МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Кафедра систем управління літальними апаратами

Лабораторна робота № 5

з дисципліни «Об'єктно-орієнтоване програмування авіаційно-транспортних систем»

Тема: «Розробка графічного інтерфейсу для розрахункових завдань і побудови графіків»

ХАІ.301.173.320.03 ЛР

Виконав студент гр. _____320_____

_____Полякова Софія___
(підпис, дата) (П.І.Б.)

Перевірив
_____ к.т.н., доц. О. В. Гавриленко
_____ ас. Є. В. Пявка
(підпис, дата) (П.І.Б.)

2024

МЕТА РОБОТИ

Застосувати теоретичні знання з основ роботи з бібліотекою tkinter на мові Python, навички використання бібліотеки matplotlib, а також об'єктно-орієнтований підхід до проектування програм, і навчитися розробляти скрипти для інженерних додатків з графічним інтерфейсом.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Описати клас, який реалізує графічний інтерфейс користувача для вирішення розрахункової задачі згідно варіанту і скрипт для роботи з об'єктом цього класу. Зазначена у задачі функція повинна бути окремим методом класу.

Func12. Описати функцію IsPowerN(K, N) логічного типу, що повертає True, якщо цілий параметр K(>0) є ступеню числа N(>1), і False інакше. Дано число N (> 1) і набір із 10 цілих додатних чисел. За допомогою функції IsPowerN знайти кількість ступенів числа N у цьому наборі.

Завдання 2. Розробити скрипт із графічним інтерфейсом, що виконує наступні функції:

A. установка початкових значень параметрів для побудови графіка (змінні Tkinter)

B. створення текстового файлу з двома стовпцями даних: аргумент і значення функції відповідно до варіанту. Роздільник в кожному рядку файлу: '#';

C. зчитування з файлу масивів даних;

D. підрахунок і відображення мінімального / максимального значення аргументу / функції у зчитаних масивах;

E. відображення масивів даних за допомогою пакета matplotlib у вигляді графіка функції в декартовій системі координат з назвою функції, позначенням осей, оцифруванням і сіткою;

F. заголовок вікна повинен містити текст текст:

lab # - <# групи> -v <# варіанту> - <прізвище> - <ім'я>.

3	$y[k] = 2 \cdot \left(1 - \frac{\xi \cdot T_0}{T}\right) \cdot y[k-1] + \left(\frac{2 \cdot \xi \cdot T_0}{T} - 1 - \frac{T_0^2}{T^2}\right) \cdot y[k-2] + \frac{K \cdot T_0^2}{T^2} \cdot U$	U [0] = 0.1 рад / с, y [0] == y [1] = 0	$T = 0{,}05$ $K = 3$ $\xi = 0{,}25$	y – υ, рад U – δ_B, рад

ВИКОНАННЯ РОБОТИ

Завдання 1. Вирішення задачі Func12.

Вхідні дані (ім’я, опис, тип, обмеження):

n — число, ціле, n > 1;

k — ряд з десяти чисел, де є числа ступеню n, цілі числа, k > 0.

Вихідні дані (ім’я, опис, тип):

count — число, яке означає кількість ступенів числа n в ряду, ціле число, count ≥ 0.

Алгоритм вирішення показано на рис.1

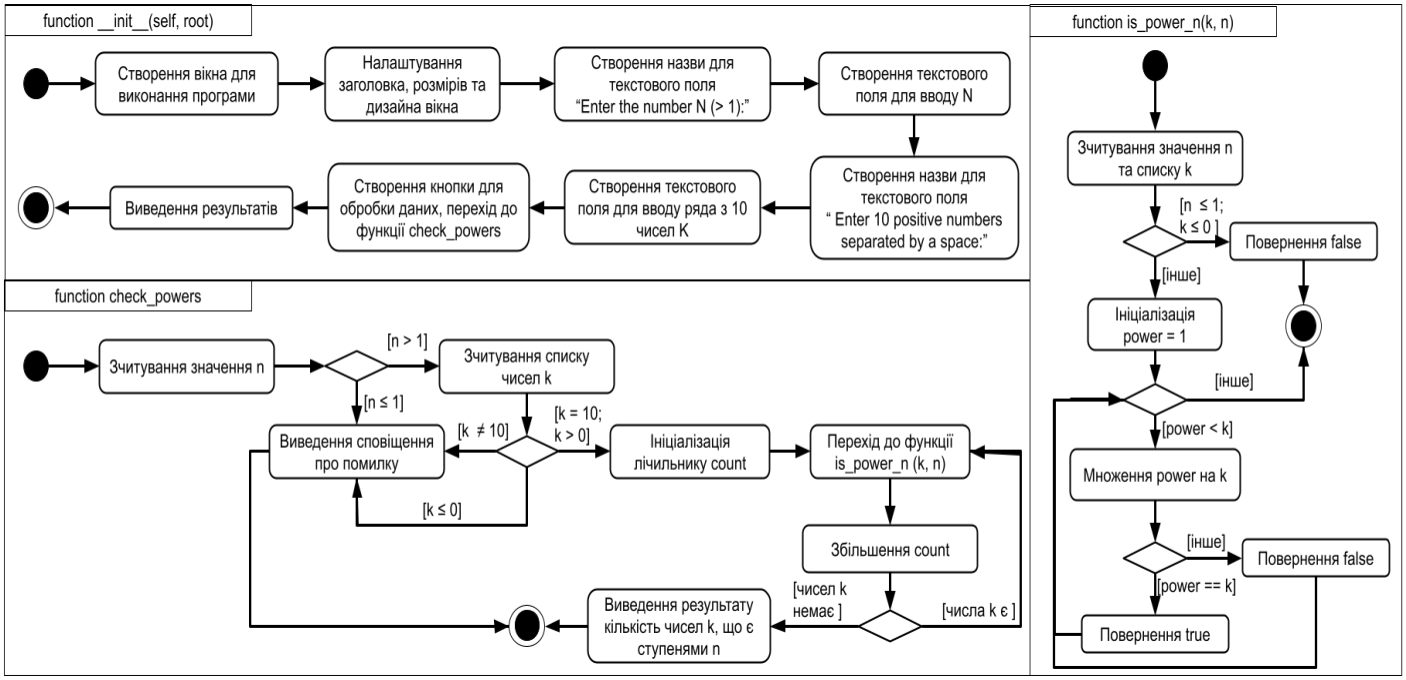

Рисунок 1 – Алгоритм вирішення задачі Func12

Лістинг коду вирішення задачі наведено в дод. А (стор. 7).

Екран роботи програми показаний на рис. В.1-3.

Завдання 3. Вирішення задачі 3.

Вхідні дані (ім’я, опис, тип, обмеження):

T = 0.05 — часовий крок(можна змінювати), дійсне значення, T > 0;

N = 100 — кількість точок(можна змінювати), цілі числа, N > 0;

K = 3 — константа(можна змінювати), цілі числа, K > 0;

Z = 0.25 — коефіцієнт демпфування(можна змінювати), дійсне значення, Z > 0;

U = 0.1 — число, дійсне значення, U > 0;

Вихідні дані (ім’я, опис, тип):

k — константи для розрахунку, дійсні числа, k > 0;

y — значення розрахунку, дійсні числа, y > 0;

Алгоритм вирішення показано на рис.2-3.

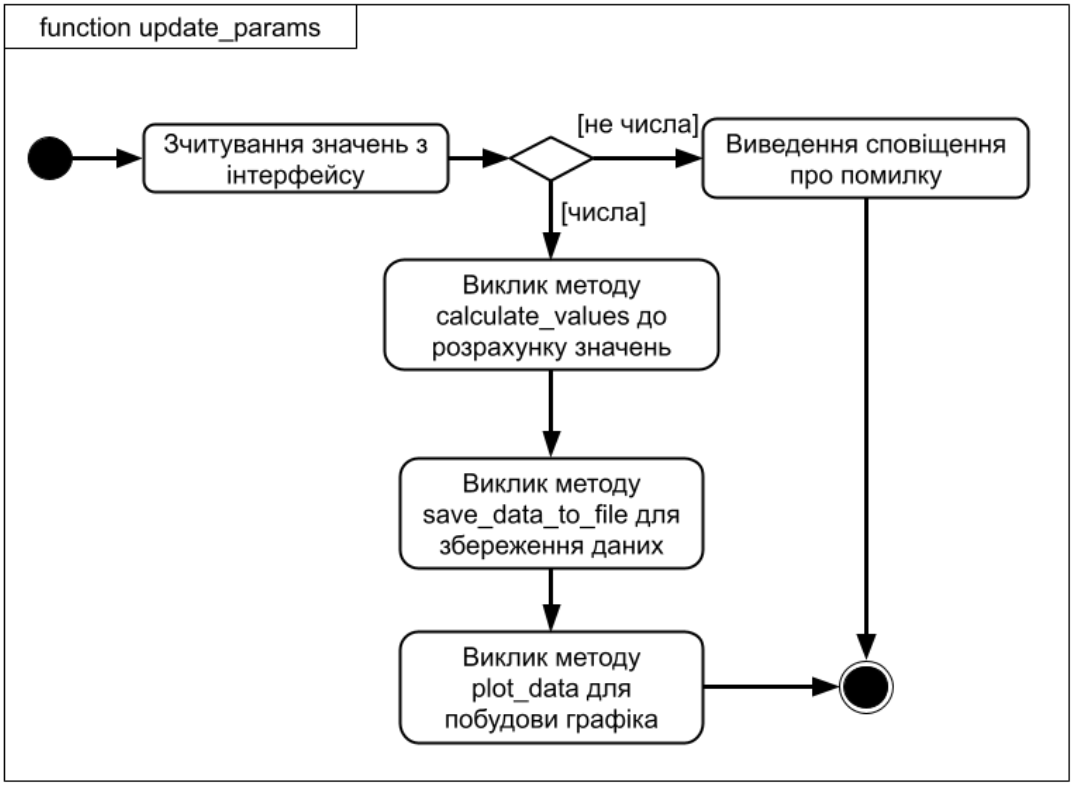

Рисунок 2 — Алгоритм вирішення функції update_params

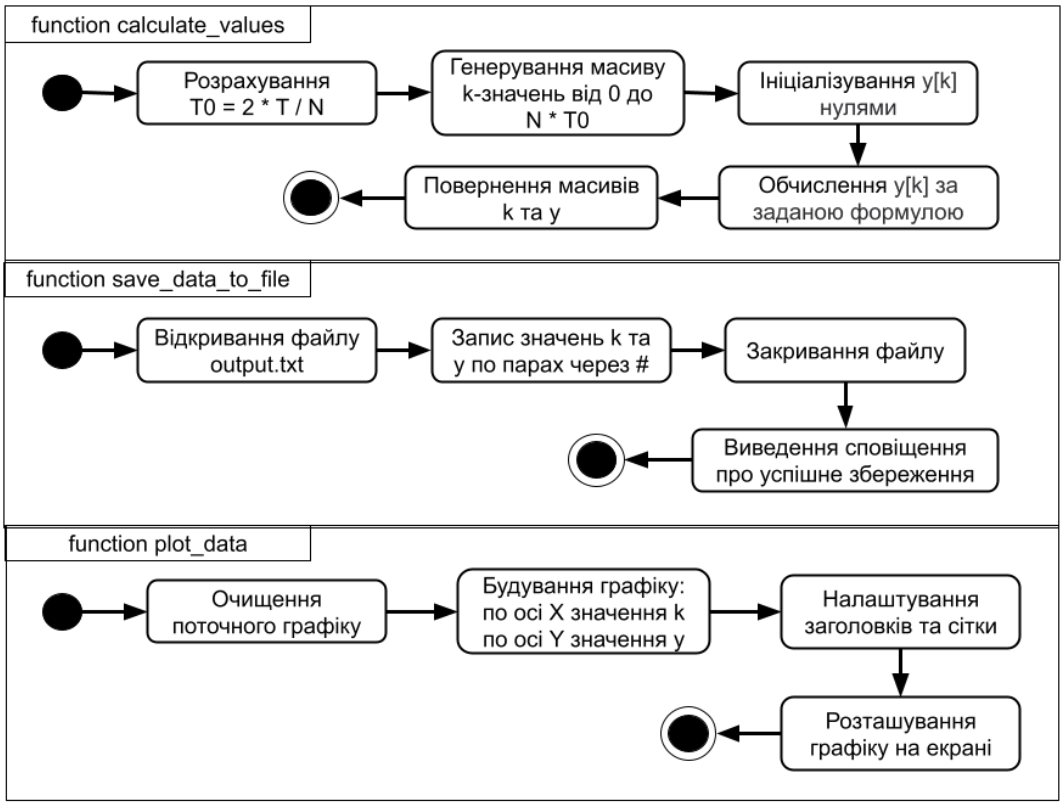

Рисунок 3 — Алгоритм вирішення функції calculate_values, save_data_to_file та plot_data

Лістинг коду вирішення задачі наведено в дод. А (стор. 9).
Екран роботи програми показаний на рис. В.4-6.

ВИСНОВКИ

В ході лабораторної роботи було вивчено теоретичний матеріал з основ роботи з бібліотекою tkinter на мові Python. Закріплено на практиці використання бібліотеки matplotlib, а також об'єктно-орієнтований підхід до проектування програм. Отримано навички з розробки скриптів для інженерних додатків з графічним інтерфейсом.

ДОДАТОК А
Лістинг коду програми до задачі Func12

```
import tkinter as tk
from tkinter import messagebox
class PowerCheckerApp:

  def __init__(self, root):
    self.root = root
    self.root.title("Checking the powers of N")
    self.root.geometry("500x300")
    self.root.configure(bg="pink")

    # entering the number N
    self.label_n = tk.Label(root, text=" Enter the number N (> 1):" , bg="pink",
fg="black")
    self.label_n.pack()
    self.entry_n = tk.Entry(root, width=20)
    self.entry_n.pack()

    # entering a list of K numbers
    self.label_k = tk.Label(root, text=" Enter 10 positive numbers separated by
a space:", bg="pink", fg="black")
    self.label_k.pack()
    self.entry_k = tk.Entry(root, width=30)
    self.entry_k.pack()

    # button to perform a check
    self.check_button = tk.Button(root, text="Check", bg="light blue",
fg="black", command=self.check_powers)
    self.check_button.pack()

    # displaying the result
    self.result_label = tk.Label(root, text="", fg="black", bg="pink")
    self.result_label.pack()

  def check_powers(self):
     try:
       # reading the number N
       n = int(self.entry_n.get())
       if n <= 1:
           raise ValueError("The number N must be greater than 1!")

       # reading a list of K numbers
       k_values = list(map(int, self.entry_k.get().split()))
       if len(k_values) != 10:
           raise ValueError("It is necessary to enter exactly 10 numbers!")
       if any(k <= 0 for k in k_values):
           raise ValueError("All numbers in the list must be positive!")
```

```
        # checking each number from the list
        count = 0
        for k in k_values:
            if self.is_power_n(k, n):
                count += 1

        # outputting the result
        self.result_label.config(text=f"The number of powers of a number {n}:
{count}")

      except ValueError as e:
        # display an error message
        messagebox.showerror("Error", str(e))

  @staticmethod
  def is_power_n(k, n):

      if k <= 0 or n <= 1:
          return False
      power = 1
      while power < k:
          power *= n
      return power == k

if __name__ == "__main__":
  root = tk.Tk()
  app = PowerCheckerApp(root)
  root.mainloop()
```

ДОДАТОК Б
Лістинг коду програми до задачі 3

```
import tkinter as tk
from tkinter import messagebox
import numpy as np
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import matplotlib.pyplot as plt


class FunctionApp:
    def __init__(self, root):
        self.root = root
        self.root.title("lab4_2-320-3-Poliakova-Sofiia")
        self.root.geometry("800x600")

        # initialization of parameters
        self.T = 0.05
        self.N = 100
        self.K = 3
        self.Z = 0.25
        self.U = 0.1

        # graph
        self.figure = plt.Figure(figsize=(5, 4), dpi=100)
        self.ax = self.figure.add_subplot(111)
        self.canvas = FigureCanvasTkAgg(self.figure, master=self.root)
        self.canvas_widget = self.canvas.get_tk_widget()
        self.canvas_widget.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)

        # creation of interface elements
        self.create_widgets()

    def create_widgets(self):
        frame = tk.Frame(self.root)
        frame.pack(side=tk.LEFT, padx=10, pady=10)

        # entering parameters
        label_T = tk.Label(frame, text="T (time step):")
        label_T.grid(row=0, column=0)
        self.entry_T = tk.Entry(frame)
        self.entry_T.insert(0, str(self.T))
        self.entry_T.grid(row=0, column=1)

        label_N = tk.Label(frame, text="N (number of points):")
        label_N.grid(row=1, column=0)
        self.entry_N = tk.Entry(frame)
```

```
        self.entry_N.insert(0, str(self.N))
        self.entry_N.grid(row=1, column=1)

        label_K = tk.Label(frame, text="K (constant):")
        label_K.grid(row=2, column=0)
        self.entry_K = tk.Entry(frame)
        self.entry_K.insert(0, str(self.K))
        self.entry_K.grid(row=2, column=1)

        label_Z = tk.Label(frame, text="Z (damping factor):")
        label_Z.grid(row=3, column=0)
        self.entry_Z = tk.Entry(frame)
        self.entry_Z.insert(0, str(self.Z))
        self.entry_Z.grid(row=3, column=1)

        label_U = tk.Label(frame, text="U (input):")
        label_U.grid(row=4, column=0)
        self.entry_U = tk.Entry(frame)
        self.entry_U.insert(0, str(self.U))
        self.entry_U.grid(row=4, column=1)

        # button to update parameters
        button_update = tk.Button(frame, text="Update and Plot",
command=self.update_params)
        button_update.grid(row=5, column=0, columnspan=2, pady=10)

    def update_params(self):
        try:
            # update settings
            self.T = float(self.entry_T.get())
            self.N = int(self.entry_N.get())
            self.K = int(self.entry_K.get())
            self.Z = float(self.entry_Z.get())
            self.U = float(self.entry_U.get())

            # calculation of values
            k_values, y_values = self.calculate_values(self.T, self.N, self.K,
self.Z, self.U)
            self.save_data_to_file(k_values, y_values, 'output.txt')
            self.plot_data(k_values, y_values, f"Function for Variant {self.K}")
        except ValueError:
            messagebox.showerror("Error", "Invalid input values. Please check
the values entered.")

    def calculate_values(self, T, N, K, Z, U):
        T0 = 2 * T / N
        k_values = np.linspace(0, N * T0, N)
        y_values = np.zeros(N)
```

```
        # using the formula for y[k]
        for k in range(2, N):
            y_values[k] = 2 * (1 - (Z * T0) / T) * y_values[k - 1] + ((2 * Z *
T0) / T - 1 - (T0**2 / T**2)) * y_values[k - 2] + ((K * T0**2) / T**2) * U
        return k_values, y_values

    def save_data_to_file(self, k_values, y_values, filename):
        with open(filename, 'w') as f:
            for k, y in zip(k_values, y_values):
                f.write(f"{k} # {y}\n")
        messagebox.showinfo("Info", f"Data saved to {filename}")

    def plot_data(self, k_values, y_values, title):
        self.ax.clear()
        self.ax.plot(k_values, y_values, label='Function')
        self.ax.set_title(title)
        self.ax.set_xlabel('Time (s)')
        self.ax.set_ylabel('Function Value')
        self.ax.grid(True)
        self.ax.legend()
        self.canvas.draw()

root = tk.Tk()
app = FunctionApp(root)
root.mainloop()
```

ДОДАТОК В
Скрін-шоти вікна виконання програми

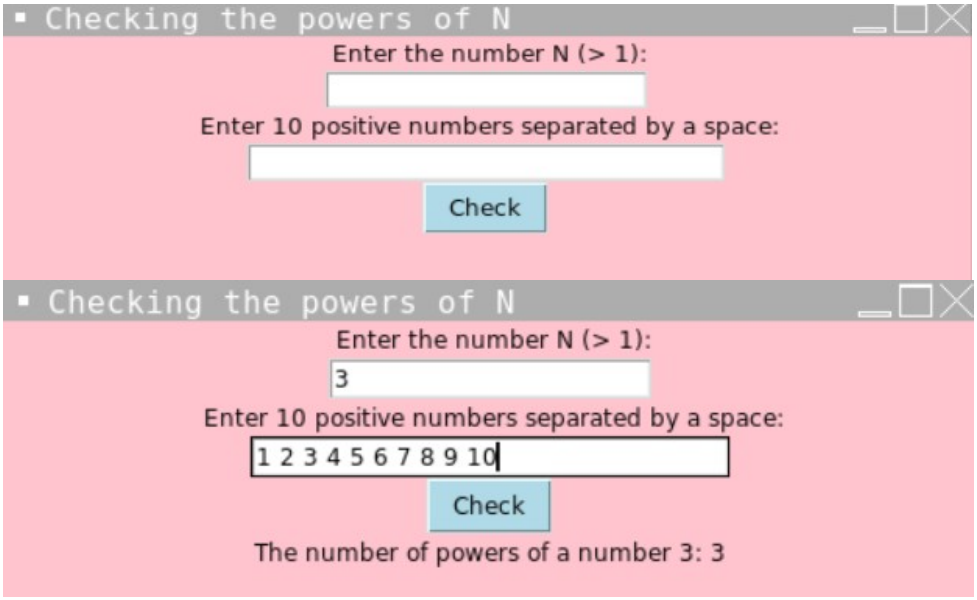

Рисунок В.1 – Екран виконання програми для вирішення завдання Func12

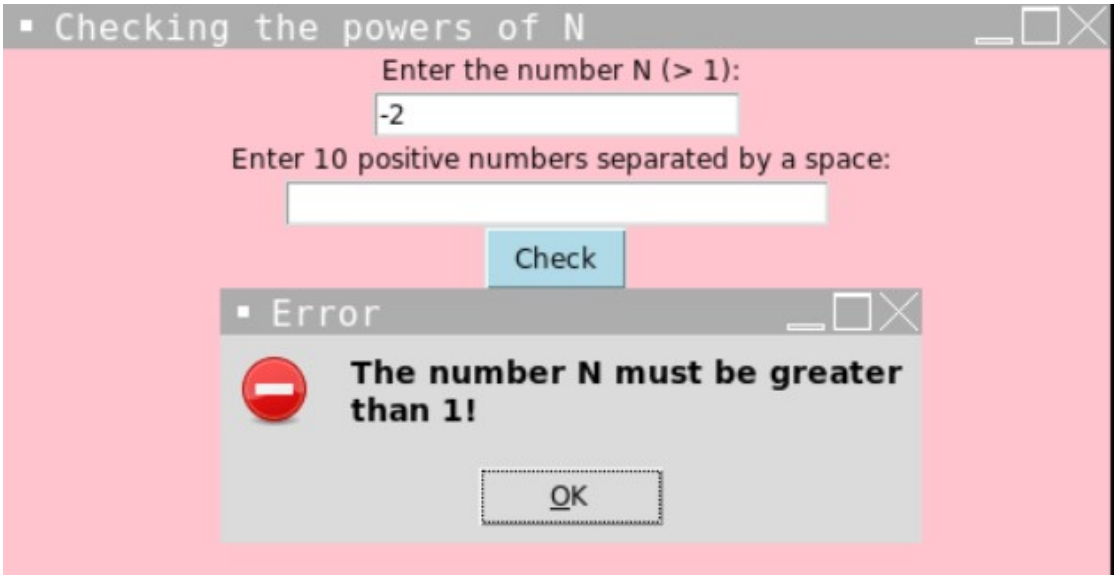

Рисунок В.2 – Екран виконання програми для вирішення завдання Func12

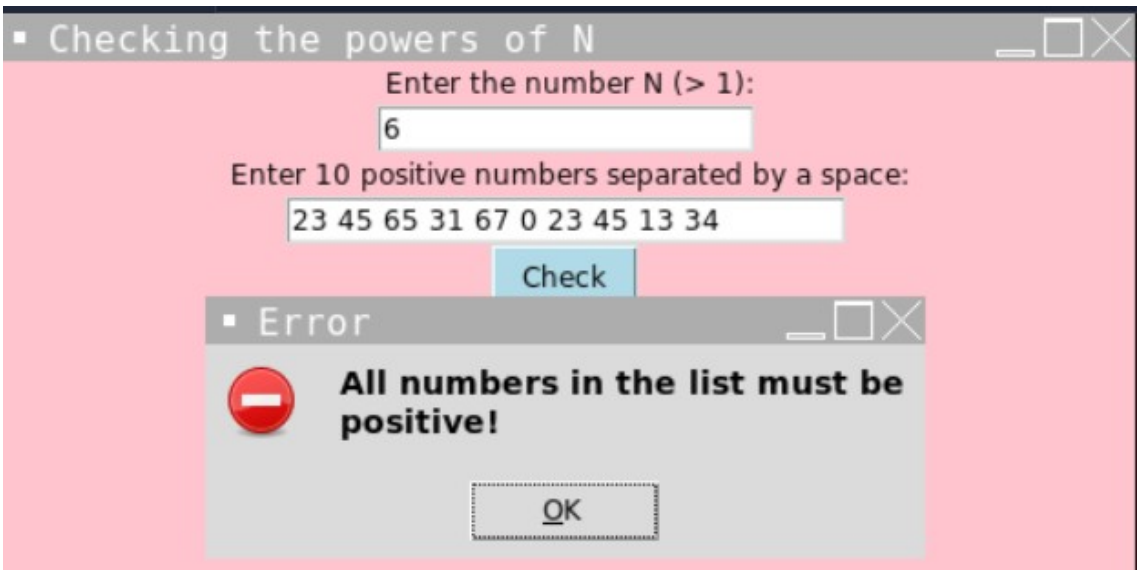

Рисунок В.3 – Екран виконання програми для вирішення завдання Func12

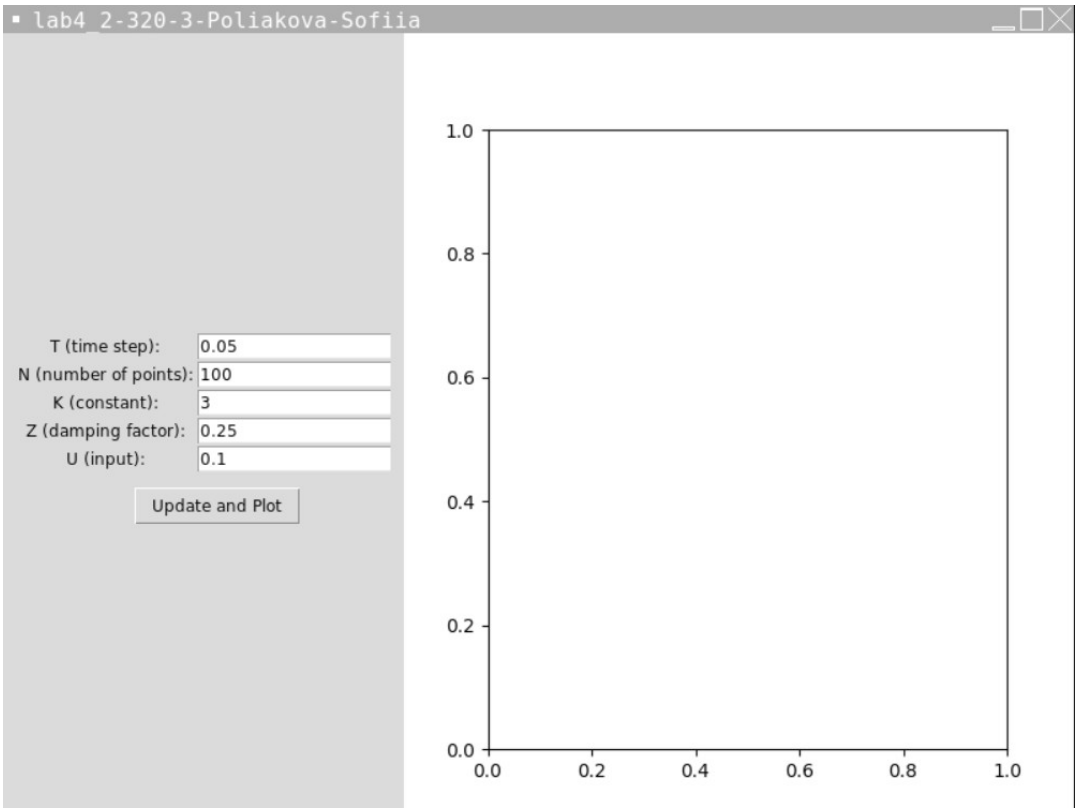

Рисунок В.4 – Екран виконання програми для вирішення завдання 3

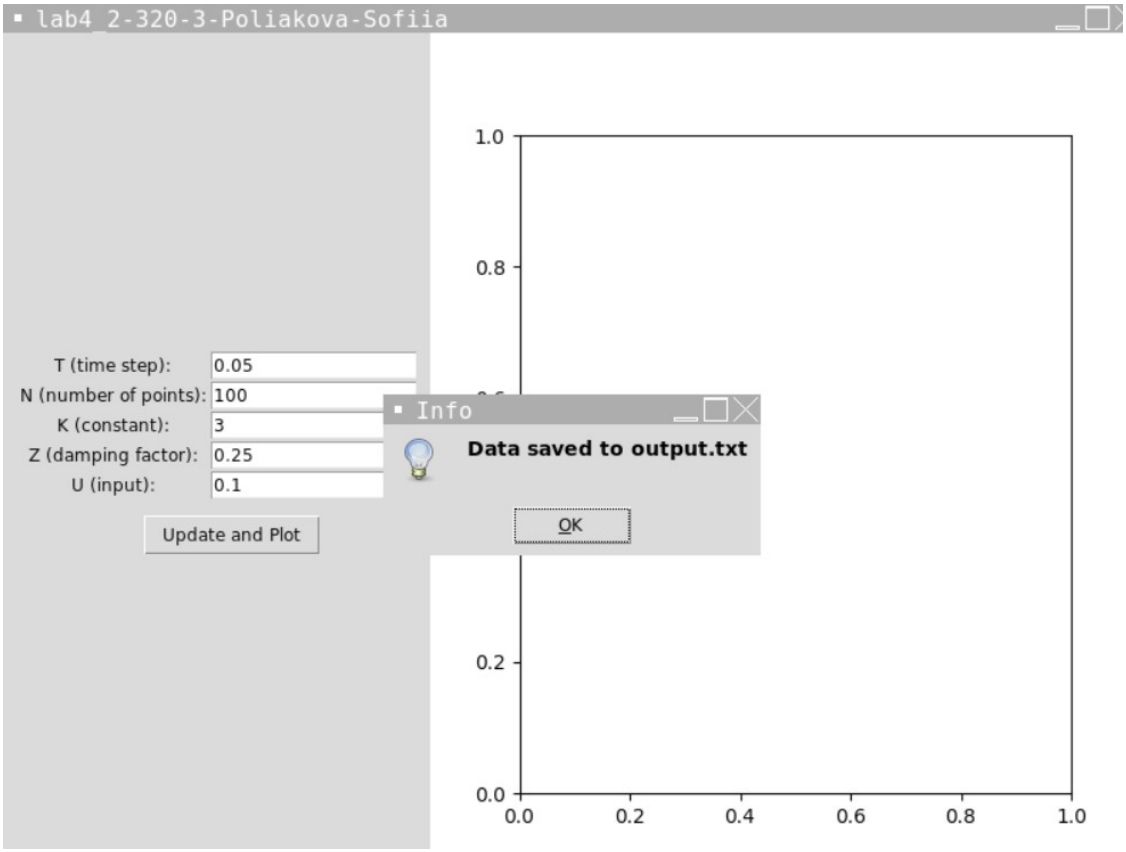

Рисунок В.5 – Екран виконання програми для вирішення завдання 3

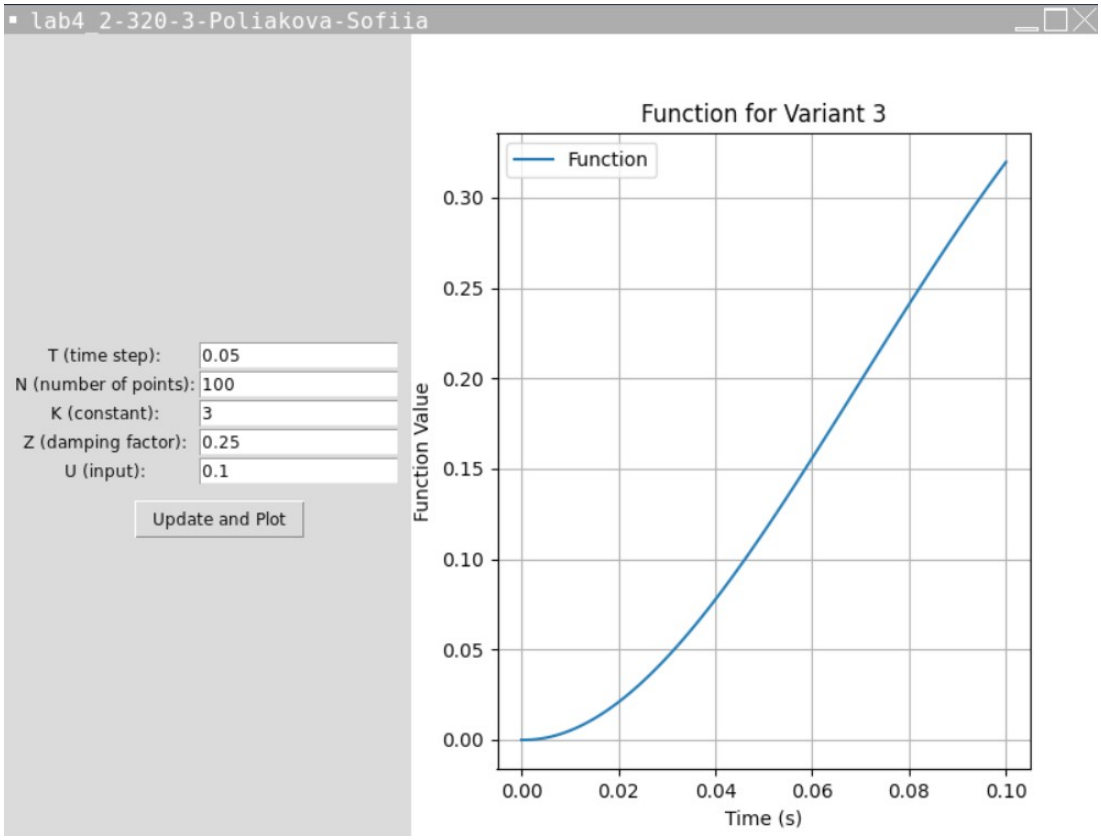

Рисунок В.6 – Екран виконання програми для вирішення завдання 3