PROJECT for FINAL EXAMINATION

Subject: **Advanced Control Theory** Level: Master's program

Class: **MST 2024** Lecturer: **Assoc. Prof. Do Xuan Phu**

Assigned date: **3 November 2024** Submitted date: **23 November 2024 @ 9:00 AM**

Problem (100 pts):

Read the article as follows:

Learning to Assist Different Wearers in Multitasks: Efficient and Individualized Human-In-the-Loop Adaptation Framework for Lower-Limb Exoskeleton

Yu Chen, Shu Miao, Gong Chen, Jing Ye, Chenglong Fu, Bin Liang, Shiji Song, and Xiang Li

1. Define "**trajectory generation**" and "**interaction control**" following the content of the manuscript. (10 pts)

Trajectory generation:

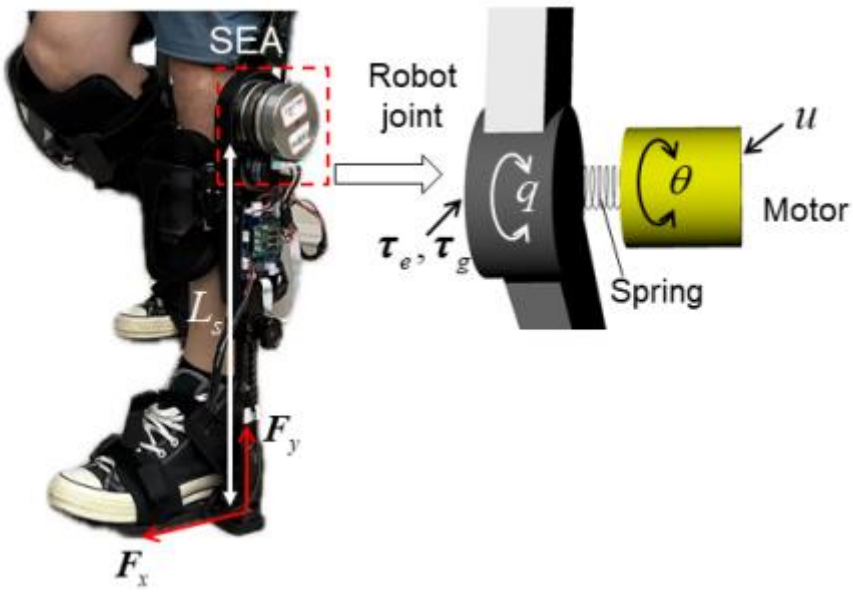
- **Trajectory generation**:

Trajectory generation in the context of exoskeletons refers to the process of creating and optimizing movement paths that the exoskeleton follows to help the wearer. This process is guided by a cost function—a mathematical objective that evaluates and shapes the trajectory based on specific criteria, such as comfort, flexibility, and adaptability. By incorporating these factors, the cost function ensures that the generated trajectories are efficient, comfortable, and well-suited to the wearer's individual needs, creating a user-centred experience across various tasks like walking, climbing stairs, and squatting. Advanced kinematic models further refine these trajectories, allowing smooth transitions in speed, slope, and other movement variations for improved adaptability. This framework relies solely on proprioceptive sensors (Proprioceptive sensors are devices that detect and provide feedback on the position, movement, and force within the body or a robotic system) within the exoskeleton, enhancing portability and usability.

- **Interaction control:**

In this manuscript, interactive control is defined as the process of controlling robot exoskeletons to follow a given trajectory while ensuring user safety. This is achieved by incorporating principles of human behaviour and biomechanics, specifically through the application of physical impedance in the limbs to resolve physical conflicts, such as discrepancies between the user's intended motion and the current joint angle of the exoskeleton.

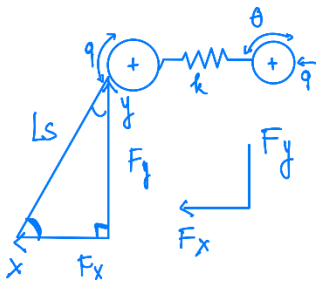
2. Use **figure 2**, find **the governing equation following the free-body-diagram**, and then **summarize** following the proposed model as shown in the manuscript. (5 pts)



Components:

1) Motor Torque: The control input u is applied by the motor, affecting the joint through the spring.
2) Elastic Spring: Between the motor and joint, which store energy and feedback, thereby ensuring structural safety.
3) Interaction Torque: This represents the net torque at the joint, influenced by the control input, elastic response of the spring, and external loads.
4) Ground Reaction Forces: Resulting from the ground reaction forces Fx and Fy at the foot of the exoskeleton.
5) Length of the shank: Ls
6) Rotational Angles:
    a. q: Rotational angle of the robot joint
    b. 0: Angle of the motor rotor

Free Body Diagram



Governing equation:

- Torque generated by spring in the SEA between the motor and the joint:

$$\tau_o = k * (\theta - q)$$

- The net torque applied by the motor to the joint after accounting for the spring torque:

$$\tau_e = u - \tau_o$$

- Torque resulting from the ground reaction forces:
  - The overall equation: $\tau = F * d$
  - ⇨ We have: $\tau_g = F_x * L_s * \cos(q) + F_y * L_s * \sin(q)$

For an object that changes its state of motion around an axis, the moment of inertia is here to measure the rotation of the system – this is describing the rotation of the knee joint, so this is the govern equation:

- For rotational systems, Newton's second law in rotational form:

  $I * \alpha = \sum \tau$

  Where:

  1. I: the moment of inertia
  2. $\alpha = \frac{d^2q}{dt^2}$: the angular acceleration of the joint
  3. $\sum \tau$: the sum of torques acting on the joint
- From that we can come up with this system:
  - Motion Status:

$$I * \frac{d^2q}{dt^2} = \tau_o + \tau_e + \tau_g$$

Summarize: From the above and the aim of this manuscript we just focus on Motion Status, so with the following equation (1) in the manuscript:

$$M(q) * \ddot{q} + C(\dot{q}, q) * \dot{q} + g(q) = \tau_o + \tau_e + \tau_g$$

With some additional concepts for complex dynamics as Coriolis matrix, gravitational torque acting on the joint, but in the overall, it is still following the concept of Newton's second law in rotational form: $\boldsymbol{I * \alpha = \sum \tau}$

3. Why we use the **equation (2)** in the manuscript? Translate the meaning when apply this equation. (5 pts)

- We use the equation (2) in the manuscript because:
  - Equation (1) describes the dynamics of the robot joint (i.e., the exoskeleton's movement), but it does not directly account for the characteristics or performance of the motor driving the system. Without Equation (2), the control would only focus on the joint behaviour, regardless of the motor's capabilities. This could make the system less effective and even unsafe if the motor is not adequately controlled, especially if it is not strong enough or if its control is not finely tuned.
  - Equation (2) is used to control the motor's behaviour. It allows the motor to generate the required bending and stretching forces for the exoskeleton's movement while using

the spring in the SEA to store energy. This energy storage helps reduce the motor's workload, preventing it from overexerting and potentially overheating. By moderating the motor's output through Equation (2), the system achieves a balance between control effectiveness and motor safety.

4. Why we use the **equation (7)** in the manuscript? How about the system when changing $\varepsilon \neq 0$? (5 pts)

This question will be related to the formation of formula (5) and its transformation into (6) and then (7), so we will start from formula (5):

- In equation (5), the primary objective í to control the motor so that it synchronizes with the joint, ensuring stability in the exoskeleton system during movement. However, this is not enough because there is also a phase difference in the response time between the moving joint and the motor. Therefore, in the manuscript, equation (3) is introduced, which separates the control input into motor control and joint control.
- In equation (7), the parameter ε is introduced to separate the motor control and joint control into two distinct parts, because the system operates on two different time scales: a fast time scale for the robot and slow time scale for the SEA. When ε is very small, the effect of the SEA on the robot dynamics become slow and must be treated separately
- When $\varepsilon \to 0$, the dynamic part of the SEA becomes stable, and the acting torque $\tau_o = K\ (\theta - q)$ will become zero if $\theta = q$. When $\varepsilon = 0$, equation (7) shows that the torque τ_o will reach a stable value $\tau_o = u_s - B\ddot{q}$, while the fast-time scale control (SEA) will no longer directly affect the robot dynamics.

When $\varepsilon \neq 0$, the system will change:

This means that θ and q will be out of phase, and the dynamics of the SEA and the robot will not be fully synchronized, which affects τ_o. As a result, the control becomes more challenging because it must be continuously adjusted to synchronize the motion of the motor and the joint. This is clearly reflected in equation (7) when $\varepsilon \neq 0$. The τ_o cannot be immediately stabilized when $\varepsilon \neq 0$, and its variation is adjusted according to the system's speed, with control factors such as u_s and $\ddot{q}$.

5. Use **figure 3**, **define the offline learning, HIL optimization, and impedance control** following the model in the manuscript. **What is the role of the impedance control when applying in the robotic**? (10 pts)

Offline learning:

Outputs:

- **To: HIL Optimization: Translated parameters – Anomaly score**
- **To: Impedance Control: Anomaly score**

Following the model in the manuscript, in this phase, data from previous human-exoskeleton interactions or simulations are analysed to understand optimal movement patterns and determine control parameters that will enhance performance. Key goals include reducing tracking errors and maximizing user comfort. The output of this phase includes *translated parameters*—optimized values for control settings that guide the exoskeleton's response. Additionally, an *anomaly score* is generated to detect any unexpected or unsafe behaviour in the system.

HIL optimization:

Outputs:

- **To Impedance Control: Trajectory**

During HIL optimization, the *translated parameters* from offline learning are tested and fine-tuned in a real-time, hardware-inclusive environment (either on the actual exoskeleton or a high-fidelity simulator). This phase refines the movement *trajectory* to ensure it is feasible and safe for physical application, accommodating real-world interactions and feedback.

Impedance control:

Output:

- **Control output to Exoskeleton**

The impedance control module uses the *trajectory* from HIL optimization and the *anomaly score* from offline learning to generate the final *control output* sent to the exoskeleton's actuators. This control output defines how the exoskeleton should move to follow the desired trajectory while adapting to the user's natural movements. By modulating stiffness and damping in response to user input, impedance control allows the exoskeleton to interact smoothly and compliantly with the user.

Role of Impedance Control in Robotics:

Impedance control is an important control method in robotics, especially in exoskeleton systems that assist and interact with humans. This method allows robots to adjust their responses based on the interaction forces from the user or the environment, ensuring flexible and agile interactions. By adjusting the three main dynamic components of stiffness, damping, and virtual mass, impedance control allows robots to respond strongly or softly to position changes, maintain stability and absorb vibrations, and create a suitable sense of inertia when the robot accelerates or decelerates. As a result, impedance control helps to minimize conflicts with the user, increase stability, and create flexible and friendly interactions, especially when assisting with moving or lifting activities. In this role, impedance control not only enhances comfort but also ensures user safety, becoming the foundation for robots to assist humans effectively and naturally in complex tasks.

6. Use the **equation (12)**. Why we use this equation? Give an example with the equivalence function as shown in **equation (12)** for the learning section. (5 pts)

$$\mathcal{L}_{VAE}(x_t) = \|x_t - \hat{x}_t\|^2 + KL[N(\mu, \sigma), N(0, I)] \quad (12)$$

- Equation (12) relates to Variational Autoencoders (VAEs), a generative model in machine learning designed to create new data that reflects variations of the input data it was trained on. In addition to this generative capability, VAEs can also perform tasks typical of standard autoencoders, such as denoising.
- In this manuscript, VAEs are utilized as a loss function to process proprioceptive data, including joint motion and torques, and generate a latent representation of this data. This latent representation aids in detecting anomalies by calculating an anomaly score, which is then used to dynamically adjust the exoskeleton's behaviour.
- Equation (12) consists of two main components: reconstruction loss and KL divergence loss:
  - **Reconstruction Loss:** The VAE generates a forecasted output $\hat{x}_t$ based on the input motion x_t. The reconstruction loss $\|x_t - \hat{x}_t\|^2$ measures the difference between the input and predicted motion. A large reconstruction loss indicates unstable or abnormal motion
  - **KL Divergence Loss:** This term regularizes the latent space (a compressed representation of data that captures its essential features, making it easier to analyse or generate similar data), ensuring that the learned distribution stays close to a standard normal distribution N (0, I). When reconstruction error is large, the KL divergence also increases, signalling abnormal motion.

Example:

1. Reconstruction loss: $\|x_t - \hat{x}_t\|^2$
2. Divergence : $KL[N(\mu, \sigma), N(0, I)]$
3. Following the Appendix – Algorithm 1: Anomaly Detection

I. RECONSTRUCTION LOSS:

1. Require $\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{K}(\boldsymbol{\theta} - \boldsymbol{q})$

We will have like an example – assump that sliding window will be n=3

3 stage for getting data points same as number of sliding window:

N=1: $q = 30,\ \dot{q} = 5,\ \theta = 32,\ \dot{\theta} = 6,\ K(\theta - q) = 2$

N=2: $q = 31,\ \dot{q} = 4.5,\ \theta = 32,\ \dot{\theta} = 6,\ K(\theta - q) = 1$

N=3: $q = 29,\ \dot{q} = 5.5,\ \theta = 32,\ \dot{\theta} = 6.5,\ K(\theta - q) = 3$

Final, we will get a sliding window queue is full:

$$Sliding\ Window = \begin{bmatrix} 30 & 5 & 32 & 6 & 2 \\ 31 & 4.5 & 32 & 6 & 1 \\ 29 & 5.5 & 32 & 6.5 & 3 \end{bmatrix}$$

2. **Flattened Window Data**

$$x_t = [30,5,32,6,2,31,4.5,32,6,1,29,5.5,32,6.5,3]$$

3. **Encode the Flattened Data:**

Here we will apply neural networks here:

For example, we will apply with 3 hidden layers:

This will be the architecture for this

- Fully Connected (Dense) layers: Linear transformations of the input.
- Nonlinear activations: Commonly ReLU or similar functions for nonlinearity.
- Output layer: Linear or no activation to produce the latent representation.

It will have an equation like this:

$$x_t^e = Encoder(x_t) = Dense_3\left(ReLU\left(Dense_2\left(ReLU\left(Dense_1(x_t)\right)\right)\right)\right)$$

Dense layer n=1:

$$h_1 = ReLU(W_1 x_t + b_1)$$

Where:

- W_1: Weight matrix (shape 10x15). => This will be random matrix
- b_1: Bias vector (shape 10x10). => assume as very small number => 0
- ReLU: Rectified Linear Unit activation function:

$$ReLU(z) = \max(0, z)$$

  - z: stand for "pre-activation value" of a neuron, and applying ReLU introduces nonlinearity to the model

Then I have the output:

$$x_t^e = W_3 h_2 + b_3 = [0.8, -0.4]$$

4. **Latent Space Parameters: The same with Encoder – this stage will have it own algorithm for decoding:**

   a. Mean (μ):

$$\mu = FC_\mu(x_t^e) = [0.2, -0.1]$$

   b. Variance (σ^2):

$$\sigma^2 = \exp\left(FC_\sigma(x_t^e)\right) = [0.5, 0.3]$$

5. **Reparametrize Latent Variable:**

Sample noise $\in from\ N(0, I)$:

$$\in = [0.1, -0.2]$$

Compute latent variable ρ :

$$\rho = \mu + \sigma \odot \in = [0.2, -0.1] + [0.707, 0.547] \odot [0.1, -0.2]$$

$$\rho = [0.2707, -0.2094]$$

6. **Decode latent Variable :**

With ρ through the decoder to reconstruct the flattened input $\hat{x}_t$:

$$\hat{x}_t = Decoder(\rho) = [30.1, 5.1, 32.2, 5.9, 1.8, 31.2, 4.4, 31.8, 6.1, 0.9, 28.8, 5.6, 31.9, 6.4, 3.1]$$

7. **Calculate Reconstruction Error:**

Calculate between x_t and $\hat{x}_t$:

$$s = MSE = \frac{1}{15}\sum_{1}^{15}\left(x_{t,i} - \hat{x}_{t,i}\right)^2$$

$$s = \frac{1}{15} * [(30 - 30.1)^2 + (5 - 5.1)^2 + \cdots + (3 - 3.1)^2]$$

$$s = \frac{1}{15} * 0.45 = 0.03$$

8. **Anomaly Score and Assumption**

The system accurately reconstructs the input data with a small error (s = 0.03), indicating that the joint positions, velocities, and control efforts are within normal operational ranges. If s exceeds the threshold, the system will flag it as an anomaly.

II. KL DIVERGENCE LOSS:

$$KL[N(\mu, \sigma), N(0, I)] = -0.5 * \sum_{j=1}^{2}(1 + ln\sigma_j^2 - u_j^2 - \sigma_j^2)$$

With above value for $\mu = [0.2, -0.1]\ and\ \sigma^2 = [0.5, 0.3]$

Substituting into the equation:

$$KL = -0.5 * ((1 + \ln(0.5) - 0.2^2 - 0.5) + (1 + \ln(0.3) - (-0.1)^2 - 0.3))$$

$$KL = -0.5 * (-0.223 - 0.514) = 0.3735$$

Total loss:

$$\mathcal{L}_{VAE} = Reconstruction\ Loss + KL\ Divergence = 0.03 + 0.3735 = 0.4035$$

7. Use **the equations (20-22)**. Translate the meaning of this group equation and explain the meaning of every parameter in these equations. (10 pts)

$$w_{ij}(t+1) = w_{ij}(t) + \Psi_j * Q_{ij}(t+1)e_{r,ij}(t) \quad (20)$$

Where:

- $Q_{ij}(t+1) = \frac{1}{\Lambda}(Q_{ij}(t) - \frac{Q_{ij}(t)^2}{\frac{\Lambda}{\Psi_j} + Q_{ij}(t)}$ (21)
- $e_{r,ij}(t) = \frac{1}{(2\pi)^2} * \ddot{q}_{demo,i} + \alpha_0 * \left(\beta_0 * q_{demo,i} + \frac{1}{2\pi}\dot{q}_{demo,i}\right) - w_{ij}(t)$ (22)
- $\Psi_j = \exp[h(\cos(2\pi * t - c_j) - 1$ (18)

Equation 20: ILWR (Incremental Locally Weighted Regression)

This group of the equation means to achieve trajectory parameterization. The algorithm ILWR is used for training on pre-recorded trajectories (joint movements overtime) and learns to approximate the mean trajectory q_{demo}.

Equation 21: This equation updates by incorporating both the past value of and the forgetting factor. It allows the model to adjust weights smoothly over time without needing to remember all historical data explicitly.

Equation 22: This error term is a composite measure of the model's deviation from the desired trajectory, accounting for position, velocity, and acceleration. It guides the weight updates in Equation (20) by showing how far the current model output is from the ideal trajectory at each time step.

Equation 18: Localized weight to each trajectory segment, letting the ILWR algorithm adjust weights with greater precision by concentrating on specific parts of the data.

q_{demo}: the mean trajectory, which is calculated by taking the average of the joint trajectories recorded across multiple wearers performing multiple tasks.

t: time.

$w_{ij}(t)$: represents the weights of the i-th joint to be updated which can be treated as the trajectory parameters - this is the current weight of the i-th joint at time t

Ψ_j: Gaussian-like kernel functions that determines how much influence each joint's data has on the weight update

h: width, typically set as 2.5J – controls the spread of the kernel function

c_j: center, evenly space in $[0, 2\pi]$

α_0: positive constant (weight given to velocity) in the error calculation

β_0: positive constant (weight given to position) in the error calculation

Λ: a forgetting factor – the factor that helps the model the intentional process of removing specific data or influences from a model's training history, often to update its knowledge for better performance.

$Q_{ij}(t+1)$: auxiliary variable used to control the rate of weight update

$e_{r,ij}$: regression error, representing the difference between the current model output and the desired trajectory at time t

$q_{demo,i}$: being the demonstration on joint i (i.e., the i-th element of the vector q_{demo})

$\dot{q}_{demo,i}$: The velocity of the mean trajectory for joint i (the speed of movement for each joint)

$\ddot{q}_{demo,i}$: The acceleration of the mean trajectory for joint i (the rapidly of the joint's movement changes)

8. Use section VI.

(a) Use Matlab, simulate the proposed control in this section. Please show the code in the word file. Discuss the simulation results. (25 pts)

Code file:

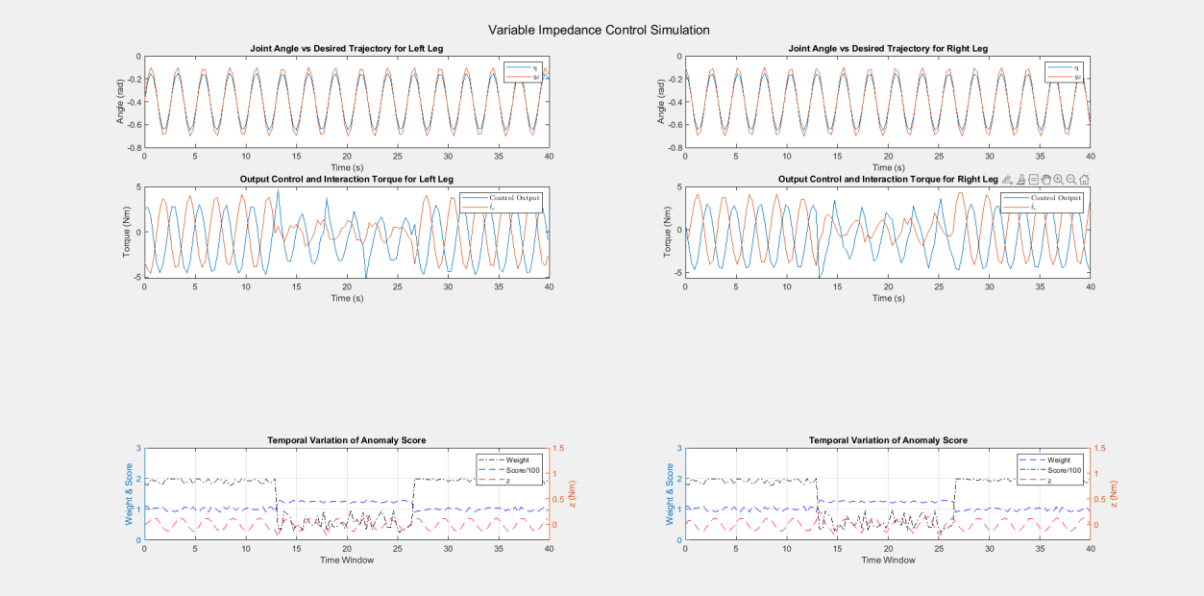
```
%% Parameters Initialization
% Tuning parameters for q and qd
A1 = 0.25;
A2 = 0.3;
T = 2.6;
B1 = 2*pi/(T); %2.035;

% Parameter for bag bang control
maxTorque = 50;
tolerance = 0.015;


% Parameters for Exoskeleton System
I2 = eye(2);
B = 3.17 * 10^-4;
K = 635;
C_q_dot_q = 0;
M_q = 3.12 * 10^-2;
Cd = 15 * I2;
Kd = 13 * I2;
Kz = 25 * I2;
Kv = (10^-3) * I2;
M_q_1 = 1/M_q;

% Impedance Control Parameters
lambda1 = 1;
lambda2 = 1;
chi1 = 10;
chi2 = 12;
k_g = 0.001;

% Simulation parameters
T = 40;
dt = 0.3;
time = 0:dt:T;

% Assumption data
sigma1 = 0.05; % ||M'(q)||
sigma2 = M_q; % ||M(q)||
beta = 1; % scaling factor

A_inv = 0.5 * (sigma1 + 2 * beta * sigma2);

numSteps = length(time);

z = zeros(2, numSteps);
qr_dot = zeros(2, numSteps);
torque_e = zeros(2, numSteps);

w_s = zeros(numSteps,1);

sgn = zeros(2, numSteps);
us = zeros(2, numSteps);
qr_dot_dot = zeros(2, numSteps);
q = zeros(2, numSteps);
q_dot = zeros(2, numSteps);
q_dot_dot = zeros(2, numSteps);
qd = zeros(2, numSteps);
qd_dot = zeros(2, numSteps);
qd_dot_dot = zeros(2, numSteps);

theta = zeros(2, numSteps);

torque_err =zeros(2,numSteps);
torque_e_bang_bang = zeros(2, numSteps);
us_bang_bang = zeros(2, numSteps);
s1 =zeros(2, numSteps);
s2 =zeros(2, numSteps);
us_test = zeros(2, numSteps);
flag = zeros(2, numSteps);
us_2 = zeros(2, numSteps);
z_dot= zeros(2, numSteps);
s = zeros(numSteps,1);
qr_dot_1 = zeros(numSteps,1);
```

```
z_after = zeros(2, numSteps);
%
torque_head_e =zeros(2, numSteps);
p_qdot_q = zeros(2,numSteps);
L_q_qdot = zeros(numSteps,1);
g_q = zeros(2,numSteps);
torque_tilde_e = zeros(2,numSteps);
torque_e_dot = zeros(2,numSteps);
%% Simulation
for i = 1:numSteps
% Adding f1 and f2 to simulate back the steps in reality
f1 = random("Uniform",0.02,0.05);
% Compute state variables
q(:, i) = [(A1 * sin(B1 * time(i) + f1)-0.4);
                  (A1 * sin(B1 * time(i) + f1 + (pi / 2))-0.4)];
q_dot(:, i) = [(A1 * B1 * cos(B1 * time(i) + f1));
                        (A1 * B1 * cos(B1 * time(i) + f1 + (pi / 2)))];
q_dot_dot(:, i) = [(-A1 * B1 * B1 * sin(B1 * time(i) + f1));
                        (-A1 * B1 * B1 * sin(B1 * time(i) + f1 + (pi / 2)))];

qd(:, i) = [(A2 * sin(B1 * time(i) + f1)-0.4);
                  (A2 * sin(B1 * time(i) + f1 + (pi / 2))-0.4)];
qd_dot(:, i) = [(A2 * B1 * cos(B1 * time(i) + f1));
                        (A2 * B1 * cos(B1 * time(i) + f1 + (pi / 2)))];
qd_dot_dot(:, i) = [(-A2 * B1 * B1 * sin(B1 * time(i) + f1));
                        (-A2 * B1 * B1 * sin(B1 * time(i) + f1 + (pi / 2)))];

theta(:, i) = [(0.35 * sin(B1 * time(i) + f1)-0.4);
                  (0.45 * sin(B1 * time(i) + f1 + (pi / 2))-0.4)];

g_q(:,i) = 2.2 * sin(q(:, i));

%Visualize back the situation when there is anomaly score make the w_s
%into 0
if i < (numSteps)/3 || i > 2*numSteps/3
       s(i) = random('Uniform',90,110);
else
       s(i) = random('Uniform',120,130);
end

w_s(i,:) = lambda1 * tanh((-s(i) / chi1) + chi2) + lambda2;

torque_e(:, i) = w_s(i,:) * (Cd * (q_dot(:, i) - qd_dot(:, i)) + Kd * (q(:, i) - qd(:,
i)));

torque_e_dot(:,i) = w_s(i,:) * (Cd * (q_dot_dot(:, i) - qd_dot_dot(:, i)) + Kd * (q_dot(:,
i) - qd_dot(:, i)));

p_qdot_q(:,i) = A_inv*q_dot(:,i);

L_q_qdot(i,:) = A_inv*M_q_1;

if i >= 2
       torque_tilde_e(:,i) = torque_tilde_e(:,i-1) - L_q_qdot(i,:) * torque_tilde_e(:,i-
       1) * dt - torque_e_dot(:,i) * dt;
       torque_head_e(:, i) = torque_e(:, i) + torque_tilde_e(:, i);
else
       torque_head_e(:, i) = torque_e(:, i) + torque_tilde_e(:, i);
end

qr_dot(:, i) = qd_dot(:, i) - pinv(Cd) * Kd * (q(:, i) - qd(:, i)) +
((1/w_s(i,:))*pinv(Cd) * torque_head_e(:, i));

z(:, i) = q_dot(:, i) - qr_dot(:, i);

sgn(:, i) = sign(z(:, i));

us(:,i) = -Kz * z(:, i) - torque_head_e(:,i) - k_g * sgn(:, i) + (M_q + B) * qr_dot_dot(:,
i) + C_q_dot_q * qr_dot(:, i) + g_q(:,i);
%% Plot Results
figure;
% Joint Angle vs Desired Trajectory for Left Leg
subplot(4, 2, 1);
plot(time, q(1, :)); hold on;
plot(time, qd(1, :));
title('Joint Angle vs Desired Trajectory for Left Leg');
```

```
xlabel('Time (s)');
ylabel('Angle (rad)');
legend('q', '$q_d$', 'Interpreter', 'latex');

% Joint Angle vs Desired Trajectory for Right Leg
subplot(4, 2, 2);
plot(time, q(2, :)); hold on;
plot(time, qd(2, :));
title('Joint Angle vs Desired Trajectory for Right Leg');
xlabel('Time (s)');
ylabel('Angle (rad)');
legend('q', '$q_d$', 'Interpreter', 'latex');

% Output Control and Interaction Torque for Left Leg
subplot(4, 2, 3);
plot(time, us(1,:));hold on;
plot(time, torque_head_e(1,:));
title('Output Control and Interaction Torque for Left Leg');
xlabel('Time (s)');
ylabel('Torque (Nm)');
legend('Control Output', '$\hat{t}_e$', 'Interpreter', 'latex');

% Output Control and Interaction Torque for Right Leg
subplot(4, 2, 4);
plot(time, us(2, :));hold on;
plot(time, torque_head_e(2, :));
title('Output Control and Interaction Torque for Right Leg');
xlabel('Time (s)');
ylabel('Torque (Nm)');
legend('Control Output', '$\hat{t}_e$', 'Interpreter', 'latex');


% % Output Control and Interaction Torque for Left Leg
% subplot(4,2,5 );
% plot(time, us_bang_bang(1, :)); hold on;
% plot(time, torque_e_bang_bang(1, :));
% title('Output Control and Interaction Torque for Left Leg - Bang Bang');
% xlabel('Time (s)');
% ylabel('Deviation (rad/s)');
% legend('Control Output', 'Torque_e');
%
% % Output Control and Interaction Torque for Right Leg
% subplot(4,2,6);
% plot(time, us_bang_bang(2, :)); hold on;
% plot(time, torque_e_bang_bang(2, :));
% title('Output Control and Interaction Torque for Right Leg - Bang Bang');
% xlabel('Time (s)');
% ylabel('Deviation (rad/s)');
% legend('Control Output', 'Torque_e');


subplot(4,2,7);

% Plot on the left y-axis
yyaxis left
plot(time, w_s, 'k-.', 'DisplayName', 'Weight');hold on;
plot(time, s/100, 'b--', 'DisplayName', 'Score/100');
ylabel('Weight & Score');
ylim([0 3]);  % Adjust based on your data

% Plot on the right y-axis
yyaxis right
plot(time, z(1,:), 'r--', 'DisplayName', 'z');
ylabel('z (Nm)');
ylim([-0.3 1.5]);  % Adjust based on your data

% Set x-axis label and title
xlabel('Time Window');
title('Temporal Variation of Anomaly Score');
grid on;

% Create a legend with all items from both y-axes
legend('Weight','Score/100', 'z');

% Set the main title for the entire figure
sgtitle('Variable Impedance Control Simulation');
```

```
subplot(4,2,8);

% Plot on the left y-axis
yyaxis left
plot(time, s/100, 'b--', 'DisplayName', 'Score/100'); hold on;
plot(time, w_s, 'k-.', 'DisplayName', 'Weight');
ylabel('Weight & Score');
ylim([0 3]);  % Adjust based on your data

% Plot on the right y-axis
yyaxis right
plot(time, z(2,:), 'r--', 'DisplayName', 'z');
ylabel('z (Nm)');
ylim([-0.3 1.5]);  % Adjust based on your data

% Set x-axis label and title
xlabel('Time Window');
title('Temporal Variation of Anomaly Score');
grid on;

% Create a legend with all items from both y-axes
legend('Weight','Score/100', 'z');

% Set the main title for the entire figure
sgtitle('Variable Impedance Control Simulation');
```

Result:



Discussion:

In this script code, because of the limitations about the parameters and experiments, so there are lots of assumptions parameters such as $q, q_d, \ldots.$, some parameters that I took in the manuscript in Result Section. From my result, the processing and constraint will base on this following equation:

Weight function:

$$w(s) = \lambda_1 \tanh\left(-\frac{s}{\chi_1} + \chi_2\right) + \lambda_2 \quad (40)$$

The desired impedance model is reformulated with the weight function:

$$C_d(\dot{q} - \dot{q_d}) + K_d(q - q_d) = \frac{1}{w(s)}\tau_e \quad (41)$$

The analytics solution of matrix A:

$$A^{-1} = \frac{1}{2}(\sigma_1 + 2\beta\sigma_2)I,\ \left\|M(\dot{q})\right\| \leq \sigma_1, \|M(q)\| \leq \sigma_2 \quad (54)$$

The observer gain matrix, and vector can be expressed as:

$$\begin{cases} L(q,\dot{q}) = A^{-1}M^{-1}(q) \\ p(\dot{q},q) = A^{-1}\dot{q} \end{cases} \quad (46)$$

The definition of the observation error:

$\tilde{\tau}_e = \hat{\tau_e} - \tau_e$ (Observation error)

The closed-loop equation of the DOB:

$\dot{\tilde{\tau}_e} = -L(\dot{q},\dot{q})\tilde{\tau}_e - \dot{\tau_e}$ (47)

- ⇨ Applying Euler's Formula:
  - $\tilde{\tau}_e(t) = \tilde{\tau}_e(t-1) + (-L(q,\dot{q}) * \tilde{\tau}_e(t-1) - \tau_e(t)) * delta(t)$

The impedance vector:

$z = \dot{q} - \dot{q_r}$ (43)

Base on the manuscript and following my understanding, at the beginning the equation (44) will be depends on the interaction torque τ_e which is using the sensor for torque measurement, but in the following manuscript, the author don't want to rely on the sensor anymore – so he comes up with estimation interaction torque, a variable that can be measured by model-based disturbance observer (DOB). To adapt to the estimate interaction torque – I have modified back the $\dot{q_r}$ which will direct affect to z.

$\dot{q_r} = \dot{q_d} - C_d^{-1}K_d\,(q - q_d) + \frac{1}{w(s)}C_d^{-1}\hat{\tau}_e$ (44)

The sign function:

$$sgn(z) = \begin{cases} 1, & z > 0 \\ 0, & z = 0 \\ -1, & z < 0 \end{cases} \quad (56)$$

The desired impedance model which designed for control input:

$u_s = -K_z z - \hat{\tau}_e - k_g * sgn(z) + (M(q) + B)\ddot{q}_r + C(\dot{q},q)\dot{q}_r + g(q)$ (55)

Assuming parameters:

- I have assumed trajectories of right leg and left leg which will be lagged for pi/2 (which means 90 degrees) – all the same in amplitude.
- Desired trajectories just different from base trajectories in amplitude.
- For the σ_1 – because this is the derivative of the inertia matrix reflects the system's dynamics' sensitivity to changes in q. This make sense this parameter must be relatively small, so I assume a small bound for it – through many times tunning.
- σ_2 : The inertia matrix varies with the configuration q but remains bounded in realistic systems. This ensures the controller or algorithm can work within predictable limits for torque or force requirements.
- β : Using a scaling factor of 1 is a common assumption when testing or developing the system in its base configuration.

Result:

Based on the assumed parameters and simulation results, the control system demonstrated its effectiveness in tracking the desired trajectory for both legs, with minimal errors and only small discrepancies in some cycles, especially in the transient phase. During the resistance period of 14–26 seconds, the abnormality score increased sharply, indicating that the system recognized the conflict and responded promptly. The weight w(s)) decreased smoothly during this period, corresponding to the decrease in the impedance parameters to minimize the conflict and maintain a safe and smooth interaction between the human and the robot. At the same time, when the interaction force from the user decreased, the control system had to increase the control input to compensate for the force that the user no longer provided, ensuring that the desired trajectory was maintained.

However, if the impedance vector (z) is not maintained stable around 0, the system is at risk of instability due to the provision of control signals that do not match the actual impact force. This may result in erroneous control signals, causing large fluctuations or conflicts between the user and the system, reducing the support efficiency and potentially causing safety loss. In this simulation, by controlling $\{z\}$ to be stable around 0, the system achieved a good balance in adjusting the interaction force and input control signals, limiting the deviation and ensuring smooth interaction. After the resistance phase, the abnormal scores and trajectories quickly returned to a stable state, demonstrating the strong resilience and adaptability of the control system.

Test case: I change the peak period for the anomaly score from t = 14 – 26s:

Based on the changes I made during the 14–26 second period related to the anomaly score, the system responded appropriately to maintain stability and safety in the human-robot interaction. During this period, the anomaly score was adjusted upward, reflecting the presence of resistance or conflict between the user and the system. This triggered the system's adjustment mechanism, causing the w(s) to gradually decrease to reduce the impedance parameter, thereby softening the system and limiting conflicts with the user. Reducing the impedance parameter w(s) plays a very important role in maintaining safety, helping the system control the impedance vector (z) to stabilize around 0, thereby regulating the exchange between the interaction force and the control signal. As the user force decreased during this period, the control system had to increase the control input signal to compensate for the lack of force, ensuring that the joint trajectory still followed the desired trajectory. This demonstrates the effective adaptation mechanism of the system. If the impedance vector (z) is not well controlled, the control signal may deviate from the actual interaction force, leading to unwanted oscillations and possible instability. However, in this simulation result, the system-maintained stability, as both z and the joint trajectory returned to normal after 14–26 seconds. This demonstrates

that the system not only effectively adapts to sudden changes but also maintains safety and stability throughout the entire interaction process.

(b) Use bang-bang model of the optimal control and then applying to the model in the manuscript. Simulate the bang-bang control by Matlab and discuss the results. Please show the code in the word file. (25 pts)

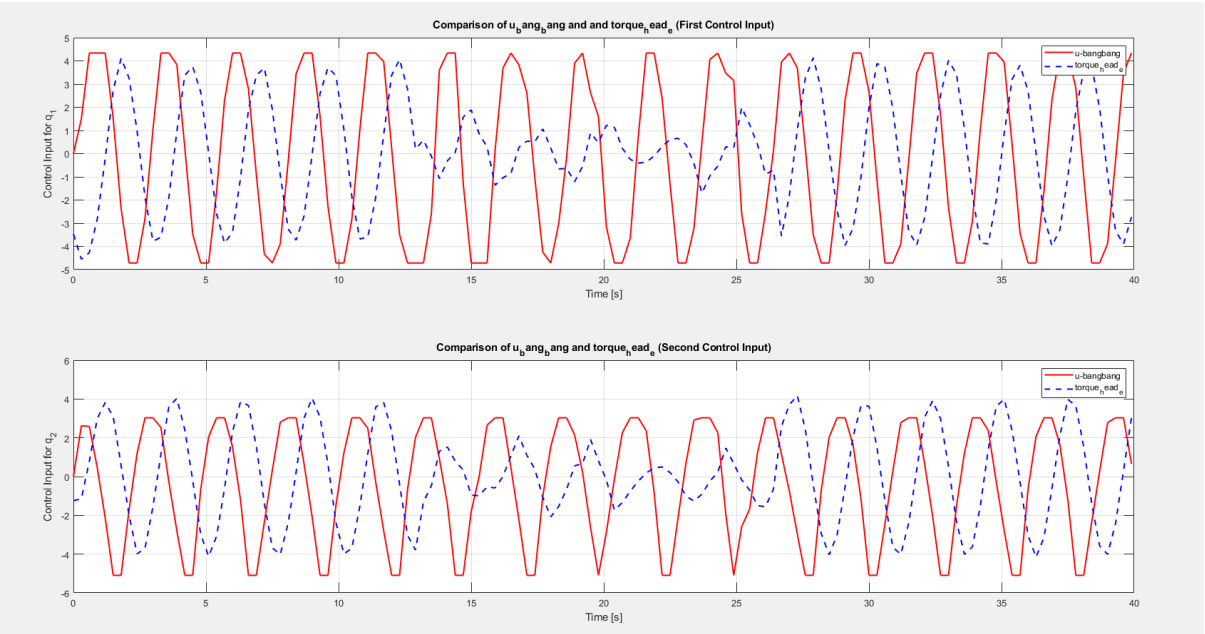
Code file:

```
%% Parameters Initialization
% Tuning parameters for q and qd
A1 = 0.25;
A2 = 0.3;
T = 2.6;
B1 = 2*pi/(T); %2.035;

% Parameter for bag bang control
maxTorque = 50;
tolerance = 0.015;
f1 = 0;

% Parameters for Exoskeleton System
I2 = eye(2);
B = 3.17 * 10^-4;
K = 635;
C_q_dot_q = 0;
M_q = 3.12 * 10^-2;
Cd = 15 * I2;
Kd = 13 * I2;
Kz = 25 * I2;
Kv = (10^-3) * I2;
M_q_1 = 1/M_q;

% Impedance Control Parameters
lambda1 = 1;
lambda2 = 1;
chi1 = 10;
chi2 = 12;
k_g = 0.001;

% Simulation parameters
T = 40;
dt = 0.3;
time = 0:dt:T;

% Assumption data
sigma1 = 0.05; % ||M'(q)||
sigma2 = M_q; % ||M(q)||
beta = 1; % scaling factor

A_inv = 0.5 * (sigma1 + 2 * beta * sigma2);

numSteps = length(time);

z = zeros(2, numSteps);
qr_dot = zeros(2, numSteps);
torque_e = zeros(2, numSteps);

w_s = zeros(numSteps,1);

sgn = zeros(2, numSteps);
us = zeros(2, numSteps);
qr_dot_dot = zeros(2, numSteps);
q = zeros(2, numSteps);
q_dot = zeros(2, numSteps);
q_dot_dot = zeros(2, numSteps);
qd = zeros(2, numSteps);
qd_dot = zeros(2, numSteps);
qd_dot_dot = zeros(2, numSteps);

theta = zeros(2, numSteps);
```

```
torque_err =zeros(2,numSteps);
torque_e_bang_bang = zeros(2, numSteps);
us_bang_bang = zeros(2, numSteps);

z_dot= zeros(2, numSteps);
s = zeros(numSteps,1);
qr_dot_1 = zeros(numSteps,1);
%
torque_head_e =zeros(2, numSteps);
p_qdot_q = zeros(2,numSteps);
L_q_qdot = zeros(numSteps,1);
g_q = zeros(2,numSteps);
torque_tilde_e = zeros(2,numSteps);
torque_e_dot = zeros(2,numSteps);

% bang bang control
u_bang_bang = zeros(2,numSteps);
error_Threshold = [0.1;0.1];
error = zeros(2,numSteps);

%% Simulation
for i = 1:numSteps
% Adding f1 and f2 to simulate back the steps in reality
% Compute state variables
q(:, i) = [(A1 * sin(B1 * time(i) + f1)-0.4);
                (A1 * sin(B1 * time(i) + f1 + (pi / 2))-0.4)];
q_dot(:, i) = [(A1 * B1 * cos(B1 * time(i) + f1));
                (A1 * B1 * cos(B1 * time(i) + f1 + (pi / 2)))];
q_dot_dot(:, i) = [(-A1 * B1 * B1 * sin(B1 * time(i) + f1));
                      (-A1 * B1 * B1 * sin(B1 * time(i) + f1 + (pi / 2)))];

qd(:, i) = [(A2 * sin(B1 * time(i) + f1)-0.4);
                (A2 * sin(B1 * time(i) + f1 + (pi / 2))-0.4)];
qd_dot(:, i) = [(A2 * B1 * cos(B1 * time(i) + f1));
                (A2 * B1 * cos(B1 * time(i) + f1 + (pi / 2)))];
qd_dot_dot(:, i) = [(-A2 * B1 * B1 * sin(B1 * time(i) + f1));
                      (-A2 * B1 * B1 * sin(B1 * time(i) + f1 + (pi / 2)))];

theta(:, i) = [(0.35 * sin(B1 * time(i) + f1)-0.4);
                (0.45 * sin(B1 * time(i) + f1 + (pi / 2))-0.4)];

g_q(:,i) = 2.2 * sin(q(:, i));

%Visualize back the situation when there is anomaly score make the w_s
%into 0
if i < (numSteps)/3 || i > 2*numSteps/3
       s(i) = random('Uniform',90,110);
else
       s(i) = random('Uniform',120,130);
end

w_s(i,:) = lambda1 * tanh((-s(i) / chi1) + chi2) + lambda2;

torque_e(:, i) = w_s(i,:) * (Cd * (q_dot(:, i) - qd_dot(:, i)) + Kd * (q(:, i) - qd(:,
i)));

torque_e_dot(:,i) = w_s(i,:) * (Cd * (q_dot_dot(:, i) - qd_dot_dot(:, i)) + Kd * (q_dot(:,
i) - qd_dot(:, i)));

p_qdot_q(:,i) = A_inv*q_dot(:,i);

L_q_qdot(i,:) = A_inv*M_q_1;

if i >= 2
       torque_tilde_e(:,i) = torque_tilde_e(:,i-1) - L_q_qdot(i,:) * torque_tilde_e(:,i-
       1) * dt - torque_e_dot(:,i) * dt;
       torque_head_e(:, i) = torque_e(:, i) + torque_tilde_e(:, i);
else
       torque_head_e(:, i) = torque_e(:, i) + torque_tilde_e(:, i);
end

qr_dot(:, i) = qd_dot(:, i) - pinv(Cd) * Kd * (q(:, i) - qd(:, i)) +
((1/w_s(i,:))*pinv(Cd) * torque_head_e(:, i));

z(:, i) = q_dot(:, i) - qr_dot(:, i);
```

```
sgn(:, i) = sign(z(:, i));

us(:,i) = -Kz * z(:, i) - torque_head_e(:,i) - k_g * sgn(:, i) + (M_q + B) * qr_dot_dot(:,
i) + C_q_dot_q * qr_dot(:, i) + g_q(:,i);

end

%% bang bang control

max_control = zeros(2, 1);
min_control = zeros(2, 1);

% Define max and min for each leg
max_control(1) = max(us(1,:));
min_control(1) = min(us(1,:));
max_control(2) = max(us(2,:));
min_control(2) = min(us(2,:));

for i = 1:numSteps
error(:,i) = q_dot(:,i) - qr_dot(:,i);

% Right leg
if abs(error(1,i)) > error_Threshold(1,1)
        if error(1,i) > 0
                u_bang_bang(1,i) = max_control(1);
        else
                u_bang_bang(1,i) = min_control(1);
        end
else
        u_bang_bang(1,i) = max_control(1) * error(1,i) / error_Threshold(1,1);
end

% Left leg
if abs(error(2,i)) > error_Threshold(2,1)
        if error(2,i) > 0
                u_bang_bang(2,i) = max_control(2);
        else
                u_bang_bang(2,i) = min_control(2);
        end
else
        u_bang_bang(2,i) = max_control(2) * error(2,i) / error_Threshold(2,1);
end
end

%
% Plotting the comparison between u_bang_bang and u
figure;
subplot(2,1,1);
plot(time, u_bang_bang(1,:), 'r', 'LineWidth', 1.5); hold on;
plot(time, torque_head_e(1,:), 'b--', 'LineWidth', 1.5);
xlabel('Time [s]');
ylabel('Control Input for q_1');
title('Comparison of u_bang_bang and and torque_head_e (First Control Input)');
legend('u-bangbang', 'torque_head_e');
grid on;

subplot(2,1,2);
plot(time, u_bang_bang(2,:), 'r', 'LineWidth', 1.5); hold on;
plot(time, torque_head_e(2,:), 'b--', 'LineWidth', 1.5);
xlabel('Time [s]');
ylabel('Control Input for q_2');
title('Comparison of u_bang_bang and torque_head_e (Second Control Input)');
legend('u-bangbang', 'torque_head_e');
grid on;
```

Result:



Discussion:

From my own research, Bang-bang control, also known as open-close control, is a feedback control method that switches abruptly between two states, typically using a Heaviside step function in its discrete form. In this article, the bang-bang control approach is applied using the Heaviside function to represent the control strategy. This method alternates between maximum and minimum control efforts based on the error between the desired and actual joint velocities of the exoskeleton. If the error exceeds a predefined threshold, the control input is set to either the maximum or minimum value, depending on the error's direction (positive or negative). Specifically, the maximum control value corresponds to the condition where the error is large and positive, while the minimum control value is used when the error is large and negative. When the error is within the threshold, the control input is scaled proportionally to the error, thus reducing the control effort.

The error used in this control strategy is derived from the impedance vector, as it reflects the relationship between the desired motion and the actual joint motion, thus offering a more direct and dynamic response. The impedance vector essentially captures the mismatch between the desired and actual movements, and its magnitude is used to determine whether the system should apply maximum, minimum, or scaled control inputs.

From the graph, we can observe the characteristic behavior of the bang-bang control method, which is marked by sharp switching between maximum and minimum control inputs depending on whether the error between the desired and actual joint velocities exceeds a predefined threshold. This abrupt switching leads to oscillations in the torque, causing significant fluctuations in the system's behavior. The system tends to overshoot as it rapidly corrects errors, and this overcompensation results in the

system adjusting in the opposite direction. The lack of smoothness in the control input, due to the sudden transitions, affects the stability and comfort of the exoskeleton, making it less suitable for tasks requiring fine precision or prolonged use. The graph also highlights the limitations of the method, including mechanical stress caused by frequent switching, sensitivity to error threshold values that can lead to either unnecessary oscillations or delayed responses, and issues with comfort and precision. Although bang-bang control is efficient for correcting large errors quickly, the oscillatory nature, overshooting, and lack of smoothness indicate a need for more advanced control strategies to improve system performance, stability, and user comfort.