ФИО: Жидков Артемий Андреевич

группа: R4136c

1. Импортировать библиотеки в Python.

```
import numpy as np
import matplotlib.pyplot as plt
import os
import random
import scipy
import torch

torch.cuda.synchronize()
torch.cuda.empty_cache()

cuda = torch.device('cuda')
print(torch.cuda.get_device_properties(cuda))
```

```
_CudaDeviceProperties(name='NVIDIA GeForce RTX 3080 Laptop
```

2. Загрузка и подготовка данных.

```
name = random.choice(os.listdir("dataset"))

# name = 'testLab1Var7.csv'

print(f"Dataset: {name}")

dataset = np.genfromtxt(f"dataset/{name}", delimiter=',')

dataset = [dataset[:, i] for i in range(dataset.shape[1])]
title = ["time", "current", "voltage"]

dataset_dict = dict(zip(title, dataset))
```

```
Dataset: testLab1Var11.csv
```

3. Нарисовать графики тока и напряжения.

Для удобства отображения отображу не весь график, а некоторый его случайный диапазон заданного размера, установив лимиты на данные.

```
"""


"""
time_period = 0.1

time_interval = random.random() * (dataset_dict["time"][-1]
time_interval = (time_interval, time_interval + time_perio

print(f"            {time_interval}")
```

```
            (28.65413982391623, 28.75413982391623)
```

```
plt.plot(dataset_dict["time"], dataset_dict["current"])
plt.xlim(time_interval)
plt.xlabel('   ,  ')
plt.ylabel('      ,  ')
```

4. Рассчитать значения параметров L и R.

Упрощённая модель двигателя постоянного тока.
Модель двигателя постоянного тока описывается следующей системой дифференциальных уравнений:

$$\begin{cases} u = e + R \times i + L \times \dfrac{di}{t} \\ M - M_C = J\dfrac{d\omega}{t} \\ M = C_M \times \Phi \times i \\ e = C_\omega \times \Phi \times \omega \end{cases}$$

где

u - напряжение на якорной обмотке двигателя,

e - электродвижущая сила (ЭДС) якоря,

i - ток якоря,

5 Рассчитать средние значения и стандартное отклонение.

Для нахождения ошибки между реальным значением Y и его предсказанием моделью $X \times K$, можно просто посчитать их разность $e = Y - X \times K$

Тогда Сумма квадратов ошибки будет:

$$S(K) = \sum e_i^2 = e^T \times e = (Y - X \times K)^T \times (Y - X \times K)$$

А среднеквадратичное отклонение:

$$\sigma_Y = \sqrt{\frac{S(K)}{n}}$$

```
e2_Y = torch.mm(Y_tensor.T - torch.mm(X_tensor, K_approx).T
sigma2_Y = torch.divide(e2_Y, Y_tensor.shape[0])

sigma_Y = torch.sqrt(sigma2_Y)
```