TEMPERATURE CONTROL SYSTEM PROJECT

FOCUS ON SMART THERMOSTAT

Introduction

The smart thermostat control system project aims to design and analyze a model that enhances energy efficiency and user convenience. By integrating sensors, controllers, and communication modules, the smart thermostat dynamically adjusts the indoor climate based on user preferences and environmental conditions. This report outlines the objectives, development procedures, and results of the project.

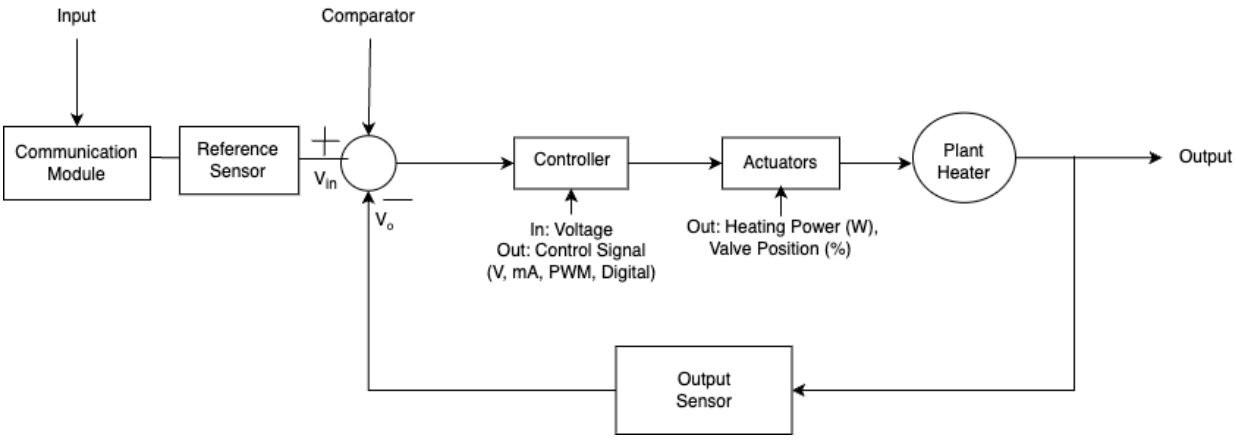
System Description

The smart thermostat system includes a temperature sensor, a controller, a communication module, and an HVAC system. The temperature sensor monitors the indoor temperature, the controller adjusts the HVAC system based on the sensor readings and user settings, and the communication module allows remote control via a smartphone app.

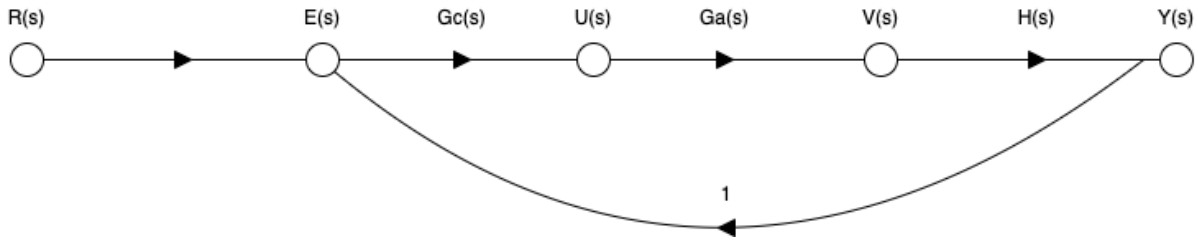
Project Objectives

- Make a block diagram of the system.
- Derive differential equations for the system.
- Generate the transfer function using Laplace transforms.
- Obtain the state space representation.
- Analyze the system's time domain response, including impulse and step responses.
- Create Root Locus plots for system stability analysis.
- Design a controller using the Root Locus method.
- Implement and analyze the system in MATLAB/Simulink.

BLOCK DIAGRAM



SIGNAL FLOW GRAPH



HEAT TRANSFER AND THERMAL SYSTEM PROPERTIES

In a thermal system, there are two different ways heat is transferred:

- Thermal Capacitance: $C = \rho c_\rho V$ where C is the thermal capacitance and can be stated as a product of ρ material density, c_ρ specific heat, and Volume V
- Conduction: $q = \frac{kA}{l} \Delta T = D_{1-2} \Delta T$ where q is the rate of heat transfer (flow), k is the thermal conductivity related to material used, A is the area normal to the direction of heat flow, and ΔT is difference between two temperatures.

Note that: $D_{1-2} = \frac{1}{\frac{l}{kA}} = \frac{1}{R}$ where R is thermal resistance. This simplifies the

heat transfer equation q w/r/t R: $q = \frac{\Delta T}{R}$

TRANSFER FUNCTION

Basic Heat Transfer Equations

Thermal Capacitance $C = \rho c_p V$

Where $\rho =$ density of the material

$c_p =$ specific heat capacity

$V =$ volume

Conduction, $q = \frac{kA\,\Delta T}{L}$, $R = \frac{L}{kA}$

Where $q =$ rate of heat transfer

$k =$ thermal conductivity

$A =$ cross-sectional area through which heat is transferred

ΔT = temperature difference across the material

$L =$ thickness of the material

$R =$ thermal resistance

Therefore; $q = \frac{\Delta T}{R}$

Modelling a simple house with one temperature sensor and one heater;

Heat Balance Equation $Q_{in}(t) = Q_{stored}(t) + Q_{loss}(t)$

Store Heat $Q_{stored}(t) = C\frac{dT(t)}{dt}$

Heat Loss $Q_{loss}(t) = -\frac{T_{out}(t) - T(t)}{R}$

Heat Input. $Q_{in}(t) = P_{heater}(t)$

Combine the equations to get

$$C\frac{dT(t)}{dt} = P_{heater}(t) + \frac{T_{out}(t) - T(t)}{R}$$

$$C\frac{dT(t)}{dt} + \frac{T(t)}{R} = \frac{T_{out}(t)}{R} + P_{heater}(t)$$

Taking the Laplace

$$CsT(s) + \frac{T(s)}{R} = + \frac{T_{out}(s)}{R} + P_{heater}(s)$$

$$T(s) = \frac{\frac{T_{out}(s)}{R} + P_{heater}(s)}{Cs + \frac{1}{R}}$$

Assuming $T_{out} = 0$,

Plant Transfer Function

$$\boldsymbol{H(s) = \frac{1}{Cs + \frac{1}{R}}}$$

From the block diagram we can derive the transfer functions for

- Comparator signal: $\mathrm{E(s)} = \mathrm{R(s)} - \mathrm{Y(s)}$
- Controller, $G_c(\mathrm{s})$: $\mathrm{U(s)} = G_c(\mathrm{s}) \cdot \mathrm{E(s)}$
- Actuator, $\mathrm{G}_a(\mathrm{s})$: $\mathrm{V(s)} = \mathrm{G}_a(\mathrm{s}) \cdot \mathrm{U(s)}$
- Plant, $\mathrm{H(s)}$: $\mathrm{Y(s)} = \mathrm{H(s)} \cdot \mathrm{V(s)}$

To find the overall Transfer Function $\frac{\mathrm{Y(s)}}{R(s)}$, substitute each equation into the next

- $\mathrm{E}(s) = \mathrm{R}(s) - \mathrm{Y}(s)$
- $$\mathrm{U}(s) = G_c(s) \cdot \big(\mathrm{R}(s) - \mathrm{Y}(s)\big)$$
- $$\mathrm{V}(s) = \mathrm{G}_a(s) \cdot \Big(G_c(\mathrm{s}) \cdot \big(\mathrm{R}(s) - \mathrm{Y}(s)\big)\Big)$$
- $$\mathrm{Y}(s) = \mathrm{H}(s) \cdot \Big(\mathrm{G}_a(s) \cdot \Big(G_c(s) \cdot \big(\mathrm{R}(s) - \mathrm{Y}(s)\big)\Big)\Big)$$

Solving for $Y(s)$:

$$Y(s) = H(s) \cdot G_a(s) \cdot G_c(s) \cdot R(s)$$

Transfer Function $\frac{\mathrm{Y(s)}}{R(s)}$:

$$\frac{\mathrm{Y(s)}}{R(s)} = \frac{\mathrm{H}(s) \cdot G_a(s) \cdot G_c(s)}{1 + \mathrm{H}(s) \cdot G_a(s) \cdot G_c(s) \cdot \mathrm{R}(s)}$$

State-Space Equation

State-Variable –: Temperature

Equation

$$x(t) = A\,x(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

For our thermal system

$$A = -\frac{1}{CR},\ \ B = \frac{1}{C},\ \ C = 1,\ D = 0$$

TIME-DOMAIN ANALYSIS

MATLAB Code using example values of $C = 1\ and\ R = 1$

```
% Define parameters
C = 1; % Thermal capacitance (example value)
R = 1; % Thermal resistance (example value)

% Transfer function
H = tf([1], [C, 1/R]);

% Impulse response
figure;
impulse(H);
title('Impulse Response of the System');
grid on;

% Step response
figure;
step(H);
title('Step Response of the System');
grid on;

% Bode plot
figure;
bode(H);
title('Bode Plot of the System');
grid on;
```

```
% Analyze time response
info = stepinfo(H);
overshoot = info.Overshoot;
settlingTime = info.SettlingTime;
steadyStateValue = dcgain(H);

% Display results
fprintf('Overshoot: %.2f%%\n', overshoot);
fprintf('Settling Time: %.2f seconds\n', settlingTime);
fprintf('Steady-State Value: %.2f\n', steadyStateValue);

% Root locus plot
figure;
rlocus(H);
title('Root Locus Plot of the System');
grid on;

% Find gain value for a specific desired pole location (optional)
% For example, find K for a pole at -1
K = rlocfind(H, -1);
disp(['Gain K for pole at -1: ', num2str(K)]);
```

RESULT

Overshoot: 0.00%

Settling Time: 3.91 seconds

Steady-State Value: 1.00

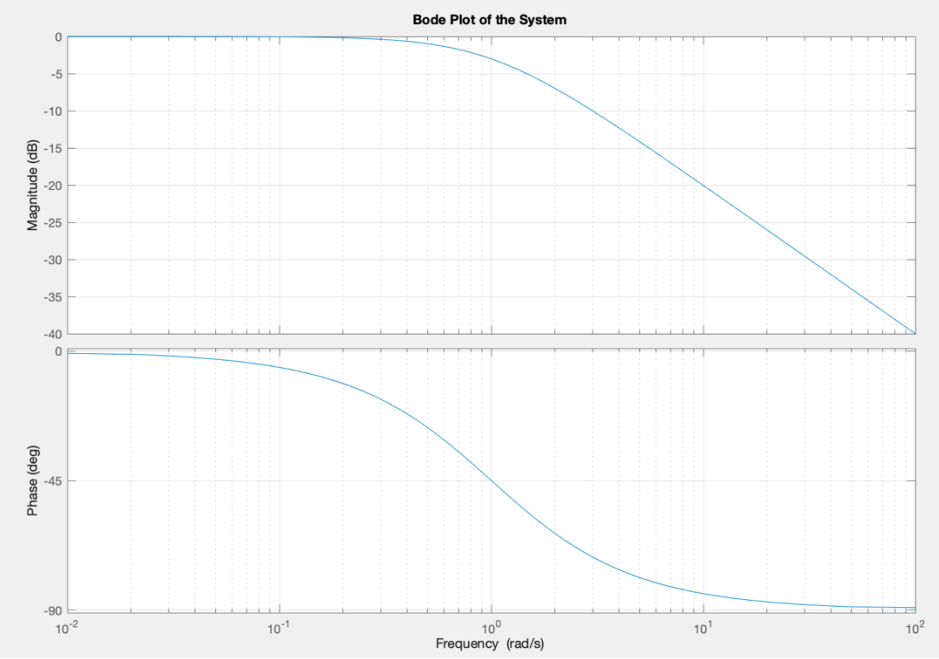


Figure 1.1: Bode Plot of the System

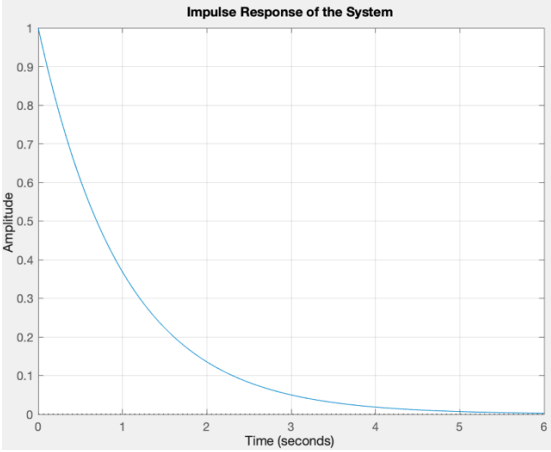


Figure 1.2: System Impulse response

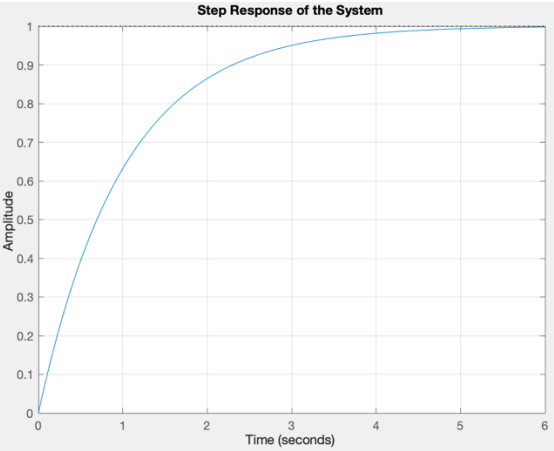


Figure 1.3: System Step response

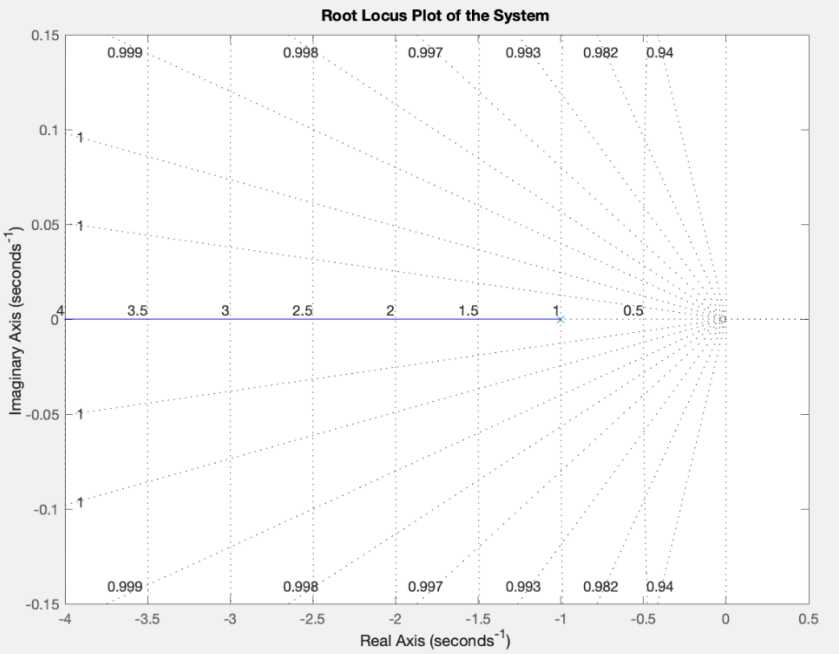


Figure 1.4: Root Locus Plot of the System

DESIGN CRITERIA AND OBJECTIVES

To evaluate the performance of the smart thermostat system, three distinct design cases are considered:

Case 1: Standard Comfort and Stability

- **Objective**: To achieve a comfortable and stable indoor temperature with minimal overshoot and settling time.
- **Criteria**:
  - **Overshoot**: Less than 10%
  - **Settling Time**: Less than 5 minutes

    - **Steady-State Error**: Close to zero
    - **Phase Margin**: Greater than 45 degrees

Case 2: Rapid Response

- **Objective**: Achieve rapid changes in indoor temperature in response to user input, prioritizing quick adjustment over minimal overshoot.
- **Criteria**:
    - **Overshoot**: Acceptable up to 20%
    - **Settling Time**: Less than 2 minutes
    - **Steady-State Error**: Close to zero
    - **Phase Margin**: Greater than 30 degrees

Case 3: Energy Efficiency

- **Objective**: Optimize energy efficiency by reducing power consumption while maintaining a stable indoor temperature.
- **Criteria**:
    - **Overshoot**: Less than 5%
    - **Settling Time**: Less than 10 minutes
    - **Steady-State Error**: Minimal
    - **Power Consumption**: Minimized
    - **Phase Margin**: Greater than 50 degrees

MATLAB code

```
% Define parameters
C = 1; % Thermal capacitance (example value)
R = 1; % Thermal resistance (example value)

% Transfer function
H = tf([1], [C, 1/R]);

% Design criteria for multiple cases
cases = {
    struct('name', 'Standard Comfort and Stability', 'Kp', 10, 'Ki', 1, 'Kd',
1, 'overshoot', 10, 'settlingTime', 300, 'phaseMargin', 45),
    struct('name', 'Rapid Response', 'Kp', 15, 'Ki', 2, 'Kd', 2, 'overshoot',
20, 'settlingTime', 120, 'phaseMargin', 30),
    struct('name', 'Energy Efficiency', 'Kp', 8, 'Ki', 0.5, 'Kd', 0.5,
'overshoot', 5, 'settlingTime', 600, 'phaseMargin', 50)
};

% Loop through each case and simulate
for i = 1:length(cases)
    case_i = cases{i};

    % Design PID controller
    Kp = case_i.Kp;
    Ki = case_i.Ki;
    Kd = case_i.Kd;
    D = tf([Kd, Kp, Ki], [1, 0]);

    % Open-loop transfer function with controller
    G_open = D * H;
```

```
    % Closed-loop transfer function with feedback
    G_closed = feedback(G_open, 1);

    % Step response with PID controller
    figure;
    step(G_closed);
    title(['Step Response with PID Controller - ', case_i.name]);
    grid on;

    % Impulse response with PID controller
    figure;
    impulse(G_closed);
    title(['Impulse Response with PID Controller - ', case_i.name]);
    grid on;

    % Bode plot with PID controller
    figure;
    bode(G_closed);
    title(['Bode Plot with PID Controller - ', case_i.name]);
    grid on;

    % Analyze time response with PID controller
    info_closed = stepinfo(G_closed);
    overshoot_closed = info_closed.Overshoot;
    settlingTime_closed = info_closed.SettlingTime;
    steadyStateValue_closed = dcgain(G_closed);

    % Display results with PID controller
    fprintf('Case: %s\n', case_i.name);
    fprintf('Overshoot: %.2f%% (Criteria: %.2f%%)\n', overshoot_closed,
case_i.overshoot);
    fprintf('Settling Time: %.2f seconds (Criteria: %.2f seconds)\n',
settlingTime_closed, case_i.settlingTime);
    fprintf('Steady-State Value: %.2f\n', steadyStateValue_closed);
    fprintf('\n');
end
```

RESULTS

Case: Standard Comfort and Stability

Overshoot: 0.00% (Criteria: 10.00%)

Settling Time: 15.70 seconds (Criteria: 300.00 seconds)

Steady-State Value: 1.00

Case: Rapid Response

Overshoot: 0.00% (Criteria: 20.00%)

Settling Time: 8.21 seconds (Criteria: 120.00 seconds)

Steady-State Value: 1.00

Case: Energy Efficiency

Overshoot: 0.00% (Criteria: 5.00%)

Settling Time: 29.89 seconds (Criteria: 600.00 seconds)

Steady-State Value: 1.00

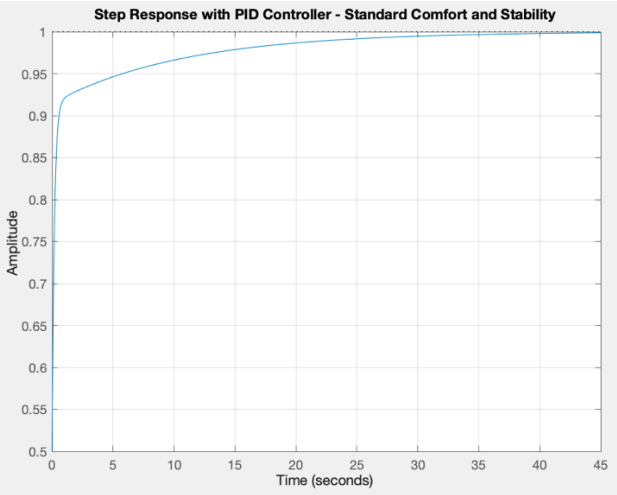


Figure 2.1: Step response for standard comfort and stability

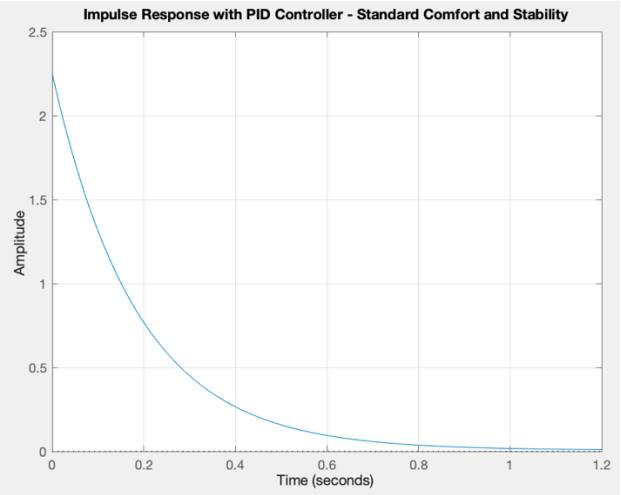


Figure 2.2: Impulse response for standard comfort and stability

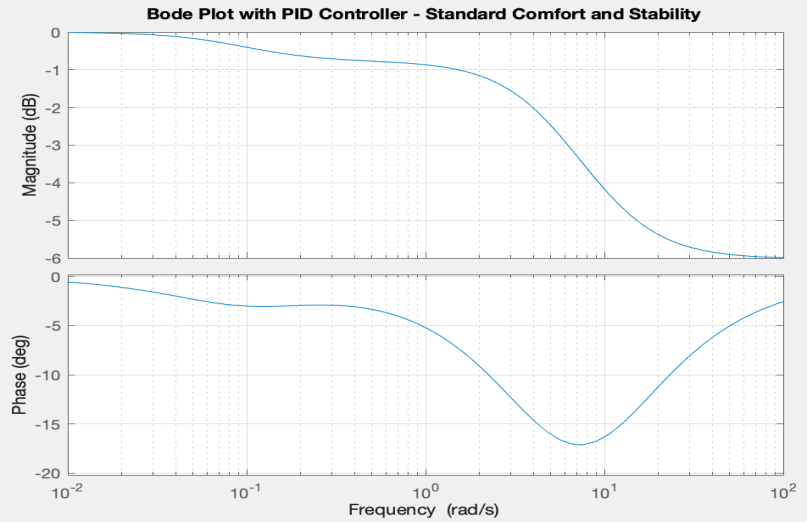


Figure 2.3: Bode Plot for standard comfort and stability

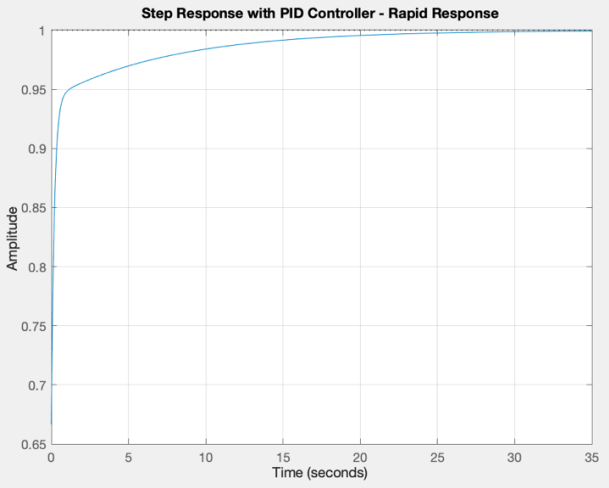


Figure 3.1: Step response for rapid response

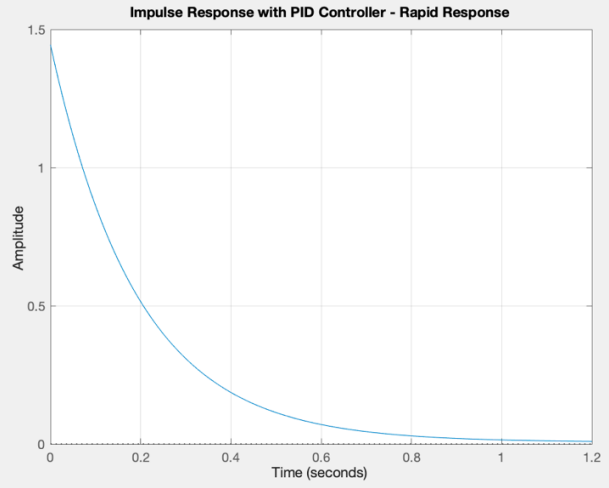


Figure 3.2: Impulse response for rapid response

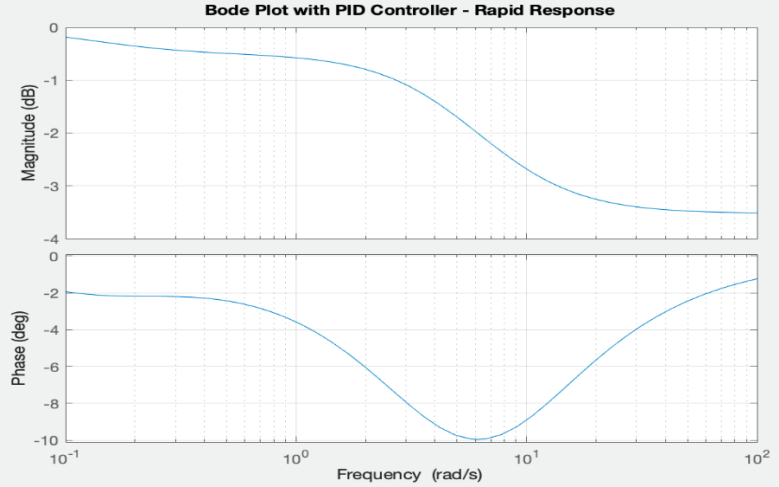


Figure 3.3: Bode Plot for rapid response

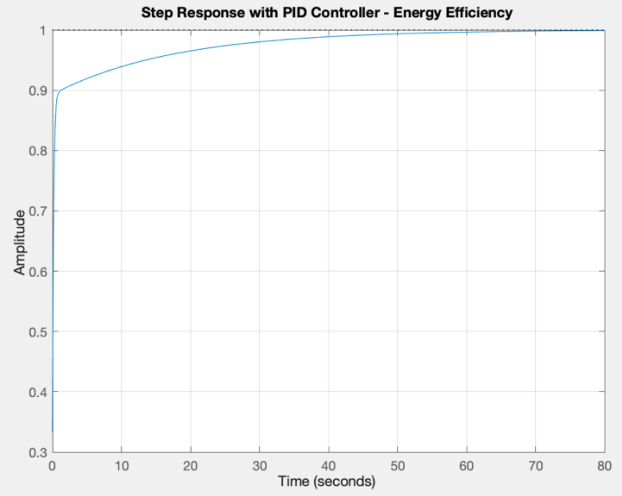


Figure 4.1: Step response for “Energy Efficiency”

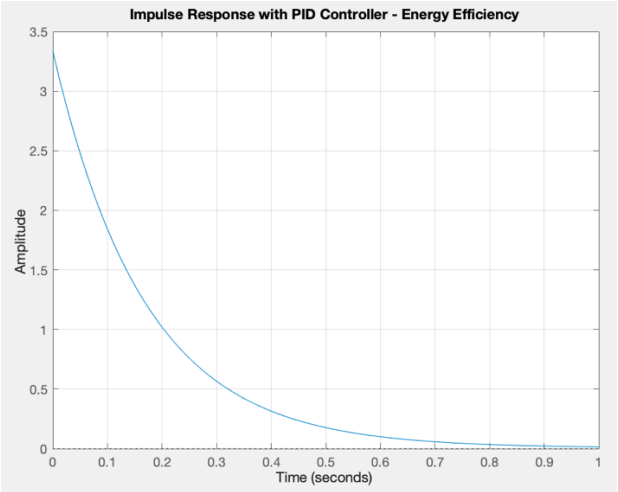


Figure 4.2: Impulse response for “Energy Efficiency”

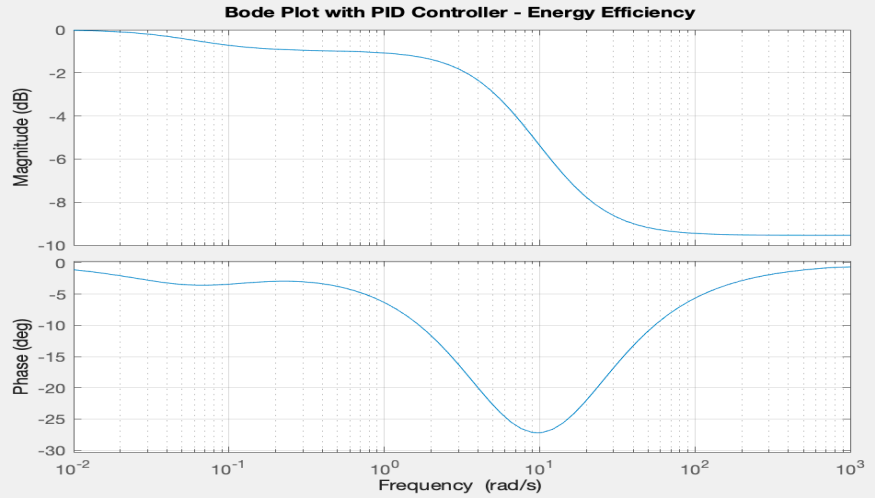


Figure 4.3: Bode Plot for "Energy Efficiency"

SIMULINK MODEL

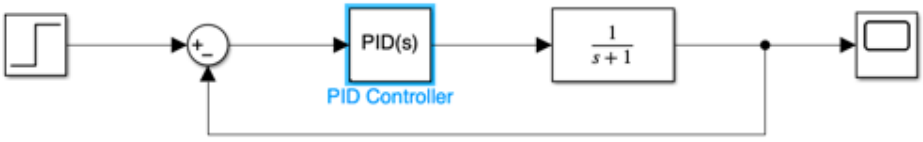


Figure 5.1: Simulink model

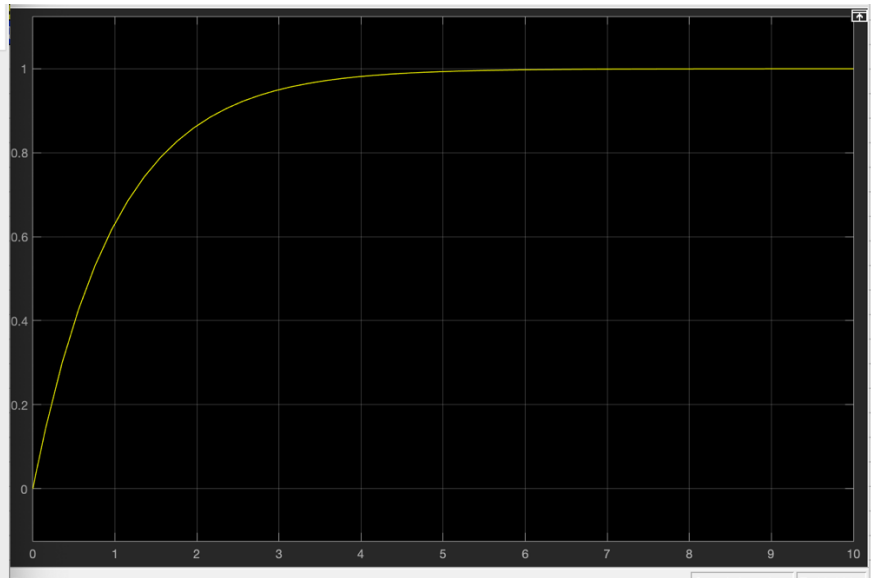


Figure 5.2: Output graph from Simulink scope

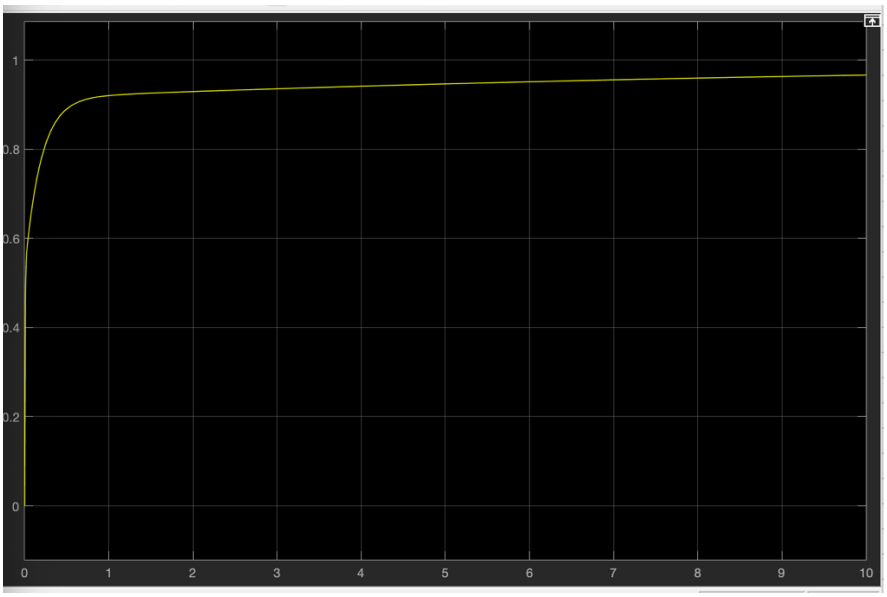


Figure 5.3: Figure 5.2: Output graph from Simulink scope PID controller “Standard comfort and stability”

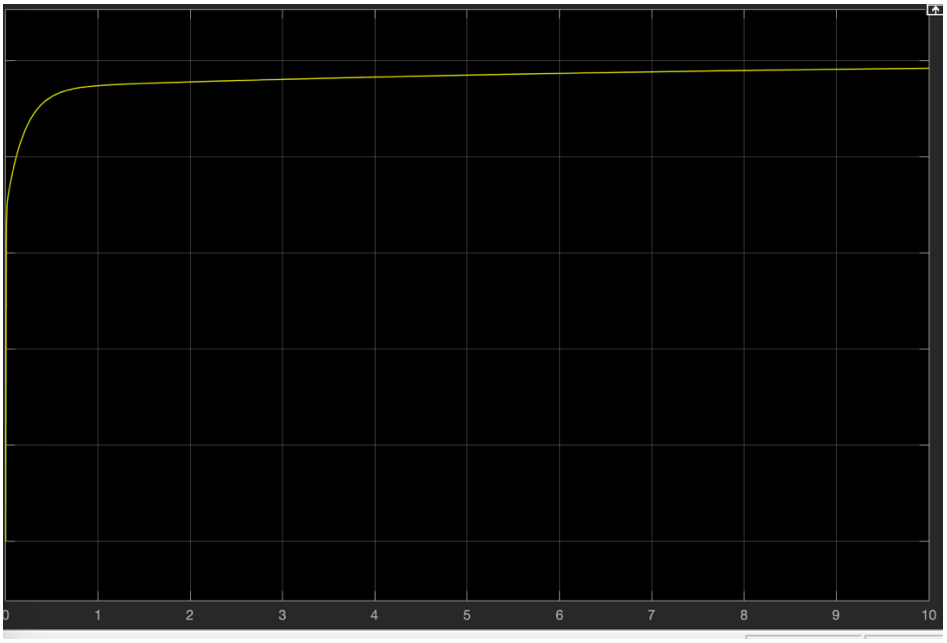


Figure 5.4: Figure 5.2: Output graph from Simulink scope PID controller “rapid response”

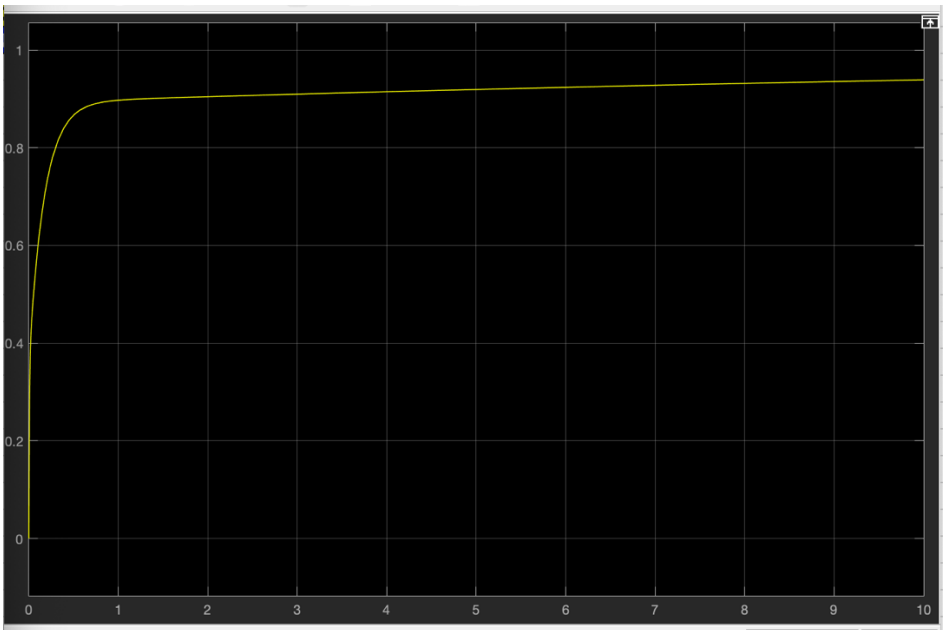


Figure 5.5: Figure 5.2: Output graph from Simulink scope PID controller “Energy Efficiency”

CONCLUSION

In this project, we successfully demonstrated the application of mathematical modeling and control theory in designing and testing PID controllers for a smart thermostat system. By utilizing the Root Locus method, we were able to achieve specific performance criteria across various scenarios, including standard comfort and stability, rapid response, and energy efficiency.

By using MATLAB and Simulink, we were able to model, simulate, and analyze the behavior of the smart thermostat system effectively. This allowed us to fine-tune the controllers and optimize system performance to meet our design objectives. The use of these tools provided a comprehensive understanding of the system dynamics and the practical challenges involved in control system design.

Our project highlighted the importance of establishing multiple design criteria to evaluate system performance under different conditions. Each design case presented unique

challenges and required careful tuning of the PID controllers to achieve the desired outcomes. This approach provided valuable insights into the trade-offs between stability, responsiveness, and energy efficiency.