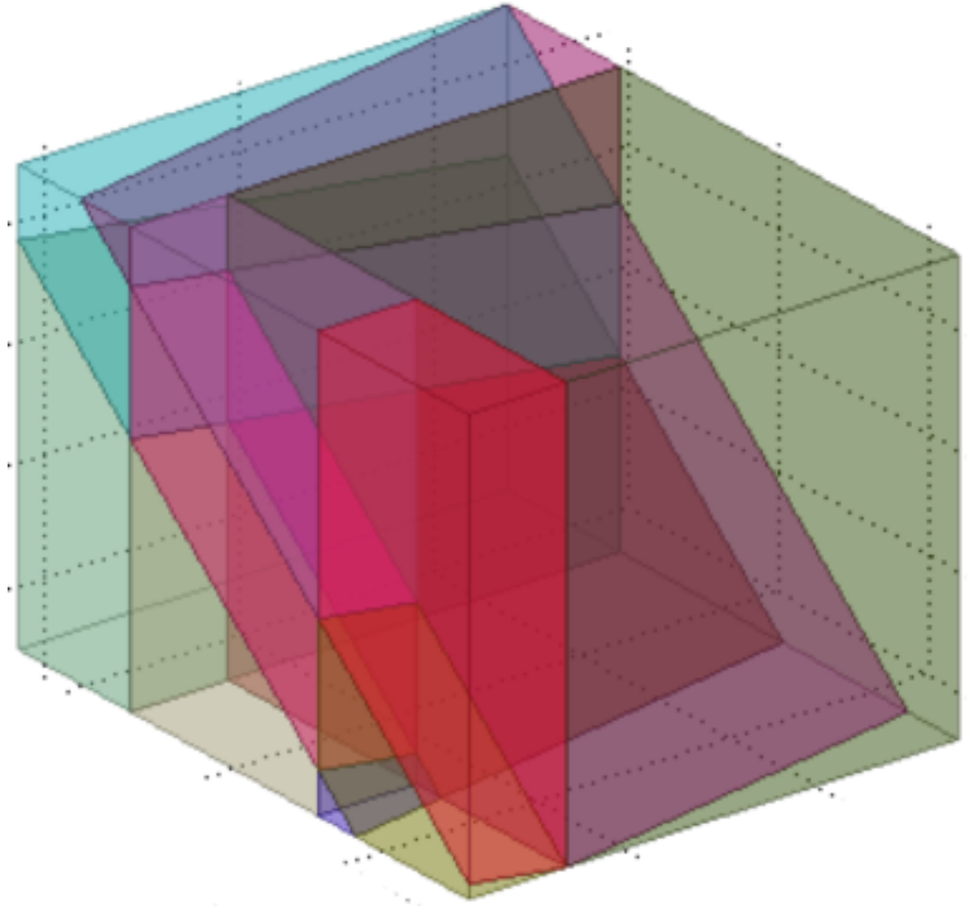
MODEL PREDICTIVE CONTROL

Alberto Bemporad

academic year 2017/18



IMT SCHOOL FOR ADVANCED STUDIES LUCCA

http://cse.lab.imtlucca.it/~bemporad

COURSE STRUCTURE

- General concepts of model predictive control (MPC)
- Linear MPC and extensions to time-varying and nonlinear MPC
- MPC computations: quadratic programming (QP), explicit MPC
- Hybrid MPC
- Stochastic MPC

- MATLAB Toolboxes:

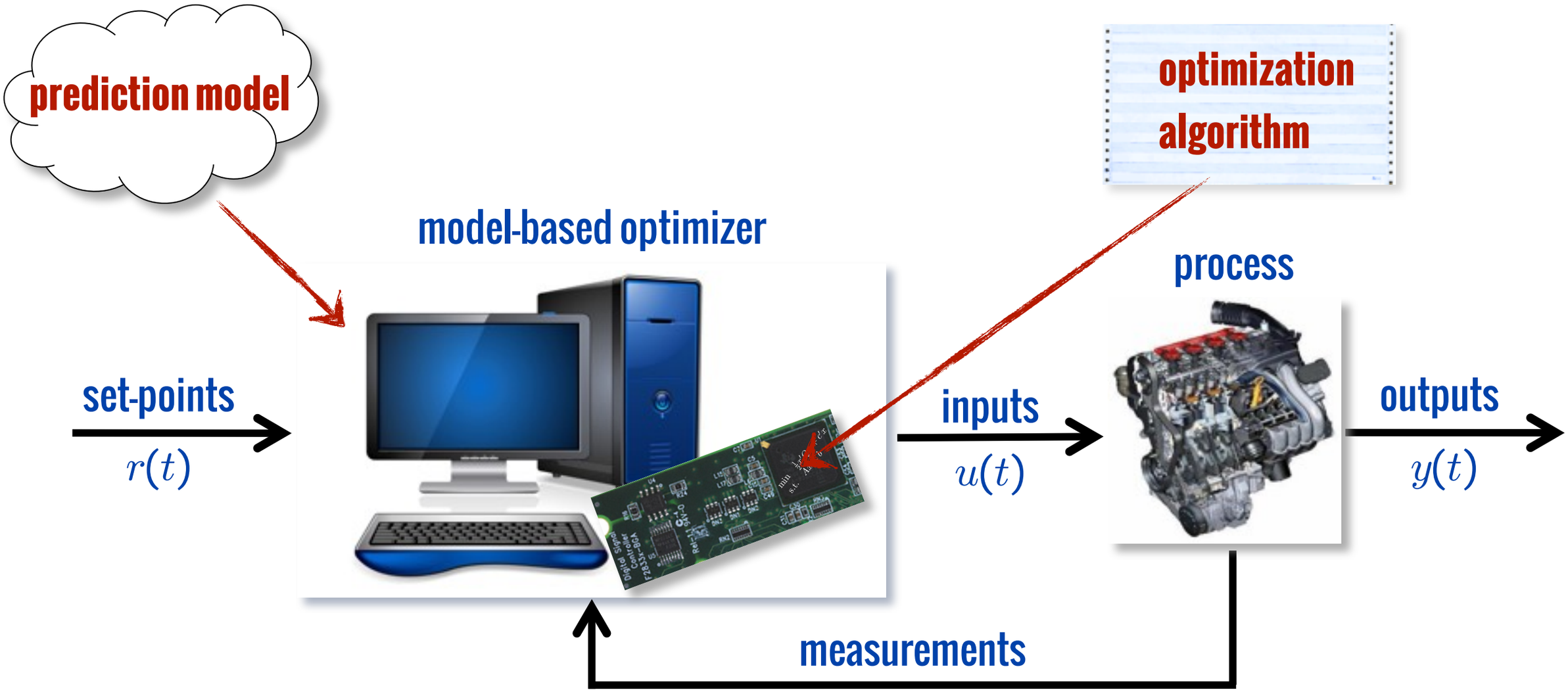
  MPC Toolbox (linear/explicit/parameter-varying MPC)
  Hybrid Toolbox (explicit MPC, hybrid systems)

- Course page:

  http://cse.lab.imtlucca.it/~bemporad/mpc_course.html

MODEL PREDICTIVE CONTROL: BASIC CONCEPTS

MODEL PREDICTIVE CONTROL (MPC)



simplified — likely

Use a dynamical model of the process to predict its future evolution and choose the ~~"best"~~ a good control action

MODEL PREDICTIVE CONTROL (MPC)

- At each time t, find the best control sequence over a future horizon of N steps

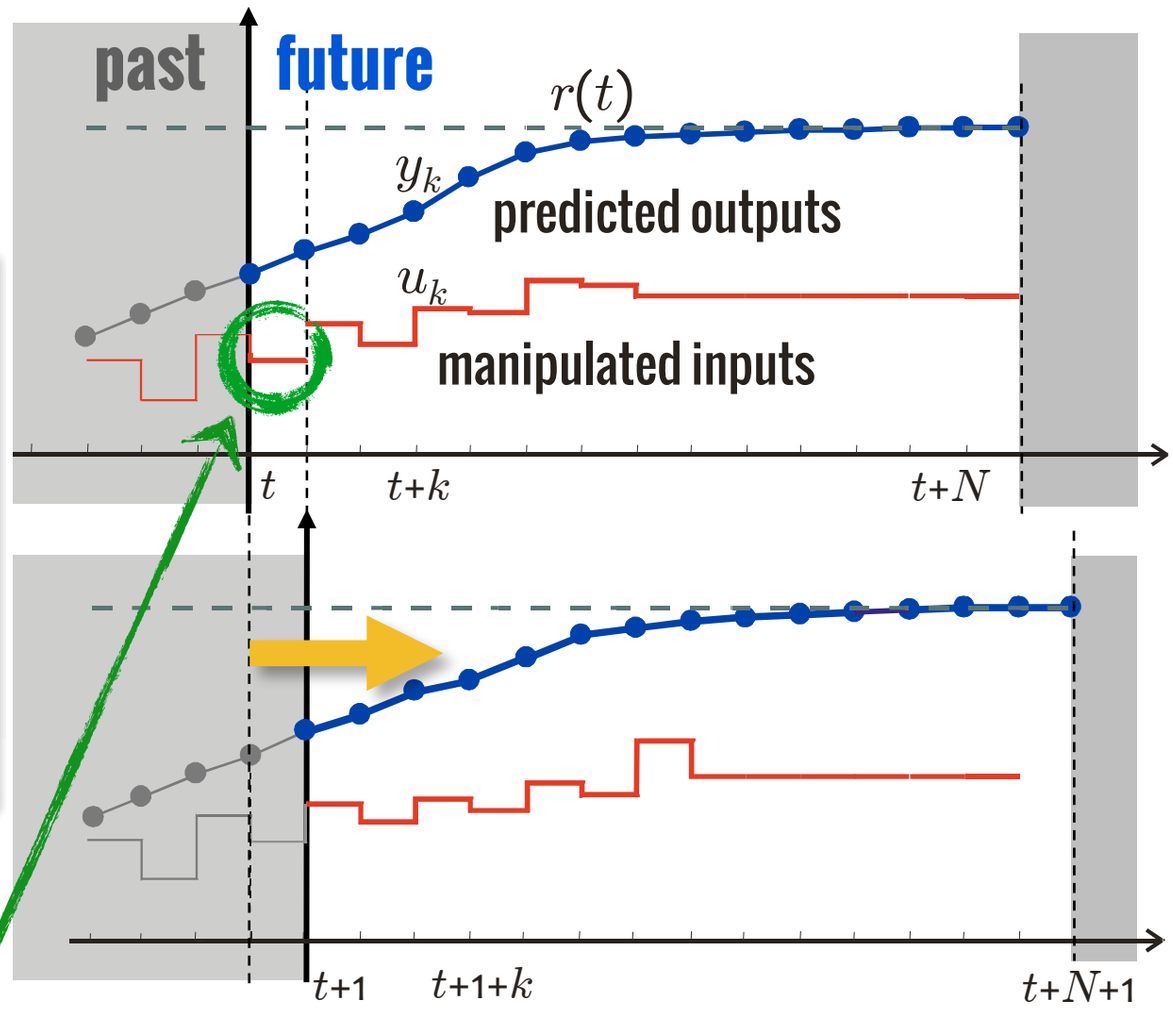
penalty on tracking error

penalty on actuation

$$\min \sum_{k=0}^{N-1} \|W^y(y_k - r(t))\|^2 + \|W^u(u_k - u^{\text{ref}}(t))\|^2$$

s.t.
$$x_{k+1} = f(x_k, u_k, t)$$
$$y_k = g(x_k, u_k, t)$$
constraints on u_k, y_k
$$x_0 = x(t)$$ ← feedback !

optimization problem

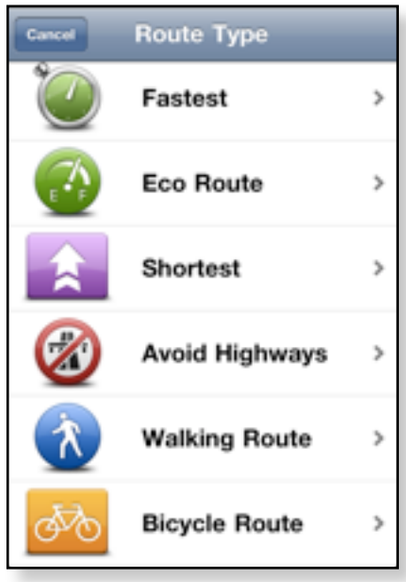


- Problem solved w.r.t. $\{u_0,\ldots,u_{N-1}\}$
- Apply the first optimal move $u(t)=u_0^*$, throw the rest of the sequence away
- At time $t+1$: Get new measurements, repeat the optimization. And so on …

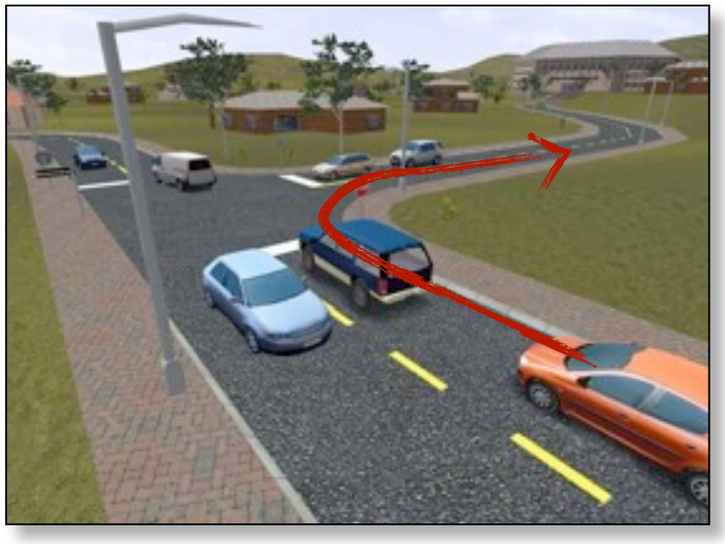
DAILY-LIFE EXAMPLES OF MPC

- MPC is like **playing chess** !

- On-line (event-based) re-planning used in GPS navigation

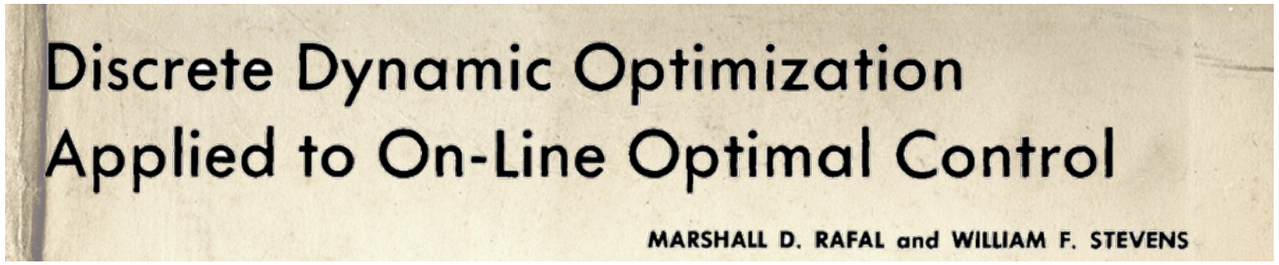


- You use MPC too when you drive !



MPC IN INDUSTRY

- The MPC concept for process control dates back to the 60's


Discrete Dynamic Optimization
Applied to On-Line Optimal Control

MARSHALL D. RAFAL and WILLIAM F. STEVENS

(Rafal, Stevens, *AiChE Journal*, 1968)

- MPC used in the process industries since the 80's


©SimulateLive.com

MPC is the standard for advanced control in the process industry.

(Qin, Badgewell, 2003) (Bauer & Craig, 2008)

- Research in MPC is still very active !

MPC IN INDUSTRY

(Qin, Badgewell, 2003)

- Industrial survey of MPC applications conducted in mid 1999

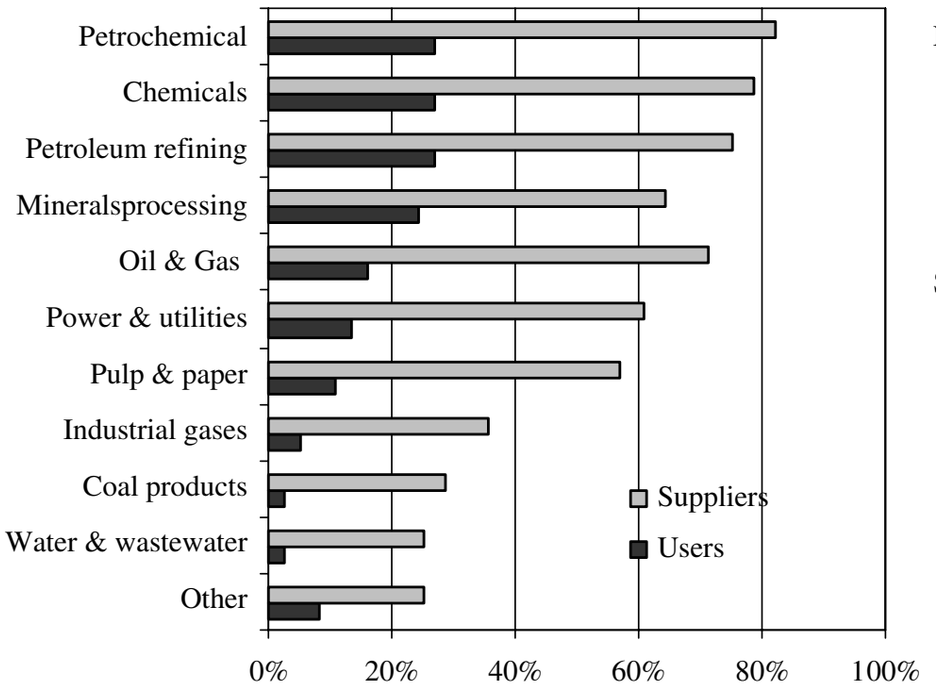
Area	Aspen Technology	Honeywell Hi-Spec	Adersa[b]	Invensys	SGS[c]	Total
Refining	1200	480	280	25		1985
Petrochemicals	450	80	—	20		550
Chemicals	100	20	3	21		144
Pulp and paper	18	50	—	—		68
Air & Gas	—	10	—	—		10
Utility	—	10	—	4		14
Mining/Metallurgy	8	6	7	16		37
Food Processing	—	—	41	10		51
Polymer	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 IDCOM-M:1987 OPC:1987	PCT:1984 RMPCT:1991	IDCOM:1973 HIECON:1986	1984	1985	
Largest App.	603 × 283	225 × 85	—	31 × 12	—	

Estimates based on vendor survey

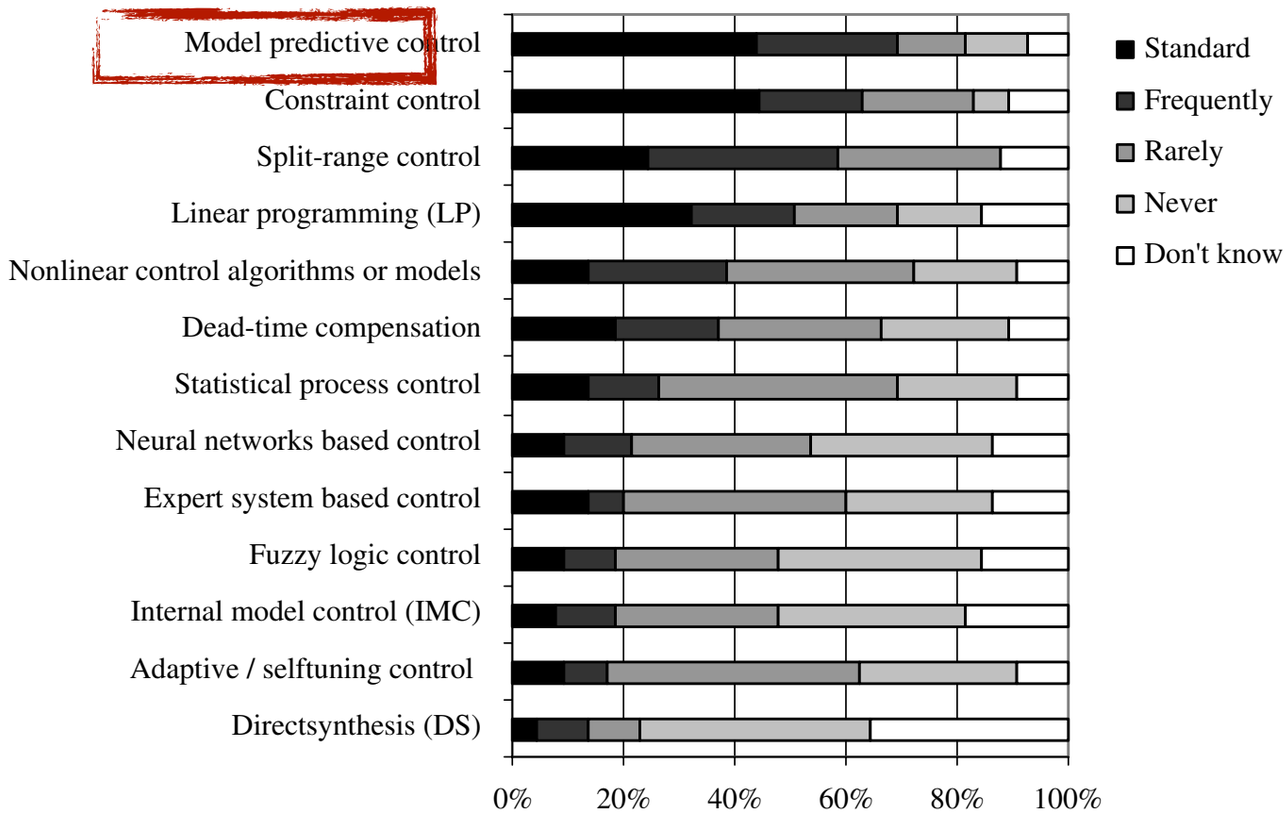
MPC IN INDUSTRY

(Bauer & Craig, 2008)

- Economic assessment of Advanced Process Control (APC)



participants of APC survey by industry (worldwide)



Industrial use of APC methods: survey results

MPC IN INDUSTRY

(Samad, *IEEE CS Magazine*, Feb 2017)

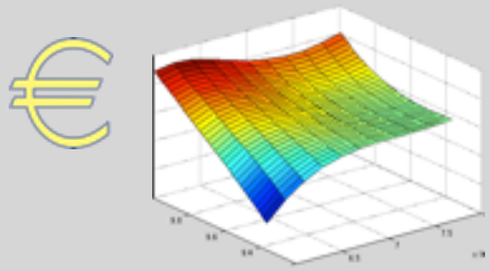
- Impact of advanced control technologies in industry

TABLE 1 A list of the survey results in order of industry impact as perceived by the committee members.

Rank and Technology	High-Impact Ratings	Low- or No-Impact Ratings
PID control	100%	0%
Model predictive control	78%	9%
System identification	61%	9%
Process data analytics	61%	17%
Soft sensing	52%	22%
Fault detection and identification	50%	18%
Decentralized and/or coordinated control	48%	30%
Intelligent control	35%	30%
Discrete-event systems	23%	32%
Nonlinear control	22%	35%
Adaptive control	17%	43%
Robust control	13%	43%
Hybrid dynamical systems	13%	43%

TYPICAL USE OF MPC

static optimizer

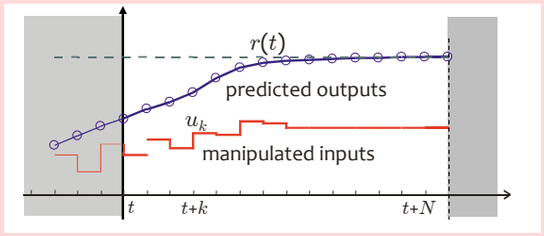


Steady-state optimization

- (economic) set-point optimization
- static I/O model

output set points and
corresponding optimal actuator set points

dynamic optimizer



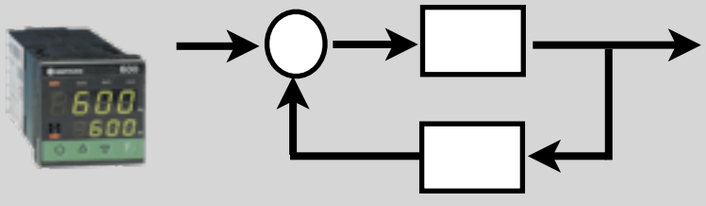
MPC

- coordinate multiple inputs
- performance optim.
- constraints handling

measurements

set points to actuators

regulators



low-level controllers

- fast-sampling
- single-loop

AUTOMOTIVE APPLICATIONS OF MPC

Bemporad, Bernardini, Borrelli, Cimini, Di Cairano, Esen, Giorgetti, Graf-Plessen, Hrovat, Kolmanovsky, Levijoki, Ripaccioli, Trimboli, Tseng, Yanakiev, ... (2001-present)

Powertrain

- direct-inj. engine control
- A/F ratio control
- magnetic actuators
- robotized gearbox
- power MGT in HEVs
- cabin heat control in HEVs
- electrical motors

Vehicle dynamics

- traction control
- active steering
- semiactive suspensions
- autonomous driving

Ford Motor Company

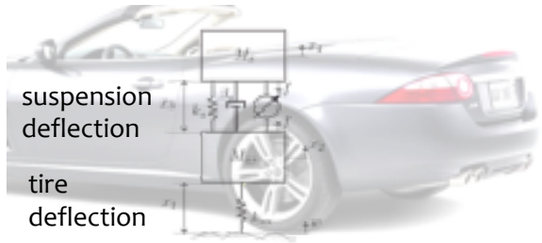
Jaguar

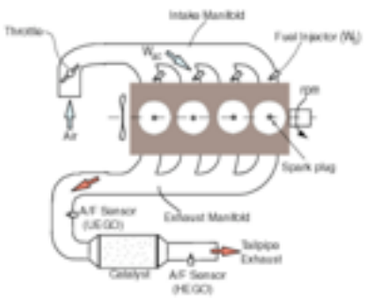
DENSO Automotive

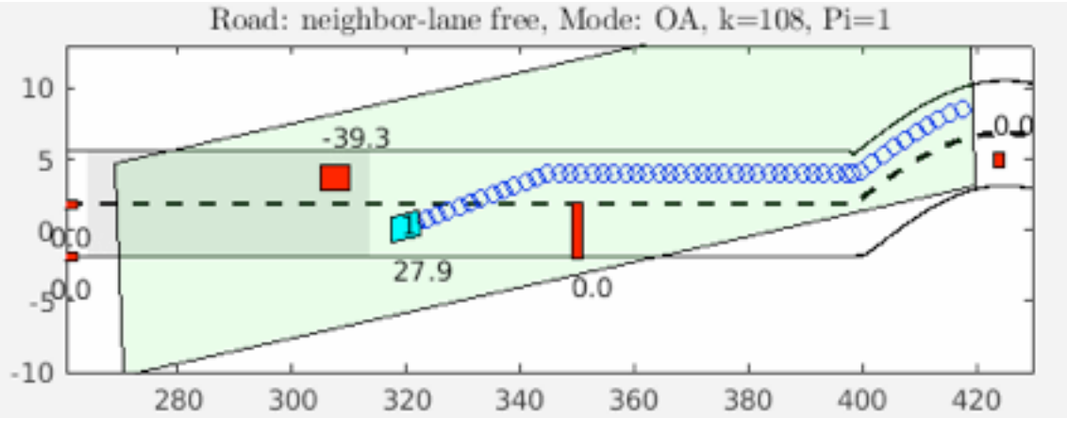
FIAT

General Motors

ODYS
Advanced Controls & Optimization







Most automotive OEMs are adopting MPC solutions today

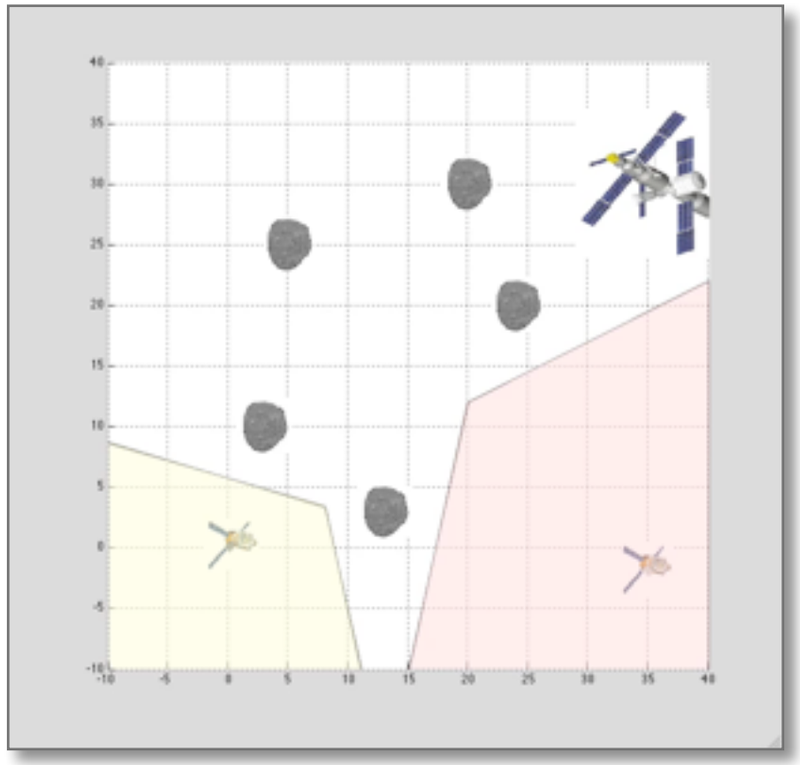
AEROSPACE APPLICATIONS OF MPC

- MPC capabilities explored in new space applications

powered descent

(Pascucci, Bennani, Bemporad, 2016)

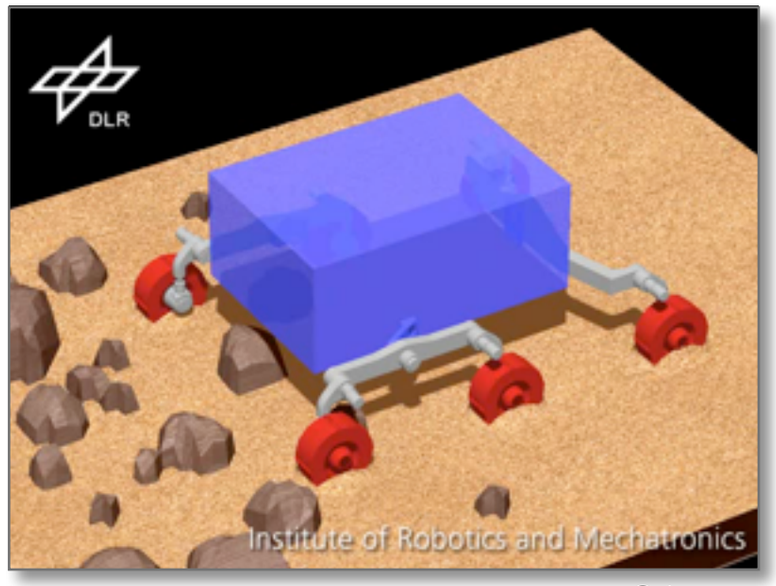
cooperating UAVs



(Bemporad, Rocchi, 2011)

planetary rover

(Krenn et. al., 2012)



MPC IN AERONAUTIC INDUSTRY

PRESS RELEASE

Pratt & Whitney's F135 Advanced Multi-Variable Control Team Receives UTC's Prestigious George Mead Award for Outstanding Engineering Accomplishment

EAST HARTFORD, CONN., THURSDAY, MAY 27, 2010

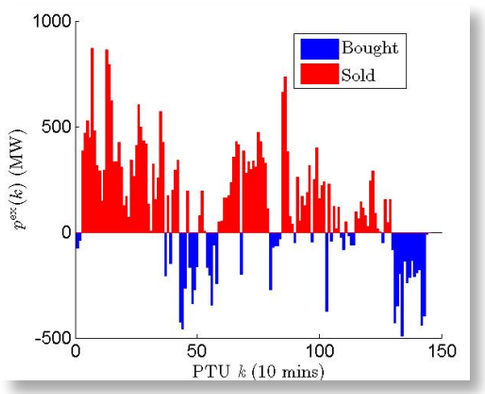
Pratt & Whitney engineers Louis Celiberti, Timothy Crowley, James Fuller and Cary Powell won the George Mead Award – United Technologies Corp.'s highest award for outstanding engineering achievement – for their pioneering work in developing the world's first advanced multi-variable control (AMVC) design for the only engine that powers the F-35 Lightning II flight test program. Pratt & Whitney is a United Technologies Corp. (NYSE:UTX) company.

The AMVC, which uses a proprietary model predictive control methodology, is the most technically advanced propulsion system control ever produced by the aerospace industry, demonstrating the highest pilot rating for flight performance and providing independent control of vertical thrust and pitch from five sources. This innovative and industry-leading advanced design is protected with five broad patents for Pratt & Whitney and UTC, and is the new standard for propulsion system control for Pratt & Whitney military and commercial engines.

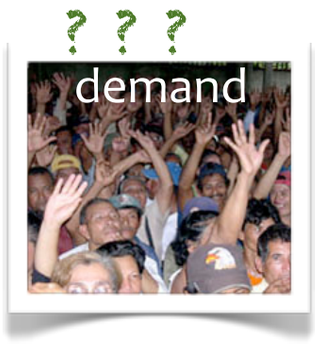
Pratt & Whitney
A United Technologies Company

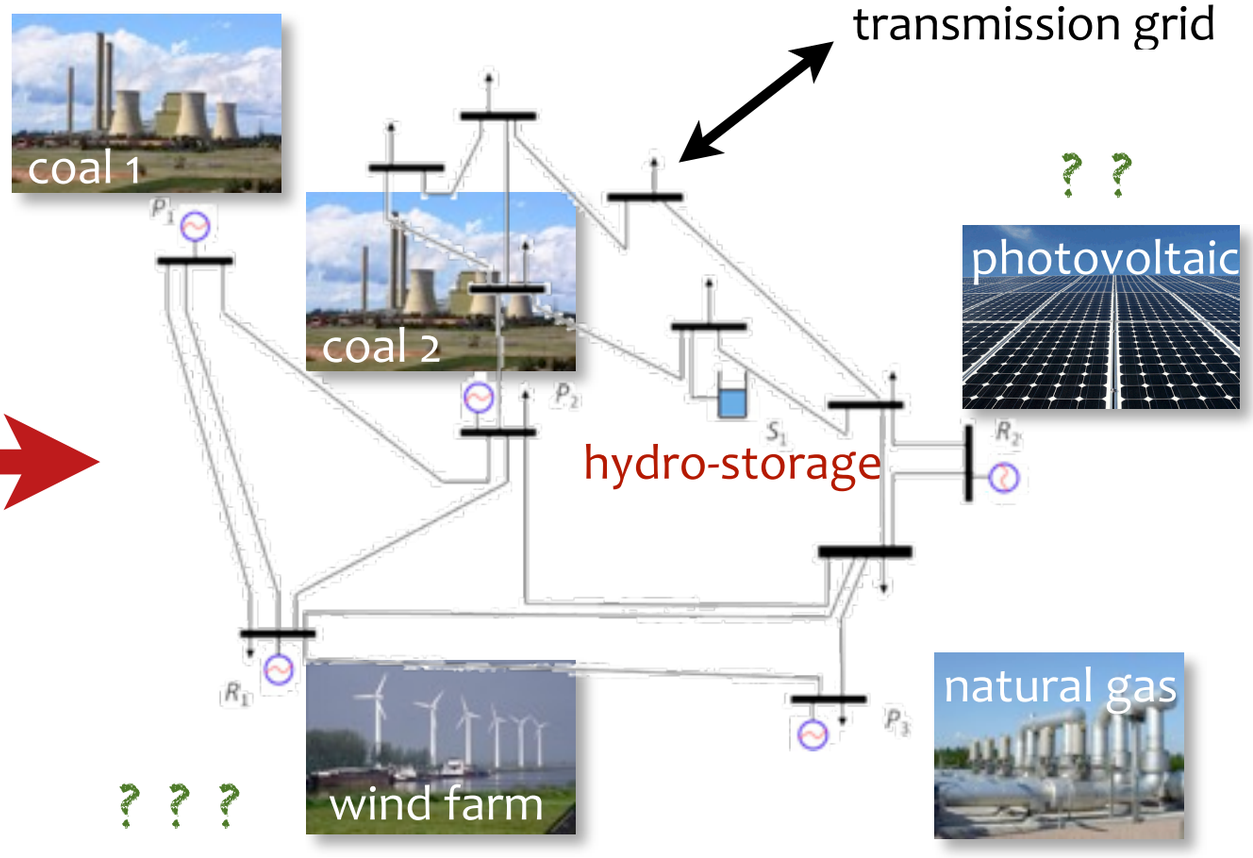
http://www.pw.utc.com/Press/Story/20100527-0100/2010

MPC FOR SMART ELECTRICITY GRIDS



? ? ?





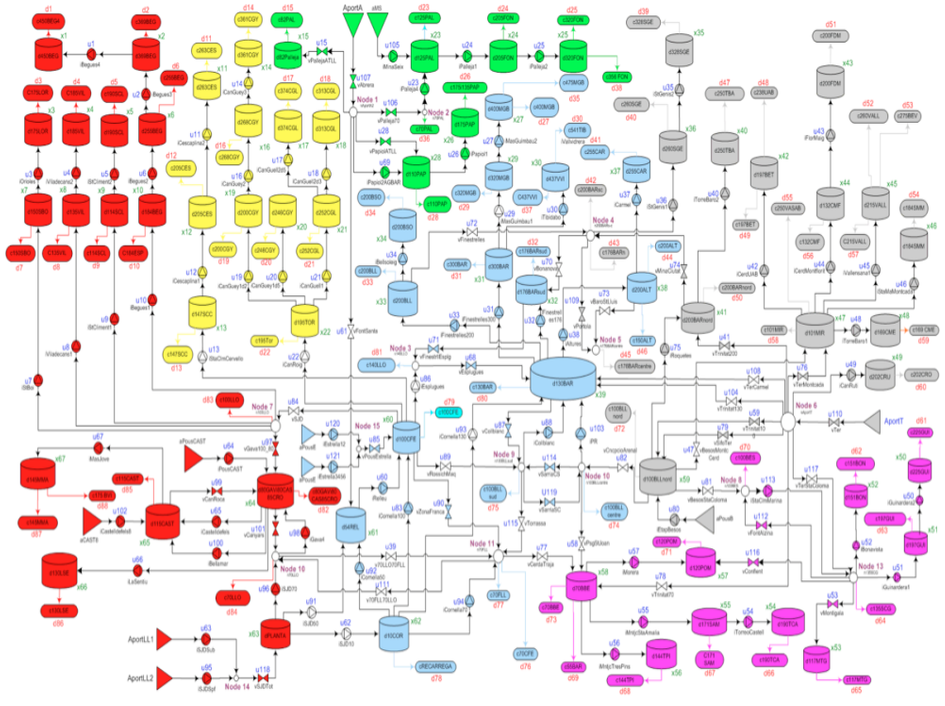
Dispatch power in smart distribution grids, **trade energy** on energy markets

Challenges: account for **dynamics**, network **topology**, physical **constraints**, and **stochasticity** (of renewable energy, demand, electricity prices)

FP7-ICT project "**E-PRICE - Price-based Control of Electrical Power Systems**" (2010-2013)

SMPC OF DRINKING WATER NETWORKS

(Sampathirao, Sopasakis, Bemporad, Patrinos, 2017)

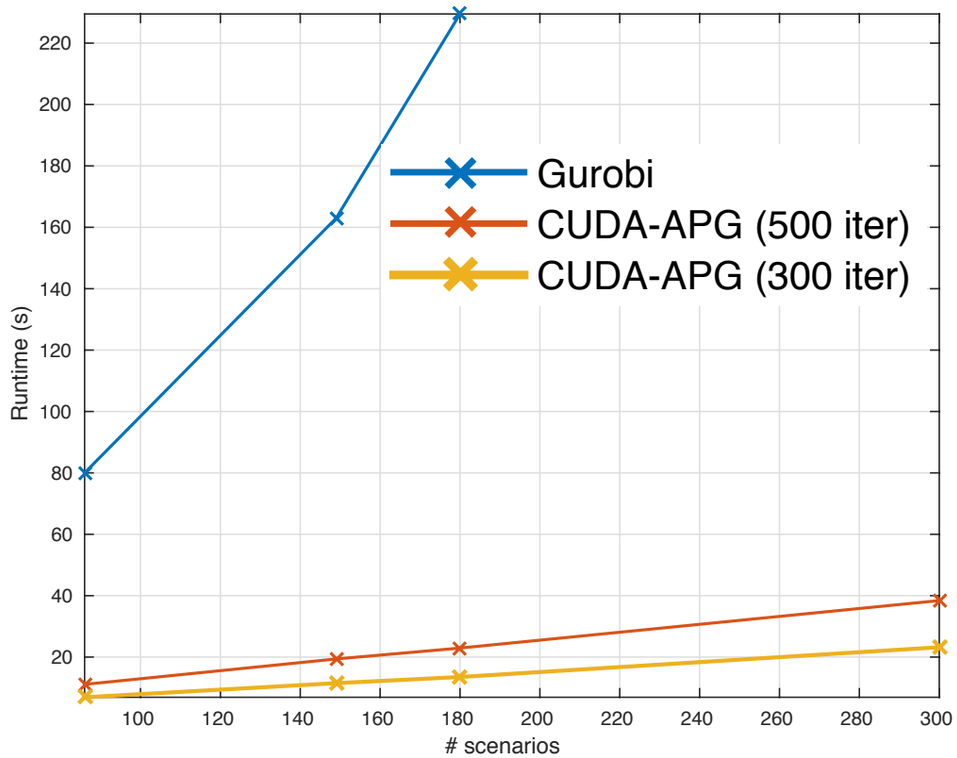


Drinking water network of Barcelona:

63 tanks
114 controlled flows
17 mixing nodes

Main goals:

- Reduce **electricity consumption** for pumping
- Meet **demand** requirements
- Keep storage tanks above safety **limits**



APG = Accelerated Proximal Gradient, parallel implemented on NVIDIA Tesla 2075 CUDA platform

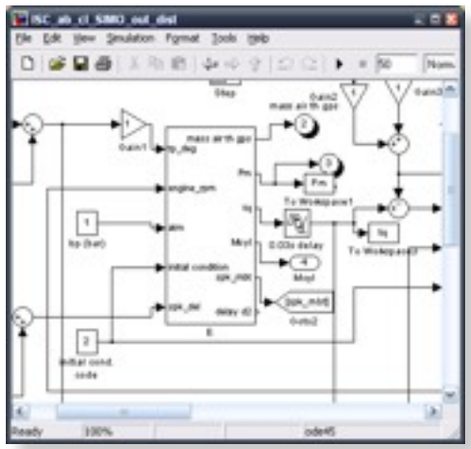
FP7-ICT project "**EFFINET - Efficient Integrated Real-time Monitoring and Control of Drinking Water Networks**" (2012-2015)

MPC RESEARCH IS DRIVEN BY APPLICATIONS

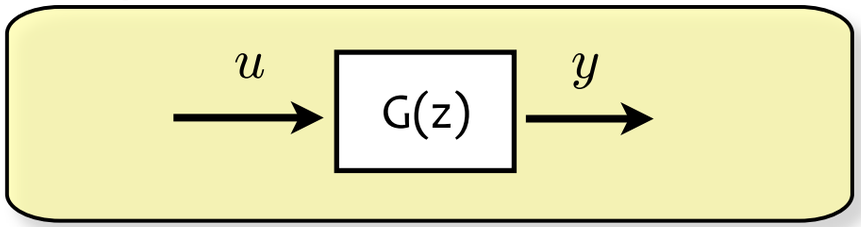
- **Process** control → **linear** MPC (some **nonlinear** too) *1970-2000*
- **Automotive** control → **explicit, hybrid** MPC *2001-2010*
- **Aerospace** systems and UAVs → **linear time-varying** MPC *>2005*
- **Information and Communication Technologies (ICT)** *>2005*
  (wireless nets, cloud computers) → **distributed**/**decentralized** MPC
- **Energy, finance, automotive, water** → **stochastic** MPC *>2010*
- **Industrial production** → **embedded solvers** for MPC *today*

MPC DESIGN FLOW

High-fidelity simulation model



(simplified) control-oriented prediction model



performance index & constraints

MPC design

real-time code

```
/* z=A*x0; */
for (i=0;i<m;i++) {
    z[i]=A[i]*x[0];
    for (j=1;j<n;j++) {
        z[i]+=A[i+m*j]*x[j];
    }
}
```

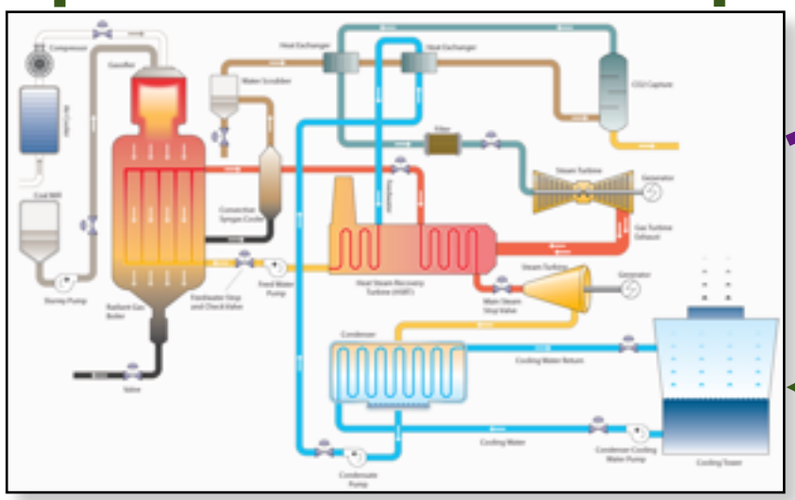
closed-loop simulation

revise MPC setup

experiments

physical modeling + parameter estimation

system identification



physical process

MPC TOOLBOXES

- **MPC Toolbox (The Mathworks, Inc.)** (Bemporad, Ricker, Morari, 1998-present)
  - Part of Mathworks' official toolbox distribution
  - Great for **education and research**

- **Hybrid Toolbox** (Bemporad, 2003-present)
  - Free download: **http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox/**
  - Great for **research and education**

~6800 downloads /
~1.5 downloads/day

- **ODYS' MPC Toolbox (ODYS S.r.l.)** (Bemporad, Bernardini, 2013-present)
  - Flexible and customized MPC **design** and **seamless integration** in production systems
  - Real-time **MPC code** and **QP solver** written in **plain C**
  - Supports (multiple) linear and linear time-varying models
  - Designed for **industrial production**

ODYS
Advanced Controls & Optimization

PROS AND CONS OF MPC

✓ Extremely flexible control design approach:

- Prediction model can be **multivariable**, w/**delays**, **time-varying**, w/ **disturbances**, …
- Can exploit available **preview** on future references and measured disturbances
- Handles **constraints** on inputs and outputs
- (Auto)**tuning** similar to Linear Quadratic Regulator (**LQR**)
- Mature code and development **tools** available

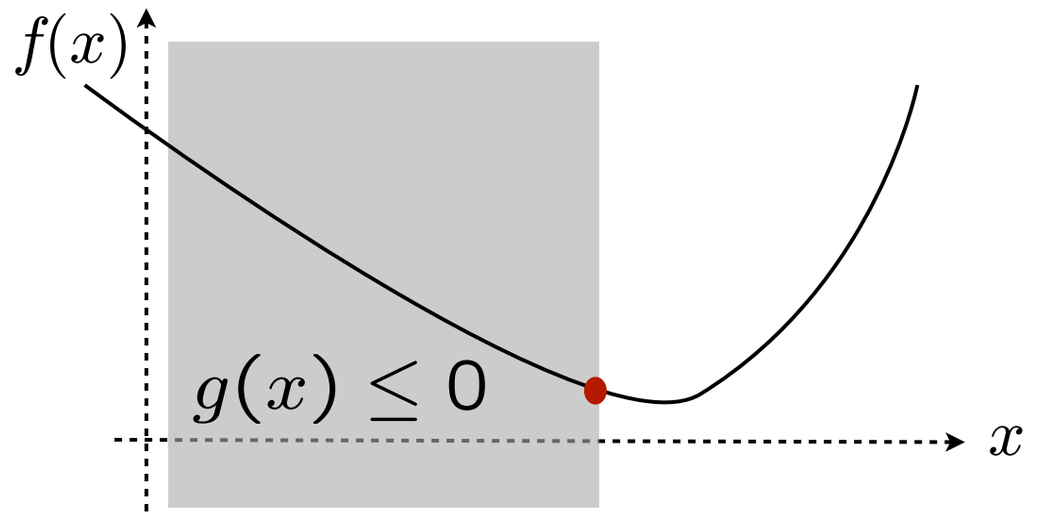
- Price to pay:
  - Requires a (simple) **model** (experiments, systems identification, linearization)
  - Many **degrees of freedom** (weights, horizons, constraints, …)
  - Requires **real-time computations** to solve the optimization problem

BASICS OF CONSTRAINED OPTIMIZATION

MATHEMATICAL PROGRAMMING

$$\begin{array}{ll} \min_x & f(x) \\ \text{s.t.} & g(x) \le 0 \end{array}$$

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



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

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

In general, problem is difficult to solve ⟹ **use software tools**

OPTIMIZATION SOFTWARE

- Comparison on benchmark problems:

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

- Taxonomy of most known solvers, for different classes of optimization problems: http://www.neos-guide.org

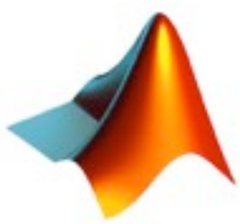
- NEOS server for remotely solving optimization problems:

  https://neos-server.org

- Good open-source optimization software:

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

- GitHub, MATLAB Central, Google ...

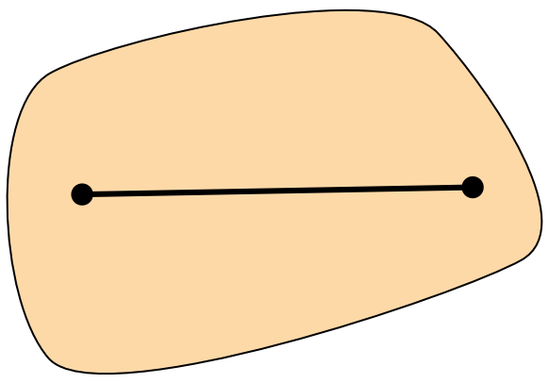




CONVEX SETS

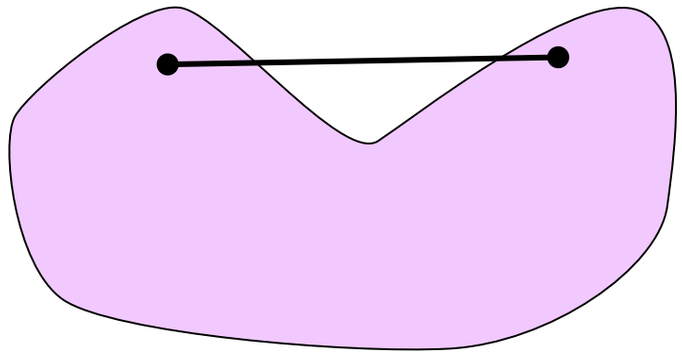
A **set** $S \in X$ is convex if for all $x_1, x_2 \in S$

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

convex set



nonconvex set

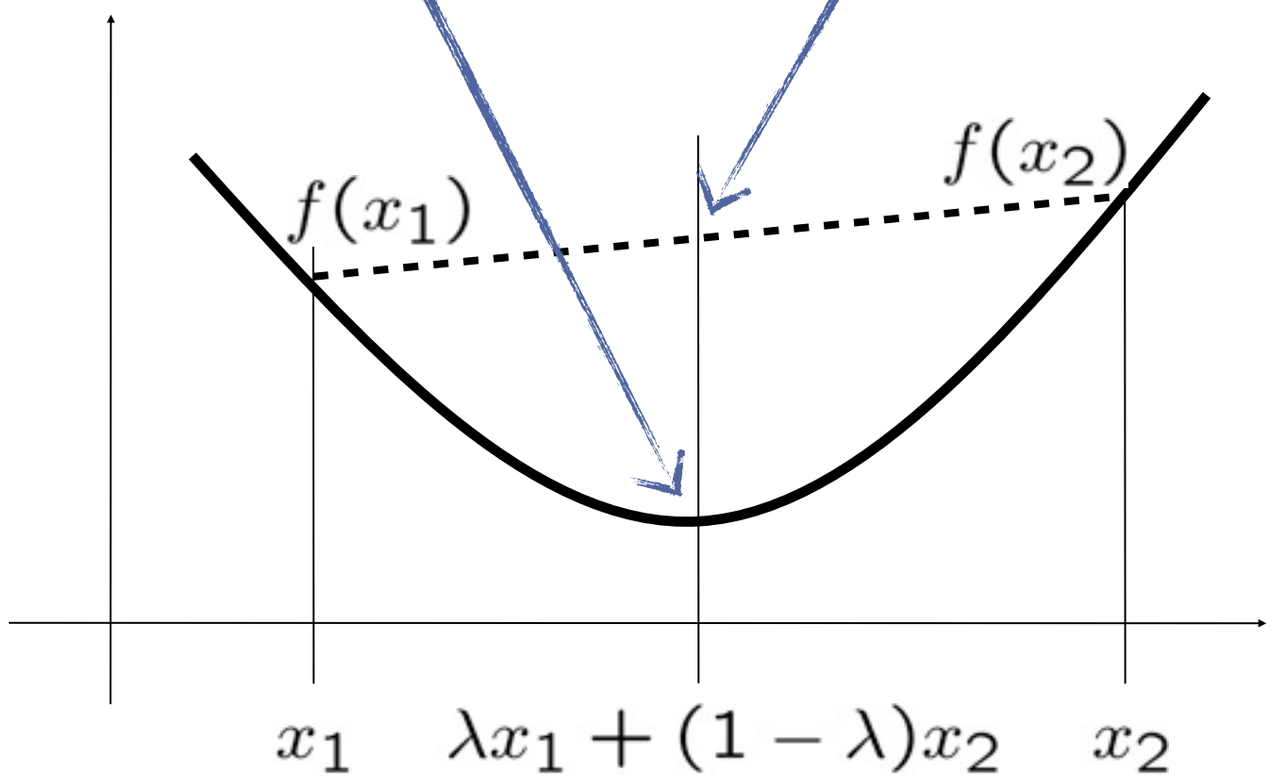


CONVEX FUNCTION

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

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

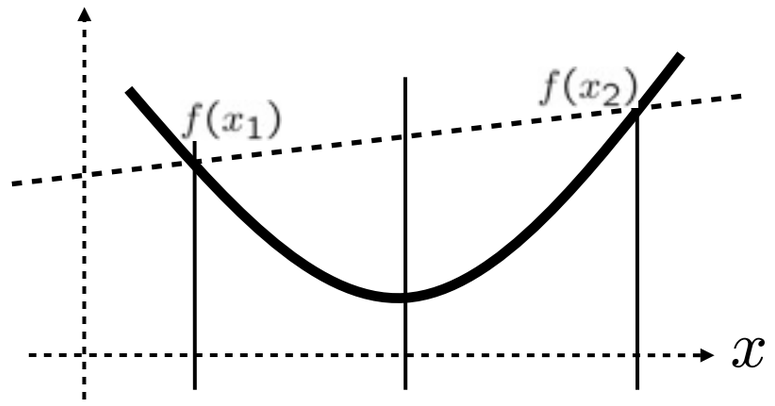
for all $x_1, x_2 \in S, \lambda \in [0,1]$



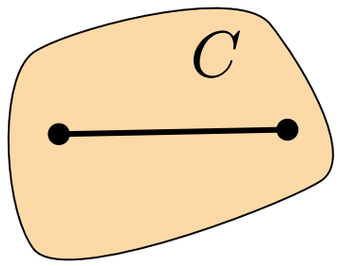
CONVEX OPTIMIZATION PROBLEM

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & x \in C \end{aligned}$$

f = convex function
C = convex set



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



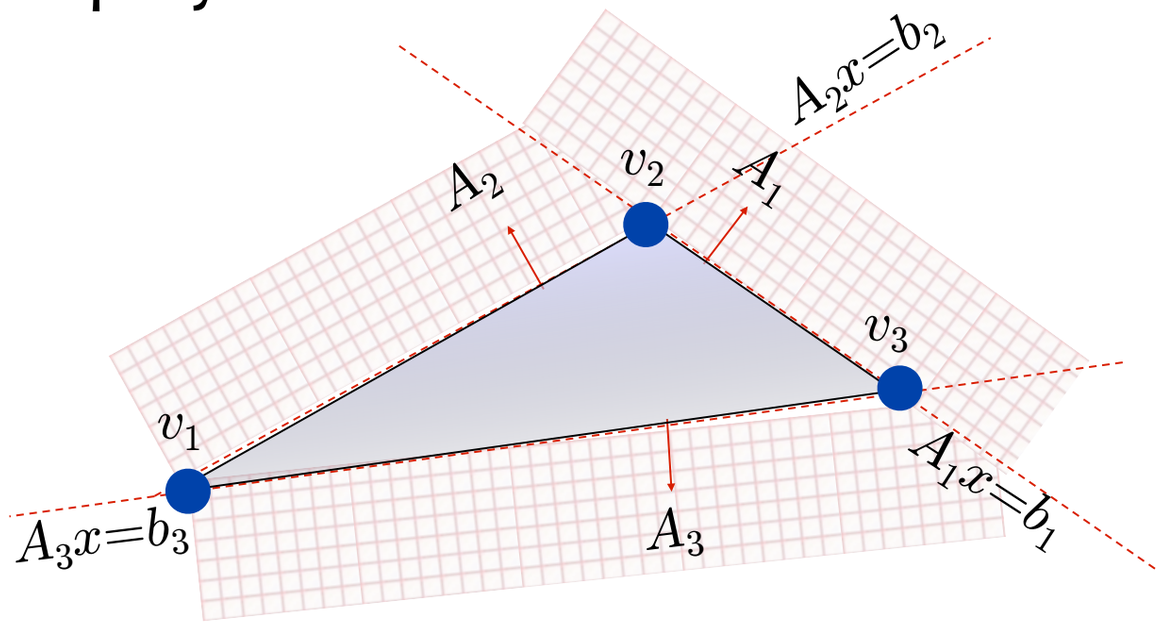
Excellent textbook: "Convex Optimization" by S. Boyd & L. Vandenberghe

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:

$$P = \{x \in \mathbb{R}^n : \ Ax \leq b\}$$

vertex enumeration

convex hull

- Vertex representation:

$$P = \{x \in \mathbb{R}^n : \ x = \sum_{i=1}^{q} \alpha_i v_i\}$$

$$\alpha_i \geq 0, \ \sum_{i=1}^{q} \alpha_i = 1, \ v_i \in \mathbb{R}^n$$

LINEAR PROGRAMMING

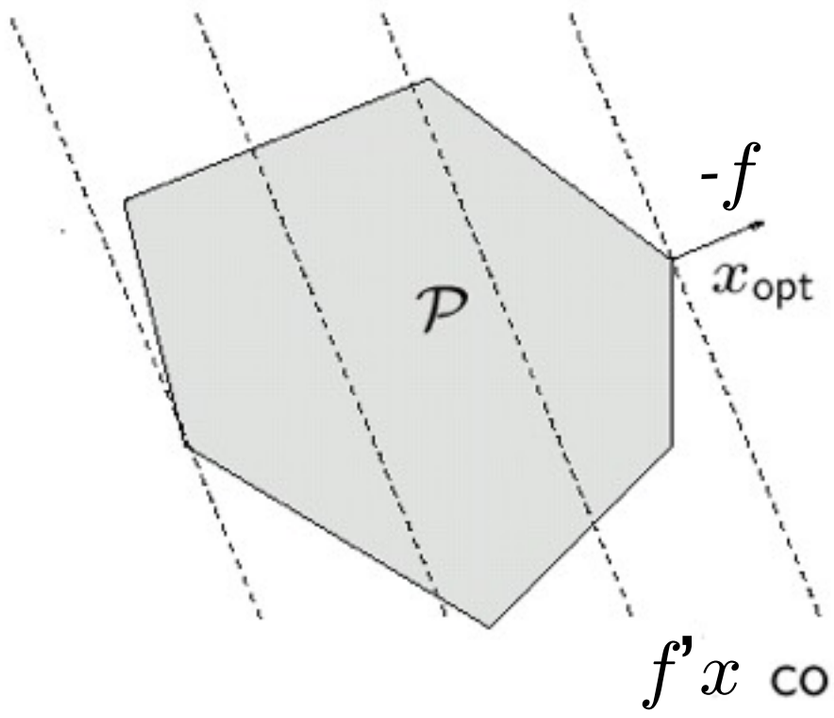
$$\begin{aligned} \min \quad & f'x \\ \text{s.t.} \quad & Ax \le b, \quad x \in \mathbb{R}^n \end{aligned}$$

linear program (LP)

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

LP in standard form

George Dantzig
(1914 - 2005)



slack variables

$$\begin{cases} \sum_{j=1}^{n} a_{ij}x_j \le b_i \\ x_j \ge 0 \end{cases} \Rightarrow \begin{cases} \sum_{j=1}^{n} a_{ij}x_j + s_i = b_i \\ s_i, x_j \ge 0 \end{cases}$$

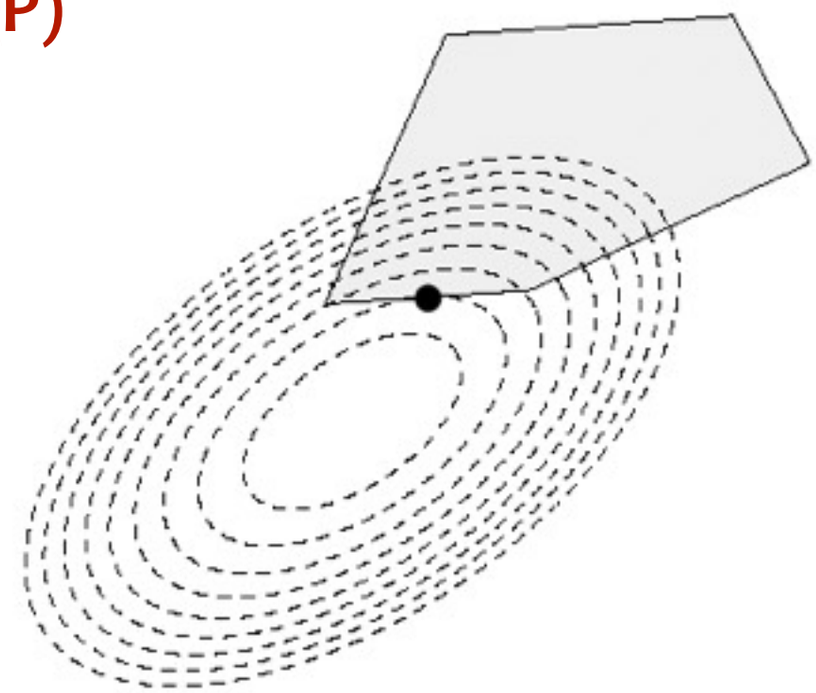
split positive and negative parts

$$\begin{cases} \sum_{j=1}^{n} a_{ij}x_j = b_i \\ x_j \text{ free} \end{cases} \Rightarrow \begin{cases} \sum_{j=1}^{n} a_{ij}(x_j^+ - x_j^-) = b_i \\ x_j^+, x_j^- \ge 0 \end{cases}$$

QUADRATIC PROGRAMMING

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

quadratic program (QP)



- Convex optimization if $P \geq 0$ ($P =$ positive semidefinite matrix)
- Hard problem if $P \not\geq 0$ ($P =$ indefinite matrix)

MIXED-INTEGER PROGRAMMING (MIP)

$$\begin{aligned} \min \quad & f'x \\ \text{s.t.} \quad & Ax \le b, \;\; x = \begin{bmatrix} x_c \\ x_b \end{bmatrix} \\ & x_c \in \mathbb{R}^{n_c}, \;\; x_b \in \{0,1\}^{n_b} \end{aligned}$$

mixed-integer linear program (MILP)

$$\begin{aligned} \min \quad & \frac{1}{2}x'Px + f'x \\ \text{s.t.} \quad & Ax \le b, \;\; x = \begin{bmatrix} x_c \\ x_b \end{bmatrix} \\ & x_c \in \mathbb{R}^{n_c}, \;\; x_b \in \{0,1\}^{n_b} \end{aligned}$$

mixed-integer quadratic program (MIQP)

- Some variables are continuous, some are discrete $(0/1)$
- A $\mathcal{NP}$-hard problem, in general
- A rich variety of algorithms/solvers is available

MODELING LANGUAGES

- **AMPL** (A Modeling Language for Mathematical Programming) most used modeling language, supports several solvers

- **OPL** (Optimization Programming Language), associated with commercial package IBM-CPLEX

- **MOSEL**, associated with commercial package FICO Xpress

- **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)

MODELING LANGUAGES

- **YALMIP** MATLAB-based modeling language

- **CVX (CVXPY)** Modeling language for convex problems in MATLAB (Python)

- **PYOMO** Python-based modeling language

- **PICOS** A Python interface to conic optimization solvers

- **PuLP** An linear programming modeler for Python

- **JuMP** A modeling language for linear, quadratic, and nonlinear constrained optimization problems embedded in Julia

LINEAR MPC

LINEAR MPC - UNCONSTRAINED CASE

- Linear prediction model:

$$\begin{cases} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{cases}$$

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

$x_0 = x(t)$

$$x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j}$$

- Performance index

$$J(z, x_0) = x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k$$

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

$$\begin{aligned} R &= R' \succ 0 \\ Q &= Q' \succeq 0 \\ P &= P' \succeq 0 \end{aligned}$$

- **Goal**: find sequence $z^* = \begin{bmatrix} u_0^* \\ u_1^* \\ \vdots \\ u_{N-1}^* \end{bmatrix}$ that steers the state to the origin optimally, i.e., minimizing $J(z,x_0)$

[COMPUTATION OF COST FUNCTION]

$$J(z,x_0) = x_0'Qx_0 + \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}' \overbrace{\begin{bmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & Q & 0 \\ 0 & 0 & \dots & 0 & P \end{bmatrix}}^{\bar{Q}} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}$$

$$+ \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \underbrace{\begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & R \end{bmatrix}}_{\bar{R}} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \underbrace{\begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}}_{\bar{S}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{z} + \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{\bar{T}} x_0$$

$$\begin{aligned} J(z,x_0) &= (\bar{S}z + \bar{T}x_0)'\bar{Q}(\bar{S}z + \bar{T}x_0) + z'\bar{R}z + x_0'Qx_0 \\ &= \frac{1}{2}z' \underbrace{2(\bar{R} + \bar{S}'\bar{Q}\bar{S})}_{H} z + x_0' \underbrace{2\bar{T}'\bar{Q}\bar{S}}_{F'} z + \frac{1}{2}x_0' \underbrace{2(Q + \bar{T}'\bar{Q}\bar{T})}_{Y} x_0 \end{aligned}$$

LINEAR MPC - UNCONSTRAINED CASE

$$J(z,x_0) = \frac{1}{2}z'Hz + x_0'F'z + \frac{1}{2}x_0'Yx_0$$

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

The optimum is obtained by zeroing the gradient

$$\nabla_z J(z,x_0) = Hz + Fx_0 = 0$$

and hence $z^* = \begin{bmatrix} u_0^* \\ u_1^* \\ \vdots \\ u_{N-1}^* \end{bmatrix} = -H^{-1}Fx_0$ **(batch least squares)**

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

LINEAR MPC - CONSTRAINED CASE

- Linear prediction model:

$$\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \end{cases}$$

$$x_0 = x(t)$$

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

- Constraints to enforce:

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

- Constrained optimal control problem (quadratic performance index):

$$\min_z \quad x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k$$

$$\text{s.t.} \quad u_{\min} \le u_k \le u_{\max}, \quad k = 0, \ldots, N-1$$
$$y_{\min} \le y_k \le y_{\max}, \quad k = 1, \ldots, N$$

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

LINEAR MPC - CONSTRAINED CASE

- State response: $x_k = A^k x_0 + \sum_{i=0}^{k-1} A^i B u_{k-1-i}$

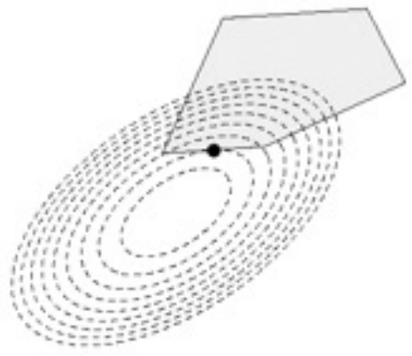
- Optimization problem:

$$V(x_0) = \tfrac{1}{2} x_0' Y x_0 + \min_z \ \tfrac{1}{2} z' H z + x_0' F' z \quad \text{(quadratic)}$$

$$\text{s.t.} \quad Gz \le W + Sx_0 \quad \text{(linear)}$$

Convex QUADRATIC PROGRAM (QP)

- $z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \in \mathbb{R}^s, \ \ s \triangleq Nm$ 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

[COMPUTATION OF CONSTRAINT MATRICES]

- Input constraints $u_{\min} \le u_k \le u_{\max},\ k = 0, \ldots, N-1$

$$\begin{aligned} u_k &\le u_{\max} \\ -u_k &\le -u_{\min} \end{aligned}$$

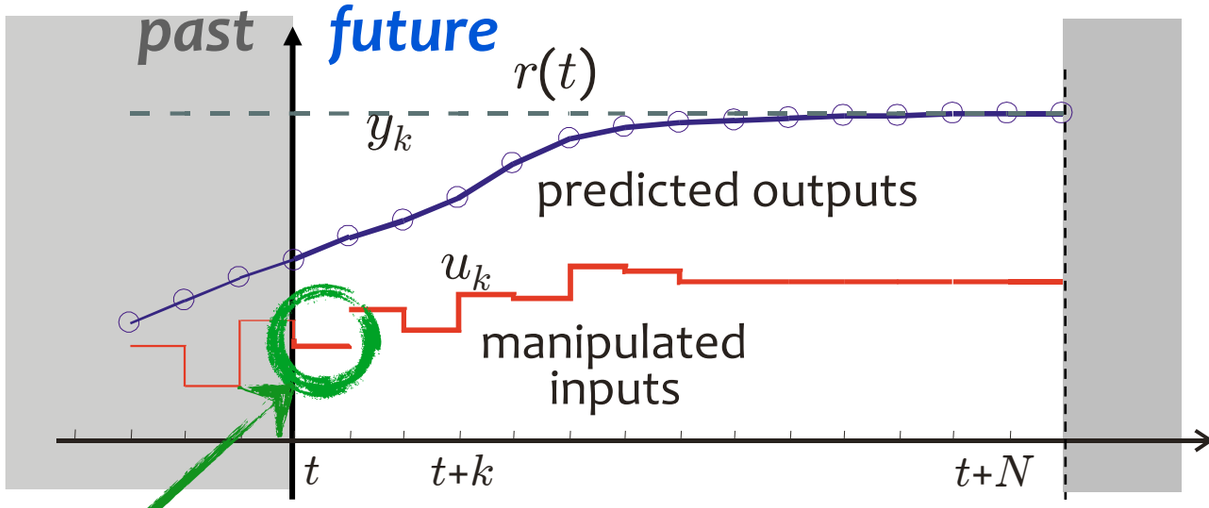
$$\begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & 1 \\ -1 & 0 & \dots & 0 \\ 0 & -1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & -1 \end{bmatrix} z \le \begin{bmatrix} u_{\max} \\ u_{\max} \\ \vdots \\ u_{\max} \\ -u_{\min} \\ -u_{\min} \\ \vdots \\ -u_{\min} \end{bmatrix} \qquad z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

- Output constraints $y_k = CA^k x_0 + \sum_{i=0}^{k-1} CA^i B u_{k-1-i} \le y_{\max},\ k = 1, \ldots, N$

$$\begin{bmatrix} CB & 0 & \dots & 0 \\ CAB & CB & \dots & 0 \\ \vdots & & & \vdots \\ CA^{N-1}B & \dots & CAB & CB \end{bmatrix} z \le \begin{bmatrix} y_{\max} \\ y_{\max} \\ \vdots \\ y_{\max} \end{bmatrix} - \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix} x_0$$

LINEAR MPC ALGORITHM

@ each sampling step t:



- Measure (or estimate) the current state $x(t)$
- Get the solution $z^* = \{u_0^*, \ldots, u_{N-1}^*\}$ of the QP

feedback !

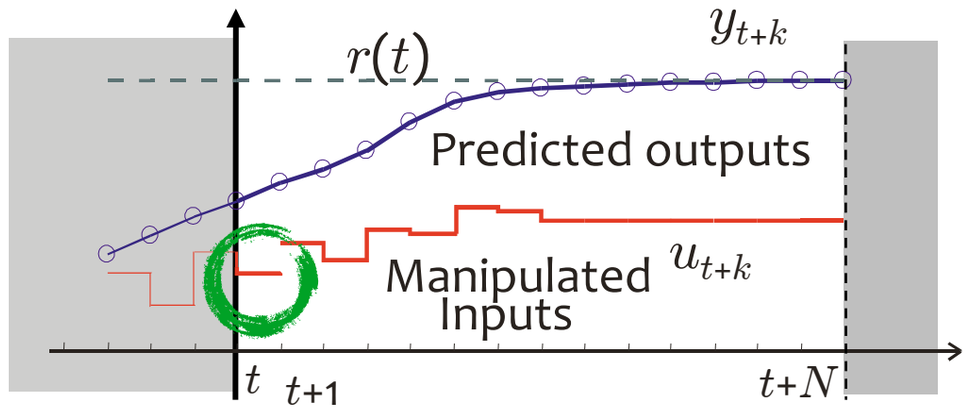
$$\min_z \quad \tfrac{1}{2} z' H z + x'(t) F' z$$
$$\text{s.t.} \quad Gz \le W + Sx(t)$$

- Apply only $u(t) = u_0^*$, discard remaining optimal inputs $u_1^*, \ldots, u_{N-1}^*$

LINEAR MPC - UNCONSTRAINED CASE

- Minimize quadratic function, no constraints

$$\min_z \; f(z) = \frac{1}{2} z' H z + x'(t) F' z$$



- Solution: $\nabla f(z) = Hz + Fx(t) = 0$

$$\longrightarrow \; z^* = -H^{-1} F x(t)$$

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

$$\Longrightarrow \quad u(t) = -[I \; 0 \; \ldots \; 0] H^{-1} F x(t) \triangleq K x(t)$$

Unconstrained linear MPC = linear state-feedback !

DOUBLE INTEGRATOR EXAMPLE

- System: $y(\tau) = \iint u(\tau) d\tau$

$$\frac{d^2y}{d\tau^2} = u$$

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: $\min \left(\sum_{k=0}^{1} y_k^2 + \frac{1}{10} u_k^2 \right) + x_2' \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_2$

- Optimization problem matrices:

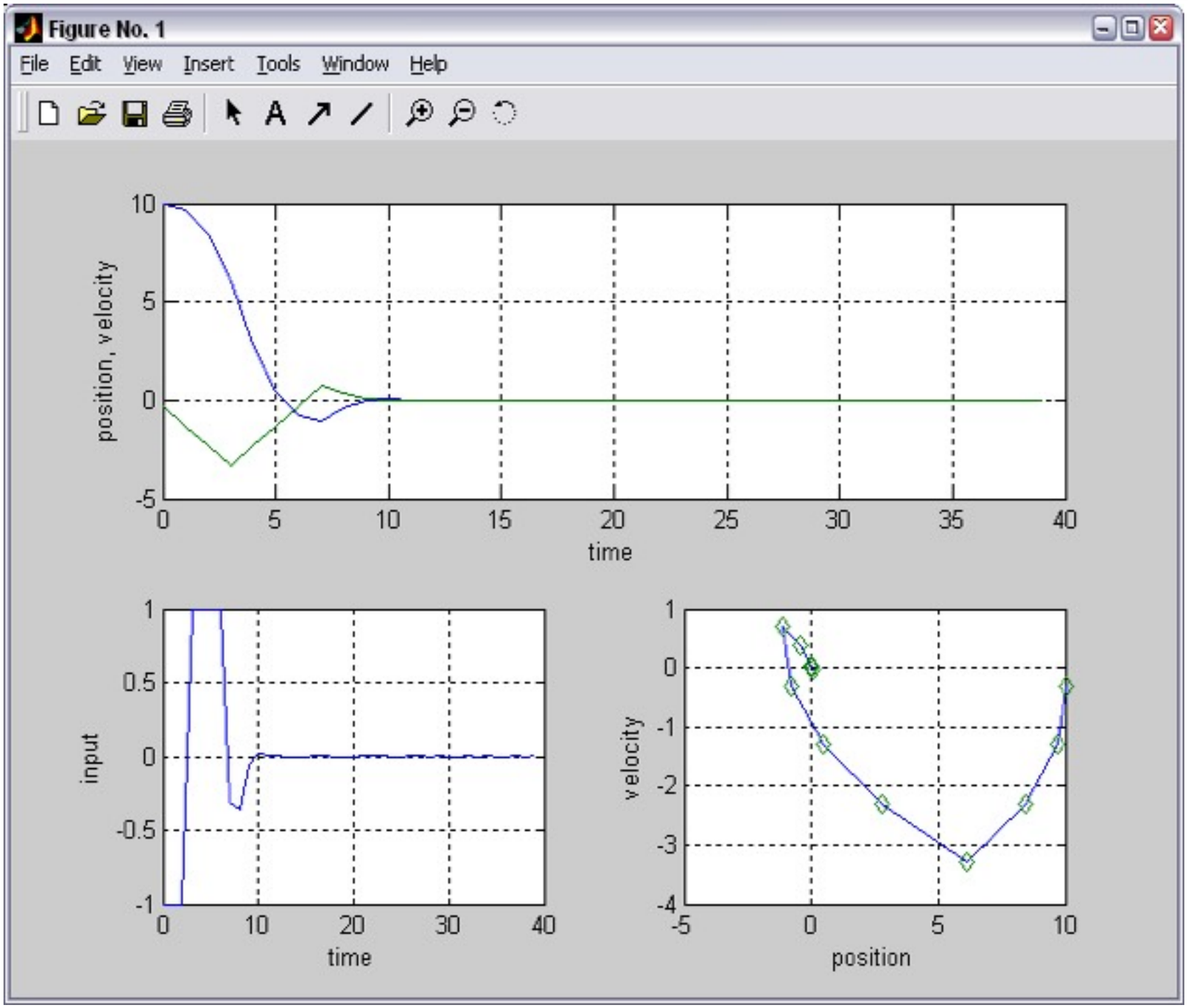
$$H = \begin{bmatrix} 4.2 & 2 \\ 2 & 2.2 \end{bmatrix}, \; F = \begin{bmatrix} 2 & 6 \\ 0 & 2 \end{bmatrix}, \; Y = \begin{bmatrix} 4 & 6 \\ 6 & 12 \end{bmatrix}$$

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

cost: $\frac{1}{2} z' H z + x'(t) F' z$

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

DOUBLE INTEGRATOR EXAMPLE



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

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

DOUBLE INTEGRATOR EXAMPLE

- Add constraint on second state: $x_{2,k} \geq -1, \;\; k = 1$

- Optimization problem matrices:

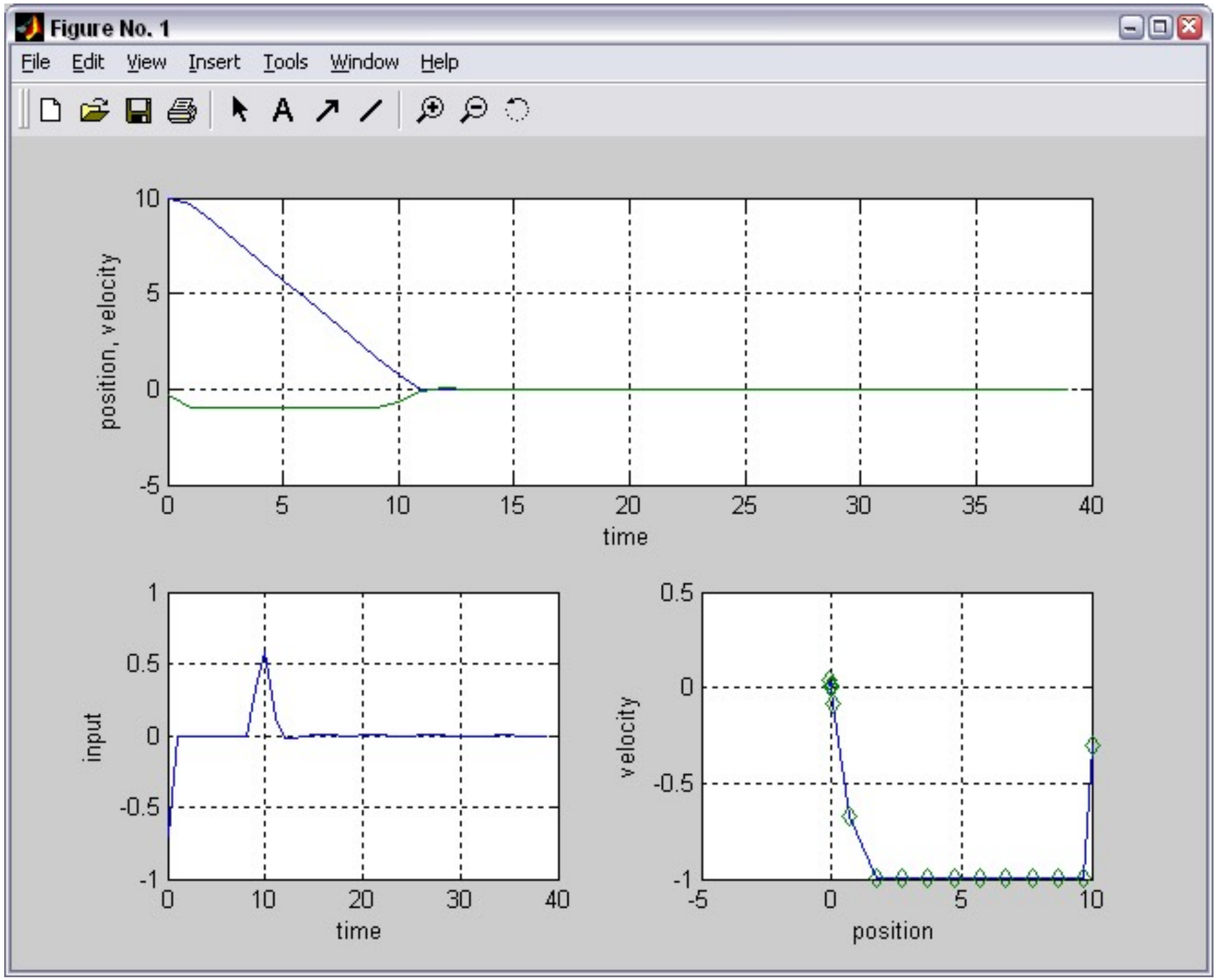
$$H = \begin{bmatrix} 4.2 & 2 \\ 2 & 2.2 \end{bmatrix}, \; F = \begin{bmatrix} 2 & 6 \\ 0 & 2 \end{bmatrix}, \; Y = \begin{bmatrix} 4 & 6 \\ 6 & 12 \end{bmatrix}$$

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

cost: $\frac{1}{2} z' H z + x'(t) F' z$

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

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 $x_u(t)=u(t\text{-}1)$

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

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

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):

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

$$[\Delta u_k \triangleq u_k - u_{k-1}],\ u_{-1} = u(t-1)$$

$$\text{subj. to} \quad u_{\min} \le u_k \le u_{\max},\ k = 0,\dots,N-1$$
$$y_{\min} \le y_k \le y_{\max},\ k = 1,\dots,N$$
$$\Delta u_{\min} \le \Delta u_k \le \Delta u_{\max},\ k = 0,\dots,N-1$$

optimization vector

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

- Note: $\|Wa\|^2 = (Wa)'(Wa) = a'(W'W)a = a'Qa$

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

- Optimization problem:

Convex QUADRATIC PROGRAM (QP)

$$\min_z \quad J(z, x(t)) = \tfrac{1}{2}z'Hz + [x'(t)\ r'(t)\ u'(t-1)]F'z$$

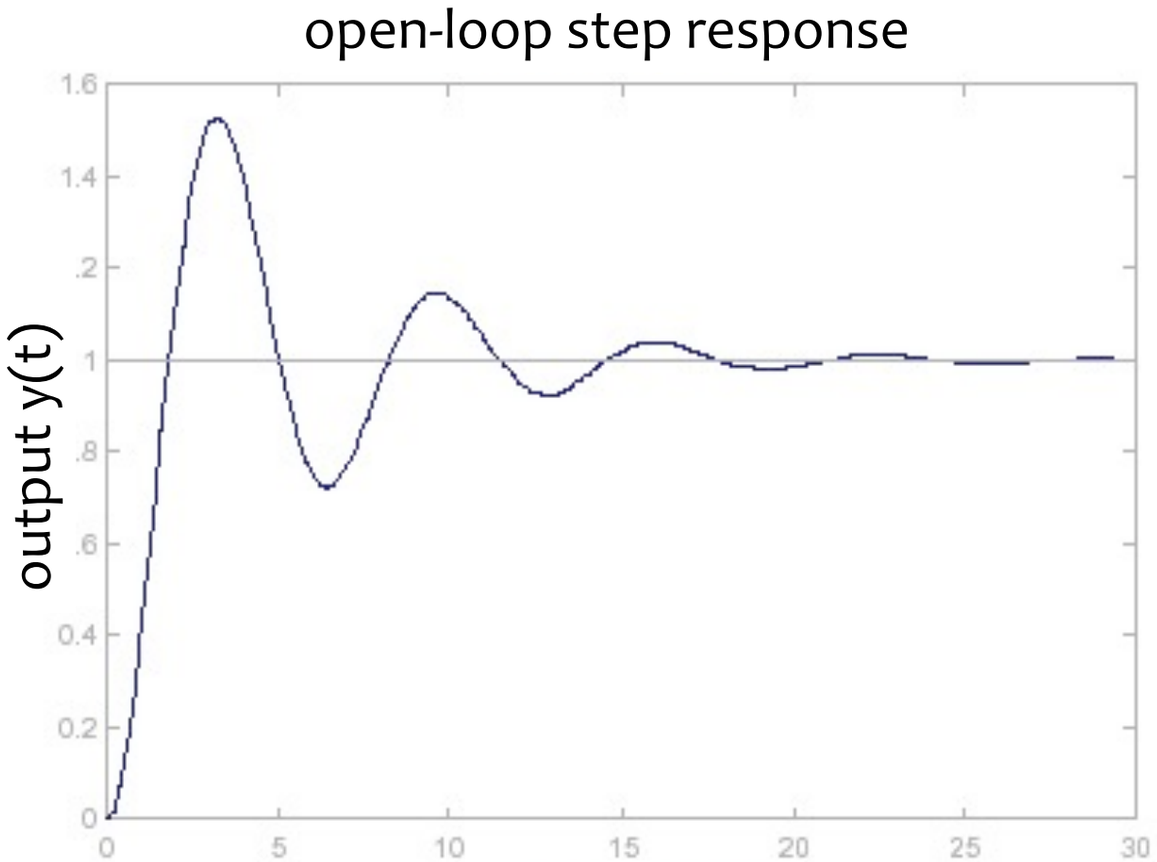
$$\text{s.t.} \quad Gz \le W + S\begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \end{bmatrix}$$

- Input references can be also handled by adding the extra penalty $\|W^u(u_k - u_{\text{ref}}(t))\|^2$
- Constraints may also depend on $r(t)$, e.g., on tracking errors $e_{\min} \le y_k - r(t) \le e_{\max}$

LINEAR MPC - TRACKING EXAMPLE

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

- Sampling time: $T_s = 0.5$ sec.



- Model:

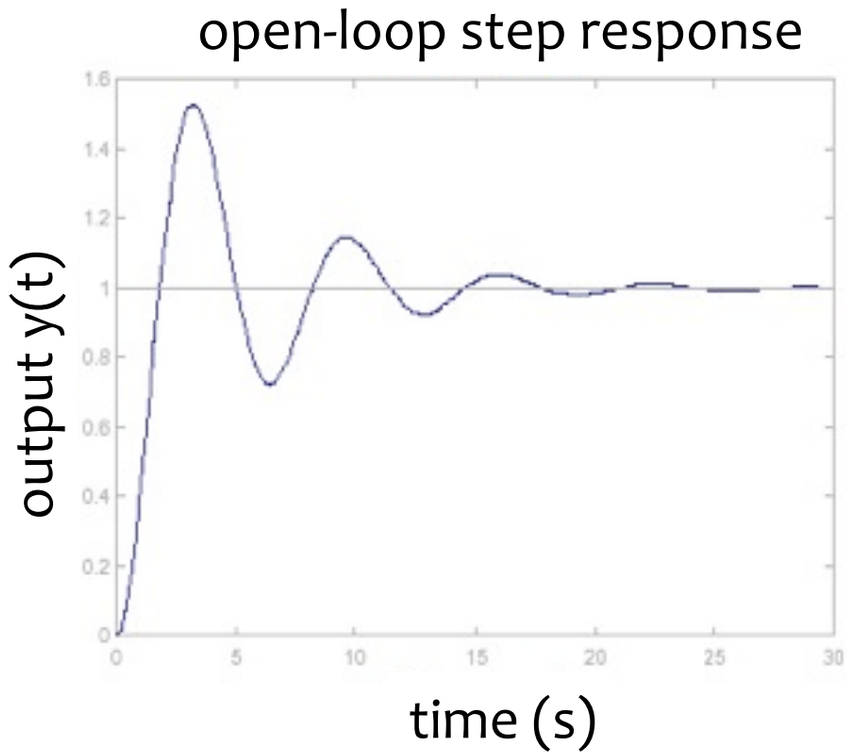
$$\begin{cases} x(t+1) = \begin{bmatrix} 1.597 & -0.4094 \\ 2 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} u(t) \\ y(t) = \begin{bmatrix} 0.2294 & 0.1072 \end{bmatrix} x(t) \end{cases}$$

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

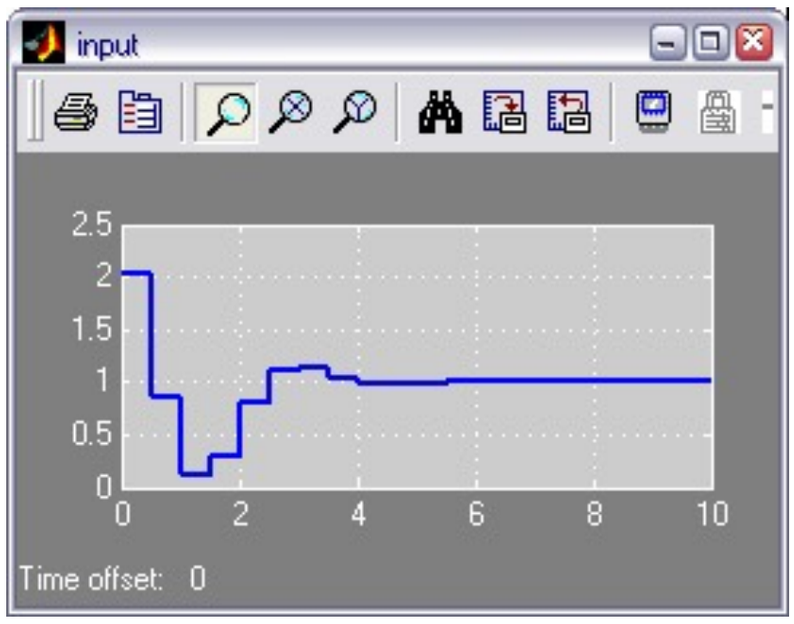
LINEAR MPC - TRACKING EXAMPLE

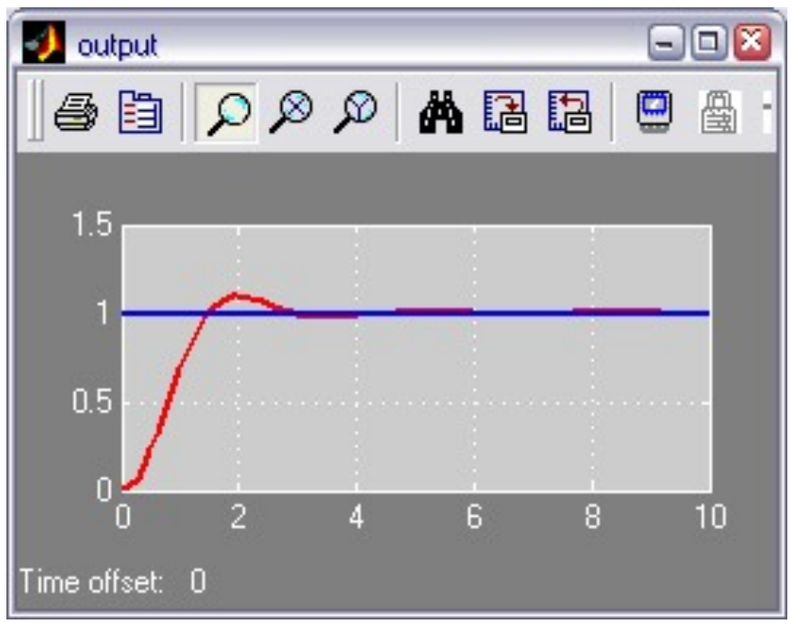
- Performance index:

$$\min \sum_{k=0}^{9} (y_{k+1} - r(t))^2 + 0.04\Delta u_k^2$$



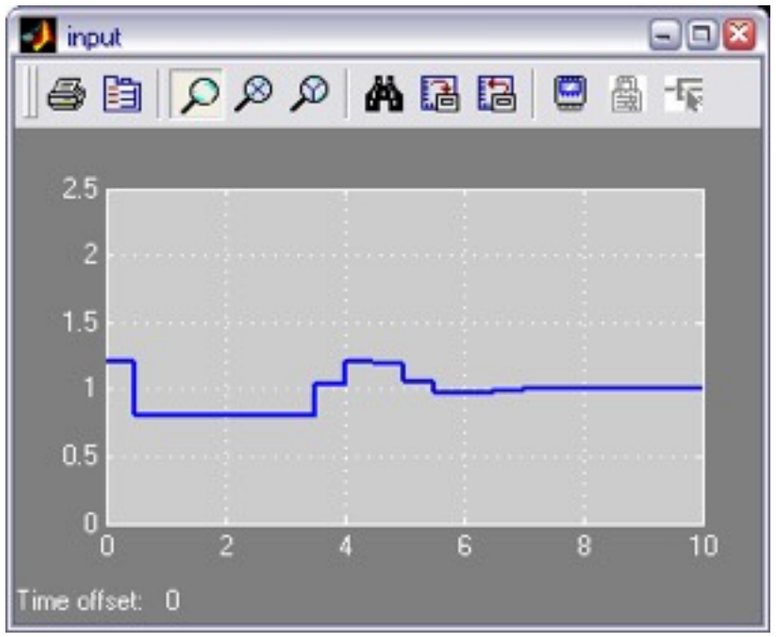
- Closed-loop MPC:

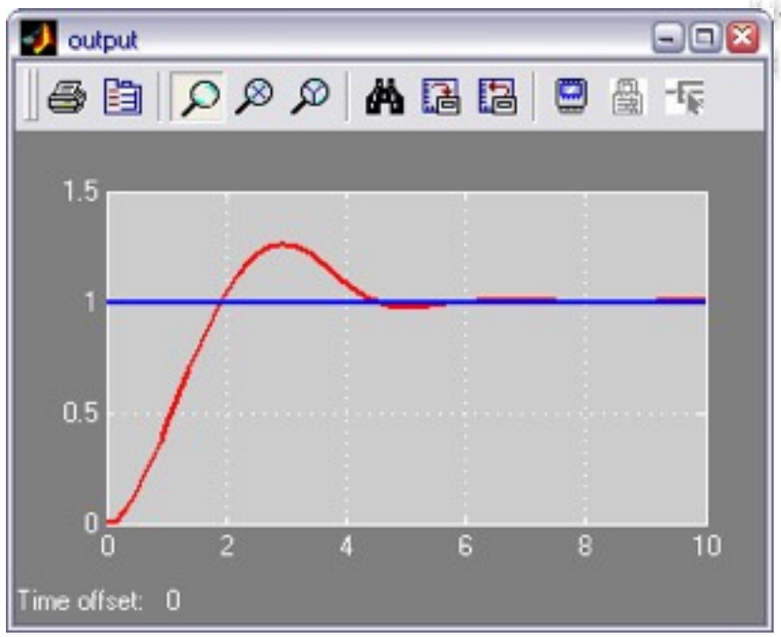


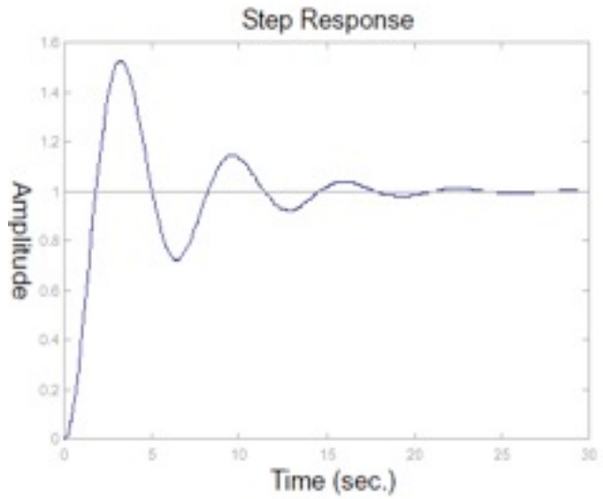


LINEAR MPC - TRACKING EXAMPLE

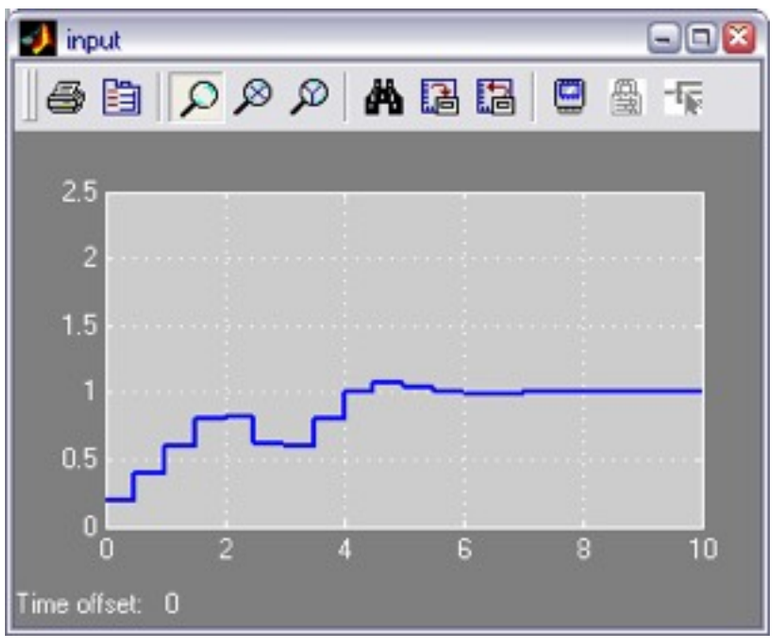
- Constraint $0.8 \leq u(t) \leq 1.2$ (amplitude)

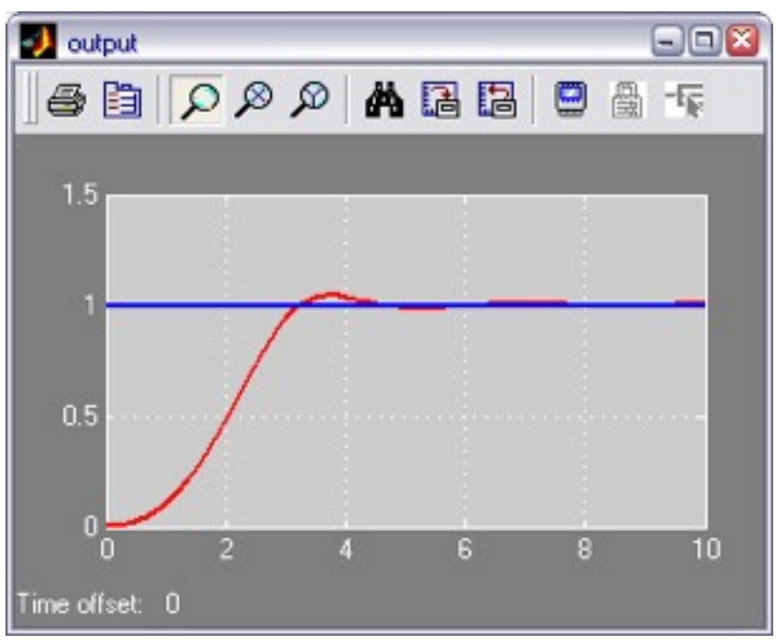






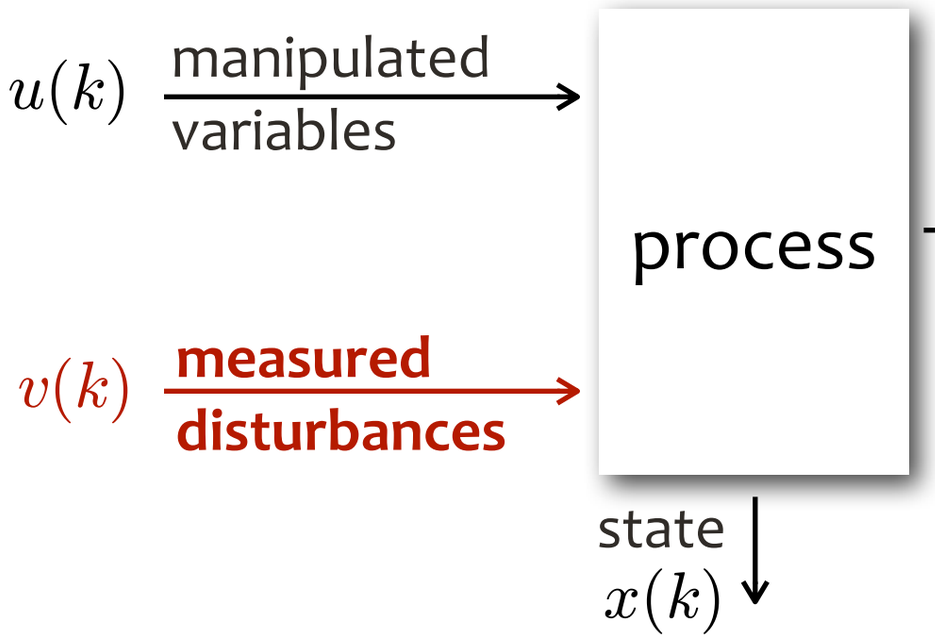
- Constraint $-0.2 \leq \Delta u(t) \leq 0.2$ (slew-rate)





MEASURED DISTURBANCES

- We also have an **input** v that is **measured** but not **manipulated**



$$\begin{cases} x_{k+1} = Ax_k + B + B_v v(k) \\ y_k = Cx_k + D_v v(k) \end{cases}$$

$v \in \mathbb{R}^{n_v}$

$$x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j} + AB_v v_{k-1-j}$$

- Same performance index, same constraints. We still have a QP:

$$\min_z \quad J(z, x(t)) = \tfrac{1}{2} z' H z + [x'(t) \;\; r'(t) \;\; u'(t-1) \;\; v'(t)] F' z$$

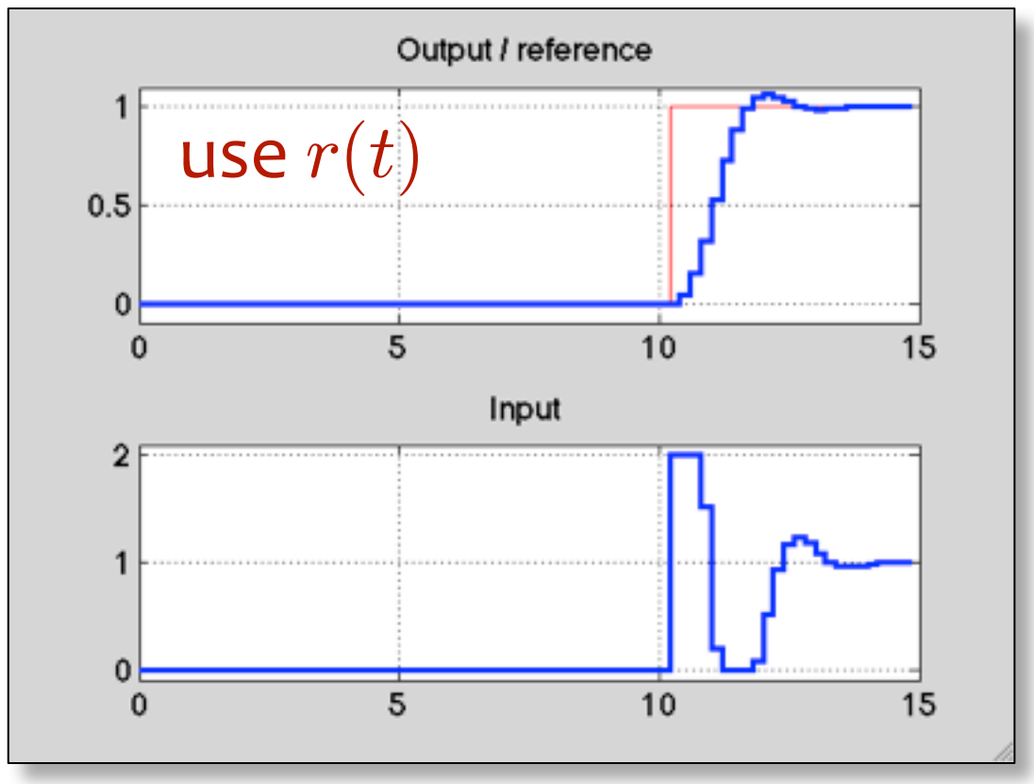
$$\text{s.t.} \quad Gz \le W + S \begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \\ v(t) \end{bmatrix}$$

ANTICIPATIVE ACTION (A.K.A. "PREVIEW")

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

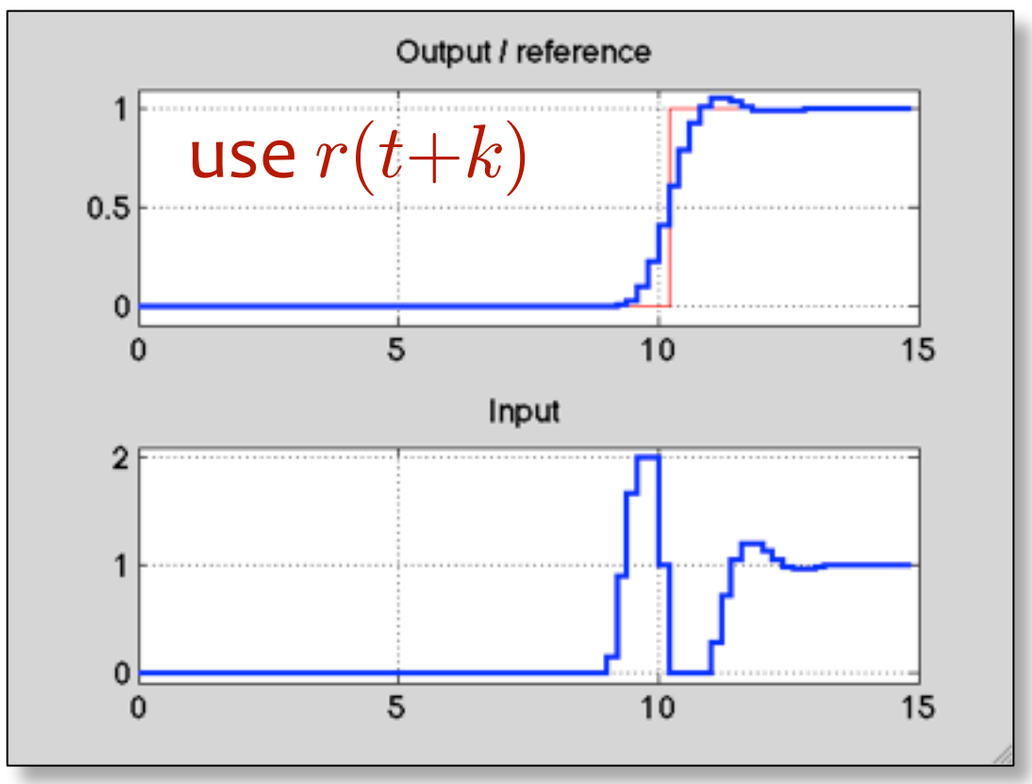
- Reference not known in advance (causal):

$$r_k \equiv r(t), \forall k = 0, \dots, N-1$$



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

$$r_k = r(t+k), \forall k = 0, \dots, N-1$$



Same idea also applies to reject **measured disturbances** entering the process

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

SOFT CONSTRAINTS

- To prevent QP infeasibility, relax output constraints:

$$\begin{aligned}
\min_{z} \quad & \sum_{k=0}^{N-1} \|W^y(y_{k+1} - r(t))\|^2 + \|W^{\Delta u}\Delta u_k\|^2 + \rho_\epsilon \epsilon^2 \\
\text{subj. to} \quad & u_{\min} \le u_k \le u_{\max}, \;\; k = 0, \ldots, N-1 \\
& \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}, \;\; k = 0, \ldots, N-1 \\
& y_{\min} - \epsilon V_{\min} \le y_k \le y_{\max} + \epsilon V_{\max}, \;\; k = 1, \ldots, N
\end{aligned}$$

ϵ = "panic" variable

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

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

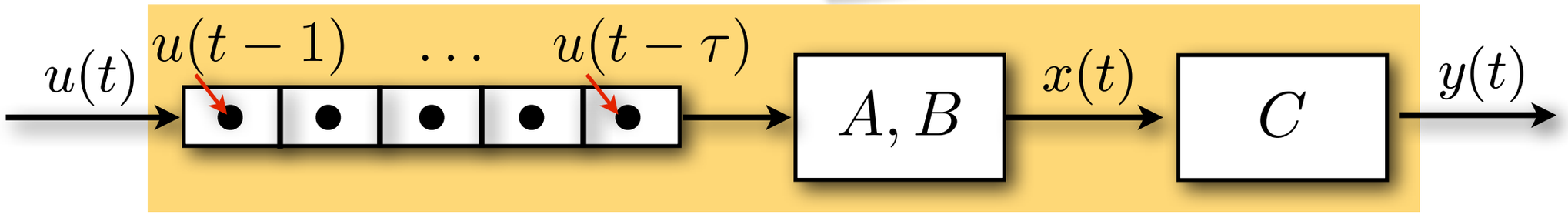
$V_{\min}$, $V_{\max}$ = vectors with entries ≥ 0 (the larger the i-th entry of vector V, the relatively softer the corresponding i-th constraint)

- 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 $z=0$:

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

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

- 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:

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

- 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:

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

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

GOOD MODELS FOR (MPC) CONTROL

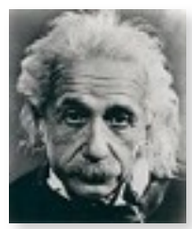
Note: computational complexity and theoretical properties (e.g. stability) depend on chosen model/objective/constraints

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 THEORY

- *After* the industrial success of MPC, a lot of research efforts were done in multiple directions:

  - linear MPC — linear prediction model
  - nonlinear MPC — nonlinear prediction model
  - robust MPC — uncertain (linear) prediction model
  - stochastic MPC — stochastic prediction model
  - distributed/decentralized MPC — multiple MPCs cooperating together
  - economic MPC — MPC based on arbitrary (economic) performance indices
  - hybrid MPC — prediction model integrating logic and dynamics
  - explicit MPC — off-line (exact/approximate) computation of MPC
  - solvers for MPC — on-line numerical algorithms for solving MPC problems

- Main theoretical issues: **feasibility**, **stability**, solution **algorithms**

(Mayne, *Automatica*, 2014)

- Research on **solution algorithms** with guaranteed theoretical properties has the most impact in industrial practice

FEASIBILITY

$$\begin{aligned} \min_{z} \quad & \sum_{k=0}^{N-1} \|W^y(y_{k+1} - r(t))\|^2 + \|W^{\Delta u}\Delta u_k\|^2 \\ \text{subj. to} \quad & u_{\min} \le u_k \le u_{\max}, \;\; k = 0, \ldots, N-1 \\ & y_{\min} \le y_k \le y_{\max}, \;\; k = 1, \ldots, N \\ & \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}, \;\; k = 0, \ldots, N-1 \end{aligned}$$

QUADRATIC PROGRAM (QP)

- **Feasibility**: Will the QP problem be feasible at all sampling instants t ?

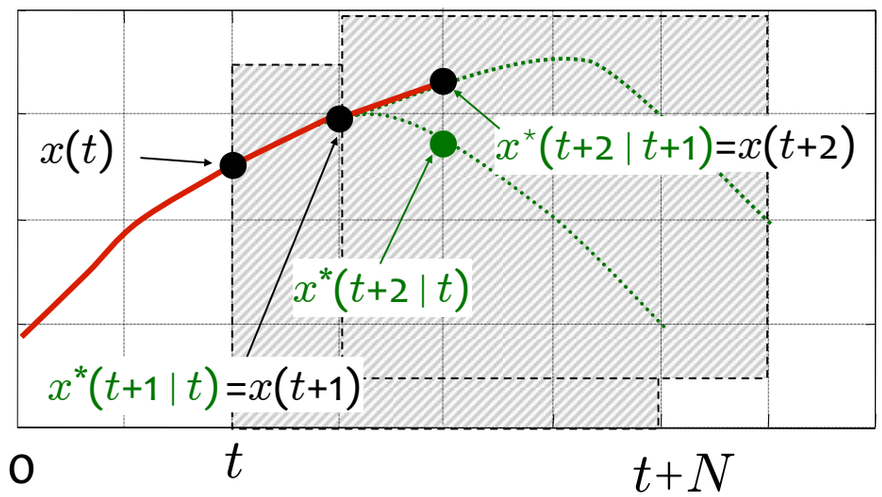
- **Input constraints only:** no feasibility issues !

- **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, 1991) (Kerrigan & Maciejowski, 2000) (Chmielewski & Manousiouthakis, 1996)

PREDICTED AND ACTUAL TRAJECTORIES

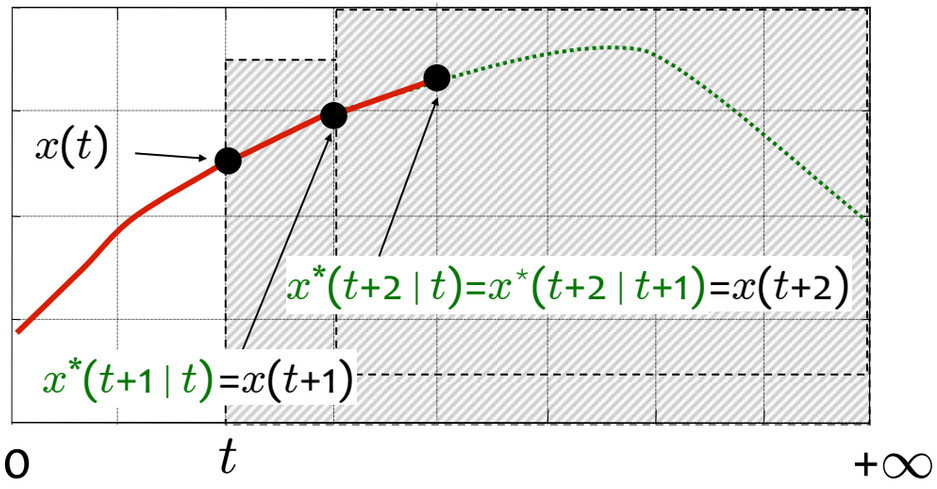
- Even assuming perfect model & no disturbances:



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

PREDICTED AND ACTUAL TRAJECTORIES

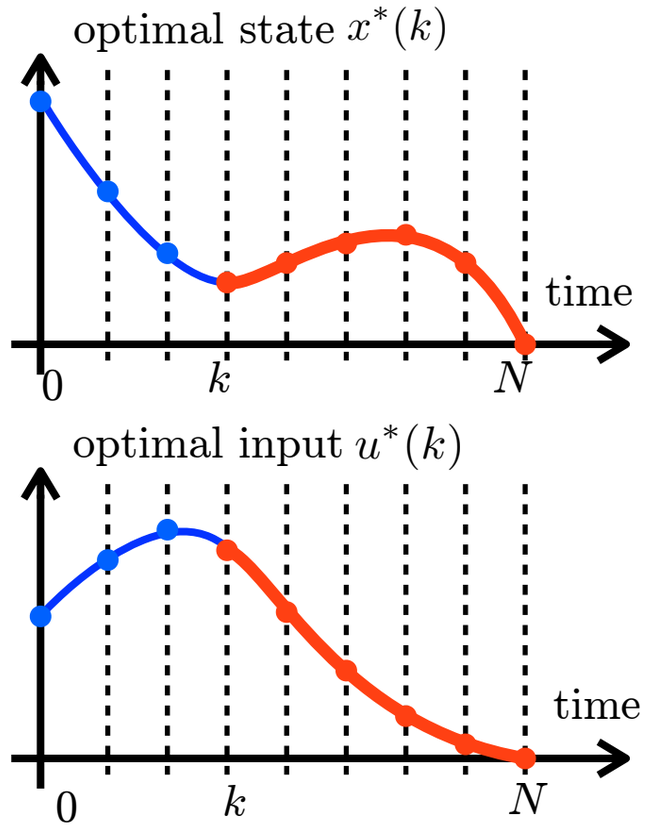
- Special case: for **infinite horizon**, open-loop trajectories and closed-loop trajectories coincide.



Richard Bellman
(1920 - 1984)

- This follows by Bellman's **principle of optimality:**

Given the optimal sequence $u^*(0),\ldots,u^*(N\text{-}1)$ and the corresponding optimal trajectory $x(0),x^*(1)\ldots,x^*(N)$ the subsequence $u^*(k),\ldots,u^*(N\text{-}1)$ is optimal for the problem on the horizon $[k,N]$, starting from the optimal state $u^*(k)$



"An optimal policy has the property that, regardless of the decisions taken to enter a particular state, the remaining decisions made for leaving that stage must constitute an optimal policy." (Bellman, 1957)

CONVERGENCE AND STABILITY

$$\begin{aligned} \min_{z} \quad & x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k \\ \text{s.t.} \quad & u_{\min} \le u_k \le u_{\max}, \;\; k = 0, \ldots, N-1 \\ & y_{\min} \le y_k \le y_{\max}, \;\; k = 1, \ldots, N \end{aligned}$$

QUADRATIC PROGRAM (QP)

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

- 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

BASIC CONVERGENCE PROPERTIES

Theorem. 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 optimizing

$$V^*(x(t)) = \min \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k$$

$$\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k$$

$$u_{\min} \le u_k \le u_{\max}$$

$$y_{\min} \le Cx_k \le y_{\max}$$

$$x_N = 0 \quad \text{“terminal constraint”}$$

with $R,Q>0$. If the optimization problem **is feasible at time** $t=0$ then

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

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

and the constraints are satisfied at all time $t\ge 0$, for all $R,Q>0$

(Keerthi and Gilbert, 1988) (Bemporad, Chisci, Mosca, 1994)

For general **stability result** see (Lazar, Heemels, Weiland, Bemporad, IEEE TAC, 2006)

CONVERGENCE PROOF

Main idea: Use **value function** $V^*(x(t))$ as a **Lyapunov** function

- Let z_t = optimal control sequence at time t, $z_t = [u_0^t \ \dots \ u_{N-1}^t]'$
- By construction $\bar{z}_{t+1} = [u_1^t \ \dots \ u_{N-1}^t \ 0]'$ is a feasible sequence at time $t+1$
- The cost of $\bar{z}_{t+1}$ is $V^*(x(t)) - x'(t)Qx(t) - u'(t)Ru(t) \geq V^*(x(t+1))$
- $V^*(x(t))$ is monotonically decreasing and ≥ 0, so $\exists \lim_{t\to\infty} V^*(x(t)) \triangleq V_\infty$
- Hence $0 \leq x'(t)Qx(t) + u'(t)Ru(t) \leq V^*(x(t)) - V^*(x(t+1)) \to 0$ for $t \to \infty$
- Since $R, Q > 0$, $\lim_{t\to\infty} x(t) = 0$, $\lim_{t\to\infty} u(t) = 0$ □

Global optimum is not needed to prove convergence !

STABILITY CONSTRAINTS

1. No stability constraint, infinite prediction horizon: $N \to \infty$

(Keerthi and Gilbert, 1988) (Rawlings and Muske, 1993) (Bemporad, Chisci, Mosca, 1994)

2. End-point constraint: $x_N = 0$

(Kwon and Pearson, 1977)(Keerthi and Gilbert, 1988) (Bemporad, Chisci, Mosca, 1994)

3. Relaxed terminal constraint: $x_N \in \Omega$

(Scokaert and Rawlings, 1996)

4. Contraction constraint: $\|x_{k+1}\| \leq \alpha \|x(t)\|, \;\; \alpha < 1$

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

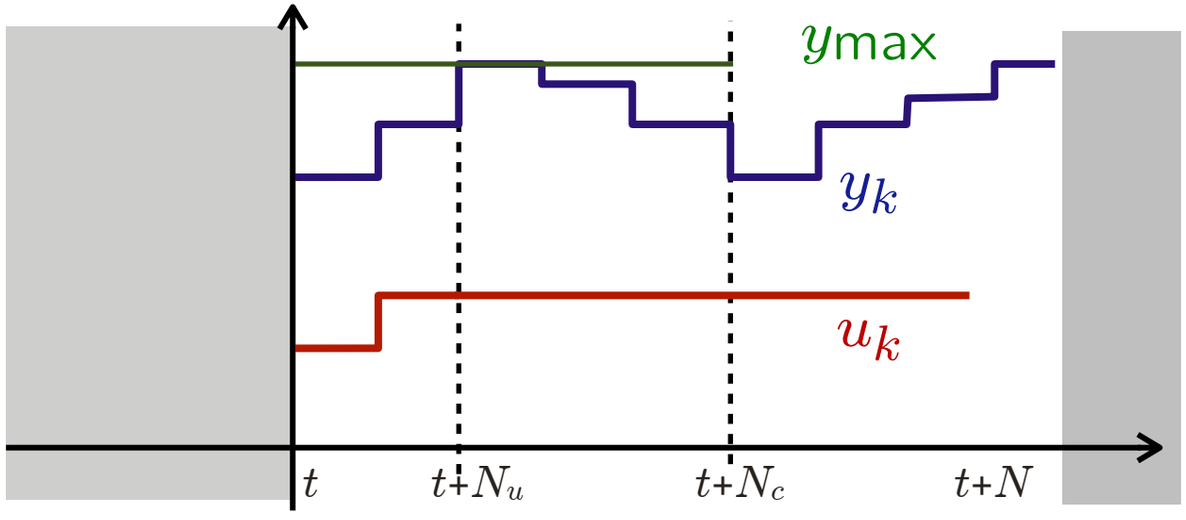
All the proofs in (1,2,3) use the value function $V^*(x(t)) = \min_z J(z, x(t))$ as a Lyapunov function

REDUCING COMPLEXITY: DIFFERENT PREDICTION HORIZONS

$$\begin{aligned}
\min_{z} \quad & \sum_{k=0}^{N-1} \|W^y(y_k - r(t))\|^2 + \|W^{\Delta u}\Delta u_k\|^2 + \rho_\epsilon \epsilon^2 \\
\text{subj. to} \quad & u_{\min} \le u_k \le u_{\max}, \;\; k = 0, \ldots, N-1 \\
& \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}, \;\; k = 0, \ldots, N-1 \\
& \Delta u_k = 0, \;\; k = N_u, \ldots, N-1 \\
& y_{\min} - \epsilon V_{\min} \le y_k \le y_{\max} + \epsilon V_{\max}, \;\; k = 1, \ldots, N_c,
\end{aligned}$$

- The **input horizon** N_u limits the number of free variables
  - Loss of performance
  - Decreased computation time (QP is smaller, less primal variables)

  typically N_u=1÷10



- The **output constraint horizon** N_c limits the number of constraints
  - Higher chances of violating output constraints
  - Decreased computation time (QP is smaller, less dual variables)

MPC AND LINEAR QUADRATIC REGULATION (LQR)

- Special case: $J(z, x_0) = \min_z x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k$

with matrix P solving the Algebraic Riccati Equation

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

Jacopo Francesco Riccati (1676 - 1754)

- **(unconstrained) MPC = LQR** (for any choice of the prediction horizon N)

Proof. Easily follows from Bellman's principle of optimality (dynamic programming): $x'_N P x_N$ = optimal "cost-to-go" from time N to ∞.

MPC AND LINEAR QUADRATIC REGULATION (LQR)

- Consider again the constrained MPC law based on minimizing

$$\min_z \quad x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k$$

$$\begin{aligned} \text{s.t.} \quad & u_{\min} \le u_k \le u_{\max}, \;\; k = 0, \ldots, N-1 \\ & y_{\min} \le y_k \le y_{\max}, \;\; k = 1, \ldots, N \\ & u_k = K x_k, \;\; k = N_u, \ldots, N-1 \end{aligned}$$

- Choose matrix P and terminal gain K by solving the LQR problem

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

- In a polyhedral region around the origin **constrained MPC = constrained LQR** (for any choice of the prediction and control horizons N,N_u)

  (Sznaier & Damborg, 1987) (Chmielewski & Manousiouthakis, 1996) (Scokaert & Rawlings, 1998) (Bemporad, Morari, Dua Pistikopoulos, 2002)

- 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: $\min \sum_{k=0}^{\infty} y_k^2 + \frac{1}{100} u_k^2$

LQ gain

$$u_k = K_{LQ} x_k, \quad \forall k \ge N_u$$

$$N_u = N = 2$$

$$\min \left(\sum_{k=0}^{1} y_k^2 + \frac{1}{100} u_k^2 \right) + x_2' \begin{bmatrix} 2.1429 & 1.2246 \\ 1.2246 & 1.3996 \end{bmatrix} x_2$$

solution of algebraic Riccati equation

- Optimization problem

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

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

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

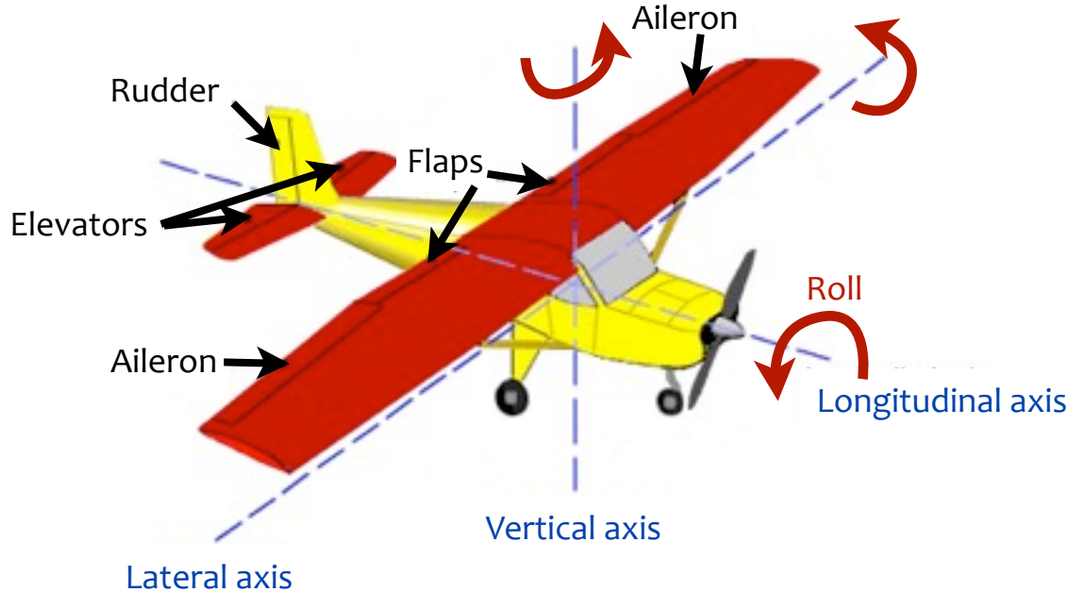
EXAMPLE: AFTI-16

- Linearized model:

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

Yaw Pitch

- 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: $-7.6636, -0.0075 \pm 0.0556j, 5.4530$)



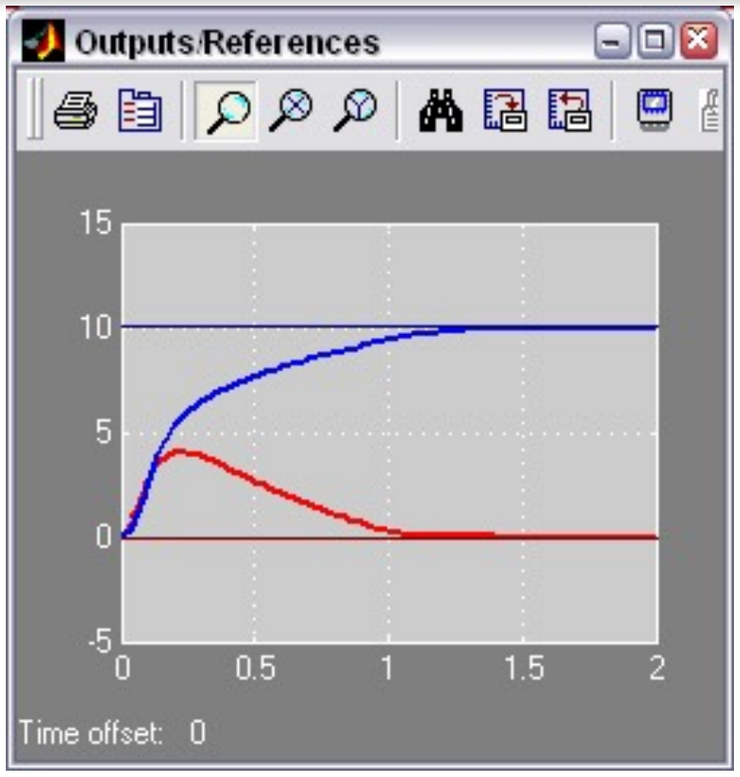
go to demo **/demos/linear/afti16.m**

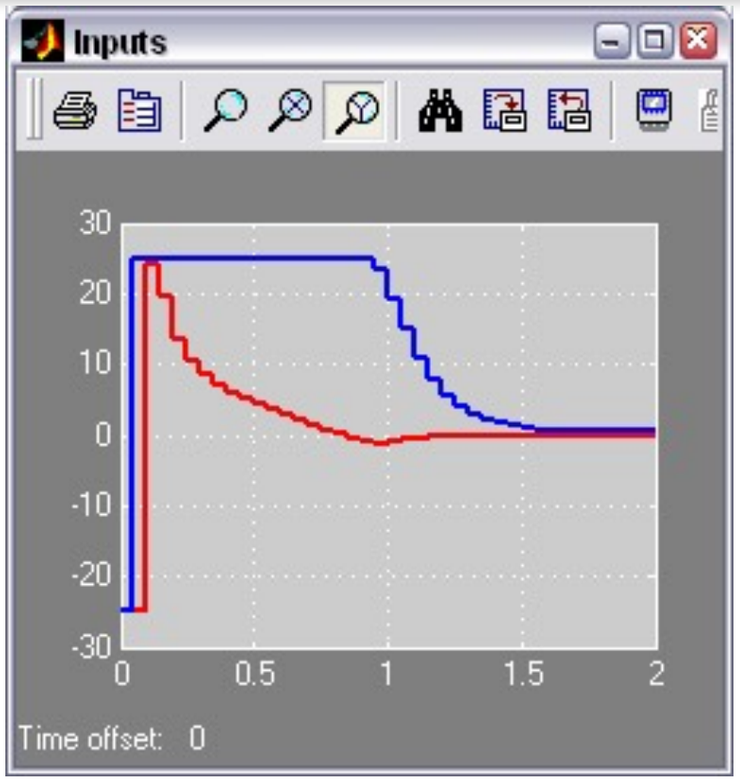
afti16.m

(Hyb-Tbx)

(MPC-Tbx)

EXAMPLE: AFTI-16

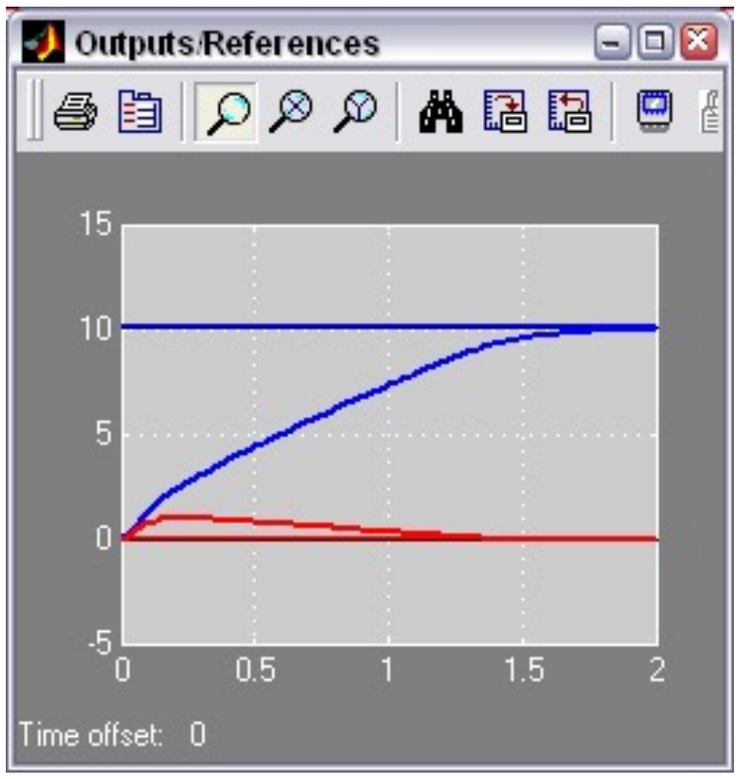


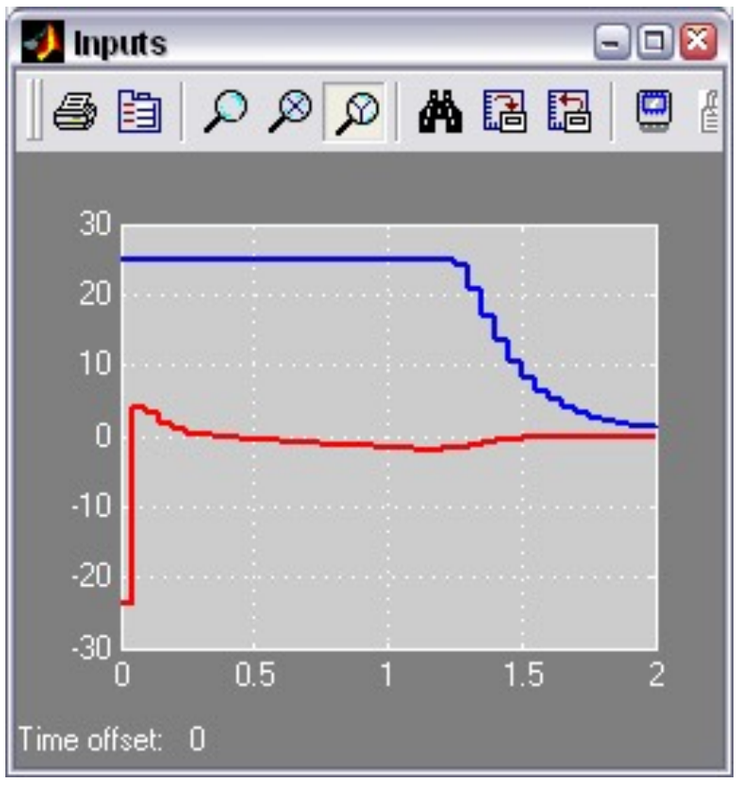


$$N_y = 10,\ N_u = 3,$$

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

$$u_{\min} = -25°,\ u_{\max} = 25°$$



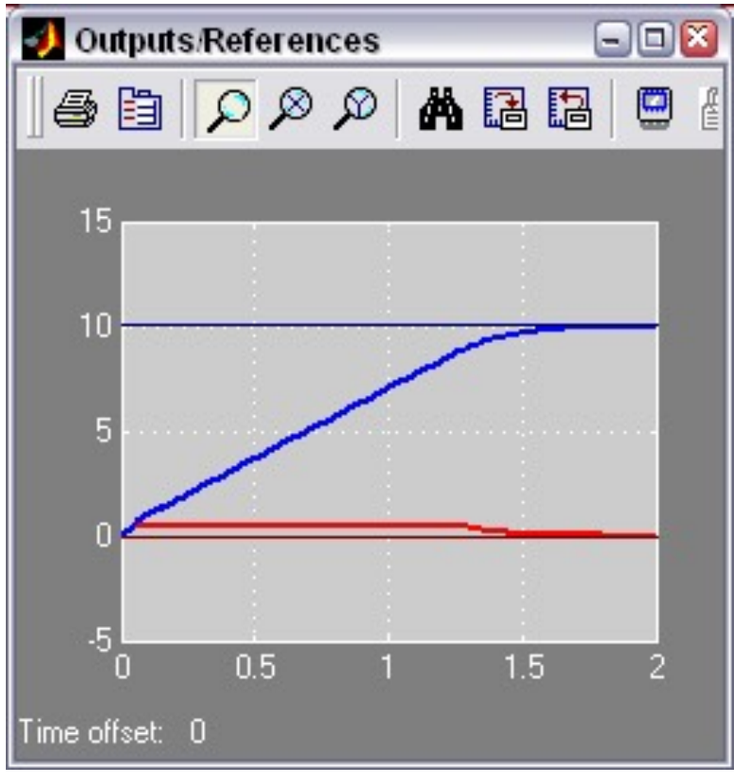


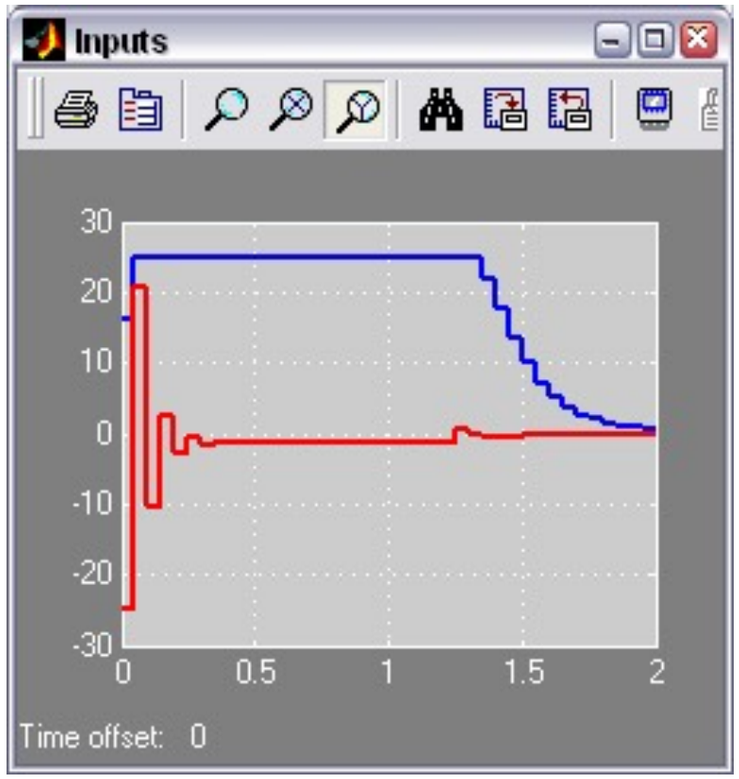
$$N_y = 10,\ N_u = 3,$$

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

$$u_{\min} = -25°,\ u_{\max} = 25°$$

EXAMPLE: AFTI-16





$$N_y = 10,\ N_u = 3,$$

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

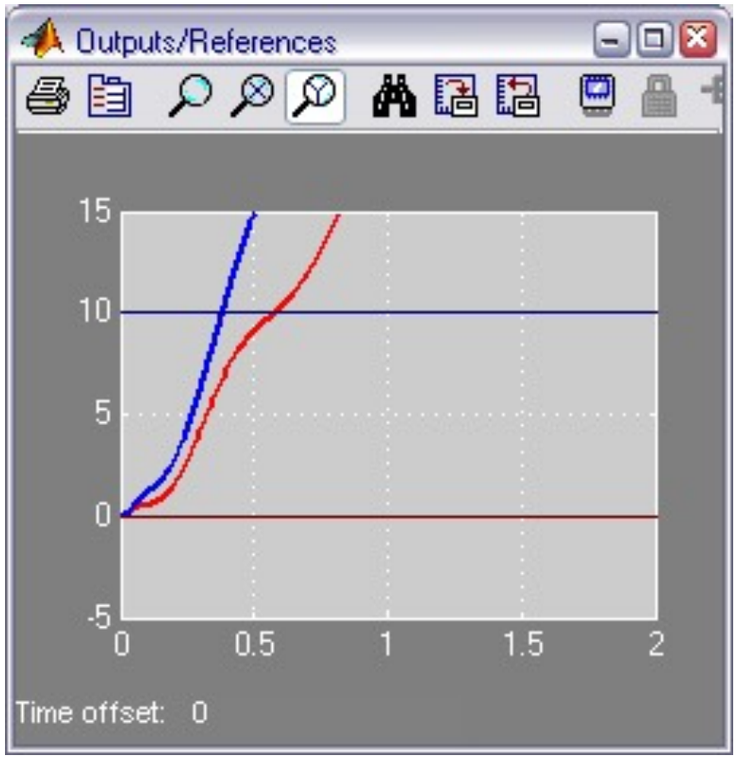
$$u_{\min} = -25°,\ u_{\max} = 25°,$$

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

EXAMPLE: AFTI-16

Unconstrained MPC (=linear controller, ≈ LQR)

+ actuator saturation ±25°



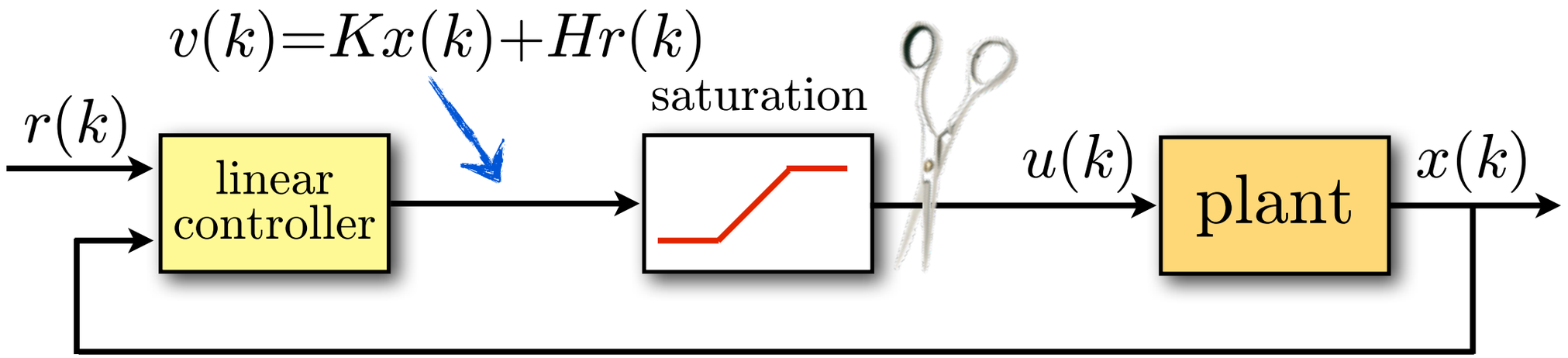
$$N_y = 10,\ N_u = 3,$$
$$w_y = \{10, 10\},\ w_{\delta u} = \{.01, .01\},$$

UNSTABLE !!!

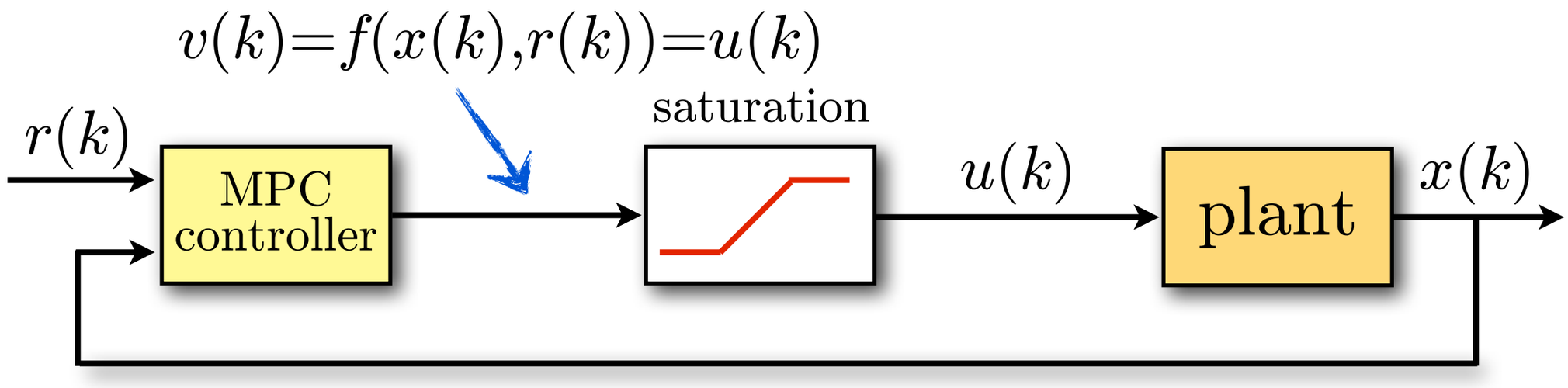
Saturation needs to be considered in the control design !

SATURATION

- Saturation is dangerous because it breaks the control loop



- MPC takes it into account automatically (and optimally)



TUNING GUIDELINES

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

$$\text{subj. to} \quad \begin{aligned} &\Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}, \ k = 0, \dots, N_u - 1 \\ &\Delta u_k = 0, \ k = N_u, \dots, N-1 \\ &u_{\min} \le u_k \le u_{\max}, \ k = 0, \dots, N_u - 1 \\ &y_{\min} - \epsilon V_{\min} \le y_k \le y_{\max} + \epsilon V_{\max}, \ k = 1, \dots, N_c \end{aligned}$$

- **Weights:** the larger the ratio $W^y/W^{\Delta u}$ the more aggressive the controller
- **Input horizon:** the larger N_u , the more "optimal" but more complex the controller
- **Prediction horizon:** the smaller N, the more aggressive the controller
- **Constraints horizon:** the smaller N_c , the simpler the controller
- **Limits:** controller less aggressive if $\Delta u_{\min}$, $\Delta u_{\max}$ are small
- **Penalty** ρ_ε**:** pick up smallest ρ_ε that keeps soft constraints reasonably satisfied

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: $y_1 \in [\text{-1e-4}, \text{1e-4}]$ (V)

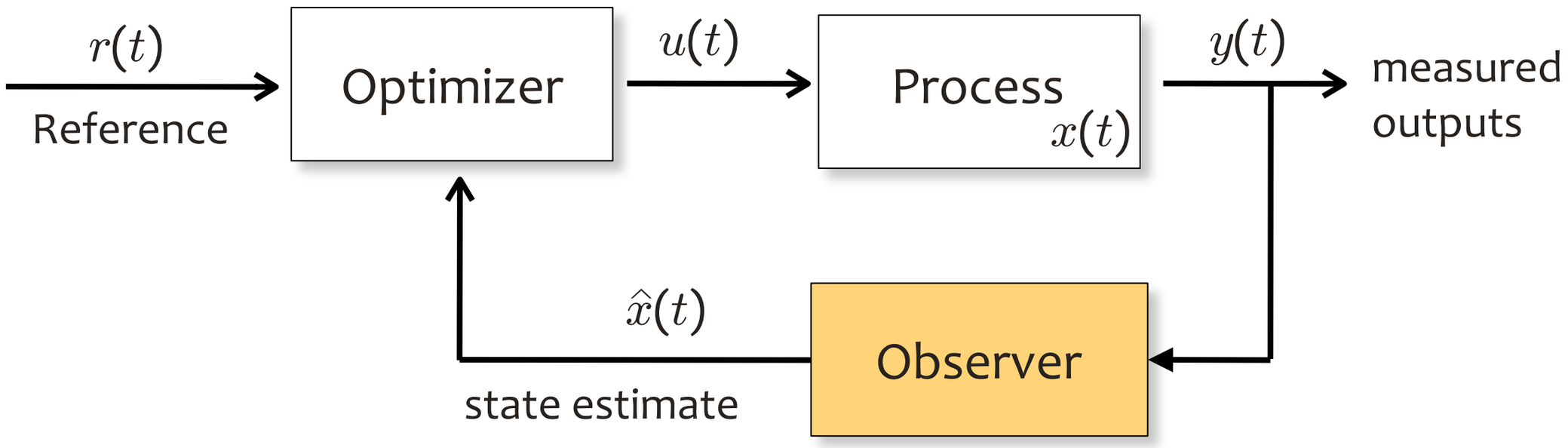
$y_2 \in [\text{-1e4}, \text{1e4}]$ (Pa)

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

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

- Ideally all variables should be in $[\text{-1}, \text{1}]$. For example, one can replace y with y/y_{max}
- Scaling also possible after formulating the QP problem (a.k.a. "preconditioning")

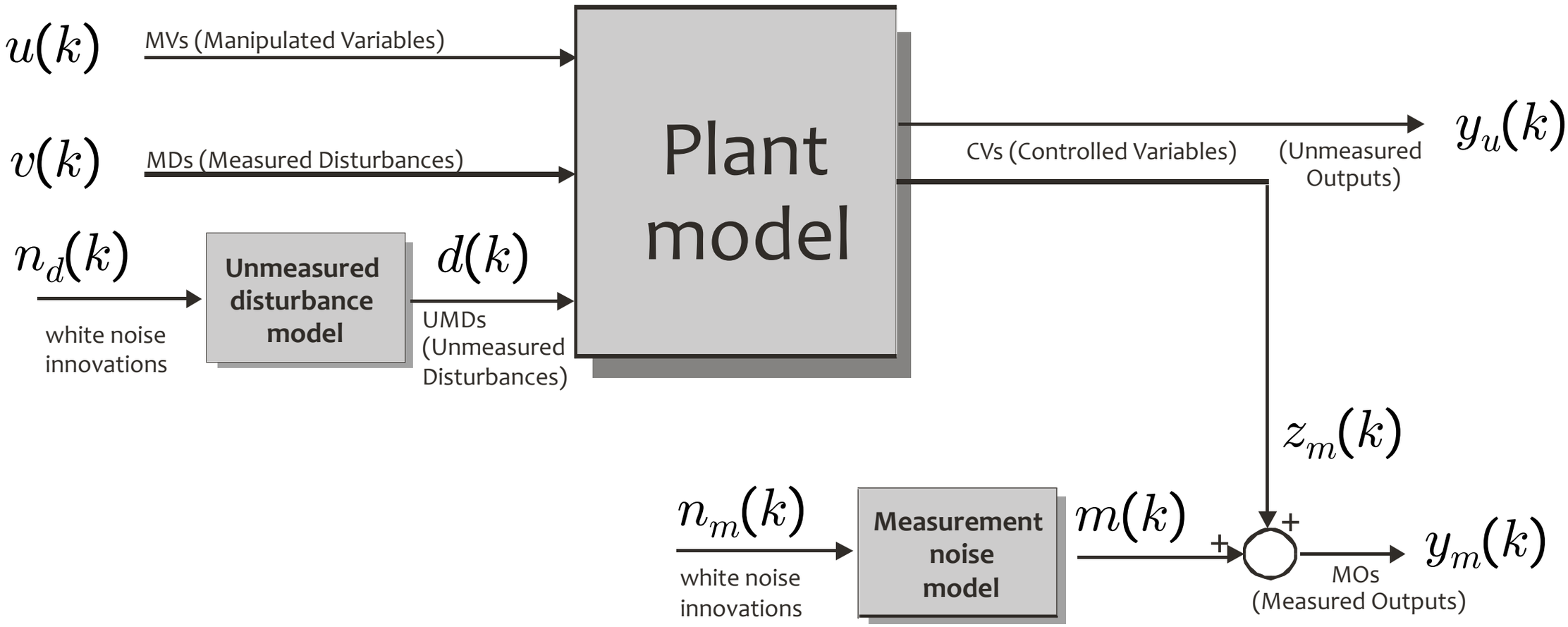
OBSERVER DESIGN FOR MPC



- Full state $x(t)$ of process may not be available, only outputs $y(t)$
- Even if $x(t)$ is available, noise should be filtered out
- Prediction and process models may be different !
  (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: measurement noise model not needed for optimization !

KALMAN FILTER DESIGN

Rudolf Emil Kalman (1930 - 2016)

- Full model for designing Kalman filter

$$\begin{bmatrix} x_p(k+1) \\ x_d(k+1) \\ x_m(k+1) \end{bmatrix} = \begin{bmatrix} A & B_d\bar{C} & 0 \\ 0 & \bar{A} & 0 \\ 0 & 0 & \tilde{A} \end{bmatrix} \begin{bmatrix} x_p(k) \\ x_d(k) \\ x_m(k) \end{bmatrix} + \begin{bmatrix} B_u \\ 0 \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} B_v \\ 0 \\ 0 \end{bmatrix} v(k) +$$
$$\begin{bmatrix} B_d\bar{D} \\ \bar{B} \\ 0 \end{bmatrix} n_d(k) + \begin{bmatrix} 0 \\ 0 \\ \tilde{B} \end{bmatrix} n_m(k) + \begin{bmatrix} B_u \\ 0 \\ 0 \end{bmatrix} n_u(k)$$
$$y_m(k) = \begin{bmatrix} C_m & D_{dm}\bar{C} & \tilde{C} \end{bmatrix} \begin{bmatrix} x_p(k) \\ x_d(k) \\ x_m(k) \end{bmatrix} + D_{vm}v(k) + \bar{D}_m n_d(k) + \tilde{D}_m n_m(k)$$

- $n_d(k)$: 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

OBSERVER DESIGN

- Measurement update

$$\begin{aligned} y_m(k|k-1) &= C_m x(k|k-1) \\ x(k|k) &= x(k|k-1) + M(y_m(k) - y_m(k|k-1)) \end{aligned}$$

- Time update

$$x(k+1|k) = Ax(k|k-1) + Bu(k) + L(y_m(k) - y_m(k|k-1))$$

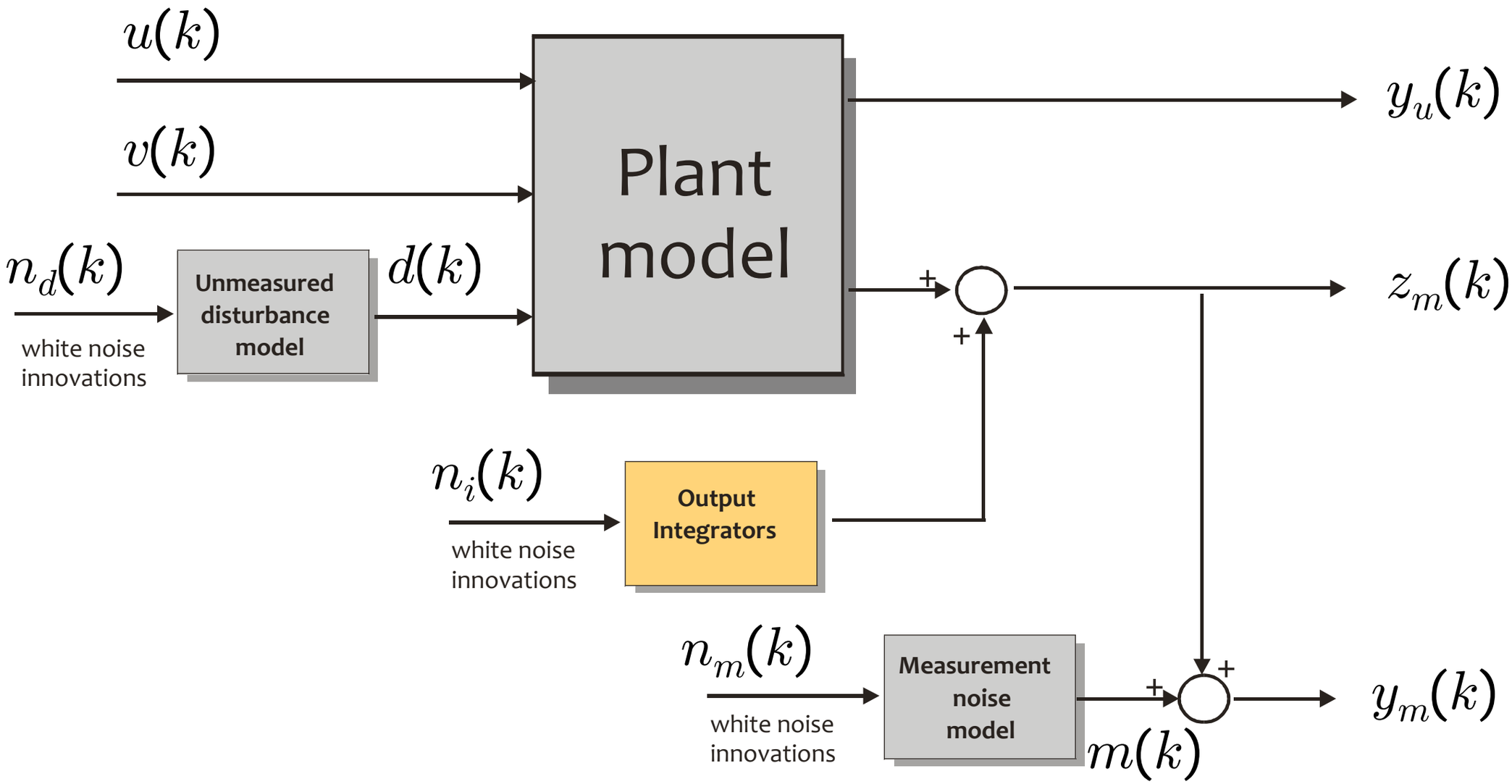
- NOTE: **separation principle** holds ! (under certain assumptions)

(Muske, Meadows, Rawlings, ACC94)

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

- Add model for constant unknown disturbance

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + B_d d_k \\ d_{k+1} = d_k \\ y_k = Cx_k + D_d d_k \end{cases}$$

(Pannocchia, Rawlings, 2003)

special case: **output integrators**

$$B_d = 0, \;\; D_d = I$$

- Use the extended model to design a state observer (e.g., Kalman filter) that estimates both the state $\hat{x}(t)$ and disturbance $d(t)$ from $y(t)$

- **Main idea:** observer makes $y(t) - (C\hat{x}(t) + D_d\hat{d}(t)) \to 0$ (estimation error)

  MPC makes $C\hat{x}(t) + D_d\hat{d}(t) \to r(t)$ (predicted tracking error)

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

- **Explanation:** in steady state, the term $D_d\hat{d}(t)$ compensates for model mismatch (=same effect of integral action, but at the observer level)

ERROR FEEDBACK

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

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

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

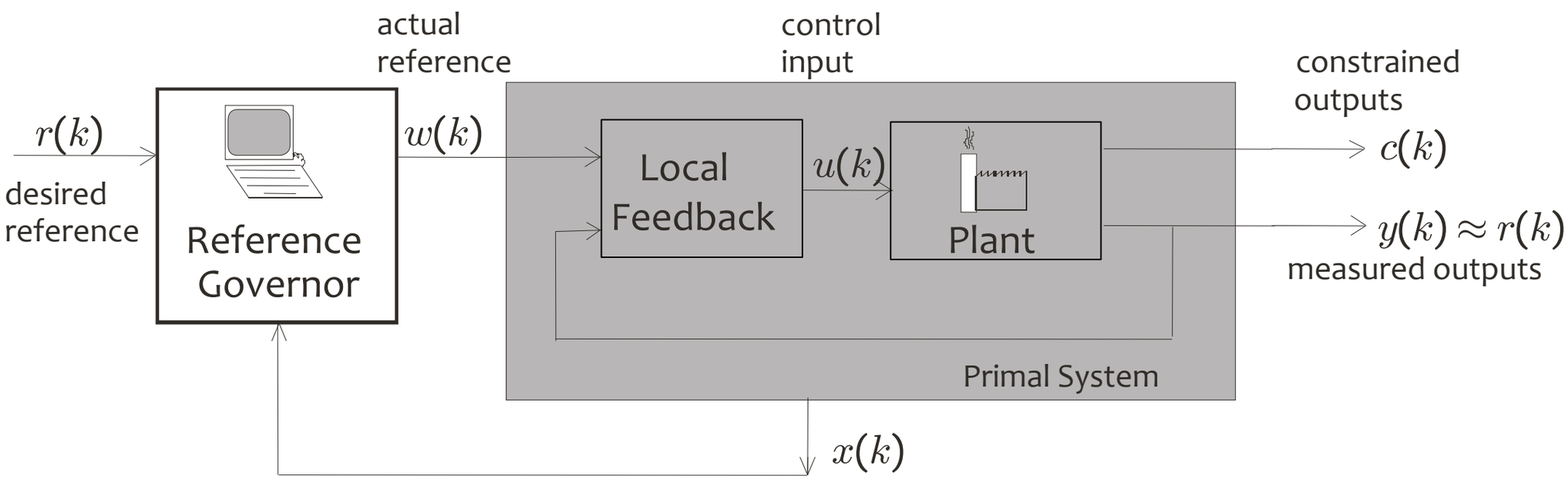
Alternative: treat $r(k)$ as a measured disturbance affecting the dynamics of q

- Explanation: if closed-loop asymptotically stable, then $q(k)$ converges to a constant, and hence $y(k) \to r(k)$

REFERENCE / COMMAND GOVERNOR

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

(Bemporad, Mosca, 1994) (Gilbert, Kolmanovsky, Tan, 1994)
(Garone, Di Cairano, Kolmanovsky, 2016)



- 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*

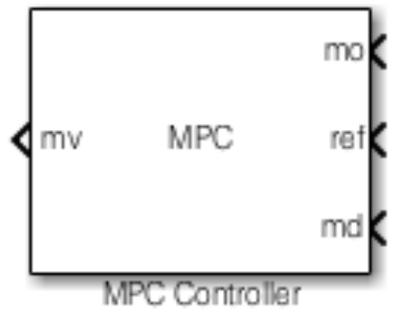
Objective:

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

MODEL PREDICTIVE CONTROL TOOLBOX

(Bemporad, Ricker, Morari, 1998-present)

- Several **linear MPC** design features available:



– explicit MPC
– time-varying/adaptive models, weights, constraints
– stability/frequency analysis of closed-loop (inactive constraints)
– …

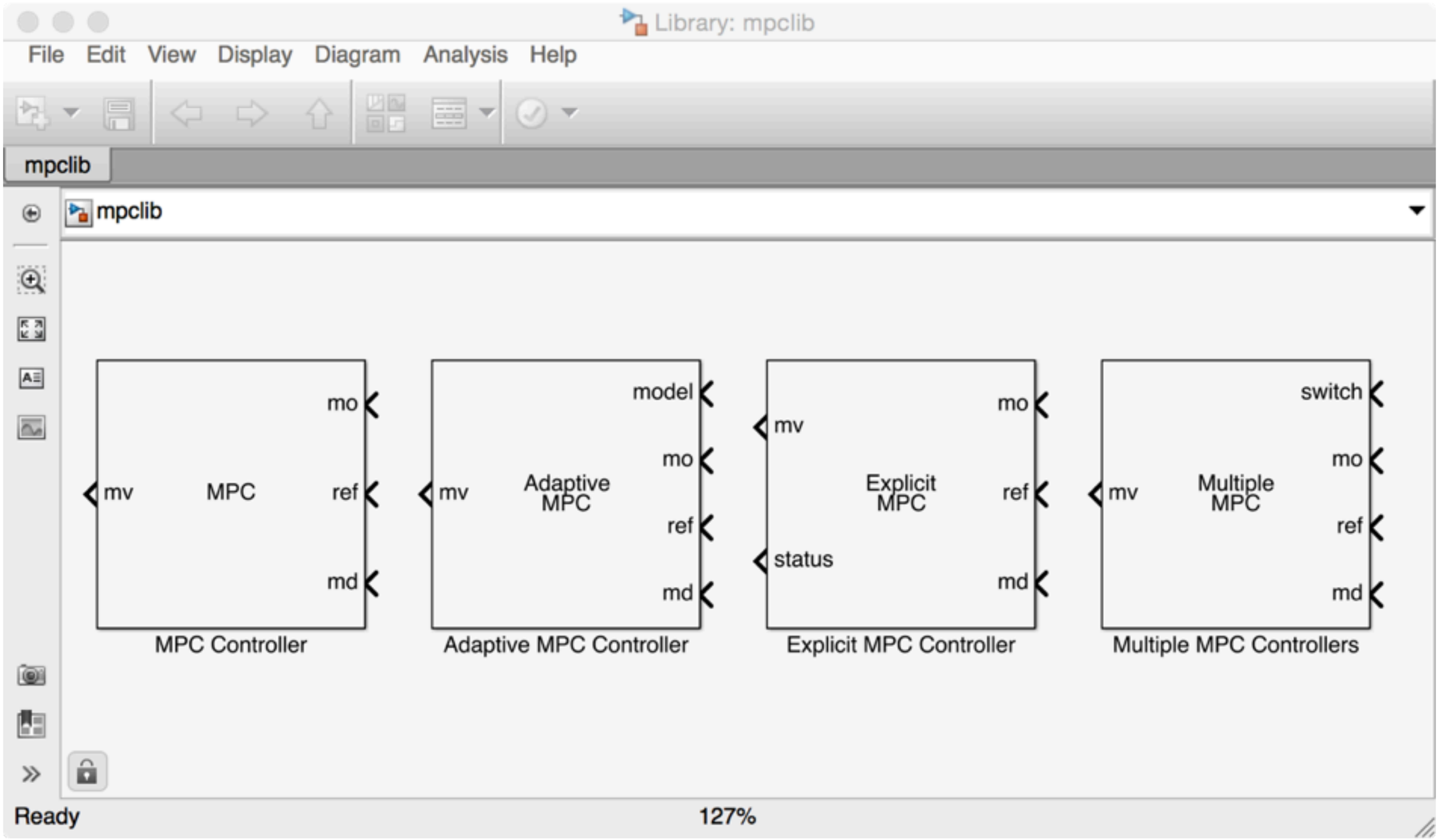
- Prediction models can be generated by the Identification Toolbox or automatically linearized from Simulink

- Fast command-line/Simulink closed-loop simulation (compiled EML-code)

- Graphical User Interface

- Simulink library (compiled EML-code)

MPC SIMULINK LIBRARY




```
>> mpclib
```

MPC SIMULINK BLOCK

Function Block Parameters: MPC Controller

MPC (mask) (link)

The MPC Controller block lets you design and simulate a model predictive controller defined in the Model Predictive Control Toolbox.

Parameters

MPC Controller: mpcobj — Design

Initial Controller State: [] — Review

Block Options

General | Online Features | Others

Additional Inports

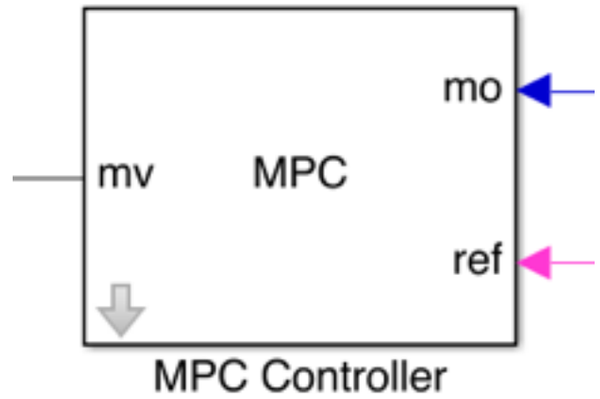
- [ ] Measured disturbance (md)
- [ ] External manipulated variable (ext.mv)

Additional Outports

- [ ] Optimal cost (cost)
- [ ] Optimal control sequence (mv.seq)
- [ ] Optimization status (qp.status)
- [ ] Estimated plant, disturbance and noise model states (est.state)

State Estimation

- [ ] Use custom estimated states instead of measured outputs (x[k|k])



MPC SIMULINK BLOCK

Function Block Parameters: MPC Controller

MPC (mask) (link)

The MPC Controller block lets you design and simulate a model predictive controller defined in the Model Predictive Control Toolbox.

Parameters

MPC Controller: mpcobj — Design

Initial Controller State: [] — Review

Block Options

General | Online Features | Others

Constraints

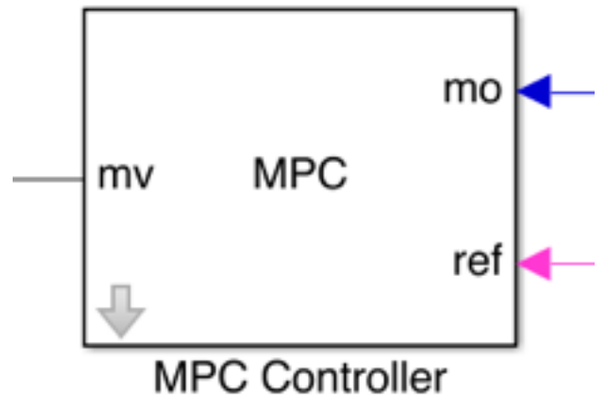
- [ ] Plant input and output limits (umin, umax, ymin, ymax)

Weights

- [ ] Weights on plant outputs (y.wt)
- [ ] Weights on manipulated variables (u.wt)
- [ ] Weights on manipulated variable changes (du.wt)
- [ ] Weight on overall constraint softening (ecr.wt)

MV Targets

- [ ] Targets for manipulated variables (mv.target)



MPC SIMULINK BLOCK

Function Block Parameters: MPC Controller

MPC (mask) (link)

The MPC Controller block lets you design and simulate a model predictive controller defined in the Model Predictive Control Toolbox.

Parameters

MPC Controller: mpcobj — Design

Initial Controller State: [] — Review

Block Options

General | Online Features | Others

Signal Attributes

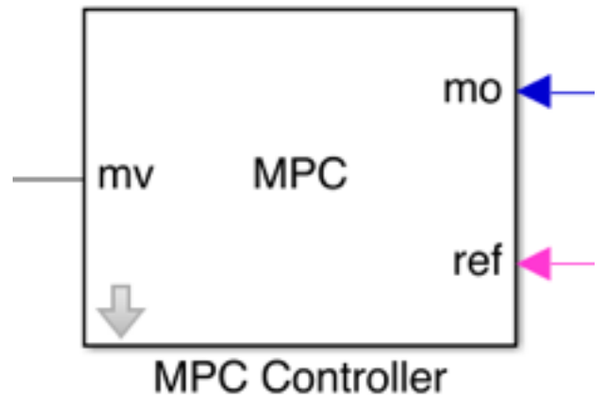
Block Data Type: double >>

Sample Time

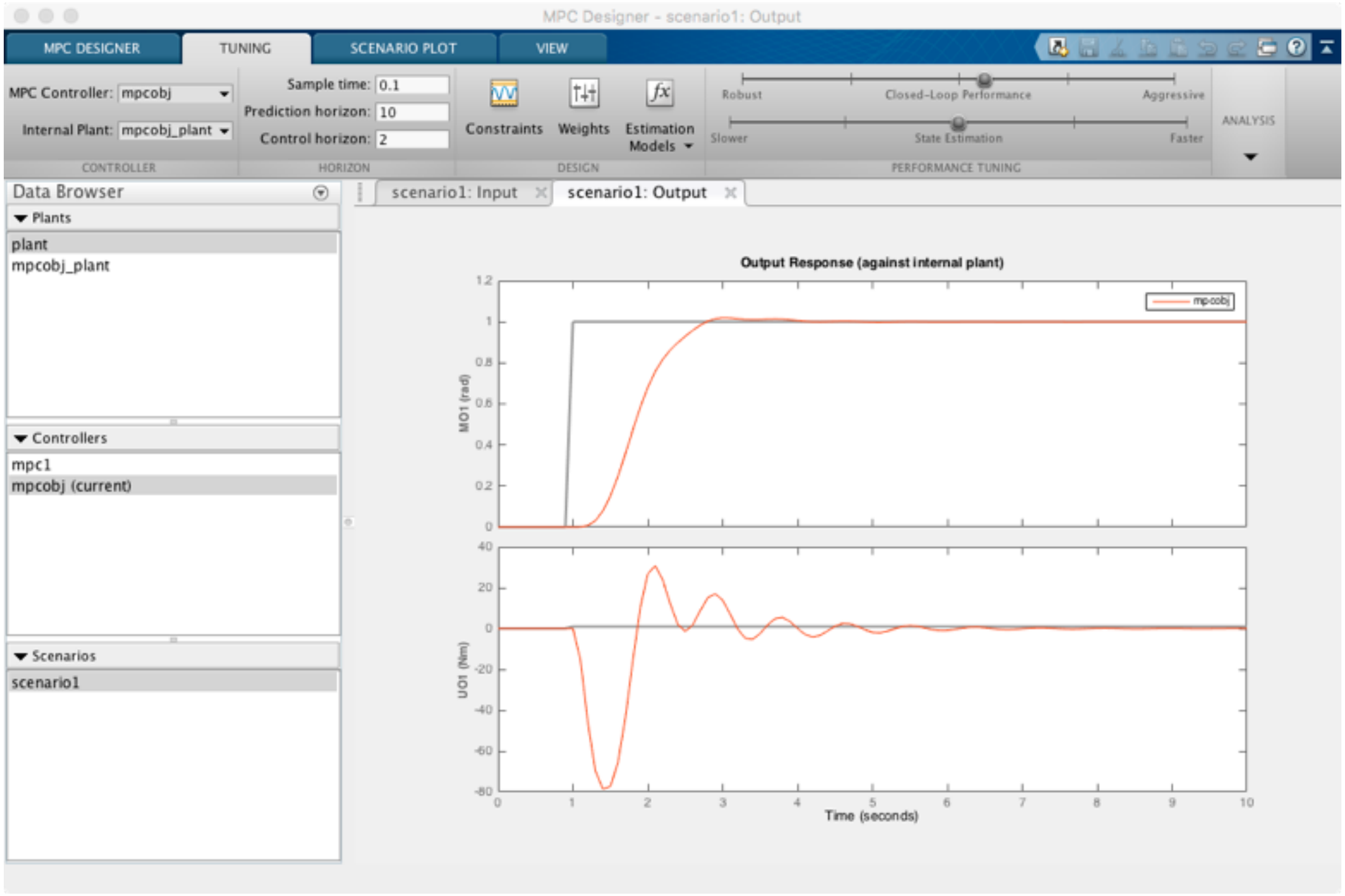
Inherit sample time (must be the same as controller sample time)

Optimization Settings

Use external signal to enable or disable optimization (switch)



MPC GRAPHICAL USER INTERFACE

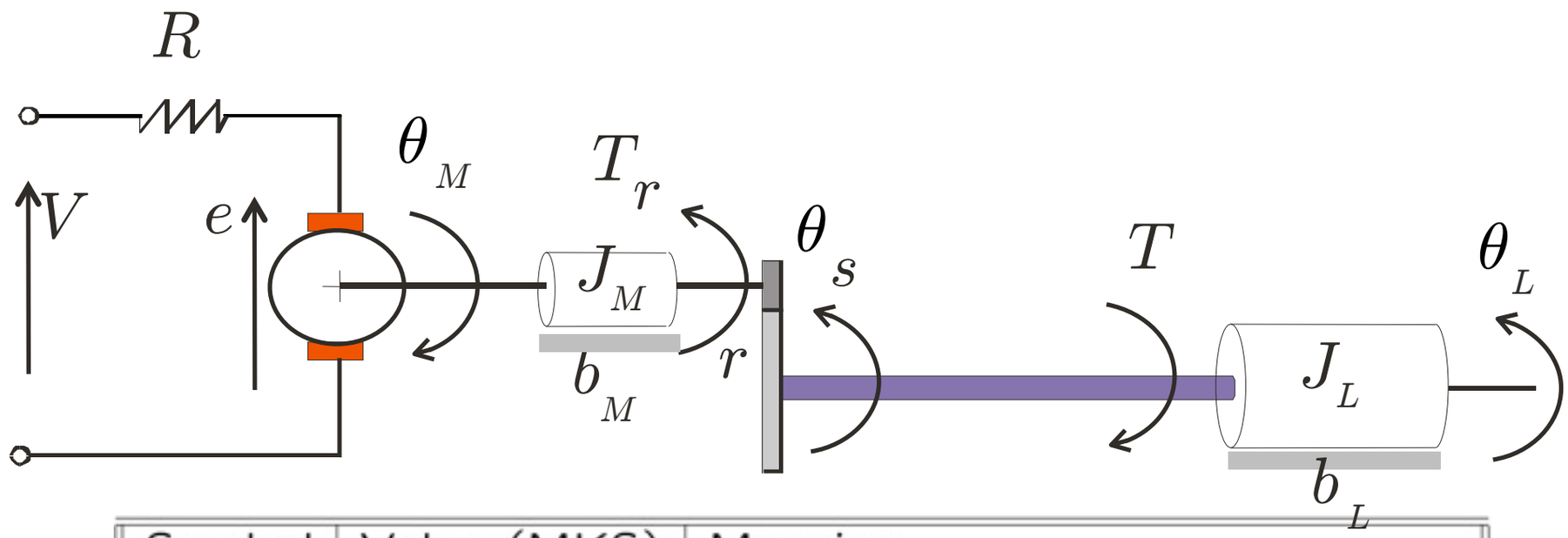



```
>> mpcDesigner
```

(old versions: `>>mpctool`)

See video on Mathworks' web site (click this `link`)

EXAMPLE: MPC OF A DC SERVOMOTOR

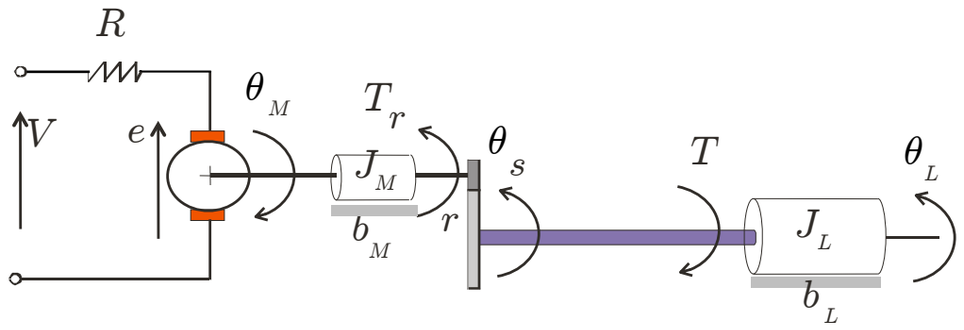


Symbol	Value (MKS)	Meaning
L_S	1.0	shaft length
d_S	0.02	shaft diameter
J_S	negligible	shaft inertia
J_M	0.5	motor inertia
β_M	0.1	motor viscous friction coefficient
R	20	resistance of armature
K_T	10	motor constant
ρ	20	gear ratio
k_θ	1280.2	torsional rigidity
$\hat{J}_L$	$20J_M$	nominal load inertia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

```
>> edit mpcmotor
```

(also /**demos/linear/dcmotor.m** in Hybrid Toolbox)

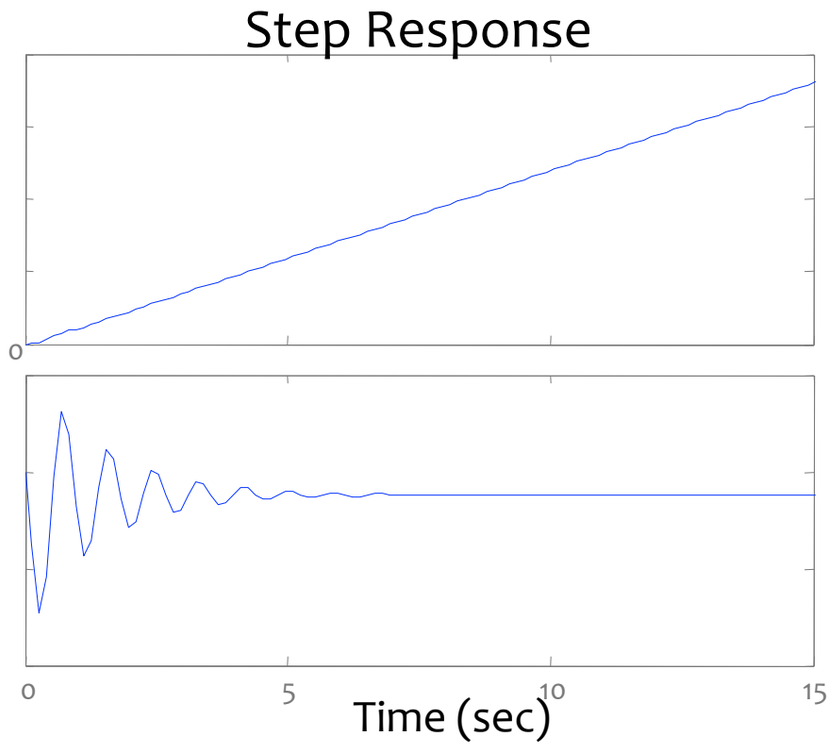
DC SERVOMOTOR MODEL



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

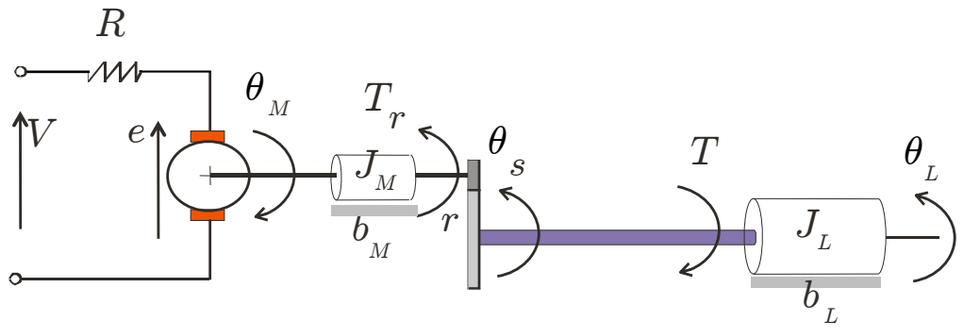
$$\theta_L = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} x$$

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




```
>> [plant, tau] = mpcmotormodel;
>> plant = setmpcsignals(plant,'MV',1,'MO',1,'UO',2);
```

DC SERVOMOTOR SPECIFICATIONS



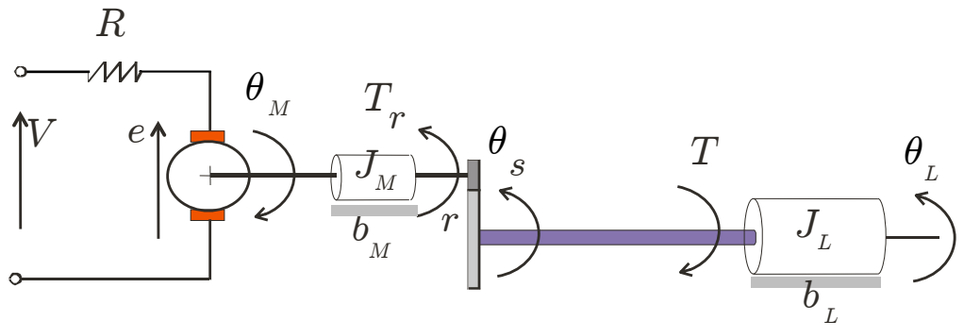
- Finite shear strength of steel shaft ($\tau_{adm} = 50$ N/mm^2)
  $\Rightarrow |T| \leq 78.5398\ Nm$

- DC voltage limits $|V| \leq 220\ V$

- Sampling time: $T_s = .1$ s (+ zero-order hold)

```
>> MV = struct('Min',-220,'Max',220);
>> OV = struct('Min',{-Inf,-tau},'Max',{Inf,tau});
>> Ts = 0.1;
```

DC SERVOMOTOR: MPC SETUP



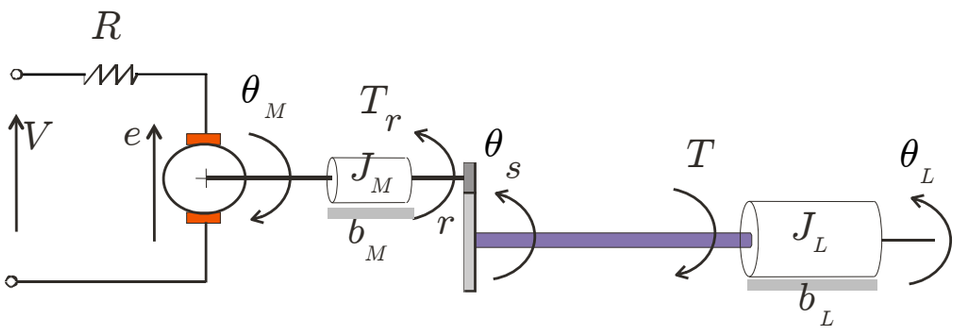
$$\min_{\Delta U} \quad \sum_{k=0}^{p-1} \|W^y(y_{k+1} - r(t))\|^2 + \|W^{\Delta u}\Delta u_k\|^2 + \rho_\epsilon \epsilon^2$$

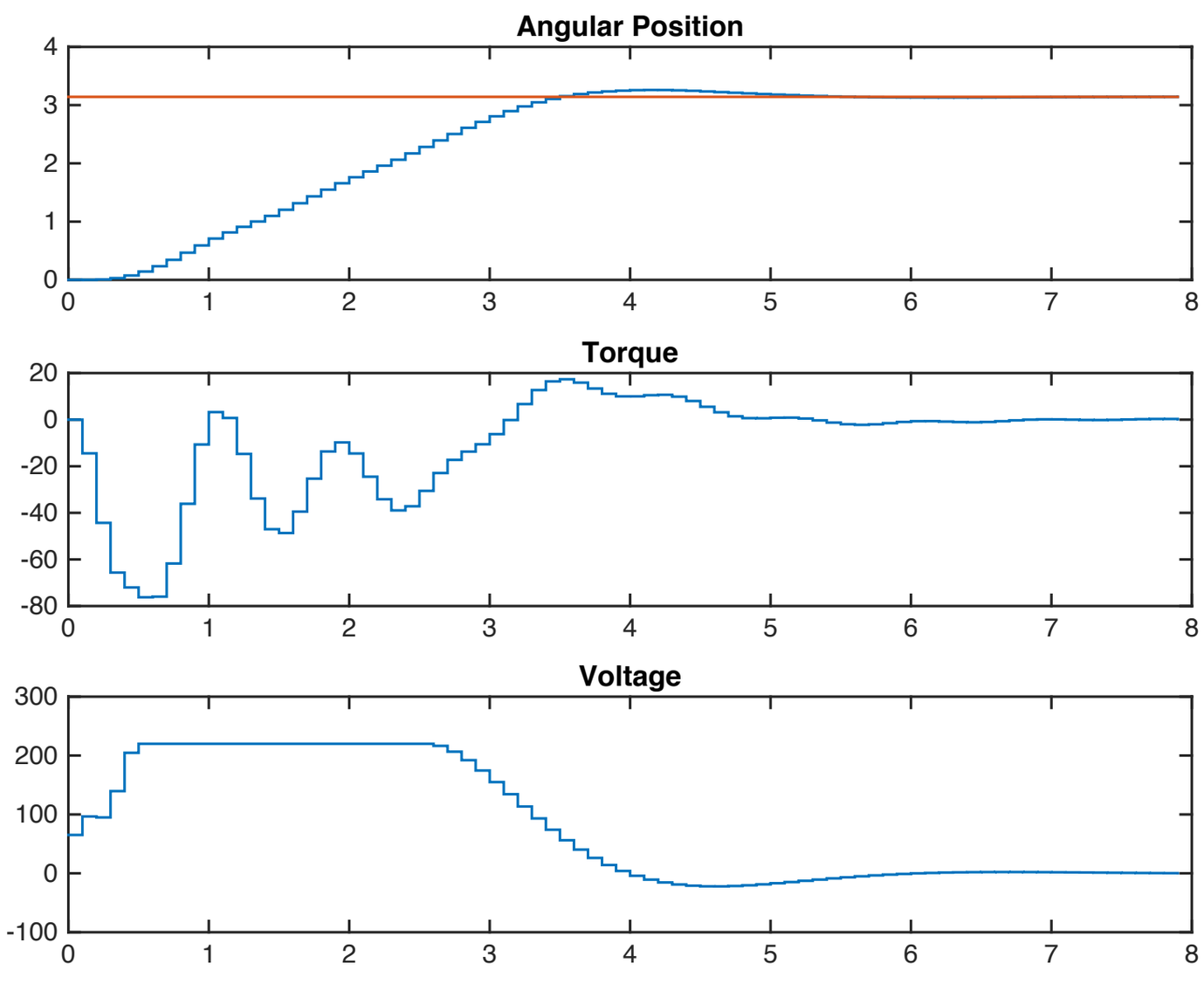
$$\begin{aligned}
\text{subj. to} \quad & \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}, \;\; k = 0, \ldots, m-1 \\
& \Delta u_k = 0, \;\; k = m, \ldots, p-1 \\
& u_{\min} \le u_k \le u_{\max}, \;\; k = 0, \ldots, m-1 \\
& y_{\min} - \epsilon V_{\min} \le y_k \le y_{\max} + \epsilon V_{\max}, \;\; k = 1, \ldots, p
\end{aligned}$$

```
>> Weights = struct('MV',0,'MVRate',0.1,'OV',[0.1 0]);
>> p = 10;
>> m = 2;
>> mpcobj = mpc(plant,Ts,p,m,Weights,MV,OV);
```

DC SERVOMOTOR: CLOSED-LOOP SIMULATION (SIM)

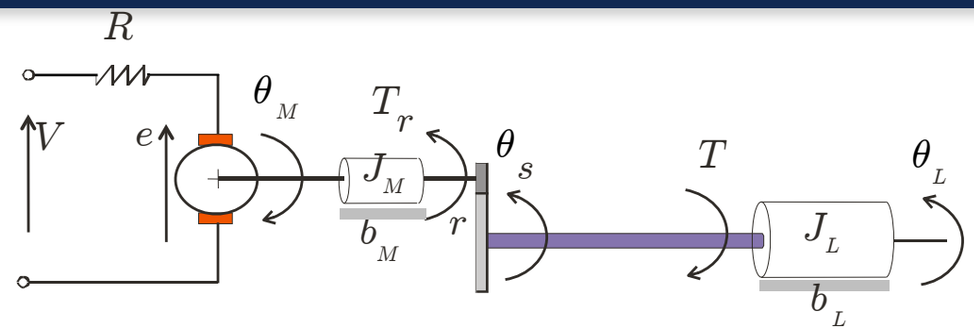
```
>> Tstop = 8;                  % seconds
>> Tf = round(Tstop/Ts);       % simulation iterations
>> r = [pi*ones(Tf,1) zeros(Tf,1)];
>> [y1,t1,u1] = sim(mpcobj,Tf,r);
```

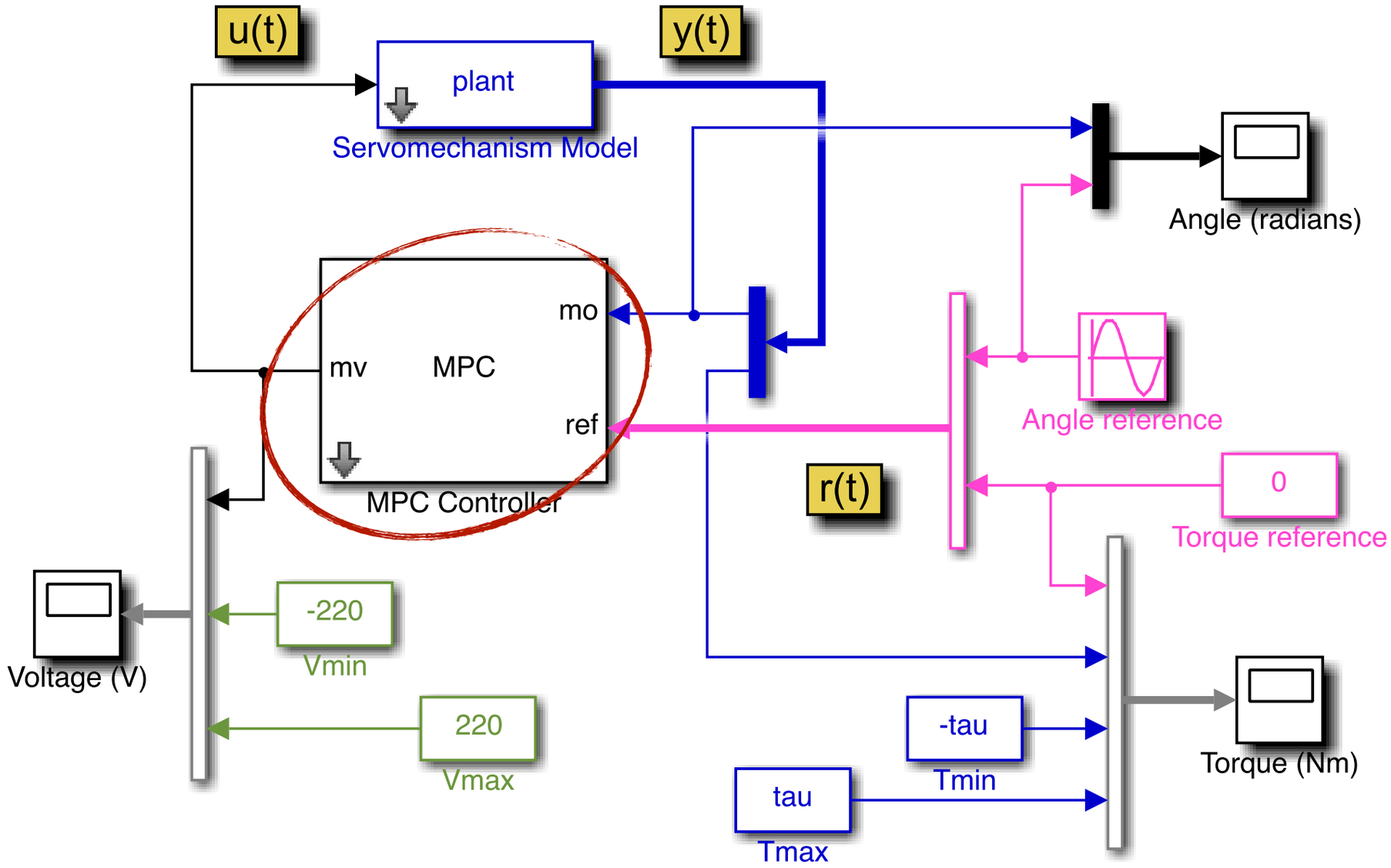




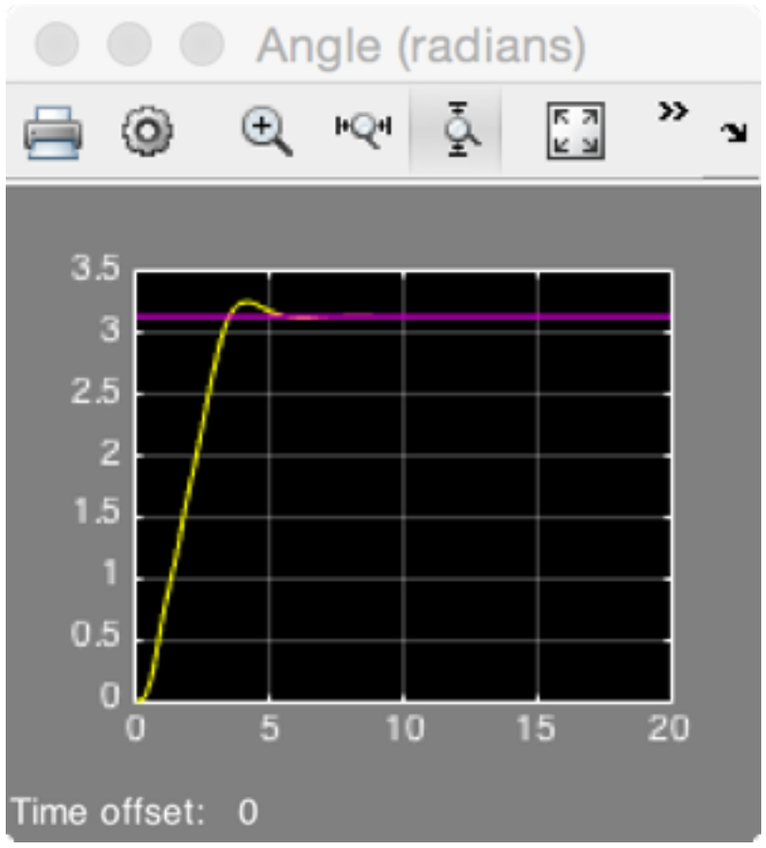
DC SERVOMOTOR: SIMULATION (SIMULINK)

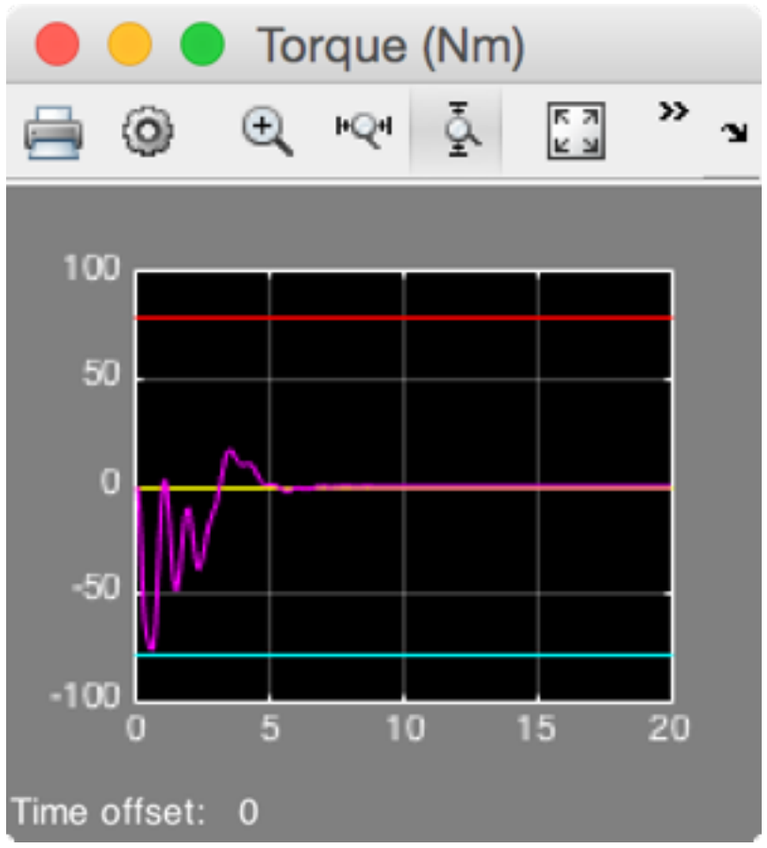
```
>> mdl = 'mpc_motor';
>> open_system(mdl)
>> sim(mdl)
```

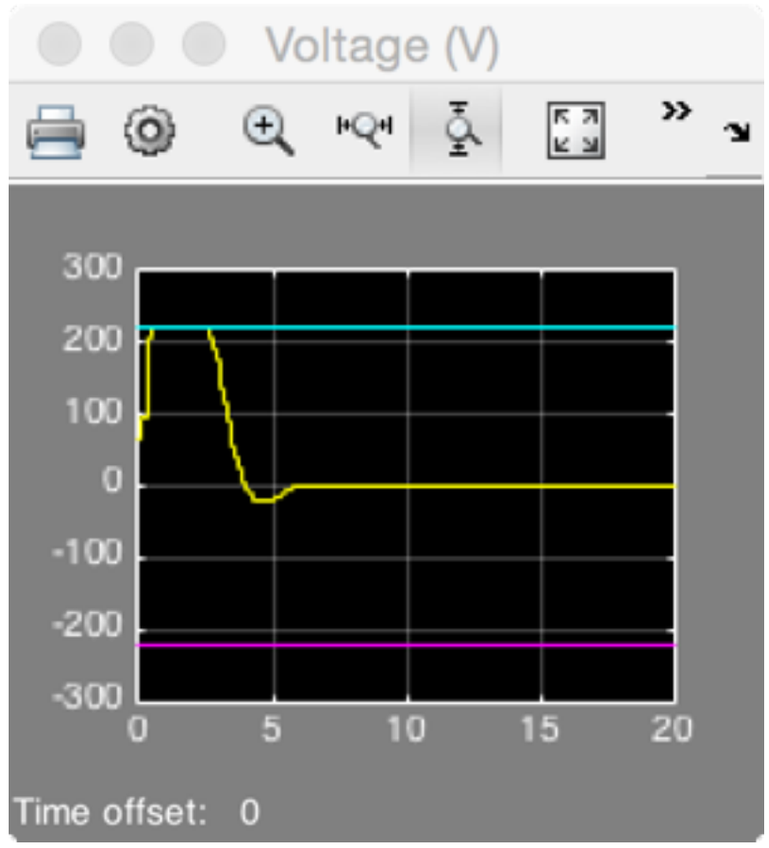




DC SERVOMOTOR: SIMULATION (SIMULINK)







$r(t) \equiv 180$ deg

$$p = 10 \qquad m = 2$$

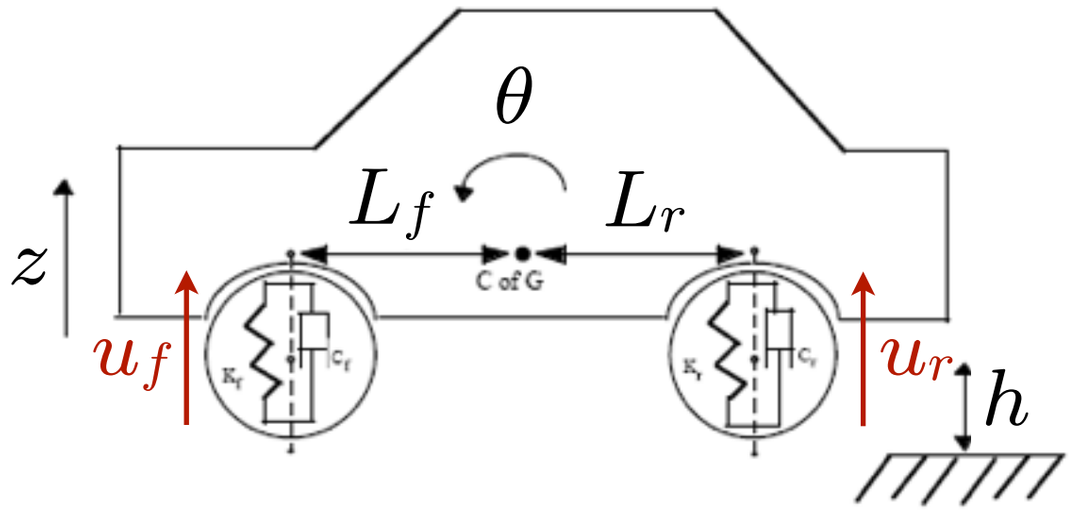
$$w_y = \{10, 0\} \qquad w_{\delta u} = .05 \qquad w_u = 0$$

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

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

AUTOMOTIVE SUSPENSION EXAMPLE

- Consider suspension demo from Mathworks (`sldemo_suspn`)



$$F_{front} = 2K_f(L_f\theta - (z+h)) + 2C_f(L_f\dot{\theta} - \dot{z})$$

F_{front}, F_{rear} = upward force on body from front/rear suspension

K_f, K_r = front and rear suspension spring constant

C_f, C_r = front and rear suspension damping rate

L_f, L_r = horizontal distance from c.g. to front/rear suspension

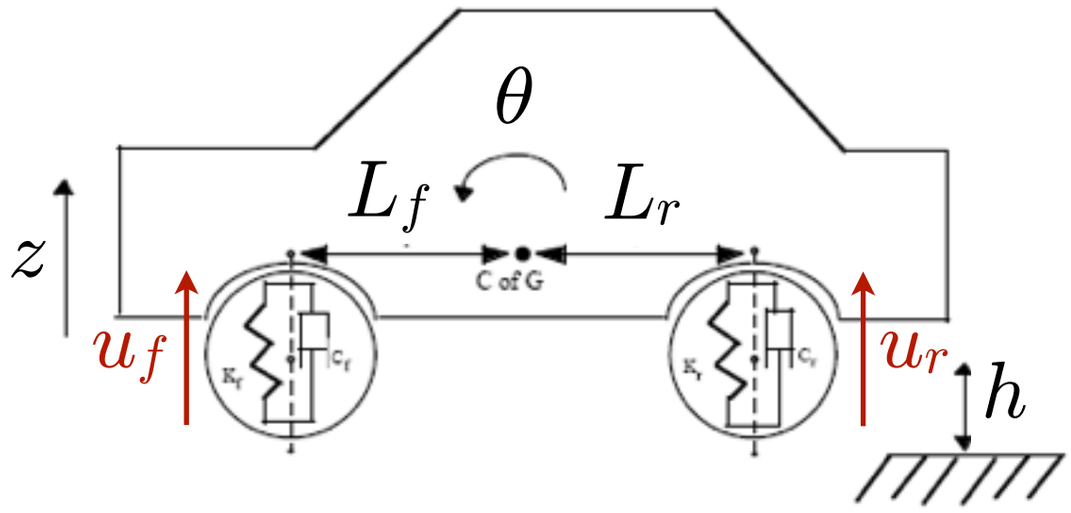
$\theta, \dot{\theta}$ = pitch (rotational) angle and its rate of change

$z, \dot{z}$ = bounce (vertical) distance and its rate of change

- Add external forces u_f, u_r as manipulated variables for control purposes

AUTOMOTIVE SUSPENSION EXAMPLE

- Consider suspension demo from Mathworks (`sldemo_suspn`)



$$F_{front} = 2K_f(L_f\theta - (z+h)) + 2C_f(L_f\dot{\theta} - \dot{z}) + u_f$$

F_{front}, F_{rear} = upward force on body from front/rear suspension

K_f, K_r = front and rear suspension spring constant

C_f, C_r = front and rear suspension damping rate

L_f, L_r = horizontal distance from c.g. to front/rear suspension

$\theta, \dot{\theta}$ = pitch (rotational) angle and its rate of change

$z, \dot{z}$ = bounce (vertical) distance and its rate of change

- The system has 4 states: θ, $d\theta/dt$, z, dz/dt

- Add external forces u_f, u_r as manipulated variables for control purposes

AUTOMOTIVE SUSPENSION EXAMPLE

- Step #1: get a linear discrete-time model (4 inputs, 3 outputs, 4 states)

```
>> plant_mdl = 'suspension_model';
>> op = operspec(plant_mdl);
>> [op_point, op_report] = findop(plant_mdl,op);
>> sys = linearize(plant_mdl, op_point);
>> Ts = 0.025; % sample time (s)
>> plant = c2d(sys,Ts);

>> plant = setmpcsignals(plant,'MV',[1 2],'MD',...
     [3 4],'MO',[1 2],'UO',3);
```

AUTOMOTIVE SUSPENSION EXAMPLE

- Step #2: design MPC controller

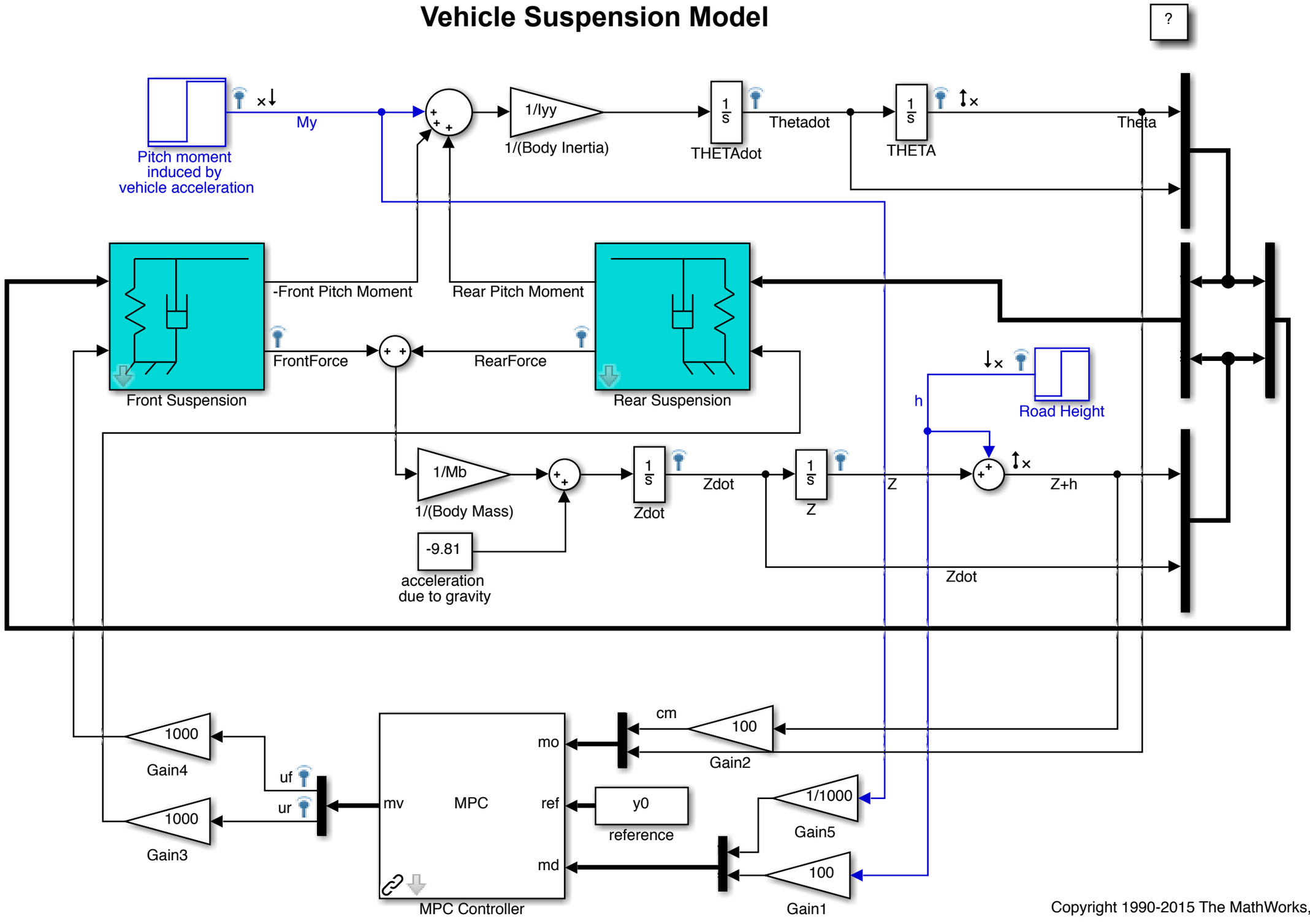
```
>> dfmax = .1/.1; % [kN/s]
>> MV = struct('RateMin',{-Ts*dfmax, ...
      Ts*dfmax},'RateMax',{Ts*dfmax,Ts*dfmax});
>> OV = [];
>> Weights = struct('MV',[0 0],...
                    'MVRate',[.01 .01],...
                    'OV',[.01 0 10]);

>> p = 50; % Prediction horizon
>> m = 5;  % Control horizon

>> mpcobj = mpc(plant,Ts,p,m,Weights,MV,OV);
```

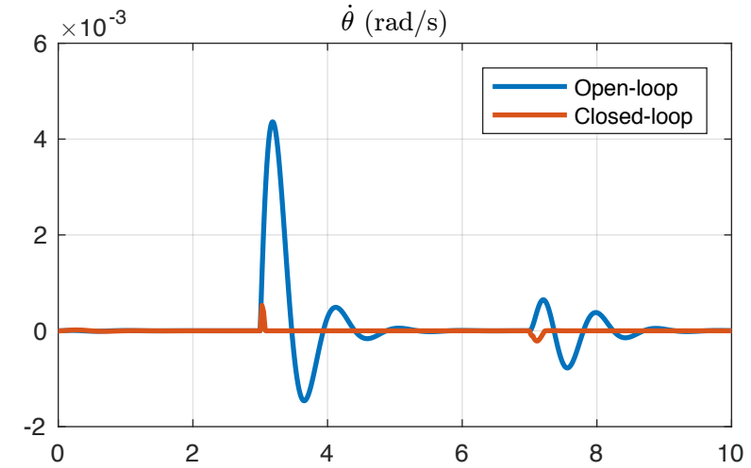
```
>> edit mpcsuspension
```

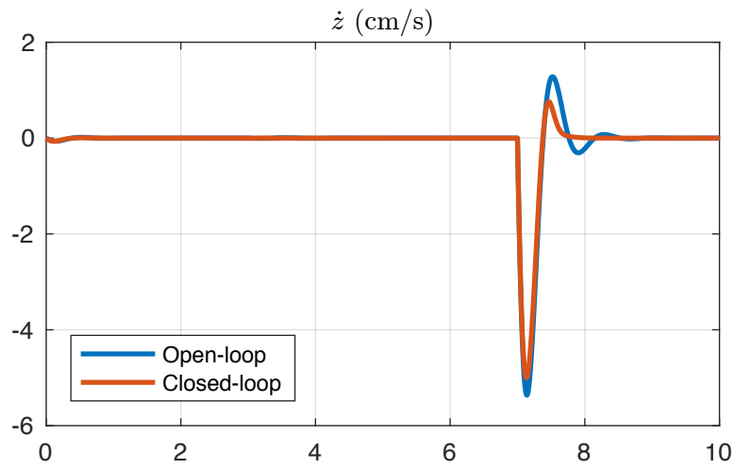
AUTOMOTIVE SUSPENSION EXAMPLE

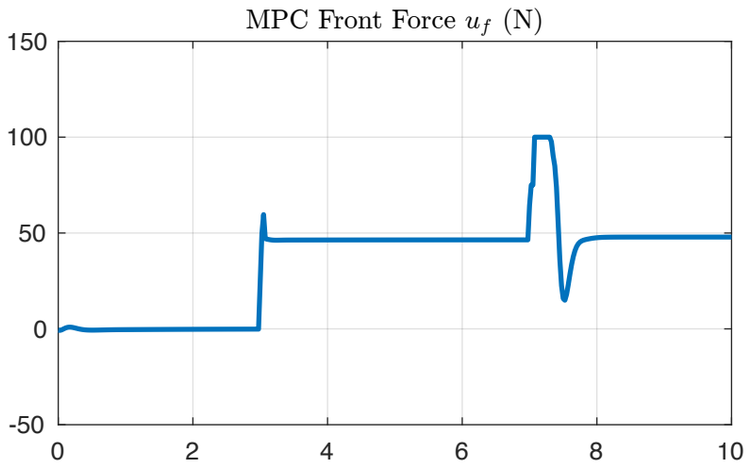


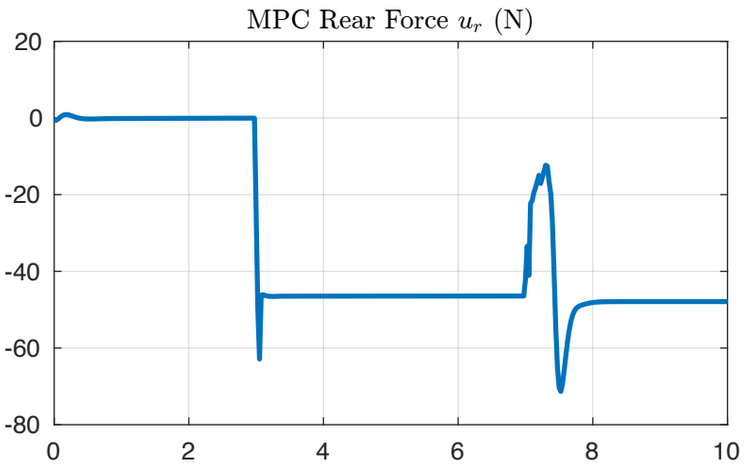
AUTOMOTIVE SUSPENSION EXAMPLE

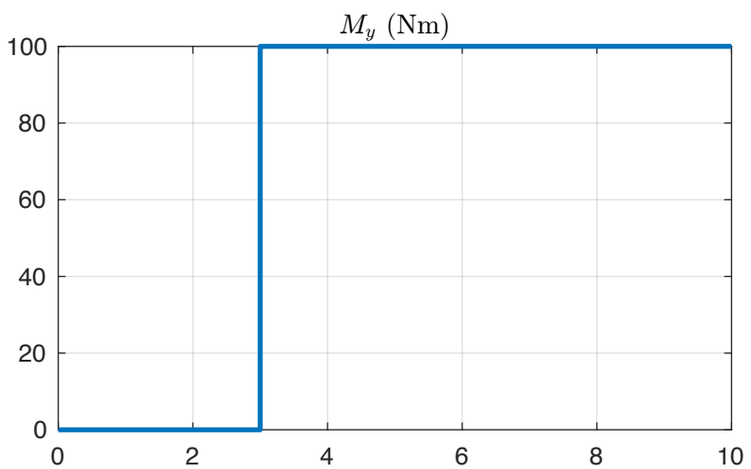
- Closed-loop MPC

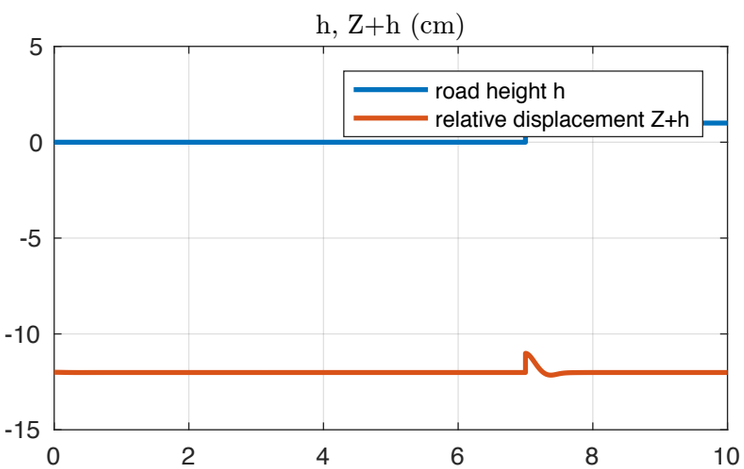








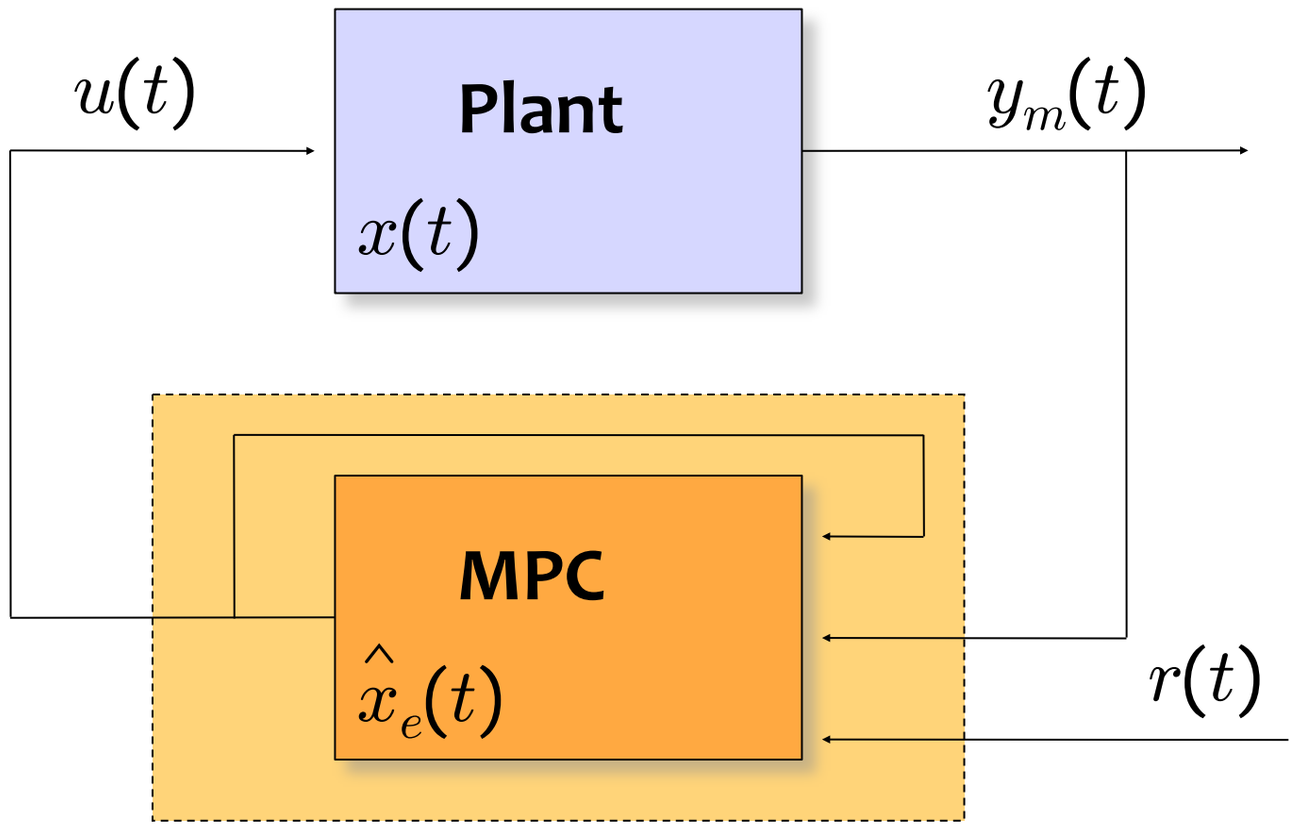




FREQUENCY ANALYSIS OF MPC

- 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, 2010)

- Given the controller $u=K_{fv}\,x$, **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: $-24 \le u(k) \le 24$ and output constraint $y(k) \geq -5$

- 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:

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

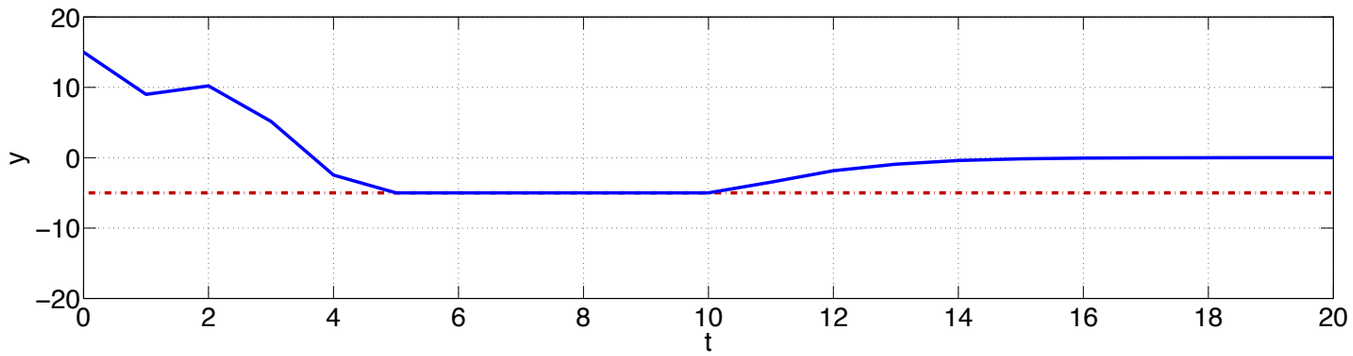
controller matching based on inverse LQR

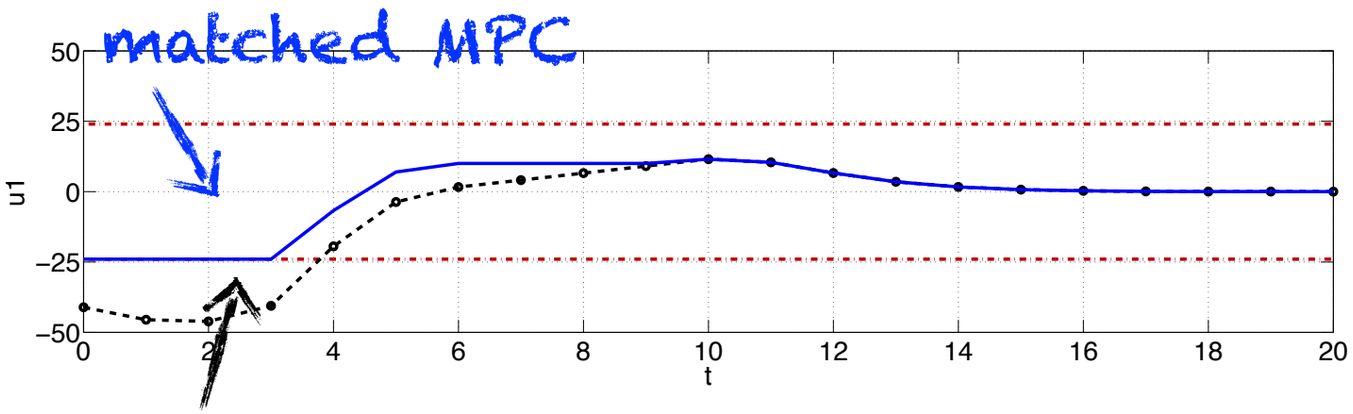
$$Q^* = \begin{bmatrix} 6.401 & 0.064 & -0.001 & 0.020 \\ 0.064 & 6.605 & 0.006 & 0.080 \\ -0.001 & 0.006 & 6.647 & -0.020 \\ 0.019 & 0.080 & -0.020 & 6.378 \end{bmatrix}$$

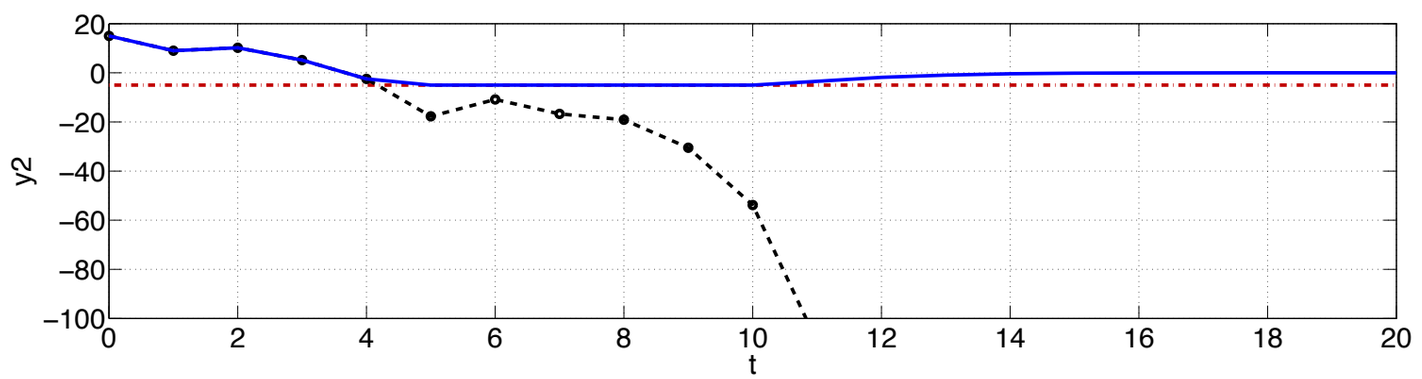
$$R^* = 1$$

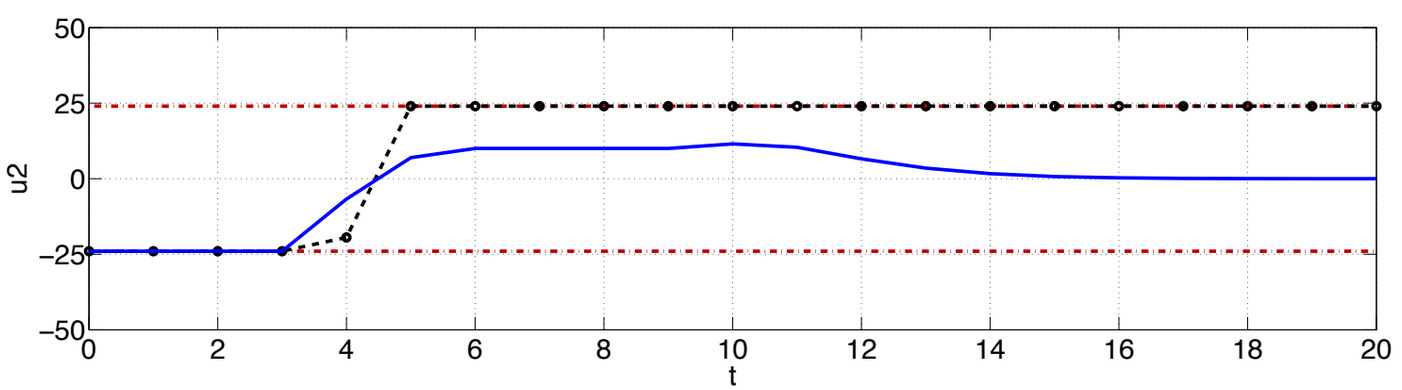
$$P^* = \begin{bmatrix} 422.7 & 241.7 & 50.39 & 201.4 \\ 241.7 & 151.0 & 32.13 & 120.4 \\ 50.39 & 32.13 & 19.85 & 26.75 \\ 201.4 & 120.4 & 26.75 & 106.6 \end{bmatrix}$$

CONTROLLER MATCHING PROBLEM - EXAMPLE









Note: This is not trivially a saturation of PID controller. In this case sat(PID) leads to instability

EMBEDDED MPC WITHOUT A MODEL

~~dynamical model (based on data)~~

optimization algorithm

$$\begin{aligned} \min_z \quad & \frac{1}{2} z'Qz + c'z \\ \text{s.t.} \quad & Az \le b \end{aligned}$$

embedded model-based optimizer

reference $r(t)$

input $u(t)$

process

output $y(t)$

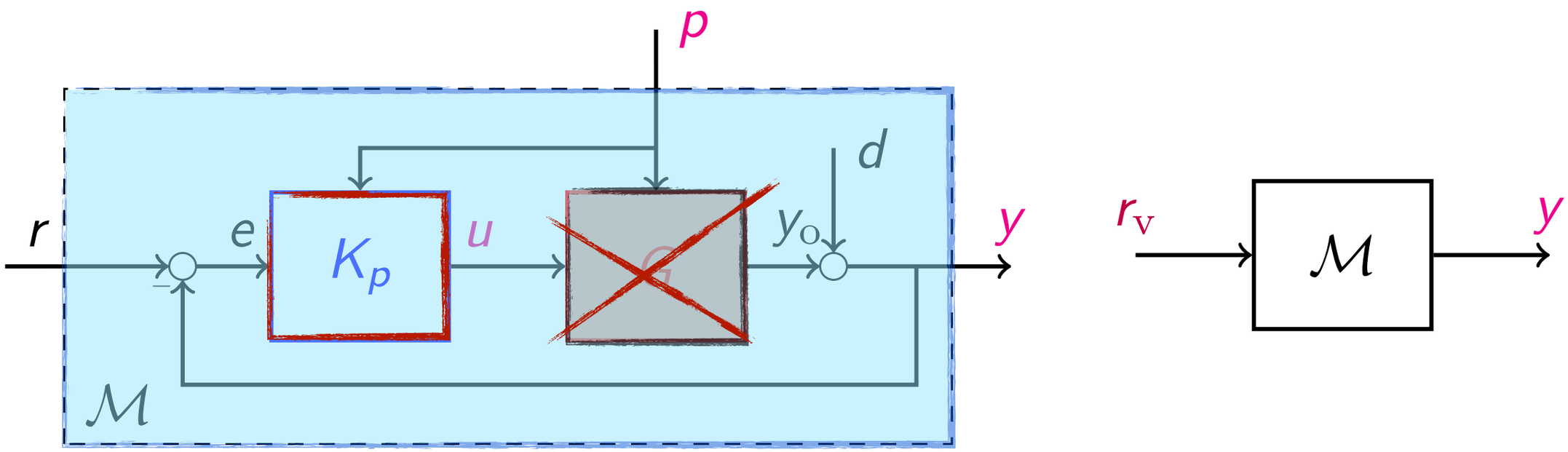
measurements

- Can we run MPC **without** a **model** of the open-loop process ?

YES !

DATA-DRIVEN DIRECT CONTROLLER SYNTHESIS

(Campi, Lecchini, Savaresi, 2002)
(Formentin et al., 2015)

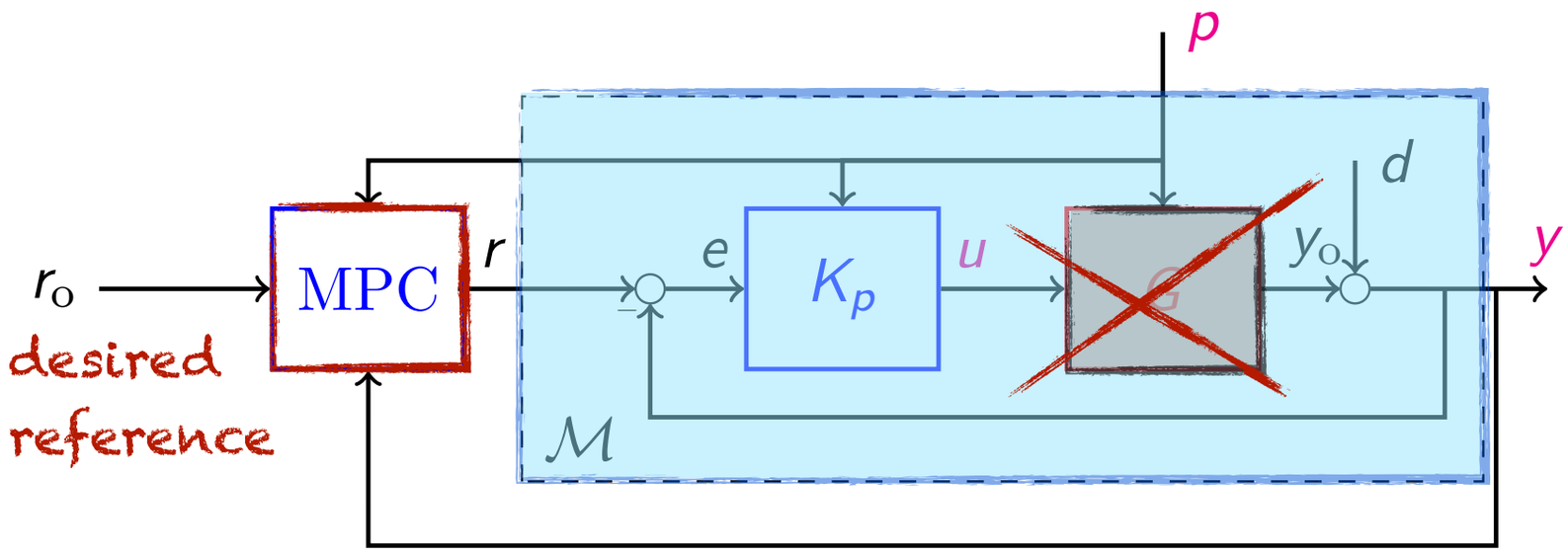


- Collect a set of observations $\{u(k),y(k),p(k)\}$, $k=1,...,N$
- Specify a desired closed-loop linear model $\mathcal{M}$ from r to y
- Compute $r_v(k)=\mathcal{M}^{\#}y(k)$ from pseudo-inverse model $\mathcal{M}^{\#}$ of $\mathcal{M}$
- Identify linear (LPV) model K_p from $e_v=r_v - y$ (virtual tracking error) to u

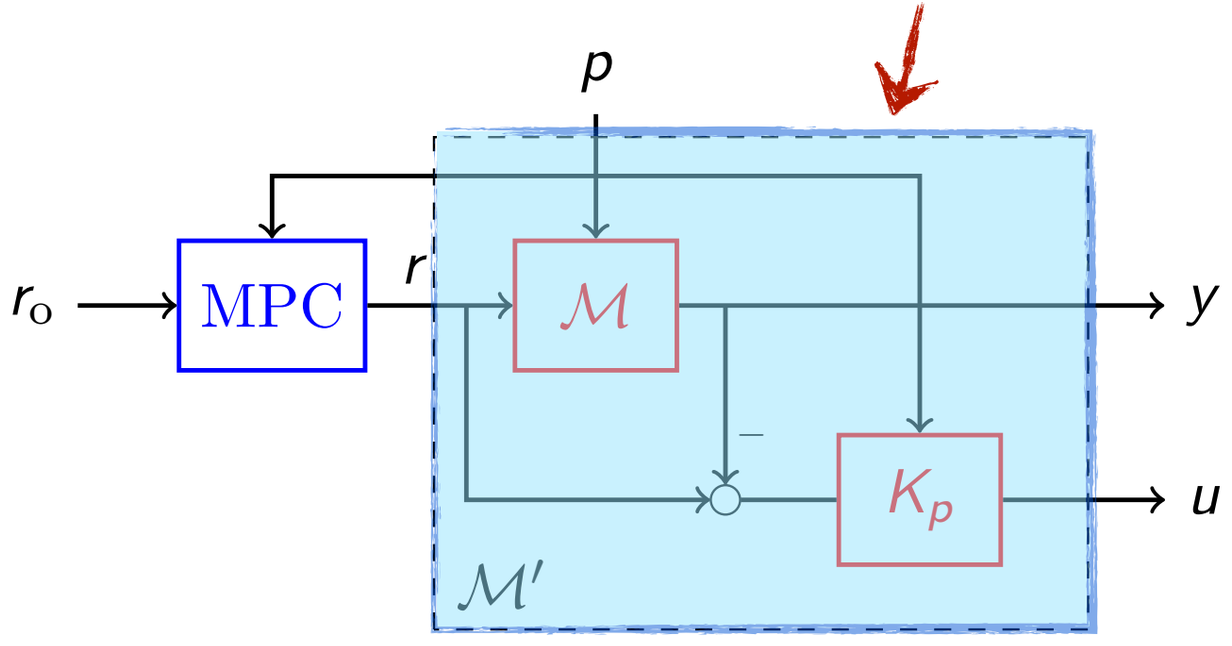
DATA-DRIVEN MPC SYNTHESIS OF CONTROLLERS

- Design a linear MPC controller (reference governor) to generate command r

(Bemporad, Mosca, 1994)
(Gilbert, Kolmanovsky, Tan, 1994)



Linear prediction model (totally known !)

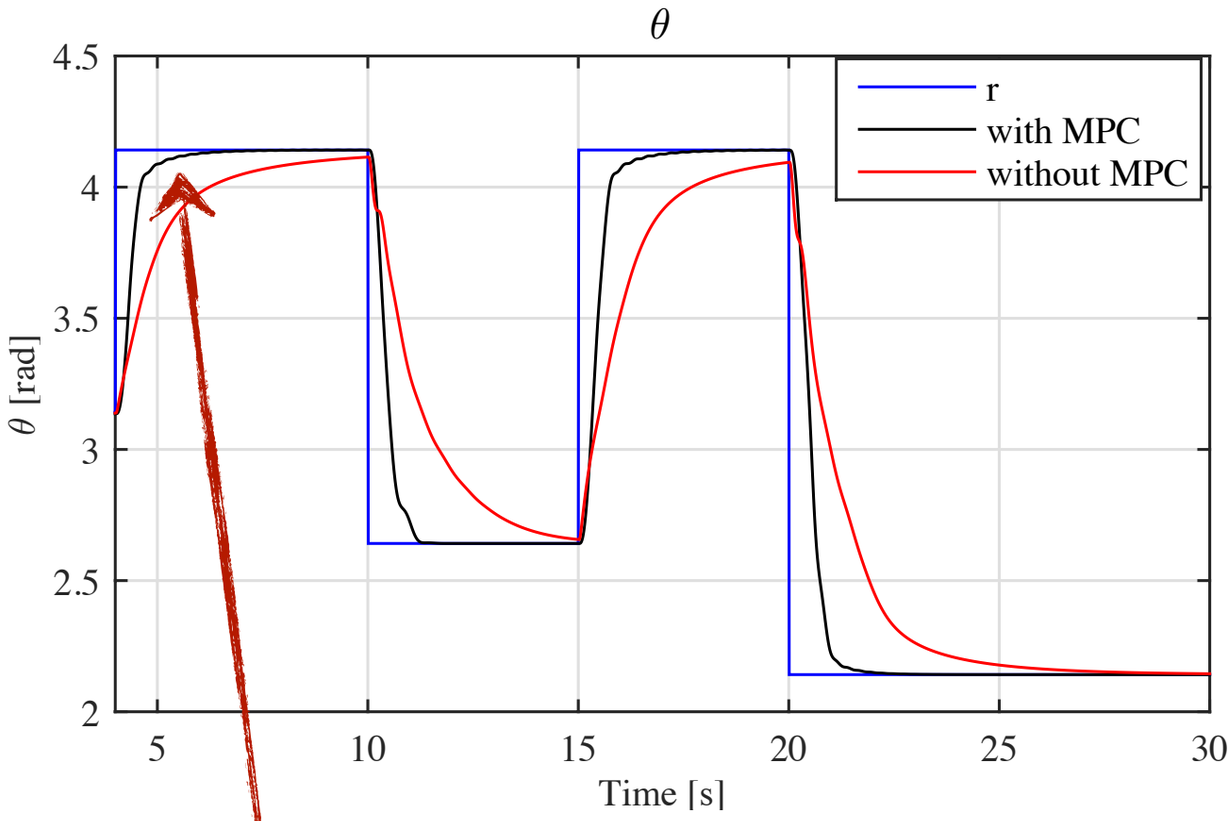


MPC can handle constraints on inputs and outputs, and improve closed-loop performance

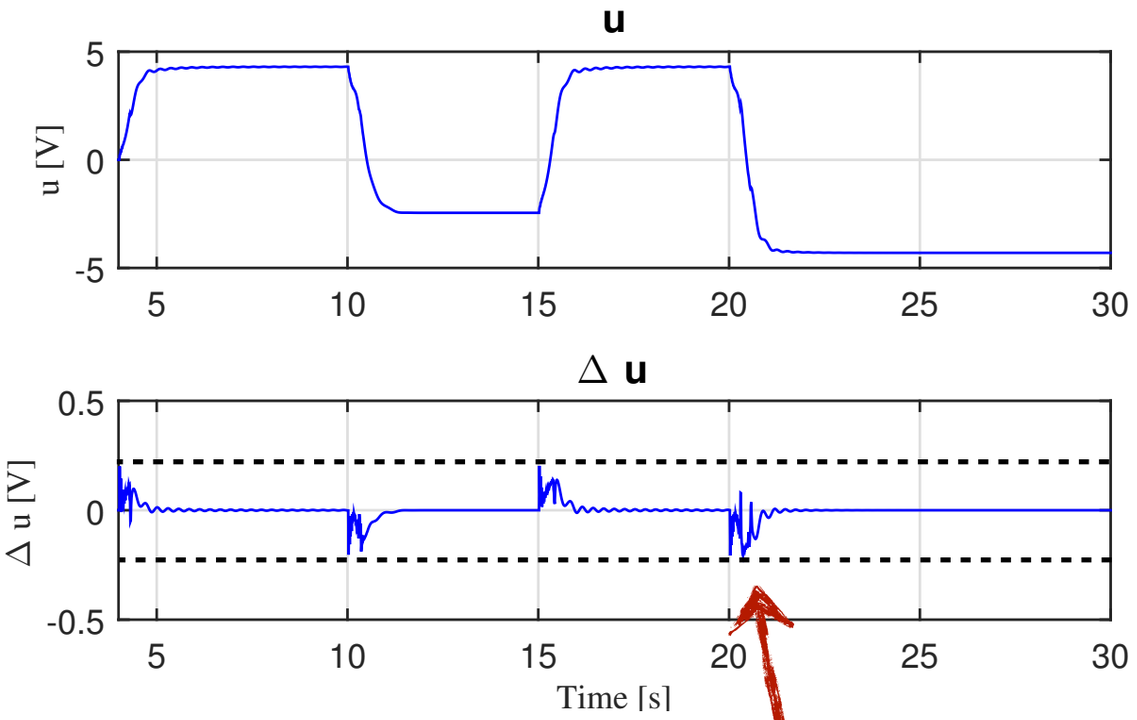
(Piga, Formentin, Bemporad, 2016)

DATA-DRIVEN MPC SYNTHESIS OF CONTROLLERS - AN EXAMPLE

- Experimental results



desired tracking performance achieved



constraints on input increments satisfied

No model of open-loop process identified to design the MPC controller !

LINEAR MPC BASED ON LINEAR PROGRAMMING

LINEAR MPC BASED ON LP

(Propoi, 1963)

(Bemporad, Borrelli, Morari, 2003)

- Linear prediction model:

$$\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \end{cases}$$

$$x_0 = x(t)$$

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

- Constraints to enforce:

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

- Constrained optimal control problem (∞-norm performance index):

$$\min_U \quad \|Px_N\|_\infty + \sum_{k=0}^{N-1} \|Qx_k\|_\infty + \|Ru_k\|_\infty$$

$$\text{s.t.} \quad u_{\min} \le u_k \le u_{\max}, \quad k = 0, \dots, N-1$$
$$y_{\min} \le y_k \le y_{\max}, \quad k = 1, \dots, N$$

Q, R, P full rank

$$\|v\|_\infty \triangleq \max_{i=1,\dots,n} |v_i|$$

LINEAR MPC BASED ON LP

- Basic trick:

$$\begin{array}{ll} \min & |x| \\ x \in \mathbb{R} & \end{array} \quad \Longleftrightarrow \quad \begin{array}{ll} \min & \epsilon \\ \text{s.t.} & \epsilon \geq x \\ & \epsilon \geq -x \end{array}$$

- Introduce slack vars:

$$\begin{array}{rcl} \epsilon_k^x & \geq & \|Qx_k\|_\infty \\ \epsilon_k^u & \geq & \|Ru_k\|_\infty \\ \epsilon_N^x & \geq & \|Px_k\|_\infty \end{array} \quad \Longrightarrow \quad \begin{array}{rcll} \epsilon_k^x & \geq & Q^i x_k & i = 1, \ldots, n \\ \epsilon_k^x & \geq & -Q^i x_k & k = 0, \ldots, N-1 \\ \epsilon_k^u & \geq & R^i u_k & i = 1, \ldots, m \\ \epsilon_k^u & \geq & -R^i u_k & k = 0, \ldots, N-1 \\ \epsilon_N^x & \geq & P^i x_N & i = 1, \ldots, n \\ \epsilon_N^x & \geq & -P^i x_N & \end{array}$$

$Q^i = i$th row of matrix Q

LINEAR MPC BASED ON LP

- Substitution: $$x_{k+1} = A^k x(t) + \sum_{j=0}^{k-1} A^j B u_{k-1-j}$$

- Optimization problem:

$$V^*(x(t)) = \min_z \begin{bmatrix} 1 & \dots & 1 & 0 & \dots & 0 \end{bmatrix} z \quad \text{(linear)}$$

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

LINEAR PROGRAM (LP)

- $z \triangleq [\epsilon_0^u \ \dots \ \epsilon_{N-1}^u \ \epsilon_1^x \ \dots \ \epsilon_N^x \ u_0', \dots, u_{N-1}']' \in \mathbb{R}^s$, $s \triangleq N(m+2)$, is the optimization vector

- G, W, S are obtained from weights Q, R, P, and model matrices A, B, C

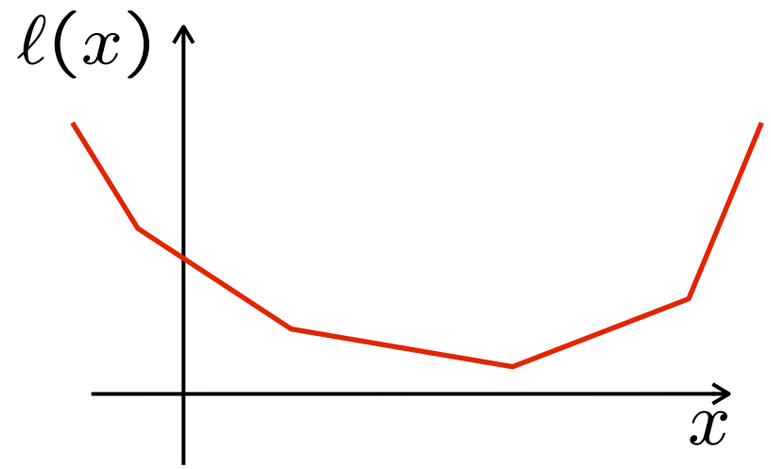
- Q, R, P can be selected to guarantee closed-loop stability

(Bemporad, Borrelli, Morari, 2003)

EXTENSION TO ARBITRARY CONVEX PWA FUNCTIONS

- Constrained optimal control problem:

$$\begin{aligned} \min_{U} \quad & \ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k) \\ \text{s.t.} \quad & g_k(x_k, u_k) \leq 0, \;\; k = 0, \ldots, N-1 \\ & g_N(x_N) \leq 0 \end{aligned}$$



where ℓ_k, ℓ_N, g_k, g_N are arbitrary convex piecewise affine (PWA) functions

Result: Every convex piecewise affine function $C : \mathbb{R}^n \to \mathbb{R}$ can be represented as the max of affine functions, and vice versa

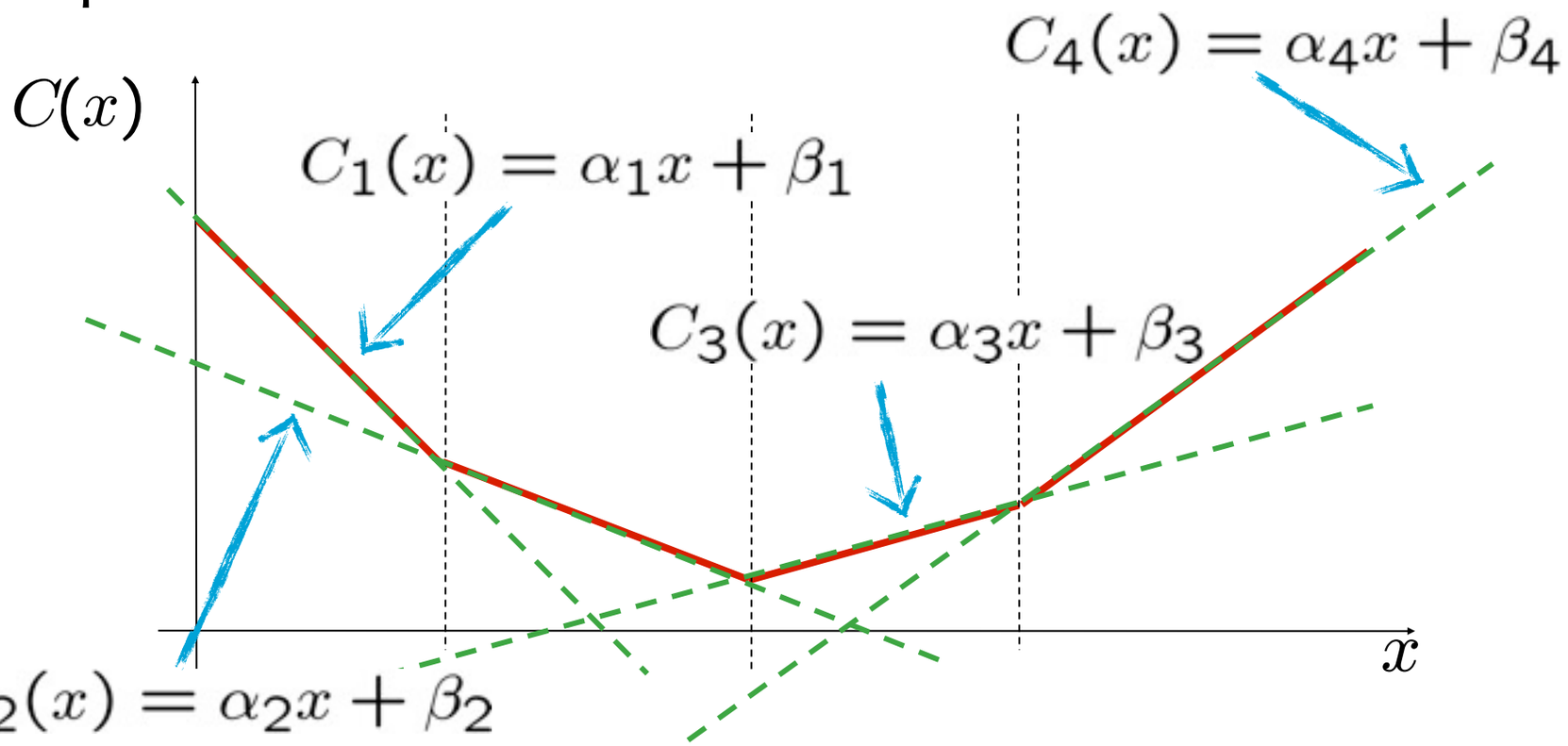
(Schechter, 1987)

$$C(x) = \max\left\{a_1'x + b_1, \ldots, a_M'x + b_M\right\}$$

Example: $|x| = \max\{x, -x\}$

CONVEX PWA COSTS

Example:

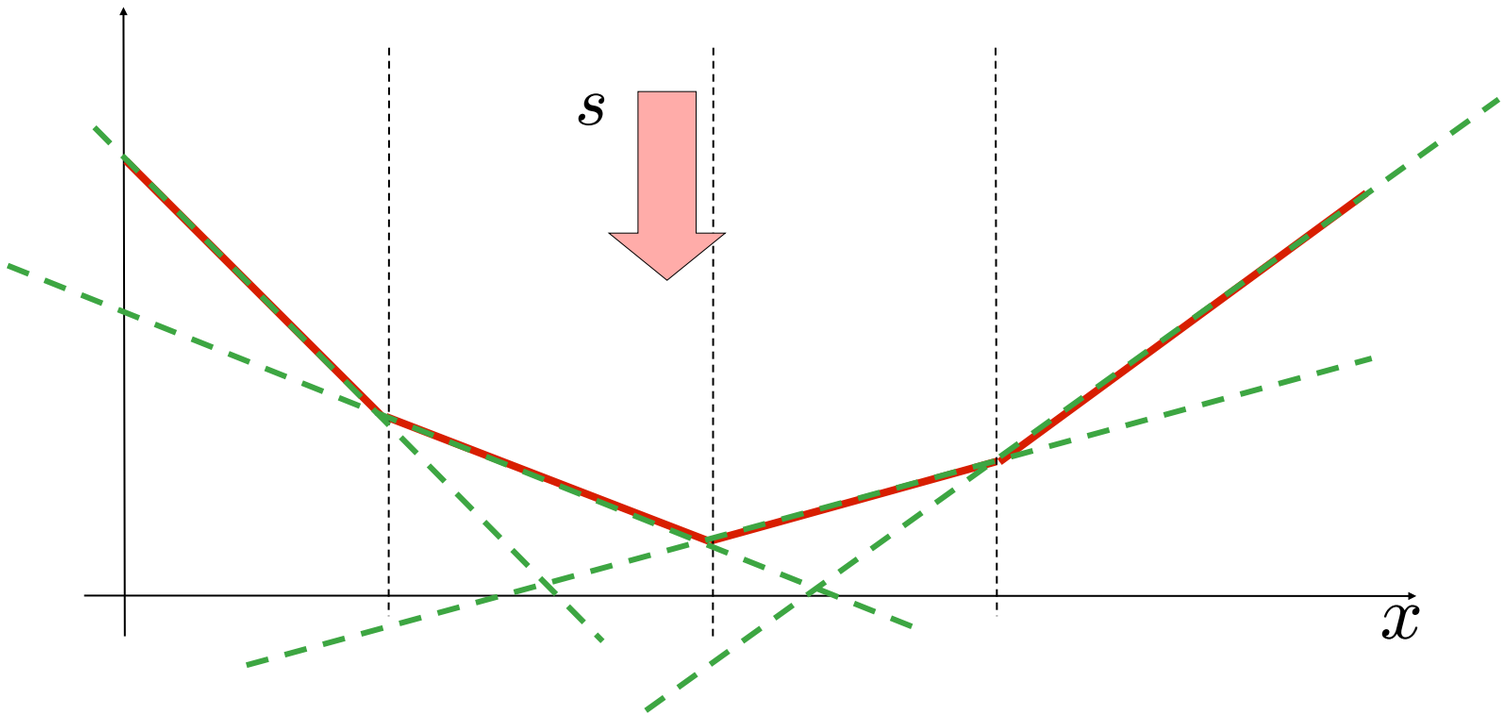


It is easy to see that:

$$C(x) = \max\{\alpha_1 x + \beta_1, \alpha_2 x + \beta_2, \alpha_3 x + \beta_3, \alpha_4 x + \beta_4\}$$

CONVEX PWA OPTIMIZATION PROBLEMS AND LP

Minimization of a convex PWA function $C(x)$:



$$\begin{aligned} \min \quad & s \\ \text{subj.to} \quad & \begin{cases} s \geq \alpha_1 x + \beta_1 \\ s \geq \alpha_2 x + \beta_2 \\ s \geq \alpha_3 x + \beta_3 \\ s \geq \alpha_4 x + \beta_4 \end{cases} \end{aligned}$$

$$s \geq \max\{\alpha_1 x + \beta_1, \alpha_2 x + \beta_2, \alpha_3 x + \beta_3, \alpha_4 x + \beta_4\}$$

Variabile s is an upper-bound on the max

It is easy to show (by contradiction) that at optimality we have:

$$s = \max\{\alpha_1 x + \beta_1, \alpha_2 x + \beta_2, \alpha_3 x + \beta_3, \alpha_4 x + \beta_4\}$$

Convex PWA constraints $C(x) \leq 0$:

Simply impose $\alpha_j x + \beta_j \leq 0$ for all $j = 1,2,3,4$

LP-BASED VS. QP-BASED MPC

- QP- and LP-based share the same set of feasible inputs ($GU \leq W + Sx$), so when constraints dominate over performance there is little difference between them (e.g., during transients)

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

- Small-signal response, however, is usually **less smooth** with LP than with QP

- Explanation: In linear programs an optimal point is always on a vertex

$$\left\{ U : \begin{array}{l} GU \leq W + Sx \\ \bar{G}U + \bar{E}\epsilon \leq \bar{W} + \bar{S}x \end{array} \right\}$$

$$\{U : \; GU \leq W + Sx\}$$

additional slack vars to represent convex PWA costs