



Virtual Commissioning for Advanced Control

Brad Schofield

Motivation

Case Study: Air Handling Units (AHU)

Model Predictive Control (MPC)

Lab Setup for Virtual Commissioning

Demo

Motivation

Motivation

Demands on controller performance and efficiency, together with increasingly complex processes to be controlled, mean that traditional control design methods are often no longer sufficient. There exist more advanced control design schemes which may be used to meet these challenges, but they typically require extensive validation and testing before deployment.

Advanced Control

Advanced control may refer to many things, such as:

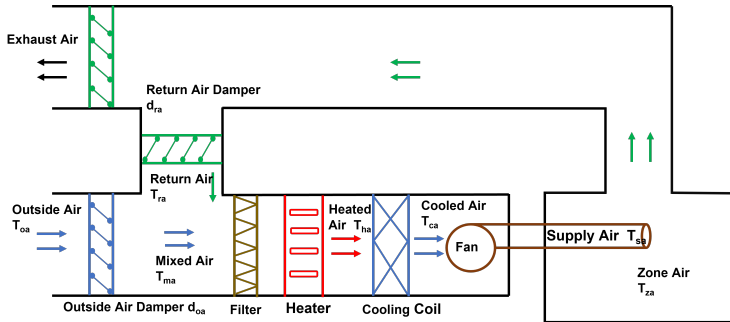
- Model Predictive Control (MPC)
- Neural Networks
- Model-based estimation and control

Challenges for design and deployment of advanced controllers:

- Implementation (code) generally more complex
- Often many more parameters to be identified, estimated or tuned

Case Study: Air Handling Units (AHU)

AHU Control Problem



Controlled Variables

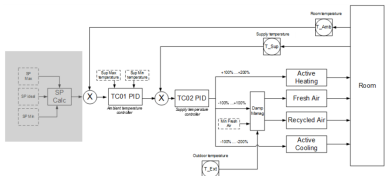
- Zone air temperature (range 15-24 degrees)
- Supply air temperature (15-30 degrees)

Manipulated Variables

- Outside and return air dampers
- Fan speed
- Heating and cooling coils

AHU Control Problem

A classical controls approach would be to use cascaded PID controllers:



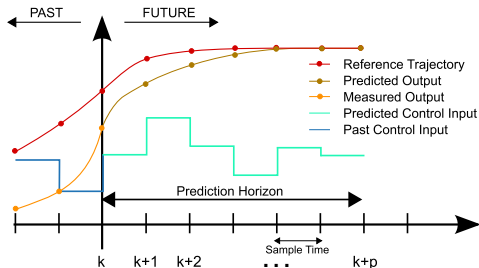
We are forced to choose specific setpoints (which we can try to vary to minimize energy consumption). However, this is not easy to do without introducing dynamic interactions.

We would like to have a controller that:

- Can express the control objectives in terms of ranges and constraints
- Can explicitly take into account energy consumption
- Can easily take into account present and even future process disturbances (for example, via a weather forecast)

Model Predictive Control (MPC)

What is MPC?



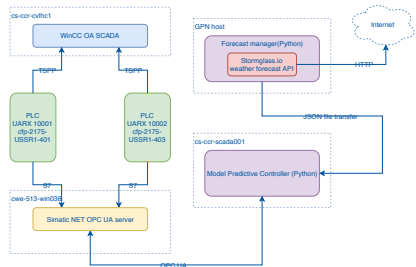
- Want to find a set of control actions that minimizes some cost over a fixed horizon, while respecting some constraints
- Need to have some kind of dynamic plant model to predict future behavior
- Leads to an optimization problem which must be solved
- **Apply the first of the sequence of control actions, then repeat the process!**

Formulation of the MPC problem

$$\begin{aligned}
 & \min_{\mathbf{d}, \boldsymbol{\omega}, \boldsymbol{\epsilon}, \mathbf{T}} \sum_{k=0}^N P_f(\omega_k) + P_h(T_{k+1}^{ha}) + P_c(T_{k+1}^{ca}) + r_{sp}(T_k^{za} - T_{sp}^{za})^2 + \mathbf{r}^T \boldsymbol{\epsilon}^2 \\
 \text{s.t. } & T_{k+1}^{za} = f_{za}(T_{k,k}^{za}, T_k^{sa}, T_k^{oa}, \omega_k), T_{k+1}^{ma} = f_{ma}(T_k^{za}, T_k^{oa}, d_k^{oa}), T_{k+1}^{sa} = f_{fan}(T_k^{ca}) \\
 & T_{k+1}^{ca} - \epsilon_{ha} \leq T_{k+1}^{ha} \leq T_{ha}^{max} + \epsilon_{ha}, T_{ca}^{min} - \epsilon_{ca} \leq T_{k+1}^{ca} \leq T_{k+1}^{ha} + \epsilon_{ca} \\
 & T_{sa}^{min} - \epsilon_{sa} \leq T_{k+1}^{sa} \leq T_{sa}^{max} + \epsilon_{sa}, T_{za}^{min} - \epsilon_{za} \leq T_{k+1}^{za} \leq T_{za}^{max} + \epsilon_{za} \\
 & d_{oa}^{min} \leq d_k^{oa} \leq d_{oa}^{max}, \omega_k^{min} \leq \omega_k \leq \omega_k^{max} \\
 & -\dot{d}_{oa} \leq d_{k+1}^{oa} - d_k^{oa} \leq \dot{d}_{oa}, -\dot{\omega} \leq \omega_{k+1} - \omega_k \leq \dot{\omega} \\
 & -\dot{T}^{ca} \leq T_{k+1}^{ca} - T_k^{ca} \leq \dot{T}^{ca}, -\dot{T}^{ha} \leq T_{k+1}^{ha} - T_k^{ha} \leq \dot{T}^{ha} \\
 & T_{za}[0] = T_{za}^{init}, T_{ma}[0] = T_{ma}^{init}, T_{ha}[0] = T_{ha}^{init}, \\
 & T_{ca}[0] = T_{ca}^{init}, \epsilon_{sa} > 0, \epsilon_{za} > 0, \epsilon_{ca} > 0, \epsilon_{ha} > 0, \mathbf{r} > 0
 \end{aligned}$$

Implementation

The process models are simple and continuous, but nonlinear. The MPC optimization problem is a Nonlinear Program (NLP). Not realistic to implement solver on classic PLC. Instead, implement in Python and run on remote server. Solve with CasADI and IPOPT



Communication between PLC and MPC is through OPC UA. We keep cascade PID, and add tracking mode for bumpless switch to/from MPC

Implementation: Closer Look

On the PLC:

- Keep existing controls, but allow switching to MPC
- Implement communication with external MPC application (OPC UA)
- Implement watchdog to check status of MPC application
- Implement tracking value calculation to allow bumpless mode change between MPC and PID

In the MPC application:

- Implement the MPC problem including models and constraints
- Collect process data from PLC via OPC UA
- Collect weather forecast data for prediction from external API
- Carry out parameter estimation for model parameters
- Solve the optimization problem and monitor status of solution

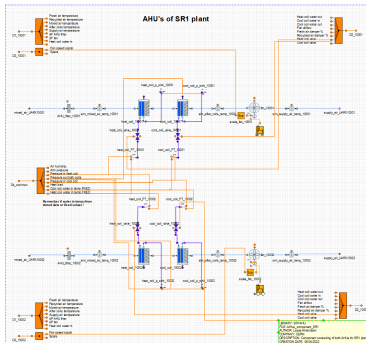
Requirements for Testing and Commissioning

- Need to be able to test the correct operation of the advanced control scheme. Normally needs a good system model.
- If the architecture has changed, need to validate the new one (particularly communication)
- Need to validate all logic changes at the PLC level
- Need to validate changes to operator interface

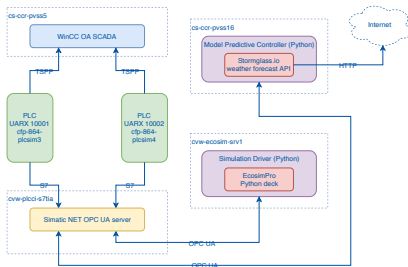
Lab Setup for Virtual Commissioning

Virtual Commissioning

Need a more realistic model to reliably develop and test the new controller. Created in EcosimPro:



Recreate the production setup in the lab:



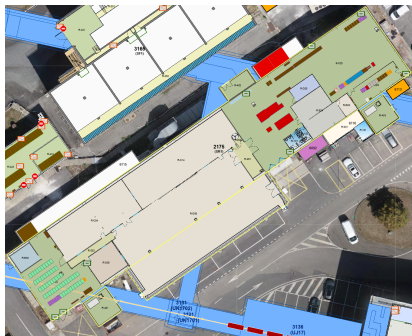
Lab setup includes PLCs, SCADA and Python simulation driver application.

Virtual Commissioning Steps

- Initially, the MPC controller code was validated against the EcosimPro model directly, without the PLC in the loop
- The PLC was then added in the loop, to develop and test the logic updates to interface the MPC controller and test communication
- Then, the MPC controller was connected to process data from the production PLC and run in open loop (not sending commands to PLC)
- Finally, 'live' data from the process was used to drive the model, to validate the MPC controller in closed loop for current operating conditions

Operational Deployment

HVAC for SR1 (Point 1)



- One of two AHUs in the building controlled by MPC
- The other retains existing controls
- Deployed mid-September 2023
- Practically no commissioning: less than 2 hours from intervention start to MPC running

Digital Twins

- Even after deployment, the virtual commissioning setup continues to be used as a 'Digital Twin', with the model being fed by certain process data, and with the controller running in closed loop against the model
- Any modifications to the MPC application are validated in this way before deploying
- No need to disturb the process to test updates; MPC has been operational almost 100% since deployment
- Using `git` for source control; separate branches for lab and production

Demo



home.cern