Assignment 04

SAURAV KUMAR

26 March, 2024

1 Overview of the Assignment

1.1 Aim of the Assignment

In this question, we are designing gain-scheduled controllers to swing and stabilize a pendulum to its upright position and demonstrate its performance in MATLAB simulations.

2 Matlab Code

```
clc;
clear all;
% System parameters
g = 9.81;%gravity
L = 1;   %length
b = 0.5; % damping coefficient
m = 1;   % mass

% Sampling parameters
num_points = 100;
sampling_time = 0.01;

% Operating point sampling
theta_values = linspace(-pi, pi, num_points);

% Linearized system matrices
```

```
A_matrices = cell(num_points, 1);
B_matrices = cell(num_points, 1);
% Control gains and inputs
K_gains = cell(num_points, 1);
U_inputs = cell(num_points, 1);

% Compute linearized system dynamics and control gains
for i = 1:num_points
    theta_i = theta_values(i);
    % Trimming: Compute input corresponding to the operating point
    U_i = m * g * L * sin(theta_i);
    % Linearize the system around the current initial angle
    A_i = [0, 1; -g * cos(theta_i) / L, -b / (m * L^2)];
    B_i = [0; 1 / (m * L^2)];
    % Store matrices
    A_matrices{i} = A_i;
    B_matrices{i} = B_i;
    U_inputs{i} = U_i;
    % Design controller gains using LQR
    Q = diag([50, 50]); % State cost
    R = 1; % Control cost
    sysd = c2d(ss(A_i, B_i, eye(2), zeros(2, 1)), sampling_time); % Convert to dis
    K_gains{i} = dlqr(sysd.A, sysd.B, Q, R);
end

% Simulation parameters
final_state = [pi; 0]; % Final state
final_time = 25; % Final time
time = 0:sampling_time:final_time; % Time vector
% Initialize state and control input vectors
state_vector = zeros(2, length(time)); % State vector
control_input_vector= zeros(1, length(time)); % Control input vector
% Additive noise parameters
noise_mean = 0;
noise_variance = 0.001;

% Simulate system dynamics with noise disturbance
for i = 1:length(time)-1
```

```
    % Add noise to the current state
    state_vector(:, i) = state_vector(:, i) + noise_variance * randn(2, 1) + noise
    % Get the index of the closest operating point
    [~, idx] = min(abs(theta_values - state_vector(1, i)));
    % Applying control law based on gain scheduling
    control_input_vector(i) = -K_gains{idx} * (state_vector(:, i) - final_state) +
    % Update system dynamics
    state_vector(:, i + 1) = state_vector(:, i) + [state_vector(2, i); -g / L * si
end

% Plot results
figure;

% Plot angle, angular velocity, and control input vs time
subplot(2, 1, 1);
plot(time, state_vector(1, :), 'black', 'LineWidth', 2);
hold on;
plot(time, state_vector(2, :), 'b', 'LineWidth', 2);
ylabel('Angle (rad), Angular Velocity (rad/s)');
title('Pendulum Angle and Angular Velocity vs Time');
legend('Angle', 'Angular Velocity', 'Location', 'best');
grid on;

subplot(2, 1, 2);
plot(time, control_input_vector, 'r', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Control Input');
title('Control Input vs Time');
grid on;
```

3 Part 1

Computations of the linearized system dynamics:

The linearized system dynamics can be obtained by linearizing the nonlinear equations of the pendulum system about an equilibrium point. Let

(x_{1e}, x_{2e}) denote the equilibrium point, where $x_{1e} = \pi$ (upright position) and $x_{2e} = 0$ (zero angular velocity). The linearized system dynamics can be represented in state-space form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

where $\mathbf{x} = [x_1, x_2]^T$ is the state vector, $\mathbf{u} = [\tau]^T$ is the control input vector, and $\mathbf{A}$ and $\mathbf{B}$ are the system matrices obtained by taking partial derivatives of the nonlinear equations with respect to the states and inputs, respectively, and evaluating them at the equilibrium point.

The expressions for $\mathbf{A}$ and $\mathbf{B}$ are given by:

$$\mathbf{A} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}_{(x_{1e}, x_{2e})} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{bmatrix}_{(x_{1e}, x_{2e})}$$

where $f_1 = x_2$ and $f_2 = -\frac{g}{L}\sin(x_1) - \frac{b}{mL^2}x_2 + \frac{1}{mL^2}\tau$ represent the equations of motion of the pendulum system.

4 Part 2

In this control system, we're using gain-scheduled control, where the control gains are adjusted based on the operating point of the system. The control law for each operating point is given by:

$$u(t) = -K_i(x(t) - x_f) + U_i$$

Here:
- $u(t)$ is the control input at time t. - K_i is the control gain matrix corresponding to the i^{th} operating point.
- $x(t)$ is the state vector at time t.
- x_f is the desired final state.
- U_i is the input corresponding to the i^{th} operating point.

The control gains K_i are determined using Linear Quadratic Regulator (LQR) design. LQR minimizes a cost function that consists of both state and

control costs. The performance requirements are represented by the weighting matrices Q and R.

In this implementation, the state cost matrix Q is chosen as a diagonal matrix with higher weights on the angle (θ) and angular velocity ($\dot{\theta}$) to prioritize the control effort towards stabilizing these states. The control cost R is chosen to balance between achieving control effort minimization and system performance.

The reasoning behind these choices is to design a control system that stabilizes the pendulum around its upright position ($\theta = \pi$) while minimizing overshoots and ensuring fast convergence. By tuning the control gains using LQR, we aim to optimize the closed-loop performance of the system for each sampled operating point.

5 Part 3

The sequence of operating points was selected to cover the entire range of possible angles of the pendulum, from $-\pi$ to π. This choice ensures that the linearized system dynamics are adequately represented across the entire range of motion of the pendulum. By sampling the operating points uniformly, we obtain a diverse set of conditions under which the system can operate, allowing the controller to adapt to various scenarios.

The implementation of the control algorithm in MATLAB involves several steps:

1. **Linearization**: The nonlinear pendulum dynamics are linearized around each operating point by computing the system matrices A and B using partial derivatives of the system equations.

2. **Gain Scheduling**: For each sampled operating point, control gains are computed using the Linear Quadratic Regulator (LQR) method. This ensures that the controller is optimized for stabilization around each specific operating point.

3. **Simulation**: The system is simulated over a specified time period, during which the control algorithm is applied to stabilize the pendulum around the desired upright position. Additive noise is introduced to the system dynamics to mimic real-world conditions.

Pros and Cons of Varying the Number of Operating Points

Pros:

- **Increased Accuracy**: With more operating points, the linearized system dynamics better capture the nonlinear behavior of the system, leading to more accurate control.
- **Improved Adaptability**: A larger number of operating points allows the controller to adapt more effectively to variations in system parameters or external disturbances.

Cons:

- **Higher Computational Cost**: Increasing the number of operating points requires more computations during the control design phase, potentially increasing computational complexity.
- **Diminished Robustness**: While more operating points offer better adaptability, they may also introduce complexity that could lead to reduced robustness of the control system.

In summary, the choice of operating points and the number thereof in the control algorithm implementation plays a crucial role in achieving effective stabilization of the pendulum system. While a larger number of operating points can offer improved accuracy and adaptability, it comes with increased computational cost and potential challenges in maintaining robustness.

6 Part 4

The nature of the output plots obtained from the simulation of the pendulum system provides valuable insights into its behavior under control inputs and noise disturbances:

1. Pendulum Angle and Angular Velocity vs Time (Upper Plot): - The angle plot shows the oscillatory behavior of the pendulum as it swings back and forth. - The angular velocity plot illustrates how the rate of change

of the angle varies over time. - The oscillations in both plots indicate the dynamic nature of the system as it responds to control inputs and disturbances. - The amplitude and frequency of the oscillations depend on factors such as the initial conditions, system parameters, control gains, and the presence of noise.

2. Control Input vs Time (Lower Plot): - The control input plot depicts how the control signal changes over time to stabilize the pendulum. - Fluctuations in the control input reflect the system's effort to counteract the effects of gravity, damping, and other disturbances. - Peaks or spikes in the control input may indicate sudden adjustments made by the controller to correct deviations from the desired state. - The overall trend of the control input curve shows the controller's ability to maintain stability and regulate the pendulum's motion.

Overall, these plots provide a visual representation of the dynamic behavior of the pendulum system under control. They help in understanding how the system responds to control inputs and noise disturbances, allowing for further analysis and refinement of the control strategy if necessary.

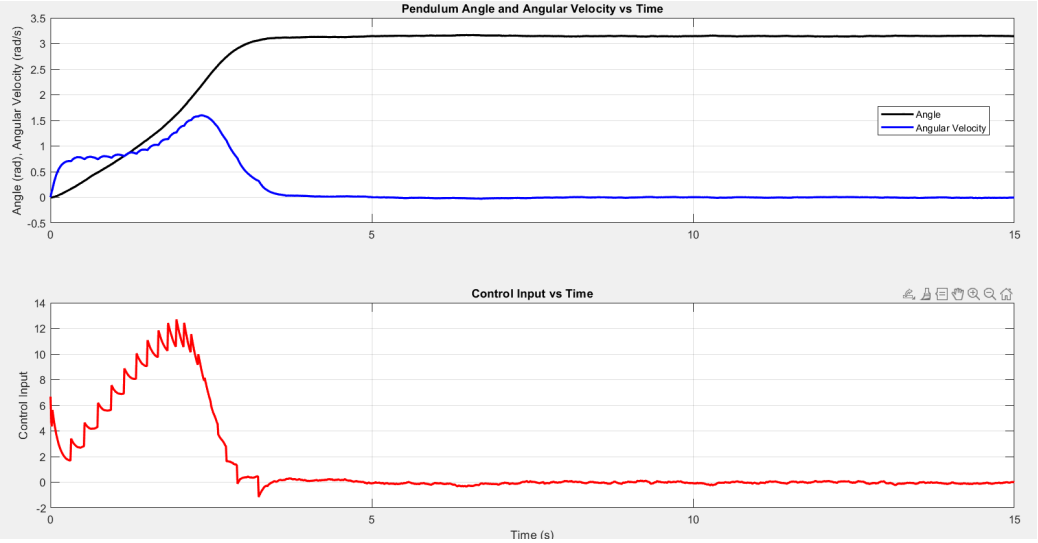

Figure 1: Simulations