МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ЗВІТ

про виконання лабораторної роботи №5

з теми

ІНТЕГРУВАННЯ СИСТЕМ ДИФЕРЕНЦІАЛЬНИХ РІВНЯНЬ. РОЗРАХУНОК ПЕРЕХІДНОГО ПРОЦЕСУ ДЛЯ RLC-ЛАНОК.

з навчальної дисципліни: “Чисельні методи”

Виконав:

студент групи ІР-24

Шийка Андрій

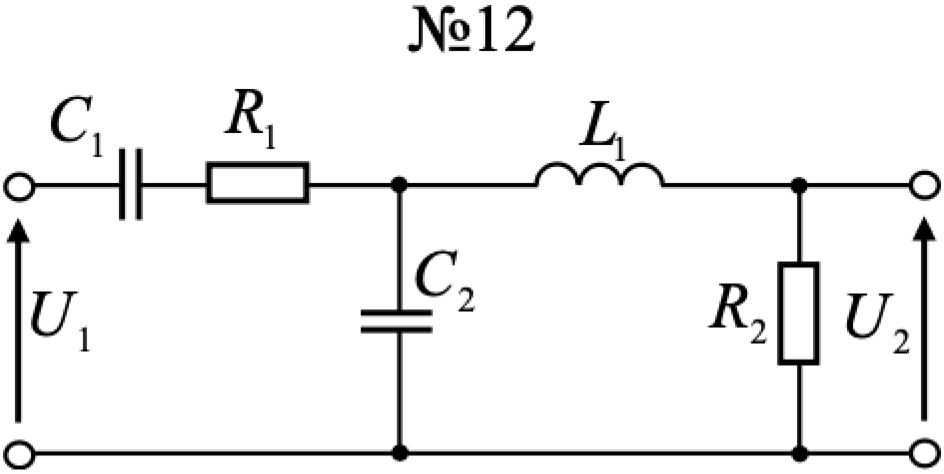
Прийняв:

Сиротюк С. В.

Львів - 2025

Мета роботи: Вивчити основні методи розв'язування систем диференціальних рівнянь першого порядку (методи Ейлера) та застосувати їх для розрахунку перехідних процесів у RLC-ланках.

Завдання:



Для заданої RCL-ланки скласти систему диференціальних рівнянь.

Виконати розрахунок перехідного процесу вихідної напруги U_2 методами:

- Явним методом Ейлера;
- Модифікованим методом Ейлера;
- Неявним методом Ейлера.

Побудувати графік перехідного процесу.

Короткі теоретичні відомості:

Задача Коші полягає у знаходженні розв'язку системи диференціальних рівнянь dx/dt = f(x, t), що задовольняє початковим умовам x(t0) = x0.

1. ***Явний метод Ейлера***

   Це найпростіший однокроковий метод. Значення функції в наступній точці обчислюється на основі похідної в поточній точці:

   x[n+1] = x[n] + h * f(x[n], t[n])

   Має похибку порядку O(h). Метод простий у реалізації, але має низьку точність і може бути не стійким.

2. ***Модифікований метод Ейлера***

Метод другого порядку точності. Використовує прогнозування (предиктор) і корекцію (коректор).

Спочатку робиться "півкрок" або прогноз:

x_progn = x[n] + h * f(x[n], t[n])

Потім обчислюється середнє значення похідної на початку і в кінці інтервалу:

x[n+1] = x[n] + (h / 2) * (f(x[n], t[n]) + f(x_progn, t[n+1]))

3. ***Неявний метод Ейлера***

   Шукане значення x[n+1] входить у праву частину рівняння:

   x[n+1] = x[n] + h * f(x[n+1], t[n+1])

   Для знаходження x[n+1] необхідно розв'язувати систему алгебраїчних (часто нелінійних) рівнянь на кожному кроці. Метод є безумовно стійким, що дозволяє використовувати його для жорстких систем.

Аналітичний вивід рівнянь:

Змінні стану:

- x1 = uC1 (напруга на конденсаторі C1)

- x2 = uC2 (напруга на конденсаторі C2)

- x3 = iL1 (струм через котушку L1 та резистор R2)

Закони Кірхгофа:

1. *Рівняння для вузла (точка з'єднання R1, C2 та гілки L1):*

   Струм i1, що втікає через R1, розгалужується на струм через C2 (iC2) та струм через індуктивність (iL1):

   i1 = iC2 + iL1

   Звідси струм конденсатора C2:

   iC2 = i1 - iL1

   Струм i1 за законом Ома:

   i1 = (U1(t) - uC1 - uC2) / R1

2. *Рівняння для елементів:*

   Для C1: i1 = C1 * (duC1/dt)

   Для C2: iC2 = C2 * (duC2/dt)

   Для гілки L1-R2: Напруга на цій гілці дорівнює напрузі на C2 (паралельне з'єднання).

   uC2 = uL1 + uR2 = L1 * (diL1/dt) + iL1 * R2

3. *Система диференціальних рівнянь у формі Коші:*

   (1) dx1/dt = i1 / C1 де i1 = (U_max*sin(wt) - x1 - x2) / R1

   (2) dx2/dt = (i1 - x3) / C2

   (3) dx3/dt = (x2 - x3 * R2) / L1

Вихідна напруга U2 знімається з резистора R2: U2 = iL1 * R2 = x3 * R2

Список ідентифікаторів констант, змінних, функцій:

Константи:

- U_MAX: Амплітуда вхідної напруги (100 В)

- F: Частота мережі (50 Гц)

- R1, R2: Опори резисторів (Ом)

- L1: Індуктивність (Гн)

- C1, C2: Ємності конденсаторів (Ф)

- H: Крок інтегрування (0.00001 с)

- T_MAX: Час моделювання (0.2 с)

Змінні:

- t: Поточний час

- x: Масив змінних стану [uC1, uC2, iL]

- derivatives: Масив похідних [duC1/dt, duC2/dt, diL/dt]

- u2: Вихідна напруга

- data_expl, data_mod, data_impl: Списки для збереження результатів розрахунку відповідними методами

Вхідні дані:

$U_1 = U_{max} \sin(2\pi f t), U_{max} = 100$ В, $f = 50$ Гц.

Елементи схеми: $R_1 = 5$ Ом, $R_2 = 4$ Ом, $L_1 = 0.01$ Гн, $C_1 = 300$ мкФ, $C_2 = 150$ мкФ.

Параметри інтегрування: $t_{int} = 0.2$ с, крок $h = 0.00001$ с.

Код програми:

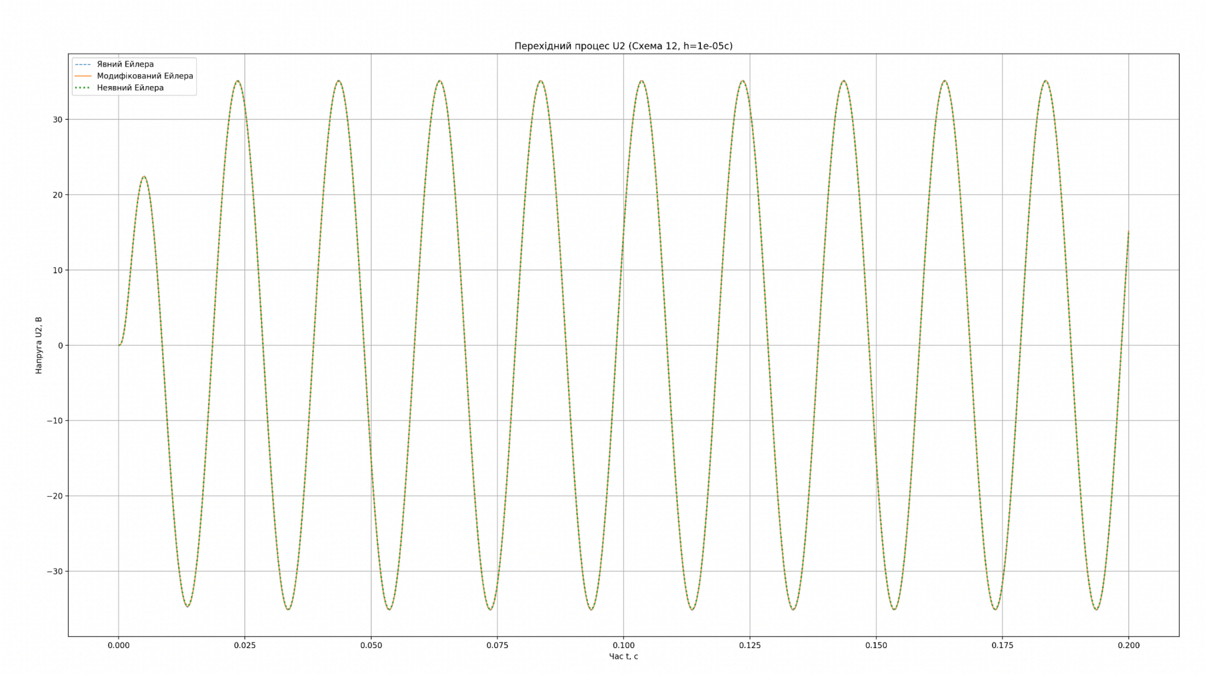
```
import math
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fsolve

U_MAX = 100.0         # В
F = 50.0              # Гц
R1 = 5.0              # Ом
R2 = 4.0              # Ом
L1 = 0.01             # Гн
C1 = 300e-6           # Ф
C2 = 150e-6           # Ф
T_MAX = 0.2           # с
H = 0.00001           # крок інтегрування (10 мкс)

# Початкові умови: [u_c1, u_c2, i_l]
X0 = [0.0, 0.0, 0.0]

def get_u1(t):
   return U_MAX * math.sin(2 * math.pi * F * t)

def get_derivatives(t, x):
   """
   x[0] -> u_c1
   x[1] -> u_c2
```

```
    x[2] -> i_l
    """
    u_c1, u_c2, i_l = x
    u1_val = get_u1(t)

    # Струм через R1
    i1 = (u1_val - u_c1 - u_c2) / R1

    # Система диференціальних рівнянь
    d_uc1 = i1 / C1
    d_uc2 = (i1 - i_l) / C2
    d_il  = (u_c2 - i_l * R2) / L1

    return np.array([d_uc1, d_uc2, d_il])

def solve_explicit_euler():
    t = 0.0
    x = np.array(X0)
    time_vals = []
    u2_vals = []

    while t <= T_MAX:
        u2 = x[2] * R2
        time_vals.append(t)
        u2_vals.append(u2)

        x = x + H * get_derivatives(t, x)
        t += H
    return time_vals, u2_vals

def solve_modified_euler():
    t = 0.0
    x = np.array(X0)
    time_vals = []
    u2_vals = []
```

```
    while t <= T_MAX:
        u2 = x[2] * R2
        time_vals.append(t)
        u2_vals.append(u2)

        k1 = get_derivatives(t, x)
        x_predict = x + H * k1
        k2 = get_derivatives(t + H, x_predict)

        x = x + (H / 2.0) * (k1 + k2)
        t += H
    return time_vals, u2_vals

def solve_implicit_euler():
    t = 0.0
    x = np.array(X0)
    time_vals = []
    u2_vals = []

    while t <= T_MAX:
        u2 = x[2] * R2
        time_vals.append(t)
        u2_vals.append(u2)

        t_next = t + H

        # Рівняння: x_new - x_old - h * f(x_new) = 0
        func = lambda x_new: x_new - x - H * get_derivatives(t_next,
x_new)

        x = fsolve(func, x)
        t = t_next
    return time_vals, u2_vals
```

```python
if __name__ == "__main__":
    print("Розрахунок перехідного процесу для Варіанту 12 (Схема 12)...")

    # Розрахунок трьома методами
    t_expl, u_expl = solve_explicit_euler()
    t_mod, u_mod = solve_modified_euler()
    t_impl, u_impl = solve_implicit_euler()

    # Побудова графіків
    plt.figure(figsize=(10, 6))
    plt.plot(t_expl, u_expl, label='Явний Ейлера', linestyle='--', linewidth=1)
    plt.plot(t_mod, u_mod, label='Модифікований Ейлера', linestyle='-', alpha=0.7)
    # Неявний графік може зливатися з модифікованим через високу точність
    plt.plot(t_impl, u_impl, label='Неявний Ейлера', linestyle=':', linewidth=2)

    plt.title(f"Перехідний процес U2 (Схема 12, h={H}c)")
    plt.xlabel("Час t, c")
    plt.ylabel("Напруга U2, В")
    plt.legend()
    plt.grid(True)
    plt.savefig('./result.png')
    plt.show()

    print("Графік побудовано. Кінцеве значення U2 (Mod): {:.4f} В".format(u_mod[-1]))
```

Результат:



Висновок:

В ході виконання лабораторної роботи було розраховано перехідний процес для RLC-ланки при підключенні до джерела синусоїдальної напруги.

1. Математична модель: Складено систему з трьох диференціальних рівнянь першого порядку на основі законів Кірхгофа.
2. Реалізація: Програма реалізує три числові методи.
   a. Явний метод Ейлера найшвидший у розрахунку, але накопичує похибку.
   b. Модифікований метод дав більш точний результат завдяки корекції нахилу.
   c. Неявний метод потребував використання солвера (fsolve) для розв'язання рівнянь на кожному кроці, але гарантує стійкість.
3. Результати: Графіки вихідної напруги U2, отримані різними методами, співпадають візуально при малому кроці інтегрування (h=10^-5), що свідчить про правильність роботи алгоритмів. Перехідний процес має коливальний характер, що пояснюється наявністю реактивних елементів (L та C) у колі.