054414 Process Control System Design

LECTURE 12: MODEL PREDICTIVE CONTROL

Daniel R. Lewin
Department of Chemical Engineering
Technion, Haifa, Israel

Objectives

On completing this section, you should:

❶ Be able to explain to a layman how model predictive control (MPC) works.

❷ Understand why MPC can improve on the performance that can be expected from decentralized controllers.

❸ Be able to correctly tune MPC to obtain a good response.

❹ Be able to implement MPC using MATLAB.

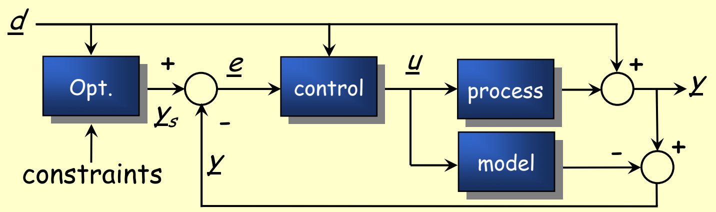
Refs: 1) Seborg, D. E., Edgar, T. F. and Mellichamp, D. A., Process Dynamics and Control, 2nd Ed., Wiley (2004).
2) Ogunnaike, B. A. and Ray, W. H., Process Dynamics, Modeling and Control, Oxford, pp. 1000-1008 (1994).

Introduction

Thus far, digital controllers have been designed using relatively simple models to predict the future values of the process outputs. We shall now consider a more general form of model predictive control in which:

❶ The set points are computed based on a constrained optimization of a steady-state process model.

❷ A model-based predictive algorithm ensures that the process outputs track the setpoints over a prespecified horizon.



Constrained Optimization

The most common approach is to perform constrained optimization using a static process model at regular intervals. This ensures that the process is maintained at the optimal operating level when confronted with such things as:

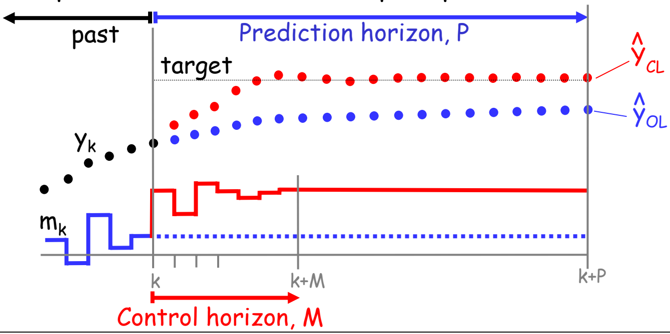
- Changes in product demands.
- Disturbances such as feedstock changes
- Changes in equipment availability

Despite the fact that processes can be highly nonlinear, the most commonly-used method is LP. This has been covered in other courses (054374, 054402), so it will not be discussed further here. This leaves MPC to cover…

MPC made simple…

MPC is a model-based control strategy using models in 2 ways:

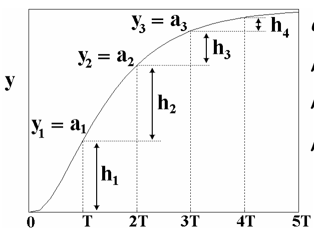
- Uses a reliable **model** to predict effect of past control moves on P future outputs, assuming no future moves.
- Uses the same **model** to compute the optimal M controller moves. Implement first move and repeat procedure.



Discrete Convolution Models

Evidently, a discrete model describing the effect of inputs on outputs is required. Commonly, discrete impulse response, or convolution, models are used.

Consider the response of a typical process to a unit step input:



a_i are the step response coefficients

h_i are the impulse response coefficients

$h_i = a_i - a_{i-1}, i = 1,\ldots,H$ - model horizon.

$h_0 = 0$ (what does this assume?)

H, the model horizon, is selected to capture 99% of the transient.

A discrete convolution model is: $\tilde{y}_{k+1} = \sum_{i=1}^{H} h_i m_{k+1-i}$

Predicted value of y

Manipulated variable at $k+1-i$

Discrete Convolution Models

Discrete convolution model: $\tilde{y}_{k+1} = \sum_{i=1}^{H} h_i m_{k+1-i}$ (11.1)

$$\tilde{y}_{k+1} = h_1 m_k + h_2 m_{k-1} + \ldots + h_H m_{k+1-H}$$
where $m_{k+1-H} = 0$ when $k + 1 - H < 0$ (11.2)

Define $\Delta m_k = m_k - m_{k-1}$ and substituting in (11.2):

$$\tilde{y}_{k+1} = h_1 m_k + h_2 m_{k-1} + \ldots + h_H m_{k+1-H}$$
$$= (a_1 - a_0) m_k + (a_2 - a_1) m_{k-1} + \ldots + (a_H - a_{H-1}) m_{k+1-H}$$
$$= a_0 m_k + a_1 (m_k - m_{k-1}) + \ldots + a_{H-1} (m_{k+1-H} - m_{k+1-H}) + a_H m_{k+1-H}$$

Hence: $\tilde{y}_{k+1} = \sum_{i=1}^{H} a_i \Delta m_{k+1-i}$ (11.3)

Note that this assumes that $\Delta m_{k+1-H} = m_{k+1-H} - m_{k-H} = m_{n+1-H}$

Discrete Convolution Models

Applying Eq. (11.3), a step change in m at $k = 0 \Rightarrow \Delta m_0 = m_0 - m_{-1}$, with $\Delta m_k = 0$, $k > 0$:

$k = 0$: $\tilde{y}_1 = \sum_{i=1}^{H} a_i \Delta m_{1-i}$
$$= a_1 \Delta m_0 + a_2 \underbrace{\Delta m_{-1}}_{0} + \ldots = a_1 \Delta m_0$$

$k = 1$: $\tilde{y}_2 = \sum_{i=1}^{H} a_i \Delta m_{2-i}$
$$= a_1 \underbrace{\Delta m_1}_{0} + a_2 \Delta m_0 + a_3 \underbrace{\Delta m_{-1}}_{0} + \ldots = a_2 \Delta m_0$$

In general, $\tilde{y}_k = a_k \Delta m_0$ – predicted values are step response coefficients multiplied by Δm_0.

ZT-Convolution Model Equivalence

We shall now see how these convolution models are equivalent to transfer function models studied so far. As an example, consider the process: $p(s) = \frac{1}{s+1}$ with a hold and T = 0.2.

Then $HP(z) = \dfrac{\left(1 - e^{-0.2}\right) z^{-1}}{1 - e^{-0.2} z^{-1}} = \dfrac{0.1813 z^{-1}}{1 - 0.8187 z^{-1}}$

By long division: $HP(z) = 0.1813 z^{-1} + 0.1484 z^{-2} + 0.1215 z^{-3} + 0.0995 z^{-4} + 0.0815 z^{-5} + \ldots \qquad (11.4)$

To obtain unit step response: $\hat{Y}(z) = HP(z) M(z) = HP(z) \cdot \frac{1}{1-z^{-1}}$

y(T) y(2T) y(3T)

By long division: $\hat{Y}(z) = 0.1813 z^{-1} + 0.3297 z^{-2} + 0.4512 z^{-3} + 0.5507 z^{-4} + 0.6321 z^{-5} + \ldots \qquad (11.5)$

This is identical to the response: $y(kT) = 1 - e^{-kT}$

12-9 PROCESS CONTROL SYSTEM DESIGN - (c) Daniel R. Lewin Model Predictive Control

ZT-Convolution Model Equivalence

Note that Eq. (11.3) is $\tilde{y}_{k+1} = \sum_{i=1}^{H} a_i \Delta m_{k+1-i}$

Taking Z-transforms of this equation gives:

$$\hat{Y}(z) z^{k+1} = \sum_{i=1}^{H} a_i \Delta \hat{M}(z) z^{k+1-i} \;\Rightarrow\; \frac{\hat{Y}(z)}{\Delta \hat{M}(z)} = \sum_{i=1}^{H} a_i z^{-i} \qquad (11.6)$$

Hence, when $\Delta m_o = 1$ ($\Delta \hat{M}(z)$), Eq. (11.6) is equivalent to Eq. (11.5), so coefficients $a_1 = 0.1813$, $a_2 = 0.3297$, etc., can be obtained by measuring the unit-step response.

Furthermore, we can write Eq. (11.1): $\tilde{y}_{k+1} = \sum_{i=1}^{H} h_i m_{k+1-i}$

in the Z-domain: $\dfrac{\hat{Y}(z)}{\hat{M}(z)} = HP(z) = \sum_{i=1}^{H} h_i z^{-i} \qquad (11.7)$

Hence, Eq. (11.7) is equivalent to Eq. (11.4), whose coefficients are h_1, h_2, etc. and we can check that $h_i = a_i - a_{i-1}$.

12-10 PROCESS CONTROL SYSTEM DESIGN - (c) Daniel R. Lewin Model Predictive Control

Matrix Form for Predictive Control

In general, at sample k, we will compute:

M future input changes - $\Delta m_k, \Delta m_{k+1}, \ldots, \Delta m_{k+M-1}$
input horizon

and **P** predicted future outputs - $\tilde{y}_{k+1}, \tilde{y}_{k+2}, \ldots, \tilde{y}_{k+P}$
prediction horizon

Generalizing Eq. (11.4):

$$\tilde{y}_{k+1} = \sum_{i=1}^{H} a_i \Delta m_{k+1-i} = \underbrace{a_1 \Delta m_k}_{future} + \underbrace{\sum_{i=2}^{H} a_i \Delta m_{k+1-i}}_{past}$$

$$\tilde{y}_{k+2} = \sum_{i=1}^{H} a_i \Delta m_{k+2-i} = \underbrace{\sum_{i=1}^{2} a_i \Delta m_{k+2-i}}_{future} + \underbrace{\sum_{i=3}^{H} a_i \Delta m_{k+2-i}}_{past}$$

12-11 PROCESS CONTROL SYSTEM DESIGN - (c) Daniel R. Lewin Model Predictive Control

Matrix Form for Predictive Control

$$\tilde{y}_{k+M} = \underbrace{\sum_{i=1}^{M} a_i \Delta m_{k+M-i}}_{future} + \underbrace{\sum_{i=M+1}^{H} a_i \Delta m_{k+M-i}}_{past}$$

$$\tilde{y}_{k+P} = \underbrace{\sum_{i=P-M+1}^{P} a_i \Delta m_{k+P-i}}_{future} + \underbrace{\sum_{i=P+1}^{H} a_i \Delta m_{k+P-i}}_{past}$$

In summary, at sample interval $n+j$:

$$\tilde{y}_{k+j} = \underbrace{\sum_{i=1}^{j} a_i \Delta m_{k+j-i}}_{\text{future adjustments}} + \underbrace{\sum_{i=j+1}^{H} a_i \Delta m_{k+j-i}}_{\text{past adjustments}} \qquad j \le M$$

$$\tilde{y}_{k+j} = \underbrace{\sum_{i=j-M+1}^{j} a_i \Delta m_{k+j-i}}_{\text{future adjustments}} + \underbrace{\sum_{i=j+1}^{H} a_i \Delta m_{k+j-i}}_{\text{past adjustments}} \qquad M < j \le P \qquad (11.8)$$

12-12 PROCESS CONTROL SYSTEM DESIGN - (c) Daniel R. Lewin Model Predictive Control

Matrix Form for Predictive Control

Thus, in matrix form, the predicted trajectory is:

$$\underbrace{\begin{bmatrix} \tilde{y}_{k+1} \\ \tilde{y}_{k+2} \\ \tilde{y}_{k+3} \\ \vdots \\ \tilde{y}_{k+M} \\ \vdots \\ \tilde{y}_{k+P} \end{bmatrix}}_{\underline{\tilde{y}}_k} = \underbrace{\begin{bmatrix} a_1 & 0 & 0 & \cdots & 0 \\ a_2 & a_1 & 0 & \cdots & 0 \\ a_3 & a_2 & a_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ a_M & a_{M-1} & a_{M-2} & \cdots & a_1 \\ \vdots & \vdots & \vdots & & \vdots \\ a_P & a_{P-1} & a_{P-2} & \cdots & a_{P-M+1} \end{bmatrix}}_{\underline{\underline{A}}} \underbrace{\begin{bmatrix} \Delta m_k \\ \Delta m_{k+1} \\ \Delta m_{k+2} \\ \vdots \\ \Delta m_{k+M-1} \end{bmatrix}}_{\underline{\Delta m}_k} + \underbrace{\begin{bmatrix} \tilde{y}^0_{k+1} \\ \tilde{y}^0_{k+2} \\ \tilde{y}^0_{k+3} \\ \vdots \\ \tilde{y}^0_{k+M} \\ \vdots \\ \tilde{y}^0_{k+P} \end{bmatrix}}_{\underline{\tilde{y}}^0_k} + \underbrace{\begin{bmatrix} w_k \\ w_{k+1} \\ w_{k+2} \\ \vdots \\ w_{k+M-1} \\ \vdots \\ w_{k+P-1} \end{bmatrix}}_{\underline{w}_k}$$

In the above, $\tilde{y}^0_{k+j}$ is the output predicted without further control action: $\tilde{y}^0_{k+j} = \sum_{i=j+1}^{H} a_i \Delta m_{k+j-i}$ and $\underline{w}_k$ is disturbance effect.

Dynamic Matrix Control

That, is, in matrix form: $\underline{\tilde{y}}_k = \underline{\underline{A}} \cdot \underline{\Delta m}_k + \underline{\tilde{y}}^0_k + \underline{w}_k$ (11.9)

In DMC, the effect of disturbances on the output at $k+1$ is considered to be modeling error, and is estimated by using the actual value of y measured at instant k:

$$\hat{w}_k = y_k - \tilde{y}_k \qquad (11.10)$$

Now, our objective is to obtain a desired trajectory, $\underline{\tilde{y}}^*_k$

Thus, we need to minimize the closed loop prediction error:

$$\underline{e}^*_k = \underline{\tilde{y}}^*_k - \underline{\underline{A}} \cdot \underline{\Delta m}_k - \underline{\tilde{y}}^0_k - \underline{\hat{w}}_k$$

Rearranging: $$\underbrace{\underline{e}^*_k}_{\text{Closed-loop prediction error}} = \underbrace{-\underline{\underline{A}} \cdot \underline{\Delta m}_k}_{\text{Vector of future control moves}} + \underbrace{\left(\underline{\tilde{y}}^*_k - \underline{\tilde{y}}^0_k - \underline{\hat{w}}_k\right)}_{\text{Open-loop prediction error}} \qquad (11.11)$$

Dynamic Matrix Control

Clearly, we would like to minimize the closed-loop prediction error, $\underline{e}_k^*$. This can be formally stated as follows:

$$\min_{\Delta \underline{m}_k} \phi = \left[\underline{e}_k^0 - \underline{\underline{A}} \cdot \Delta \underline{m}_k\right]^T \left[\underline{e}_k^0 - \underline{\underline{A}} \cdot \Delta \underline{m}_k\right] \qquad (11.12)$$

where $\underline{e}_k^0 = \underline{\tilde{y}}_k^* - \underline{\tilde{y}}_k^0 - \underline{\hat{w}}_k$ is the open-loop prediction error.

Differentiating Eq. (11.12) with respect to Δm_k, and equating to zero gives:

$$\frac{\partial \phi}{\partial \Delta \underline{m}_k} = -\underline{\underline{A}}^T \left[\underline{e}_k^0 - \underline{\underline{A}} \cdot \Delta \underline{m}_k\right] = 0$$

Solving gives: $$\Delta \underline{m}_k = \underbrace{\left(\underline{\underline{A}}^T \underline{\underline{A}}\right)^{-1} \underline{\underline{A}}^T}_{\text{A MxP matrix of controller gains}} \underline{e}_k^0 \qquad (11.13)$$

12-15 PROCESS CONTROL SYSTEM DESIGN - (c) Daniel R. Lewin Model Predictive Control

Dynamic Matrix Control

The generic DMC algorithm is implemented as follows:

❶ The process output, y_k, is measured, and used to estimate the process model error using Eq. (11.10).

❷ The open-loop prediction error, $\underline{e}_k^0 = \underline{\tilde{y}}_k^* - \underline{\tilde{y}}_k^0 - \underline{\hat{w}}_k$, is updated (accounting for changes in setpoint and effect of previous controller moves)

❸ Solve Eq. (11.13): $\Delta \underline{m}_k = \left(\underline{\underline{A}}^T \underline{\underline{A}}\right)^{-1} \underline{\underline{A}}^T \underline{e}_k^0$

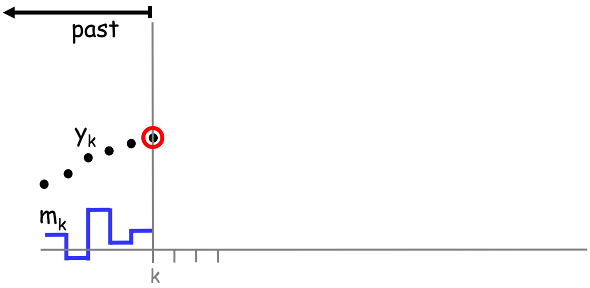
❹ Δm_k (first step only) is implemented.

❺ Counter is updated: $k = k + 1$.

12-16 PROCESS CONTROL SYSTEM DESIGN - (c) Daniel R. Lewin Model Predictive Control

MPC in Action

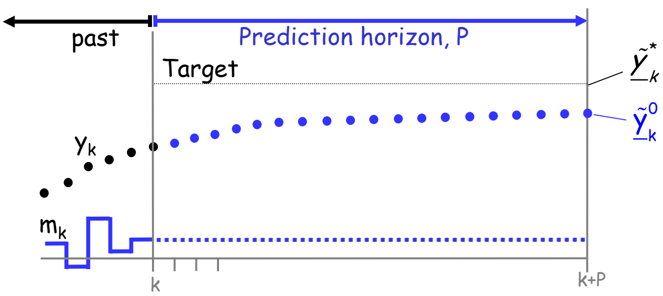
❶ The process output, $\underline{y}_k$, is measured, and used to estimate the process model error using Eq. (11.10).



12-17 PROCESS CONTROL SYSTEM DESIGN - (c) Daniel R. Lewin Model Predictive Control

MPC in Action

❷ The open-loop prediction error, $\underline{e}_k^0 = \underline{\tilde{y}}_k^* - \underline{\tilde{y}}_k^0 - \underline{\hat{w}}_k$, is updated (accounting for changes in setpoint and effect of previous controller moves)

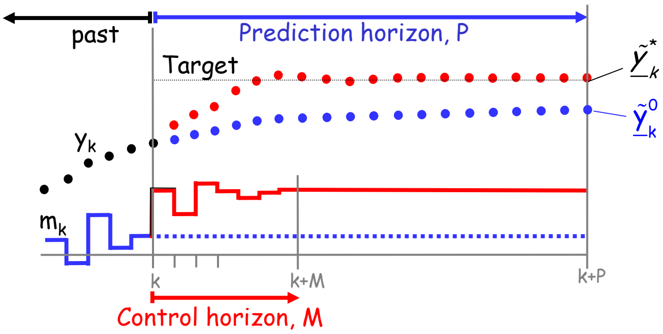


12-18 PROCESS CONTROL SYSTEM DESIGN - (c) Daniel R. Lewin Model Predictive Control

MPC in Action

❸ Solve Eq. (11.13): $\Delta \underline{m}_k = \left(\underline{\underline{A}}^T \underline{\underline{A}}\right)^{-1} \underline{\underline{A}}^T \underline{e}_k^0$

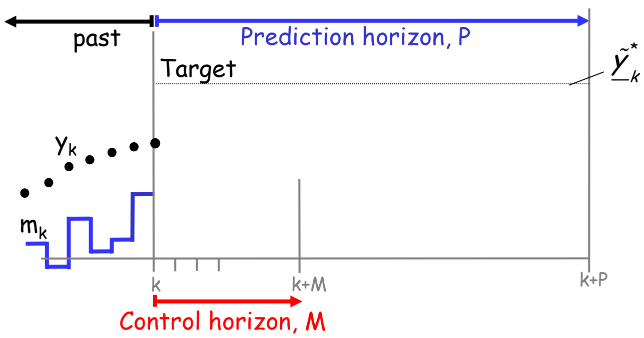
❹ Δm_k (first step only) is implemented.



12-19 PROCESS CONTROL SYSTEM DESIGN - (c) Daniel R. Lewin Model Predictive Control

MPC in Action

❺ Counter is updated: $k = k + 1$.



Problem. Control law can result in ringing in Δm_k. This occurs when $\underline{\underline{A}}^T \underline{\underline{A}}$ is ill-conditioned or singular.

12-20 PROCESS CONTROL SYSTEM DESIGN - (c) Daniel R. Lewin Model Predictive Control

Dynamic Matrix Control

Solution. Modify the objective function, to penalize for excessive control moves:

$$\phi_{MS} = \left[\underline{e}_k^0 - \underline{\underline{A}} \cdot \Delta \underline{m}_k\right]^T \left[\underline{e}_k^0 - \underline{\underline{A}} \cdot \Delta \underline{m}_k\right] + \Delta \underline{m}_k^T \underline{\underline{W}}_2 \Delta \underline{m}_k \quad (11.14)$$

The inclusion of the positive definite matrix, $\underline{\underline{W}}_2$, in the objective function is refereed to as "move suppression."

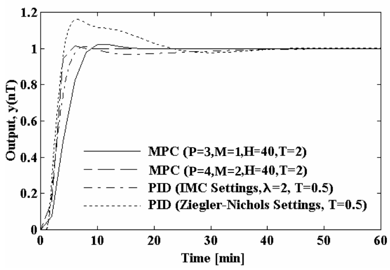
Minimizing with respect to Δm_k gives: $\Delta \underline{m}_k = \left(\underline{\underline{A}}^T \underline{\underline{A}} + \underline{\underline{W}}_2\right)^{-1} \underline{\underline{A}}^T \underline{e}_k^0$

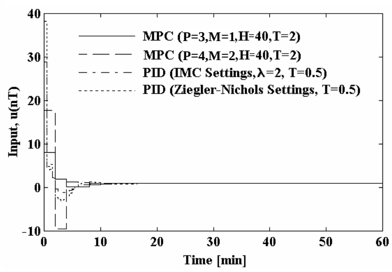
Example. Design MPC/DMC for the process: $p(s) = \frac{1}{(10s+1)(5s+1)} e^{-s}$

Use T = 2 min, and H = 40, and test: (a) M = 1, P = 3, and (b) M = 2, P = 4. Compare results with PID tuned by (c) ZN rule, and (d) IMC-PID method. Test the performance of the designs for unit step change in setpoint for nominal process model and for the case with 50% gain and delay uncertainty.

Comparing MPC with PID (SISO)

Nominal servo responses.

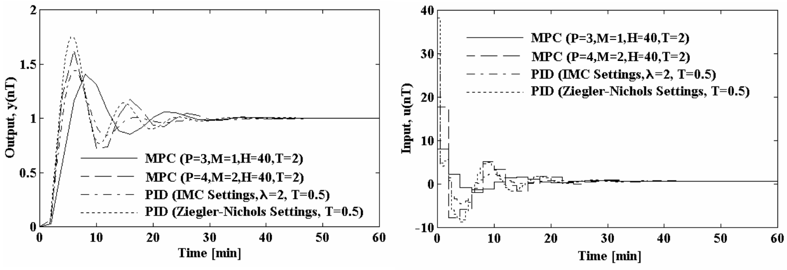




ZN tuning is: K_c = 4.2, τ_I = 6, τ_D = 4. IMC-PID tuning (with λ = 2) is: K_c = 3.75, τ_I = 15, τ_D = 3.33. Note that IMC-PID gives similar performance to MPC, superior to ZN (Is this surprising?)

Comparing MPC with PID (SISO)

Servo responses with 50% uncertainty in gain and delay.



Increasing M and P leads to more aggressive control action and less robustness.

MIMO MPC

The results obtained so far are easily extended to cover MIMO systems. Consider for example the 2x2 system described by the following discrete convolution model:

$$\tilde{y}_{1,k+1} = \sum_{i=1}^{H} a_{11,i}\Delta m_{1,k+1-i} + \sum_{i=1}^{H} a_{12,i}\Delta m_{2,k+1-i}$$

$$\tilde{y}_{2,k+1} = \sum_{i=1}^{H} a_{21,i}\Delta m_{1,k+1-i} + \sum_{i=1}^{H} a_{22,i}\Delta m_{2,k+1-i}$$

This model can be transformed into the standard dynamic form:

$$\underline{E}_k^* = -\underline{\underline{A}} \cdot \Delta\underline{M}_k + \underline{E}_k^0$$

where $\underline{E}_k^*$ and $\underline{E}_k^0$ are vectors of length 2P and $\Delta\underline{M}_k$ is a vector of length 2M:

$$\underline{E}_k^* = \left[e_{1,k+1}^*, e_{1,k+2}^*, \ldots, e_{1,k+P}^*, e_{2,k+1}^*, e_{2,k+2}^*, \ldots, e_{2,k+P}^*\right]^T$$

$$\Delta\underline{M}_k = \left[\Delta m_{1,k}, \Delta m_{1,k+1}, \ldots, \Delta m_{1,k+M-1}, \Delta m_{2,k}, \Delta m_{2,k+1}, \ldots, \Delta m_{2,k+M-1}\right]^T$$

MIMO MPC

In this 2x2 case, the matrix $\underline{\underline{A}}$ has the following structure:

$\underline{\underline{A}} = \begin{bmatrix} \underline{\underline{A}}_{11} & \underline{\underline{A}}_{12} \\ \underline{\underline{A}}_{21} & \underline{\underline{A}}_{22} \end{bmatrix}$ where each sub-matrix $\underline{\underline{A}}_{ij}$ has the structure:

$$\underline{\underline{A}}_{ij} = \begin{bmatrix} a_{ij,1} & 0 & 0 & \cdots & 0 \\ a_{ij,2} & a_{ij,1} & 0 & \cdots & 0 \\ a_{ij,3} & a_{ij,2} & a_{ij,1} & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ a_{ij,P} & a_{ij,P-1} & a_{ij,P-2} & \cdots & a_{ij,P-M+1} \end{bmatrix}$$

Even more general objective function is:

$$\phi = \underline{E}_k^{*T} \underline{\underline{W}}_1 \underline{E}_k^* + \Delta \underline{M}_k^T \underline{\underline{W}}_2 \Delta \underline{M}_k$$

Minimizing gives: $\Delta \underline{M}_k = \left(\underline{\underline{A}}^T \underline{\underline{W}}_1 \underline{\underline{A}} + \underline{\underline{W}}_2 \right)^{-1} \underline{\underline{A}}^T \underline{\underline{W}}_1 \underline{E}_k^0$

Comparing MPC with PID (MIMO)

Example. Consider the Wood and Berry distillation column:

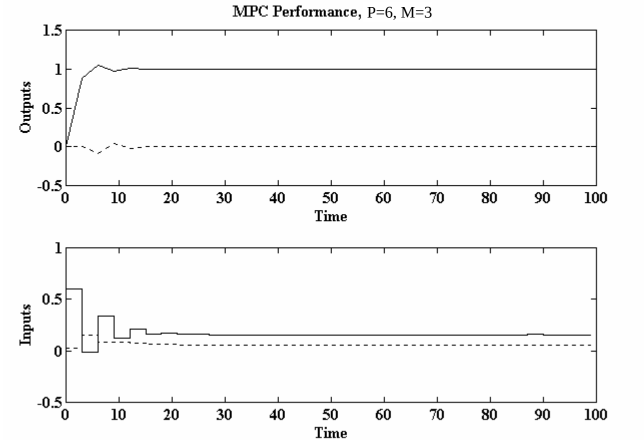
$$\begin{bmatrix} x_D \\ x_B \end{bmatrix} = \begin{bmatrix} \dfrac{12.8}{16.7s+1}e^{-s} & \dfrac{-18.9}{21s+1}e^{-3s} \\ \dfrac{6.6}{10.9s+1}e^{-7s} & \dfrac{-19.4}{14.4s+1}e^{-3s} \end{bmatrix} \begin{bmatrix} L \\ V \end{bmatrix}$$

We will compare two MIMO control systems for the process:
(a) an MPC design for H=30, P=6, M = 3, T=3, and $\underline{\underline{W}}_2 = 0$;
(b) a diagonal IMC-PID control system, tuned with $\underline{\lambda} = [6, 2.5]$.

Results. The following slides indicate that: (a) Nominally, MPC provides superior performance to decentralized PID, since it decouples the process interactions; (b) It is more sensitive to model uncertainties (50% in gain and delay) than PID.

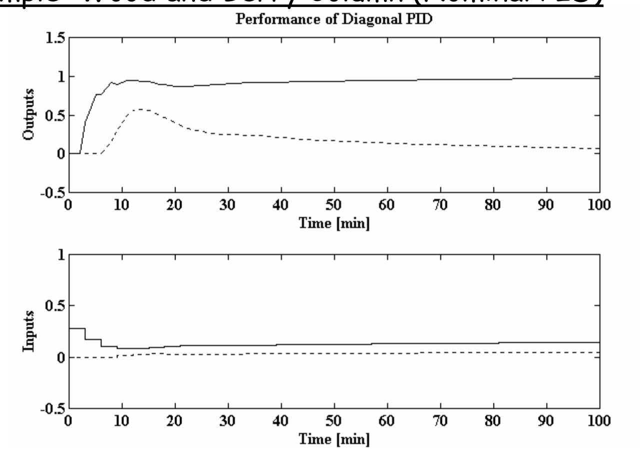
Comparing MPC with PID (MIMO)

Example: Wood and Berry Column (Nominal MPC)



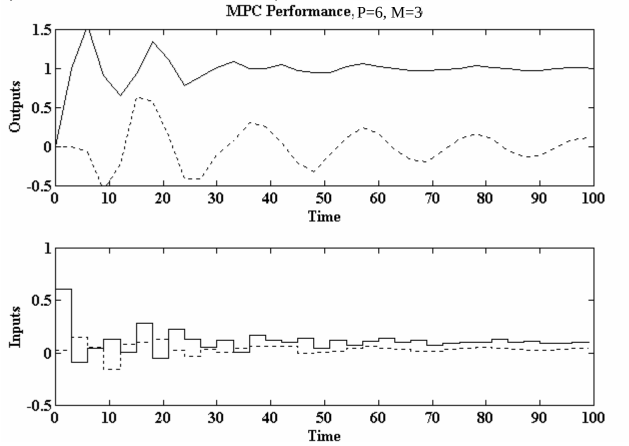
Comparing MPC with PID (MIMO)

Example: Wood and Berry Column (Nominal PID)



Comparing MPC with PID (MIMO)

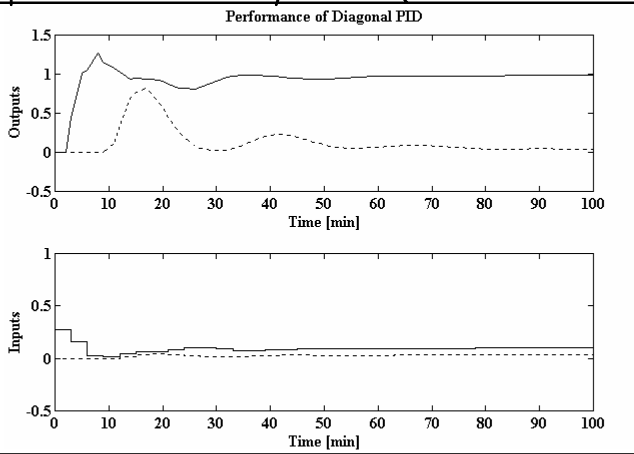
Example: Wood and Berry Column (MPC - Model Error)



12-29 PROCESS CONTROL SYSTEM DESIGN - (c) Daniel R. Lewin Model Predictive Control

Comparing MPC with PID (MIMO)

Example: Wood and Berry Column (PID - Model Error)



12-30 PROCESS CONTROL SYSTEM DESIGN - (c) Daniel R. Lewin Model Predictive Control

Tuning the MPC Controller

The following recommendations are often considered when selecting the tunable parameters in MPC:

❶ H - model horizon.
- HT should be selected so that it is greater than the open-loop settling time (defined as 99 or 95% of response). Commonly $20 \le H \le 70$.

❷ T – sampling period.
- Should be selected to capture important dynamic information.

❸ P and M - prediction and control horizons.
- Generally, P > M. Increasing M increases controller aggressiveness and decreases robustness. A suitable first guess $P \approx t_{60}/T$ (number of samples to reach 60% of open-loop response).

❹ W_1 and W_2 – weighting matrices.
- W_2, generally defined as a diagonal matrix, permits selectively penalizing particular manipulated variables for MIMO systems ("move suppression"). W_1 allows output scaling to be handled, and is referred to as "equal concern error" in DMC.

Process Constraints

In practice, the manipulated variables are constrained to physical, or allowable limits. Furthermore, the output variables may also be subject to constraints. Thus, in practice, a constrained optimization problem is solved:

$$\min_{\Delta \underline{M}_k} \phi = \underline{E}_k^{*T} \underline{\underline{W}}_1 \underline{E}_k^{*} + \Delta \underline{M}_k^T \underline{\underline{W}}_2 \Delta \underline{M}_k \quad (11.15)$$

subject to: $\Delta \underline{m}^L \le \Delta \underline{m}_k \le \Delta \underline{m}^H$ (move limits)

$\underline{m}^L \le \underline{m}_k \le \underline{m}^H$ (manipulated variable bounds)

$\underline{y}^L \le \underline{y}_k \le \underline{y}^H$ (output variable bounds)

This type of problem can be solved using quadratic programming. Nonlinear versions are solved using successive quadratic programming (SQP).

Why Model Predictive Control ?

Real processes are **multivariable** and often involve significant **interaction**. Here's a good analogy to convince your future boss to implement MPC.

Driving a car down Haifa's, Freud Street

- When driving a car, we make coordinated use of:
  - Steering wheel
  - Brake pedal
  - Acceleration pedal
  - Gear stick
- Consider how well you could maneuver a car at speed down Freud Street using only the steering wheel, with all other controls fixed? Equivalent to SISO control…

Summary

On completing this section, you should:

❶ Be able to explain to a layman how model predictive control (MPC) works.

- Recall that MPC uses a model in two ways: (a) to **predict** the effects of past moves; (b) to **optimize** future moves.

❷ Understand why MPC can improve on the performance that can be expected from decentralized controllers.

- Recalling that MPC: (a) implicitly handles constraints and thus maximizes DOF; (b) effectively decouples process interactions.

❸ Be able to correctly tune MPC to obtain a good response.

❹ Be able to implement MPC using MATLAB.