Faculty of Engineering & Technology

Electrical & Computer Engineering Department

ENCS 4380

Interfacing Techniques

Modeling of System

HW 2

Student Name: Hamza Al Shaer

Student ID: 1211162

Instructor: Dr. Wasel Ghanem

Date: 27/10/2024

Section: 2

❖ Part One: System Analysis and Results

This section presents an analysis of the behavior of the spring-mass and RLC systems under different damping conditions (overdamped, critically damped, and underdamped) with various input types (unit step, unit ramp, and sinusoidal). The results include plots of system responses and a comparative analysis to understand the impact of each damping condition and input type.

- ### Spring-Mass System Analysis

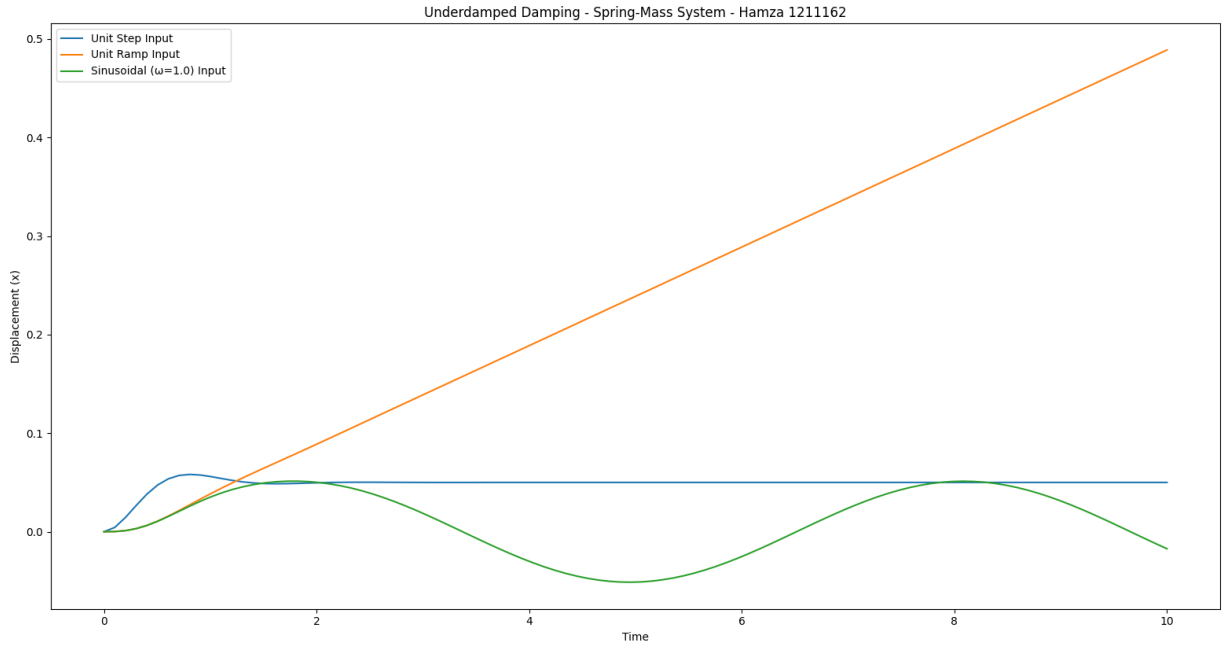

Figure 1: Critically Damped Response - Spring-Mass System

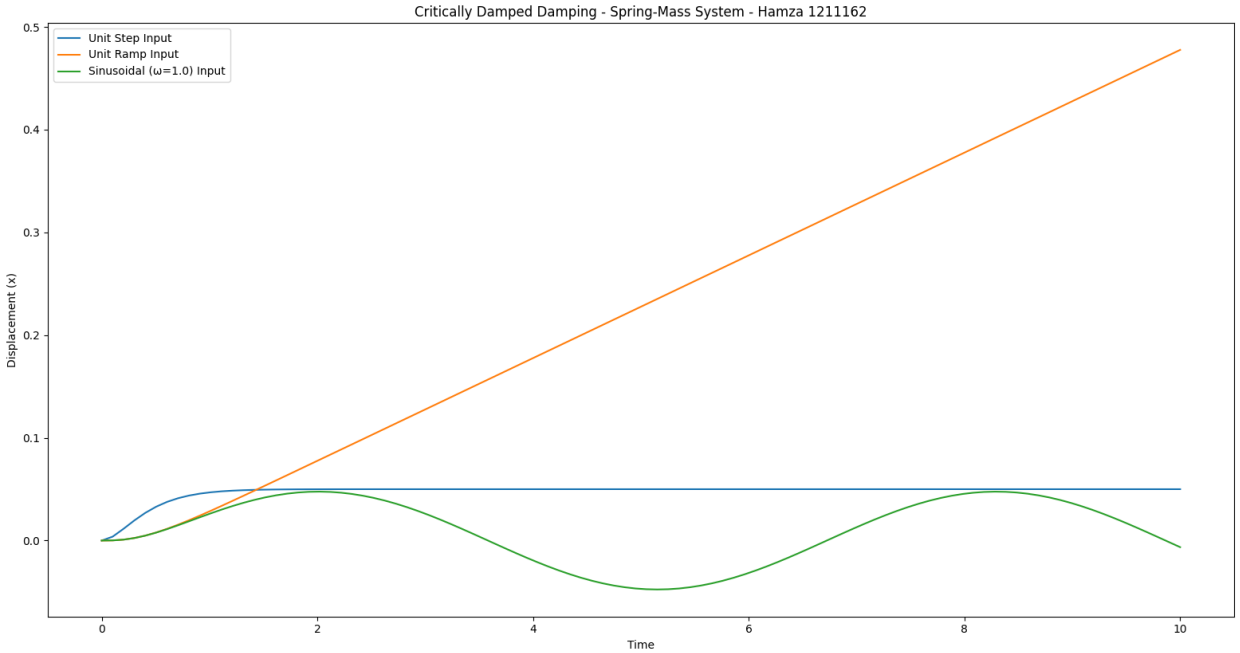

Figure 2: Underdamped Response - Spring-Mass System

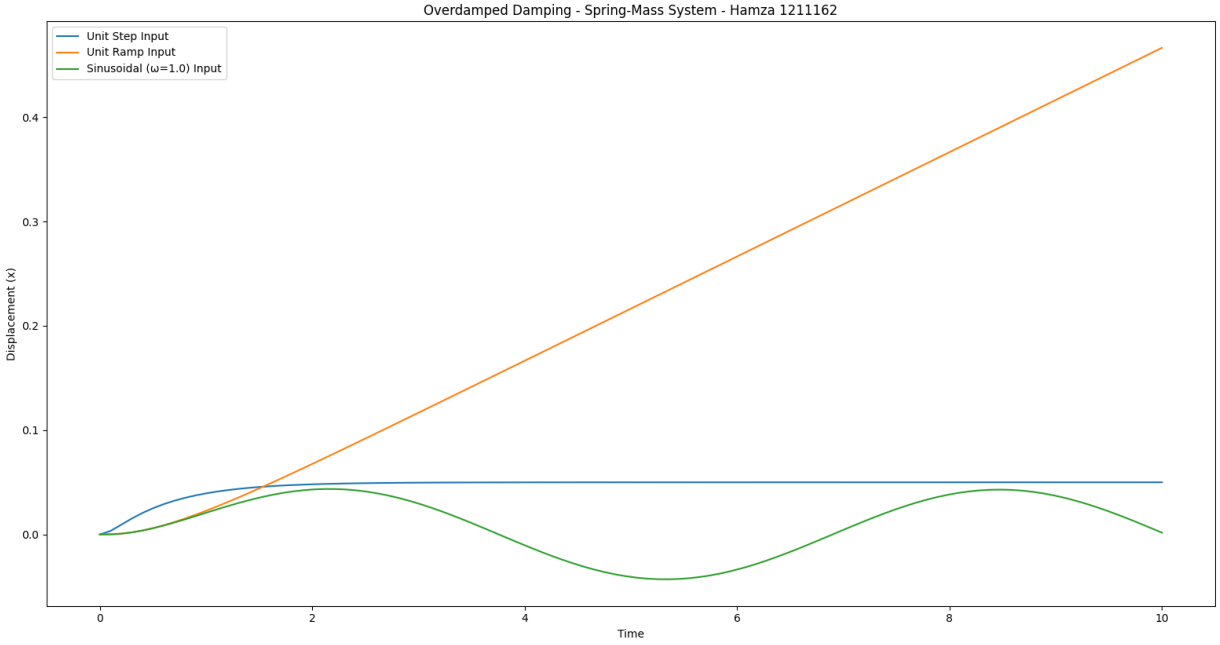

Figure 3: Overdamped Response - Spring-Mass System

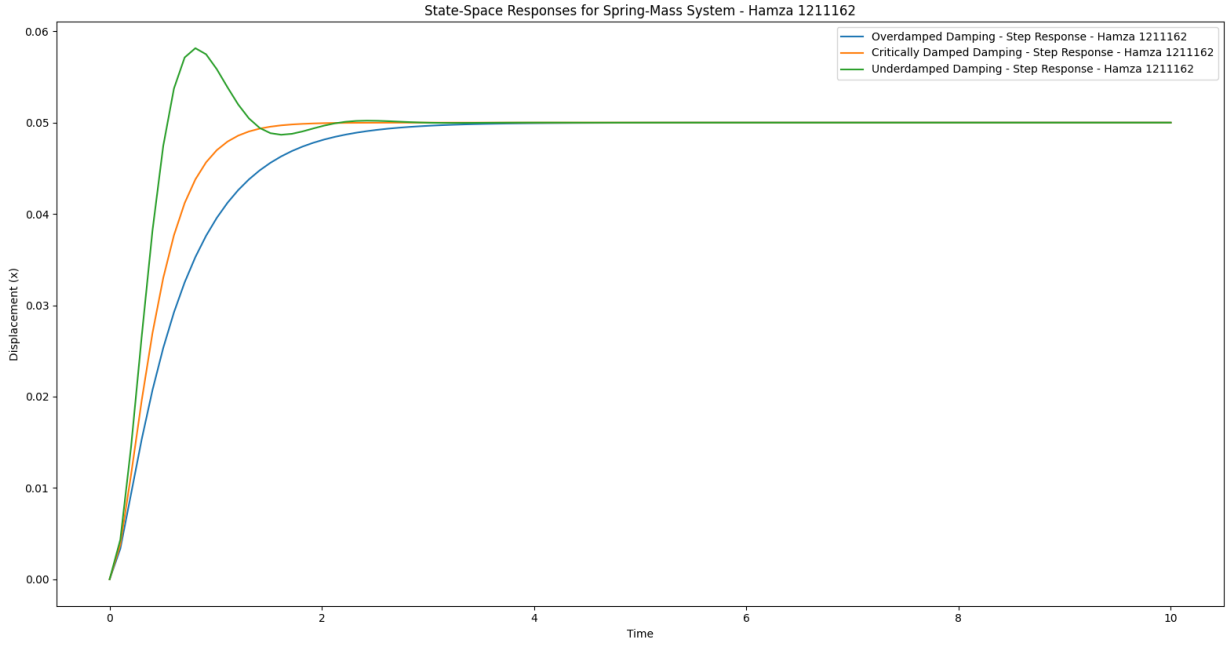

Figure 4: State-Space Comparison - Spring-Mass System

- ## Spring-Mass System Output Analysis

1. **Critically Damped Response**: The critically damped response of the spring-mass system shows a smooth, non-oscillatory approach to the steady state. This configuration allows the system to reach equilibrium as quickly as possible without overshooting.

2. **Underdamped Response**: In the underdamped case, the system exhibits oscillatory behavior as it approaches the steady state, which is typical in lightly damped systems. The system oscillates before gradually stabilizing.

3.**Overdamped Response**: The overdamped response is the slowest, with no oscillations, as the high damping level prevents rapid movement. This results in a slower approach to the steady state.

4. **State-Space Comparison**: The state-space plot shows a comparison of the step responses across different damping conditions, with the critically damped case reaching stability fastest, followed by the overdamped and underdamped cases.

- ## RLC Circuit Analysis

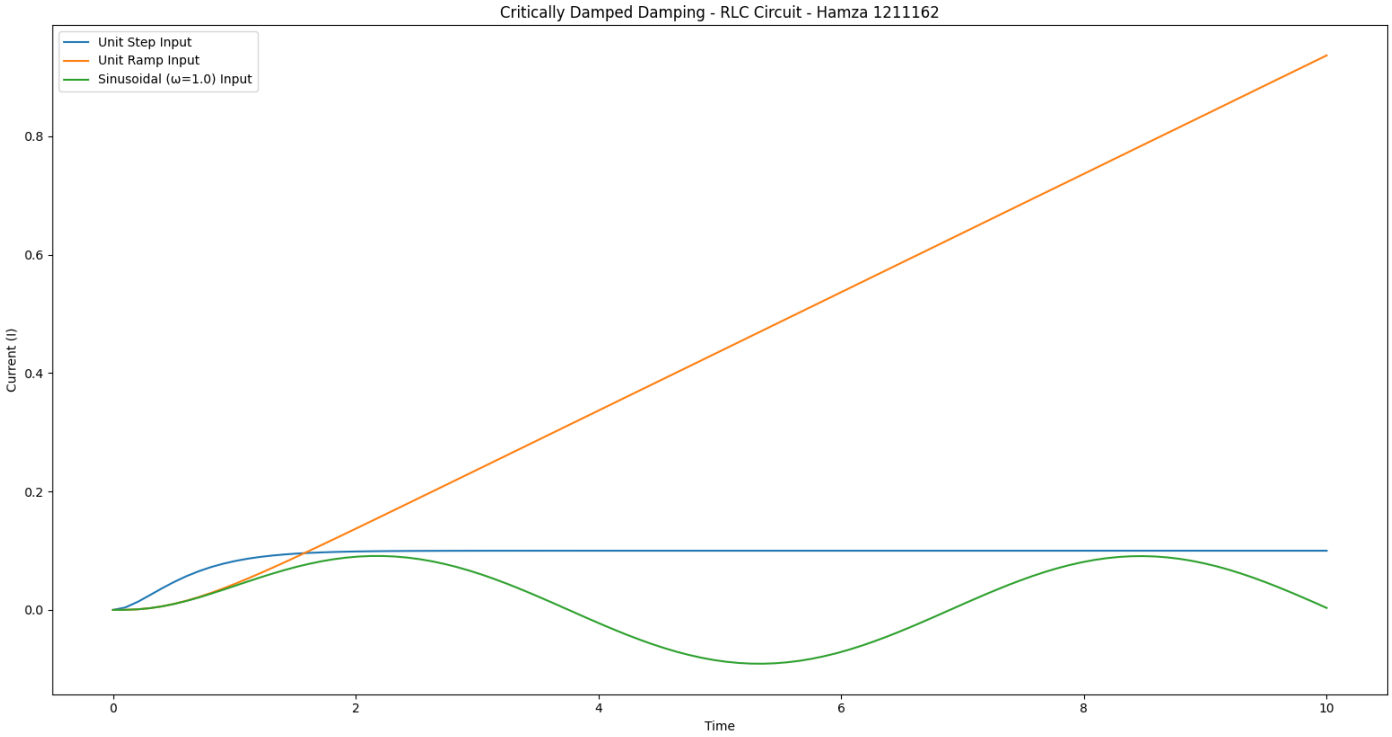

Figure 5: Critically Damped Response - RLC Circuit

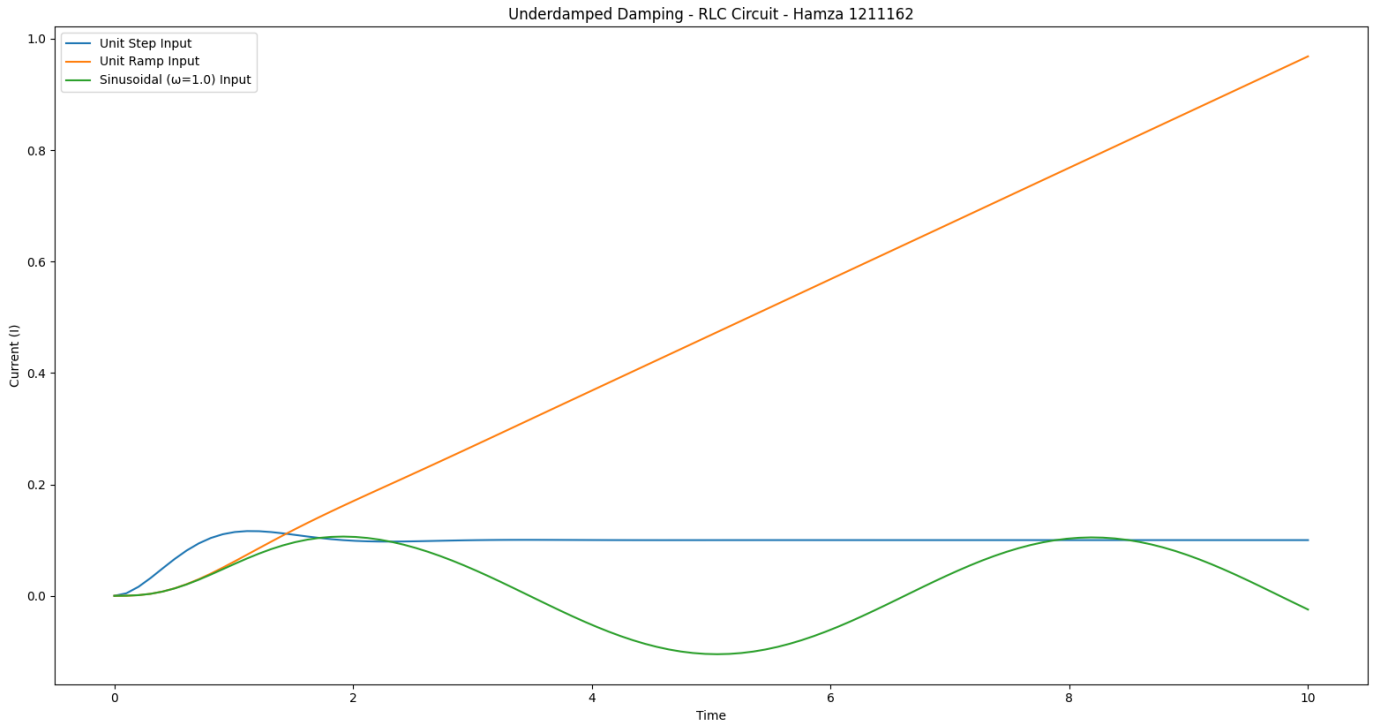

Figure 6: Underdamped Response - RLC Circuit

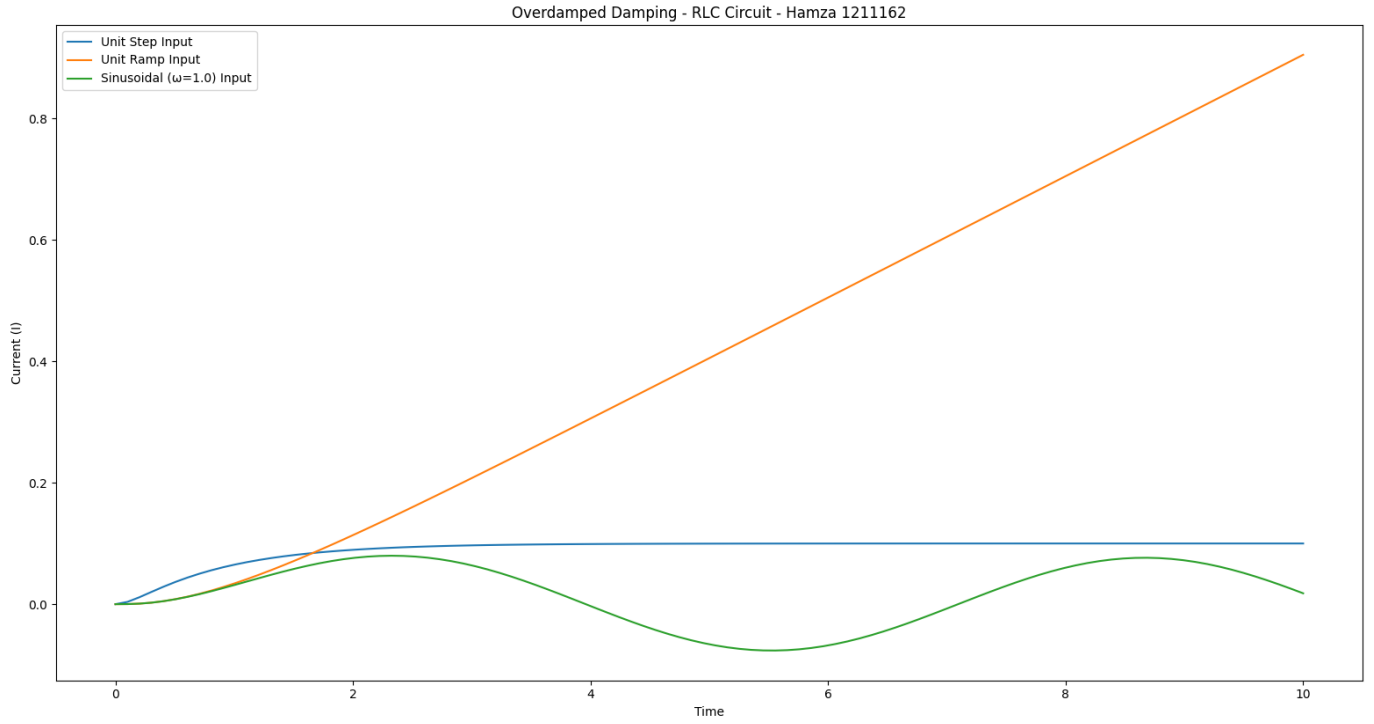

Figure 7: Overdamped Response - RLC Circuit

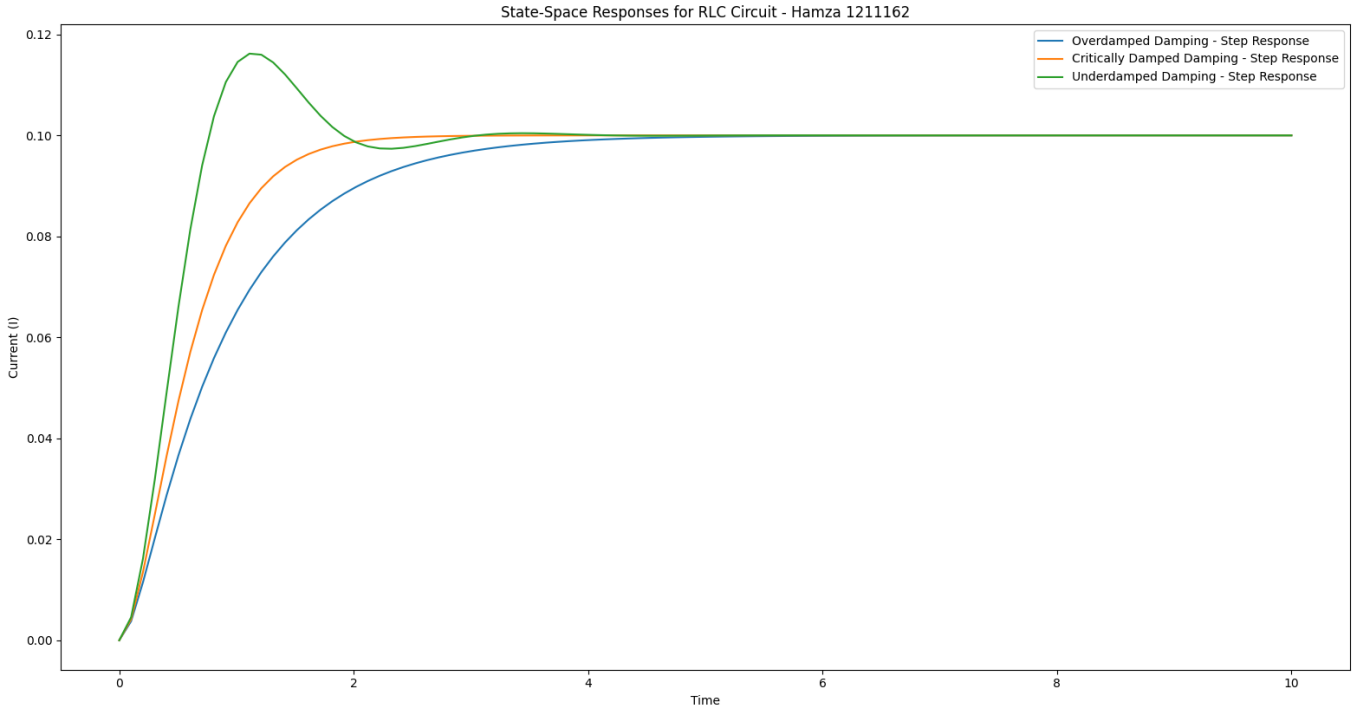

Figure 8: State-Space Comparison - RLC Circuit

- ## RLC System Output Analysis

1. **Critically Damped Response**: Similar to the spring-mass system, the critically damped RLC circuit response reaches a steady state without oscillations. This setting provides the fastest response time without overshooting.

2. **Underdamped Response**: The underdamped RLC circuit shows oscillations, resonating with the natural frequency of the circuit, and eventually settles to a steady state.

3. **Overdamped Response**: The overdamped response is gradual and non-oscillatory, as the high resistance dampens the system significantly, resulting in a slower approach to equilibrium.

4. **State-Space Comparison**: The state-space plot for the RLC circuit provides a comparative view of the system's response under different damping conditions. The critically damped circuit stabilizes the fastest, while the overdamped and underdamped circuits take longer to reach equilibrium.

- **Appendix: Python Code**

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
from scipy.signal import lti, step, lsim
from sympy import symbols, Function, Eq, dsolve, laplace_transform, Heaviside,
sin
from sympy.abc import t, s
from sympy import Rational, simplify
from scipy.signal import StateSpace, step

# example form me
m = 1.0  # Mass for spring-mass system
k = 20.0  # Spring constant for spring-mass system
L = 1.0  # Inductance for RLC circuit
C = 0.1  # Capacitance for RLC circuit

b_critical = 2 * np.sqrt(m * k)  # Critical damping (ξ = 1)
b_over = b_critical * 1.5  # Over damped (ξ > 1)
b_under = b_critical * 0.5  # Under damped (ξ < 1)
b_values = {'Overdamped': b_over, 'Critically Damped': b_critical, 'Underdamped':
b_under}

R_critical = 2 * np.sqrt(L / C)  # Critical damping for RLC circuit
R_over = R_critical * 1.5  # Over damped for RLC
R_under = R_critical * 0.5  # Under damped for RLC
R_values = {'Overdamped': R_over, 'Critically Damped': R_critical, 'Underdamped':
R_under}

# Time range for simulation & Time evaluation points
t_span = (0, 10)
t_eval = np.linspace(*t_span, 100)


# Define each input type
def unit_step(t):
    return 1

def unit_ramp(t):
    return t

def sinusoidal(t, omega=1.0):
    return np.sin(omega * t)

# Spring-Mass System ODE definition with force function
def spring_mass_system(t, y, m, b, k, F_func):
    x, v = y  # x is displacement, v is velocity
    F = F_func(t)  # Evaluate the force function at time t
    dxdt = v
    dvdt = (F - b * v - k * x) / m
    return [dxdt, dvdt]

# RLC Circuit ODE definition with voltage function
def rlc_circuit_system(t, y, L, R, C, V_func):
    I, dI_dt = y  # I is current, dI_dt is rate of change of current
    V_prime = V_func(t)  # Evaluate the voltage function at time t
    d2I_dt2 = (V_prime - R * dI_dt - (1 / C) * I) / L
    return [dI_dt, d2I_dt2]


# Simulate Spring-Mass System for Each Damping Condition and Input Type
for damping_type, b in b_values.items():
    print(f"--- Spring-Mass System: {damping_type} Damping ---")
```

```
    for F_func, label in [(unit_step, 'Unit Step'), (unit_ramp, 'Unit Ramp'),
                          (lambda t: sinusoidal(t, omega=1.0), 'Sinusoidal
(ω=1.0)')]:
        solution = solve_ivp(spring_mass_system, t_span, [0.0, 0.0], args=(m, b,
k, F_func), t_eval=t_eval)

        plt.plot(solution.t, solution.y[0], label=f"{label} Input")
    plt.xlabel('Time')
    plt.ylabel('Displacement (x)')
    plt.title(f"{damping_type} Damping - Spring-Mass System - Hamza 1211162")
    plt.legend()
    plt.show()

# Simulate RLC Circuit for Each Damping Condition and Input Type
for damping_type, R in R_values.items():
    print(f"--- RLC Circuit: {damping_type} Damping ---")

    for V_func, label in [(unit_step, 'Unit Step'), (unit_ramp, 'Unit Ramp'),
                          (lambda t: sinusoidal(t, omega=1.0), 'Sinusoidal
(ω=1.0)')]:
        solution_rlc = solve_ivp(rlc_circuit_system, t_span, [0.0, 0.0], args=(L,
R, C, V_func), t_eval=t_eval)

        plt.plot(solution_rlc.t, solution_rlc.y[0], label=f"{label} Input")
    plt.xlabel('Time')
    plt.ylabel('Current (I)')
    plt.title(f"{damping_type} Damping - RLC Circuit - Hamza 1211162")
    plt.legend()
    plt.show()

# Laplace Domain Transfer Functions (Symbolic)
s = symbols('s')
X, F = symbols('X F')
omega_n = np.sqrt(k / m)  # Natural frequency for spring-mass

# Transfer Function for Spring-Mass System
H_s_spring_mass = 1 / (m * s**2 + b_critical * s + k)
simplified_H_s_spring_mass = simplify(H_s_spring_mass)
print(f"Spring-Mass System Transfer Function: {simplified_H_s_spring_mass} -
Hamza 1211162")

# Transfer Function for RLC Circuit
H_s_rlc = 1 / (L * s**2 + R_critical * s + (1 / C))
simplified_H_s_rlc = simplify(H_s_rlc)
print(f"RLC Circuit Transfer Function: {simplified_H_s_rlc} - Hamza 1211162")

# Spring-Mass System State-Space
for damping_type, b in b_values.items():
    A_spring_mass = np.array([[0, 1], [-k / m, -b / m]])
    B_spring_mass = np.array([[0], [1 / m]])
    C_spring_mass = np.array([[1, 0]])
    D_spring_mass = np.array([[0]])

    system_spring_mass = StateSpace(A_spring_mass, B_spring_mass, C_spring_mass,
D_spring_mass)

    t, y = step(system_spring_mass, T=t_eval)
    plt.plot(t, y, label=f"{damping_type} Damping - Step Response - Hamza
1211162")

plt.xlabel('Time')
plt.ylabel('Displacement (x)')
plt.title("State-Space Responses for Spring-Mass System - Hamza 1211162")
plt.legend()
plt.show()
```

```
# RLC Circuit State-Space
for damping_type, R in R_values.items():
    A_rlc = np.array([[0, 1], [-1 / (L * C), -R / L]])
    B_rlc = np.array([[0], [1 / L]])
    C_rlc = np.array([[1, 0]])
    D_rlc = np.array([[0]])

    system_rlc = StateSpace(A_rlc, B_rlc, C_rlc, D_rlc)
    t, y = step(system_rlc, T=t_eval)
    plt.plot(t, y, label=f"{damping_type} Damping - Step Response")

plt.xlabel('Time')
plt.ylabel('Current (I)')
plt.title("State-Space Responses for RLC Circuit - Hamza 1211162")
plt.legend()
plt.show()
```

❖ Part Two: Summary of State-Space Modeling Assignment

This assignment explores **dynamic system modeling** and **analysis techniques** using a **spring-mass-damper** system and an **RLC circuit**. Through state-space representation, Laplace transforms, and time-domain solutions, it demonstrates how these techniques reveal system behavior under various damping conditions and input types. This analysis provides insights into system stability, transient response, and steady-state behavior.

1. **Mathematical Foundations and Modeling:**
    - **Differential Equations**:

      Each system's physical dynamics are described by second-order differential equations. For the spring-mass system, Newton's second law is applied to derive the motion equation, where the force balance includes mass mmm, damping coefficient bbb, and spring constant kkk. Similarly, Kirchhoff's voltage law is applied to derive the differential equation for the RLC circuit, involving inductance LLL, resistance RRR, and capacitance CCC.

    - **Damping Conditions**:

      Three damping cases (overdamped, critically damped, and underdamped) are analyzed for each system. These conditions determine how quickly and smoothly the system returns to equilibrium after being disturbed:
        - **Overdamped**: Slow response without oscillations.
        - **Critically Damped**: Fastest non-oscillatory response.
        - **Underdamped**: Oscillatory response that gradually decays to equilibrium.

2. **Time-Domain Analysis**:

- **Numerical Solutions**: Time-domain responses are computed numerically using techniques like **Euler's method** and **Runge-Kutta methods** (ODE solvers). These methods approximate the system's behavior by discretizing the differential equations over small time intervals, revealing transient and steady-state responses.
- **Input Types and Responses**: The system is subjected to different types of inputs—unit step, ramp, and sinusoidal—to observe its behavior:
    - **Unit Step**: Provides insight into how the system initially responds and stabilizes over time.
    - **Ramp Input**: Reveals the system's ability to track a continuously increasing input, highlighting its response speed and damping effectiveness.
    - **Sinusoidal Input**: Demonstrates resonance characteristics, particularly in underdamped systems, where the system's natural frequency may amplify the response.

3. **Laplace-Domain (s-Domain) Analysis**:
    - **Transfer Functions**: Each system's transfer function is derived from its differential equation, expressing the relationship between the Laplace-transformed input and output. This allows for an algebraic (rather than differential) analysis, simplifying the study of system stability and frequency response.
    - **Poles and Zeros**: The location of poles (roots of the characteristic equation) in the complex plane determines the system's stability and response type (real poles for non-oscillatory, complex poles for oscillatory responses). The analysis of poles provides a theoretical foundation for understanding the observed time-domain behavior under each damping condition.

4. **State-Space Representation**:

- **State Variables**: By defining appropriate state variables (e.g., displacement and velocity for the spring-mass system, current and its rate of change for the RLC circuit), the systems are represented in a matrix form:
- **State Equation**: Describes the evolution of the state vector over time.
- **Output Equation**: Defines how the state vector relates to the observable output.
- **System Matrices** (AAA, BBB, CCC, DDD): The matrices encode system parameters, allowing compact and scalable analysis. Matrix AAA represents system dynamics, BBB captures input effects, CCC maps state variables to the output, and DDD accounts for any direct input-to-output influence.

- **Stability and Control**: The state-space model offers a basis for advanced analysis and control design, such as placing poles for desired dynamic characteristics or designing observers to estimate unmeasured states.

5. **Comparative Analysis of Damping Effects**:

- **Overdamped Systems**: Characterized by a slow return to equilibrium without oscillations, resulting in stable but gradual responses to disturbances. This is particularly effective in systems where oscillations are undesirable.
- **Critically Damped Systems**: Achieve the fastest return to equilibrium without oscillation. This balance is optimal for systems where quick stabilization is crucial.
- **Underdamped Systems**: Display oscillations as they return to equilibrium, showing resonance effects when input frequencies match the system's natural frequency. This case demonstrates how insufficient damping leads to sustained oscillations and highlights resonance phenomena, which are critical in designing systems to avoid amplifying inputs at specific frequencies.

This assignment illustrates fundamental principles in mechanical and electrical system modeling:

- **System Stability**: Analyzing damping effects provides insights into system stability, where overdamped and critically damped systems remain stable under various inputs, while underdamped systems risk instability due to oscillations.
- **Transient vs. Steady-State Response**: The choice of damping impacts transient behavior (response speed and smoothness) and the steady-state accuracy, especially under different input types.
- **Resonance and Frequency Response**: The sinusoidal input response, especially in underdamped cases, shows resonance effects, which are essential in applications where vibration control or frequency filtering is required.
- Through this assignment, the process of **state-space modeling** and **differential equation analysis** provides a comprehensive view of how physical parameters affect system behavior, offering valuable tools for designing stable, responsive, and efficient systems in both mechanical and electrical domains.