Part II

OVERVIEW OF INDUSTRIAL MPC TECHNIQUES

Contents

Chapter 1

INTRODUCTION TO MODEL PREDICTIVE CONTROL

1.1 BACKGROUND FOR MPC DEVELOPMENT

Two main driving forces for a new process control paradigm in the late 70's $\sim$ early 80's:

- Energy crisis + global competition + environmental reg.

  $\Downarrow$

  - process integration
  - reduced design / safety margin
  - real-time optimization
  - tighter quality control

  $\Downarrow$

  higher demand on process control.

- (Remarkable) advances in microprocessor technology.

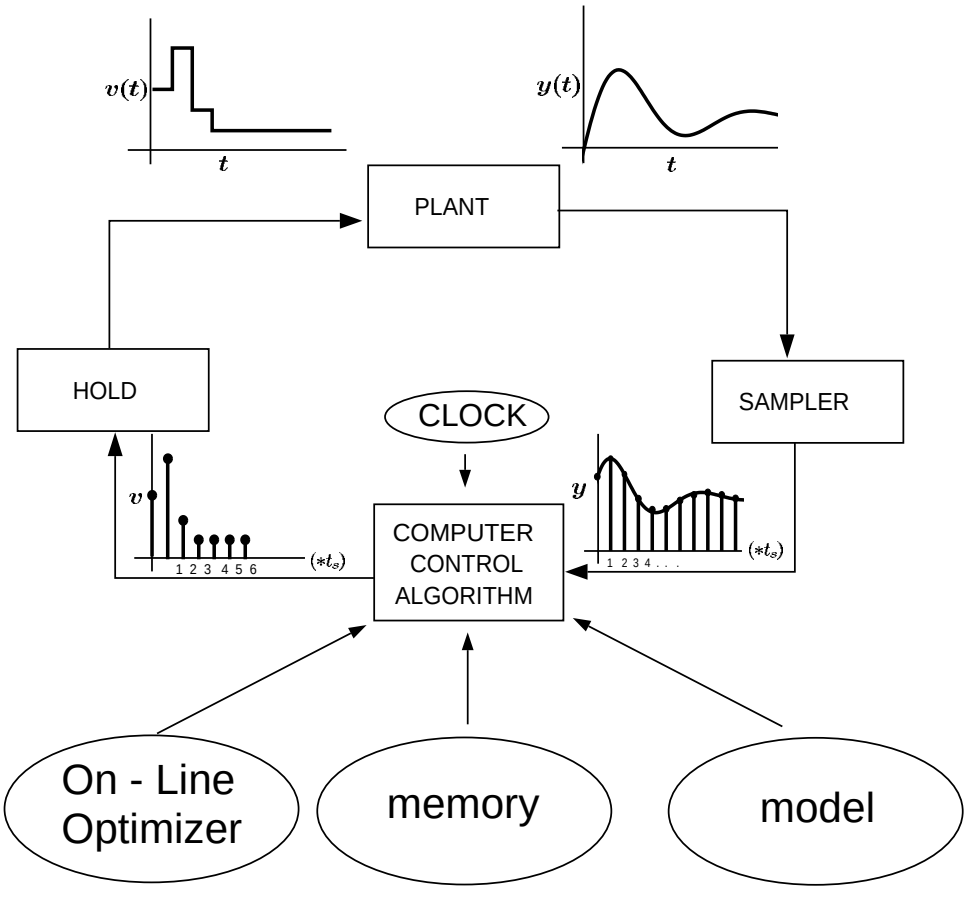
– cheap, fast and reliable medium for implementation.

– network environment(e.g., DCS) conductive to hierarchical approach.

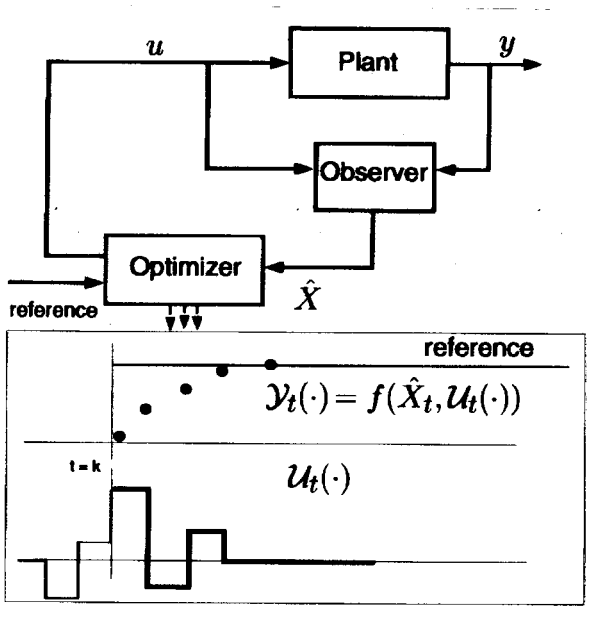
Industry's response ⇒ MPC

1.2 WHAT'S MPC

It's a computer control system.



It's a computer control system consisting of an observer & an optimizer.



$$\min_{\mathcal{U}_t(\cdot)} \int_t^{t+p} l_1[\text{Error}(\tau)] + l_2[\text{Input}(\tau)]d\tau$$

$$\mathcal{U}(\cdot) \in U, \quad \mathcal{Y}_t(\cdot) \in Y$$

The optimization is based on prediction of future behavior of y.

MPC(software packages) is sold under different names:

- DMC (Dynamic Matrix Control, now AspenTech)
- IDCOM (Setpoint, now AspenTech)
- SMCA (Setpoint, now AspenTech)
- RMPCT (Honeywell)
- PCT (Profimatics)

- HEICON (Adersa)
- OPC (Treiber)
- MAC
- IMC
- GPC
- GMC
- UPC

⋮

It's major features are

- model based
- *explicit* prediction of future system behavior
- *explicit* consideration of constraints
- use of on-line mathematical programming
- receding horizon control : repeated computation of open-loop optimal trajectory with feedback update ⇒ implicit *feedback* control.

1.3 WHY MPC?

Difficult elements for process control:

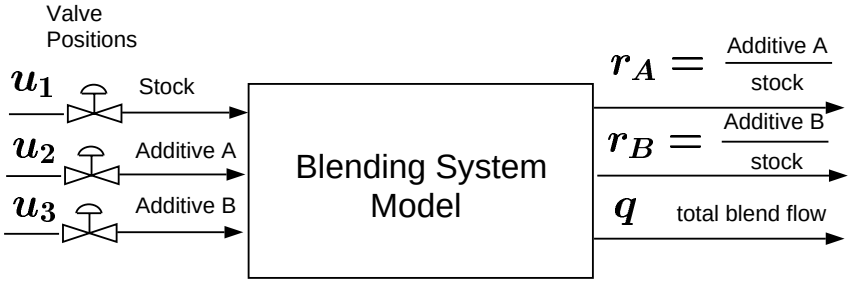
- delay, inverse response
- interaction

- constraints
- competing optimization requirements

MPC provides a systematic, unified solution to problems with these characteristics.

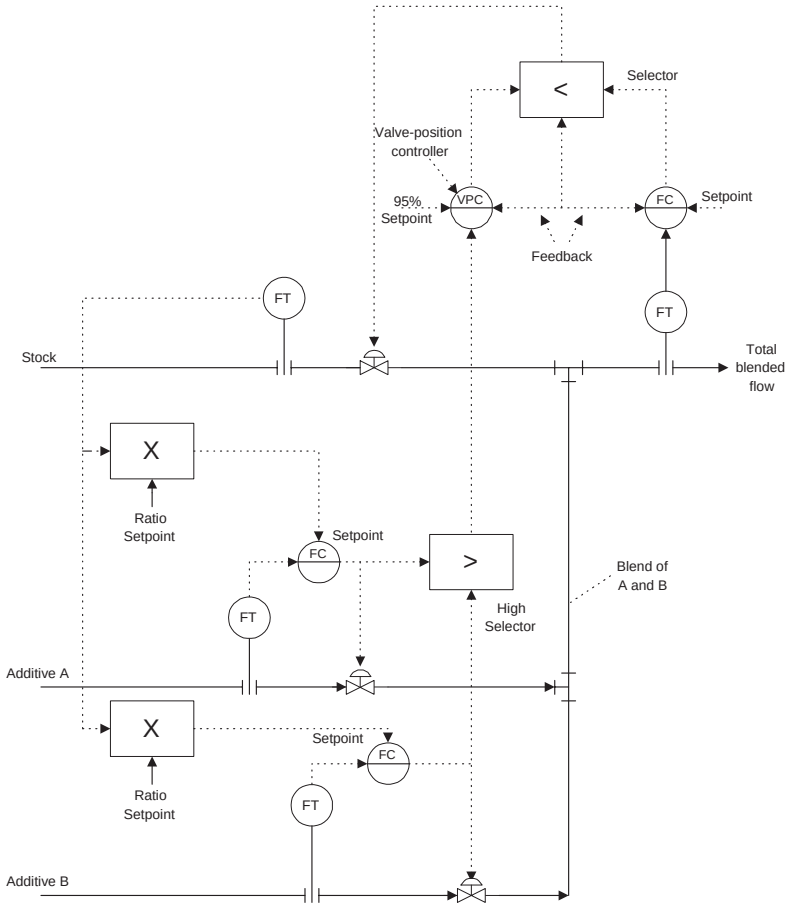
1.3.1 SOME EXAMPLES

Example I : Blending systems (input constraints)



- control r_A & r_B (first priority).
- control q if possible (second priority).
- possibility of valve saturation must be taken into account.

Classical Solution :



MPC Solution :

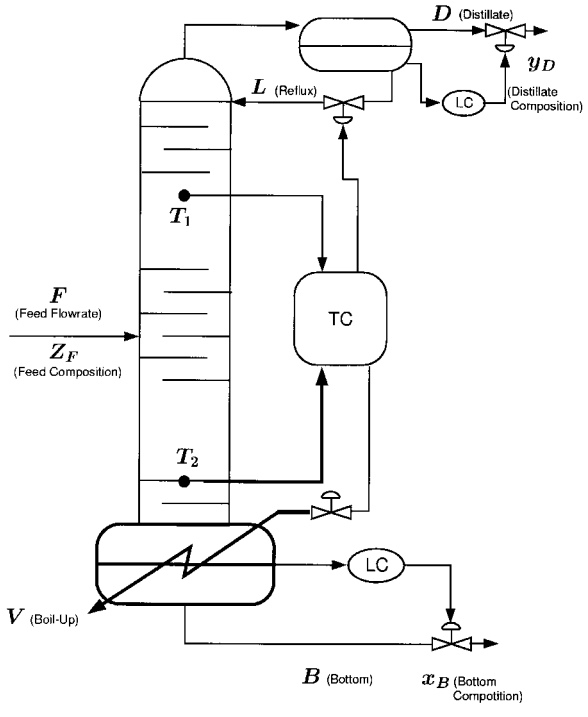
At t=k, solve

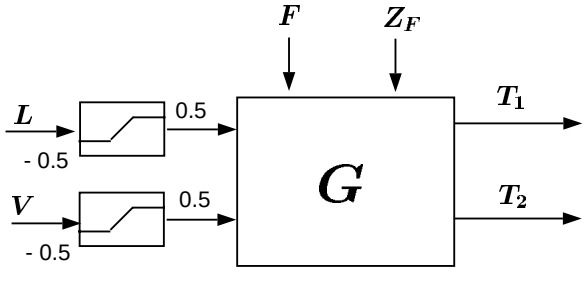
$$\min_{u_i} \quad \sum_{i=1}^{p} \left\| \begin{bmatrix} (r_A)_{k+i|k} \\ (r_B)_{k+i|k} \end{bmatrix} - \begin{bmatrix} (r_A)_{ref} \\ (r_B)_{ref} \end{bmatrix} \right\|_Q^2 + \|q_{k+i|k} - q_{ref}\|_R^2$$

$$Q \gg R$$

$$\begin{bmatrix} (u_1)_{min} \\ (u_2)_{min} \\ (u_3)_{min} \end{bmatrix} \leq \begin{bmatrix} (u_1)_j \\ (u_2)_j \\ (u_3)_j \end{bmatrix} \leq \begin{bmatrix} (u_1)_{max} \\ (u_3)_{max} \\ (u_2)_{max} \end{bmatrix}, \quad j = 0, \cdots, p-1$$

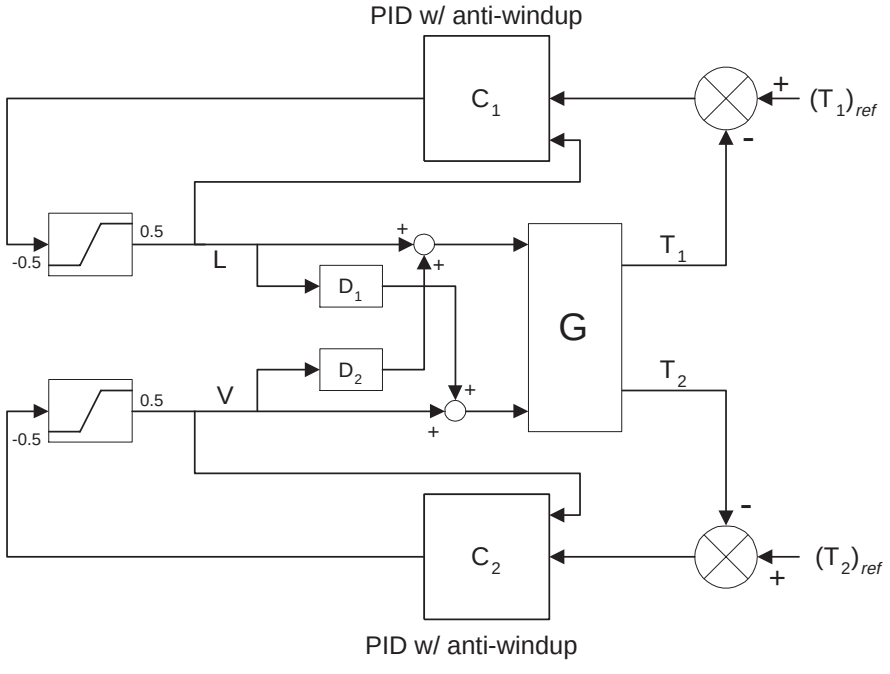
Example II : Two-point control in a distillation column (input constraints, interaction)





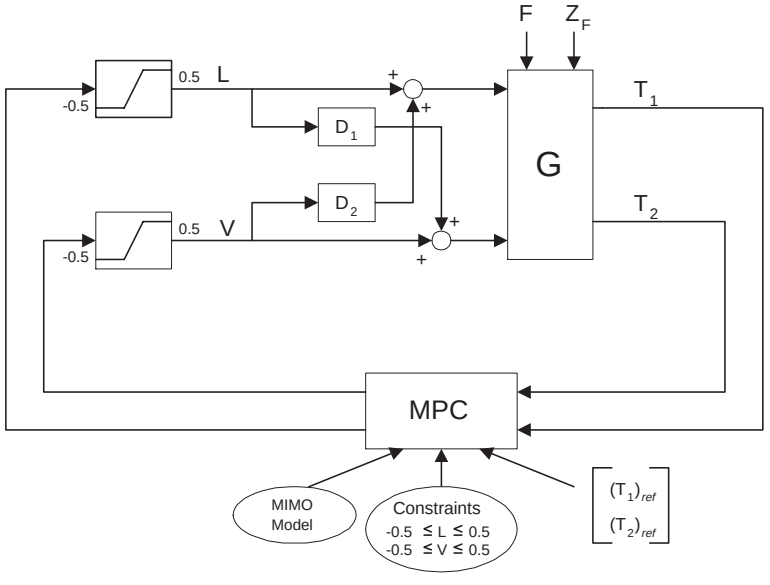
- strong interaction
- "wind-up" during saturation
- saturation of an input requires recoordination of the other input

Clasical Solution: Two single-loop controllers with anti-windup scheme (decouplers not shown)



- T_1 controller does not know that V has saturated and vice versa $\Rightarrow$ coordination of the other input during the saturation of one input is impossible.
- mode-switching logic is difficult to design / debug (can you do it?) and causes "bumps", etc.

MPC Solution:



At $t = k$, solve

$$\min_{\Delta \mathcal{U}_k} \sum_{i=1}^{p} \left\| \begin{bmatrix} (T_1)_{k+i|k} \\ (T_2)_{k+i|k} \end{bmatrix} - \begin{bmatrix} (T_1)_{\text{ref}} \\ (T_2)_{\text{ref}} \end{bmatrix} \right\|_Q^2 + \sum_{i=0}^{m-1} \left\| \begin{bmatrix} \Delta L_{k+i|k} \\ \Delta V_{k+i|k} \end{bmatrix} \right\|_R^2$$

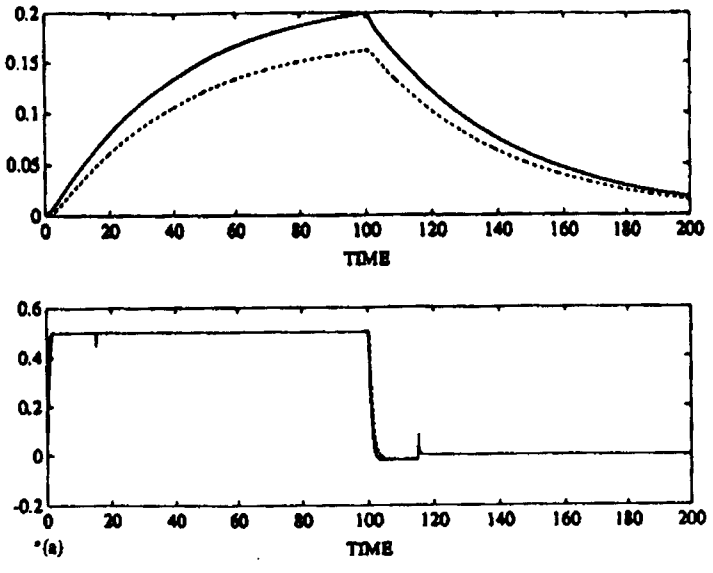
with

$$\begin{bmatrix} L_{\min} \\ V_{\min} \end{bmatrix} \le \begin{bmatrix} L_{k+i|k} \\ V_{k+i|k} \end{bmatrix} \le \begin{bmatrix} L_{\max} \\ V_{\max} \end{bmatrix} \qquad \text{for } i = 0, \cdots, m-1$$

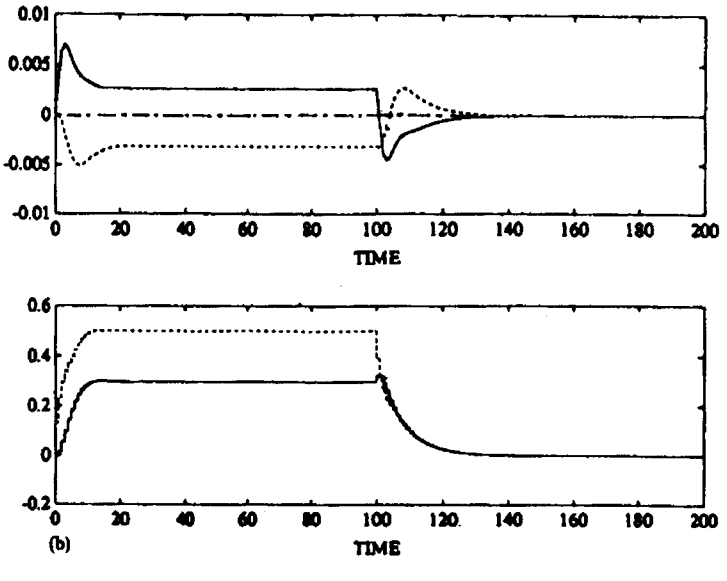
- easy to design / debug / reconfigure.
- anti-windup is automatic.
- optimal coordination of the inputs is automatic.

Performance of classical solution vs. MPC

SISO loops w/ anti-windup & decoupler (no mode switching):



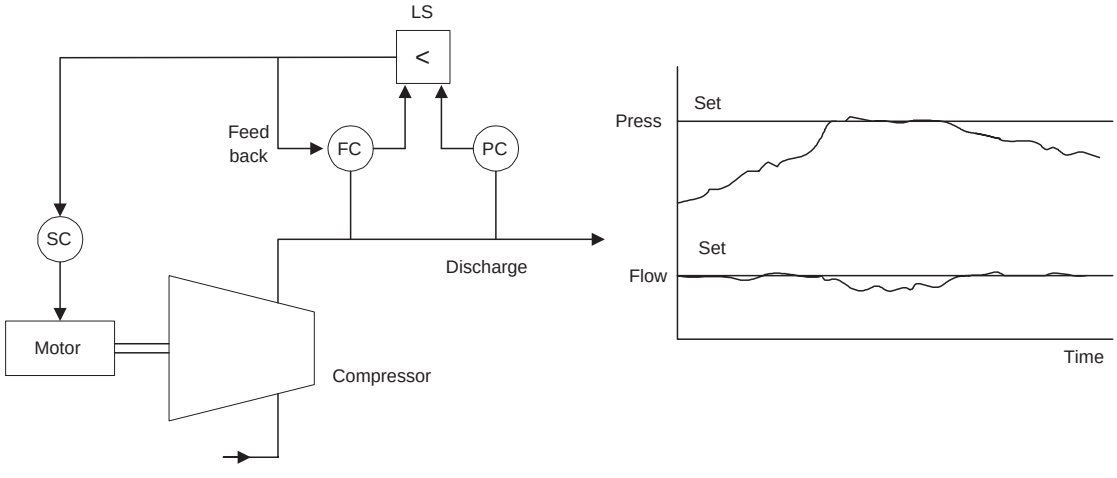
MPC:



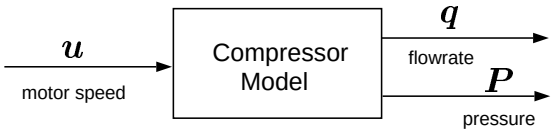
Example III : Override control in compressor(output constraint)

- control the flowrate
- but maintain $P \leq P_{max}$

Classical Solution :



MPC Solution:



At $t = k$, solve

$$\min_{\Delta \mathcal{U}_k} \sum_{i=1}^{p} \left\| q_{k+i|k} - q_{ref} \right\|_Q^2 + \sum_{i=0}^{m-1} \left\| \Delta u_{k+i|k} \right\|_R^2$$

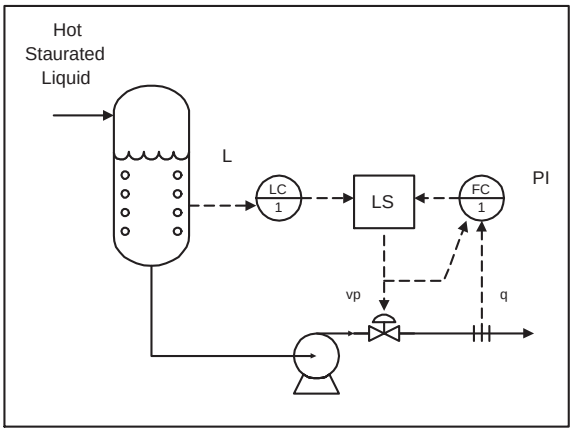
with

$$P_{k+i|k} \leq P_{max} \qquad \text{for } i = 1, \cdots, p$$

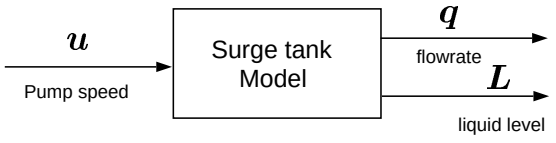
Example IV : Override control in surge tank(output constraints)

- control the outlet flowrate
- but maintain $L \geq L_{min}$

Classical Solution :



MPC Solution:



At $t = k$, solve

$$\min_{\Delta \mathcal{U}_k} \sum_{i=1}^{p} \left\| q_{k+i|k} - q_{ref} \right\|_Q^2 + \sum_{i=0}^{m-1} \left\| \Delta u_{k+i|k} \right\|_R^2$$

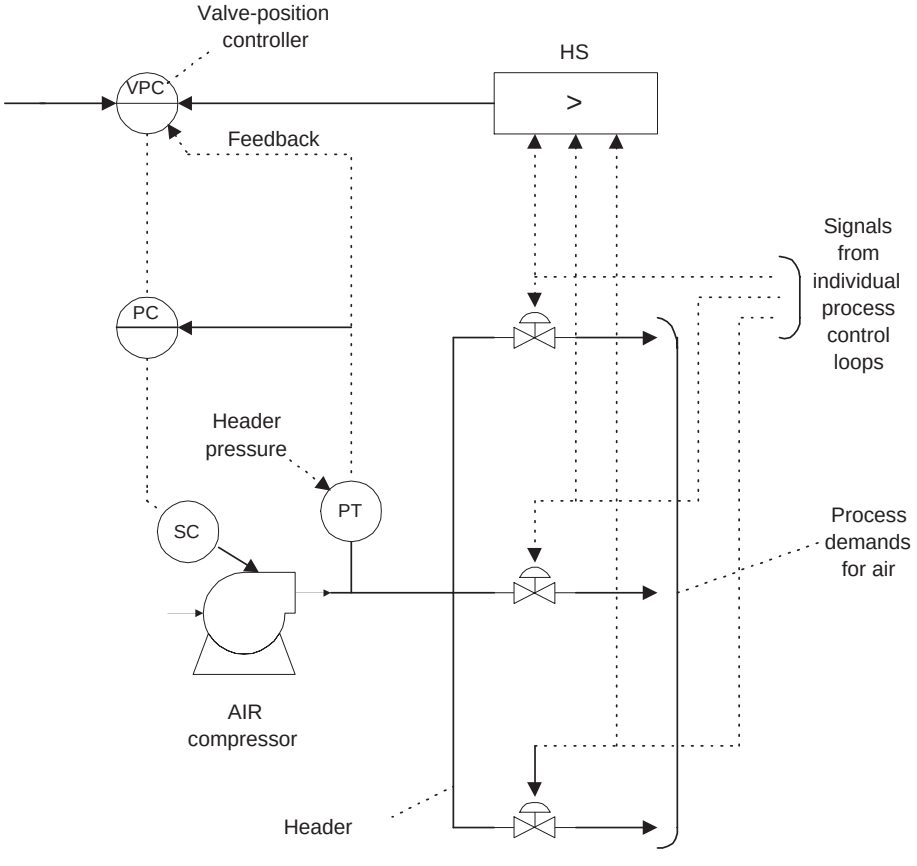
with

$$L_{k+i|k} \geq L_{min} \quad \text{for } i = 1, \cdots, p$$

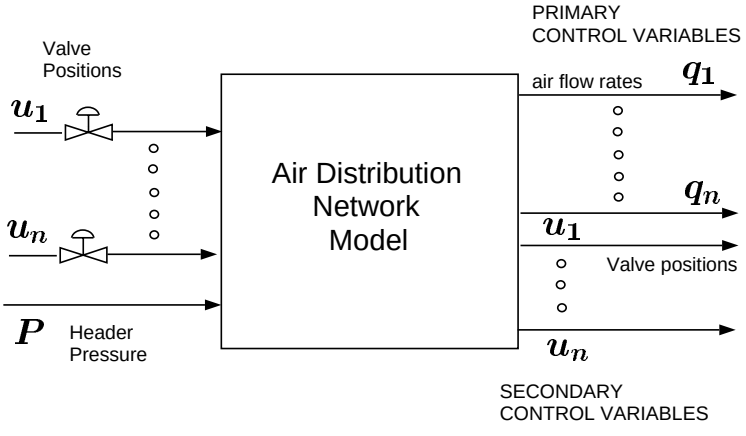
Example V : Valve position control in air distribution network (optimization requirement)

- control the flowrates of individual channels
- minimize the air compression

Classical Solution :



MPC Solution :



At $t = k$, solve

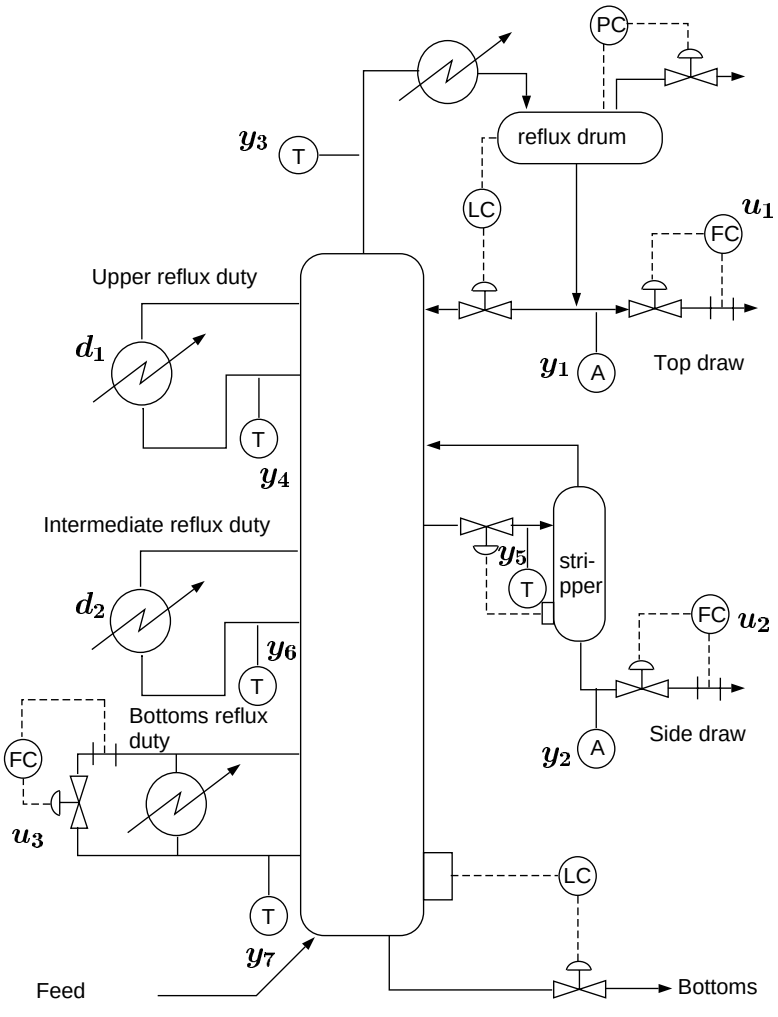
$$\min_{\Delta \mathcal{U}_k} \sum_{i=1}^{p} \left\| \begin{bmatrix} (q_1)_{k+i|k} \\ \vdots \\ (q_n)_{k+i|k} \end{bmatrix} - \begin{bmatrix} (q_1)_{ref} \\ \vdots \\ (q_n)_{ref} \end{bmatrix} \right\|_Q^2 + \sum_{i=1}^{m-1} \left\| P_{k+i|k} - P_{min} \right\|_R^2$$

with $Q \gg R$ and

$$\begin{bmatrix} P_{min} \\ (u_1)_{min} \\ \vdots \\ (u_n)_{min} \end{bmatrix} \leq \begin{bmatrix} P_{k+i|k} \\ (u_1)_{k+i|k} \\ \vdots \\ (u_n)_{k+i|k} \end{bmatrix} \leq \begin{bmatrix} P_{max} \\ (u_1)_{max} \\ \vdots \\ (u_n)_{max} \end{bmatrix} \qquad \text{for } i = 0, \cdots, m-1$$

Example VI : Heavy oil fractionator (all of the above)

- y_7 must be kept above T_{min}.
- y_1 and y_2 is to be kept at setpoint(measurements delayed).
- BRD must be minimized to maximize the heat recovery.

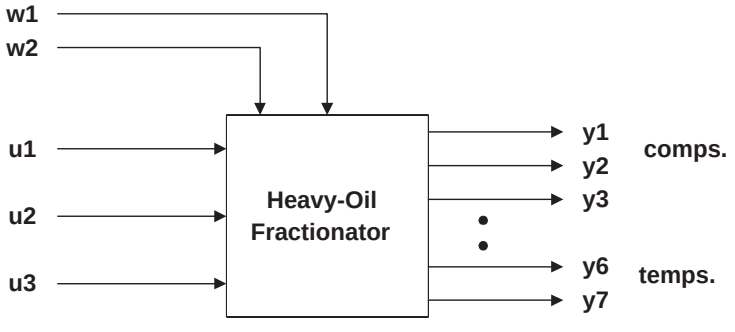


Classical Solution:

Not clear

- how to use temperature measurements to fight the effect of delays, unreliability, etc. of analyzers.
- how to accommodate the optimization requirement.

MPC Solution :

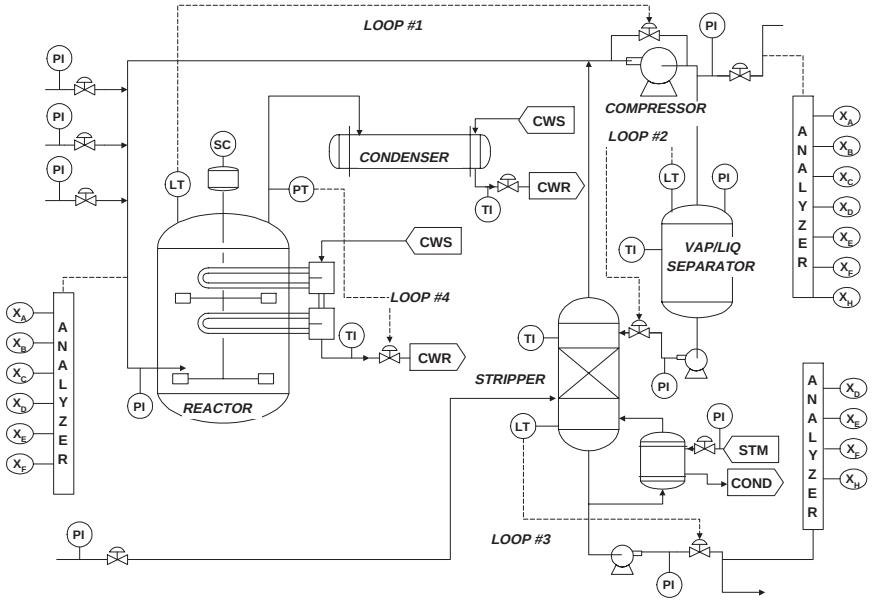


$$\min_{\Delta \mathcal{U}_k} \sum_{l=1}^{p} \left\| \begin{bmatrix} y_1 \\ y_2 \\ u_3 \end{bmatrix}_{k+l|k} - \begin{bmatrix} y_1 \\ y_2 \\ u_3 \end{bmatrix}_{ref} \right\|_Q^2 + \sum_{i=1}^{m} \left\| \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ \Delta u_3 \end{bmatrix}_{k+i|k} \right\|_R^2$$

$$y_7 \geq T_{min}$$

plus other input constraints.

Example VII : Tennessee Eastman process(supervisory control requirements)



Tier	Loop #	Controlled Variables	Manipulated Variables
I	1	Reactor Level	Compressor recycle valve
	2	Separator Level	Separator liquid flow
	3	Stripper Level	Stripper liquid flow
	4	Reactor Pressure	Reactor cooling water flow

$$\min_{\Delta \mathcal{U}_k} \sum_{l=1}^{p} \left\| \begin{bmatrix} Q \\ G/H \end{bmatrix}_{k+l|k} - \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}_{k+l|k} \right\|_Q^2 + \sum_{i=0}^{m-1} \left\| \Delta u_{k+i|k} \right\|_R^2$$

$$P_r \leq (P_r)_{max}$$

$$(H_r)_{min} \leq H_r \leq (H_r)_{max}$$

where
P_r: reactor pressure, $(P_r)_s$: setpoint to reactor pressure loop
H_r: reactor level, $(H_r)_s$: setpoint to reactor level loop
Q: total product flow G/H: mass ratio between products G and H
F_D: D feed flow F_E: E feed flow

1.3.2 SUMMARY

Advantages of MPC over Traditional APC

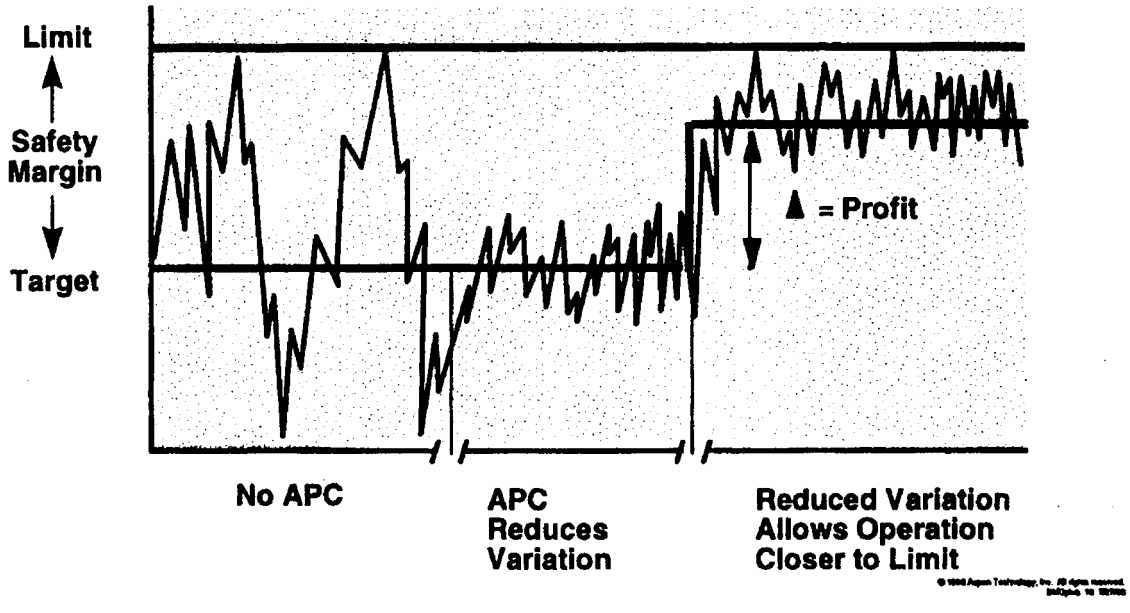
- control of processes with complex dynamics
- decoupling and feedforward control are "built in" (traditional approaches are difficult for systems larger than 2×2).
- constraint handling
- utilizing degrees of freedom
- consistent methodology
- realized benefits: higher on-line times and cheaper implementation / maintenance

1.4 INDUSTRIAL USE OF MPC: OVERVIEW

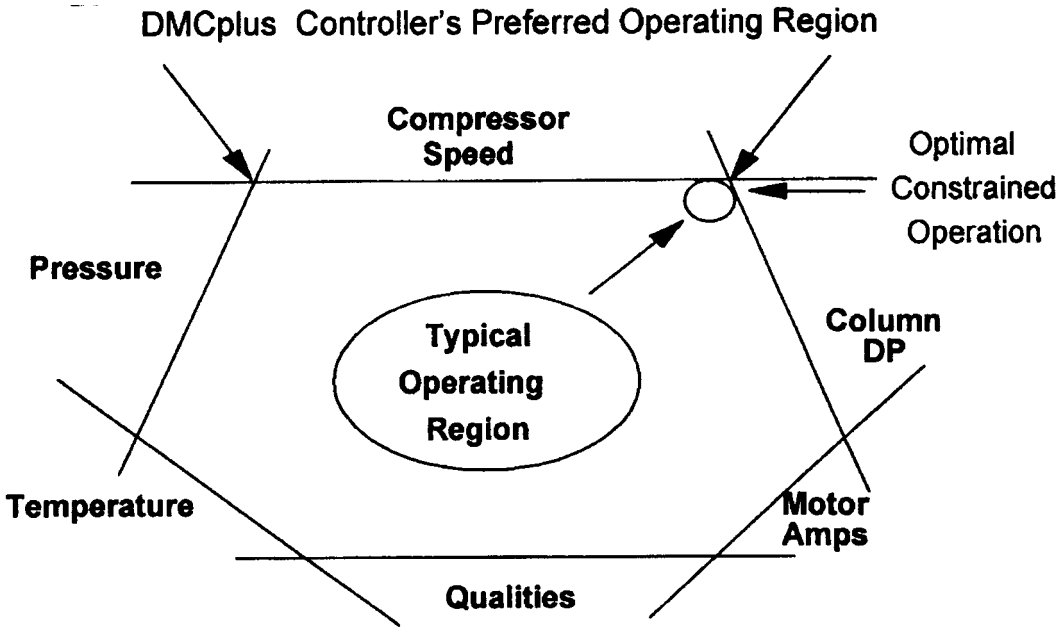
1.4.1 MOTIVATION

Profitability potential with multivariable control

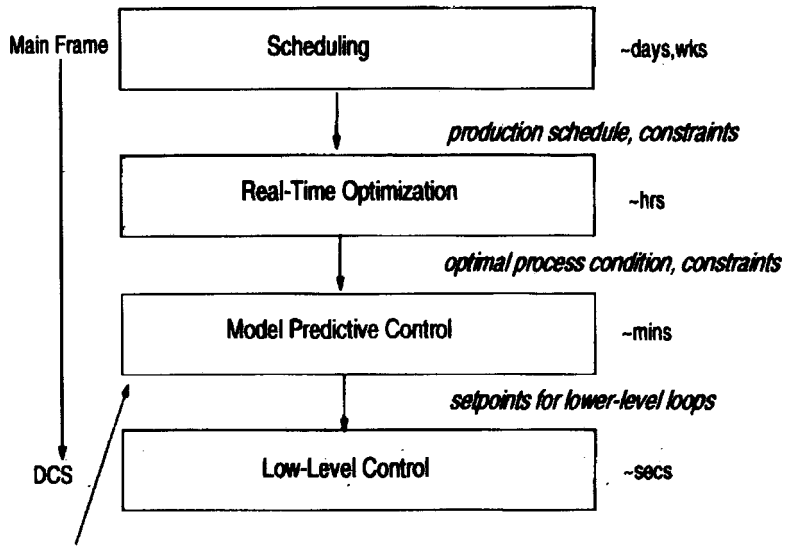
- reduce variability of key variables by 50 % or more.
- increase yield of more valuable products,



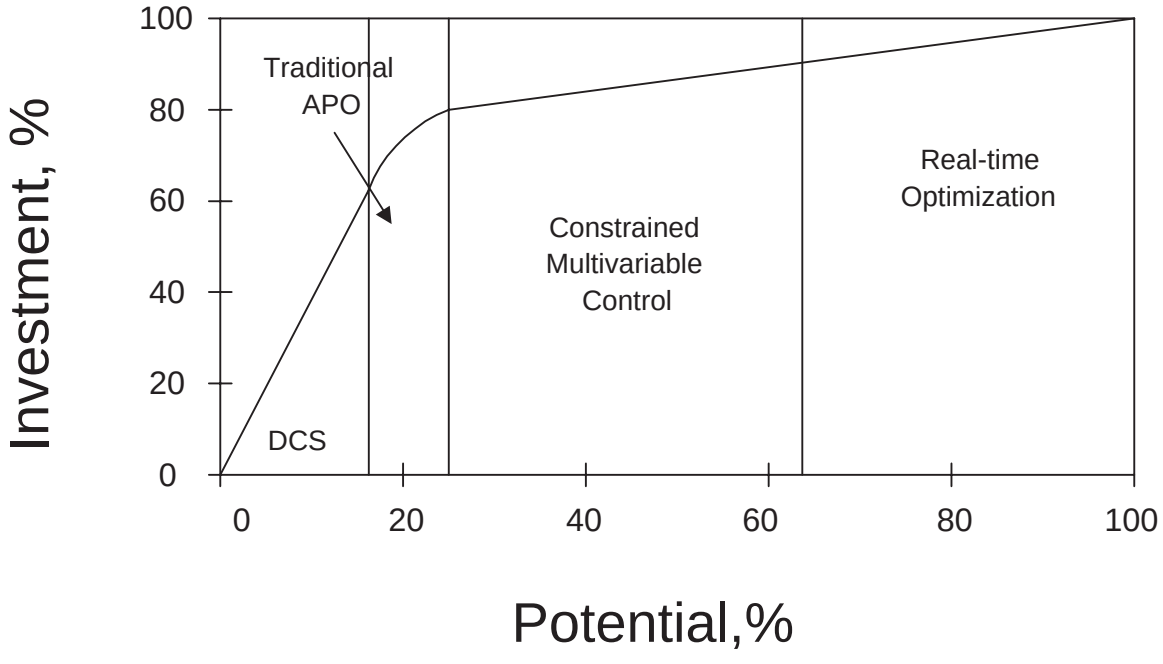
Benefits of control and optimization



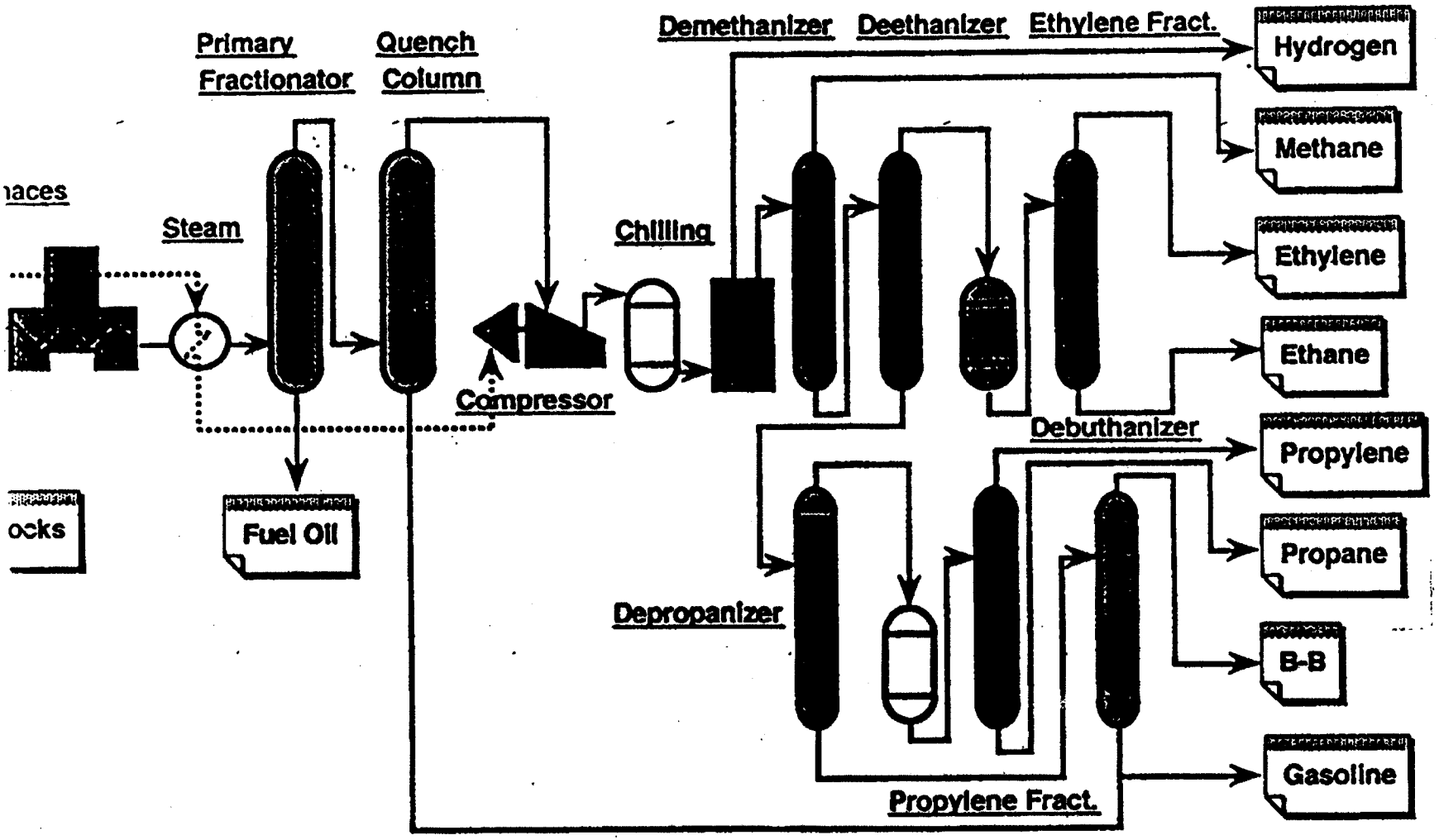
MPC within plant automation hierarchy



Move the plant to the current optimal condition fast and smoothly w/o violating constraints


Investment, %
100
80
60
40
20
0
Traditional
APO
DCS
Constrained
Multivariable
Control
Real-time
Optimization
0
20
40
60
80
100
Potential,%

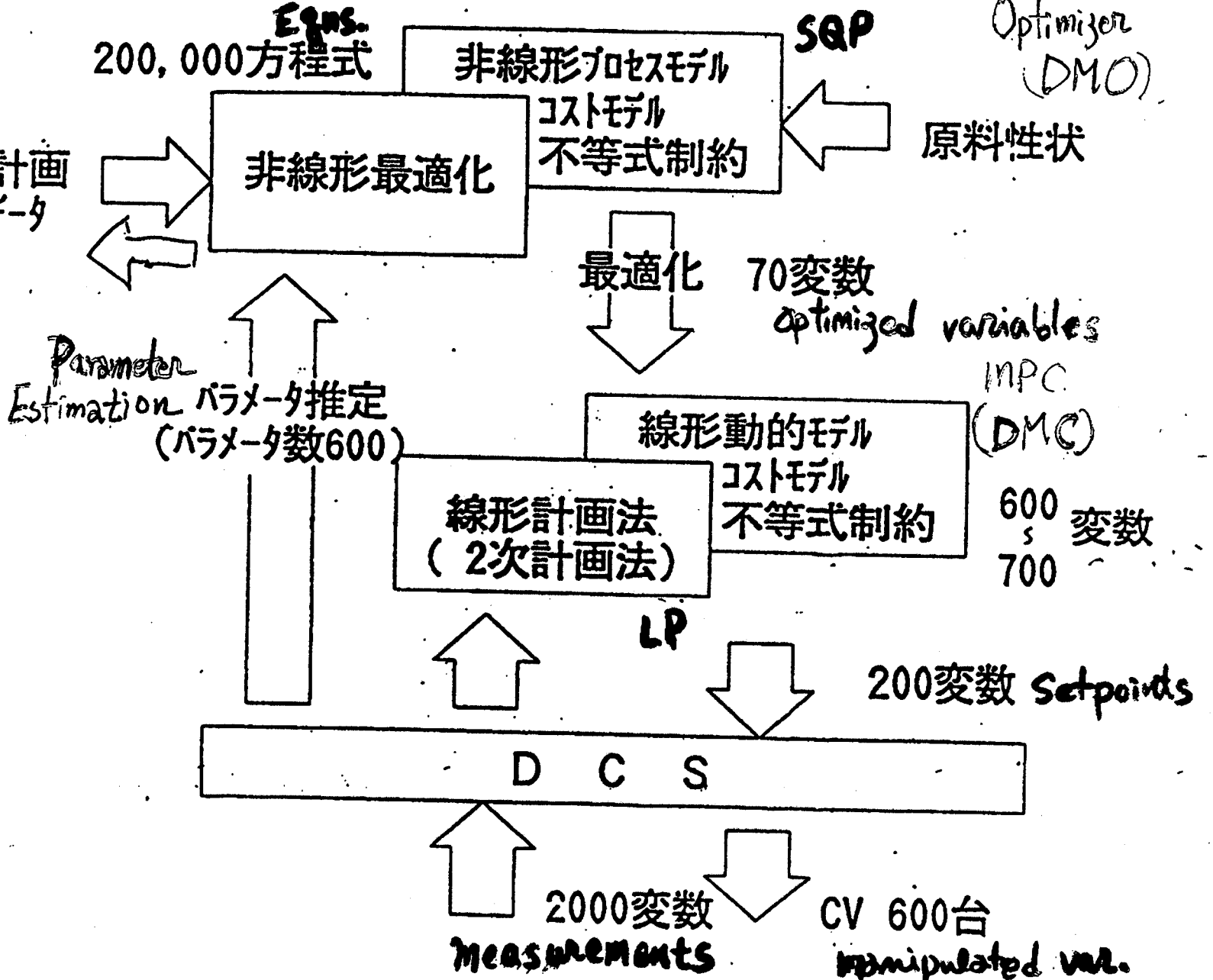
Block Diagram of Olefins Plant



Ethylene Plant

DEC/ALPHA

プロセスオリエンティッドな制御システム例



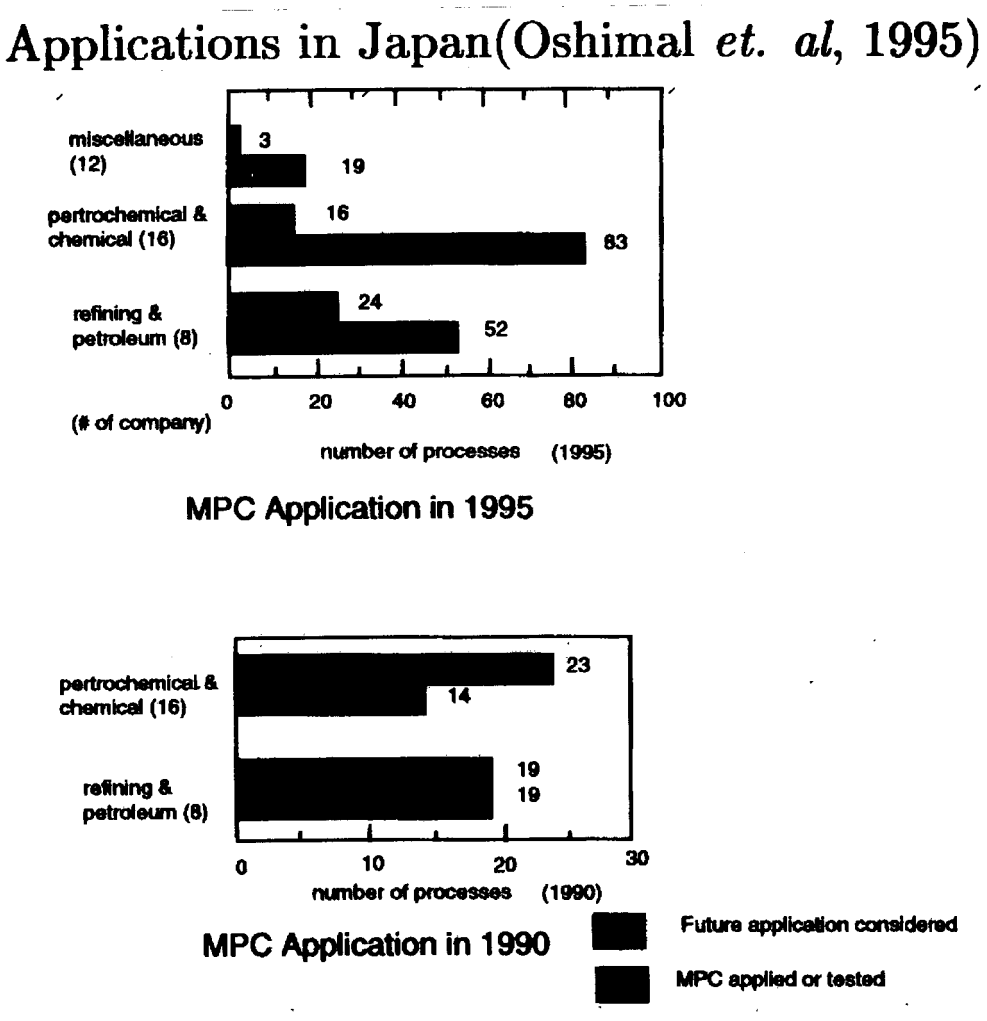
1.4.2 SURVEY OF MPC USE

Current status of MPC application(North America/Europe)

Applications by five major MPC vendors (Badgewell, 1996)

Area	DMC Coop.	Setpoint Inc.	Honeywell Profimatics	Adersa	Treiber Controls	Total
Refining	360	320	290	280	250	1500
Petrochemicals	210	40	40	-	-	290
Chemicals	10	20	10	3	150	193
Pulp and Paper	10	-	30	-	5	45
Gas	-	-	5	-	-	5
Utility	-	-	2	-	-	2
Air Separation	-	-	-	-	5	5
Mining/Metallurgy	-	2	-	7	6	15
Food Processing	-	-	-	41	-	41
Furnaces	-	-	-	42	-	42
Aerospace/Defence	-	-	-	13	-	13
Automotive	-	-	-	7	-	7
Other	10	20	-	45	-	75
Total	600	402	377	438	416	2233
First App	DMC:1985	IDCOM-M:1987 SMCA:1993	PCT:1984 RMPCT:1991	IDCOM:1973 HIECON:1986	OPC:1987	
Largest App	603×283	35×28	28 ×20	-	24×19	

Current status of MPC application(Japan):



1.5 HISTORICAL PERSPECTIVE

- The idea of using a model for prediction and optimal control computation has been around for long time.

Note that research on optimal control was most vigorous in the 50s and 60s. Most of the results during this period were for open-loop optimal control. For feedback implementation, they hinted the idea of *receding horizon control*. However, most of the results were impractical due to the lack of implementation hardware.

Some remarkable results of this period include

- Pontryagin's maximum principle.
- Hamilton-Jacobi-Bellman equation for optimal feedback control.
- Feldbaum's dual control concept.

- Due to the lack of sophisticated hardware, it was highly desirable to derive a closed-form control law that could be implemented with computational equipments available at reasonable costs. The work of Kalman represents a major achievement in this regard.

  Kalman derived analytical solutions to

  - linear quadratic optimal control problem for deterministic systems $\Rightarrow$ (∞-horizon) LQR
  - the same problem for Gaussian stochastic systems $\Rightarrow$ LQG = LQR + Kalman filter

  These solutions were important because they represented very few *analytial* solutions to optimal feedback control problems.

  However his work (based on the idea of stage-wise solution using dynamic programming) could not be extended to constrained systems or nonlinear systems.

- In the 70's, Kwon and Pearson discussed the idea of *receding horizon control* (a cornerstone of MPC) in the context of LQ optimal control and how to achieve stability with such a control law.

  However, they did not consider constrained / nonlinear problems and failed to motivate the need for such an approximation.

- In the late 70s and early 80s, there were several reports of successful use of optimal control concepts in oil industries. For instance, Charlie Cutler reported the success of implementing the so-called Dynamic Matrix Control in Shell Oil refining units.

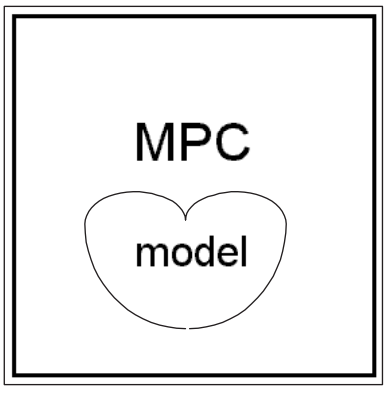
  This started an avalanch of similar algorithms and industrial projects. From here on, process control would never be the same.

- The industrial success renewed the academics' enthusiasm for optimal control. Prototypical algorithms were analyzed and improved. Also, connections to the Kalman's work and other optimal control theories were brought forth.

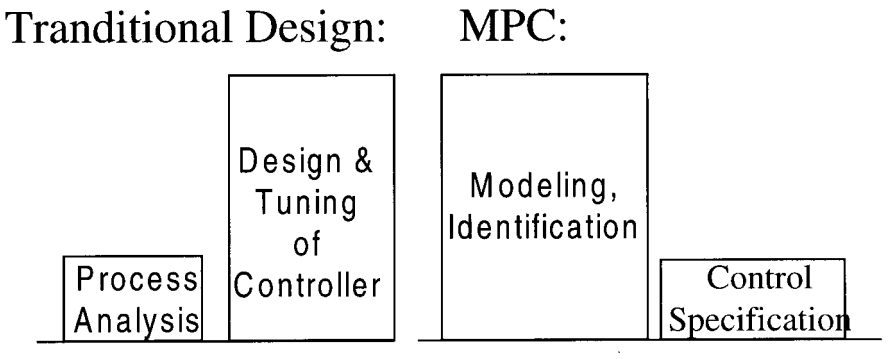
- Now, MPC is an essential tool-of-trade for process control engineers. There are more than a dozen vendors offering commercial software packages and engineering services. There is probably not a single major oil industry where MPC is not employed in its new instaillations or revamps. MPC is also very well understood from a theoretical standpoint.

1.6 CHALLENGES

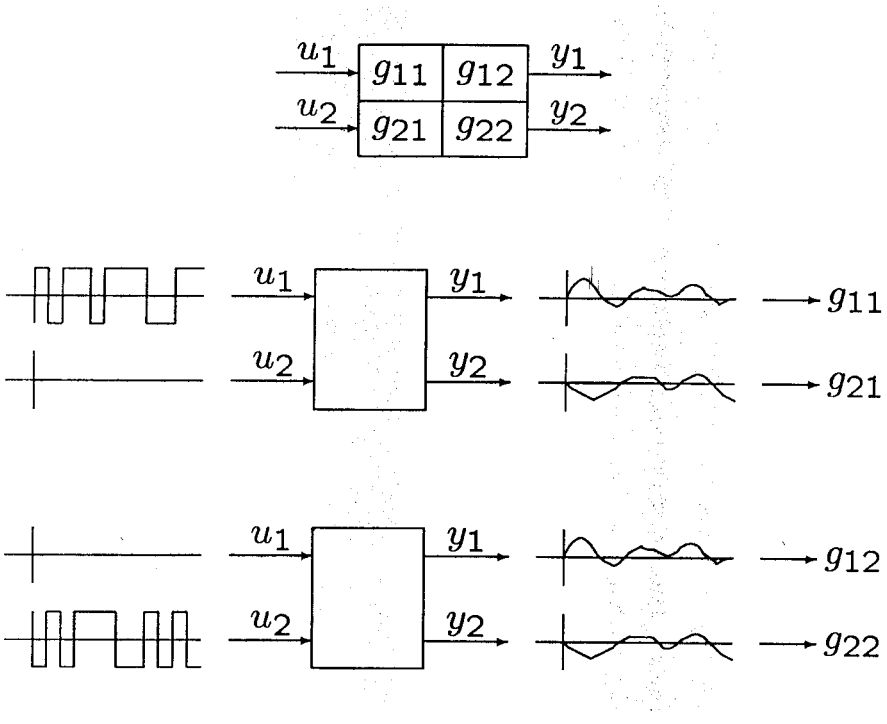
1.6.1 MODELING & IDENTIFICATION



- most models for MPC design came from identification rater than fundamental modeling.
- system ID takes up to 80-90% of the cost and time in a typical implementation of a model based controller.



Current Practice



- Example illustrating the difficulty in multivariable system identification

True plant(Model A)

$$\frac{1}{10s+1}\begin{bmatrix} 0.878 & -0.864 \\ 1.086 & -1.092 \end{bmatrix}$$

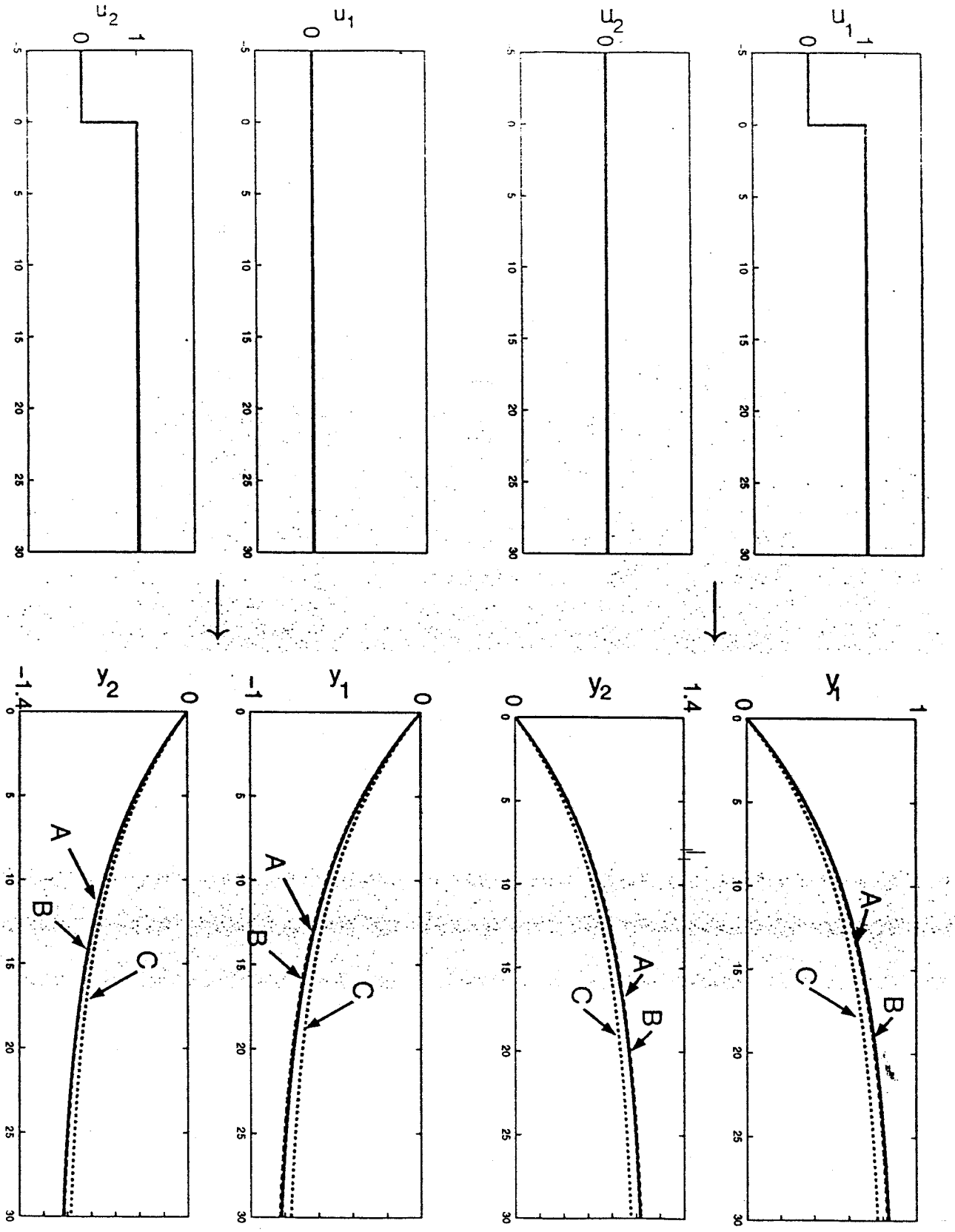
Model B

$$\frac{1}{10s+1}\begin{bmatrix} 0.867 & -0.875 \\ 1.095 & -1.083 \end{bmatrix}$$

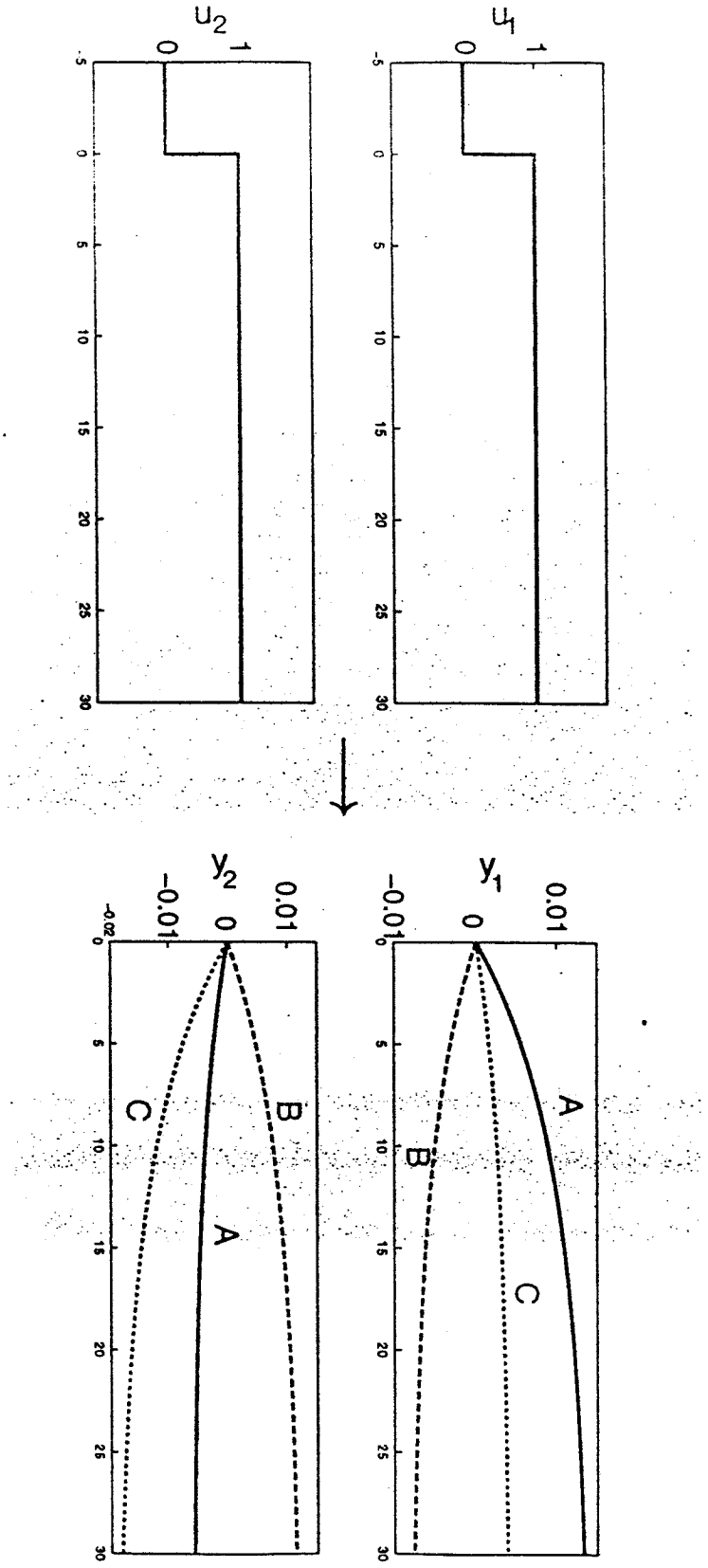
Model C

$$\frac{1}{10s+1}\begin{bmatrix} 0.808 & -0.804 \\ 1.006 & -1.025 \end{bmatrix}$$

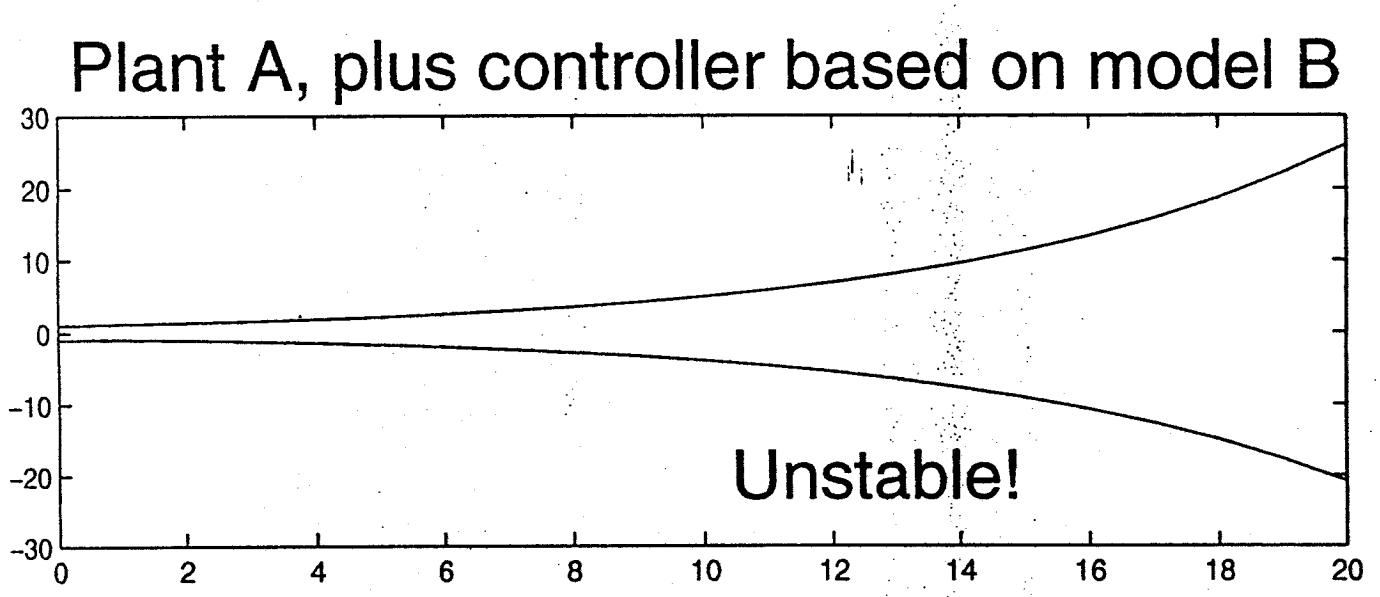
Open-loop response fits

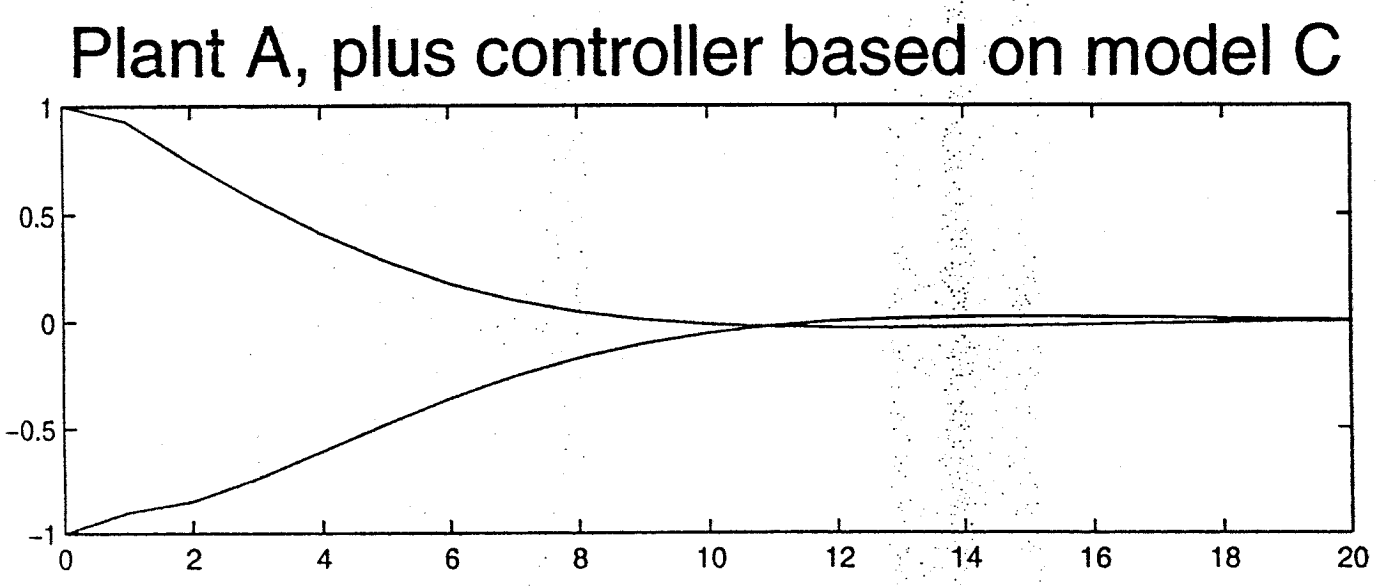


Open-loop step response fits with simultaneous input change.



Close-loop responses

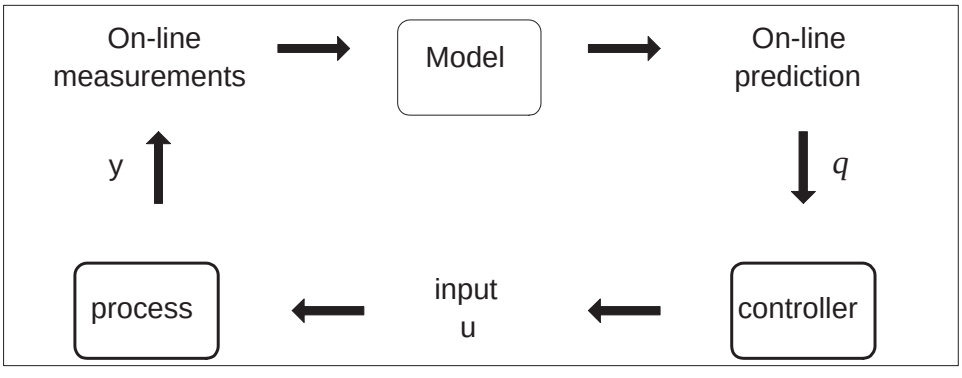




1.6.2 INCORPORATION OF STATISTICAL CONCEPTS

- *Improved Disturbance Rejection:*
  One can capture the temporal / spatial correlations of disturbance effects in the form of statistical models, using historical data, and use them for better prediction / control.
- *Inferential Control of Composition / Quality Variables*
  Many quality variables (e.q., variables directly related to end-use property) and compositions are not on-line measurable or difficult to measure on-line. Delays and large sample time involved in the laboratory analysis can make tight control impossible. In this case, correlations with other more easily measurable variables can be captured and utilized for inferencing.
- *Control System Performance Monitoring / Failure Diagnosis*
  The concept of Statistical Process Control (SPC) can be used to detect unusual behavior and also diagnose the cause of performance deterioration of MPC.

Motivational Example: Batch Pulp Digester

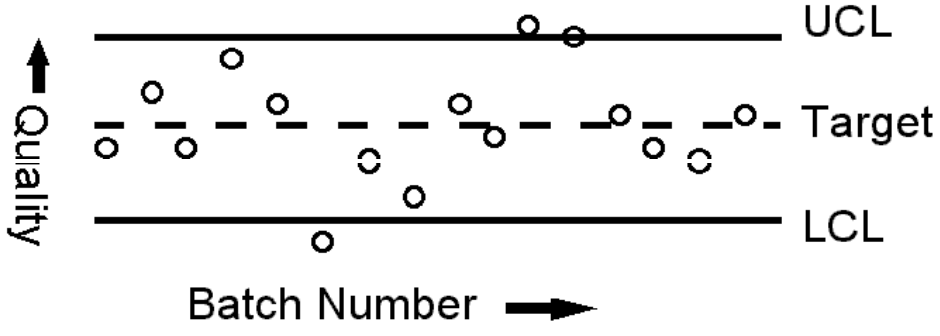


Key Aspects of the Problem

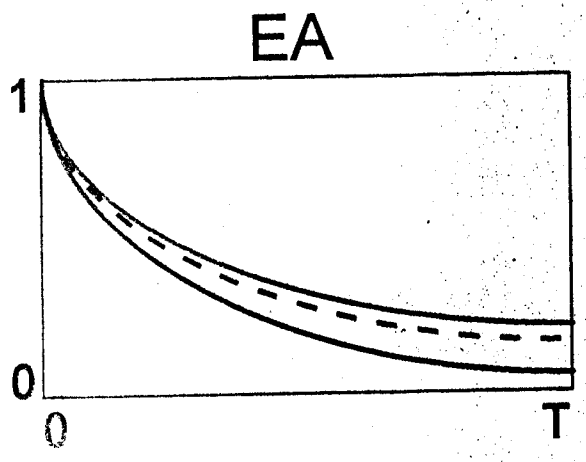
- Frequent Batch-to-Batch Variations in Operating Conditions
  - Feed conditions
  - Process parameters
    - → heat transfer coefficients
    - → reaction rate parameters
- Lack of On-Line Quality Measurements
  - Prohibits real-time feedback control

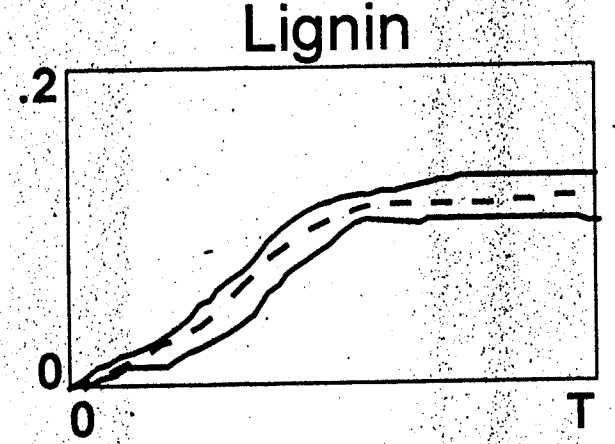
Traditional Control Method: Statistical Quality Control

- SPC Based on On-Line Measurements



Batch Digester Example



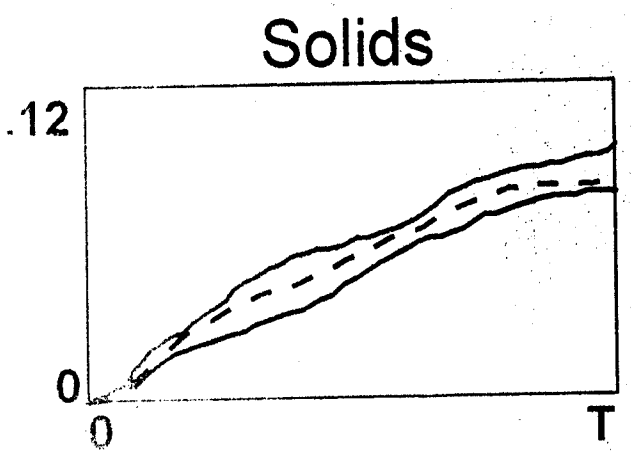


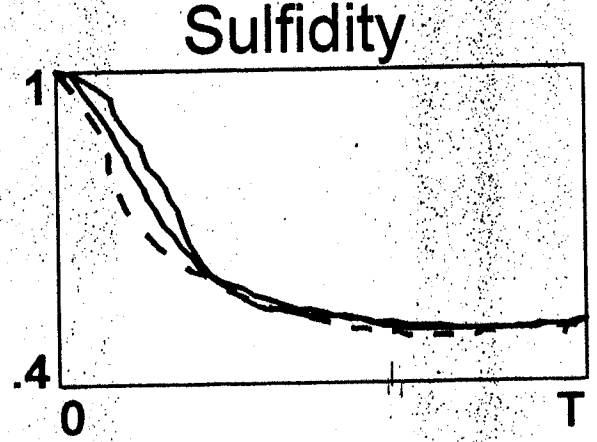
Kappa #

40 - target

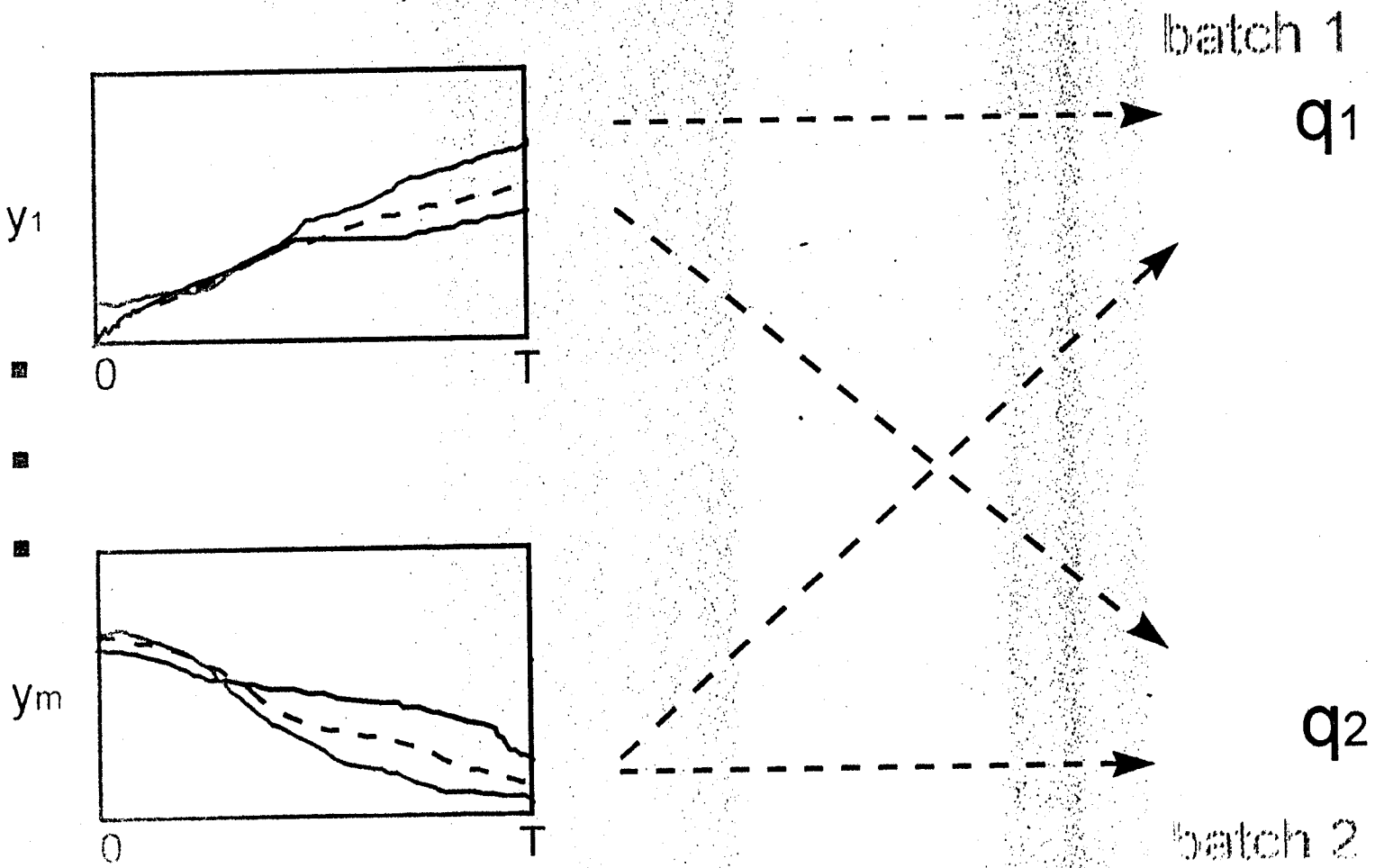
41

50

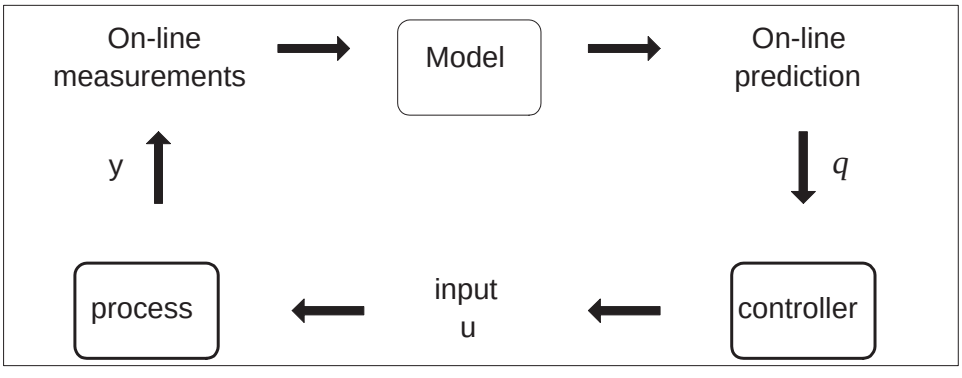




Relating On-line Data to Final Product Quality



Inferential Prediction and Control of Product Quality



1.6.3 NONLINEAR CONTROL

Linear model-based control can be inadequate for

- highly nonlinear processes (reactors, high-purity distillation column, batch processes, etc)
- process with large operating windows

MPC is a promising approach, but difficulties are in

- obtaining models (esp. through identification)
- computational complexity (NLP must be solved on-line)
- lack of theoretical understanding on the stability and robustness.

1.6.4 OTHER ISSUES

- control system maintenance
- integration with on-line optimization
- discrete-event systems, hybrid systems(e.q., start up)

Chapter 2

DYNAMIC MATRIX CONTROL

Dynamic Matrix Control

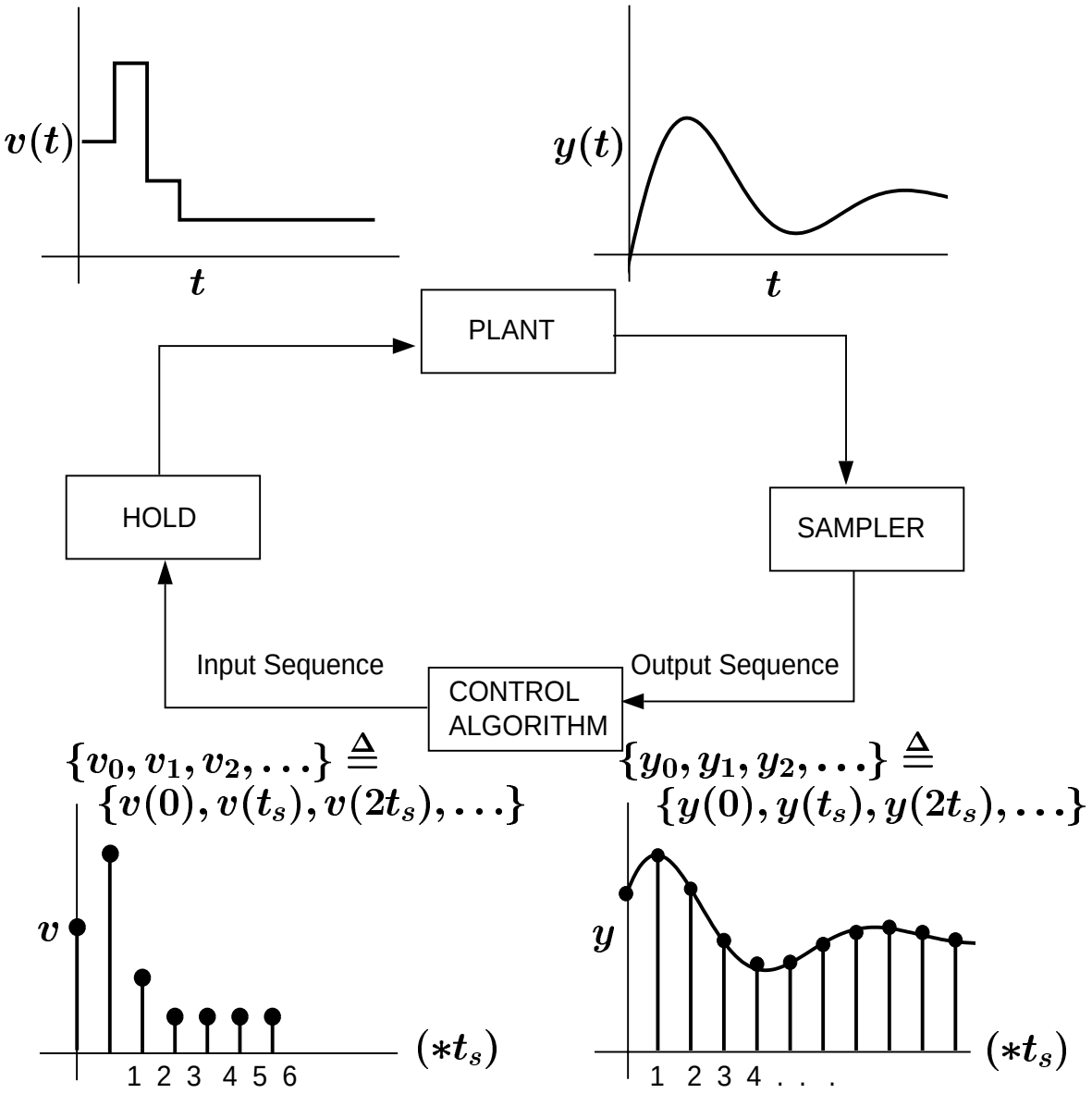
- Proposed by C. Cutler at Shell (later became the President of DMCC).
- Based on a system representation using step response coefficients.
- Currently being marketed by AspenTech (in the name of DMC-Plus).
- Prototypical of commercial MPC algorithms used in the process industries.

We will discuss the core features of the algorithm. There may be some differences in details.

2.1 FINITE IMPULSE AND STEP RESPONSE MODEL

2.1.1 OVERVIEW OF COMPUTER CONTROL

Computer Control System



Model for Computer Control

Should provide the following relation:

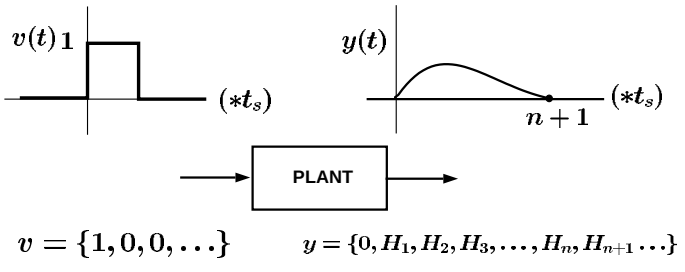
$$\{v(0), v(1), v(2), \cdots, v(\infty)\} \overset{model}{\rightarrow} \{y(1), y(2), \cdots, y(\infty)\}$$

We will concentrate on linear models. v and y are deviation variables, i.e., steady state is defined as

$$v'(k) = 0 \quad \forall k \qquad \rightarrow \qquad y'(0) = 0 \quad \forall k$$

2.1.2 IMPULSE RESPONSE AND IMPULSE RESPONSE MODEL

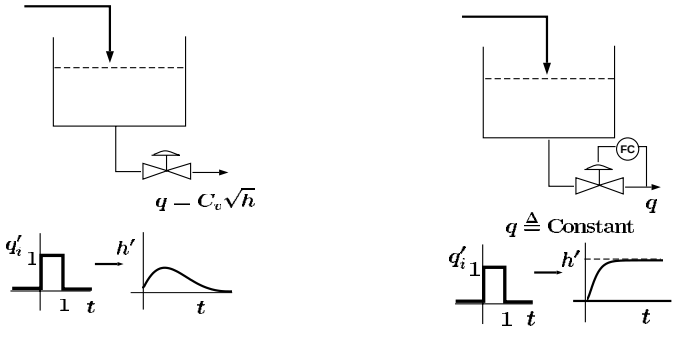
Impulse Response



Assumptions:

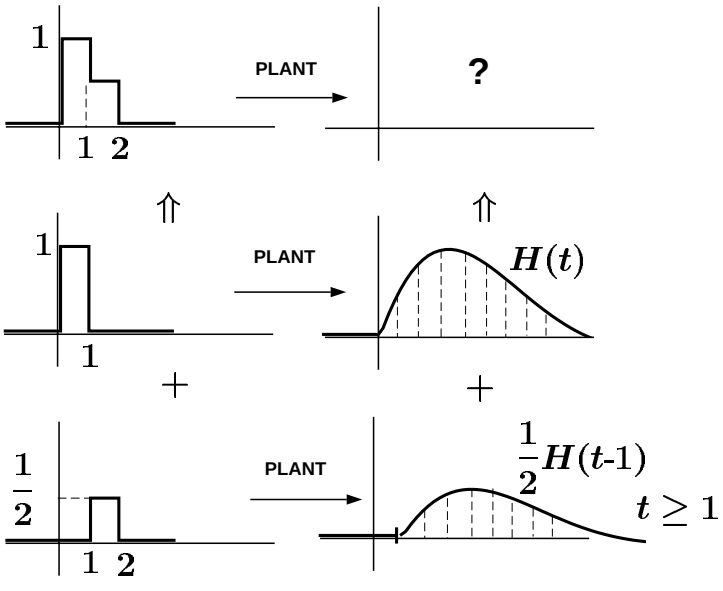
- $H_0 = 0$: no immediate effect of impulse response
- $\exists \;\; n$ s.t.$H_{n+1} = H_{n+2} = \cdots = 0$: "Finite Impulse Response" (reasonable for stable processes).

Examples:



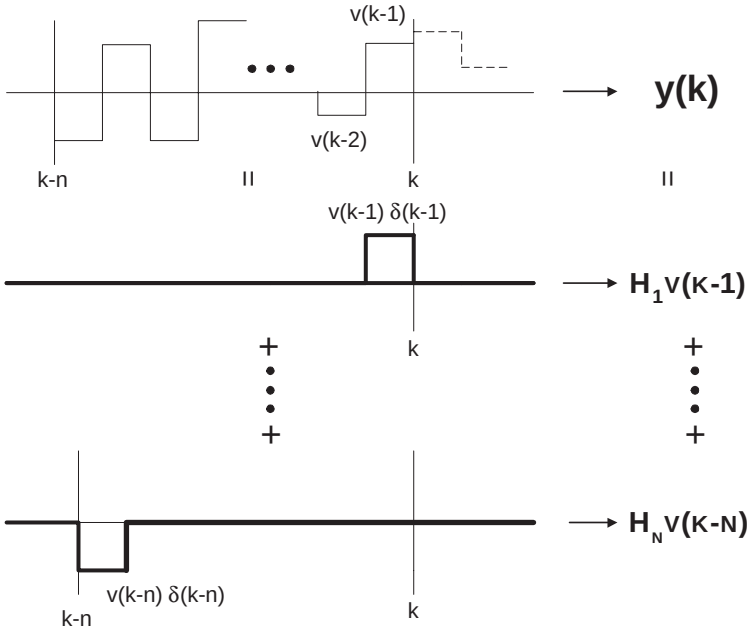
Finte Impulse Response Model

Superposition means $\Longrightarrow$ "Response adds and scales."



Using the superposition described above,

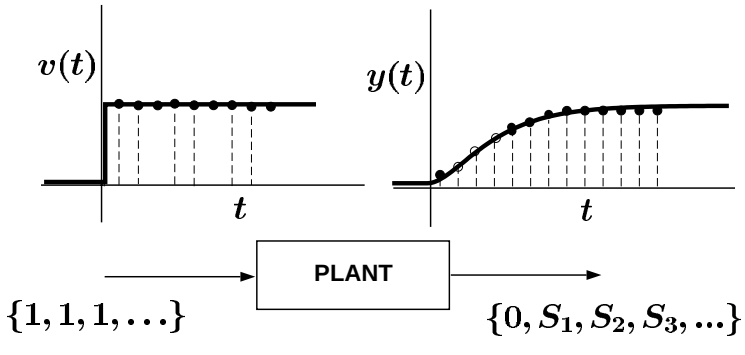
$$y(k) = H_1 v(k-1) + H_2 v(k-2) + \cdots\cdots + H_n v(k-n)$$



NOTE: need to have n-past inputs $(v(k-1), \cdots, v(k-n))$ in the memory.

2.1.3 STEP RESPONSE AND STEP RESPONSE MODEL

Step Response



Assumptions:

- $S_0 = 0$: no immediate effect of step input
- $S_{n+1} = S_{n+2} = \cdots = S_\infty$: equivalent "Finite Impulse Response"

(reasonable for stable processes)

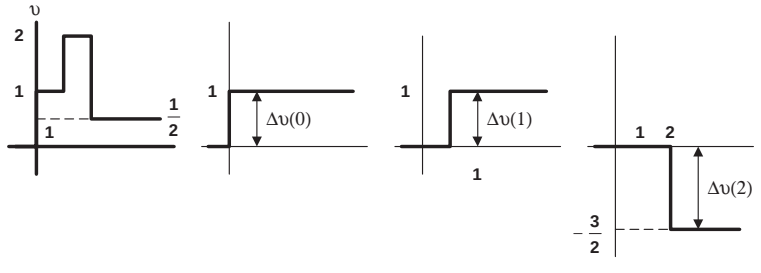
Relation between Impulse Response and Step Response:

$$S_k = \sum_{i=1}^{k} H_i v(k-i)$$

where $v(k-i) = 1$ for $1 \le i \le k$. Hence,

$$\begin{aligned} S_k &= \textstyle\sum_{i=1}^{k} H_i \\ H_k &= S_k - S_{k-1} \end{aligned}$$

Truncated Step Response Model



As shown above, any z.o.h. signal $v(t)$ can be represented as a sum of steps:

$$v(t) = \sum_{i=0}^{\infty} \Delta v(i) \mathcal{S}(t-i)$$

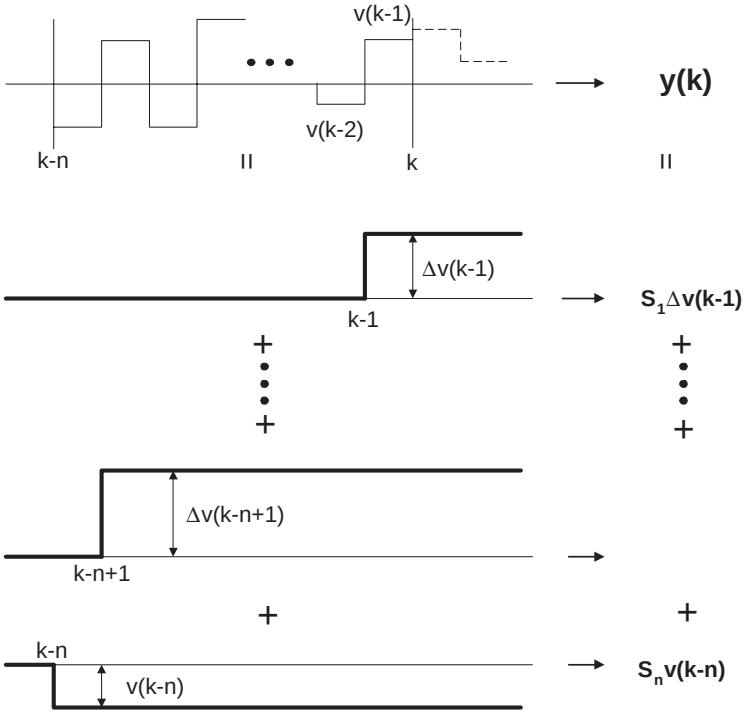
where $\Delta v(i) = v(i) - v(i-1)$ and $\mathcal{S}(t-i)$ is a unit step starting at the i_{th} time step.

Using this and superposition,

$$\begin{aligned} y(k) &= S_1 \Delta v(k-1) + S_2 \Delta v(k-2) + \cdots\cdots \\ &\quad + S_n \underbrace{\left(\Delta v(k-n) + \Delta v(k-n-1) + \cdots + \Delta v(0)\right)}_{v(k-n)} \end{aligned}$$

More compactly,

$$y(k) = \sum_{i=1}^{n-1} S_i \Delta v(k-i) + S_n v(k-n)$$



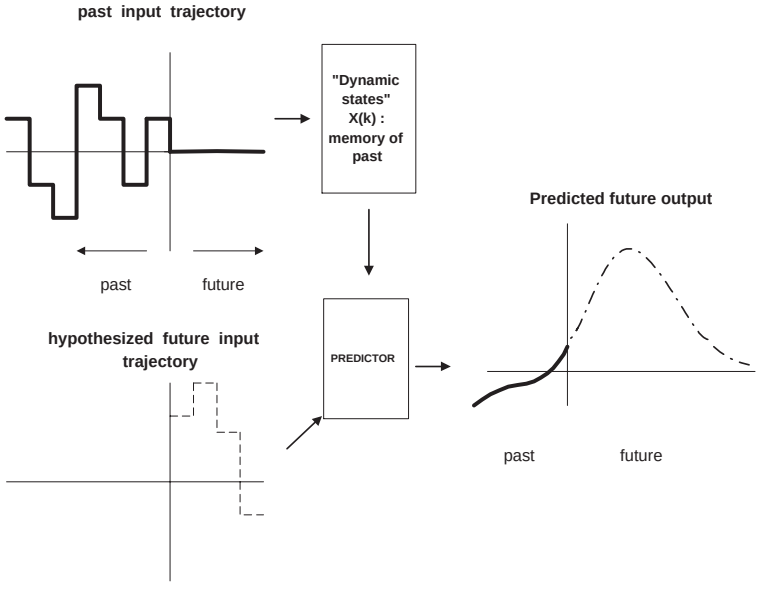
Note:

1. $\Delta v(k-i)$ instead of $v(k-i)$ appears in the model.
2. $v(k-n), \Delta v(k-n+1), \ldots, \Delta v(k-2), \Delta v(k-1)$ must be stored in the memory (Same storage requirement as in the FIR model).

2.2 MULTI-STEP PREDICTION

2.2.1 OVERVIEW

- In control, we are often interested in describing the *future* output behavior with a model.
- For a dynamic system, future output behavior depends on both *past* and future inputs.



Hence, past inputs must be remembered in some form for prediction. *Dynamic states* (in an input / output description) are defined as

> *memory* about the past inputs needed for prediction of the future output behavior

For a same system, states can be defined in many different ways, i.e., there are many ways to remember the past for the purpose of future prediction).

- For instance, states can consist of the entire past input trajectory:

$$x(k) = [v(k-1),\ v(k-2),\ \ldots\ldots,\ v(0)]^T$$

This choice is not practical since the memory size keeps growing with time.

- For an FIR system, one only has to keep n past inputs (Why?) :

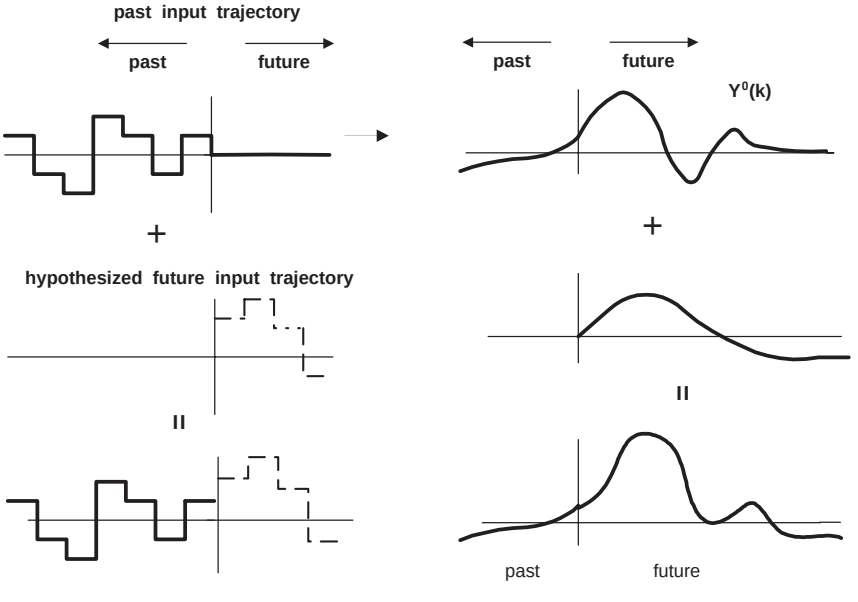
$$x(k) = [v(k-1),\ v(k-2),\ \ldots\ldots,\ v(k-n)]^T$$

With this choice of $x(k)$, we can certainly build the prediction of the future output behavior.

- Since the ultimate purpose of the memory is to predict future output, the *past* may be more conveniently tracked in terms of its effect on the future rather than the past itself. This is discussed next.

2.2.2 RECURSIVE MULTI-STEP PREDICTION FOR AN FIR SYSTEM

- **Separating Past and Future Input Effects**
  For linear systems, due to the separation principle, the effect of past and (hypothesized) future inputs can be computed separately and added:



- **Past Effects As Memory**

  Define $Y^0(k)$ as *future* output deviation due to *past* input deviation:

$$Y^0(k) = [y^0(k/k),\ y^0(k+1/k),\ \ldots\ldots, y^0(\infty/k)]^T$$

  where

$$y^0(i/k) \stackrel{\Delta}{=} y(i) \text{ assuming } v(k+j) = 0 \text{ for } j \geq 0$$

  Note that

$$y^0(k/k) = y(k)$$

  since the assumption of $v(k+j) = 0, j \geq 0$ does not affect the output at time k.

  Although $Y^0(k)$ is infinite dimensional, for FIR system, we only have to keep n terms (why?):

$$Y^0(k) = [y^0(k/k),\ y^0(k+1/k),\ \ldots\ldots, y^0(n/k)]^T$$

  This vector can be chosen as *states* since it describes the effect of *past* input deviation on future output deviation.

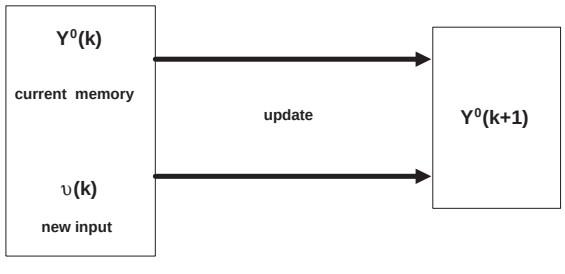
Future output can be written as

$$\begin{bmatrix} y(k+1) \\ y(k+2) \\ \vdots \\ \vdots \\ y(k+p) \end{bmatrix} = \underbrace{\begin{bmatrix} y^0(k+1/k) \\ y^0(k+2/k) \\ \vdots \\ \vdots \\ y^0(k+p/k) \end{bmatrix}}_{\text{Effect of Past Inputs From } Y^0(k)} + \underbrace{\begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ \vdots \\ H_p \end{bmatrix} v(k) + \begin{bmatrix} 0 \\ H_1 \\ \vdots \\ \vdots \\ H_{p-1} \end{bmatrix} v(k+1) + \ldots\ldots + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ H_1 \end{bmatrix} v(k+p-1)}_{\text{Effect of Hypothesized Future Inputs}}$$

We can see that such a defition of states can be very convenient for predictive control.

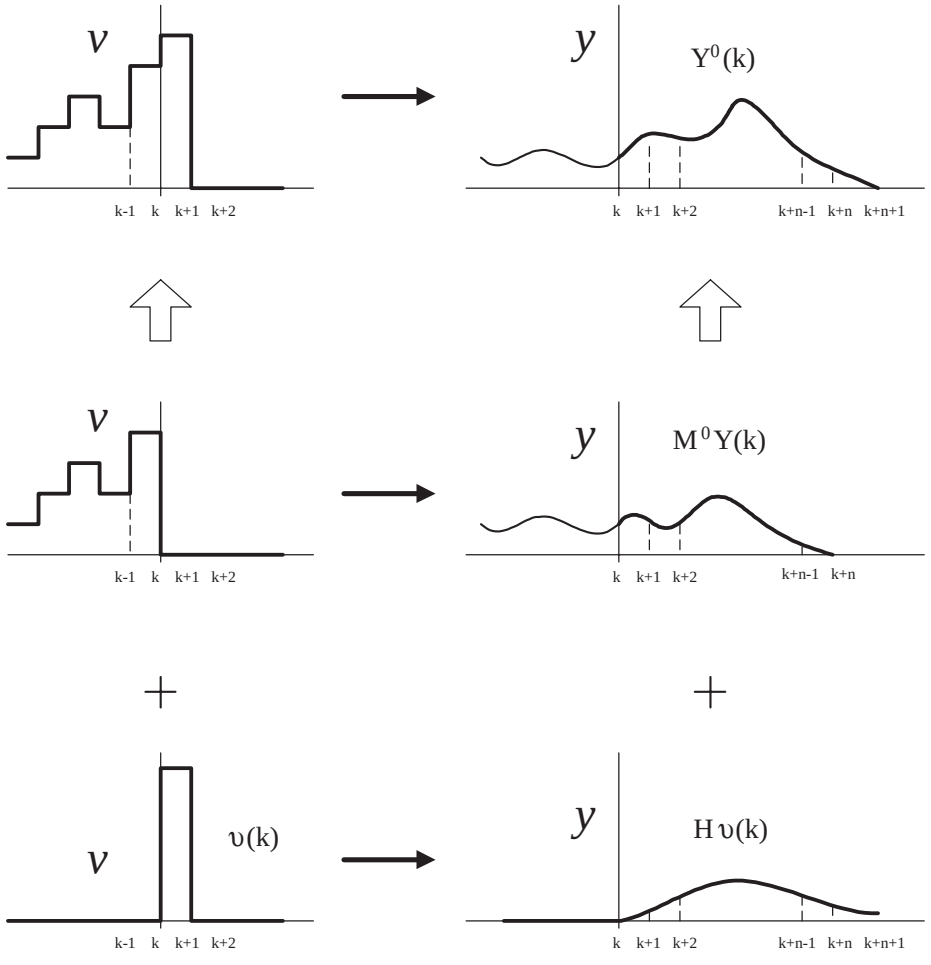
- **Recursive Update of Memory**

  Memory should be updated from one time step to next. For computer implementation, the update should occur in a recursive manner.



$Y^0(k)$ can be updated recursively as follows:

$$
\begin{array}{ccccccc}
Y^0(k) & \rightarrow & M^0Y^0(k) & + & Hv(k) & = & Y^0(k+1) \\
\begin{bmatrix} y^0(k/k) \\ y^0(k+1/k) \\ \vdots \\ \vdots \\ y^0(k+n-2/k) \\ y^0(k+n-1/k) \end{bmatrix} & & \begin{bmatrix} y^0(k+1/k) \\ y^0(k+2/k) \\ \vdots \\ \vdots \\ y^0(k+n-1/k) \\ y^0(k+n/k) \end{bmatrix} & + & \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ \vdots \\ H_{n-1} \\ H_n \end{bmatrix} v(k) & = & \begin{bmatrix} y^0(k+1/k+1) \\ y^0(k+2/k+1) \\ \vdots \\ \vdots \\ y^0(k+n-1/k+1) \\ y^0(k+n/k+1) \end{bmatrix} \\
0 & & & & & & \\
\vdots & & & & & &
\end{array}
$$



Mathematically, the above can be represented as

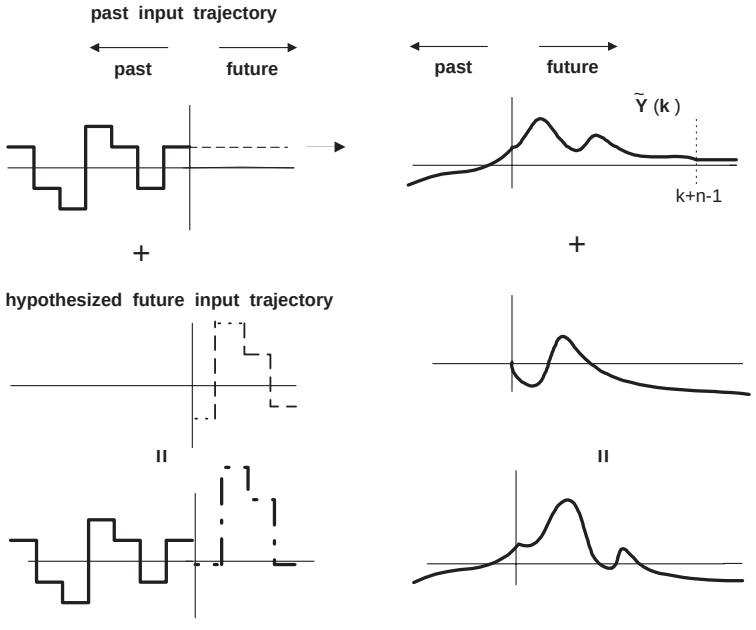
$$Y^0(k+1) = \underbrace{\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}}_{M^0} Y^0(k) + \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_{n-1} \\ H_n \end{bmatrix} v(k)$$

Note that multiplication by M^0 in the above represents the shift operation (which can be efficiently implemented on a computer).

2.2.3 RECURSIVE MULTI-STEP PREDICTION FOR AN FIR SYSTEM WITH DIFFERENCED INPUT

Multi-step prediction model can be developed in terms of step response coefficients as well.

- **Separating Past and Future Input Effects**
  Apply the superposition as before, but in a slightly different manner:



- **Past Effects As Memory**

  Define $\tilde{Y}(k)$ as *future* output deviation due to *past* input deviation plus current bias:

$$\tilde{Y}(k) = [\tilde{y}(k/k),\ \tilde{y}(k+1/k),\ \ldots\ldots,\tilde{y}(k+n-1/k)]^T$$

  where

$$\tilde{y}(i/k) \stackrel{\Delta}{=} y(i) \text{ assuming } \Delta v(k+j) = 0 \text{ for } j \geq 0$$

  Note that $\tilde{y}(k+n-1/k) = \tilde{y}(k+n/k) = \ldots = \tilde{y}(\infty/k)$, thus allowing the finite-dimensional representation of future output trajectory. This vector can be chosen as *states*.

Future output can be written as

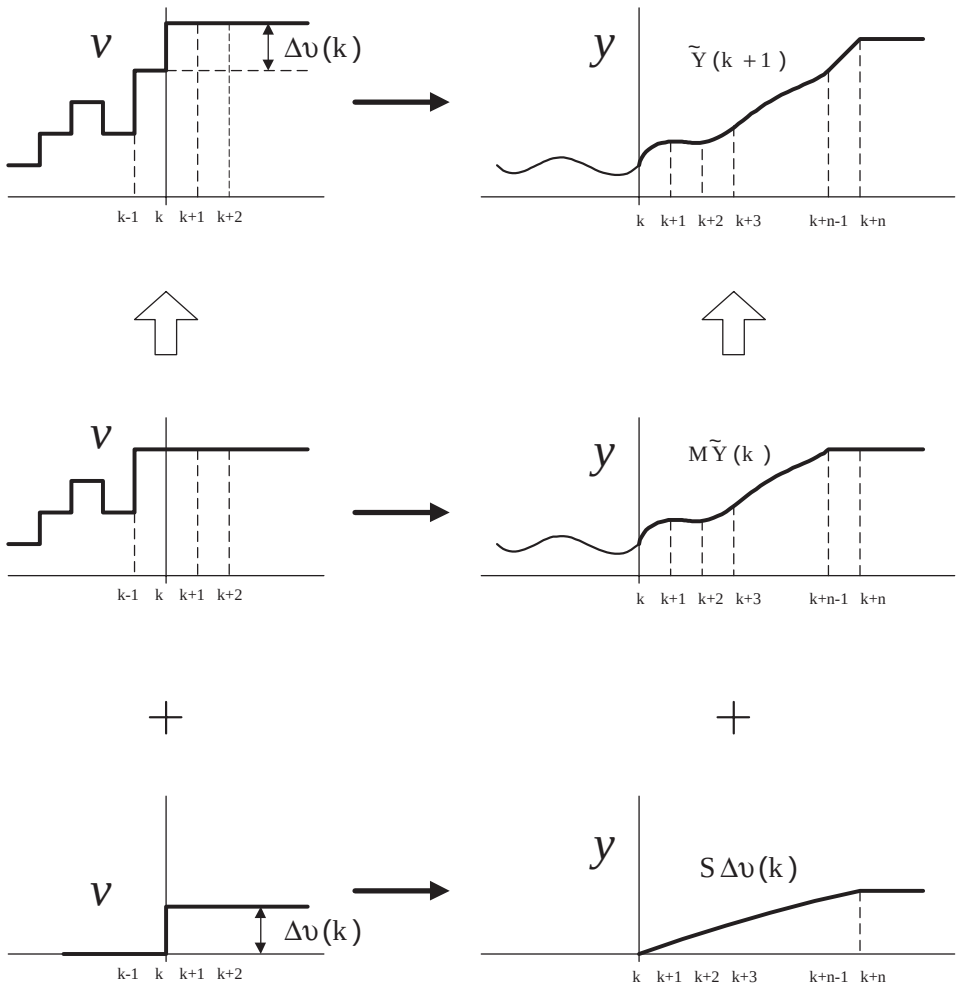
$$\begin{bmatrix} y(k+1) \\ y(k+2) \\ \vdots \\ \vdots \\ y(k+p) \end{bmatrix} = \underbrace{\begin{bmatrix} \tilde{y}(k+1/k) \\ \tilde{y}(k+2/k) \\ \vdots \\ \vdots \\ \tilde{y}(k+p/k) \end{bmatrix}}_{\text{Effect of Past Inputs + Current Bias (from } \tilde{Y}(k))} + \underbrace{\begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ \vdots \\ S_p \end{bmatrix} \Delta v(k) + \begin{bmatrix} 0 \\ S_1 \\ \vdots \\ \vdots \\ S_{p-1} \end{bmatrix} \Delta v(k+1) + \ldots + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ S_1 \end{bmatrix} \Delta v(k+p-1)}_{\text{Effect of Hypothesized Future Input Changes}}$$

We can see that such a defition of states can be very convenient for predictive control.

- **Recursive Update of Memory**

  $\tilde{Y}(k)$ can be updated recursively as follows:

$$\begin{array}{ccccc} \tilde{Y}(k) & & M\tilde{Y}(k) & + \quad S\Delta v(k) & = \quad \tilde{Y}(k+1) \\ \begin{bmatrix} \tilde{y}(k/k) \\ \tilde{y}(k+1/k) \\ \vdots \\ \vdots \\ \tilde{y}(k+n-2/k) \\ \tilde{y}(k+n-1/k) \end{bmatrix} & & \begin{bmatrix} \tilde{y}(k+1/k) \\ \tilde{y}(k+2/k) \\ \vdots \\ \vdots \\ \tilde{y}(k+n-1/k) \\ \tilde{y}(k+n/k) \end{bmatrix} & + \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ \vdots \\ S_{n-1} \\ S_n \end{bmatrix} \Delta v(k) & = \begin{bmatrix} \tilde{y}(k+1/k+1) \\ \tilde{y}(k+2/k+1) \\ \vdots \\ \vdots \\ \tilde{y}(k+n-1/k+1) \\ \tilde{y}(k+n/k+1) \end{bmatrix} \end{array}$$

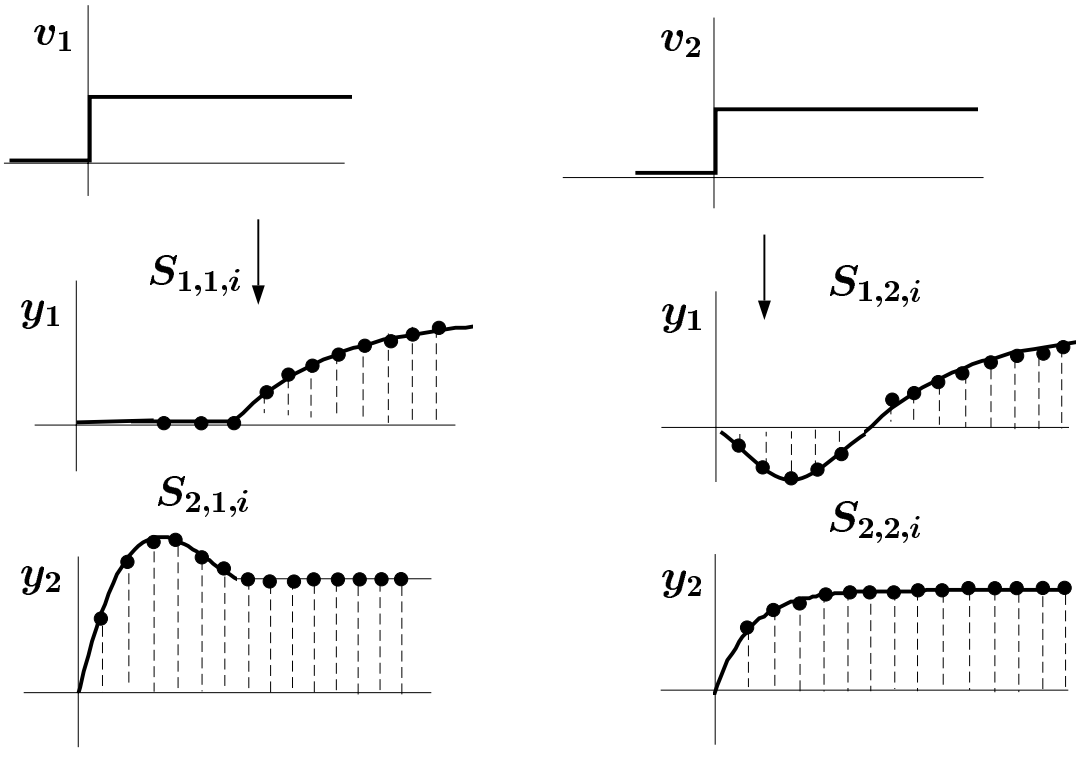


Hence,

$$\tilde{Y}(k+1) = \underbrace{\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}}_{M} \tilde{Y}(k) + \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n-1} \\ S_n \end{bmatrix} \Delta v(k)$$

Note that multiplication by M in the above represents a shift operation of different kind (the last element is repeated).

2.2.4 MULTIVARIABLE GENERALIZATION



$$S_i \triangleq i_{\text{th}} \text{ step response coefficient matrix} = \begin{bmatrix} S_{1,1,i} & S_{1,2,i} \\ S_{2,1,i} & S_{2,2,i} \end{bmatrix}$$

In general

$$S_i = \begin{bmatrix} S_{1,1,i} & S_{1,2,i} & \cdots & \cdots & S_{1,n_v,i} \\ S_{2,1,i} & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ S_{n_y,1,i} & S_{n_y,2,i} & \cdots & \cdots & S_{n_y,n_v,i} \end{bmatrix}$$

Again, define $\tilde{Y}_{k+1}$ and $\tilde{Y}_k$ in the same manner as before (now they are $(n \cdot n_y)$-dimensional vectors). Then,

$$\tilde{Y}(k+1) = M\tilde{Y}(k) + S\Delta v(k)$$

where

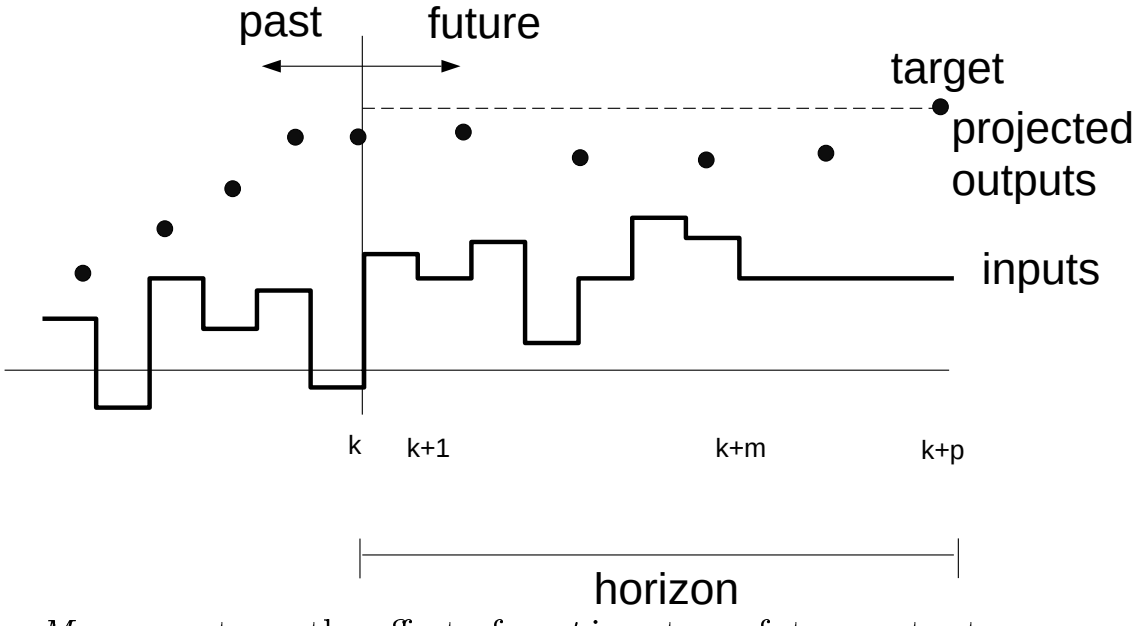
$$M = \begin{bmatrix} 0 & I & 0 & \cdots & 0 \\ 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & I \\ 0 & 0 & 0 & \cdots & I \end{bmatrix}$$

$$S = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n-1} \\ S_n \end{bmatrix}$$

where I is an $n_y \times n_y$ indentity matrix. Again, it merely represents the shift-operation; such a matrix does not need to be created in reality.

2.3 DYNAMIC MATRIX CONTROL ALGORITHM

2.3.1 MAJOR CONSTITUENTS



- *Memory:* stores the effect of *past* inputs on future outputs.
- *Predictor:* combines information stored in the memory with model information to predict the effect of hypothesized future input adjustments on future outputs.

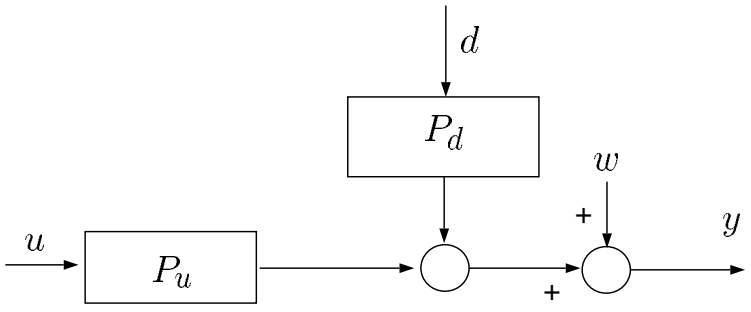
  $$(y(k+1|k), y(k+2|k), \ldots, y(k+p|k)) = f\left(\mathcal{I}(k), (\Delta u(k), \ldots, \Delta u(k+m-1))\right)$$

  where $\mathcal{I}(k)$ denotes the all the available information at k (stored in the memory).
- *Objective function and constraints*
- *Optimization program*

User-chosen parameters are the prediction horizon, control horizon, and parameters in the objective function and constraints.

2.3.2 BASIC PROBLEM SETUP

The basic system description we assume is



u : manpulated variable

d : measured / modelled disturbance

w_y : unmeasured disturbance + model / bias error effect

2.3.3 DEFINITION AND UPDATE OF MEMORY

Define the memory (state vector) as the effect of *past* deviation + current bias of *known* inputs (u and d) on the future output behavior:

$$\tilde{Y}(k) = \begin{bmatrix} y(k) \\ y(k+1) \\ \vdots \\ y(k+n-1) \end{bmatrix} \quad \text{with} \quad \begin{aligned} \Delta u(k) = \Delta u(k+1) = \cdots\cdots = 0 \\ \Delta d(k) = \Delta d(k+1) = \cdots\cdots = 0 \\ w_y(k) = w_y(k+1) = \cdots\cdots = 0 \end{aligned}$$

The memory update simply consists of:

$$M\tilde{Y}(k-1) + \begin{bmatrix} S^u & S^d \end{bmatrix} \overbrace{\begin{bmatrix} \Delta u(k-1) \\ \Delta d(k-1) \end{bmatrix}}^{\Delta v(k-1)} \quad \rightarrow \quad \tilde{Y}(k)$$

2.3.4 PREDICTION EQUATION

We can develop the prediction based on $\tilde{Y}(k)$ in the following manner ($y(k+\ell|k)$ denotes $y(k+\ell)$ predicted at $t=k$):

$$\begin{bmatrix} y(k+1|k) \\ y(k+2|k) \\ \vdots \\ \vdots \\ y(k+p|k) \end{bmatrix} = \underbrace{\begin{bmatrix} \tilde{y}(k+1/k) \\ \tilde{y}(k+2/k) \\ \vdots \\ \vdots \\ \tilde{y}(k+p/k) \end{bmatrix}}_{\substack{\Delta u(k)=\Delta u(k+1)=\cdots\cdots=0 \\ y(k+1),\cdots,y(k+p) \text{ with } \Delta d(k)=\Delta d(k+1)=\cdots\cdots=0 \\ w_y(k)=w_y(k+1)=\cdots\cdots=0}}$$

$$+\begin{bmatrix} S_1^u \\ S_2^u \\ \vdots \\ \vdots \\ S_p^u \end{bmatrix} \Delta u(k|k) + \begin{bmatrix} 0 \\ S_1^u \\ S_2^u \\ \vdots \\ S_{p-1}^u \end{bmatrix} \Delta u(k+1|k) + \cdots + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ S_1^u \end{bmatrix} \Delta u(k+p-1|k)$$

$$+\begin{bmatrix} S_1^d \\ S_2^d \\ \vdots \\ \vdots \\ S_p^d \end{bmatrix} \Delta d(k) + \begin{bmatrix} 0 \\ S_1^d \\ S_2^d \\ \vdots \\ S_{p-1}^d \end{bmatrix} \Delta d(k+1|k) + \cdots + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ S_1^d \end{bmatrix} \Delta d(k+p-1|k)$$

$$+\begin{bmatrix} w(k+1|k) \\ w(k+2|k) \\ \vdots \\ \vdots \\ w(k+p|k) \end{bmatrix}$$

There are more than a few terms (marked with $(\cdot|k)$) on the right-hand side

that are unavailabe at time k.

- Assume *piece-wise constant disturbances*, i.e.,

$$\Delta d(k+1|k) = \Delta d(k+2|k) = \cdots\cdots = 0$$

- Assume the effect of unmeasured disturbance, model error, etc. are described as a *piece-wise constant* signal.

$$w_y(k+1|k) = w_y(k+2|k) = \cdots\cdots = w_y(k|k) \approx \underbrace{y(k)}_{\text{real meas.}} - \underbrace{\tilde{y}(k/k)}_{\text{model prediction}}$$

- For flexibility in adjusting the computational load, consider only m ($\leq p$) input moves ($\Delta u(k|k), \Delta u(k+1|k), \cdots, \Delta u(k+m-1|k)$). This means, in your prediction, assume

$$\Delta u(k+m|k) = \Delta u(k+m-1|k) = \cdots = \cdots = \Delta u(k+p-1|k) = 0$$

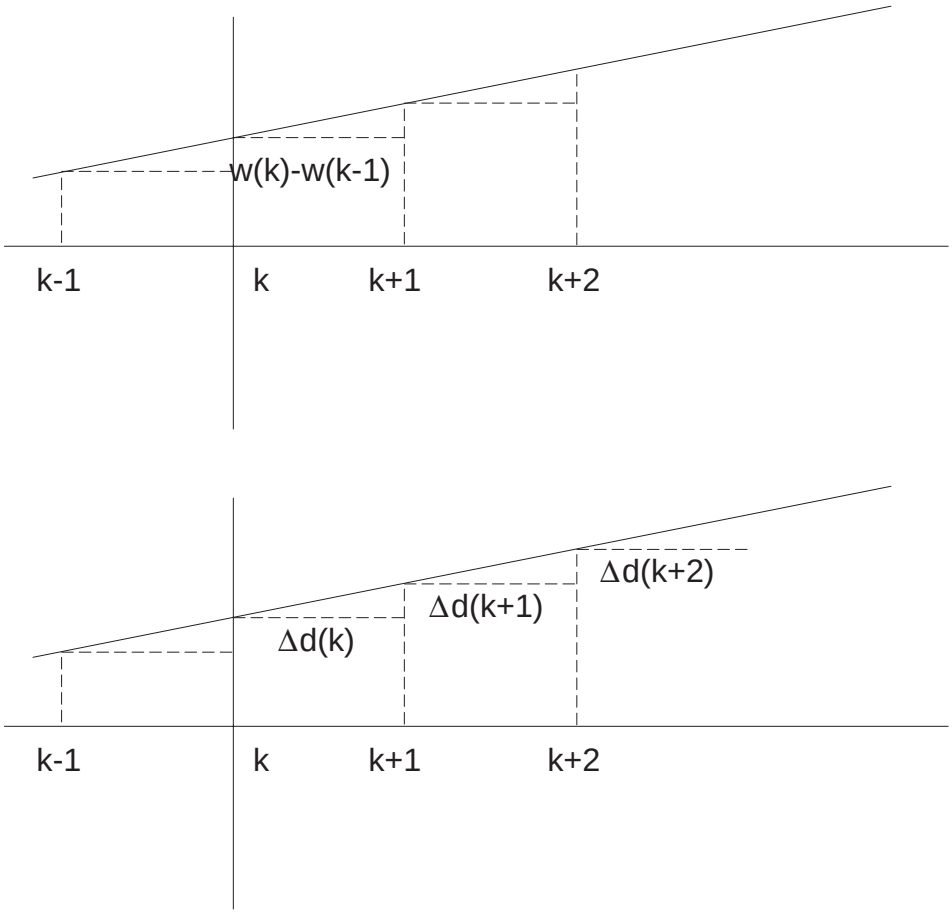
In summary,

$$\mathcal{Y}_{k+1|k} = \underbrace{\begin{bmatrix} \tilde{y}(k+1/k) \\ \tilde{y}(k+2/k) \\ \vdots \\ \vdots \\ \tilde{y}(k+p/k) \end{bmatrix}}_{\substack{M_p\tilde{Y}(k) \\ \text{from} \\ \text{memory}}} + \underbrace{\begin{bmatrix} S_1^d \\ S_2^d \\ \vdots \\ \vdots \\ S_n^d \end{bmatrix} \Delta d(k)}_{\substack{S^d\Delta d(k) \\ \text{feedforward} \\ \text{term}}}$$

$$+ \underbrace{\begin{bmatrix} y(k)-\tilde{y}(k/k) \\ y(k)-\tilde{y}(k/k) \\ \vdots \\ \vdots \\ y(k)-\tilde{y}(k/k) \end{bmatrix}}_{\substack{\mathcal{I}_p(y(k)-\tilde{y}(k/k)) \\ \text{feedback} \\ \text{term}}} + \underbrace{\begin{bmatrix} S_1^u & 0 & \cdots & \cdots & 0 \\ S_2^u & S_1^u & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ S_m^u & S_{m-1}^u & \cdots & \cdots & S_1^u \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ S_p^u & S_{p-1}^u & \cdots & \cdots & S_{p-m+1}^u \end{bmatrix}}_{\substack{\mathcal{S}^u \\ \text{dynamic} \\ \text{matrix}}} \underbrace{\begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \vdots \\ \Delta u(k+m-1|k) \end{bmatrix}}_{\substack{\Delta\mathcal{U}(k) \\ \text{future} \\ \text{input} \\ \text{moves}}}$$

NOTE: More complex (dynamic) extrapolation of the feedback errors is possible. For instance, for ramp disturbances, use

$$\mathbf{\Delta d(k)} = \Delta d(k+1|k) = \cdots\cdots = \Delta d(k+p-1|k)$$
$$w_y(k+\ell|k) = w_y(k|k) + \ell(w_y(k|k) - w_y(k-1|k-1))$$



2.3.5 QUADRATIC CRITERION

$$\min_{\Delta u(j|k)} [V(k) \triangleq \sum_{j=1}^{p}(r(k+i|k)-y(k+i|k))^T Q(r(k+i|k)-y(k+i|k)) + \sum_{\ell=0}^{m-1} \Delta u^T(k+\ell|k) R \Delta u(k+\ell|k)]$$

Q and R are weighting matrices; they are typically chosen as diagonal matrices.

Note that the objective function can be rewritten as

$$V(k) = \begin{bmatrix} r(k+1|k) - y(k+1|k) \\ r(k+2|k) - y(k+2|k) \\ \vdots \\ r(k+p|k) - y(k+p|k) \end{bmatrix}^T \begin{bmatrix} Q & & & \\ & Q & & \\ & & \ddots & \\ & & & Q \end{bmatrix} \begin{bmatrix} r(k+1|k) - y(k+1|k) \\ r(k+2|k) - y(k+2|k) \\ \vdots \\ r(k+p|k) - y(k+p|k) \end{bmatrix}$$
$$+ \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+m-1|k) \end{bmatrix}^T \begin{bmatrix} R & & & \\ & R & & \\ & & \ddots & \\ & & & R \end{bmatrix} \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+m-1|k) \end{bmatrix}$$

$$\Downarrow$$

$$V(k) = (\mathcal{R}(k+1|k) - \mathcal{Y}(k+1|k))^T \bar{Q} (\mathcal{R}(k+1|k) - \mathcal{Y}(k+1|k)) + \Delta\mathcal{U}^T(k) \bar{R} \Delta\mathcal{U}(k)$$

where

$$\mathcal{R}(k+1|k) = \begin{bmatrix} r(k+1|k) \\ r(k+2|k) \\ \vdots \\ \vdots \\ r(k+p|k) \end{bmatrix}; \quad \mathcal{Y}(k+1|k) = \begin{bmatrix} y(k+1|k) \\ y(k+2|k) \\ \vdots \\ \vdots \\ y(k+p|k) \end{bmatrix}$$

and

$$\bar{Q} = \text{blockdiag}[Q,\ Q,\ \ldots, \ldots,\ Q]; \quad \bar{R} = \text{blockdiag}[R,\ R,\ \ldots, \ldots,\ R]$$

Note that

$$\mathcal{Y}(k+1|k) = \underbrace{M_p \tilde{Y}(k) + S^d \Delta d(k) + \mathcal{I}_p \left(y(k) - \tilde{y}(k/k)\right)}_{\text{known term}} + \mathcal{S}^{\mathcal{U}} \Delta\mathcal{U}(k)$$

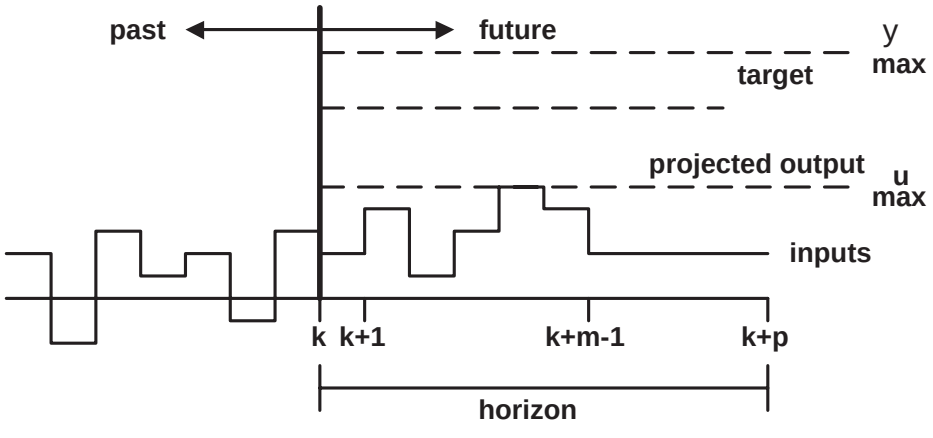
Hence, $V(k)$ is a quadratic function of $\Delta\mathcal{U}(k)$.

2.3.6 CONSTRAINTS

Constraints include

- Input magnitude constraints
- Input rate constraints
- Output magnitude constraints

At $t = k$, one has



$$
\begin{aligned}
& u_{\min} \le u(k+\ell|k) \le u_{\max} \\
& |\Delta u(k+\ell|k)| \le \Delta u_{\max}, \qquad \ell = 0, \cdots, m-1 \\
& y_{\min} \le y(k+j|k) \le y_{\max}, \quad j = 1, \cdots, p
\end{aligned}
$$

We want to express the above constraints as a linear inequalty in the form of

$$\mathcal{C}^{\mathcal{U}} \Delta \mathcal{U}(k) \le \mathcal{C}(k)$$

Manipulated Input Magnitude Constraints

$$u_{\min} \le u(k+\ell|k) \le u_{\max}, \qquad \ell = 0, \cdots, m-1$$

$$\Downarrow$$

$$\begin{aligned} \overbrace{u(k-1) + \sum_{i=0}^{\ell} \Delta u(k+i|k)}^{u(k+\ell|k)} &\ge u_{\min} \\ \underbrace{-u(k-1) - \sum_{i=0}^{\ell} \Delta u(k+i|k)}_{-u(k+\ell|k)} &\ge -u_{\max} \end{aligned}$$

$$\Downarrow$$

$$\begin{bmatrix} \begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & 0 & \vdots \\ \vdots & \vdots & \ddots & 0 \\ I & I & \cdots & I \end{bmatrix} \\ -\begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & 0 & \vdots \\ \vdots & \vdots & \ddots & 0 \\ I & I & \cdots & I \end{bmatrix} \end{bmatrix} \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+m-1|k) \end{bmatrix} \ge \begin{bmatrix} \begin{bmatrix} u_{\min} - u(k-1) \\ u_{\min} - u(k-1) \\ \vdots \\ u_{\min} - u(k-1) \end{bmatrix} \\ -\begin{bmatrix} u_{\max} - u(k-1) \\ u_{\max} - u(k-1) \\ \vdots \\ u_{\max} - u(k-1) \end{bmatrix} \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} I_L \\ -I_L \end{bmatrix} \Delta\mathcal{U}(k) \ge \begin{bmatrix} \begin{bmatrix} u_{\min} - u(k-1) \\ \vdots \\ u_{\min} - u(k-1) \end{bmatrix} \\ -\begin{bmatrix} u_{\max} - u(k-1) \\ \vdots \\ u_{\max} - u(k-1) \end{bmatrix} \end{bmatrix}$$

Manipulated Variable Rate Cosntraints

$$|\Delta u(k+\ell|k)| \le \Delta u_{\max}, \qquad \ell = 0, \cdots, m-1$$

$$\Downarrow$$

$$-\Delta u_{\max} \le \Delta u(k+\ell|k) \le \Delta u_{\max}$$

$$\Downarrow$$

$$\Delta u(k+\ell|k) \ge -\Delta u_{\max}$$
$$-\Delta u(k+\ell|k) \ge -\Delta u_{\max}$$

$$\Downarrow$$

$$\begin{bmatrix} \begin{bmatrix} I & 0 & \cdots & 0 \\ 0 & I & 0 & \vdots \\ \vdots & \cdots & \ddots & 0 \\ 0 & 0 & \cdots & I \end{bmatrix} \\ -\begin{bmatrix} I & 0 & \cdots & 0 \\ 0 & I & 0 & \vdots \\ \vdots & \cdots & \ddots & 0 \\ 0 & 0 & \cdots & I \end{bmatrix} \end{bmatrix} \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+m-1|k) \end{bmatrix} \ge \begin{bmatrix} -\begin{bmatrix} \Delta u_{\max} \\ \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{bmatrix} \\ \begin{bmatrix} \Delta u_{\max} \\ \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{bmatrix} \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} I \\ -I \end{bmatrix} \Delta \mathcal{U}(k) \ge \begin{bmatrix} -\begin{bmatrix} \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{bmatrix} \\ \begin{bmatrix} \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{bmatrix} \end{bmatrix}$$

Output Magnitude Constraints

$$y_{\min} \le y(k+j|k) \le y_{\max}, \qquad j = 1, \cdots, p$$

$$\Downarrow$$

$$\begin{aligned} y(k+j|k) &\ge y_{\min} \\ -y(k+j|k) &\ge -y_{\max} \end{aligned}$$

$$\Downarrow$$

$$\begin{bmatrix} M_p\tilde{Y}(k) + \mathcal{S}^d\Delta d(k) + \mathcal{I}_p(y(k) - \tilde{y}(k/k)) + \mathcal{S}^{\mathcal{U}}\Delta\mathcal{U}(k) \\ -M_p\tilde{Y}(k) - \mathcal{S}^d\Delta d(k) - \mathcal{I}_p(y(k) - \tilde{y}(k/k)) - \mathcal{S}^{\mathcal{U}}\Delta\mathcal{U}(k) \end{bmatrix} \ge \begin{bmatrix} \mathcal{Y}_{\min} \\ -\mathcal{Y}_{\max} \end{bmatrix}$$

where

$$\mathcal{Y}_{\min} = \begin{bmatrix} y_{\min} \\ y_{\min} \\ \vdots \\ y_{\min} \end{bmatrix} \qquad \mathcal{Y}_{\max} = \begin{bmatrix} y_{\max} \\ y_{\max} \\ \vdots \\ y_{\max} \end{bmatrix}$$

since

$$\mathcal{Y}(k+1|k) \triangleq \begin{bmatrix} y(k+1|k) \\ \vdots \\ y(k+p|k) \end{bmatrix} = M_p\tilde{Y}(k) + \mathcal{S}^d\Delta d(k) + \mathcal{I}_p(y(k) - \tilde{y}(k/k)) + \mathcal{S}^{\mathcal{U}}\Delta\mathcal{U}(k)$$

$$\Downarrow$$

$$\begin{bmatrix} \mathcal{S}^{\mathcal{U}} \\ -\mathcal{S}^{\mathcal{U}} \end{bmatrix} \Delta\mathcal{U}(k) \ge \begin{bmatrix} \mathcal{Y}_{\min} - M_p\tilde{Y}(k) - \mathcal{S}^d\Delta d(k) - \mathcal{I}_p(y(k) - \tilde{y}(k/k)) \\ -\mathcal{Y}_{\max} + M_p\tilde{Y}(k) + \mathcal{S}^d\Delta d(k) + \mathcal{I}_p(y(k) - \tilde{y}(k/k)) \end{bmatrix}$$

In summary, we have

$$\begin{bmatrix} I_L \\ -I_L \\ I \\ -I \\ \mathcal{S}^{\mathcal{U}} \\ -\mathcal{S}^{\mathcal{U}} \end{bmatrix} \Delta\mathcal{U}(k) \geq \begin{bmatrix} -\begin{bmatrix} u_{\min} - u(k-1) \\ \vdots \\ u_{\min} - u(k-1) \end{bmatrix} \\ \begin{bmatrix} u_{\max} - u(k-1) \\ \vdots \\ u_{\max} - u(k-1) \end{bmatrix} \\ -\begin{bmatrix} \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{bmatrix} \\ \begin{bmatrix} \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{bmatrix} \\ \begin{bmatrix} \mathcal{Y}_{\min} - M_p\tilde{Y}(k) - \mathcal{S}^d\Delta d(k) - \mathcal{I}_p(y(k) - \tilde{y}(k/k)) \\ -\mathcal{Y}_{\max} + M_p\tilde{Y}(k) + \mathcal{S}^d\Delta d(k) + \mathcal{I}_p(y(k) - \tilde{y}(k/k)) \end{bmatrix} \end{bmatrix}$$

The above is in the form of linear equality,

$$\mathcal{C}^{\mathcal{U}}\Delta\mathcal{U}(k) \geq \mathcal{C}(k)$$

Note that $\mathcal{C}^{\mathcal{U}}$ is a constant matrix while $\mathcal{C}(k)$ must be updated at each time step.

Although not treated here, time-varying constraints can easily be incorporated into the formulation.

2.3.7 QUADRATIC PROGRAMMING

Problem:

At each sample time, we have a minimization with an objective function

$$V(k) = (\mathcal{R}(k+1|k) - \mathcal{Y}(k+1|k))^T \bar{Q} (\mathcal{R}(k+1|k) - \mathcal{Y}(k+1|k)) + \Delta\mathcal{U}^T(k) \bar{R} \Delta\mathcal{U}(k)$$

with the prediction equation constraint

$$\mathcal{Y}(k+1|k) = M_p \tilde{Y}(k) + S^d \Delta d(k) + \mathcal{I}_p \left(y(k) - \tilde{y}(k/k)\right) + \mathcal{S}^{\mathcal{U}} \Delta\mathcal{U}(k)$$

and the inequality constraint

$$\mathcal{C}^{\mathcal{U}} \Delta\mathcal{U}(k) \geq \mathcal{C}(k)$$

Putting Into the Standard QP Form:

Substituting the prediction equation constraint into the objective gives

$$\begin{aligned} V(k) &= \mathcal{E}^T(k) \bar{Q} \mathcal{E}(k) - \underbrace{2\mathcal{E}^T(k) \bar{Q} \mathcal{S}^{\mathcal{U}}}_{\mathcal{G}^T(k)} \Delta\mathcal{U}(k) + \Delta\mathcal{U}^T(k) \underbrace{\left(\mathcal{S}^{\mathcal{U}T} \bar{Q} \mathcal{S}^{\mathcal{U}} + \bar{R}\right)}_{\mathcal{H}} \Delta\mathcal{U}(k) \\ \mathcal{E}(k) &= \mathcal{R}(k+1|k) - M_p \tilde{Y}(k) - S^d \Delta d(k) - \mathcal{I}_p \left(y(k) - \tilde{y}(k/k)\right) \end{aligned}$$

Note that $\mathcal{E}(k)$ can be computed with information given to us at time k. Hence, $V(k)$ is a quadratic function of $\Delta\mathcal{U}(k)$ with hessian matrix $\mathcal{H}$ and gradient vector $\mathcal{G}(k)$.

Since we have a minimization of a quadratic objective with a linear inequality constraint, we have a quadratic programming (QP). The standard form of quadratic programming is

$$\begin{gathered} \min_x x^T H x^T - g^T x \\ Cx \geq c \end{gathered}$$

The parameters that should be supplied to the QP solver are H, g, C and c.

In our case, at $t = k$,

$$
\begin{aligned}
&x : \Delta\mathcal{U}(k) \\
&H : \mathcal{H} \triangleq (\mathcal{S}^{\mathcal{U}})^T \bar{Q} \mathcal{S}^{\mathcal{U}} + \bar{R} \\
&g : \mathcal{G}(k) \triangleq 2(\mathcal{S}^{\mathcal{U}})^T \bar{Q}(M_p \tilde{Y}(k) + \mathcal{S}^d \Delta d(k) + \mathcal{I}_p(y(k) - \tilde{y}(k/k)) - \mathcal{R}_{k+1|k}) \\
&C : \mathcal{C}^u \triangleq \begin{bmatrix} I_L \\ -I_L \\ I \\ -I \\ \mathcal{S}^{\mathcal{U}} \\ -\mathcal{S}^{\mathcal{U}} \end{bmatrix} \\
&c : \mathcal{C}(k) \triangleq \begin{bmatrix} -\begin{bmatrix} u_{\min} - u(k-1) \\ \vdots \\ u_{\min} - u(k-1) \end{bmatrix} \\ \begin{bmatrix} u_{\max} - u(k-1) \\ \vdots \\ u_{\max} - u(k-1) \end{bmatrix} \\ -\begin{bmatrix} \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{bmatrix} \\ \begin{bmatrix} \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{bmatrix} \\ \begin{bmatrix} \mathcal{Y}_{\min} - M_p \tilde{Y}(k) - \mathcal{S}^d \Delta d(k) - \mathcal{I}_p(y(k) - \tilde{y}(k/k)) \\ -\mathcal{Y}_{\max} + M_p \tilde{Y}(k) + \mathcal{S}^d \Delta d(k) + \mathcal{I}_p(y(k) - \tilde{y}(k/k)) \end{bmatrix} \end{bmatrix}
\end{aligned}
$$

Following are some comments on quadratic programming.

- QP is convex and therefore fundamentally tractable.
- The solution doesn't necessarily lie at the vertices of feasible region(unlike LPs). One may have any number of active constraints

(up to the QP dimension).

- The size of QP is $m \times n_u$ where m is the control input horizon and n_u is the number of input. Computational time of QP depends on many things (e.g., Hessian size, its structure, number of constraints, the proximity of the solution to the active constraints) and is difficult to predict.
- Off-the-shelf QP solver is available, but is often not the best choice in terms of computational efficiency. Because the hessian and gradient for QP tends to be highly structured (sparse), an algorithm tailored to take advantage of this is recommended.
- QP solver requires inversion of the hessian. Since the hessian is a constant matrix (given fixed input / output weights and model parameters), it only needs to be inverted once off-line. This eliminates the time-consuming step of inverting the hessian at each QP run. Only when the weighting matrices are model parameters are changed, hessian needs to be recomputed and inverted in the background.
- Since most QPs are feasible-path algorithms, the number of inequality constraints also affect the computational time. One should use the constraints sparingly.
- The most well-known solution strategy is the active set strategy. In this method, first a feasible solution is found. Then, the least squares problem is solved with the active constraints as equality constraints. The optimality of the solution is checked through Kuhn-Tucker conditions. If they are not satisfied, the active constraint set is updated and the procedure is repeated.
- Another emerging method is the interior point (IP) method. In the IP method, a barrier function is used to trap the solution *within* the feasible region. Newton interation is used to converge to the optimum. This method has many attractive features like quick convergence (most

problems converge with 5-50 iterations regardless of the problem size) and ability exploit the problem structure.

2.3.8 SUMMARY OF REAL-TIME IMPLMENTATION

1. *Initialization:* Initialize the memory vector $\tilde{Y}(0)$ and the reference vector $\mathcal{R}(1|0)$. Set $k = 1$.
2. *Memory Update:*

$$\tilde{Y}(k-1) \quad \rightarrow \quad M\tilde{Y}(k-1)+S^u\Delta u(k-1)+S^d\Delta d(k-1) \quad \rightarrow \quad \tilde{Y}(k)$$

$$\begin{bmatrix} \tilde{y}(k-1/k-1) \\ \tilde{y}(k/k-1) \\ \vdots \\ \vdots \\ \tilde{y}(k+n-3/k-1) \\ \tilde{y}(k+n-2/k-1) \\ \tilde{y}(k+n-1/k-1) \\ \vdots \end{bmatrix} \begin{bmatrix} \tilde{y}(k/k-1) \\ \tilde{y}(k+1/k-1) \\ \vdots \\ \vdots \\ \tilde{y}(k+n-2/k-1) \\ \tilde{y}(k+n-1/k-1) \end{bmatrix} + \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ \vdots \\ S_{n-1} \\ S_n \end{bmatrix} \Delta v(k-1) = \begin{bmatrix} \tilde{y}(k/k) \\ \tilde{y}(k+1/k) \\ \vdots \\ \vdots \\ \tilde{y}(k+n-2/k) \\ \tilde{y}(k+n-1/k) \end{bmatrix}$$

3. *Reference Vector Update:* Update $\mathcal{R}(k+1|k)$ by shifting $\mathcal{R}(k|k-1)$ and entering a new reference value.

$$\begin{bmatrix} r(k|k-1) \\ r(k+1|k-1) \\ \vdots \\ \vdots \\ r(k+p-2|k-1) \\ r(k+p-1|k-1) \end{bmatrix} \qquad \begin{bmatrix} r(k+1|k) \\ r(k+2|k) \\ \vdots \\ \vdots \\ r(k+p-1|k) \\ r(k+p|k) \end{bmatrix}$$

4. *Measurement Intake:* Take in new measurement $y(k)$ and $\Delta d(k)$.

5. *Calculation of the Gradient Vector and Constraint Vector:*

$$\tilde{G}(k) = 2(\mathcal{S}^{\mathcal{U}})^T \bar{Q}(M_p \tilde{Y}(k) + \mathcal{S}^d \Delta d(k) + \mathcal{I}_p(y(k) - \tilde{y}(k/k)) - \mathcal{R}(k+1|k))$$

   Update the constraint vector $\mathcal{C}(k)$ similarly.

6. *Solve QP:* Call the QP subroutine with pre-inverted $\mathcal{H}$,$\mathcal{C}^{\mathcal{U}}$ and computed $\mathcal{G}(k), \mathcal{C}(k)$.

7. *Implementation of input*: Implement $\Delta u(k|k)$:

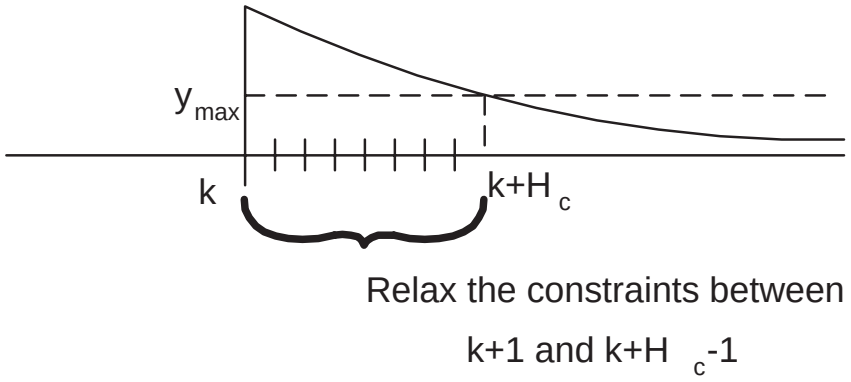
$$u(k) = u(k-1) + \Delta u(k|k)$$

8. Go back to Step 2 after setting $k = k + 1$.

2.4 ADDITIONAL ISSUES

2.4.1 FEASIBILITY ISSUE AND CONSTRAINT RELAXATION

- Output constraints can become infeasible (impossible to satisfy). For example, if we require $-\epsilon \leq y(k+\ell|k) \leq \epsilon$ for all ℓ, as $\epsilon \to 0$, the constraint becomes infeasible.
- When the QP is declared infeasible, one must relax the output constraints. Various ways to relax the constraints exist:
  - Relax the constraint starting from the initial time one by one until it is feasible.



– Soften the constraint and penalize the degree of softening:

$$\min_{\epsilon,\Delta\mathcal{U}(k)}[\text{Usual Objective}] + \lambda\epsilon^2$$

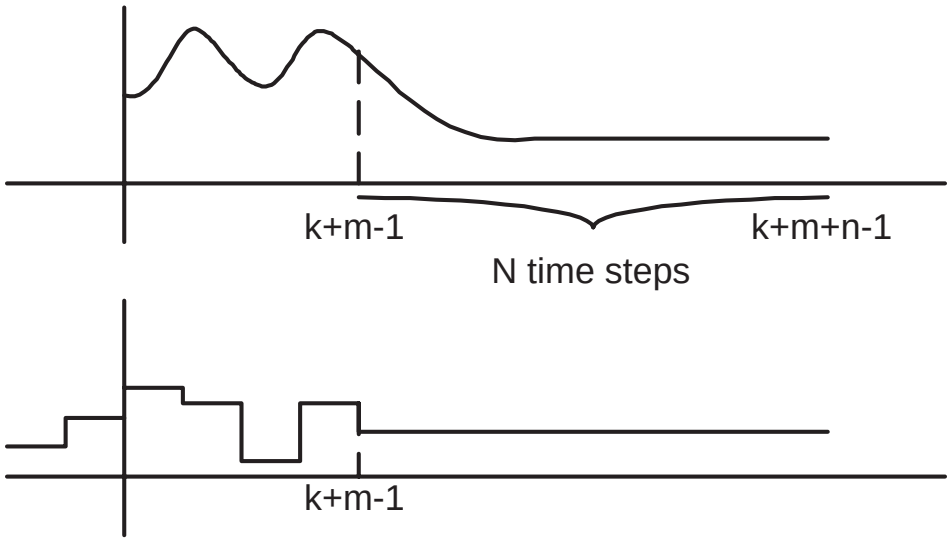
$$y_{\min} - \epsilon \le y(k+\ell|k) \le y_{\max} + \epsilon$$
plus other constraints

2.4.2 GUIDELINES FOR CHOOSING THE HORIZON SIZE

In order to obtain good closed-loop properties and consistent tuning effect from problem to problem, it is recommended to use a very large or preferably infinite prediction horizon (Long-sighted decision making produces better results in general). ∞-horizon DMC can be implemented in the following way:

- choose m as large as possible (within the computational limit).
- choose
$$p = m + n$$
where n is the number of time steps for step responses to settle.
- add constraint
$$y(k+p|k) = 0$$

Note that the above choice of p with the equality constraint amounts to choosing $p = \infty$. Stability of the closed-loop system is guaranteed under this cuhoice (regardless of choice of m). Choice of m is not critical for stability; a larger m should result in better performance at the expense of increased computational requirement.



The lesson is

- Use large enough horizon for system responses to settle.
- Try to penalize the endpoint error more (if not constrain to zero).

2.4.3 BI-LEVEL FORMULATION

In the DMC algorithm, control computation *at each sample time* is done in two steps:

- *Steady State Optimization:* Here model prediction at steady state is used to determine the optimal steady state. The steady-state model is

in the form of

$$y(\infty|k) = K_s \underbrace{(u(\infty|k) - u(k-1))}_{\Delta u_s(k)} + b(k)$$

With only m moves considered,

$$\Delta u_s(k) = \Delta u(k|k) + \Delta u(k+1|k) + \ldots\ldots + \Delta u(k+m-1|k)$$

and with FIR assumption,

$$y(\infty|k) = y(k+m+n-1|k)$$

and $K_s = S_n$. Hence, the steady prediction equation can be easily extracted from the dynamic prediciton equation we had earlier.

In terms of the optimization criterion, various choices are possible.

- Most typically, some kind of linear economic criterion is used along with constraints on the inputs and outputs:

  $$\min_{\Delta u_s(k)} [\ell(u(\infty|k), y(\infty|k))]$$

  In this case, a linear programming (LP) results.

- Sometimes, the objective is chosen to minimize the input move size while satisfying various input / output constraints (posed by control requirements, actuator limits plus those set by the rigorous plant optimizer):

  $$\min_{\Delta u_s(k)} [|\Delta u_s(k)|]$$

  Again, an LP results.

- In the pure regulation problems where setpoint for the output is fixed, one may use

  $$\min_{\Delta u_s(k)} [(r - y(\infty|k))^T Q (r - y(\infty|k))]$$

This combined with subsequently discussed QP results in *Infinite-Horizon MPC*.

- *Dynamic Optimization:* Once the steady-state target is fixed, the following QP is solved to drive the outputs (and sometimes also inputs) to their chosen targets quickly without violating constraints:

$$\begin{aligned}\min_{\Delta u(j|k)} \quad & \left[\sum_{i=1}^{m+n-2}(y(k+i|k)-y^*(\infty|k))^T Q(y(k+i|k)-y^*(\infty|k))\right. \\ & \left.+\sum_{j=0}^{m-1}\Delta u^T(k+j|k)R\Delta u(k+j|k)\right]\end{aligned}$$

subject to

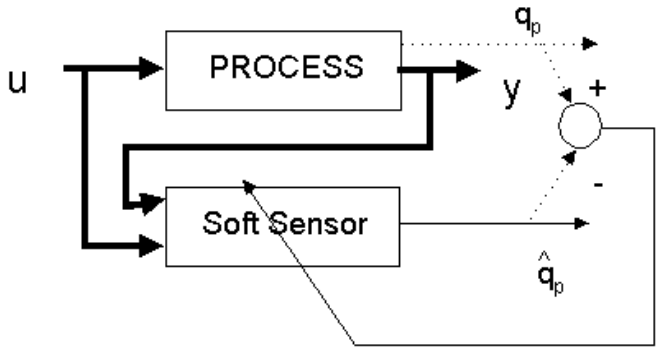
$$\Delta u(k|k)+\Delta u(k+1|k)+\ldots\ldots+\Delta u(k+m-1|k)=\Delta u_s^*(k)$$

plus various other constraints. This is a QP.

The last constraint forces $y(k+m+n-1|k)$ to be at the optimal steady-state value $y^*(k+\infty|k)$.

Note: The above steady-state optimization is to be distinguished from the rigorous plant-wide optimization. The above is performed at every sample time of MPC while the rigorous optimization is done much more infrequently.

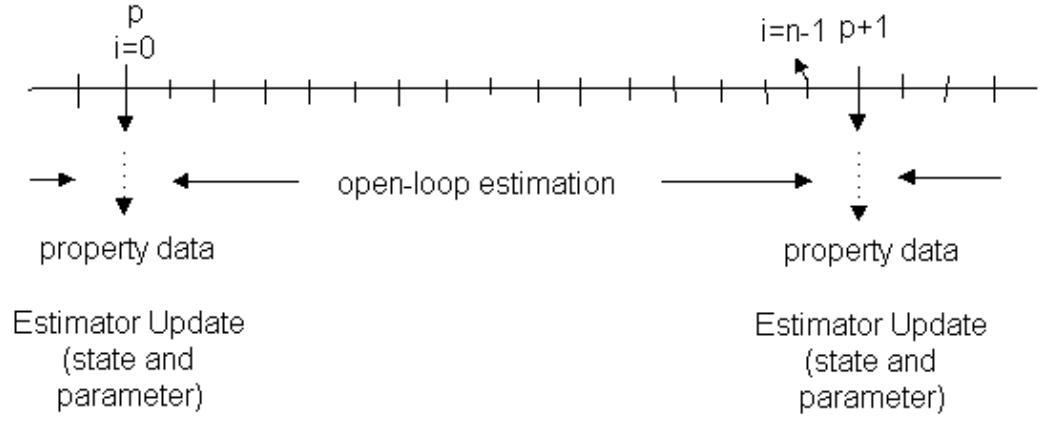
2.4.4 PROPERTY ESTIMATION



- Property data q are usually obtained through on-line analyzer or lab. analysis.
- Both have significant delays and limited sampling capabilities (more so for lab. analysis).
- On-line anayzers are highly unreliable (prone to failures).
- Using more reliable fast process measurments $\mathbf{y}$ (and possibly $\mathbf{u}$), we can estimate product properties at a higher frequency with a minimal delay.
- The property estimator (sometimes called *soft sensor*) can be constructed from a fundamental model or more commonly through data regression.
- Almost all estimators used in practice today are designed as *static* estimators.
- Since process variables exhibit different response times, *ad hoc* dynamic compensations (e.g., lead / lag elements, delays) are often added to the static estimator.

- If the number of process measurements is too large, the dimension can be reduced through PCA(principal component analysis) or orther correlation analyses.
- In some cases where nonlinearity is judged to be significant, Artificial Neural Networks are used for regression.
- Analyzer or lab results can be used to remove bias from the soft sensor. Suppose the soft sensor takes the form of $\hat{q}_s(p,i) = f(y(p,i))$. Then,

$$\hat{q}(p,i) = \hat{q}_s(p,i) + \underbrace{\lambda(q(p,0) - \hat{q}_s(p,0))}_{\text{bias correction}}, \quad 0 \le \lambda \le 1$$

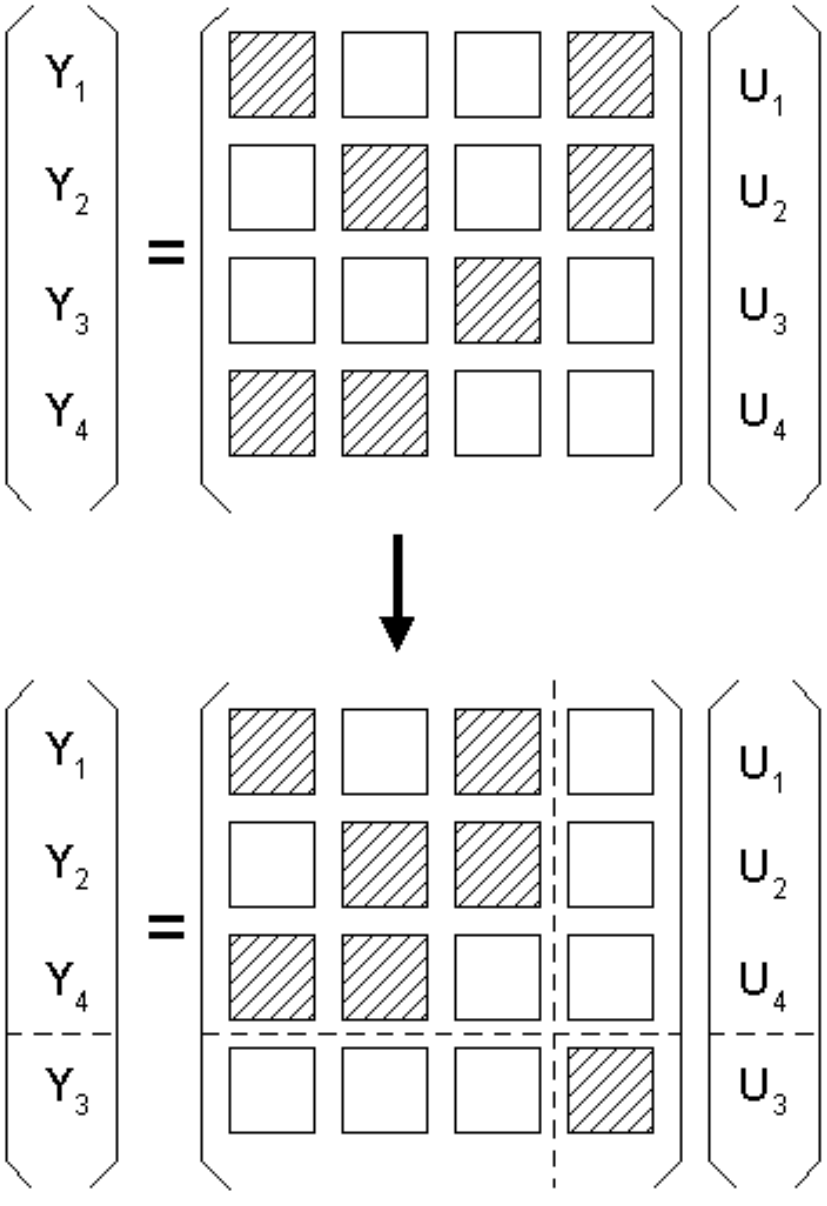


2.4.5 SYSTEM DECOMPOSITION

In MIMO processes, some input-output pairs have no or only weak coupling. Such systems can be decomposed into several subsystems and separate MPC can be designed for each subsystem.
The decentralized MPC design can reduce computational demand and improve numerical stability.

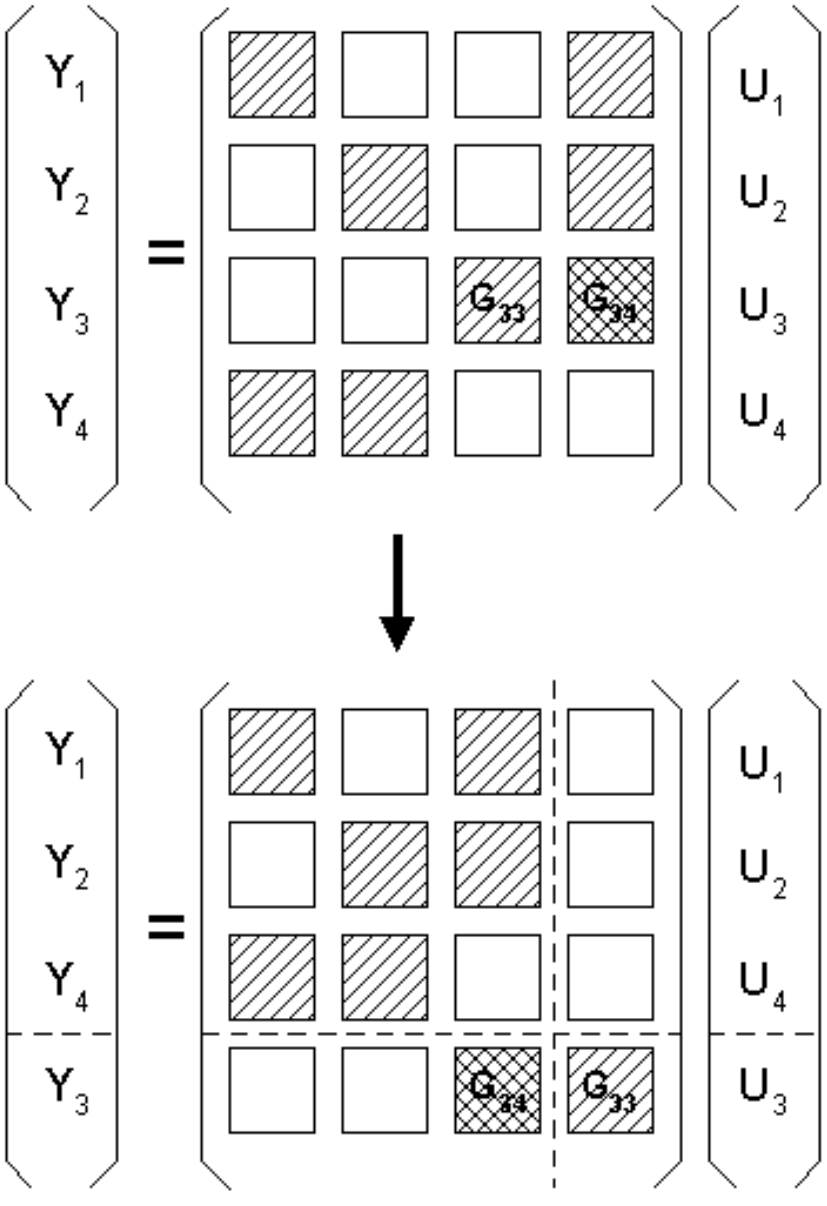
- Number of floating point computation in matrix algebra is proportional to n^2 or n^3.
- If we can decompose an n-dimensional system into two subsystems with equal size, the number of computation can be reduced from $O(n^2)$ or $O(n^3)$ to $O(n^2/4)$ or $O(n^3/8)$.
- System decomposition is not a trivial task in general. It is one of the continuing research issues studied under the title of *Control Structure Synthesis* or *Design of Decentralized Control*.
- Some of the rather obvious cases are as follows:

Case 1 : Complete Separation



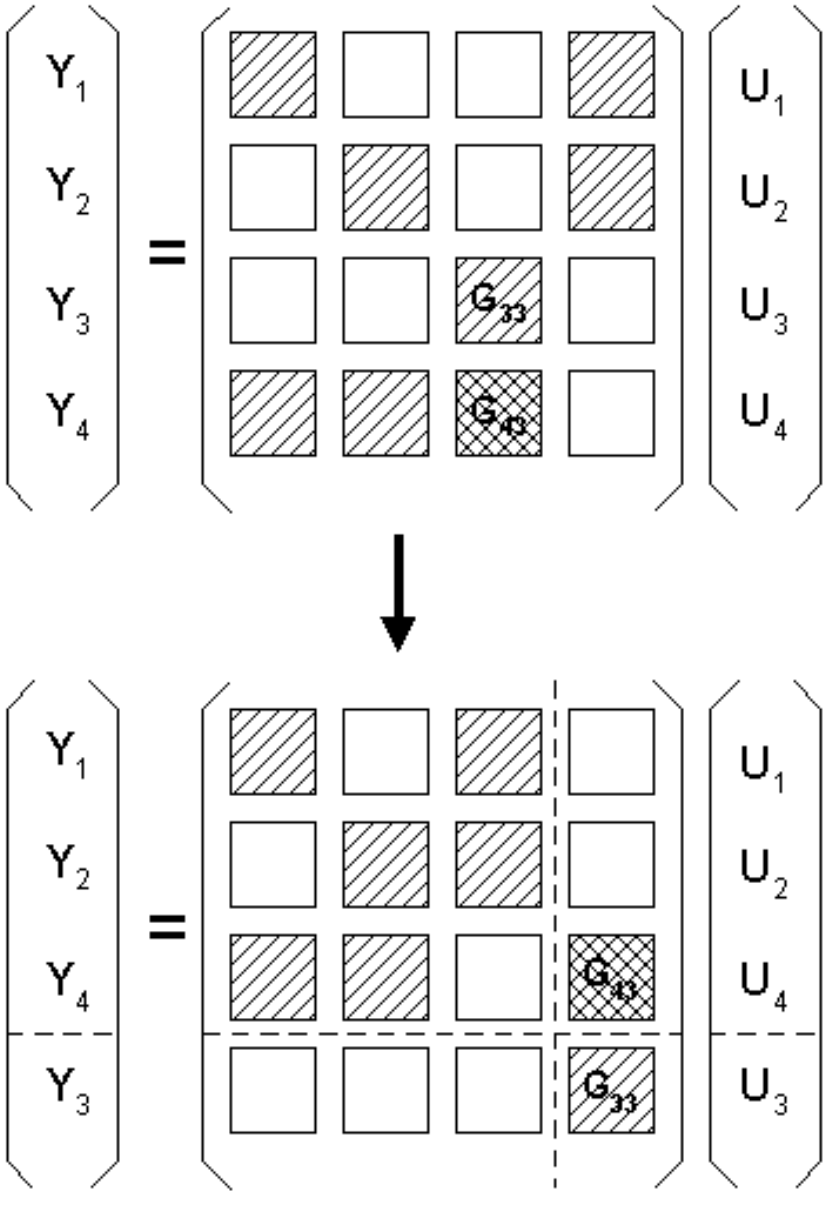
- The system can be decomposed into $(U_1\ U_2\ U_4) - (Y_1\ Y_2\ Y_4)$ and $U_3 - Y_3$ disjoint pairs.

Case 2 : Partial Separation I



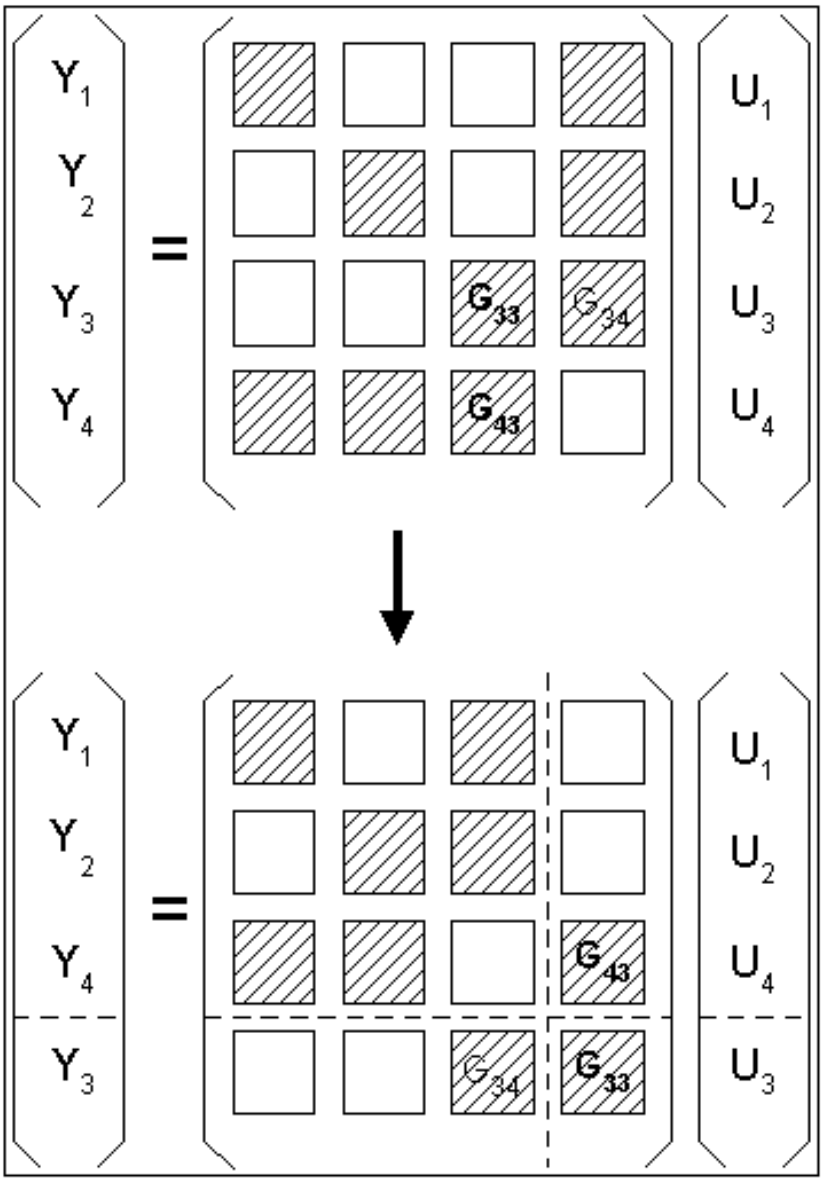
- $(Y_1\ Y_2\ Y_4)$ is not affected by U_3. But Y_3 is affected by U_4
- The system can be decomposed into two subsystems. In this case, U_4 can be treated as a measurable disturbance to the $U_3 - Y_3$ loop.

Case 3 : Partial Separation II



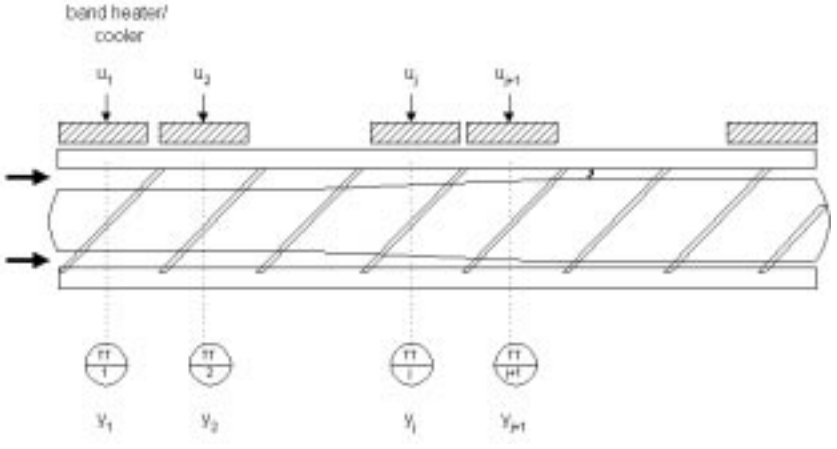
- Y_3 is not influenced by $(U_1\ U_2\ U_4)$. But, U_3 has an influence on Y_4.
- Similarly to above, the problem can be decomposed into two subproblems. U_3 acts as a measurable disturbance to the first block.

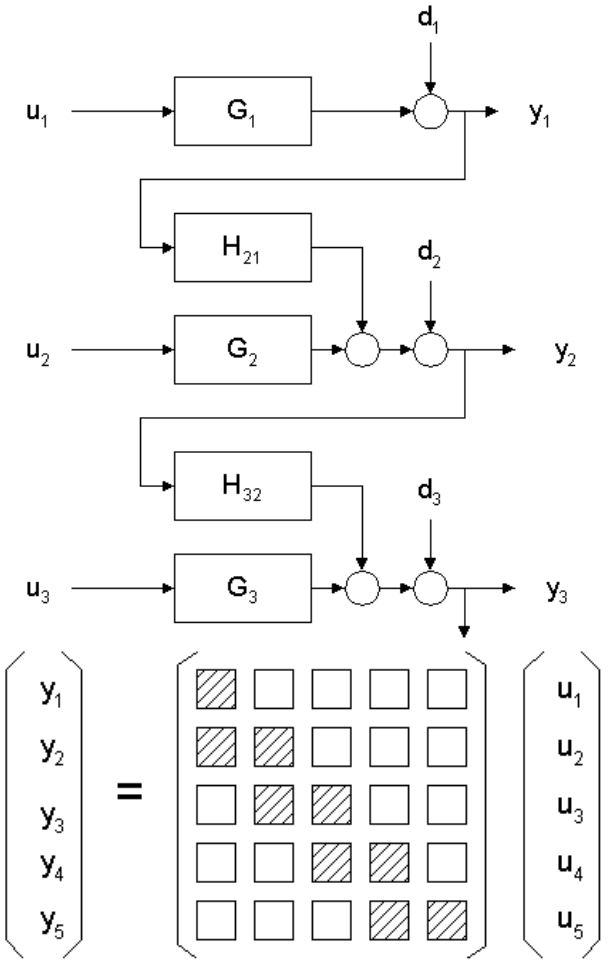
Case 4 : Partial Separation III



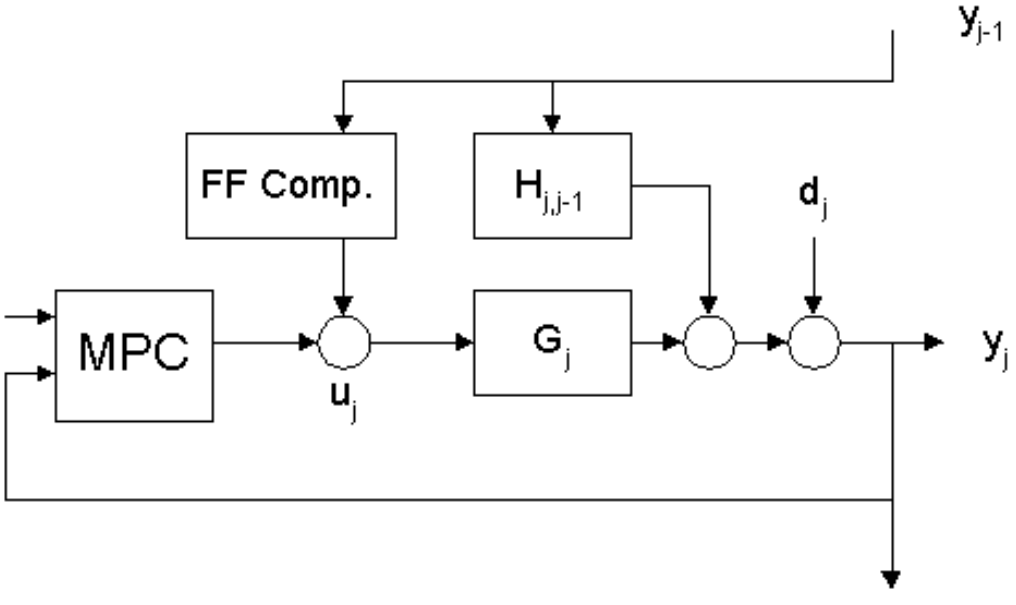
- If G_{34} and G_{43} have slower dynamics and smaller steady state gains than the other transfer functions, we may decompose the system as shown in the figure.

Example: Extrusion Process This example shows how the feedforward control can be constructed in a real process situation.





According to the input-output map, $u_j - y_j$ pair is decoupled from others while u_{j-1} plays a measurable disturbance to y_j.
Instead of treating u_{j-1} as a measured disturbance, however, it is better to take y_{j-1} as the measured disturbance and compensate its effect through the feedforward loop.



Decentralization Options

- Decentralization for both model update and optimization.
- Full model update, but decentralized optimization.
- Full model update, full steady-state optimization (LP), but decentralized dynamic optimization (QP).

2.4.6 MODEL CONDITIONING

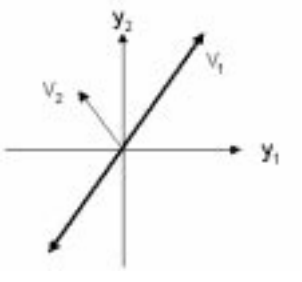
Ill-Conditioned Process ?

- Consider the following process:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 3 & 6 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} u_1 + \begin{bmatrix} 4 \\ 6 \end{bmatrix} u_2 = \begin{bmatrix} 2 \\ 3 \end{bmatrix} (u_1 + 2u_2)$$

  Two column vectors of the steady state gain matrix are colinear. As a result, $[y_1\ y_2]^T$ lies on $\mathbf{v}_1$ for any u_1 and u_2.

  If the set point is given outside $\mathbf{v}_1$, it can never be attained.



- This time, let the steady state gain matrix be

$$\begin{bmatrix} 2 & 4 \\ 3 & 6.2 \end{bmatrix}$$

  Two column vectors are nearly colinear.

  Assume that $\mathbf{y}^{sp} = (-2,\ 3)^T \perp \mathbf{v}_1$.

  The input is

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 3 & 6.2 \end{bmatrix}^{-1} \begin{bmatrix} -2 \\ 3 \end{bmatrix} = \begin{bmatrix} -61 \\ 30 \end{bmatrix}$$

  On the other hand, for $\mathbf{y}^{sp} = (2\ 3)^T = \mathbf{v}_1$, the input is

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

  - It is possible to control $\mathbf{y}$ along the $\mathbf{v}_2$ direction but a large input possibly beyond the constraints is required.
  - Does it make sense to try to control both y_1 and y_2 independently ? The answer will depend on the requirements of the process, but in many cases would be 'Not !"
  - If we give up one of the outputs, say y_2, and control only y_1 at y_1^{sp}, only a small control energy will be required. In this case, y_2 will stay at around $1.5y_1^{sp}$.

- Since the above gain matrix has a very large condition number, we say that the process is *ill-conditioned* or has a strong directionality.

Analysis using SVD

- Let

$$\mathbf{G} = [\ \mathbf{W}_1\ \mathbf{W}_2\ ] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^T \\ \mathbf{V}_2^T \end{bmatrix}$$

- Assume that $\Sigma_1 \gg \Sigma_2$ where $\dim(\Sigma_1) = m < n$,

$$\mathbf{y} = \mathbf{G}\mathbf{u} \ \Rightarrow\ \mathbf{y} \approx \mathbf{W}_1\Sigma_1\mathbf{V}_1^T\mathbf{u}$$

  The output is dominated by the modes with large singular values.

- On the other hand,

$$\mathbf{u} = \mathbf{G}^+\mathbf{y} = [\ \mathbf{V}_1\ \mathbf{V}_2\ ] \begin{bmatrix} \Sigma_1^{-1} & 0 \\ 0 & \Sigma_2^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{W}_1^T \\ \mathbf{W}_2^T \end{bmatrix} \ \Rightarrow\ \mathbf{u} \approx \mathbf{V}_2\Sigma_2^{-1}\mathbf{W}_2^T\mathbf{y}$$

  where $^+$ denotes the pseudo-inverse. The input is mostly determined by less significant modes associated with small singular values.

- SVD can be extended to a dynamic gain.

$$\mathbf{G}(j\omega) = \mathbf{W}(j\omega)\Sigma(j\omega)\mathbf{V}^T(j\omega)$$

Model Conditioning in Commercial Packages

step 0 It is assumed that operating regime for output $\mathbf{y}$ is given with priority for each output.

step 1 From the SVD of $\mathbf{G}$(steady state gain matrix), count the number of significant modes. Let it be m. Notify that $n - m$ outputs are better to be removed from the controlled variables (CVs).

step 2 Using $\mathbf{u} = \mathbf{G}^{+}\mathbf{y}$, check if the input constraints can be violated for any $\mathbf{y}$ within the defined set. If unacceptable, do the next step.

step 3 The designer takes out some of low propority outputs from CVs. Let the reduced input-output model be $\mathbf{y}_r = \mathbf{G}_r\mathbf{u}$. Repeat step 2 for the reduced model until the estimated input is acceptable for all possible $\mathbf{y}_r$.

This idea can be slightly generalized to include quantitative weighting for each output (rather than strict priority).

Model conditioning is needed not only to prevent input constraint violation (which would be automatically handled by the constrained MPC), but because low-gain directions are very difficult to identify and gains typically carry large multiplicative errors (sometimes more than 100%).

2.4.7 BLOCKING

- Consider an MPC problem with $m = 30$, $n_u = 4$. At every sampling instance, MPC has to decide 120 variables through QP. Are all these values truly impotant in the prediction of major output modes ?
- The technique to reduce the number of input decision variables while minimizing the degradation of the intended control performance is called *blocking*.
- Blocking can enhance robustness of MPC, too.

Concept of Blocking

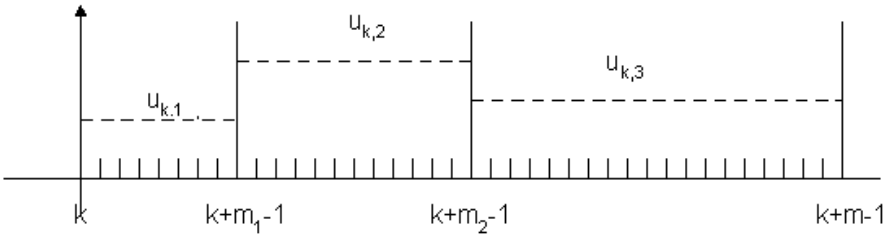
- Blocking is an approximation of the future input values by a linear combination of appropriately chosen small number of basis vectors.

$$\Delta\mathcal{U}_k \approx \mathbf{b}_1 \Delta u^*_{1k} + \mathbf{b}_b \Delta u^*_{bk} = \mathbf{B}\Delta\mathcal{U}^*_k \qquad b \ll m \times n_u$$

  where $m \times n_u$ is the dimension of $\mathcal{U}_k$.
  $\mathbf{B}$ is called a *blocking matrix*.
- QP determines $\Delta\mathcal{U}^*_k$ instead of $\Delta\mathcal{U}_k$.
- The most important step in blocking is to choose an appropriate blocking matrix.

Time-Domain Blocking - Signal Approximation

Divide the control horizon into several subintervals and decide piecewise constant input value for each subinterval.



$$\underbrace{\begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \vdots \\ \Delta u_{k+m_1-1} \\ \Delta u_{k+m_1} \\ \Delta u_{k+m_1+1} \\ \vdots \\ \Delta u_{k+m_2-1} \\ \Delta u_{k+m_2} \\ \Delta u_{k+m_2+1} \\ \vdots \\ \Delta u_{k+m-1} \end{bmatrix}}_{\Delta \mathcal{U}_k} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} \Delta u_{k,1} \\ \Delta u_{k,3} \\ \Delta u_{k,3} \end{bmatrix}}_{\Delta \mathcal{U}_k^*}$$

Rather heuristic.

Many industrial algorithms employ the above technique.

SVD-based Blocking

SVD of the pulse reponse matrix informs us which input directions excite the significant output directions.
Let the SVD of the truncated pulse response matrix over the control and prediction horizons be

$$\mathbf{H} = \begin{bmatrix} W_1 & W_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}$$

where

$$\mathbf{H} = \begin{bmatrix} h_1 & 0 & \cdots & 0 \\ h_2 & h_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ h_m & h_{m-1} & \cdots & h_1 \\ \vdots & \vdots & \vdots & \vdots \\ h_p & h_{p-1} & \cdots & h_{p-m+1} \end{bmatrix}$$

If $\Sigma_1 \gg \Sigma_2$, we can choose

$$\mathbf{B} = V_1$$

$\Rightarrow$ Approximate $\Delta\mathcal{U}_k$ as a linear combination of dominant input principal vectors.

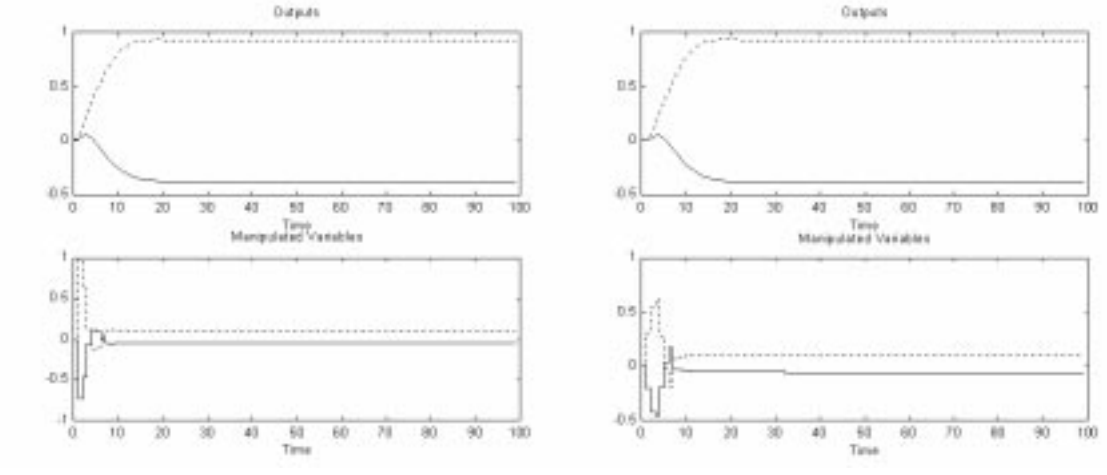
- considered to be better than the time-domain blocking in that it provides structural approximation of MIMO systems, too.

[Ex. 1] No Model Error Case

$$G(s) = G^{model}(s) = \begin{bmatrix} \frac{17}{60.48s^2+15.6s+1} & \frac{5}{19.36s^2+7.04s+1} \\ \frac{3}{10.89s^2+4.62s+1} & \frac{10}{36s^2+12s+1} \end{bmatrix}$$

$$\mathbf{Q} = \mathbf{I}, \quad \mathbf{R} = 0.01\mathbf{I}, \quad p = m = 50$$

$$-1 \leq \mathcal{U}_k \leq 1$$



Regular MPC($m \times n_u = 100$) SVD of H (b=20)

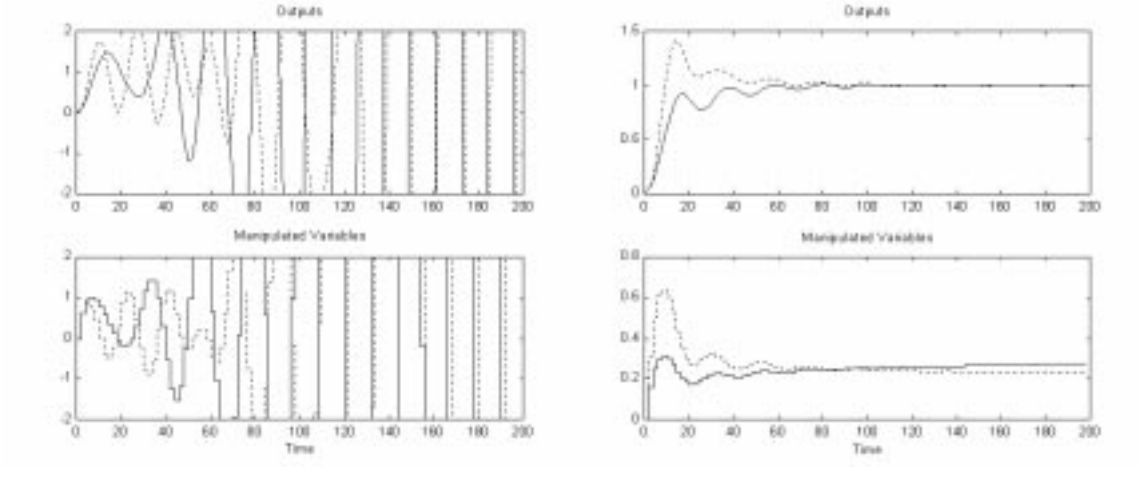
[Ex. 2] Model Error Case

$$G(s) = \begin{bmatrix} \frac{17}{(10s+1)(10s^2+s+1)} & \frac{2.3}{30s+1} \\ \frac{1.3}{20s+1} & \frac{2.8}{(10s+1)(5s^2+s+1)} \end{bmatrix}$$

$$G^{model}(s) = \begin{bmatrix} \frac{1.5}{10s+1} & \frac{2.2}{30s+1} \\ \frac{1.2}{20s+1} & \frac{2.6}{10s+1} \end{bmatrix}$$

$$\mathbf{Q} = \mathbf{I}, \quad \mathbf{R} = \mathbf{I}, \quad p = m = 90$$

No constraints



Regular MPC($m \times n_u = 180$) SVD of H (b=20)

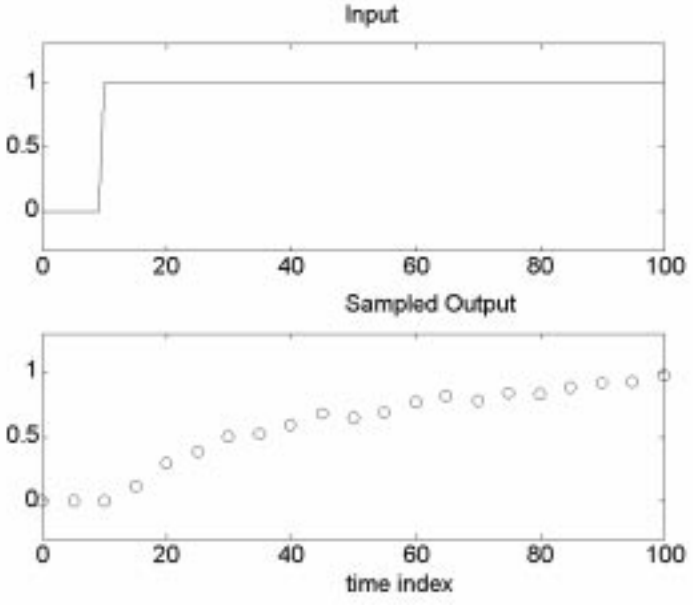
Chapter 3

SYSTEM IDENTIFICATION

The quality of model-based control absolutely relies on the quality of the model.

3.1 DYNAMIC MATRIX IDENTIFICATION

3.1.1 STEP TESTING



Procedure

1. Assume operation at steady-state with

$$\begin{aligned} \text{controlled var.(CV)}: &\quad y(t) = y_0 \quad \text{for} \quad t < t_0 \\ \text{manipulated var.(MV)}: &\quad u(t) = u_0 \quad \text{for} \quad t < t_0 \end{aligned}$$

2. Make a step change in u of a specified magnitude, Δu for

$$u(t) = u_0 + \Delta u \quad \text{for} \quad t \geq t_0$$

3. Measure $y(t)$ at regular intervals:

$$y_k = y(t_0 + kh) \quad \text{for} \quad k = 1, 2, \ldots, N$$

where

h	is the sampling interval
Nh	is approximate time required to reach steady state.

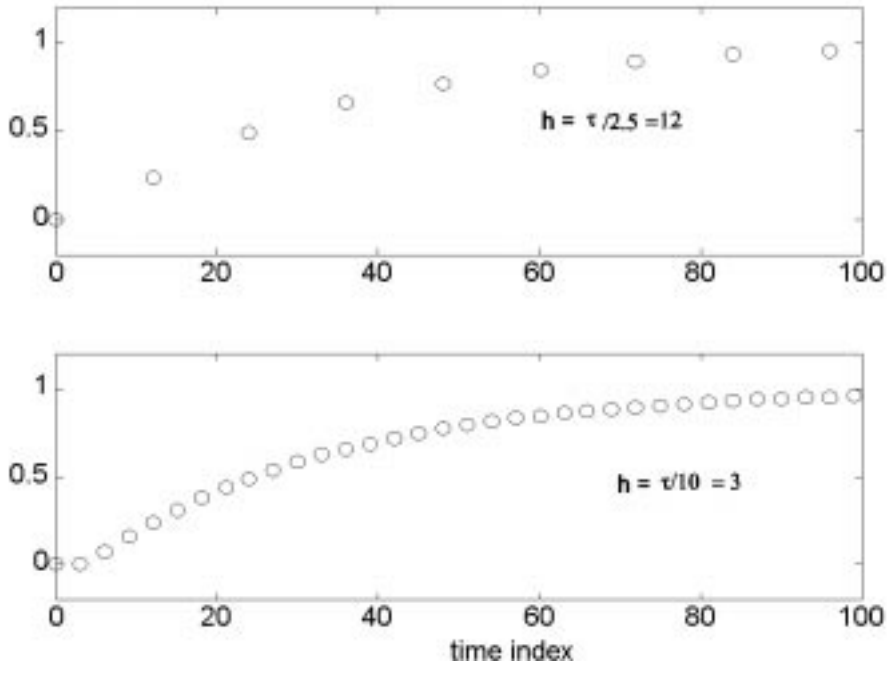
4. Calculate the *step response coefficients* from the data

$$s_k = \frac{y_k - y_0}{\Delta u} \quad \text{for} \quad k = 1, \ldots, N$$

Discussions

1. Choice of sampling period

   - For modeling, best h is one such that $N = 30 \sim 40$.
     Ex : If $g(s) = Ke^{-ds}/(\tau s + 1)$,
     then settling time $\approx 4\tau + d$
     Therefore, $h \approx \frac{4\tau+d}{N} = \frac{4\tau+d}{40} = 0.1\tau + 0.025d$



- May be adjusted depending upon control objectives.

2. Choice of step size (Δu)

   - too small :

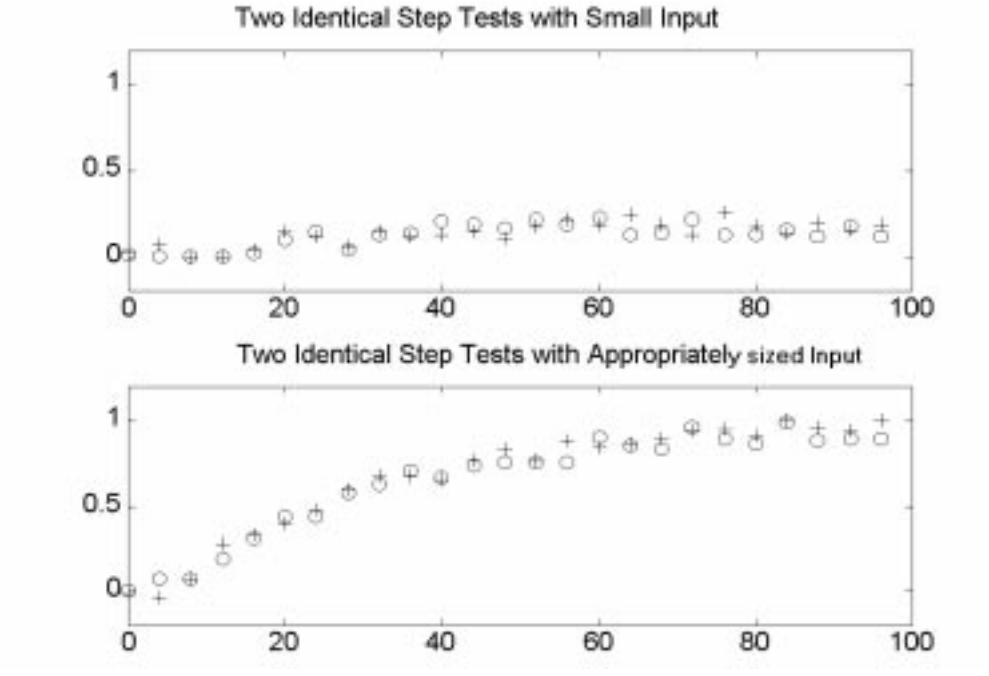
     May not produce enough output change

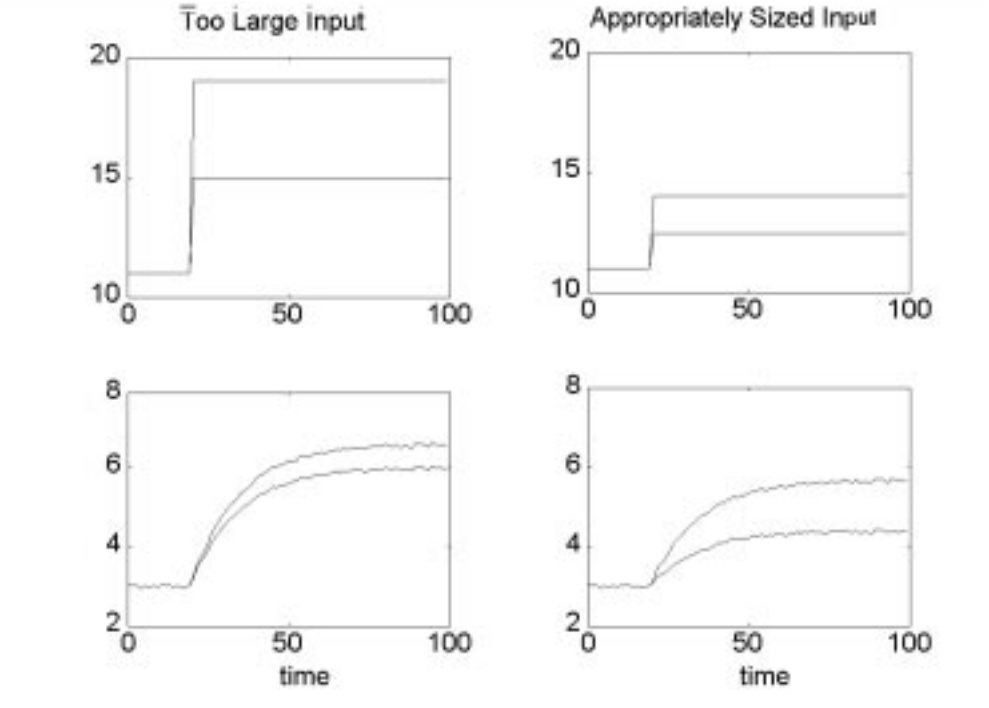
     Low signal to noise ratio

   - too big :

     Shift the process to an undesirable condition

     Nonlinearity may be induced.





- Trial and error is needed to determine the optimum step size.

3. Choice of number of experiments

- Averaging results of multiple experiments reduces impact of disturbances on calculated s_k's
- Multiple experiments can be used to check model accuracy by cross-validation.
  Data sets for Identification $\leftrightarrow$ Data set for Validation

4. An appropriate method to detect steady state is requried.
5. While the steady state (low frequency) charateristics are accurately identified, high frequency dynamics may be inaccurately charactierized.

3.1.2 PULSE TESTING

Procedure

1. Steady operation at y_0 and u_0.
2. Send a pulse of size δu lasting for 1 sampling period.
3. Calculate pulse response coefficients

$$h_k = \frac{y_k - y_0}{\delta u} \quad \text{for} \quad k = 1, \ldots, N$$

4. Calculate the step response coefficients as a cumulative sum of h_k.

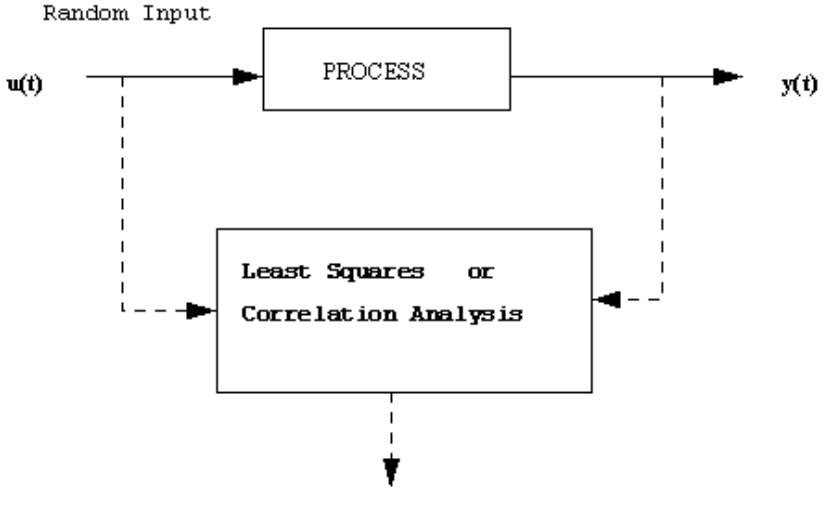
$$s_k = \sum_{i=1}^{k} h_i \quad \text{for} \quad k = 1, 2, \ldots, N$$

Discussions

1. Select h and N as for the step testing.
2. Usually need $\delta u \gg \Delta u$ for adequate S/N ratio.
3. Multiple experiments are recommended for the same reason as in the step testing.
4. An appropriate method to detect steady state is requried.
5. Theoretically, pulse is a perfect (unbiased) excitation for linear systems.

3.1.3 RANDOM INPUT TESTING

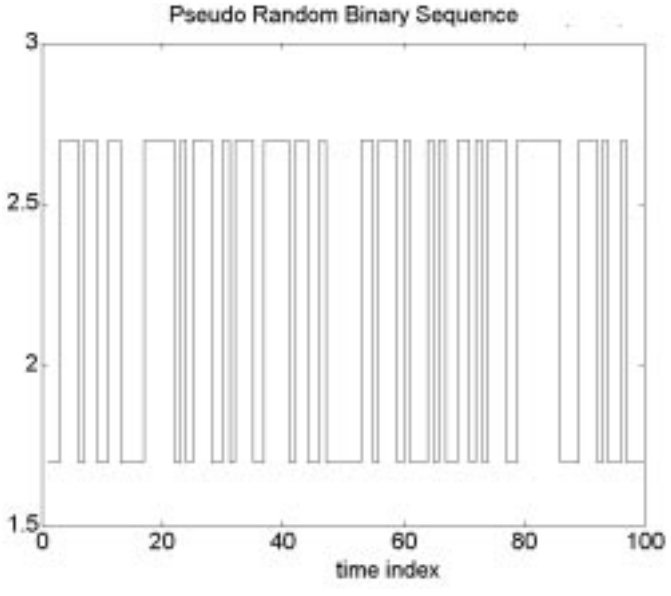
Concept



$\{h_k\}$ or $\{A, B, C\}$ or $G(s)$

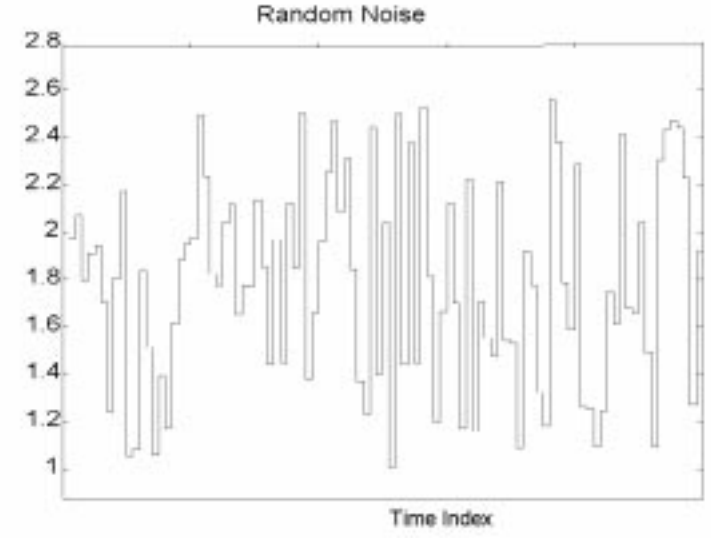
Type of Inputs

1. Pseudo-Random Binary Signal(PRBS)



In MATLAB, ≫ `u=u0+del*2*sign(rand(100,1))-0.5;`
or ≫ `u=mlbs(12);`

2. Random Noise



In MATLAB, ≫ `u=u0+del*2*(rand(100,1)-0.5);`

Data Analysis - Least Squares Fit

Given $\{u_1, u_2, \ldots, u_M\}$ and $\{y_1, y_2, \ldots, y_M\}$, determine the best fit FIR(finite impulse response) model $\{h_1, h_2, \ldots, h_N\}$.

Consider

$$y_k = h_1 u_{k-1} + h_2 u_{k-2} + \ldots + h_N u_{k-N} + d_k$$

Assume the effects of initial condition are negligible.

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} u_0 & u_{-1} & \ldots & u_{1-N} \\ u_1 & u_0 & \ldots & u_{2-N} \\ \vdots & \vdots & \vdots & \vdots \\ u_{M-1} & u_{M-2} & \ldots & u_{M-N} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_N \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_M \end{bmatrix}$$

$$\mathbf{y} = \mathbf{U}\mathbf{h} + \mathbf{d}$$

The least squares solution which minimizes

$$(\mathbf{y} - \mathbf{U}\mathbf{h})^T (\mathbf{y} - \mathbf{U}\mathbf{h}) = \sum_{i=1}^{M} \left(y_i - \sum_{j=1}^{N} h_j u_{i-j} \right)^2$$

is

$$\hat{\mathbf{h}} = \left(\mathbf{U}^T \mathbf{U}\right)^{-1} \mathbf{U}^T \mathbf{y}$$

In MATLAB, $\gg$ `hhat=y\U;`

Discussions

1. Random input testing, if appropriately designed, gives better models than the step or pulse testing does since it can equally excite low to high frequency dynamics of the process.
2. If $\mathbf{U}^T\mathbf{U}$ is singular, the inverse doesn't exist and identification fails.
   $\rightarrow$ *persistent excitation* condition.

3. When the number of coefficients is large, $\mathbf{U}^T\mathbf{U}$ can be easily singular (or nearly singular). To avoid the numerical, a regularization term is addded the the cost function. $\rightarrow$ *ridge regression*

$$\min_{\mathbf{h}} \left[(\mathbf{y}-\mathbf{U}\mathbf{h})^T(\mathbf{y}-\mathbf{U}\mathbf{h}) + \alpha\mathbf{h}^T\mathbf{h}\right]$$

$$\rightarrow \hat{\mathbf{h}} = \left(\mathbf{U}^T\mathbf{U} + \alpha\mathbf{I}\right)^{-1}\mathbf{U}^T\mathbf{y}$$

4. *Unbiasedness*: If $d(\cdot)$ and/or $u(\cdot)$ is zero-mean and $u(i)$ is uncorrelated with $d(j)$ for all (i,j) pairs (these conditions are easily satisfied.), the estimate is unbiased.

$$\hat{\mathbf{h}} = \left(\mathbf{U}^T\mathbf{U}\right)^{-1}\mathbf{U}^T\left(\mathbf{U}\mathbf{h}+\mathbf{d}\right) = \mathbf{h} + \left(\mathbf{U}^T\mathbf{U}\right)^{-1}\mathbf{U}^T\mathbf{d}$$

Since

$$E\{\left(\mathbf{U}^T\mathbf{U}\right)^{-1}\mathbf{U}^T\mathbf{d}\} = 0$$

we have

$$E\{\hat{\mathbf{h}}\} = \mathbf{h}$$

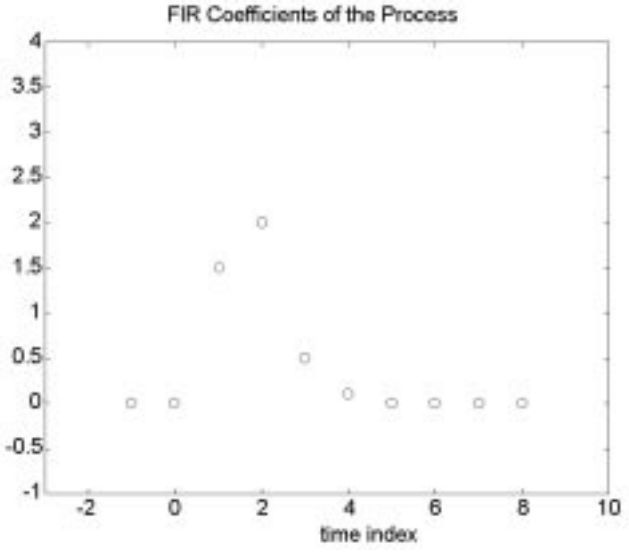
5. *Consistency*: In addition to the unbiasedness,

$$\hat{\mathbf{h}} \rightarrow \mathbf{h} \text{ (or, equivalently } E\left\{(\hat{\mathbf{h}}-\mathbf{h})(\hat{\mathbf{h}}-\mathbf{h})^T\right\} \rightarrow 0) \text{ as } M \rightarrow \infty.$$

6. Extension to MIMO identification is straightforward.
   The above properties are inherited to the MIMO case, too.

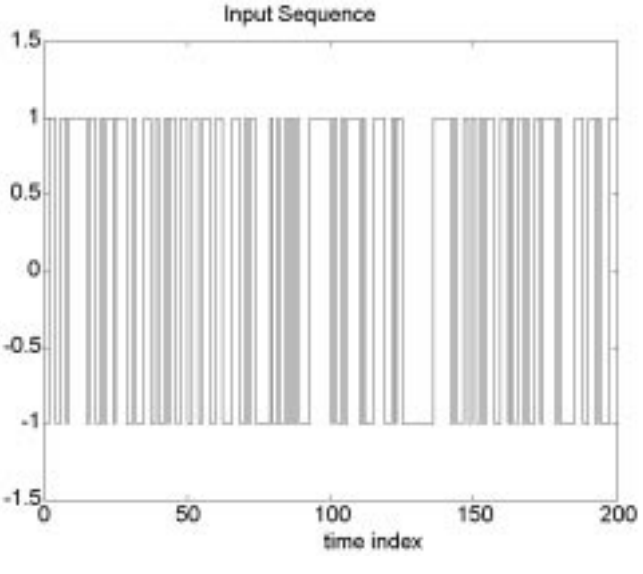
Example

• Process : $h = [h_1,\ h_2,\ h_3,\ h_4,\ h_5, \cdots] = [1.5\ 2.0\ 5.5\ 0.1\ 0\ \cdots]$

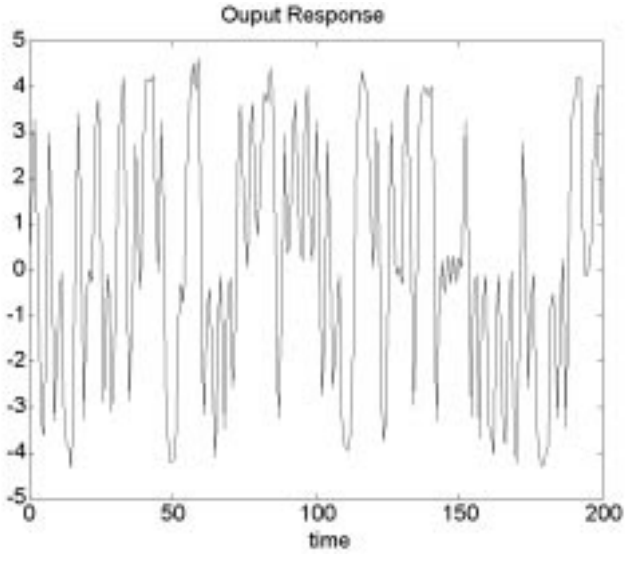


• Input : PRBS with

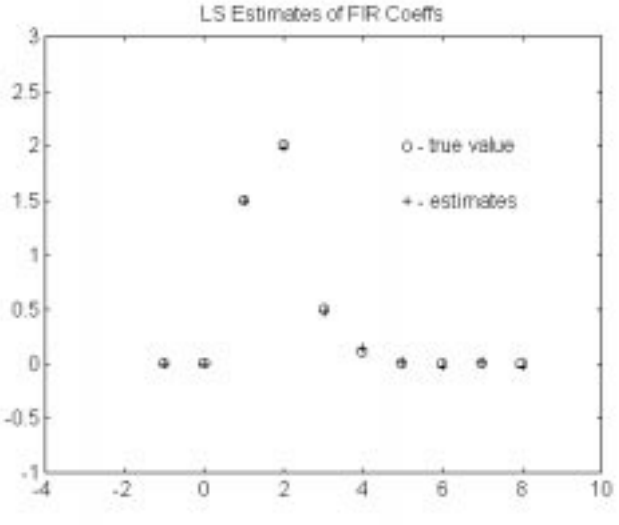
$$N = 200 \quad \bar{u} = 0$$



• The resulting output response corrupt with measurement noise with $\sigma_n^2 = 0.25^2$ is



• Estimates of $\{h_j\}$ appear as



3.1.4 DATA PRETREATMENT

The data need to be processed before they are used in identification.

(a) **Spike/Outlier Removal**

- Check plots of data and remove *obvious* outliers (*e.g.*, that are impossible with respect to surrounding data points). Fill in by interpolation.
- After modeling, plot of actual *vs* predicted output (using measured input and modeling equations) may suggest additional outliers. Remove and redo modelling, if necessary.
- But don't remove data unless there is a clear justification.

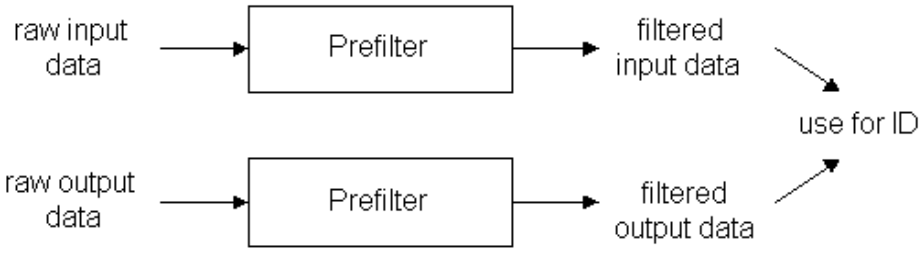
(b) **Bias Removal and Normalization**

- The input/output data are biaesd by the nonzero steady state and also by disturbance effects. To remove the bias, difference is taken for the input/output data. Then the differenced data is conditioned by scaling before using in identification.

$$\left.\begin{array}{rcl} y(k) & = & (y^{proc}(k) - y^{proc}(k-1))/c_y \\ u(k) & = & (u^{proc}(k) - u^{proc}(k-1))/c_u \end{array}\right\} \rightarrow \text{identification}$$

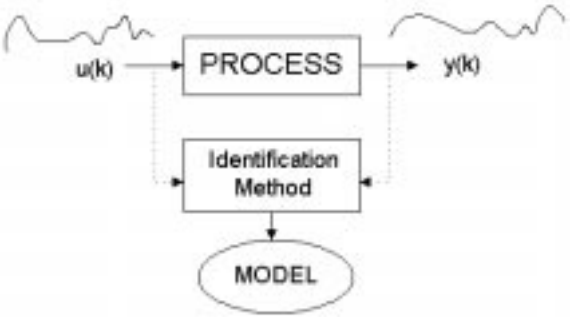
(c) **Prefiltering**

- If the data contain too much frequency components over an undesired range and/or if we want to obtain a model that fits well the data over a certain frequency range, data prefiltering (via digital filters) can be done.

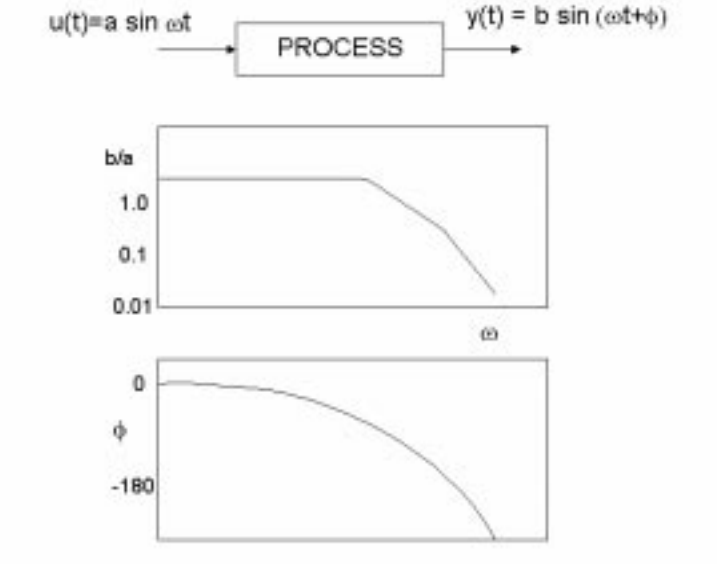


The two filters should be same.

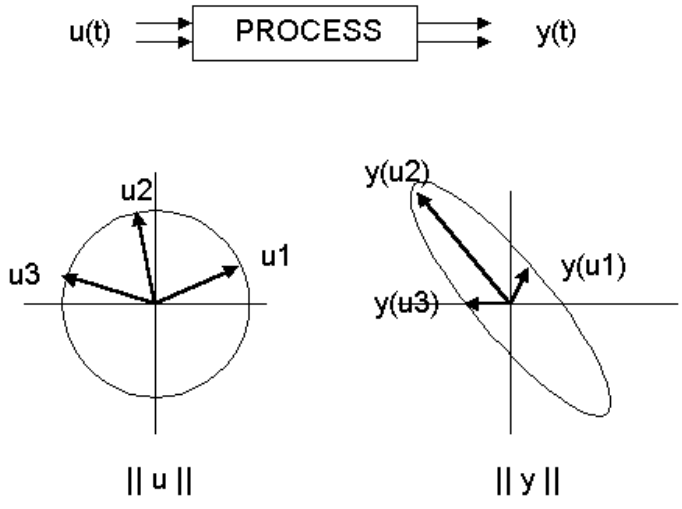
3.2 BASIC CONCEPTS OF IDENTIFICATION



- $u(k)$ is processed to $y(k)$ by the *process*, *i.e.*, $y(k)$ contains the process information. By treating $\{y_k\}$ together with $\{u(k)\}$, we can extract the process characteristics.
- A multivariable process has directional as well as frequency-dependent characteristics.



Frequency-dependent (Filtering) Characteristic of a Process



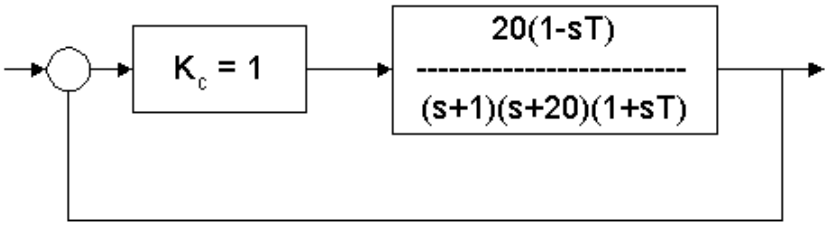
Directional Charateristics

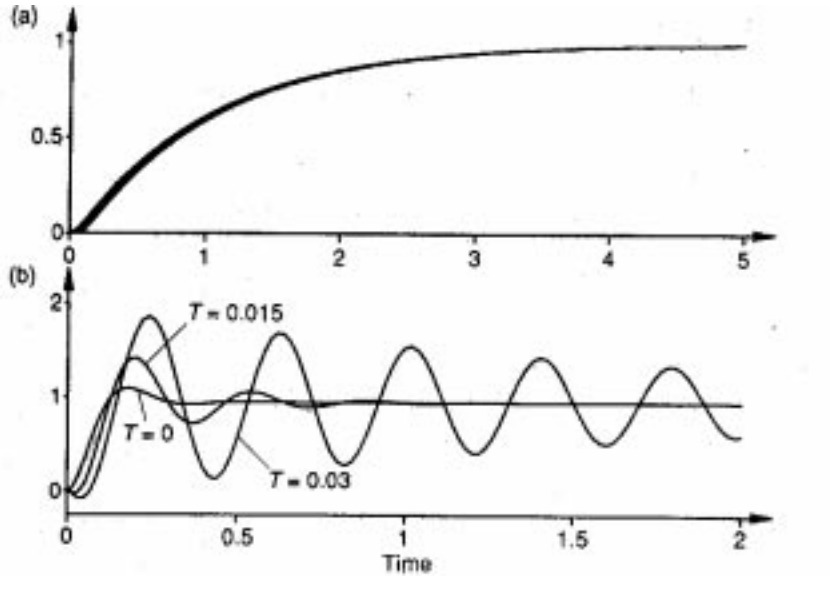
The directional gain changes with frequencies.

- To extract all the information, the input should be properly designed to excite all these charateristics uniformly.
- Furthermore, the process variables (inputs as well as outputs) are subject to various random disturbances. To remove the disturbance effects, some kind of averaging is needed. For perfect averaging, infinite number of data should be collected. The excitation input, however, has limited magnitude and duration.
- In reality, it is neither necessary nor feasible to identify all the facets of a process. Depending on the purpose of the model, some characteristics should be accurately identified while the others are not.
- To find an appropriate model to given purposes, the following three elements need to be judiciously selected and/or designed.
  - Model description
  - Experimental condition

– Identification method

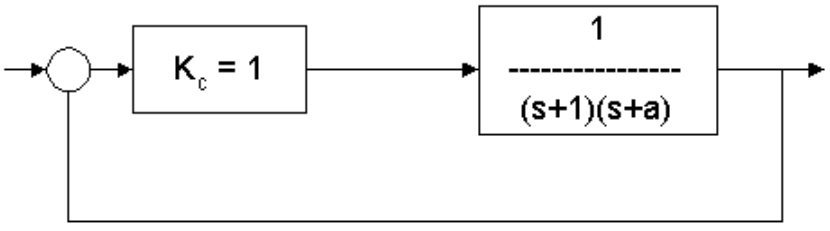
Ex. Accurate fit of the step response does not necessarily imply a good model for control.

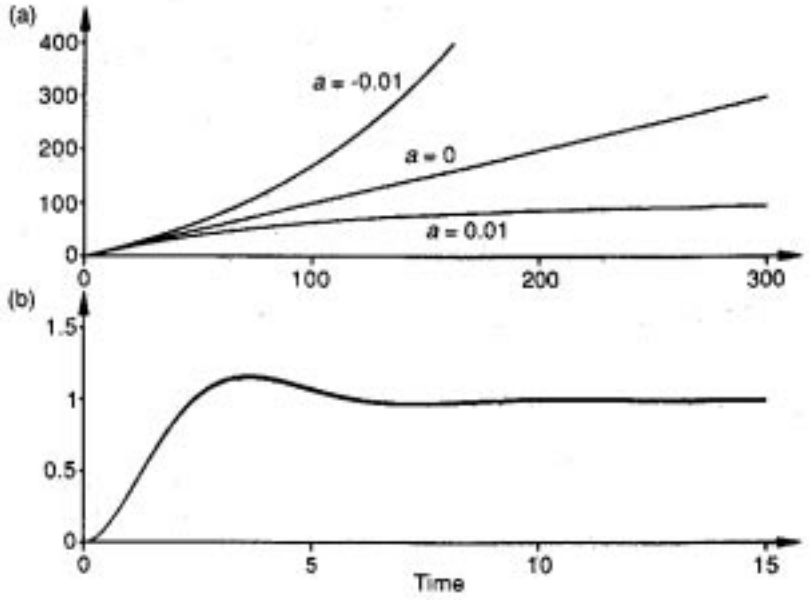




(a) open-loop step response

(b) closed-loop step response





(a) open-loop step response

(b) closed-loop step response

3.3 MODEL DESCRIPTION

3.3.1 NONPARAMETRIC MODEL

- Models that are not described by a finite number of parameters.
  - Pulse response model : $\{h_0,\ h_1,\ h_2,\ \cdots\}$
  - Step response model : $\{s_0,\ s_1,\ s_2,\ \cdots\}$
  - Frequency response model : $G(j\omega)$
- Pulse or step response models can be directly identified from pulse or step test.
- The pulse and step tests are very simple to conduct. However, the step test too much emphasizes low-frequency excitation while the pulse test gives too-widely-spread excitation over the whole frequency ranges, hence may not provide an appropriate model adequate to control purpose.
- In general, a parametric model is identified first and then converted to a nonparametric model.

3.3.2 PARAMETRIC METHOD

ARMAX Model

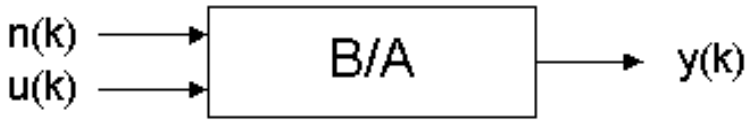
$$\begin{aligned} y(k) &= a_1 y(k-1) + \cdots + a_{n_a} y(k-n_a) + b_1 u(k-1) + \cdots + b_{n_b} u(k-n_b) \\ &+ n(k) + c_1 n(k-1) + \cdots + c_{n_c} n(k-n_c) \end{aligned}$$

or

$$\underbrace{A(q^{-1})y(k)}_{AR} = \underbrace{B(q^{-1})u(k)}_{X} + \underbrace{C(q^{-1})n(k)}_{MA}$$

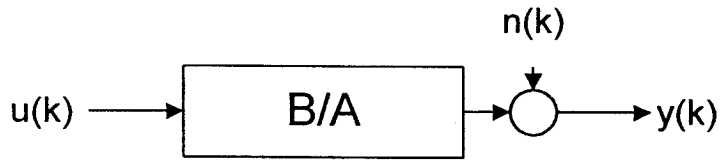
where $\{n(k)\}$ is a zero-mean i.i.d.(independent and identically distributed) sequence or, equivalently a white noise sequence.

- $(C(q^{-1})/A(q^{-1}))n(k)$ represents the disturbance model. Depending on the nature of the disturbance, a simpler form can be used.
  When the disturbance mostly occurs at the input in the form of white noise,



$$A(q^{-1})y(k) = B(q^{-1})u(k) + n(k)$$

  might be enough.
  When the disturbance mostly occurs at the output in the form of white noise,



$$y(k) = \frac{B(q^{-1})}{A(q^{-1})}u(k) + n(k) \quad \rightarrow \quad C(q^{-1}) = A(q^{-1})$$

the above output error(OE) model can be an appropriate description.

- The ARMAX model is a kind of general model that can fit most linear discrete-time dynamic systems.
- Many of the well-established classical identification methods such as least squares, extended least squares, generlaized least squares, and output error methods are based on a special case of the ARMAX model or its variants.
- Orders of each term (n_a, n_b, n_c) should be selected by the user.
  When these values are chosen less than required, parameter estimates may be biased.
  When too large values are chosen, more data points are needed for accurate parameter estimation. $\rightarrow$ Not Parsimonious
- In MIMO identification, more paramters than are necessary should be determined since no MIMO canonical ARMAX model with minimum number of paramters has not been known yet.
  A generally employed approach for MIMO system identification is to conduct indendent MISO(Multi-Input Single-Output) identification for each output and combine the results.

$$\mathbf{A}(q^{-1})\mathbf{y}(k) = \mathbf{B}(q^{-1})\mathbf{u}(k) + \mathbf{C}(q^{-1})\mathbf{n}(k) \quad \rightarrow$$

$$\begin{bmatrix} A_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_n \end{bmatrix} \begin{bmatrix} y_1(k) \\ \vdots \\ y_n(k) \end{bmatrix} = \begin{bmatrix} B_{11} & \cdots & B_{1m} \\ \vdots & \ddots & \vdots \\ B_{n1} & \cdots & B_{nm} \end{bmatrix} \begin{bmatrix} u_1(k) \\ \vdots \\ u_m(k) \end{bmatrix} + \begin{bmatrix} C_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & C_n \end{bmatrix} \begin{bmatrix} n_1(k) \\ \vdots \\ n_n(k) \end{bmatrix}$$

$$\begin{array}{rcll} A_1(q^{-1})y_1(k) & = & B_{11}(q^{-1})u_1(k) + \cdots + B_{1m}(q^{-1})u_m(k) & +C_n(q^{-1})n_1(k) \\ \vdots & \vdots & \vdots & \vdots \\ A_n(q^{-1})y_n(k) & = & B_{n1}(q^{-1})u_1(k) + \cdots + B_{nm}(q^{-1})u_m(k) & +C_n(q^{-1})n_n(k) \end{array}$$

FIR (Finite Impulse Response) Model

$$y(k) = H_1 u(k-1) + \cdots + H_{n_b} u(k-n_b) + w(k)$$

where H_i is a impulse response coefficient (matrix) and $w(k)$ is a zero-mean random noise (not necessarily i.i.d.).

- Cannot be used for description of unstable systems.
- Requires (much) more parameters than the corresponding ARMAX or state space model does.
  For description of a SISO stable system, usually 40 or more pulse response coefficients are needed if the sampling interval is appropriately chosen (not too short and too long).
- Irrespective of the nature of $w(k)$ as far as it is of zero-mean, unbiased parameter estimates can be obtained using a simple least squares method.

State space model

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w(k) \\ y(k) &= Cx(k) + v(k) \end{aligned}$$

where $\{w(k)\}$ and $\{v(k)\}$ are white noise sequences.

- Adequate to MIMO system description.
  Many useful canonical forms are well developed
- Powerful identification methods called the subspace method which directly finds a state space model in a balanced form has been recently developed.
  A version of the subspace method was commercialized by SETPOINT.

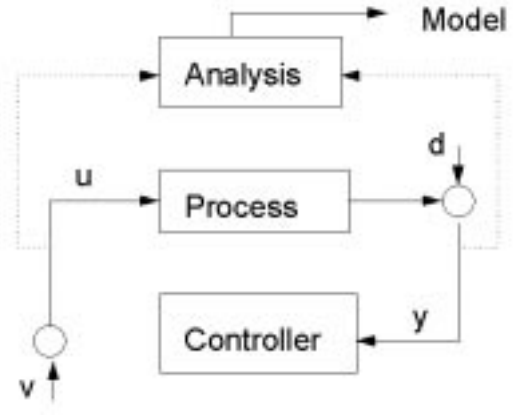
3.4 EXPERIMENTAL CONDITIONS

3.4.1 SAMPLING INTERVAL

- Too long sampling interval $\rightarrow$ too much loss of information
  Too short sampling interval $\rightarrow$ too much computation
- There are many different guidelines. $h \approx \tau/10$ is thought to be adequate for most applications.

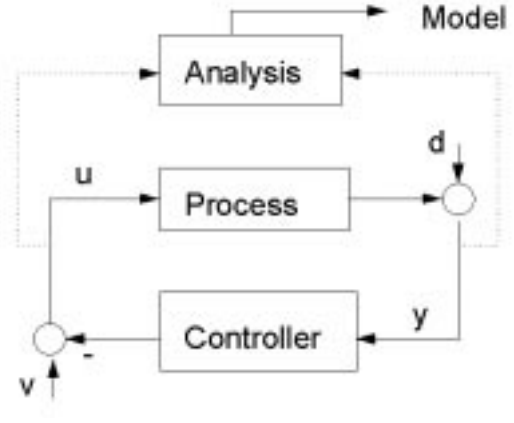
3.4.2 OPEN-LOOP VS. CLOSED-LOOP EXPERIMENTS

Open-loop experiment



u = process input
y = process output

Closed-loop experiment



u = process input, controller output
y = process output, controller input

$$\Rightarrow \quad \text{Identified Model} \approx \text{Process or } 1/\text{Controller}$$

- For nonparametric models (typically transfer functions),

$$\begin{aligned} \hat{G}_{model}(s) &\approx G_{process}(s) \text{ when } d = 0 \\ \hat{G}_{model}(s) &\approx G_{control}(s) \text{ when } v = 0 \end{aligned}$$

- For parametric models (FIR, ARMAX, State Space ..),

$$\hat{G}_{model}(s) \approx G_{process}(s)$$

if *identifiability* is satisfied.

Identifiability is *in most case* satisfied if

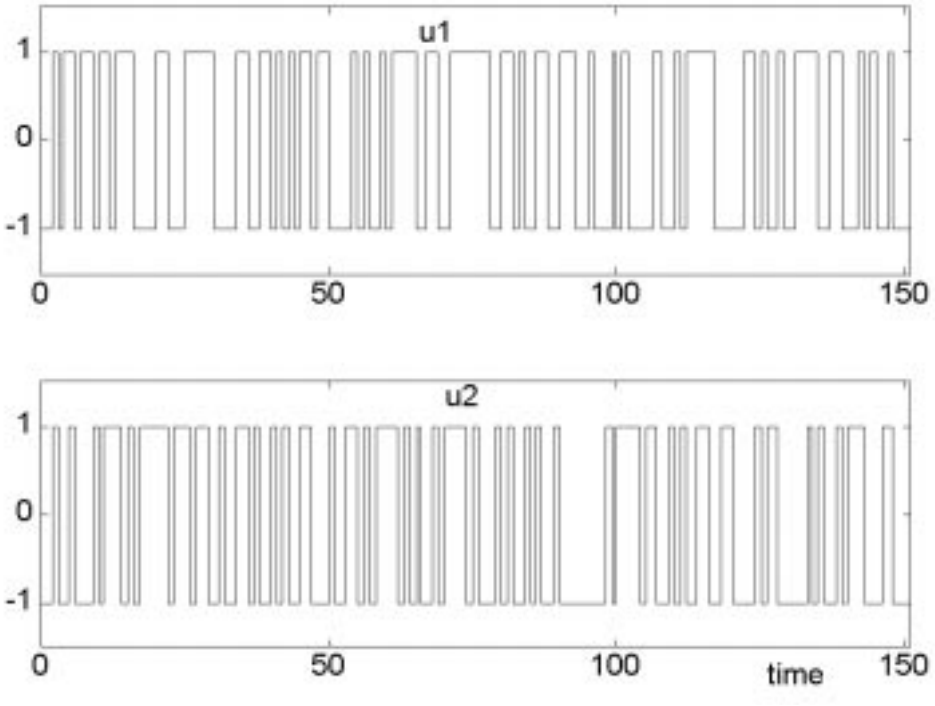
1. a FIR model is used and/or
2. a high-order controller is used and/or
3. independent excitation is given on v.

3.4.3 INPUT DESIGN

- Remember that the excitation input has limited energy with finite magnitudes over a finite duration. Hence, it is inevitable that the identified model has biased information of the process.
- Depending on the way how to distribute the input energy over different frequencies and also over different input principal directions (for MIMO cases), the identified model may have different characteristics.
- The input should preferrably be designed to sufficiently excite the system modes which are associated with the desired closed-loop performance.
  For a SISO process, process information near the crossover frequency is most important. The Ziegler-Nichols tuning method (*i.e.*, continuous

cycling method) is justified in this sense.

- In general, the PRBS(Pseudo-Random Binary Sequence) is used as an excitation signal. By adjusting the minimum step length, we can change the frequency contents in the PRBS.



3.5 IIDENTIFICATION METHODS

3.5.1 PREDICTION ERROR METHOD

- Estimate model parameters which minimizes the *optimally determined one-step-ahead output prediction error.*
- The output predictor is constructed using the model.
- The existing identification methods such as LSM, OEM, GLSM, ELSM, PLRM, MLM etc. are special cases of PEM which are derived for different model types.
  Hence, the PEM can be considered a kind of generalized framwork for system identification.

Example 1: Identification of an ARMAX process using an ARX Model

True process :

$$y(k) + \bar{a}y(k-1) = \bar{b}u(k-1) + n(k) + \bar{c}_1 n(k-1)$$

where $n(k)$ is white noise($\sim \ (0, \sigma_n^2)$).
Model :

$$y(k) + ay(k-1) = bu(k-1) + e(k) \ \text{ ARX model}$$

where $e(k)$ is assumed to be a zero-mean white noise.

Procedure

1. Given $\{y(k-1), y(k-2), \cdots\}$ and $\{u(k-1), u(k-2), \cdots\}$, the best one-step ahead output prediction is

$$\hat{y(k)} = -ay(k-1)+bu(k-1) = [\ -y(k-1)\ u(k-1)\ ] \begin{bmatrix} a \\ b \end{bmatrix} = \phi(k)^T\theta$$

2. The prediction errror at k is

$$\varepsilon(k,\theta) = y(k) - \hat{y}(k) = y(k) - \phi(k)^T\theta$$

3. θ which minimizes the sum of squared prediction error can be found as

$$\min_{\theta} \quad \sum_{k=1}^{N} \varepsilon(k,\theta)^T\varepsilon(k,\theta)$$

4. The above precedure can be rewritten in the vector form as

$$\begin{bmatrix} \varepsilon(1,\theta) \\ \vdots \\ \varepsilon(N,\theta) \end{bmatrix} = \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix} - \begin{bmatrix} \phi(1)^T \\ \vdots \\ \phi(N)^T \end{bmatrix} \theta \quad \rightarrow \quad E_N(\theta) = Y_N - \Phi_N\theta$$

$$\min_{\theta} E_N(\theta)^T E_N(\theta)$$

$$\hat{\theta}_{LS} = (\Phi_N^T\Phi_N)^{-1}\Phi_N^T Y_N$$

Discussions

- The PEM tries to seek a parameter which minimizes the prediction error. In case that the model has a different structure from the process, the parameter is determined such that the PE is minimized under its structural constraints. This usually leads to *unbiased estimate* as shown below. The process output can be written as

$$\begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} \phi(1)^T \\ \vdots \\ \phi(N)^T \end{bmatrix} \bar{\theta} + \begin{bmatrix} n(1) + \bar{c}n(0) \\ \vdots \\ n(N) + \bar{c}n(N-1) \end{bmatrix} \rightarrow Y_N = \Phi_N \bar{\theta} + V_N$$

Hence,

$$\hat{\theta} = (\Phi_N^T \Phi_N)^{-1} \Phi_N^T (\Phi_N \bar{\theta} + V_N) = \bar{\theta} + (\Phi_N^T \Phi_N)^{-1} \Phi_N^T V_N$$

Taking expectation gives

$$E\left\{\hat{\theta}_{LS}\right\} = \bar{\theta} + \underbrace{E\left[(\Phi_N^T \Phi_N)^{-1} \Phi_N^T V_N\right]}_{=\ 0\ ?}$$

Now,

$$\begin{aligned} \Phi_N^T V_N &= \begin{bmatrix} -y(0) & -y(1) & \cdots & -y(N-1) \\ u(0) & u(1) & \cdots & u(N-1) \end{bmatrix} \begin{bmatrix} n(1) + \bar{c}n(0) \\ n(2) + \bar{c}n(1) \vdots \\ n(N) + \bar{c}n(N-1) \end{bmatrix} \\ &= \begin{bmatrix} -y(0)\left(n(1) + \bar{c}n(0)\right) - y(1)\left(n(2) + \bar{c}n(1)\right) - \cdots \\ \cdots\cdots \end{bmatrix} \end{aligned}$$

Since $y(k)$ and $n(k)$ have correlation ($E\{y(k)n(k)\} = \sigma^2$),

$$E\left\{\hat{\theta}\right\} \neq \bar{\theta} \quad \rightarrow \quad \text{BIASED !!}$$

If $\bar{c} = 0$, unbased estimate !!

Example 2: Revisit of Example 1 with an ARMAX Model

This time, we consider an ARMAX model

$$y(k) + ay(k-1) = bu(k-1) + e(k) + ce(k-1)$$

- Note that $e(k)$ is not directly measurable. However, it affects $y(k)$. Hence, by treating $y(k)$ we can obtain an estimate of $e(k)$.

Procedure

1. Let the estimate of $e(k)$ be $\hat{e}(k)$.
   At $k-1$, the best one-step-ahead output prediction is

$$\begin{aligned}\hat{y}(k) &= -ay(k-1) + bu(k-1) + c\hat{e}(k-1) \\ &= \begin{bmatrix} -y(k-1) & u(k-1) & \hat{e}(k-1) \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \phi_k^T \theta\end{aligned}$$

2. As was in Ex. 1,

$$\hat{\theta} = (\Phi_N^T \Phi_N)^{-1} \Phi_N^T Y_N$$

3. In the above, $\hat{e}(k)$ can be obtained by inverting the model equation.

$$\hat{e}(k) = -\hat{c}\hat{e}(k-1) + y(k) + \hat{a}y(k-1) - \hat{b}u(k-1), \quad \hat{e}(0) = 0$$

Discussions

- To find $\hat{e}(k)$, $\hat{\theta}$ should known. On the other hand, $\hat{e}(k)$ is needed to find $\hat{\theta}$. $\rightarrow$ Nonlinear equation. Backsubstitution or other nonlinear solver is required.
- Due to the structural consistency, unbiased estimate is obtained.

General Properties of PEM

1. Need a priori infomation on the model structure (model type and orders of each term)
2. Structural inconsistency may lead to biased parameter estimates. The bias is revealed differently for differentl input excitation.
3. An ARMAX model with sufficiently large orders is OK for most applications.
   To find a parsimonious model, however, trial and error procedure with different orders is usaully necessary.
4. Generally, nonlinear equation should be solved to find an estimate. Optimumj solution is not always guaranteed.
5. Recursive (or On-line) PEM algorithms are avaliable, too
6. The PEM can be extended to MIMO identification, too.
   However, lack of an appropriate canonical form for the MIMO ARMAX model leads to an overparameterized model structure.
   The industrial practice for MIMO identification is to seperate the model into n_y MISO (multi-input single-output) susbsystems, conduct MISO identification independently, and combine the results.

$$\mathbf{A}(q^{-1})\mathbf{y}(k) = \mathbf{B}(q^{-1})\mathbf{u}(k) + \mathbf{C}(q^{-1})\mathbf{n}(k) \quad \rightarrow$$

$$\begin{bmatrix} A_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_n \end{bmatrix} \begin{bmatrix} y_1(k) \\ \vdots \\ y_n(k) \end{bmatrix} = \begin{bmatrix} B_{11} & \cdots & B_{1m} \\ \vdots & \ddots & \vdots \\ B_{n1} & \cdots & B_{nm} \end{bmatrix} \begin{bmatrix} u_1(k) \\ \vdots \\ u_m(k) \end{bmatrix} + \begin{bmatrix} C_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & C_n \end{bmatrix} \begin{bmatrix} n_1(k) \\ \vdots \\ n_n(k) \end{bmatrix}$$

$$\begin{aligned} A_1(q^{-1})y_1(k) &= B_{11}(q^{-1})u_1(k) + \cdots + B_{1m}(q^{-1})u_m(k) + C_n(q^{-1})n_1(k) \\ &\vdots \qquad\qquad \vdots \qquad\qquad\qquad\qquad \vdots \qquad\qquad\qquad\qquad \vdots \\ A_n(q^{-1})y_n(k) &= B_{n1}(q^{-1})u_1(k) + \cdots + B_{nm}(q^{-1})u_m(k) + C_n(q^{-1})n_n(k) \end{aligned}$$

7. MISO identification is in effect same as SISO identification
8. MIMO identification via MISO identification cannot take the directional characterisitcs of MIMO systems into account.

3.5.2 SUBSPACE IDENTIFICATION

- Subspace identification(SSID) is a very powerful method that has been emergered from early 90s.
- SSID
  1. is an off-line identification method (at least currently),
  2. does not require virtually any a priori information on the model structure,
  3. can be seamlessly extended to MIMO identification,
  4. provides an optimally balanced stochastic state space model, $(A,\ B,\ C)$ and noise covariances, of minimum order using input /output data.
- Moreover, SSID does not solve a nonlinear equation as in the PEM, but only relies on numerically stable linear operations.
- The ID package, IPCOS, from SETPOINT is based on a version of SSID.
- More comments will be given later.

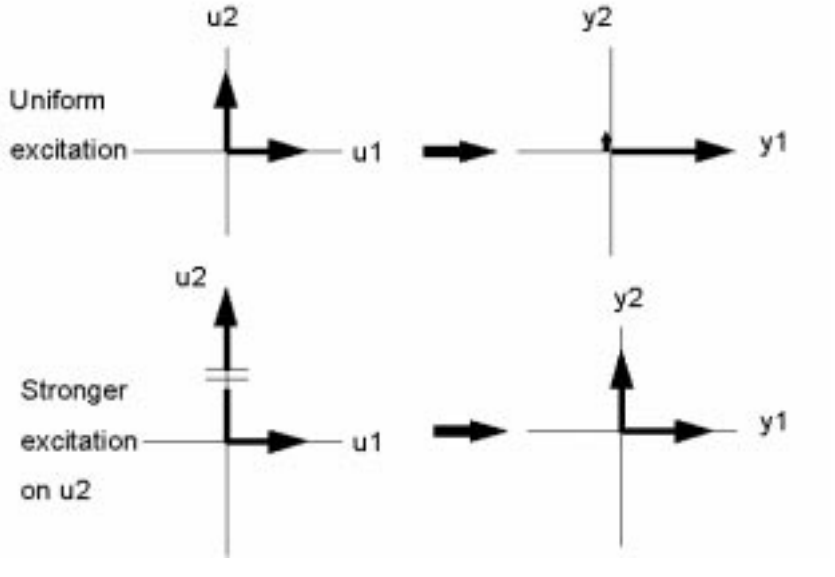
3.6 IDENTIFICATION OF A PROCESS WITH STRONG DIRECTIONALITY

Suppose that we want to control the following 2×2 system.

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \begin{bmatrix} a_1 & 0 \\ 0 & a_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$$

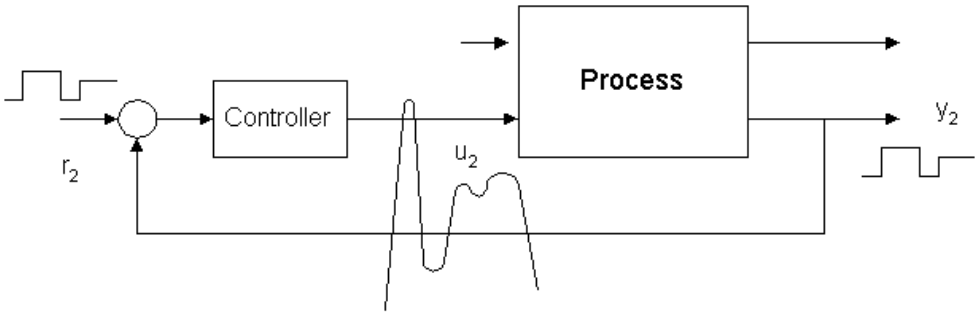
where $a_1 = 100,\ a_2 = 0.1$ and $|e_1|, |e_2| \approx 1$.
For controller design, a_1 and a_2 should be known.

- To identify a_1 and a_2, assume that we apply excitation signals of magnitude 1 to both inputs u_1 and u_2
  S/N(signal to noise ratio) for $y_2 \approx 0.1$ while S/N for $y_1 \approx 100$
  The consequence is that $\hat{a}_2$ is not correctly identified. In the worst case, the sign may be reversed.
  $\Rightarrow$ The y_2 control loop performs poorly or may be unstable.
- In order to get over the trouble,
  - we either apply large exciation to u_2 in open-loop



or

– close the y_2 loop and apply an excitation from outside the loop.



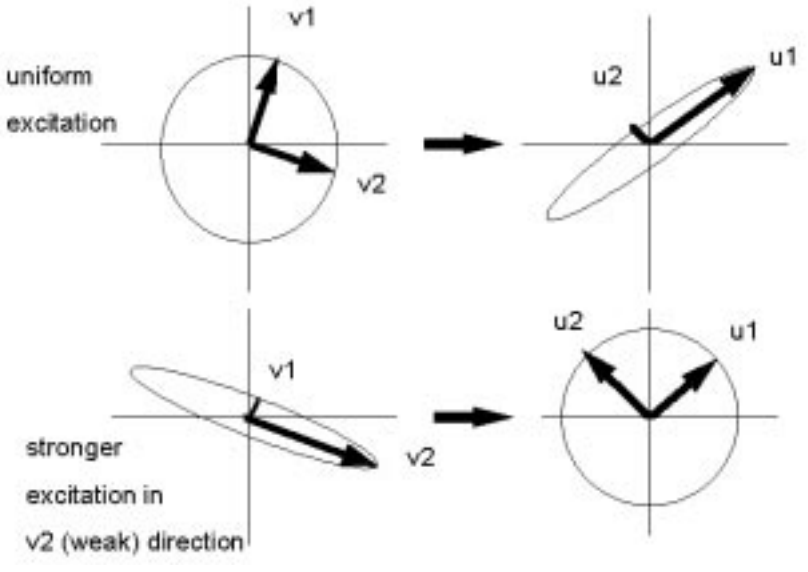
Now consider a more general case.

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \mathbf{G} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$$

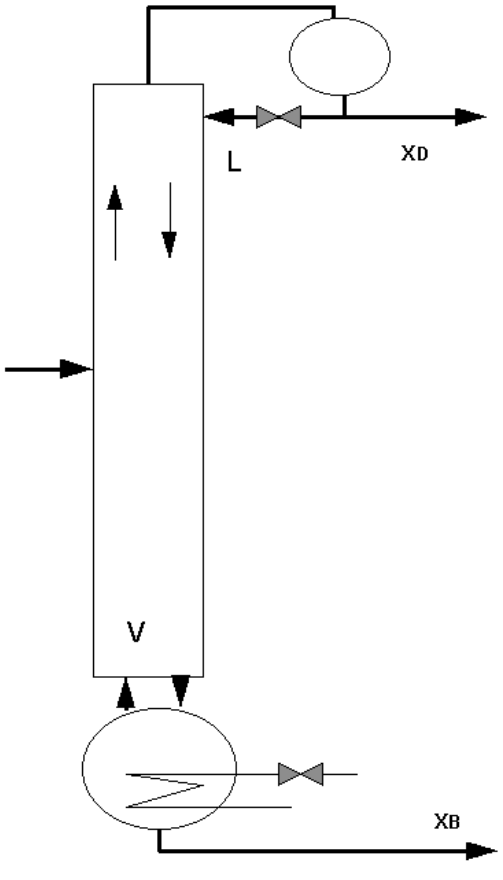
$\mathbf{G}$ is decomposed as

$$\mathbf{G} = [\mathbf{u}_1 \ \mathbf{u}_2] \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \end{bmatrix} \quad --\text{SVD}$$

Here, $\mathbf{u}_1 \perp \mathbf{u}_2$, $\mathbf{v}_1 \perp \mathbf{v}_2$, $\|\mathbf{u}_1\| = \|\mathbf{u}_2\| = \|\mathbf{v}_1\| = \|\mathbf{v}_2\| = 1$.
If $\sigma_1 \gg \sigma_2$, the same problem as before arises.
To avoid the problem, it is necessary to apply large input along the weak direction to the process either in an open-loop or a closed-loop manner.



[Example :] High Purity Binary Distillation



In high purity separation,

$$\begin{bmatrix} \Delta L \\ \Delta V \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} \Delta x_D \\ \Delta x_B \end{bmatrix} = \begin{bmatrix} +10^{-4} \\ -10^{-4} \end{bmatrix}$$

$$\begin{bmatrix} \Delta L \\ \Delta V \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \Rightarrow \begin{bmatrix} \Delta x_D \\ \Delta x_B \end{bmatrix} = \begin{bmatrix} 10^{-1} \\ 10^{-1} \end{bmatrix}$$

$$\begin{bmatrix} \Delta x_D \\ \Delta x_B \end{bmatrix} = [\mathbf{u}_1 \;\; \mathbf{u}_2] \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \Delta L \\ \Delta V \end{bmatrix}$$

where $\sigma_1 \gg \sigma_2$