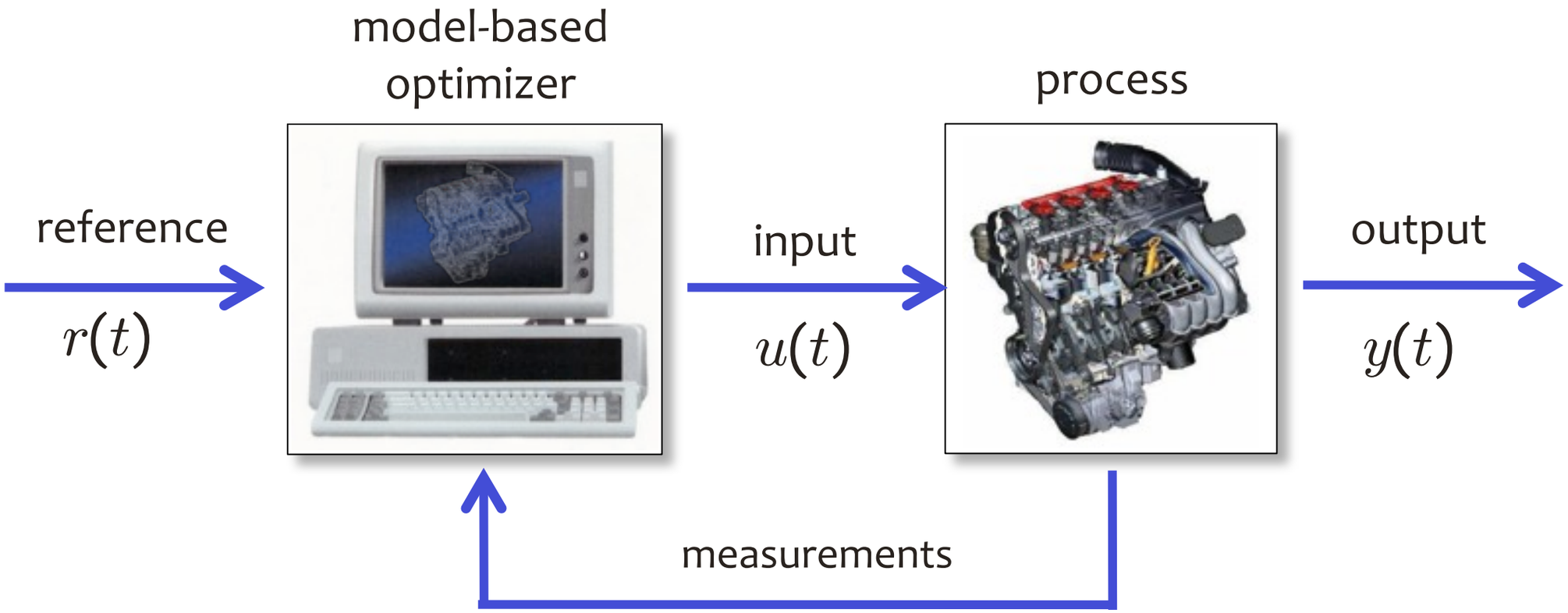
# Model Predictive Control: Basic Concepts

# Model Predictive Control (MPC)



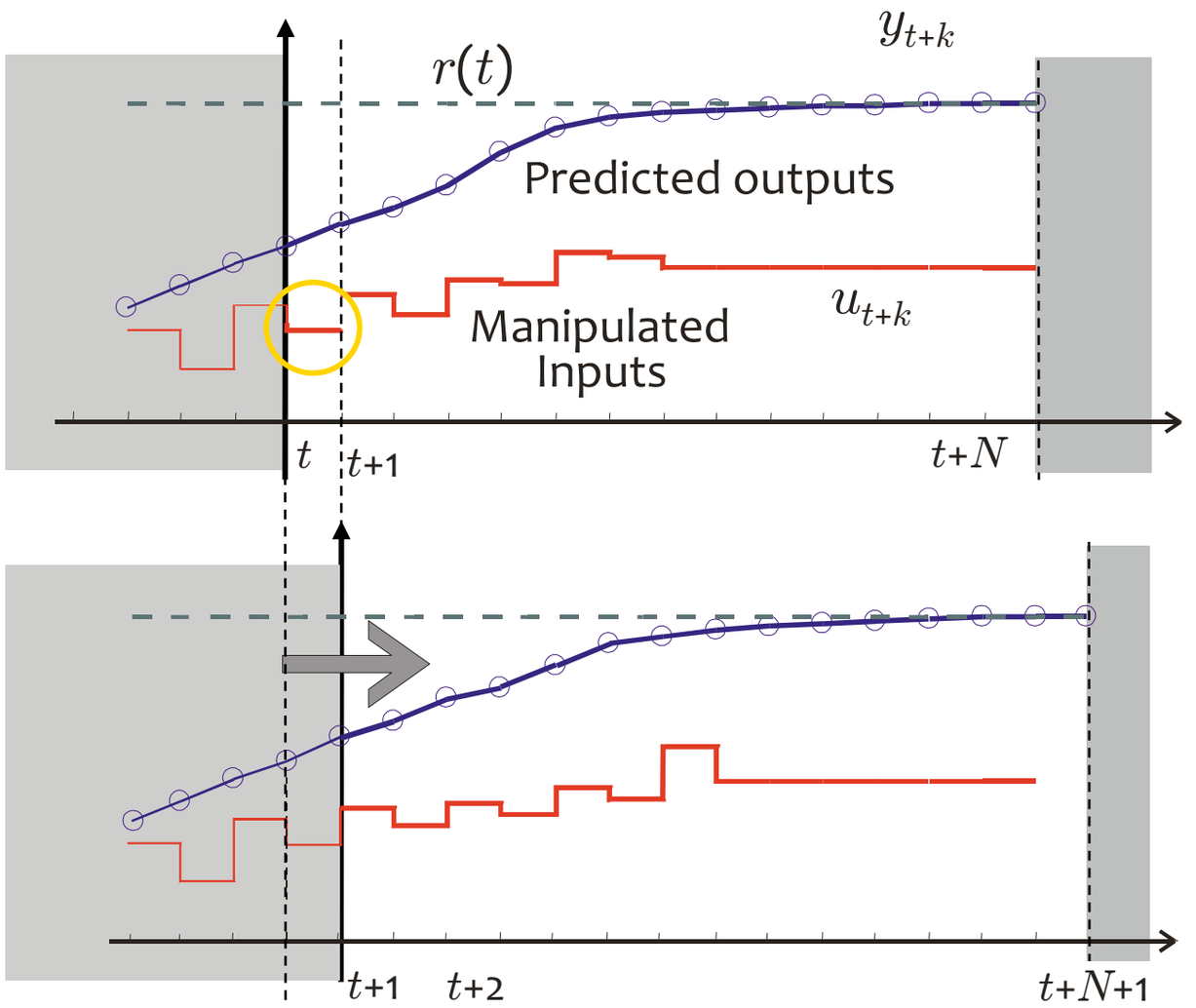

*A **model** of the process is used to **predict** the future evolution of the process to optimize the **control** signal*

# Receding horizon philosophy

- At time $t$: solve an **optimal control** problem over a finite future horizon of $N$ steps:

$$\min_{u_t,\dots,u_{t+N-1}} \left\{ \sum_{k=0}^{N-1} \|y_{t+k} - r(t)\|^2 + \rho\|u_{t+k} - u_r(t)\|^2 \right\}$$

$$\text{s.t.} \quad x_{t+k+1} = f(x_{t+k}, u_{t+k})$$
$$y_{t+k} = g(x_{t+k}, u_{t+k})$$
$$u_{\min} \le u_{t+k} \le u_{\max}$$
$$y_{\min} \le y_{t+k} \le y_{\max}$$
$$x_t = x(t), \ \ k = 0, \dots, N-1$$

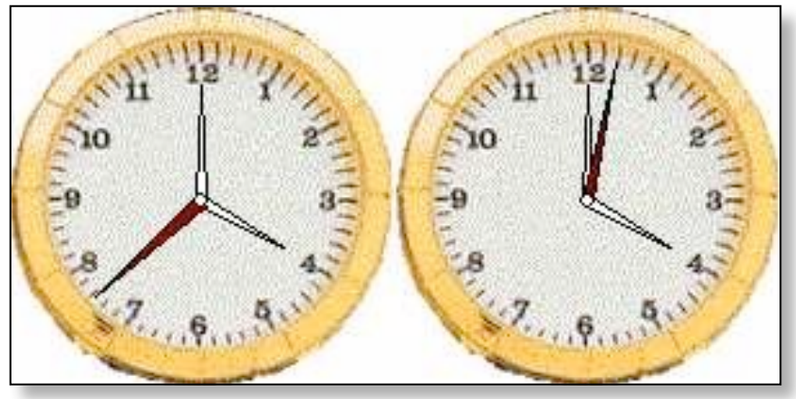



- Only apply the first optimal move $u^*(t)$
- At time $t$+1: Get new measurements, repeat the optimization. And so on …

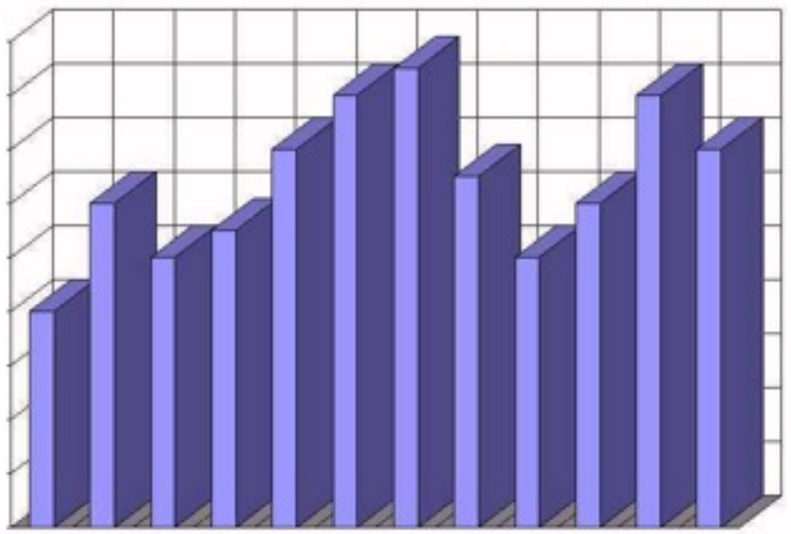
Advantage of repeated on-line optimization: **FEEDBACK!**

# Receding Horizon - Examples

- MPC is like **playing chess** !



- "Rolling horizon" policies are also used frequently **in finance**

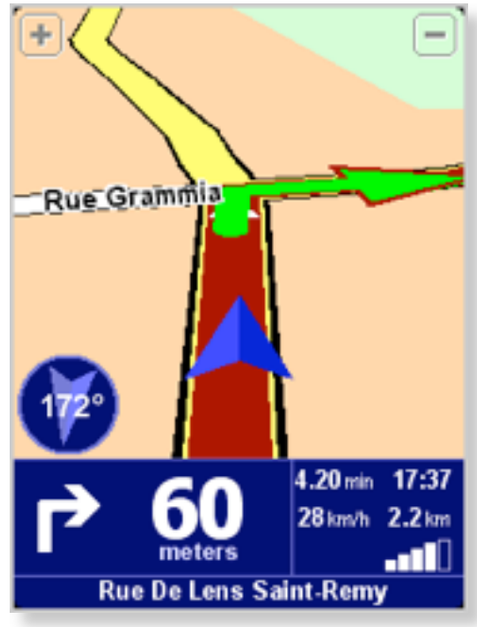


# Receding Horizon - Examples

- **prediction model** how vehicle moves on the map

- **constraints** drive on roads, respect one-way roads, etc.

- **disturbances** mainly driver's inattention !




- **set point** desired location

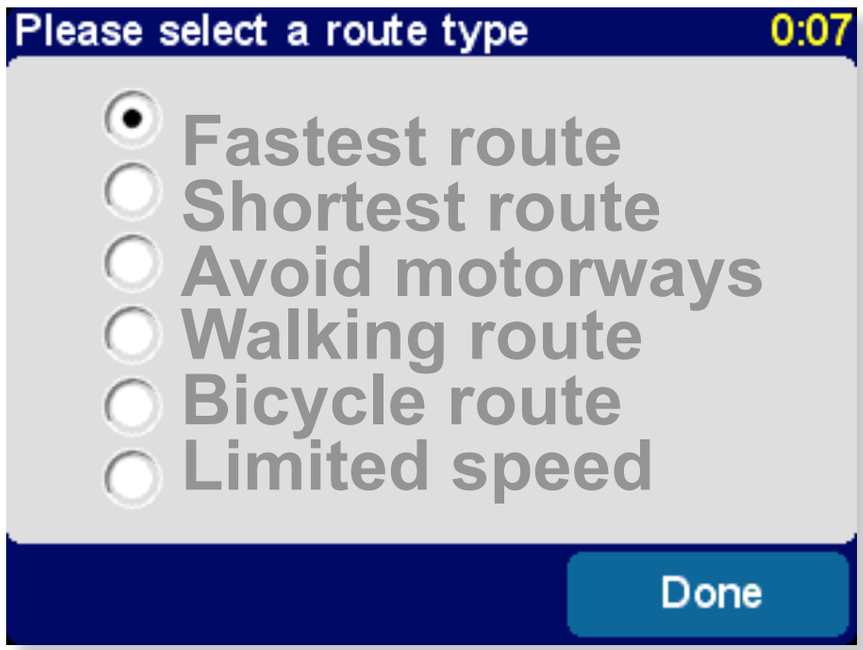
- **cost function** minimum time,
  minimum distance, etc.

- **receding horizon mechanism**

  event-based
  (optimal route re-planned when path is lost)

$x$ = GPS position

$u$ = navigation commands



# Good Models for (MPC) Control

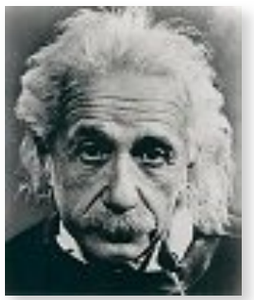
**Note:** computational complexity and theoretical properties (e.g. stability) depend on chosen model/objective/constraints

Good models for MPC:

- **Descriptive** enough to capture the most significant dynamics of the system


- **Simple** enough for solving the optimization problem

***"Make everything as simple as possible, but not simpler."***
— Albert Einstein



# MPC in Industry

- History: Computer control ("Manual" MPC)

Fluid catalytic cracking
(courtesy of Shell / M. Morari)

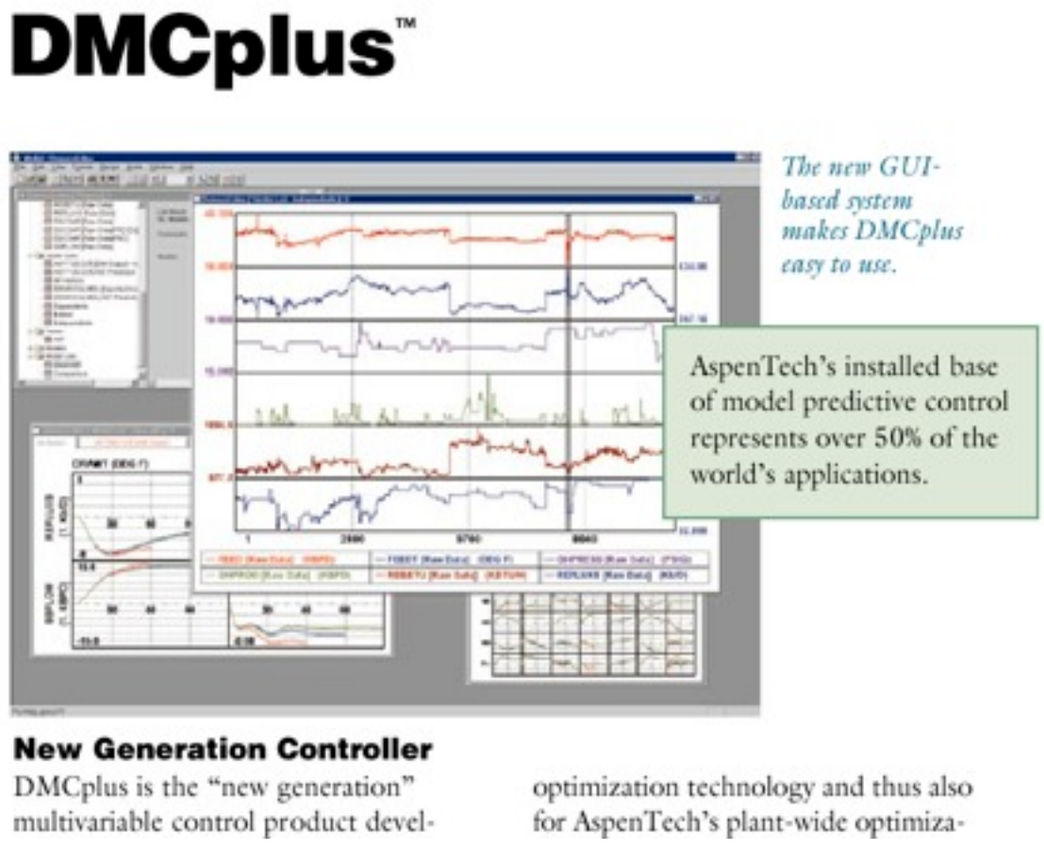
**Fluid catalytic cracking** (FCC) is the most important conversion process used in petroleum refineries. It is widely used to convert the high-boiling hydrocarbon fractions of petroleum crude oils to more valuable gasoline, olefinic gases and other products

(http://en.wikipedia.org/wiki/Catalytic_cracking)

# MPC in Industry

- History: 1979 Dynamic Matrix Control (DMC) by Shell (Motivation: multivariable, constrained)

- Present Industrial Practice
  - linear impulse/step response models
  - sum of squared errors objective function
  - executed in supervisory mode



DMCplus™

The new GUI-based system makes DMCplus easy to use.

AspenTech's installed base of model predictive control represents over 50% of the world's applications.

**New Generation Controller**

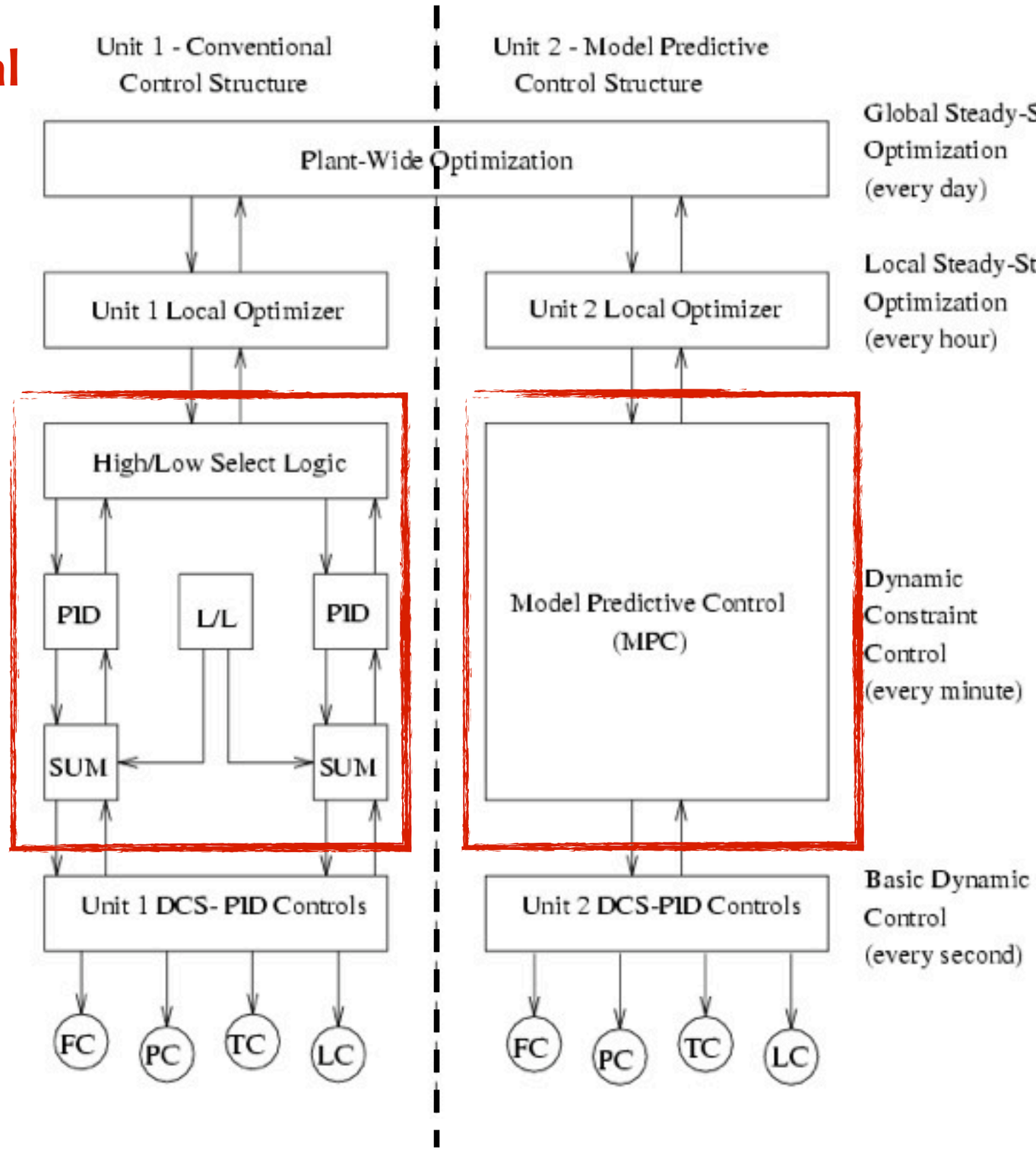
DMCplus is the "new generation" multivariable control product devel- optimization technology and thus also for AspenTech's plant-wide optimiza-

- Particularly suited for problems with
  - many inputs and outputs
  - constraints on inputs, outputs, states
  - varying objectives and limits on variables (e.g. because of faults)

# MPC in Industry

## Hierarchy of control system functions:

**Conventional** **MPC**




(Qin, Badgewell, 1997)

# MPC in Industry

| Area | Aspen Technology | Honeywell Hi-Spec | Adersa[b] | Invensys | SGS[c] | 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<br>IDCOM-M:1987<br>OPC:1987 | PCT:1984<br>RMPCT:1991 | IDCOM:1973<br>HIECON:1986 | 1984 | 1985 | |
| Largest App. | 603 × 283 | 225 × 85 | — | 31 × 12 | — | |

(snapshot survey conducted in mid-1999)

(Qin, Badgewell, 2003)

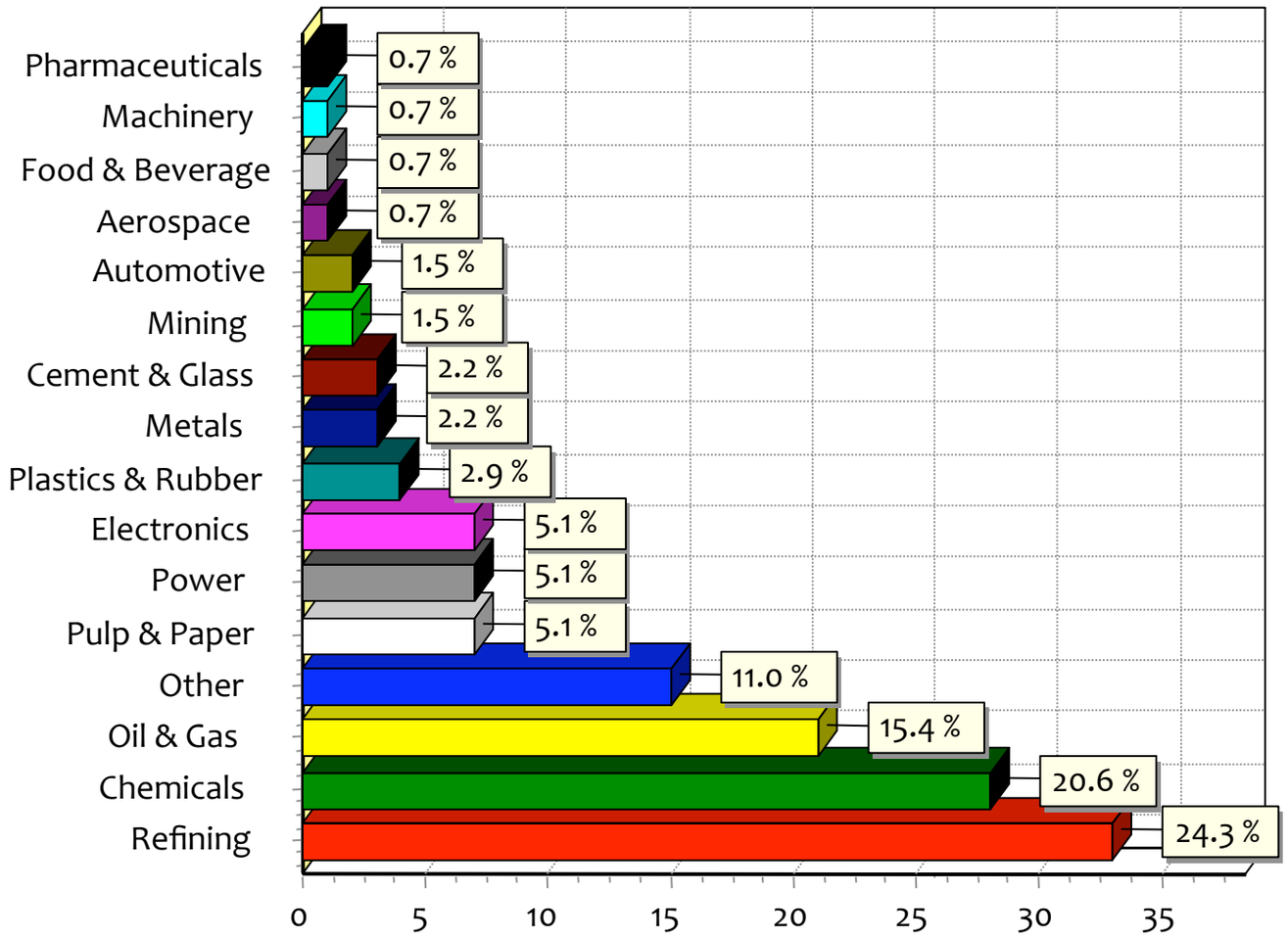
"For us multivariable control is predictive control "

Tariq Samad, *Honeywell* (IEEE Control System Society, President) (1997)

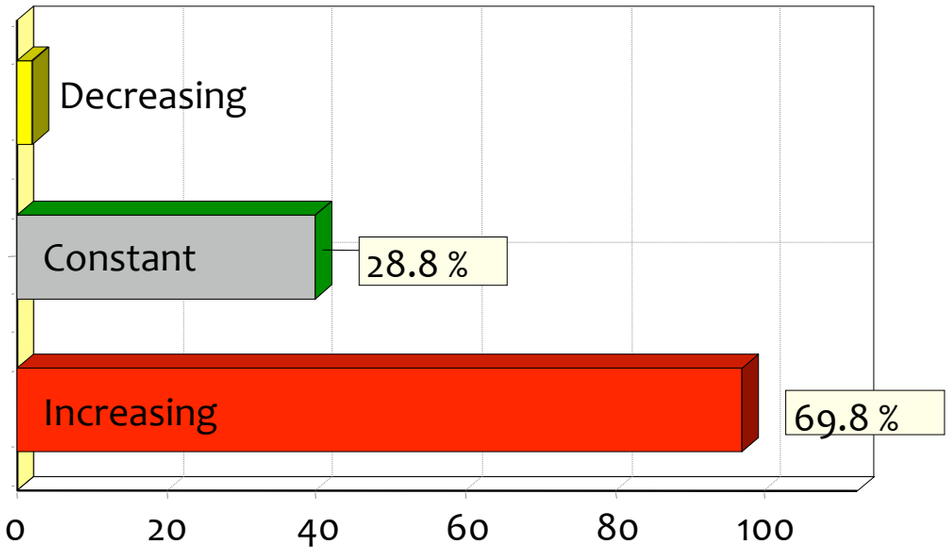
# MPC in Industry

Results from a recent survey (November 7, 2005) about the use of MPC techniques / real-time optimization in a set of US industries:

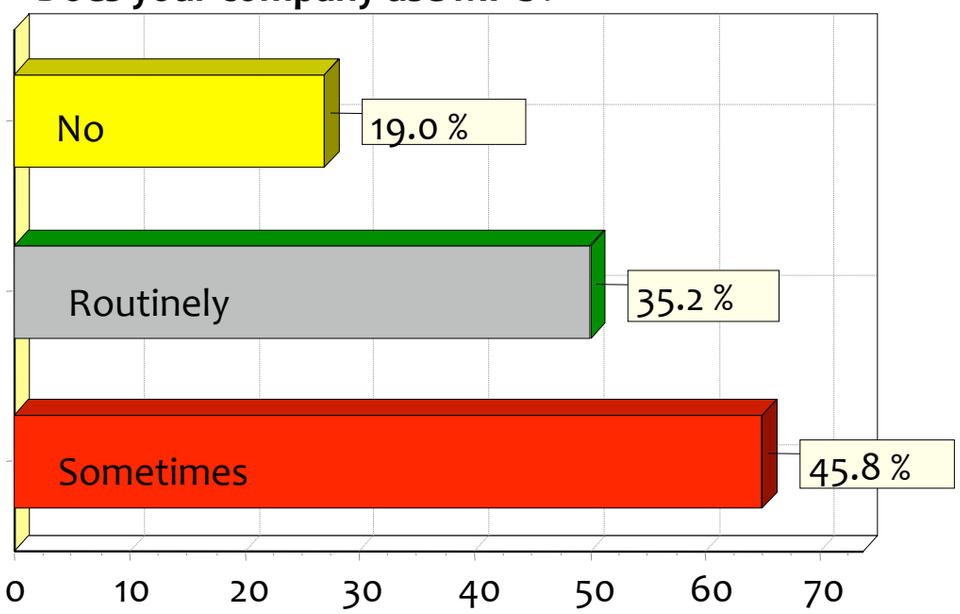
**Industrial area of respondents to the survey:**




**Do you see your use of MPC accelerating, staying constant, or declining?**



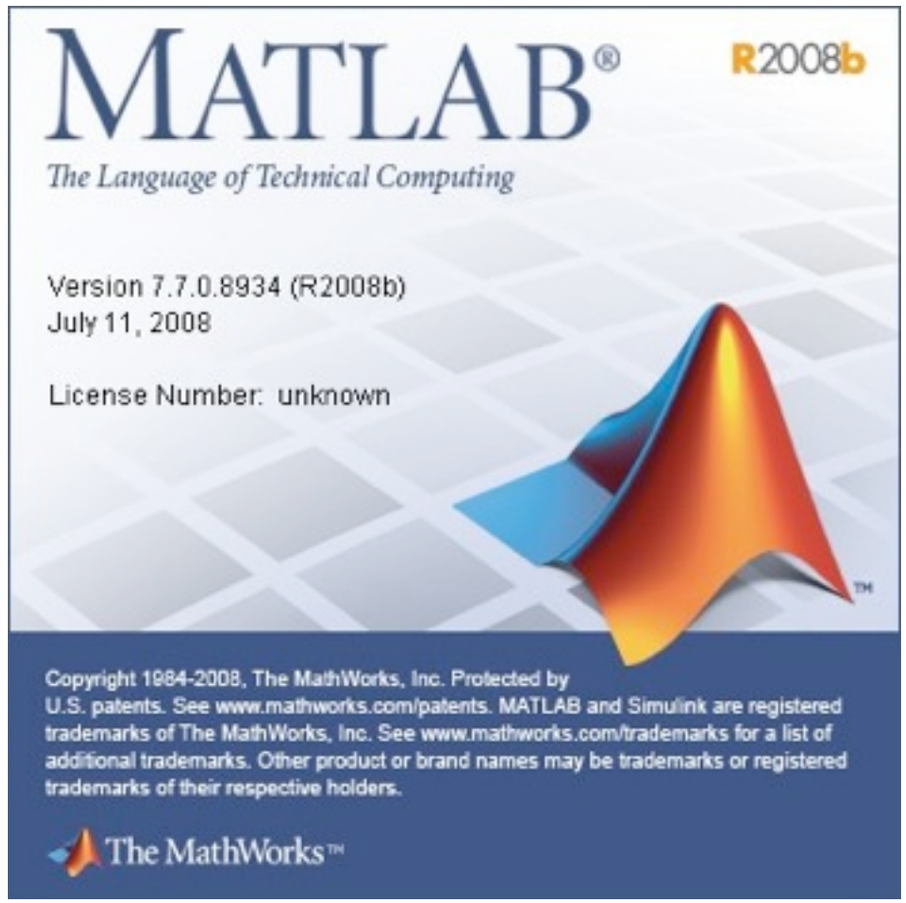

**Does your company use MPC ?**




courtesy: ARC Advisory Group

# Model Predictive Control Toolbox

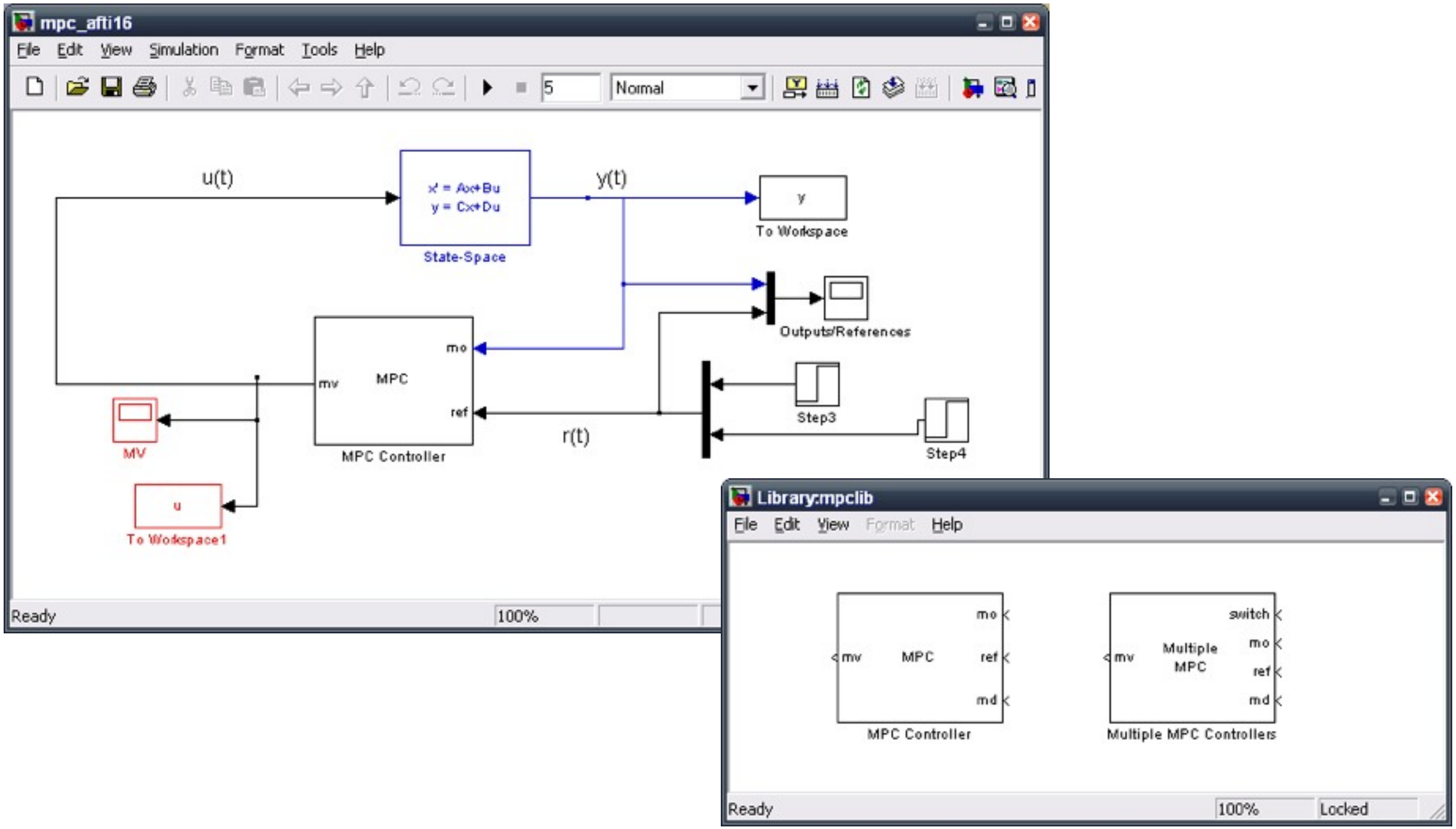
- **MPC Toolbox 3.0** (Bemporad, Ricker, Morari, 1998-today):
  - Object-oriented implementation (MPC object)
  - MPC Simulink Library
  - MPC Graphical User Interface
  - RTW extension (code generation)
    [xPC Target, dSpace, etc.]
  - Linked to OPC Toolbox v2.0.1




Only linear models are handled

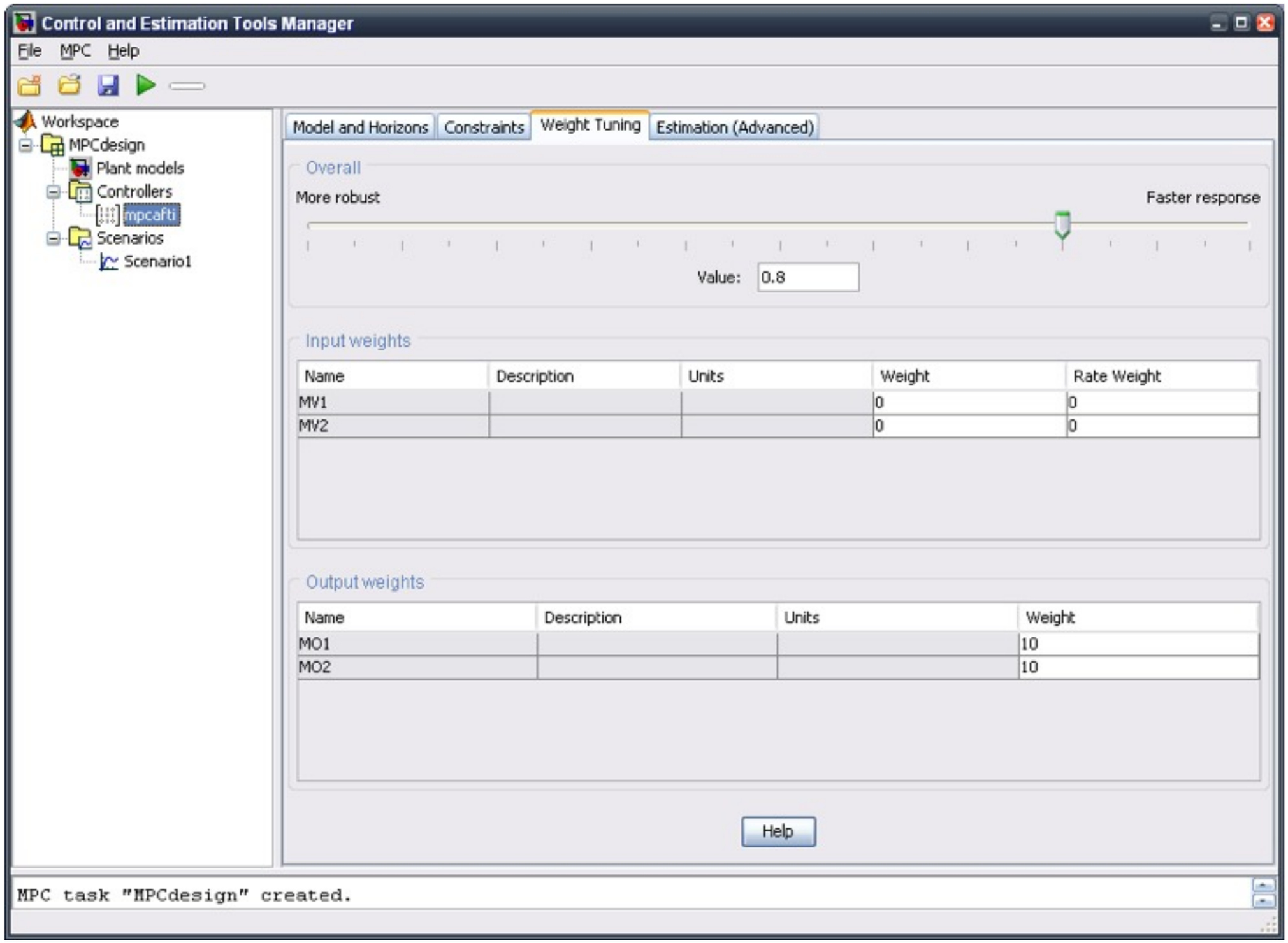
**http://www.mathworks.com/products/mpc/**

# MPC Simulink Library




- Single MPC and multiple swicthed MPC blocks supported
- Reference/disturbance preview and time-varying limits supported

# MPC Graphical User Interface



# MPC Tuning Advisor

MPC Tuning Advisor for Task - MPCdesign

Scenario in design: **Scenario1** Controller in design: **mpcafti** Select a performance function: ISE

Output Weights

| Name | Performance Weight | Sensitivity | Tuning Direction | Current Tuning |
|---|---|---|---|---|
| | 10 | -2.208 | Increase | 5 |
| | 10 | 0.9202 | Decrease | 12 |

Input Weights

| Name | Performance Weight | Sensitivity | Tuning Direction | Current Tuning |
|---|---|---|---|---|
| | 0 | -9.68e-005 | Increase | 0 |
| | 0 | 0.01158 | Decrease | 0 |

Input Rate Weights

| Name | Performance Weight | Sensitivity | Tuning Direction | Current Tuning |
|---|---|---|---|---|
| | 1e-010 | 0.001367 | Decrease | 1e-010 |
| | 1e-010 | -0.0005747 | Increase | 1e-010 |

Baseline Baseline Performance: **166.7** Analyze Current Tuning Performance: **125.4**

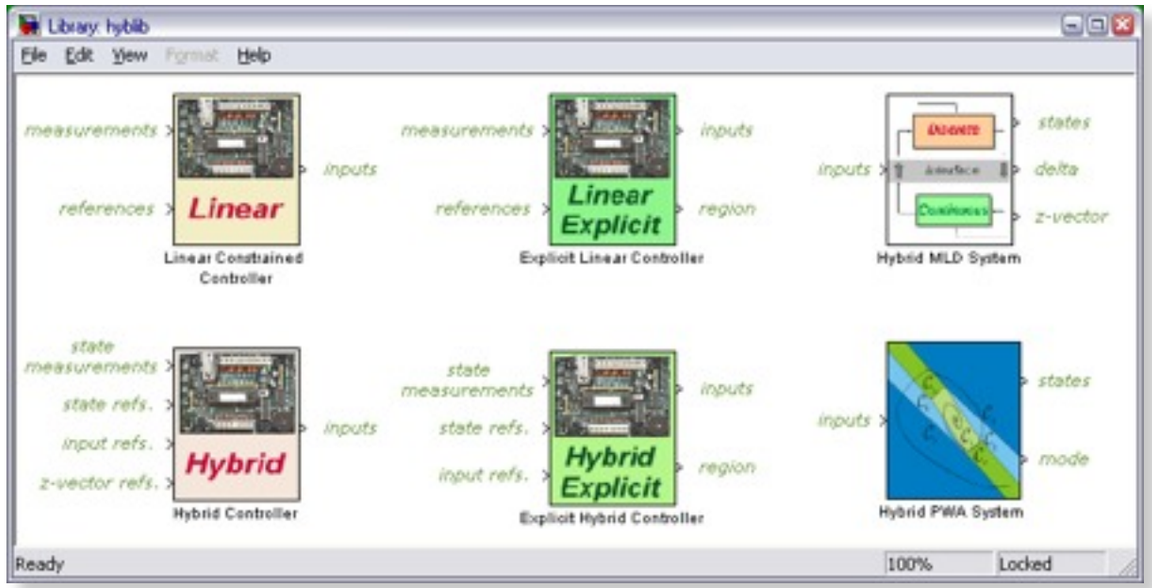
Tuning Advisor is Modal | Restore Baseline Weights | Update Controller in MPC Tool | Close | Help

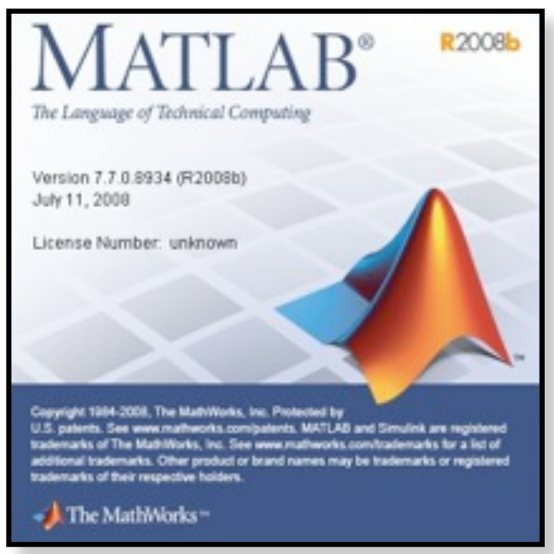
# Hybrid Toolbox for MATLAB

**Features:**

(Bemporad, 2003-2009)

- Hybrid models: design, simulation, verification
- Control design for linear systems w/ constraints
  and hybrid systems (on-line optimization via QP/MILP/MIQP)
- Explicit MPC control (via multi-parametric programming)
- C-code generation
- Simulink library




**2450+ download requests since October 2004**

**http://www.dii.unisi.it/hybrid/toolbox**

# Basics of Constrained Optimization

# Mathematical Programming

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ \text{s.t.} & g(x) \leq 0 \end{array}$$

$$\left(\begin{array}{ll} \max\limits_{x} & f(x) \\ \text{s.t.} & g(x) \leq 0 \end{array}\right)$$

$x \in \mathbb{R}^n,\ f : \mathbb{R}^n \to \mathbb{R},\ g : \mathbb{R}^n \to \mathbb{R}^m$

$$f(x) = f(x_1, \ldots, x_n) \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad g(x) = \begin{bmatrix} g_1(x_1, \ldots, x_n) \\ g_2(x_1, \ldots, x_n) \\ \vdots \\ g_m(x_1, \ldots, x_n) \end{bmatrix}$$

In general, problem is difficult to solve

⟹ **use software tools**

# Optimization Software

- Taxonomy of most known solvers, for different classes of optimization problems:

  **http://www-fp.mcs.anl.gov/otc/Guide/SoftwareGuide/**

- Network Enabled Optimization Server (NEOS) for remotely solving optimization problems:

  **http://neos.mcs.anl.gov/neos/solvers/**

- Comparison on benchmark problems:

  **http://plato.la.asu.edu/bench.html**

- Good open-source optimization software

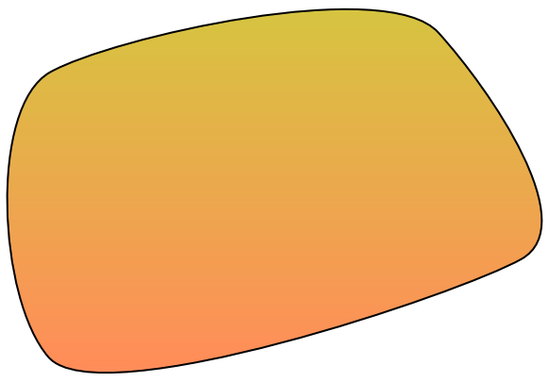
  **http://www.coin-or.org/**

# Convex sets

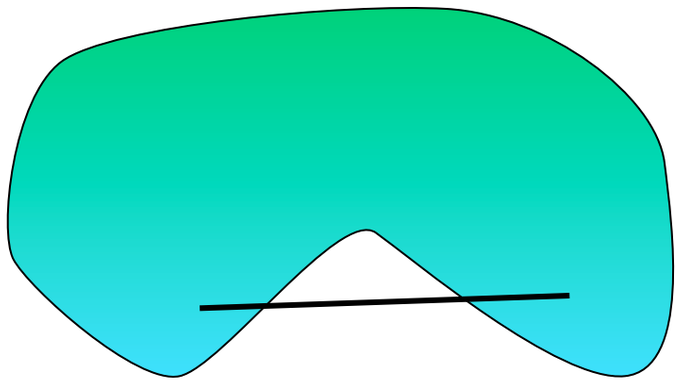
A **set** $S \in X$ is convex if for all $x_1, x_2 \in S$

$$\lambda x_1 + (1-\lambda)x_2 \in S, \text{ for all } \lambda \in [0,1]$$

Convex set



Nonconvex set

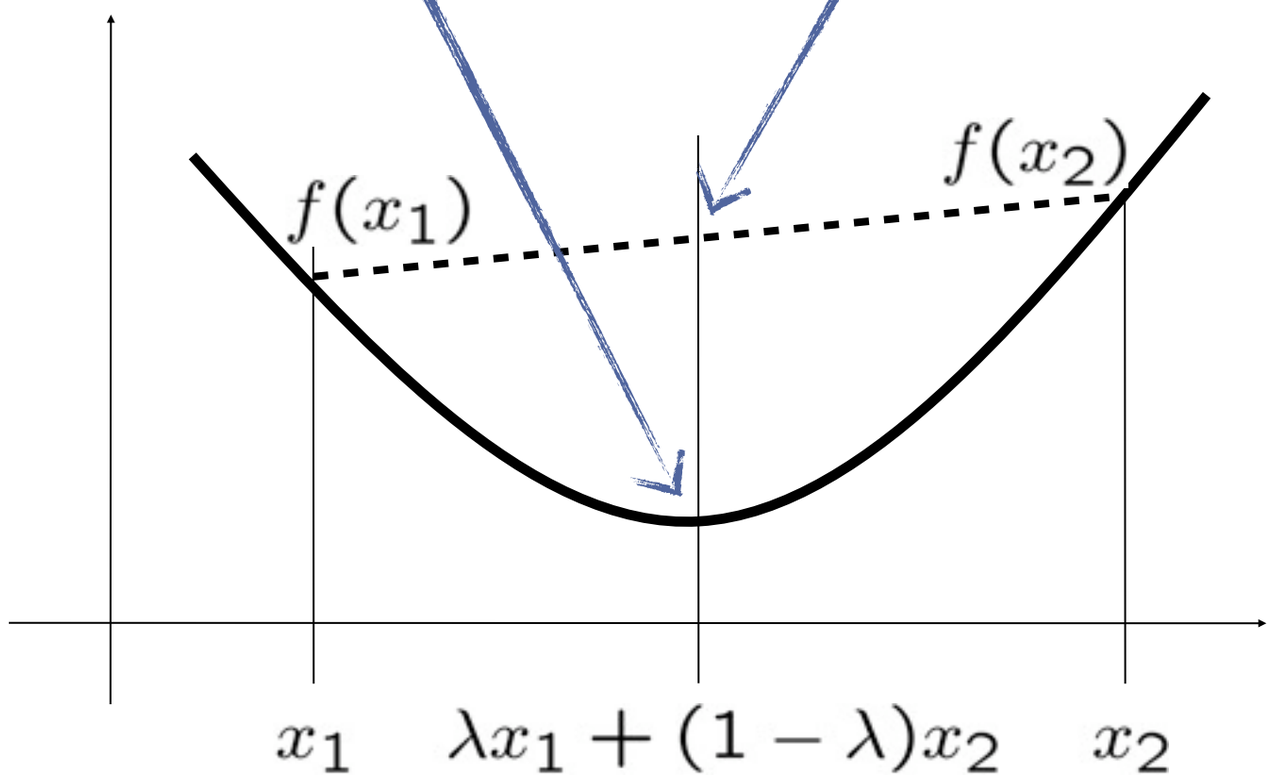


# Convex function

A **function** $f : S \to \mathbb{R}$ is convex if $S$ is convex and

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$

for all $x_1, x_2 \in S, \lambda \in [0,1]$



# Convex Optimization Problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C \end{array}$$

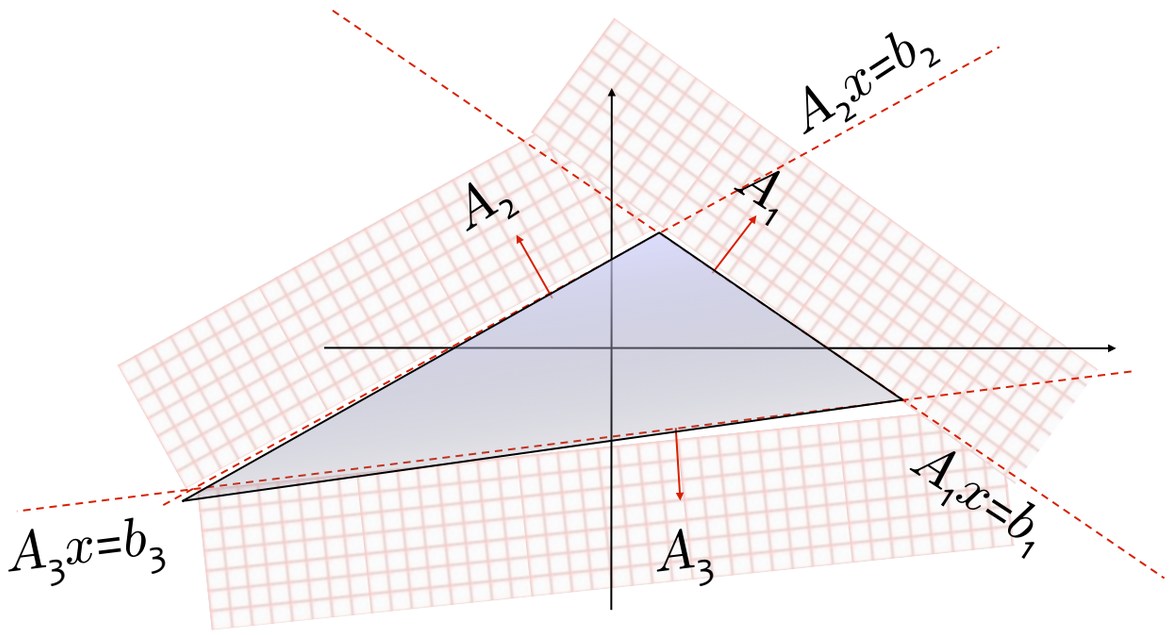
$f$ and $C$ convex

- Very efficient numerical algorithms exist
- Global solution attained
- Extensive useful theory
- Often occurring in engineering problems
- Tractable in theory and practice

Excellent reference textbook: "Convex Optimization" by S. Boyd and L. Vandenberghe **http://www.stanford.edu/~boyd/cvxbook/**

# Polyhedra

- A convex polyhedron is the intersection of a finite set of halfspaces of $\mathbb{R}^d$
- A convex polytope is a bounded convex polyhedron



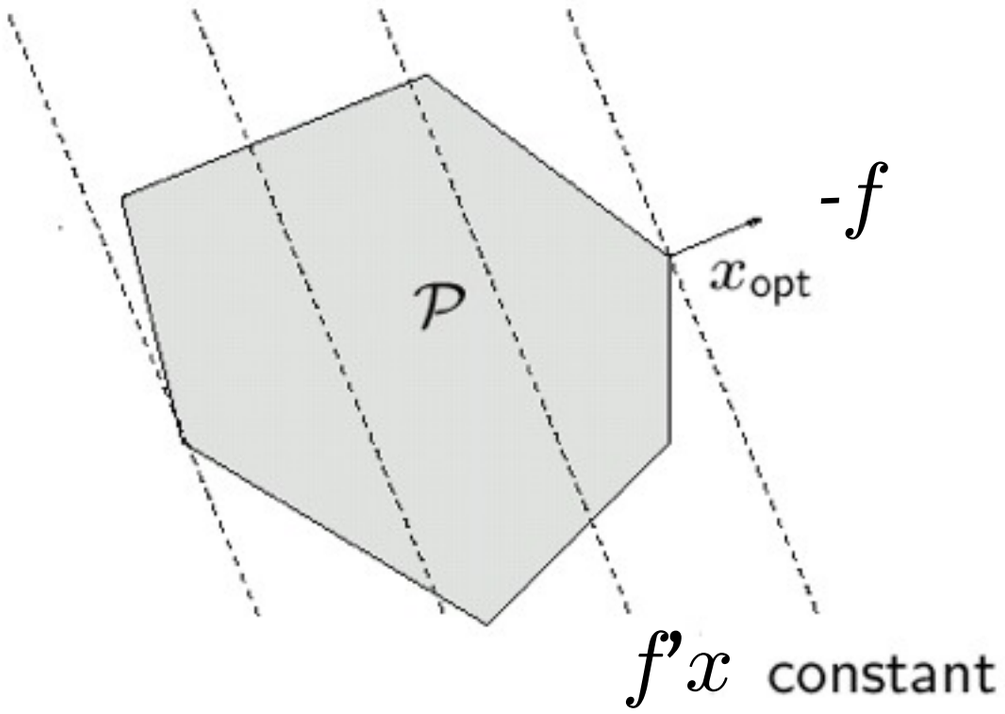

- Hyperplane representation:

$$Ax \leq b$$

# Linear Program

George Dantzig
(1914 - 2005)

$$\begin{aligned} \min \quad & f'x \\ \text{s.t.} \quad & Ax \le b, \;\; x \in \mathbb{R}^n \end{aligned}$$




Standard form:

$$\begin{aligned} \min \quad & f'x \\ \text{s.t.} \quad & Ax = b \\ & x \ge 0, \;\; x \in \mathbb{R}^n \end{aligned}$$

## Slack variables

$$\sum_{j=1}^{N} a_j x_j \le b \quad \Rightarrow \quad \sum_{j=1}^{N} a_j x_j + s = b, \;\; s \ge 0$$

# Linear Program

trasformation
from max to min:

$$\max_x c^T x = -\left(\min_x -c^T x\right)$$

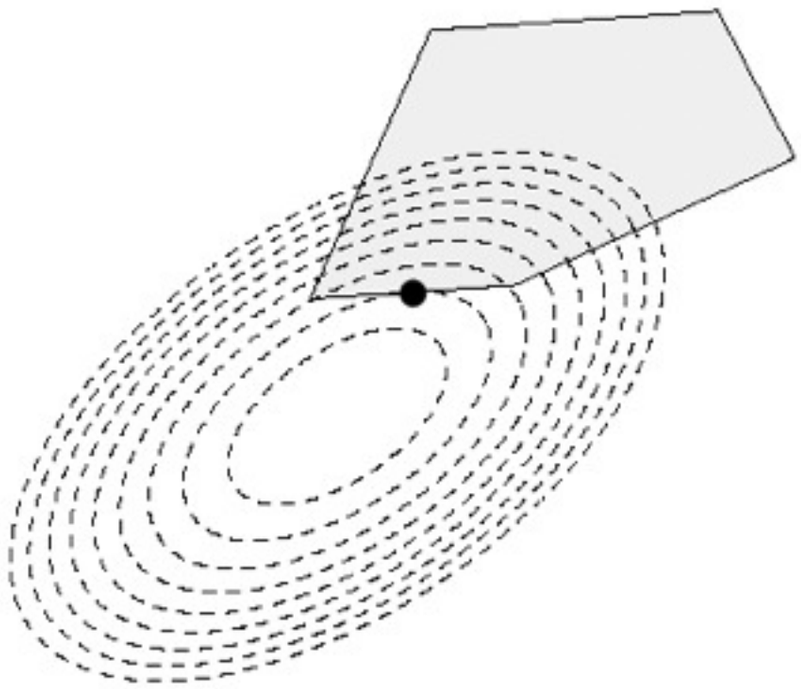
Change inequality direction:

$$\sum_{j=1}^{N} a_{ij}x_j \geq b_i \quad \Longrightarrow \quad \sum_{j=1}^{N} -a_{ij}x_j \leq -b_i$$

It is always possible to formulate LP problems using "**min**" and "$\leq$" inequalities

# Quadratic Program

$$\begin{aligned} \min \quad & \frac{1}{2}x'Px + f'x \\ \text{s.t.} \quad & Ax \leq b, \;\; x \in \mathbb{R}^n \end{aligned}$$



- Convex optimization if $P \succeq 0$
- Hard problem if $P \not\succeq 0$

# Mixed-Integer Linear Program

$$\begin{aligned} \min \quad & f'x + d'\delta \\ \text{s.t.} \quad & Ax + B\delta \le c \\ & x \in \mathbb{R}^n, \ \delta \in \{0,1\}^m \end{aligned}$$

- Some variables are continuous, some are discrete (0/1)
- In general, it is a NP-hard problem
- Rich variety of algorithms/solvers available

# Modeling languages

- **MOSEL**, associated with commercial package Xpress-MP
- **OPL** (Optimization Programming Language), associated with commercial package Ilog-CPLEX
- **AMPL** (A Modeling Language for Mathematical Programming) most used modeling language, supports several solvers
- **GAMS** (General Algebraic Modeling System) is one of the first modeling languages
- **LINGO**, modeling language of Lindo Systems Inc.
- **GNU MathProg**, a subset of AMPL associated with the *free* package GLPK (GNU Linear Programming Kit)
- **FLOPC++** *open source* modeling language (C++ class library)
- **CVX** matlab-based modeling language (from Stanford)
- **YALMIP** another matlab-based modeling language

# Linear MPC

# Unconstrained Optimal Control

- Linear model:

$$\begin{cases} x(t+1) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases}$$

$x \in \mathbb{R}^n,\ u \in \mathbb{R}^m$
$y \in \mathbb{R}^p$

- Goal: find $u^*(0),\ u^*(1),\ \ldots,\ u^*(N-1)$

$$J(x(0), U) = \sum_{k=0}^{N-1} \left[ x'(k)Qx(k) + u'(k)Ru(k) \right] + x'(N)Px(N)$$

$$U = [u'(0)\ u'(1)\ \ldots\ u'(N-1)]'$$

$u^*(0),\ u^*(1),\ \ldots,\ u^*(N-1)$ is the input sequence that steers the state to the origin "optimally"

# [computation of cost function]

$$J(x(0),U) = x'(0)Qx(0) + \begin{bmatrix} x'(1) & x'(2) & \dots & x'(N-1) & x'(N) \end{bmatrix} \overbrace{\begin{bmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & Q & 0 \\ 0 & 0 & \dots & 0 & P \end{bmatrix}}^{\bar{Q}} \cdot$$

$$\cdot \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(N-1) \\ x(N) \end{bmatrix} + \begin{bmatrix} u'(0) & u'(1) & \dots & u'(N-1) \end{bmatrix} \underbrace{\begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & R \end{bmatrix}}_{\bar{R}} \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N-1) \end{bmatrix}$$

$$\begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(N) \end{bmatrix} = \overbrace{\begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}}^{\bar{S}} \begin{bmatrix} u(0) \\ u(1) \\ \dots \\ u(N-1) \end{bmatrix} + \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{\bar{T}} x(0)$$

$$\begin{aligned} J(x(0),U) &= x'(0)Qx(0) + (\bar{S}U + \bar{T}x(0))'\bar{Q}(\bar{S}U + \bar{T}x(0)) + U'\bar{R}U \\ &= \frac{1}{2}U' \underbrace{2(\bar{R} + \bar{S}'\bar{Q}\bar{S})}_{H} U + x'(0) \underbrace{2\bar{T}'\bar{Q}\bar{S}}_{F} U + \frac{1}{2}x'(0) \underbrace{2(Q + \bar{T}'\bar{Q}\bar{T})}_{Y} x(0) \end{aligned}$$

# Unconstrained Optimal Control

$$J(x(0),U) = \frac{1}{2}U'HU + x'(0)FU + \frac{1}{2}x'(0)Yx(0)$$

$$U = [u'(0)\ u'(1)\ \dots\ u'(N-1)]'$$

The optimum is obtained by zeroing the gradient

$$\nabla_U J(x(0),U) = HU + F'x(0) = 0$$

and hence

$$U^* = \begin{bmatrix} u^*(0) \\ u^*(1) \\ \vdots \\ u^*(N-1) \end{bmatrix} = -H^{-1}F'x(0)$$

**batch least squares**

Alternative approach: use dynamic programming to find $U^*$ (Riccati iterations)

# Constrained Optimal Control

- Linear model:

$$\begin{cases} x(t+1) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases} \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^m,\ y \in \mathbb{R}^p$$

- Constraints:

$$\begin{cases} u_{\min} \le u(t) \le u_{\max} \\ y_{\min} \le y(t) \le y_{\max} \end{cases}$$

- Constrained optimal control problem (quadratic performance index):

$$\min_{u(0),\ldots,u(N-1)} \sum_{k=0}^{N-1} \left[x'(k)Qx(k) + u'(k)Ru(k)\right] + x'(N)Px(N)$$

$$\text{s.t.} \quad \begin{array}{l} u_{\min} \le u(k) \le u_{\max},\ \ k = 0,\ldots,N-1 \\ y_{\min} \le y(k) \le y_{\max},\ \ k = 1,\ldots,N \end{array}$$

$$Q = Q' \succeq 0,\ R = R' \succ 0,\ P \succeq 0$$

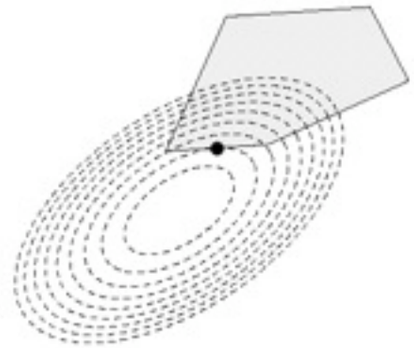
# Constrained Optimal Control

- Optimization problem:

$$V(x(0)) = \tfrac{1}{2}x'(0)Yx(0) + \min_{U} \ \tfrac{1}{2}U'HU + x'(0)FU \quad \text{(quadratic)}$$

$$\text{s.t.} \quad GU \leq W + Sx(0) \quad \text{(linear)}$$

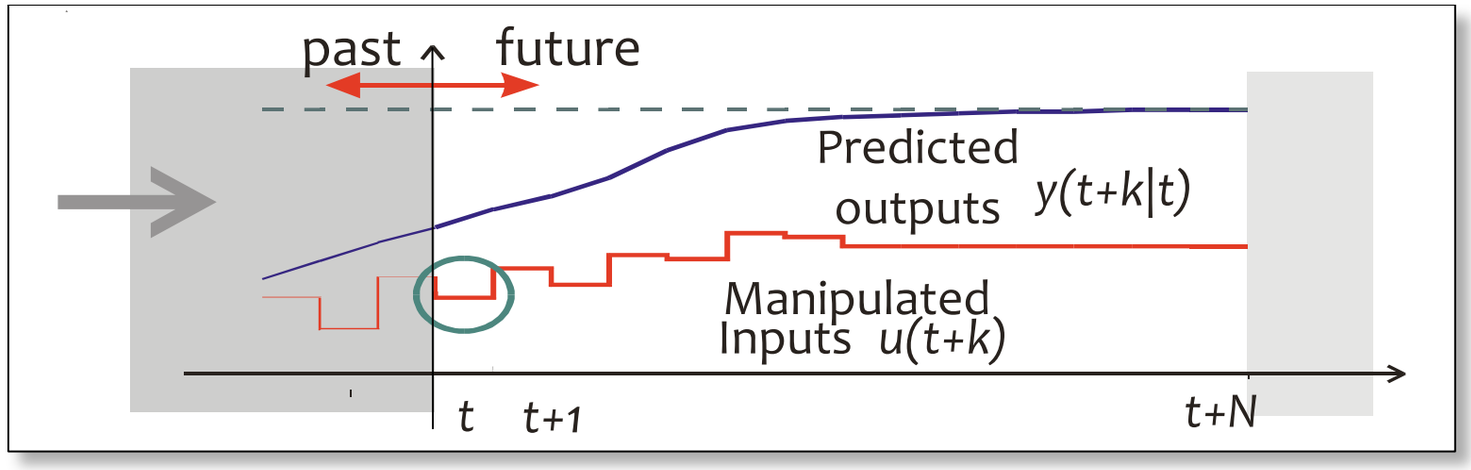
**Convex QUADRATIC PROGRAM (QP)**



- $U \triangleq [u'(0) \ \ldots \ u'(N-1)]' \in \mathbb{R}^s$, $s \triangleq Nm$, is the optimization vector

- $H = H' \succ 0$, and $H$, $F$, $Y$, $G$, $W$, $S$ depend on weights $Q$, $R$, $P$, upper and lower bounds $u_{\min}$, $u_{\max}$, $y_{\min}$, $y_{\max}$, and model matrices $A$, $B$, $C$

# Linear MPC Algorithm




At time $t$:

- Get/estimate the current state $x(t)$
- Solve the QP problem

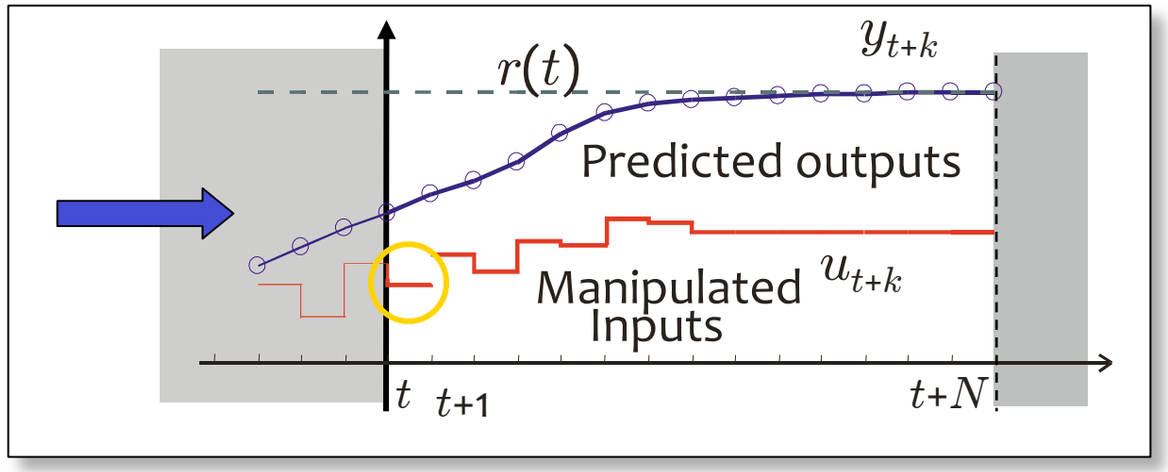
$$\min_U \ \tfrac{1}{2}U'HU + x'(t)FU$$
$$\text{s.t.} \quad GU \le W + Sx(t)$$

and let $U=\{u^*(0),\ldots,u^*(N\text{-}1)\}$ be the solution
(=finite-horizon constrained open-loop optimal control)

- Apply only $u(t) = u^*(0)$ and discard the remaining optimal inputs
- Repeat optimization at time $t$+1. And so on ...

# Unconstrained Linear MPC

- Assume no constraints




- Problem to solve on-line:

$$\min_U \quad J(x(t),U) = \frac{1}{2}U'HU + x'(t)FU$$

- Solution: $\nabla_U J(x(t),U) = HU + F'x(t) = 0$

$$\longrightarrow \quad U^* = -H^{-1}F'x(t)$$

$$U = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$\Longrightarrow \quad \boxed{u(t) = -[I \ 0 \ \dots \ 0]H^{-1}Fx(t) \triangleq Kx(t)}$$

Unconstrained linear MPC is nothing else than a standard linear state-feedback law !

# Double Integrator Example

- System: $y(\tau) = \frac{1}{s^2} u(\tau)$

sampling + ZOH
$T_s$=1 s

$$\begin{aligned} x(t+1) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \end{aligned}$$

- Constraints: $-1 \le u(\tau) \le 1$

- Control objective: min $\left( \sum_{k=0}^{1} y^2(k) + \frac{1}{10} u^2(k) \right) + x'(2) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x(2)$

- Optimization problem matrices:

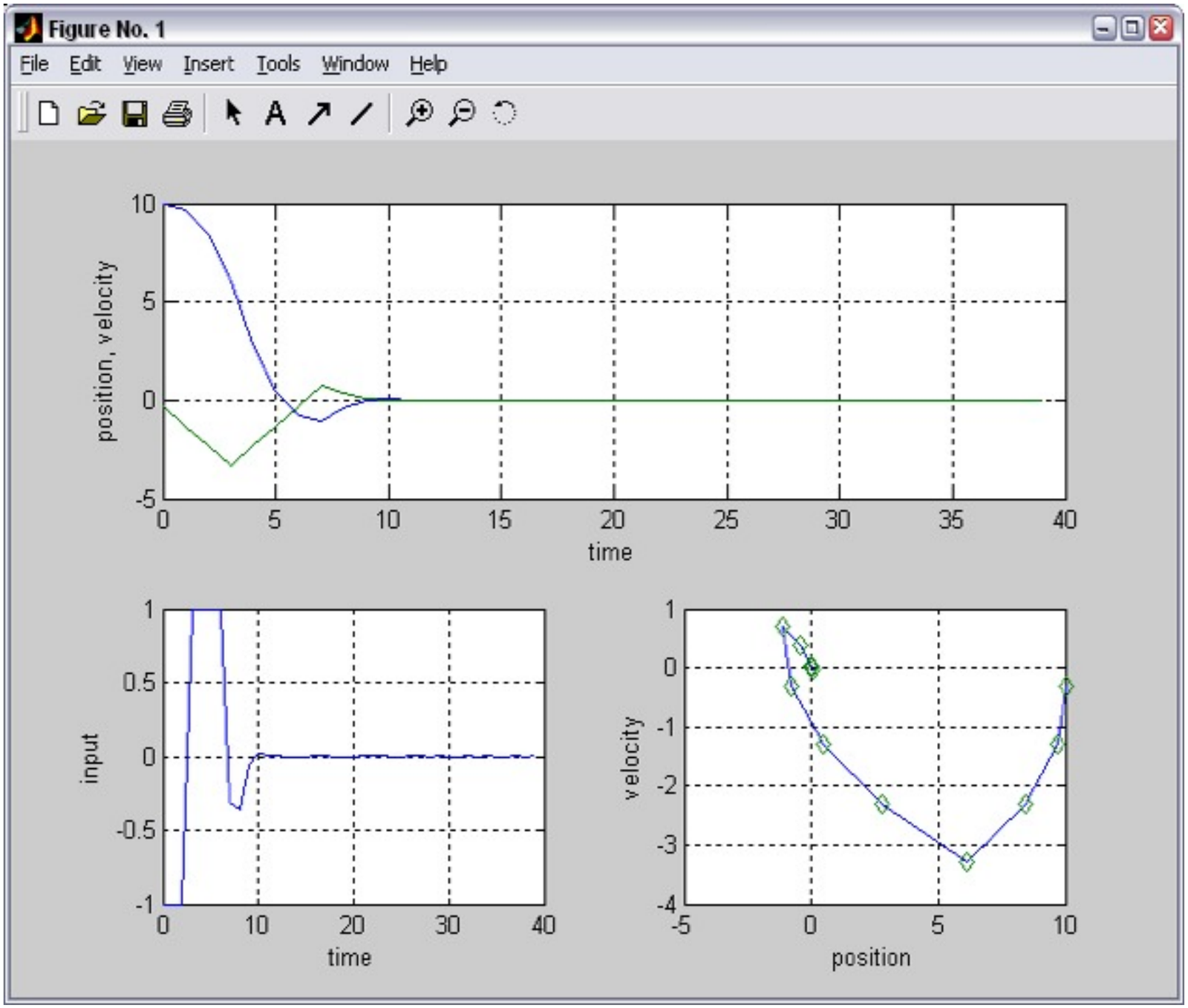
$$H = \begin{bmatrix} 4.2 & 2 \\ 2 & 2.2 \end{bmatrix}, \; F = \begin{bmatrix} 2 & 0 \\ 6 & 2 \end{bmatrix}, \; Y = \begin{bmatrix} 4 & 6 \\ 6 & 12 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, \; W = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \; S = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

cost: $\frac{1}{2} U'HU + x'(t)FU + \frac{1}{2} x'(t) Y x(t)$

constraints: $GU \le W + Sx(t)$

# Double Integrator Example




go to demo **/demos/linear/doubleint.m** (Hyb-Tbx)

see also **mpcdoubleint.m** (MPC-Tbx)

# Double Integrator Example

- Add a state constraint:

$$x_2(k) \geq -1, \;\; k = 1$$

- Optimization problem matrices:

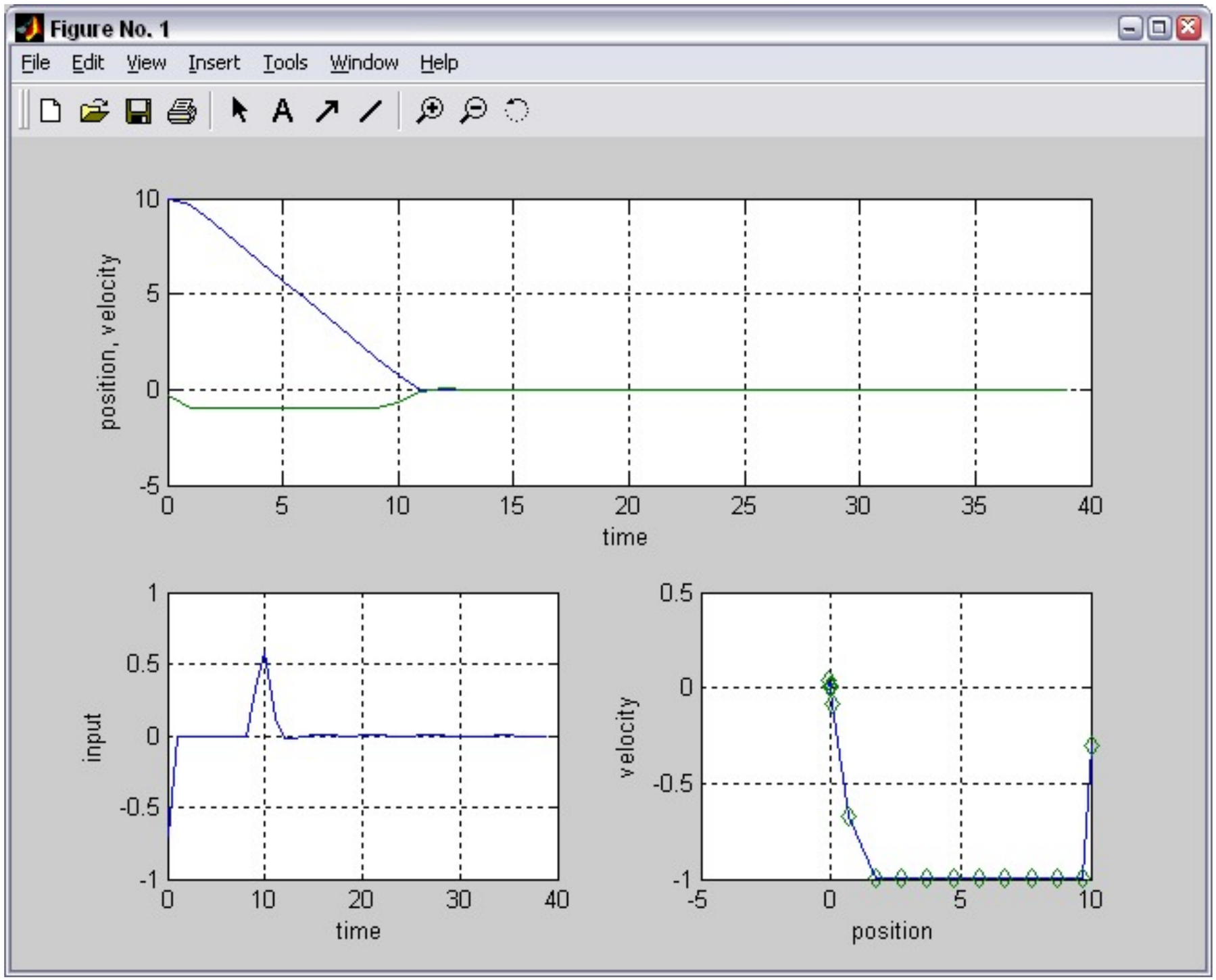
$$H = \begin{bmatrix} 4.2 & 2 \\ 2 & 2.2 \end{bmatrix}, \; F = \begin{bmatrix} 2 & 0 \\ 6 & 2 \end{bmatrix}, \; Y = \begin{bmatrix} 4 & 6 \\ 6 & 12 \end{bmatrix}$$

$$G = \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, \; W = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \; S = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

cost: $\frac{1}{2}U'HU + x'(t)FU + \frac{1}{2}x'(t)Yx(t)$

constraints: $GU \leq W + Sx(t)$

# Double Integrator Example



# Linear MPC - Tracking

- Objective: make the output $y(t)$ track a reference signal $r(t)$
- Idea: parameterize the problem using input increments

$$\boxed{\Delta u(t) = u(t) - u(t-1)} \quad \Rightarrow \quad u(t) = u(t-1) + \Delta u(t)$$

- Extended system: let $x_u(t)$=$u(t$-1$)$

$$\begin{cases} x(t+1) &= Ax(t) + Bu(t-1) + B\Delta u(t) \\ x_u(t+1) &= x_u(t) + \Delta u(t) \end{cases}$$

$$\Downarrow$$

$$\begin{cases} \begin{bmatrix} x(t+1) \\ x_u(t+1) \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} x(t) \\ x_u(t) \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \Delta u(t) \\ y(t) = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x_u(t) \end{bmatrix} \end{cases}$$

Again a linear system with states $x(t)$, $x_u(t)$ and input $\Delta u(t)$

# Linear MPC - Tracking

- Optimal control problem (quadratic performance index):

$$\min_{\Delta U} \quad \sum_{k=0}^{N-1} \|W^y(y(k+1)-r(t))\|^2 + \|W^{\Delta u}\Delta u(k)\|^2$$

$$[\Delta u(k) \triangleq u(k)-u(k-1)]$$

$$\text{subj. to} \quad u_{\min} \le u(k) \le u_{\max}, \ \ k=0,\dots,N-1$$

$$\Delta u_{\min} \le \Delta u(k) \le \Delta u_{\max}, \ \ k=0,\dots,N-1$$

$$y_{\min} \le y(k) \le y_{\max}, \ \ k=1,\dots,N$$

optimization vector:

$$\Delta U = \begin{bmatrix} \Delta u(0) \\ \Delta u(1) \\ \vdots \\ \Delta u(N-1) \end{bmatrix}$$

- Note: $\|Wz\|^2 = (Wz)'(Wz) = z'(W'W)z = z'Qz$

⟹ same formulation as before ($W$=Cholesky factor of weight matrix $Q$)

- Optimization problem:

Convex Quadratic Program (QP)

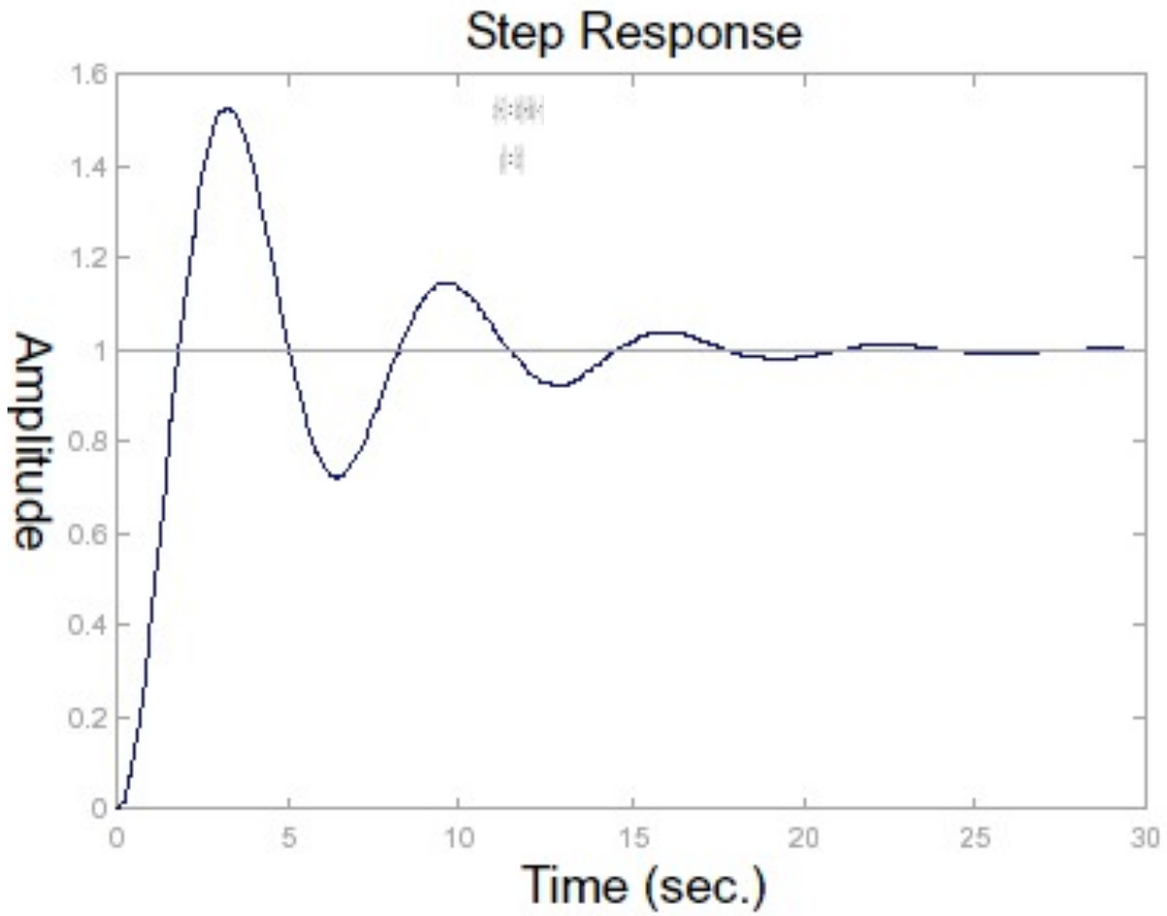
$$\min_{\Delta U} \quad J(\Delta U, x(t)) = \tfrac{1}{2}\Delta U' H \Delta U + [x'(t)\ \ r'(t)\ \ u'(t-1)]F\Delta U$$

$$\text{s.t.} \quad G\Delta U \le W + S\begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \end{bmatrix}$$

# Linear MPC - Example

- Plant: $G(s) = \dfrac{1}{s^2 + 0.4s + 1}$

- Sampling time: $T_s = 0.5$ sec.




- Model:

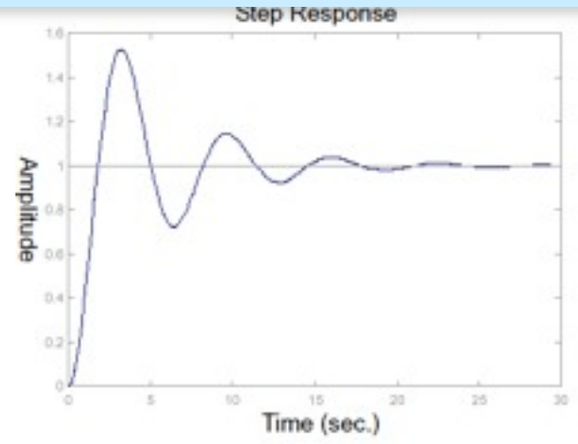
$$\begin{cases} x(t+1) = \begin{bmatrix} 1.597 & -0.4094 \\ 2 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} u(t) \\ y(t) = \begin{bmatrix} 0.2294 & 0.1072 \end{bmatrix} x(t) \end{cases}$$

go to demo **`linear/example3.m`** (Hyb-Tbx)

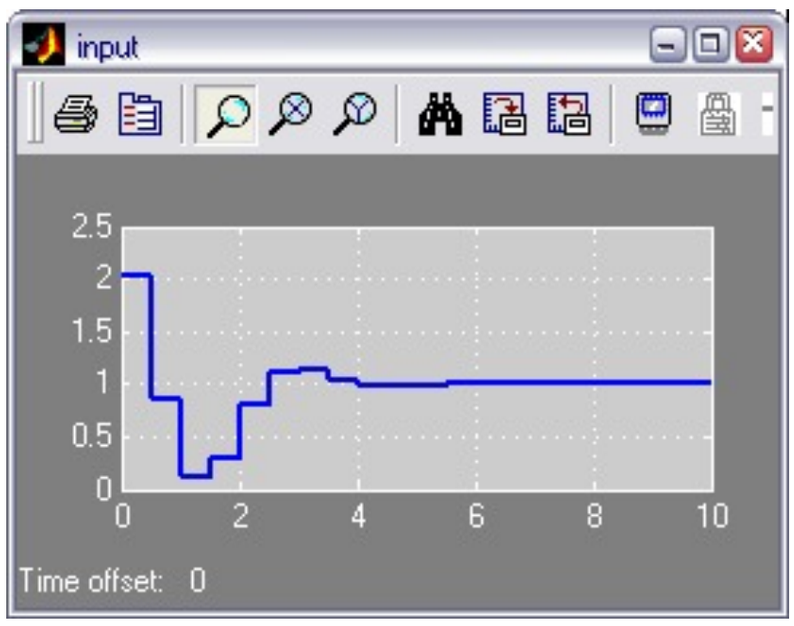
# Linear MPC - Example

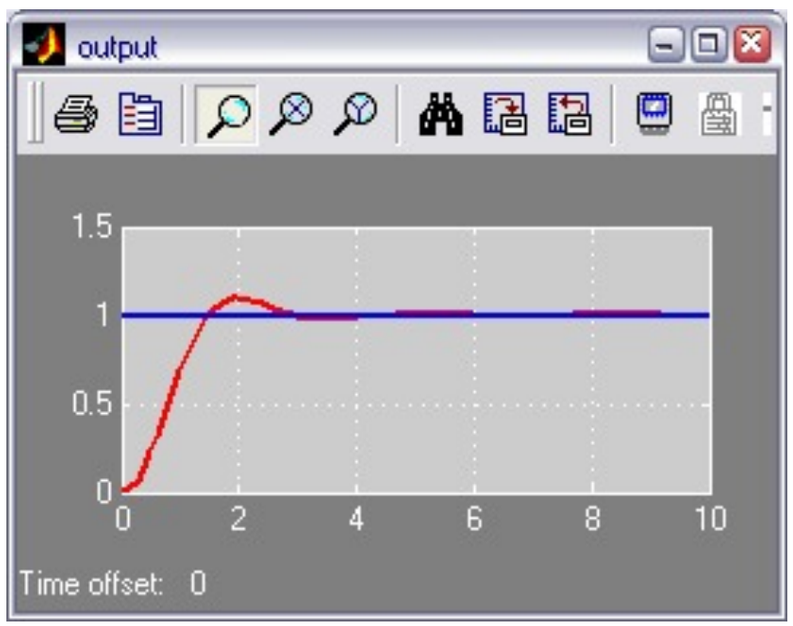
- Performance index:

$$J(U,t) \triangleq \sum_{k=0}^{9} [y(t+k+1|t) - r(t)]^2 + 0.04\Delta u^2(t)$$




- Closed-loop MPC:

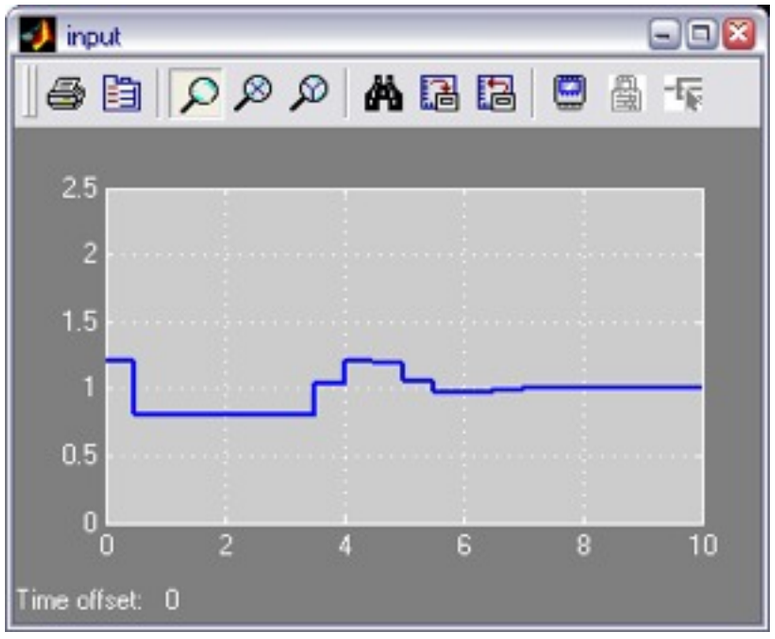


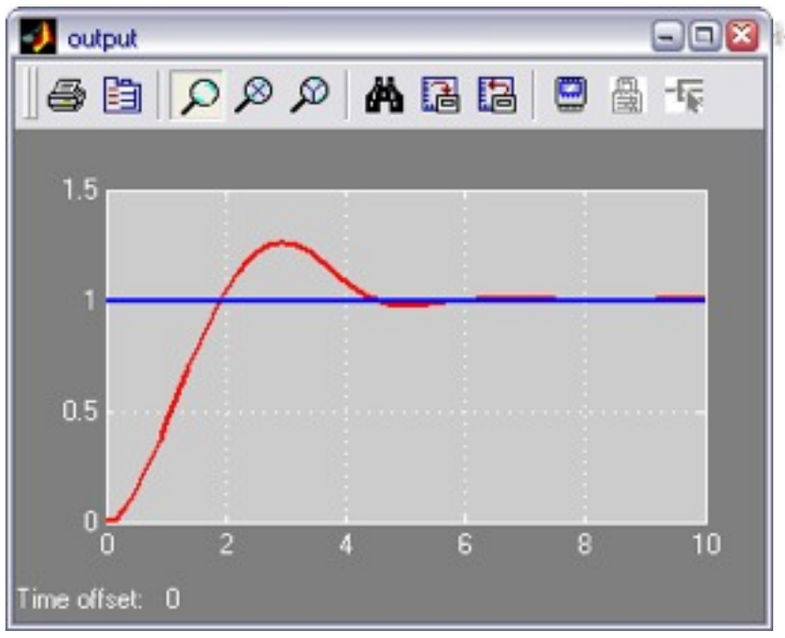



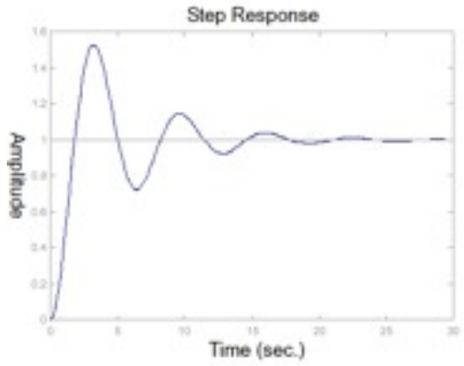

go to demo **`linear/example3.m`** (Hyb-Tbx)

# Linear MPC - Example

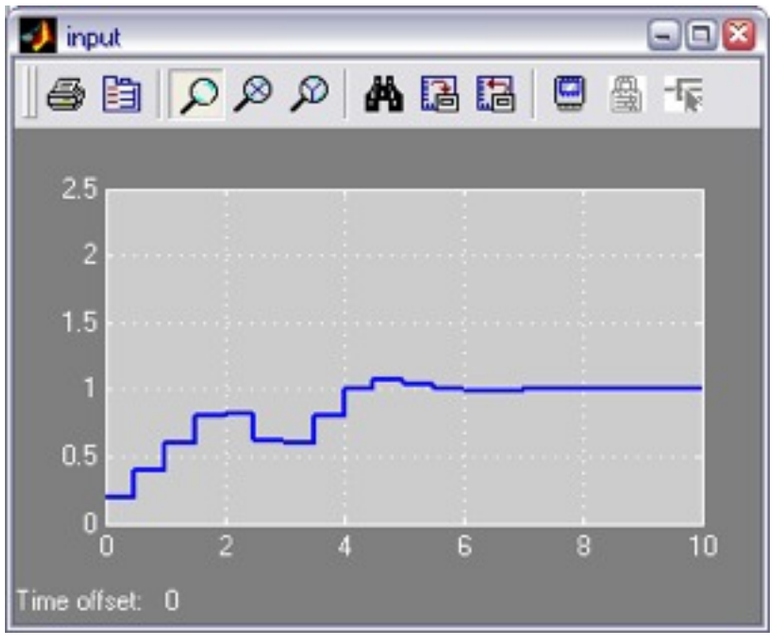
- Constraint $0.8 \leq u(t) \leq 1.2$ (amplitude)

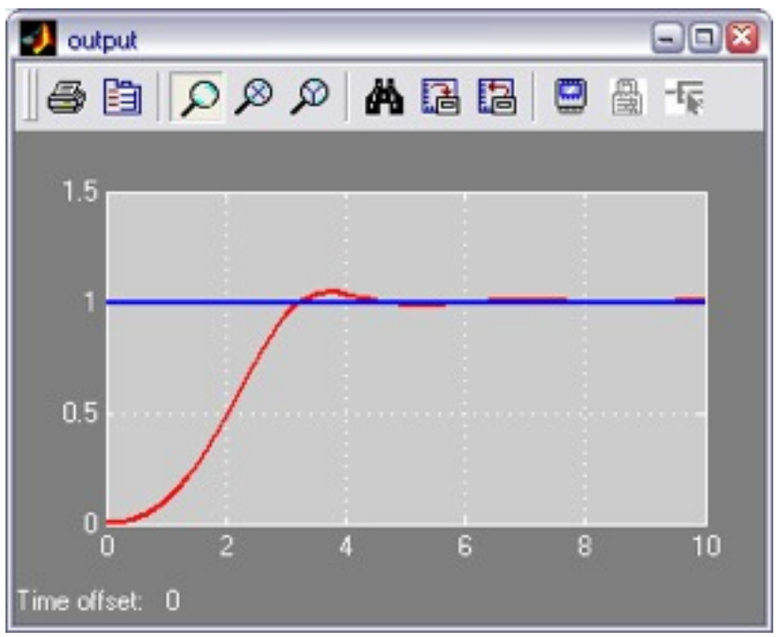



- Constraint $-0.2 \leq \Delta u(t) \leq 0.2$ (slew-rate)



# Anticipative Action

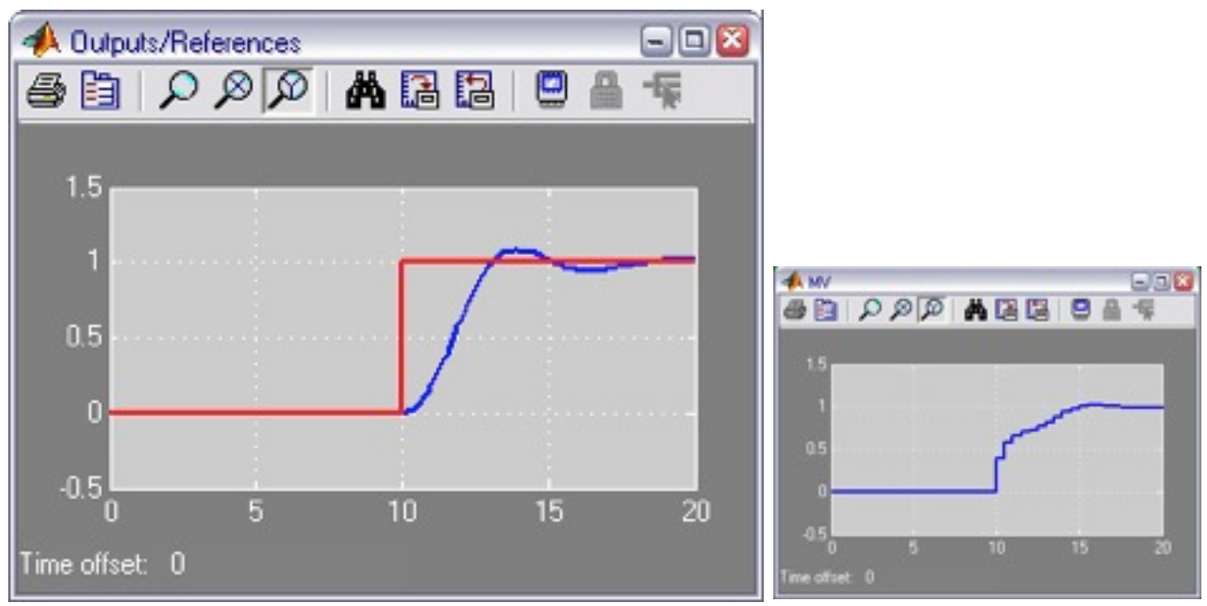
$$\min_{\Delta U} \sum_{k=0}^{N-1} \|W^y(y(k+1) - r(t+k+1|t))\|^2 + \|W^{\Delta u}\Delta u(k)\|^2$$

- Future reference samples (partially) known in advance (anticipating action):

$$r(t+k|t) = \begin{cases} r(t+k) & \text{if } k = 0,\dots,N_r \\ r(t+N_r) & \text{if } k > N_r \end{cases}$$

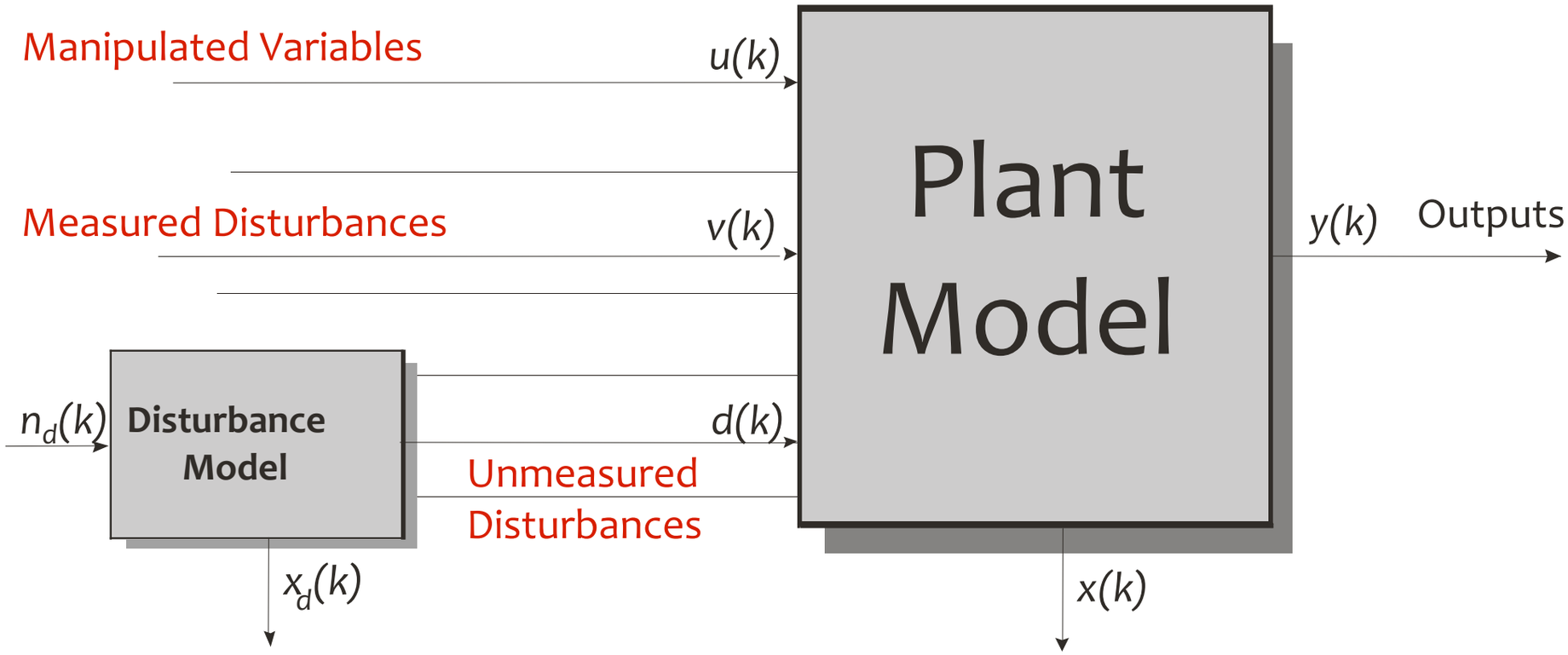

- Reference not known in advance (causal):

$$r(t+k|t) \equiv r(t), \quad \forall k \geq 0$$




go to demo **`mpcpreview.m`** (MPC-Tbx)

# Measured and Unmeasured Disturbances




Linear model for MPC optimization

$$\begin{cases} x(k+1) = Ax(k) + B_u u(k) + B_v v(k) + B_d d(k) \\ y(k) = Cx(k) + D_v v(k) + D_d d(k) \end{cases}$$

$$\begin{cases} x_d(k+1) = \bar{A} x_d(k) + \bar{B} n_d(k) \\ d(k) = \bar{C} x_d(k) + \bar{D} n_d(k). \end{cases}$$

($n_d(k)$ = white Gaussian noise, $n_d(t+k|t)$=0 over the prediction horizon)

more details about disturbance models later on …

# Soft Constraints

- To prevent QP infeasibility, relax output constraints:

$$\min_z \quad \sum_{k=0}^{N-1} \|W^y(y(t+k+1|t) - r(t))\|^2 + \|W^{\Delta u}\Delta u(t)\|^2 + \rho_\epsilon \epsilon^2$$

$$\text{subj. to} \quad u_{\min} \le u(t+k) \le u_{\max}, \;\; k = 0, \ldots, N-1$$
$$\Delta u_{\min} \le \Delta u(t+k) \le \Delta u_{\max}, \;\; k = 0, \ldots, N-1$$
$$y_{\min} - \epsilon V_{\min} \le y(t+k|t) \le y_{\max} + \epsilon V_{\max}, \;\; k = 1, \ldots, N$$

$\epsilon$ = "panic" variable

$$z = [\Delta u'(0) \;\; \Delta u'(1) \;\; \ldots \;\; \Delta u'(N-1) \;\; \epsilon]'$$

$$\rho_\epsilon \gg W^y, W^{\Delta u}$$

$V_{\min}$, $V_{\max}$ = vectors with entries $\geq$0 (the larger the entry, the relatively softer the corresponding constraint)

- Infeasibility can be due to:
  - modeling errors
  - disturbances
  - wrong MPC setup (e.g., prediction horizon is too short)

# Delays – Method 1

- Linear model w/ delays:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t-\tau) \\ y(t) &= Cx(t) \end{aligned}$$

- Map delays to poles in $z$=0:

$$x_k(t) \triangleq u(t-k) \quad \Rightarrow \quad x_k(t+1) = x_{k-1}(t) \qquad\qquad k = 1, \ldots, \tau$$

$$\begin{bmatrix} x \\ x_\tau \\ x_{\tau-1} \\ \vdots \\ x_1 \end{bmatrix}(t+1) = \begin{bmatrix} A & B & 0 & 0 & \ldots & 0 \\ 0 & 0 & I_m & 0 & \ldots & 0 \\ 0 & 0 & 0 & I_m & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 0 \end{bmatrix} \begin{bmatrix} x \\ x_\tau \\ x_{\tau-1} \\ \vdots \\ x_1 \end{bmatrix}(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ I_m \end{bmatrix} u(t)$$

- Apply MPC to the extended system

# Delays – Method 2

- Linear model w/ delays:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t-\tau) \\ y(t) &= Cx(t) \end{aligned}$$

- Delay-free model:

$$\bar{x}(t) \triangleq x(t+\tau)$$

$$\begin{aligned} \bar{x}(t+1) &= A\bar{x}(t) + Bu(t) \\ \bar{y}(t) &= C\bar{x}(t) \end{aligned}$$

- Design MPC for delay-free model:

$$u(t) = f_{\mathsf{MPC}}(\bar{x}(t))$$

- Compute the predicted state

$$\bar{x}(t) = x(t+\tau) = A^{\tau}x(t) + \sum_{j=0}^{\tau-1} A^{j}Bu(t-1-j)$$

- Compute MPC action accordingly:

$$u(t) = f_{\mathsf{MPC}}(\bar{x}(t+\tau))$$

For better closed-loop performance one can predict $x(t+\tau)$ with a much more complex model than (A,B,C) !

# MPC vs. Conventional Control

Single input/single output control loop w/ constraints:

equivalent performance can be obtained with other simpler control techniques (e.g.: PID + anti-windup)

HOWEVER

MPC allows (in principle) UNIFORMITY
(i.e. same technique for wide range of problems)

- reduce training
- reduce cost
- easier design maintenance

Satisfying control specs and walking on water is similar …

both are not difficult if frozen !

# MPC Features

- Multivariable constrained “non-square” systems (i.e. #inputs and #outputs are different)
- Delay compensation
- Anticipative action for future reference changes
- “Integral action”, i.e. no offset for step-like inputs

Price to pay:

- Substantial on-line computation
- For simple small/fast systems other techniques dominate (e.g. PID + anti-windup)
- New possibilities for MPC: explicit piecewise linear forms

# MPC Theory

- Historical Goal: Explain the success of DMC
- Present Goal: Improve, simplify, and extend industrial algorithms
- Areas:
  - Linear MPC: linear model
  - Nonlinear MPC: nonlinear model
  - Robust MPC: uncertain (linear) model
  - Hybrid MPC: model integrating logic, dynamics, and constraints
- Issues:
  - Feasibility
  - Stability (Convergence)
  - Computations

(Mayne, Rawlings, Rao, Scokaert, 2000)

# Feasibility

$$\min_{\Delta U} \quad \sum_{k=0}^{N-1} \|W^y(y(t+k+1|t) - r(t))\|^2 + \|W^{\Delta u}\Delta u(k)\|^2$$

$$\text{subj. to} \quad u_{\min} \le u(t+k) \le u_{\max}, \;\; k = 0, \ldots, N-1$$
$$\Delta u_{\min} \le \Delta u(t+k) \le \Delta u_{\max}, \;\; k = 0, \ldots, N-1$$
$$y_{\min} \le y(t+k|t) \le y_{\max}, \;\; k = 1, \ldots, N$$

**QUADRATIC PROGRAM (QP)**

- **Feasibility**: Guarantee that the QP problem is feasible at all sampling times $t$

- Input constraints only: no feasibility issues !

- Hard output constraints:

  - When N$<\infty$ there is no guarantee that the QP problem will remain feasible at all future time steps $t$

  - N=$\infty$ ⟹ infinite number of constraints !

  - Maximum output admissible set theory: N$<\infty$ is enough
    (Gilbert, Tan, *IEEE TAC*,1991), (Kerrigan, Maciejowski,*CDC*,2000),
    (Chmielewski, Manousiouthakis, *Sys. Cont. Letters*, 1996)

# Stability

$$\min_{u(0),\ldots,u(N-1)} \sum_{k=0}^{N-1} \Big[x'(k)Qx(k) + u'(k)Ru(k)\Big] + x'(N)Px(N)$$

$$\text{s.t.} \quad \begin{aligned} u_{\min} &\le u(k) \le u_{\max}, \quad k = 0,\ldots,N-1 \\ y_{\min} &\le y(k) \le y_{\max}, \quad k = 1,\ldots,N \end{aligned}$$

$$Q = Q' \succeq 0, \; R = R' \succ 0, \; P \succeq 0$$

- Stability is a complex function of the MPC parameters
  $N, Q, R, P, u_{min}, u_{max}, y_{min}, y_{max}$

- **Stability constraints** and weights on the terminal state can be imposed over the prediction horizon to ensure stability of MPC

# Convergence Result

**Theorem 1** *Consider the linear system*

$$\begin{cases} x(t+1) &= Ax(t)+Bu(t) \\ y(t) &= Cx(t) \end{cases}$$

*and the MPC control law based on*

$$\min_U J(U,x(t)) = \sum_{k=0}^{N-1} \left\{ x'(t+k|t)Qx(t+k|t) + u'(t+k)Ru(t+k) \right\}$$

subj. to

$$y_{min} \le y(t+k) \le y_{max}$$
$$u_{min} \le u(t+k) \le u_{max}$$
$$x(t+N|t) = 0$$

*Assume that the optimization problem is feasible at time $t = 0$. Then, for all $R > 0$, $Q > 0$,*

$$\lim_{t\to\infty} x(t) = 0,$$
$$\lim_{t\to\infty} u(t) = 0,$$

*and the constraints are satisfied at all time instants $t \ge 0$.*

(Keerthi and Gilbert, 1988)(Bemporad *et al.*, 1994)

Proof: Use value function as Lyapunov function

# Convergence Proof

- Let $\mathcal{U}_t^*$ denote the optimal control sequence @$t$ $\{u_t^*(0), \ldots, u_t^*(N-1)\}$
- Let $V(t) \triangleq J(\mathcal{U}_t^*, x(t))$=value function ➡ Lyapunov function
- By construction, $\mathcal{U}_1 = \{u_t^*(1), \ldots, u_t^*(N-1), 0\}$ is feasible @$t+1$, and hence

$$V(t+1) = J(\mathcal{U}^*{}_{t+1}, x(t+1)) \leq J(\mathcal{U}_1, x(t+1)) =$$
$$= V(t) - x'(t)Qx(t) - u'(t)Ru(t)$$

- $V(t)$ is decreasing and lower-bounded by 0 $\Rightarrow \exists V_\infty = \lim_{t\to\infty} V(t) \Rightarrow$ $V(t+1) - V(t) \to 0$, which implies $x'(t)Qx(t), u'(t)Ru(t) \to 0$
- Since $R, Q > 0$, $u(t), x(t) \to 0$

**Global optimum is not needed to prove convergence !**

# Convergence Result (a little more general)

**Theorem 1** *Consider the linear system*

$$\begin{cases} x(t+1) &= Ax(t)+Bu(t) \\ y(t) &= Cx(t) \end{cases}$$

*and the MPC control law based on*

$$\min_U J(U,x(t)) = \sum_{k=0}^{N-1} \left\{ y'(t+k|t)Qy(t+k|t) + u'(t+k)Ru(t+k) \right\}$$

subj. to

$$y_{min} \le y(t+k) \le y_{max}$$
$$u_{min} \le u(t+k) \le u_{max}$$

*Assume that the optimization problem is feasible at time $t=0$. Then, for either $N \to \infty$ or with the extra constraint $x(t+N|t)=0$, for all $R>0$, $Q \ge 0$,*

$$\lim_{t\to\infty} y(t) = 0,$$
$$\lim_{t\to\infty} u(t) = 0,$$

*while fulfilling the constraints. Moreover, provided that $(Q^{\frac{1}{2}}C, A)$ is a detectable pair, $\lim_{t\to\infty} x(t) = 0$.*

(Keerthi and Gilbert, 1988)(Bemporad *et al.*, 1994)

Proof: Use value function as Lyapunov function

# Convergence Proof

- Assume we set the terminal constraint $x(t+N|t)=0$
- Let $\mathcal{U}_t^*$ denote the optimal control sequence @$t$ $\{u_t^*(0),\dots,u_t^*(N-1)\}$
- Let $V(t) \triangleq J(\mathcal{U}_t^*, x(t))$=value function ➡ Lyapunov function
- By construction, $\mathcal{U}_1 = \{u_t^*(1),\dots,u_t^*(N-1),0\}$ is feasible @$t+1$, and hence

$$V(t+1) = J(\mathcal{U}^*{}_{t+1}, x(t+1)) \leq J(\mathcal{U}_1, x(t+1)) =$$
$$= V(t) - y'(t)Qy(t) - u'(t)Ru(t)$$

- $V(t)$ is decreasing and lower-bounded by 0 $\Rightarrow \exists V_\infty = \lim_{t\to\infty} V(t) \Rightarrow V(t+1)-V(t)\to 0$, which implies $y'(t)Qy(t), u'(t)Ru(t)\to 0$
- Since $R>0$, $u(t)\to 0$
- Assume for simplicity that $C=I$ (i.e., $y(t)=x(t)$) and $Q>0$. Then also $x(t)\to 0$.

**Global optimum is not needed to prove convergence !**

# Convergence Proof

If $Q \geq 0$ and/or $C \neq I$:

- For all $\forall k = 0, \ldots, n-1$, we have

$$\lim_{t\to\infty} y'(t+k)Qy(t+k) = \lim_{t\to\infty} \|Q^{\frac{1}{2}}C(A^k x(t) + \sum_{j=0}^{k-1} A^j Bu(t+k-1-j))\|^2 = 0$$

- As $u(t) \to 0$, also $Q^{\frac{1}{2}}CA^k x(t) \to 0$, and hence $\Theta x(t) \to 0$, where $\Theta$ is the observability matrix of $(Q^{\frac{1}{2}}C, A)$.

- If $(Q^{\frac{1}{2}}C, A)$ is observable, this also implies $x(t) \to 0$.

- If $(Q^{\frac{1}{2}}C, A)$ is only detectable, through a canonical decomposition one can observe that, as $u(t) \to 0$, unobservable modes go to zero spontaneously.

# Convergence Proof

- Similar argument for infinite prediction horizon $N = \infty$:
  - Let $\mathcal{U}_t^*$ denote the infinite optimal control sequence @$t$ $\{u_t^*(0), u_t^*(1) \ldots\}$
  - Let $V(t) \triangleq J(\mathcal{U}_t^*, x(t))$=value function ($\Rightarrow$ Lyapunov function)
  - Because constraints were checked up to $t+k = \infty$, $\mathcal{U}_1 = \{u_t^*(1), u_t^*(2), \ldots\}$ is feasible @$t+1$ by construction.
  - Hence

$$V(t+1) = J(\mathcal{U}^*{}_{t+1}, x(t+1)) \leq J(\mathcal{U}_1, x(t+1)) = \\ = V(t) - y'(t)Qy(t) - u'(t)Ru(t)$$

  - Repeat same arguments as before

# Stability Constraints

1. No constraint, infinite output horizon:

$$N \to \infty$$

(Keerthi and Gilbert, 1988) (Rawlings and Muske, 1993)

2. End-point constraint:

$$x(t+N|t) = 0$$

(Kwon and Pearson, 1977) (Keerthi and Gilbert, 1988)

3. Relaxed terminal constraint:

$$x(t+N|t) \in \Omega$$

(Scokaert and Rawlings, 1996)

4. Contraction constraint:

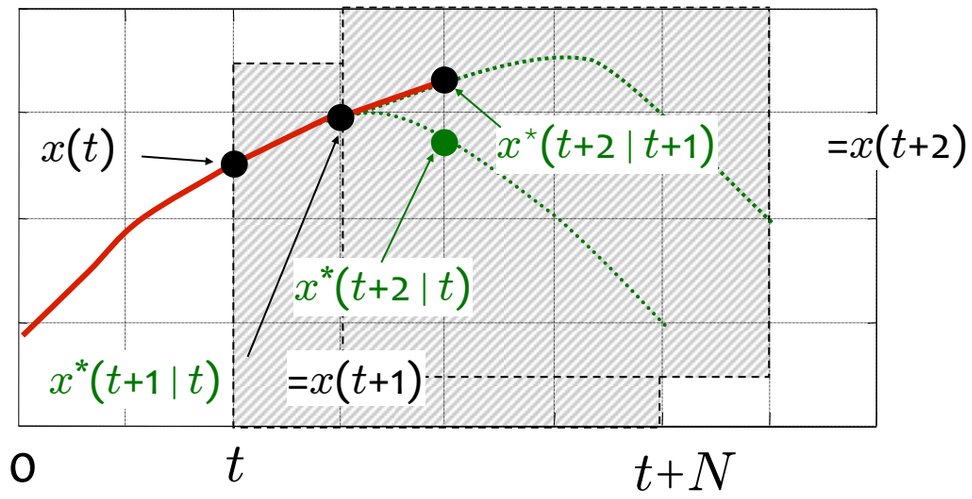
$$\|x(t+1|t)\| \leq \alpha \|x(t)\|, \quad \alpha < 1$$

(Polak and Yang, 1993) (Bemporad, 1998)

All the proofs in (1,2,3) use the value function $V(t) = \min_U J(U,t)$ as a Lyapunov function

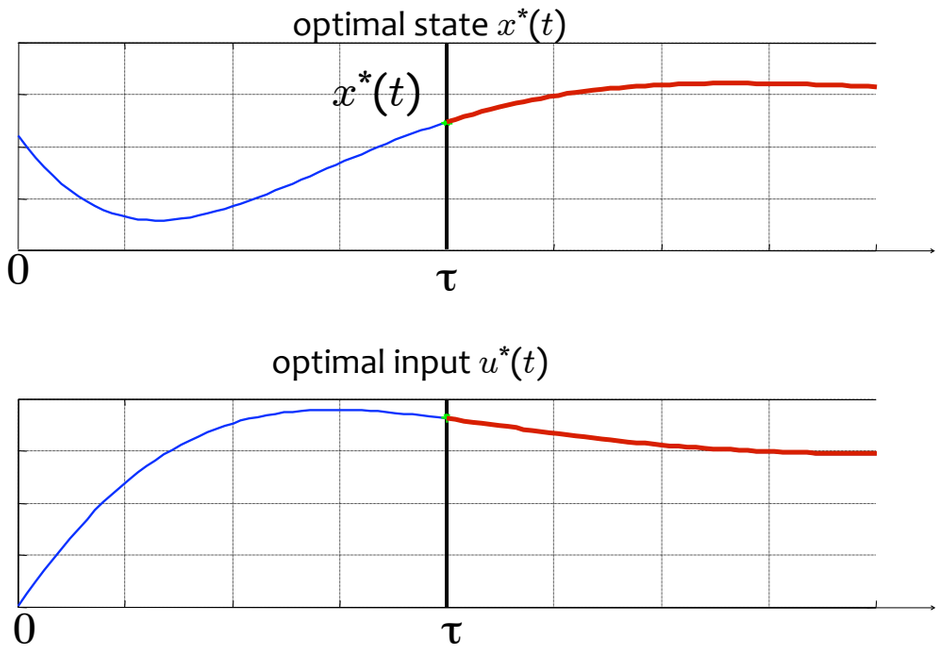
# Predicted and Actual Trajectories

- Even assuming perfect model & no disturbances:




**predicted** open-loop trajectories may be different from **actual** closed-loop trajectories

- Special case: for **infinite horizon**, open-loop trajectories and losed-loop trajectories coincide. This follows by Bellman's principle of optimality.




Richard Bellman
(1920 - 1984)

# Input and Output Horizons

$$\min_{\Delta U} \quad \sum_{k=0}^{N-1} \|W^y(y(t+k+1|t) - r(t))\|^2 + \|W^{\Delta u}\Delta u(t)\|^2$$

$$\text{subj. to} \quad u_{\min} \le u(t+k) \le u_{\max}, \;\; k = 0, \ldots, N-1$$

$$\Delta u_{\min} \le \Delta u(t+k) \le \Delta u_{\max}, \;\; k = 0, \ldots, N_u - 1$$

$$y_{\min} \le y(t+k|t) \le y_{\max}, \;\; k = 1, \ldots, N$$

$$\Delta u(t+k) = 0, \;\; k = N_u, \ldots, N-1$$

- Input horizon $N_u$ can be shorter than output horizon $N$
- $N_u < N$ = less degrees of freedom, and hence:
  - Loss of performance
  - Decreased computation time (QP is smaller)
  - Feasibility still maintained (constraints are still checked up to $N$)

typically $N_u$=1÷10

# MPC and LQR

Jacopo Francesco Riccati (1676 - 1754)

- Consider the MPC control law:

$$\min_U J(U,t) = \; x'(t+N|t)Px(t+N|t) + \sum_{k=0}^{N-1} \left\{ x'(t+k|t)Qx(t+k|t) + u'(t+k)Ru(t+k) \right\}$$

$R = R' > 0$, $Q = Q' \geq 0$, and $P$ satisfies the Riccati equation

$$P = A'PA - A'PB(B'PB + R)^{-1}B'PA + Q$$

**(Unconstrained) MPC = LQR**

# MPC and LQR

- Consider the MPC control law:

$$\min_U J(U,t) = \; x'(t+T|t)Px(t+T|t) + \sum_{k=0}^{N-1} \left\{ x'(t+k|t)Qx(t+k|t) + u'(t+k)Ru(t+k) \right\}$$

subj. to

$$y_{min} \le y(t+k|t) \le y_{max}, \;\; k = 1,\dots,N$$
$$u_{min} \le u(t+k) \le u_{max}, \;\; k = 0,\dots,N-1$$
$$u(t+k) = Kx(t+k|t), \;\; k = N_u,\dots,N-1$$

Jacopo Francesco Riccati (1676 - 1754)

$R = R' > 0$, $Q = Q' \ge 0$, and $P$, $K$ satisfy the Riccati equation

$$\begin{aligned} K &= -(R + B'PB)^{-1}B'PA \\ P &= (A+BK)'P(A+BK) + K'RK + Q \end{aligned}$$

- In a polyhedral region around the origin the MPC control law is equivalent to the constrained LQR controller with weights $Q$,$R$

**MPC = constrained LQR**

(Chmielewski, Manousiouthakis, 1996)

(Scokaert and Rawlings, 1998)

- The larger the horizon, the larger the region where MPC=LQR

# Double Integrator Example

- System: $y(t) = \frac{1}{s^2}u(t)$

sampling + ZOH
$T_s$=1 s

$$\begin{aligned} x(t+1) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \end{aligned}$$

- Constraints: $-1 \le u(t) \le 1$

- Control objective: min $\sum_{t=0}^{\infty} y^2(t) + \frac{1}{100}u^2(t)$

$$u(k) = K_{LQ}x(k), \ \forall k \ge N = 2$$

LQ gain

$$\Rightarrow \left( \sum_{k=0}^{1} y^2(k) + \frac{1}{100}u^2(k) \right) + x'(2) \begin{bmatrix} 2.1429 & 1.2246 \\ 1.2246 & 1.3996 \end{bmatrix} x(2)$$

solution of algebraic Riccati equation

- Optimization problem

$$H = \begin{bmatrix} 0.8365 & 0.3603 \\ 0.3603 & 0.2059 \end{bmatrix}, \ F = \begin{bmatrix} 0.4624 & 1.2852 \\ 0.1682 & 0.5285 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, \ W = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \ S = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

(cost function was normalized by max svd(H))

# Example: AFTI-16

- Linearized model:

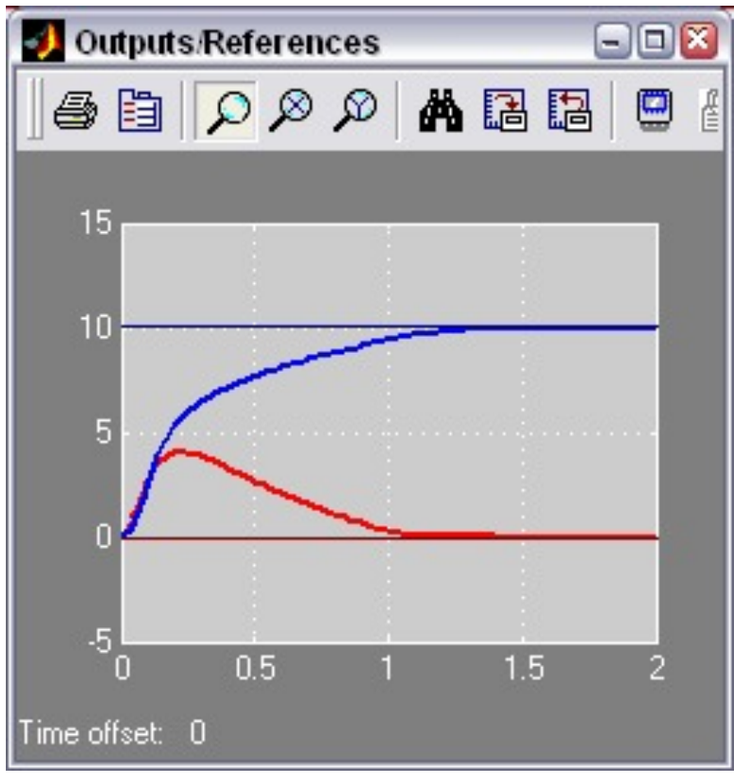
$$\begin{cases} \dot{x} = \begin{bmatrix} -.0151 & -60.5651 & 0 & -32.174 \\ -.0001 & -1.3411 & .9929 & 0 \\ .00018 & 43.2541 & -.86939 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x + \\ \qquad \begin{bmatrix} -2.516 & -13.136 \\ -.1689 & -.2514 \\ -17.251 & -1.5766 \\ 0 & 0 \end{bmatrix} u \\ y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x, \end{cases}$$

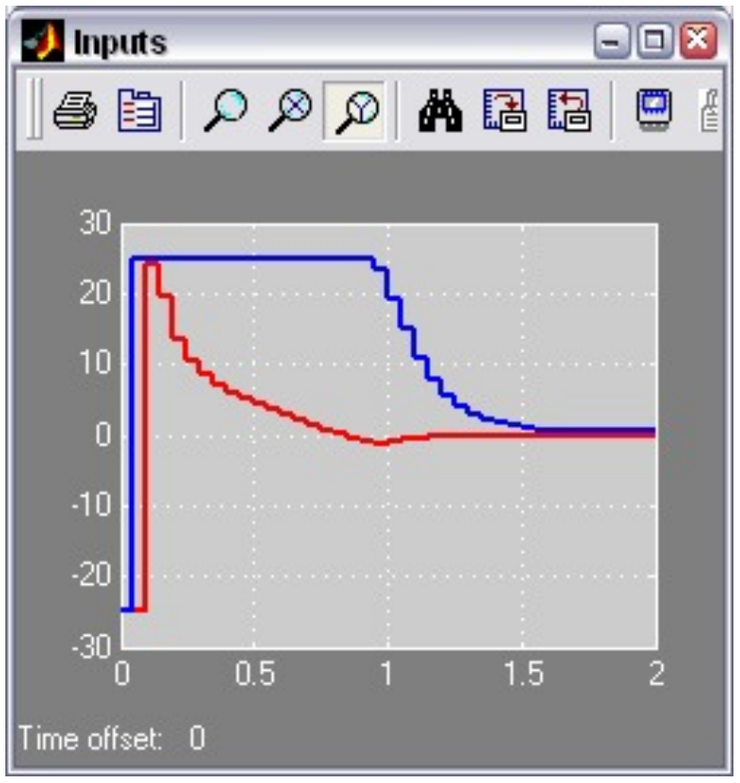
- Inputs: elevator and flaperon angle
- Outputs: attack and pitch angle
- Sampling time: $T_s = .05$ s (+ zero-order hold)
- Constraints: max 25° on both angles
- Open-loop response: unstable
  (open-loop poles: $-7.6636, -0.0075 \pm 0.0556j, 5.4530$)

go to demo **`/demos/linear/afti16.m`** (Hyb-Tbx)

**`afti16.m`** (MPC-Tbx)

# Example: AFTI-16

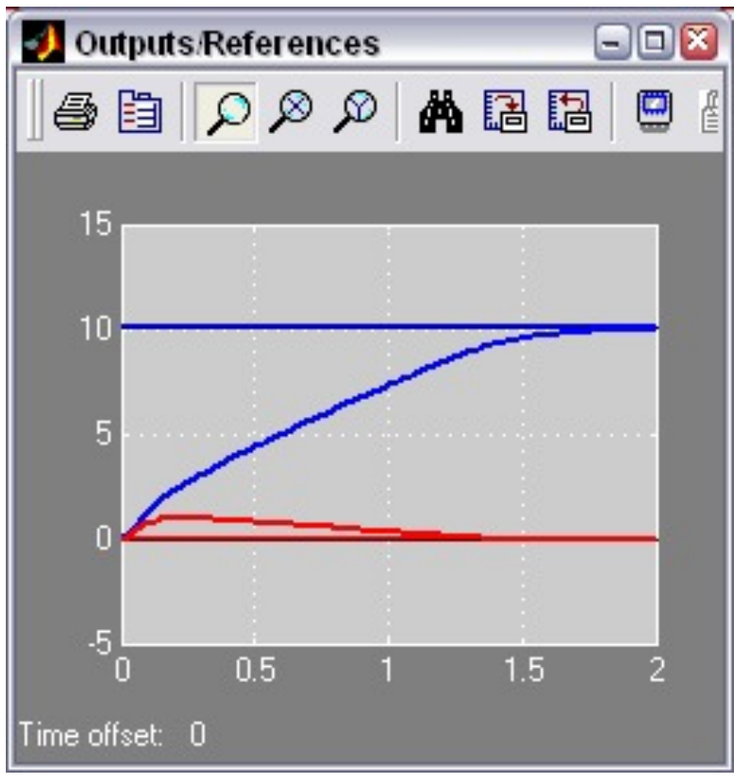


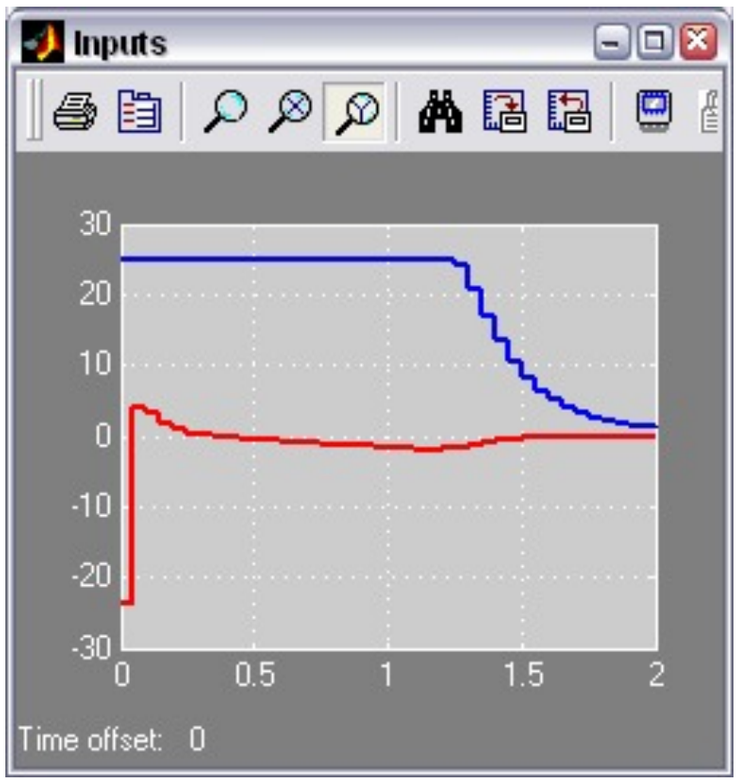




$N_y = 10,\ N_u = 3,$

$w_y = \{10, 10\},\ w_{\delta u} = \{.01, .01\},$

$u_{\min} = -25°,\ u_{\max} = 25°$



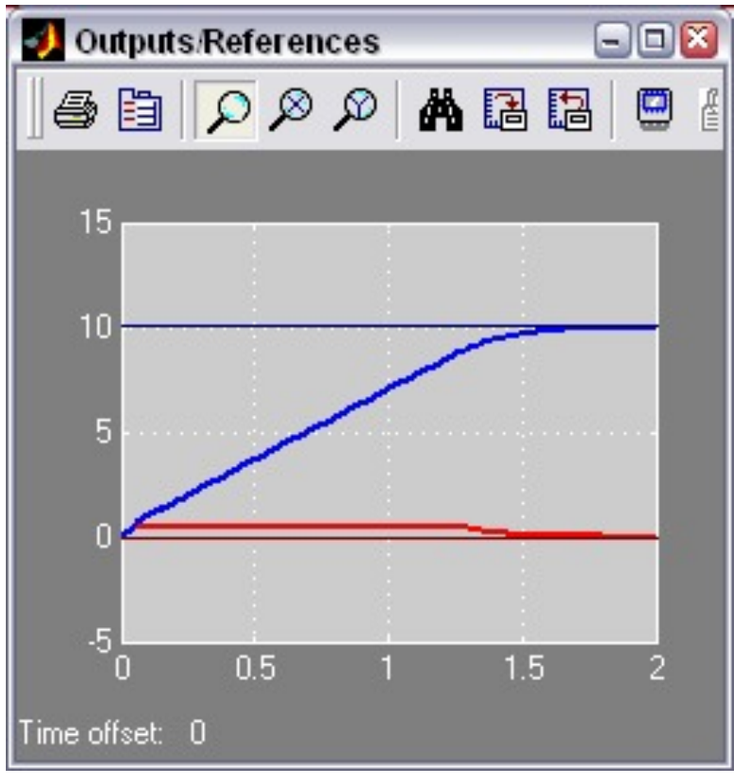



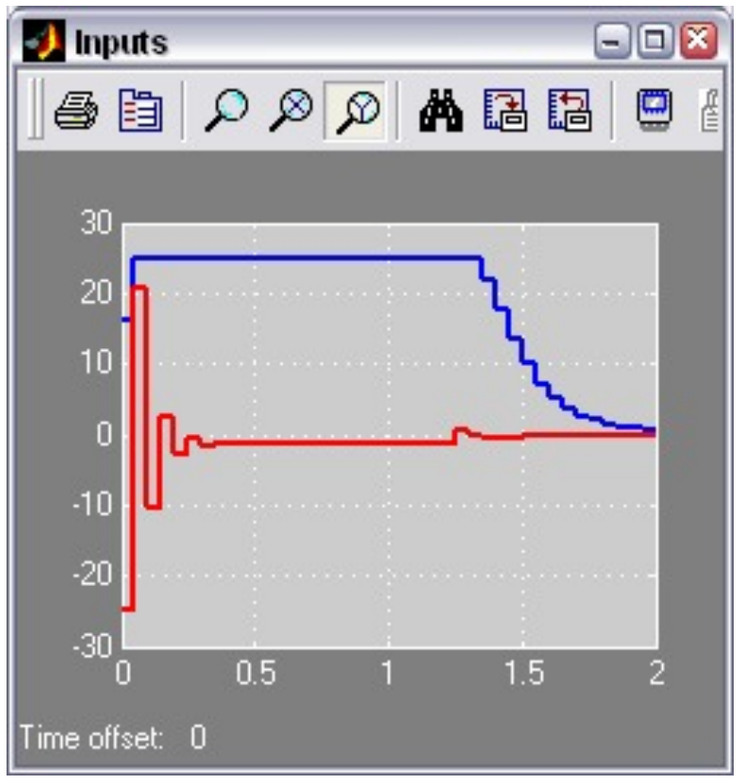

$N_y = 10,\ N_u = 3,$

$w_y = \{100, 10\},\ w_{\delta u} = \{.01, .01\},$

$u_{\min} = -25°,\ u_{\max} = 25°$

# Example: AFTI-16




$N_y = 10,\ N_u = 3,$

$w_y = \{10, 10\},\ w_{\delta u} = \{.01, .01\},$

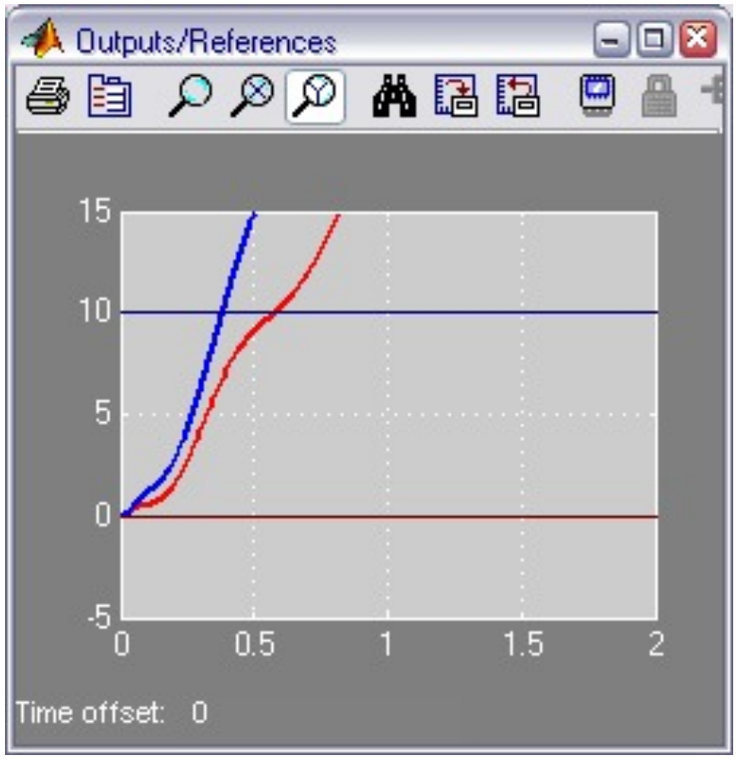
$u_{\min} = -25°,\ u_{\max} = 25°,$

$y_{1,\min} = -0.5°,\ y_{1,\max} = 0.5°$

# Example: AFTI-16

Unconstrained MPC (=linear controller, ≈ LQR)

+ actuator saturation ±25°





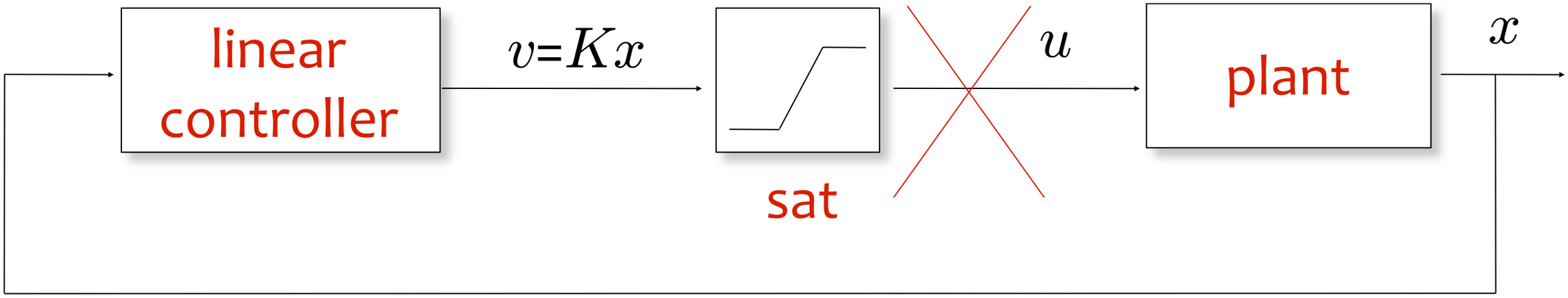
$$N_y = 10,\ N_u = 3,$$
$$w_y = \{10, 10\},\ w_{\delta u} = \{.01, .01\},$$

UNSTABLE !!!

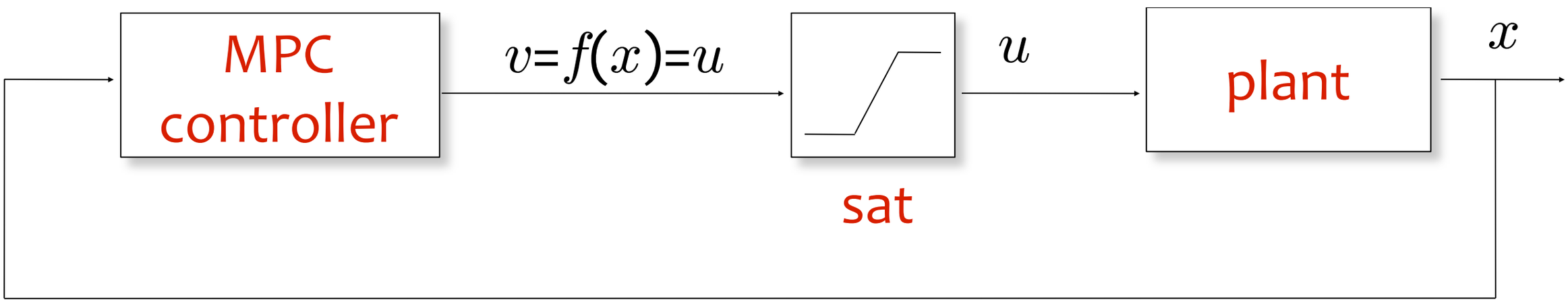
Saturation needs to be considered in the control design !

# Saturation

- Saturation needs to be considered in the control design




- MPC takes it into account automatically (and optimally)



# Tuning Guidelines

$$\min_{\Delta U} \quad \sum_{k=0}^{N-1} \|W^y(y(t+k+1|t) - r(t))\|^2 + \|W^{\Delta u}\Delta u(t)\|^2$$

$$\text{subj. to} \quad \begin{aligned} & u_{\min} \le u(t+k) \le u_{\max}, \;\; k = 0, \ldots, N-1 \\ & \Delta u_{\min} \le \Delta u(t+k) \le \Delta u_{\max}, \;\; k = 0, \ldots, N_u - 1 \\ & y_{\min} \le y(t+k|t) \le y_{\max}, \;\; k = 1, \ldots, N \\ & \Delta u(t+k) = 0, \;\; k = N_u, \ldots, N-1 \end{aligned}$$

- **Weights:** the larger the ratio $W^y \triangleleft W^{\Delta u}$ the more aggressive the controller
- **Input horizon:** the larger $N_u$, the more "optimal" but the more complex the controller
- **Output horizon:** the smaller $N$, the more aggressive the controller
- **Limits:** controller less aggressive if $\Delta u_{\min}$, $\Delta u_{\max}$ are small

Always try to set $N_u$ as small as possible !

# Scaling

- Humans think infinite precision ...
- Computers do not !
- Numerical difficulties may arise if variables assume very small or very large values

**Example:** $y_1 \in$ [-1e-4,1e-4] (V)

$y_2 \in$ [-1e4,1e4] (Pa)

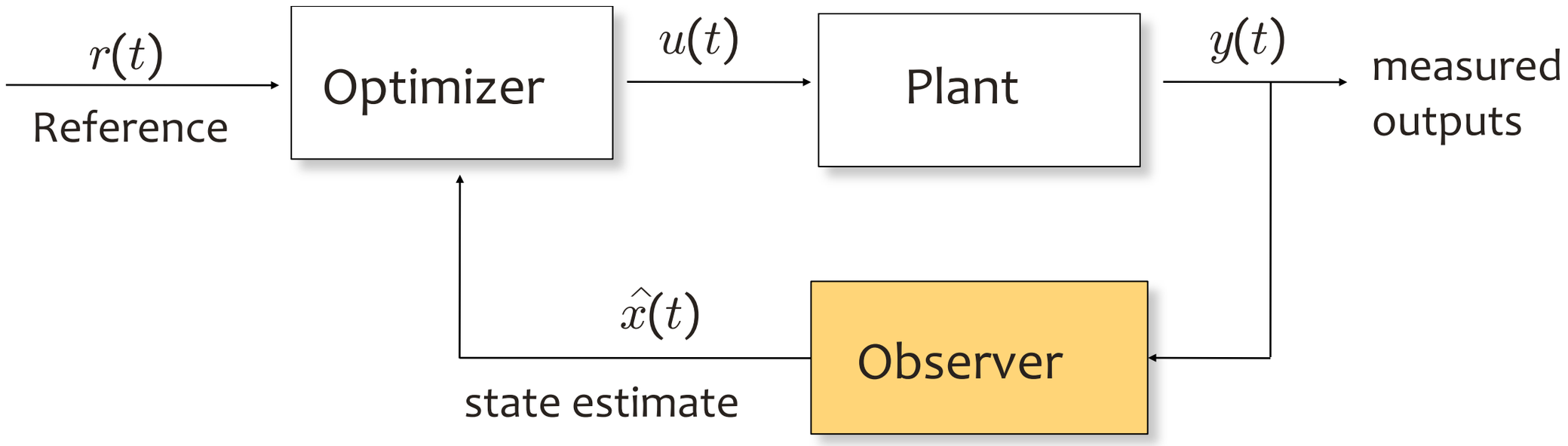
use instead: $y_1 \in$ [-0.1,0.1] (mV)

$y_2 \in$ [-10,10] (kPa)

- Ideally all variables should range in [-1,1]. For example, one can replace $y$ with $y/y_{\max}$

# Observer Design in MPC

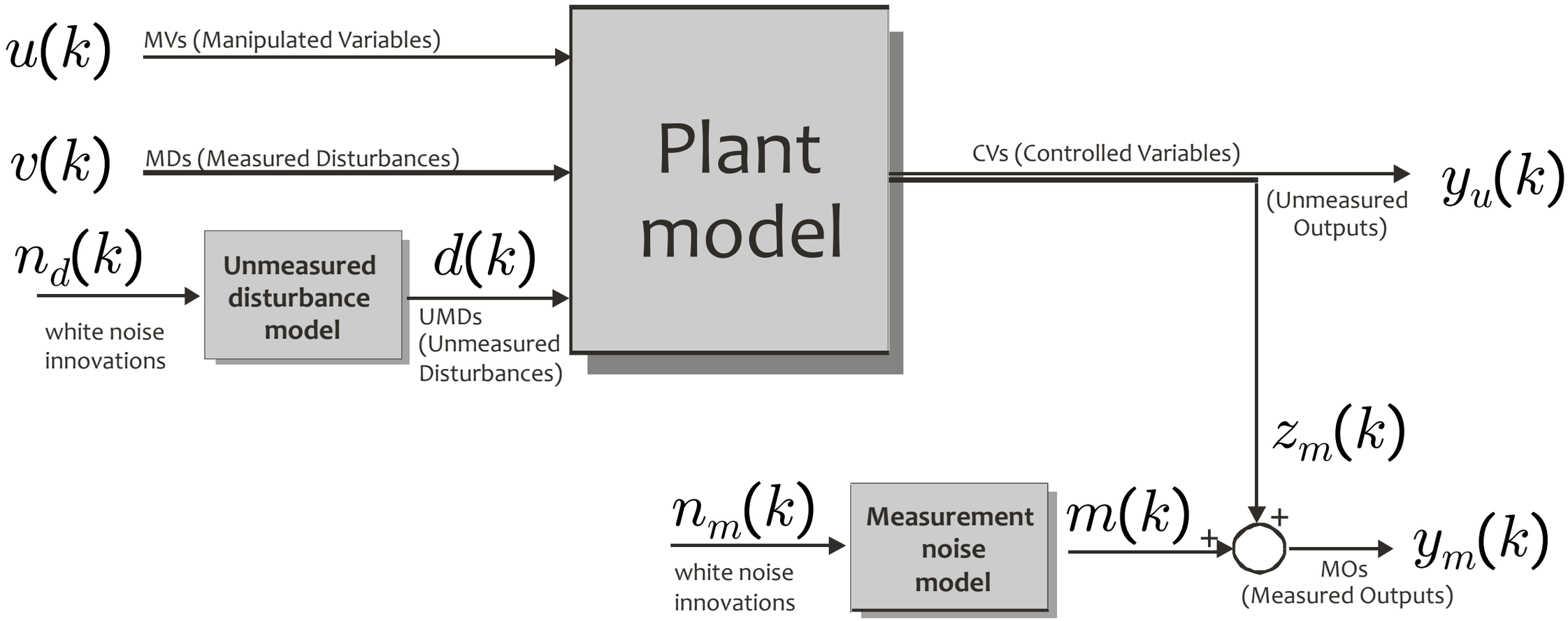
# Observer Design for MPC




- Full state $x(t)$ may not be available
- Even if available, noise should be filtered out
- State of prediction model may be different from plant model $x(t)$ (e.g.: model reduction, identification)

**we need to use a state observer**

# Model for Observer Design




unmeasured disturbance model

$$\begin{cases} x_d(k+1) = \bar{A}x_d(k) + \bar{B}n_d(k) \\ d(k) = \bar{C}x_d(k) + \bar{D}n_d(k) \end{cases}$$

measurement noise model

$$\begin{cases} x_m(k+1) = \tilde{A}x_m(k) + \tilde{B}n_m(k) \\ m(k) = \tilde{C}x_m(k) + \tilde{D}n_m(k) \end{cases}$$

- Note: meas. noise model not needed for optimization !

# Observer Design

- Measurement update

$$\begin{bmatrix} \widehat{x}(k|k) \\ \widehat{x}_d(k|k) \\ \widehat{x}_m(k|k) \end{bmatrix} = \begin{bmatrix} \widehat{x}(k|k-1) \\ \widehat{x}_d(k|k-1) \\ \widehat{x}_m(k|k-1) \end{bmatrix} + M\left[y_m(k) - \widehat{y}_m(k)\right]$$

- Time update

$$\begin{aligned} \widehat{x}(k+1|k) &= A\widehat{x}(k|k) + B_u u(k) + B_v v(k) + B_d \bar{C}\widehat{x}_d(k|k) \\ \widehat{x}_d(k+1|k) &= \bar{A}\widehat{x}_d(k|k) \\ \widehat{x}_m(k+1|k) &= \tilde{A}\widehat{x}(k|k) \\ \widehat{y}_m(k) &= C_m\widehat{x}(k|k-1) + D_{vm}v(k) + \\ &\quad D_{dm}\bar{C}\widehat{x}_d(k|k-1) + \tilde{C}\widehat{x}_m(k|k-1) \end{aligned}$$

- NOTE: **separation principle** holds ! (under certain assumptions)

(Muske, Meadows, Rawlings, ACC94)

# Kalman Filter Design

Rudolf Emil Kalman (1930 - )

- Full model for designing observer gain $M$

$$\begin{bmatrix} x(k+1) \\ x_d(k+1) \\ x_m(k+1) \end{bmatrix} = \begin{bmatrix} A & B_d\bar{C} & 0 \\ 0 & \bar{A} & 0 \\ 0 & 0 & \tilde{A} \end{bmatrix} \begin{bmatrix} x(k) \\ x_d(k) \\ x_m(k) \end{bmatrix} + \begin{bmatrix} B_u \\ 0 \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} B_v \\ 0 \\ 0 \end{bmatrix} v(k) + \begin{bmatrix} B_d\bar{D} \\ \bar{B} \\ 0 \end{bmatrix} n_d(k) + \begin{bmatrix} 0 \\ 0 \\ \tilde{B} \end{bmatrix} n_m(k) + \begin{bmatrix} B_u \\ 0 \\ 0 \end{bmatrix} n_u(k)$$

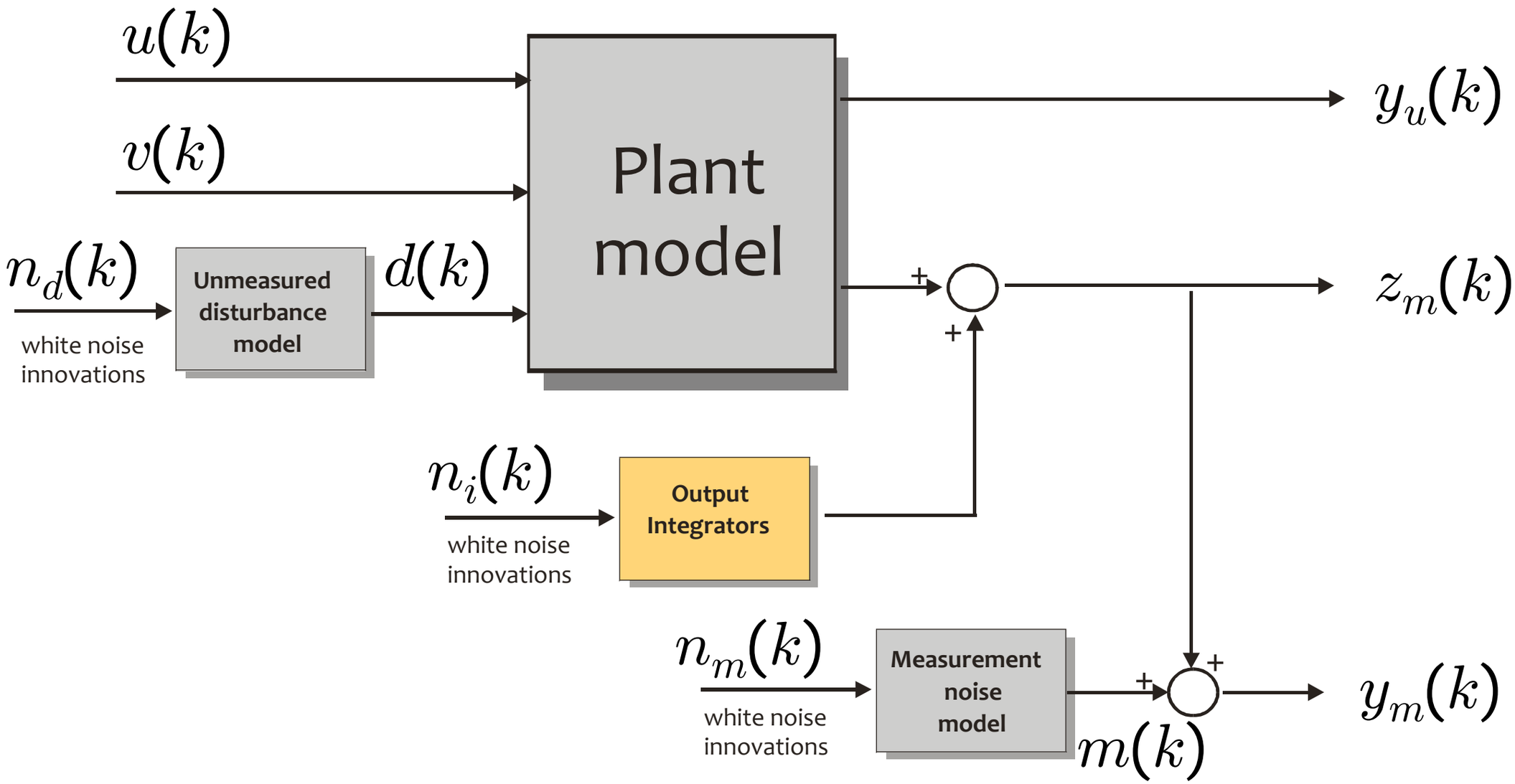
$$y_m(k) = \begin{bmatrix} C_m & D_{dm}\bar{C} & \tilde{C} \end{bmatrix} \begin{bmatrix} x(k) \\ x_d(k) \\ x_m(k) \end{bmatrix} + D_{vm}v(k) + \bar{D}_m n_d(k) + \tilde{D}_m n_m(k)$$

- $n_d(k)$: represents source for modeling errors
- $n_m(k)$: represents source for measurement noise
- $n_u(k)$: white noise on all inputs $u$ added for solvability of the Riccati equation

# Integral Action in MPC

(and not only in MPC)

# Output Integrators



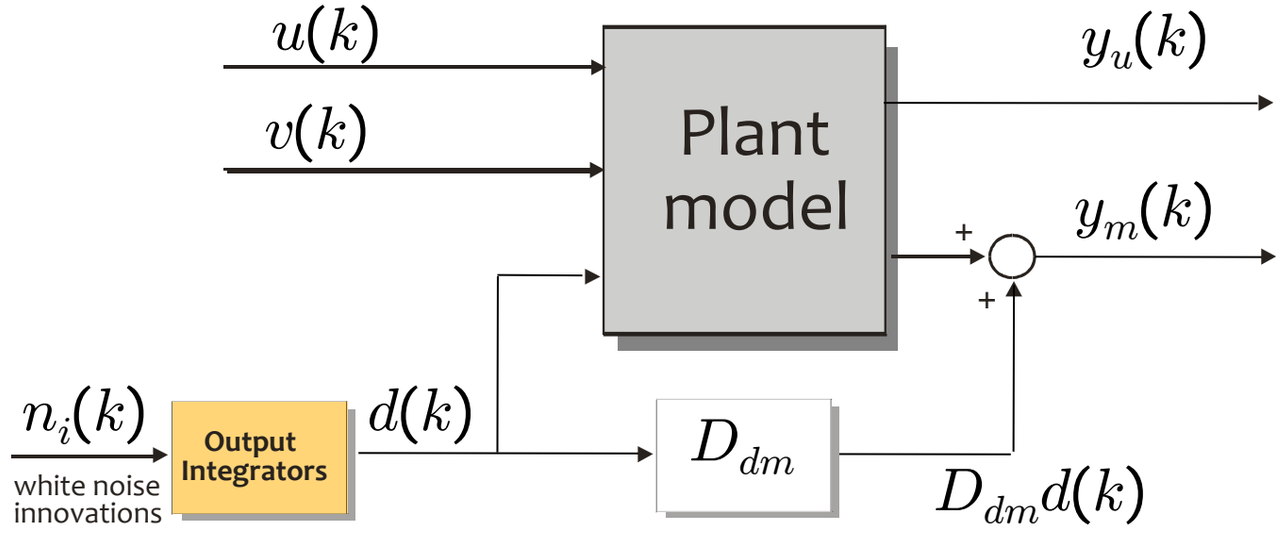

- Introduce output integrators as additional disturbance models
- Under certain conditions, observer + controller provide **zero offset** in steady-state

# Integrators and Steady-State Offsets

- More generally, add integrators on states + outputs:

$$\begin{cases} x(k+1) = Ax(k) + B_u u(k) + B_v v(k) + B_d d(k) \\ d(k+1) = d(k) \\ y_m(k) = C_m x(k) + D_{vm} v(k) + D_{dm} d(k) \end{cases}$$

$u \in \mathbb{R}^m, \quad y_m \in \mathbb{R}^p, \quad x \in \mathbb{R}^n$




- Use the above model + meas.noise model to design an observer (e.g. Kalman filter)

- Main idea: observer makes $y_m - (C_m\hat{x} + D_{dm}\hat{d}) \to 0$ (estimation error)

  MPC makes $C_m\hat{x} + D_{dm}\hat{d} \to r$ (predicted tracking error)

  $\Rightarrow$ the combination makes $y_m \to r$ (actual tracking error)

- Explanation: $D_{dm}\hat{d}$ compensates model mismatch in steady-state

# Error feedback

- **Idea:** add integrals of measured outputs as additional states (similar to linear state-feedback case) (Kwakernaak, 1972)

- Extended prediction model:

$$\begin{bmatrix} x(k+1) \\ q(k+1) \\ r(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 & 0 \\ C & I & -I \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ q(k) \\ r(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} C & 0 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ q(k) \\ r(k) \end{bmatrix}$$

- Implementation:

$$q(k+1) = q(k) + [y(k) - r(k)]$$

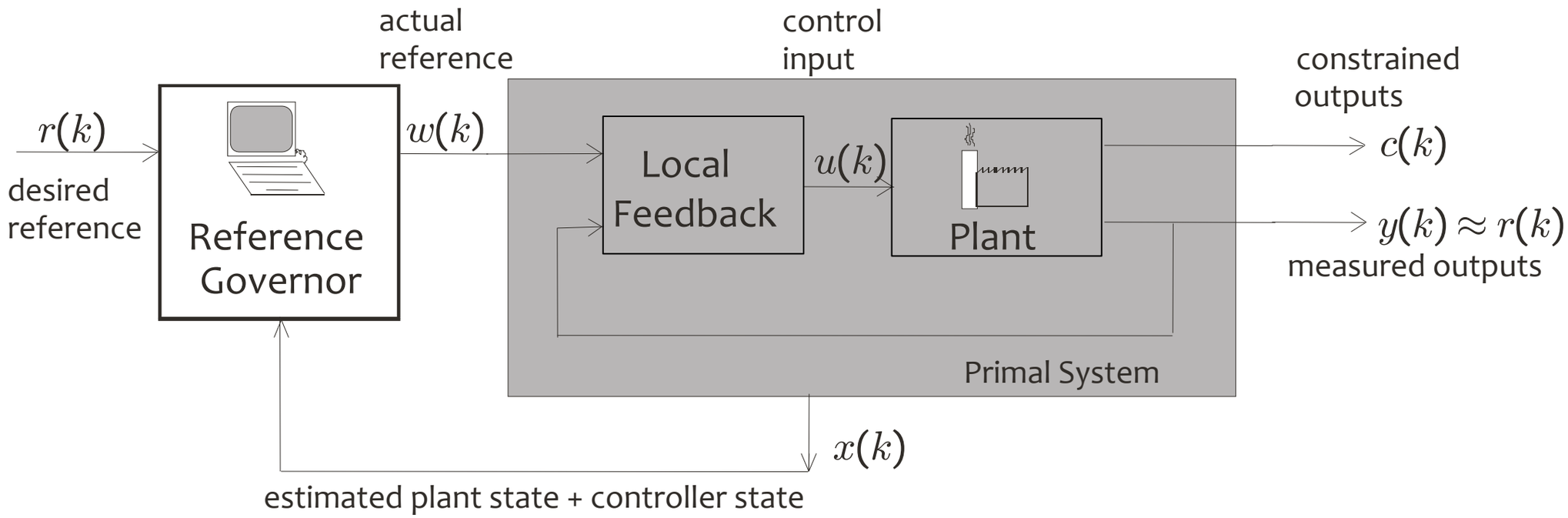
$$u(k) = f_{\text{MPC}}\left( \begin{bmatrix} x(k) \\ q(k) \\ r(k) \end{bmatrix} \right)$$

- Explanation: if closed-loop asymptotically stable $\Rightarrow q(k) \rightarrow$ cost. and hence $y(k) \rightarrow r(k)$

# Reference Governor

- Idea: Apply MPC as the set-point generator to linear feedback loops (e.g.: PID)

(Bemporad, 1997)




- Separation of problems:

  - Local feedback designed for stability, disturbance attenuation, good tracking, *without taking care of constraints*

  - Actual reference $w(t)$ generated on-line by an MPC algorithm *to take care of constraints*

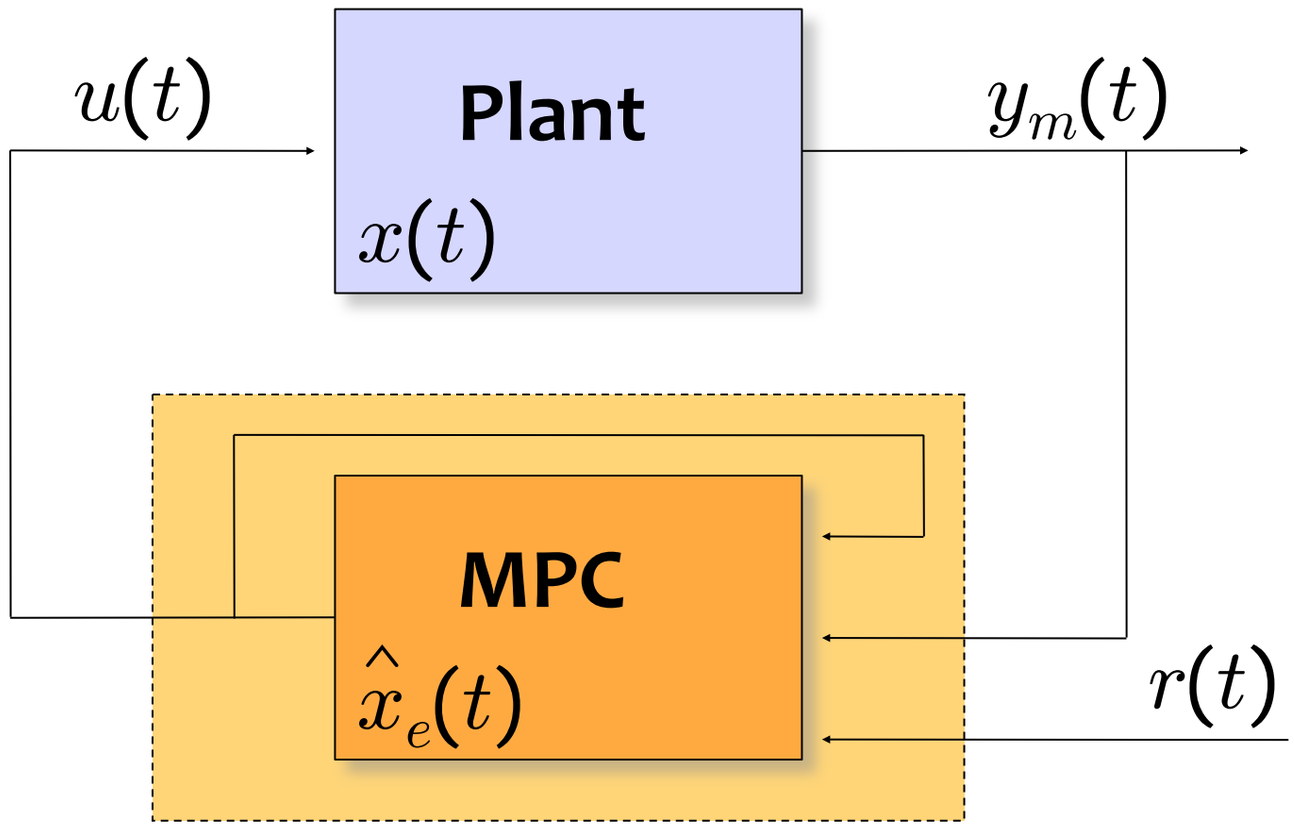
Objective:

$$\begin{array}{ll} \min\limits_{w} & \|w - r\| \\ \text{s.t.} & \text{constraints} + \text{closed-loop dynamics} \end{array}$$

# MPC frequency analysis (inactive constraints)

- Unconstrained MPC gain + linear observer = linear dynamical system (= 2 d.o.f. dynamic controller)

- Closed-loop MPC analysis can be performed using standard frequency-domain tools (e.g. Bode plots for sensitivity analysis)




In MPC Tbx: `ss(mpc)` or `tf(mpc)` return the LTI discrete-time form of the linearized (=no constraints) MPC object

# Controller matching problem

(Di Cairano, Bemporad, ECC'09)

- Given the controller $u=Kx$, **find weights** $Q,R,P$ for the MPC problem such that

$$-\left[\begin{array}{cccc} I & 0 & \dots & 0 \end{array}\right] H^{-1}F = K_{fv}$$

that is, the **MPC controller coincides with** $K_{fv}$ when the constraints are **not active**

- QP matrices: $H = (\mathcal{R} + \mathcal{S}'\mathcal{Q}\mathcal{S}),\ \ F = \mathcal{T}'\mathcal{Q}\mathcal{S}$

$$\begin{array}{ll} \min\limits_{U} & \frac{1}{2}U'HU + x'(t)F'U + \frac{1}{2}x(t)Yx(t) \\ \text{subj. to} & GU \leq W + Sx(t), \end{array}$$

$$\mathcal{S} = \begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix} \quad \mathcal{Q} = \begin{bmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & Q & 0 \\ 0 & 0 & \dots & 0 & P \end{bmatrix}$$

$$\mathcal{T} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} \qquad \mathcal{R} = \begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & R \end{bmatrix}$$

# Controller matching problem - Example

- Open-loop process:

$$y(k) = 1.8y(k-1) + 1.2y(k-2) + u(k-1)$$

- Constraints: $-24 \le u(k) \le 24$

- Desired controller (**PID**): $K_I = 0.248,\ K_P = 0.752,\ K_D = 2.237$

$$u(k) = -\left(K_I \mathcal{I}(k) + K_P y(k) + \frac{K_D}{T_s}(y(k) - y(k-1))\right)$$
$$\mathcal{I}(k) = \mathcal{I}(k-1) + T_s y(k)$$

- State-space form:

$$x(k) = \begin{bmatrix} y(k-1) \\ y(k-2) \\ \mathcal{I}(k-1) \\ u(k-1) \end{bmatrix}$$

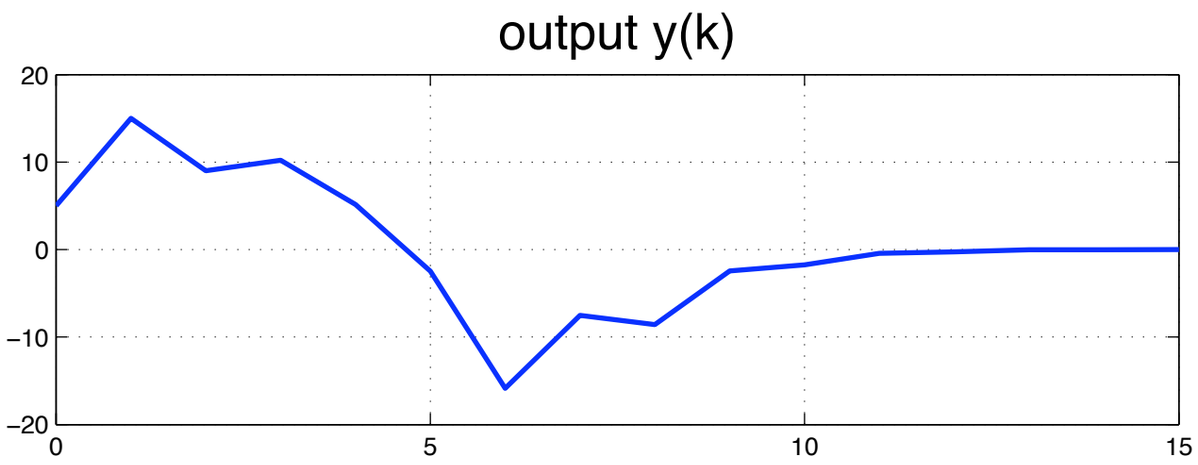
controller matching based on inverse LQR

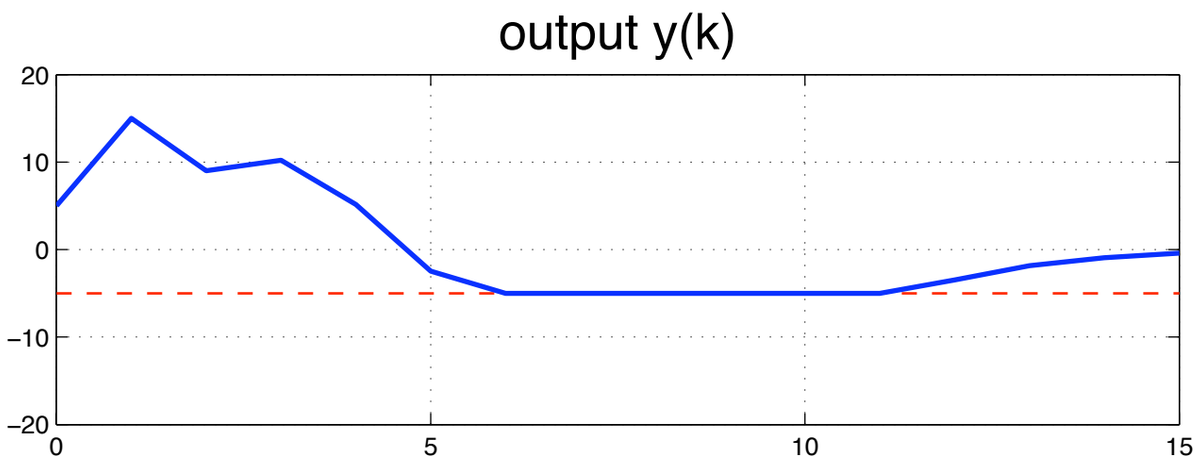
$$Q^* = \begin{bmatrix} 6.401 & 0.064 & -0.001 & 0.020 \\ 0.064 & 6.605 & 0.006 & 0.080 \\ -0.001 & 0.006 & 6.647 & -0.020 \\ 0.019 & 0.080 & -0.020 & 6.378 \end{bmatrix}$$

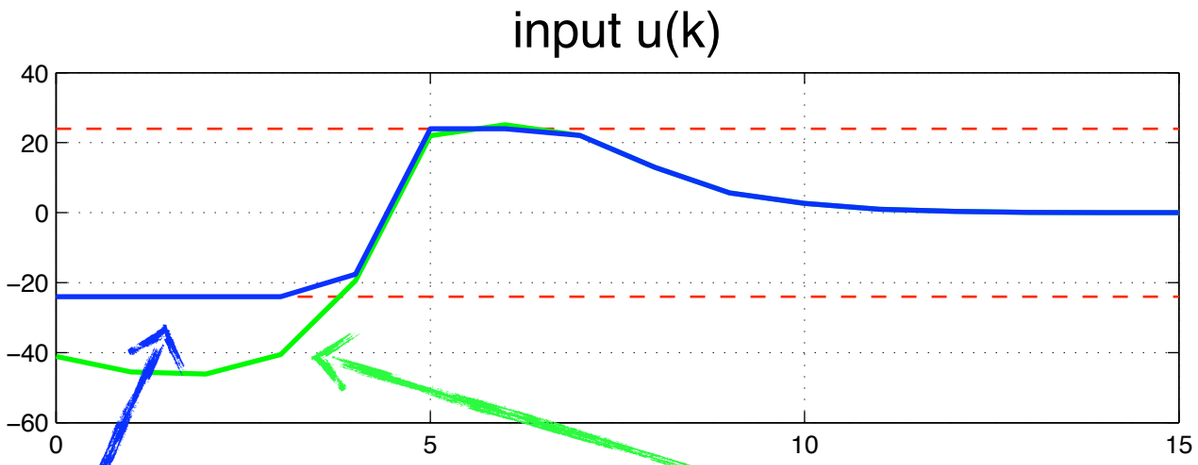
$$R^* = 1$$

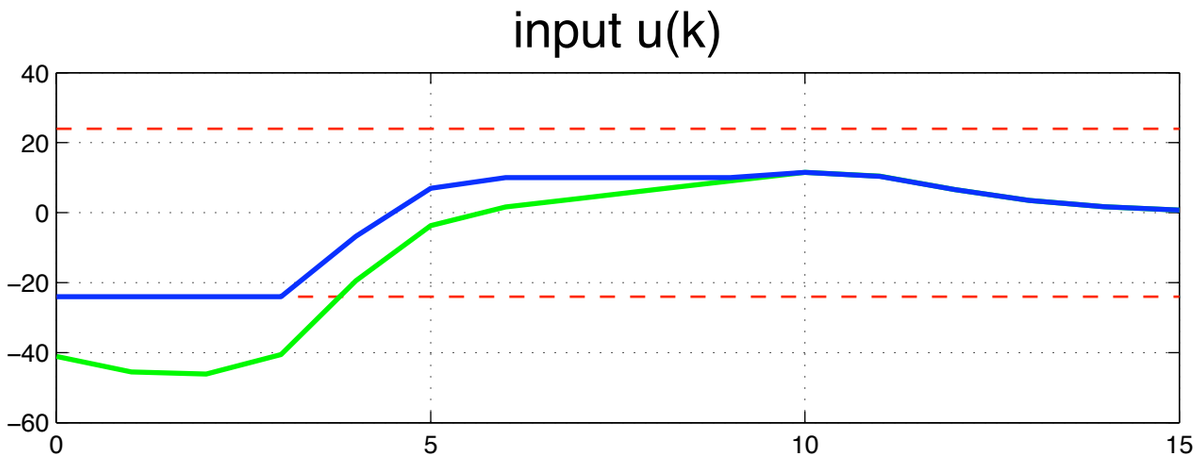
$$P^* = \begin{bmatrix} 422.7 & 241.7 & 50.39 & 201.4 \\ 241.7 & 151.0 & 32.13 & 120.4 \\ 50.39 & 32.13 & 19.85 & 26.75 \\ 201.4 & 120.4 & 26.75 & 106.6 \end{bmatrix}$$

# Controller matching problem - Example



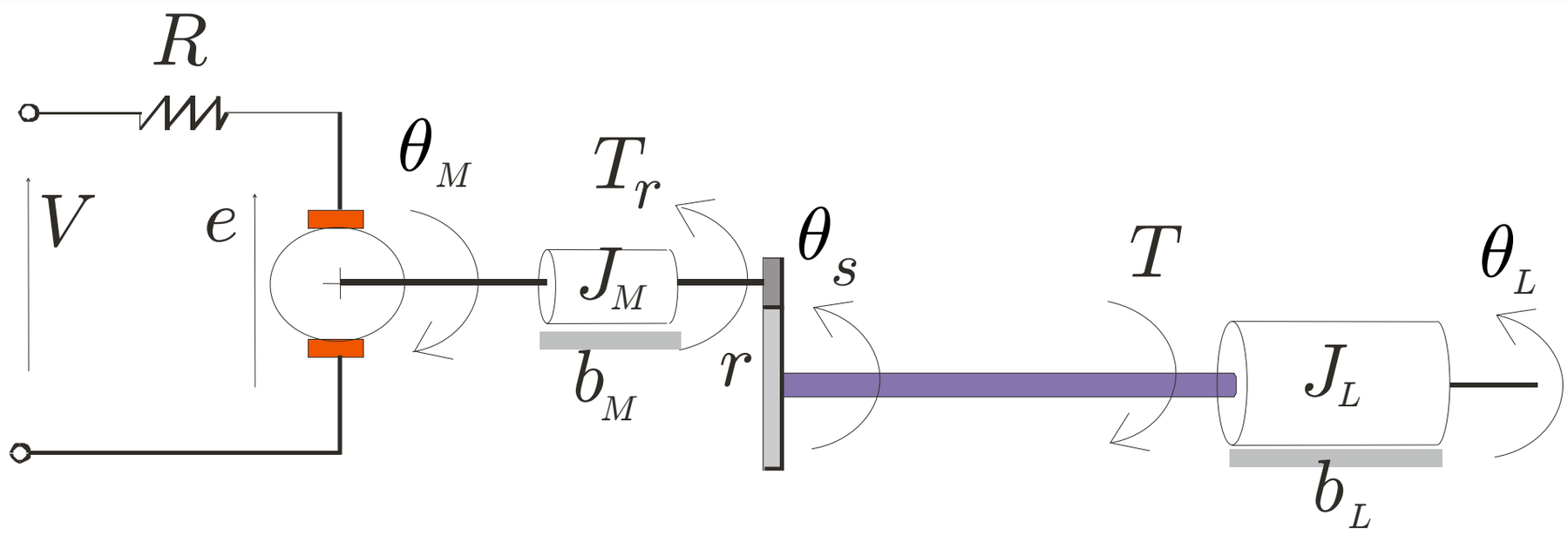




add output constraint $y(k) \geq -5$

Note: it's not trivially a saturation of PID controller. In this case sat(PID) leads to instability

# Example: MPC of a DC Servomotor

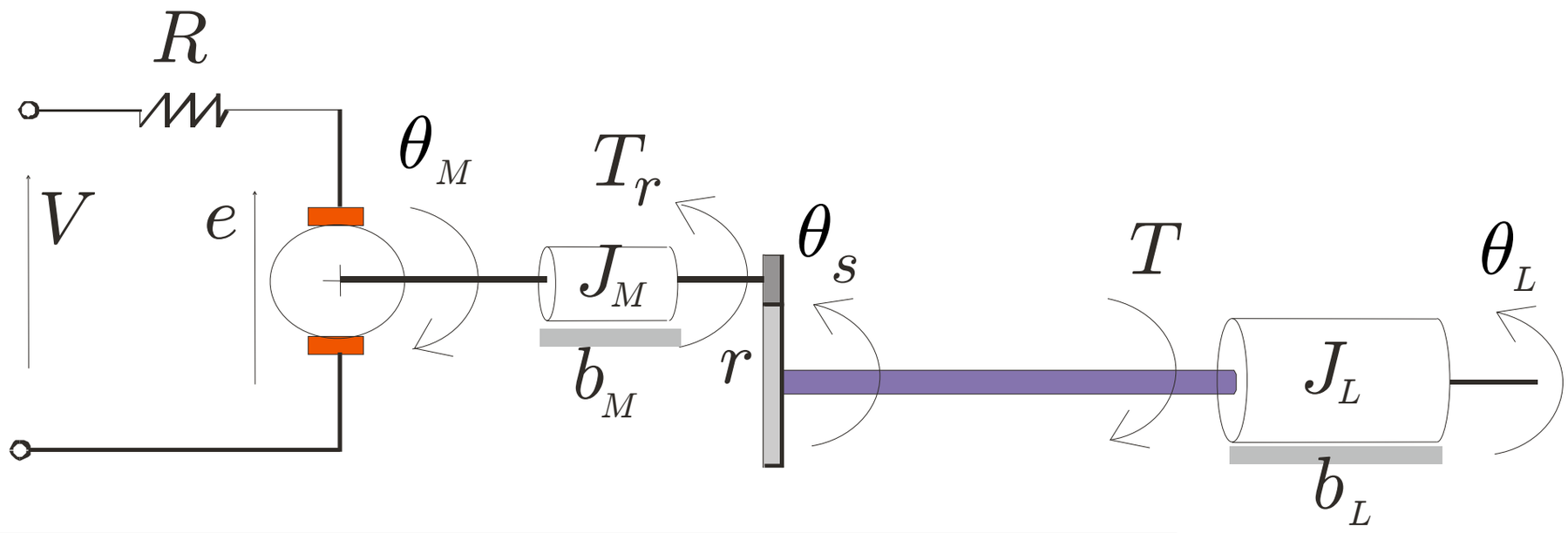



| Symbol | Value (MKS) | Meaning |
|---|---|---|
| $L_S$ | 1.0 | shaft length |
| $d_S$ | 0.02 | shaft diameter |
| $J_S$ | negligible | shaft inertia |
| $J_M$ | 0.5 | motor inertia |
| $\beta_M$ | 0.1 | motor viscous friction coefficient |
| $R$ | 20 | resistance of armature |
| $K_T$ | 10 | motor constant |
| $\rho$ | 20 | gear ratio |
| $k_\theta$ | 1280.2 | torsional rigidity |
| $\hat{J}_L$ | $20J_M$ | nominal load inertia |
| $\beta_L$ | 25 | load viscous friction coefficient |

go to demo **`/demos/linear/dcmotor.m`** (Hyb-Tbx)

**`mpcmotor.m`** (MPC-Tbx)

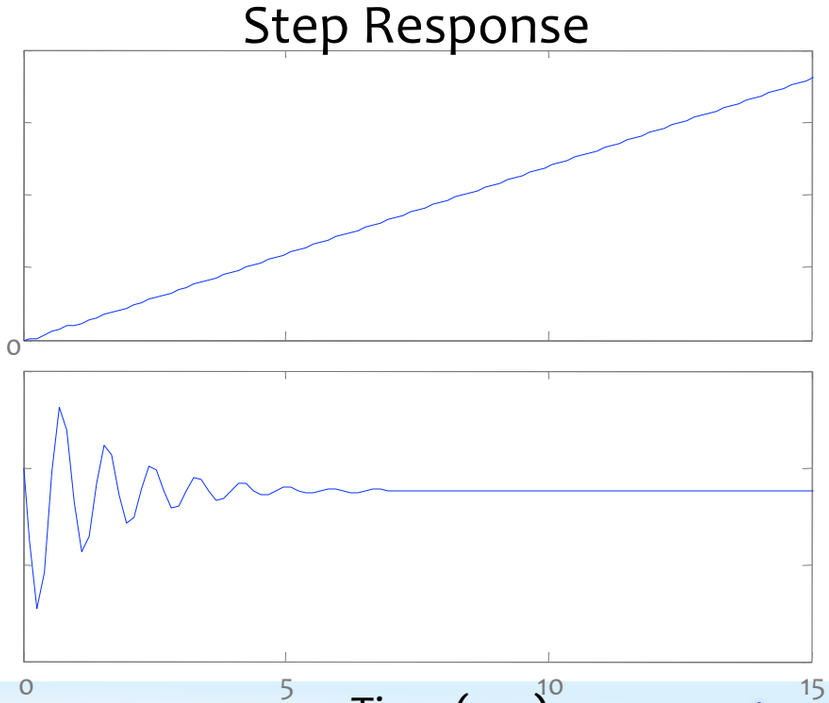
# DC Servomotor - Model



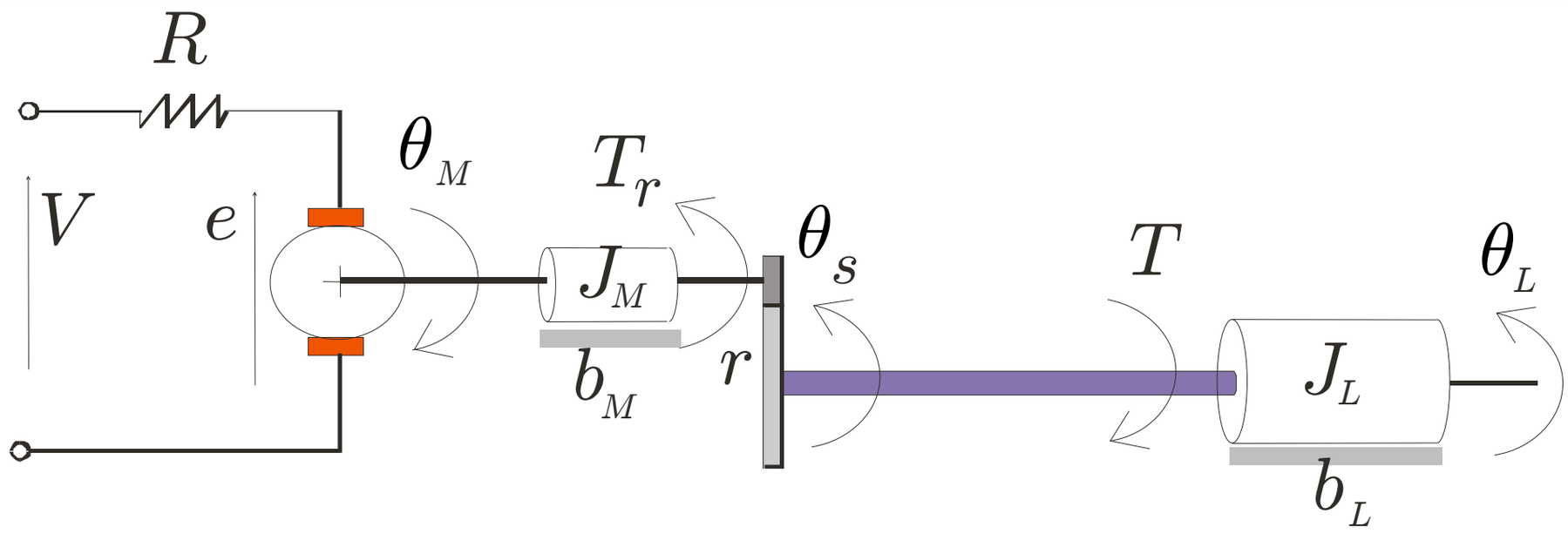

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_\theta}{J_L} & -\frac{\beta_L}{J_L} & \frac{k_\theta}{\rho J_L} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k_\theta}{\rho J_M} & 0 & -\frac{k_\theta}{\rho^2 J_M} & -\frac{\beta_M + k_T^2/R}{J_M} \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{k_T}{R J_M} \end{bmatrix} V$$

$$\theta_L = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} x$$

$$T = \begin{bmatrix} k_\theta & 0 & -\frac{k_\theta}{\rho} & 0 \end{bmatrix} x$$



# DC Servomotor: Specs




- Finite shear strength of steel shaft ($\tau_{adm} = 50$ N/mm$^2$)
  $\Rightarrow |T| \leq 78.5398\ Nm$
- DC voltage limits $|V| \leq 220\ V$
- Sampling time: $T_s = .1$ s (+ zero-order hold)

# DC Servomotor: Exercise

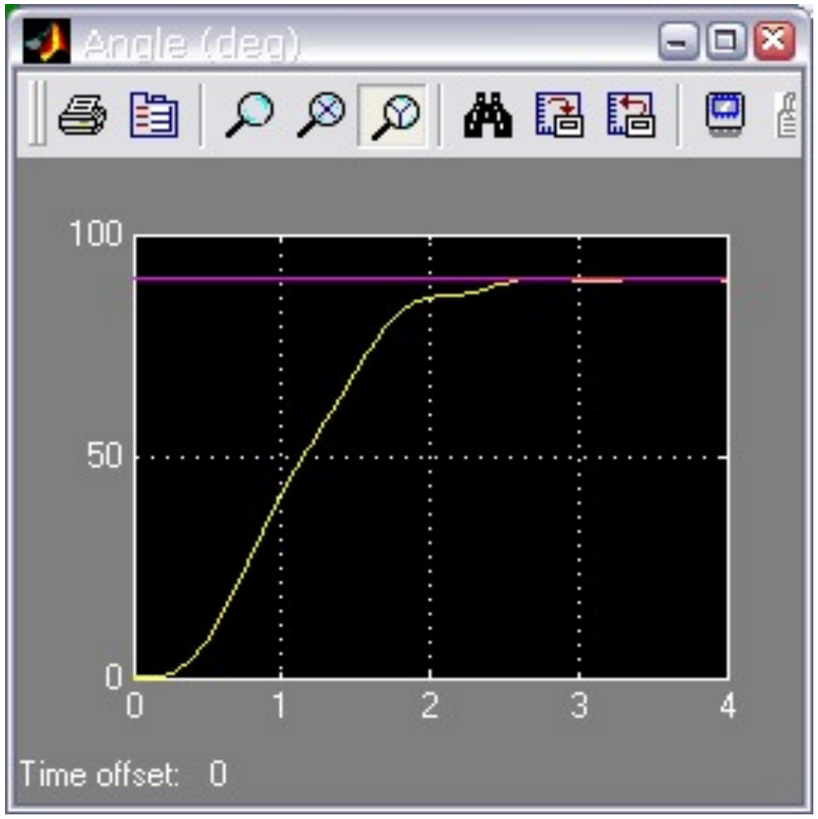
Design a model predictive controller for tracking an angle reference step signal $r$. Assume that only the position $\theta_L$ is measured.

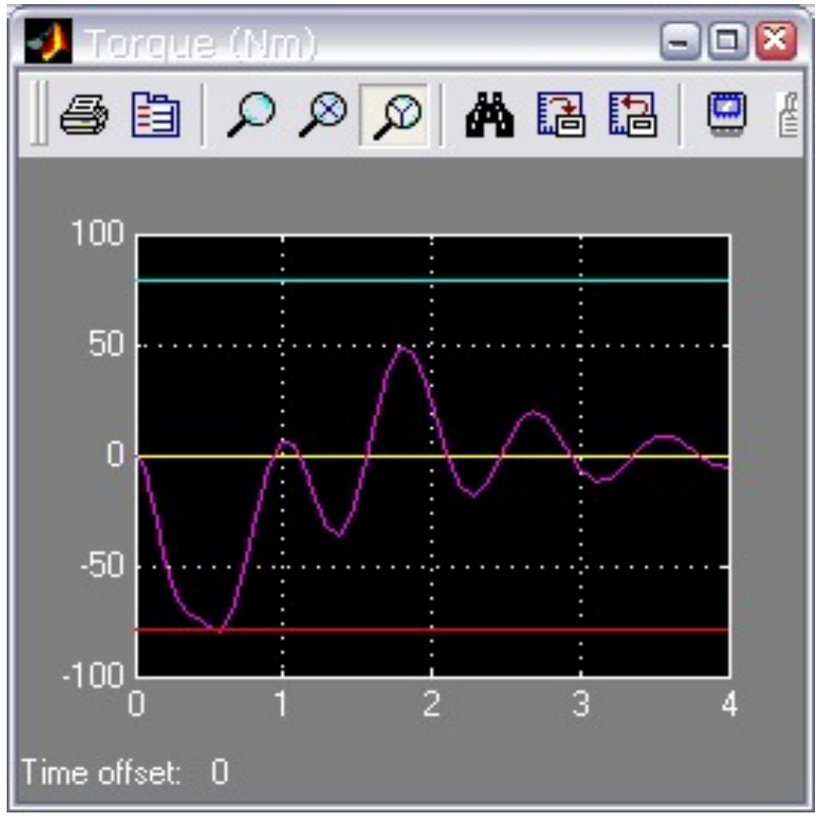
- Try unconstrained MPC (no voltage and torque constraints) for tracking $r = \frac{\pi}{2}$ rad. Tune the controller to fulfill voltage and torque constraints. What happens if you change the reference signal to $r = \pi$ rad ?

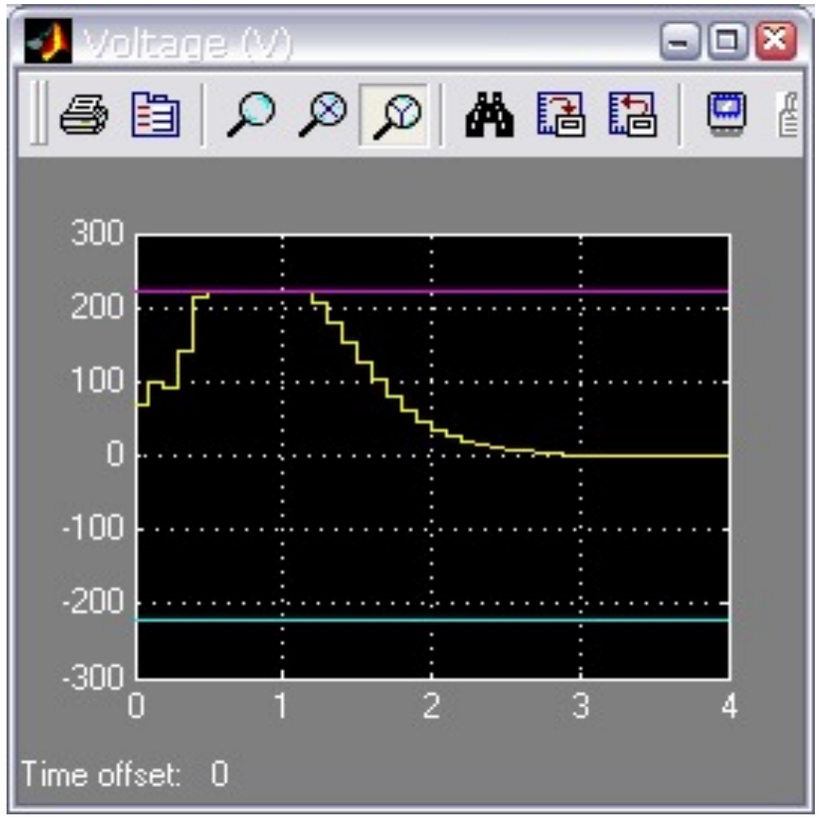
- Tune a constrained MPC for $r = \frac{\pi}{2}$ rad. What happens if you change the reference signal to $r = \pi$ rad ? And $r(t) = 2\pi \sin(0.4t)$ rad ?

- Try different numbers of control moves $M$ and prediction horizons $P$. What happens for increasing $M$ ? For increasing $P$ ? What happens if $P/M$ are too small ? Look for the smallest $M$ which provides a reasonably good performance.

# DC Servomotor: MPC



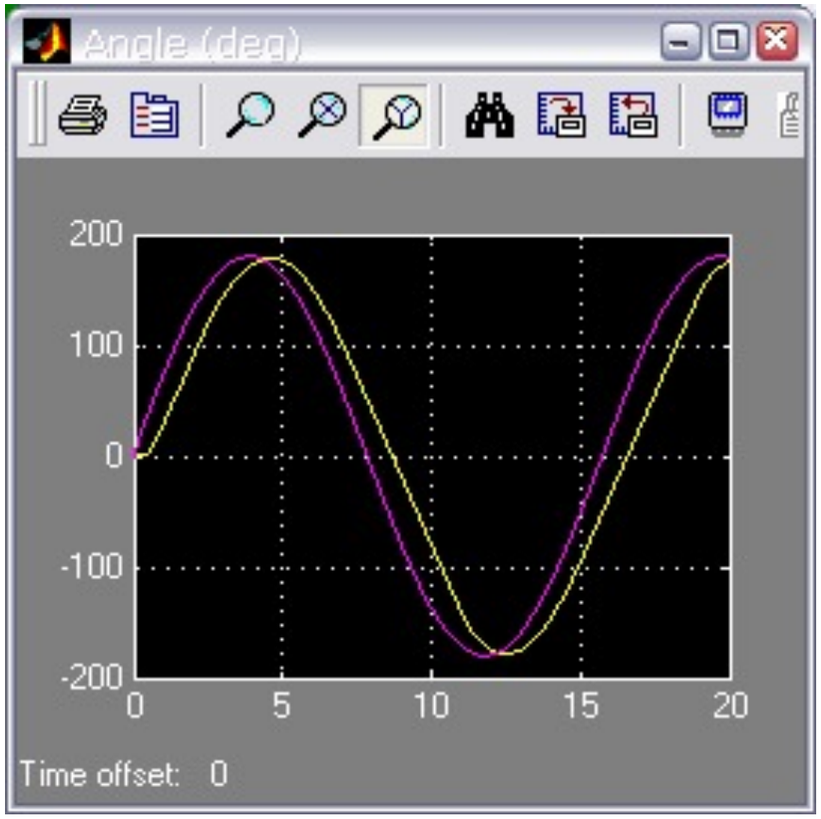



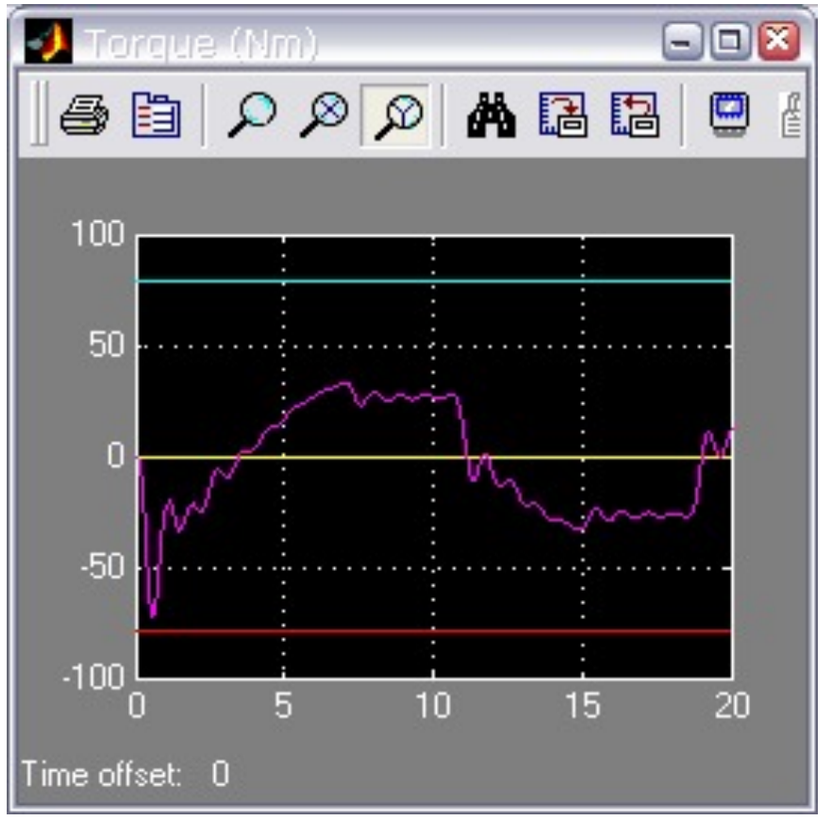



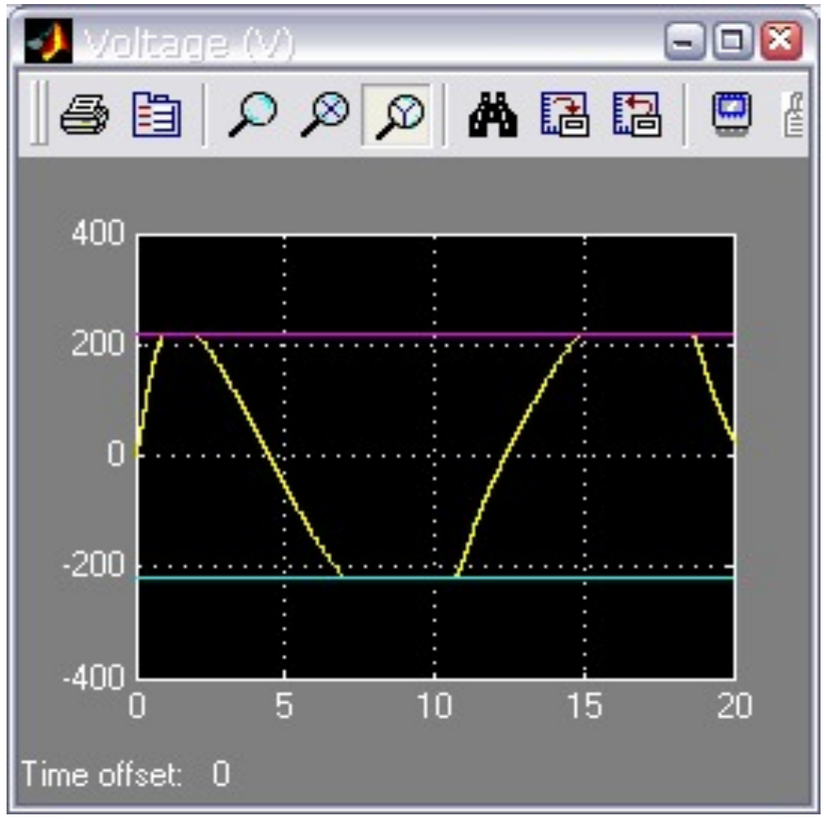

$r(t) \equiv 90$ deg

$$N = 10 \qquad N_u = 3$$
$$w_y = \{10, 0\} \qquad w_{\delta u} = .05 \qquad w_u = 0$$
$$u_{\min} = -220 \text{ V} \qquad u_{\max} = 220 \text{ V}$$
$$y_{\min} = \{-\infty, -78.5398\} \text{ Nm} \qquad y_{\max} = \{\infty, 78.5398\} \text{ Nm}$$

# DC Servomotor: MPC




$$r(t) = 180\sin(0.4t) \text{ deg}$$

$$N = 10 \qquad N_u = 3$$

$$w_y = \{10, 0\} \qquad w_{\delta u} = .05 \qquad w_u = 0$$

$$u_{\min} = -220 \text{ V} \qquad u_{\max} = 220 \text{ V}$$

$$y_{\min} = \{-\infty, -78.5398\} \text{ Nm} \qquad y_{\max} = \{\infty, 78.5398\} \text{ Nm}$$