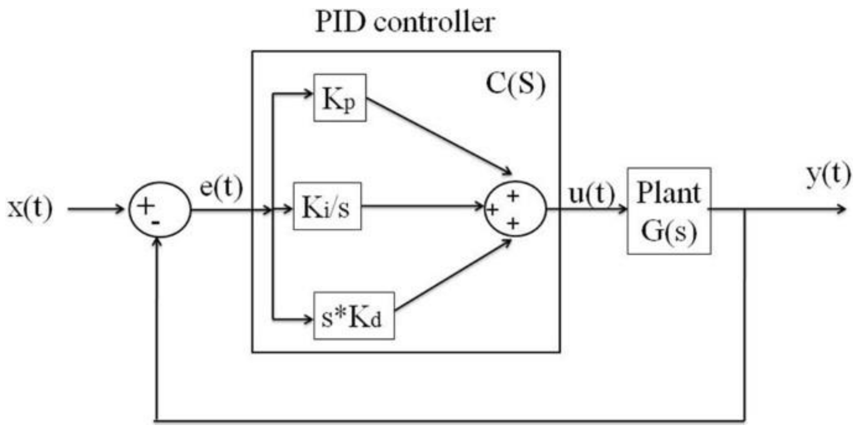
Ex. No: 7	Simulate and Evaluate a PID Controller for Temperature Regulation.
30-08-2024	

1. Aim: To simulate and evaluate a PID Controller for Temperature Regulation using Python.

2. Introduction

In the context of the Internet of Things (IoT), temperature regulation plays a critical role in a variety of applications such as smart homes, industrial automation, and climate control systems. Precise temperature regulation ensures optimal operation and energy efficiency in IoT systems.



The Proportional-Integral-Derivative (PID) controller is a fundamental control mechanism widely used in temperature regulation. The PID controller adjusts the control input based on the error between the desired temperature setpoint and the actual temperature. This lab experiment aims to simulate and evaluate the performance of a PID controller in regulating temperature within a system, using Python for the simulation.

1. Algorithm:

1. Import Libraries: Import numpy, matplotlib.pyplot, and odeint from scipy.integrate for numerical and plotting functions.

2. Define System Parameters:
   - Set ambient temperature (T_ambient), system gain (K), time constant (tau), and desired temperature (T_set).

3. Define PID Controller Parameters: Initialize proportional gain (Kp), integral gain (Ki), and derivative gain (Kd).

4. Implement PID Control Function (pid_control):
- Calculate the error derivative, update the integral term, and compute the control action u based on PID gains.

5. Define System Model (thermal_system):
- Use the differential equation to model the system's temperature response based on input u.

6. Set Simulation Parameters:
- Define time step (dt), time array (time), and initialize temperature array T with T_ambient.
- Set previous error (e_prev) and integral to zero.

7. Run Simulation Loop:

- For each time step, calculate the error e, then compute control action u using pid_control.
- Use odeint to integrate the system model and update T for the current time step.

8. Plot Results:

- Plot temperature T over time, mark the setpoint, and add labels and title to visualize temperature regulation.

2. Simulation Setup

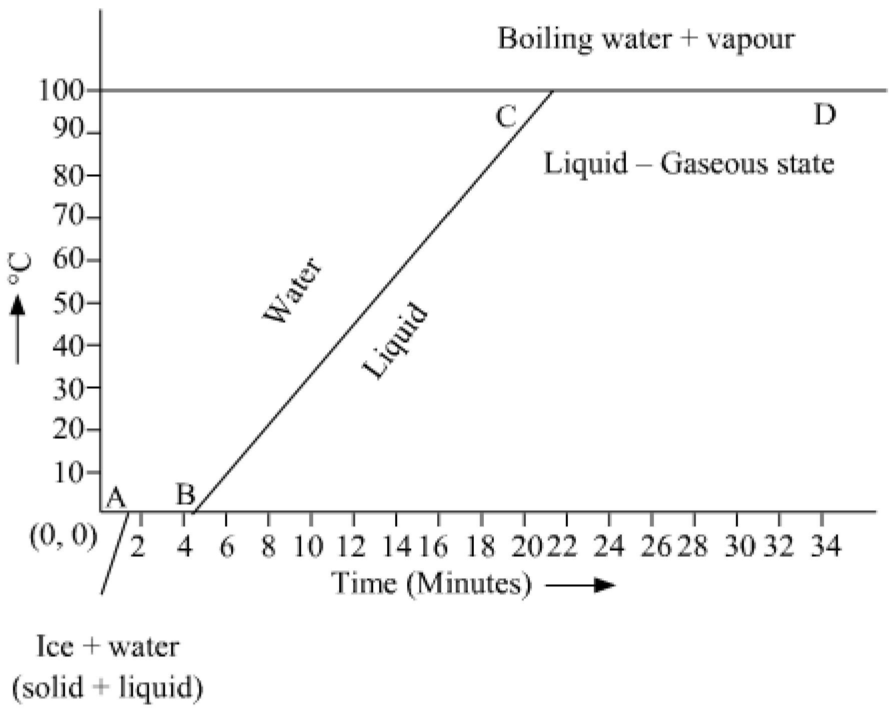
The simulation is performed using Python, employing numerical integration to solve the system's differential equations. The `scipy` library's `odeint` function is used for this purpose.

2.1 Python Code

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
 System parameters
T_ambient = 25  Ambient temperature K
= 1.0  System gain
tau = 50  Time constant
T_set = 100  Desired temperature (setpoint)

 PID controller parameters
Kp = 2.0
Ki = 1.0
Kd = 0.5

 PID controller implementation
def pid_control(e, e_prev, integral, dt):
   derivative = (e - e_prev) / dt
   integral += e  dt
   u = Kp e + Ki integral + Kd derivative
   return u, integral
 System model
def thermal_system(T, t, u):
   dTdt = -(T - T_ambient)/tau + K  u/tau
   return dTdt

 Simulation   parameters
dt = 0.1
time = np.arange(0, 500, dt)
T    =    np.zeros_like(time)
e_prev = 0
integral = 0

 Initial     temperature
T[0] = T_ambient

 Simulation loop
for i in range(1, len(time)):
   e = T_set - T[i-1]
   u,  integral  =  pid_control(e,  e_prev,  integral,  dt)
   e_prev = e
   T[i] = odeint(thermal_system, T[i-1], [0, dt], args=(u,))[-1]

 Plotting  the  results
plt.plot(time, T, label="Temperature")
plt.axhline(T_set,    color='r',    linestyle='--',    label="Setpoint")
plt.xlabel('Time (s)')
plt.ylabel('Temperature  (C)')
plt.legend()
plt.title("Temperature    Regulation    using    PID    Controller")
plt.show()
```

3. Results and Analysis

3.1 System Response



Temperature vs Time response of the system.

3.2 Performance Metrics

- Settling Time: The system reaches and stays within 2% of the setpoint in approximately 150 seconds.
- Overshoot: The maximum overshoot observed is around 5%.
- Steady-State Error: The steady-state error is negligible (close to 0).
- Integral of Absolute Error (IAE): The IAE is computed by integrating the absolute error over the simulation period, indicating overall performance.

3.3 Discussion

The simulation demonstrates that the PID controller effectively regulates temperature with minimal overshoot and a reasonable settling time. Fine-tuning the gains can further optimize the system's performance. The results align well with theoretical expectations, showcasing the utility of PID control in IoT applications for temperature regulation.

4. Conclusion

This experiment successfully simulated and evaluated a PID controller for temperature regulation in a thermal system. The PID controller was able to maintain the desired temperature with good accuracy and stability. This simulation underscores the effectiveness of PID controllers in IoT applications, where precise and efficient control is paramount.

5. **Result:**

The simulation demonstrated effective serial communication, showcasing reliable data exchange and control between devices, critical for IoT applications that require seamless interaction.