Тестовое задание в компанию "Ритм"

Решение задачи Коши для идеального LC-контура

Кириллова А.Р.

4 октября 2023 г.

Задача: Дана электрическая схема, состоящая из идеального конденсатора ёмкостью $C = 1$ Ф, идеального индуктора индуктивностью $L = 1$ Гн и заземления. Пусть в начальный момент времени конденсатор заряжен до 1 В, а ток в цепи отсутствует.

Построить соответствующую электрической схеме задачу Коши для системы обыкновенных дифференциальных уравнений и написать программу для ее решения любым численным методом на интервале $t \in [0; 100]$.

1 Построение математической модели

Будем считать контур идеальным, внешние поля отсутствуют, потенциал заземленного проводника равен 0, и ток земли равен 0.

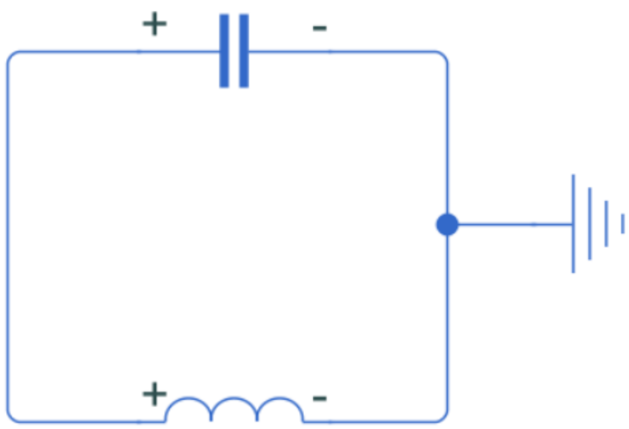

Рис. 1: LC-контур

Для построения дифференциального уравнения, описывающего колебания в контуре, будем использовать законы Кирхгофа.

Напряжение на конденсаторе $U_C = \frac{q}{C}$, где q - заряд на конденсаторе C.

ЭДС индукции катушки индуктивности $\mathcal{E}_L = -L \cdot \dot{I}$, где I - ток через индуктивность L.

В цепи имеется один контур:

$$-L \cdot \dot{I} = \frac{q}{C} \Rightarrow$$
$$\ddot{q} + \frac{1}{LC} q = 0$$

Соответствующее уравнение для напряжения на конденсаторе:

$$\ddot{U_C} + \frac{1}{LC} U_C = 0 \tag{1}$$

1.1 Аналитическое решение

Решение этого дифференциального уравнения с учетом начальных условий $U_C(0) = U_0 = 1$ В, $I(0) = 0 \Rightarrow \dot{U_C}(0) = 0$ представляет собой гармонические колебания с частотой $\omega = \dfrac{1}{\sqrt{LC}}$:

$$\begin{aligned} U_C(t) &= U_0 \cdot \cos(\omega t), \\ I(t) &= -U_0 \cdot \omega \cdot \sin(\omega t) \end{aligned} \tag{2}$$

Амплитуда колебаний напряжения на конденсаторе равна 1 В и амплитуда колебаний тока в контуре 1 А (так как круговая частота ω равна 1 c^{-1}). Период колебаний составляет примерно 6.28 c.

1.2 Численное решение

Одним из стандартных способов численного решения подобной задачи Коши является явный метод Рунге-Кутты четвертого порядка, опирающийся на итерационную формулу:

$$
\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{y}_n) \\
\mathbf{k}_2 &= \mathbf{f}(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2} \cdot \mathbf{k}_1) \\
\mathbf{k}_3 &= \mathbf{f}(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2} \cdot \mathbf{k}_2) \\
\mathbf{k}_4 &= \mathbf{f}(t_n + h, \mathbf{y}_n + h \cdot \mathbf{k}_3) \\
\mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{h}{6} \cdot (\mathbf{k}_1 + 2 \cdot \mathbf{k}_2 + 2 \cdot \mathbf{k}_3 + \mathbf{k}_4)
\end{aligned}
\tag{3}
$$

Эта схема вычислений применяется для численного решения системы дифференциальных уравнений первого порядка, заданных в виде:

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0 \tag{4}$$

Сведение дифференциального уравнения второго порядка (1) к системе двух дифференциальных уравнений первого порядка происходит следующим образом:

$$\mathbf{y} = \begin{Vmatrix} U_C(t) \\ \dot{U}_C(t) \end{Vmatrix}, \mathbf{f}(t, \mathbf{y}) = \begin{Vmatrix} y_2(t) \\ -\omega^2 \cdot y_1(t) \end{Vmatrix}$$

Программный код, реализующий данную схему вычислений написан на языке C++ (см. Приложение).

2 Результаты

Графики построены с помощью пакета matplotlib языка Python (код можно помотреть в Приложении).

На графиках 2 и 3 представлены зависимости искомых величин от времени при численном расчете.

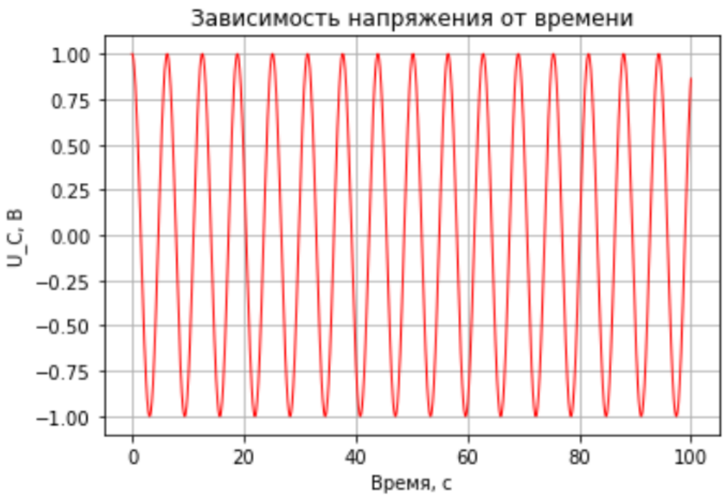

Рис. 2: График зависимости напряжения на конденсаторе от времени

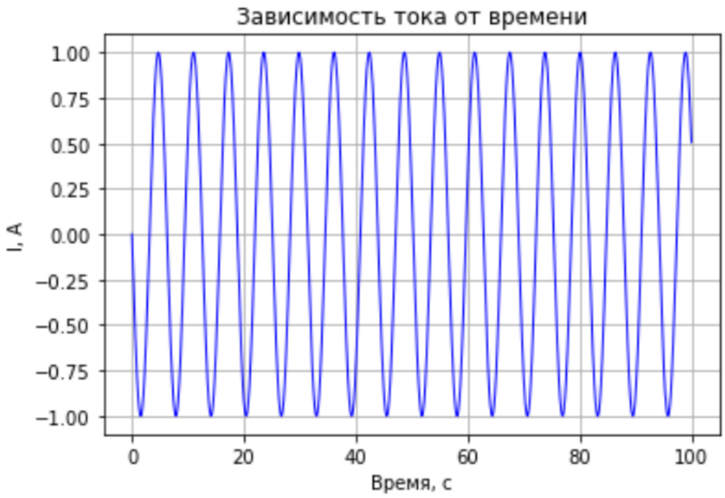

Рис. 3: График зависимости тока в цепи от времени

Численный расчет проводился на сетке размером 4000 точек на весь расчетный интервал. Разница аналитического решения и численного представлена на графике 4.

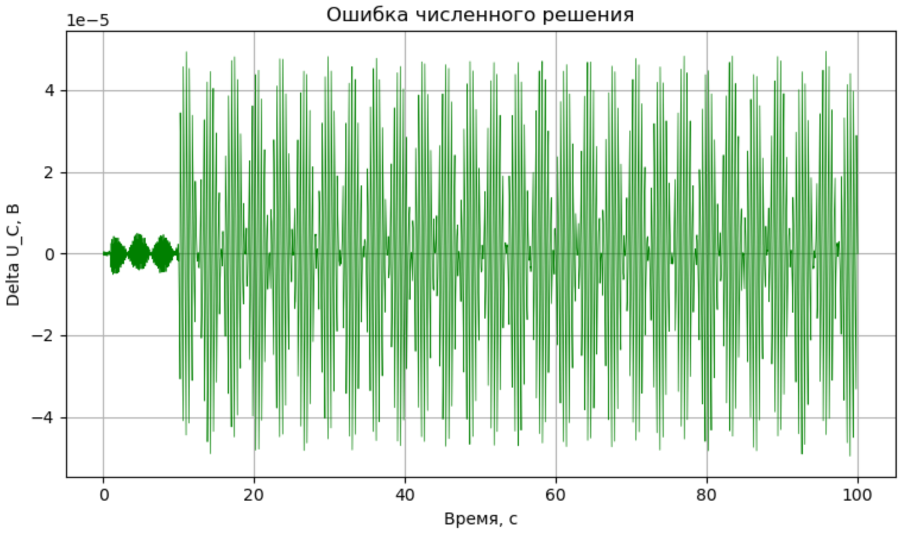

Рис. 4: Ошибка численного расчета

Видим, что ошибка составляет 0.005 процента. Как показали экспериметы, дальнейшее уменьшение шага по сетке не приведет к значительному повышению точности. Для увеличения точности можно использовать неявные методы Рунге-Кутты.

3 Приложение

3.1 Численный расчет на C++

```
#include <iostream>
#include <vector>
using namespace std;

double sq_w;

struct vector2D {
	double x1;
	double x2;
};

vector2D diff_func(vector2D& y, double& t) {
	vector2D dy;
	dy.x1 = y.x2;
	dy.x2 = -sq_w * y.x1;
	return dy;
}

vector<vector2D> method_runge_kutta(vector2D(*)(vector2D&, double&), vector<double>& t, vector2D& y_start) {
	vector<vector2D> y_rez(t.size());
	y_rez[0].x1 = y_start.x1;
	y_rez[0].x2 = y_start.x2;
	double step = t[1] - t[0];
	vector2D y_shift, k1, k2, k3, k4;
	double shift = step / 2;
	for (int i = 1; i < t.size(); ++i) {
		k1 = diff_func(y_rez[i - 1], t[i - 1]);
		double new_t = t[i - 1] + shift;
		y_shift.x1 = y_rez[i - 1].x1 + shift * k1.x1;
		y_shift.x2 = y_rez[i - 1].x2 + shift * k1.x2;
		k2 = diff_func(y_shift, new_t);
		y_shift.x1 = y_rez[i - 1].x1 + shift * k2.x1;
		y_shift.x2 = y_rez[i - 1].x2 + shift * k2.x2;
		k3 = diff_func(y_shift, new_t);
		new_t = t[i - 1] + step;
		y_shift.x1 = y_rez[i - 1].x1 + step * k3.x1;
		y_shift.x2 = y_rez[i - 1].x2 + step * k3.x2;
		k4 = diff_func(y_shift, new_t);
		y_rez[i].x1 = y_rez[i - 1].x1 + step / 6 * (k1.x1 + 2 * k2.x1 + 2 * k3.x1 + k4.x1);
		y_rez[i].x2 = y_rez[i - 1].x2 + step / 6 * (k1.x2 + 2 * k2.x2 + 2 * k3.x2 + k4.x2);
	}
	return y_rez;

}

int main()
{
	vector2D y_start;
	y_start.x1 = 1;
	y_start.x2 = 0;
	int N = 4000;
	double T = 100;
	double dt = T / (N-1);
	vector<double> t(N);
	for (int i = 0; i < N; ++i) {
		t[i] = i * dt;
	}
	sq_w = 1;
	vector2D(*ptr_f)(vector2D&, double&);
	ptr_f = diff_func;
	vector<vector2D> rez = method_runge_kutta(ptr_f, t, y_start);
	return 0;
}
```

3.2 Построение графиков на Python

```python
# метод Рунге-Кутты
from matplotlib import pyplot as plt
ar1 = []
ar2 = []
ar3 = []
with open("data_runge_kutta.txt", "r") as f:
    s = f.readlines()
for elem in s:
    a, b, c = elem.split()
    ar1.append(float(a))
    ar2.append(float(b))
    ar3.append(float(c))

plt.plot(ar1, ar2, "red", linewidth = 1)
plt.title('Зависимость напряжения от времени')
plt.xlabel('Время, с')
plt.ylabel('U_C, В')
#plt.plot(ar1, ar3, "b", linewidth = 1)
#plt.title('Зависимость тока от времени')
#plt.xlabel('Время, с')
#plt.ylabel('I, A')
plt.grid()
plt.show()
```

```python
import numpy as np
import matplotlib.pyplot as plt
ar1 = []
ar2 = []
ar3 = []
with open("data_runge_kutta.txt", "r") as f:
    s = f.readlines()
for elem in s:
    a, b, c = elem.split()
    ar1.append(float(a))
    ar2.append(float(b))
    ar3.append(float(c))
t_anal = ar1
y_anal = np.cos(t_anal)
t_rk = ar1
y_rk = ar2
plt.plot(t_rk, y_anal-y_rk, 'c', linewidth = 0.5)
plt.title('Ошибка численного решения')
plt.xlabel('Время, с')
plt.ylabel('Delta U_C, В')
plt.grid()
plt.show()
```