Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новосибирский государственный технический университет»

Кафедра Автоматизированных электроэнергетических систем

**Отчет ПО ПРАКТИКЕ**

Учебная практика: практика по получению первичных навыков работы с программным обеспечением

(наименование практики в соответствии с учебным планом)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направление подготовки: **\_\_\_**«Электроэнергетика и электротехника»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Выполнил:  Студент Кутбиддинов Т. Р.  (Ф.И.О.)  Группа ЭН2-31  Факультет ФЭН.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись  «01» 06 2024 г. | Проверил:  Руководитель от НГТУ \_\_Дулов И.В.\_\_\_\_\_\_\_  (Ф.И.О.)  Балл: \_\_\_\_\_\_\_\_\_\_, ECTS\_\_\_\_\_\_\_\_\_\_\_\_\_,  Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «отлично», «хорошо», «удовлетворительно», «неуд.»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

Новосибирск 2024

Оглавление

[1. Цель работы 3](#_Toc167891375)

[2. Задание 3](#_Toc167891376)

[3. Описание исходных данных 3](#_Toc167891377)

[4. Описание языка программирования Python 4](#_Toc167891378)

[5. Пояснения к заданию 5](#_Toc167891379)

[5.1. Математическая модель 5](#_Toc167891380)

[5.2. Численное интегрирование при помощи встроенной функции 7](#_Toc167891381)

[5.3. Численное интегрирование при помощи метода левых прямоугольников, реализованным самостоятельно 9](#_Toc167891382)

[5.4. Алгоритм решения задачи в графическом виде 11](#_Toc167891383)

[5.5. Разработка интерфейса 12](#_Toc167891384)

[5.6. Тестирование программы 20](#_Toc167891385)

[6. Выводы 20](#_Toc167891386)

[7. Список используемых источников 21](#_Toc167891387)

# 1. Цель работы

Получение навыков алгоритмизации, разработки и тестирования программ на языке Python на примере задачи из области электроэнергетики.

# 2. Задание

1. Ознакомиться с поставленной задачей.
2. Выбрать подходящую среду разработки.
3. Определить библиотеки для решения задачи.
4. Составить математическую модель решения задачи с описанием используемых библиотек и функций. Каждый вариант должен быть решён двумя способами:
   1. Численное интегрирование при помощи встроенной функции.
   2. Численное интегрирование при помощи метода левых/правых прямоугольников, реализованным самостоятельно.
5. Составить алгоритм решения задачи в графическом виде.
6. Разработать интерфейс программы, содержащий следующий набор элементов:
   1. Поля ввода, позволяющие вводить исходные данные.
   2. Выпадающий список для выбора способа решения (реализованная или встроенная функции).
   3. Кнопка для запуска программы.
   4. Текст с результатами расчёта.
   5. Графики с необходимой графической информацией.
7. Выполнить реализацию составленного алгоритма в выбранной среде разработки.
8. Провести тестирование программы.

# 3. Описание исходных данных

Проводник в поперечном сечении имеет форму овала Кассини. Контур проводника описывается геометрической кривой:

где:

– координаты кривой, ;

– параметры овала, равные и соответственно.

Известно, что сопротивление проводника постоянному току можно определить по выражению:

где:

– удельное электрическое сопротивление проводника, определяемое материалом, из которого изготовлен проводник, ;

– площадь поперечного сечения, ;

– длина проводника, .

Необходимо по известному значению длины проводника и по заданному материалу проводника (медь, алюминий и т.п.) найти величину его сопротивления. Графически отобразить форму проводника в масштабе.

# 4. Описание языка программирования Python

Python — это высокоуровневый интерпретируемый язык программирования, который был разработан в конце 1980-х годов Гвидо ван Россумом в Нидерландах. Язык Python предоставляет простой и элегантный синтаксис, что делает его популярным средством для быстрого и эффективного написания программ на различных платформах.

### История

Python был первоначально создан как универсальный язык программирования, который мог бы облегчить задачи разработки. Со временем Python стал одним из самых популярных языков программирования, используемых в различных областях, таких как веб-разработка, научные исследования, анализ данных и машинное обучение.

### Типы и структуры данных

Python поддерживает различные типы данных, включая числа, строки, списки, кортежи, множества, словари и другие. Язык также предоставляет возможность создания пользовательских типов данных и структур.

### Синтаксис

Синтаксис Python легко читается и понимается, что делает его идеальным выбором для начинающих программистов. Python использует отступы для определения блоков кода, что способствует более читаемому и структурированному коду.

### Возможности

Python обладает мощными функциональностями, такими как динамическая типизация, автоматическое управление памятью, модульность, расширяемость и большое количество сторонних библиотек.

### Основные библиотеки

В Python существует огромное количество стандартных модулей и библиотек, таких как NumPy, Pandas, Matplotlib, TensorFlow, Flask и многие другие, которые обеспечивают расширенные возможности для решения различных задач.

### Применимость к практическим задачам

Python охватывает широкий спектр областей применения, включая веб-разработку, научные исследования, обработку данных, машинное обучение, искусственный интеллект, автоматизацию задач, разработку игр и многое другое. Язык Python позволяет быстро и эффективно решать разнообразные задачи.

### Применимость в энергетике

Python также широко применяется в энергетике для анализа данных, моделирования и прогнозирования, контроля и управления энергосистемами, разработки алгоритмов оптимизации и других задач. Python обладает богатым набором инструментов и библиотек для работы с временными рядами, машинным обучением, глубоким обучением и другими технологиями, которые могут быть полезны в энергетической отрасли.

# 5. Пояснения к заданию

## 5.1. Математическая модель

График функции, задающей овал Кассини, представлен на рисунке 5.1.1.

|  |
| --- |
|  |
| Рис. 5.1.1 – График функции, задающей овал Кассини |

Пусть график пересекает ось абсцисс в точках . Тогда формула для нахождения площади фигуры будет выглядеть так:

Для нахождения программа будет проверять определённость функции. Для этого должно выполняться следующее условие:

Все уравнения, необходимые для расчётов, представлены в листинге 5.1.1.

Данный программный код содержится в модуле common/equation.py.

### Листинг 5.1.1 – Уравнения, необходимые для расчётов

import numpy as np

def radicant(x):

    a = 0.8

    c = 0.7

    if x == 0:

        result = 0.0

    else:

        result = np.sqrt(np.power(a, 4) + 4 \* np.square(c) \* np.square(x)) - np.square(x) - np.square(a)

    return result

def integrand(x):

    result = np.sqrt(radicant(x))

    return result

def resistance(rho, area, length):

    rho = float(rho)

    area = float(area)

    length = float(length)

    result = rho \* area \*np.power(10.0, -4) / length

    return result

## 5.2. Численное интегрирование при помощи встроенной функции

Для нахождения определённого интеграла требуются следующие библиотеки:

1. numpy – для определения диапазонов определённых интегралов;
2. scipy – для вычисления численного решения интеграла.

Блок-схема алгоритма численного интегрирования при помощи встроенной функции представлена на рисунке 5.2.1.

|  |
| --- |
|  |
| Рис. 5.2.1 – Блок-схема алгоритма численного интегрирования при помощи встроенной функции |

Техническая реализация численного интегрирования при помощи встроенной функции представлена в листинге 5.2.1.

Данный программный код содержится в модуле common/built\_in\_integration.py.

### Листинг 5.2.1. – Техническая реализация численного интегрирования при помощи встроенной функции

import numpy as np

from scipy import integrate

import equation

def area():

    h = 0.0001

    x = 0

    x\_0 = x

    while equation.radicant(x\_0) >= 0:

        x\_0 += h

    x\_0 += -h

    integral = integrate.quad(equation.integrand, 0, x\_0)

    result = 4 \* integral[0]

    return result

## 5.3. Численное интегрирование при помощи метода левых прямоугольников, реализованным самостоятельно

**Метод прямоугольников** — метод численного интегрирования функции одной переменной, заключающийся в замене подынтегральной функции на константу, на каждом элементарном отрезке. Если рассмотреть график подынтегральной функции, то метод будет заключаться в приближённом вычислении площади под графиком суммированием площадей конечного числа прямоугольников, ширина которых будет определяться расстоянием между соответствующими соседними узлами интегрирования, а высота — значением подынтегральной функции в этих узлах.

Значение интеграла для элементарного участка находится по следующей формуле:

Блок-схема алгоритма численного интегрирования при помощи метода левых прямоугольников, реализованным самостоятельно, представлена на рисунке 5.3.1.

|  |
| --- |
|  |
| Рис. 5.3.1 – Блок-схема алгоритма численного интегрирования при помощи метода левых прямоугольников, реализованным самостоятельно |

Техническая реализация численного интегрирования при помощи метода левых прямоугольников, представлена в листинге 5.3.1.

Данный программный код содержится в модуле common/my\_integration.py.

### Листинг 5.3.1. – Техническая реализация численного интегрирования при помощи метода левых прямоугольников

import numpy as np

import equation

def area():

    h = 0.0001

    x = 0

    x\_0 = x

    while equation.radicant(x\_0) >= 0:

        x\_0 += h

    x\_0 += -h

    integral = integrate(equation.integrand, 0, x\_0)

    result = 4 \* integral

    return result

def integrate(function, a, b):

    integral = 0

    x = a

    h = np.power(10.0, -5)

    while x < b:

        integral += function(x) \* h

        x += h

    return integral

## 5.4. Алгоритм решения задачи в графическом виде

График, изображённый на рисунке 5.1.1, можно представить в качестве набора простых геометрических фигур, площади которых находятся по известным формулам. Данная процедура показана на рисунке 5.4.1.

|  |
| --- |
|  |
| Рис. 5.4.1 – Процедура разбиения сложной фигуры на несколько простых |

Таким образом, алгоритм решения задачи в графическом виде состоит в следующем:

1. разбить сложную фигуру на несколько простых;
2. вычислить значения площадей простых фигур;
3. сложить их.

## 5.5. Разработка интерфейса

Взаимодействие пользователя с интерфейсом отражено с помощью блок-схемы на рисунке 5.5.1.

|  |
| --- |
|  |
| Рис. 5.5.1 – Взаимодействие пользователя с интерфейсом |

Для более эстетичного внешнего вида была применена тёмная тема из набора “Azure ttk”.

Техническая реализация интерфейса представлена в листингах 5.5.1, 5.5.2, 5.5.3.

Данный программный код содержится в модуле common/interface.py.

### Листинг 5.5.1 – Техническая реализация интерфейса

import tkinter as tk

from tkinter import ttk

from ctypes import windll

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import equation

import built\_in\_integration

import my\_integration

import plotting

windll.shcore.SetProcessDpiAwareness(1)

root = tk.Tk()

root.title("")

root.geometry("1600x900")

root.resizable(False, False)

# Задание темы

root.tk.call("source", "common/azure.tcl")

root.tk.call("set\_theme", "dark")

root.columnconfigure(index=0, weight=2)

root.columnconfigure(index=1, weight=1)

for index in [0, 2]:

    root.rowconfigure(index=index, weight=1)

graph\_frame = ttk.Frame(root)

graph\_frame.grid(

    row=0, column=0, sticky="nsew", rowspan=3

)

conditions\_frame = ttk.LabelFrame(root, text="Начальные условия", padding=(20, 10))

conditions\_frame.grid(

    row=0, column=1, padx=10, pady=(20, 10), sticky="nsew"

)

for index in range(5):

    conditions\_frame.rowconfigure(index=index, weight=1)

solve\_frame = ttk.Frame(root, padding=(0, 10))

solve\_frame.grid(

    row=1, column=1, padx=0, pady=(20, 0), sticky="nsew"

)

answer\_frame = ttk.LabelFrame(root, text="Ответ", padding=(20, 10))

answer\_frame.grid(

    row=2, column=1, padx=10, pady=(20, 10), sticky="nsew"

)

for index in range(2):

    answer\_frame.rowconfigure(index=index, weight=1)

def y(x):

    return 0 \* x

fig = plotting.plotting(y, 0, 0)

graph\_canvas = FigureCanvasTkAgg(fig, master = graph\_frame)

graph\_canvas.draw()

graph\_canvas.get\_tk\_widget().grid(row = 0, column = 0)

equation\_u\_frame = ttk.Frame(conditions\_frame)

equation\_u\_frame.grid(

    row=0, column=0, padx=10, pady=(20, 10), sticky="nsew"

)

equation\_u\_image = tk.PhotoImage(file='gfx/u.png')

equation\_u\_label = ttk.Label(equation\_u\_frame, image=equation\_u\_image)

equation\_u\_label.grid(

    row=0, column=0, sticky='nsew'

)

equation\_U\_frame = ttk.Frame(conditions\_frame)

equation\_U\_frame.grid(

    row=1, column=0, padx=10, pady=(20, 10), sticky='nsew'

)

equation\_U\_image = tk.PhotoImage(file='gfx/u\_sqrt.png')

equation\_U\_label = ttk.Label(equation\_U\_frame, image=equation\_U\_image)

equation\_U\_label.grid(

    row=0, column=0, sticky='nsew'

)

f\_frame = ttk.Frame(conditions\_frame)

f\_frame.grid(

    row=2, column=0, padx=10, pady=(20, 10), sticky='nsew'

)

f\_label = ttk.Label(f\_frame, text="f = ")

f\_label.grid(

    row=0, column=0, sticky='nsew'

)

f\_var = tk.DoubleVar(value=50)

f\_entry = ttk.Entry(f\_frame, textvariable=f\_var)

f\_entry.grid(

    row=0, column=1, sticky='nsew'

)

f\_unit\_label = ttk.Label(f\_frame, text="Гц")

f\_unit\_label.grid(

    row=0, column=2, sticky='nsew'

)

U\_frame = ttk.Frame(conditions\_frame)

U\_frame.grid(

    row=3, column=0, padx=10, pady=(20, 10), sticky='nsew'

)

U\_label = ttk.Label(U\_frame, text="U = ")

U\_label.grid(

    row=0, column=0, sticky='nsew'

)

U\_var = tk.DoubleVar(value=220)

U\_entry = ttk.Entry(U\_frame, textvariable=U\_var)

U\_entry.grid(

    row=0, column=1, sticky='nsew'

)

U\_unit\_label = ttk.Label(U\_frame, text="В")

U\_unit\_label.grid(

    row=0, column=2, sticky='nsew'

)

optionmenu\_frame = ttk.Frame(conditions\_frame)

optionmenu\_frame.grid(

    row=4, column=0, padx=10, pady=(20, 10), sticky='nsew'

)

optionmenu\_list = ["", "Решение встроенной функцией", "Решение реализованной функцией"]

optionmenu\_var = tk.StringVar(value=optionmenu\_list[1])

optionmenu = ttk.OptionMenu(

    optionmenu\_frame, optionmenu\_var, \*optionmenu\_list

)

optionmenu.config(width = 33)

optionmenu.grid(

    row=0, column=0, sticky='nsew'

)

def solve\_command():

    f = f\_entry.get()

    U = U\_entry.get()

    if optionmenu\_var.get() == "Решение встроенной функцией":

        alpha = built\_in\_integration.alpha(f)

    if optionmenu\_var.get() == "Решение реализованной функцией":

        alpha = my\_integration.alpha(f)

    T = equation.T(f)

    U\_0 = equation.U\_0(alpha, U)

    y = lambda x: equation.u(x, f, U\_0)

    fig = plotting.plotting(y, 0, T)

    graph\_canvas = FigureCanvasTkAgg(fig, master = graph\_frame)

    graph\_canvas.draw()

    graph\_canvas.get\_tk\_widget().grid(row = 0, column = 0)

    T = round(T, 3)

    T\_var.set(T)

    U\_0 = round(U\_0, 3)

    U\_0\_var.set(U\_0)

solve\_button = ttk.Button(

    solve\_frame, text="Решить", style="Accent.TButton", command=solve\_command

)

solve\_button.config(width=30)

solve\_button.grid(row=0, column=0, padx=(250, 0), sticky="nsew")

T\_frame = ttk.Frame(answer\_frame)

T\_frame.grid(

    row=0, column=0, padx=10, pady=(20, 10), sticky='nsew'

)

T\_label = ttk.Label(T\_frame, text="T = ")

T\_label.grid(

    row=0, column=0, sticky='nsew'

)

T\_var = tk.DoubleVar(value=0)

T\_entry = ttk.Entry(T\_frame, state="readonly", textvariable=T\_var)

T\_entry.grid(

    row=0, column=1, sticky='nsew'

)

T\_unit\_label = ttk.Label(T\_frame, text="с")

T\_unit\_label.grid(

    row=0, column=2, sticky='nsew'

)

U\_0\_frame = ttk.Frame(answer\_frame)

U\_0\_frame.grid(

    row=1, column=0, padx=10, pady=(20, 10), sticky='nsew'

)

U\_0\_label = ttk.Label(U\_0\_frame, text="U\_0 = ")

U\_0\_label.grid(

    row=0, column=0, sticky='nsew'

)

U\_0\_var = tk.DoubleVar(value=0)

U\_0\_entry = ttk.Entry(U\_0\_frame, state="readonly", textvariable=U\_0\_var)

U\_0\_entry.grid(

    row=0, column=1, sticky='nsew'

)

U\_0\_unit\_label = ttk.Label(U\_0\_frame, text="В")

U\_0\_unit\_label.grid(

    row=0, column=2, sticky='nsew'

)

root.update()

x\_cordinate = int((root.winfo\_screenwidth() / 2) - (root.winfo\_width() / 2))

y\_cordinate = int((root.winfo\_screenheight() / 2) - (root.winfo\_height() / 2))

root.geometry("+{}+{}".format(x\_cordinate, y\_cordinate-20))

root.mainloop()

Данный программный код содержится в модуле common/plotting.py

### Листинг 5.5.2 – Построение графика

import numpy as np

import matplotlib.pyplot as plt

from matplotlib import figure

import locale

def plotting(function, a, b):

    fig = figure.Figure()

    ax = fig.add\_subplot()

    locale.setlocale(locale.LC\_NUMERIC, "de\_RU")

    font = {'family': 'Segoe Ui',

            'size': 10}

    plt.rc('font', \*\*font)

    ax.ticklabel\_format(useLocale=True)

    ax.grid(linewidth = 0.5, color='#7b7b7b')

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

    ax.spines['right'].set\_color('none')

    ax.yaxis.tick\_left()

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

    ax.spines['top'].set\_color('none')

    ax.xaxis.tick\_bottom()

    ax.plot(0, 0)

    ax\_color = '#ffffff'

    ax.set\_facecolor('#333333')

    fig.set\_facecolor('#333333')

    ax.spines['bottom'].set\_color(ax\_color)

    ax.spines['left'].set\_color(ax\_color)

    ax.tick\_params(axis='x', colors=ax\_color)

    ax.tick\_params(axis='y', colors=ax\_color)

    x = np.linspace(a, b, 1000)

    y = function(x)

    ax.plot(x, y, color='#007fff')

    fig.set\_size\_inches(7.2, 7.2)

    return fig

Данный программный код содержится в модуле common/equation.py.

### Листинг 5.5.3 – Уравнения, используемые для расчётов

import numpy as np

def U\_0(alpha, U):

    alpha = float(alpha)

    U = float(U)

    result = alpha \* U

    return result

def u(t, f, U\_0):

    f = float(f)

    U\_0 = float(U\_0)

    p = np.pi

    result = U\_0 \* np.sin(2 \* p \* f \* t) + 0.1 \* U\_0 \* np.sin(6 \* p \* f \* t)

    return result

def T(f):

    result = 1 / float(f)

    return result

Видеокадр программы представлен на рисунке 5.5.2.

|  |
| --- |
|  |
| Рис. 5.5.2 – Видеокадр программы |

## 5.6. Тестирование программы

Тестирование программы представлено в таблице 5.6.1.

Таблица 5.6.1 – Тестирование программы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № |  |  | Решение встроенной функцией | | Решение реализованной функцией | |
|  |  |  |  |
| 1 | 50 | 200 |  |  |  |  |
| 2 | 50 | 220 |  |  |  |  |
| 3 | 60 | 200 |  |  |  |  |
| 4 | 60 | 220 |  |  |  |  |

# 6. Выводы

Были получены навыки алгоритмизации, разработки и тестирования программ на языке Python на примере задачи из области электроэнергетики.

# 7. Список используемых источников

1. <https://www.geeksforgeeks.org/how-to-find-definite-integral-using-python/>
2. <https://ru.wikipedia.org/wiki/Метод_прямоугольников>
3. <https://github.com/rdbende/Azure-ttk-theme>