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 Куделко А. Е.  (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](#_Toc168054756)

[2. Задание 3](#_Toc168054757)

[3. Описание исходных данных 3](#_Toc168054758)

[4. Описание языка программирования Python 4](#_Toc168054759)

[5. Пояснения к заданию 5](#_Toc168054760)

[5.1. Математическая модель 5](#_Toc168054761)

[5.2. Численное интегрирование при помощи встроенной функции 6](#_Toc168054762)

[5.3. Численное интегрирование при помощи метода левых прямоугольников, реализованным самостоятельно 8](#_Toc168054763)

[5.4. Алгоритм решения задачи в графическом виде 10](#_Toc168054764)

[5.5. Разработка интерфейса 11](#_Toc168054765)

[5.6. Тестирование программы 18](#_Toc168054766)

[6. Выводы 18](#_Toc168054767)

[7. Список используемых источников 19](#_Toc168054768)

# 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

Электроустановка подключена к электросети через устройство выпрямления напряжения. Выпрямленное напряжение изменяется по следующему закону:

где:

– максимальное напряжение сети, ;

– частота тока, ;

– время, .

В работе электроустановка будет потреблять энергию, которую можно определить как:

где:

– сопротивление электроустановки, ;

– время работы электроустановки, .

Необходимо по заданным величинам максимального напряжения сети, времени работы электроустановки и ее потребляемой энергии за это время определить сопротивление электроустановки. Построить график изменения напряжения от времени.

# 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

В уравнении является постоянной величиной, поэтому в уравнении её можно вынести за знак интеграла:

Тогда сопротивление электроустановки можно найти следующим образом:

Если коэффициент пропорциональности обозначить за , то формула для расчёта сопротивления электроустановки выглядит так:

где:

## 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

def alpha(T):

    p = np.pi

    f = 50

    integrand = lambda t: np.square(2 \* p \* f \* t)

    integral = integrate.quad(integrand, 0, T)

    result = 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

def alpha(T):

    p = np.pi

    f = 50

    integrand = lambda t: np.square(2 \* p \* f \* t)

    integral = integrate(integrand, 0, T)

    result = integral

    return result

def integrate(function, a, b):

    integral = 0

    x = a

    h = np.power(10.0, -6)

    while x < b:

        integral += function(x) \* h

        x += h

    return integral

## 5.4. Алгоритм решения задачи в графическом виде

Алгоритм решения задачи в графическом виде представлен на рисунке 5.4.1488.

|  |
| --- |
|  |
| Рис. 5.4.1488 – Решение задачи в графическом виде |

## 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 numpy as np

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\_frame = ttk.Frame(conditions\_frame)

equation\_frame.grid(

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

)

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

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

equation\_label.grid(

    row=0, column=0, sticky='nsew'

)

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

U\_m\_frame.grid(

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

)

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

U\_m\_label.grid(

    row=0, column=0, sticky='nsew'

)

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

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

U\_m\_entry.grid(

    row=0, column=1, sticky='nsew'

)

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

U\_m\_unit\_label.grid(

    row=0, column=2, sticky='nsew'

)

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

W\_frame.grid(

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

)

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

W\_label.grid(

    row=0, column=0, sticky='nsew'

)

W\_var = tk.DoubleVar(value=1)

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

W\_entry.grid(

    row=0, column=1, sticky='nsew'

)

W\_unit\_label = ttk.Label(W\_frame, text="Дж")

W\_unit\_label.grid(

    row=0, column=2, sticky='nsew'

)

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

T\_frame.grid(

    row=3, 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=1)

T\_entry = ttk.Entry(T\_frame, 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'

)

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():

    U\_m = float(U\_m\_entry.get())

    W = float(W\_entry.get())

    T = float(T\_entry.get())

    print(type(T))

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

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

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

        alpha = my\_integration.alpha(T)

    R = equation.R(alpha, U\_m, W)

    y = lambda x: equation.u(x, U\_m)

    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)

    R = round(R, 3)

    R\_var.set(R)

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")

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

R\_frame.grid(

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

)

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

R\_label.grid(

    row=0, column=0, sticky='nsew'

)

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

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

R\_entry.grid(

    row=0, column=1, sticky='nsew'

)

R\_unit\_label = ttk.Label(R\_frame, text="Ом")

R\_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 R(alpha, U\_m, W):

    alpha = float(alpha)

    U\_m = float(U\_m)

    W = float(W)

    result = alpha \* np.square(U\_m) / (3600 \* W)

    return result

def u(t, U\_m):

    p = np.pi

    f = 50.0

    result = U\_m \* np.abs(2 \* p \* f \* t)

    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 | 220 |  |  |
| 2 | 440 |  |  |

# 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>