"""

Grupp 9

AU2

"""

import numpy as np

import matplotlib.pyplot as plt

def uppgift1 ():

    # Parametrar för RLC-kretsen

    #R = 10  # Resistans (Ω)

    #C = 65e-6  # Kapacitans (F)

    #L = 53e-3  # Induktans (H)

    #U0 = 1  # Spänning från källan (V)

    omega = 800  # Vinkelfrekvens (rad/s)

    # Beräkna fasvinkeln

    XC = 1 / (omega \* C)

    XL = omega \* L

    phi = np.arctan((XC - XL) / R)

    # Beräkna överföringsfunktionen Z

    Z = np.sqrt(R\*\*2 + (1 / (omega \* C) - omega \* L)\*\*2)

    # Beräkna strömmen I(t) med differentialekvationen

    I0 = U0 / Z

    # Steglängd och tidsvektor

    dt = 0.000001

    tmax = 0.08

    t = np.arange(0, tmax, dt)

    dim = len(t)

    # Initialisera vektorer

    I\_numerical = np.zeros(dim)

    I\_derivata = np.zeros(dim)

    I\_2derivata = np.zeros(dim)

    # Begynnelsevillkor

    I\_numerical[0] = I0

    I\_derivata[0] = -omega \* U0 \* np.cos(omega \* t[0]) / R

    # Numerisk lösning med Eulers metod

    for i in range(dim-1):

    # dIdt = (U0 - I\_numerical[i] \* R) / L - I\_numerical[i] / (R \* C)

    # I\_numerical[i+1] = I\_numerical[i] + dIdt \* dt

        I\_2derivata[i] = (U0 \* omega \* np.cos(omega \* t[i]) - R \* I\_derivata[i] - I\_numerical[i] / C) / L

        I\_derivata[i+1] = I\_derivata[i] + I\_2derivata[i] \* dt

        I\_numerical[i+1] = I\_numerical[i] + I\_derivata[i] \* dt

    # Plotta de olika kurvorna

    plt.figure(figsize=(8, 6))

    plt.plot(t, Z \* I0 \* np.sin(omega \* t + phi), label="Z · i(t) analytisk", linestyle="--", color="black")

    plt.plot(t, Z\* I\_numerical, label="Z · i(t) numerisk", color="orange")

    plt.plot(t, U0 \* np.sin(omega \* t), label="U(t)", color="blue")

    plt.xlabel("Tid (s)")

    plt.ylabel("Värden")

    plt.title("Överföringsfunktion och spänning i en RLC-krets")

    plt.ylim(-3, 2.3)

    plt.yticks([-3, -2, -1, 0, 1, 2], ["-3", "-2", "-1", "0", "1", "2"])

    plt.legend()

    plt.grid(True)

    plt.show()

    # Skriv ut fasvinkeln för strömmen

    print(f"Fasvinkeln φ ≈ {np.degrees(phi):.2f} grader")

def uppgift2():

   # Parametrar för RLC-kretsen

    #R = 10  # Resistans (Ω)

    #C = 65e-6  # Kapacitans (F)

    #L = 53e-3  # Induktans (H)

    #U0 = 1  # Spänning från källan (V)

    omega = 800  # Vinkelfrekvens (rad/s) w

    TFV = []

    omega\_values = np.linspace(0.0, 3.0, 1000) \* omega

    for omega in omega\_values:

        Z = np.sqrt(R \*\* 2 + (1 / (omega \* C) - omega \* L) \*\* 2) # Impedans

        H = (omega \* L / Z)

        TFV.append(H)

    NF = omega\_values / 540

    plt.plot(NF, TFV)

    plt.xlim(0.00 , 3.00)

    plt.ylim (0.00, 3.00)

    plt.xlabel("w / w0")

    plt.ylabel("H(w)")

    plt.title("Överföringsfunktion för spole")

    plt.show()

def uppgift3():

    #L = 53e-3

    #C = 65e-6

    omega\_0 = 1 / np.sqrt(L \* C)  # Resonance frequency (rad/s)

    # Dynamic range for testing

    omega\_values = np.linspace(0.1 \* omega\_0, 10 \* omega\_0, 500)

    # Initialize variables to track the best performance

    best\_R = None

    lowest\_variance = np.inf

    best\_H = []

    # loop to test differant R to find the optimal one

    for R in np.linspace(1, 100, 100):

        H\_values = [omega \* L / np.sqrt(R\*\*2 + (1 / (omega \* C) - omega \* L)\*\*2) for omega in omega\_values]

        variance = np.var(H\_values)  # Calculate variance as a smoothness measure

        if variance < lowest\_variance:

            lowest\_variance = variance

            best\_R = R

            best\_H = H\_values  # Best H to be saved

        # Debug output to track progress

        print(f"Testing R={R:.2f} Ohm, Variance={variance:.4f}") # can be removed if dont wanna see in terminal

    # if sats to check if best\_R was set and plot the result

    if best\_R is not None:

        # Plot the optimized transfer function

        plt.figure(figsize=(10, 6))

        plt.plot(omega\_values / omega\_0, best\_H)

        plt.xlabel('$\omega / \omega\_0$')

        plt.ylabel('$H(\omega)$')

        plt.title('Optimized Transfer Function for an RLC Circuit')

        plt.grid(True)

        plt.legend()

        plt.show()

        print(f"Optimal Resistance for Smoothness: {best\_R:.2f} Ohm")

    else:

        print("No optimal resistance found.")

def configure():

    global R, C, L, U0, omega

    R = 10  # Resistans (Ω)

    C = 65e-6  # Kapacitans (F)

    L = 53e-3  # Induktans (H)

    U0 = 1  # Spänning från källan (V)

    omega = 800  # Vinkelfrekvens (rad/s) w

def configureOwn():

    print()

def runAlone():

    choice = input("Välj mellan uppgift 1/2/3\n ")

    if choice == "1":

        uppgift1()

    elif choice == "2":

        uppgift2()

    elif choice == "3":

        uppgift3()

    else:

        print("1, 2, 3 ?:\n")

def chooseOwn ():

    choice = input("Välj mellan uppgift 1/2/3\n ")

    global R, C, L, U0, omega

    if choice == "1":

        rinput=float(input("Hur mycket R\n"))

        R = rinput

        cinput= float(input("Hur mycket C\n"))

        C = cinput

        linput= float( input ("Hur mycket L\n"))

        L = linput

        U0input= float(input ("Hur mycket U0\n"))

        U0 = U0input

        uppgift1()

    elif choice == "2":

        rinput=float(input("Hur mycket R\n"))

        R = rinput

        cinput= float(input("Hur mycket C\n"))

        C = cinput

        linput= float( input ("Hur mycket L\n"))

        L = linput

        U0input= float(input ("Hur mycket U0\n"))

        U0 = U0input

        uppgift2()

    elif choice == "3":

        cinput= float(input("Hur mycket C\n"))

        C = cinput

        linput= float( input ("Hur mycket L\n"))

        L = linput

        U0input= float(input ("Hur mycket U0\n"))

        U0 = U0input

        uppgift3()

    else:

        print("1, 2, 3 ?:\n")

def main():

    choose = input ("Vill du välja värden själv. ja/nej\n")

    if choose == "ja":

        chooseOwn()

    elif choose == "nej":

        configure()

        runAlone()

if \_\_name\_\_ == "\_\_main\_\_":

    main()