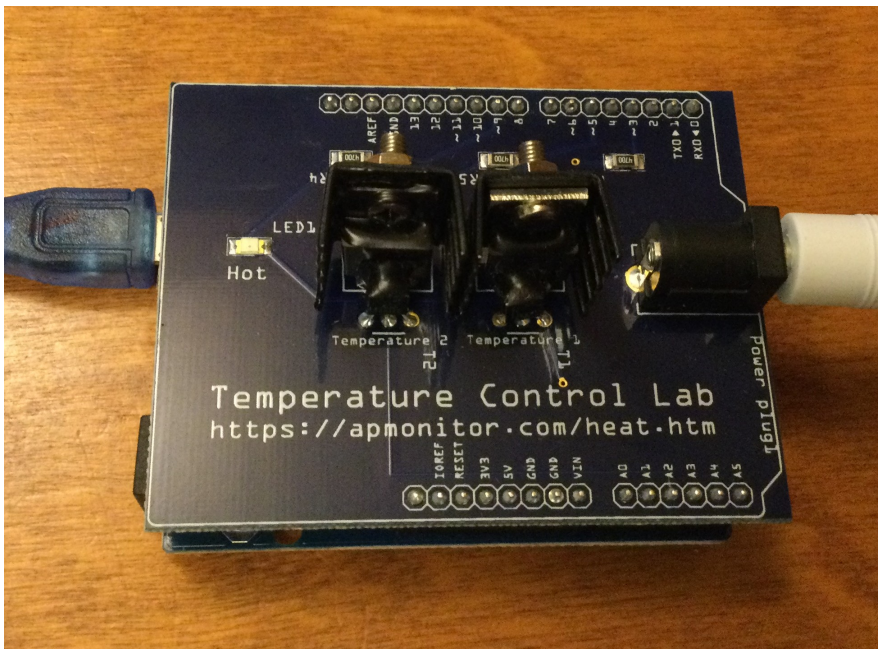
Dynamic Matrix Control of the Temperature Control Lab Device

Kent H. Rasmussen

January 2021 - Version 2.0

Abstract

This document provides an introduction to the theory of Dynamic Matrix Control (DMC). The control theory is illustrated using the *Temperature Control Lab* device that was developed for educational purposes.



1 Introduction

The Temperature Control Lab (*TCLab*) device has two temperature sensors and two electric heaters. The device is relatively inexpensive ($35 on Amazon). Hedengren and Kantor [1] describe how the device has been used in Process Control Education for Chemical Engineers at BYU and other Universities since 2013. The device can be controlled from popular software such as Matlab, Octave and Python.

The intent of this document is to illustrate the theory of Dynamic Matrix Control as described in the papers by Cutler and Ramamker [2] and Cutler, Morshedi and Haydel [3]. The control problem will be formulated using impulse response models which are closely related to step response models. The focus is on ensuring the math is easy to follow rather than on a computational efficient implementation.

All examples in this document are developed in Python [8]. The Python scripts can communicate with the *TCLab* device through a USB serial connection. Python has libraries for Advanced Matrix Calculations, Optimization, Visualization of Data, as well as libraries for MODBUS and OPC connectivity. Python is also very popular in the Data Science community.

2 Step Testing

A Dynamic Model of the the Process is required for Control. To estimate a Dynamic Model from Process data it is typically required to perform some kind of step testing to ensure the data has sufficient information content. During the step testing the heat input to each of the two heaters are varied independently. The data from a step test is shown in figure 1.

- The temperatures T_1 and T_2 respond well to changes in heat input Q_1 and Q_2.
- The data shows Heater-1 has more power than Heater-2.
- Heater-1 affect both T_1 and T_2, and that the effect on T_1 is stronger as expected.
- Heater-2 affect both T_1 and T_2, but the effect on T_1 is relatively small.
- Heater-1 strongly affects T_2, almost as much as Heater-2.

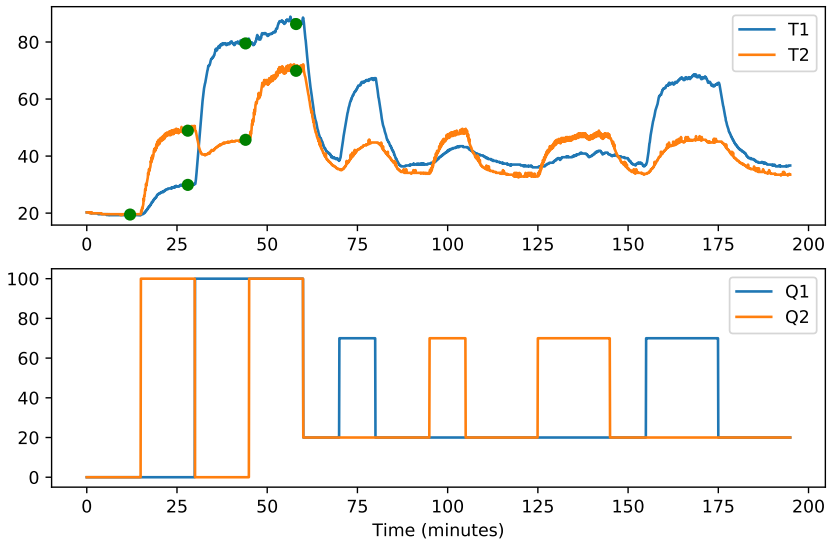


Figure 1: The upper part of the plot shows the response from the two temperature sensors. The lower part of the plot shows the heat input to the two heaters.

3 Model development

In this section it will be shown how to devvelop models for the *TCLab* device. First a steady model will be developed from the step test data, then a dynamic model suitable for control will be estimated.

3.1 Steady State Model

During the initial part of the step test each heater was turned on to 100 percent one at a time and the temperatures were allowed to settle. The table below shows the temperature data at select times after the temperatures have stabilized. These data points are marked with green dots in figure 1.

Time	T_1	T_2	Q_1	Q_2
12	19.29	19.62	0.0	0.0
25	29.93	48.95	0.0	100.0
44	79.56	45.72	100.0	0.0
58	86.33	69.90	100.0	100.0

From the data in the table, an estimate of the steady state gain matrix relating changes in heat input to changes in temperature, can be derived. The gain matrix shows a one percent change in Q_1 will cause a 0.593 °C increase in T_1.

$$\begin{bmatrix} \Delta T_1 \\ \Delta T_2 \end{bmatrix} = \frac{1}{100} \begin{bmatrix} 79.56 - 19.29 & 29.93 - 19.29 \\ 45.72 - 19.62 & 48.95 - 19.62 \end{bmatrix} \begin{bmatrix} \Delta Q_1 \\ \Delta Q_2 \end{bmatrix} = \begin{bmatrix} 0.5930 & 0.0967 \\ 0.2546 & 0.2869 \end{bmatrix} \begin{bmatrix} \Delta Q_1 \\ \Delta Q_2 \end{bmatrix}$$

The gain matrix can be used to estimate the temperatures when both heaters are fully turned on. The estimated temperatures are slightly higher than the observed temperatures listed in the last row of the data table. The temperature response may not be exactly linear as the heat loss increases when the temperatures increase.

$$\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} 19.29 \\ 19.62 \end{bmatrix} + \begin{bmatrix} 0.5930 & 0.0967 \\ 0.2546 & 0.2869 \end{bmatrix} \begin{bmatrix} 100 \\ 100 \end{bmatrix} = \begin{bmatrix} 89.23 \\ 74.41 \end{bmatrix} \approx \begin{bmatrix} 86.33 \\ 69.90 \end{bmatrix}$$

3.2 Dynamic Model

The model type that will be used to represent dynamics is the Finite Impulse Response Model (FIR). The FIR model describes changes in a dependent variables (temperature) at the current time as a function of changes in the independent variables (heat inputs) at past times.

The change in temperature for the two temperature sensors at the time t depends on the current and past heat input levels.

$$\begin{aligned} y_1(t) &= h_{11}^0 u_1(t) + h_{11}^1 u_1(t-1) + \cdots + h_{11}^N u_1(t-N) \\ &+ h_{12}^0 u_2(t) + h_{12}^1 u_2(t-1) + \cdots + h_{12}^N u_2(t-N) \\ y_2(t) &= h_{21}^0 u_1(t) + h_{21}^1 u_1(t-1) + \cdots + h_{21}^N u_1(t-N) \\ &+ h_{22}^0 u_2(t) + h_{22}^1 u_2(t-1) + \cdots + h_{22}^N u_2(t-N) \end{aligned}$$

The time $t-1$ indicates the time one sample period prior to time t. And the variables y_i and u_i indicate changes from some initial point where $y_1(t) = \Delta T_1(t) = T_1(t) - T_1(0)$, $y_2(t) = \Delta T_2(t) = T_2(t) - T_2(0)$, $u_1(t) = \Delta Q_1(t) = Q_1(t) - Q_1(0)$ and $u_2(t) = \Delta Q_2(t) = Q_2(t) - Q_2(0)$.

The FIR model describes changes in temperatures based upon changes in heat inputs from a starting point at time $t = 0$. The h_{ij}^k coefficients are the impulse response coefficients that represent the dynamic response. In the rest of this document the static part at time $t = 0$ will be omitted, and so will the Δ symbol indicating change. All models used will be based upon changes from some starting point.

FIR example
A very simple impulse response model written in vector format is:

$$y(t) = \begin{bmatrix} 0 & 0.7 & 0.2 & 0.1 & 0.0 \end{bmatrix} \begin{bmatrix} u(t) & u(t-1) & u(t-2) & u(t-3) & u(t-4) \end{bmatrix}^T$$

Assume the process is steady ($y(t) = u(t) = 0$ for $t < 0$) and that there is a step change in the input at time zero ($u(t) = 1$ for $t \geq 0$), then the output at different time intervals will be:

$$\begin{aligned}
y(0) &= \begin{bmatrix} 0 & 0.7 & 0.2 & 0.1 & 0.0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T = 0 \\
y(1) &= \begin{bmatrix} 0 & 0.7 & 0.2 & 0.1 & 0.0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \end{bmatrix}^T = 0 + 0.7 = 0.7 \\
y(2) &= \begin{bmatrix} 0 & 0.7 & 0.2 & 0.1 & 0.0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \end{bmatrix}^T = 0 + 0.7 + 0.2 = 0.9 \\
y(3) &= \begin{bmatrix} 0 & 0.7 & 0.2 & 0.1 & 0.0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \end{bmatrix}^T = 0 + 0.7 + 0.2 + 0.1 = 1.0 \\
y(4) &= \begin{bmatrix} 0 & 0.7 & 0.2 & 0.1 & 0.0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T = 0 + 0.7 + 0.2 + 0.1 + 0 = 1.0 \\
&\vdots \\
y(k) &= \begin{bmatrix} 0 & 0.7 & 0.2 & 0.1 & 0.0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T = 1.0
\end{aligned}$$

If there are no further changes in the input, the output from the model will be constant into the future ($y(k) = 1$ for $k \geq 4$) and we call the observed output sequence the step response. Assume the impulse response model represents changes in temperature for changes in heat input and that the process is steady with a heat input of 50 and a temperature of 30. When the heat input is changed to 51 at time $t = 0$ the fist 5 steps of the dynamic temperaure response will be:

$$\begin{bmatrix} T(0) & T(1) & T(2) & T(3) & T(4) \end{bmatrix} = \begin{bmatrix} 30 & 30.7 & 30.9 & 31 & 31 \end{bmatrix}$$

3.3 Model Identifcation

The impulse response coefficients can be estimated from the data collected from the step test. The step test data set is of length $M = 2340$ (195 minutes at 5 second sample interval). The first data point in the step test data gets the time stamp $t = 0$ and the last data point gets the time stamp $t = 2339$. The time stamps reflect the sample interval number and not the actual time in seconds.

We want to estimate a FIR model with $N + 1$ coefficients from each of the heaters to the observed temperatures. Based upon the (unknown) FIR model, the predicted temperature

at time k ($k \geq N$) for each temperature sensor can be written as:

$$\begin{aligned} y_1(k) &= h_{11}^0 u_1(k) + h_{11}^1 u_1(k-1) + \cdots + h_{11}^N u_1(k-N) \\ &+ h_{12}^0 u_2(k) + h_{12}^1 u_2(k-1) + \cdots + h_{12}^N u_2(k-N) \\ y_2(k) &= h_{21}^0 u_1(k) + h_{21}^1 u_1(k-1) + \cdots + h_{21}^N u_1(k-N) \\ &+ h_{22}^0 u_2(k) + h_{22}^1 u_2(k-1) + \cdots + h_{22}^N u_2(k-N) \end{aligned}$$

Note that the equation cannot be used to describe $y(k)$ for $k < N$ unless we make the assumption that $u_1(k) = u_1(0)$ for $k < 0$ - i.e. the process is steady before the start of the plant test. This equation can be repeated for each data point until the last data point. This allows for writing the predicted time series for each temperature in a compact format as:

$$\begin{aligned} Y_1 &= U_1 H_{11} + U_2 H_{12} = U H_1 \\ Y_2 &= U_1 H_{21} + U_2 H_{22} = U H_2 \end{aligned}$$

where

$$Y_1 = \begin{bmatrix} y_1(N) \\ \vdots \\ y_1(M-1) \end{bmatrix} \quad Y_2 = \begin{bmatrix} y_2(N) \\ \vdots \\ y_2(M-1) \end{bmatrix}$$

$$U_1 = \begin{bmatrix} u_1(N) & \cdots & u_1(0) \\ \vdots & & \vdots \\ u_1(M-1) & \cdots & u_1(M-1-N) \end{bmatrix}$$

$$U_2 = \begin{bmatrix} u_2(N) & \cdots & u_2(0) \\ \vdots & & \vdots \\ u_2(M-1) & \cdots & u_2(M-1-N) \end{bmatrix}$$

$$\begin{aligned} U &= \begin{bmatrix} U_1 & U_2 \end{bmatrix} \\ H_1 &= \begin{bmatrix} H_{11} & H_{12} \end{bmatrix} \\ H_2 &= \begin{bmatrix} H_{21} & H_{22} \end{bmatrix} \\ H_{ij} &= \begin{bmatrix} h_{ij}^0 & h_{ij}^1 & \cdots & h_{ij}^N \end{bmatrix}^T \end{aligned}$$

The goal of the model parameter estimation is to minimze the difference between the observed temperature $y_i^*(k)$ and the predicted temperature $y_i(k)$ at time k. The observed temperatures can be aranged in vector format similar to the predicted temperatures.

$$Y_1^* = \begin{bmatrix} y_1^*(N) \\ \vdots \\ y_1^*(M-1) \end{bmatrix} \quad Y_2^* = \begin{bmatrix} y_2^*(N) \\ \vdots \\ y_2^*(M-1) \end{bmatrix}$$

The parameter estimation can be formulated as an optimization problem where the sum of the squared error (a scalar) is mimized by adjusting the impulse response coefficients for each temperature sensor. A small error represents close agreement between the temperature response and the predicted temperature response. The sum of the squared error can be formulated as a vector product which can be expressed in short format as a vector norm (squared).

$$\begin{aligned} J_1 &= \sum_{k=N}^{M-1} (y_1^*(k) - y_1(k))^2 = (Y_1^* - UH_1)^T (Y_1^* - UH_1) = \left\| Y_1 - UH_1 \right\|_2^2 \\ J_2 &= \sum_{k=N}^{M-1} (y_2^*(k) - y_2(k))^2 = (Y_2^* - UH_2)^T (Y_2^* - UH_2) = \left\| Y_2 - UH_2 \right\|_2^2 \end{aligned}$$

The solution that minimizes the objective functions is the normal Least Squares regression for an overdetermined problem ($M > N$). The estimate of the impulse response coefficients is:

$$\begin{aligned} H_1 &= [U^T U]^{-1} U^T Y1 \\ H_2 &= [U^T U]^{-1} U^T Y2 \end{aligned}$$

The same matrix $U^T U$ has to be inverted to determine the coefficients for each of the temperatures. This matrix will be singular (non invertable) if there is no movement in one of the inputs, or if the inputs are correlated.

Linear Regression example

The estimation of the impulse response coefficients is very similar to how the slope of a line going through the origin is estimated. If there is only one data point $(x_1, y_1) = (1, 2)$, then the best fit for the slope is clearly $\alpha = 2$. If we have a second data point $(x_2, y_2) = (2, 5)$ the best estimate can be found using linear regression. The data can be aranged in vectors:

$$U^T = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T = \begin{bmatrix} 1 & 2 \end{bmatrix}^T \qquad Y^T = \begin{bmatrix} y_1 & y_2 \end{bmatrix}^T = \begin{bmatrix} 2 & 5 \end{bmatrix}^T$$

When only one parameter is estimated, the matrices $U^T U$ and $U^T Y$ both become scalars:

$$U^T U = \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix}^T = 5 \qquad U^T Y = \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 2 & 5 \end{bmatrix}^T = 12$$

The estimate of the slope then becomes:

$$\alpha = [U^T U]^{-1} U^T Y = \frac{1}{5} \times 12 = 2\frac{2}{5}$$

3.3.1 Detrending

The data is filtered before the identification to make it have zero mean and remove drift. This is necessary because the FIR model used can only represent dynamics and not the steady state information. For FIR model estimation it is common to use differenced data (a high pass filter):

$$\begin{aligned} \Delta u_i(t) &= u_i(t) - u_i(t-1) \\ \Delta y_i(t) &= y_i(t) - y_i(t-1) \end{aligned}$$

The difference filter does emphasize the high frequency content in the data. Sometimes additional filtering such as central average filtering is used to emphasize the lower frequency information in the data to get a better estimate of the gains.

3.3.2 Smoothening

The estimated impulse response model can be erractic. Dayal and MacGregor [4] describe various techniques to smooth the estimated impulse response.

- Penalize the difference between two adjecent impulse response coefficients.
- Use additional penality on the later coefficients assuming the response is becomming steady as time goes on.

The difference between adjacent coefficients can be written as a vector.

$$\Delta H_{ij} = \left[\begin{array}{cccc} h_{ij}^0 - 0 & h_{ij}^1 - h_{ij}^0 & \cdots & h_{ij}^N - h_{ij}^{N-1} \end{array} \right]$$

Minimizing the vector product $\Delta H_{ij}^T \Delta H_{ij}$ (sum of squared differences) as part of the estimation problem will penalize movement between the impulse response coefficients. Weights can be used to add additional penality on movement between impulse response coefficents that represent past data. This has the effect of forcing the model to reach steady state. A simple method is to have the weights increase linearly.

$$\left[\begin{array}{cccc} 1 \times (h_{ij}^0 - 0) & 2 \times (h_{ij}^1 - h_{ij}^0) & \cdots & (N+1) \times (h_{ij}^N - h_{ij}^{N-1}) \end{array} \right]$$

The weighted sum of the squared differences can be written in matrix form as:

$$[\Delta H_{ij}]^T L \Delta H_{ij} = [H_{ij} A]^T L H_{ij} A = H_{ij}^T \left[A^T L A \right] H_{ij}$$

The weight matrix L is a diagonal matrix with increasing elements and A is a matrix that is used to calculate the differences between the impulse response coefficeints:

$$
\begin{aligned}
A &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & & & & & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \\
L &= \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 2 & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & N+1 \end{bmatrix}
\end{aligned}
$$

Instead of simply minimizing the sum of the squared errors between the predicted temperatures and the actual observed temperatures the objective function will be modified to include an additional term that penalizes differences between the impulse response coefficients.

$$
\begin{aligned}
J_1^* &= \sum_{k=N}^{M-1} (y_1^*(k) - y_1(k))^2 = (Y_1^* - UH_1)^T (Y_1^* - UH_1) + \kappa H_1^T W H_1 \\
J_2^* &= \sum_{k=N}^{M-1} (y_2^*(k) - y_2(k))^2 = (Y_2^* - UH_2)^T (Y_2^* - UH_2) + \kappa H_2^T W H_2
\end{aligned}
$$

where κ is a factor that determines the amount of smoothening. If all impulse response coefficients are zero, the last term in the objective function will be zero. The matrix W is a block diagonal matrix (one block for each of the inputs):

$$
W = \begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix}^T \begin{bmatrix} L & 0 \\ 0 & L \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix}
$$

The smoothened impulse response coefficients h_{ijk}^* are found as the optimal solution that minimizes the objective functions J_1^* and J_2^*.

$$
\begin{aligned}
H_1^* &= [U^T U + \kappa * W]^{-1} U^T Y1 \\
H_2^* &= [U^T U + \kappa * W]^{-1} U^T Y2
\end{aligned}
$$

Figure 2 shows the estimated impulse response coefficients with and without smoothening for the model between the heat input Q_1 to the first heater and the temperature sensor T_1.

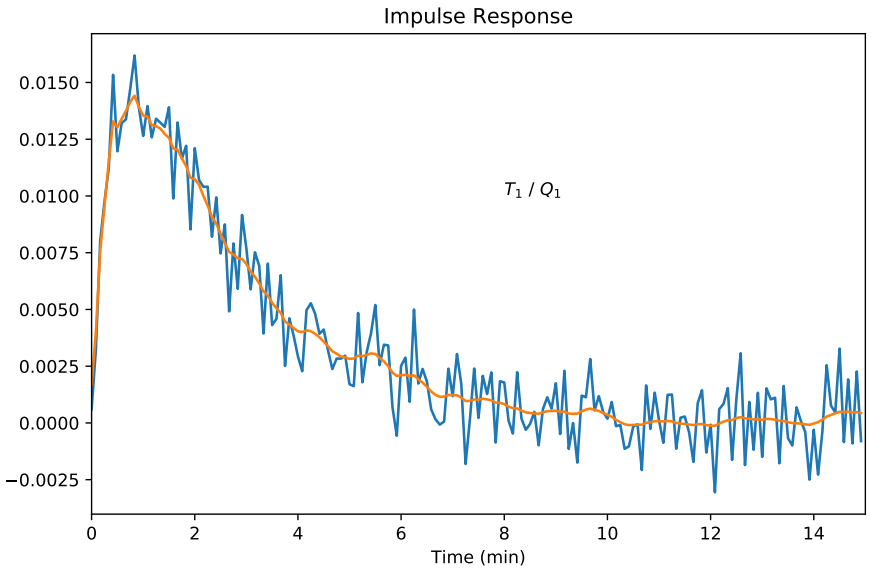


Figure 2: The plot shows 180 estimated impulse response coefficients, with smoothening (orange) and without smoothening (blue), for the relationship between temperature sensor number one T_1 and heater number one Q_1.

3.3.3 Step Response

The step response model can be derived from the impulse response model. Each of the $N+1$ step response coefficients s_{ijk} are calculated as a sum of the impulse response coefficients.

$$\begin{aligned} s_{ij}^0 &= h_{ij}^0 \\ s_{ij}^1 &= h_{ij}^0 + h_{ij}^1 = s_{ij}^0 + h_{ij}^1 \\ s_{ij}^2 &= h_{ij}^0 + h_{ij}^1 + h_{ij}^2 = s_{ij}^1 + h_{ij}^2 \\ &\vdots \\ s_{ij}^N &= \sum_{k=0}^{N} h_{ij}^k = s_{ij}^{N-1} + h_{ij}^N \end{aligned}$$

The estimated model gain is the last step response coefficient s_{ij}^N. Figure 3 shows the step response model between the two heater and the two temperature sensors. The estimated model gains are very close to what was observed during the step test when the temperatures were allowed to settle out at maximum power to each heater. The step response will settle out if the sequence of estimated impulse responses approach zero.

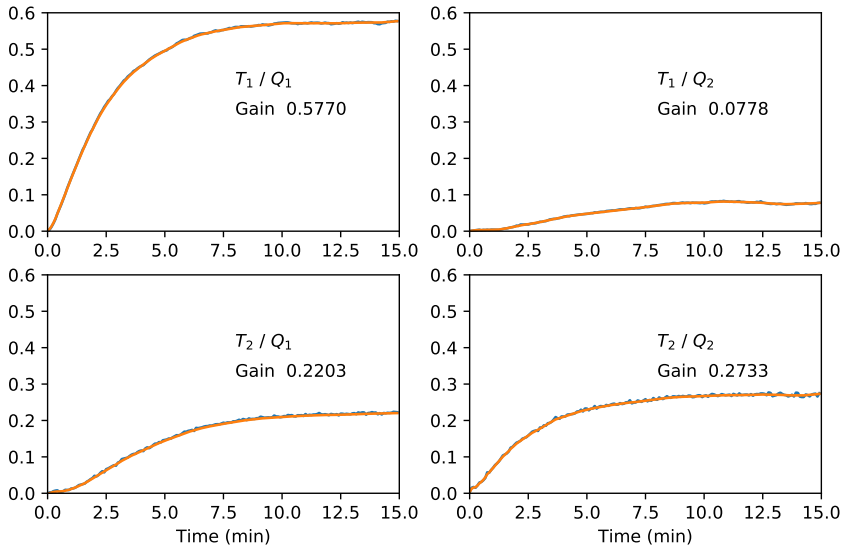


Figure 3: Step response model for the *TCLab* device. The estimate is shown with smoothening (orange) and without smoothening (blue). Note that the gains calculated from the step response model are slightly different from the steady state model obtained from the step testing.

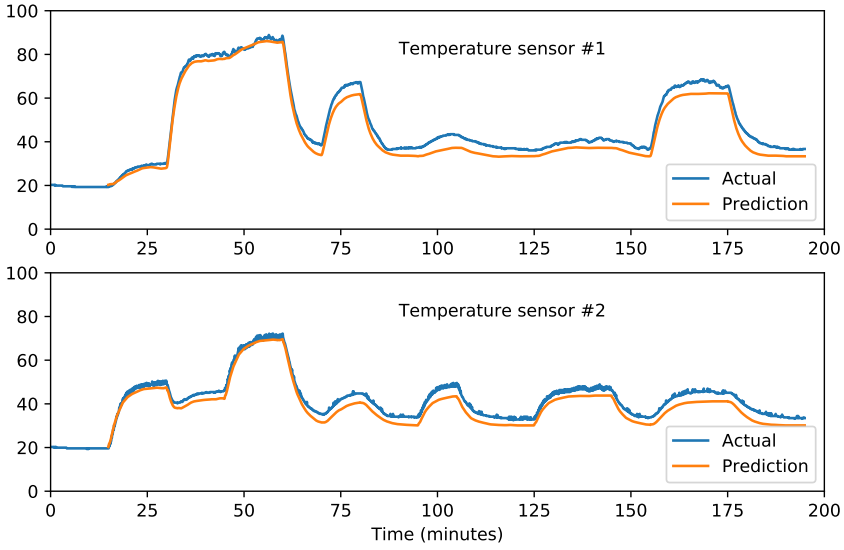


Figure 4: Prediction of the temperature data from the step test using the identified model.

3.3.4 Prediction

Once the impulse response coefficients have been identified, the temperatures can be predicted. The predicted temperatures are calculated from the changes in heat input over time. Figure 4 shows the actual temperatures and the predicted temperatures. The model is able to predict the dynamic response of the temperatures well.

$$\begin{aligned} Y_1 &= U_1 H_{11} + U_2 H_{12} = U H_1 \\ Y_2 &= U_1 H_{21} + U_2 H_{22} = U H_2 \end{aligned}$$

The matrix U is built from the heat inputs where the initial point is subtracted $u_i(t) = Q_i(t) - Q_i(0)$. Note that for the prediction, the matrix U is not built from differenced (detrended) data which was used for the model identification.

4 Control

The goal is to have the outputs (temperatures) follow a target $y_i^r(t)$ by adjusting the inputs (heaters). If both temperatures are to be controlled, then the control problem becomes a multivariable optimization problem where the squared error between both temperatures and their targets are minimized into the future by adjusting a sequence of P future inputs $U_i^P = \begin{bmatrix} u_i(t+1) & u_i(t+2) & \cdots & u_i(t+P) \end{bmatrix}^T$ for each of the two heat inputs. The input will be held constant into the future after the last move $u_i(t+P)$. The objective function to be minimized has the form shown below where the relative importance between the temperatures are weighted with the coefficients w_i

$$J_C = \sum_{k=1}^{\infty} w_1^2 (y_1^r(t+k) - y_1(t+k))^2 + w_2^2 (y_2^r(t+k) - y_2(t+k))^2$$

In this section the individual pieces needed for the Dynamic Matrix Control algorithm will be discussed.

- The first part is to predict where the outputs (temperatures) will move in the future based upon past changes in heat input.
- The second part is to calculate the correction to the heat input that brings the temperatures from the predicted future values to the desired target.

4.1 Future Prediction with Control

The number of future moves P is typically less than the number of coefficients $N+1$. At the time $t+P+N$ there will be no further change in the output as the last $N+1$ inputs to the model are all identical and equal to the last move $u_i(t+P)$. In practice, the objective function J_c is not evaluated into infinty, but over a finite control horizon of length N_c. Since the outputs are no longer changing after the time $t+P+N$ it makes sense to at least have $N_c \geq P+N$ to allow the effect of the last input change to settle. The sequence of future heat inputs needed for predicting into the future $U_i^C = \left[\begin{array}{cccc} u_i(t+1) & u_i(t+2) & \cdots & u_i(t+N_c) \end{array} \right]^T$ can be derived from the sequence of future moves.

$$U_i^C = \mathcal{M} U_i^P$$

where

$$\mathcal{M} = \left[\begin{array}{cccc} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ & \vdots & \ddots & \\ 0 & 0 & \cdots & 1 \\ \hline 0 & 0 & \cdots & 1 \\ & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{array} \right] \qquad (N_c \times P)$$

If we assume the system is steady with $y_i(t) = 0$ and $u_i(t) = 0$ at time t, then the first $N+1$ future prediction with the control moves are.

$$\begin{aligned} y_i(t+1) &= h_{i1}^0 u_1(t+1) \\ &+ h_{i2}^0 u_2(t+1) \\ y_i(t+2) &= h_{i1}^1 u_1(t+1) + h_{i1}^0 u_1(t+2) \\ &+ h_{i2}^1 u_2(t+1) + h_{i2}^0 u_2(t+2) \\ y_i(t+3) &= h_{i1}^2 u_1(t+1) + h_{i1}^1 u_1(t+2) + h_{i1}^0 u_1(t+3) \\ &+ h_{i2}^2 u_2(t+1) + h_{i2}^1 u_2(t+2) + h_{i2}^0 u_2(t+3) \\ &\vdots \\ y_i(t+N+1) &= h_{i1}^N u_1(t+1) + h_{i1}^{N-1} u_1(t+2) + \cdots + h_{i1}^0 u_1(t+N+1) \\ &+ h_{i2}^N u_2(t+1) + h_{i2}^{N-1} u_2(t+2) + \cdots + h_{i2}^0 u_2(t+N+1) \end{aligned}$$

The future predictions after the first $N+1$ follow a similar pattern, but the coefficients h_{ij}^{N+k} are zero for $k > 0$.

$$\begin{aligned} y_i(t+N+1+k) &= h_{i1}^{N+k} u_1(t+1) + h_{i1}^{N+k-1} u_1(t+2) + \cdots + h_{i1}^0 u_1(t+N+1+k) \\ &+ h_{i2}^{N+k} u_2(t+1) + h_{i2}^{N+k-1} u_2(t+2) + \cdots + h_{i2}^0 u_2(t+N+1+k) \end{aligned}$$

In summary, the future prediction for each temperature output can be calculated from the two sequences of future heat inputs.

$$Y_i^C = \mathcal{H}_{i1} U_1^C + \mathcal{H}_{i2} U_2^C = \mathcal{H}_{i1} \mathcal{M} U_1^P + \mathcal{H}_{i2} \mathcal{M} U_2^P$$

where $\mathcal{H}_{ij}$ consists of a triangular matrix constructed from the impulse response coefficients. The matix $\mathcal{H}_{ij}$ is called the Dynamic Matrix.

$$\mathcal{H}_{ij} = \begin{bmatrix} h_{ij}^0 & 0 & 0 & \cdots & 0 \\ h_{ij}^1 & h_{ij}^0 & 0 & \cdots & 0 \\ h_{ij}^2 & h_{ij}^1 & h_{ij}^0 & \cdots & 0 \\ & \vdots & & \ddots & \\ h_{ij}^{N_C-1} & h_{ij}^{N_C-2} & h_{ij}^{N_C-3} & \cdots & h_{ij}^0 \end{bmatrix} \qquad (N_c \times N_c)$$

Note that the coefficients h_{ij}^{N+k} are zero for $k > 0$.

4.2 Calculating the Future Moves

The objective function will be minimized over a finite future horizon N_C. The objective function is simply the sum of the squared control error over time. The objective function for finding the future moves is very similar to the objective function for the model identification. The optimimal vector U_P minmizes the objective function, which can be expressed as a vector norm.

$$\begin{aligned} J_C &= \sum_{k=1}^{N_C} w_1^2 (y_1^r(t+k) - y_1(t+k))^2 + w_2^2 (y_2^r(t+k) - y_2(t+k))^2 \\ &= w_1 \left[Y_1^r - Y_1^C\right]^T \left[Y_1^r - Y_1^C\right] + w_2 \left[Y_2^r - Y_2^C\right]^T \left[Y_2^r - Y_2^C\right] \\ &= w_1 \left[Y_1^r - \mathcal{H}_{11}\mathcal{M}U_1^P - \mathcal{H}_{12}\mathcal{M}U_2^P\right]^T \left[Y_1^r - \mathcal{H}_{11}\mathcal{M}U_1^P - \mathcal{H}_{12}\mathcal{M}U_2^P\right] \\ &+ w_2 \left[Y_2^r - \mathcal{H}_{21}\mathcal{M}U_1^P - \mathcal{H}_{22}\mathcal{M}U_2^P\right]^T \left[Y_2^r - \mathcal{H}_{21}\mathcal{M}U_1^P - \mathcal{H}_{22}\mathcal{M}U_2^P\right] \\ &= \left[W_C Y^R - W_C \mathcal{H}_C \mathcal{M}_C U_P\right]^T \left[W_C Y^R - W_C \mathcal{H}_C \mathcal{M}_C U_P\right] \\ &= \left[B_C - A_C U_P\right]^T \left[B_C - A_C U_P\right] \\ &= \left\| B_C - A_C U_P \right\|_2^2 \end{aligned}$$

where

$$\begin{aligned} \mathcal{H}_C &= \begin{bmatrix} \mathcal{H}_{11} & \mathcal{H}_{12} \\ \mathcal{H}_{21} & \mathcal{H}_{22} \end{bmatrix} \qquad (2N_C \times 2N_C) \\ \mathcal{M}_C &= \begin{bmatrix} \mathcal{M} & 0 \\ 0 & \mathcal{M} \end{bmatrix} \qquad (2N_C \times 2P) \end{aligned}$$

W_C is a diagonal matrix

$$W_C = \left[\begin{array}{ccc|ccc} w_1 & \cdots & 0 & & & \\ & \ddots & & & 0 & \\ 0 & \cdots & w_1 & & & \\ \hline & & & w_2 & \cdots & 0 \\ & 0 & & & \ddots & \\ & & & 0 & \cdots & w_2 \end{array}\right] \qquad (2N_C \times 2N_C)$$

$$\begin{aligned} Y_r &= \begin{bmatrix} Y_1^r \\ Y_2^r \end{bmatrix} \qquad (2N_C \times 1) \\ U_P &= \begin{bmatrix} U_1^P \\ U_2^P \end{bmatrix} \qquad (2P \times 1) \\ B_C &= W_C Y^R \qquad (2N_C \times 1) \\ A_C &= W_C \mathcal{H}_C \mathcal{M}_C \quad (2N_C \times 2P) \end{aligned}$$

4.2.1 Dynamic Matrix Control Solution

Provided the inverse exist, the future moves U_P that minimizes the norm $\left\| B_C - A_C U_P \right\|_2^2$ is found as the normal Least Squares Solution.

$$U_P = \left[A_C^T A_C \right]^{-1} A_C^T B_C$$

Control Example - One Future Move

The very simple impulse response model with $N+1=5$ will be used again.

$$h = \begin{bmatrix} 0 & 0.7 & 0.2 & 0.1 & 0.0 \end{bmatrix}$$

The system is steady at $y(t) = u(t) = 0$ for $t \leq 0$. The control target $y_r(t)$ will be changed from zero to one at time $t = 1$. If only one future move is allowed ($P = 1$), then the solution $u(t) = 1$ ensures the target is met and the infinite horizon objective function is minimized. The solution $U_p = 1$ results in the objective function $J_c = 1.1$. To find the optimal solution for the finite objective function, the following vectors and matrices are needed for $N_C = 5$.

$$Y_r = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T \qquad (5 \times 1)$$

$$W_C = \begin{bmatrix} 1 & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & 1 \end{bmatrix} \qquad (5 \times 5)$$

$$\mathcal{M}_C = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T \qquad (5 \times 1)$$

$$\mathcal{H}_C = \begin{bmatrix} 0. & 0. & 0. & 0. & 0. \\ 0.7 & 0. & 0. & 0. & 0. \\ 0.2 & 0.7 & 0. & 0. & 0. \\ 0.1 & 0.2 & 0.7 & 0. & 0. \\ 0. & 0.1 & 0.2 & 0.7 & 0. \end{bmatrix} \qquad (5 \times 5)$$

The optimal solution is found to be $U_P = [1.091]$ which results in an objective function of $J_c = 1.072$. Figure 5 shows the response of the output. The output sequence does not approach the value of one because the horizon is finite. It is possible to force the solution towards one by extending the horizon. The same result can be achieved by increasing the weight on the last diagonal element in W_C because the output does not change if the horizon is extened past $t = 5$. If $W_C(5,5) = 10,000$ the solution approaches one at $t = 5$.

4.3 Move Penalty

It is often necessary to slow down the movement in the independent variables to achieve robustness against model error. This can be achieved by penalizing changes in the independent variables between to adjacent time intervals. The sequence of input changes can be formulated as:

$$\Delta U_i^C = \begin{bmatrix} \Delta u_i(t+1) \\ \vdots \\ \Delta u_i(t+N_C) \end{bmatrix} = \begin{bmatrix} u_i(t+1) - u_i(t+0) \\ \vdots \\ u_i(t+N_C) - u(t+N_C-1) \end{bmatrix} = QU_i^C \qquad (1)$$

where $u_i(t+0)$ is the input at time zero which is assumed to be zero and thus not part of the sequence. The matrix Q has the form:

$$Q = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ & & & \vdots & & \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \qquad (N_C \times N_C)$$

To make the control less aggressive a term, J_C^U that penalizes large changes in the input, will be added to the objective function, where m_i penalizes movement in input i.

$$\begin{aligned} J_C^U &= \sum_{k=1}^{N_C} m_1^2 \left[u_1(t+k) - u_1(t+k-1)\right]^2 + m_2^2 \left[u_2(t+k) - u_2(t+k-1)\right]^2 \\ &= \sum_{k=1}^{N_C} m_1^2 \left[\Delta u_1(t+k)\right]^2 + m_2^2 \left[\Delta u_2(t+k)\right]^2 \\ &= \left[m_1 \Delta U_1^C\right]^T \left[m_1 \Delta U_1^C\right] + \left[m_2 \Delta U_2^C\right]^T \left[m_2 \Delta U_2^C\right] \\ &= \left\|m_1 Q U_1^C\right\|_2^2 + \left\|m_2 Q U_2^C\right\|_2^2 \\ &= \left\|m_1 Q M U_1^P\right\|_2^2 + \left\|m_2 Q M U_2^P\right\|_2^2 \\ &= \left\|\mathcal{Q}_C \mathcal{M}_C U_P\right\|_2^2 \end{aligned}$$

Where

$$\mathcal{Q}_C = \begin{bmatrix} m_1 \mathcal{Q} & 0 \\ 0 & m_2 \mathcal{Q} \end{bmatrix} \qquad (2N_C \times 2N_C)$$

The term J_C^U will be zero if all future inputs do not change. The combined objective function that minimizes the control error and penalizes movement can be written as:

$$J_C = \left\|B_C - A_C U_P\right\|_2^2 + \left\|\mathcal{Q}_C \mathcal{M}_C U_P\right\|_2^2 = \left\| \begin{bmatrix} B_C \\ O \end{bmatrix} - \begin{bmatrix} A_C \\ -\mathcal{Q}_C \mathcal{M}_C \end{bmatrix} U_P \right\|_2^2 = \left\|B - A U_P\right\|_2^2$$

The solution to the Dynamic Matrix Control problem is the vector U_P that minimizes the combined objective function and can be solved as a Least Squares problem.

$$U_P = \left[A^T A\right]^{-1} A^T B$$

The control problem assumed the system was steady with zero input in the beginning. In practice the future move plan will be calculated from the difference between the future

prediction and the desired target, i.e. the difference between where the prediction shows the process is moving and where the process is desired to be. The calculated future moves will be additive to the current heater inputs.

Control Example - Four Future Moves with Move Penalty
The very simple impulse response model with $N+1=5$ will be used again. This time four future moves will be used ($P=4$). We will use $Nc=8$ in this example. The matrix W_C is a unity matrix of dimension (8×8). The matrix $\mathcal{M}_C$ takes the form:

$$\mathcal{M}_C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (8\times 4)$$

The matrix Q has dimension (8×8). The optimal solution will be found for two different values of the move penalty.

- Solution 1: $m=0.2$ which give $J_C=1.08$ and $U_P = \begin{bmatrix} 1.2993 & 1.0638 & 0.9534 & 1.0013 \end{bmatrix}^T$
- Solution 2: $m=1.0$ which give $J_C=1.85$ and $U_P = \begin{bmatrix} 0.6679 & 0.9108 & 0.9789 & 0.9976 \end{bmatrix}^T$

The input and outputs are shown in figure 6 for the two solutions. The second solution is clearly slower and more damped. The last variable in both input sequences is close to one which would be the solution if the horizon was infinite.

4.4 Open Loop Control

The identified model can be used to generate a control signal that increases the temperature of the two heaters by 20 °C each. The four identified models have $N+1=180$ coefficients each. The plan is to implement the move plan on the *TCLab* device without feedback (Open Loop Control) to see how close the observed temperatures follow what is predicted by the model.

The number of future moves for each of the two heaters will be $P=100$. The horizon will be $N_C = P+N = 100+179 = 279$. The following tuning parameters will be used: $w_1=1.0$,

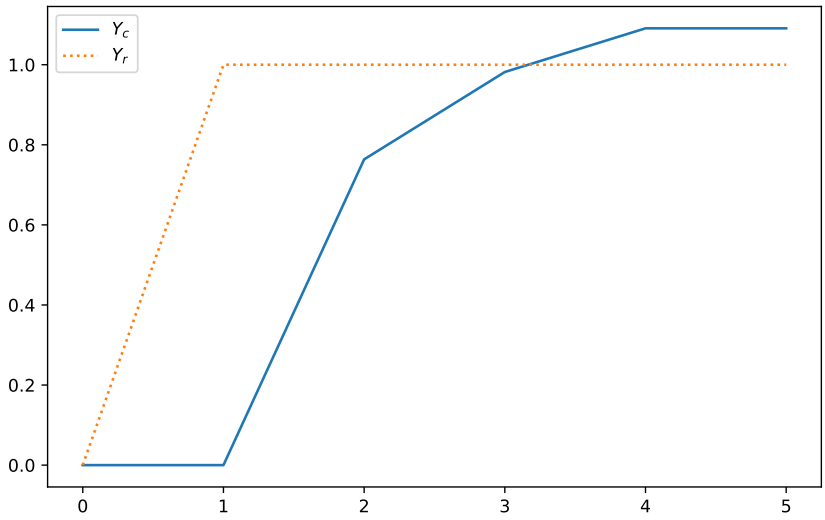


Figure 5: Optimal solution for one future move example. The target is increased to one at $t = 1$. The output does not reach the target because the horizon is final.

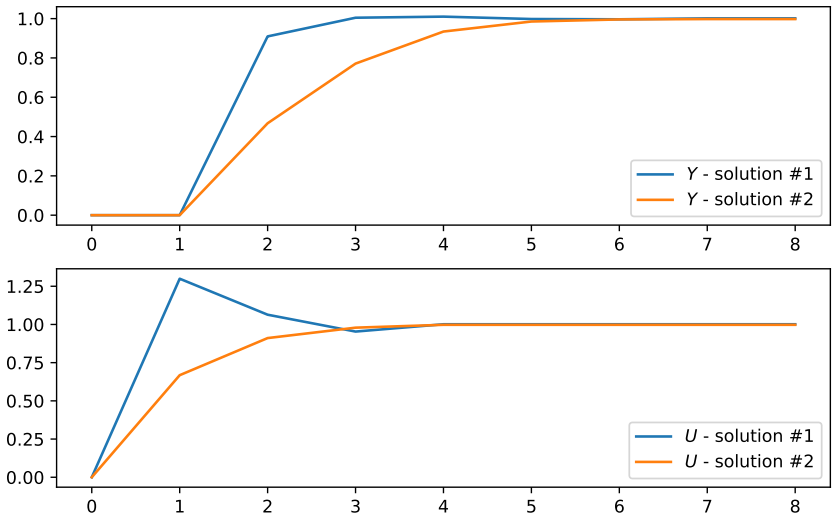


Figure 6: Example of four future moves on simple impulse response model example. Solution 1 uses a move penalty of $m = 0.2$, and solution 2 uses a move penalty of $m = 1.0$. Solution 2 provides a less aggressive response with no overshoot in the input.

$w_2 = 1.0$, $m_1 = 2.1$ and $m_2 = 2.1$. The matrices used in the optimization problem will have the following dimensions.

$$\begin{array}{cccc} \mathcal{H}_C & (558 \times 558) & \mathcal{W}_C & (558 \times 558) \\ \mathcal{Q}_C & (558 \times 558) & \mathcal{M}_C & (558 \times 200) \\ A & (1116 \times 200) & B & (1116 \times 1) \end{array}$$

Thus, to find the 2×100 future moves the matrix $A^T A$ of dimension 200×200 will have to be inverted. The plot in figure 7 shows the input to the two heaters found as the optimal solution. There is overshoot in each of the two heat inputs which helps bring the temperatures to the target faster. The observed temperatures matches the predicted temperatures fairly well for the two heaters, but the model gains are not quite correct as the observed temperatures increase a little more than predicted.

4.4.1 Blocking

It is possible to use fewer future moves by spacing them out over the control horizon, this is called blocking. Doing so will reduce the size of the matrix that has to be inverted. This is acomplished by holding the output constant between moves. If the plan is to use seven ($P = 7$) future moves at the time intervals: 1, 3, 7, 15, 27, 51 and 81, then the vector U_i^P becomes:

$$U_i^P = \left[\begin{array}{ccccccc} u_i(t+1) & u_i(t+3) & u_i(t+7) & \cdots & u_i(t+51) & u_i(t+81) \end{array} \right]^T$$

The matrix $\mathcal{M}$ that maps the actual move sequence U_i^C to U_i^P will take the form:

$$\mathcal{M} = \left[\begin{array}{cccccc} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ & \vdots & & & & \vdots \\ 0 & 0 & & \cdots & 1 & 0 \\ 0 & 0 & & \cdots & 1 & 0 \\ 0 & 0 & & \cdots & 0 & 1 \\ \hline 0 & 0 & & \cdots & 0 & 1 \\ & \vdots & & & & \vdots \\ 0 & 0 & & \cdots & 0 & 1 \end{array} \right] \begin{array}{l} \\ \\ \leftarrow 1 \\ \\ \leftarrow 3 \\ \\ \\ \\ \leftarrow 7 \\ \\ \leftarrow 79 \\ \leftarrow 80 \\ \leftarrow 81 \\ \leftarrow 82 \\ \\ \leftarrow 260 \end{array} \qquad (N_C \times P)$$

With the last move at $t + 81$, the horizon is $N_C = N + 81 = 260$. The following tuning paramters will be used: $w_1 = 1.0$, $w_2 = 1.0$, $m_1 = 1.0$ and $m_2 = 1.0$. The matrices used in the optimization problem will have the following dimensions.

$$\begin{array}{cccc} \mathcal{H}_C & (520 \times 520) & \mathcal{W}_C & (520 \times 520) \\ \mathcal{Q}_C & (520 \times 520) & \mathcal{M}_C & (520 \times 14) \\ A & (1040 \times 14) & B & (1040 \times 1) \end{array}$$

Thus, to find the 2×7 future moves the matrix $A^T A$ of dimension 14×14 will have to be inverted. The plot in figure 8 shows the heater inputs and the predicted temperatures. The future moves are spaced out over the horizon.

4.4.2 Constraints

It is possible to have constraints on the future moves. The future input may be limited (heat input is limited in range 0% to 100%).

$$u_i^{low} \leq u_i(k) \leq u_i^{high}$$

It is also possible to limit the size of each of the future moves.

$$|\Delta u_i(k)| = |u_i(k) - u_i(k-1)| \leq u_i^{move}$$

where u_i^{move} is the maximum allowable change in the input. When constraints are applied to the inputs the optimization problem that minimizes the objective function and satisfies the constraints becomes a Quadratic Program (QP) instead of a simple Least Squares problem.

4.5 Future Prediction without Control

As seen in the modelling section, the predicted temperature measurement for each of the two sensors at the current time t is a function of the last $N + 1$ (including current) heat inputs.

$$\begin{array}{rcl} y_1(t) & = & h_{11}^0 u_1(t) + h_{11}^1 u_1(t-1) + \cdots + h_{11}^N u_1(t-N) \\ & + & h_{12}^0 u_2(t) + h_{12}^1 u_2(t-1) + \cdots + h_{12}^N u_2(t-N) \\ y_2(t) & = & h_{21}^0 u_1(t) + h_{21}^1 u_1(t-1) + \cdots + h_{21}^N u_1(t-N) \\ & + & h_{22}^0 u_2(t) + h_{22}^1 u_2(t-1) + \cdots + h_{22}^N u_2(t-N) \end{array}$$

4.5.1 Prediction Error

The prediction error at time t is defined as the difference between the observed temperature y_i^* and the predicted temperature y_i.

$$e_i(t) = y_i^*(t) - y_i(t)$$

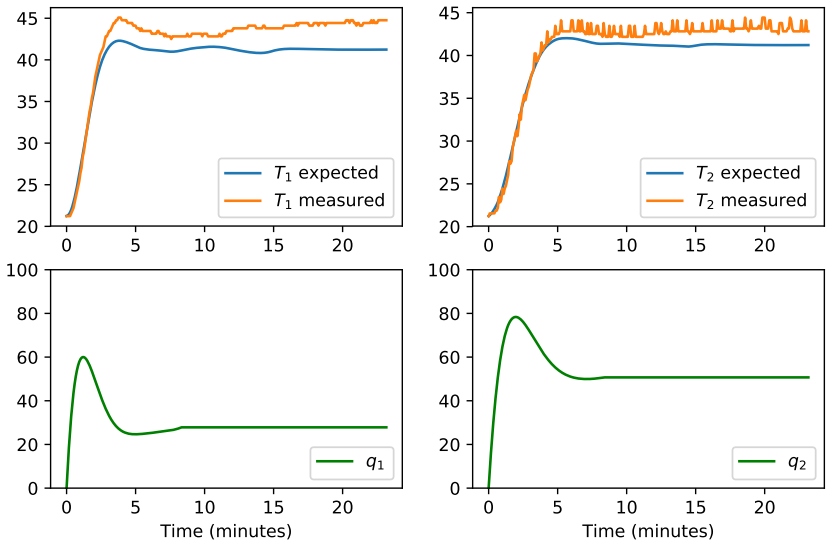


Figure 7: Open loop control example for *TCLab* device. The heater input for each of the two heaters are changed as shown on the plot (green trace). The observed output (orange) from the temperature sensors follows the predicted temperatures (blue) well.

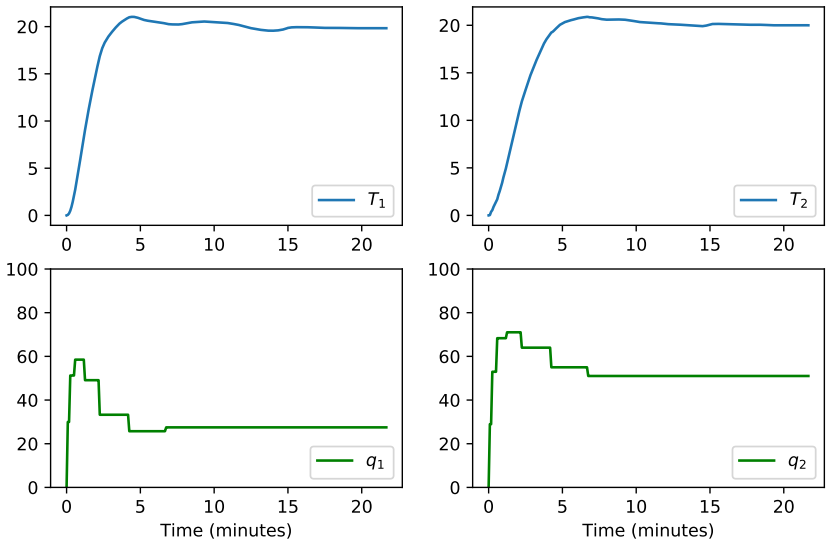


Figure 8: Open loop control example for *TCLab* device with seven future moves. The future heater input moves (green trace) are spaced out over the horizon. The predicted temperature response (blue) for each sensor both increase by 20 °C.

4.5.2 Predicting into the Future

The last input $u_i(t)$ is held constant (i.e. no control) into the future $u_i(t+k) = u_i(t)$ for $k \geq 1$. The future predictions for each output can be written in vector/matrix form.

$$Y_i^P = \mathcal{H}_{i1}^P U_1^P + \mathcal{H}_{i2}^P U_2^P$$

where

$$Y_i^P = \begin{bmatrix} y_i(t+1) \\ y_i(t+2) \\ \vdots \\ y_i(t+N) \\ y_i(t+N+1) \\ \vdots \\ y_i(t+N_c-1) \\ y_i(t+N_c) \end{bmatrix} \qquad U_j^P = \left[\begin{array}{c} u_j(t-N+1) \\ u_j(t-N+2) \\ u_j(t-N+3) \\ \vdots \\ u_j(t) \\ \hline u_j(t+1) \\ u_j(t+2) \\ u_j(t+3) \\ \vdots \\ u_j(t+N_c-1) \\ u_j(t+N_c) \end{array}\right] \begin{array}{l} \\ \\ \\ \\ \leftarrow \text{ Past History } (N) \\ \\ \\ \\ \leftarrow \text{ Future constant inputs } (N_C) \\ \\ \\ \end{array}$$

and

$$\mathcal{H}_{ij}^P = \left[\begin{array}{ccccc|ccccc} h_{ij}^N & h_{ij}^{N-1} & h_{ij}^{N-2} & \cdots & h_{ij}^1 & h_{ij}^0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & h_{ij}^N & h_{ij}^{N-1} & \cdots & h_{ij}^2 & h_{ij}^1 & h_{ij}^0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & h_{ij}^N & \cdots & h_{ij}^3 & h_{ij}^2 & h_{ij}^1 & h_{ij}^0 & \cdots & 0 & 0 \\ \vdots & & & & & \vdots & & & & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & h_{ij}^N & h_{ij}^{N-1} & h_{ij}^{N-2} & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & h_{ij}^N & h_{ij}^{N-1} & \cdots & 0 & 0 \\ \vdots & & & & & & & & & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & h_{ij}^0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & h_{ij}^1 & h_{ij}^0 \end{array}\right] \qquad N_C \times (N + N_C)$$

Future Prediction without Control

The very simple impulse response model with $N+1=5$ will be used again.

$$h = \left[\begin{array}{ccccc} h_0 & h_1 & h_2 & h_3 & h_4 \end{array} \right] = \left[\begin{array}{ccccc} 0 & 0.7 & 0.2 & 0.1 & 0.0 \end{array} \right]$$

We will use $N_C = 5$ in this example as in the one future move example. We will assume $u(t) = 0$ for $t < 0$ and that $u(t) = 1$ for $t \geq 0$. All future inputs will be assumed constant (no control) $u(t+k) = u(t)$ for $k \geq 1$.

$$\begin{aligned} U^P &= \left[\begin{array}{cccc|ccccc} u(t-3) & u(t-2) & u(t-1) & u(t) & u(t+1) & u(t+2) & u(t+3) & u(t+4) & u(t+5) \end{array} \right]^T \\ &= \left[\begin{array}{cccc|ccccc} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \right]^T \end{aligned}$$

The prediction matrix $\mathcal{H}^P$ becomes:

$$\mathcal{H}^P = \left[\begin{array}{cccc|ccccc} 0 & 0.1 & 0.2 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0.2 & 0.7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0.2 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0.2 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0.2 & 0.7 & 0 \end{array} \right]$$

and the predicted future outputs are:

$$\begin{aligned} Y^P = \mathcal{H}^P U^P &= \left[\begin{array}{ccccc} y(t+1) & y(t+2) & y(t+3) & y(t+4) & y(t+5) \end{array} \right]^T \\ &= \left[\begin{array}{ccccc} 0.7 & 0.9 & 1 & 1 & 1 \end{array} \right]^T \end{aligned}$$

As expected, the future output value settles out at the value 1.

4.6 The DMC Algorithm

All the necessary pieces are now in place to implment the DMC Algorithm. The following steps are executed at each time interval:

1. Sample data (inputs and outputs)
2. Calculate current prediction error
3. Calculate future predictions based upon past inputs
4. Get desired target for outputs (may come from steady state optimizer not yet discussed)
5. Calculate the future moves that bring the outputs to the targets
6. Implement first move and repeat sequence next time interval

DMC Algorithm - Closed Loop Control

The *TCLab* device will be controlled with $P = 100$ future moves as in the Open Loop Control example. The tuning parameters will be selected to provide a slightly slower response than the Open Loop Control example (less overshoot). The following tuning paramters will be used:

$$\begin{aligned} w_1 &= w_2 = 0.5 \\ m_1 &= m_2 = 4.0 \end{aligned}$$

The closed loop DMC control is shown in figure 9. The future input target (last move in the future move sequence) at each time interval is shown together with the actual input. If there are no unmeasured disturbances or model error, the input target will be constant if the set-point for the output is constant.

The data is sampled at each interval and the control calculations are performed. Once the control calculations are complete the calculated heat inputs are sent to the device. The control calculations and communication with the device is complete in slightly less than one second with most of the time spent on the communication as the calculations for a small problem are very fast. The control calculations are repeated every 5 seconds.

Figure 10 illustrates the control calculation after 4 minutes. The future prediction without control shows that the output does not reach the target because the input sequence has not yet stabilized.

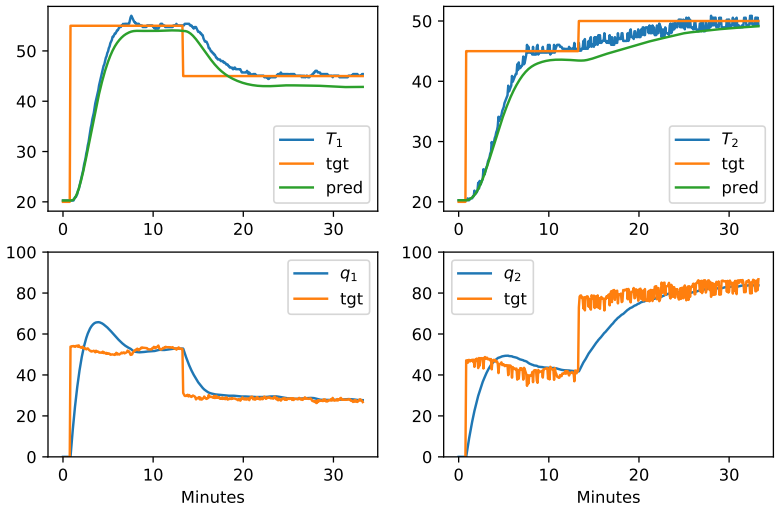


Figure 9: Closed loop DMC control example for *TCLab* device. The upper plots show the temperatures (blue), the set-points (orange) and the open loop prediction (green) calculated from the heater inputs. The set-points are changed after 13 minutes. The lower plots show the heater inputs (blue) and the calculated future target (orange) at steady state.

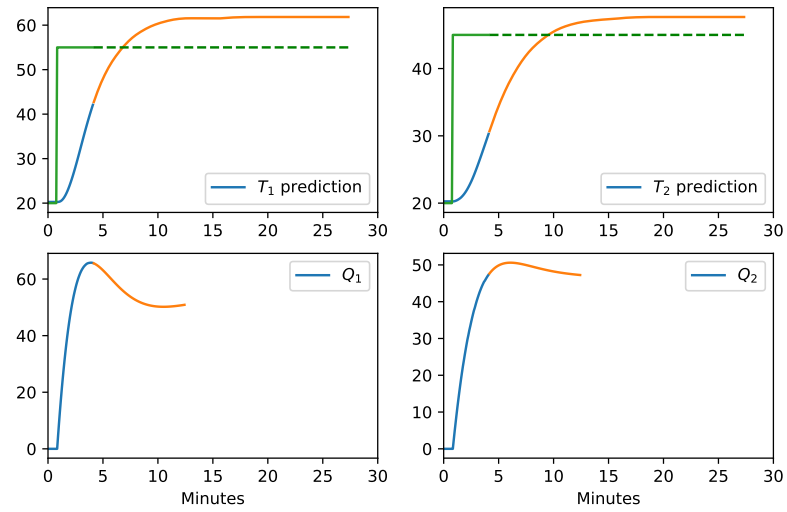


Figure 10: Upper plots show predictions at 4 minutes. The blue trace shows past prediction, the orange trace shows the future prediction with the heat input held constant at current value (no control). The green trace is the desired temperature target. The lower plots show the past heat inputs (blue) and the 100 future planned heat input changes (orange) not yet implemented.

5 Optimization

Industrial DMC type controllers often have an optimizer that sets the desired targets for the inputs and outputs, Qin and Badgwell [5]. This allows for managing systems where the number of inputs are different from the number of outputs, and it allows for driving the process in a desired economical direction.

The steady state model for the *TCLab* device can be written as:

$$\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} - \begin{bmatrix} T_{10} \\ T_{20} \end{bmatrix} = \begin{bmatrix} 0.5930 & 0.0967 \\ 0.2546 & 0.2869 \end{bmatrix} \begin{bmatrix} Q_1 - Q_{10} \\ Q_2 - Q_{20} \end{bmatrix} = G \begin{bmatrix} \Delta Q_1 \\ \Delta Q_2 \end{bmatrix}$$

Assuming the initial temperaures are $T_{10} = T_{20} = 19$ °C (typical indoor temperaure) and that the initial heat inputs are $Q_{10} = Q_{20} = 0$ the possible operating map can be found knowing the heat inputs are restriced to the range $0 \leq Q_i < 100$. The possible predicted operating range is shown as the red box on figure 11. The real data from the steps test (blue dots) are overlayed on the plot. The model predicts the possible operating range well.

5.1 Constraints

The limits on the inputs are typical hard limits than cannot be vilolated.

$$\begin{aligned} Q_1^L \leq \;& Q_1 \;\leq Q_1^H \\ Q_2^L \leq \;& Q_2 \;\leq Q_2^H \end{aligned}$$

The limits on the temperatures are typically soft as this allows for managing constraints that are not possible to achieve in a real life situtation. The limits on the temperatures can be changed by the addition of positive slack variables. If the optimization problem is feasible, the slack variables will be zero.

$$\begin{aligned} T_1^L - S_1^L \leq \;& T_1 \;\leq T_1^H + S_1^L \\ T_2^L - S_2^L \leq \;& T_2 \;\leq T_2^H + S_1^H \end{aligned}$$

Combining the limits on the inputs and outputs we get

$$\begin{bmatrix} T_1^L - T_{10} \\ T_2^L - T_{20} \\ \hline Q_1^L - Q_{10} \\ Q_2^L - Q_{20} \end{bmatrix} \leq \begin{bmatrix} T_1 - T_{10} + S_1^L - S_1^H \\ T_2 - T_{20} + S_2^L - S_2^H \\ \hline Q_1 - Q_{10} \\ Q_2 - Q_{20} \end{bmatrix} \leq \begin{bmatrix} T_1^H - T_{10} \\ T_2^H - T_{20} \\ \hline Q_1^H - Q_{10} \\ Q_2^H - Q_{20} \end{bmatrix}$$

or in more compact form as:

$$\left[\begin{array}{c} T_1^L - T_{10} \\ T_2^L - T_{20} \\ \hline Q_1^L - Q_{10} \\ Q_2^L - Q_{20} \end{array}\right] \leq \left[\begin{array}{c} G \\ \hline I \end{array}\right] \left[\begin{array}{c} \Delta Q_1 \\ \Delta Q_2 \end{array}\right] + \left[\begin{array}{c} I \\ \hline 0 \end{array}\right] \left[\begin{array}{c} S_1^L \\ S_2^L \end{array}\right] + \left[\begin{array}{c} -I \\ \hline 0 \end{array}\right] \left[\begin{array}{c} S_1^H \\ S_2^H \end{array}\right] \leq \left[\begin{array}{c} T_1^H - T_{10} \\ T_2^H - T_{20} \\ \hline Q_1^H - Q_{10} \\ Q_2^H - Q_{20} \end{array}\right]$$

Combining the limits on inputs and outputs with the condition that the slack variables are positive, the constraints can be written in matrix form as

$$x^L \leq Ax \leq x^H$$

where

$$x^T = \left[\begin{array}{ccc} \Delta Q & S^L & S^H \end{array}\right]^T$$

and

$$A = \left[\begin{array}{ccc} G & I & -I \\ I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{array}\right] \qquad x^L = \left[\begin{array}{c} T_1^L - T_{10} \\ T_2^L - T_{20} \\ \hline Q_1^L - Q_{10} \\ Q_2^L - Q_{20} \\ \hline 0 \\ 0 \\ \hline 0 \\ 0 \end{array}\right] \qquad x^H = \left[\begin{array}{c} T_1^H - T_{10} \\ T_2^H - T_{20} \\ \hline Q_1^H - Q_{10} \\ Q_2^H - Q_{20} \\ \hline \infty \\ \infty \\ \hline \infty \\ \infty \end{array}\right]$$

5.2 Constrained Optimization

The optimization problem can be solved as a Quadratic Program (QP) if a linear cost variable is assigned to each heat input and a quadratic penalty is assigned to each slack variable. The penalty assigned to the slack variables are often large positive numbers to drive the slack variables to zero if possible.

$$\begin{array}{rl} \text{minimize} & \frac{1}{2} x^T P x + q^T x \\ \text{subject to} & x^L \leq Ax \leq x^H \end{array}$$

where

$$P = \left[\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_1^L & 0 & 0 & 0 \\ 0 & 0 & 0 & w_2^L & 0 & 0 \\ 0 & 0 & 0 & 0 & w_1^H & 0 \\ 0 & 0 & 0 & 0 & 0 & w_2^H \end{array}\right] \qquad q = \left[\begin{array}{c} c_1 \\ c_2 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}\right]$$

Simple Optimization

Assume we have the following constraints for the *TCLab* device.

$$\begin{aligned} 10 \leq \quad & Q_1 \quad \leq 90 \\ 20 \leq \quad & Q_2 \quad \leq 80 \\ 30 \leq \quad & T_1 \quad \leq 60 \\ 30 \leq \quad & T_2 \quad \leq 50 \end{aligned}$$

The temperaure and heat input constraints are shown in figure 11. Starting from the initial temperaures $T_{10} = T_{20} = 19$ °C and zero initial heat input, the constraints can be written without slack variables as:

$$\begin{bmatrix} 30-19 \\ 30-19 \\ \hline 10 \\ 20 \end{bmatrix} \leq \begin{bmatrix} G \\ \hline I \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} + \begin{bmatrix} 60-19 \\ 50-19 \\ \hline 90 \\ 80 \end{bmatrix}$$

The following cost variables will be assigned: $c_1 = -0.1$ and $c_2 = -1.0$. Without slack variables the matrix P becomes a zero matrix and the optimization problem is a Linear Program (LP). The LP will be solved using the OSQP algorithm. The OSQP Algorithm is very fast and has been used for control of embedded systems, Stellato [7]. The OSQP solver is available as a Python library. The optimal solution (shown as a purple dot on figure 11) becomes

$$Q = \begin{bmatrix} 31.61 & \mathbf{80.00} \end{bmatrix}^T \qquad \text{or} \qquad T = \begin{bmatrix} 45.48 & \mathbf{50.00} \end{bmatrix}^T$$

where the active constraints are marked in **bold**. Heater #2 is at the maximum, and the limit on temperature sensor #2 prevents heater #1 from increasing the input further.

Constraint Violation

If the temperature constraints are made narrower, they become impossible to satisfy within the hard limits defined by the heat inputs.

$$\begin{aligned} 55 \leq \quad & T_1 \quad \leq 60 \\ 30 \leq \quad & T_2 \quad \leq 35 \end{aligned}$$

If the QP is solved for $w_1^L = w_1^H = w_2^L = w_2^H = 10000$, the solution is:

$$x = \begin{bmatrix} Q_1 & Q_2 & S_1^L & S_2^L & S_1^H & S_2^H \end{bmatrix}^T = \begin{bmatrix} 54.8 & 20 & 1.58 & 0 & 0 & 3.68 \end{bmatrix}^T$$

with both temperatures $T = \begin{bmatrix} 53.41 & 38.68 \end{bmatrix}^T$ being outside the specified limits. Note that two of the slack variables are non zero. The QP smears the error when constraints cannot be met.

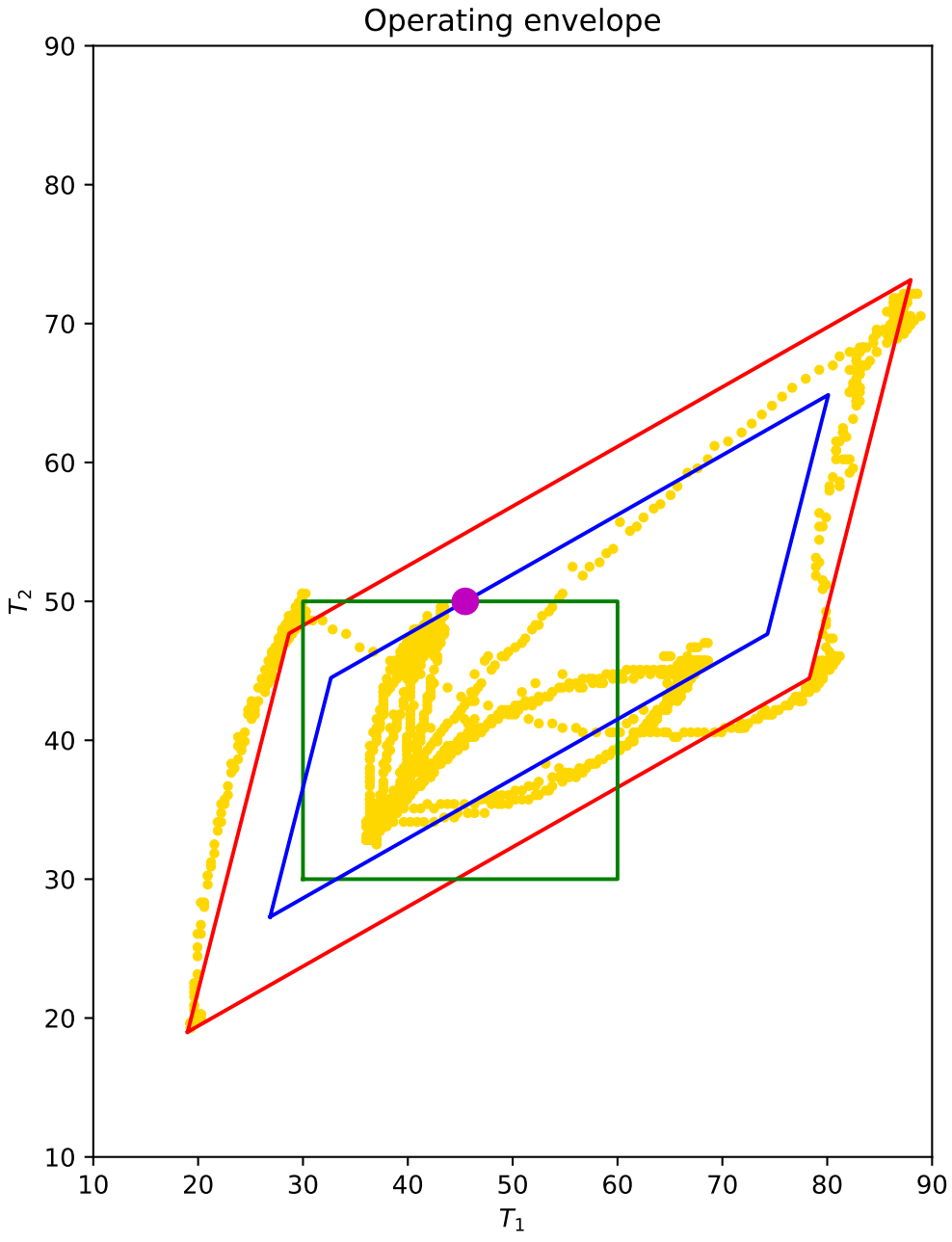


Figure 11: The step test data (orange dots) are overlayed on the predicted operating map defined by the red lines. The green square shows the constraints on the temperatures and the blue lines show the constraints on the heat inputs. The optimal solution from the simple optimization example is shown as the purple dot.

5.3 Optimizing Control

The optimizer can be connected to the control algorithm by evaluating the optimization problem at the predicted future steady state (corrected for the prediction error). When solved this way, the change in heat input ΔQ is the steady change that moves the process to the optimal operating point. The optimization problem will be solved at each control execution. Stability analysis of DMC type controllers driven by an LP or QP optimizer was analyzed by Ying and Joseph [6].

Optimizing Control
Assume we have the following constraints:

$$0 \le Q_1 \le 75 \qquad 0 \le Q_2 \le 75$$
$$30 \le T_1 \le 60 \qquad 30 \le T_2 \le 52$$

If the initial cost vector is $q^T = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$ the steady state solution will be $T_1 = T_2 = 30$ °C. The plot in figure 12 shows closed loop control with the optimizer active. At 20 minutes, the cost vector is changed to $q^T = \begin{bmatrix} -0.1 & -1.0 \end{bmatrix}^T$ and the active constraint set changes to $T_2 = 52$ °C and $q_2 = 75$. At this constraint set the temperature T_2 is essentially controlled by q_1. Examples like this where the primary control handle for a variable is tied up with another constraint often arise when optimizing industrial processes.

6 Final Words

Hopefully, this document inspired you to learn about the math used in Dynamic Matrix Control. Understanding the underlaying math used for solving the Dynamic Matrix Control algorithm is not a requirement to work with this technology in practice, but it is useful to understand how the math works to better understand the tradeoffs when changing tuning constants.

Controlling the *TCLab* is more interesting than controlling a simulated process. The second temperature sensor on the device used for the examples in this document has erratic readings. This is possibly caused by the glue holding the sensor close to the heater being broken which allows for the sensor to slightly move. Such problems are similar to the problems encountered on industrial equipment where it often is necessary to work around broken equipment and instrumentation.

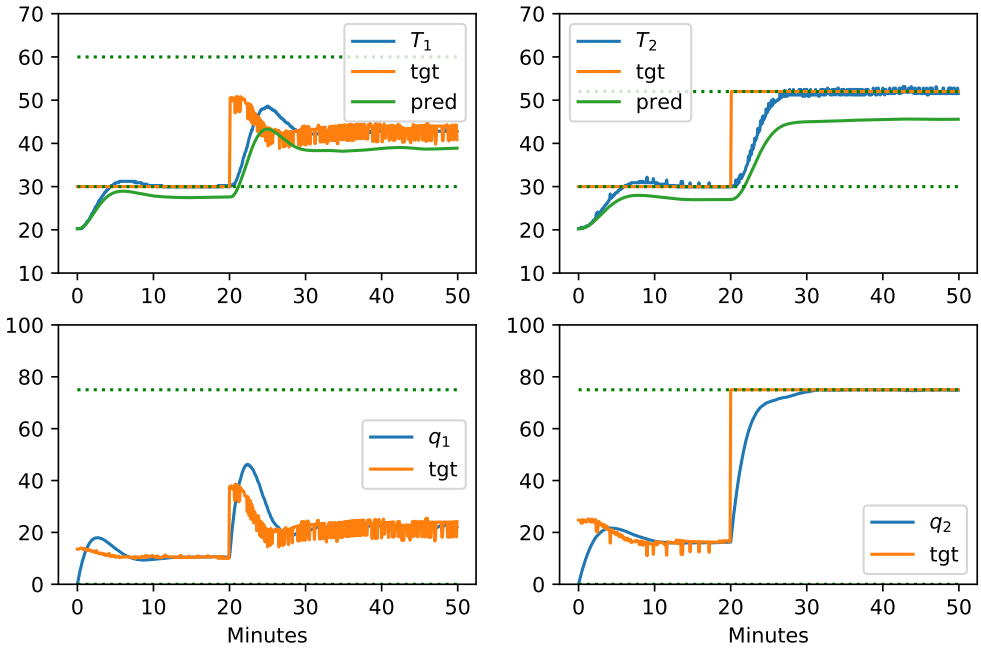


Figure 12: Optimizing DMC Control. Initial cost factors are $c_1 = c_2 = 1$ which results in the optimal solution $T_1 = T_2 = 30$. After 20 minutes the cost factors are changed to $c_1 = -0.1$ and $c_2 = -1.0$ which results in the optimal solution $T_2 = 52$ and $q_2 = 75$. At this constraint set, T_2 is primarily being controlled by q_1. The future targets for T_1 and q_1 calculated at 20 minutes when the costs are changed are different from where these values finally settle out after 50 minutes. This is a sign of model errors.

References

[1] Hedengren, J., Kantor, J., Computer Programming and Process Control Take-Home Lab, *CACHE News* Summer **2020**

https://cache.org/sites/default/files/sum-2020-take-home-lab.pdf.

http://apmonitor.com/pdc/index.php/Main/ArduinoTemperatureControl

[2] Cutler, C.R. and Ramaker, B.L., Dynamic Matrix Control - A Computer Control Algorithm, *AICHE 86th National Meeting*, Paper S1b, April **1979**.

[3] Cutler, C.R., Morshedi, A.M., Haydel, J.J., An Industrial Perspective on Advanced Control, *AICHE meeting Washington D.C.* October **1983**.

[4] Dayal, B., MacGregor J., Identification of Finite Impulse Response Models: Methods and Robustness Issues, *Ind. Eng. Chem. Res.* pp. 4078-4090, 35, **1996**

[5] Qin, J., Badgwell, T., A Survey of Industrial Model Predictive Control Technology *Control Engineering Practice* 11(7):733-764, **2003**

[6] Ying, C., Joseph, B. Performance and Stability Analysis of LP-MPC and QP-MPC Cascade Control Systems, *AIChE Journal*, Vol 45, No. 7, 1521-1534, **1999**

[7] Stellato, B., Banjac, G., Goulart, P., Bemporad, A., Boyd, S., OSQP: an operator splitting solver for quadratic programs, *Mathematical Programming Computation* 12:637–672, **2020**

https://web.stanford.edu/ boyd/papers/pdf/osqp.pdf

[8] Python and relevant packages

https://www.python.org/.

https://numpy.org/

https://scipy.org/

https://matplotlib.org/

https://pypi.org/project/osqp/

Appendix

Python Examples

Python is used to illustrate all concepts in this document. Below is a list of the Python scripts with a short description. The scripts appear in the order they are used in the document. All examples can be run from the command line using Python.

test_connection.py Performs a communication test to the *TCLab* device.

mylib.py This is a python library with common functions used in the scripts listed above.

steptest.py Performs a step test of the device where the heat input is changed and the temperatures are recorded. The data is used to develop the dynamic model.

steptest_plot.py This script plots the data from the step test shown on figure 1.

model_id.py This script identifies the dynamic models of the *TCLab* device based upon the step test data. The dynamic models are shown on figures 2 and 3.

predict.py This script uses the identified model to predict the data from the step test shown on figure 4.

one_move.py This script shows an example of control with one future move for a very simple impulse response model and generates the plot shown on figure 5.

four_moves.py This script illustrates the effect of move penalty shown in figure 6.

open_loop.py This script performs open loop control of the *TCLab* device and attempts to increase the temperature by 20 °C of each temperature sensor.

open_loop_plot.py Plots the data from the open loop experiment shown in figure 7.

open_loop_blocking.py This script created the plot in figure 8

future_prediction.py This script is used in the Future Prediction Example.

control.py DMC control of the *TCLab* device.

control_plot.py The DMC control data is shown on figure 9.

control_plot2.py The DMC control data is shown on figure 10.

envelope.py The operating envelope shown on figure 11.

violate.py The calculations for the QP constraint violation example.

optimize.py Optimizing DMC control of the *TCLab* device.

optimize_plot.py The optimized DMC control is shown on figure 12.

Python Environment

The examples were developed using the Anaconda environment for Python. Anaconda allows for configuring a local Python installation where the versions of the packages used can be controlled. This allows the user to ensure the packages will work together. All examples were developed using the older version 2.7 of Python.

The following Python packages are needed in addition to the standard python installation.

numpy Numerical algorithms for solving matrix equations

scipy Numerical algorithms for sparse matrices

matplotlib Plotting of data

osqp QP solver

pyserial Serial communication

tclab Package to communicate with the *TCLab* device.

A *conda* environment named *test* with the required python packages can be installed with the following commands.

1. *conda create –name test python=2.7*
2. *conda activate test*
3. *conda install numpy scipy matplotlib pyserial*
4. *pip install tclab*
5. *pip install osqp*

The installation can be checked with the following command which creates the plot shown in figure 5:

```
python one_move.py
```

The connection to the *TCLab* device can be checked with the following command:

```
python test_connection.py
```

Linux Troubleshooting

The *TCLab* device was able to communicate with Python on Windows 10. Getting the communication working from Ubuntu Linux (Version 20.04 LTS) took a little more time. Luckily, the *TCLab* device is built on Arduino which makes it easy to find information on the internet.

The problem turned out to be lack of permission to access the USB serial connection used to communicate between Python and the *TCLab* Arduino board. Below is a series of troubleshooting steps that may be useful.

The Python statement *a=tclab.TCLab()* fails with the error **Failed to Connect**.

```
Python 2.7.18 |Anaconda, Inc.| (default, Apr 23 2020, 22:42:48)
[GCC 7.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import tclab
>>> a=tclab.TCLab()
TCLab version 0.4.9
Traceback (most recent call last):
    .......
    raise RuntimeError('Failed to Connect.')
RuntimeError: Failed to Connect.
```

Once the Arduino board is connected to the USB port, the command *dmesg* will indicate an Arduino device is connected to the serial port *ttyACM0* (name may be different on other computers).

```
[35203.646163] usb 1-1.1: new full-speed USB device number 7 using ehci-pci
[35203.758130] usb 1-1.1: New USB device found, idVendor=2341, idProduct=8036  .....
[35203.758135] usb 1-1.1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[35203.758138] usb 1-1.1: Product: Arduino Leonardo
[35203.758140] usb 1-1.1: Manufacturer: Arduino LLC
[35203.758818] cdc_acm 1-1.1:1.0: ttyACM0: USB ACM device
```

The output from the command *lsusb* also indicates an Arduino device is connected to USB.

```
Bus 001 Device 006: ID 2341:8036 Arduino SA Leonardo (CDC ACM, HID)
```

The Python serial package provides a command tool to check for serial ports. The command *python -m serial.tools.list_ports -v* produces the following output.

```
/dev/ttyACM0
    desc: Arduino Leonardo
    hwid: USB VID:PID=2341:8036 LOCATION=1-1.1:1.0
/dev/ttyS4
    desc: n/a
    hwid: n/a
2 ports found
```

Trying to establish a connection to the serial port using Python fails with the error **Permission denied**.

```
Python 2.7.18 |Anaconda, Inc.| (default, Apr 23 2020, 22:42:48)
[GCC 7.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import serial
>>> a = serial.Serial('/dev/ttyACM0', baudrate=9600, timeout=2.0)
Traceback (most recent call last):
  .....
  serial.serialutil.SerialException:
    [Errno 13] could not open port /dev/ttyACM0:
    [Errno 13] Permission denied: '/dev/ttyACM0'
```

The command *ls -la /dev/ttyACM0* shows the serial device exists and that only members of the *dialout* group has read/write permissions.

```
crw-rw---- 1 root dialout 166, 0 Oct 11 13:56 /dev/ttyACM0
```

One solution is to execute the command: *sudo chmod a+rw /dev/ttyACM0* every time the *TCLab* board is connected to the USB port. The command *ls -la /dev/ttyACM0* will indcate the permissions were changed and that all users access now have access.

```
crw-rw-rw- 1 root dialout 166, 0 Oct 12 02:25 /dev/ttyACM0
```

A more permanent solution is to add the current user to the *dialout* group with the following command: *sudo usermod -a -G dialout $USER*. This change seems to require a reboot (not just log-out/log-in) to take effect. After the change the command *groups* will show the user is a member of the *dialout* group.

Once the serial connecton to the Arduino device is working, the Python statement

```
a=tclab.TCLab()
```

will produce output similar to what is listed below.

```
TCLab version 0.4.9
Arduino Leonardo connected on port /dev/ttyACM0 at 115200 baud.
TCLab Firmware 1.4.3 Arduino Leonardo/Micro.
```