Model Predictive Control

Alberto Bemporad

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

Institutions Markets Technologies
IMT
INSTITUTE FOR ADVANCED STUDIES LUCCA

DYSCO Research Unit
Dynamical Systems, Control, and Optimization

Course structure

- Basic concepts of MPC and optimization
- Linear MPC and MPC theory
- Explicit MPC (multiparametric programming)
- Hybrid MPC
- Stochastic MPC

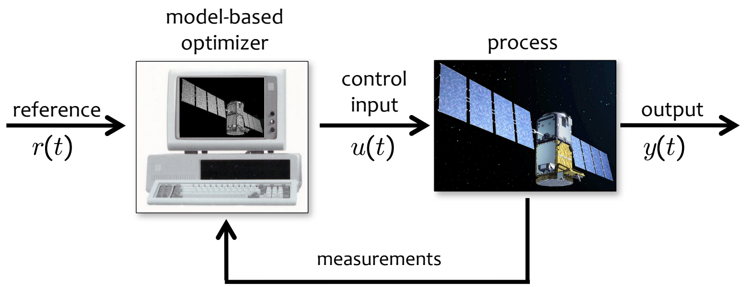
- MATLAB:
  - MPC Toolbox (linear MPC)
  - Hybrid Toolbox (explicit MPC, hybrid systems)

- Lecture notes:

  http://cse.lab.imtlucca.it/~bemporad/teaching/mpc/stuttgart_2012

Model Predictive Control: Basic Concepts

Model Predictive Control (MPC)



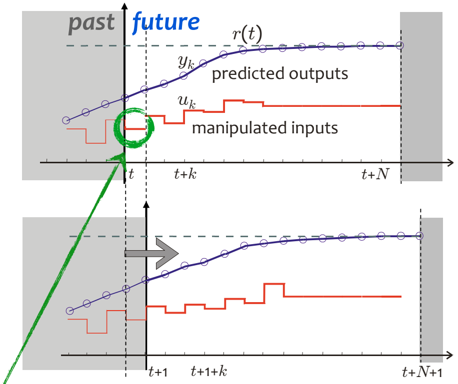
*Use a dynamical **model** of the process to **predict** its future evolution and choose the "best" **control** action*

Receding horizon control

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

penalty on tracking error — *penalty on actuation effort*

$$\min \sum_{k=0}^{N-1} \ell(y_k - r(t+k), u_k)$$
s.t. $x_{k+1} = f(x_k, u_k)$
$y_k = g(x_k, u_k)$
constraints on u_k, x_k, y_k
$x_0 = x(t)$

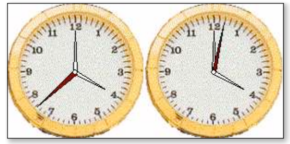


- Apply only the first optimal move $u^*(t)$, throw the rest of the sequence away
- **At time $t+1$**: Get new measurements, repeat the optimization. And so on …

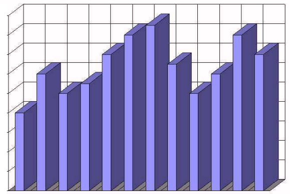
MPC transforms open-loop optimal control into feedback control

Receding horizon examples

- MPC is like **playing chess** !

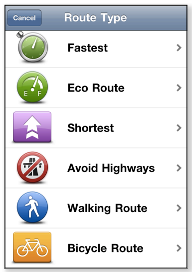


- “Rolling horizon” policies are also used frequently **in finance**



Receding horizon planning: GPS example

- **prediction model** — how vehicle moves on map
- **constraints** — drive on roads, respect one-way roads, etc.
- **disturbances** — (driver does not follow directions properly)
- **set point** — desired location
- **cost function** — minimum time, minimum distance, etc.
- **receding horizon mechanism** (event-based: optimal route re-planned when path is lost)



x = GPS position
u = navigation commands

Good models for (MPC) control

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

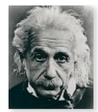
Good models for MPC:

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

TRADE OFF

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

“Make everything as simple as possible, but not simpler.”
— Albert Einstein



MPC in industry

Area	Aspen Technology	Honeywell Hi-Spec	Adersa[b]	Invensys	SGS[c]	Total
Refining	1200	480	280	25		1985
Petrochemicals	450	80	—	20		550
Chemicals	100	20	3	21		144
Pulp and paper	18	50	—	—		68
Air & Gas	—	10	—	—		10
Utility	—	10	—	4		14
Mining/Metallurgy	8	6	7	16		37
Food Processing	—	—	41	10		51
Polymer	17	—	—	—		17
Furnaces	—	—	42	3		45
Aerospace/Defense	—	—	13	—		13
Automotive	—	—	7	—		7
Unclassified	40	40	1045	26	450	1601
Total	1833	696	1438	125	450	4542
First App.	DMC:1985 IDCOM-M:1987 OPC:1987	PCT:1984 RMPCT:1991	IDCOM:1973 HIECON:1986	1984	1985	
Largest App.	603 × 283	225 × 85	—	31 × 12	—	

(snapshot survey conducted in mid-1999) (Qin, Badgewell, 2003)

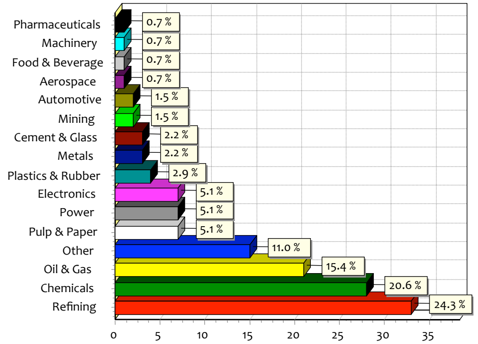
"For us multivariable control is predictive control"

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

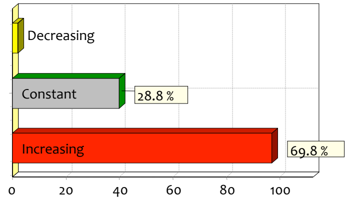
MPC in industry

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

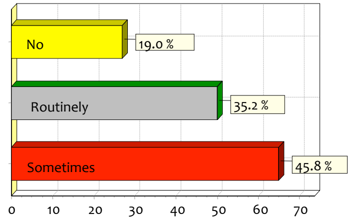
Industrial area of respondents to the survey:



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



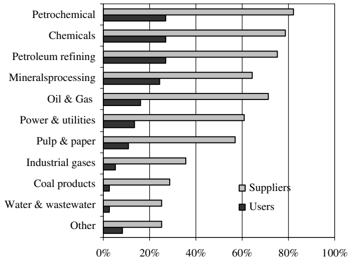
Does your company use MPC ?



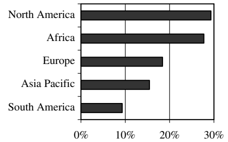
courtesy: ARC Advisory Group

MPC in industry

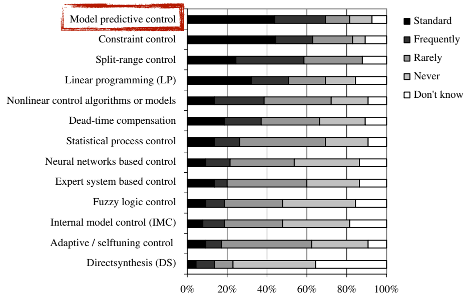
- Economic assessment of Advanced Process Control (APC) (Bauer & Craig, 2008)



participants of APC survey by industry



participants of APC survey by continent



Industrial use of APC methods: survey results

MPC research is driven by applications

- **Process** control → **linear** MPC (some **nonlinear** too) 1980-2000
- **Automotive** control → **explicit, hybrid** MPC 2001-2010
- **Aerospace** systems and UAVs → **linear time-varying** MPC >2005
- **Information and Communication Technologies (ICT)** (wireless nets, cloud computers) → **distributed**/**decentralized** MPC >2005
- **Energy, finance, automotive** → **stochastic** MPC >2010

Basics of Constrained Optimization

Mathematical programming

$$\min_x \ f(x) \quad \text{s.t.} \ g(x) \leq 0$$

$$\left(\max_x \ f(x) \quad \text{s.t.} \ g(x) \leq 0 \right)$$

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

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

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

Optimization software

- 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/index.php/NEOS_Wiki

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

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

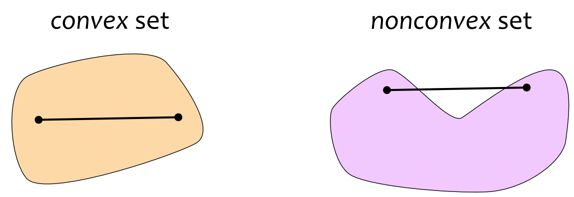
- Good open-source optimization software

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

Convex sets

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

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

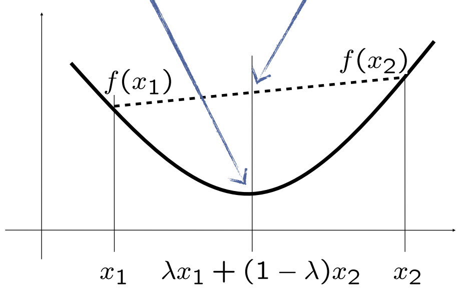


Convex function

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

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

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



Convex optimization problem

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

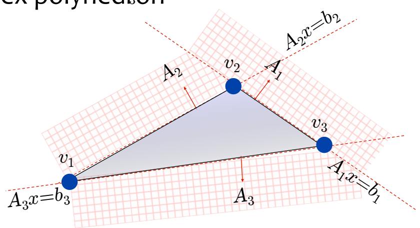
f = convex function
C = convex set

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

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

Polyhedra

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



- Hyperplane representation:

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

vertex enumeration

convex hull

- Vertex representation:

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

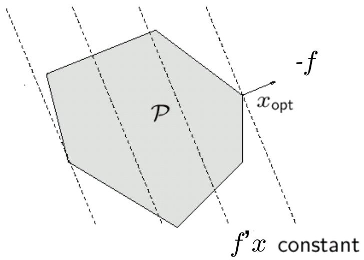
$\alpha_i \geq 0, \; \sum_{i=1}^{q} \alpha_i = 1, \; v_i \in \mathbb{R}^n$

Linear programming

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

linear program (LP)

George Dantzig (1914 - 2005)



standard form

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

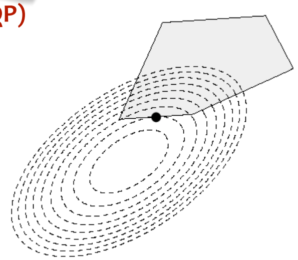
slack variables

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i \;\Rightarrow\; \sum_{j=1}^{n} a_{ij}x_j + s_i = b_i, \; s_i \geq 0$$

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

Mixed-integer programming (MIP)

$$\begin{array}{ll} \min & f'x \\ \text{s.t.} & Ax \leq b, \;\; x = \left[\begin{smallmatrix} x_c \\ x_b \end{smallmatrix}\right] \\ & x_c \in \mathbb{R}^{n_c}, \;\; x_b \in \{0,1\}^{n_b} \end{array}$$

mixed-integer linear program (MILP)

$$\begin{array}{ll} \min & \frac{1}{2}x'Px + f'x \\ \text{s.t.} & Ax \leq b, \;\; x = \left[\begin{smallmatrix} x_c \\ x_b \end{smallmatrix}\right] \\ & x_c \in \mathbb{R}^{n_c}, \;\; x_b \in \{0,1\}^{n_b} \end{array}$$

mixed-integer quadratic program (MIQP)

- Some variables are continuous, some are discrete $(0/1)$
- A $\mathcal{NP}$-hard problem, in general
- Rich variety of algorithms/solvers 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 Ilog-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)
- **CVX** MATLAB-based modeling language (Stanford Univ.)
- **YALMIP** another MATLAB-based modeling language

Linear MPC

Linear MPC - Unconstrained case

- Linear prediction model:

$$\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \end{cases} \qquad x_0 = x(t)$$

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

- Performance index

$$J(x_0, U) = \min_U x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k$$

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

- Goal: find sequence $U^* = \begin{bmatrix} u_0^* \\ u_1^* \\ \vdots \\ u_{N-1}^* \end{bmatrix}$ that steers the state to the origin "optimally"

[computation of cost function]

$$J(x_0,U) = x_0'Qx_0 + \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}' \overbrace{\begin{bmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & Q & 0 \\ 0 & 0 & \dots & 0 & P \end{bmatrix}}^{\bar{Q}} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} + \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}' \underbrace{\begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & R \end{bmatrix}}_{\bar{R}} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

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

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

Linear MPC - Unconstrained case

$$J(x_0, U) = \frac{1}{2}U'HU + x_0'FU + \frac{1}{2}x_0'Yx_0 \qquad U = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

The optimum is obtained by zeroing the gradient

$$\nabla_U J(x_0, U) = HU + F'x_0 = 0$$

and hence $U^* = \begin{bmatrix} u_0^* \\ u_1^* \\ \vdots \\ u_{N-1}^* \end{bmatrix} = -H^{-1}F'x_0$ **(batch least squares)**

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

Linear MPC - Constrained case

- Linear prediction model:

$$\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \end{cases} \qquad x_0 = x(t)$$

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

- 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):

$$\begin{aligned} \min_U \quad & x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \\ \text{s.t.} \quad & u_{\min} \le u_k \le u_{\max}, \quad k = 0, \dots, N-1 \\ & y_{\min} \le y_k \le y_{\max}, \quad k = 1, \dots, N \end{aligned}$$

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

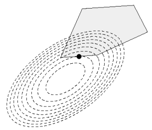
Linear MPC - Constrained case

- Optimization problem:

$$V(x_0) = \frac{1}{2}x_0'Yx_0 + \min_U \frac{1}{2}U'HU + x_0'FU \quad \text{(quadratic)}$$

$$\text{s.t.} \quad GU \leq W + Sx_0 \quad \text{(linear)}$$

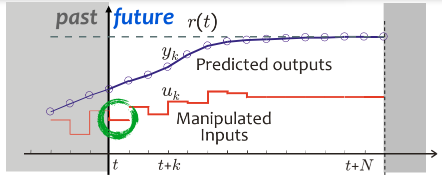
Convex QUADRATIC PROGRAM (QP)



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

Linear MPC algorithm



At time t:

- Measure (or estimate) the current state $x(t)$
- Solve the QP problem

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

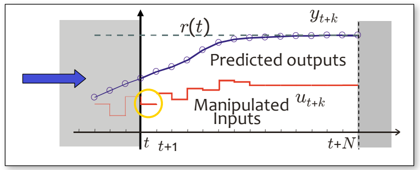
$$\text{s.t.} \quad GU \leq W + Sx(t)$$

Let $U = \{u^*(0), \ldots, u^*(N\text{-}1)\}$ be the solution

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

Linear MPC - Unconstrained case

- Assume no constraints
- Problem to solve on-line:



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

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

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

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

$$\Rightarrow u(t) = -[I \; 0 \; \ldots \; 0]H^{-1}Fx(t) \triangleq Kx(t)$$

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

Double integrator example

- System: $y(\tau) = \frac{1}{s^2}u(\tau)$ $\Longrightarrow$ sampling + ZOH T_s=1 s

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

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

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

- Control objective: $\min \left(\sum_{k=0}^{1} y_k^2 + \frac{1}{10}u_k^2 \right) + x_2' \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right] x_2$

- Optimization problem matrices:

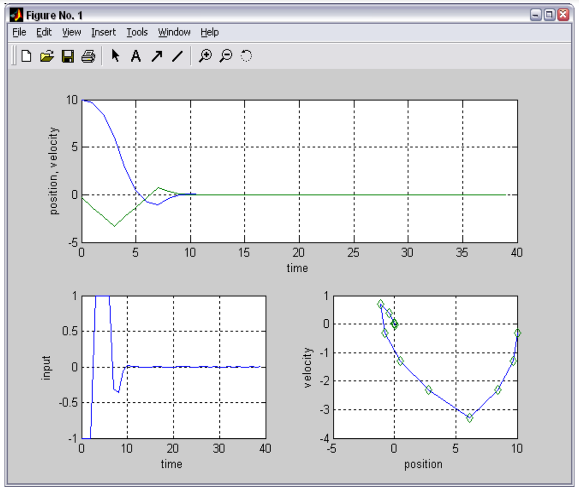
$$H = \begin{bmatrix} 4.2 & 2 \\ 2 & 2.2 \end{bmatrix}, \; F = \begin{bmatrix} 2 & 0 \\ 6 & 2 \end{bmatrix}, \; Y = \begin{bmatrix} 4 & 6 \\ 6 & 12 \end{bmatrix}$$

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

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

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

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:

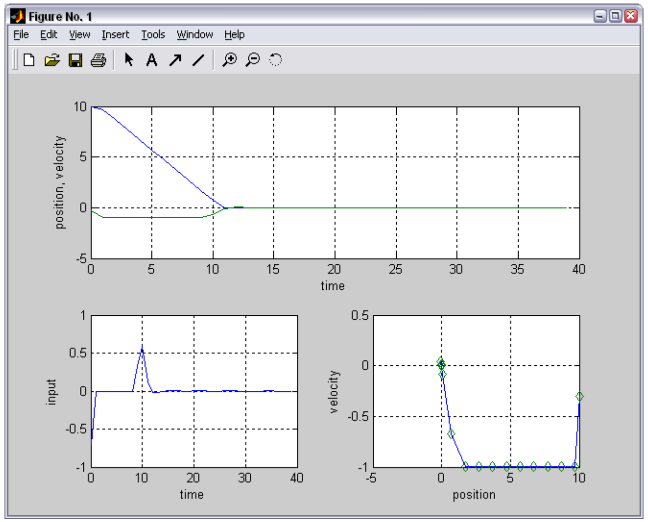
$$H = \begin{bmatrix} 4.2 & 2 \\ 2 & 2.2 \end{bmatrix}, \; F = \begin{bmatrix} 2 & 0 \\ 6 & 2 \end{bmatrix}, \; Y = \begin{bmatrix} 4 & 6 \\ 6 & 12 \end{bmatrix}$$

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

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

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

Double integrator example



Linear MPC - Tracking

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

$$\boxed{\Delta u(t) = u(t) - u(t-1)} \;\Longrightarrow\; u(t) = u(t-1) + \Delta u(t)$$

- 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):

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

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

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

optimization vector

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

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

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

- Optimization problem:

Convex Quadratic Program (QP)

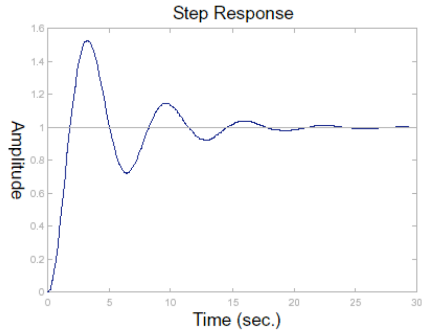
$$\min_{\Delta U} J(\Delta U, x(t)) = \tfrac{1}{2}\Delta U' H \Delta U + [x'(t)\ r'(t)\ u'(t-1)]F\Delta U$$

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

- Constraints on tracking errors can be also included: $e_{\min} \le y_k - r(t) \le e_{\max}$

Linear MPC - Tracking example

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



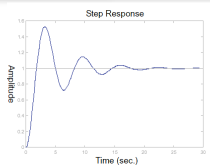
- Sampling time: $T_s = 0.5$ sec.

- 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)

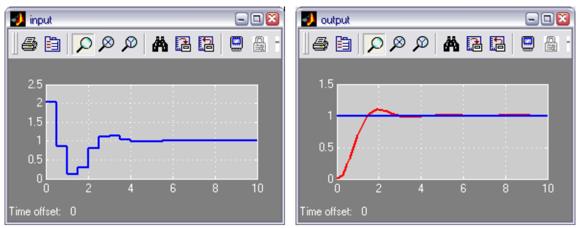
Linear MPC - Tracking example



- Performance index:

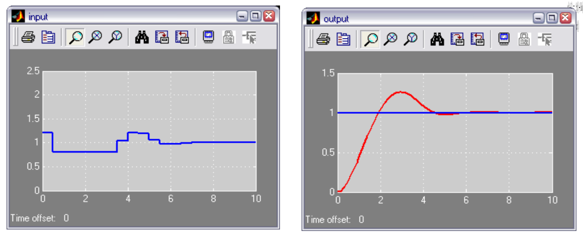
$$J(U,t) \triangleq \sum_{k=0}^{9} [y(t+k+1|t) - r(t)]^2 + 0.04\Delta u^2(t)$$

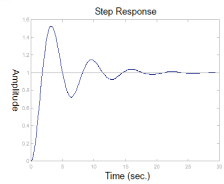
- Closed-loop MPC:



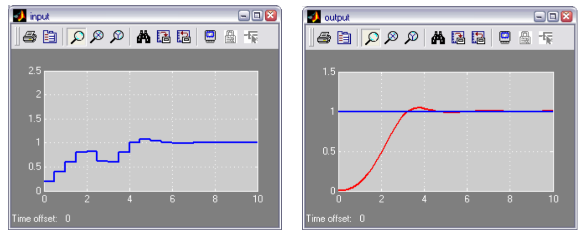
Linear MPC - Tracking example

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





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

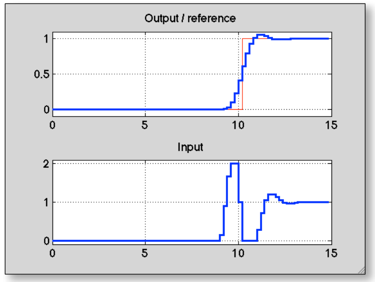


Anticipative action (or preview)

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

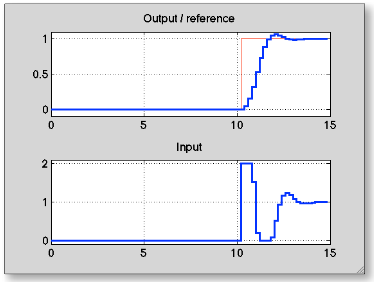
- Future reference samples (partially) known in advance (anticipative action):

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



- Reference not known in advance (causal):

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



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

Soft constraints

- To prevent QP infeasibility, relax output constraints:

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

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

ϵ = "panic" variable

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

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

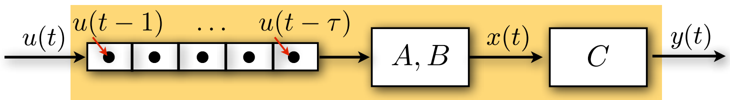
$V_{\min}$, $V_{\max}$ = vectors with entries ≥ 0 (the larger the entry, the relatively softer the corresponding constraint)

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

Delays – Method #1

- Linear model w/ delays:

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



- Map delays to poles in z=0:

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

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

- Apply MPC to the extended system

Delays – Method #2

- Linear model w/ delays:

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

- Delay-free model:

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

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

- Design MPC for delay-free model: $u(t) = f_{\text{MPC}}(\bar{x}(t))$

- 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) !

MPC vs. conventional control

Single input/single output (SISO) control loops with constraints: equivalent performance can be obtained with other simpler control techniques (e.g.: PID + anti-windup)

however

MPC allows **uniformity** = same technique for wide range of problems:
- MIMO
- constraints
- preview
- delays
- time-varying models
- ...

and therefore makes **design maintenance easier**

Satisfying control specs and walking on water are similar …

No problem when they are frozen !

MPC theory

- Historical goal: Explain the success of DMC
- Present goal: Improve, simplify, and extend industrial algorithms
- Areas:
  - Linear MPC linear prediction model
  - Nonlinear MPC nonlinear prediction model
  - Robust MPC uncertain (linear) prediction model
  - Stochastic MPC stochastic prediction model
  - Hybrid MPC prediction model integrating logic and dynamics
  - Explicit MPC off-line (exact/approximate) computation of MPC
- Theoretical issues:
  - Feasibility
  - Convergence and stability
  - Solution algorithms (=computations)

(Mayne, Rawlings, Rao, Scokaert, 2000)

Feasibility

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

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

QUADRATIC PROGRAM (QP)

- **Feasibility**: Guarantee that the QP problem is feasible at all sampling times t
- **Input constraints only**: no feasibility issues !
- **Hard output constraints:**
  - When $N<\infty$ there is no guarantee that the QP problem will remain feasible at all future time steps t
  - $N=\infty$ ⟹ infinite number of constraints !
  - Maximum output admissible set theory: $N<\infty$ is enough !

(Gilbert, Tan, *IEEE TAC*,1991), (Kerrigan, Maciejowski,*CDC*,2000), (Chmielewski, Manousiouthakis, *Sys. Cont. Letters*, 1996)

Stability

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

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

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 result

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

$$\begin{aligned} V^*(x(t)) = \min \quad & \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \\ \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k \\ & u_{\min} \leq u_k \leq u_{\max} \\ & y_{\min} \leq Cx_k \leq y_{\max} \\ & x_N = 0 \end{aligned}$$

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 $t\geq 0$, for all $R,Q>0$

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

More 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 U_t = optimal control sequence at time t, $U_t = [u_0^t \ \dots \ u_{N-1}^t]'$
- By construction $\bar{U}_{t+1} = [u_1^t \ \dots \ u_{N-1}^t \ 0]'$ is a feasible sequence at time $t+1$
- The cost of $\bar{U}_{t+1}$ 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 constraint, infinite prediction horizon: $N \to \infty$

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

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

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

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

   (Scokaert and Rawlings, 1996)

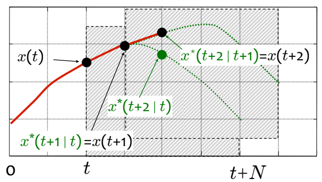
4. Contraction constraint: $\|x_{k+1}\| \leq \alpha \|x(t)\|, \ \alpha < 1$

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

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

Predicted and actual trajectories

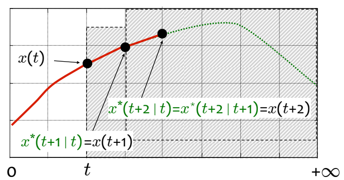
- Even assuming perfect model & no disturbances:



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

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

Richard Bellman (1920 - 1984)



At time $t+1$ the input sequence $\{u^*(t+1|t), u^*(t+2|t), \dots\}$ is also the optimal sequence for the sub-problem with initial state $x^*(t+1|t) = x(t+1)$. Therefore $x^*(t+k|t+1) = x^*(t+k|t)$, $\forall k \geq 1$, $\forall t \geq 0$

Input and output horizons

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

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

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

typically N_u=1÷10

MPC and Linear Quadratic Regulation (LQR)


Jacopo Francesco Riccati (1676 - 1754)

- Consider again the MPC control law based on minimizing

$$J(x_0,U)=\min_U x_N'Px_N + \sum_{k=0}^{N-1} x_k'Qx_k + u_k'Ru_k$$

- Given $R=R'\succ 0,\ Q=Q'\succeq 0$, choose matrix P by solving the Riccati equation

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

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

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

MPC and Linear Quadratic Regulation (LQR)

- Consider again the constrained MPC law based on minimizing


Jacopo Francesco Riccati (1676 - 1754)

$$\min_U\ x_N'Px_N + \sum_{k=0}^{N-1} x_k'Qx_k + u_k'Ru_k$$

$$\begin{aligned}
\text{s.t. } & u_{\min} \leq u_k \leq u_{\max},\ k=0,\ldots,N-1 \\
& y_{\min} \leq y_k \leq y_{\max},\ k=1,\ldots,N \\
& u_k = Kx_k,\ k=N_u,\ldots,N-1
\end{aligned}$$

- 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 = LQR** (for any choice of the prediction and control horizons N,N_u)
  (Sznaiera and Damborg, 1987)(Chmielewski and Manousiouthakis, 1996) (Scokaert and Rawlings, 1998)
- The larger the horizon, the larger the region where MPC=LQR

Double integrator example

- System: $y(t)=\frac{1}{s^2}u(t)$ → sampling + ZOH T_s=1 s

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

- Constraints: $-1\leq u(t)\leq 1$
- Control objective: $\min \sum_{k=0}^{\infty} y_k^2+\frac{1}{100}u_k^2$

$u_k = K_{LQ}x_k,\ \forall k\geq N_u$ (LQ gain)

$N_u=N=2$

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

- Optimization problem

$H=\begin{bmatrix}0.8365 & 0.3603\\ 0.3603 & 0.2059\end{bmatrix},\ F=\begin{bmatrix}0.4624 & 1.2852\\ 0.1682 & 0.5285\end{bmatrix}$ (cost function is normalized by max svd(H))

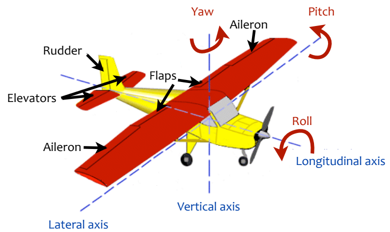
$G=\begin{bmatrix}1 & 0\\ -1 & 0\\ 0 & 1\\ 0 & -1\end{bmatrix},\ W=\begin{bmatrix}1\\1\\1\\1\end{bmatrix},\ S=\begin{bmatrix}0 & 0\\ 0 & 0\\ 0 & 0\\ 0 & 0\end{bmatrix}$

Example: AFTI-16

- Linearized model:

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

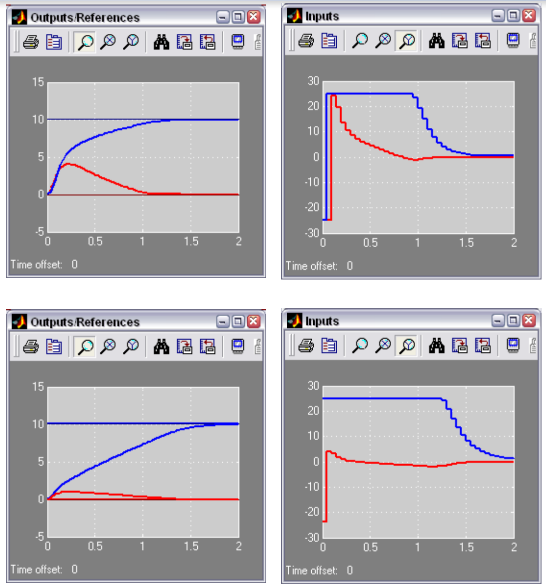
- Inputs: elevator and flaperon angle
- Outputs: attack and pitch angle
- Sampling time: $T_s = .05$ s (+ zero-order hold)
- Constraints: max 25° on both angles
- Open-loop response: unstable
  (open-loop poles: $-7.6636, -0.0075 \pm 0.0556j, 5.4530$)



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

afti16.m (MPC-Tbx)

Example: AFTI-16



$N_y = 10,\ N_u = 3,$

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

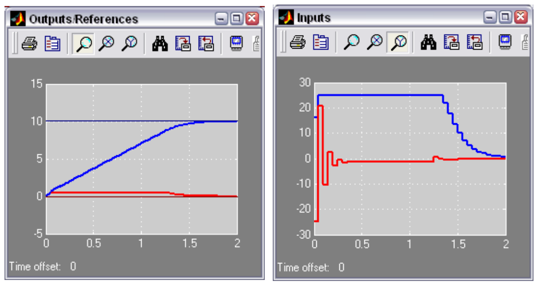
$u_{\min} = -25°,\ u_{\max} = 25°$

$N_y = 10,\ N_u = 3,$

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

$u_{\min} = -25°,\ u_{\max} = 25°$

Example: AFTI-16



$N_y = 10,\ N_u = 3,$

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

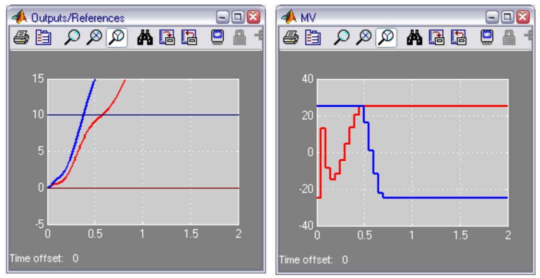
$u_{\min} = -25°,\ u_{\max} = 25°,$

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

Example: AFTI-16

Unconstrained MPC (=linear controller, ≈ LQR)

+ actuator saturation ±25°



$N_y = 10,\ N_u = 3,$

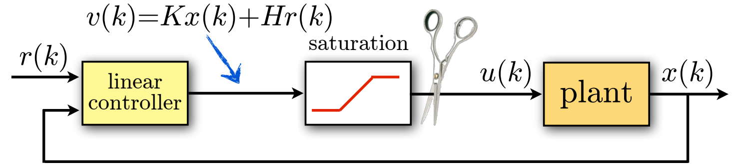
$w_y = \{10, 10\},\ w_{\delta u} = \{.01, .01\},$

UNSTABLE !!!

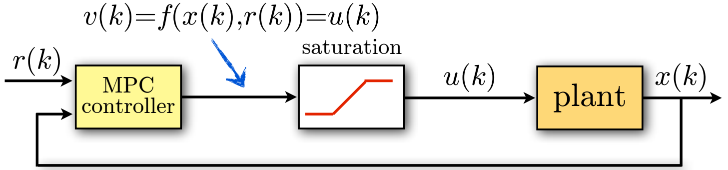
Saturation needs to be considered in the control design !

Saturation

- Saturation is dangerous because it breaks the control loop



- MPC takes it into account automatically (and optimally)



Tuning guidelines

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

$$\text{subj. to } \Delta u_{\min} \leq \Delta u_k \leq \Delta u_{\max},\ k=0,\ldots,N_u-1$$
$$\Delta u_k = 0,\ k = N_u,\ldots,N-1$$
$$u_{\min} \leq u_k \leq u_{\max},\ k=0,\ldots,N_u-1$$
$$y_{\min} - \epsilon V_{\min} \leq y_k \leq y_{\max} + \epsilon V_{\max},\ k=1,\ldots,N$$

- **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 the more complex the controller
- **Output horizon:** the smaller N, the more aggressive the controller
- **Limits:** controller less aggressive if $\Delta u_{\min}$, $\Delta u_{\max}$ are small

Always try to set N_u as small as possible !

Scaling

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

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

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

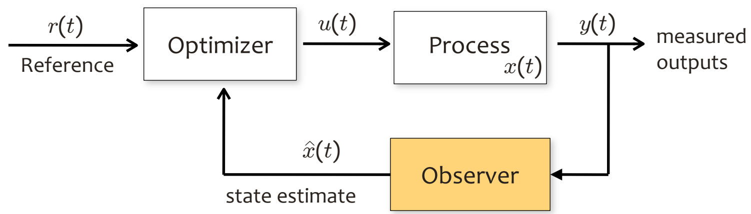
use instead: $y_1 \in$[-0.1,0.1] (mV)

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

- Ideally all variables should range in [-1,1].

  For example, one can replace y with $y/y_{\max}$

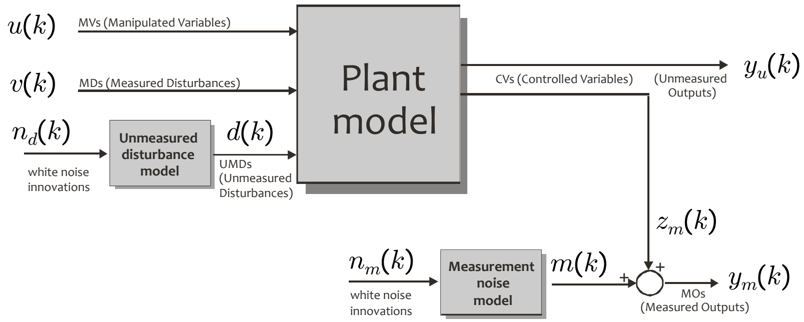
Observer design for MPC



- 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 !

Observer design

- Measurement update

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

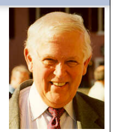
- Time update

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

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

(Muske, Meadows, Rawlings, ACC94)

Kalman filter design



Rudolf Emil Kalman (1930 -)

- Full model for designing observer gain M

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

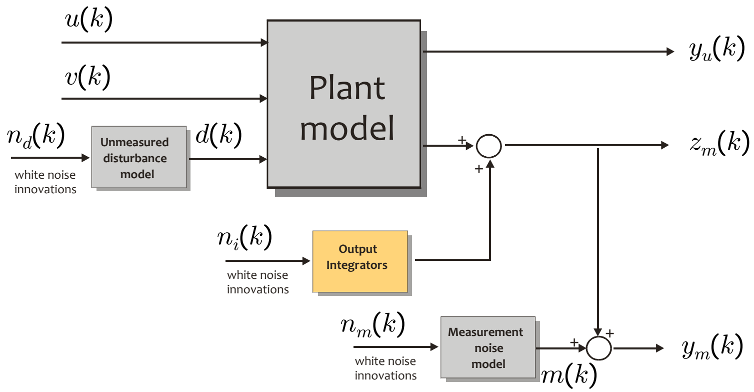
$$y_m(k) = \begin{bmatrix} C_m & D_{dm}\bar{C} & \tilde{C} \end{bmatrix}\begin{bmatrix} x(k) \\ x_d(k) \\ x_m(k) \end{bmatrix} + D_{vm}v(k) + \bar{D}_m n_d(k) + \tilde{D}_m n_m(k)$$

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

Integral Action in MPC

(and not only in MPC)

Output Integrators

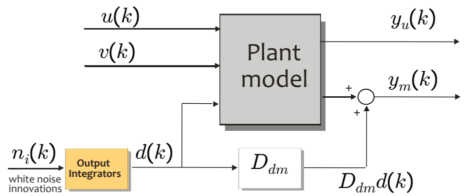


- Introduce output integrators as additional disturbance models
- Under certain conditions, observer + controller provide **zero offset** in steady-state

Integrators and steady-state offsets

- More generally, add integrators on states + outputs:

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



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

- Use the above model + meas.noise model to design an observer (e.g. Kalman filter)
- **Main idea:** observer makes $y_m - (C_m\hat{x} + D_{dm}\hat{d}) \to 0$ (estimation error)

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

  $\Rightarrow$ the combination makes $y_m \to r$ (actual tracking error)
- Explanation: $D_{dm}\hat{d}$ compensates model mismatch in steady-state

Error feedback

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

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

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

- Implementation:

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

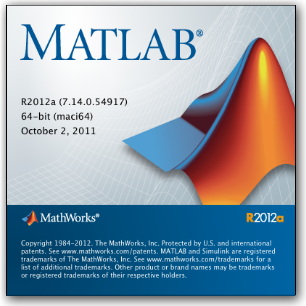
$$u(k) = f_{\rm MPC}\left(\begin{bmatrix} x(k) \\ q(k) \\ r(k) \end{bmatrix}\right)$$

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

Model Predictive Control Toolbox

(Bemporad, Ricker, Morari, 1998-2012)

- **MPC Toolbox 4.0** (The Mathworks, Inc.)
  - Object-oriented implementation (MPC object)
  - MPC Simulink Library
  - MPC Graphical User Interface
  - Code generation [RTW, xPC Target, dSpace, etc.]
  - Linked to OPC Toolbox, System ID Toolbox, ...



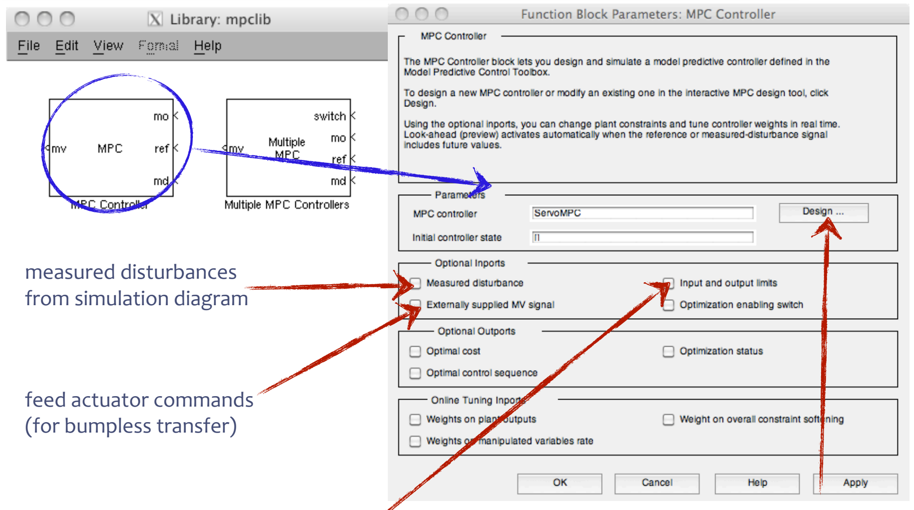
http://www.mathworks.com/products/mpc/

Model Predictive Control Toolbox

(Bemporad, Ricker, Morari, 1998-present)

- Several **linear MPC** design features available:
  - preview on references/measured disturbances
  - time-varying weights and constraints, non-diagonal weights
  - integral action for offset-free tracking
  - soft constraints
- Prediction models generated by Identification Toolbox supported
- Automatic linearization of prediction models from Simulink diagrams
- Linear stability/frequency analysis of closed-loop (inactive constraints)
- Very fast command-line closed-loop simulation (compiled EML-code), with very versatile simulation options (e.g., for analysis of model mismatch effects)

MPC Simulink library



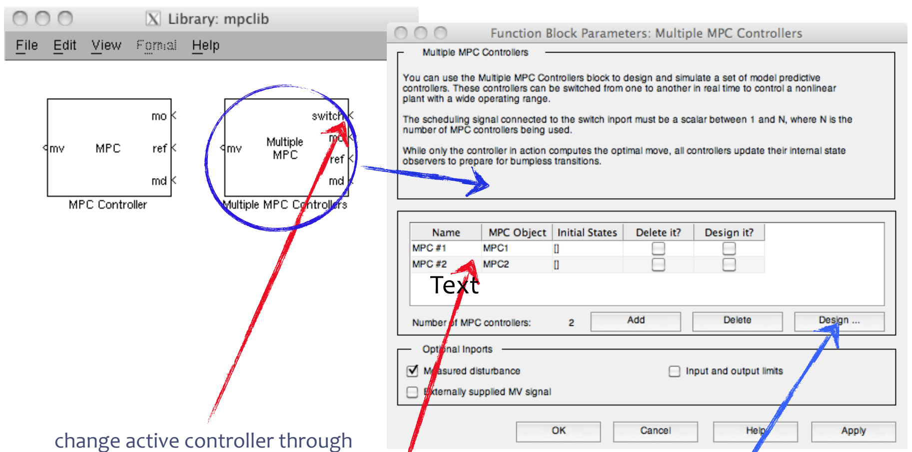
measured disturbances from simulation diagram

feed actuator commands (for bumpless transfer)

input and output limits change during simulation

open MPC GUI for design

MPC Simulink library

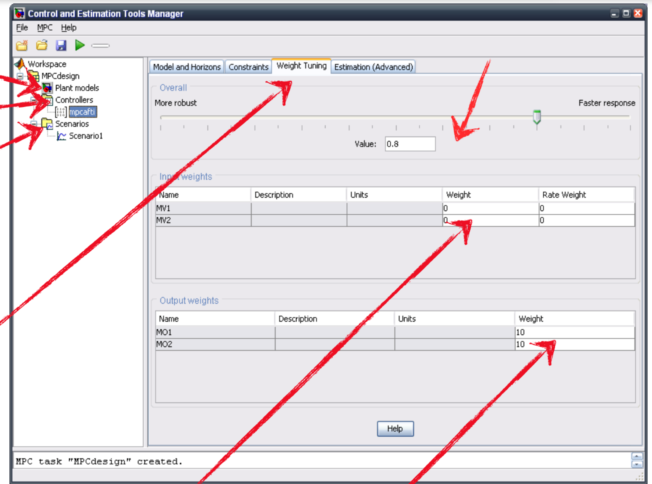


change active controller through an integer switching signal

specify a set of N linear MPC controllers

open MPC GUI for design

MPC Graphical User Interface



linear prediction models

MPC designs

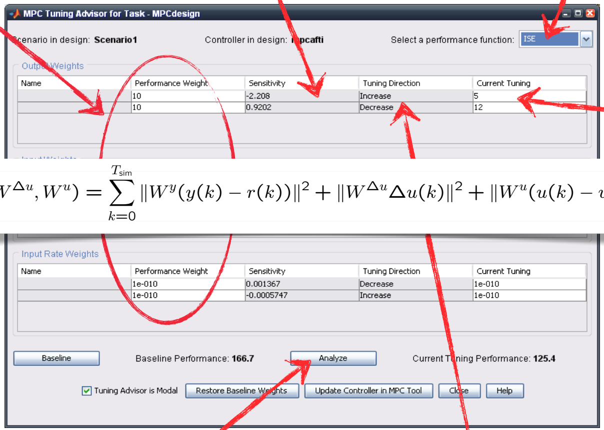
simulation trials with different combinations of controllers, plants, references, etc.

MPC tuning panel

input weights

output weights

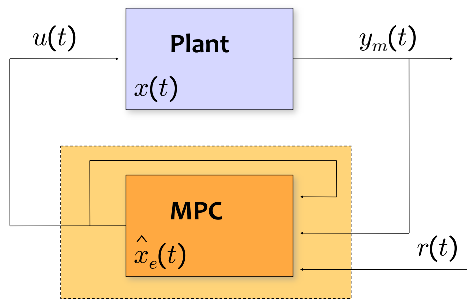
MPC Tuning Advisor



$$J(W^y, W^{\Delta u}, W^u) = \sum_{k=0}^{T_{\rm sim}} \|W^y(y(k)-r(k))\|^2 + \|W^{\Delta u}\Delta u(k)\|^2 + \|W^u(u(k)-u_r(k))\|^2$$

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

$$-\left[\,I\;0\;\ldots\;0\,\right] H^{-1}F = K_{fv}$$

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

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

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

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

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

Controller matching problem - Example

- Open-loop process: $y(k) = 1.8y(k-1) + 1.2y(k-2) + u(k-1)$
- Constraints: $-24 \leq u(k) \leq 24$
- Desired controller (**PID**): $K_I = 0.248,\; K_P = 0.752,\; K_D = 2.237$

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

- State-space form:

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

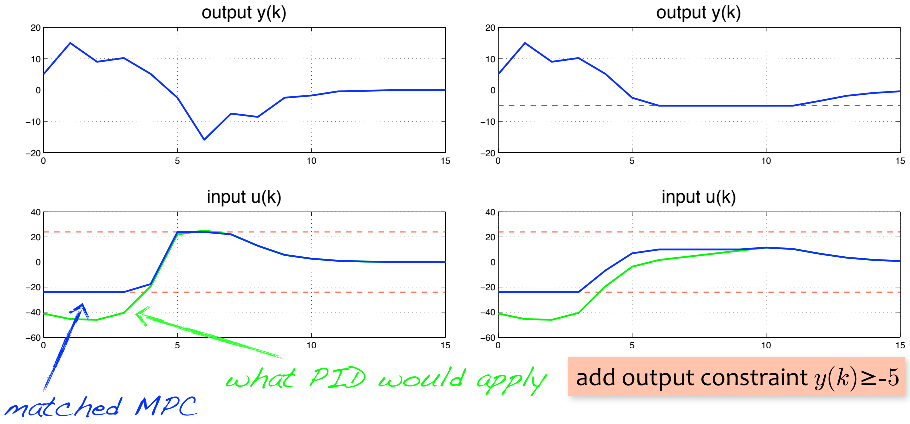
controller matching based on inverse LQR

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

$$R^* = 1$$

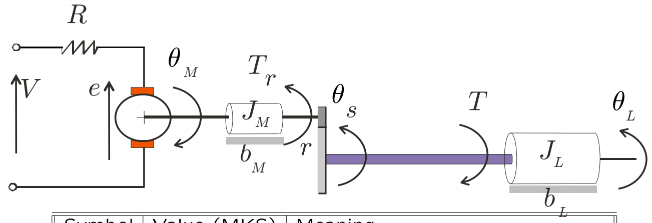
$$P^* = \begin{bmatrix} 422.7 & 241.7 & 50.39 & 201.4\\ 241.7 & 151.0 & 32.13 & 120.4\\ 50.39 & 32.13 & 19.85 & 26.75\\ 201.4 & 120.4 & 26.75 & 106.6\end{bmatrix}$$

Controller matching problem - Example



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

Example: MPC of a DC servomotor

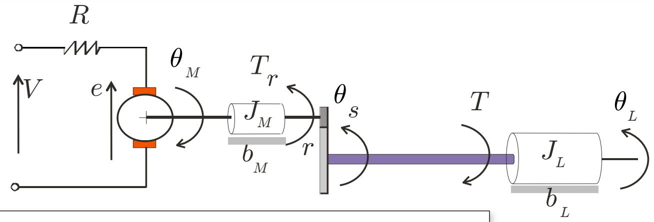


Symbol	Value (MKS)	Meaning
L_S	1.0	shaft length
d_S	0.02	shaft diameter
J_S	negligible	shaft inertia
J_M	0.5	motor inertia
β_M	0.1	motor viscous friction coefficient
R	20	resistance of armature
K_T	10	motor constant
ρ	20	gear ratio
k_θ	1280.2	torsional rigidity
J_L	$20J_M$	nominal load inertia
β_L	25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

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

mpcmotor.m (MPC-Tbx)

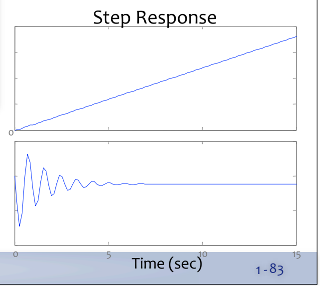
DC servomotor - Model



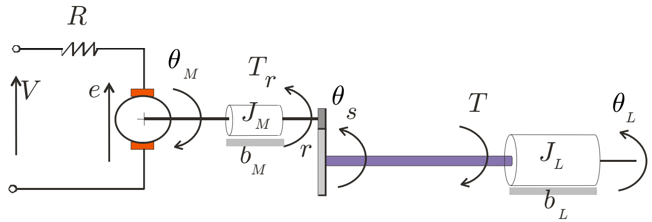
$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_\theta}{J_L} & -\frac{\beta_L}{J_L} & \frac{k_\theta}{\rho J_L} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k_\theta}{\rho J_M} & 0 & -\frac{k_\theta}{\rho^2 J_M} & -\frac{\beta_M + k_T^2/R}{J_M} \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{k_T}{R J_M}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

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

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

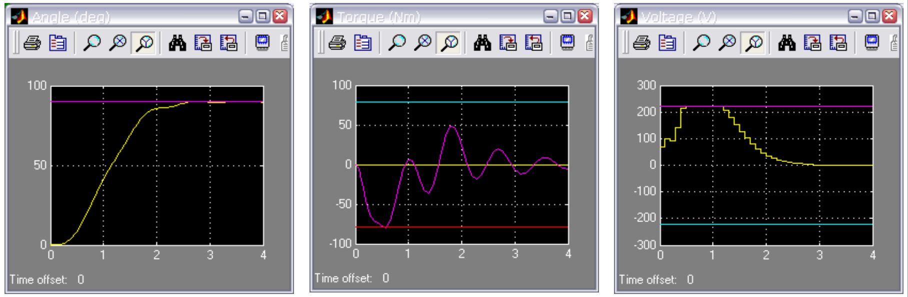


DC servomotor - Specs



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

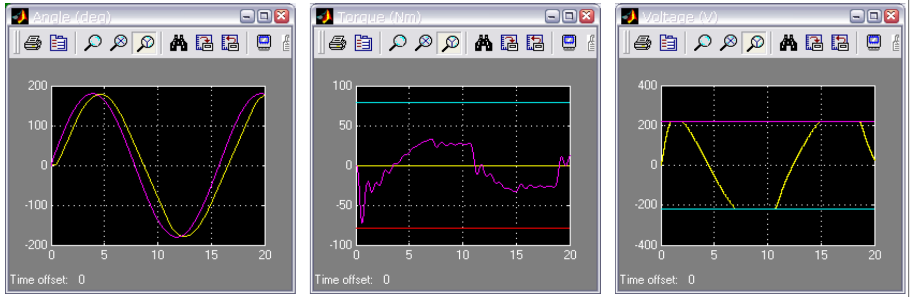
DC servomotor - MPC results



$r(t) \equiv 90$ deg

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

DC servomotor - MPC results



$r(t) = 180 \sin(0.4t)$ deg

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

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

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

$$
\begin{array}{ll}
\min\limits_{U} & \|Px_N\|_\infty + \sum\limits_{k=0}^{N-1} \|Qx_k\|_\infty + \|Ru_k\|_\infty \\
\text{s.t.} & u_{\min} \leq u_k \leq u_{\max}, \ k = 0, \ldots, N-1 \\
 & y_{\min} \leq y_k \leq y_{\max}, \ k = 1, \ldots, N
\end{array}
$$

Q, R, P full rank

$\|v\|_\infty \triangleq \max\limits_{i=1,\ldots,n} |v_i|$

Linear MPC based on LP

- Basic trick:

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

- Introduce slack vars:

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

Q^i = ith 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:

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

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

LINEAR PROGRAM (LP)

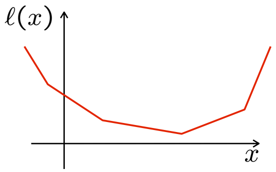
- $z \triangleq [\epsilon_0^u \ \ldots \ \epsilon_{N-1}^u \ \epsilon_1^x \ \ldots \ \epsilon_N^x \ u_0', \ldots, u_{N-1}']' \in \mathbb{R}^s$, $s \triangleq N(m+2)$, is the optimization vector
- G, W, S are obtained from weights Q, R, P, and model matrices A, B, C
- Q, R, P can be selected to guarantee closed-loop stability

(Bemporad, Borrelli, Morari, 2003)

Extension to arbitrary convex PWA functions

- Constrained optimal control problem:

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



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

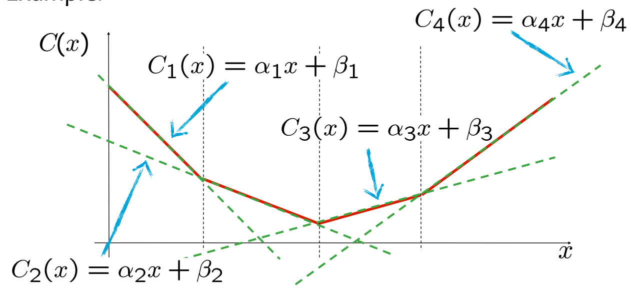
(Schechter, 1987)

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

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

Convex PWA costs

Example:

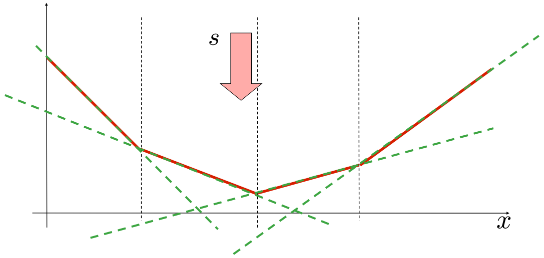


It is easy to see that:

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

Convex PWA optimization problems and LP

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



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

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

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)
- 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

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

additional slack vars to represent convex PWA costs