Department of Electrical Engineering
University of Moratuwa
Control Systems – Design Home Assignment 1
220647K- M.Thiruvarankan

Assumptions

- The classroom air behaves as a lumped thermal mass (temperature is uniform throughout).
- Heat loss is through walls, windows, and doors only, modeled as one overall heat transfer coefficient.
- AC units provide constant cooling power when ON.
- No internal heat gains (e.g., people, lights, or equipment are not considered).
- The on-off controller maintains room temperature between 19.5°C and 20.5°C.
- Room operates for 5 hours per day and 210 days per year.
- Electricity cost is Rs 52 per unit (kWh) for institutional usage.

System Parameters

- Room dimensions: 10 m × 8 m × 4.5 m
- Volume, V = 360 m^3
- Air density, ρ = 1.18 kg/m^3
- Mass of air, m = 424.8 kg
- Specific heat, c = 1 kJ/kg·°C
- Heat transfer coefficient, U = 0.13 kJ/min°Cm^2
- Heat transfer area, A = 2[2(10x4.5)+(8x4.5)]+10x8=242 m^2
- Outside temperature, θ_{out} = 32°C
- Initial room temperature, θ_o = 31°C
- Setpoint temperature, θ_{set} = 20°C
- Cooling power of 2 AC units = 2 x 12000 = 24000 kJ/hr = 400 kJ/min

Energy Balance Equation

$$\mathbf{Q_{in} - Q_{out} = mc(d\theta/dt)}$$

where:

- Q_{in} is the heat input to the room (due to the air conditioner),
- Q_{out} is the heat loss through convection with the surroundings,
- m=ρV is the mass of air in the room
- c is the specific heat capacity of air,
- dθ/dt is the rate of change of the temperature in the room.

a)

Heat in flow rate due to leakages,

$$Qin = UA(\theta out - \theta)$$

For heat balance of the system,

$$Qin - Qout = mc\frac{d\theta}{dt}$$

$$UA(\theta outd - \theta) - Qout = mc\frac{d\theta}{dt}$$

$$-UA\theta + UA\theta outd - Qout = mc\frac{d\theta}{dt}$$

$$-\theta + \left(\theta outd - \frac{Qout}{UA}\right) = \frac{mc}{UA}\frac{d\theta}{dt}$$

Let $K1 = (\theta outd - \frac{Qout}{UA})$

$$-\theta + K1 = \frac{mc}{UA}\frac{d\theta}{dt}$$

$$\int dt = -\frac{mc}{UA}\int \frac{d\theta}{(\theta - K1)} + c1$$

Where c1 is the constant of the integration

$$t = -\frac{mc}{UA}\ln(\theta - K1) + c1$$

$$\ln(\theta - K1) = -\frac{UA}{mc}(t - c1)$$

$\theta - K1 = e^{-UA(t-c1)/mc}$
$\theta - K1 = e^{UAc1/mc} \times e^{-UAt/mc}$
$\theta - K1 = B \times e^{-UAt/mc}$
$\theta(t) = B\, e^{-UAt/mc} + K1$

B is a constant determined by using the boundary condition $\theta(0) = \theta o$,

$\theta(0) = \theta o = B + K1$
$B = \theta o - K1$
$\theta(t) = (\theta o - K1)\, e^{-UAt/mc} + K1$

where $K1 = (\theta outd - \frac{Qout}{UA})$

Analysis Without On-Off Controller:

In the case where no on-off control is implemented (i.e., the system operates without a thermostat or other regulation mechanism), the final steady-state temperature can be derived by taking the limit as $t \to \infty$:

$$\lim_{t\to\infty} \theta(t) = K1$$

Substituting the given values into the equation for K_1:

$$K_1 = 32 - \frac{(12000\ x\ 2)}{(60\ x\ 0.94787\ x\ 0.13\ x\ 242)} = 18.5861^{\circ}C$$

In theory, without any control mechanism, the air conditioner would keep running continuously, and the room temperature would slowly approach the calculated steady-state value of 18.5861°C. However, this leads to overcooling, energy wastage, and discomfort for occupants.

To avoid this, an on-off (bypass) controller is implemented. It works by:

- Turning the AC off when the temperature drops below a lower threshold (e.g., 19.5°C),
- Turning the AC on again when it rises above an upper threshold (e.g., 20.5°C).

This type of control:

- Maintains a comfortable temperature band,
- Reduces unnecessary power consumption, and
- Keeps the system efficient.

In our case, the fact that the room temperature only dropped to around 23°C, instead of reaching 18.5861°C, confirms the presence of an on-off controller. This ensures the system stays within a practical and energy-efficient range rather than continuing to cool beyond comfort levels.

Hysteresis and Controller Behavior

Let us now consider a hysteresis control band of ±0.5˚C around a setpoint temperature θ_{set}. The temperatures at which the air conditioner activates and deactivates are as follows:

- $\theta(t_1)=\theta_{set}-0.5$
- $\theta(t_2)=\theta_{set}+0.5$
- $\theta(t_3)=\theta_{set}-0.5$

The respective time durations during which the system remains within this band can be computed using Equation (1). The expressions for these times are:

$$t_1 = -\frac{mc}{UA}\ln\left(\frac{\theta set-0.5-K1}{\theta 0-K1}\right)$$

$$t_2 = -\frac{mc}{UA}\ln\left(\frac{\theta set+0.5-\theta out}{\theta set-0.5-\theta out}\right)$$

$$t_3 = -\frac{mc}{UA}\ln\left(\frac{\theta set-0.5-K1}{\theta set+0.5-K1}\right)$$

These equations account for the temperature changes as the system cycles through the hysteresis control limits.

Average Outdoor Temperature Calculation

Using monthly and weekly outdoor temperature data, we calculate the yearly average outdoor temperature θ_{avg}:

$$\theta_{avg}=\frac{(27x44)+(28x70)+(29x181)+(30x216)+(31x209)+(32x198)+(33x80)+(34x52)}{(44+70+181+216+209+198+80+52)}=30.57^{\circ}C.$$

Substituting this into the heat balance equation,

$$K_1=30.57-\frac{(12000\ x\ 2)}{(60\ x\ 0.94787\ x\ 0.13\ x\ 242)}=17.1561^{\circ}C$$

Duty Cycle and Energy Estimation

To estimate the energy usage and duty cycle, we assume the room's initial temperature θ_0=31˚C. After using the temperature model, we calculate the times during which the air conditioner operates,

Cooling: Time from 31˚C to 19˚C,

$t_1 = -\frac{10\ X\ 8\ X\ 4.5\ X\ 1.18\ X\ 1}{0.13\ X\ 242} \ln\left(\frac{20-0.5-17.1561}{31-17.1561}\right)$ = 23.98 minutes

Heating: Time from 19.5°C to 20.5°C (AC is OFF)

$t_2 = -\frac{10\ X\ 8\ X\ 4.5\ X\ 1.18\ X\ 1}{0.13\ X\ 242} \ln\left(\frac{20+0.5-30.57}{20-0.5-30.57}\right)$ = 1.25 minutes

Cooling again: 20.5°C to 19.5°C (AC is ON again)

$t_3 = -\frac{10\ X\ 8\ X\ 4.5\ X\ 1.18\ X\ 1}{0.13\ X\ 242} \ln\left(\frac{20-0.5-17.1561}{20+0.5-17.1561}\right)$ = 4.8 minutes

Thus, the duty cycle is:

$$\frac{t3}{t2+t3} = \frac{4.8}{1.25+4.8} = 0.7938$$

Energy Consumption Estimate

The energy consumption is calculated for daily, monthly, and annual usage based on the duty cycle. The energy used per day is calculated as,

Cooling time to reach steady temperature = 2 x 23.98 = 47.96min

Remaining time in a 5-hour day(300 minutes) = 300 – 47.96 = 252.04min

Daily energy usage(2 AC units),

Power = 2812+120/1000 = 2.932kW

Total daily energy = (47.96+0.7938x252.04) x (2.932/60) = 12.07 kWh/day

The annual consumption is,

= 12.07 x 210 = 2534.7 kWh

The monthly usage is,

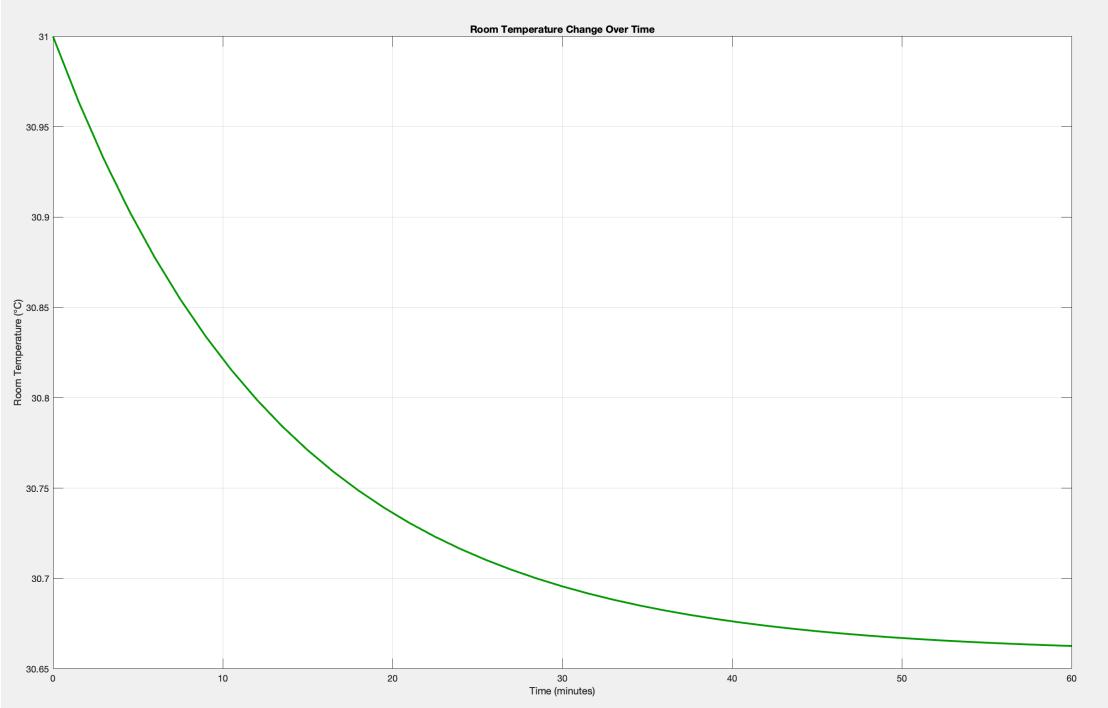
= 2534.7/12 = 211.225 kWh

Finally, the annual cost is calculated as ,

= (52 x 211.225 + 2000) x12 = Rs. 155,804.40

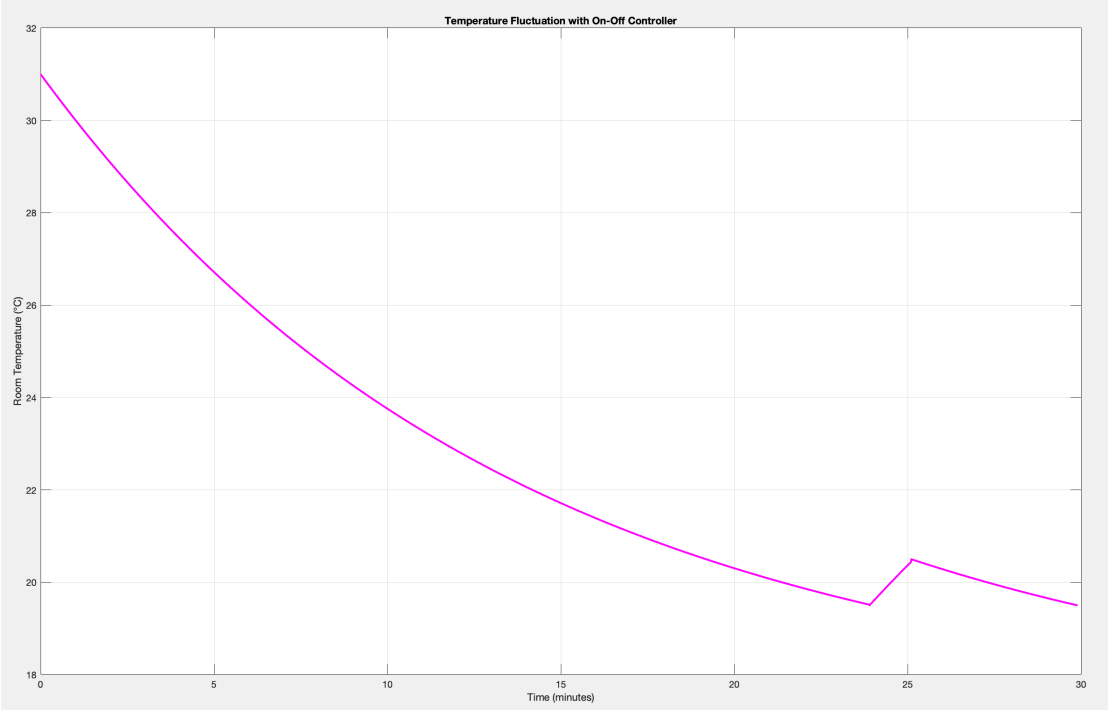
Graphs

1)



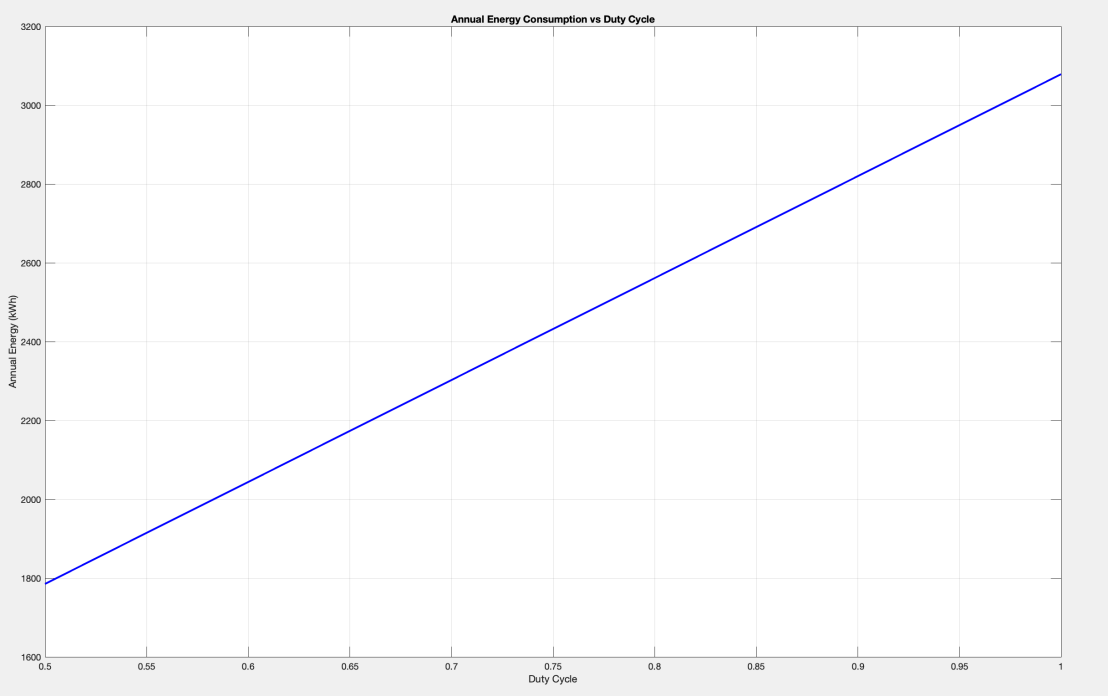
This graph shows how the room cools down over time without any control. The temperature keeps dropping and goes below the comfortable level.

2)



This graph shows how the on-off controller keeps the room temperature between 19.5°C and 20.5°C by turning the AC on and off.

3)



This graph shows the estimated yearly energy use based on how often the AC runs. Higher duty cycle means more energy used.

b) To estimate the annual savings when the school sets the air conditioner temperature exclusively at 26°C, we assume a hysteresis band width for the on-off control of $2\Delta\theta = 1°C$.

Using the system model, the cooling and heating durations during the temperature cycle are calculated as follows:

- Cooling Time (t_1): 6.84 minutes
- Heating Time (t_2): 2.96 minutes
- Re-cooling Time (t_3): 1.53 minutes

The duty cycle is calculated by dividing the re-cooling time (t_3) by the total cycle time ($t_2 + t_3$),

$$\text{Duty Cycle} = t_3/(t_2+t_3) = 1.53 / (1.53+2.96) = 0.34$$

To estimate daily energy consumption, we use the duty cycle and the operational time for the air conditioners. The energy used per day is calculated as,

Daily Energy Consumption = (6.84 x 2 + (5x60-6.84x2)0.34)(1/(60x1000))x(2812+120)

= 5.42kWh/day

Annual Energy Consumption = 5.42 x 210 = 1138.2 kWh/year

Monthly Energy Consumption = 1138.2/12 = 94.85kWh/month

The cost of energy is calculated by multiplying the monthly energy consumption by the cost per unit of electricity (Rs. 52 per kWh), and adding a fixed monthly cost (Rs. 1000) for institutional use,

Annual Energy Cost = 52 x 94.85kWh + 1000 x 12 = Rs.34,764

Annual Savings,

Savings = 155,804.40 – 34,764 = Rs.121,040.40

By setting the air conditioning temperature to 26°C exclusively, the school can save approximately **Rs. 121,040.40** annually.

c)

Yes, I recommend using inverter-type air conditioners instead of on-off type units for better energy performance. Because,

1. Continuous Adjustment for Efficiency: Inverter air conditioners adjust their compressor speed depending on the room temperature and the set temperature. This means they can work at lower speeds when the room is close to the set temperature, using less power. On-off systems, on the other hand, constantly turn the compressor on and off, which uses more energy.
2. Energy Efficiency: When an on-off air conditioner starts, it uses a lot of energy due to inrush currents. In contrast, inverter units gradually ramp up their compressor speed, avoiding this high energy consumption at startup. Once the desired temperature is reached, the inverter system can maintain it efficiently, saving energy in the process.
3. Stable Temperature: Inverter air conditioners maintain a steady temperature. On-off systems can cause temperature swings, where the room becomes too hot or too cold because the compressor stops and starts repeatedly. With an inverter system, the temperature stays closer to the set point, providing more comfort and better control.
4. Longer Compressor Life: The constant starting and stopping of the compressor in on-off systems can wear out the components over time. Inverter units, by gradually adjusting the compressor speed, reduce wear and tear, leading to longer lifespan for the compressor.

Proportional (P) Controller for Inverter ACs:

For an inverter-based air conditioning system, the energy balance equation can be written as,

$$UA(\theta_{out}-\theta)+ \dot{Q} = mc(d\theta/dt)$$

where $\dot{Q} = k_p(\theta_{set}-\theta)$ represents the heat input controlled by the P controller. This gives the following equation,

$$mc(d\theta/dt) + (UA+k_p)\theta = UA\theta_{out} + k_p\theta_{set}$$

Rewriting this, we get:

$$a = \frac{UA+kp}{mc}, b = \frac{UA\theta out + kp\theta set}{mc}$$

The solution to this equation is,

$$\theta(t)=Ce^{-at}+ (b/a)$$

Where $\theta(t)$ is the temperature at time t, and the steady-state value θ_∞ is,

$$\theta_\infty=b/a$$

The goal is to set the steady-state temperature equal to the desired set temperature (θ_{set}) but with a proportional controller, there will still be some steady-state error. The error depends on the proportional gain k_p, and increasing k_p reduces the response time, but it doesn't eliminate the steady-state error.

Estimating Response Time:

To estimate the time it takes to reach 95% of the steady-state value, we use the following equation,

$$\theta(t)=\theta_\infty+(\theta_0-\theta_\infty)e^{-at}$$

Here, $\theta_0=31°C$ is the initial temperature, and $\theta_{set}=26°C$ is the target temperature. We can solve for t, which gives us the response time,

$$t = -\frac{\ln(0.05)}{a}$$

This shows that the response time is inversely proportional to the proportional gain k_p. In other words, increasing k_p results in a faster response time, but it also increases the steady-state error.

Proportional-Integral (PI) Controller:

To minimize the steady-state error and improve performance, a PI controller is used. The energy balance equation for a PI-controlled system is,

$$UA(\theta out - \theta) + kp(\theta set - \theta) + kI\int(\theta set - \theta)dt = mc\frac{d\theta}{dt}$$

The PI controller adds an integral term to the equation, which helps eliminate the steady-state error by integrating the difference between the setpoint and the actual temperature over time. This allows for a faster and more accurate response**.**

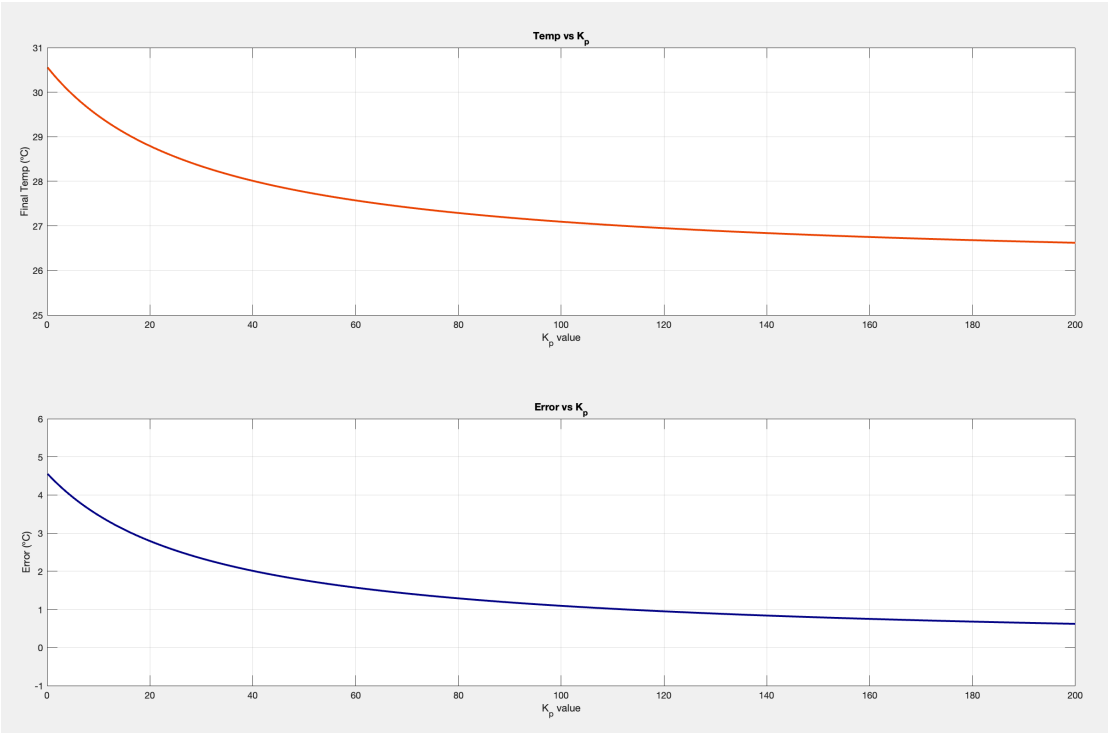
By adjusting the proportional (k_p) and integral (k_I) gains, we can achieve a balance between fast response and minimal steady-state error. For example, setting $k_p=10$ and $k_I=1200$ results in a system with low steady-state error and a fast response.

Increasing k_p results in a faster response, but adjusting k_I helps in reducing temperature fluctuations and improving the accuracy of temperature control.

- Inverter-based ACs are more efficient and sustainable than on-off systems because they adjust cooling power based on the room temperature.
- Using PI control strategies helps achieve faster response times and minimal steady-state error, which leads to better performance in maintaining the set temperature and reducing energy waste.

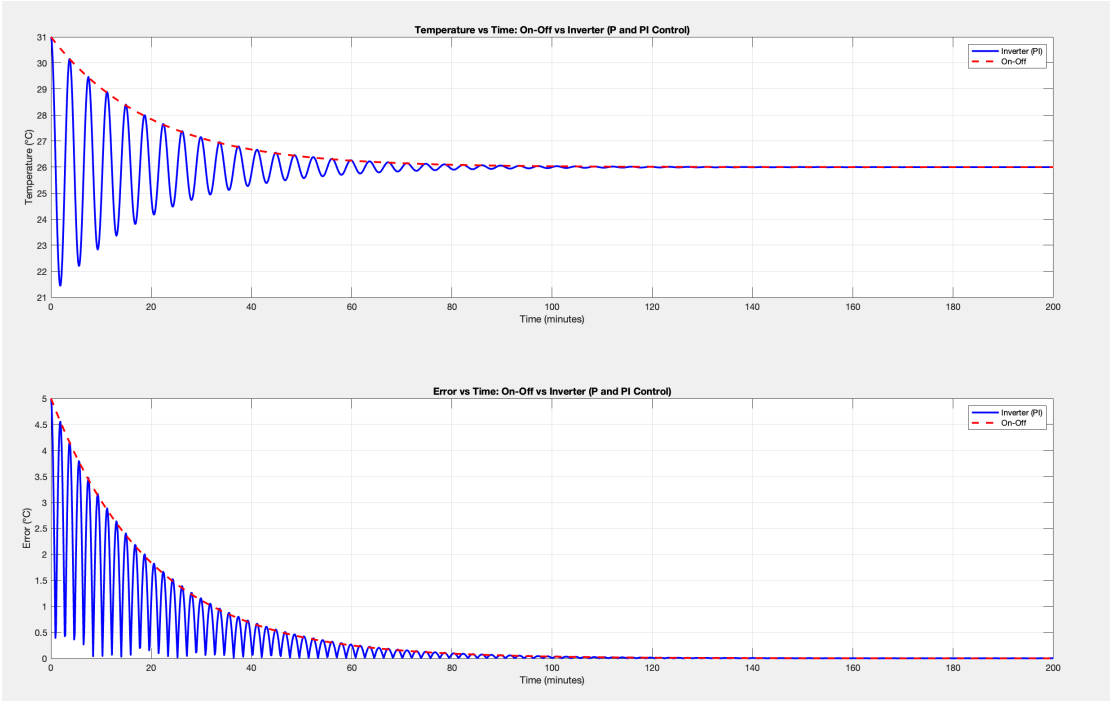
Graphs

1)



This graph shows how the final temperature and error change as the proportional gain (Kp) increases. As Kp increases, the system reaches a higher final temperature and the steady-state error decreases.

2)



These graphs compare the temperature and error over time for both on-off and inverter systems under proportional (P) and PI control. The temperature stabilizes faster and with less error in the inverter system, especially with the PI controller, compared to the on-off system.

d)

(i) Sudden Change in Set Temperature

Objective: Observe how well each controller handles changes in the desired indoor temperature.

Initial conditions:

- Indoor temperature is 35°C
- Outdoor temperature: Constant at 32°C
- Setpoint: 25°C for the first 10 minutes, then changed to 26°C
- Thermal parameters:
    - Heat transfer coefficient: U = 0.13 kJ/m^2·°C ·min
    - Surface area: A=242 m^2

P Controller Analysis

A basic P controller was tested by varying the proportional gain Kp, with the integral gain Ki set to zero. As shown in **Graph : P Controller Response for Varying Kp** (Ki = 0)**,** the system approaches the setpoint more quickly as Kp increases. However, a noticeable steady-state error (SSE) persists in all cases, since a P controller alone cannot eliminate SSE. This issue becomes more evident during setpoint changes or when external disturbances are introduced.

PI Controller Analysis

To evaluate the performance of the PI controller, we first selected an optimal proportional gain Kp that achieved effective temperature regulation without causing excessive overshoot. After testing several values, we found that Kp=140 offered a good balance between responsiveness and system stability.

We then varied the integral gain Ki to study the influence of integral action. The system responses are shown in **Graph : PI Controller Response for Varying Ki (Kp = 140)**. When Ki is small, the controller gradually corrects the steady-state error (SSE). As Ki increases, the integral component becomes more dominant, allowing quicker SSE elimination. However, excessively high Ki values lead to overshooting, where the temperature briefly drops below the setpoint due to excessive cooling.

(ii) Gradual Rise in Outdoor Temperature

Objective: Assess controller response when ambient temperature rises from 32°C to 34°C over an hour.

We modeled the variation of outdoor temperature using the function:

$$\theta out(t) = \frac{(34-32)}{60}t + 32 = \frac{1}{30}t + 32 \quad \text{, with t in minutes}$$

The indoor setpoint was kept constant at 26°C.

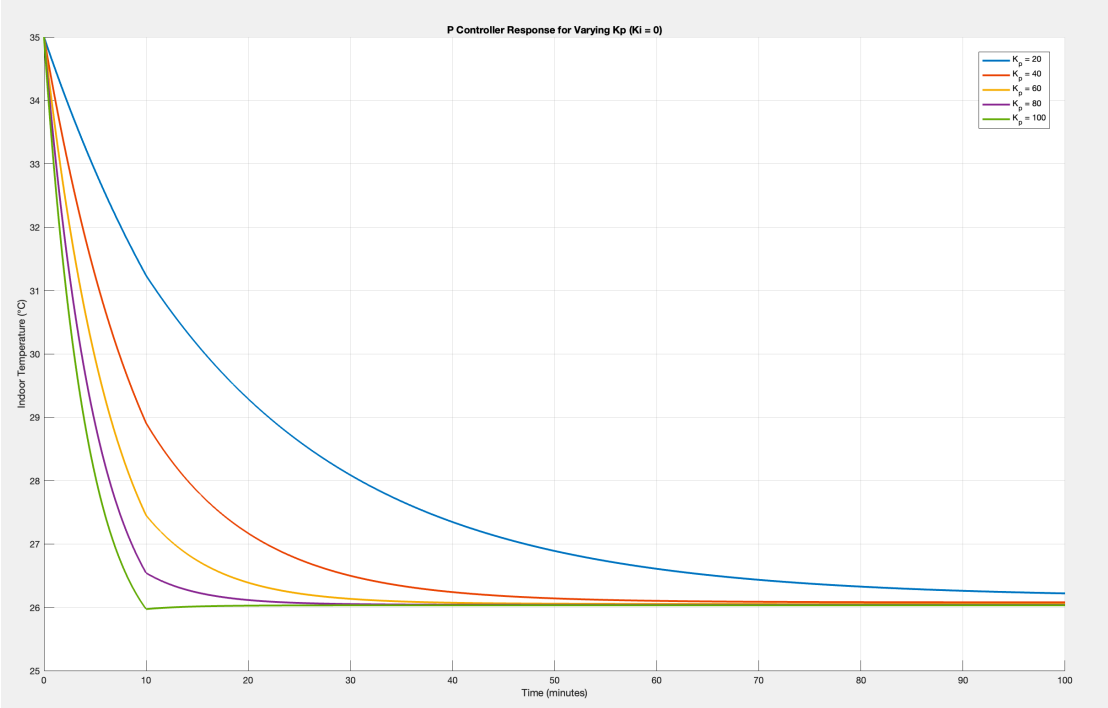
We analyzed the performance of the PI controller (with Kp=140) across different values of integral gain Ki. As shown in **Graph: Temperature Response for Varying Integral Gains**, when Ki=1, the controller was unable to maintain the setpoint effectively, and the temperature gradually drifted toward 27°C.
As Ki increased to 10 and 50, the system responded more effectively to the rising outdoor temperature, showing smaller fluctuations around the setpoint.
At higher values such as Ki=75 and Ki=100, the controller successfully minimized fluctuations, maintaining the indoor temperature within the desired range of 25°C to 27°C.

- **P Controller:** Cannot eliminate steady-state error (SSE) and performs poorly when setpoints change or disturbances occur.

- **PI Controller:** Eliminates SSE effectively when tuned correctly. It adapts to setpoint changes and external disturbances but requires careful tuning to avoid instability or oscillations.

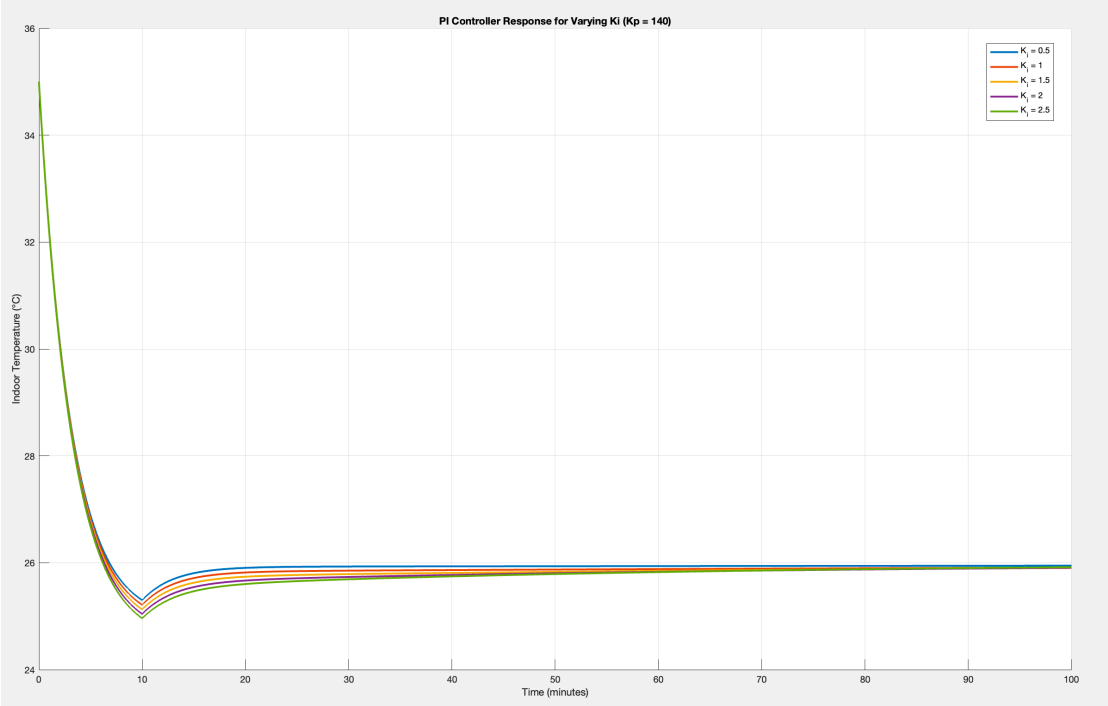
Graphs

1)



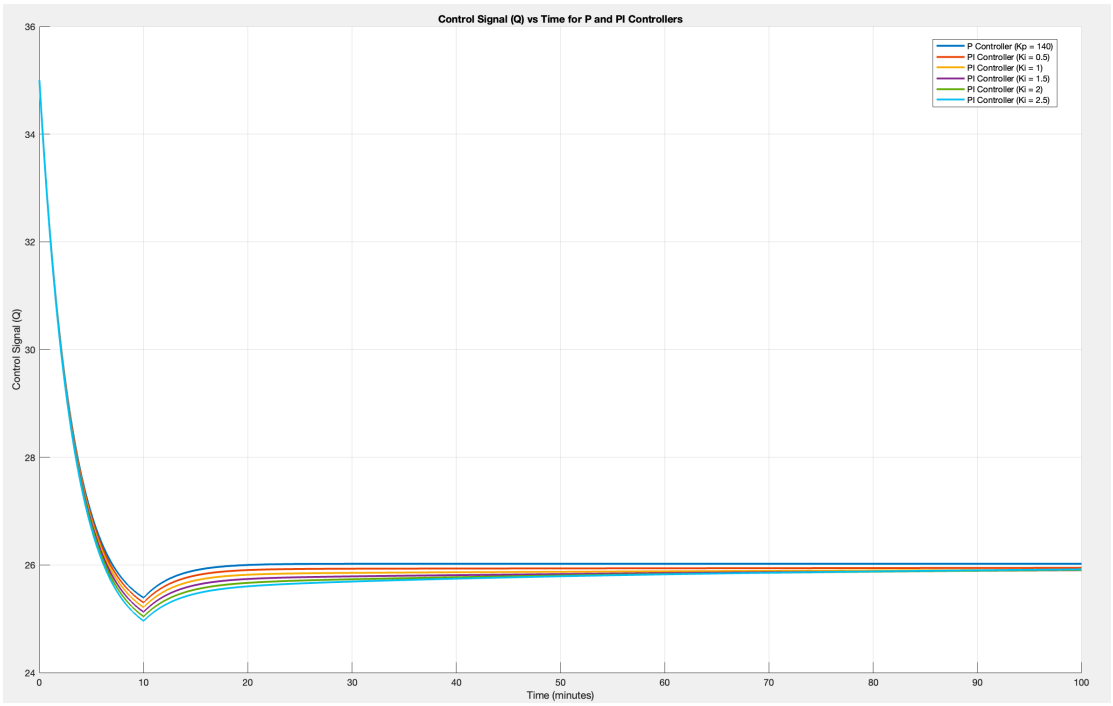
This graph shows how the indoor temperature (°C) changes over time for a P controller with different proportional gains (Kp values). It helps visualize the system's behavior with varying controller strength.

2)



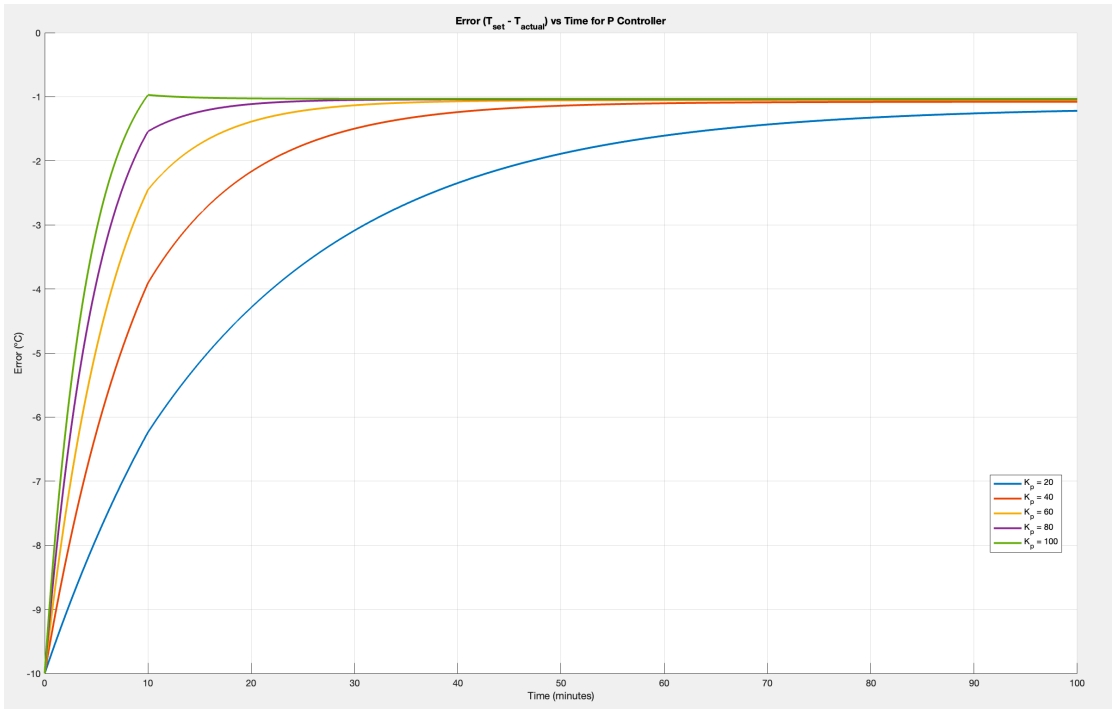
This graph illustrates the indoor temperature (°C) over time for a PI controller with varying integral gains (Ki values), while keeping the proportional gain (Kp) constant at 140. It shows how the integral term influences temperature control.

3)



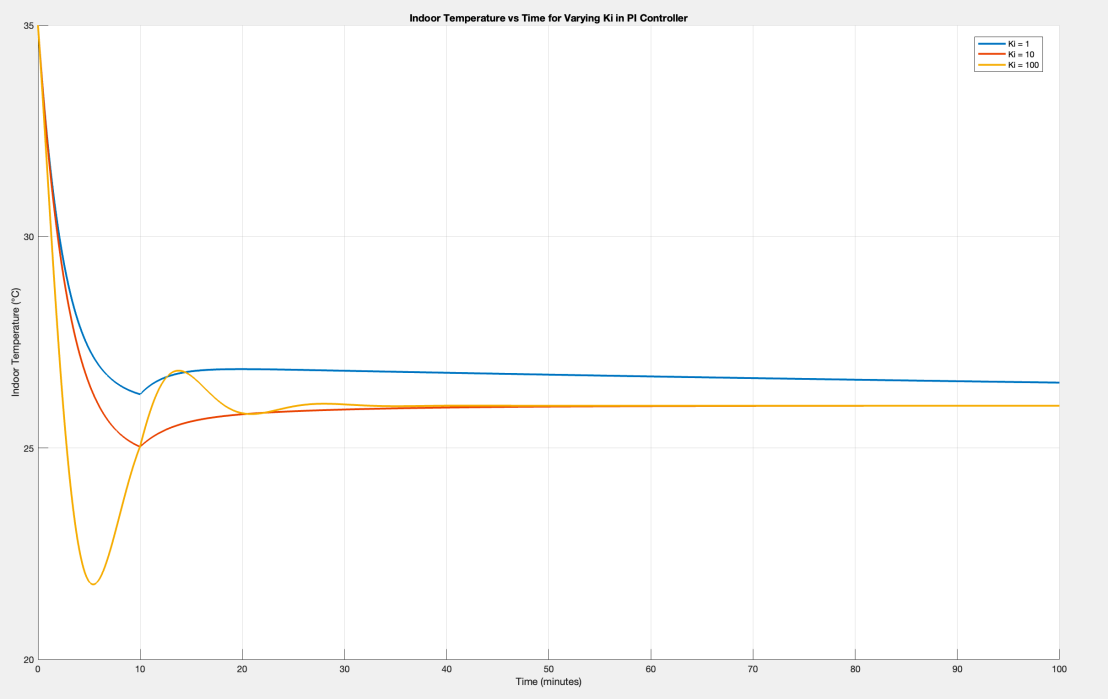
This graph compares the control signal (Q) over time for both P and PI controllers. It shows how the control effort changes with time for different gains, helping to compare their effectiveness in regulating the temperature.

4)



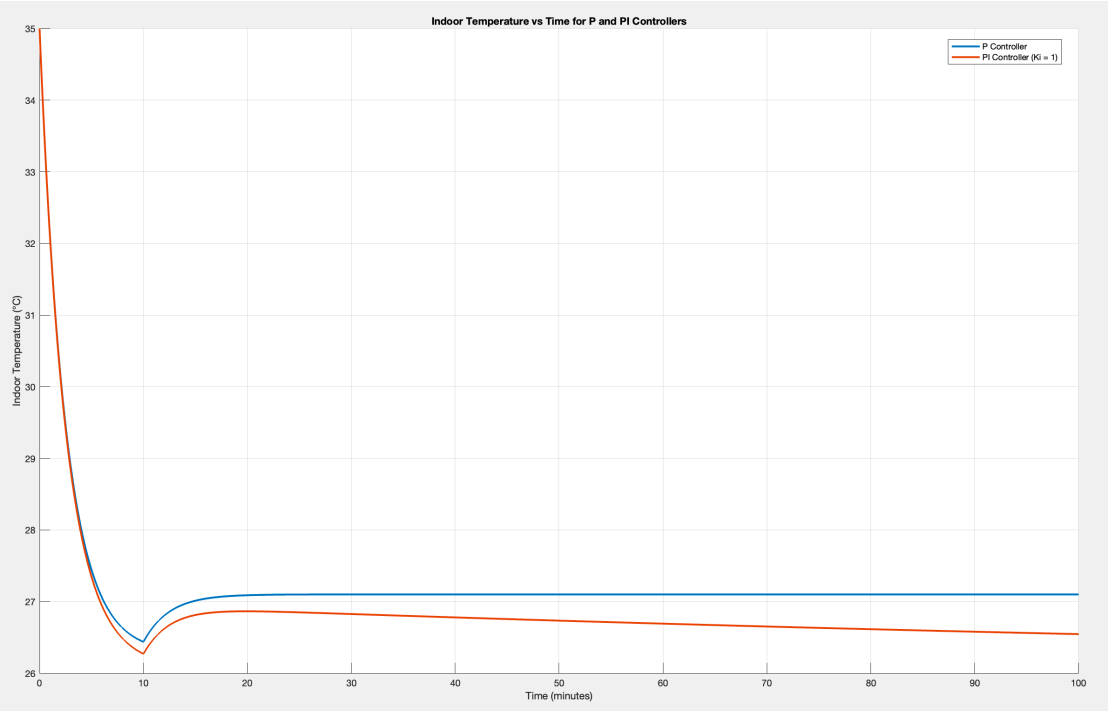
This graph plots the error between the target temperature (T_set) and the actual indoor temperature (T_actual) for varying Kp values in a P controller. It helps visualize how quickly the error reduces for different proportional gains.

5)



This graph shows how the indoor temperature (°C) evolves over time for a PI controller with different integral gains (Ki values), demonstrating how the integral component affects temperature stabilization.

6)



This graph compares the indoor temperature (°C) over time for both P and PI controllers, showing how each controller regulates temperature differently, with the PI controller typically providing smoother and more stable control.

e) We assume the following PI controller gains:

- Proportional gain Kp=120
- Integral gain Ki=0.3

The system cools the indoor temperature from **31°C** to a setpoint of **26°C**. The time needed for this change is calculated using the formula,

$$t = -\frac{mc}{UA}\ln\left(\frac{\theta\text{set} - \text{K1}}{\theta\text{o} - \text{K1}}\right)$$

Where,

- m=10×8×4.5×1.18=424.8 kg(thermal mass)
- U=0.13 kJ/min.m^2·°C (heat transfer coefficient)
- A=242 m^2(surface area)
- c = 1 kJ/kg°C(specific heat capacity)
- K_1=18.5861
- θset=26°C, θo=31°C

By substituting these values into equation of t,

t ≈ 7 minutes

Energy Consumption Calculation,

- Inverter power rating = 0.75 kW
- Time to reach setpoint = 7 minutes
- Usage per day = 2 times

Daily energy use,

= 0.75 x [(7 x 2)/60] = 0.175kWh

For 210 days in a year,

Annual Energy = 0.175 x 210 = 36.75kWh

Monthly Energy = 36.75/12 = 3.0625 kWh

Cost Calculation,

- Electricity cost = Rs. 4 per kWh
- Fixed monthly charge = Rs. 75

Monthly Cost = (4 x 3.0625) +75 = Rs.87.25

Annual Cost = 87.25 x 12 = Rs.1047

Matlab Codes

Graph 1:Room temperature Change over time

```
U = 0.13;
A = 242;
Q = 42.19987;
m = 10 * 8 * 4.5 * 1.18;
c = 1;
theta_out = 32;
theta0 = 31;

dtheta_dt = @(t, theta) (U*A/(m*c)) * (theta_out - theta) - Q/(m*c);

t_min = [0, 60];

[t, theta] = ode45(dtheta_dt, t_min, theta0);

plot(t, theta, 'Color', [0.2 0.6 0.1], 'LineWidth', 2);
xlabel('Time (minutes)');
ylabel('Room Temperature (°C)');
title('Room Temperature Change Over Time');
grid on;
```

Graph 2:Temperature Fluctuation with On-Off Controller

```
U = 0.13;
A = 242;
m = 10 * 8 * 4.5 * 1.18;
c = 1;
K1 = 17.1561;
theta_set = 20;
theta_low = theta_set - 0.5;
theta_high = theta_set + 0.5;

tau = (m * c) / (U * A);

t1 = 0:0.1:23.98;
t2 = 0:0.1:1.25;
t3 = 0:0.1:4.8;
```

```
theta1 = (31 - K1) * exp(-U*A*t1/(m*c)) + K1;
theta2 = (theta_low - 30.57) * exp(-U*A*t2/(m*c)) + 30.57;


theta3 = (theta_high - K1) * exp(-U*A*t3/(m*c)) + K1;

t_total = [t1, t1(end)+t2, t1(end)+t2(end)+t3];
theta_total = [theta1, theta2, theta3];

figure;
plot(t_total, theta_total, 'm', 'LineWidth', 2);
xlabel('Time (minutes)');
ylabel('Room Temperature (°C)');
title('Temperature Fluctuation with On-Off Controller');
grid on;
```

Graph 3:Annual Energy Consumption vs Duty Cycle

```
duty_cycle = linspace(0.5, 1, 100);
power_kw = 2.932;
daily_minutes = 300;
energy_daily = (power_kw / 60) * (47.96 + duty_cycle *
(daily_minutes - 47.96));
energy_annual = energy_daily * 210;

figure;
plot(duty_cycle, energy_annual, 'b', 'LineWidth', 2);
xlabel('Duty Cycle');
ylabel('Annual Energy (kWh)');
title('Annual Energy Consumption vs Duty Cycle');
grid on;
```

Graph 4: Final Temperature and Steady-State Error vs. Proportional Gain (Kp) for P Controller

```
T_out = 30.57;
T_set = 26;
H = 0.13 * 242;
Kp = linspace(0.1, 200, 500);
T_end = (H * T_out + Kp .* T_set) ./ (H + Kp);
T_error = T_end - T_set;

figure;

subplot(2,1,1);
plot(Kp, T_end, 'Color', [0.85 0.33 0.1], 'LineWidth', 2);
xlabel('K_p value');
ylabel('Final Temp (°C)');
title('Temp vs K_p');
grid on;
ylim([25 31]);

subplot(2,1,2);
plot(Kp, T_error, 'Color', [0 0 0.5], 'LineWidth', 2);
xlabel('K_p value');
ylabel('Error (°C)');
title('Error vs K_p');
grid on;
ylim([-1 6]);
```

Graph 5: Temperature and Error Response Over Time — On-Off vs. Inverter (PI Control)

```
UA = 0.13 * 242;
theta_out = 30;
theta_set = 26;
m = 10 * 8 * 4.5 * 1.18;
c = 1;
tspan = [0 200];
x0 = [31; 0];
Kp = 10;
Ki = 1200;

odefun = @(t, x) [
    (UA * (theta_out - x(1)) + Kp * (theta_set - x(1)) + Ki * x(2))
/ (m * c);
    theta_set - x(1)
];

[t, x_pi] = ode45(odefun, tspan, x0);

t_on_off = linspace(0, 200, 200);
theta_on_off = 31 + (theta_set - 31) * (1 - exp(-t_on_off / 20));

figure;
subplot(2,1,1);
plot(t, x_pi(:,1), 'b', 'LineWidth', 2);
hold on;
plot(t_on_off, theta_on_off, 'r--', 'LineWidth', 2);
xlabel('Time (minutes)');
ylabel('Temperature (°C)');
title('Temperature vs Time: On-Off vs Inverter (P and PI Control)');
legend('Inverter (PI)', 'On-Off');
grid on;

subplot(2,1,2);
plot(t, abs(x_pi(:,1) - theta_set), 'b', 'LineWidth', 2);
hold on;
plot(t_on_off, abs(theta_on_off - theta_set), 'r--', 'LineWidth',
2);
xlabel('Time (minutes)');
ylabel('Error (°C)');
title('Error vs Time: On-Off vs Inverter (P and PI Control)');
legend('Inverter (PI)', 'On-Off');
grid on;
```

Graph 6: P Controller Response to Setpoint Change for Varying Kp (Ki=0)

```
clc;
clear;
close all;

U = 0.13 / 60;
A = 242;
mass = 10 * 8 * 4.5 * 1.18;
c = 1;

Ki = 0;
T_out = 32;
```

```
T_set_initial = 25;


T_set_final = 26;
T_init = 35;
I_init = 0;

t_sim = linspace(0, 100, 10000);
init_state = [T_init; I_init];

Kp_list = [20, 40, 60, 80, 100];

figure;
hold on;

for Kp = Kp_list
    [t, y] = ode15s(@(t, y) p_control_model(t, y, U, A, mass, c, Kp,
Ki, T_out, T_set_initial, T_set_final), ...
                    t_sim, init_state, odeset('MaxStep', 0.005));
    plot(t, y(:,1), 'LineWidth', 2, 'DisplayName', ['K_p = '
num2str(Kp)]);
end

xlabel('Time (minutes)');
ylabel('Indoor Temperature (°C)');
title('P Controller Response for Varying Kp (Ki = 0)');
legend('Location', 'best');
grid on;
hold off;

function dydt = p_control_model(t, y, U, A, m, c, Kp, Ki, T_ext,
T_target_initial, T_target_final)
    temp = y(1);
    error_sum = y(2);

    if t > 10
        T_target = T_target_final;
    else
        T_target = T_target_initial;
    end

    err = T_target - temp;
    Q = Kp * err + Ki * error_sum;

    dydt = zeros(2,1);
    dydt(1) = (U * A * (T_ext - temp) + Q) / (m * c);
    dydt(2) = err;
end
```

Graph 7: P Controller Response for Varying Ki (Kp=140)

```
clc;
clear;
close all;

U = 0.13 / 60;
A = 242;
m = 10 * 8 * 4.5 * 1.18;
c = 1;
```

```
Kp = 140;
T_out = 32;
T_set_initial = 25;
T_set_final = 26;
T_init = 35;
I_init = 0;

t_sim = linspace(0, 100, 10000);
init_state = [T_init; I_init];

K_i_values = [0.5, 1.0, 1.5, 2.0, 2.5];

figure;
hold on;

for K_i = K_i_values
    [t, y] = ode15s(@(t, y) pi_control_model(t, y, U, A, m, c, Kp,
K_i, T_out, T_set_initial, T_set_final), ...
                    t_sim, init_state, odeset('MaxStep', 0.005));
    plot(t, y(:,1), 'LineWidth', 2, 'DisplayName', ['K_i = '
num2str(K_i)]);
end

xlabel('Time (minutes)');
ylabel('Indoor Temperature (°C)');
title('PI Controller Response for Varying Ki (Kp = 140)');
legend('Location', 'best');
grid on;
hold off;

function dydt = pi_control_model(t, y, U, A, m, c, Kp, K_i, T_ext,
T_target_initial, T_target_final)
    temp = y(1);
    integral_error = y(2);

    if t > 10
        T_target = T_target_final;
    else
        T_target = T_target_initial;
    end

    error = T_target - temp;
    Q = Kp * error + K_i * integral_error;

    dydt = zeros(2, 1);
    dydt(1) = (U * A * (T_ext - temp) + Q) / (m * c);
    dydt(2) = error;
end
```

Graph 8: Control Signal (Q) vs Time for P and PI Controllers

```
clc;
clear;
close all;

U = 0.13 / 60;
A = 242;
m = 10 * 8 * 4.5 * 1.18;
c = 1;
```

```
T_out = 32;
T_set_initial = 25;
T_set_final = 26;
T_init = 35;
I_init = 0;

t_sim = linspace(0, 100, 10000);
init_state = [T_init; I_init];

Kp = 140;
Ki = 0; % For P controller
K_i_values = [0.5, 1.0, 1.5, 2.0, 2.5]; % For PI controller

% P Controller Analysis
figure;
hold on;
for Kp = Kp
    [t, y] = ode15s(@(t, y) control_signal_model(t, y, U, A, m, c,
Kp, Ki, T_out, T_set_initial, T_set_final), ...
                    t_sim, init_state, odeset('MaxStep', 0.005));
    plot(t, y(:,1), 'LineWidth', 2, 'DisplayName', ['P Controller
(Kp = ' num2str(Kp) ')']);
end

% PI Controller Analysis
for K_i = K_i_values
    [t, y] = ode15s(@(t, y) control_signal_model(t, y, U, A, m, c,
Kp, K_i, T_out, T_set_initial, T_set_final), ...
                    t_sim, init_state, odeset('MaxStep', 0.005));
    plot(t, y(:,1), 'LineWidth', 2, 'DisplayName', ['PI Controller
(Ki = ' num2str(K_i) ')']);
end

xlabel('Time (minutes)');
ylabel('Control Signal (Q)');
title('Control Signal (Q) vs Time for P and PI Controllers');
legend('Location', 'best');
grid on;
hold off;

function dydt = control_signal_model(t, y, U, A, m, c, Kp, Ki,
T_ext, T_target_initial, T_target_final)
    temp = y(1);
    integral_error = y(2);

    if t > 10
        T_target = T_target_final;
    else
        T_target = T_target_initial;
    end

    error = T_target - temp;
    Q = Kp * error + Ki * integral_error;

    dydt = zeros(2,1);
    dydt(1) = (U * A * (T_ext - temp) + Q) / (m * c);
    dydt(2) = error;
end
```

Graph 9: Error vs Time for P Controller (Varying Kp, Ki=0)

```
clc;
clear;
close all;

U = 0.13 / 60;
A = 242;
m = 10 * 8 * 4.5 * 1.18;
c = 1;

T_out = 32;
T_set_initial = 25;
T_set_final = 26;
T_init = 35;
I_init = 0;

t_sim = linspace(0, 100, 10000);
init_state = [T_init; I_init];

Kp_list = [20, 40, 60, 80, 100];

figure;
hold on;

for Kp = Kp_list
    [t, y] = ode15s(@(t, y) error_model(t, y, U, A, m, c, Kp, 0,
T_out, T_set_initial, T_set_final), ...
                    t_sim, init_state, odeset('MaxStep', 0.005));
    plot(t, T_set_initial - y(:,1), 'LineWidth', 2, 'DisplayName',
['K_p = ' num2str(Kp)]);
end

xlabel('Time (minutes)');
ylabel('Error (°C)');
title('Error (T_{set} - T_{actual}) vs Time for P Controller');
legend('Location', 'best');
grid on;
hold off;

function dydt = error_model(t, y, U, A, m, c, Kp, Ki, T_ext,
T_target_initial, T_target_final)
    temp = y(1);
    integral_error = y(2);

    if t > 10
        T_target = T_target_final;
    else
        T_target = T_target_initial;
    end

    error = T_target - temp;
    Q = Kp * error + Ki * integral_error;

    dydt = zeros(2,1);
    dydt(1) = (U * A * (T_ext - temp) + Q) / (m * c);
    dydt(2) = error;
end
```

Graph 10: Indoor Temperature vs Time for Varying Ki in PI Controller

```
clc;
clear;
close all;

U = 0.13;
A = 242;
m = 10 * 8 * 4.5 * 1.18;
c = 1;
Kp = 140;
T_out = 32;
T_set_initial = 25;
T_set_final = 26;
T_init = 35;
I_init = 0;

t_sim = linspace(0, 100, 10000);
init_state = [T_init; I_init];

K_i_values = [1, 10, 100];

figure;
hold on;

for K_i = K_i_values
    [t, y] = ode15s(@(t, y) pi_control_model(t, y, U, A, m, c, Kp,
K_i, T_out, T_set_initial, T_set_final), ...
                    t_sim, init_state, odeset('MaxStep', 0.005));
    plot(t, y(:,1), 'LineWidth', 2, 'DisplayName', ['Ki = '
num2str(K_i)]);
end

xlabel('Time (minutes)');
ylabel('Indoor Temperature (°C)');
title('Indoor Temperature vs Time for Varying Ki in PI Controller');
legend('Location', 'best');
grid on;
hold off;

function dydt = pi_control_model(t, y, U, A, m, c, Kp, Ki, T_ext,
T_target_initial, T_target_final)
    temp = y(1);
    integral_error = y(2);

    if t > 10
        T_target = T_target_final;
    else
        T_target = T_target_initial;
    end

    error = T_target - temp;
    Q = Kp * error + Ki * integral_error;

    dydt = zeros(2,1);
    dydt(1) = (U * A * (T_ext - temp) + Q) / (m * c);
    dydt(2) = error;
end
```

Graph 11: Indoor Temperature vs Time for P and PI Controllers

```
clc;
clear;
close all;

U = 0.13;
A = 242;
m = 10 * 8 * 4.5 * 1.18;
c = 1;
Kp = 140;
T_out = 32;
T_set_initial = 25;
T_set_final = 26;
T_init = 35;
I_init = 0;

t_sim = linspace(0, 100, 10000);
init_state = [T_init; I_init];

% P Controller (Ki = 0)
figure;
hold on;

[t, y] = ode15s(@(t, y) pi_control_model(t, y, U, A, m, c, Kp, 0,
T_out, T_set_initial, T_set_final), ...
                t_sim, init_state, odeset('MaxStep', 0.005));
plot(t, y(:,1), 'LineWidth', 2, 'DisplayName', 'P Controller');

% PI Controller (Ki = 1)
[t, y] = ode15s(@(t, y) pi_control_model(t, y, U, A, m, c, Kp, 1,
T_out, T_set_initial, T_set_final), ...
                t_sim, init_state, odeset('MaxStep', 0.005));
plot(t, y(:,1), 'LineWidth', 2, 'DisplayName', 'PI Controller (Ki =
1)');

xlabel('Time (minutes)');
ylabel('Indoor Temperature (°C)');
title('Indoor Temperature vs Time for P and PI Controllers');
legend('Location', 'best');
grid on;
hold off;

function dydt = pi_control_model(t, y, U, A, m, c, Kp, Ki, T_ext,
T_target_initial, T_target_final)
    temp = y(1);
    integral_error = y(2);

    if t > 10
        T_target = T_target_final;
    else
        T_target = T_target_initial;
    end

    error = T_target - temp;
    Q = Kp * error + Ki * integral_error;

    dydt = zeros(2,1);
    dydt(1) = (U * A * (T_ext - temp) + Q) / (m * c);
    dydt(2) = error;
end
```

Note: I have uploade my Matlab codes and Respective output graphs here,
https://dms.uom.lk/s/fLZXW9KNtenfTt2