Project Report: Model Predictive Control (MPC) for Process Systems

Author: Sohel Rana

Position: Digital Currency Investor & Technical Lead

Nationality: Bangladeshi

1. Introduction

Model Predictive Control (MPC) is an advanced control methodology widely used in industrial automation and process systems. Unlike traditional controllers, MPC predicts future system behavior using a mathematical model and optimizes control inputs accordingly. This project demonstrates the implementation of MPC for a first-order plus dead-time (FOPDT) process system using Octave, without relying on external packages. The goal is to showcase MPC's ability to track setpoints, handle disturbances, and respect input constraints.

2. Objectives

- Implement MPC for a synthetic process system.
- Simulate system response under disturbances and constraints.
- Evaluate MPC performance in terms of setpoint tracking, disturbance rejection, and input optimization.

3. Process Model

The process is modeled as a FOPDT system, represented by the difference equation:

$$y(k) = a * y(k-1) + b * u(k-L)$$

Where:

- K = 2 -> Process gain
- T = 10 -> Time constant
- L = 3 -> Dead time (samples)
- Ts = 1 -> Sampling time

This model captures the essential dynamics of many chemical and industrial processes.

4. MPC Design

Parameters:

- Prediction Horizon: 20 steps
- Control Horizon: 5 steps
- Weights:
  - Q = 1 (output error penalty)
  - lambda = 0.1 (input move penalty)
- Constraints: Input bounded between 0 and 100

Strategy:

- Predict future outputs over the prediction horizon.
- Evaluate candidate inputs using the cost function:

  J = sum(Q * (r - y_pred)^2) + lambda * (delta_u)^2

- Select the input that minimizes the cost.

5. Implementation

The MPC algorithm was implemented in Octave using arrays and loops.

Steps:

1. Initialization: Output, input, setpoint, disturbance, and dead-time buffer.
2. Simulation Loop:
   - Update process output using the difference equation.
   - Test candidate inputs across the allowed range.
   - Predict future outputs for each candidate.
   - Compute cost function and select optimal input.

3. Disturbance Handling: A disturbance of +10 was introduced between steps 40?60 to test robustness.

6. Results

- Output Response: The system tracked the setpoint of 50 effectively, with minor deviations during disturbance.
- Control Input: MPC adjusted inputs smoothly, respecting constraints and avoiding aggressive changes.
- Disturbance Rejection: The controller compensated for the disturbance, restoring the output to the desired setpoint.

Plots generated:

- Output vs Setpoint (tracking performance).
- Control Input vs Time (controller actions).

7. Conclusion

This project demonstrates the effectiveness of MPC in controlling a process system with dead-time and disturbances. Even with a simplified greedy search approach, MPC achieved reliable setpoint tracking and disturbance rejection. The implementation highlights MPC's ability to balance performance and smoothness while respecting input constraints.

8. Future Work

- Extend the implementation to multi-input multi-output (MIMO) systems.
- Incorporate quadratic programming solvers for more efficient optimization.
- Apply MPC to real-world industrial datasets for validation.