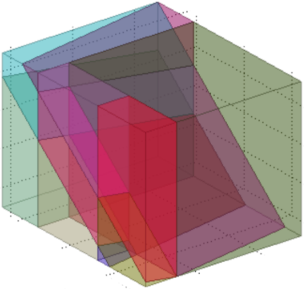
MODEL PREDICTIVE CONTROL

Alberto Bemporad

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

Academic year 2017-2018



IMT SCHOOL FOR ADVANCED STUDIES LUCCA

COURSE STRUCTURE

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

MATLAB Toolboxes:

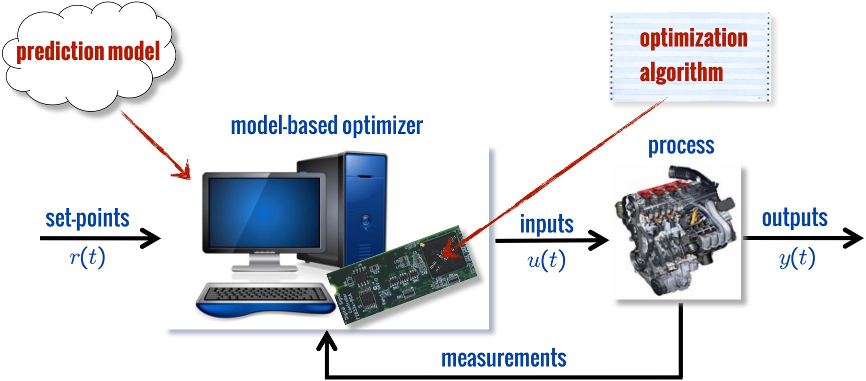
- **MPC Toolbox** (linear/explicit/parameter-varying MPC)
- **Hybrid Toolbox** (explicit MPC, hybrid systems)

Course page:

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

MODEL PREDICTIVE CONTROL: BASIC CONCEPTS

MODEL PREDICTIVE CONTROL (MPC)



Use a simplified dynamical **model** of the process to **predict** its likely future evolution and choose ~~the “best” control~~ a good action

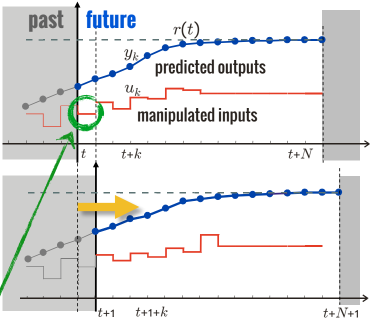
MODEL PREDICTIVE CONTROL (MPC)

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

$$\min \sum_{k=0}^{N-1} \overbrace{\|W^y(y_k - r(t))\|_2^2}^{\text{penalty on tracking error}} + \overbrace{\|W^u(u_k - u_r(t))\|_2^2}^{\text{penalty on actuators}}$$

$$\begin{aligned} \text{s.t. } & x_{k+1} = f(x_k, u_k, t) \\ & y_k = g(x_k, u_k, t) \\ & \text{constraints on } u_k, y_k \\ & x_0 = x(t) \Leftarrow \text{feedback !} \end{aligned}$$

➡ **optimization problem**

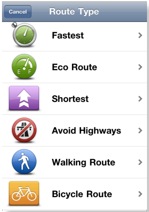


- Solve the resulting optimization problem with respect to $\{u_0, \ldots, u_{N-1}\}$
- Apply only the first optimal move $u(t) = u_0^*$ and discard the remaining samples
- At time $t+1$: Get new measurements, repeat the optimization. And so on ...

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

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

MPC is the standard for advanced control in the process industry

- Research in MPC is still very active !

MPC IN INDUSTRY

(Qin, Badgewell, 2003)

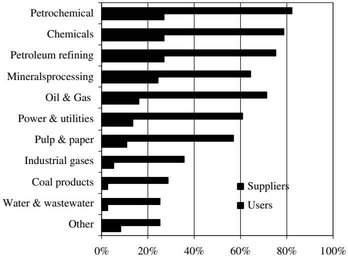
- Industrial survey of MPC applications conducted in mid 1999

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	—	

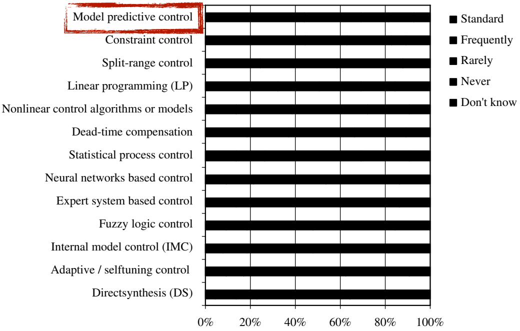
Estimates based on vendor survey

(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

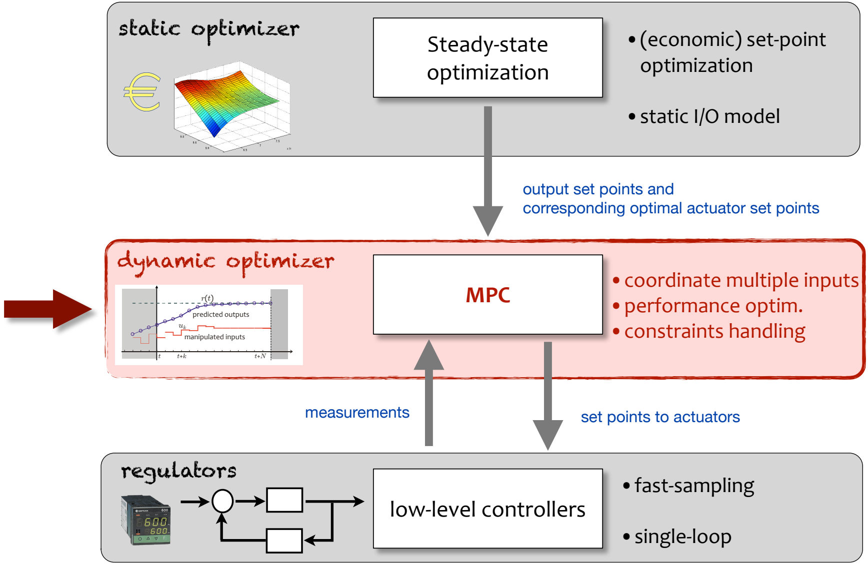
(Samad, IEEE CS Magazine, 2017)

- 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



AUTOMOTIVE APPLICATIONS OF MPC

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

Powertrain

engine control, magnetic actuators, robotized gearbox, power MGT in HEVs, cabin heat control, electrical motors

Vehicle dynamics

traction control, active steering, semiactive suspensions, autonomous driving

Ford Motor Company

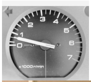
Jaguar

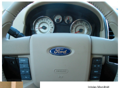
DENSO Automotive

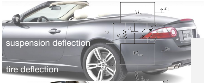
FCA

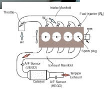
General Motors

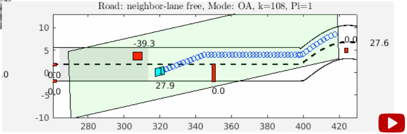
ODYS
Advanced Controls & Optimization









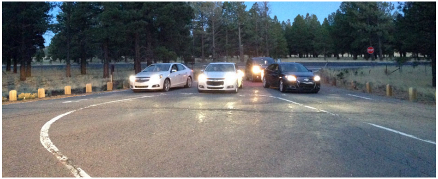


Most automotive OEMs are looking into MPC solutions today

MPC GOES TO AUTOMOTIVE PRODUCTION THIS YEAR !

General Motors and **ODYS** have developed a multivariable constrained MPC system for torque tracking in turbocharged gasoline engines. The control system is **scheduled for production by GM in 2018**.

- Multivariable system, **4 inputs, 4 outputs**. On-line QP solved in **real-time on ECU**
  (Bemporad, Bernardini, Long, Verdejo, 2018)



- Supervisory **MPC for powertrain control** also in production in 2018
  (Bemporad, Bernardini, Livshiz, Pattipati, 2018)

First known mass production of MPC in the automotive industry

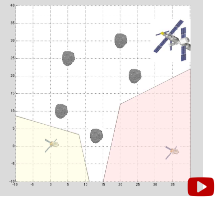
http://www.odys.it/odys-and-gm-bring-online-mpc-to-production/

ODYS
Advanced Controls & Optimization

AEROSPACE APPLICATIONS OF MPC

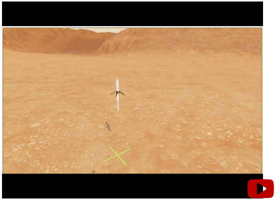
- MPC capabilities explored in new space applications

cooperating UAVs



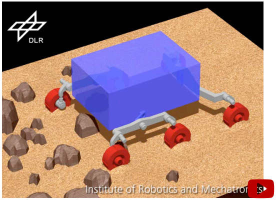
(Bemporad, Rocchi, 2011)

powered descent



(Pascucci, Bennani, Bemporad, 2016)

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.

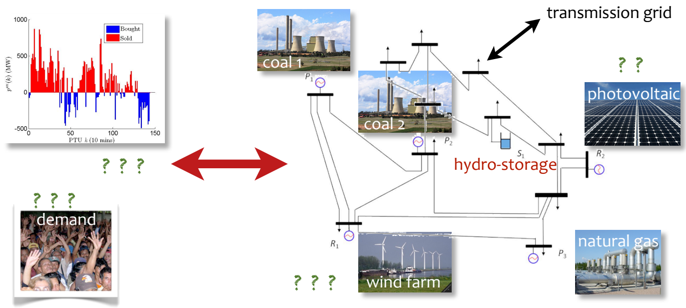
The AMVC, which uses a proprietary model predictive control methodology, is the most technically advanced propulsion system control ever produced by the aerospace industry, demonstrating the highest pilot rating for flight performance and providing independent control of vertical thrust and pitch from 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

(Patrinos, Trimboli, Bemporad, 2011)



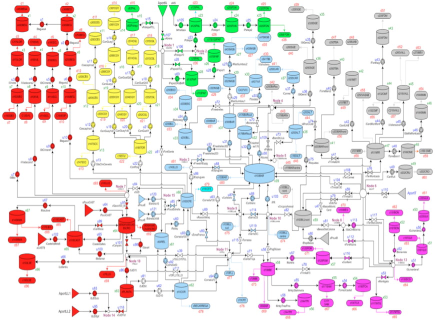
Dispatch power in smart distribution grids, trade energy on energy markets

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)

MPC OF DRINKING WATER NETWORKS

(Sampathirao, Sopasakis, Bemporad, Patrinos, 2017)



Drinking water network of Barcelona:

63 tanks

114 controlled flows

17 mixing nodes

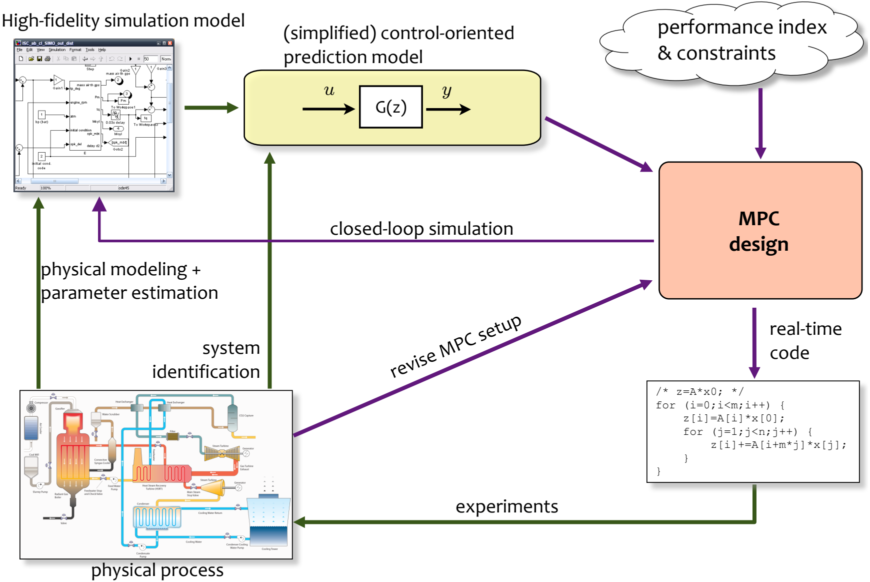
FP7-ICT project “EFFINET - Efficient Integrated Real-time Monitoring and Control of Drinking Water Networks” (2012-2015)

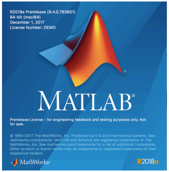
EFFINET

MPC RESEARCH IS DRIVEN BY APPLICATIONS

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

MPC DESIGN FLOW





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

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

7200 downloads

1.5 downloads/day

- **ODYS MPC Toolbox**: (Bemporad, Bernardini, Cimini 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: **multivariable**, w/**delays**, **time-varying**, **uncertain**, **stochastic**, **nonlinear**, ...
  - Handles **constraints** on inputs and outputs
  - Can exploit **preview** on future references and measured disturbances
  - **Tuning** similar to LQR (Linear Quadratic Regulator)
  - Mature **offline design tools** and **real-time code** are available

$$
\begin{aligned}
\min \quad & \sum_{k=0}^{N-1} \|W^y(y_k - r(t))\|_2^2 \\
& \qquad + \|W^u(u_k - u_r(t))\|_2^2 \\
\text{s.t.} \quad & x_{k+1} = f(x_k, u_k) \\
& y_k = g(x_k, u_k \\
& h(u_k, x_k, y_k) \leq 0
\end{aligned}
$$

- Price to pay:
  - Requires a (simple) **model** (experiments + system identification, linearization, ...)
  - Many **design/calibration knobs** (weights, horizon, constraints, ...)
  - Requires **real-time computations** to solve the optimization problem

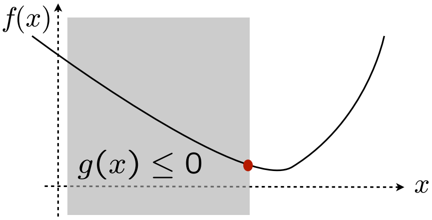
BASICS OF CONSTRAINED OPTIMIZATION

MATHEMATICAL PROGRAMMING

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

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

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



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

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

OPTIMIZATION SOFTWARE

- Comparison on benchmark problems:

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

- Taxonomy of many solvers for different classes of optimization problems:

  `http://www.neos-guide.org`

- NEOS server for remotely solving optimization problems:

  `http://www.neos-server.org`

- Good open-source optimization software:

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

- GitHub, MATLAB Central, Google, ...

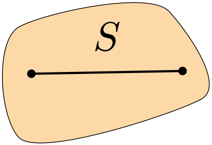
CONVEX SETS

DEFINITION

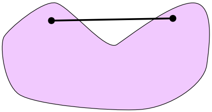
A set $S \subseteq \mathbb{R}^n$ is **convex** if for all $x_1, x_2 \in S$

$$\lambda x_1 + (1-\lambda)x_2 \in S,\ \forall \lambda \in [0,1]$$

convex set



nonconvex set



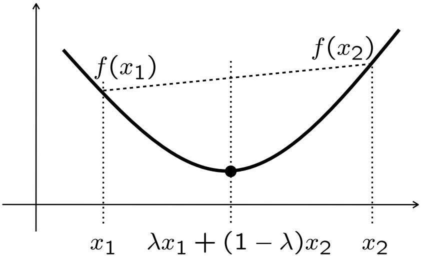
CONVEX FUNCTIONS

DEFINITION

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

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$
$$\forall x_1, x_2 \in S,\ \lambda \in [0,1]$$

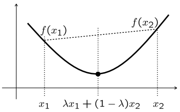
Jensen's inequality



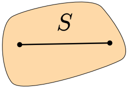
CONVEX OPTIMIZATION PROBLEM

The optimization problem

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



is a **convex optimization problem** if S is a convex set and $f : S \to \mathbb{R}$ is a convex function



- Often S is defined by **linear equality constraints** $Ax = b$ and **convex inequality constraints** $g(x) \leq 0$, $g : \mathbb{R}^n \to \mathbb{R}^m$ convex
- Every local solution is also a global one (we will see this later)
- Efficient solution algorithms exist
- Often occurring in many problems in engineering, economics, and science

Excellent textbook: "Convex Optimization" (Boyd, Vandenberghe, 2002)

POLYHEDRA

DEFINITION

Convex **polyhedron** = intersection of a finite set of half-spaces of $\mathbb{R}^n$

Convex **polytope** = bounded convex polyhedron

- **Hyperplane (H-)representation**:

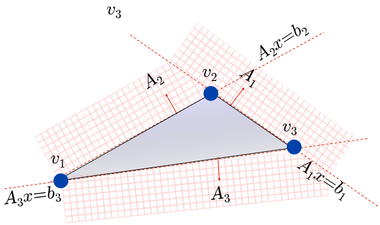
$$P = \{x \in \mathbb{R}^n : \ Ax \le b\}$$

- **Vertex (V-)representation**:

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

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

when $q = 0$ the polyhedron is a **cone**



Convex hull = transformation from V- to H-representation

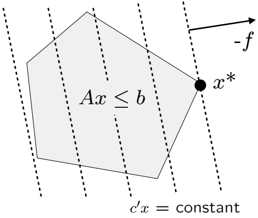
Vertex enumeration = transformation from H- to V-representation

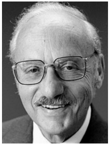
v_i = **vertex**, r_j = **extreme ray**

LINEAR PROGRAMMING

- Linear programming (LP) problem:

$$\begin{array}{rl} \min & c'x \\ \text{s.t.} & Ax \le b,\ x \in \mathbb{R}^n \end{array}$$





George Dantzig
(1914–2005)

- LP in standard form:

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

- Conversion to standard form by introducing slack variables

$$\left\{ \begin{array}{l} \sum_{j=1}^{n} a_{ij}x_j \le b_i \\ x_j \ge 0 \end{array} \right. \Rightarrow \left\{ \begin{array}{l} \sum_{j=1}^{n} a_{ij}x_j + s_i = b_i \\ s_i, x_j \ge 0 \end{array} \right.$$

- Nonnegative variables by splitting positive and negative part of x

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

LINEAR PROGRAMMING (LP)

- Converting maximization to minimization

$$\max_x c'x = -(\min_x -c'x)$$

- Equalities to double inequalities (not recommended)

$$\sum_{j=1}^n a_{ij}x_j = b_i \;\Rightarrow\; \left\{ \begin{array}{l} \sum_{j=1}^n a_{ij}x_j \leq b_i \\ \sum_{j=1}^n a_{ij}x_j \geq b_i \end{array} \right.$$

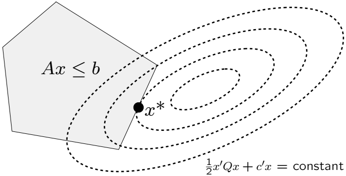
- Change direction of an inequality

$$\sum_{j=1}^n a_{ij}x_j \geq b_i \;\Rightarrow\; \sum_{j=1}^n -a_{ij}x_j \leq -b_i$$

An LP can be always formulated using "$\min$" and "$\leq$"

- Quadratic programming (QP) problem:

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



- Convex optimization problem if $Q \succeq 0$ (Q = positive semidefinite matrix) [1]

- Hard problem if $Q \not\succeq 0$ (Q = indefinite matrix)

---

[1] A matrix $P \in \mathbb{R}^{n\times n}$ is **positive semidefinite** ($P \succeq 0$) if $x'Px \geq 0$ for all x.
It is **positive definite** ($P \succ 0$) if in addition $x'Px > 0$ for all $x \neq 0$.
It is **negative (semi)definite** ($P \prec 0$, $P \preceq 0$) if $-P$ is positive (semi)definite.
It is **indefinite** otherwise.

MIXED-INTEGER PROGRAMMING (MIP)

$$\begin{aligned} \min \quad & c'x \\ \text{s.t.} \quad & Ax \le b,\ x = \left[\begin{smallmatrix} x_c \\ x_b \end{smallmatrix}\right] \\ & x_c \in \mathbb{R}^{n_c},\ x_b \in \{0,1\}^{n_b} \end{aligned}$$

mixed-integer linear program (MILP)

$$\begin{aligned} \min \quad & \frac{1}{2}x'Qx + c'x \\ \text{s.t.} \quad & Ax \le b,\ x = \left[\begin{smallmatrix} x_c \\ x_b \end{smallmatrix}\right] \\ & x_c \in \mathbb{R}^{n_c},\ x_b \in \{0,1\}^{n_b} \end{aligned}$$

mixed-integer quadratic program (MIQP)

- Some variables are real, some are binary (0/1)
- A $\mathcal{NP}$-hard problem, in general
- Many good solvers are available (CPLEX, Gurobi, GLPK, Xpress-MP, CBC, ...)
  For comparisons see `http://plato.la.asu.edu/bench.html`

MODELING LANGUAGES FOR OPTIMIZATION PROBLEMS

- **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 FOR OPTIMIZATION PROBLEMS

- **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

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

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

Notation:

$x_0 = x(t)$

$x_k = x(t+k|t)$

$u_k = u(t+k|t)$

- Relation between input and states: $x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j}$

- Performance index

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

$$\begin{aligned} R &= R' \succ 0 \\ Q &= Q' \succeq 0 \\ P &= P' \succeq 0 \end{aligned} \qquad z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

- **Goal:** find the sequence z^* that minimizes $J(z, x_0)$, i.e., that steers the state x to the origin optimally

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

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

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

condensed form of MPC

- 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" solution)

- **Alternative #1:** find z^* via **dynamic programming** (Riccati iterations)
- **Alternative #2:** keep also $x_1, \ldots, x_N$ as optimization variables and the equality constraints $x_{k+1} = Ax_k + Bu_k$ (**non-condensed form**)

CONSTRAINED LINEAR MPC

- Linear prediction model: $\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \end{cases}$ $\quad x \in \mathbb{R}^n, \; u \in \mathbb{R}^m, \; y \in \mathbb{R}^p$
- 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 (quadratic performance index):

$$\begin{array}{ll} \min\limits_{z} & x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \\ \text{s.t.} & u_{\min} \leq u_k \leq u_{\max}, \; k = 0, \ldots, N-1 \\ & y_{\min} \leq y_k \leq y_{\max}, \; k = 1, \ldots, N \end{array}$$

$$\begin{array}{rcl} R & = & R' \succ 0 \\ Q & = & Q' \succeq 0 \\ P & = & P' \succeq 0 \end{array} \qquad z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

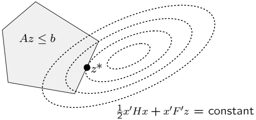
CONSTRAINED LINEAR MPC

- Linear prediction model: $x_k = A^k x_0 + \sum_{i=0}^{k-1} A^i B u_{k-1-i}$
- Optimization problem (condensed form):

$$
\begin{aligned}
V(x_0) = \tfrac{1}{2} x_0' Y x_0 + \quad & \min_z \quad \tfrac{1}{2} z' H z + x_0' F' z && \text{(quadratic objective)} \\
& \text{s.t.} \quad Gz \leq W + S x_0 && \text{(linear constraints)}
\end{aligned}
$$

convex Quadratic Program (QP)

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

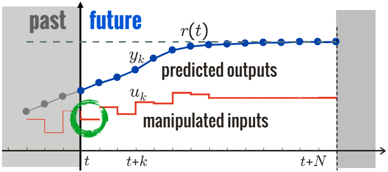
COMPUTATION OF CONSTRAINT MATRICES

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

$$\left\{\begin{array}{rcl} u_k & \leq & u_{\max} \\ -u_k & \leq & -u_{\min} \end{array}\right. \quad \Rightarrow \quad \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & 1 \\ -1 & 0 & \ldots & 0 \\ 0 & -1 & \ldots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & -1 \end{bmatrix} z \leq \begin{bmatrix} u_{\max} \\ u_{\max} \\ \vdots \\ u_{\max} \\ -u_{\min} \\ -u_{\min} \\ \vdots \\ -u_{\min} \end{bmatrix} \qquad z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

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

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



@ each sampling step t:

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

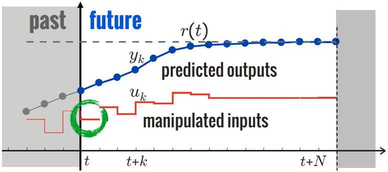
- Get the solution $z^* = \begin{bmatrix} u_0^* \\ u_1^* \\ \vdots \\ u_{N-1}^* \end{bmatrix}$ of the QP

$$\left\{ \begin{array}{ll} \min\limits_z & \frac{1}{2} z' H z + \overbrace{x'(t) F' z}^{\text{feedback}} \\ \text{s.t.} & Gz \leq W + S \underbrace{x(t)}_{\text{feedback}} \end{array} \right.$$

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

UNCONSTRAINED LINEAR MPC ALGORITHM

@ each sampling step t:



- Minimize quadratic function, no constraints

$$\min_z \; f(z) = \frac{1}{2} z' H z + x'(t) F' z \qquad\qquad z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

- solution: $\nabla f(z) = Hz + Fx'(t) = 0 \;\Rightarrow\; z^* = -H^{-1} F x'(t)$

$$u(t) = -\begin{bmatrix} I & 0 & \dots & 0 \end{bmatrix} H^{-1} F x(t) = K x(t)$$

Unconstrained linear MPC = linear state-feedback!

DOUBLE INTEGRATOR EXAMPLE

- System: $y(t) = \iint u(\tau)d\tau$ (or equivalently $\frac{d^2y}{dt^2} = u$)
- Sampling with period $T_s = 1$ s: $\left\{\begin{array}{rcl} x(t+1) & = & \left[\begin{smallmatrix}1 & 1\\ 0 & 1\end{smallmatrix}\right] x(t) + \left[\begin{smallmatrix}0\\ 1\end{smallmatrix}\right] u(t) \\ y(t) & = & \left[\begin{smallmatrix}1 & 0\end{smallmatrix}\right] x(t)\end{array}\right.$
- Constraints: $-1 \leq u(t) \leq 1$
- Performance index: $\min \left(\sum_{k=0}^{1} y_k^2 + \frac{1}{10}u_k^2\right) + x_2' \left[\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right] x_2$
- QP matrices:

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

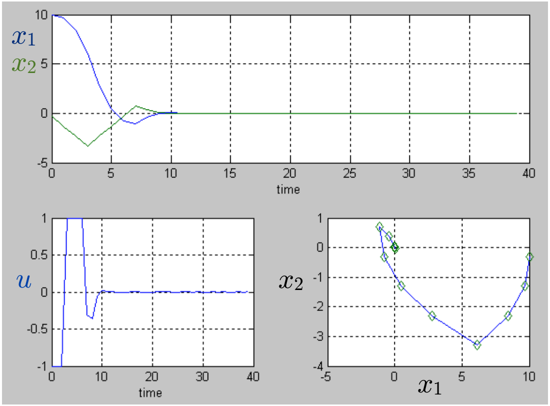
constraints: $Gz \leq W + Sx(t)$

$$H = \left[\begin{smallmatrix}4.2 & 2\\ 2 & 2.2\end{smallmatrix}\right], \ F = \left[\begin{smallmatrix}2 & 6\\ 0 & 2\end{smallmatrix}\right]$$

$$Y = \left[\begin{smallmatrix}4 & 6\\ 6 & 12\end{smallmatrix}\right], \ G = \begin{bmatrix}1 & 0\\ -1 & 0\\ 0 & 1\\ 0 & -1\end{bmatrix}$$

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

DOUBLE INTEGRATOR EXAMPLE



go to demo `linear/doubleint.m` (Hybrid Toolbox)
see also `mpcdoubleint.m` (MPC Toolbox)

DOUBLE INTEGRATOR EXAMPLE

- Add constraint on second state at prediction time $t+1$:

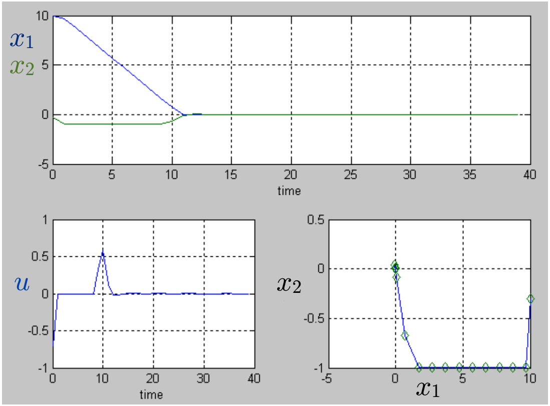
$$x_{2,k} \geq -1,\ k = 1$$

- New QP matrices:

$$H = \left[\begin{smallmatrix} 4.2 & 2 \\ 2 & 2.2 \end{smallmatrix}\right],\ F = \left[\begin{smallmatrix} 2 & 6 \\ 0 & 2 \end{smallmatrix}\right],\ Y = \left[\begin{smallmatrix} 4 & 6 \\ 6 & 12 \end{smallmatrix}\right]$$

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

DOUBLE INTEGRATOR EXAMPLE



LINEAR MPC - TRACKING

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

$$\Delta u(t) = u(t) - u(t-1)$$

- As $u(t) = u(t-1) + \Delta u(t)$ we need to extend the system with a new state $x_u(t) = u(t-1)$

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

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

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

- Optimal control problem (quadratic performance index):

$$\begin{array}{ll}\min_z & \sum_{k=0}^{N-1} \|W^y(y_{k+1}-r(t))\|_2^2 + \|W^{\Delta u}\Delta u_k\|_2^2 \\ & [\Delta u_k \triangleq u_k - u_{k-1}],\ u_{-1} = u(t-1) \\ \text{s.t.} & u_{\min} \leq u_k \leq u_{\max},\ k=0,\ldots,N-1 \\ & y_{\min} \leq y_k \leq y_{\max},\ k=1,\ldots,N \\ & \Delta u_{\min} \leq \Delta u_k \leq \Delta u_{\max},\ k=0,\ldots,N-1\end{array}$$

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

weight W = diagonal matrix (more generally, Cholesky factor of $Q = W'W$)

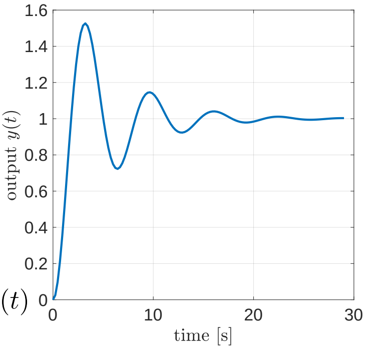
$$\begin{array}{ll}\min_z & J(z,x(t)) = \frac{1}{2}z'Hz + [x'(t)\ r'(t)\ u'(t-1)]F'z \\ \text{s.t.} & Gz \leq W + S\begin{bmatrix}x(t) \\ r(t) \\ u(t-1)\end{bmatrix}\end{array}$$

convex Quadratic Program

- Add the extra penalty $\|W^u(u_k - u_{\text{ref}}(t))\|_2^2$ to track **input references**
- Constraints may depend on $r(t)$, such as $e_{\min} \leq y_k - r(t) \leq e_{\max}$

LINEAR MPC TRACKING EXAMPLE

- System: $y(t) = \frac{1}{s^2+0.4s+1} u(t)$
  (or equivalently $\frac{d^2y}{dt^2} + 0.4\frac{dy}{dt} + y = u$)



- Sampling with period $T_s = 0.5$ s:

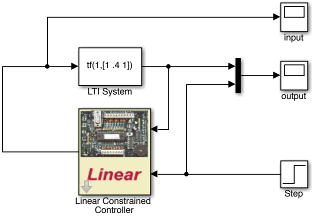
$$\begin{cases} x(t+1) &= \begin{bmatrix} 1.597 & -0.8187 \\ 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 0.2294 & 0.2145 \end{bmatrix} x(t) \end{cases}$$

go to demo `linear/example3.m` (Hybrid Toolbox)

LINEAR MPC TRACKING EXAMPLE

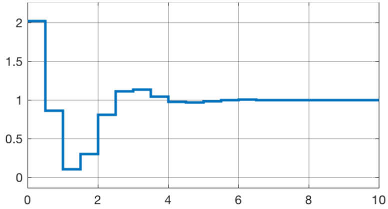
- Performance index:

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

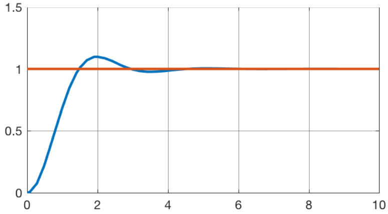


- Closed-loop MPC results:

$u(t)$

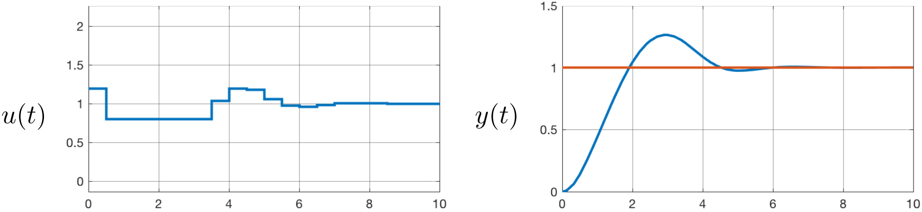


$y(t)$

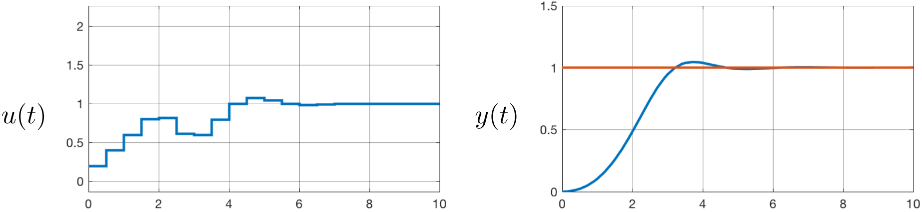


LINEAR MPC TRACKING EXAMPLE

- Impose constraint $0.8 \le u(t) \le 1.2$ (amplitude)



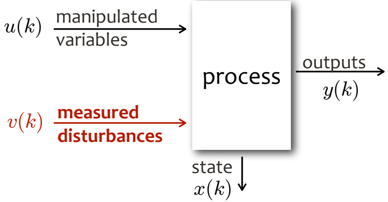
- Impose instead constraint $-0.2 \le \Delta u(t) \le 0.2$ (slew-rate)



- **Measured disturbance** $v(t)$ = input that is measured but not manipulated

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

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



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

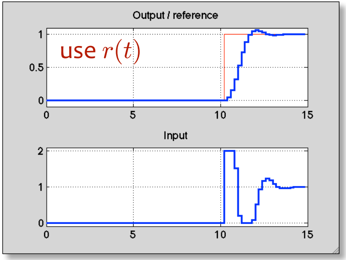
$$\begin{aligned} \min_z \quad & J(z, x(t)) = \tfrac{1}{2} z' H z + [x'(t)\ r'(t)\ u'(t-1)\ v'(t)] F' z \\ \text{s.t.} \quad & Gz \le W + S \begin{bmatrix} x(t) \\ r(t) \\ u(t-1) \\ v(t) \end{bmatrix} \end{aligned}$$

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

$$\min_{\Delta U} \sum_{k=0}^{N-1} \|W^y(y_{k+1} - r_{k+1})\|_2^2 + \|W^{\Delta u}\Delta u(k)\|_2^2$$

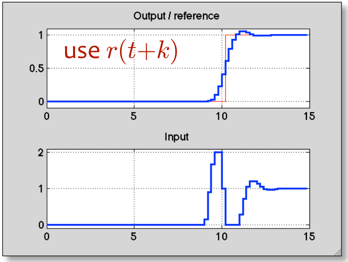
- Reference **not known** in advance (**causal**):

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



- Future refs (partially) **known** in advance (**anticipative action**):

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



go to demo `mpcpreview.m` (MPC Toolbox)

- Same idea also applies for **preview of measured disturbances**

SOFT CONSTRAINTS

- **Relax** output constraints to prevent QP infeasibility

$$
\begin{array}{ll}
\min_{z} & \sum_{k=0}^{N-1} \|W^y(y_{k+1}-r(t))\|_2^2 + \|W^{\Delta u}\Delta u_k\|_2^2 + \rho_\epsilon \epsilon^2 \\
\text{subj. to} & u_{\min} \le u_k \le u_{\max},\ k=0,\ldots,N-1 \\
& \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max},\ k=0,\ldots,N-1 \\
& y_{\min} - \epsilon V_{\min} \le y_k \le y_{\max} + \epsilon V_{\max},\ k=1,\ldots,N
\end{array}
\qquad
z = \begin{bmatrix} \Delta u(0) \\ \vdots \\ \Delta u(N-1) \\ \epsilon \end{bmatrix}
$$

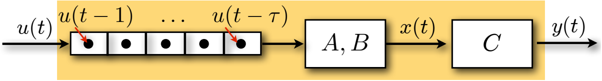
- ϵ = "panic" variable, with weight $\rho_\epsilon \gg W^y, W^{\Delta u}$
- $V_{\min}, V_{\max}$ = vectors with entries ≥ 0. The larger the i-th entry of vector V, the relatively softer the corresponding i-th constraint

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

DELAYS - METHOD #1

- Linear model with **delays**

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



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

$$x_k(t) \triangleq u(t-k) \quad \Rightarrow \quad x_k(t_1) = x_{k-1}(t),\ k = 1, \dots, \tau$$

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

- Apply MPC to the extended system

- **Note:** the prediction horizon $N \geq \tau$, otherwise $u_0, \dots, u_{N-1}$ have no effect on the output!

DELAYS - METHOD #2

- Linear model with **delays**: $\begin{cases} x(t+1) &=& Ax(t) + Bu(t-\tau) \\ y(t) &=& Cx(t) \end{cases}$
- **Delay-free** model: $\bar{x}(t) \triangleq x(t+\tau)$ ➡ $\begin{cases} \bar{x}(t+1) &=& A\bar{x}(t) + Bu(t) \\ \bar{y}(t) &=& C\bar{x}(t) \end{cases}$
- Design MPC for delay-free model, $u(t) = f_{\rm MPC}(\bar{x}(t))$
- Compute the predicted state

$$\bar{x}(t) = \hat{x}(t+\tau) = A^\tau x(t) + \sum_{j=1}^{\tau-1} A^j B \underbrace{u(t-1-j)}_{\text{past inputs!}}$$

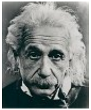
- Compute the MPC control move $u(t) = f_{\rm MPC}(\hat{x}(t+\tau))$
- For better closed-loop performance, $\hat{x}(t+\tau)$ can be computed by a more complex model than (A, B, C)

GOOD MODELS FOR (MPC) CONTROL

- Computation complexity and theoretical properties (e.g., stability) depend on chosen model/performance index/constraints
- Good models for MPC must be
  - **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 $\Rightarrow$ linear prediction model
  - **nonlinear** MPC $\Rightarrow$ nonlinear prediction model
  - **robust** MPC $\Rightarrow$ uncertain (linear) prediction model
  - **stochastic** MPC $\Rightarrow$ stochastic prediction model
  - **distributed/decentralized** MPC$\Rightarrow$ multiple MPCs cooperating together
  - **economic** MPC $\Rightarrow$ MPC based on arbitrary (economic) performance indices
  - **hybrid** MPC $\Rightarrow$ prediction model integrating logic and dynamics
  - **explicit** MPC $\Rightarrow$ off-line (exact/approximate) computation of MPC
  - **solvers** for MPC $\Rightarrow$ on-line numerical algorithms for solving MPC problems
- Main theoretical issues: **feasibility**, **stability**, solution **algorithms** (Mayne, 2014)

Research on <u>solution algorithms</u> with guaranteed theoretical properties has the most impact in industrial practice.

FEASIBILITY

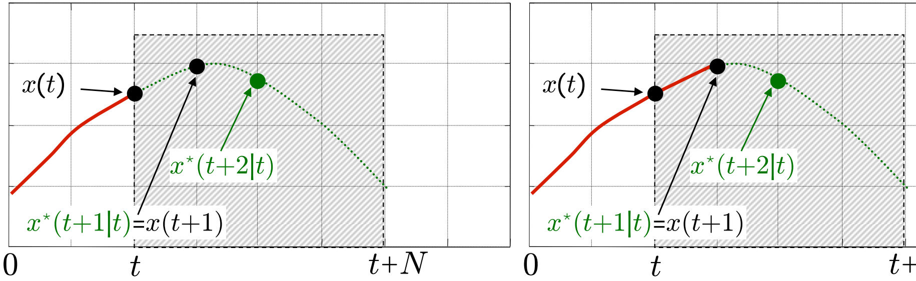
$$
\begin{array}{ll}
\min_z & \sum_{k=0}^{N-1} \|W^y(y_{k+1}-r(t))\|^2 + \|W^{\Delta u}\Delta u_k\|^2 \\
\text{subj. to} & u_{\min} \leq u_k \leq u_{\max},\ k=0,\dots,N-1 \\
& y_{\min} \leq y_k \leq y_{\max},\ k=1,\dots,N \\
& \Delta u_{\min} \leq \Delta u_k \leq \Delta u_{\max},\ k=0,\dots,N-1
\end{array}
$$

QP problem

- **Feasibility**: Will the QP problem be feasible at all sampling instants t?
- **Input constraints** only: always feasible if $u/\Delta u$ constraints are consistent
- **Hard output constraints**:
  - When $N < \infty$ there is no guarantee that the QP problem will remain feasible at all future time steps t
  - $N = \infty \Rightarrow$ infinite number of constraints!
  - **Maximum output admissible set** theory: $N < \infty$ is enough (Gutman, Ckwikel, 1987) (Gilbert, Tan, 1991) (Chmielewski, Manousiouthakis, 1996) (Kerrigan, Maciejowski, 2000)

PREDICTED AND ACTUAL TRAJECTORIES

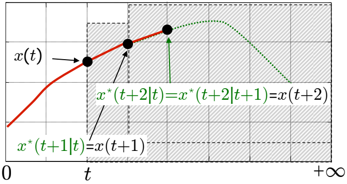
- Consider **predicted** and **actual** trajectories

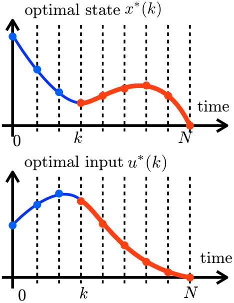


Even assuming perfect model and no disturbances, **predicted** open-loop trajectories and **actual** closed-loop trajectories can be different

PREDICTED AND ACTUAL TRAJECTORIES

- Special case: when the horizon is **infinite**, open-loop trajectories and closed-loop trajectories coincide.





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

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

` `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)

Richard Bellman
(1920–1984)

CONVERGENCE AND STABILITY

$$\begin{array}{ll}\min\limits_{z} & x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \\ \text{s.t.} & u_{\min} \leq u_k \leq u_{\max},\; k = 0, \ldots, N-1 \\ & y_{\min} \leq C x_k \leq y_{\max},\; k = 1, \ldots, N \end{array}$$

QP problem

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

- Stability is a complex function of the model (A, B, C) and 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 $x(t+1) = Ax(t) + Bu(t)$. Let the MPC law be based on

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

with $R, Q \succ 0$, $u_{\min} < 0 < u_{\max}$, $y_{\min} < 0 < y_{\max}$.

*If the **optimization problem is feasible at time** $t = 0$ then*

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

and the constraints are satisfied at all time $t \geq 0$, for all $R, Q \succ 0$.

For general stability results see (Lazar, Heemels, Weiland, Bemporad, 2006)

CONVERGENCE PROOF

Proof: Main idea: use the **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 $R, Q \succ 0$, $\lim_{t\to\infty} x(t) = 0, \ \lim_{t\to\infty} u(t) = 0$ ■

Reaching the global optimum is not needed to prove convergence!

A LITTLE MORE GENERAL CONVERGE RESULT

THEOREM

Consider again the linear system described by (A, B, C) *and the MPC control law based on optimizing*

$$\begin{array}{rl} V^*(x(t)) = \min & \displaystyle\sum_{k=0}^{N-1} y_k' Q_y y_k + u_k' R u_k \\ \text{s.t.} & x_{k+1} = A x_k + B u_k \\ & u_{\min} \le u_k \le u_{\max} \\ & y_{\min} \le y_k \le y_{\max} \end{array}$$

If the optimization problem is feasible at time $t = 0$ *then for either* $N = \infty$ *or* $x_N = 0$ *and for all* $R \succ 0$, $Q_y \succeq 0$

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

and the constraints are satisfied at all time $t \ge 0$*. Moreover, if* (LC, A) *is a detectable pair,* $Q_y = L'L$ *(Cholesky factorization), then* $\lim_{t\to\infty} x(t) = 0$*.*

CONVERGENCE PROOF

Proof:

- The shifted optimal sequence $\bar{z}_{t+1} = [u_1^t \ \dots \ u_{N-1}^t \ 0]'$ (or $\bar{z}_{t+1} = [u_1^t \ u_2^t \ \dots]'$ in case $N = \infty$) is feasible sequence at time $t+1$ and has cost

$$V^*(x(t)) - y'(t)Q_y y(t) - u'(t)Ru(t) \geq V^*(x(t+1))$$

- Therefore, by convergence of $V^*(x(t))$, we have that

$$0 \leq y'(t)Q_y y(t) + u'(t)Ru(t) \leq V^*(x(t)) - V^*(x(t+1)) \to 0$$

  for $t \to \infty$
- Since $R, Q_y \succ 0$, also $\lim_{t\to\infty} y(t) = 0, \ \lim_{t\to\infty} u(t) = 0$
- For all $k = 0, \dots, n-1$ we have that

$$0 = \lim_{t\to\infty} y'(t+k)Q_y y(t+k) = \lim_{t\to\infty} \|LC(A^k x(t) + \sum_{j=0}^{k-1} A^j Bu(t+k-1-j))\|_2^2$$

CONVERGENCE PROOF (CONT'D)

- As $u(t) \to 0$, also $LCA^k x(t) \to 0$, and hence $\Theta x(t) \to 0$, where Θ is the observability matrix of (LC, A)

- If (LC, A) is observable then Θ is nonsingular and hence $\lim_{t\to\infty} x(t) = 0$

- If (LC, A) is only detectable, a canonical observability decomposition shows that the unobservable modes go to zero spontaneously since $u(t) \to 0$. ■

STABILITY CONSTRAINTS

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

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

2. End-point constraint $x_N = 0$

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

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

   (Scokaert, Rawlings, 1996)

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

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

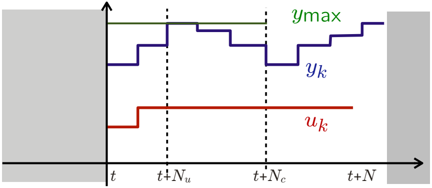
All the proofs in (1,2,3) use the value function $V^*(x(t)) = \min_z J(z, x(t))$ as a Lyapunov function.

CONTROL AND CONSTRAINT HORIZONS

$$
\begin{array}{ll}
\min\limits_{z} & \sum\limits_{k=0}^{N-1} \|W^y(y_k - r(t))\|_2^2 + \|W^{\Delta u}\Delta u_k\|_2^2 + \rho_\epsilon \epsilon^2 \\
\text{subj. to} & u_{\min} \leq u_k \leq u_{\max},\ k = 0, \dots, N-1 \\
& \Delta u_{\min} \leq \Delta u_k \leq \Delta u_{\max},\ k = 0, \dots, N-1 \\
& \Delta u_k = 0,\ k = N_u, \dots, N-1 \\
& y_{\min} - \epsilon V_{\min} \leq y_k \leq y_{\max} + \epsilon V_{\max},\ k = 1, \dots, N_c
\end{array}
$$

- The **input horizon** N_u limits the number of free variables
  - Reduced performance
  - Reduced computation time

  typically $N_u = 1 \div 5$



- The **constraint horizon** N_c limits the number of constraints
  - Higher chance of violating output constraints
  - Reduced computation time

MPC AND LINEAR QUADRATIC REGULATION (LQR)

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

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

- 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 N, the larger the region where MPC=LQR

DOUBLE INTEGRATOR EXAMPLE

- $y(t) = \iint u(\tau)d\tau$ $\Rightarrow$ $\left\{\begin{array}{rcl} x(t+1) & = & \left[\begin{smallmatrix}1&1\\0&1\end{smallmatrix}\right]x(t) + \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]u(t) \\ y(t) & = & \left[\begin{smallmatrix}1&0\end{smallmatrix}\right]x(t)\end{array}\right.$
- Constraints: $-1 \leq u(t) \leq 1$
- Objective:

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

$$u_k = Kx_k,\ \forall k \geq N_u,\ K = \text{LQR gain}$$

$$N_u = N = 2$$

$$\Rightarrow \left(\sum_{k=0}^{1} y_k^2 + \frac{1}{100}u_k^2\right) + x_2' \underbrace{\left[\begin{smallmatrix}2.1429 & 1.2246\\1.2246 & 1.3996\end{smallmatrix}\right]}_{\text{solution of algebraic Riccati equation}} x_2$$

- QP matrices (cost function normalized by max singular value of H)

$$H = \left[\begin{smallmatrix}0.8365 & 0.3603\\0.3603 & 0.2059\end{smallmatrix}\right],\ F = \left[\begin{smallmatrix}0.4624 & 1.2852\\0.1682 & 0.5285\end{smallmatrix}\right]$$

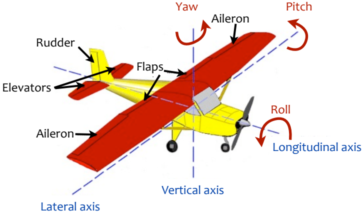
$$G = \begin{bmatrix}1 & 0\\-1 & 0\\0 & 1\\0 & -1\end{bmatrix},\ W = \begin{bmatrix}1\\1\\1\\1\end{bmatrix},\ S = \begin{bmatrix}0 & 0\\0 & 0\\0 & 0\\0 & 0\end{bmatrix}$$

EXAMPLE: AFTI-F16 AIRCRAFT

Linearized model:

$$\begin{cases} \dot{x} = \begin{bmatrix} -0.0151 & -60.5651 & 0 & -32.174 \\ -0.0001 & -1.3411 & 0.9929 & 0 \\ 0.00018 & 43.2541 & -0.86939 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x + \begin{bmatrix} -2.516 & -13.136 \\ -0.1689 & -0.2514 \\ -17.251 & -1.5766 \\ 0 & 0 \end{bmatrix} u \\ y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x \end{cases}$$

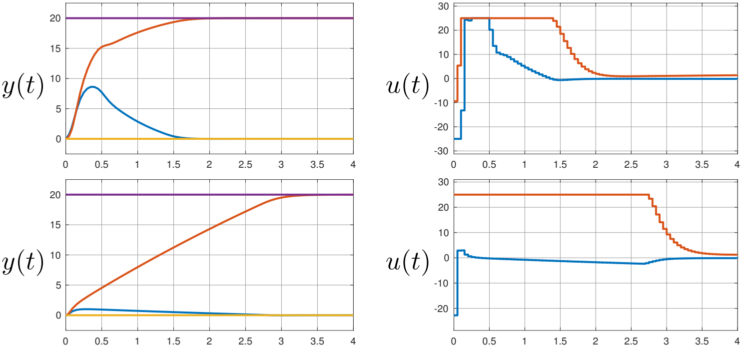
- Inputs: elevator and flaperon (=flap+aileron) angles
- Outputs: attack and pitch angles
- Sampling time: $T_s = 0.05$ sec (+ ZOH)
- Constraints: $\pm 25^\circ$ on both angles
- Open-loop unstable, poles are $-7.6636, -0.0075 \pm 0.0556j, 5.4530$



go to demo `linear/afti16.m` (Hybrid Toolbox)
see also `afti16.m` (MPC Toolbox)

EXAMPLE: AFTI-F16 AIRCRAFT

- Prediction horizon $N = 10$, control horizon $N_u = 2$
- Input weights $W^{\Delta u} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$, $W^u = 0$
- Input constraints $u_{\min} = -u_{\max} = \begin{bmatrix} 25^\circ \\ 25^\circ \end{bmatrix}$

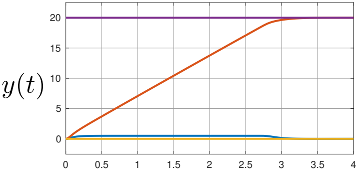


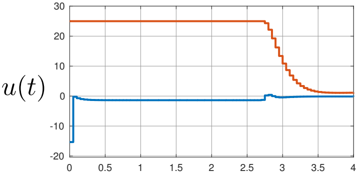
$W^y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$W^y = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}$

EXAMPLE: AFTI-F16 AIRCRAFT

- Add output constraints $y_{1,\min} = -y_{1,\max} = 0.5^\circ$

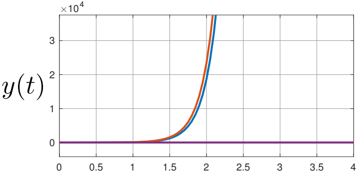


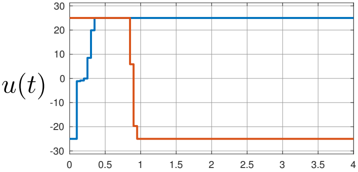


$W^y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

EXAMPLE: AFTI-F16 AIRCRAFT

- Linear control (=unconstrained MPC) + input clipping





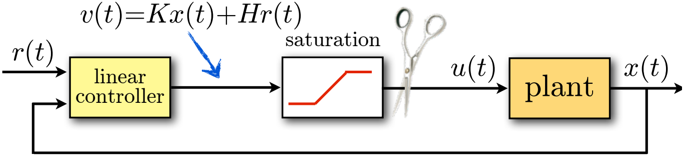
$$W^y = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$$

unstable!

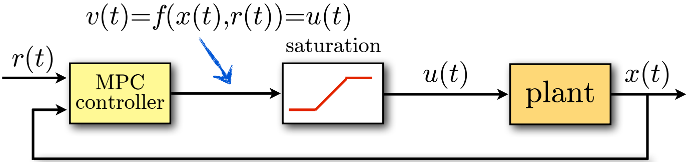
Saturation needs to be considered in the control design!

SATURATION

- Saturation is dangerous because it breaks the control loop



- MPC takes it into account automatically (and optimally)



TUNING GUIDELINES

$$
\begin{array}{ll}
\min\limits_{\Delta U} & \sum\limits_{k=0}^{N-1} \|W^y(y_{k+1}-r(t))\|_2^2 + \|W^{\Delta u}\Delta u_k\|_2^2 + \rho_\epsilon \epsilon^2 \\
\text{subj. to} & \Delta u_{\min} \leq \Delta u_k \leq \Delta u_{\max},\ k=0,\ldots,N_u-1 \\
& \Delta u_k = 0,\ k = N_u,\ldots,N-1 \\
& u_{\min} \leq u_k \leq u_{\max},\ k=0,\ldots,N_u-1 \\
& y_{\min} - \epsilon V_{\min} \leq y_k \leq y_{\max} + \epsilon V_{\max},\ k=1,\ldots,N_c
\end{array}
$$

- **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 $\rho_\epsilon \epsilon$ that keeps soft constraints reasonably satisfied

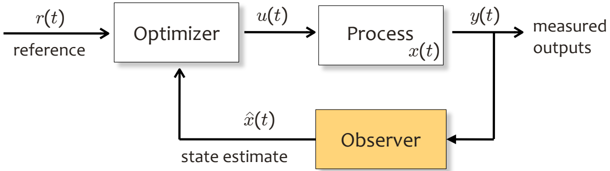
Always try to set N_u as small as possible!

SCALING

- Humans think infinite precision ... computers do not !
- Numerical difficulties may arise if variables assume very small/large values
- Example:

$$\begin{array}{lll} y_1 \in [-1e-4, 1e-4] & \text{[V]} \\ y_2 \in [-1e4, 1e4] & \text{[Pa]} \end{array} \quad \Longrightarrow \quad \begin{array}{ll} y_1 \in [-0.1, 0.1] & \text{[mV]} \\ y_2 \in [-10, 10] & \text{[kPa]} \end{array}$$

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

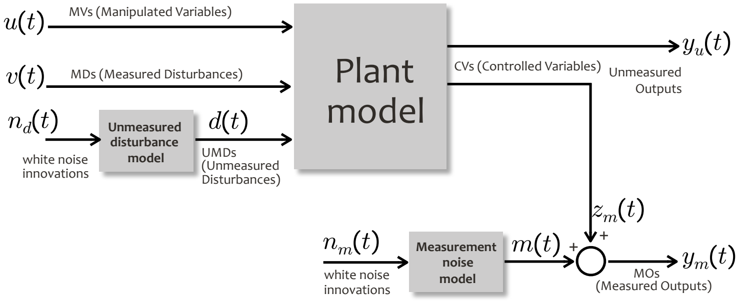


- Full state $x(t)$ of process may not be available, only outputs $y(t)$
- Even if $x(t)$ is available, noise should be filtered out
- Prediction and process models may be quite different
- The state $x(t)$ may not have any physical meaning
  (e.g., in case of model reduction or subspace identification)

We need to use a state observer

- Example: Luenberger observer $\hat{x}(t+1) = A\hat{x}(t) + Bu(t) + L(y(t) - C\hat{x}(t))$

EXTENDED MODEL FOR OBSERVER DESIGN



unmeasured disturbance model

$$\left\{\begin{array}{rcl} x_d(t+1) & = & \bar{A}x_d(t) + \bar{B}n_d(t) \\ d(t) & = & \bar{C}x_d(t) + \bar{D}n_d(t) \end{array}\right.$$

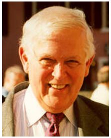
measurement noise model

$$\left\{\begin{array}{rcl} x_m(t+1) & = & \tilde{A}x_m(t) + \tilde{B}n_m(t) \\ m(t) & = & \tilde{C}x_m(t) + \tilde{D}n_m(t) \end{array}\right.$$

- Note: the measurement noise model is not needed during optimization

- Plant model

$$\left\{\begin{array}{rcl} x(t+1) &=& Ax(t)+B_u u(t)+B_v v(t)+B_d d(t)\\ y(t) &=& Cx(t)+D_v v(t)+D_d d(t)\end{array}\right.$$



Rudolf Emil Kalman
(1930–2016)

- Full model for designing Kalman filter

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

- $n_d(k)$ = source of **modeling errors**
- $n_m(k)$ = source of **measurement noise**
- $n_u(k)$ = white noise on input u (added to compute the Kalman gain)

OBSERVER IMPLEMENTATION

- Measurement update

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

- Time update

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

- Note that the **separation principle** holds (under certain assumptions)

  (Muske, Meadows, Rawlings, 1994)

I/O FEEDTHROUGH

- We always assumed **no feedthrough from u to y**

$$y_k = Cx_k + Du_k + D_v v_k + D_d d_k, \quad D = 0$$

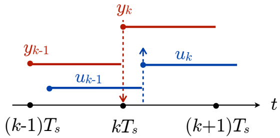
- This avoids **static loops** between state observer ($x(t|t)$ depends on $u(t)$) and MPC ($u(t)$ depends on $x(t|t)$)

- Often $D = 0$ is not a limiting assumption as
  - often **actuator dynamics** must be considered (u is the set-point to a low-level controller of the actuators)
  - most physical models described by ordinary differential equations are **strictly causal**, and so is the discrete-time version of the model

- In case $D \neq 0$, we can assume a **delay** in executing the commanded u

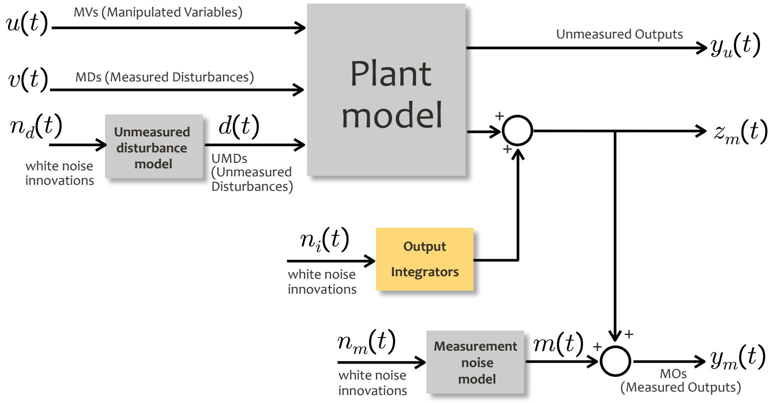
$$y_k = Cx_k + Du_{k-1}$$

and treat $u(t-1)$ as an extra state.



INTEGRAL ACTION IN MPC

OUTPUT INTEGRATORS



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

OUTPUT INTEGRATORS AND OFFSET-FREE TRACKING

- Add constant unknown disturbances on measured outputs:

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

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

- Why we get offset-free tracking in steady-state (intuitively):
  - the observer makes $C\hat{x}(t) + \hat{d}(t) \to y(t)$ **(estimation error)**
  - the MPC controller makes $C\hat{x}(t) + \hat{d}(t) \to r(t)$ **(predicted tracking error)**
  - the combination of the two makes $y(t) \to r(t)$ **(actual tracking error)**

- In steady state, the term $\hat{d}(t)$ compensates for model mismatch

OUTPUT INTEGRATORS AND OFFSET-FREE TRACKING

(Pannocchia, Rawlings, 2003)

- **More general result:** consider the disturbance model

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

special case: **output integrators**

$B_d = 0$, $D_d = I$

THEOREM

Let the number n_d of disturbances be equal to the number n_y of measured outputs. Then all pairs (B_d, D_d) satisfying

$$\operatorname{rank} \begin{bmatrix} I - A & -B_d \\ C & D_d \end{bmatrix} = n_x + n_y$$

guarantee zero offset in steady-state $(y = r)$, provided that constraints are not active in steady-state and the closed-loop system is asymptotically stable.

DISTURBANCE MODEL EXAMPLE

- Open-loop system: $A = \begin{bmatrix} 1 & -1 & -2 & 1 \\ 0 & -2 & 3 & 2 \\ 0 & 2 & -1 & -1 \\ 0 & 2 & 0 & -2 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$, $C_m = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}$

- We cannot add an **output integrator** since the pair $\left(\begin{bmatrix} A & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} C & 1 \end{bmatrix}\right)$ is not observable

- Add an **input integrator**: $B_d = B$, $D_d = 0$, $\text{rank} \begin{bmatrix} A & B_d \\ C_m & D_d \end{bmatrix} = 5 = n_x + n_y$

$$\begin{cases} x(t+1) &= Ax(t) + B(u(t) + d(t)) \\ d(t+1) &= d(t) \\ y(t) &= Cx(t) + 0 \cdot d(t) \end{cases}$$

ERROR FEEDBACK

(Kwakernaak, Sivan, 1972)

- **Idea:** add the integral of the tracking error as an additional state (original idea developed for integral action in state-feedback control)

- Extended prediction model:

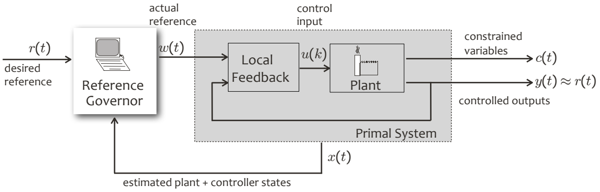
$$\left\{\begin{array}{rcl} x(t+1) & = & Ax(t)+B_u u(t)+0\cdot r(t) \quad \leftarrow \text{r(t) is seen as a meas. disturbance} \\ q(t+1) & = & q(t)+\underbrace{Cx(t)-r(t)}_{\text{tracking error}} \quad \leftarrow \text{integral action} \\ y(t) & = & Cx(t) \end{array}\right.$$

- $\|W^i q\|_2^2$ is penalized in the cost function, otherwise it is useless. W^i is a new tuning knob

- Intuitively, if the MPC closed-loop is asymptotically stable then $q(t)$ converges to a constant, and hence $y(t)-r(t)$ converges to zero.

REFERENCE / COMMAND GOVERNOR

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



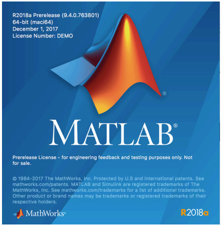
- MPC manipulates set-points to a linear feedback loop to enforce constraints
- Separation of problems:
  - Local feedback guarantees offset-free tracking in steady-state **in the absence of constraints**
  - Actual reference $w(t)$ generated by MPC to **take care of constraints**
- Advantages: small-signal properties preserved, fewer variables to optimize

$$\begin{array}{rl} w(t) = \arg\min_w & \|w - r(t)\|_2^2 \\ \text{s.t.} & c_{t+k} \in \mathcal{C} \end{array}$$

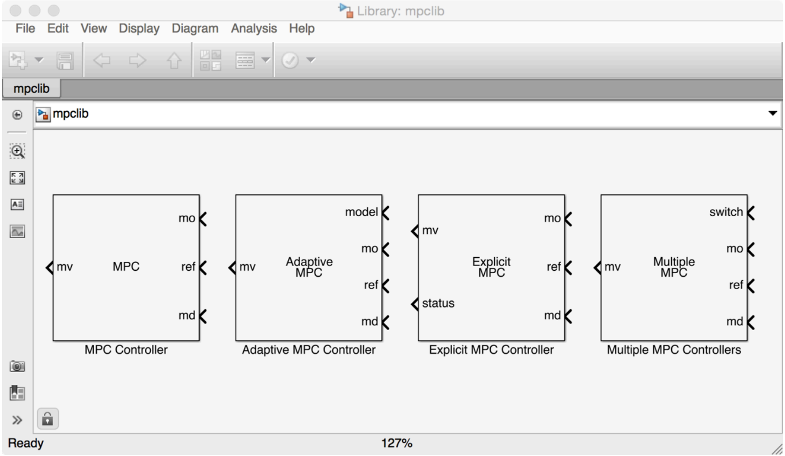
MODEL PREDICTIVE CONTROL TOOLBOX

(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** MPC functions (compiled EML-code)
- **Graphical User Interface**
- **Simulink library** (compiled EML-code)

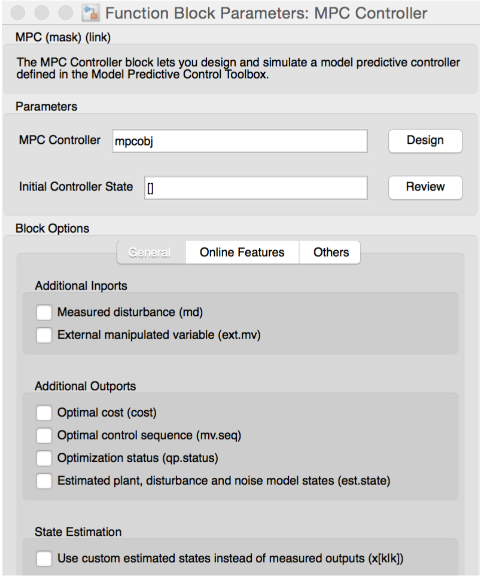


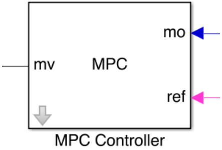
MPC SIMULINK LIBRARY



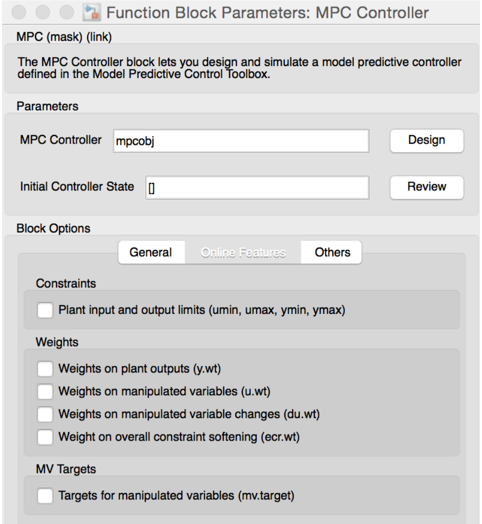

```
>> mpclib
```

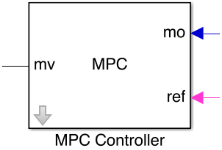
MPC SIMULINK BLOCK



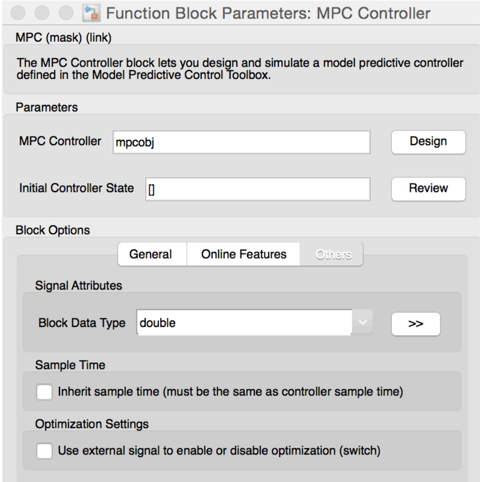


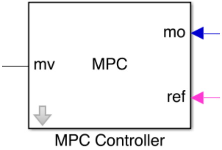
MPC SIMULINK BLOCK



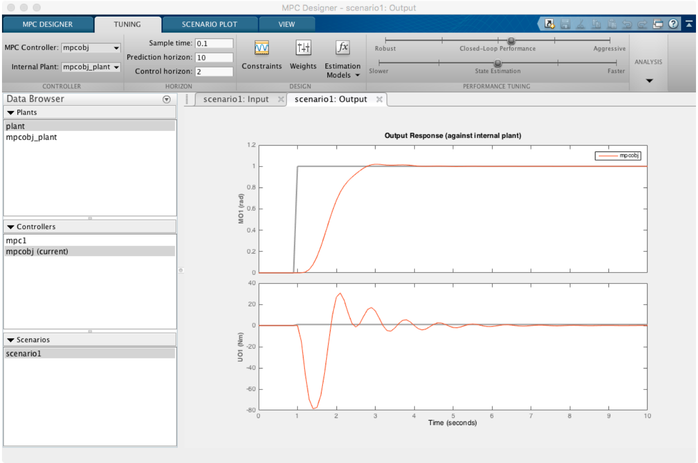


MPC SIMULINK BLOCK





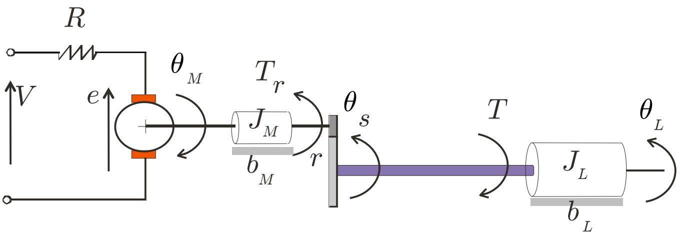
MPC GRAPHICAL USER INTERFACE



>> `mpcDesigner` (old version: >> `mpctool`)

See video on Mathworks' web site (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
J_L	$50 J_M$	nominal load inertia
β_L	25	load viscous friction coefficient
T_s	0.1	sampling time

```
>> mpcmotor
```

see also
`linear/dcmotor.m`
(Hybrid Toolbox)

DC SERVOMOTOR MODEL

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

$$x = \begin{bmatrix} \theta_L \\ \dot{\theta}_L \\ \theta_M \\ \dot{\theta}_M \end{bmatrix}$$

$$y = \begin{bmatrix} \theta_L \\ T \end{bmatrix}$$

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

DC SERVOMOTOR CONSTRAINTS

- The input DC voltage V is bounded withing the range

$$|V| \leq 220\, V$$

- Finite shear strength $\tau_{adm} = 50N/mm^2$ requires that the torsional torque T satisfies the constraint

$$|T| \leq 78.5398\, Nm$$

- Sampling time of model/controller: $T_s = 0.1s$

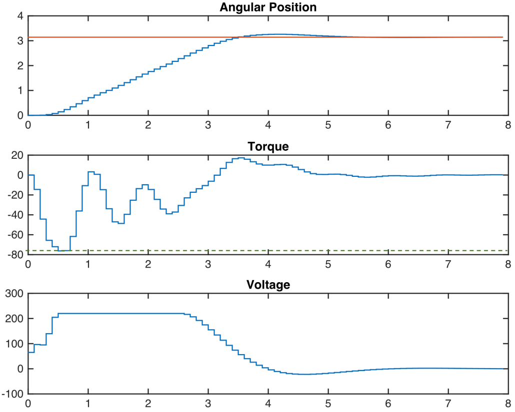
```
>> MV = struct('Min',-220,'Max',220);
>> OV = struct('Min',{-Inf,-78.5398},'Max',{Inf,78.5398});
>> Ts = 0.1;
```

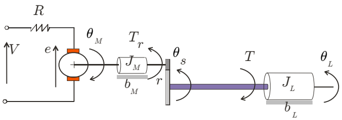
DC SERVOMOTOR - MPC SETUP

$$
\begin{array}{ll}
\min\limits_{\Delta U} & \displaystyle\sum_{k=0}^{p-1} \|W^y(y_{k+1} - r(t))\|^2 + \|W^{\Delta u}\Delta u_k\|^2 + \rho_\epsilon \epsilon^2 \\[2ex]
\text{subj. to} & \Delta u_{\min} \leq \Delta u_k \leq \Delta u_{\max},\ k = 0, \ldots, m-1 \\
& \Delta u_k = 0,\ k = m, \ldots, p-1 \\
& u_{\min} \leq u_k \leq u_{\max},\ k = 0, \ldots, m-1 \\
& y_{\min} - \epsilon V_{\min} \leq y_k \leq y_{\max} + \epsilon V_{\max},\ k = 1, \ldots, p
\end{array}
$$

```
>> 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

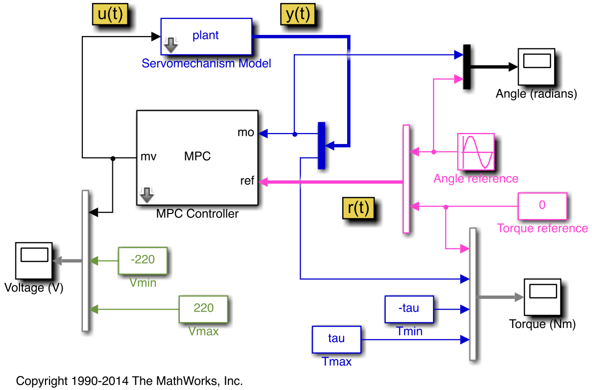




Closed-loop simulation using the `sim` command

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

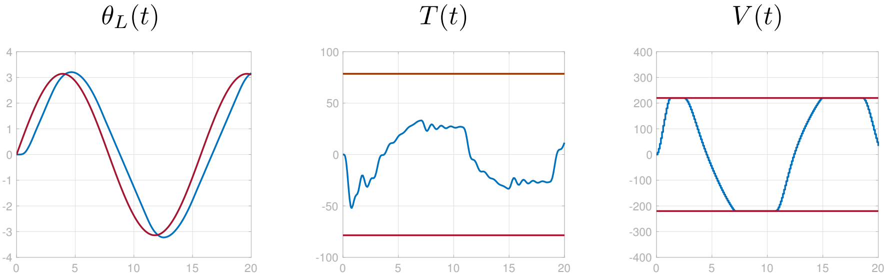
DC SERVOMOTOR - CLOSED-LOOP SIMULATION



Closed-loop simulation in Simulink

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

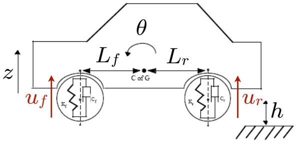
DC SERVOMOTOR - CLOSED-LOOP SIMULATION



- same MPC tuning parameters
- setpoint $r(t) = \pi \sin(0.4t)$ deg

EXAMPLE: MPC OF ACTIVE SUSPENSIONS

We consider the suspension demo `sldemo_suspn` in MATLAB



$$
\begin{aligned}
F_{front} &= 2K_f(L_f\theta - (z+h)) + 2C_f(L_f\dot{\theta} - \dot{z}) + u_f \\
F_{front}, F_{rear} &= \text{upward force on body from front/rear suspension} \\
K_f, K_r &= \text{front and rear suspension spring constant} \\
C_f, C_r &= \text{front and rear suspension damping rate} \\
L_f, L_r &= \text{horizontal distance from c.g. to front/rear suspension} \\
\theta, \dot{\theta} &= \text{pitch (rotational) angle and its rate of change} \\
z, \dot{z} &= \text{bounce (vertical) distance and its rate of change}
\end{aligned}
$$

- for control purposes we add external forces u_f, u_r as manipulated variables
- the system has 4 states: θ, $\dot{\theta}$, z, $\dot{z}$

EXAMPLE: MPC OF ACTIVE SUSPENSIONS

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

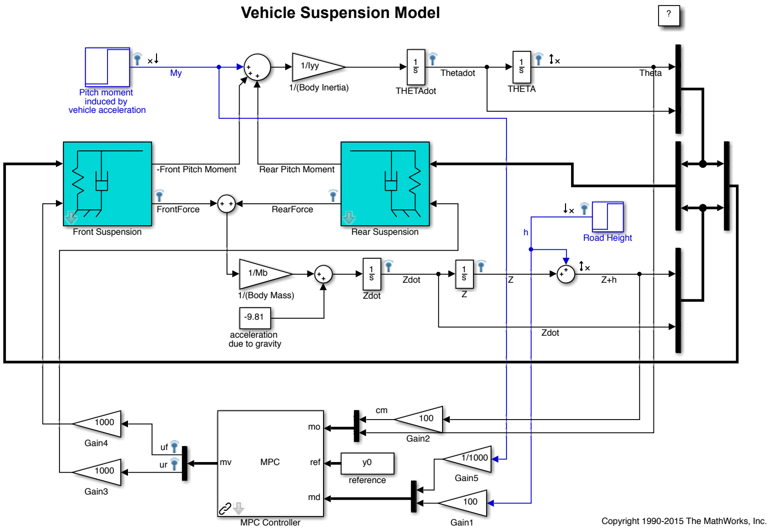
```
>> plant_mdl = 'suspension_model';
>> op = operspec(plant_mdl);
>> [op_point, op_report] = findop(plant_mdl,op);
>> sys = linearize(plant_mdl, op_point);
>> Ts = 0.025; % sample time (s)
>> plant = c2d(sys,Ts);

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

EXAMPLE: MPC OF ACTIVE SUSPENSIONS

- Step #2: design the MPC controller

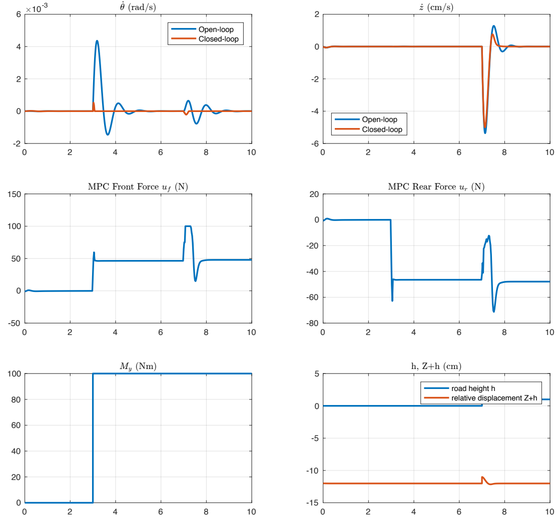
```
>> dfmax = .1/.1; % [kN/s]
>> MV = struct('RateMin',-Ts*dfmax, Ts*dfmax,...
'RateMax',Ts*dfmax,Ts*dfmax);
>> OV = [];
>> Weights = struct('MV',[0 0], 'MVRate',[.01 .01],...
'OV',[.01 0 10]);

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

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

EXAMPLE: MPC OF ACTIVE SUSPENSIONS



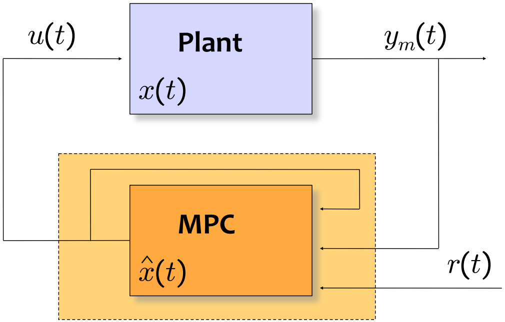
EXAMPLE: MPC OF ACTIVE SUSPENSIONS

- Closed-loop MPC results



FREQUENCY ANALYSIS OF MPC (FOR SMALL SIGNALS)

- Unconstrained MPC gain + linear observer = linear dynamical system
- Closed-loop MPC analysis can be performed using standard frequency-domain tools (e.g., Bode plots for sensitivity analysis)




```
>> sys=ss(mpcobj)
>> sys=tf(mpcobj)
```

returns the LTI object of the MPC controller (when constraints are inactive)

CONTROLLER MATCHING

(Di Cairano, Bemporad, 2010)

- Given a desired linear controller $u = K_d x$, find a set of weights Q, R, P defining an MPC problem such that

$$-\begin{bmatrix} I \ 0 \ \dots \ 0 \end{bmatrix} H^{-1} F = K_d$$

i.e., the **MPC controller coincides with K_d** when the constraints are **inactive**

- Recall that the QP matrices are $H = 2(\bar{R} + \bar{S}'\bar{Q}\bar{S})$, $F = 2\bar{S}'\bar{Q}\bar{T}$, where

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

- The above inverse optimality problem can be cast as a convex problem

(Di Cairano, Bemporad, 2010)

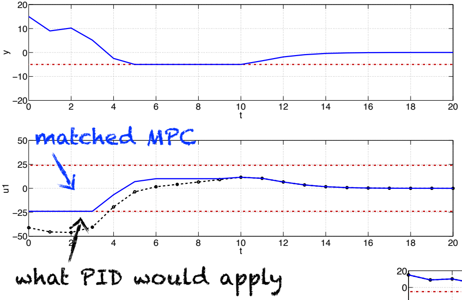
CONTROLLER MATCHING - EXAMPLE

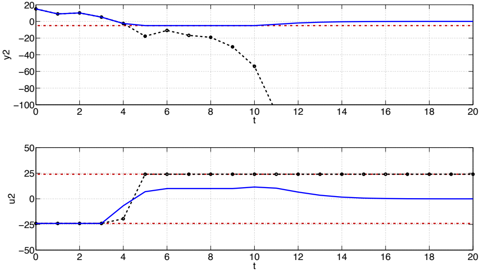
- Open-loop process: $y(t) = 1.8y(t-1) + 1.2y(t-2) + u(t-1)$
- Constraints: $-24 \le u(t) \le 24$, $y(t) \ge -5$
- Desired controller = PID with gains $K_I = 0.248$, $K_P = 0.752$, $K_D = 2.237$

$$\begin{array}{rcl} u(t) & = & -\left(K_I\mathcal{I}(t) + K_P y(t) + \frac{K_D}{T_s}(y(t) - y(t-1))\right) \\ \mathcal{I}(t) & = & \mathcal{I}(t-1) + T_s y(t) \end{array} \qquad x(t) = \begin{bmatrix} y(t-1) \\ y(t-2) \\ \mathcal{I}(t-1) \\ u(t-1) \end{bmatrix}$$

- Matching result (using inverse LQR):

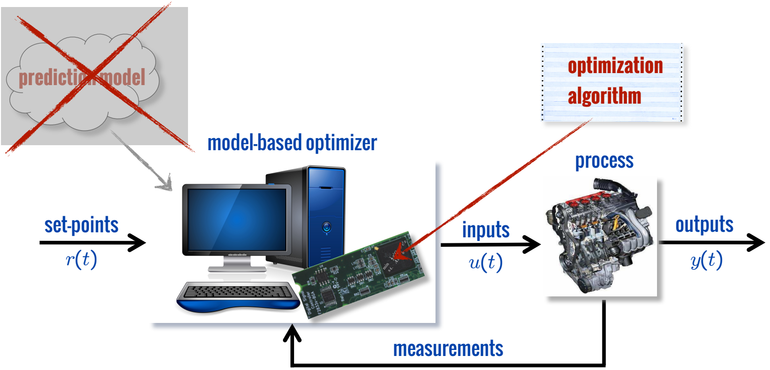
$$Q^* = \begin{bmatrix} 6.401 & 0.064 & -0.001 & 0.020 \\ 0.064 & 6.605 & 0.006 & 0.080 \\ -0.001 & 0.006 & 6.647 & -0.020 \\ 0.019 & 0.080 & -0.020 & 6.378 \end{bmatrix},\ R^* = 1,\ P^* = \begin{bmatrix} 422.7 & 241.7 & 50.39 & 201.4 \\ 241.7 & 151.0 & 32.13 & 120.4 \\ 50.39 & 32.13 & 19.85 & 26.75 \\ 201.4 & 120.4 & 26.75 & 106.6 \end{bmatrix}$$





- Note: This is not trivially a saturation of a PID controller. In this case saturating the PID output leads to closed-loop instability!

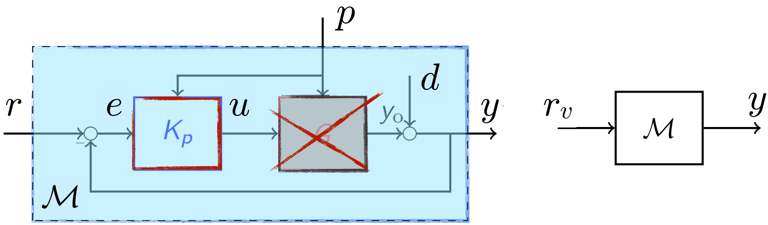
DATA-DRIVEN MPC



- Can we design an MPC controller **without a model** of the **open-loop process** ?

DATA-DRIVEN DIRECT CONTROLLER SYNTHESIS

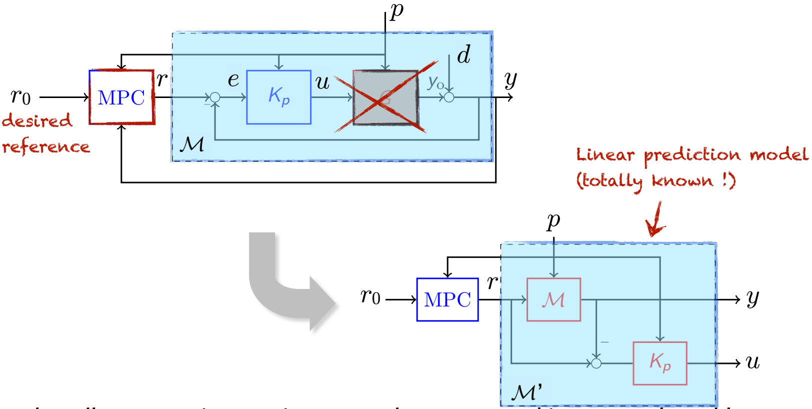
(Campi, Lecchini, Savaresi, 2002) (Formentin et al., 2015)



- Collect a set of **data** $\{u(t), y(t), p(t)\}$, $t = 1, \ldots, N$
- Specify a **desired closed-loop linear model** $\mathcal{M}$ from r to y
- Compute $r_v(t) = \mathcal{M}^{\#} y(t)$ 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

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

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



- MPC can handle constraints on inputs and outputs and improve closed-loop performance (Piga, Formentin, Bemporad, 2016)

DATA-DRIVEN MPC - AN EXAMPLE

- DC motor equations

$$\begin{bmatrix} \dot{\theta}(\tau) \\ \dot{\omega}(\tau) \\ \dot{I}(\tau) \end{bmatrix} = \begin{bmatrix} 0 & 1+\frac{\sin(\theta(\tau))}{\theta(\tau)} & 0 \\ \frac{mgl}{J}\frac{\sin(\theta(\tau))}{\theta(\tau)} & -\frac{b}{J} & \frac{K}{J} \\ 0 & -\frac{K}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} \theta(\tau) \\ \omega(\tau) \\ I(\tau) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} V(\tau)$$

$$y(\tau) = [1\ 0\ 0] \begin{bmatrix} \theta(\tau) \\ \omega(\tau) \\ I(\tau) \end{bmatrix}$$

← this model is not used!

- Desired closed-loop behavior $\mathcal{M}$ = first-order low-pass filter:

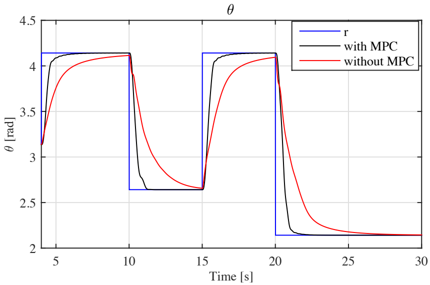
$$\begin{aligned} x_M(t+1) &= 0.99 x_M(t) + 0.01 r(t) \\ \theta(t) &= x_M(t) \end{aligned}$$

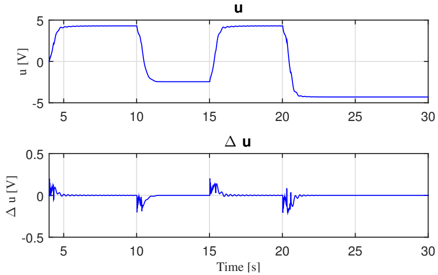
- Controller K_p synthesized from data with the following structure

$$\begin{aligned} e_I(t+1) &= e_I(t) + r_v(t) - y(t) \\ u(t) &= \textstyle\sum_{i=1}^{4} a_i(\theta(t))u(t-i) + \sum_{j=0}^{3} b_j(\theta(t))e_I(t-j) \end{aligned}$$

DATA-DRIVEN MPC - AN EXAMPLE

- MPC designed with soft constraints on u, Δu and y
- Experimental results:





desired tracking performance achieved

constraints on input increments satisfied

No open-loop model was 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: $\begin{cases} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{cases}$ $\quad x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^p$
- 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 (∞-norms): $\|v\|_\infty \triangleq \max_{i=1,\ldots,n} |v_i|$

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

R, Q, P full rank $\quad z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$

LINEAR MPC BASED ON LP

- Basic trick: introduce slack variables (Q^i = i-th row of Q)

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

- Example:

$$
\begin{array}{rl}
\min_x |x| \to \ \min_{x,\epsilon} & \epsilon \\
\text{s.t.} & \epsilon \geq x \\
& \epsilon \geq -x
\end{array}
$$

LINEAR MPC BASED ON LP

- Linear prediction model: $x_k = A^k x_0 + \sum_{i=0}^{k-1} A^i B u_{k-1-i}$
- Optimization problem:

$$\begin{aligned} V(x_0) = \quad & \min_z \quad [1\ \dots\ 1\ 0\ \dots\ 0]z \\ & \text{s.t.} \quad Gz \leq W + Sx_0 \end{aligned}$$

(linear objective)

(linear constraints)

Linear Program (LP)

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

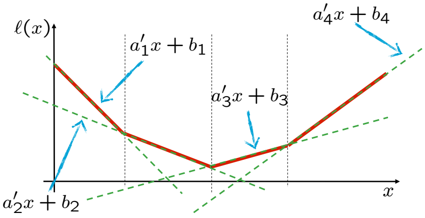
EXTENSION TO ARBITRARY CONVEX PWA FUNCTIONS

RESULT

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

(Schechter, 1987)

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



$$\ell(x)=\max\left\{a_1'x+b_1,\dots,a_4'x+b_4\right\}$$

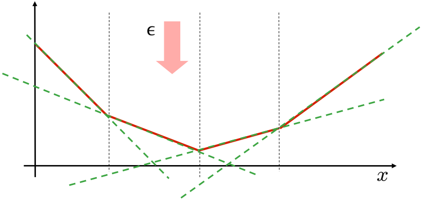
- Constrained optimal control problem

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

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

CONVEX PWA OPTIMIZATION PROBLEMS AND LP

- Minimization of a **convex PWA function** $\ell(x)$:



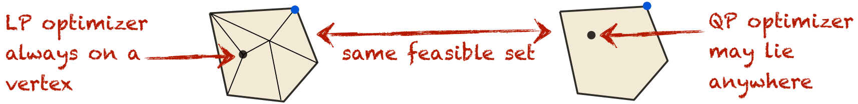
$$\begin{aligned} \min_{\epsilon,x} \quad & \epsilon \\ \text{s.t.} \quad & \begin{cases} \epsilon \geq a_1' x + b_1 \\ \epsilon \geq a_2' x + b_2 \\ \epsilon \geq a_3' x + b_3 \\ \epsilon \geq a_4' x + b_4 \end{cases} \end{aligned}$$

- By construction $\epsilon \geq \max\{a_1'x + b_1, a_2'x + b_2, a_3'x + b_3, a_4'x + b_4\}$
- By contradiction it is easy to show that at the optimum we have that

$$\epsilon = \max\{a_1'x + b_1, a_2'x + b_2, a_3'x + b_3, a_4'x + b_4\}$$

- **Convex PWA constraints** $\ell(x) \leq 0$ can be handled similarly by imposing $a_i'x + b_i \leq 0$, $\forall i = 1, 2, 3, 4$

- QP- and LP-based share the same set of feasible inputs ($Gz \leq W + Sx$)
- 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, because in LP an optimal point is always on a vertex



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

$$\{z : \ Gz \leq W + Sx\}$$

ϵ = additional slack variables introduced to represent convex PWA costs