. 5

[**2 ОПИСАНИЕ РЕАЛИЗУЕМОГО АЛГОРИТМА**](#_2s8eyo1)**................................................. 7**

[2.1 Математическое описание алгоритма](#_17dp8vu)..................................................................... 7

[2.2 Описание алгоритма, приближённое к программированию](#_3rdcrjn)................................. 7

[**3 ФУНКЦИОНАЛЬНАЯ СТРУКТУРА РАЗРАБОТАННОЙ ПРОГРАММЫ**](#_26in1rg) **9**

[**4 ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ ПРОГРАММЫ**](#_lnxbz9)**.................................. 11**

[4.1 Входные данные программы](#_35nkun2).................................................................................. 11

[4.2 Выходные данные программы](#_1ksv4uv)............................................................................... 11

[**5 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ**](#_2jxsxqh)**.......................................... 12**

[**ЗАКЛЮЧЕНИЕ**](#_3j2qqm3)**...........................................................................................................16**

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**](#_4i7ojhp)**..............................................17**

**ПРИЛОЖЕНИЕ...........................................................................................................18**

# ВВЕДЕНИЕ

Все мы пользуемся калькуляторами – вводим нужные значения, а программа вычисляет ответ. Но существуют и специализированные калькуляторы, например, те, что вычисляют максимум или минимум введенной функции, ограниченной на заданном отрезке. Такие программы очень полезны людям, занимающимся исследованием функций, поскольку сильно сокращают время работы и предоставляют нужную информацию в удобном формате.

Метод дихотомии используется в различных областях: математика, алгебра, теория вероятностей и математическая статистика, биология, астрономия и даже архитектура. Он применяется не только для оптимизации определения значения функции по заранее установленному критерию, но и для поиска нулевого значения монотонной функции на заданном отрезке, поиска соотношения между «несущими и несомыми» конструкциями в строительстве, разбиения и классификации подмножеств, групп и колец, и т.п.

Идея дихотомического деления впервые упоминается под именем диереза в диалогах древнегреческого мыслителя Платона (ок. 427—347 гг. до н.э.).

Деление целого на две части, не связанные между собой, называется дихотомией (от греч. διχό — надвое τοµoς — сечение).

# 1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

### 1.1 Описание решаемой задачи

Необходимо разработать программу наглядной демонстрации работы алгоритма нахождения локального минимума или максимума функции методом дихотомии. Программа должна решать следующие задачи:

* Построение заданного пользователем графика на указанном промежутке;
* Построение заданного пользователем графика на указанном промежутке;
* Визуальная демонстрация работы алгоритма пошагово и конечным результатом.

### 1.2 Теоретические основы

Основные элементы теории построения и анализа графиков, которые используются и упоминаются в практической работе:

* **График функции** — множество точек, у которых абсциссы являются допустимыми значениями аргумента, а ординаты — соответствующими значениями функции.
* **Точка экстремума** — точка локального минимума или максимума функции.
* **Точка минимума** – это значение переменной х, при которой функция минимальна.
* **Точка максимума** – это значение переменной х, при которой функция максимальна.
* **Точность** – мера того, насколько близки числа в вычислениях к истинным значениям. Более точные вычисления означают меньше ошибок и более надёжные результаты.

Алгоритм нахождения наибольшего и наименьшего значения функции с помощью метода дихотомии относится к методам прямого поиска.

Алгоритм нахождения наименьшего значения функции с помощью метода дихотомии основывается на делении заданного промежутка на 2 и поиска значения функций в этих точках. Затем, в зависимости от поиска минимума или максимума выбирается та точка, которая лучше удовлетворяет условию, промежуток меняется и продолжается работа уже с новым промежутком.

# 2 ОПИСАНИЕ РЕАЛИЗУЕМОГО АЛГОРИТМА

### 2.1 Математическое описание алгоритма

Считаем, что отделение корней произведено и на интервале [a,b] расположен один корень, который необходимо уточнить с погрешностью ε.

Итак, имеем f(a)f(b)<0. Метод дихотомии заключается в следующем. Определяем половину отрезка c=½(a+b) и вычисляем f(c). Проверяем следующие условия

1. Если |f(c)| < ε, то c – корень. Здесь ε - заданная точность.

2. Если f(c)f(a)<0, то корень лежит в интервале [a,c].

3. Если f(c)f(b)<0, то корень лежит на отрезке[c,b].

Продолжая процесс половинного деления в выбранных подынтервалов, можно дойти до сколь угодно малого отрезка, содержащего корень ξ.

Так как за каждую итерацию интервал, где расположен корень уменьшается в два раза, то через n итераций интервал будет равен:

**bn-an=1/2n(b-a)**, при этом **an≤ξ≤bn, | ξ-an|≤1/2n(b-a), | ξ-bn|≤1/2n(b-a)**

В качестве корня ξ. возьмем =½(bn+an). Тогда погрешность определения корня будет равна (bn – an)/2. Если выполняется условие (bn–an)/2<ε, то процесс поиска заканчивается и ε=½(bn+an).

### 2.2 Описание алгоритма, приближённое к программированию

**1. Инициализация:**

1.1 Создать экземпляр класса DihotomyDecision с передачей в виде аргументов строки функции, списка из двух точек – границ промежутка определения функции и точности в виде десятичной дроби 10 \*\* (-n), где n – число, указанное как точность в интерфейсе программы.

**2. Нахождение точек экстремума:**

2.1 Далее, в зависимости от целей пользователя, запускается одна из функций вычисления результата – пошаговая или полноценная. В случае пошагового вычисления при каждом запуске пошаговой функции промежуток двигается к той или иной границе, а изменения возвращаются.

2.2 Следующим шагом необходимо еще раз запустить эту функцию. В случае полноценного решения сначала вычисляется ξ, который бы позволял вычислить значение с указанной точностью, а затем запускается цикл вычисления экстремума, где с помощью математического алгоритма, указанного ранее, границы промежутка двигаются к точке с минимальным или максимальным значением функции.

**3. Условие остановки:**

3.1 Программа останавливается, каждый шаг, если пользователем выбрана такая опция, либо при условии нахождения достаточно малого промежутка, в котором находится точка экстремума функции.

### 3 ФУНКЦИОНАЛЬНАЯ СТРУКТУРА РАЗРАБОТАННОЙ ПРОГРАММЫ

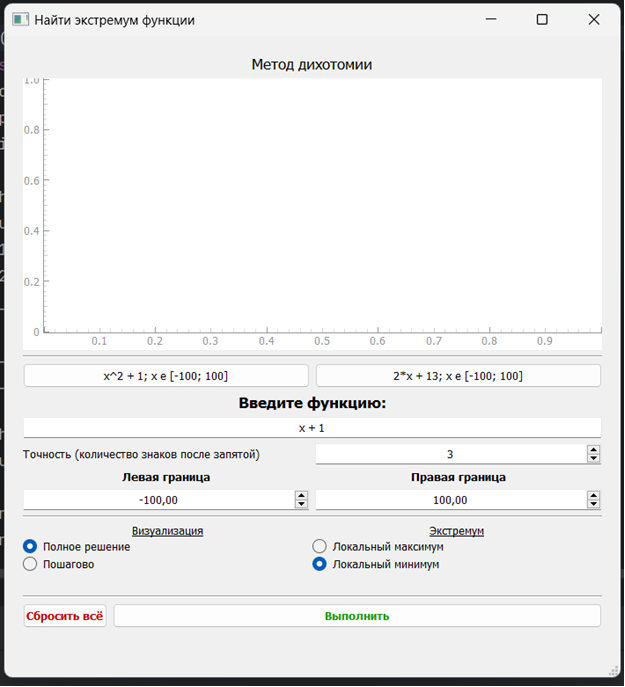


Рисунок 3.1 – Основное окно программы

**Компоненты программы:**

• Найти экстремум функции – название программы;

• Метод -дихотомии – заголовок графика;

• Белый прямоугольник с осями координат – визуализация графика функции;

• x^2 + 1; x e [-100; 100] и 2 \* x + 13; [-100; 100] – шаблоны заполнения данных;

• Введите функцию – заголовок для поля ввода текста функции;

• Поле ввода функции;

• Точность (количество знаков после запятой) – заголовок для целочисленного счётчика знаков после запятой переменной эпсилон;

• Счётчик знаков после запятой;

• Левая граница и Правая граница – заголовки для полей ввода (счётчиков) промежутка определения функции;

• Счётчики границ промежутка определения функции;

• Полное решение с задержкой в 0,00 сек -чекбокс опции полного решения;

• Пошагово – чекбокс опции решения с визуализацией пошагово;

• Локальный максимум и минимум – чекбоксы опции искомой точки;

• Сбросить всё – кнопка сброса полей и графика в исходное состояние;

• Выполнить – кнопка вычисления и визуализации метода дихотомии для указанной функции.

При запуске открывается окно программы. Пользователь указывает нужные значения в поля ввода, счётчики, выбирает желаемые опции, либо использует существующие шаблоны. Далее пользователь запускает вычисление экстремума функции нажатием на кнопку Выполнить. При выборе опции пошагового решения для визуализации следующего шага пользователю необходимо еще раз нажать на кнопку Выполнить.

# 4 ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ ПРОГРАММЫ

### 4.1 Входные данные программы

Данные, которые вводит пользователь:

* Функция;
* Точность;
* Левая граница;
* Правая граница;
* Опция визуализации;
* Опция типа экстремума.

### 4.2 Выходные данные программы

К выходным данным программы относятся:

* Визуализация работы метода дихотомии на графике
* Визуализация результата работы метода на графике
* Две точки по 0X, в пределах которых находится искомое значение с учётом заданной точности
* Две точки по 0Y, вычисленные по значениям X
* Количество итераций при вычислении точки экстремума

# 5 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ

При запуске программы открывается основное окно. В полях ввода уже заданы начальные значения, которые пользователь может изменить.

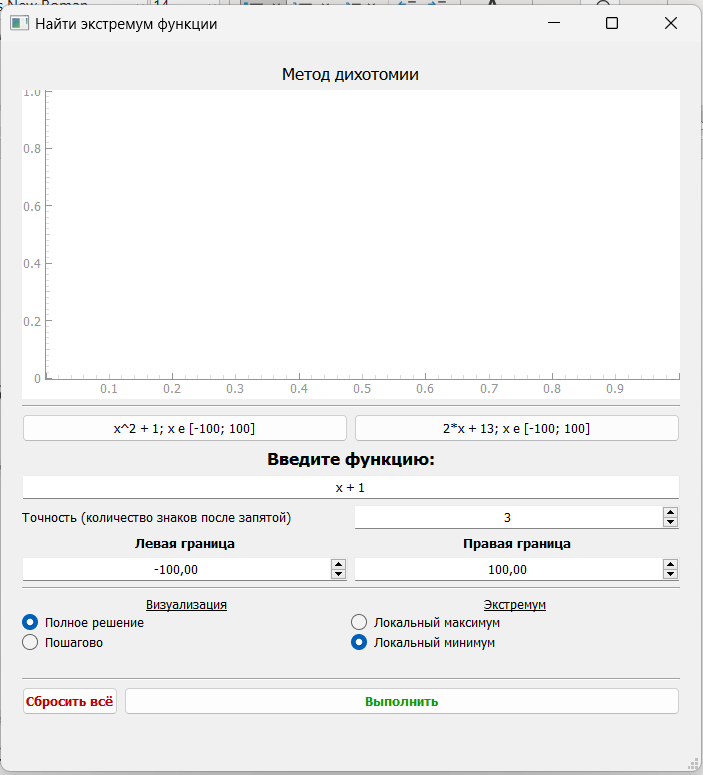


Рис 5.1. Вид приложения при запуске

Также пользователь может заполнить данные с помошью встроенных шаблонов над заголовком «Введите функцию»:

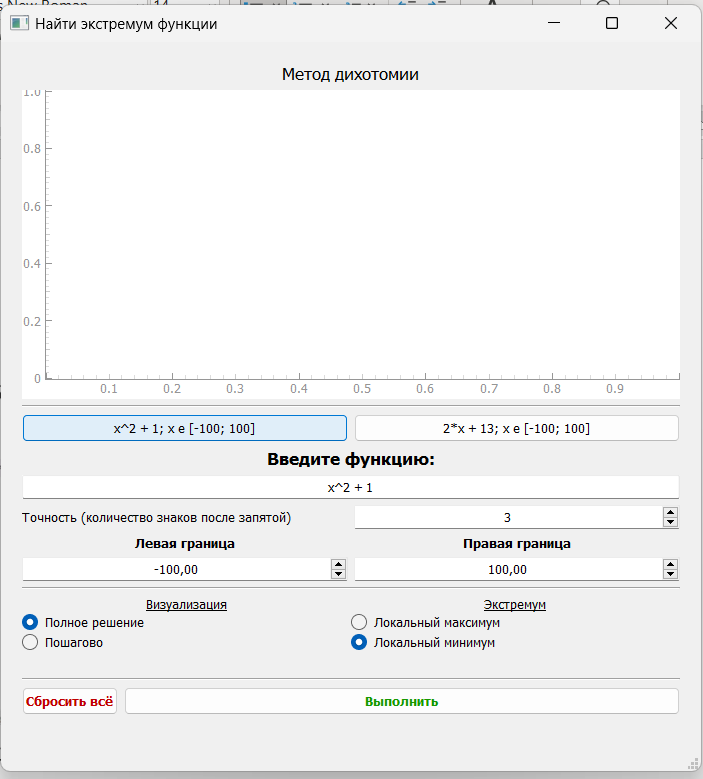


Рис 5.2. Выбран шаблон заполнения данных

Далее пользователь нажимает кнопку Выполнить и наблюдает график функции с промежутком, включающим в себя экстремум функции:

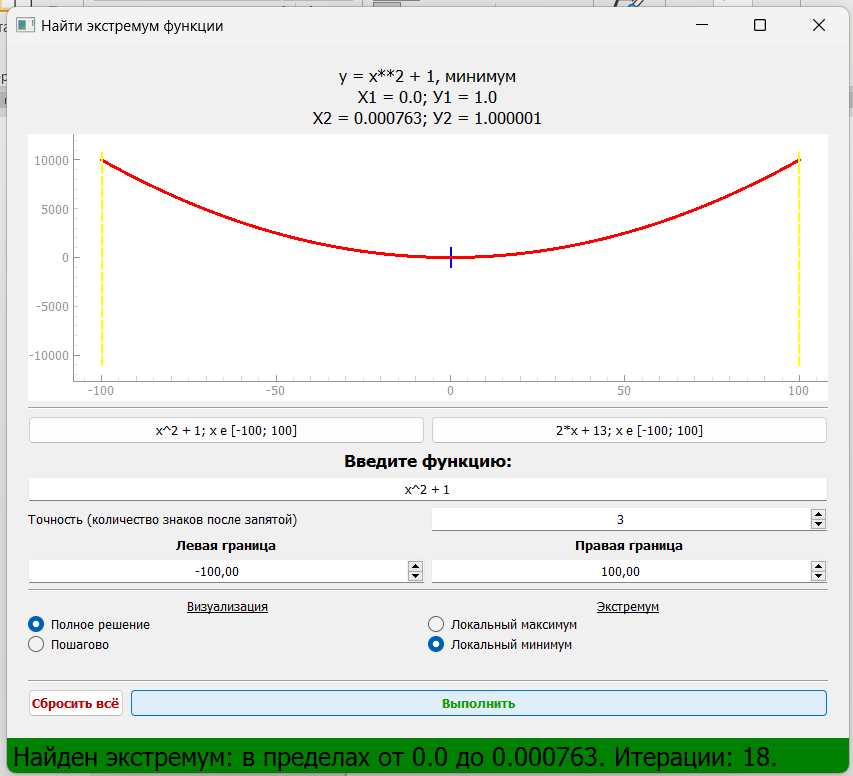


Рис 5.3. Полное нахождение экстремума - минимума

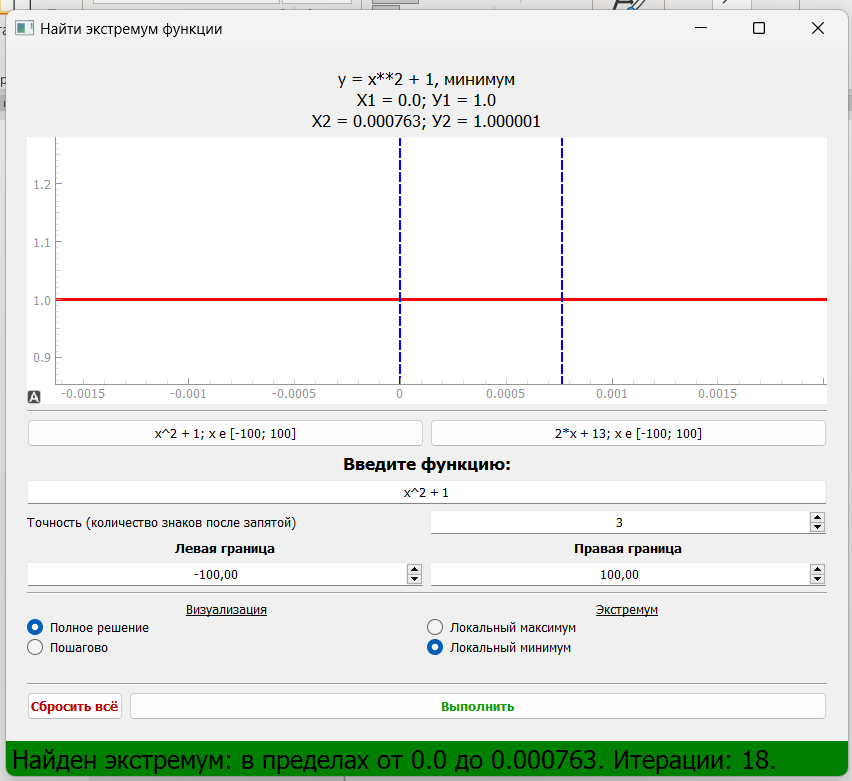


Рис 5.4. Разница между двумя границами решения

Код программы расположен в приложении.

## ЗАКЛЮЧЕНИЕ

Основной целью данной работы была реализация программы для визуализации работы метода дихотомии, которая позволяет пользователю вручную задавать нужные параметры функции и выбирать тип искомого экстремума. Программа должна была быть интуитивно понятной, универсальной и подходить для учебных и демонстрационных целей.

Программа написана на языке Python, который был выбран из-за простоты синтаксиса, значительного количества удобных библиотек для создания пользовательского интерфейса и отрисовки графика.

Написанная программа получилась удобной и максимально простой для пользования и понимания. Разработанная программа позволяет отрисовывать график по заданным параметрам на указанном промежутке, исследовать на точки экстремума с указанной точностью. Программа включает дополнительные функции для удобства пользователя: очистка данных, сброс полей ввода до начального значения, проверка и обработка некорректных данных с помощью statusbar.

В процессе учебной практики я получил знания и практические навыки в реализации программ с графическим интерфейсом, а также ознакомился с массой инструментов и методов, которые могут позволить выполнять подобные задачи лучше и эффективнее.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Доусон, М. Программируем на Python / Доусон М. – СПб.: Питер, 2014 – 416 с.
2. «PyQt6 — полное руководство для новичков». [Электронный ресурс]. Дата обращения: 17.09.2024

(<https://habr.com/ru/companies/skillfactory/articles/599599/>)

1. «Наука и жизнь – по закону дихотомии». [Электронный ресурс]. Дата обращения: 13.10.2024

(<https://m.nkj.ru/archive/articles/23134/>)

1. Кернякевич, П. С. Разработка производственных проектов [Электронный ресурс]: Учебное пособие / П. С. Кернякевич - 2018. 90 с. Дата обращения: 15.10.24

(<https://edu.tusur.ru/publications/7837?ysclid=m4bgu22w1x724446838>)

1. Вирт, Н. Алгоритмы и структуры данных (с примерами на Паскале): пер. с англ. [Электронный ресурс]: Учебное пособие / Н. Вирт; пер. Д. Б. Подшивалов. – 2-е изд., испр. – СПб.: Невский диалект, 2007. – 351 с. Дата обращения: 11.10.24

(<https://djvu.online/file/oxZRj2LL1zSNp?ysclid=m4bgyr4dkh615507556>)

**ПРИЛОЖЕНИЕ**

**Модуль main.py – основной код программы:**

from design.interface2 import Ui\_MainWindow

from PyQt5 import QtCore, QtGui, QtWidgets

from dihotomy import DihotomyDecision

import pyqtgraph as pg

class DihotomyApp(QtWidgets.QMainWindow, Ui\_MainWindow):

def \_\_init\_\_(self):

super(DihotomyApp, self).\_\_init\_\_()

self.setupUi(self)

self.setWindowTitle('Найти экстремум функции')

self.graphic.setBackground('white')

self.do\_button.clicked.connect(self.do\_button\_clicked)

self.shab1button.clicked.connect(self.shab1button\_clicked)

self.shab2button.clicked.connect(self.shab2button\_clicked)

self.drop\_all\_button.clicked.connect(self.drop\_all\_clicked)

self.func\_finder = None

self.step\_counter = 0

self.graphic\_label.setText('Метод дихотомии')

self.statusbar.setFont(QtGui.QFont('Times', 15))

self.begin\_borders = None

self.begin\_sizes = None

self.eps\_spinBox.textChanged.connect(self.setEPStext)

def do\_button\_clicked(self):

function = self.func\_line.text() # get func text

if not function: # no func exists

self.statusbar.show()

self.statusBar().setStyleSheet("background-color : red;")

self.statusbar.showMessage('Не заполнена функция!')

return

eps = 0.1 \*\* self.eps\_spinBox.value()

border = [self.left\_border\_SpinBox.value(), self.right\_border\_SpinBox.value()]

if border[0] == border[1]: # no borders

self.statusbar.show()

self.statusBar().setStyleSheet("background-color : red")

self.statusbar.showMessage('Заданы неверные границы определения!')

return

self.statusbar.hide()

to\_max = self.max\_radioButton.isChecked()

if ((self.func\_finder is None or self.func\_finder.function != function.replace(':', '/').replace('^', '\*\*')) or

(not (self.func\_finder is None) and

self.func\_finder.function != function.replace(':', '/').replace('^', '\*\*'))):

self.graphic\_label.setText('Метод дихотомии')

self.graphic.clear()

self.step\_counter = 0

self.func\_finder = DihotomyDecision(function, border, eps)

total\_graphic\_points = self.func\_finder.get\_points()

self.graphic.plot([x[0] for x in total\_graphic\_points], [y[1] for y in total\_graphic\_points],

pen=pg.mkPen(color=(255, 0, 0), width=3))

func\_resY = [sorted((-self.func\_finder.get\_func\_res(self.func\_finder.border[i])[1],

self.func\_finder.get\_func\_res(self.func\_finder.border[i])[1]))

for i in range(2)]

for i, yy in enumerate(func\_resY):

yys = [y for y in range(int(yy[0] - 1000), int(yy[1] + 1000), 5)]

self.graphic.plot([self.func\_finder.border[i] for \_ in range(len(yys))], yys,

pen=pg.mkPen(color=(255, 255, 0), width=2, style=QtCore.Qt.DashLine))

self.begin\_borders = self.func\_finder.border

if self.full\_decision\_radio.isChecked():

self.draw\_fully(to\_max)

elif self.step\_decision\_radio.isChecked():

self.draw\_by\_step(to\_max)

def draw\_fully(self, to\_max):

new\_graphic\_borders = self.func\_finder.find\_extremum(to\_max)[0]

func\_resY = [sorted((-self.func\_finder.get\_func\_res(new\_graphic\_borders[i])[1],

self.func\_finder.get\_func\_res(new\_graphic\_borders[i])[1]))

for i in range(2)]

for i, yy in enumerate(func\_resY):

yys = [y for y in range(int(yy[0] - 1000), int(yy[1] + 1000))]

self.graphic.plot([new\_graphic\_borders[i] for \_ in range(len(yys))], yys,

pen=pg.mkPen(color=(0, 0, 255), width=2, style=QtCore.Qt.DashLine))

self.statusBar().setStyleSheet("background-color : green")

self.statusbar.show()

self.statusbar.showMessage(

f'Найден экстремум: в пределах от {min(self.func\_finder.border)} до {max(self.func\_finder.border)}. '

f'Итерации: {self.func\_finder.counter}.')

print(self.func\_finder.rounder)

self.graphic\_label.setText(f'y = {self.func\_finder.function}, '

f'{"минимум" if self.min\_radioButton.isChecked() else "максимум"}\n'

f'Х1 = {self.func\_finder.border[0]}; '

f'У1 = {round(self.func\_finder.get\_func\_res(self.func\_finder.border[0])[1], self.func\_finder.rounder)}\n'

f'Х2 = {self.func\_finder.border[1]}; '

f'У2 = {round(self.func\_finder.get\_func\_res(self.func\_finder.border[1])[1], self.func\_finder.rounder)}')

def draw\_by\_step(self, to\_max):

is\_answer = self.func\_finder.find\_by\_step(to\_max)

func\_resY = [sorted((-self.func\_finder.get\_func\_res(self.func\_finder.border[i])[1],

self.func\_finder.get\_func\_res(self.func\_finder.border[i])[1]))

for i in range(2)]

for i, yy in enumerate(func\_resY):

yys = [y for y in range(int(yy[0] - 1000), int(yy[1] + 1000), 5)]

print(self.begin\_borders, func\_resY)

if not is\_answer:

if self.func\_finder.border[i] == self.begin\_borders[i]:

continue

self.graphic.plot([self.func\_finder.border[i] for \_ in range(len(yys))], yys,

pen=pg.mkPen(color=(0, 0, 255), width=2, style=QtCore.Qt.DashLine))

else:

self.graphic.plot([self.func\_finder.border[i] for \_ in range(len(yys))], yys,

pen=pg.mkPen(color=(0, 255, 0), width=2, style=QtCore.Qt.DashLine))

if is\_answer:

self.statusBar().setStyleSheet("background-color : green")

self.statusbar.show()

self.statusbar.showMessage(

f'Найден экстремум: в пределах от {min(self.func\_finder.border)} до {max(self.func\_finder.border)}. '

f'Итерации: {self.func\_finder.counter}.')

else:

self.statusBar().setStyleSheet("background-color : green")

self.statusbar.show()

self.statusbar.showMessage(

f'Итерация: {self.func\_finder.counter}')

def shab1button\_clicked(self):

self.func\_line.setText(self.shab1button.text().split(';')[0])

self.left\_border\_SpinBox.setValue(float(self.shab1button.text().split(';')[1].split('[')[1]))

self.right\_border\_SpinBox.setValue(float(self.shab1button.text().split(';')[2].split(']')[0]))

def shab2button\_clicked(self):

self.func\_line.setText(self.shab2button.text().split(';')[0])

self.left\_border\_SpinBox.setValue(float(self.shab2button.text().split(';')[1].split('[')[1]))

self.right\_border\_SpinBox.setValue(float(self.shab2button.text().split(';')[2].split(']')[0]))

def drop\_all\_clicked(self):

self.graphic\_label.setText('Метод дихотомии')

self.graphic.clear()

self.func\_finder = None

self.step\_counter = 0

self.func\_line.setText('x + 1')

self.eps\_spinBox.setValue(3)

self.left\_border\_SpinBox.setValue(-100)

self.right\_border\_SpinBox.setValue(100)

self.full\_decision\_radio.toggle()

self.min\_radioButton.toggle()

def setEPStext(self):

self.eps\_label.setText(f"Точность: 0.{'0' \* (self.eps\_spinBox.value() - 1)}1")

if \_\_name\_\_ == '\_\_main\_\_':

import sys

app = QtWidgets.QApplication(sys.argv)

form = DihotomyApp()

form.show()

sys.exit(app.exec\_())

**Модуль interface2.py – содержит описание интерфейса:**

import pyqtgraph as pg

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_MainWindow(object):

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.resize(700, 729)

MainWindow.setMinimumSize(QtCore.QSize(600, 600))

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.horizontalLayout\_6 = QtWidgets.QHBoxLayout(self.centralwidget)

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

self.verticalLayout = QtWidgets.QVBoxLayout()

self.verticalLayout.setContentsMargins(10, 10, 10, 10)

self.verticalLayout.setSpacing(6)

self.verticalLayout.setObjectName("verticalLayout")

self.graphic\_label = QtWidgets.QLabel(self.centralwidget)

font = QtGui.QFont()

font.setPointSize(10)

self.graphic\_label.setFont(font)

self.graphic\_label.setAlignment(QtCore.Qt.AlignCenter)

self.graphic\_label.setObjectName("graphic\_label")

self.verticalLayout.addWidget(self.graphic\_label)

self.graphic = pg.PlotWidget(self.centralwidget)

self.graphic.setObjectName("graphic")

self.verticalLayout.addWidget(self.graphic)

self.line = QtWidgets.QFrame(self.centralwidget)

self.line.setFrameShape(QtWidgets.QFrame.HLine)

self.line.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line.setObjectName("line")

self.verticalLayout.addWidget(self.line)

self.horizontalLayout\_8 = QtWidgets.QHBoxLayout()

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

self.shab1button = QtWidgets.QPushButton(self.centralwidget)

self.shab1button.setObjectName("shab1button")

self.horizontalLayout\_8.addWidget(self.shab1button)

self.shab2button = QtWidgets.QPushButton(self.centralwidget)

self.shab2button.setObjectName("shab2button")

self.horizontalLayout\_8.addWidget(self.shab2button)

self.verticalLayout.addLayout(self.horizontalLayout\_8)

self.func\_label = QtWidgets.QLabel(self.centralwidget)

font = QtGui.QFont()

font.setPointSize(10)

font.setBold(True)

font.setWeight(75)

self.func\_label.setFont(font)

self.func\_label.setAlignment(QtCore.Qt.AlignCenter)

self.func\_label.setObjectName("func\_label")

self.verticalLayout.addWidget(self.func\_label)

self.func\_line = QtWidgets.QLineEdit(self.centralwidget)

self.func\_line.setAlignment(QtCore.Qt.AlignCenter)

self.func\_line.setObjectName("func\_line")

self.verticalLayout.addWidget(self.func\_line)

self.horizontalLayout = QtWidgets.QHBoxLayout()

self.horizontalLayout.setObjectName("horizontalLayout")

self.eps\_label = QtWidgets.QLabel(self.centralwidget)

font = QtGui.QFont()

font.setPointSize(8)

self.eps\_label.setFont(font)

self.eps\_label.setObjectName("eps\_label")

self.horizontalLayout.addWidget(self.eps\_label)

self.eps\_spinBox = QtWidgets.QSpinBox(self.centralwidget)

font = QtGui.QFont()

font.setPointSize(8)

self.eps\_spinBox.setFont(font)

self.eps\_spinBox.setAlignment(QtCore.Qt.AlignCenter)

self.eps\_spinBox.setMinimum(1)

self.eps\_spinBox.setMaximum(15)

self.eps\_spinBox.setProperty("value", 3)

self.eps\_spinBox.setObjectName("eps\_spinBox")

self.horizontalLayout.addWidget(self.eps\_spinBox)

self.verticalLayout.addLayout(self.horizontalLayout)

self.horizontalLayout\_2 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_2.setContentsMargins(-1, -1, 0, -1)

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

self.verticalLayout\_2 = QtWidgets.QVBoxLayout()

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

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

font = QtGui.QFont()

font.setBold(True)

font.setWeight(75)

self.label\_3.setFont(font)

self.label\_3.setAlignment(QtCore.Qt.AlignCenter)

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

self.verticalLayout\_2.addWidget(self.label\_3)

self.left\_border\_SpinBox = QtWidgets.QDoubleSpinBox(self.centralwidget)

self.left\_border\_SpinBox.setAlignment(QtCore.Qt.AlignCenter)

self.left\_border\_SpinBox.setMinimum(-99999.0)

self.left\_border\_SpinBox.setMaximum(99999.0)

self.left\_border\_SpinBox.setProperty("value", -100.0)

self.left\_border\_SpinBox.setObjectName("left\_border\_SpinBox")

self.verticalLayout\_2.addWidget(self.left\_border\_SpinBox)

self.horizontalLayout\_2.addLayout(self.verticalLayout\_2)

self.verticalLayout\_3 = QtWidgets.QVBoxLayout()

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

self.label\_4 = QtWidgets.QLabel(self.centralwidget)

font = QtGui.QFont()

font.setBold(True)

font.setWeight(75)

self.label\_4.setFont(font)

self.label\_4.setAlignment(QtCore.Qt.AlignCenter)

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

self.verticalLayout\_3.addWidget(self.label\_4)

self.right\_border\_SpinBox = QtWidgets.QDoubleSpinBox(self.centralwidget)

self.right\_border\_SpinBox.setAlignment(QtCore.Qt.AlignCenter)

self.right\_border\_SpinBox.setMinimum(-99999.0)

self.right\_border\_SpinBox.setMaximum(99999.0)

self.right\_border\_SpinBox.setProperty("value", 100.0)

self.right\_border\_SpinBox.setObjectName("right\_border\_SpinBox")

self.verticalLayout\_3.addWidget(self.right\_border\_SpinBox)

self.horizontalLayout\_2.addLayout(self.verticalLayout\_3)

self.verticalLayout.addLayout(self.horizontalLayout\_2)

self.line\_2 = QtWidgets.QFrame(self.centralwidget)

self.line\_2.setFrameShape(QtWidgets.QFrame.HLine)

self.line\_2.setFrameShadow(QtWidgets.QFrame.Sunken)

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

self.verticalLayout.addWidget(self.line\_2)

self.horizontalLayout\_4 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_4.setContentsMargins(-1, -1, 0, -1)

self.horizontalLayout\_4.setSpacing(0)

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

self.verticalLayout\_4 = QtWidgets.QVBoxLayout()

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

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

font = QtGui.QFont()

font.setUnderline(True)

self.label.setFont(font)

self.label.setAlignment(QtCore.Qt.AlignCenter)

self.label.setObjectName("label")

self.verticalLayout\_4.addWidget(self.label)

self.full\_decision\_radio = QtWidgets.QRadioButton(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Maximum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.full\_decision\_radio.sizePolicy().hasHeightForWidth())

self.full\_decision\_radio.setSizePolicy(sizePolicy)

self.full\_decision\_radio.setChecked(True)

self.full\_decision\_radio.setAutoExclusive(True)

self.full\_decision\_radio.setObjectName("full\_decision\_radio")

self.buttonGroup\_2 = QtWidgets.QButtonGroup(MainWindow)

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

self.buttonGroup\_2.addButton(self.full\_decision\_radio)

self.verticalLayout\_4.addWidget(self.full\_decision\_radio)

self.step\_decision\_radio = QtWidgets.QRadioButton(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Maximum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.step\_decision\_radio.sizePolicy().hasHeightForWidth())

self.step\_decision\_radio.setSizePolicy(sizePolicy)

self.step\_decision\_radio.setChecked(False)

self.step\_decision\_radio.setAutoExclusive(True)

self.step\_decision\_radio.setObjectName("step\_decision\_radio")

self.buttonGroup\_2.addButton(self.step\_decision\_radio)

self.verticalLayout\_4.addWidget(self.step\_decision\_radio)

self.horizontalLayout\_4.addLayout(self.verticalLayout\_4)

self.verticalLayout\_5 = QtWidgets.QVBoxLayout()

self.verticalLayout\_5.setContentsMargins(-1, -1, -1, 0)

self.verticalLayout\_5.setSpacing(0)

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

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

font = QtGui.QFont()

font.setUnderline(True)

self.label\_2.setFont(font)

self.label\_2.setAlignment(QtCore.Qt.AlignHCenter | QtCore.Qt.AlignTop)

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

self.verticalLayout\_5.addWidget(self.label\_2)

self.max\_radioButton = QtWidgets.QRadioButton(self.centralwidget)

self.max\_radioButton.setAutoExclusive(True)

self.max\_radioButton.setObjectName("max\_radioButton")

self.buttonGroup = QtWidgets.QButtonGroup(MainWindow)

self.buttonGroup.setObjectName("buttonGroup")

self.buttonGroup.addButton(self.max\_radioButton)

self.verticalLayout\_5.addWidget(self.max\_radioButton)

self.min\_radioButton = QtWidgets.QRadioButton(self.centralwidget)

self.min\_radioButton.setChecked(True)

self.min\_radioButton.setAutoExclusive(True)

self.min\_radioButton.setObjectName("min\_radioButton")

self.buttonGroup.addButton(self.min\_radioButton)

self.verticalLayout\_5.addWidget(self.min\_radioButton)

self.horizontalLayout\_4.addLayout(self.verticalLayout\_5)

self.verticalLayout.addLayout(self.horizontalLayout\_4)

spacerItem = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.verticalLayout.addItem(spacerItem)

self.line\_3 = QtWidgets.QFrame(self.centralwidget)

self.line\_3.setFrameShape(QtWidgets.QFrame.HLine)

self.line\_3.setFrameShadow(QtWidgets.QFrame.Sunken)

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

self.verticalLayout.addWidget(self.line\_3)

self.horizontalLayout\_3 = QtWidgets.QHBoxLayout()

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

self.drop\_all\_button = QtWidgets.QPushButton(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Maximum, QtWidgets.QSizePolicy.Fixed)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.drop\_all\_button.sizePolicy().hasHeightForWidth())

self.drop\_all\_button.setSizePolicy(sizePolicy)

palette = QtGui.QPalette()

brush = QtGui.QBrush(QtGui.QColor(191, 0, 0))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)

brush = QtGui.QBrush(QtGui.QColor(191, 0, 0))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)

brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)

self.drop\_all\_button.setPalette(palette)

font = QtGui.QFont()

font.setFamily("MS Shell Dlg 2")

font.setPointSize(8)

font.setBold(True)

font.setWeight(75)

self.drop\_all\_button.setFont(font)

self.drop\_all\_button.setObjectName("drop\_all\_button")

self.horizontalLayout\_3.addWidget(self.drop\_all\_button)

self.do\_button = QtWidgets.QPushButton(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Fixed)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.do\_button.sizePolicy().hasHeightForWidth())

self.do\_button.setSizePolicy(sizePolicy)

palette = QtGui.QPalette()

brush = QtGui.QBrush(QtGui.QColor(223, 234, 240))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)

brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)

brush = QtGui.QBrush(QtGui.QColor(23, 153, 0))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)

brush = QtGui.QBrush(QtGui.QColor(0, 0, 0, 128))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.PlaceholderText, brush)

brush = QtGui.QBrush(QtGui.QColor(223, 234, 240))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)

brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)

brush = QtGui.QBrush(QtGui.QColor(23, 153, 0))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)

brush = QtGui.QBrush(QtGui.QColor(0, 0, 0, 128))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.PlaceholderText, brush)

brush = QtGui.QBrush(QtGui.QColor(223, 234, 240))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)

brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)

brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)

brush = QtGui.QBrush(QtGui.QColor(0, 0, 0, 128))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.PlaceholderText, brush)

self.do\_button.setPalette(palette)

font = QtGui.QFont()

font.setFamily("MS Shell Dlg 2")

font.setPointSize(8)

font.setBold(True)

font.setWeight(75)

self.do\_button.setFont(font)

self.do\_button.setObjectName("do\_button")

self.horizontalLayout\_3.addWidget(self.do\_button)

self.verticalLayout.addLayout(self.horizontalLayout\_3)

self.horizontalLayout\_6.addLayout(self.verticalLayout)

MainWindow.setCentralWidget(self.centralwidget)

self.statusbar = QtWidgets.QStatusBar(MainWindow)

self.statusbar.setObjectName("statusbar")

MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

MainWindow.setTabOrder(self.func\_line, self.eps\_spinBox)

MainWindow.setTabOrder(self.eps\_spinBox, self.left\_border\_SpinBox)

MainWindow.setTabOrder(self.left\_border\_SpinBox, self.right\_border\_SpinBox)

MainWindow.setTabOrder(self.right\_border\_SpinBox, self.full\_decision\_radio)

MainWindow.setTabOrder(self.full\_decision\_radio, self.step\_decision\_radio)

MainWindow.setTabOrder(self.step\_decision\_radio, self.do\_button)

MainWindow.setTabOrder(self.do\_button, self.drop\_all\_button)

MainWindow.setTabOrder(self.drop\_all\_button, self.graphic)

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate("MainWindow", "Нахождение локального экстремума методом дихотомии"))

self.graphic\_label.setText(\_translate("MainWindow", "График функции"))

self.shab1button.setText(\_translate("MainWindow", "x^2 + 1; x e [-100; 100]"))

self.shab2button.setText(\_translate("MainWindow", "2\*x + 13; x e [-100; 100]"))

self.func\_label.setText(\_translate("MainWindow", "Введите функцию:"))

self.func\_line.setText(\_translate("MainWindow", "x + 1"))

self.eps\_label.setText(\_translate("MainWindow", "Точность (количество знаков после запятой)"))

self.label\_3.setText(\_translate("MainWindow", "Левая граница"))

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

self.label.setText(\_translate("MainWindow", "Визуализация"))

self.full\_decision\_radio.setText(\_translate("MainWindow", "Полное решение"))

self.step\_decision\_radio.setText(\_translate("MainWindow", "Пошагово"))

self.label\_2.setText(\_translate("MainWindow", "Экстремум"))

self.max\_radioButton.setText(\_translate("MainWindow", "Локальный максимум"))

self.min\_radioButton.setText(\_translate("MainWindow", "Локальный минимум"))

self.drop\_all\_button.setText(\_translate("MainWindow", "Сбросить всё"))

self.do\_button.setText(\_translate("MainWindow", "Выполнить"))

**Модуль dihotomy.py – содержит вычислительную часть программы:**

from time import sleep

from math import sin, cos, tan

import numpy as np

class DihotomyDecision:

def \_\_init\_\_(self, function: str, border: list, eps: float):

self.function = function

self.border = [min(border), max(border)]

self.eps = eps

self.rounder = 3

self.function = self.function.replace(':', '/').replace('^', '\*\*')

self.counter = 0

def get\_func\_res(self, x):

try:

return x, eval(self.function.replace('x', f'({x})'))

except ZeroDivisionError:

return x, 0

def move\_border(self, to\_max=False):

self.border = [min(self.border), max(self.border)]

to\_max = 1 if not to\_max else -1

c = round((self.border[0] + self.border[1]) / 2, self.rounder)

x1 = c - self.eps

x2 = c + self.eps

f1 = round(self.get\_func\_res(x1)[1], self.rounder)

f2 = round(self.get\_func\_res(x2)[1], self.rounder)

if to\_max \* f1 < to\_max \* f2:

self.border[1] = c

else:

self.border[0] = c

return c, x1, x2, f1, f2, self.rounder, self.border

def find\_extremum(self, to\_max=False):

self.set\_rounder()

while abs(self.border[1] - self.border[0]) >= self.eps:

self.move\_border(to\_max)

self.counter += 1

return self.border, self.eps

def find\_by\_step(self, to\_max=False):

self.set\_rounder()

if abs(self.border[1] - self.border[0]) >= self.eps:

self.move\_border(to\_max)

self.counter += 1

return False

else:

return True

def set\_rounder(self, count=0):

if count != 0:

self.rounder = count

return

temp\_eps = self.eps

if self.rounder > 3:

return

while temp\_eps < 1:

temp\_eps \*= 10

self.rounder += 1

def get\_points(self):

if self.rounder == 1:

self.set\_rounder()

points = np.array(list(map(lambda x: self.get\_func\_res(x / 10 \*\* self.rounder),

range(int(min(self.border) \* 10 \*\* self.rounder),

int(max(self.border) \* 10 \*\* self.rounder) + 1, 10 \*\* abs(self.rounder - 2)))))

return points

if \_\_name\_\_ == '\_\_main\_\_':

print("Введите функцию: ", end='')

test = DihotomyDecision(input(), [-3, -1], 0.001)

print(test.get\_points())

print(test.find\_extremum(True))