Министерство науки и высшего образования Российской Федерации
Пензенский государственный университет
Кафедра «Математическое обеспечение и применение ЭВМ»

ОТЧЕТ

по лабораторной работе №2
по курсу «Моделирование систем»
на тему «Программная реализация имитационной модели нелинейной динамической системы»
Вариант 10

Выполнил:
ст. гр. 21ВП2 Копылов Е.А.
Принял:
д.т.н., профессор Козлов А.Ю.

Пенза, 2024

Содержание задания

В соответствии с индивидуальным вариантом задания разработать и отладить программное приложение, обеспечивающее:

1. Решение системы дифференциальных уравнений на интервале [0; T] для T = 11 с с любым шагом, задаваемым пользователем в пределах (0; T). Для демонстрации результатов обеспечить вывод графиков xi(t), i=1, 2, …, n; значения указанной в задании переменной состояния в конце интервала интегрирования xk(T) и значения относительной погрешности его определения δ.

2. Анализ зависимости точности и трудоемкости решения задачи от шага интегрирования. Вывод графиков зависимостей относительной погрешности δ и оценки трудоемкости от величины шага h.

3. Автоматический выбор величины шага интегрирования для достижения относительной погрешности не более 1% с выводом итоговых результатов, перечисленных в п. 1, для найденного шага.

Вариант задания

$$\dot{x_1} = x_2$$

$$\dot{x_2} = \frac{cu}{x_3} - g - \frac{rx_2^2}{x_3}$$

$$\dot{x_3} = -u$$

Ход работы

Результаты решения системы дифференциальных уравнений на интервале [0; T] для T = 11 с с любым шагом, задаваемым пользователем в пределах (0; T) представлены на рисунке 1.

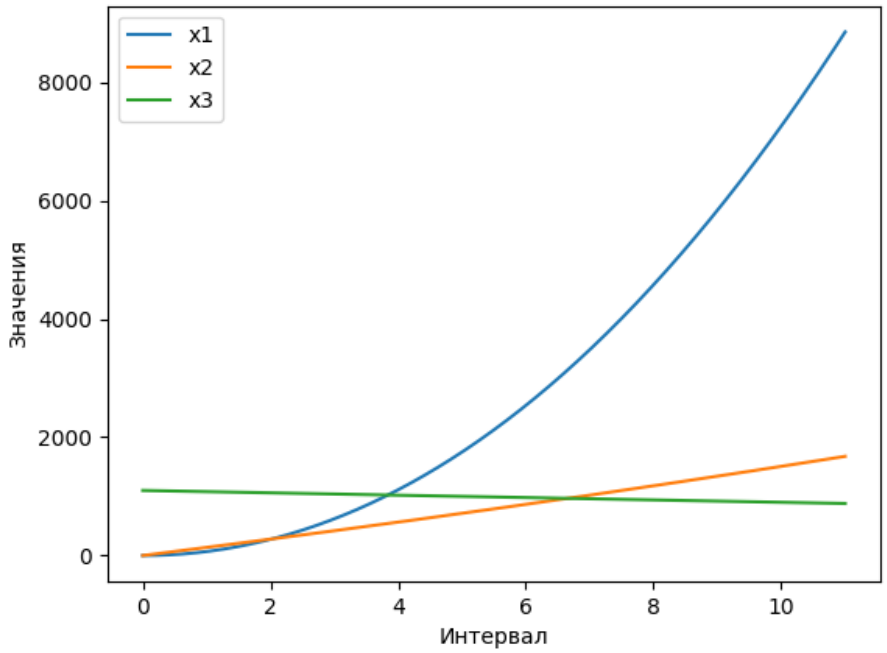

Рисунок 1 – Вывод графика для уравнения с шагом 0,0001

Графики зависимости точности и трудоемкости решения задачи от шага интегрирования приведены на рисунках 2 .

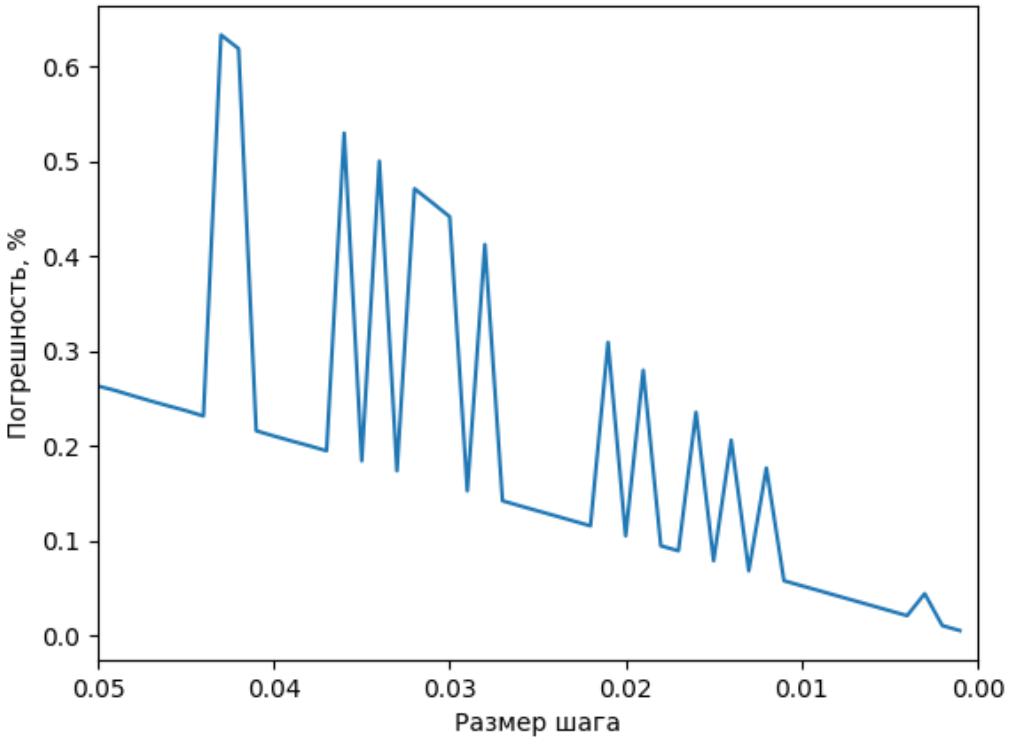

Рисунок 2 – Зависимость точности решения задачи от шага интегрирования

Код программы на языке Python приведен в листинге 1.

Листинг 1 – Program.py

```
from matplotlib import pyplot as plt
import numpy as np
import math


c = 8000
u = 20
T = 11
h_0 = 9900
g = 9.81
x1 = 0
x2 = 0
x3 = 1100
step_size = 0.01


def system(vector):
    F = np.zeros(3)
    r = 0.1 * math.e ** -vector[1] / h_0
    F[0] = vector[1]  # x1
    F[1] = (c * u) / vector[2] - g - (r * vector[1] ** 2) / vector[2]  #
(cu)/x3 - g - (rx2^2)/x3
    F[2] = -u  # -u
    return F


def Euler(vector, step_size, calculation_end):
    step_count = int(calculation_end / step_size) + 1
    result = [vector]
    for i in range(1, step_count):
        result.append(result[-1] + step_size * system(result[-1]))
    return result


def fixed_step():
    R1 = Euler(np.array([x1, x2, x3]), step_size, T)
    R2 = Euler(np.array([x1, x2, x3]), step_size / 2, T)
    steps = np.arange(0, T + step_size, step_size)
    res_x1 = [R1[i][0] for i in range(0, len(steps))]
    res_x2 = [R1[i][1] for i in range(0, len(steps))]
    res_x3 = [R1[i][2] for i in range(0, len(steps))]

    plt.plot(steps, res_x1, label='x1')
    plt.plot(steps, res_x2, label='x2')
    plt.plot(steps, res_x3, label='x3')

    plt.xlabel('Интервал')
    plt.ylabel('Значения')
    plt.legend()
    plt.show()

    print(f'Погрешность переменной X1 при шаге {step_size} составляет
{abs((R2[-1][0] - R1[-1][0]) / R2[-1][0]) * 100}%')
    print(f'Погрешность переменной X2 при шаге {step_size} составляет
{abs((R2[-1][1] - R1[-1][1]) / R2[-1][1]) * 100}%')
    print(f'Погрешность переменной X3 при шаге {step_size} составляет
{abs((R2[-1][2] - R1[-1][2]) / R2[-1][2]) * 100}%')
```

```
def dynamic_step():
    curr_step_size = 0.05
    first_step_size = curr_step_size
    prev_step = []
    prev_lost = []
    while curr_step_size > 0:
        R1 = Euler(np.array([x1, x2, x3]), curr_step_size, T)
        R2 = Euler(np.array([x1, x2, x3]), curr_step_size / 2, T)
        sigma = abs((R2[-1][0] - R1[-1][0]) / R2[-1][0]) * 100
        print(f'Размер шага {round(curr_step_size, 6)} потери {round(sigma,
2)}%')
        prev_step.append(curr_step_size)
        prev_lost.append(sigma)
        if sigma > 1:
            break
        curr_step_size -= 0.001

    fig, ax = plt.subplots()
    ax.plot(prev_step, prev_lost)
    plt.xlabel('Размер шага')
    plt.ylabel('Погрешность, %')
    ax.set_xlim(first_step_size, curr_step_size)
    plt.show()
    print(f"Итоговый размер шага: {prev_step[-1]}")


if __name__ == '__main__':
    fixed_step()
    dynamic_step()
```

Выводы

В ходе выполнения лабораторной работы было разработано и отлажено программное приложение, обеспечивающее решение системы дифференциальных уравнений на интервале [0; T] для T = 11 с с любым шагом, задаваемым пользователем в пределах (0; T), анализ зависимости точности и трудоемкости решения задачи от шага интегрирования, автоматический выбор величины шага интегрирования для достижения относительной погрешности не более 1% с выводом итоговых результатов.