A Lecture on Model Predictive Control

Jay H. Lee

School of Chemical and Biomolecular Engineering
Center for Process Systems Engineering
Georgia Inst. of Technology

Prepared for
Pan American Advanced Studies Institute Program on
Process Systems Engineering

Georgia Institute of Technology
School of Chemical & Biomolecular Engineering

Schedule

- Lecture 1: Introduction to MPC
- Lecture 2: Details of MPC Algorithm and Theory
- Lecture 3: Linear Model Identification

Lecture 1

Introduction to MPC

- Motivation
- History and status of industrial use of MPC
- Overview of commercial packages

Key Elements of MPC

- **Formulation of the control problem as an (deterministic) optimization problem**

$$\min_{u_i} \sum_{i=0}^{p} \phi_i(x_i, u_i)$$
$$g_i(x_i, u_i) \geq 0$$
$$x_{i+1} = F(x_i, u_i)$$

$\xrightarrow{?}$ $u_0 = \mu(x_0)$ HJB Eqn.

- **On-line optimization**

- **Receding horizon implementation (with feedback update)**

At $t = k$, Set $x_0 = \hat{x}_k$ (Estimated Current State)

Solve the optimization problem numerically

Implement solution u_0 as the current move.

Repeat!

Popularity of Quadratic Objective in Control

- **Quadratic objective**

$$\sum_{i=0}^{p} x_i^T Q x_i + \sum_{i=0}^{m-1} u_i^T R u_i$$

Linear State Space System Model

$$x_{k+1} = Ax_k + Bu_k$$

$$y_k = Cx_k$$

- **Fairly general**
  - State regulation
  - Output regulation
  - Setpoint tracking
- **Unconstrained linear least squares problem has an analytical solution. (Kalman's LQR)**
- **Solution is smooth with respect to the parameters**
- **Presence of inequality constraints → no analytical solution**

Classical Process Control

$$p = k_c\left(1+\frac{1}{\tau_I}\int_0^t e(t')dt'+\tau_D\frac{de}{dt}\right)$$

Ad Hoc Strategies, Heuristics

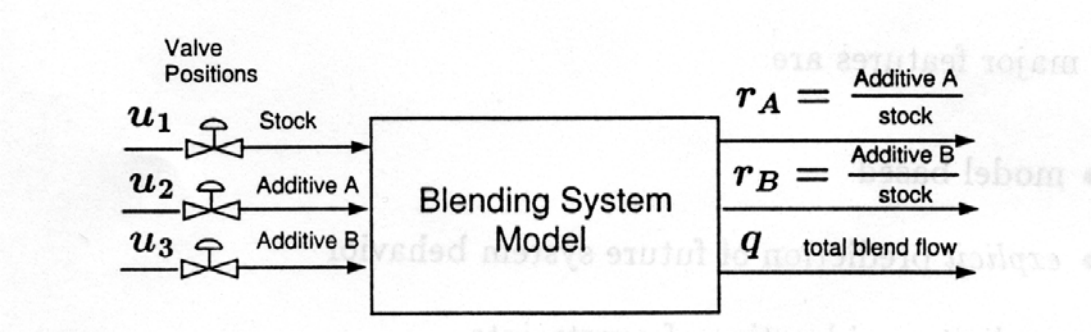
- Regulation
- Constraint handling
- Local optimization

Lead / Lag Filters
Switches
Min, Max Selectors
If / Then Logics
Sequence Logics

- **Model is not explicitly used inside the control algorithm**
- **No clearly stated objective and constraints**

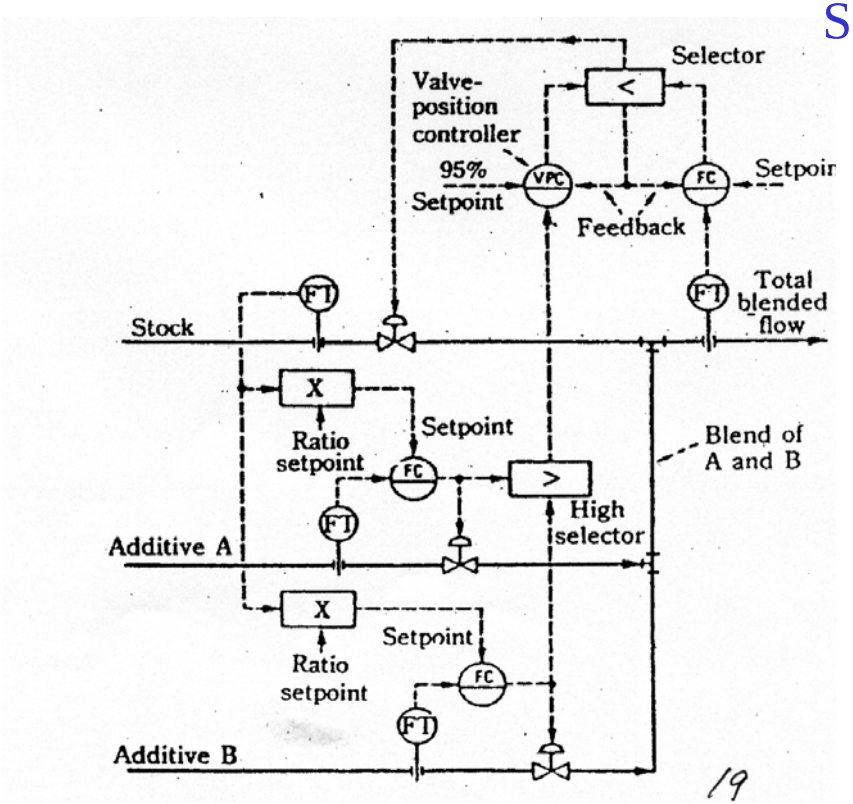
- Inconsistent performance
- Complex control structure
- Not robust to changes and failures
- Focus on the performance of a local unit

Example 1: Blending System

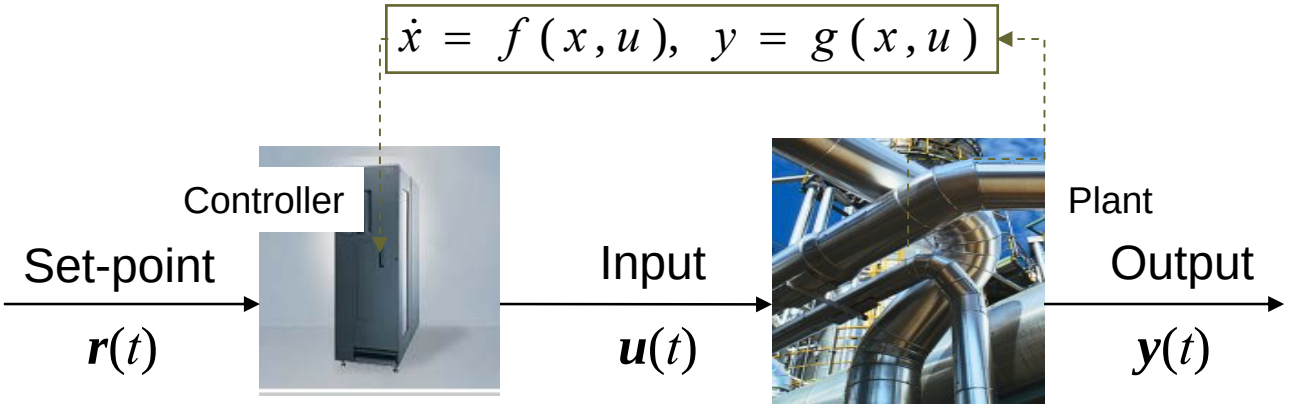


- Control r_A and r_B
- Control q if possible
- Flowrates of additives are limited

Classical Solution



Model-Based Optimal Control



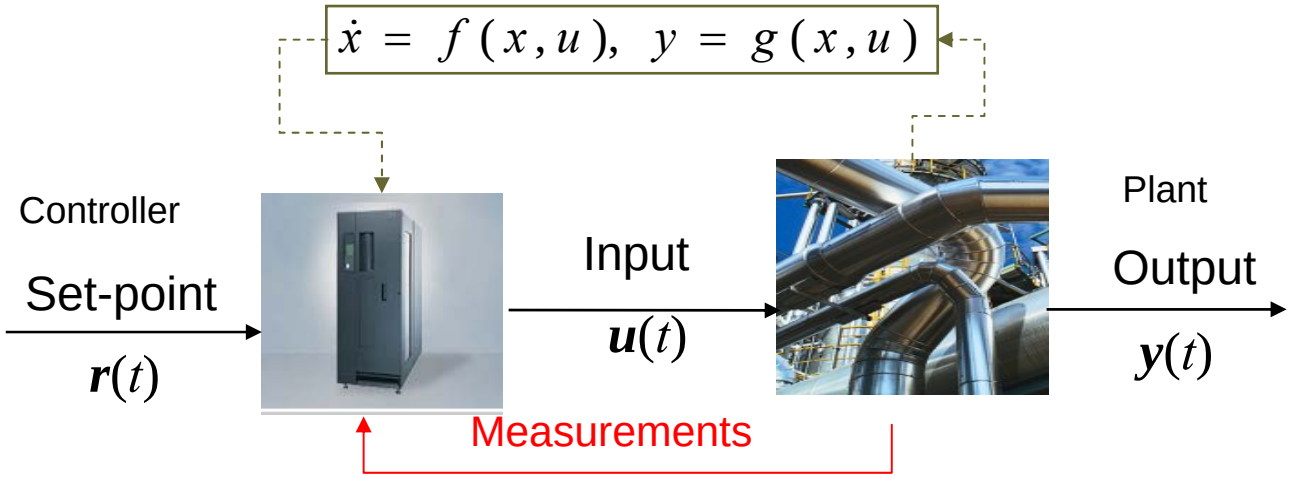
Open-Loop Optimal Control Problem

$$\min_{u_0,\ldots,u_{p-1}} \left\{ \sum_{i=0}^{p-1} \phi(x_i,u_i) + \phi_p(x_p) \right\}$$

stage-wise cost | terminal cost

$g_i(x_i,u_i) \geq 0$	Path constraints
$g_p(x_p) \geq 0$	Terminal constraints
$\dot{x} = f(x,u)$	Model constraints

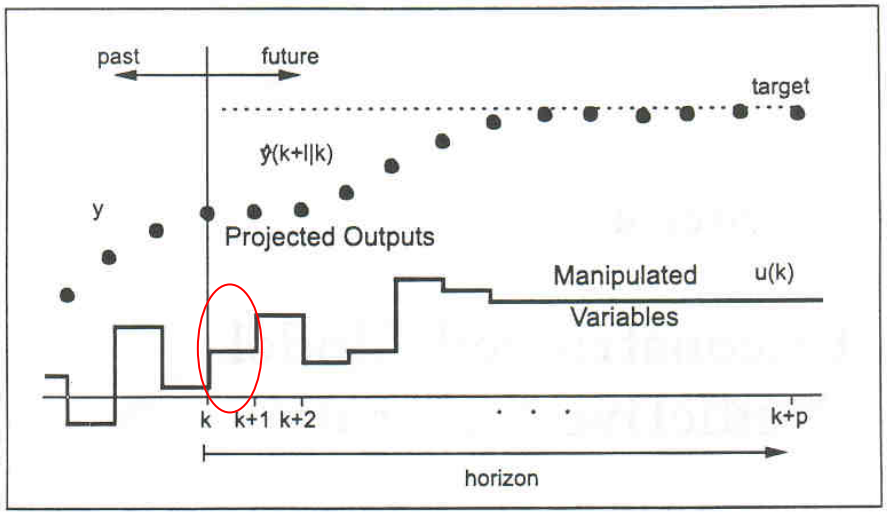
Model-Based Optimal Control



- **Open-loop optimal solution is not robust**
- **Must be coupled with on-line state / model parameter update**
- **Requires on-line solution for each updated problem**
- **Analytical solution possible only in a few cases (LQ control)**
- **Computational limitation for numerical solution, esp. back in the ’50s and ’60s**

$\dot{x} = f(x,u)$ Model constraints

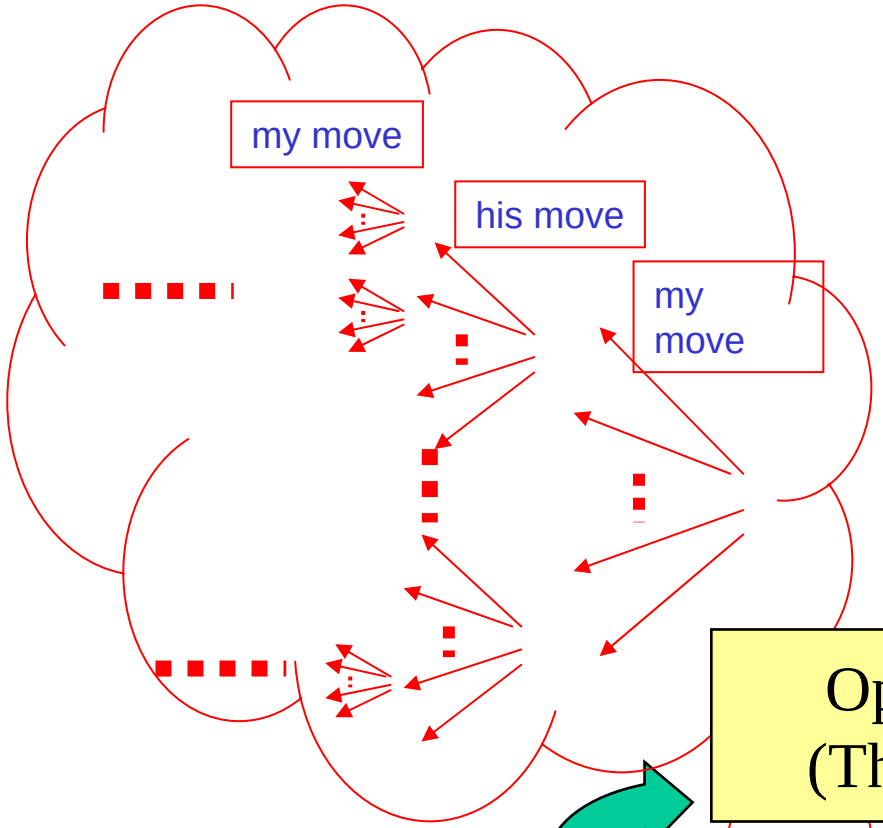
Model Predictive Control (Receding Horizon Control)



- At time k, solve the open-loop optimal control problem **on-line with $x_0 = x(k)$**
- Apply the optimal input moves **$u(k) = u_0$**
- Obtain new measurements, update the state and solve the OLOCP at time k+1 with **$x_0 = x(k+1)$**
- Continue this at each sample time

Implicitly defines the feedback law $u(k) = h(x(k))$

Analogy to Chess Playing



My
Move

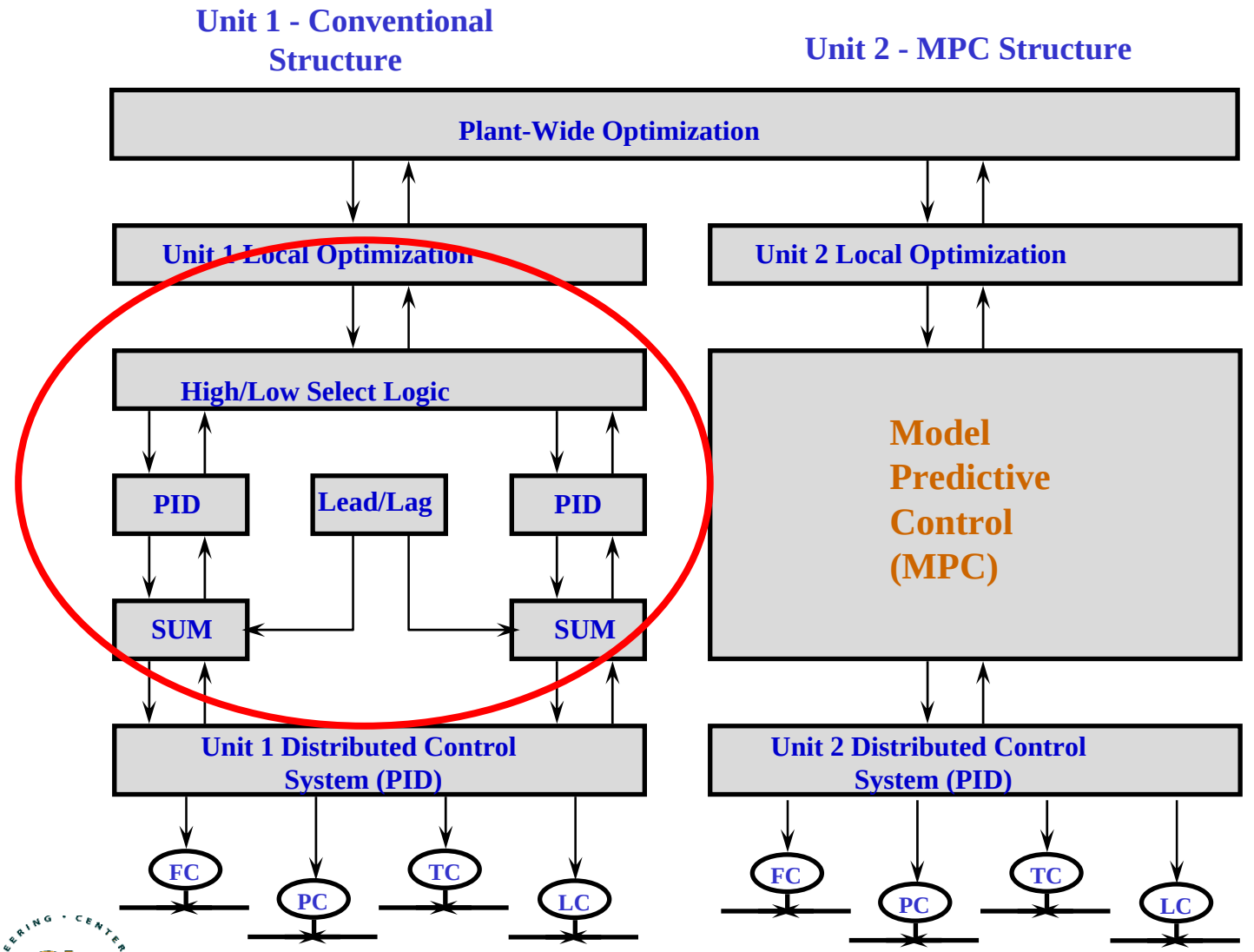
Opponent
(The Plant)

The Opponent's
Move

I
(The Controller)

New State

Operational Hierarchy Before and After MPC



Global Steady-State Optimization (every day)

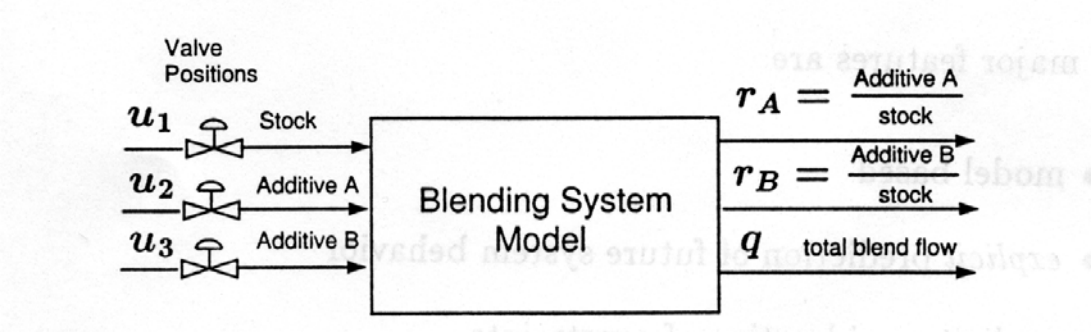
Local Steady-State Optimization (every hour)

Dynamic Constraint Control (every minute)

Supervisory Dynamic Control (every minute)

Basic Dynamic Control (every second)

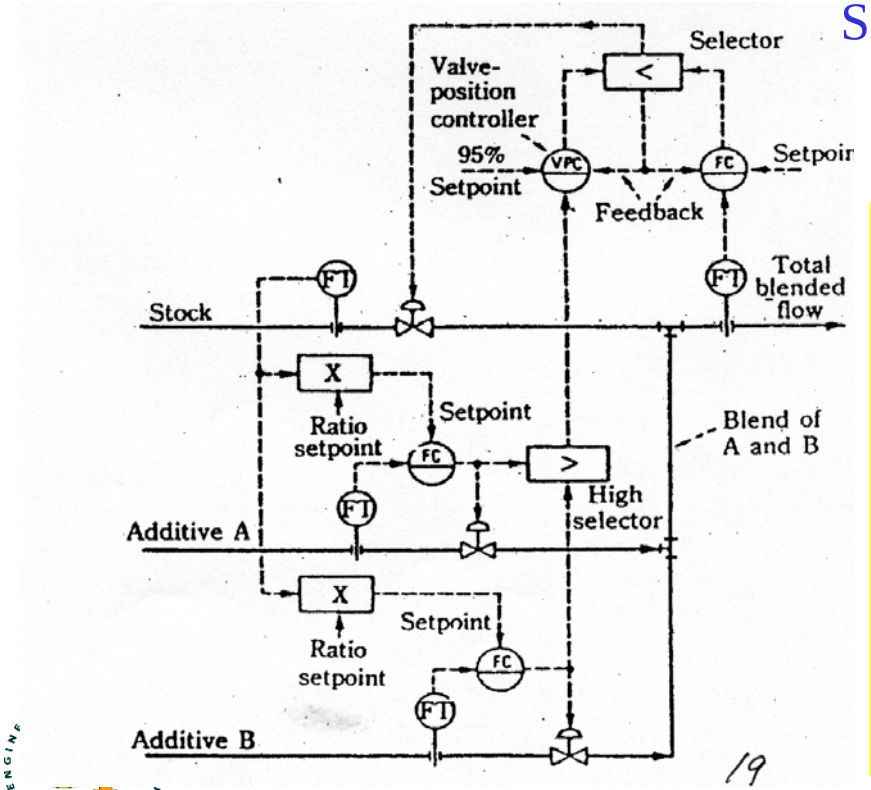
Example: Blending System



- Control r_A and r_B
- Control q if possible
- Flowrates of additives are limited

Classical Solution

MPC: Solve at each time k



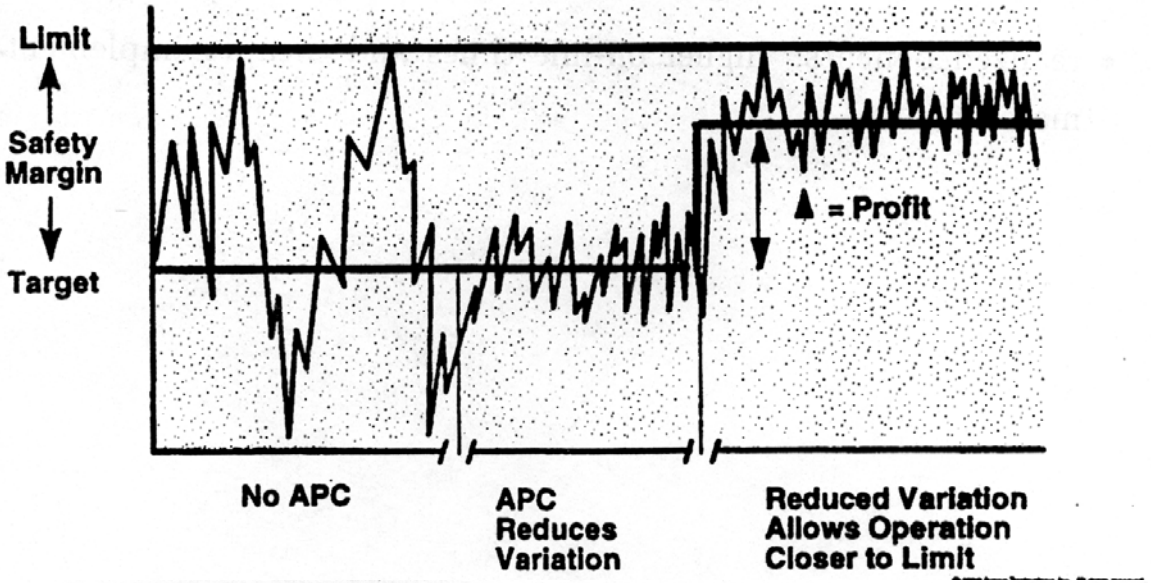
p = Size of prediction window

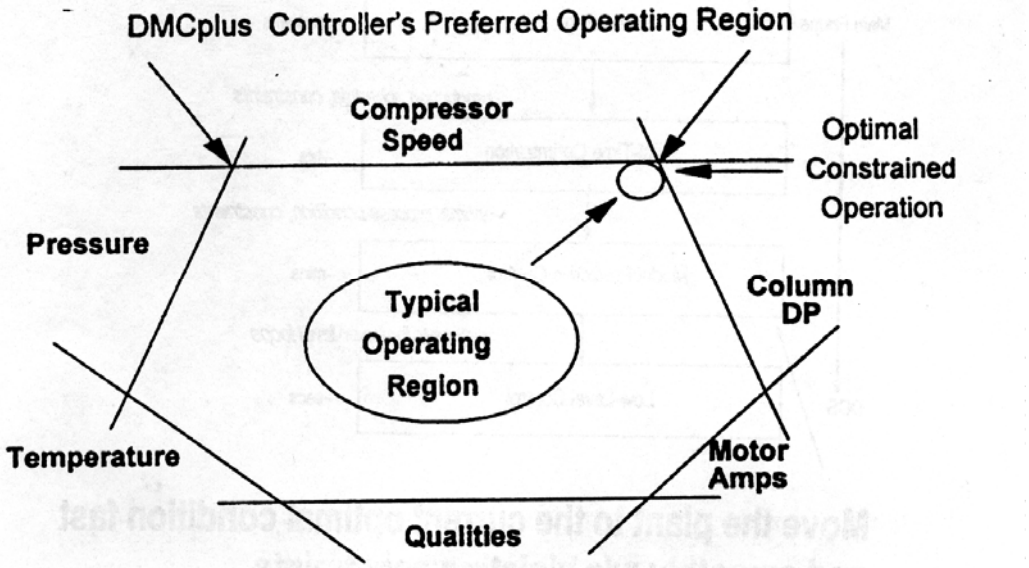
$$\min_{\substack{u_1(j),u_2(j),u_3(j)\\ j=k,\cdots,k+p-1}} \sum_{i=1}^{p} \left(r_A(k+i\,|\,k)-r_A{}^*\right)^2 + \left(r_B(k+i\,|\,k)-r_B{}^*\right)^2 + \gamma\left(q(k+i\,|\,k)-q^*\right)^2$$

$$(u_i)_{\min} \le u_i(j) \le (u_i)_{\max}, i=1,\cdots,3,$$

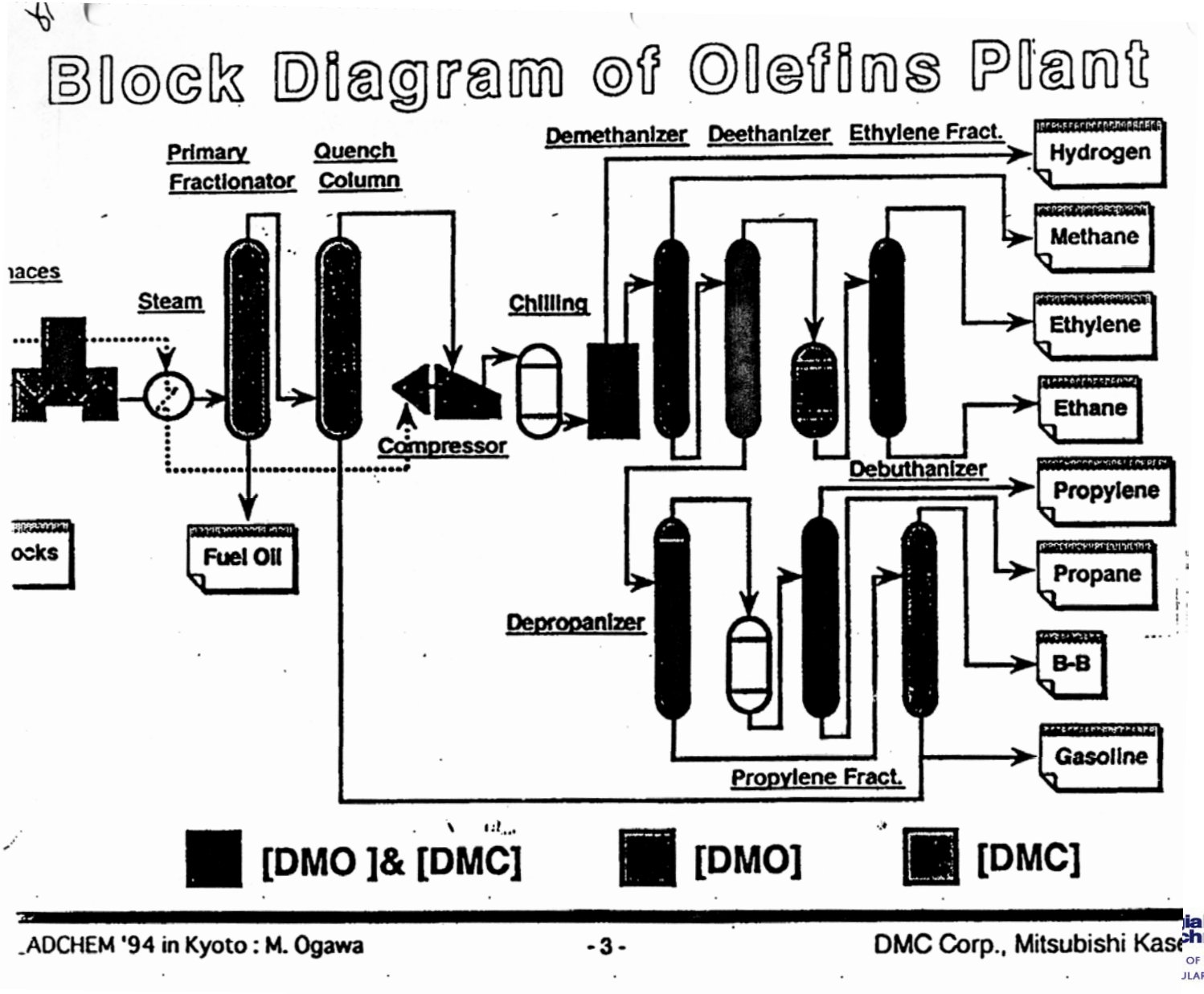
$$\gamma << 1$$

Optimization and Control

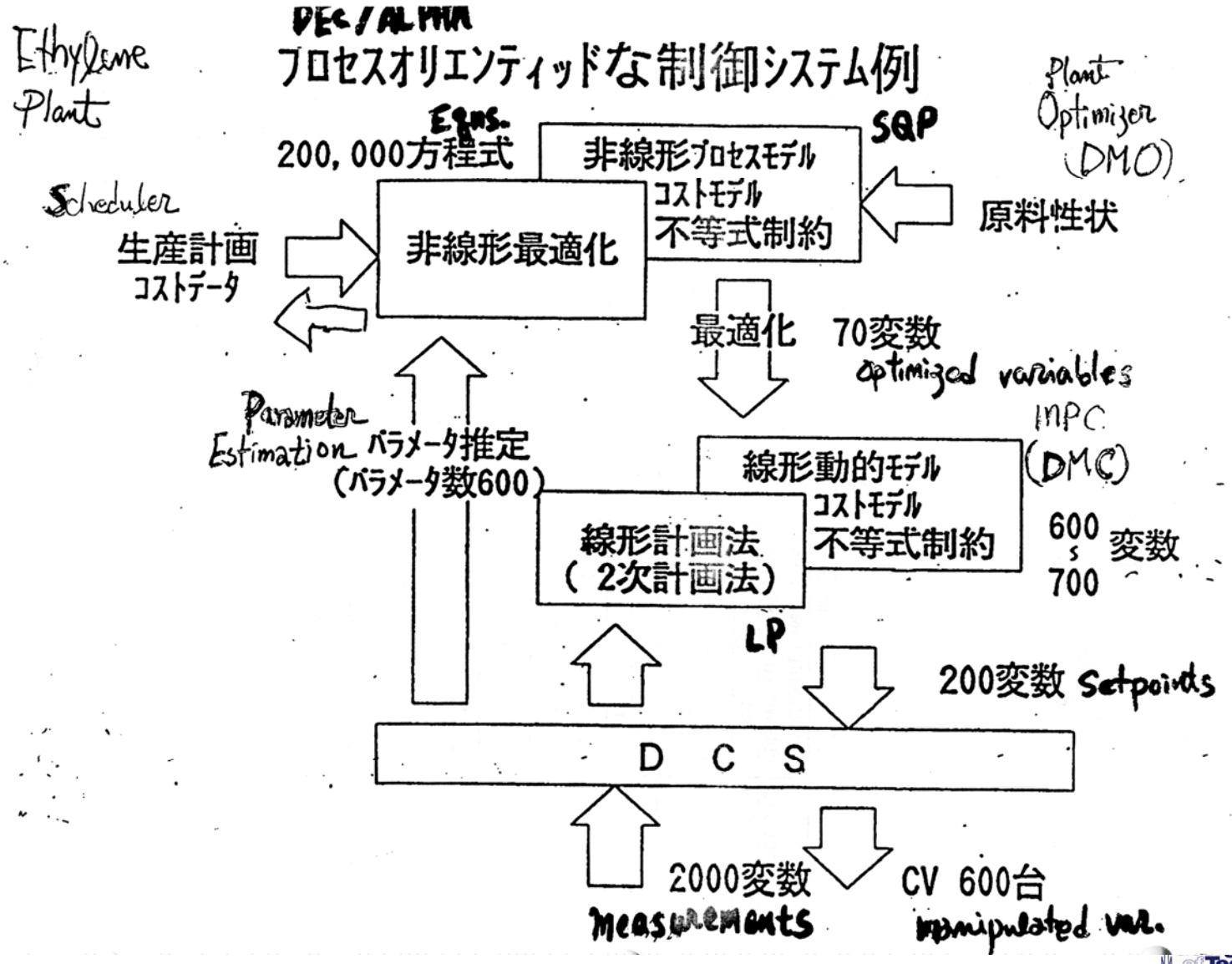




An Exemplary Application (1)



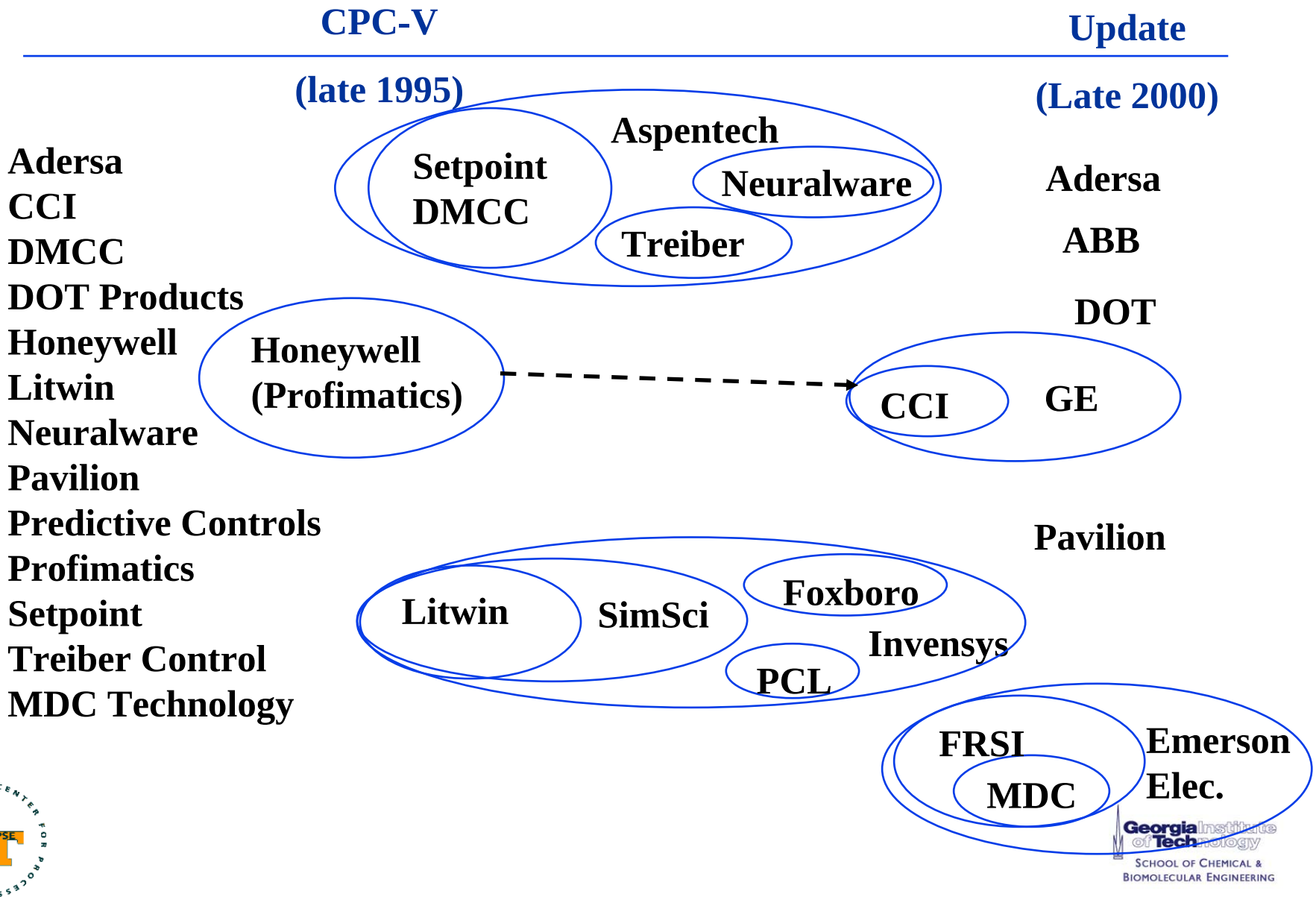
An Exemplary Application (2)



Industrial Use of MPC

- Initiated at Shell Oil and other refineries during late 70s.
- The most applied advanced control technique in the process industries.
- **>4600** worldwide installations + unknown # of “in-house” installations (Result of a survey in yr 1999).
- Majority of applications (67%) are in refining and petrochemicals. Chemical and pulp and paper are the next areas.
- Many vendors specializing in the technology
  - Early Players: DMCC, Setpoint, Profimatics
  - Today’s Players: Aspen Technology, Honeywell, Invensys, ABB
- Models used are predominantly empirical models developed through plant testing.
- Technology is used not only for **multivariable control** but for **most economic operation** within constraint boundaries.

MPC Industry Consolidation



Linear MPC Vendors and Packages

- Aspentech
  - **DMCplus**
  - **DMCplus-Model**
- Honeywell
  - **Robust MPC Technology (RMPCT)**
- Adersa
  - **Predictive Functional Control (PFC)**
  - **Hierarchical Constraint Control (HIECON)**
  - **GLIDE (Identification package)**
- MDC Technology (Emerson)
  - **SMOC (licensed from Shell)**
  - **Delta V Predict**
- Predictive Control Limited (Invensys)
  - **Connoisseur**
- ABB
  - **3d MPC**

Result of a Survey in 1999 (Qin and Badgwell)

Area	Aspen Technology	Honeywell Hi-Spec	Adersa[1]	Invensys	SGS[2]	Total
Refining	1200	480	280	25		1985
Petrochemicals	450	80	-	20		550
Chemicals	100	20	3	21		144
Pulp and Paper	18	50	-	-		68
Air & Gas	-	10	-	-		10
Utility	-	10	-	4		14
Mining/Metallurgy	8	6	7	16		37
Food Processing	-	-	41	10		51
Polymer	17	-	-	-		17
Furnaces	-	-	42	3		45
Aerospace/Defense	-	-	13	-		13
Automotive	-	-	7	-		7
Unclassified	40	40	1045	26	450	1601
Total	1833	696	1438	125	450	4542
First App.	DMC:1985 IDCOM-M:1987 OPC:1987	PCT:1984 RMPCT:1991	IDCOM:1973 HIECON:1986	1984	1985	
Largest App	603x283	225x85	-	31x12	-	

Nonlinear MPC Vendors and Packages

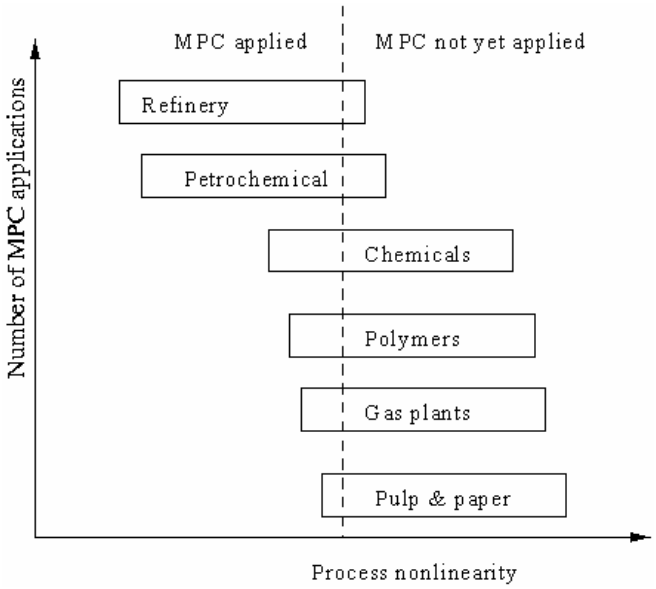
- **Adersa**
  - Predictive Functional Control (PFC)
- **Aspen Technology**
  - Aspen Target
- **Continental Controls**
  - Multivariable Control (MVC): Linear Dynamics + Static Nonlinearity
- **DOT Products**
  - NOVA Nonlinear Controller (NLC): First Principles Model
- **Pavilion Technologies**
  - Process Perfecter: Linear Dynamics + Static Nonlinearity

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}_u\mathbf{u}_k + \mathbf{B}_v\mathbf{v}_k$$

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k) = \mathbf{C}\mathbf{x}_k + \mathbf{NN}(\mathbf{x}_k)$$

Results of a Survey in 1999 for Nonlinear MPC

Area	Adersa	Aspen Technology	Continental Controls	DOT Products	Pavilion Technologies	Total
Air and Gas			18			18
Chemicals	2		15		5	22
Food Processing					9	9
Polymers		1		5	15	21
Pulp & Paper					1	1
Refining					13	13
Utilities		5	2			7
Unclassified	1		1			2
Total	3	6	36	5	43	93



Controller Design and Tuning Procedure

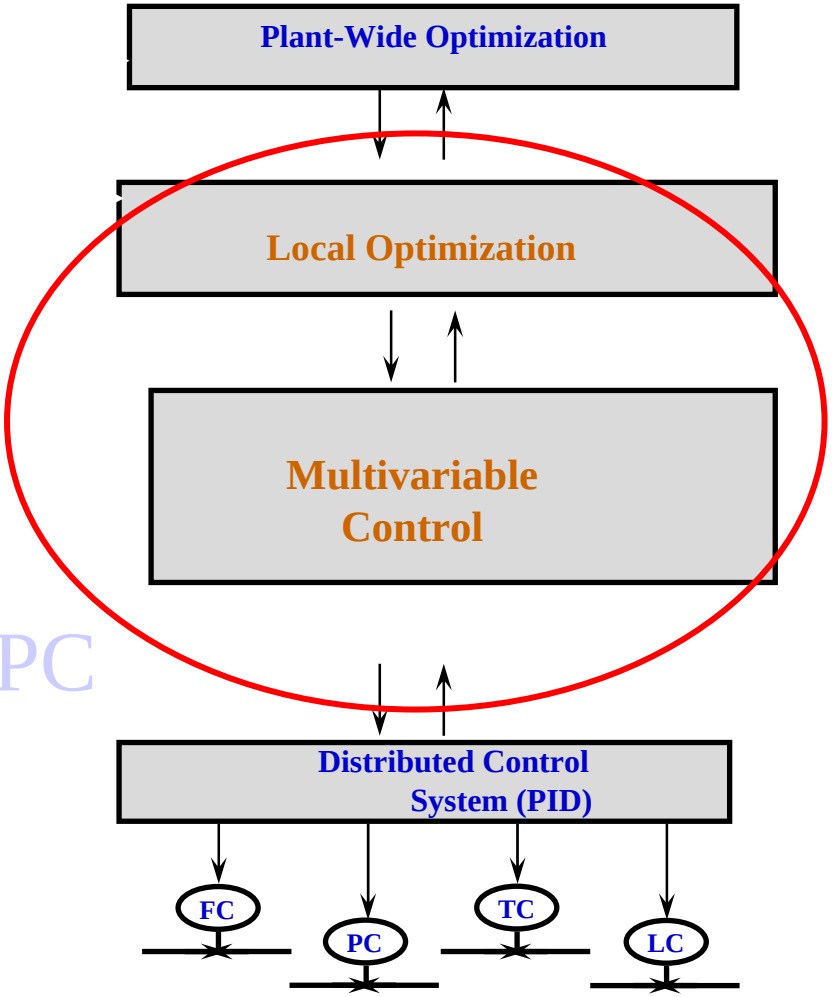
1. Determine the relevant CV's, MV's, and DV's
2. Conduct plant test: Vary MV's and DV's & record the response of CV's
3. Derive a dynamic model from the plant test data
4. Configure the MPC controller and enter initial tuning parameters
5. Test the controller off-line using closed loop simulation
6. Download the configured controller to the destination machine and test the model predictions in *open-loop* mode
7. Commission the controller and refine the tuning as needed

Role of MPC in the Operational Hierarchy

Determine **plant-wide the optimal operating condition** for the day

Make **fine adjustments** for local units

Take each local unit to the optimal condition **fast but smoothly without violating constraints**



Local Optimization

- A separate steady-state optimization to determine steady-state targets for the inputs and outputs; RMPCT introduced a dynamic optimizer recently
- Linear Program (LP) for SS optimization; the LP is used to enforce input and output constraints and determine optimal input and output targets for the thin and fat plant cases
- The RMPCT and PFC controllers allow for both linear and quadratic terms in the SS optimization
- The DMCplus controller solves a sequence of separate QPs to determine optimal input and output targets; CV's are ranked in priority so that SS control performance of a given CV will never be sacrificed to improve performance of lower priority CV's; MV's are also ranked in priority order to determine how extra degrees of freedom is used

Dynamic Optimization

At the dynamic optimization stage, all of the controllers can be described (approximately) as minimizing a performance index with up to three terms; an output penalty, an input penalty, and an input rate penalty:

$$J = \sum_{j=1}^{P} \left\| \mathbf{e}^{y}_{k+j} \right\|^2_{\mathbf{Q}_j} + \sum_{j=0}^{M-1} \left\| \Delta \mathbf{u}_{k+j} \right\|^2_{\mathbf{S}_j} + \sum_{j=0}^{M-1} \left\| \mathbf{e}^{u}_{k+j} \right\|^2_{\mathbf{R}_j}$$

A vector of inputs $\mathbf{u}^{\mathbf{M}}$ is found which minimizes *J* subject to constraints on the inputs and outputs:

$$\mathbf{u}^{\mathbf{M}} = \left(\mathbf{u}_0^T, \mathbf{u}_1^T, \ldots \mathbf{u}_{M-1}^T \right)^T$$

$$\mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k \right)$$

$$\mathbf{y}_{k+1} = g\left(\mathbf{x}_{k+1} \right) + \mathbf{b}_{k+1}$$

$$\underline{\mathbf{u}} \le \mathbf{u}_k \le \overline{\mathbf{u}}$$

$$\underline{\Delta \mathbf{u}} \le \Delta \mathbf{u}_k \le \Delta \overline{\mathbf{u}}$$

$$\underline{\mathbf{y}} \le \mathbf{y}_k \le \overline{\mathbf{y}}$$

Georgia Institute of Technology
School of Chemical & Biomolecular Engineering

Dynamic Optimization

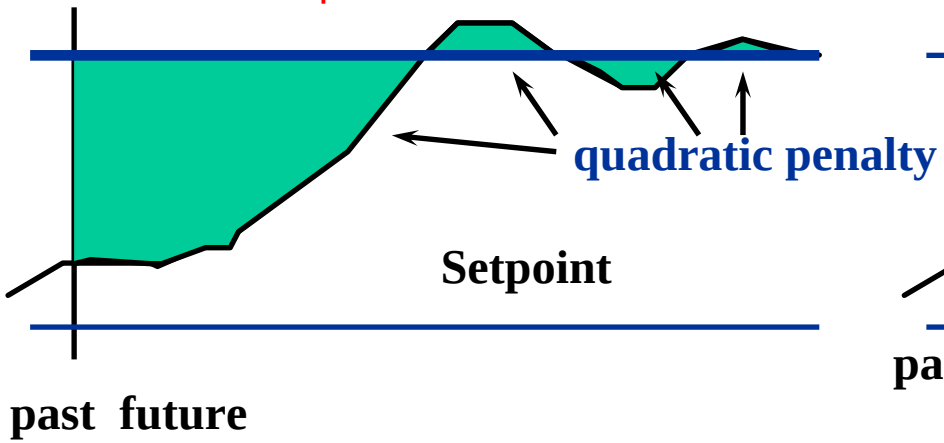
- **Most control algorithms use a single quadratic objective**
- **The HIECON algorithm uses a sequence of separate dynamic optimizations to resolve conflicting control objectives; CV errors are minimized first, followed by MV errors**
- **Connoisseur allows for a multi-model approach and an adaptive approach**
- **The RMPCT algorithm defines a funnel and finds the optimal trajectory y^r and input u^M which minimize the following objective:**

$$\min_{\mathbf{y}^r_{k+j},\mathbf{u}^M} J = \sum_{j=1}^{P} \left\| \mathbf{y}^r_{k+j} - \mathbf{y}_{k+j} \right\|_{\mathbf{Q}}^2 + \left\| \mathbf{u}_{M-1} - \mathbf{u}_{ss} \right\|_{\mathbf{S}}^2$$

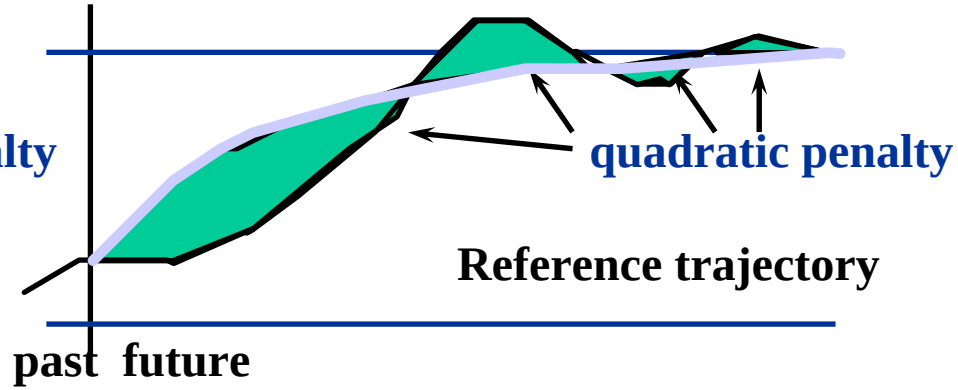
subject to a funnel constraint

Output Trajectories

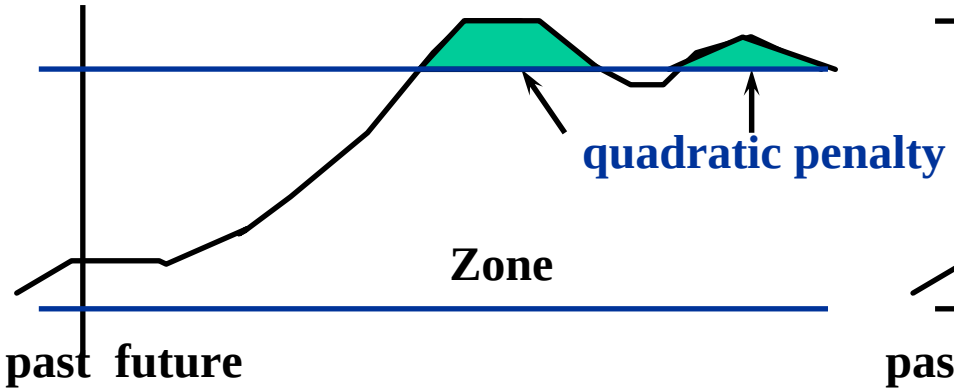
Aspen Tech's DMC



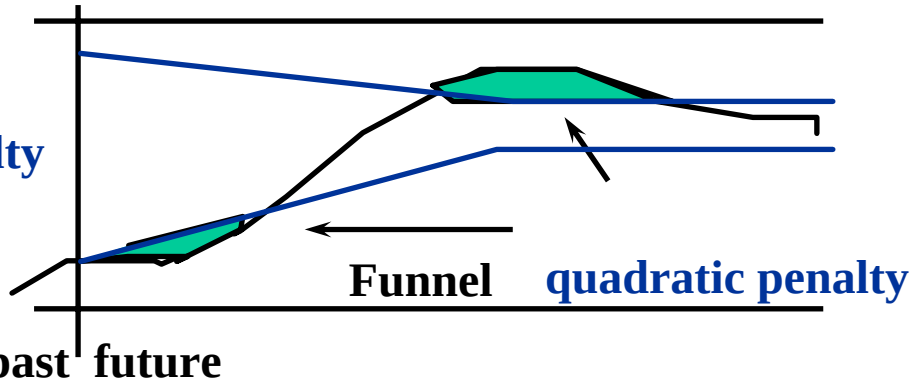
ID-COM, Adersa's



Soft Constraint, "Zone Control"

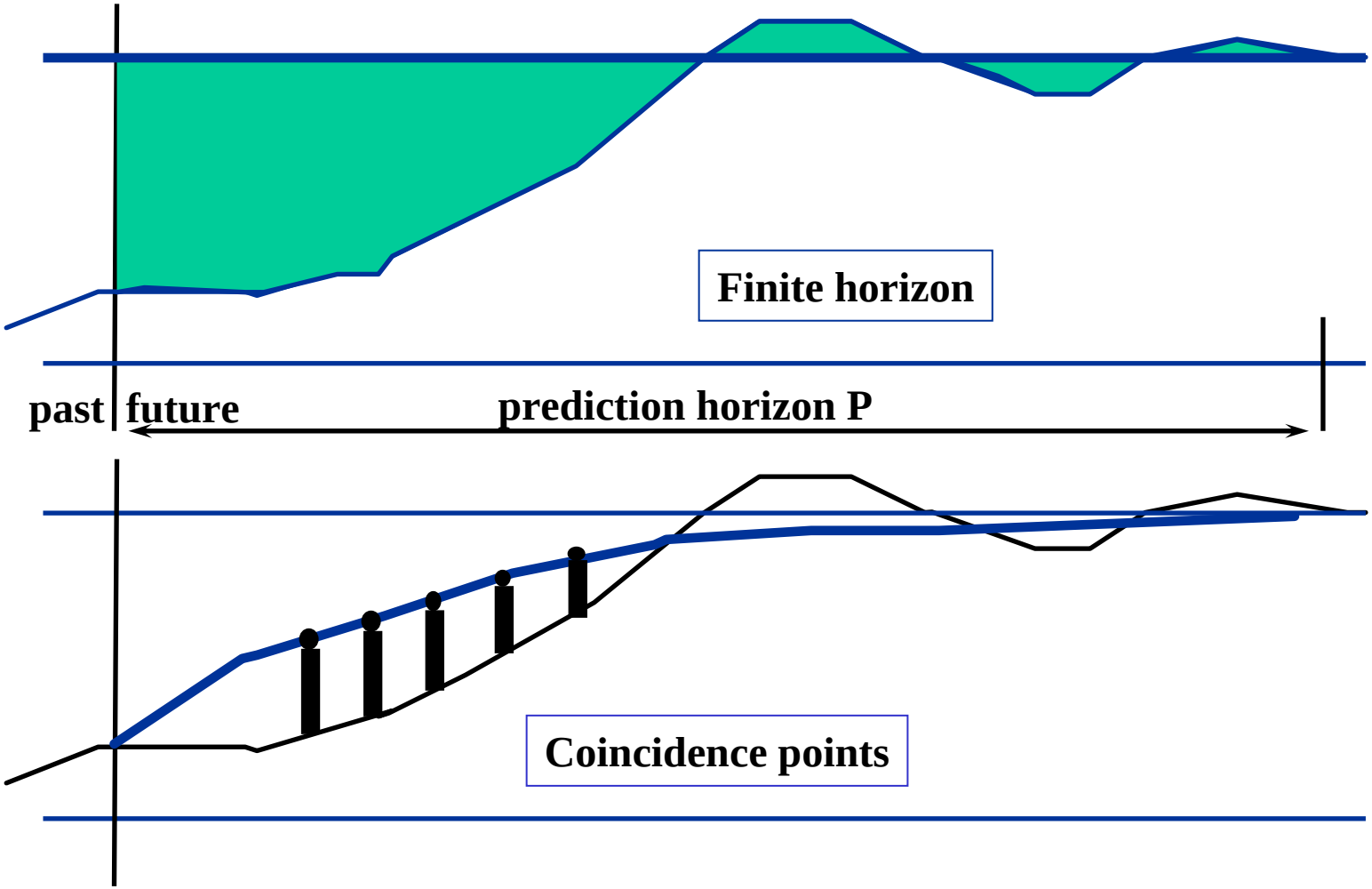


Honeywell's RMPCT

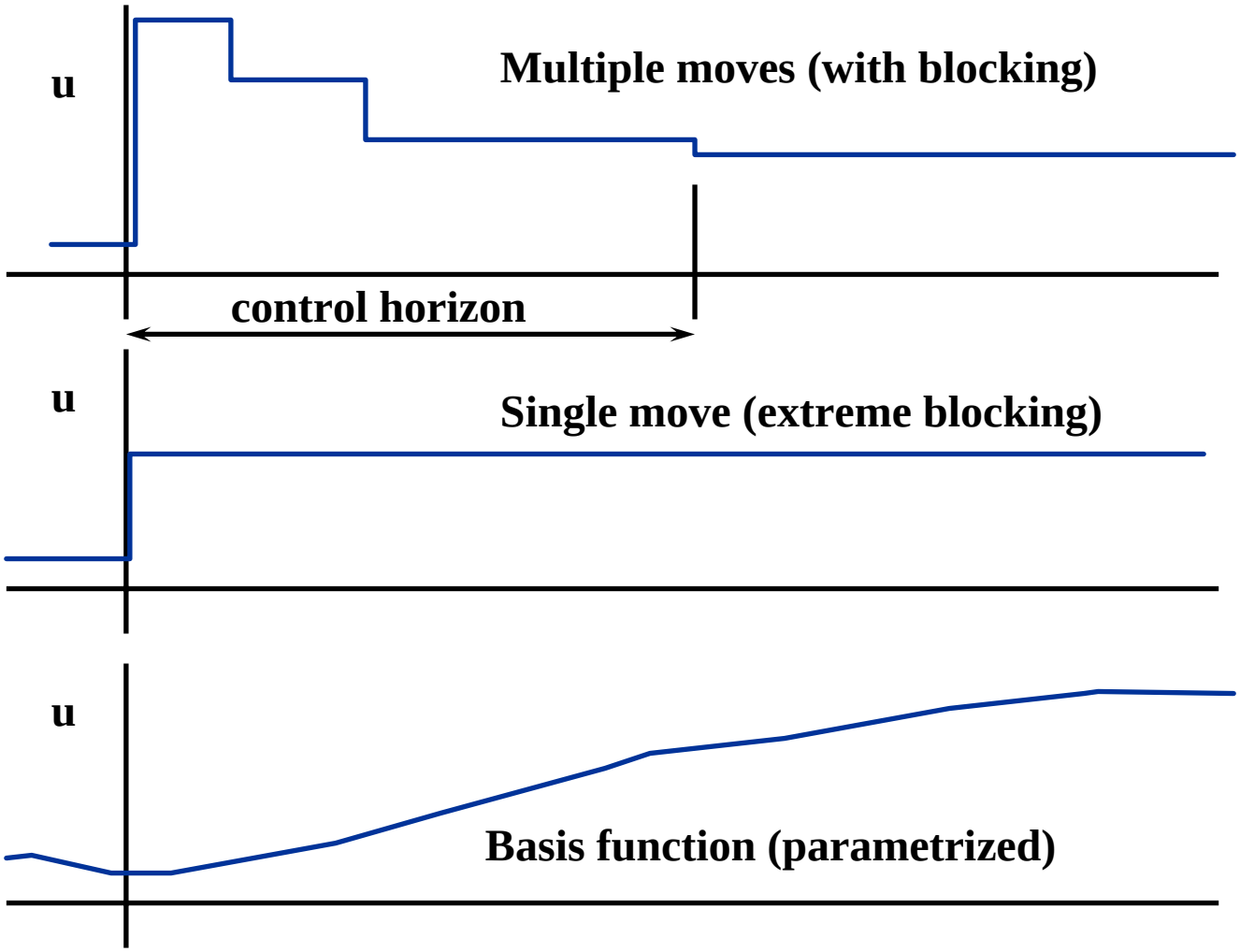


- Move suppression is necessary when reference trajectory is not used

Output Horizon



Input Parameterization



Process Model Types

Model Type	Origin	Linear/Nonlinear	Stable/Unstable
Differential Equations	physics	L,NL	S,U
State-Space	physics data	L,NL	S,U
Laplace Transfer Function	physics data	L	S,U
ARMAX/NARMAX	data	L,NL	S,U
Convolution (Finite Impulse or Step Response)	data	L	S
Other (Polynomial, Neural Net)	data	L,NL	S,U

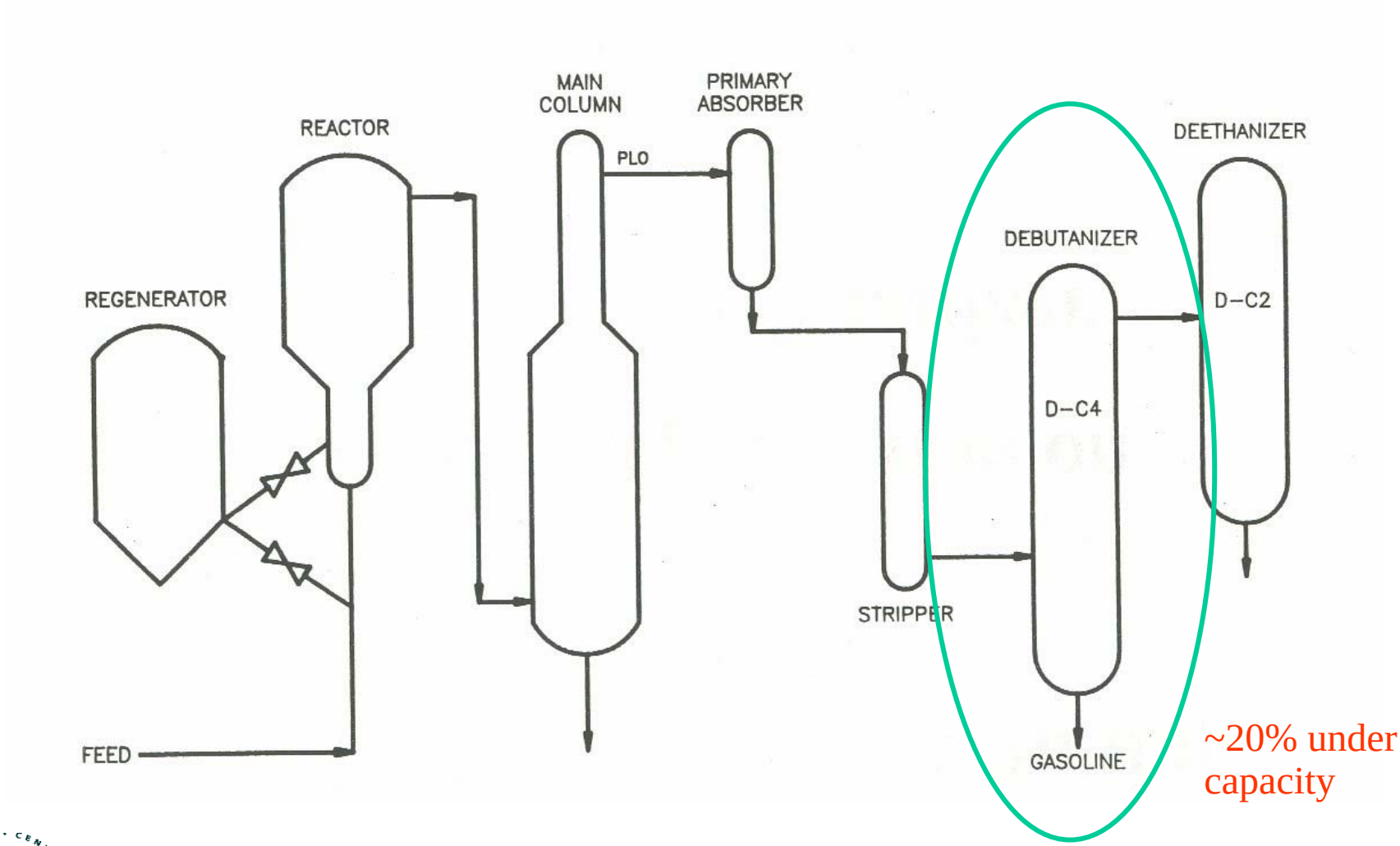
Identification Technology

- **Most products use PRBS-like or multiple steps test signals. Glide us es non-PRBS signals**
- **Most products use FIR, ARX or step response models**
  - Glide uses transfer function G(s)
  - RMPCT uses Box-Jenkins
  - SMOC uses state space models
- **Most products use least squares type parameter estimation:**
  - prediction error or output error methods
  - RMPCT uses prediction error method
  - Glide uses a global method to estimate uncertainty
- **Connoisseur has adaptive capability using RLS**
- **A few products (DMCplus, SMOC) have subspace identification metho ds available for MIMO identification**
- **Most products have uncertainty estimate, but most products do not m ake use of the uncertainty bound in control design**

Summary

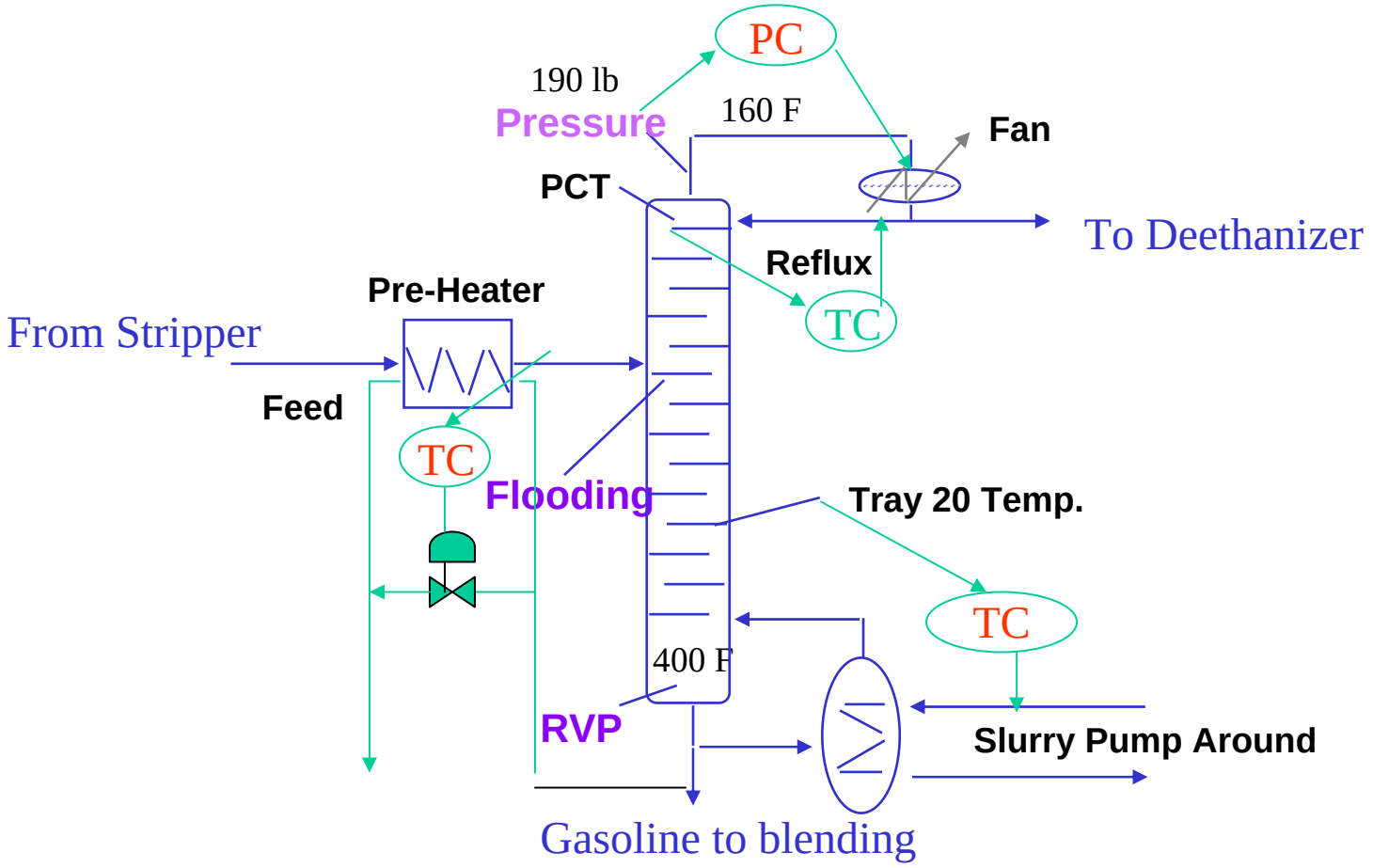
- **MPC is a mature technology!**
  - **Many commercial vendors with packages differing in model form, objective function form, etc.**
  - **Sound theory and experience**
- **Challenges are**
  - **Simplifying the model development process**
    - plant testing & system identification
    - nonlinear model development
  - **State Estimation**
    - Lack of sensors for key variables
  - **Reducing computational complexity**
    - approximate solutions, preferably with some guaranteed properties
  - **Better management of "uncertainty"**
    - creating models with uncertainty information (e.g., stochastic model)
    - on-line estimation of parameters / states
    - "robust" solution of optimization

FCCU Debutanizer



~20% under capacity

Debutanizer Diagram



Process Limitation

Operation Problems:

- Overloading
  -- over design capacity.

- Flooding
  -- usually jet flooding, causing very poor separation.

- Lack of Overhead Fan Cooling
  -- especially in summer.

Consequences:

- High RVP, giving away Octane Number
- High OVHD C5, causing problems at Alky.

Control Objectives

Constrained Control:

- Preventing safety valve from relieving
- Keep the tower from flooding
- Keep RVP lower than its target.

Regulatory Control:

- Regulate OVHD PCT or C5 at spec.
- Rejecting disturbance not through slurry, if possible.

Real-Time Optimization

Optimization Objectives:

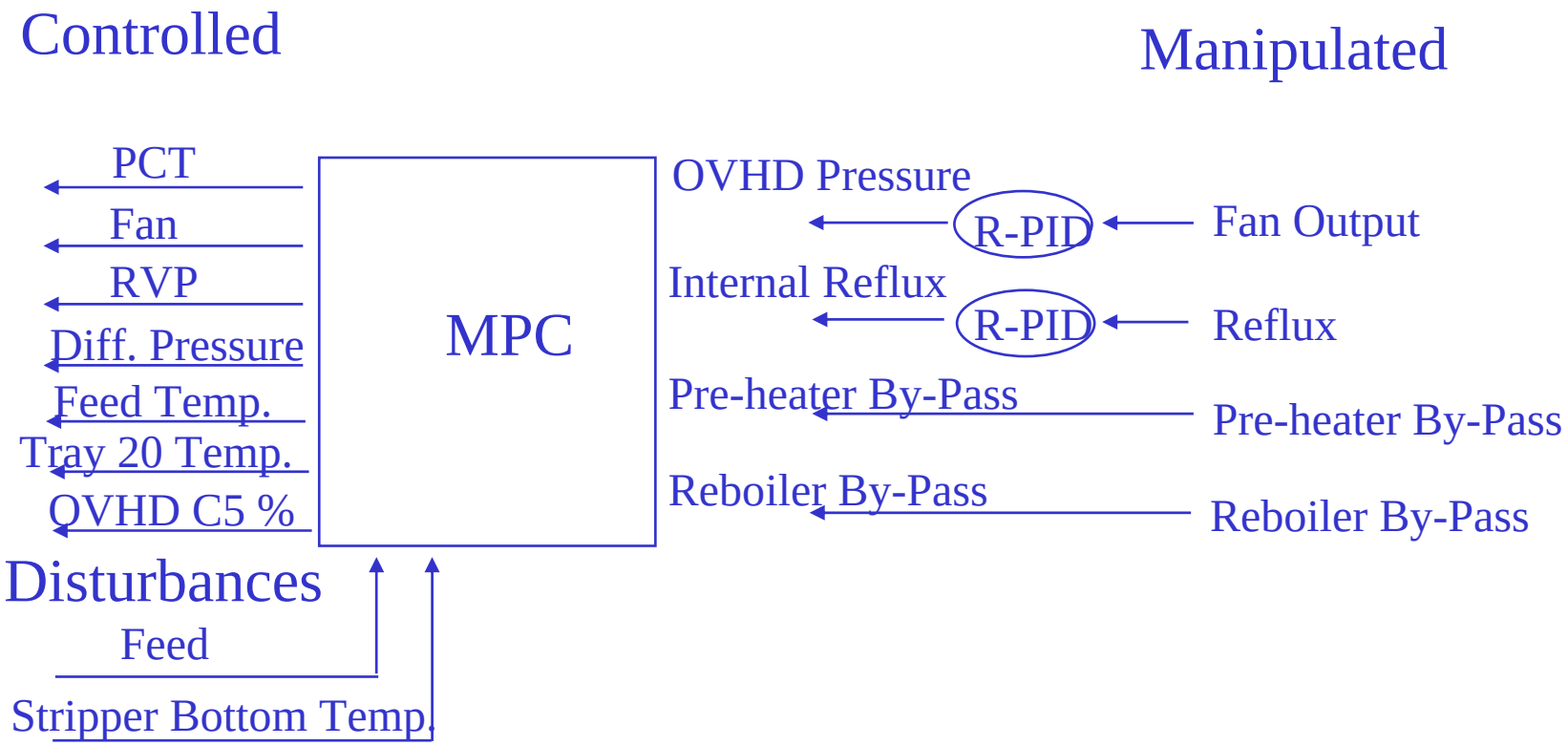
While maintaining PCT, RVP on their specifications

- Minimizing energy consumed

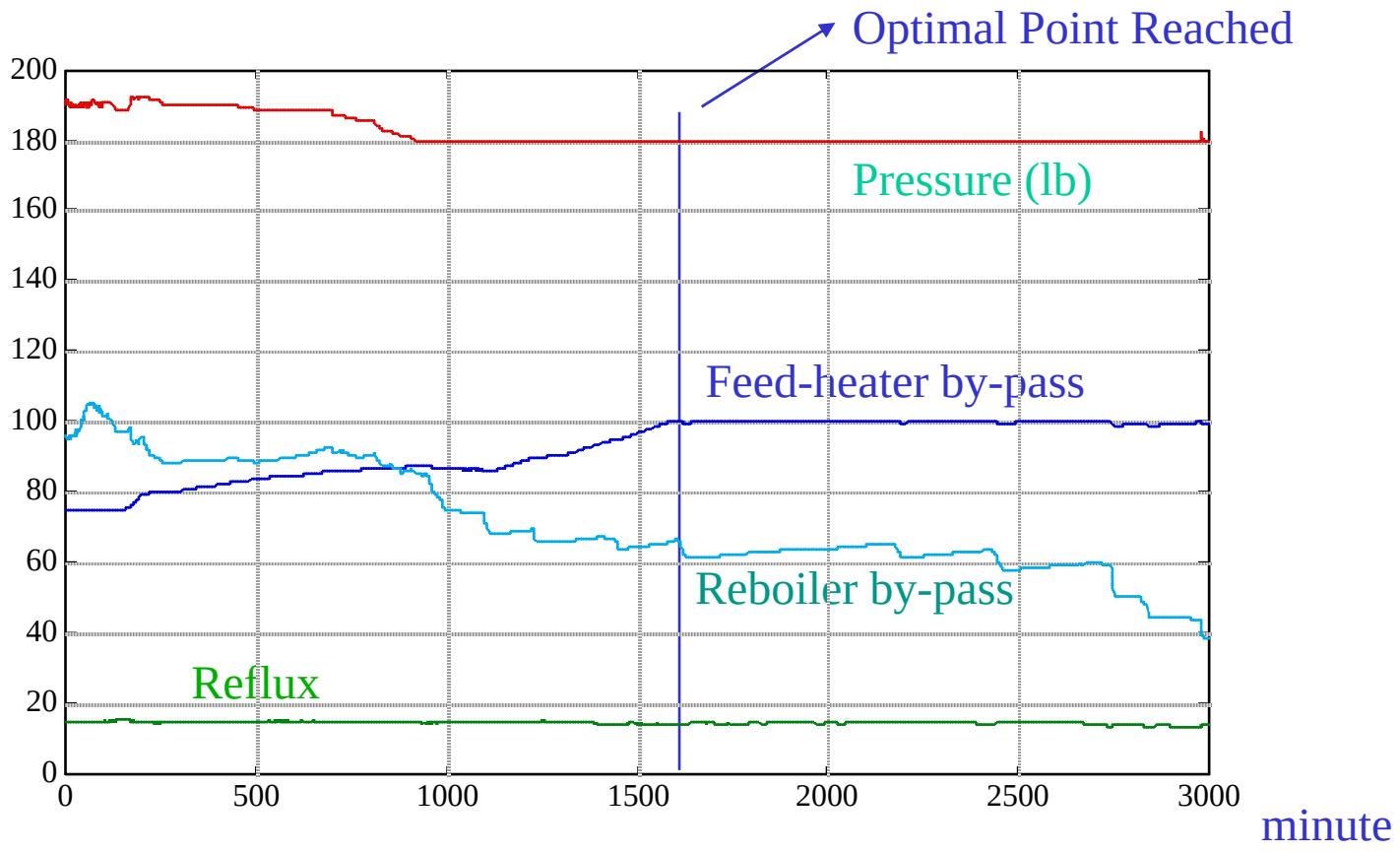
- Minimizing overhead reflux
- Minimizing overhead cooling required

- Minimizing overhead pressure
- Maximizing separation efficiency.

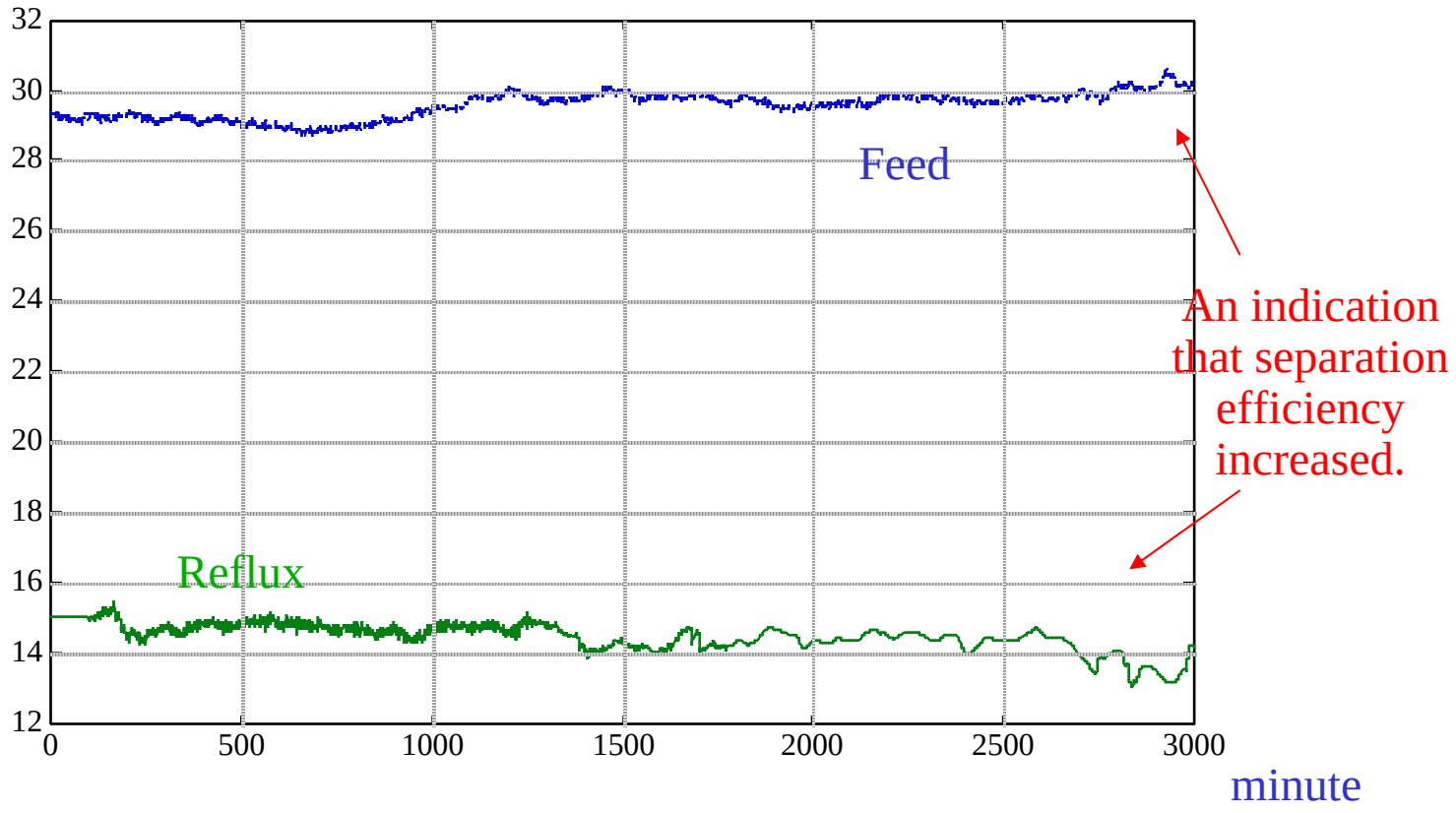
MPC Configuration



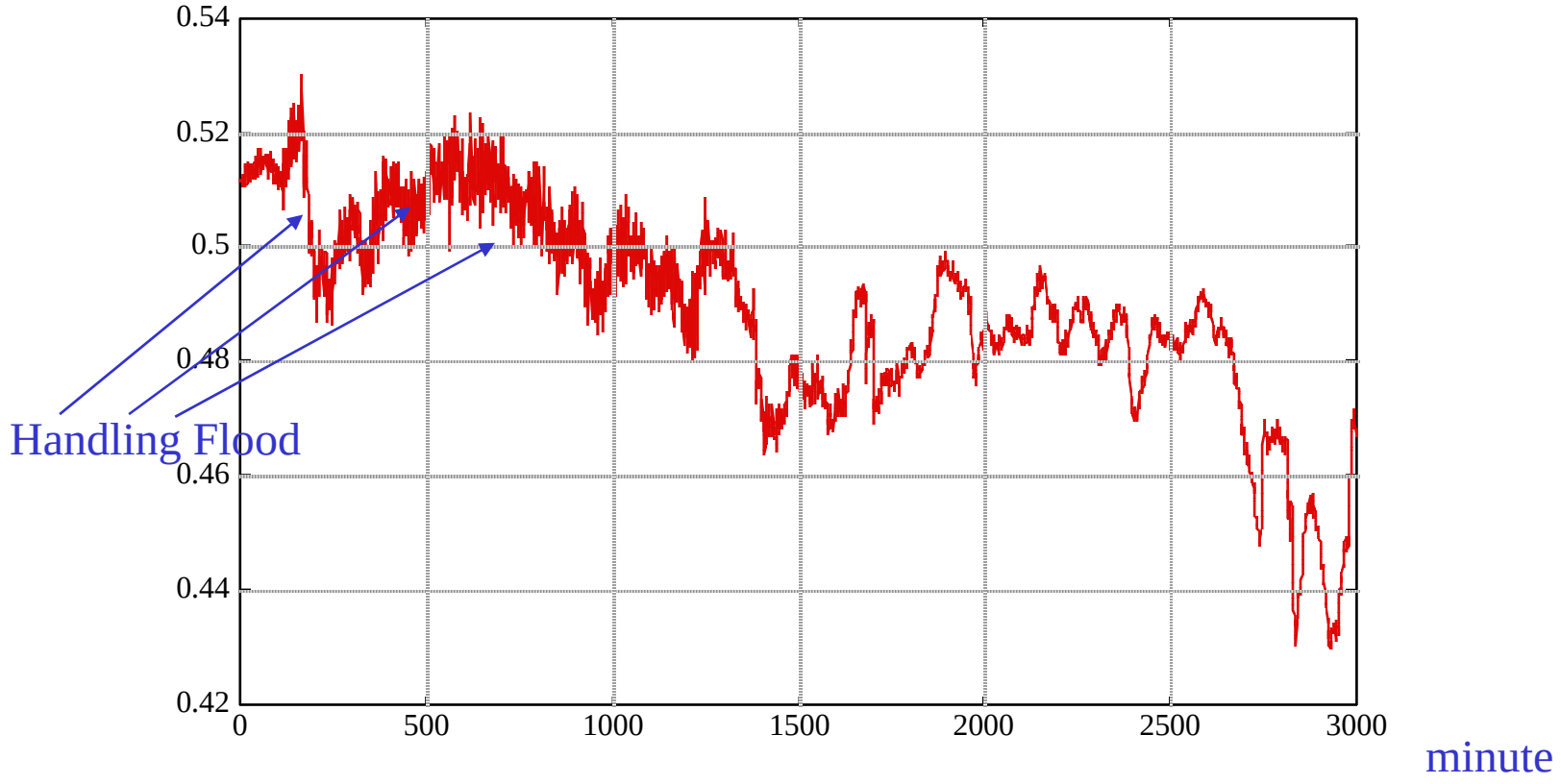
MPC's MV Moves



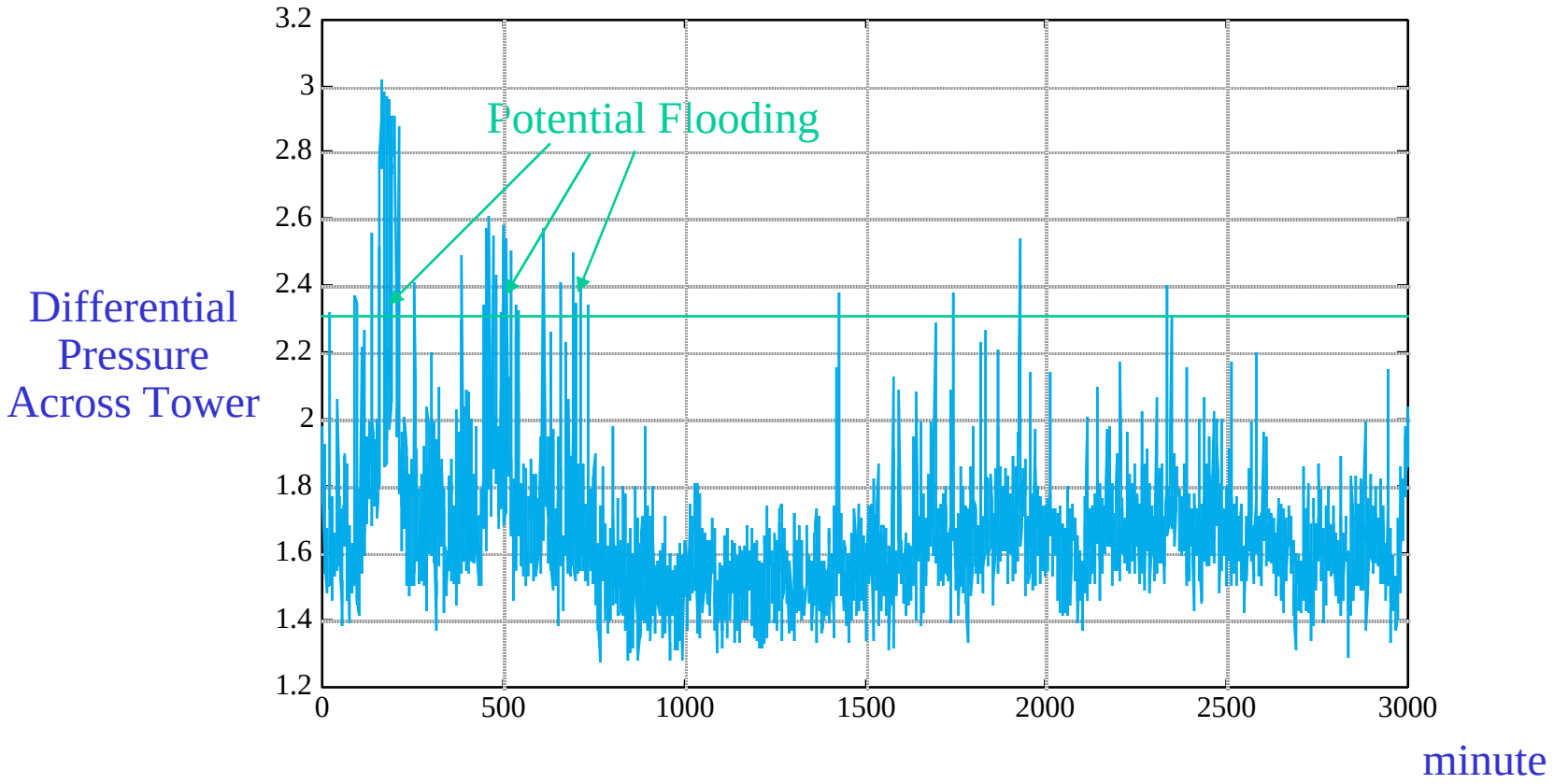
Reflux vs. Feed



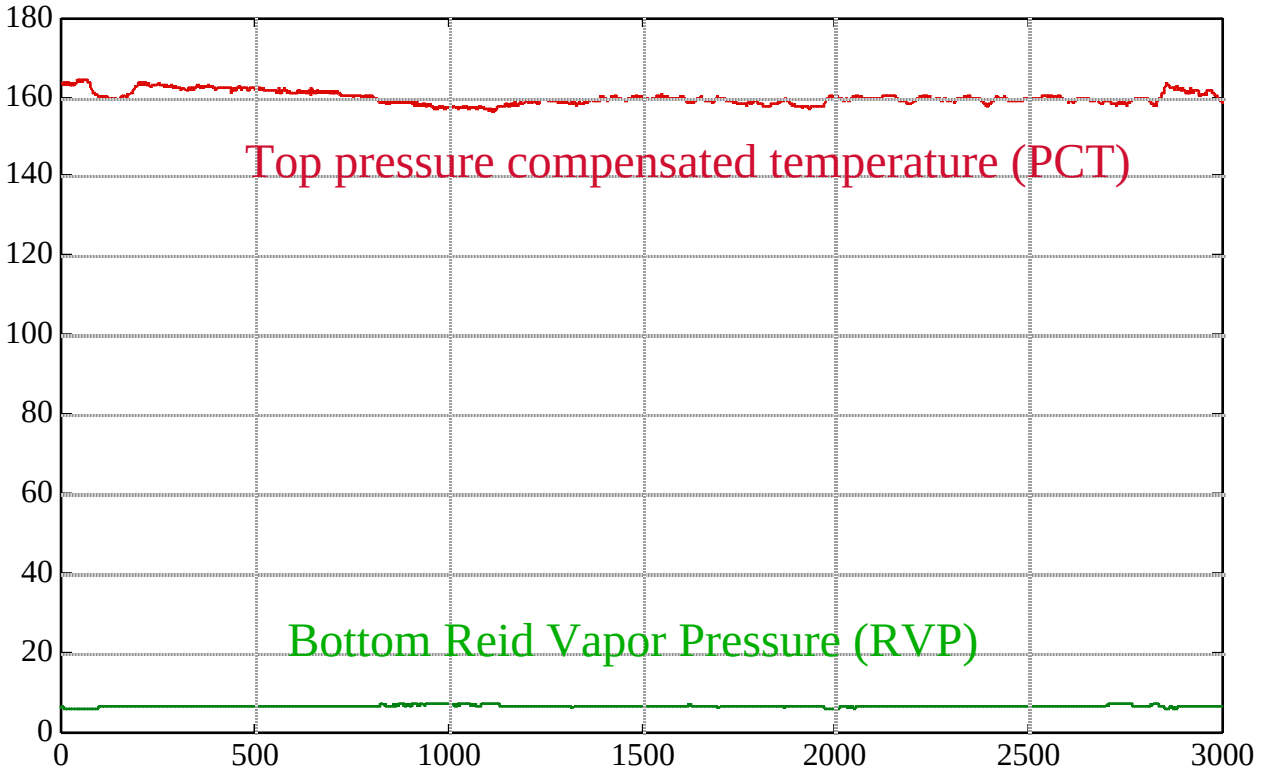
Reflux-to-Feed Ratio



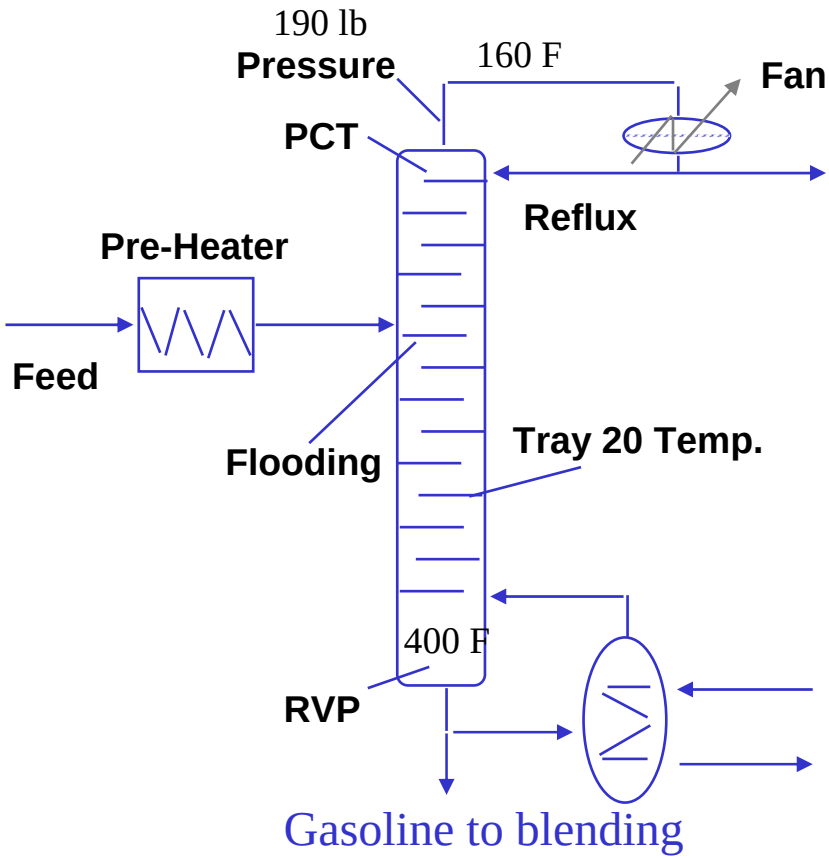
Handling Flood

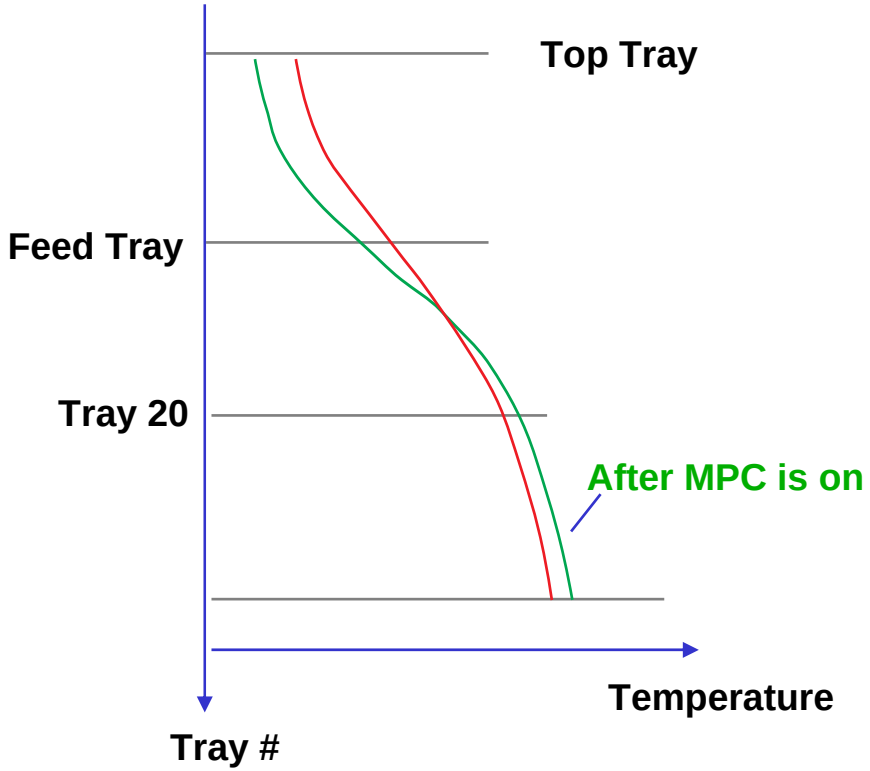


Product Spec's

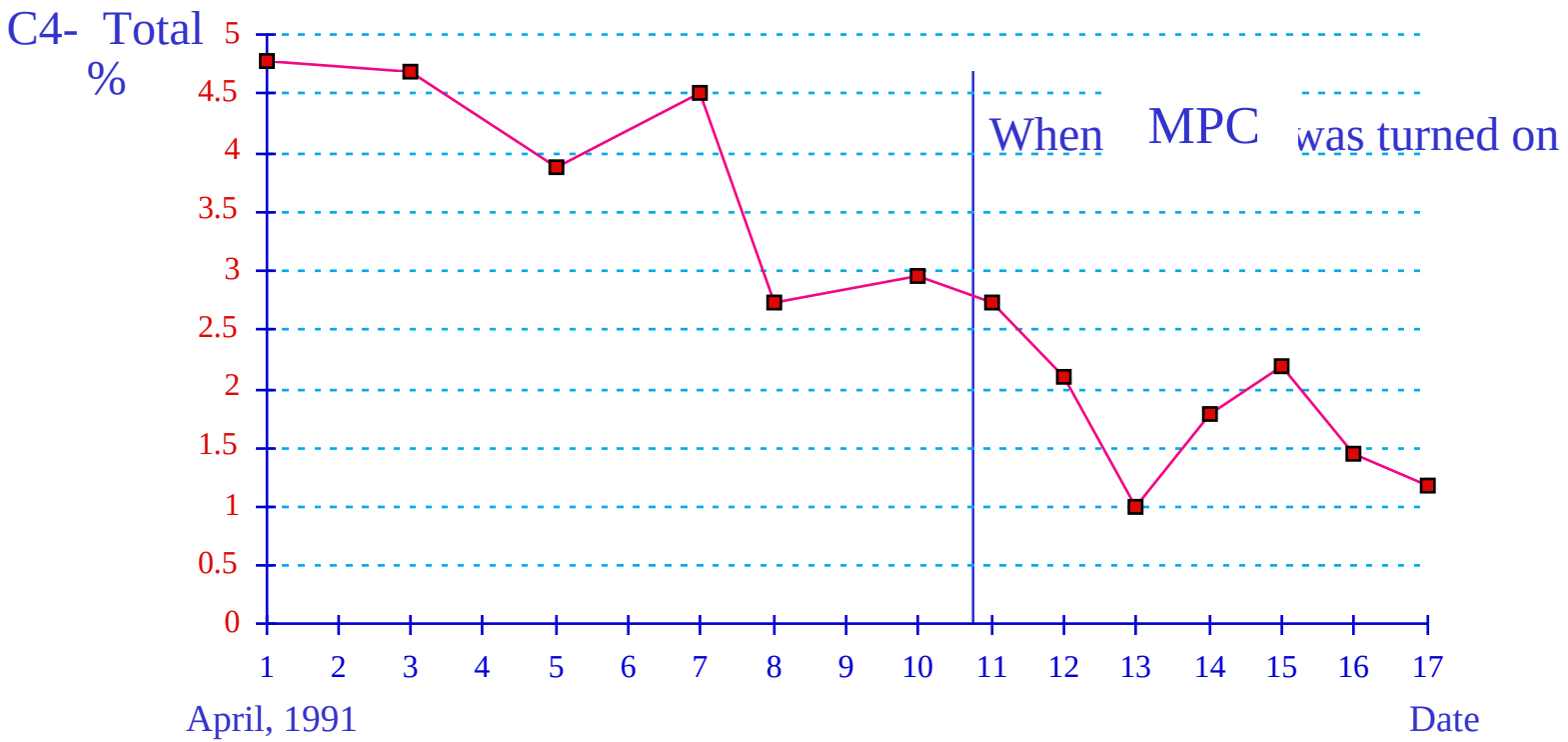


MPC Operation Overview



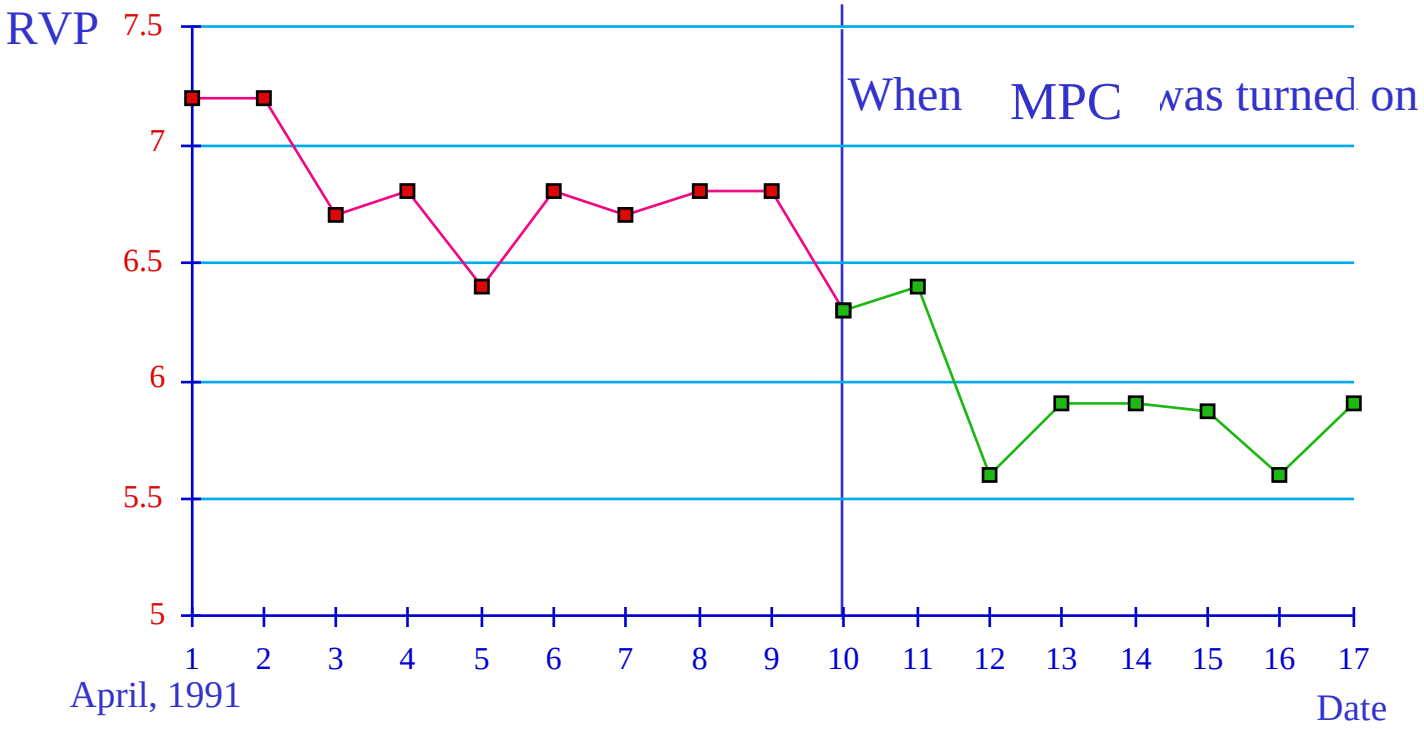


MPC Control Results (1)



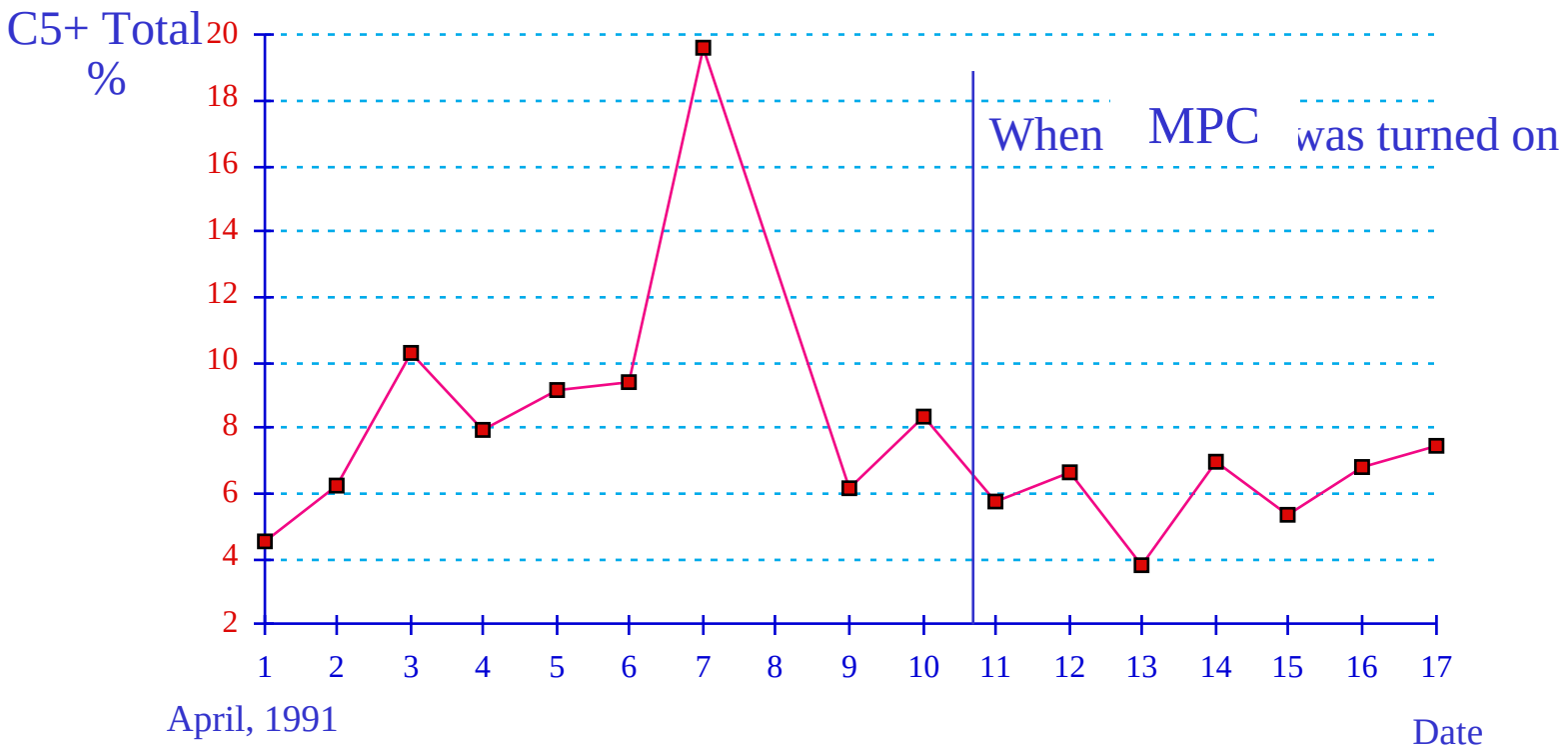
Debutanizer Bottom C4- Total (I-C4, N-C4 & C4=-)

MPC Control Results (2)



Debutanizer Bottom RVP Daily Average

MPC Control Results (3)



Debutanizer Top C5+ Total (I-C5, N-C5 & C5=+)

Other Benefits

- Reflux is 15% lower than before
- Separation efficiency is increased
- Now have room for lower RVP or PCT, if needed
- Variance on PCT and RVP is reduced;
  Note: Variance on the tray-20 temperature is increased, and it should be!!
- Energy saving (~10%)
- OVHD fan maximum bound may be avoided
- Flooding is eliminated.

Acknowledgment

- Professor S. Joe Qin, UT, Austin
- Dr. Joseph Lu, Honeywell, Phoenix, AZ
- Center for Process Systems Engineering Industrial Members

Lecture 2

Details of MPC Algorithms and Theory

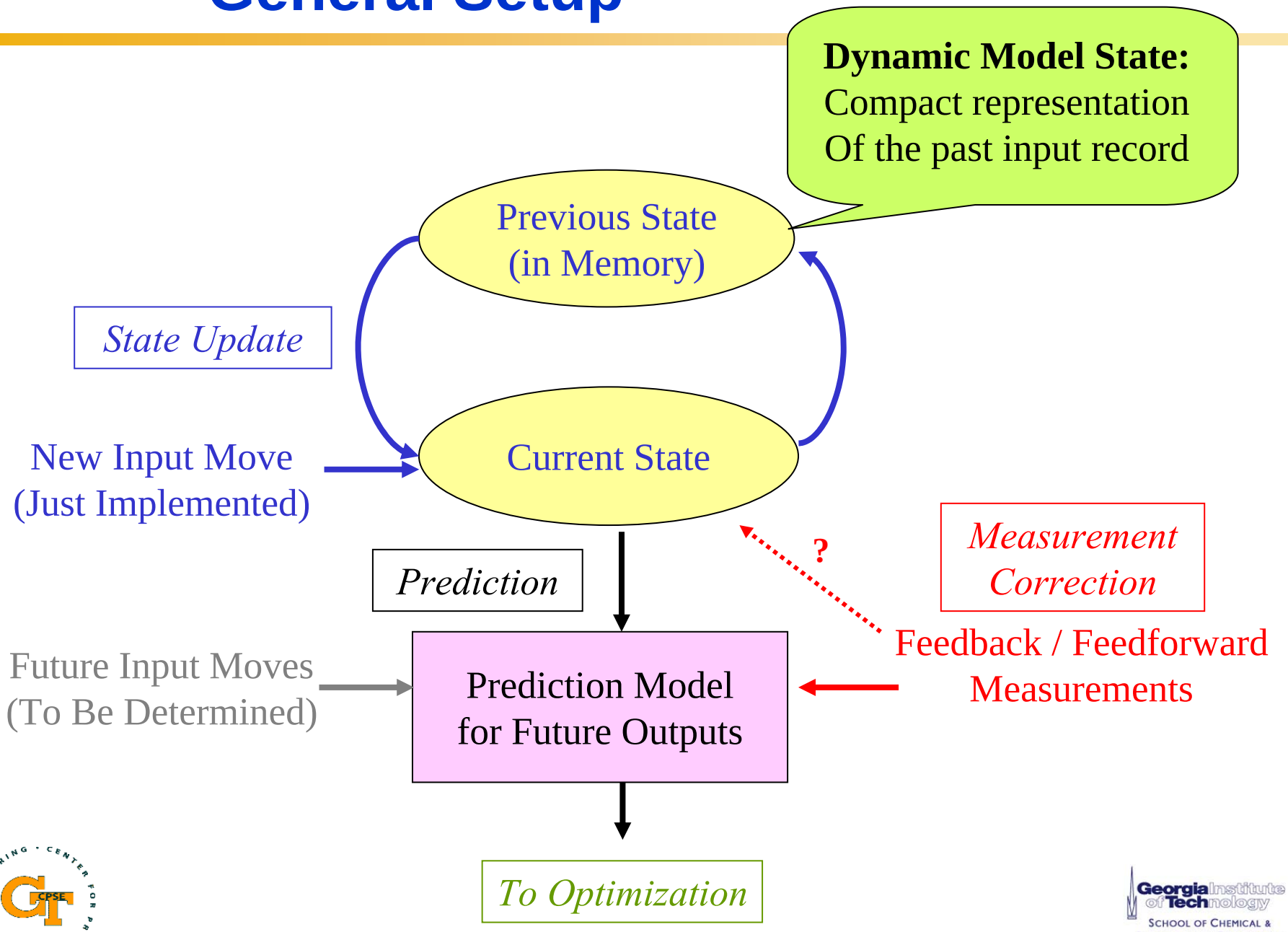
- Impulse and step response models and the prediction equation
- Use of state estimation
- Optimization
- Infinite-horizon MPC and stability
- Use of nonlinear models

Georgia Institute of Technology
School of Chemical & Biomolecular Engineering

Important Ingredients of an MPC Algorithm

- **Dynamic Model → Prediction Model**
  - Predicted future outputs = Function of current "state" (stored in memory) + feedforward measurement + feedback measurement correction + future input adjustments
- **Objective and Constraints**
- **Optimization Algorithm**
- **Receding Horizon Implementation**

General Setup



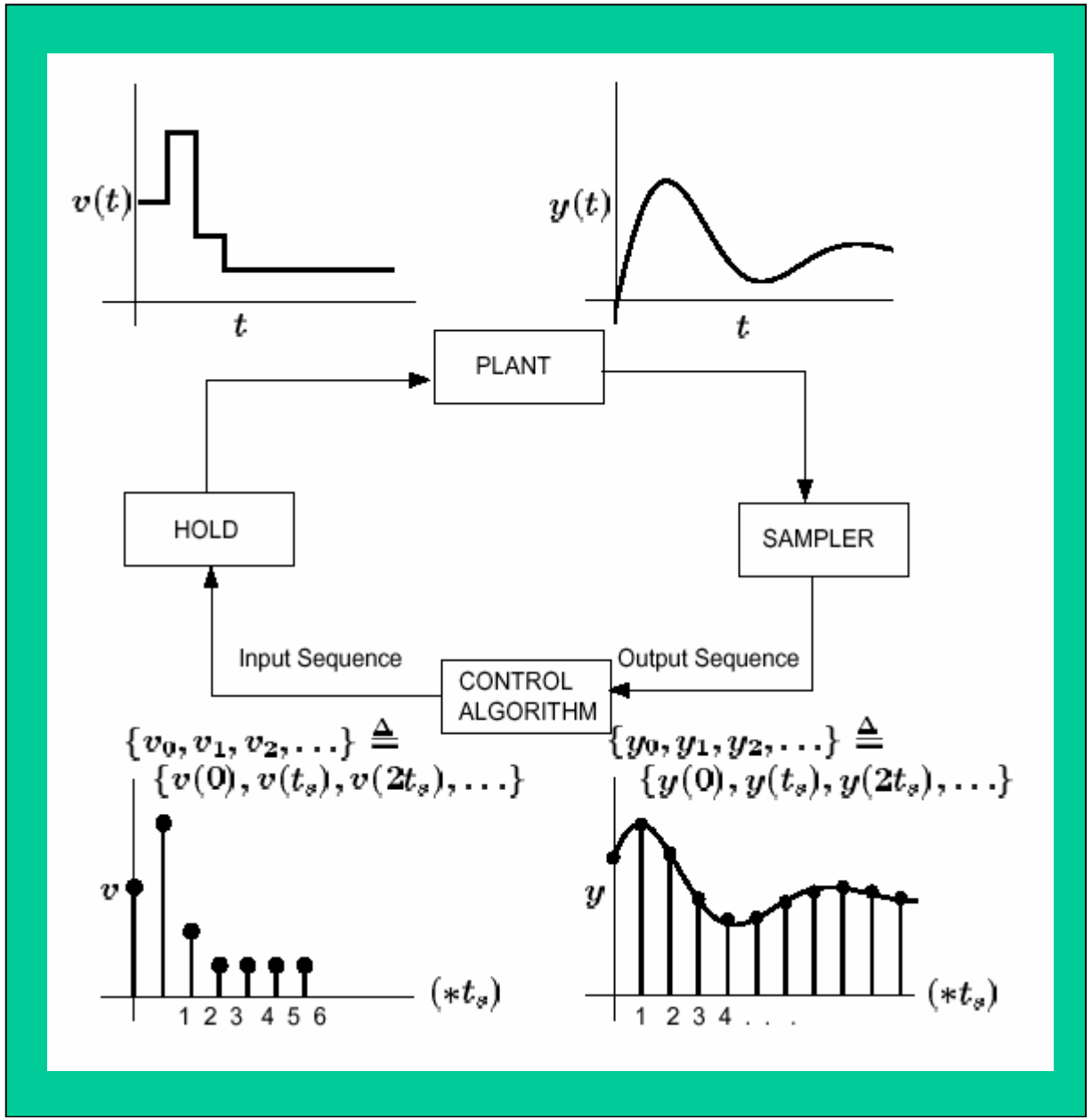
Options

- **Model Types**
  - Finite Impulse Response Model or Step Response Model
  - State-Space Model
  - Linear or Nonlinear
- **Measurement Correction**
  - To the prediction (based on open-loop state calculation)
  - To the state (through state estimation)
- **Objective Function**
  - Linear or Quadratic
  - Constrained or Unconstrained

Prediction Model for Different Model Types

- **Finite Impulse Response Model**
- **Step Response Model**
- **State-Space Model**

Sample-Data (Computer) Control



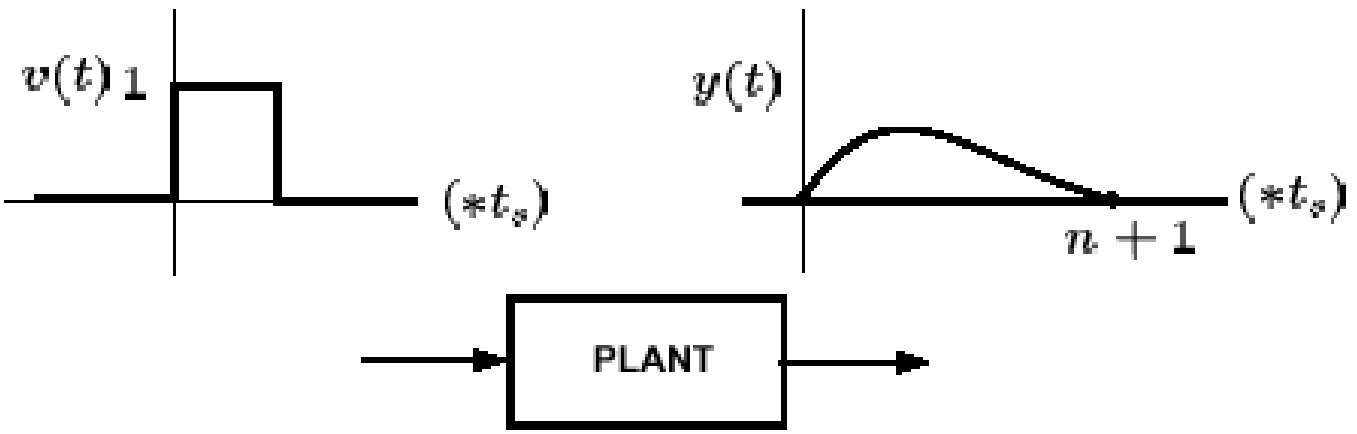
Model relates input samples to output samples

$$\{v_0, v_1, v_2, \cdots\}$$
$$\downarrow$$
$$\{y_0, y_1, y_2, \cdots\}$$

v can be a MV (*u*) or a Measured DV (*d*)

Georgia Institute of Technology
School of Chemical & Biomolecular Engineering

Finite Impulse Response Model (1)



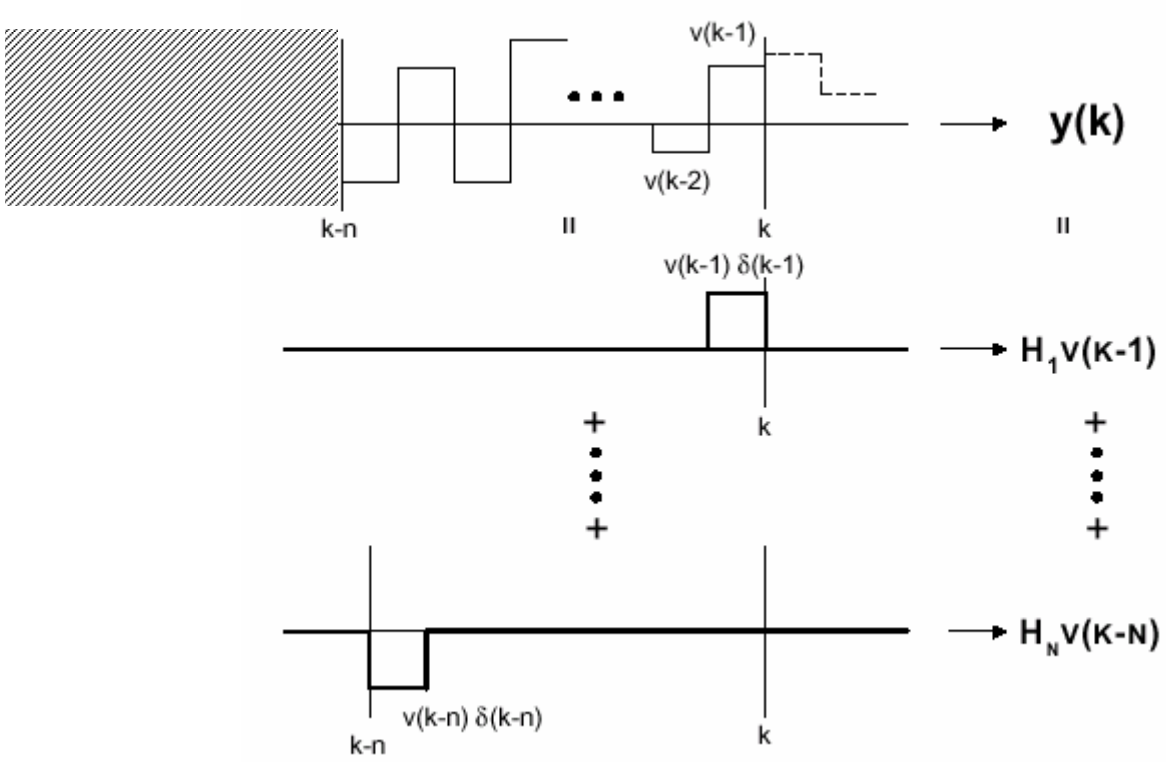
$$v = \{1, 0, 0, \ldots\} \qquad y = \{0, H_1, H_2, H_3, \ldots, H_n, H_{n+1} \ldots\}$$

Assumptions:

- H_0 = 0: **no** immediate **effect**
- **The response settles back in n steps s.t. $H_{n+1} = H_{n+2} = \ldots = 0$: "Finite Impulse Response" (reasonable for stable processes)**

Finite Impulse Response Model (2)

Linear Model→ "Superposition Principle"



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

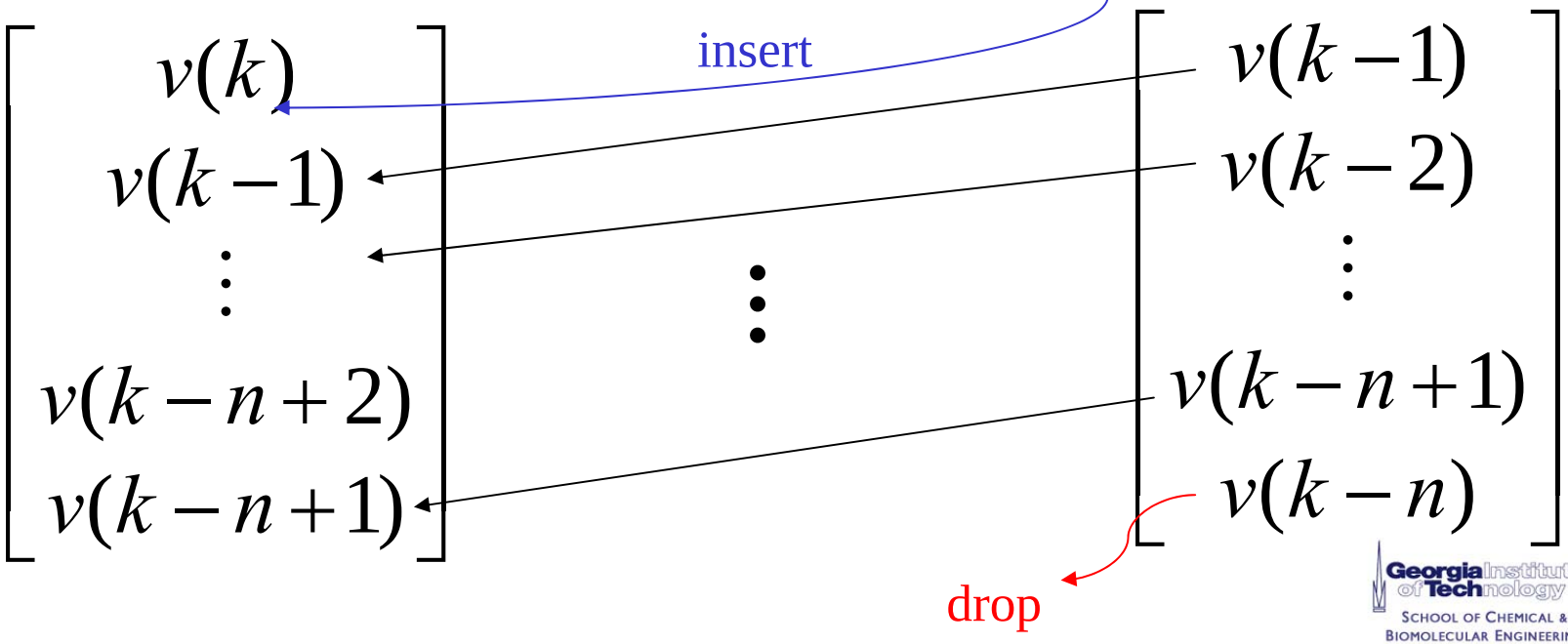
Finite Impulse Response Model (3)

- State $x(k) = \left[v^T(k-1), \cdots, v^T(k-n) \right]^T$

  n past input samples (includes both MVs and measured DVs)

- State Update: Easy!

$$x(k+1) = \underbrace{M_0}_{shift} x(k) + \underbrace{\mathrm{T}}_{insertion} v(k)$$



Finite Impulse Response Model (4)

- Prediction

Model prediction of y

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

Model prediction error

Model error+Unmeasured disturbance

$$e(k) = y_m(k) - y(k)$$

Measured output

Predicted future output samples

Past input samples stored in memory

Future input samples (to be decided)

Dynamic Matrices (made of impulse response coefficients)

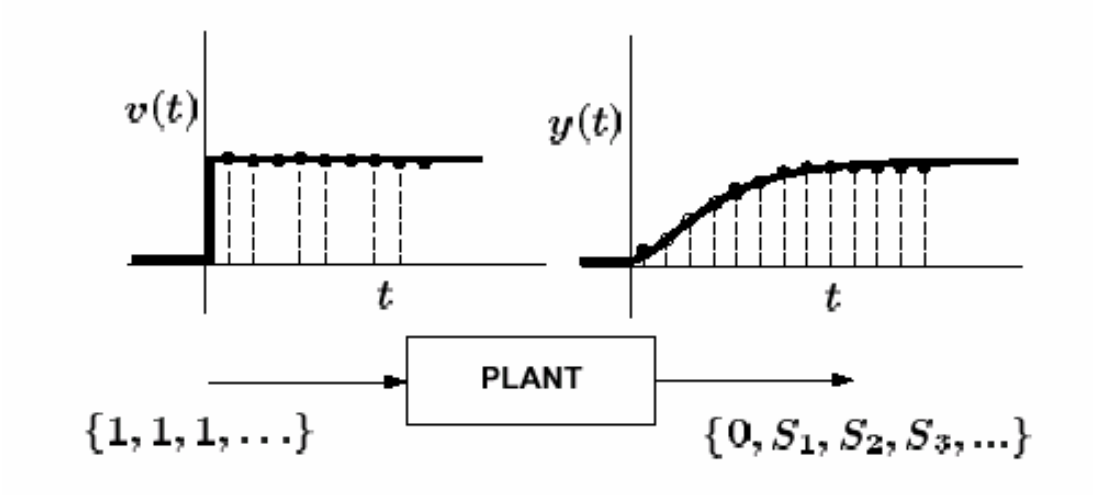
$$\begin{bmatrix} y_{k+1|k} \\ \vdots \\ y_{k+p|k} \end{bmatrix} = \Psi_1^u \begin{bmatrix} u(k-1) \\ \vdots \\ u(k-n) \end{bmatrix} + \Psi_2^u \begin{bmatrix} u(k) \\ \vdots \\ u(k+p-1) \end{bmatrix} + \Psi_1^d \begin{bmatrix} d(k-1) \\ \vdots \\ d(k-n) \end{bmatrix} + \Psi_2^d \begin{bmatrix} d(k) \\ \vdots \\ d(k+p-1) \end{bmatrix} + \begin{bmatrix} e(k) \\ \vdots \\ e(k) \end{bmatrix}$$

Feedback Error Correction

Feedforward term: Past disturbance samples stored in memory

Feedforward term: new measurement (Assume d(k)=d(k=1)=….=d(k+p-1))

Step Response Model (1)



Assumptions:

- **S_0 = 0: no** immediate **effect**
- **The response settles in n steps s.t. $S_n = S_{n+1} = \ldots = S_\infty$: the same as the finite impulse response assumption**
- **Relationship with the impulse response coefficients:**

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

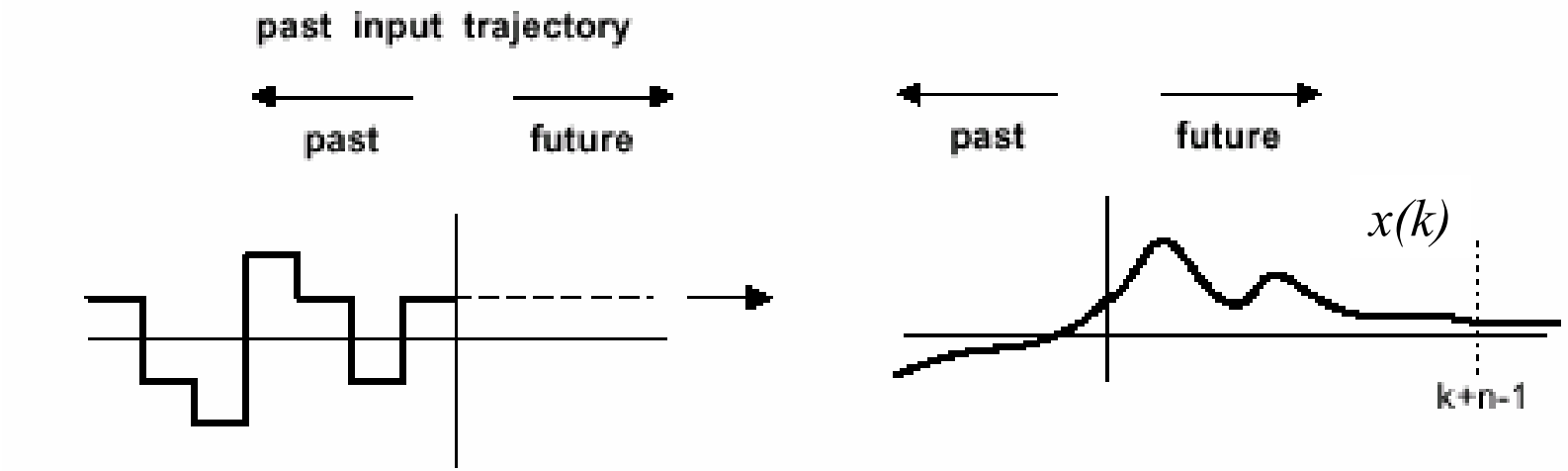
Step Response Model (2)

- **State**

$$x(k) = \left[{y_0}^T(k), \cdots, {y_{n-1}}^T(k) \right]^T$$

***n* future outputs** assuming the input remains constant at the most recent value

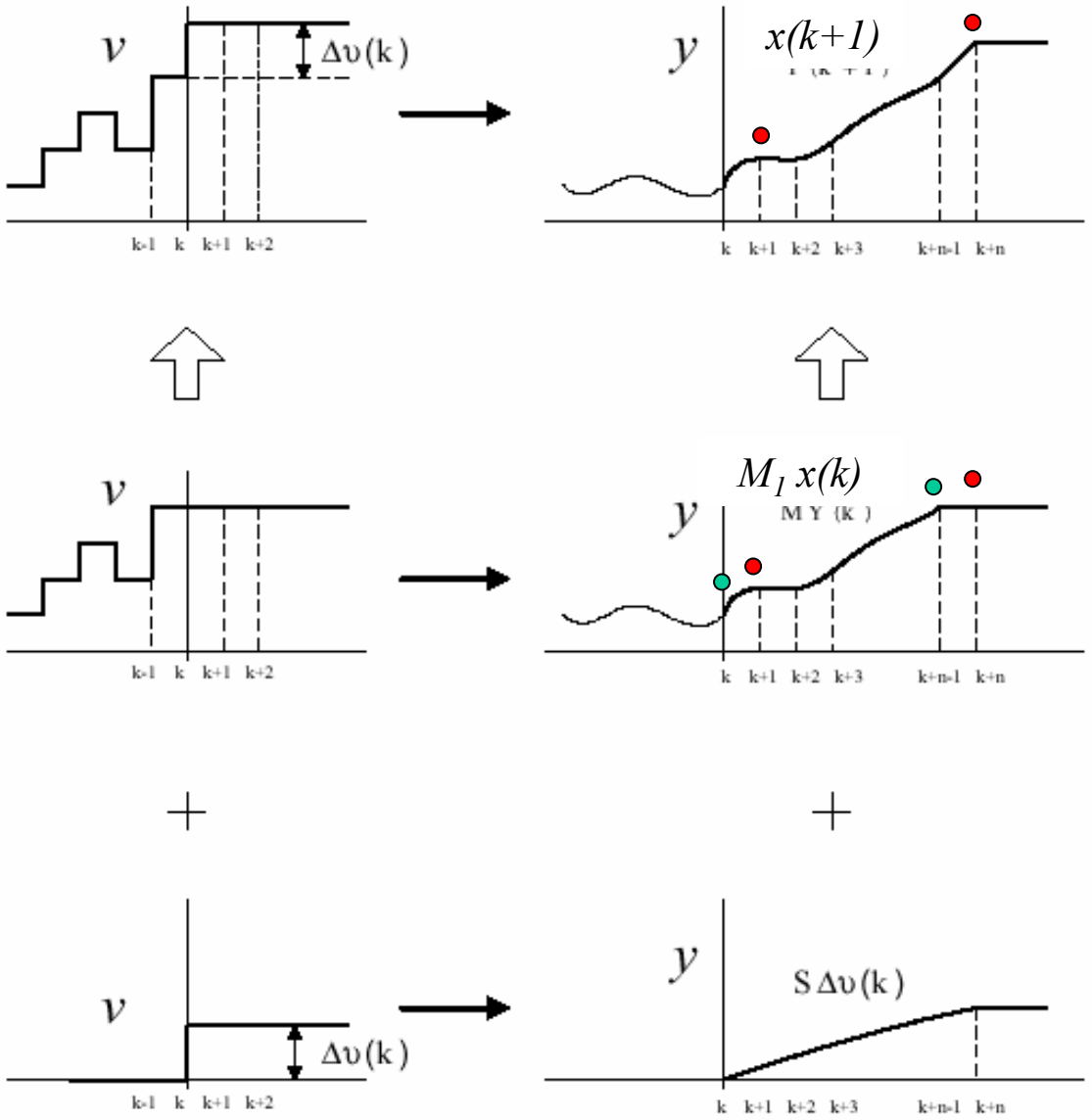
$$y_i(k) = y(k+i) \quad \text{w/} \quad \Delta u(k) = \Delta u(k+1) = \cdots = 0$$



Note $y_{n-1}(k) = y_n(k) = \cdots = y_\infty(k)$

Also $y(k) = y_0(k)$

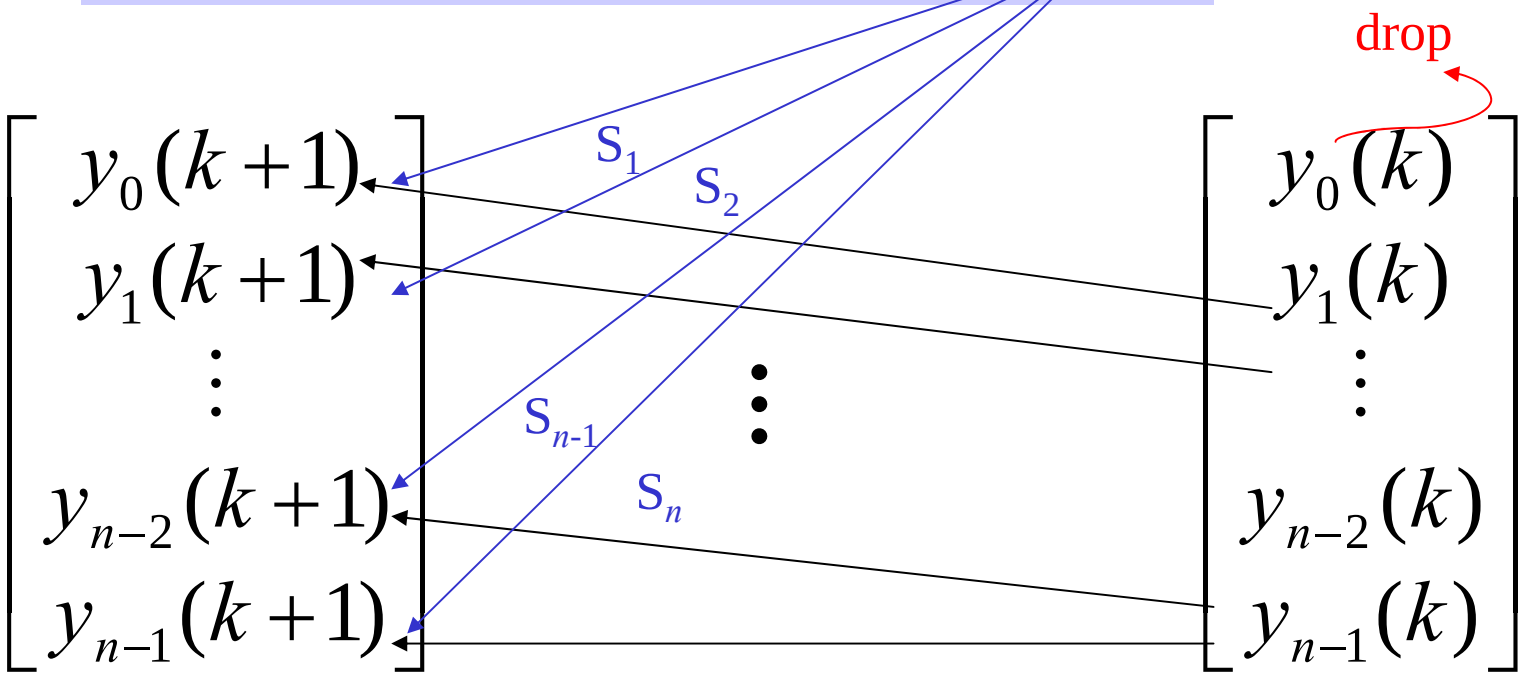
Pictorial Representation of the State Update



Step Response Model (3)

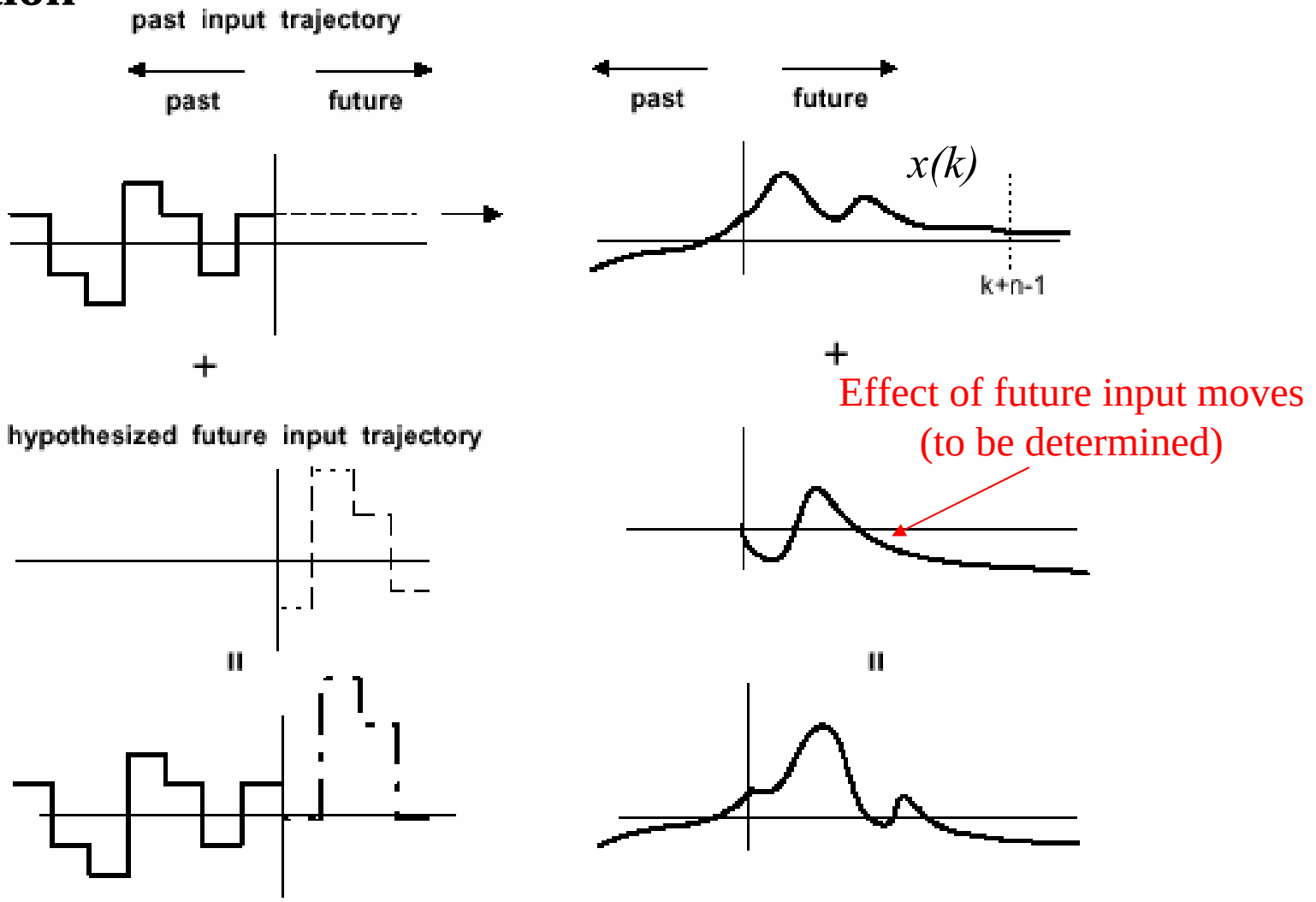
- State Update

$$x(k+1) = \underbrace{M_1}_{shift} x(k) + \underbrace{S}_{step\ response} \Delta v(k)$$



Step Response Model (4)

- **Prediction**



Step Response Model (4)

- Prediction

Model prediction of y(k) →

$$y(k) = y_0(k) = [1, 0, \cdots, 0]x(k)$$

Model prediction error →

$$e(k) = y_m(k) - y(k)$$

Predicted future output samples

The "state" stored in memory

Future input moves (to be decided)

$$\begin{bmatrix} y_{k+1|k} \\ \vdots \\ y_{k+p|k} \end{bmatrix} = \begin{bmatrix} y_1(k) \\ \vdots \\ y_p(k) \end{bmatrix} + \Omega^u \begin{bmatrix} \Delta u(k) \\ \vdots \\ \Delta u(k+p-1) \end{bmatrix}$$

Dynamic Matrices (made of step response coefficients)

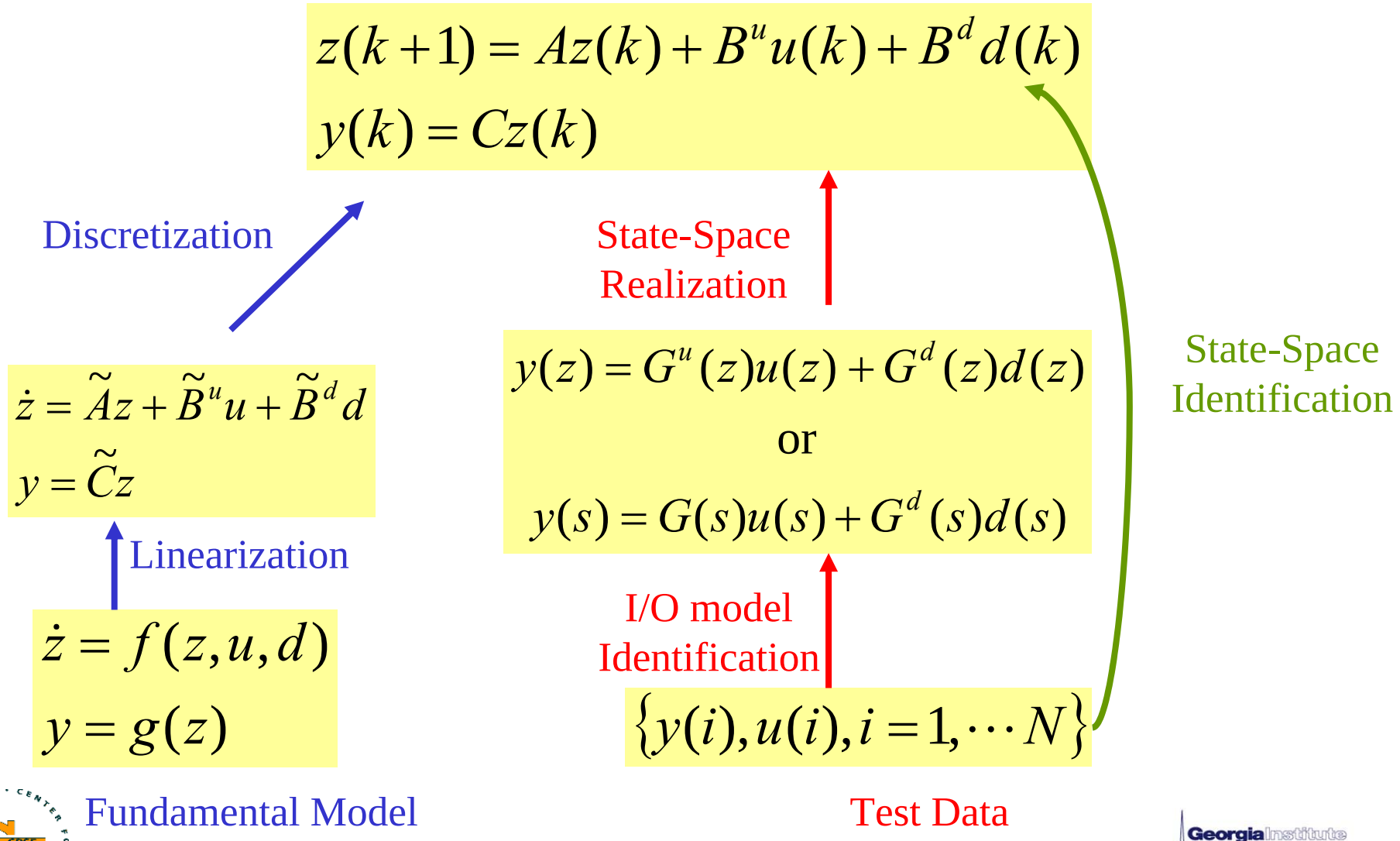
$$\Omega^d \begin{bmatrix} \Delta d(k) \\ \vdots \\ \Delta d(k+p-1) \end{bmatrix} + \begin{bmatrix} e(k) \\ \vdots \\ e(k) \end{bmatrix}$$

Feedback Error Correction

Feedforward term: new measurement

(Assume $\Delta d(k+1) = \ldots = \Delta d(k+p-1) = 0$)

State-Space Model (1)



State-Space Model (2)

$$\begin{pmatrix} z(k+1) = Az(k) + B^u u(k) + B^d d(k) \\ y(k+1) = Cz(k+1) \end{pmatrix}$$

$$- \quad \begin{pmatrix} z(k) = Az(k-1) + B^u u(k-1) + B^d d(k-1) \\ y(k) = Cz(k) \end{pmatrix}$$

$$\Delta z(k+1) = A\Delta z(k) + B^u \Delta u(k) + B^d \Delta d(k)$$

$$\Delta y(k+1) = C\Delta z(k+1) \quad \rightarrow$$

$$y(k+1) = y(k) + C\left(A\Delta z(k) + B^u \Delta u(k) + B^d \Delta d(k)\right)$$

$$\begin{bmatrix} \Delta z(k+1) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ CA & I \end{bmatrix} \begin{bmatrix} \Delta z(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} B^u \\ CB^u \end{bmatrix} \Delta u(k) + \begin{bmatrix} B^d \\ CB^d \end{bmatrix} \Delta d(k)$$

$$y(k) = \begin{bmatrix} 0 & I \end{bmatrix} \begin{bmatrix} \Delta z(k) \\ y(k) \end{bmatrix}$$

State Update

$$x(k+1) = \Phi x(k) + \Gamma^u \Delta u(k) + \Gamma^d \Delta d(k)$$

$$y(k) = \Xi x(k)$$

State-Space Model (3)

- Prediction

Model prediction of y(k) →

$$y(k) = \Xi x(k)$$

Model prediction error →

$$e(k) = y_m(k) - y(k)$$

Predicted future output samples

The "state" stored in "memory"

Future input moves (to be decided)

$$\begin{bmatrix} y_{k+1|k} \\ \vdots \\ y_{k+p|k} \end{bmatrix} = \begin{bmatrix} \Xi\Phi \\ \vdots \\ \Xi\Phi^p \end{bmatrix} x(k) + \Omega^u \begin{bmatrix} \Delta u(k) \\ \vdots \\ \Delta u(k+p-1) \end{bmatrix}$$

Dynamic Matrix (made of step response coefficients)

$$\Omega^d \begin{bmatrix} \Delta d(k) \\ \vdots \\ \Delta d(k+p-1) \end{bmatrix} + \begin{bmatrix} e(k) \\ \vdots \\ e(k) \end{bmatrix}$$

Feedback Error Correction

Feedforward term: new measurement

(Assume $\Delta d(k+1) = \ldots = \Delta d(k+p-1) = 0$)

Summary

- **Regardless of model form, one gets the prediction equation in the form of**

$$\underbrace{\begin{bmatrix} y_{k+1|k} \\ \vdots \\ y_{k+p|k} \end{bmatrix}}_{Y(k)} = \underbrace{L^x x(k) + L^d \Delta d(k) + L^e e(k)}_{known \equiv b(k)} + L^u \underbrace{\begin{bmatrix} \Delta u(k) \\ \vdots \\ \Delta u(k+p-1) \end{bmatrix}}_{\Delta U(k)}$$

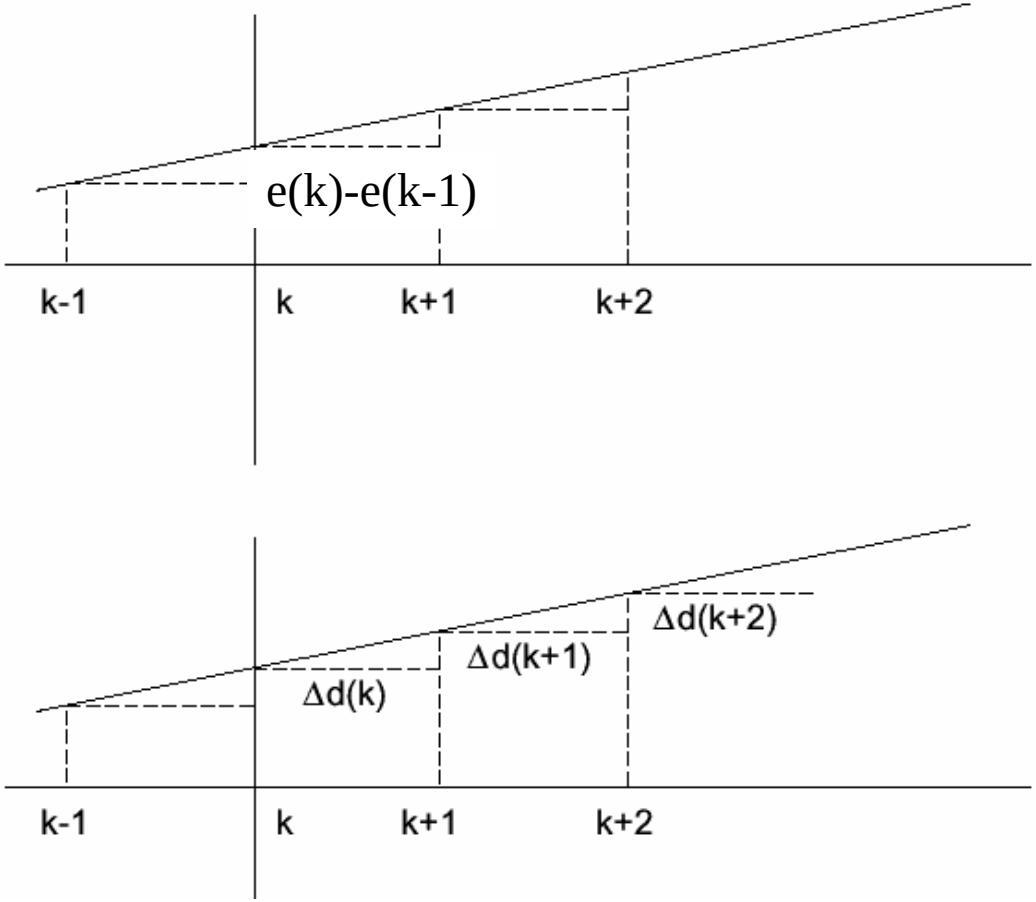
- **Assumptions**
  - Measured DV (d) remains constant at the current value of d(k)
  - Model prediction error (e) remains constant at the current value of e(k)

Ramp Type Extrapolation

- For Integrating Processes, Slow Dynamics

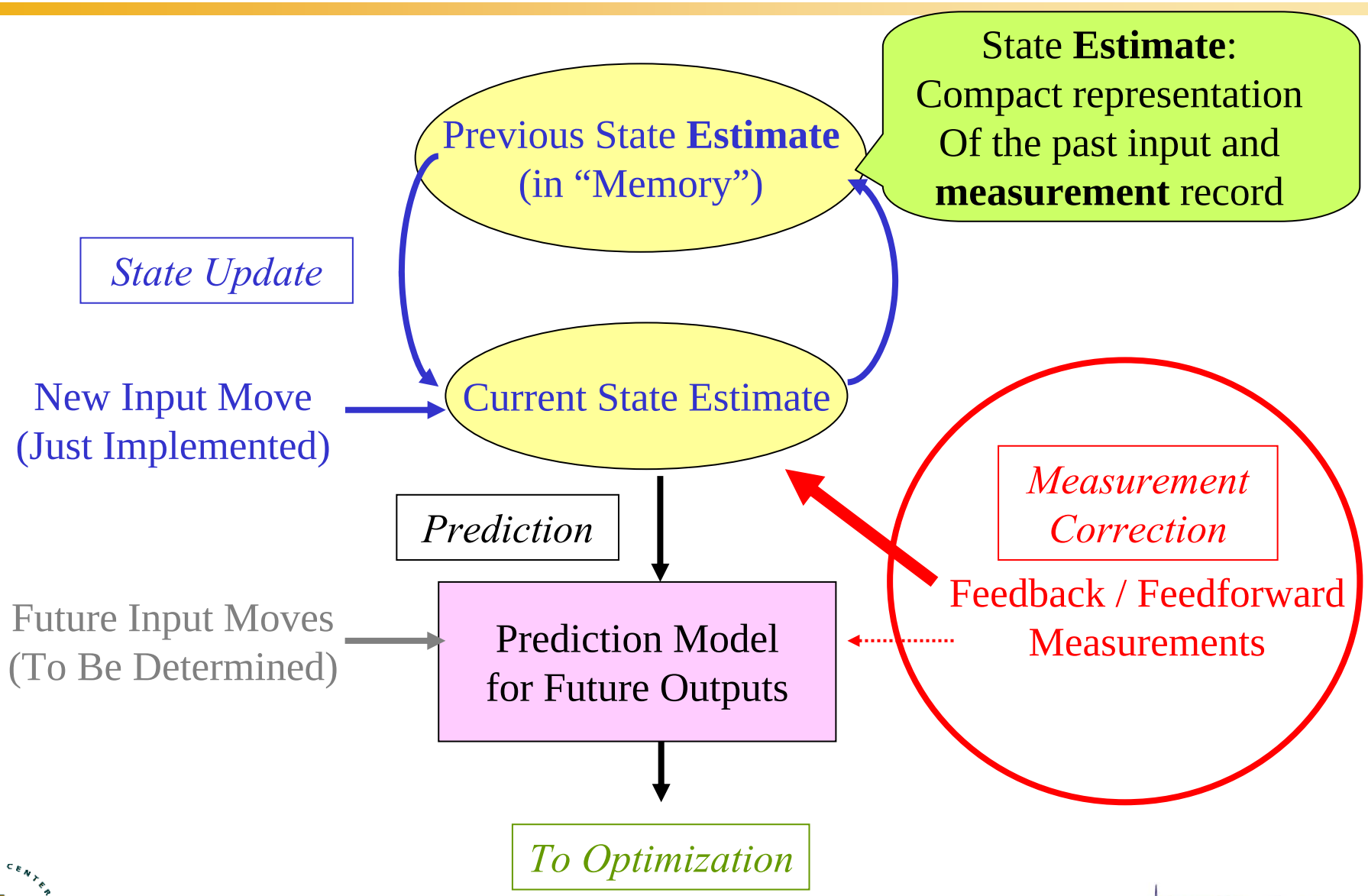
$$e(k+i) = e(k) + i(e(k) - e(k-1))$$

$$\Delta d(k) = \Delta d(k+1) = \cdots = \Delta d(k+p-1)$$



Use of State Estimation

Measurement Correction of State



State Update Equation

$$x(k+1) = \Phi x(k) + \Gamma^u \Delta u(k) + \Gamma^d \Delta d(k)$$
$$+ K(y_m(k) - \Xi x(k))$$

- K is the update gain matrix that can be found in various ways
  - *Pole placement*: Not so effective with systems with many states (most chemical processes)
  - *Kalman filtering*: Requires a **stochastic** model of form

$$x(k+1) = \Phi x(k) + \Gamma^u \Delta u(k) + \Gamma^d \Delta d(k) + w(k)$$
$$y(k) = \Xi x(k) + v(k)$$

Can be obtained using, e.g., subspace ID

White noises of known covariances
Effect of unmeasured disturbances and noise

Prediction Equation

Contains past feedback measurement corrections

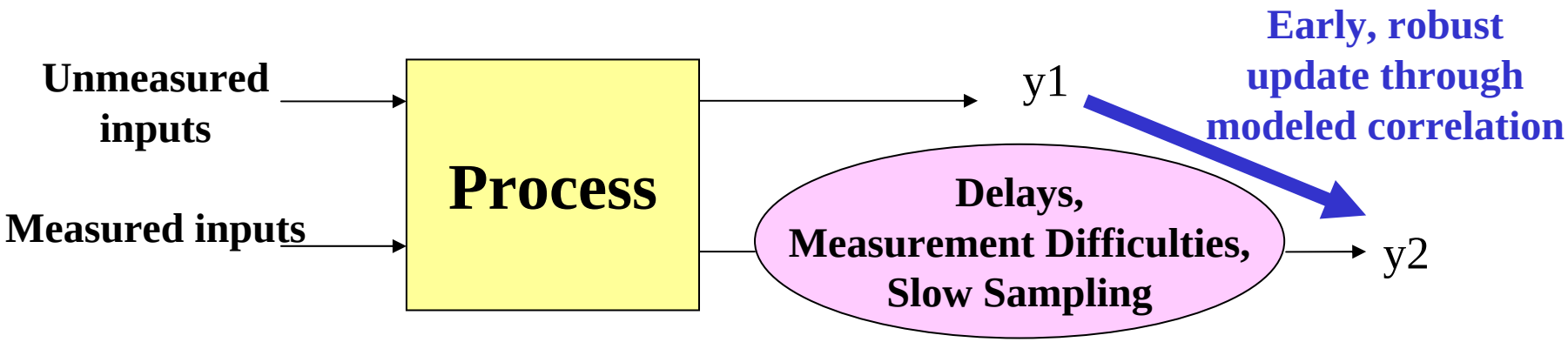
$$\begin{bmatrix} y_{k+1|k} \\ \vdots \\ y_{k+p|k} \end{bmatrix} = \begin{bmatrix} \Xi\Phi \\ \vdots \\ \Xi\Phi^{p} \end{bmatrix} x(k) + \Omega^{u} \begin{bmatrix} \Delta u(k) \\ \vdots \\ \Delta u(k+p-1) \end{bmatrix}$$

$$\Omega^{d} \begin{bmatrix} \Delta d(k) \\ \vdots \\ \Delta d(k+p-1) \end{bmatrix} + \begin{bmatrix} e(k) \\ \vdots \\ e(k) \end{bmatrix}$$

Additional measurement correction NOT needed here!

What Are the Potential Advantages?

- **Can handle** unstable **processes**
  - Integrating processes, run-away processes
- Cross-channel **measurement update**
  - More effective update of output channels with delays or measurement problems based on other channels.



- **Systematic handling of** multi-rate **measurements**
- **Optimal** extrapolation of output error **and** filtering of noise **(based on the given stochastic system model)**

Optimization

Objective Function

- **Minimization Function: Quadratic cost (as in DMC)**

$$V(k)=\sum_{i=1}^{p}(y_{k+i|k}-y^*)^T\Lambda^y(y_{k+i|k}-y^*)+\sum_{i=0}^{m-1}\Delta u^T(k+i)\Lambda^u\Delta u(k+i)$$

  - Consider only m input moves by assuming $\Delta u(k+j)=0$ for $j\geq m$
  - Penalize the tracking error as well as the magnitudes of adjustments
- **Use the prediction equation.**

First m columns of L^u

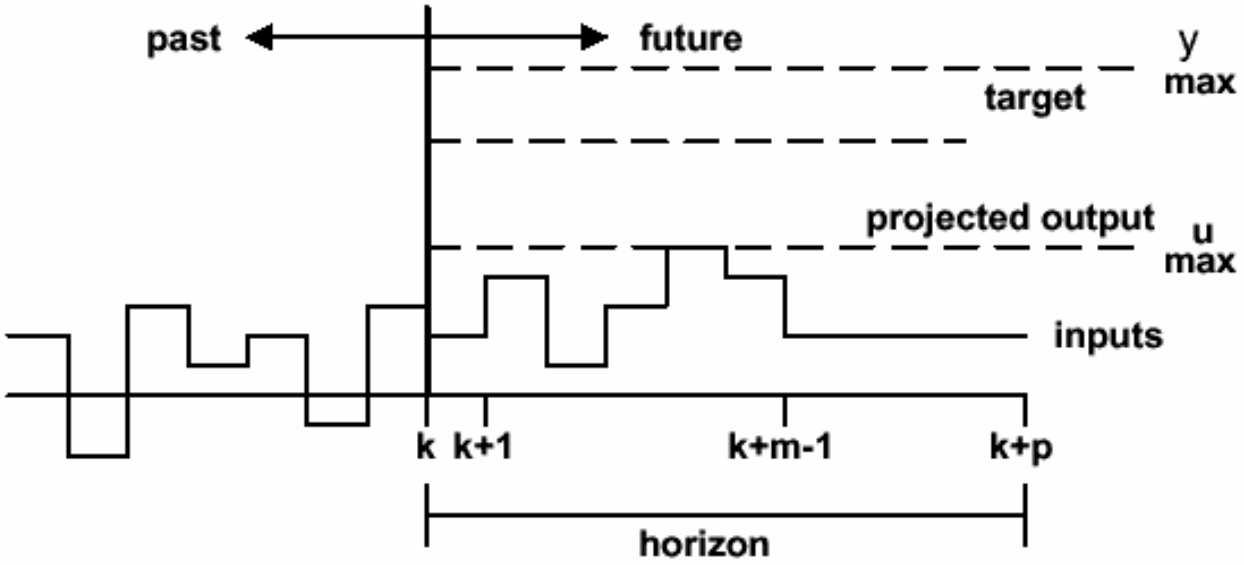
$$V(k)=\left(Y(k)-Y^*\right)^{\mathrm{T}}\mathrm{diag}(\Lambda^y)\left(Y(k)-Y^*\right)+\Delta U_m^T(k)\mathrm{diag}(\Lambda^u)\Delta U_m(k)$$

Substitute $Y(k)=b(k)+L_m^u\Delta U_m(k)$

$$V(k)=\Delta U_m^T(k)H\Delta U_m(k)+g^T(k)\Delta U_m(k)+c(k)$$

constant

Constraints



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

Substitute the prediction equation and rearrange to

$$C\Delta U_m(k) \ge h(k)$$

Optimization Problem

- **Quadratic Program**

$$\min_{\Delta U_m(k)} \Delta U_m^T(k) H \Delta U_m(k) + g^T(k) \Delta U_m(k)$$

$$\text{such that } C\Delta U_m(k) \geq h(k)$$

- **Unconstrained Solution**

$$\Delta U_m(k) = -\frac{1}{2} H^{-1} g(k)$$

- **Constrained Solution**
  - Must be solved numerically

Quadratic Program

- **Minimization of a quadratic function subject to linear constraints**
- **Convex and therefore *fundamentally tractable***
- **Solution methods**
  - Active set method: Determination of the active set of constraints on the basis of the KKT condition
  - Interior point method: Use of barrier function to “trap” the solution inside the feasible region, Newton iteration
- **Solvers**
  - Off-the-shelf software, e.g., QPSOL
  - Customization is desirable for large-scale problems

Two-Level Optimization

Steady-State Optimization (Linear Program)

$$\min_{u_s(k)} L(y_{\infty|k}, u_s(k))$$

$$C_s \begin{bmatrix} y_{\infty|k} \\ u_s(k) \end{bmatrix} \geq c_s(k)$$

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

$$y_{\infty|k} = b_s(k) + L_s \Delta u_s(k)$$

Optimal Setting Values (setpoints)

$$y^*_{\infty|k}, u^*_s(k)$$

(sometimes input deviations are included in the quadratic objective function)

Steady-State Prediction Eqn.

State
Feedforward Measurement
Feedback Error

To Dynamic Optimization (Quadratic Program)

Dynamic Prediction Eqn.

Use of Infinite Prediction Horizon and Stability

Use of ∞ Prediction Horizon – Why?

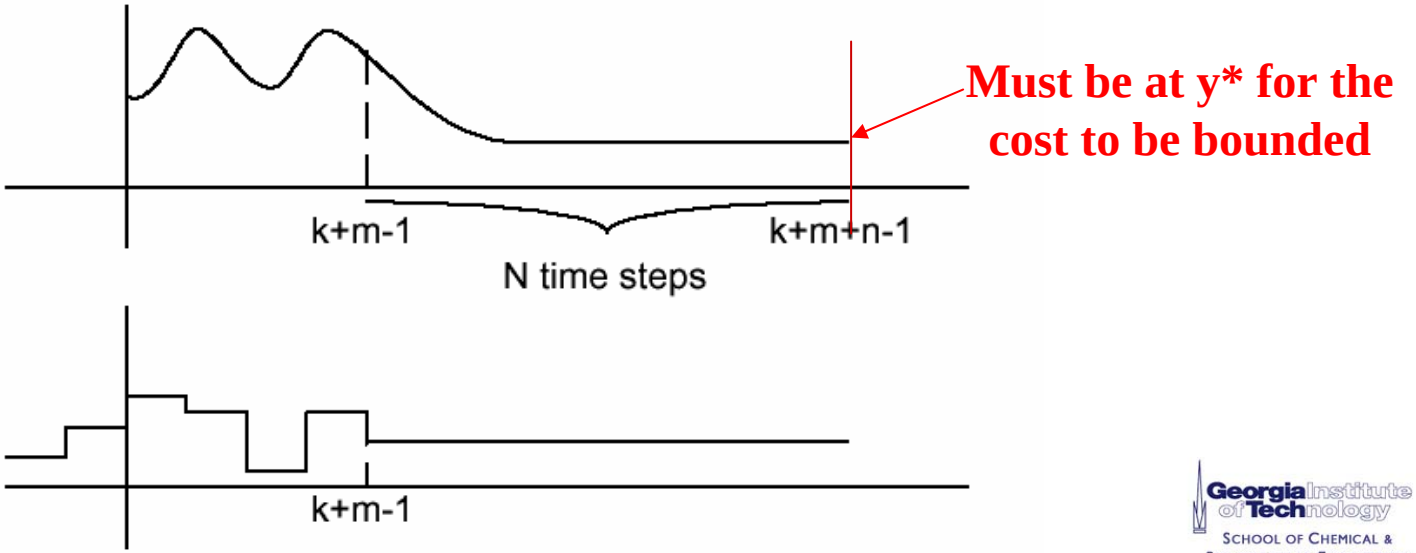
- **Stability guarantee**
  - The optimal cost function can be shown to be the control Lyapunov function
- **Less parameters to tune**
- **More consistent, intuitive effect of weight parameters**
- **Close connection with the classical optimal control methods, e.g., LQG control**

Step Response Model Case

$$V(k) = \sum_{i=1}^{\infty} (y_{k+i|k} - y^*)^T \Lambda^y (y_{k+i|k} - y^*) + \sum_{i=0}^{m-1} \Delta u^T (k+i) \Lambda^u \Delta u(k+i)$$

$$V(k) = \sum_{i=1}^{m+n-1} (y_{k+i|k} - y^*)^T \Lambda^y (y_{k+i|k} - y^*) + \sum_{i=0}^{m-1} \Delta u^T (k+i) \Lambda^u \Delta u(k+i)$$

with extra constraint $y_{k+m+n-1|k} = y^*$



Additional Comments

- **Previously, we assumed** finite **settling time.**
- **Can be generalized to state-space models**
  - More complicated procedure to turn the ∞-horizon problem into a finite horizon problem
  - Requires solving Lyapunov equation to get the terminal cost matrix
  - Also, must make sure that output constraints will be satisfied beyond the finite horizon $\rightarrow$ construction of output admissible set
- **Use of a *sufficiently large* horizon (p$\approx$ m+ the settling time) should have a similar effect**
- **Can we always satisfy the settling constraint?**
  - y=y* may not be feasible due to input constraints or insufficient m $\rightarrow$ use two-level approach

Two-Level Optimization

Steady-State Optimization
(Linear Program or Quadratic Program)

Optimal Setting Values (setpoints)

$$y^*_{\infty|k}, u^*_s(k)$$

Dynamic Optimization (∞-horizon MPC)

Constraint $y_{k+m+n-1|k} = y^*_{\infty|k}$ is guaranteed to be feasible.

Constraint $\Delta u(k) + \cdots + \Delta u(k+m-1) = \Delta u^*_s \rightarrow y_{k+m+n-1|k} = y^*_{\infty|k}$

Use of Nonlinear Model

Difficulty (1)

$$\dot{x} = f(x,u,d)$$
$$y = g(x)$$

Discretization?

$$x(k+1) = F(x(k),u(k),d(k))$$
$$y(k) = g(x(k))$$

Orthogonal Collocation

$$y_{k+1|k} = g \circ F(x(k),u(k),d(k)) + e(k)$$
$$y_{k+2|k} = g \circ F\big(F(x(k),u(k),d(k)),u(k+1),d(k)\big) + e(k)$$
$$\vdots$$
$$y_{k+p|k} = g \circ F^{p}\big(x(k),u(k),\cdots u(k+p-1),d(k)\big) + e(k)$$

The prediction equation is nonlinear w.r.t. u(k), ……, u(k+p-1)

↓

Nonlinear Program (Not so nice!)

Difficulty (2)

State Estimation

$$\dot{x} = f(x,u,d) + w$$

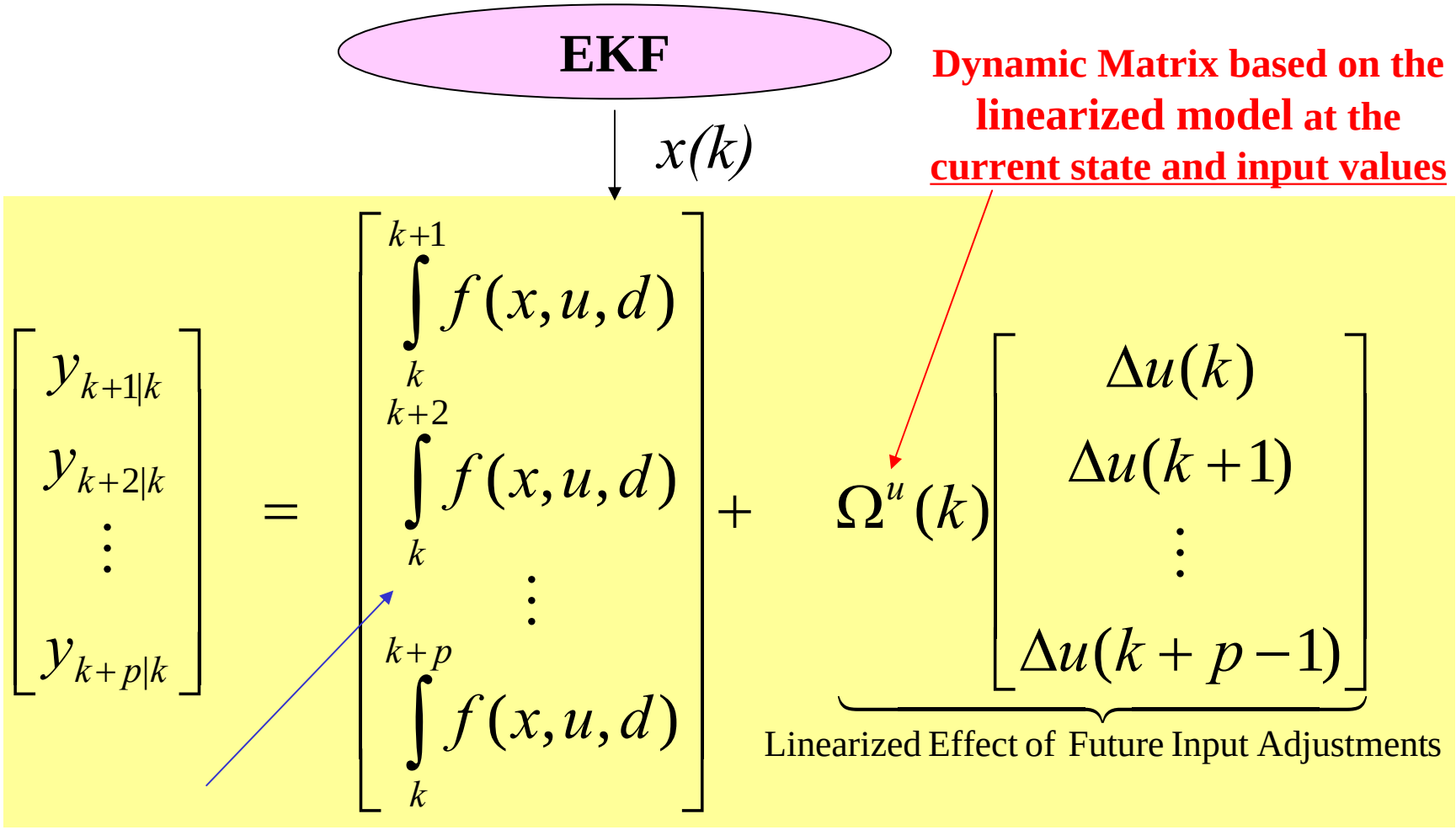
$$y = g(x) + v$$

Extended Kalman Filtering

$$x(k+1) = \int_{k}^{k+1} f(x,u,d) + K(k)\left(y_m(k) - g(x(k))\right)$$

- Computationally more demanding steps, e.g., calculation of K at each time step
- Based on linearization at each time step – not optimal, may not be stable
- Best practical solution at the current time
- Promising alternative: Moving Horizon Estimation (requires solving NLP)
- Difficult to come up with an appropriate stochastic system model (no ID technique)

Practical Algorithm



Additional Comments / Summary

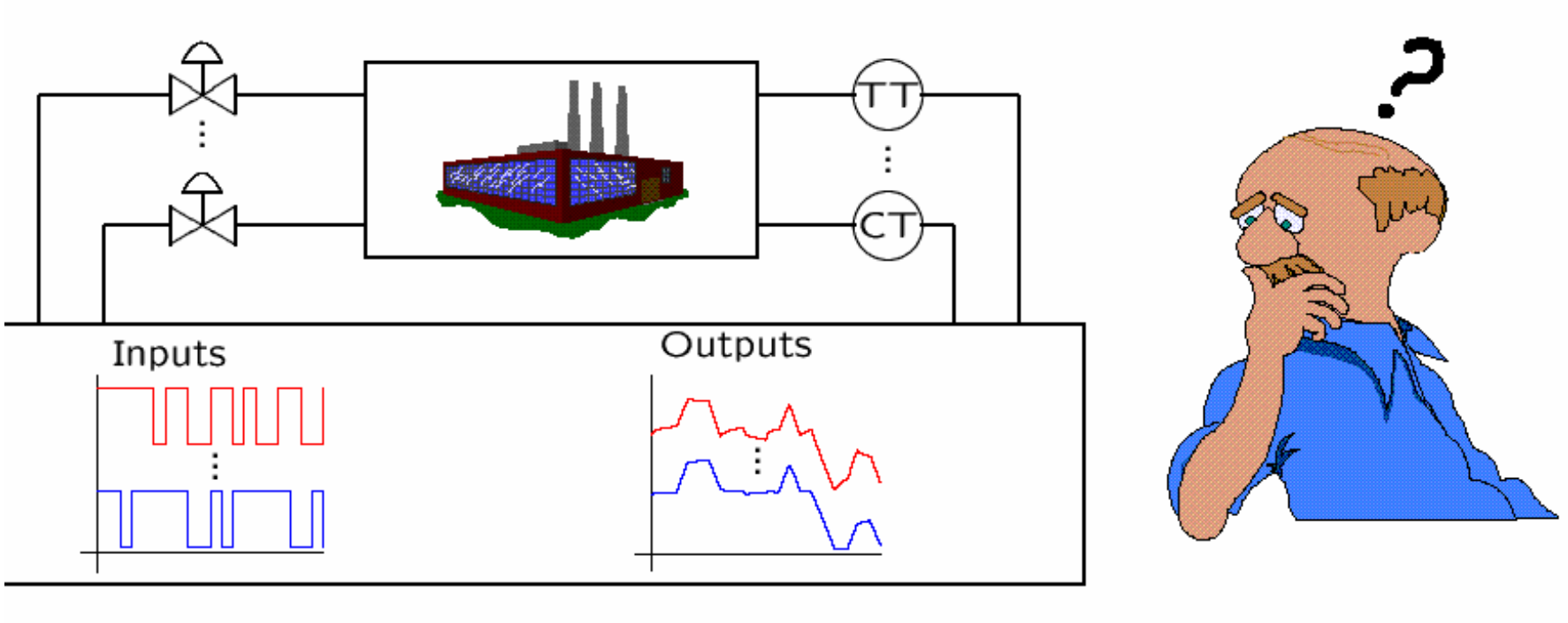
- **Some refinements to the "Practical Algorithm" are possible**
  - Use the previously calculated input trajectory (instead of the constant input) in the integration and linearization step
  - Iterate between integration/linearization and control input calculation
- **Full-blown nonlinear MPC is still computationally prohibitive in most applications**

Lecture 3

Linear Model Identification

- Model structure
- Parameter/model estimation
- Error analysis
- Plant testing
- Data pretreatment
- Model validation

System Identification

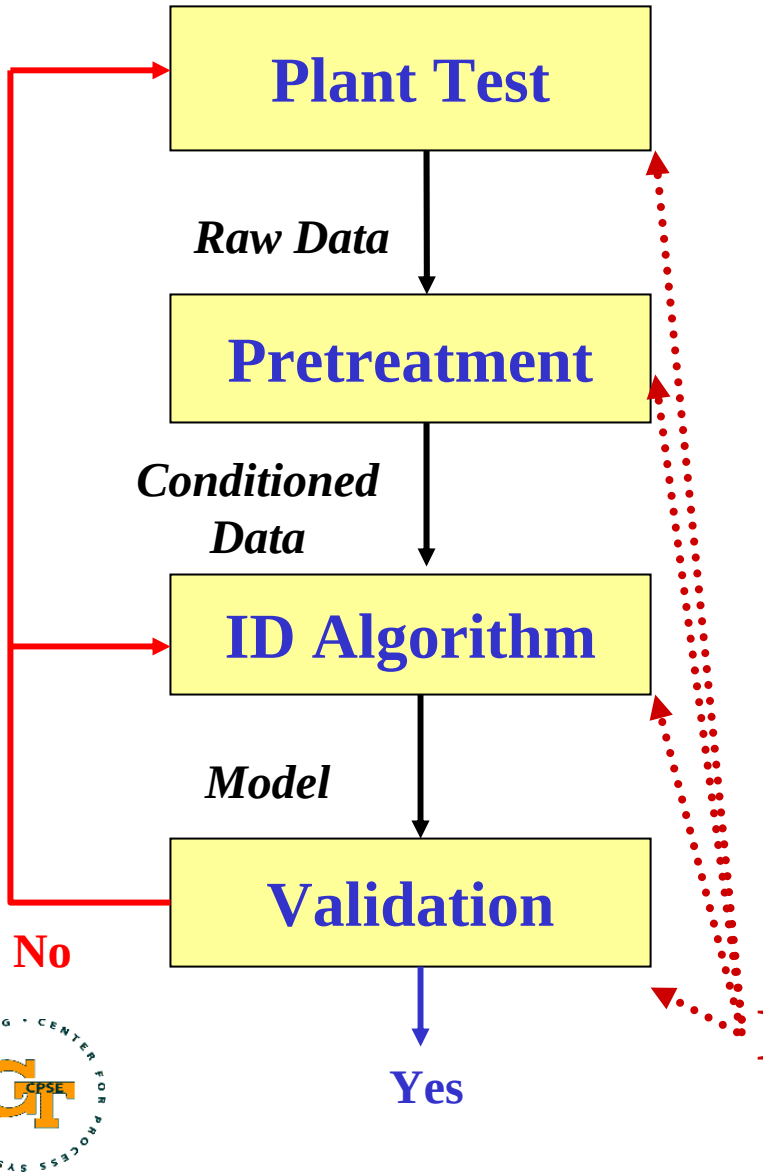


Building a dynamic system model using data obtained from the plant

Why Important?

- **Almost all industrial MPC applications use an empirical mo del obtained through system identification**
- **Poor model → Poor Prediction → Poor Performance**
- **Up to 80% of time is spent on this step**
- **Direct interaction with the plant**
  - Cost factor, safety issues, credibility issue
- **Issues and decisions are sufficiently complicated that syst ematic procedures must be used**

Steps and Decisions Involved



- Test signal (shape, size of perturbation)
- Closed-loop or open-loop?
- One-input-at-a-time or simultaneous?
- How long?
- Etc.

- Outlier removal
- Pre-filtering

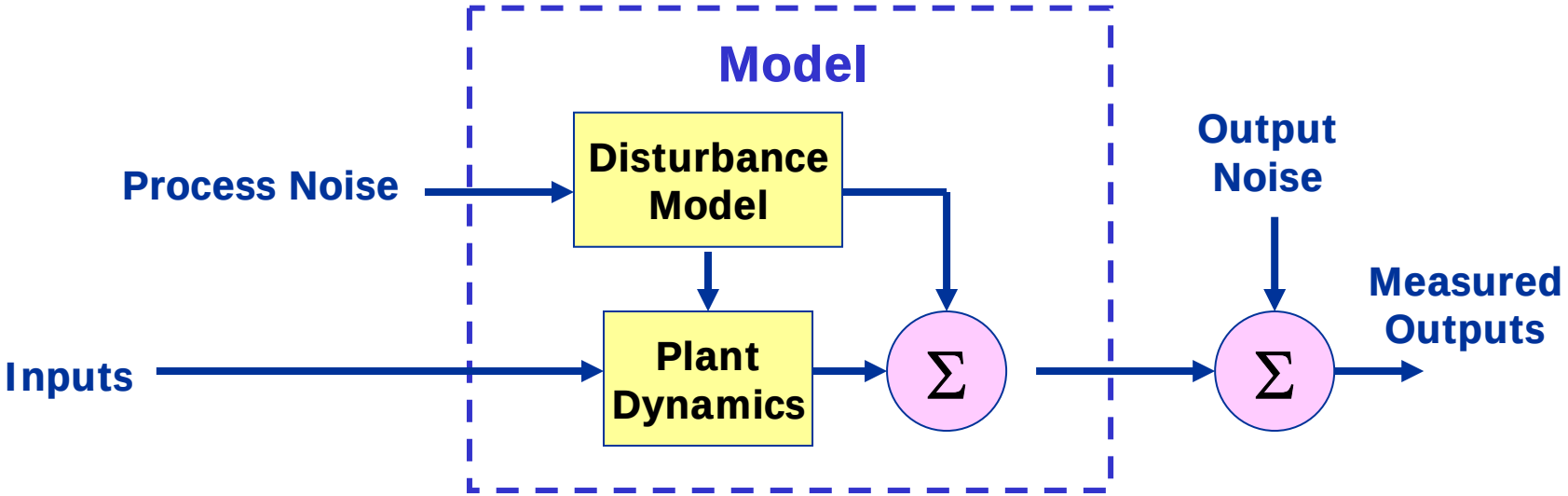
- Model structure
- (Parameter) estimation algorithm

- Source of validation data
- Criterion

End Objective: Control

Model Structure

Model Structure (1)



- **I/O Model**

White noise sequence

$$y(k) = \underbrace{G(q)u(k)}_{\text{effect of inputs}} + \underbrace{H(q)\,e(k)}_{\text{effect of disturbances, noise}}$$

Models auto- and cross-correlations of the residual (not physical cause-effect)

Assume wolog that $H(0)=1$

SISO I/O Model Structure (1)

- **FIR (Past inputs only)**

$$y(k) = b_1 u(k-1) + \cdots + b_m u(k-m) + e(k)$$

$$G(q) = b_1 q^{-1} + \cdots + b_m q^{-m}, \quad H(q) = 1$$

- **ARX (Past inputs and outputs: "Equation Error")**

$$y(k) = a_1 y(k-1) + \cdots + a_n y(k-n) + b_1 u(k-1) + \cdots + b_m u(k-m) + e(k)$$

- **ARM**

$$G(q) = \frac{b_1 q^{-1} + \cdots + b_m q^{-m}}{1 - a_1 q^{-1} - \cdots - a_n q^{-n}}, \quad H(q) = \frac{1}{1 - a_1 q^{-1} - \cdots - a_n q^{-n}}$$

$$y(k) = a_1 y(k-1) + \cdots + a_n y(k-n) + b_1 u(k-1) + \cdots + b_m u(k-m) + e(k) + c_1 e(k-1) + \cdots + c_n e(k-n)$$

$$G(q) = \frac{b_1 q^{-1} + \cdots + b_m q^{-m}}{1 - a_1 q^{-1} - \cdots - a_n q^{-n}}, \quad H(q) = \frac{1 + c_1 q^{-1} + \cdots + c_n q^{-n}}{1 - a_1 q^{-1} - \cdots - a_n q^{-n}}$$

SISO I/O Model Structure (2)

- **Output Error (No noise model or white noise error)**

$$\tilde{y}(k) = a_1\tilde{y}(k-1) + \cdots + a_n\tilde{y}(k-n) + b_1u(k-1) + \cdots + b_mu(k-m);$$
$$y(k) = \tilde{y}(k) + e(k)$$

$$G(q) = \frac{b_1q^{-1} + \cdots + b_mq^{-m}}{1 - a_1q^{-1} - \cdots - a_nq^{-n}}, \quad H(q) = 1$$

- **Box Jenkins (More general than ARMAX)**

$$G(q) = \frac{b_1q^{-1} + \cdots + b_mq^{-m}}{1 - a_1q^{-1} - \cdots - a_nq^{-n}}, \quad H(q) = \frac{1 + c_1q^{-1} + \cdots + c_nq^{-n}}{1 - d_1q^{-1} - \cdots - d_nq^{-n}}$$

MIMO I/O Model Structure

- **Inputs and outputs are vectors. Coefficients are matrices.**
- **For example, ARX model becomes**

$$y(k) = A_1 y(k-1) + \cdots + A_n y(k-n) + B_1 u(k-1) + \cdots + B_m u(k-m) + e(k)$$

$$G(q) = \left(I - A_1 q^{-1} - \cdots - A_n q^{-n}\right)^{-1} \left(B_1 q^{-1} + \cdots + B_m q^{-m}\right)$$

$$H(q) = \left(I - A_1 q^{-1} - \cdots - A_n q^{-n}\right)^{-1}$$

- ***Identifiability* becomes an issue**
  - Different sets of coefficient matrices giving exactly same $G(q)$ and $H(q)$ through pole/zero cancellations → Problems in parameter estimation → Requires special parameterization to avoid problem

A_i is an $n_y \times n_y$ matrix and B_i is an $n_y \times n_u$ matrix

State Space Model

- **Deterministic**

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + e(k)$$

$\Leftrightarrow$ **Output Error Structure**

- **Combined Deterministic / Stochastic**

$$x(k+1) = Ax(k) + Bu(k) + Ke(k)$$
$$y(k) = Cx(k) + e(k)$$

$\Leftrightarrow$ **ARMAX Structure**

$$x_d(k+1) = Ax_d(k) + Bu(k)$$
$$y_d(k) = Cx(k)$$

+

$$x_s(k+1) = Ax_s(k) + Ke(k)$$
$$y_s(k) = Cx_s(k) + e(k)$$

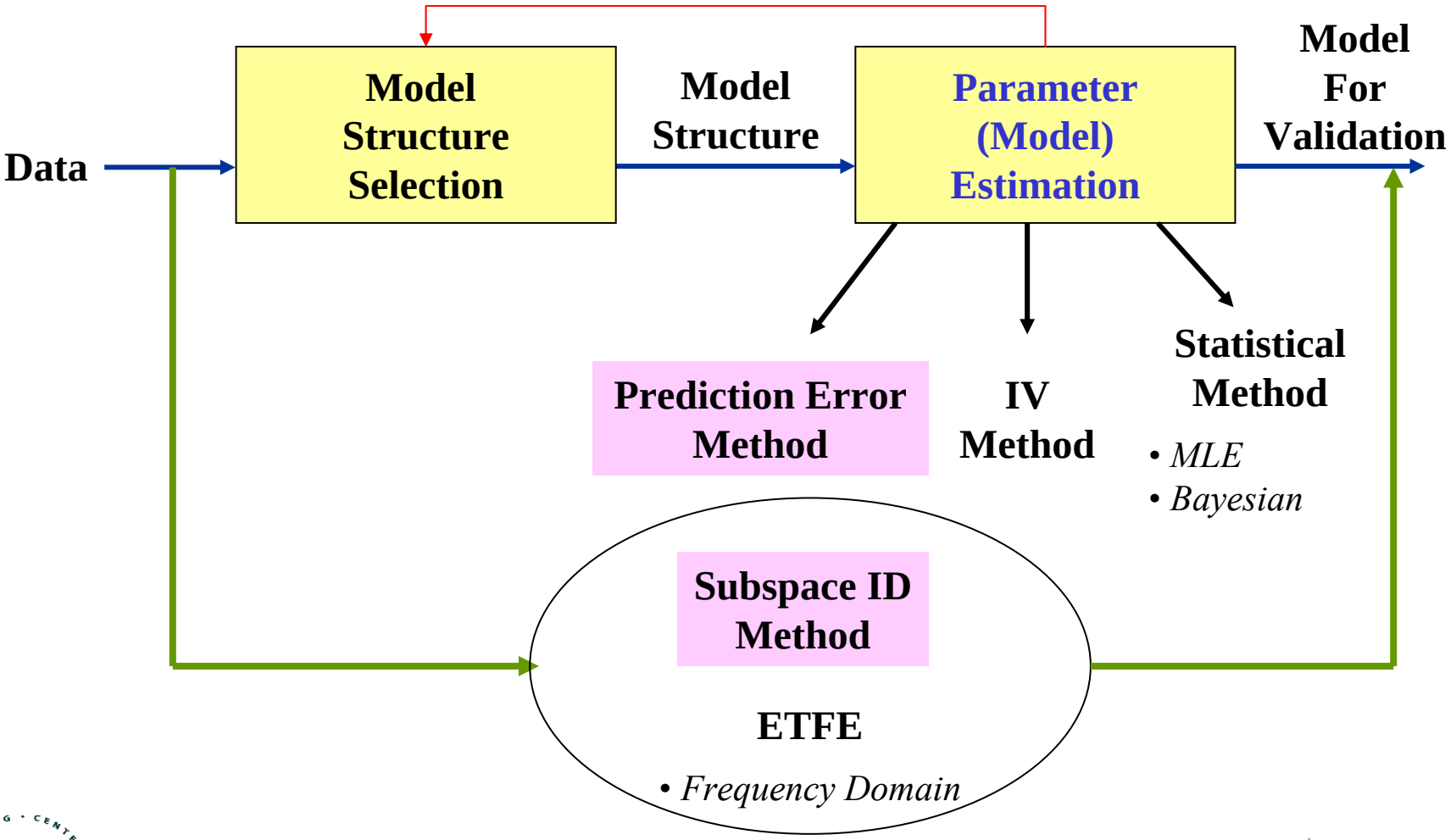
Effect of deterministic input

Auto- and cross-correlation of the residual

- ***Identifiability* can be an issue here too**
  - State coordinate transformation does not change the I/O relationship

Parameter (Model) Estimation

Overview


Data
Model Structure Selection
Model Structure
Parameter (Model) Estimation
Model For Validation
Prediction Error Method
IV Method
Statistical Method
• MLE
• Bayesian
Subspace ID Method
ETFE
• Frequency Domain

Prediction Error Method

- **Predominant method at current time**
- **Developed by Ljung and coworkers**
- **Flexible**
  - Can be applied to any model structure
  - Can be used in recursive form
- **Well developed theories and software tools**
  - Book by Ljung, System ID Toolbox for MATLAB
- **Computational complexity depends on the model st ructure**
  - ARX, FIR $\rightarrow$ Linear least squares
  - ARMAX, OE, BJ $\rightarrow$ Nonlinear optimization
- **Not easy to use for identifying *multivariable* models**

Prediction Error Method

- **Put the model in the predictor form**

$$y(k) = G(q,\theta)u(k) + H(q,\theta)e(k) \quad \rightarrow$$

$$y_{k|k-1} = G(q,\theta)u(k) + \underbrace{\left(I - H^{-1}(q,\theta)\right)}_{\text{Contains at least 1 delay}}\left(y(k) - G(q,\theta)u(k)\right)$$

$$e(k) = y(k) - y_{k|k-1} = H^{-1}(q,\theta)\left(y(k) - G(q,\theta)u(k)\right)$$

- **Choose the parameter values to minimize the sum of the pr ediction error for the given data**

$$\min_{\theta}\left\{\frac{1}{N}\sum_{k=1}^{N}\left\|e(k)\right\|_2^2\right\}$$

$$e(k) = H^{-1}(q,\theta)\left(y(k) - G(q,\theta)u(k)\right)$$

  - ARX, FIR $\rightarrow$ Linear least squares,
  - ARMAX, OE, BJ $\rightarrow$ Nonlinear least squares

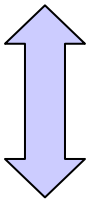
Subspace Method

- **More recent development**
- **Dates back to the classical realization theories but rediscov ered and extended by several people**
- **Identifies a state-space model**
- **Some theories and software tools**
- **Computationally simple**
  - Non-iterative, linear algebra
- **Good for identifying *multivariable* models**
  - No special parameterization is needed
- **Not optimal in any sense**
- **May need a lot of data for good results**
- **May be combined with PEM**
  - Use SS method to obtain an initial guess for PEM

Main Idea of the SS-ID Method (1)

Assumed Form of the Underlying Plant

$$x(k+1) = Ax(k) + Bu(k) + w(k)$$
$$y(k) = Cx(k) + v(k)$$



Innovation Form

(Steady-State Kalman Filter)

Equivalent to the above in I/O sense

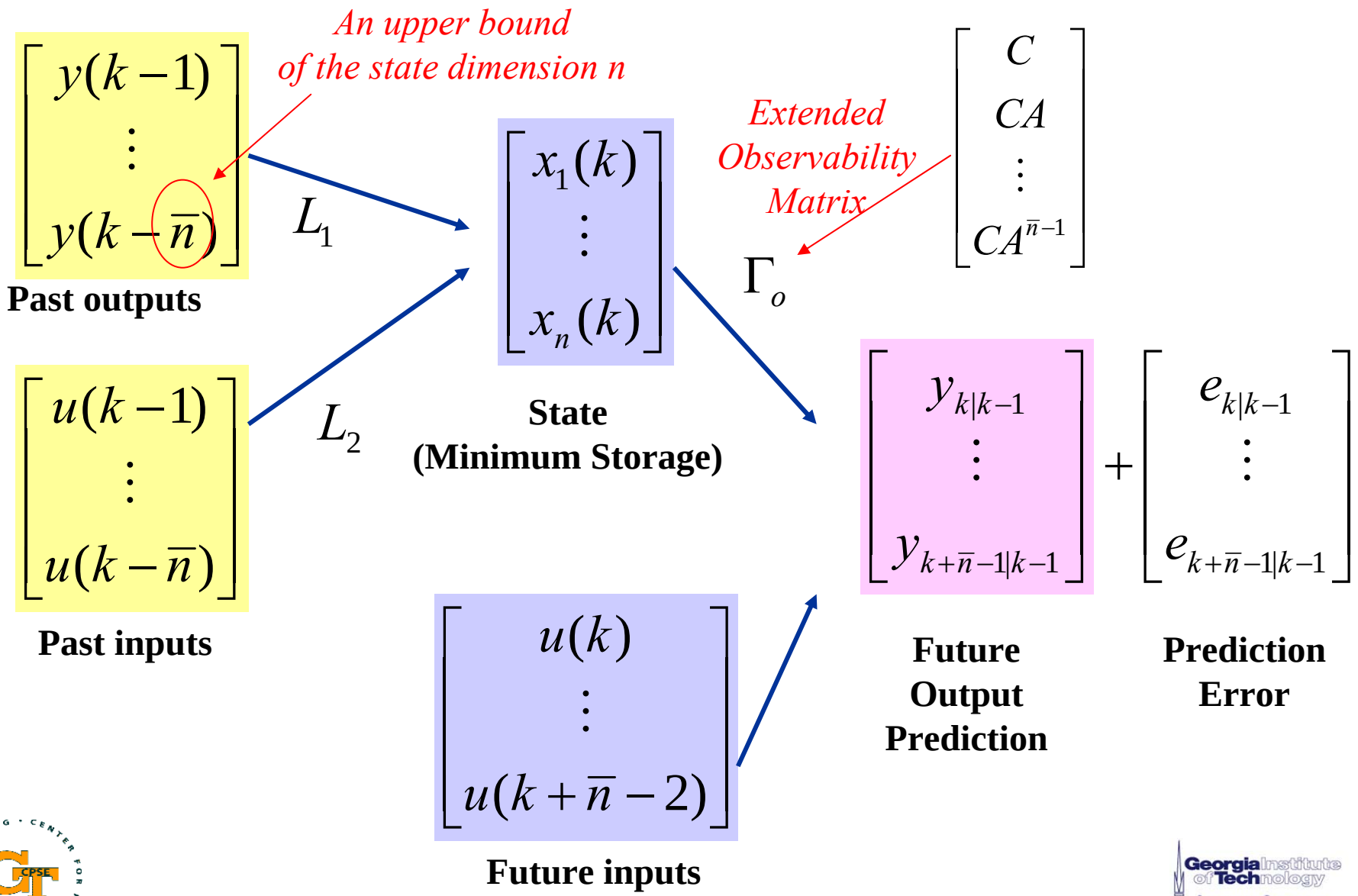
$$x(k+1) = Ax(k) + Bu(k) + Ke(k)$$
$$y(k) = Cx(k) + e(k)$$

Identify *{A, B, C, K, Cov(e)}*

within *some similarity transformation*

We are free to choose the state coordinates

Main Idea of the SS-ID Method (2)



$$\begin{bmatrix} y(k-1) \\ \vdots \\ y(k-\bar{n}) \end{bmatrix} \quad \begin{bmatrix} u(k-1) \\ \vdots \\ u(k-\bar{n}) \end{bmatrix} \quad L_1 \quad L_2 \quad \begin{bmatrix} x_1(k) \\ \vdots \\ x_n(k) \end{bmatrix} \quad \Gamma_o \quad \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{\bar{n}-1} \end{bmatrix}$$

$$\begin{bmatrix} u(k) \\ \vdots \\ u(k+\bar{n}-2) \end{bmatrix} \quad \begin{bmatrix} y_{k|k-1} \\ \vdots \\ y_{k+\bar{n}-1|k-1} \end{bmatrix} + \begin{bmatrix} e_{k|k-1} \\ \vdots \\ e_{k+\bar{n}-1|k-1} \end{bmatrix}$$

Georgia Institute of Technology
School of Chemical & Biomolecular Engineering

Main Idea of the SS-ID Method (3)

$$Y^{0+} = \underbrace{\Gamma_o \begin{bmatrix} L_1 & L_2 \end{bmatrix}}_{M} \begin{bmatrix} Y^- \\ U^- \end{bmatrix} + L_3 U^{0+} + E^{0+}$$

- **Find M through linear least squares**
  - Consistent estimation since E^{0+} is independent of the regres sors
  - Oblique projection of data matrices
- **Perform SVD on M and find n as well as Γ_o**

$$M = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} \Sigma_n & 0 \\ 0 & \approx 0 \end{bmatrix} \begin{bmatrix} P_1^T \\ P_2^T \end{bmatrix}$$

$$\Gamma_o = Q_1 \Sigma_n^{1/2}$$

The column space of Q1 is the state space

Some variations exist among different algorithms in terms of picking the state basis

Main Idea of the SS-ID Method(4)

- **Obtain the data for *x*(*k*)**

$$x(k) = \underbrace{\Gamma_o^-}_{\text{pseudo inverse}} \begin{bmatrix} Y^-(k) \\ U^-(k) \end{bmatrix}$$

- **Obtain the data for *x*(*k*+1) in a similar manner**
- **Obtain *A*,*B* and *C* through linear regression**
  - Consistent estimation since the residual is independent of the regre ssor

$$\begin{bmatrix} x(k+1) \\ y(k) \end{bmatrix} = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ y(k) \end{bmatrix} + \underbrace{\begin{bmatrix} Ke(k) \\ e(k) \end{bmatrix}}_{\text{Residual}}$$

- **Obtain *K* and *Cov*(*e*) by using the residual data**

Properties

- **N4SID (Van Overschee and DeMoor)**
  - Kalman filter interpretation
  - Proof of asymptotically unbiasedness of A, B, and C
  - Efficient algorithm using QR factorization
  - -
- **CVD (Larimore)**
  - Founded on statistical argument
  - Same idea but the criterion for choosing the state basis (Q_1) diff ers a bit from N4SID – based on “correlation” between past I/O data and future output data, rather than minimization of the pre diction error for the given data

Alternative

$$\Gamma_o \rightarrow C, A \rightarrow B$$

- **MOESP (Verhaegen)**

Error Analysis

Error Types

- Bias: **Error due to structural mismatch**
  - Bias = the error as # of data points $\rightarrow \infty$
  - Independent of # of data points collected
  - Bias distribution (e.g., in the frequency domain) depends on the input spectrum, **pre-filtering** of the data, etc.
  - Frequency-domain bias distribution under PEM - by Ljung
- Variance: **Error due to limited availability of data**
  - Vanishes as # of data points $\rightarrow \infty$
  - Depends on the number of parameters, the number of data poin ts, S/N ratio, etc. but not on pre-filtering
  - Asymptotic distribution (as n, $N \rightarrow \infty$):
- **Main tradeoff**
  - Richer structure (more parameters) $\rightarrow$ Bias$\downarrow$, Variance$\uparrow$

$$\mathrm{cov}(vec(\hat{G}_N)) \approx \underbrace{\frac{n}{N}(\Phi_u)^{-T} \otimes \Phi_d}_{\text{Noise-to-signal ratio}}$$

Plant Test for Data Generation

Test Signals

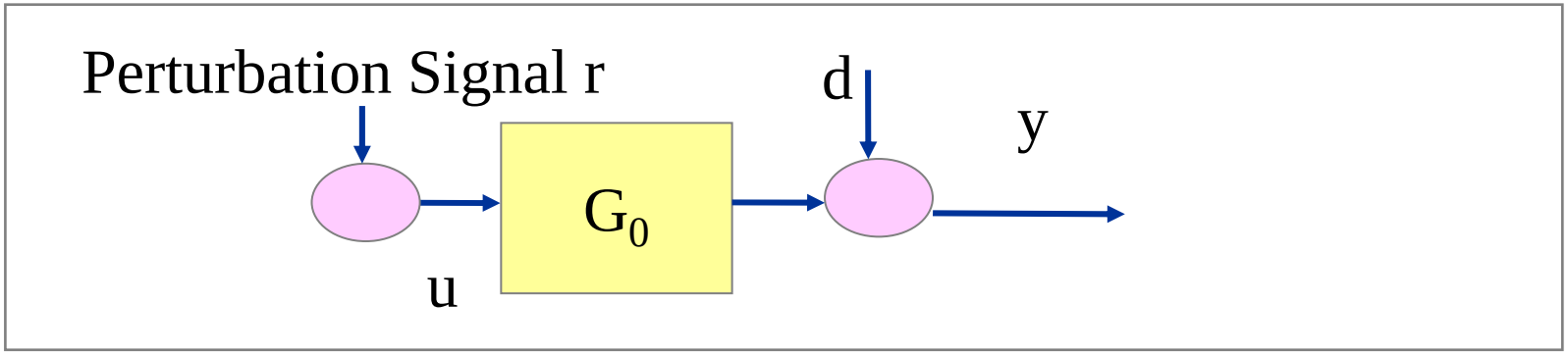
- **Very Important**
  - Signal-to-noise ratio → Distribution and size of the variance
  - Bias distribution
- **Popular Types**
  - Multiple steps: Power mostly in the low-frequency region→ Good e stimation of steady-state gains (even with step disturbances) but g enerally poor estimation of high frequency dynamics
  - PRBS: Flat spectrum → Good estimation of the entire frequency re sponse, given the error also has a flat spectrum (often not true)
  - Combine steps w/ PRBS?

Multi-Input Testing (MIT) vs. Single Input Testing (SIT)

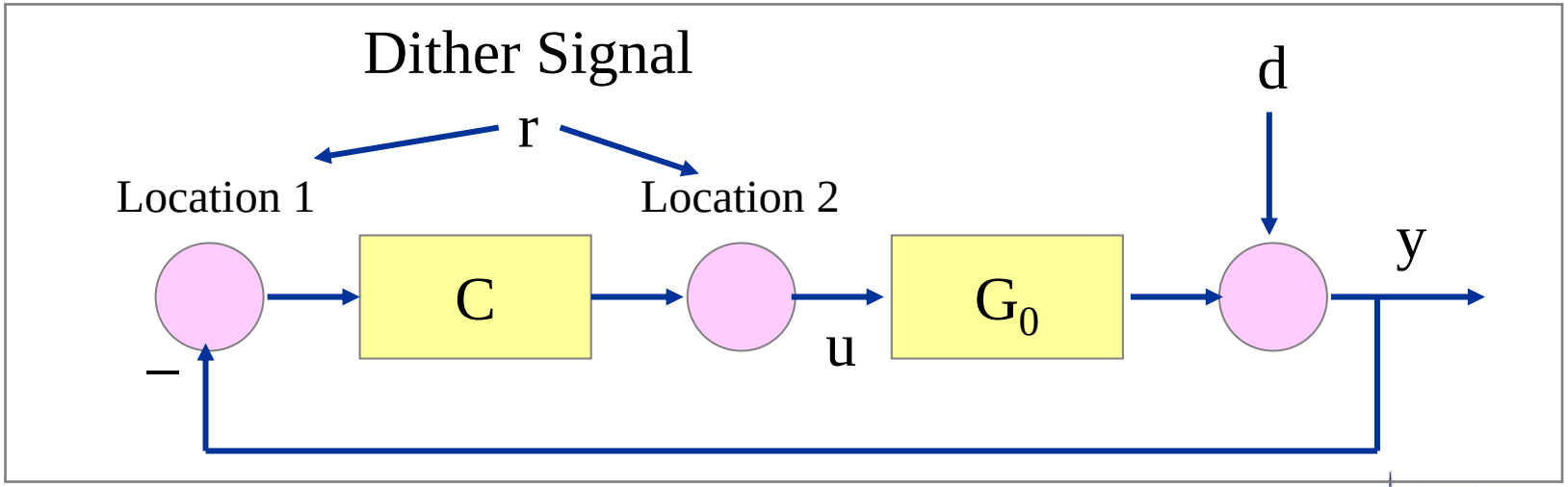
- MIT gives better signal-to-noise ratio for a given t esting time
- Control-relevant data generation requires MIT
- MIT can be necessary for identification of highly i nteractive systems (e.g., systems with large RGA)
- SIT is often preferred in practice because of the m ore predictable effect on the on-going operation

Open-Loop vs. Closed-Loop

Open-Loop Testing



Closed-Loop Testing



Pros and Cons of Closed-Loop Testing

- **Pros**
  - Safer, less damaging effect on the on-going operation
  - Generates data that are more relevant to closed-loop c ontrol
- **Cons**
  - Correlation between input perturbations and disturbanc es / noise through the feedback.
  - Many algorithms can fail or give problems – They give “ bias” unless the assumed noise structure is perfect

Important Points from Analysis

- **External perturbations ("dither") are necessary.**
  - Perturbations due to error feedback hardly contributes to *variance reduction* (since they are correlated to the errors)

$$\mathrm{cov}(vec(\hat{G}_N)) \approx \frac{n}{N}(\Phi_u^r)^{-T} \otimes \Phi_d$$

Portion of the input spectrum due to the dithering

Output error spectrum

  - The level of external perturbation signals also contribute to *the siz e of bias* due to the feedback-induced correlation

$$E\{G_0 - \hat{G}_N\} = (H_0 - H_\mu)\Phi_{eu}\Phi_u^{-1}$$

$$\Phi_{eu}\Phi_u^{-1} = \underbrace{(P_e\Phi_u^{-1})}_{\text{Noise-to-signal ratio}} \times \underbrace{(\Phi_u^e\Phi_u^{-1})}_{\text{Relative contribution of noise feedback to input spectrum}}$$

- **Specialized algorithm may be necessary to avoid bias**

Georgia Institute of Technology
School of Chemical & Biomolecular Engineering

Different Approaches to Model Identification with Closed-Loop Data

- **Direct Approach**

$$D^N = \{y(i), u(i), i = 1, \ldots, N\} \rightarrow \hat{G}_N \text{ (and } \hat{H}_N)$$

- **Indirect Approach**

$$D^N = \{y(i), r(i), i = 1, \ldots, N\} \rightarrow \hat{T}_N^{yr} \xrightarrow{\hat{G}_N = \hat{T}_N^{yr}(I - \hat{T}_N^{yr}C)^{-1}} \hat{G}_N$$

- **Joint I/O Approach**

$$D^N = \{y(i), u(i), r(i), i = 1, \ldots, N\} \rightarrow (\hat{T}_N^{yr}, \hat{T}_N^{ur}) \xrightarrow{\hat{G}_N = \hat{T}_N^{yr}(\hat{T}_N^{ur})^{-1}} \hat{G}_N$$

- **Two-Stage or Projection Approach**

$$D_1^N = \{u(i), r(i), i = 1, \ldots, N\} \rightarrow \begin{matrix} \tilde{D}_1^N = \{u(i)\} \\ \tilde{D}_2^N = \{y(i)\} \end{matrix} \rightarrow \hat{G}_N$$

Data Pretreatment

Main Issues (1)

- **Time-consuming but very important**
- **Remove outliers**
- **Remove portions of data corresponding to u nusual disturbances or operating conditions**
- **Filter the data**
  - Affects bias distribution (emphasize or de-emphasize different frequency regions)
  - Does NOT improve the S/N ratio – often a misconcept ion

Main Issues (2)

- **Difference the data?**
  ($\Delta y = y(k) - y(k\text{-}1)$, $\Delta u(k) = u(k) - u(k\text{-}1)$)
  - Removes trends (e.g., effect of step disturbances, set point changes) that can destroy the effectiveness of m any ID methods (e.g., subspace ID)
  - Often used in practice
  - Also removes the input power in the low-frequency re gion. (PRBS $\rightarrow$ zero input power at $\omega = 0$)
  - Amplifies high-frequency parts of the data (e.g., noise), so low-pass filtering may be necessary

Model Validation

Overview

- **Use fresh data different from the data used for m odel building**
- **Various methods**
  - Size of the prediction error
  - “Whiteness” of the prediction error
  - Cross correlation test (e.g., prediction error and inputs)
- **Good prediction with test data but poor predictio n with validation data**
  - Sign of “overfit”
  - Reduce the order or use more compact structures like ARM AX (instead of ARX)

Concluding Remarks on Linear ID

- **System ID is often the most expensive and difficult part of model-based controller design**
- **Involves many decisions that affect**
  - Plant operation during testing
  - Eventual performance of the controller
- **Good theories and systematic tools are available**
- **System ID can also be used for constructing monitoring models**
  - Subspace identification
  - Trend model, not a causal model
    $\rightarrow$ Active testing is not needed

Deterministic Multi-Stage Optimization

$$\min_{u_0,\ldots,u_{p-1}} \left\{ \sum_{j=0}^{p-1} \phi(x_j, u_j) \; + \; \phi_p\left(x_p\right) \right\}$$

stage-wise cost | terminal cost

$g_j(x_j, u_j) \geq 0$	Path constraints
$g_p(x_p) \geq 0$	Terminal constraints
$x_{j+1} = f(x_j, u_j)$	Model constraints

- **General formulation for deterministic control and scheduling problems.**
- **Continuous and integer state / decision variables possible**
- **In control, $p=\infty$ case is solved typically.**
- **Uncertainty is not explicitly addressed.**

Solution Approaches

- **Analytical approach: 50s-70s**
  - Derivation of closed form optimal policy ($u_j = \mu^*(x_j)$) requires solution to HJB equation (hard!)
- **Numerical approach: 80s-now**
  - **Math programming (LP, QP, NLP, MILP, etc.):**
    - Fixed parameter case solution
    - Computational limitation for large-size problem (e.g., when $p = \infty$).
  - **Parametric programming:**
    - General parameter dependent solution (e.g., a lookup table)
    - Significantly higher computational burden
  - **Practical solution:**
    - Resolve the problem on-line whenever parameters are updated or constraints are violated (e.g., in Model Predictive Control or Reactive Scheduling).

Stochastic Multi-Stage Decision Problem with Recourse

$$\min_{u_j=\mu(x_j)} E\left\{\sum_{j=0}^{\infty}\alpha^j\phi(x_j,u_j)\right\}$$

$$0<\alpha<1 \qquad \text{Discount factor}$$

$$x_{j+1}=f_h(x_j,u_j,\omega_j) \qquad \text{Markov sys. model}$$

$$\Pr\left[g(x_j,u_j)\geq 0\right]\geq\zeta \qquad \text{Chance constraint}$$

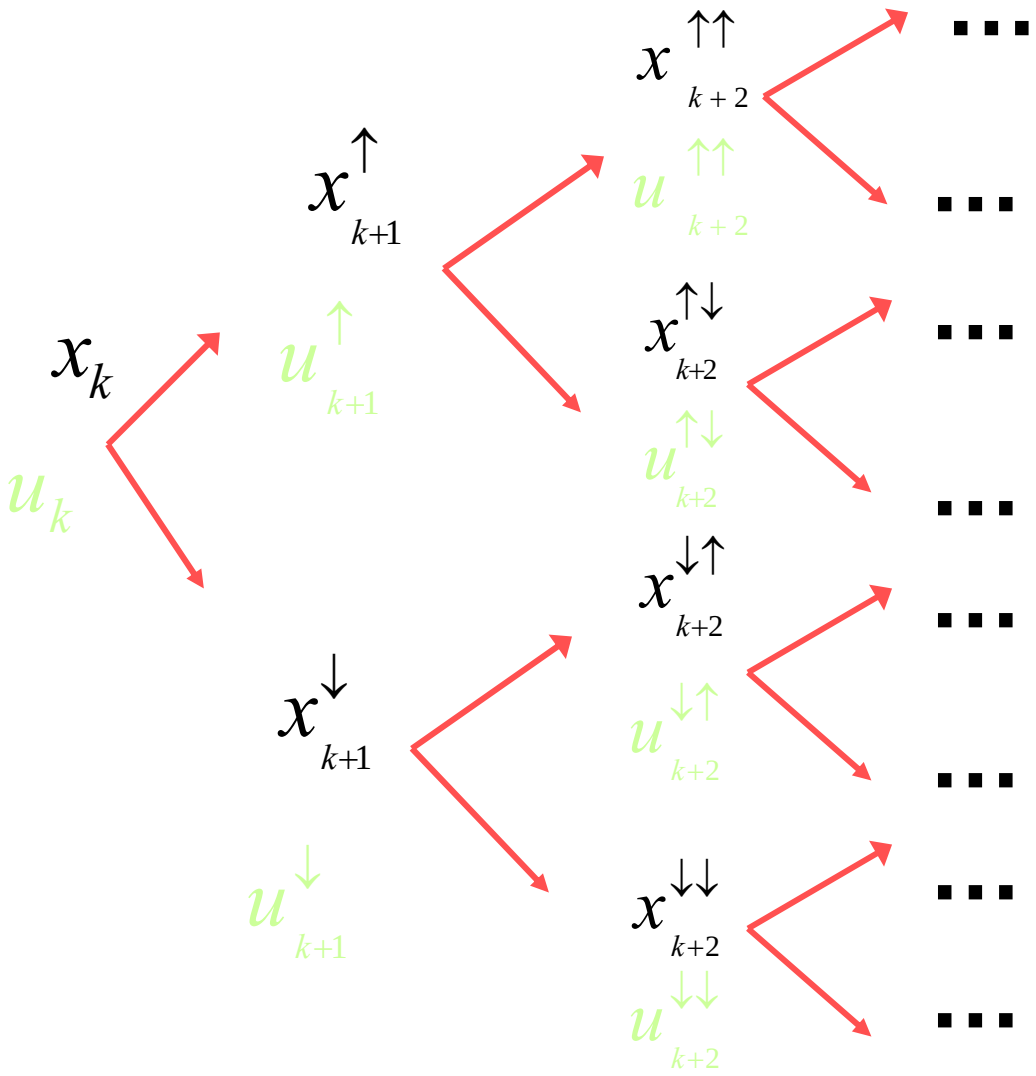
- **Next "holy-grail" of control: A general form for control, scheduling, and other real-time decision problems in an uncertain dynamic environment.**
- **No satisfactory solution approach currently available.**

CENTER FOR PROCESS SYSTEMS ENGINEERING · CPSE

Georgia Institute of Technology
School of Chemical & Biomolecular Engineering

Limitation of Stochastic Programming Approach

Simple case of 2 scenarios (↑ or ↓) per stage



Total number of decision variables $= (1+2+4+\ldots+2^{p-1})\, n_u$

Number of branches to evaluate for each candidate decisions = 2^p

CENTER FOR PROCESS SYSTEMS ENGINEERING · CPSE

Georgia Institute of Technology
School of Chemical & Biomolecular Engineering

Stochastic Programming Approach

General case (S number of scenarios per stage)

- **Total number of decision variables**

  $= (1 + S + S^2 + \ldots + S^{p-1})\, n_u$
- **Number of branches to evaluate for each decision candidate = S^p**
- **Not feasible for large *S* (large number of scenarios) and/or large *p* (large number of stages)**
  - practically limited to two stage problems with a small number of scenarios.
- **Current practical approach: Evaluate most likely branch(es) only. BUT highly limited!**

Georgia Institute of Technology
School of Chemical & Biomolecular Engineering

Dynamic Programming (DP)

ON THE THEORY OF DYNAMIC PROGRAMMING

BY RICHARD BELLMAN

THE RAND CORPORATION, SANTA MONICA, CALIFORNIA

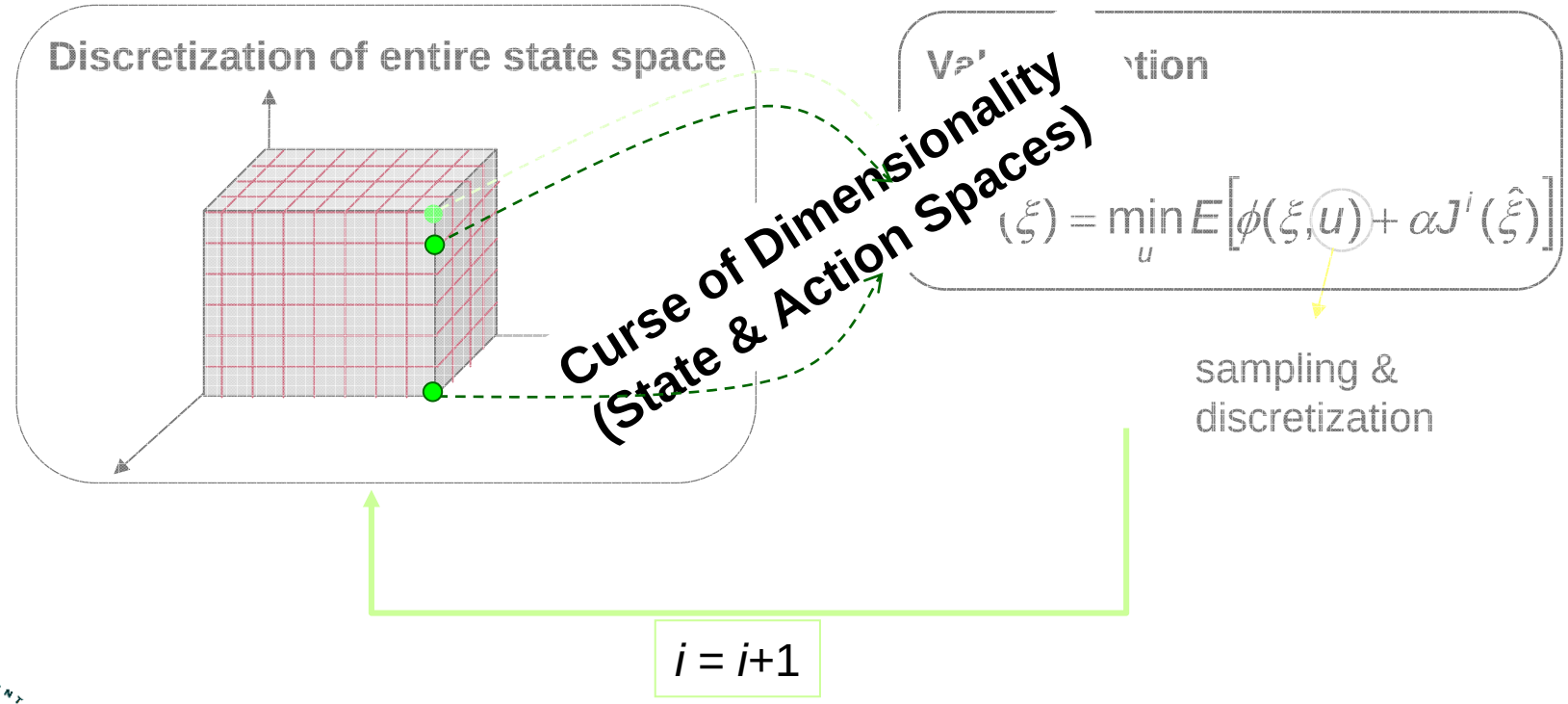
Communicated by J. von Neumann, June 5, 1952

1. Introduction.—We are interested in a class of mathematical problems which arise in connection with situations which require that a bounded or unbounded sequence of operations be performed for the purpose of achieving a desired result. Particularly important are the cases where each operation gives rise to a stochastic event, the result of which is applied to the determination of subsequent operations.

Two fundamental problems encountered in situations of this type, in some sense duals of each other, are those of maximizing the yield obtained

Value Iteration Approach to Solving DP

$$J^*(x_k) = \min_{u(k)} E\left[\phi(x_k, u_k) + \alpha J^*(f_h(x_k, u_k)\right]$$



Approximate Dynamic Programming (ADP)

- **Bellman equation needs to be solved** $\forall x \in \mathcal{X}$
  - Curse of dimensionality! Not suitable for high dimens. sys.
- **Key idea of ADP**
  - To find approximate cost-to-go function
  - Use simulations under known suboptimal policy to sample a very small "relevant" fraction of the states and initialize cost-to-go value table.
  - Iteratively improve the policy and cost-to-go function
    - Iterate over only the sampled points in the state space
    - **Use interpolation** to evaluate the cost-to-go values for non-sampled points.

Approximate Dynamic Programming (ADP)

Approximate Value Iteration

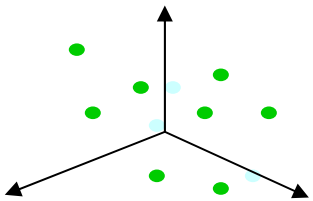
On-line Decision

$$u = \mu(x) = \arg\min_u E\left[\phi(x,u) + \alpha\tilde{J}^*\left(f(x,u)\right)\right]$$

Monte-Carlo Simulations

- Closed-loop w/ suboptimal policies
- MPC, PI, etc.

Sampled State Points



- State and input trajectories: x_k, u_k
- Initial cost-to-go: $J(x) = \sum \alpha^i \phi_i$

Iteration on the Bellman Equation

$$J^{i+1}(x) = \min_{u \in U} E\left[\phi(x,u) + \alpha\tilde{J}^i\left(f(x,u)\right)\right]$$

Sample Avg.

sampled state points

$i \leftarrow i+1$ converged solution $\tilde{J}^*$

Approximation of Cost-to-Go

$$\tilde{J}(x) := x_k \rightarrow J(x_k)$$