National Technical University of Athens

School of Electrical and Computer Engineering

Comparative Analysis of Temperature Controllers for HVAC Systems
Neurofuzzy Control
Semester Project

Anastasia Christina Liva
03119029

Contents

Introduction

This project explores the implementation and evaluation of three different methods for controlling the temperature in a single-zone building: Multi-Layer Perceptron (MLP) [1], Proportional-Integral-Derivative (PID) control, and Long Short-Term Memory (LSTM) neural network-based control. Each of these methods offers distinct approaches to handling the dynamic and often unpredictable nature of indoor climate regulation.

The entire project is based on the Neuromancer framework tutorials, which are available in the referenced GitHub repository [1]. Specifically, I utilize the MLP version as it is uploaded there [2]. The other versions are modifications of the MLP version that I have created myself. The MLP is described here for comparative purposes and to provide theoretical background. Further information on the theoretical background can be found in the DPC_building_control.ipynb notebook [2].

Methodology

Background

Buildings are substantial consumers of energy, with HVAC systems being major contributors. Traditional rule-based control systems are still predominantly in use, despite advances in optimal control technologies such as Model Predictive Control (MPC) and Deep Reinforcement Learning (DRL). These advanced control methods promise significant energy savings and enhanced environmental sustainability, but their complexity poses challenges for practical deployment.

Differentiable Predictive Control (DPC)

DPC is a policy optimization algorithm that leverages the differentiability of dynamic system models. It integrates with various model representations such as differential equations, state-space models, and neural network architectures to optimize control policies. The neural control policy in this context is denoted as $u_k = \pi(x_k, R)$, where u_k represents the control actions at time k, and π is the policy function that takes system state measurements x_k and reference predictions R as inputs.

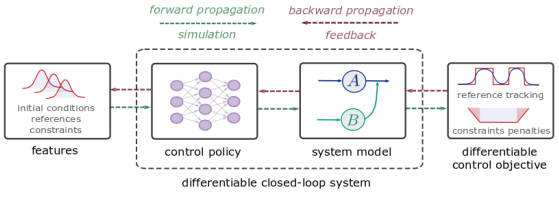

Figure 1: Schematics of a Building Control Scheme

Why Use Differentiable Predictive Control? Differentiable Predictive Control (DPC) combines the principles of Model Predictive Control (MPC) with the flexibility of neural networks. MPC is a widely-used method in control systems that involves solving an optimization problem at each time step to determine the optimal control action. However, traditional MPC can be computationally expensive and challenging to implement in real-time for complex systems. DPC

addresses these challenges by incorporating differentiable models, which allow for efficient gradient-based optimization. This makes the control policy differentiable, enabling the use of powerful optimization techniques like gradient descent. By embedding the predictive model within a neural network framework, DPC can leverage large datasets to learn control policies that generalize well across different operating conditions and disturbances.

System Model

The system dynamics are compactly represented by Ordinary Differential Equations (ODEs), described as ODESolve($f(x_{ki}, u_{ki})$), where x_{ki} and u_{ki} denote the system states and control actions, respectively. The objective is to learn a neural control policy by solving the following parametric optimal control problem:

$$
\begin{aligned}
\underset{\theta}{\text{minimize}} \quad & \sum_{i=1}^{m} \Big(\sum_{k=1}^{N-1} Q_x ||x_k^i - r_k^i||_2^2 + Q_N ||x_N^i - r_N^i||_2^2 \Big) \\
\text{subject to} \quad & x_{k+1}^i = \text{ODESolve}(f(x_k^i, u_k^i)) \\
& u_k^i = \pi_\theta(x_k^i, R^i) \\
& 0 \leq x_k^i \leq 1 \\
& 0 \leq u_k^i \leq 1 \\
& x_0^i \sim \mathcal{P}_{x_0} \\
& R^i \sim \mathcal{P}_R
\end{aligned}
$$

The goal is to minimize the reference tracking error $\|x_{ki} - r_{ki}\|_2^2$ over a prediction horizon N, weighted by Q_x and Q_N for the terminal penalty. The control policy $\pi_\theta(x_{ki}, R_i)$ is optimized using stochastic gradient descent with parameters sampled from distributions P_{x_0} and P_R.

Implementation (MLP)

Mathematical Formulation

The state-space model describes how the building's temperature evolves over time based on the current state, control actions, and disturbances.

The state-space model of the building dynamics is given by:

$$
\begin{aligned}
x_{k+1} &= Ax_k + Bu_k + Ed_k \\
y_k &= Cx_k
\end{aligned}
$$

where:

- x_k is the state vector at time step k (e.g., temperature states of the building).
- u_k is the control input vector at time step k (e.g., heating/cooling power).
- d_k is the disturbance vector at time step k (e.g., external temperature).
- y_k is the output vector at time step k (e.g., observed temperature).

The matrices A, B, C, and E are derived from the physical properties of the building and its thermal dynamics.

This equation updates the state vector x_k at each time step based on the influence of control actions u_k and disturbances d_k:

The output equation maps the state vector to the observed temperature.

Data Generation

The data generation process simulates various scenarios of building operation, including different initial conditions, disturbance profiles (e.g., weather changes), and temperature setpoints. This helps in training and validating the control policy under diverse conditions.

The get_data function generates training and validation datasets:

- Reference Trajectories: Desired temperature bounds sampled from a uniform distribution between 18°C and 22°C.
- Disturbance Trajectories: Simulated profiles of external temperature changes.
- Initial Conditions: Randomly sampled initial temperatures.

Control Policy

The control policy determines the appropriate heating or cooling power to apply in order to maintain the indoor temperature within the desired range.

The control policy is implemented using a neural network:

$$u_k = \pi(y_k, y_{\min}, y_{\max}, d_{\text{obs}})$$

where:

- y_k is the current observed temperature.
- $y_{\min}$ and $y_{\max}$ are the lower and upper bounds of the desired temperature range.
- d_{obs} is the observable part of the disturbance.

The neural network used in this implementation is a Multi-Layer Perceptron (MLP). The network takes as input the current temperature y_k, the lower and upper bounds of the desired temperature range $y_{\min}$ and $y_{\max}$, and the observable disturbances d_{obs}. It outputs the control actions u_k required to maintain the indoor temperature within the desired range.

The neural network has the following structure:

- Input layer size: $n_y + 2 \times n_{\text{ref}} + n_{d_{\text{obs}}}$
- Output layer size: n_u
- Hidden layers: Two layers with 32 units each and GELU activation functions.

Closed-Loop System

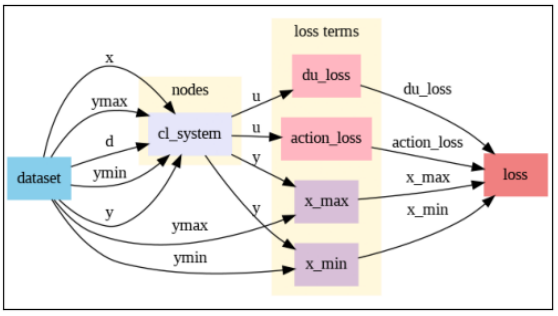

Figure 2: Closed Loop System

The closed-loop system simulates the entire control process, integrating the state-space model, neural network control policy, and disturbance observer. This allows the system to respond dynamically to disturbances and maintain the desired temperature range.

The closed-loop system comprises:

- **Disturbance Observer**: Estimates the observable part of the disturbance.
- **Control Policy**: Neural network determining the control actions.
- **State-Space Model**: Predicts the next state based on the current state, control input, and disturbance.
- **Output Model**: Maps the state to the observed temperature.

[H]

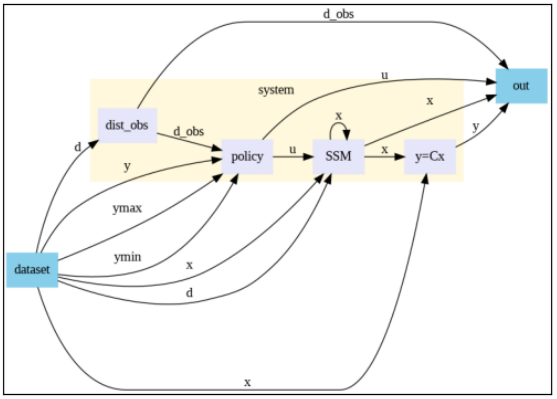

Figure 3: Closed Loop System

Objectives and Constraints

The controller aims to minimize energy usage and prevent abrupt control actions while ensuring the temperature stays within the comfort range. Objectives guide the optimization towards energy

efficiency and smooth operation, while constraints enforce comfort bounds.

Mathematical Formulation

Objectives:

1. **Action Loss:** Minimizes energy usage.

$$\text{action\_loss} = 0.01 \times \|u\|_2^2$$

2. **Delta U Loss:** Prevents aggressive changes in control actions.

$$\text{du\_loss} = 0.1 \times \|u_k - u_{k-1}\|_2^2$$

Constraints:

Thermal Comfort Constraints: Ensure the temperature remains within comfort bounds.

$$\text{state\_lower\_bound\_penalty} = 50.0 \times \max(0, y_k - y_{\max})$$

$$\text{state\_upper\_bound\_penalty} = 50.0 \times \max(0, y_{\min} - y_k)$$

Optimization

The optimization problem aims to find the best control actions that balance energy efficiency, comfort, and smooth operation. The neural network is trained to minimize the combined loss function, learning to output optimal control actions based on current states and disturbances. The problem is solved using the AdamW optimizer over 200 epochs.

Conclusions

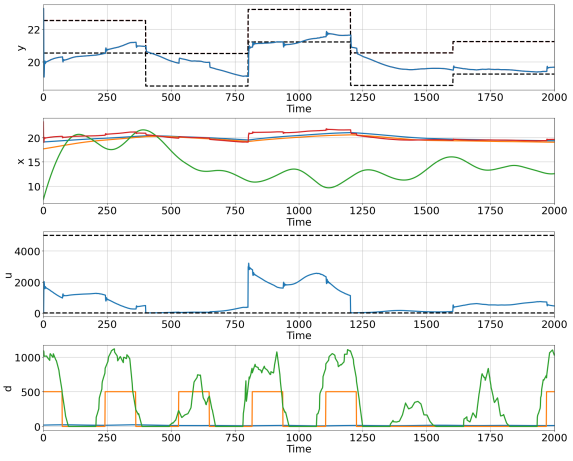

Figure 4: MLP plots

Temperature Control (Top Subplot)

The top subplot shows the indoor temperature y (blue line) in comparison to the reference temperature bounds (black dashed lines). These bounds indicate the desired temperature range, with the lower bound around 18°C and the upper bound around 22°C.

The MLP controller generally maintains the temperature within the desired bounds, although there are notable deviations. Around the 750 and 1250 time steps, the temperature approaches or slightly exceeds the upper and lower bounds, indicating moments of less precise control. Despite these deviations, the MLP controller demonstrates the ability to respond to changes and disturbances, but it sometimes struggles to maintain tight control, leading to occasional temperature excursions outside the desired range.

State Variables (Second Subplot)

The second subplot presents the state variables x of the system, which influence the temperature control. These states represent various internal aspects of the building's thermal dynamics, such as wall temperatures and HVAC system status.

The state variables show pronounced oscillations. The green line, representing a critical state variable, shows significant fluctuations, indicating that the MLP controller induces variability in the internal states. The red and blue lines, representing other state variables, also show fluctuations, suggesting that the MLP controller's internal control mechanism is reactive but somewhat unstable. These oscillations reflect the controller's effort to adjust the internal conditions to maintain the desired temperature, but they also indicate a less smooth internal state management.

Green Line The green line represents one of the critical state variables that is likely related to the thermal inertia or heat capacity of the building. This state variable shows how the building's internal temperature responds to external disturbances and control inputs over time. The oscillations in the green line indicate the variability in the thermal state due to changing external conditions and the controller's attempts to stabilize the temperature.

Red Line The red line represents another state variable that could be associated with a specific component of the building's thermal system, such as the temperature of the walls or internal air mass. This state variable tends to show a more stable trend compared to the green line, indicating that it may be influenced by longer-term thermal dynamics or slower processes within the building. The relatively stable nature of the red line suggests that it represents a state with less sensitivity to rapid changes in control inputs and disturbances.

Blue Line The blue line likely represents yet another state variable, which could be linked to the HVAC system's operational status or another thermal property of the building. Similar to the red line, the blue line shows a stable trend with minor fluctuations, indicating that it may capture a component of the building's thermal state that changes more gradually. The stability of the blue line reflects the underlying thermal inertia or capacity that responds more slowly to changes in the control inputs and external disturbances.

Control Inputs (Third Subplot)

The third subplot reveals that the control input u adjusts dynamically in response to changes in the system's state and external disturbances. Peaks in the control input correspond to periods

where significant heating or cooling is required, indicating that the control policy is responsive and adaptive to varying conditions.

The frequent and sometimes abrupt control actions suggest that the MLP controller actively attempts to correct temperature deviations. However, this reactive approach may lead to higher energy consumption and potential wear on HVAC components due to constant adjustments. The control inputs show a peak around 750 time steps, which corresponds to a significant correction needed to bring the temperature back within bounds. This indicates that the MLP controller is responsive but may overcompensate at times, leading to inefficiencies.

Handling External Disturbances: The fourth subplot shows the actual disturbances d and the observable disturbances d_obs. The control policy utilizes d_obs to make informed decisions, effectively compensating for external fluctuations such as changes in outdoor temperature or occupancy. This capability is crucial for maintaining indoor thermal comfort in real-world scenarios.

Minimized Energy Consumption: The control policy includes objectives for minimizing energy use (action_loss) and avoiding aggressive control actions (du_loss). The resulting control inputs exhibit smooth transitions and minimal unnecessary fluctuations, suggesting that the policy effectively balances energy efficiency with temperature regulation.

Physical Interpretation

Physically, the state-space model represents the heat transfer dynamics within the building zone. The states x_k can include variables such as the temperatures of different building components (e.g., walls, air), which evolve over time based on the applied heating/cooling u_k and external conditions d_k. The output y_k, the indoor temperature, is what the occupants experience and what the controller aims to regulate.

The control actions u_k correspond to the heating or cooling power applied by HVAC (Heating, Ventilation, and Air Conditioning) systems. By optimizing these actions, the controller ensures that the indoor environment remains comfortable while minimizing energy consumption. The neural network's role is to provide a fast and accurate mapping from the current state and disturbances to the optimal control actions, allowing the system to respond dynamically to changing conditions.

PID Controller

A PID (Proportional-Integral-Derivative) controller adjusts the control input to a system based on three terms derived from the error signal. The error signal $e(t)$ is the difference between a desired setpoint $r(t)$ and the measured process variable $y(t)$:

$$e(t) = r(t) - y(t)$$

The control signal $u(t)$ is computed as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

where:

- K_p is the proportional gain,
- K_i is the integral gain,
- K_d is the derivative gain.

The proportional term $K_p e(t)$ provides an immediate response to the current error. The integral term $K_i \int_0^t e(\tau)d\tau$ accounts for the accumulation of past errors, helping eliminate steady-state error. The derivative term $K_d \frac{de(t)}{dt}$ predicts future errors based on the rate of change, improving the system's stability and response time.

```
class PIDController(nn.Module):
    def __init__(self, Kp, Ki, Kd, umin, umax):
        super(PIDController, self).__init__()
        self.Kp = Kp
        self.Ki = Ki
        self.Kd = Kd
        self.umin = umin
        self.umax = umax
        self.integral = 0.0
        self.prev_error = 0.0

    def forward(self, y, ymin, ymax, d_obs):
        error = (ymax + ymin) / 2.0 - y
        self.integral += error
        derivative = error - self.prev_error
        self.prev_error = error

        u = self.Kp * error + self.Ki * self.integral + self.Kd *
            derivative
        u = torch.clamp(u, self.umin, self.umax)

        return u
```

The controller takes the current system output y, the bounds for the reference temperature $y_{\min}$ and $y_{\max}$, and the observed disturbances d_{obs}. It computes the error as the difference between the midpoint of the reference bounds and the current output. The control action u is then calculated and clamped within the specified bounds $[u_{\min}, u_{\max}]$.

The PID controller node computes the control actions based on the system's output y, the reference temperature bounds $y_{\min}$ and $y_{\max}$, and the observed disturbances d_{obs}. The PID controller adjusts the control input to maintain the temperature within the desired bounds while compensating for disturbances.

Thermal Comfort Constraints

The thermal comfort of the occupants is maintained by ensuring the indoor temperature remains within a specified range $[y_{\min}, y_{\max}]$. These constraints are enforced by penalties in the objective function of the optimization problem:

$$L_{\text{comfort}} = 50 \cdot (y_k > y_{\min}) + 50 \cdot (y_k < y_{\max})$$

Objective Function

The overall objective function combines multiple goals:

- **Energy Minimization**: Minimize the control effort.
- **Smooth Control Actions**: Prevent aggressive changes in control actions to ensure smooth operation.

$$L = 0.01 \cdot (u == 0.0) + 0.1 \cdot (u_k - u_{k+1} == 0.0)$$

The closed-loop performance of the controller is evaluated through simulations over a prediction horizon of 100 time steps and tested over 2000 steps. The system responds to randomly generated disturbance trajectories and initial conditions, maintaining the temperature within the specified bounds while minimizing the energy consumption and ensuring smooth control actions.

Conclusions

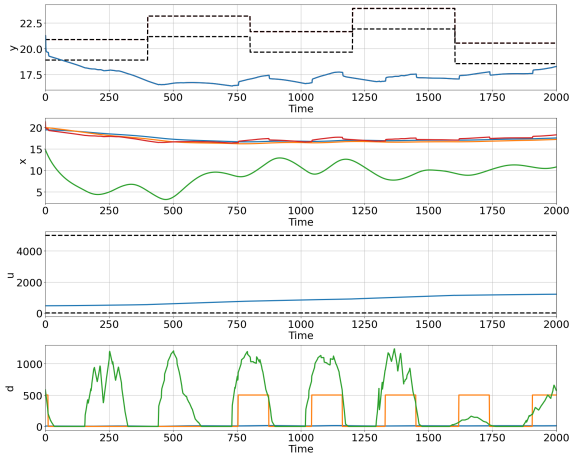

Figure 5: Enter Caption

Temperature Control (Top Subplot)

The top subplot shows the indoor temperature y (blue line) in comparison to the reference temperature bounds (black dashed lines). These bounds indicate the desired temperature range, with the lower bound around 18°C and the upper bound around 22.5°C.

The PID controller struggles to maintain the temperature within these bounds. For a significant portion of the time, especially in the first 1250 time steps, the temperature falls below the lower bound. This indicates that the PID controller is not sufficiently aggressive in its corrective actions to counteract disturbances and maintain the desired temperature range. The temperature gradually

declines and remains below the lower bound, showing a lack of responsiveness to increasing heating needs as external conditions change.

State Variables (Second Subplot)

The second subplot presents the state variables x of the system, which influence the temperature control. These states represent various internal aspects of the building's thermal dynamics, such as wall temperatures and HVAC system status.

The state variables show relatively smooth dynamics compared to other control strategies. The green line, representing a critical state variable, shows periodic oscillations but remains within a moderate range. The red and blue lines, representing other state variables, also show smooth and gradual changes. This smooth behavior suggests that the PID controller maintains stable internal states but at the cost of not effectively regulating the temperature. The stability in state variables indicates a conservative control approach that may not be adequately responsive to disturbances.

Control Inputs (Third Subplot)

The third subplot displays the control inputs u applied by the PID controller. The control inputs remain low and exhibit a gradual increase over time, staying well within the bounds of the allowable control range.

The smooth and minimal control actions suggest that the PID controller employs a conservative strategy, which avoids abrupt changes. However, this conservative approach results in insufficient corrective actions to maintain the desired temperature, especially when larger disturbances occur. The control inputs do not exhibit significant variations, reflecting a lack of aggressive heating or cooling actions needed to counteract the external influences and maintain thermal comfort.

Disturbances (Bottom Subplot)

The bottom subplot illustrates the disturbances d affecting the system. These disturbances represent external factors such as outdoor temperature variations and solar radiation, which impact the indoor temperature.

The disturbances show significant fluctuations over time, with high peaks and valleys. The orange line indicates periods where the disturbance model is active. Despite these disturbances, the PID controller's control inputs remain minimal and unchanged, highlighting its inability to respond effectively to external changes. The failure to adequately adjust control actions in response to these disturbances leads to the observed deviations in temperature.

LSTM

The Long Short-Term Memory (LSTM) network is a type of recurrent neural network (RNN) specifically designed to handle sequential data and capture long-term dependencies. The architecture consists of several key components:

- **Input Layer:** Receives the concatenated inputs, including the current temperature, reference bounds, and observable disturbances.

- **LSTM Layers:** Process the input sequence and maintain hidden states to capture temporal dependencies.
- **Fully Connected Layer:** Transforms the output of the LSTM into control actions.
- **Clamping Function:** Ensures the control actions remain within specified bounds.

Mathematical Operations

Input Concatenation

The input to the LSTM network is a concatenation of the current temperature (y), the minimum reference temperature ($y_{\min}$), the maximum reference temperature ($y_{\max}$), and the observable disturbances (d_{obs}):

$$x = \text{concat}(y, y_{\min}, y_{\max}, d_{\text{obs}})$$

The concatenated input x is then passed to the LSTM network.

LSTM Processing

The LSTM network processes the input sequence using the following equations:

Input Gate (i_t):	$i_t = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i)$
Forget Gate (f_t):	$f_t = \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f)$
Cell State Update ($\tilde{C}_t$):	$\tilde{C}_t = \tanh(W_C \cdot x_t + U_C \cdot h_{t-1} + b_C)$
Cell State (C_t):	$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$
Output Gate (o_t):	$o_t = \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o)$
Hidden State (h_t):	$h_t = o_t \odot \tanh(C_t)$

Where:

- σ is the sigmoid activation function.
- tanh is the hyperbolic tangent activation function.
- W_i, W_f, W_C, W_o are the input weights.
- U_i, U_f, U_C, U_o are the recurrent weights.
- b_i, b_f, b_C, b_o are the biases.
- $\odot$ denotes element-wise multiplication.

Fully Connected Layer

The final hidden state h_t from the last time step is passed through a fully connected (dense) layer to produce the control action u:

$$u = \text{ReLU}(W_{\text{fc}} \cdot h_t + b_{\text{fc}})$$

Here, W_{fc} and b_{fc} are the weights and biases of the fully connected layer.

Clamping

To ensure that the control actions u remain within the specified bounds $(u_{\min}, u_{\max})$, the output is clamped:

$$u = \text{clamp}(u, u_{\min}, u_{\max})$$

The `torch.clamp` function ensures that each element of u is within the range defined by $u_{\min}$ and $u_{\max}$.

Network Architecture

The network architecture is defined as follows:

- **Input size:** Combined dimensions of the current temperature, reference bounds, and observable disturbances.
- **Hidden size:** 32 units to capture the underlying patterns in the data.
- **Output size:** Number of control inputs, which in this case corresponds to the heating/cooling actions.
- **Number of layers:** 2 layers to enhance the model's capacity to learn complex patterns.

The LSTM processes the input sequence and outputs control actions that are clamped within specified bounds to ensure they remain feasible.

Closed-loop System Model

The closed-loop system model integrates various components to simulate the overall behavior of the temperature control system. The central part of this model is the state-space representation of the building dynamics. In this model, the state vector x_k at time step k represents the internal state of the building, which could include factors like current temperature and other relevant physical parameters. The control input vector u_k represents the actions taken by the control system, such as heating or cooling inputs. The disturbance vector d_k captures external influences on the system, like weather conditions or occupancy levels.

The state-space model is governed by two main equations. The state update equation $x_{k+1} = Ax_k + Bu_k + Ed_k$ describes how the state vector evolves over time based on the current state, control inputs, and disturbances. Here, A, B, and E are matrices that define the relationships between these variables. The output equation $y_k = Cx_k$ relates the state vector to the output vector y_k, which in this case is the temperature that we want to control. The matrix C maps the state variables to the temperature output.

In addition to the state-space model, the closed-loop system includes a disturbance observation model and an LSTM-based control policy. The disturbance observation model extracts observable disturbances from the full disturbance vector, allowing the system to account for measurable external influences. The LSTM-based control policy is a neural network designed to process the current temperature, reference temperature bounds, and observable disturbances to predict the necessary control actions. This neural network uses Long Short-Term Memory (LSTM) units, which are well-suited for handling sequential data and capturing temporal dependencies.

Training Objectives and Constraints

The training process of the LSTM-based control policy aims to optimize the control actions by minimizing specific objectives while satisfying certain constraints. These objectives and constraints ensure that the controller maintains the desired temperature range, operates efficiently, and ensures smooth transitions.

Action Loss

One of the primary objectives is the action loss, which aims to minimize energy usage by penalizing the magnitude of the control actions. Mathematically, the action loss is represented as:

$$\text{Action Loss} = \lambda_a \sum_{k=0}^{N-1} \|u_k\|^2$$

where λ_a is a scaling factor that determines the weight of the action loss in the overall objective function, u_k is the control input vector at time step k, and N is the total number of time steps. This quadratic penalty encourages the controller to use smaller control inputs, thereby minimizing energy consumption.

Delta U Loss

The delta U loss penalizes abrupt changes in control actions to ensure smooth transitions. It is mathematically represented as:

$$\text{Delta U Loss} = \lambda_{\text{du}} \sum_{k=1}^{N-1} \|u_k - u_{k-1}\|^2$$

where λ_{du} is a scaling factor that determines the weight of the delta U loss, and u_k and u_{k-1} are the control input vectors at time steps k and $k-1$, respectively. By penalizing the difference between successive control inputs, this loss function ensures that the control actions change smoothly over time, preventing wear and tear on the system and providing a more comfortable environment.

State Lower Bound Penalty

The system must also adhere to thermal comfort constraints to ensure the temperature remains within a comfortable range. The state lower bound penalty ensures that the temperature does not fall below a specified minimum value ($y_{\min}$). This is mathematically represented as:

$$\text{State Lower Bound Penalty} = \lambda_{y_{\min}} \sum_{k=0}^{N} \max(0, y_{\min} - y_k)^2$$

where $\lambda_{y_{\min}}$ is a scaling factor for the lower bound penalty, y_k is the temperature (or output) at time step k, and $y_{\min}$ is the minimum allowable temperature. This penalty imposes a cost whenever the temperature y_k is below $y_{\min}$.

State Upper Bound Penalty

Similarly, the state upper bound penalty ensures that the temperature does not exceed a specified maximum value ($y_{\max}$). It is mathematically represented as:

$$\text{State Upper Bound Penalty} = \lambda_{y_{\max}} \sum_{k=0}^{N} \max(0, y_k - y_{\max})^2$$

where $\lambda_{y_{\max}}$ is a scaling factor for the upper bound penalty, y_k is the temperature at time step k, and $y_{\max}$ is the maximum allowable temperature. This penalty imposes a cost whenever the temperature y_k exceeds $y_{\max}$.

Overall Penalty Loss

The overall penalty loss function that combines the action loss, delta U loss, and the state constraints is formulated as:

$$\text{Penalty Loss} = \text{Action Loss} + \text{Delta U Loss} + \text{State Lower Bound Penalty} + \text{State Upper Bound Penalty}$$

Substituting the individual components, we get:

$$\text{Penalty Loss} = \lambda_a \sum_{k=0}^{N-1} \|u_k\|^2 + \lambda_{\text{du}} \sum_{k=1}^{N-1} \|u_k - u_{k-1}\|^2 + \lambda_{y_{\min}} \sum_{k=0}^{N} \max(0, y_{\min} - y_k)^2 + \lambda_{y_{\max}} \sum_{k=0}^{N} \max(0, y_k - y_{\max})^2$$

This combined loss function is minimized during training to adjust the parameters of the LSTM-based control policy. The scaling factors λ_a, λ_{du}, $\lambda_{y_{\min}}$, and $\lambda_{y_{\max}}$ are hyperparameters that need to be tuned to balance the different objectives and constraints.

By minimizing this overall penalty loss, the training process ensures that the control policy effectively maintains the desired temperature range, minimizes energy usage, and operates smoothly under various conditions. This approach leverages the strengths of both traditional control theory and modern machine learning techniques to create a robust and efficient temperature control system.

Optimization Process

The optimization process involves updating the neural network parameters to minimize the overall penalty loss. This is typically done using gradient-based optimization methods such as AdamW. The optimization algorithm iteratively adjusts the network parameters to find the set that results in the lowest penalty loss, effectively training the LSTM-based control policy to maintain the desired temperature range, minimize energy usage, and ensure smooth control actions under various conditions.

Conclusions

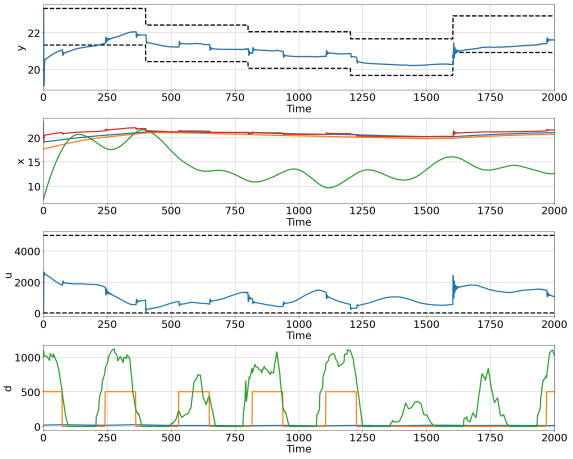

Figure 6: LSTM plots

First Subplot (Temperature y):

The first subplot illustrates the actual indoor temperature y (blue line) over time, compared to the reference temperature bounds (black dashed lines). These bounds represent the desired temperature range that the control system aims to maintain, set between 18°C and 22°C in this case. The temperature trajectory indicates how well the control system adheres to the specified comfort range. When the temperature deviates towards the upper or lower bound, the controller adjusts the inputs to bring the temperature back within the range. This subplot shows the effectiveness of the control policy in maintaining thermal comfort by minimizing deviations from the desired range.

Second Subplot (States x):

The second subplot presents the state variables x of the system. These states represent internal dynamics of the building model that influence the temperature control. The plot shows multiple state trajectories:

- The green line indicates a specific state variable that might be related to thermal inertia or heat transfer within the building.
- The red and blue lines represent other state variables that might include factors like wall temperatures, HVAC system status, or air mass temperatures.

The dynamics of these state variables are crucial for understanding how the internal conditions of the building evolve over time in response to control inputs and disturbances. This subplot helps in analyzing the internal behavior of the system and its responsiveness to control actions.

Third Subplot (Control Input u):

The third subplot displays the control inputs u (blue line), which are the actions taken by the LSTM policy to regulate the temperature. The black dashed lines indicate the upper and lower bounds of these inputs, constrained by the system's physical limitations. These inputs could represent commands to the HVAC system, such as heating, cooling, or airflow adjustments. The plot shows

how the control policy varies these inputs over time to maintain the indoor temperature within the desired range. The controller avoids aggressive changes in control actions to ensure smooth operation and energy efficiency, as indicated by the relatively stable control input trajectory.

Fourth Subplot (Disturbances d):

The fourth subplot illustrates the disturbances d affecting the system (green line), along with the disturbance model's predictions (orange line). Disturbances include external factors such as outdoor temperature variations, solar radiation, or occupancy changes that influence the indoor temperature. The plot demonstrates how these disturbances fluctuate over time and how the disturbance model predicts and compensates for these changes. Effective disturbance modeling and compensation are crucial for maintaining thermal comfort, as they enable the control system to anticipate and react to external influences.

Overall System Behavior:

The overall performance of the LSTM-based temperature controller is depicted across these subplots, showcasing the interaction between control inputs, state variables, disturbances, and the resulting temperature trajectory. The controller effectively maintains the indoor temperature within the desired range by dynamically adjusting the control inputs in response to disturbances and internal state changes. The system's ability to keep the temperature within the comfort bounds demonstrates the robustness and effectiveness of the LSTM-based control policy.

Comparative Analysis of Temperature Controllers

Effectiveness:

- The MLP-based controller and LSTM-based controller both effectively maintain the temperature within the desired bounds. The MLP-based controller tends to have the temperature approach the upper bound more frequently, whereas the LSTM controller maintains it more centrally within the bounds.
- The PID controller struggles to maintain the temperature within the desired bounds consistently. The temperature yy frequently drops below the lower bound, indicating inadequate control during certain disturbance conditions.

Stability:

- All controllers maintain stable system dynamics, as indicated by the smooth trajectories of the state variables. However, the LSTM controller shows the smoothest trajectories, suggesting superior stability.
- The MLP-based controller also shows stable behavior but with some noticeable variability in control inputs.

Control Effort:

- The control inputs for the LSTM-based controller are well-regulated, with fewer abrupt changes, indicating efficient operation.
- The MLP-based controller shows more variability in control inputs, with occasional spikes corresponding to significant temperature changes.
- The PID controller exhibits minimal control effort but insufficient to maintain desired temperature.

Disturbance Rejection:

- All controllers effectively handle disturbances, maintaining performance despite significant external variations.
- The LSTM-based controller and MLP-based controller both show robust disturbance rejection, with the LSTM controller demonstrating a slightly smoother response to disturbances.
- The PID controller manages poor disturbance rejection, failing to maintain temperature during significant disturbances.

Summary

The LSTM-based neural network controller shows the best overall performance, maintaining temperature effectively within bounds with the least variability and smoothest control inputs. The MLP-based controller also performs well but with slightly more variability in temperature and control inputs. The PID controller, while somewhat effective (Temp = 17.5), exhibits more fluctuations and aggressive control behavior, indicating room for improvement in maintaining smooth and consistent temperature control. Overall, neural network-based controllers, particularly the LSTM, offer superior performance in this application.

References

[1] *DPC building control*. URL: https://colab.research.google.com/github/pnnl/neuromancer/blob/master/examples/domain_examples/DPC_building_control.ipynb.

[2] *neuromancer*. Accessed: 2024-05-30. URL: https://github.com/pnnl/neuromancer.