Model Predictive Control

introductory seminar

Gionata Cimini

Universitá Politecnica delle Marche

Dipartimento di Ingegneria dell'Informazione

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

INSTITUTE
FOR ADVANCED
STUDIES
LUCCA

Outline

What is Model Predictive Control?

Model **P**redictive **C**ontrol (**MPC**) is an **advanced control strategy**, trying to replace the standard PID where its performance are not satisfactory

It is the **highest impact** advanced control methodology in industrial control engineering

It cannot be considered a specific control strategy

It covers several control strategies which **make an explicit use of a model** of the process to predict its future behavior, **minimizing an objective function** to obtain the control inputs

Some references:

- *Predictive Control with Constraints*, J.M. Maciejowski, Pearson Education, 2002.
- *Model Predictive Control*, E.F. Camacho and C. Bordons, Springer, 2nd Ed. 2007.

In which field was born MPC?

MPC was born in the late 70's! $\rightarrow$ it is a relatively old control

Several researchers published almost simultaneously the first theory about MPC

One of the first publication is *J. Richalet, A. Rault, J.L. Testud, and J. Papon. Model predictive heuristic control: applications to industrial processes, Automatica, 14: 413-428,* **1978**

But...

It was born in industry! Industrial practitioners pioneered a first version of MPC, which was implemented before the theoretical aspects were formalized by mathematicians or control people.
This is not so common ...

Process industry:

- refineries
- chemical plants
- petrochemical

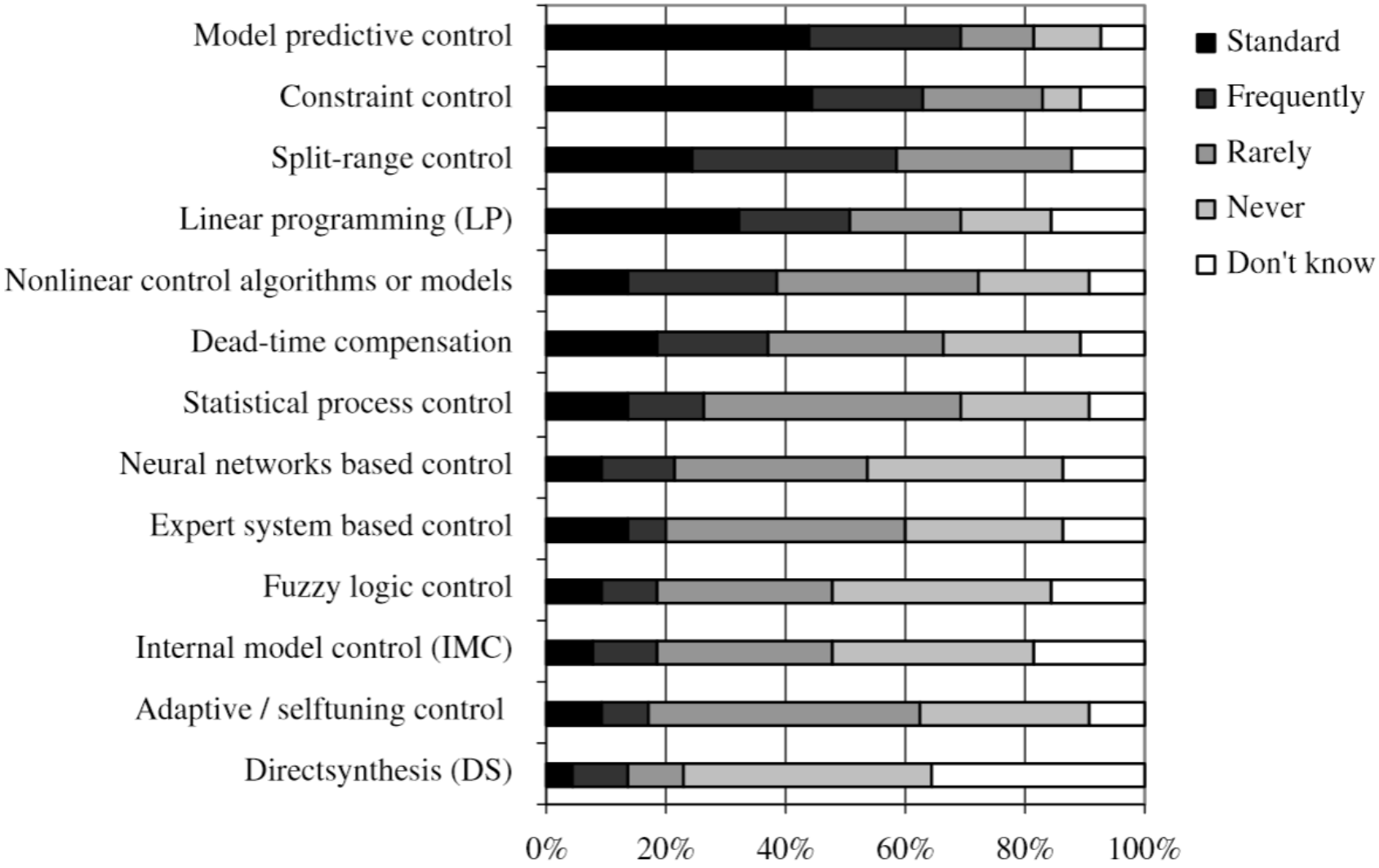
Why its success?

In process industry, MPC is a technology!

Some of the most appreciated features

- it is an optimal control
- mutivariables systems are treated in a straightforward way
- system constraints are easily taken into account
- can be successfully implemented for simple or very complex processes
  - long delays
  - nonminimum phase
  - unstable plants
- can exploit the knowledge about future references
- it accounts for feedforward with measured disturbances
- can be handled by people with limited control knowledge
- it does not have a fixed structure, modifications and new features are easily included

MPC use in industry



Margret Bauer and Ian K. Craig. "Economic assessment of advanced process control survey and framework". In: *Journal of Process Control* 18.1 (2008), pp. 2 –18. ISSN: 0959-1524

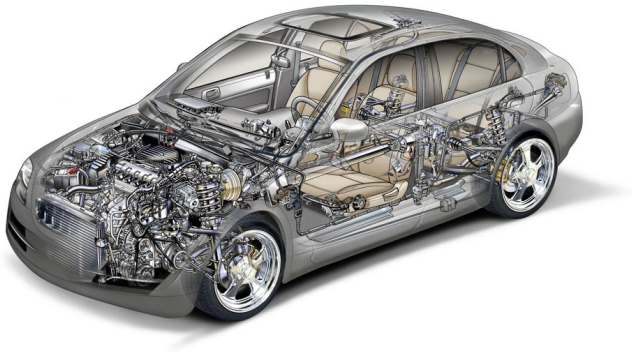
MPC developers

The most important companies that are currently developing MPC softwares

- AspenTech
- Adersa
- Honeywell Profimatics
- Setpoint Inc
- Treiber Cotrols
- ABB
- Pavillon Technologies
- Simulation Sciences

Emerging MPC applications

AUTOMOTIVE



AEROSPACE

RENEWABLES

ROBOTICS

Why it is not widely applied?

Some disadvantages of MPC that have limited the spread both in industry and in the academy:

- the quality of the model heavily affects the control performance
- stability is not often easily proven (finite horizon)
- **the online computational effort is high**
  - Process industry is weakly affected. Sampling time is in the order of seconds, or higher. Usually a PC is employed (GHz)
  - In faster processes, this is the **main issue**! The sampling interval can be in the order of ms and μs. Embedded boards are not that powerful (Mhz)

Good News!

- embedded processors are more and more powerful
- advances in optimization algorithms will make the problem easier to solve

Defining MPC

There are lot of names denoting predictive control:

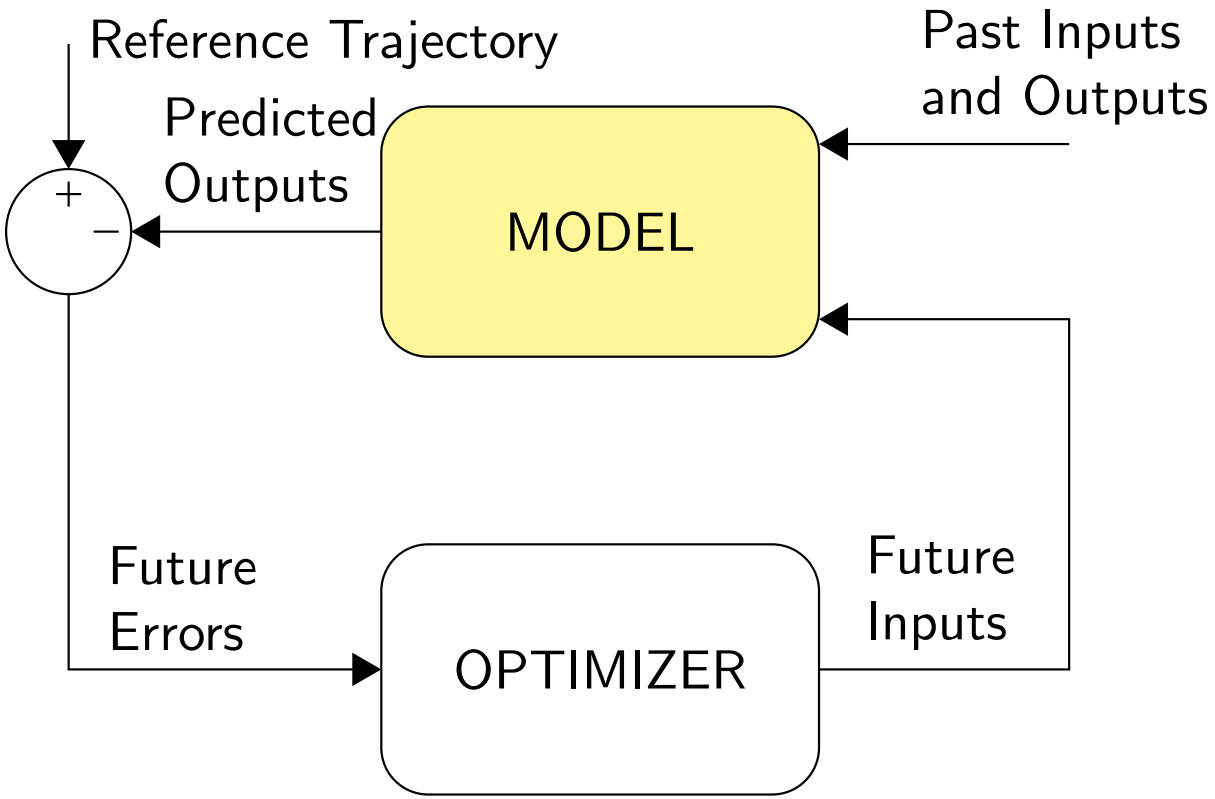
- Dynamic Matrix Control (DMC)
- Extended Prediction Self-Adaptive (EPSAC)
- Generalized Predictive Control (GPC)
- Model Algorithmic Control (MAC)
- Predictive Functional Control (PFC)
- Quadratic Dynamic Matrix Control (QDMC)
- Sequential Open Loop Optimization (SOLO)

All these controllers are simply variants of *Model Predictive Control* (MPC)

→ with MPC we denotes the family of controllers that

- predicts the future dynamics of the process with an explicit model
- involves optimization technique
- follows the receding horizon idea

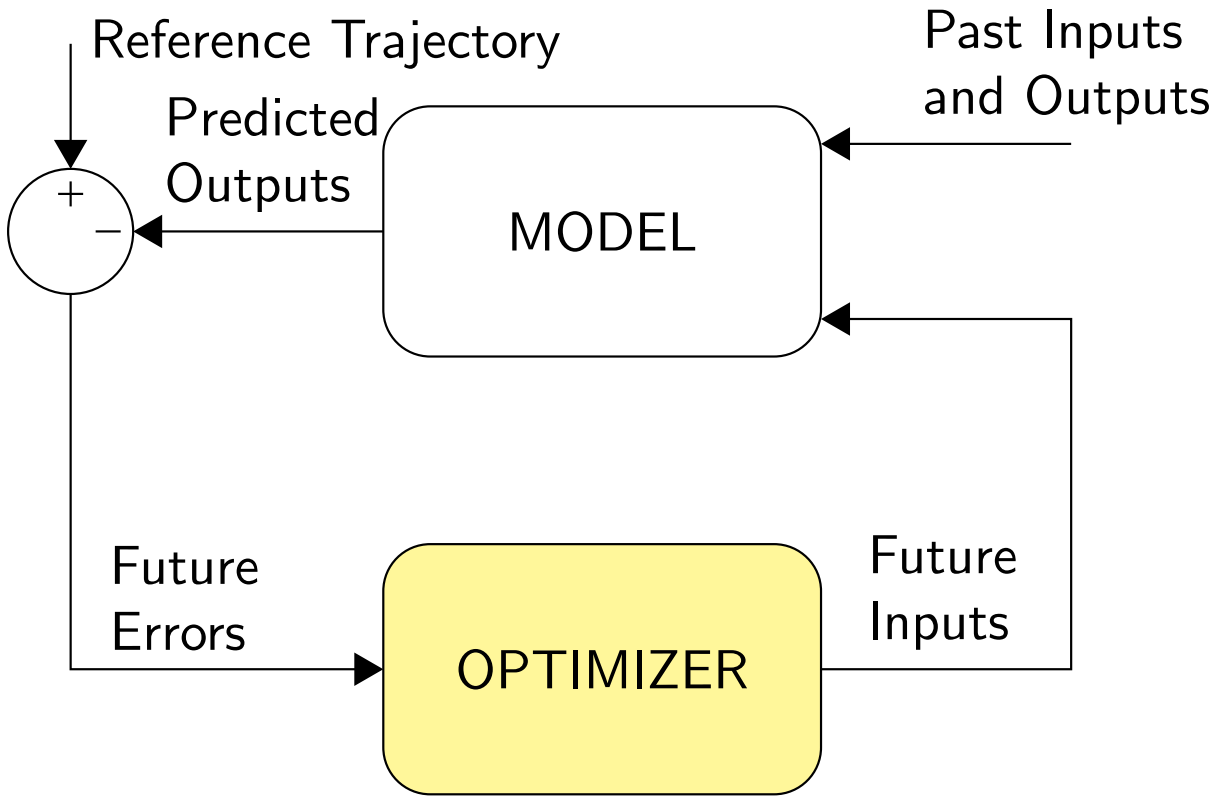
Model Based Optimization /1



Prediction Model (examples)

- Mathematical Model: a linear (ARX, ARMAX, state-space) or a non-linear representation of the model
- PieceWise Affine (PWA) function: the partition of state/input set into a finite number of polyhedral regions
- Artificial Intelligence: modeling trough Neural Networks or Fuzzy Logic

Model Based Optimization /2

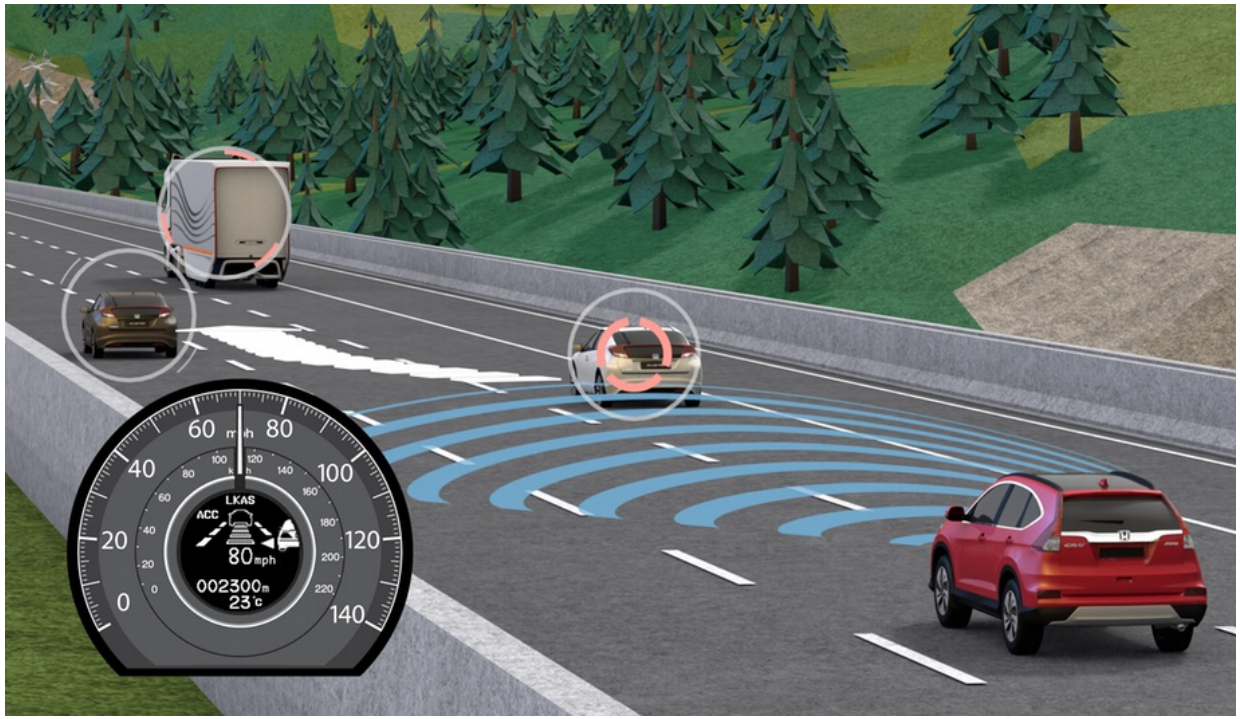


Optimizer (examples)

- Online/Offline: the optimization problem can be per-calculated, or solved at each iteration;
- Constrained: constraints on input and outputs can be imposed;
- Linear/ Non linear;
- Mixed Integer: when there are inputs/outputs that are integer (e.g. switches);
- Stochastic: when the model involves a stochastic process (wind, solar power);

Outline

Prediction



Idea

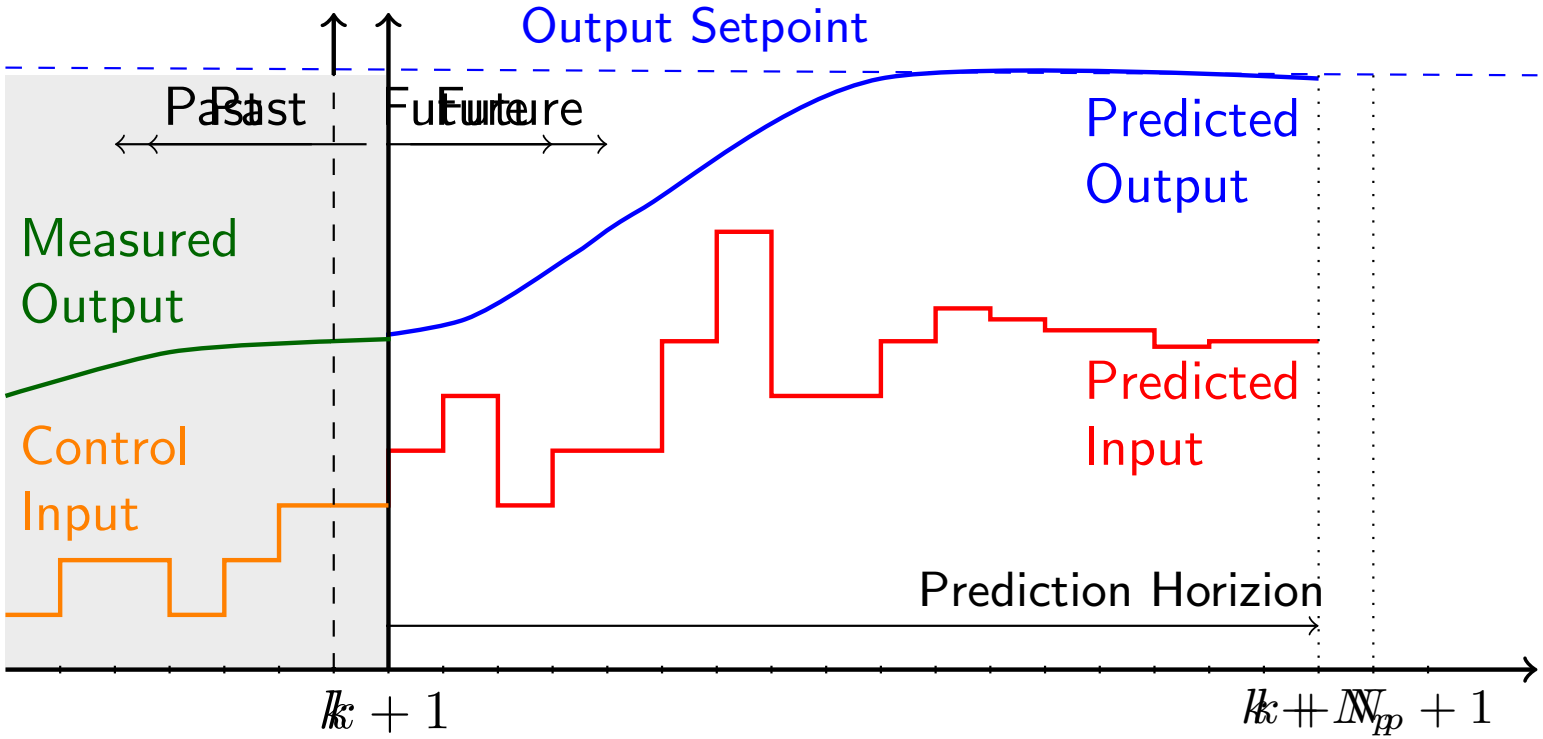
Use an explicit model of the process to predict the future behavior of the system

The prediction itself is not a novel concept...

- Smith Predictor (systems with time delay)
- Derivative action (e.g. PID)

Receding Horizon Control

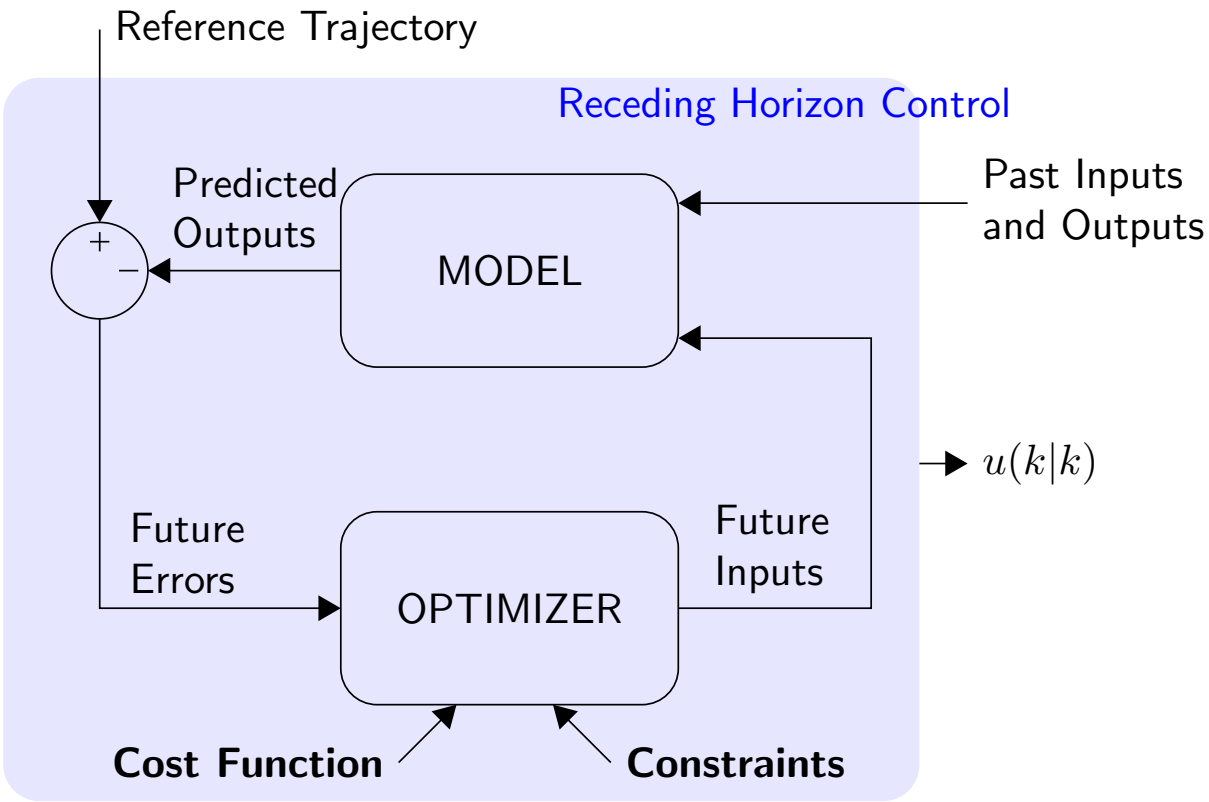
IDEA: At each iteration, the prediction horizon keeps being shifted forward!



At each iteration k

- predict the future behavior of the system (**fixed** horizon N_p)
- obtain an optimal input sequence
- apply only **the first predicted input**

Receding Horizon Control



An open loop optimal control problem is solved at each iteration time

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & h(x) \\ \text{subject to} \quad & g(x) \end{aligned}$$

$h(x)$ and $g(x)$ depends on the **model**, the **cost function** and the **constraints**.

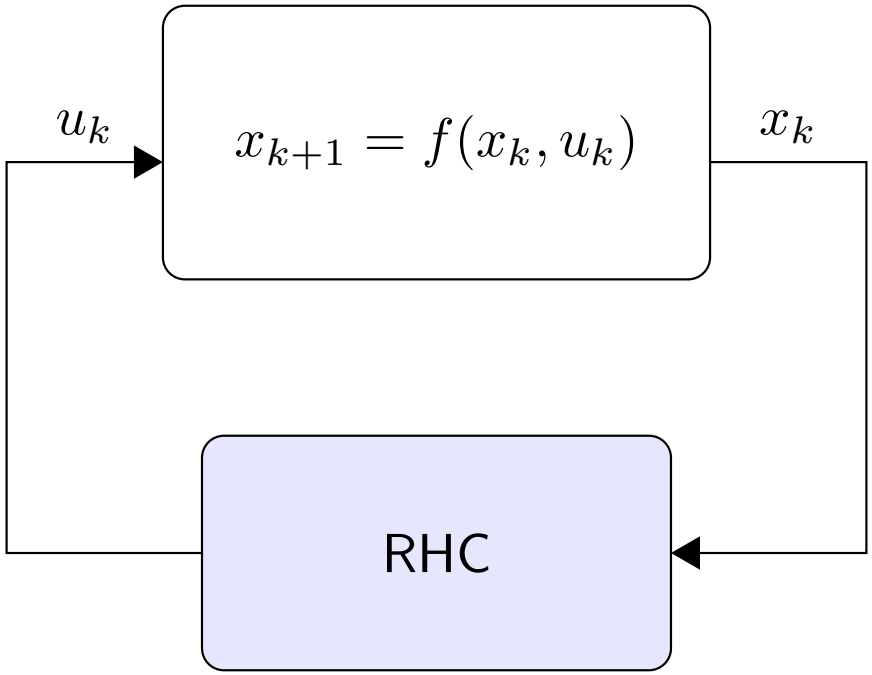
Receding Horizon Control

At each sampling instant an open loop optimization problem is solved

With perfect modeling and in the absence of disturbances, the state measured will correspond to the predicted one

It is a feedback control!

Feedback is guaranteed by the fact that **only the first input is applied** and at each k the new measurements are acquired
At each k the entire **procedure is repeated** with new x_k



MPC operation

Control design work flow:

1. obtain a model of the process, the accuracy of this model will deeply affect the results
2. define a cost function that represents a desired behavior of the system
3. implement a receding horizon policy

The most used structure consists of:

- Linear model (state space form)
- Quadratic Cost Function
- Affine Constraints

This structure is preferred for computational reasons $\rightarrow$ fast and certificated algorithms exist for this optimization problem.

N.B. An affine function is the composition of a linear function and a translation

Outline

Unconstrained Problem Formulation

- quadratic performance index
- linear model

$$\begin{aligned}
\min_{\underline{u}} \quad & \sum_{i=0}^{N_p-1} \left(\|Q y_{k+i|k}\|_2^2 + \|R u_{k+i|k}\|_2^2 \right) + \|P(y_{k+N_p|k})\|_2^2 \\
\text{s.t.} \quad & x_{k+i+1|k} = A x_{k+i|k} + B u_{k+i|k}, \\
& y_{k+i+1|k} = C x_{k+i+1|k}, \\
& x_{k|k} = x(k), \\
& i = 0, 1, \ldots, N_p - 1
\end{aligned}$$

- N_p is the prediction horizon
- Q are the weights on the outputs
- R are the weights on the inputs
- P is the final terminal cost (stability reason)
- $\underline{u} = [u_{k|k}, u_{k+1|k}, \ldots, u_{k+N_p-1|k}]$
- **trade-off between tracking error and control effort**

Unconstrained Problem Formulation

Linear Prediction Model

The dynamics of the process to control are expressed by a **linear, discrete time model**

$$x_{k+1} = Ax_k + B_u u_k$$
$$y_k = Cx_k$$

with $x \in \mathcal{R}^{n_x}$, $u \in \mathcal{R}^{n_u}$.

The objective function is therefore minimized subject to the predicted dynamics

$$x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k}$$
$$y_{k+i+1|k} = Cx_{k+i+1|k}$$

Unconstrained Problem Formulation
If the process is non-linear?

Linearization: it is possible only when the nonlinearities are negligible and when the system operates always near the linearization point. To deal with this last point, a **switching MPC** can be used

Trajectory linearization: at each sampling interval, the model is linearized around the current state; it increases the computational effort as a time-variant MPC must be used

Nonlinear MPC: the nonlinear model is directly considered . At each iteration a nonlinear optimization problem is solved. The computational effort could be very high to avoid local optima of the problem

. . . *and more*

Unconstrained Problem Formulation

How to construct the cost function? Simplest example: $C = I \rightarrow y \equiv x$

$$\text{Objective Function} \rightarrow f(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}, x_0) = x_0' Q x_0 + \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N_p-1} \\ x_{N_p} \end{bmatrix}' \underbrace{\begin{bmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & \dots & 0 & Q & 0 \\ 0 & 0 & \dots & 0 & P \end{bmatrix}}_{\bar{Q}} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N_p-1} \\ x_{N_p} \end{bmatrix} +$$

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

$$\text{System dynamics} \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N_p-1} \\ x_{N_p} \end{bmatrix} = \underbrace{\begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ A^{N_p-1}B & A^{N_p-2} & \dots & B \end{bmatrix}}_{S_u} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N_p-2} \\ u_{N_p-1} \end{bmatrix} + \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N_p-2} \\ A^{N_p-1} \end{bmatrix}}_{S_x} x_0$$

Unconstrained Problem Formulation

How to construct the cost function? Simplest example: $C = I \rightarrow y \equiv x$

The system dynamics, originally imposed as constraints, are substituted into the cost function obtaining $f(\underline{u}, x_0)$

$$f(\underline{u}, x_0) = x_0' Q x_0 + (S_u \underline{u} + S_x x_0)' \bar{Q} (S_u \underline{u} + S_x x_0) + \underline{u}' \bar{R} \underline{u} =$$

$$= 0.5 \underline{u}' \underbrace{2 \left(\bar{R} \underline{u} + S_u' \bar{Q} S_u \right)}_{H} + \underline{u}' \underbrace{2 S_u' \bar{Q} S_x}_{F} x_0 + \cancel{x_0' \left((\bar{Q} + S_x' \bar{Q} S_x) \right) x_0}$$

SOLVE each iteration (k)

$$\min_{\underline{u}} \quad f(x_0, \underline{u}) = 0.5 \underline{u}' H \underline{u} + \underline{u}' F x_0$$

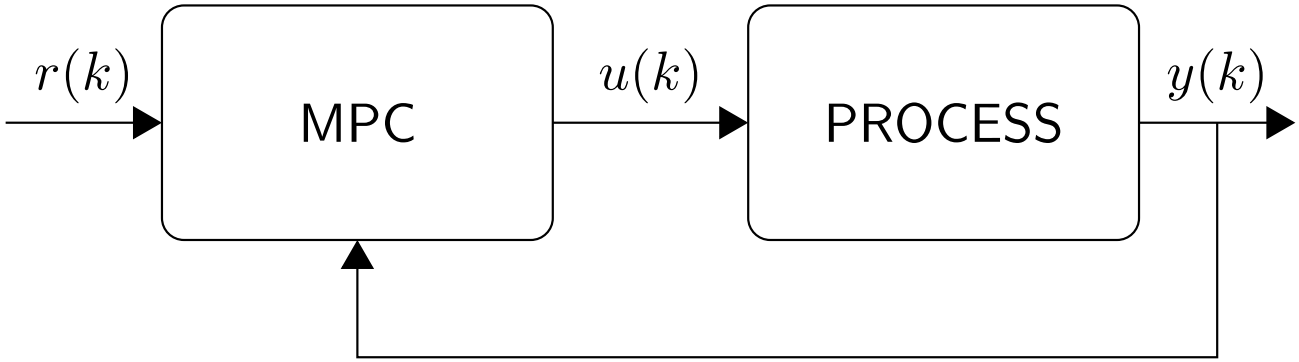
and apply only the first element of $\underline{u} \Rightarrow u_{k|k}$

The unconstrained problem is easy to solve: **the optimal $\underline{u}^*$ is the point where the gradient of the cost function vanishes**

$$\nabla f(x_0, \underline{u}) = H \underline{u} + x_0' F' = 0 \;\rightarrow\; \underline{u}^* = -H^{-1} (F x_0)'$$

Tracking

The objective of the control is to make the output $y(k)$ tracking a reference r



The tracking error must vanish $e(k) = (r(k) - y(k)) \to 0$

The error is minimized within the objective function

$$J = \sum_{i=0}^{N_p-1} \|Q(y_{k+i|k} - r)\|_2^2 + \|P(y_{k+N_p|k} - r)\|_2^2 + \eta(\underline{\boldsymbol{u}})$$

How to deal with $\eta(\underline{\boldsymbol{u}})$?
A solution could be providing a reference value for the inputs too, but "Δu formulation" is more common ...

Tracking

Idea: Δu Formulation

As the control objective is to track a reference different from 0, the required control input at the steady-state can be different from 0. Thus the problem is formulated minimizing the objective function respect to the **input increments**.

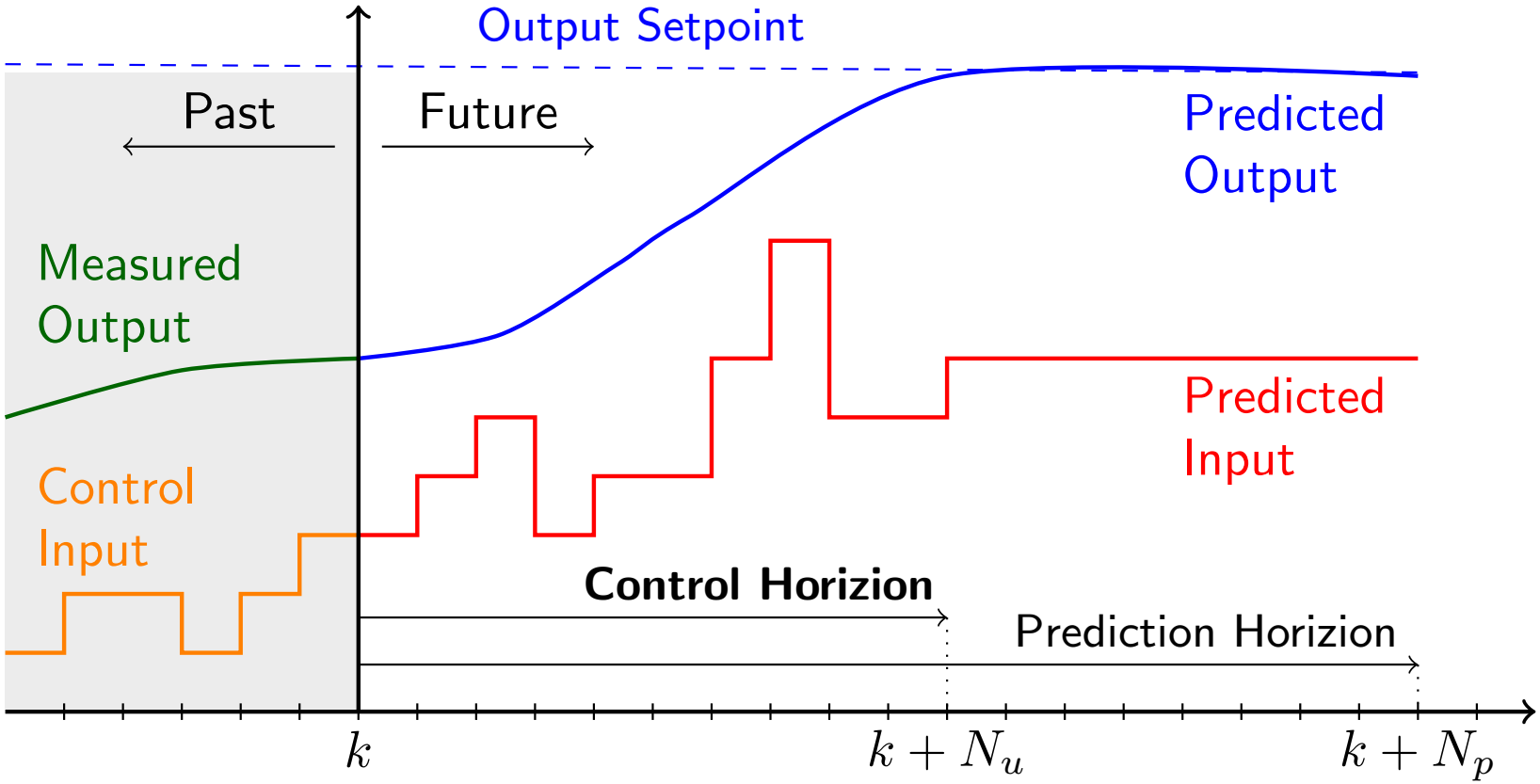
$$\Delta u(k) = u(k) - u(k-1)$$

The optimization problem is formulated as in the following

$$\begin{aligned}
\min_{\Delta \underline{u}} \quad & \sum_{i=0}^{N_p-1} \left(\|Q(y_{k+i|k} - r)\|_2^2 + \|R\Delta u_{k+i|k}\|_2^2 \right) + \|P(y_{k+N_p|k} - r)\|_2^2 \\
\text{s.t.} \quad & x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k}, \\
& y_{k+i+1|k} = Cx_{k+i+1|k}, \\
& x_{k|k} = x(k), \\
& i = 0, 1, \ldots, N_p - 1
\end{aligned}$$

Control Horizon

- Idea $\rightarrow$ differentiate between control and prediction horizon!
- Prediction Horizon: # of steps for output prediction $[N_p]$
- Control Horizon: # of inputs moves to be optimized $[N_u]$



$$\forall k \in \{N_u + 1, N_u + 2, \ldots, N_p - 1\} \rightarrow \Delta u(k) = 0 \tag{1}$$

Control Horizon

Why Control Horizon?

- The number of optimization variable depends, **only**, on the size of vector $\Delta\underline{u}$.
- A shorter control horizon **reduces the effort** to compute the optimal input sequence

$$
\begin{aligned}
\min_{\Delta\underline{u}} \quad & \sum_{i=0}^{N_p-1} \|Q(y_{k+i|k} - r)\|_2^2 + \sum_{i=0}^{N_u-1} \|R\Delta u_{k+i|k}\|_2^2 + \|P(y_{k+N_p|k} - r)\|_2^2 \\
\text{s.t.} \quad & x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k}, \\
& y_{k+i+1|k} = Cx_{k+i+1|k}, \\
& x_{k|k} = x(k), \\
& \Delta u_{k+N_u+j|k} = 0, \\
& i = 0, 1, \ldots, N_p - 1 \\
& j = 0, 1, \ldots, N - N_u - 1
\end{aligned}
$$

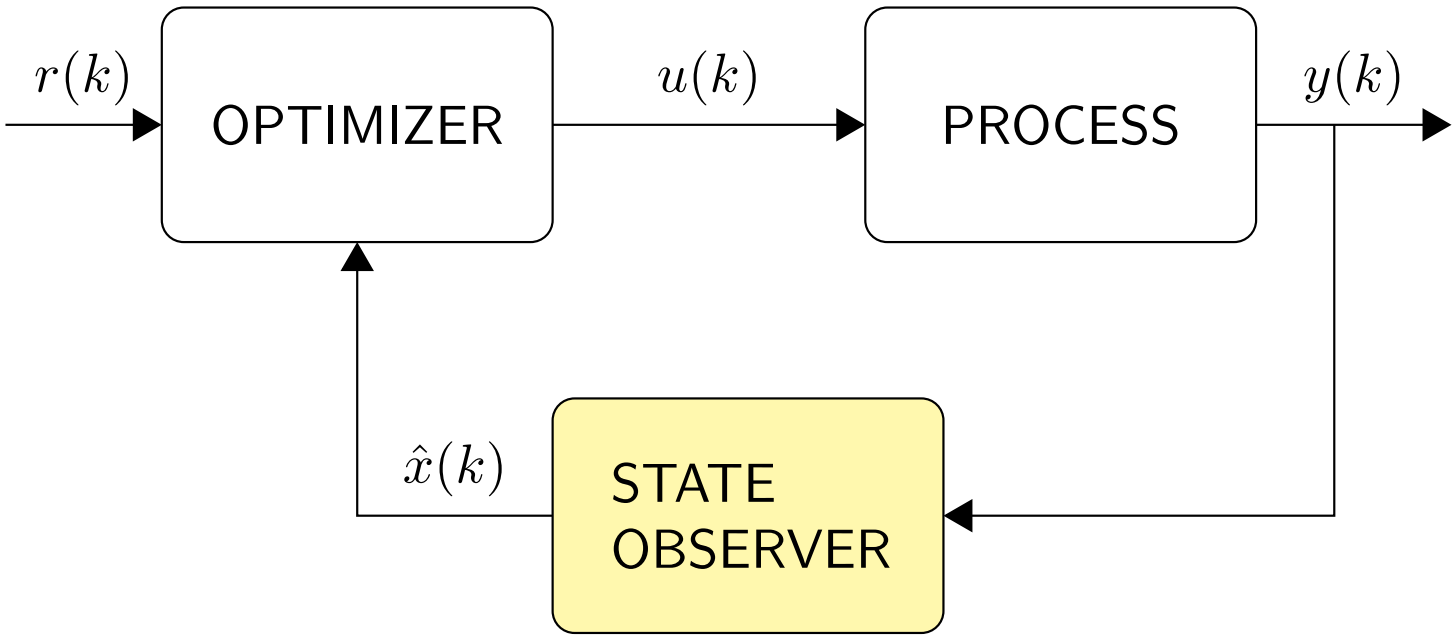
You can consider N_u as the number of available moves for your controller.

The input sequence is optimized until N_u, $N_u \ll N_p$

State Estimation in MPC

Problems...

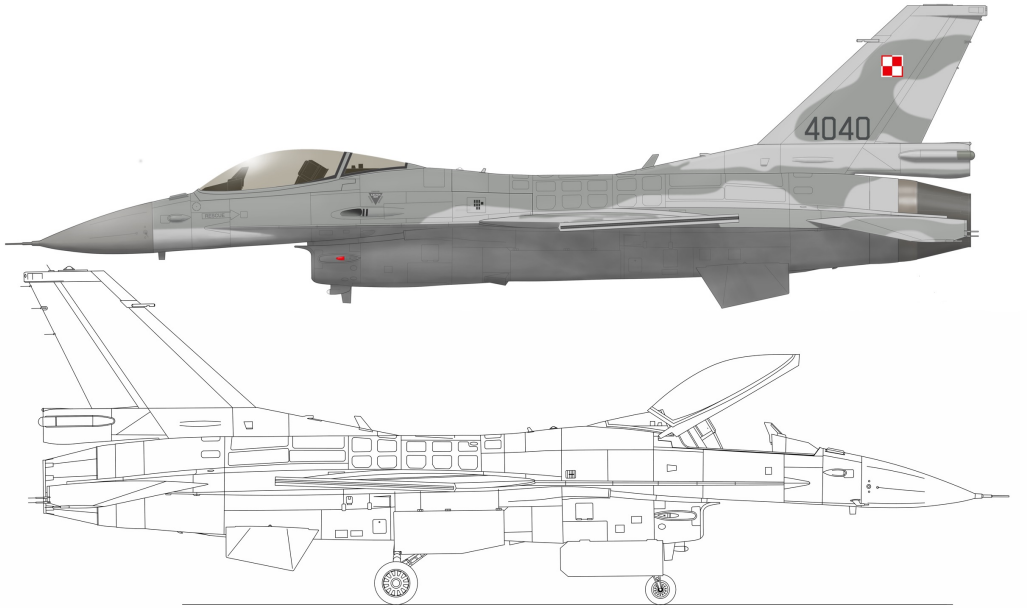
- The optimization needs the **states** of the process. They could **not be available**. Only $y(k)$ measurements can be used
- Noise can affects the measurements $y(k)$



A state observer is needed. The standard choice is a **Kalman Filter**.

AFTI F16 - example

- Inputs:
  - elevator angle
  - flaperon angle
- Outputs:
  - attack angle
  - pitch angle



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

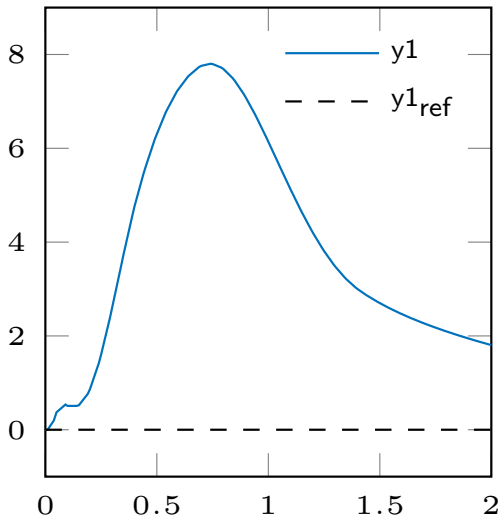
$$y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x$$

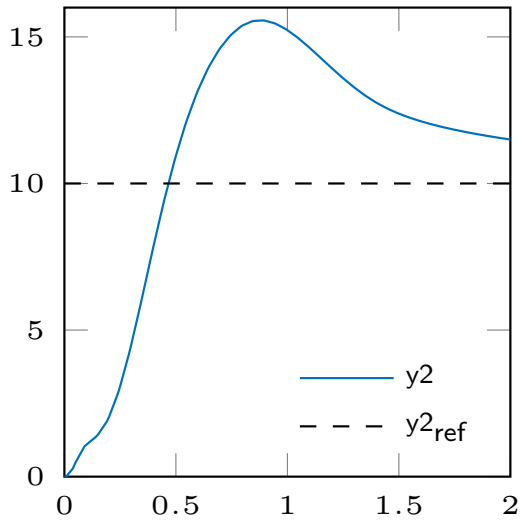
- the system has constraints on both input angles
- **the open loop response is unstable!**

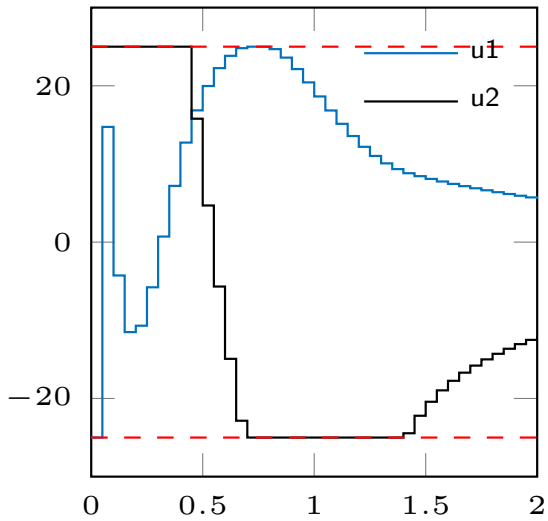
AFTI F16 - example

MPC design:

- prediction horizon $N_p = 10$
- control horizon $N_u = 2$
- output weights $Q = \text{diag}[10, 10]$
- input weights $R = \text{diag}[0.01, 0.01]$
- the angles' constrains are treated as saturation $u_1, u_2 \in \{-25°, +25°\}$







Outline

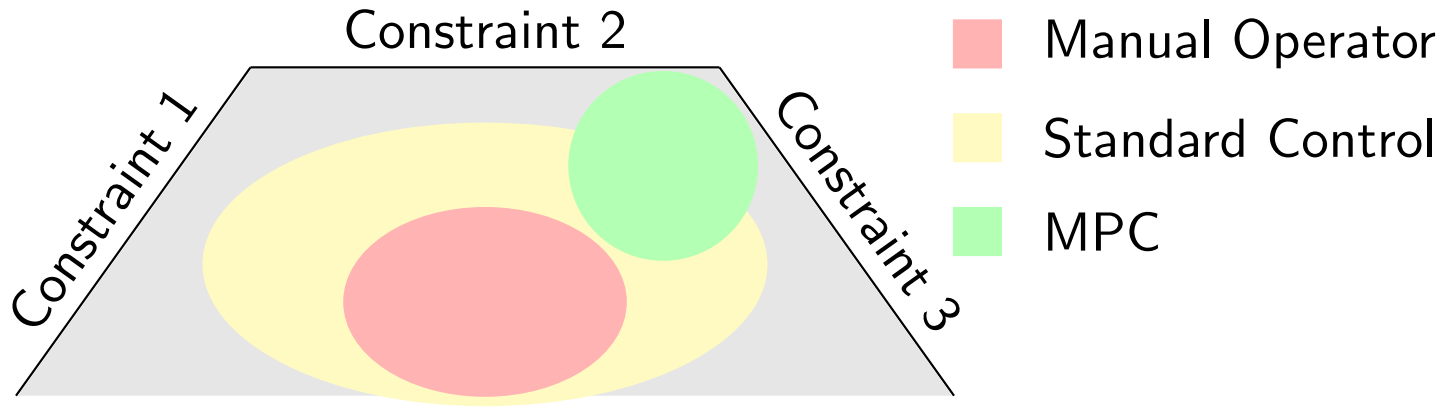
System Constraints

All processes are subject to constraints!!

- **Actuators**:
  - limited range of action
  - limited slew rate
- **System variables**:
  - constructive reasons (e.g. electrical, mechanical)
  - safety reasons
  - sensors' range
- **Performance**
  - energy saving
  - economic goal
  - quality

The most profitable operation for a system is obtained, commonly, when it operates **near the constraints**

MPC operates near the constraints



MPC Drives the system close to the constraints $\rightarrow$ **higher profit**

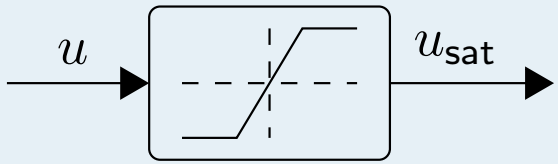
Some examples:

- Underwater and aerial vehicles have to perform maneuvers as quickly as possible, but they typically have constraints on allowable pitch an roll angles
- Products from manufacturing must meet a level of a certain quality
- Power converters must react as quickly as possible to load changes, but constraints on current peaks must be respected to avoid damages

How to handle constraints?

With standard control, the constraints on the system are handled with **limiters**, but...

Saturation is dangerous!



The feedback control structure is broken and this **could lead to instability**.

With MPC the constraints:

- are directly taken into account;
- an optimal solution is found, which satisfies the constraints
- it is possible to impose constraints both on inputs, outputs and their combinations;

Constraints in MPC

$$
\begin{aligned}
\min_{\Delta \underline{u}} \quad & \sum_{i=0}^{N_p-1} \|Q(y_{k+i|k} - r)\|_2^2 + \sum_{i=0}^{N_u-1} \|R\Delta u_{k+i|k}\|_2^2 + \|P(y_{k+N_p|k} - r)\|_2^2 \\
\text{s.t.} \quad & x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k}, \\
& y_{k+i+1|k} = Cx_{k+i+1|k}, \\
& x_{k|k} = x(k), \\
& \Delta u_{k+N_u+j|k} = 0, \\
& u_{k+i|k} \in \mathbb{U}, \\
& x_{k+i+1|k} \in \mathbb{X}, \\
& \Delta u_{k+i|k} \in \mathbb{U}_\Delta, \\
& i = 0, 1, \ldots, N_p - 1 \\
& j = 0, 1, \ldots, N - N_u - 1
\end{aligned}
$$

- input constraints: $u_{k+i|k} \in \mathbb{U}$
- input rate constraints: $\Delta u_{k+i|k} \in \mathbb{U}_\Delta$
- output constraints: $x_{k+i+1|k} \in \mathbb{X}$

Constraints in MPC

The simplest case is represented by **box constraints**:

$$u_{\min} \leq u(k) \leq u_{\max}$$
$$\Delta u_{\min} \leq \Delta u(k) \leq \Delta u_{\max}$$
$$x_{\min} \leq x(k) \leq x_{\max}$$

When the constraints are linear, the three contributions can be casted in a unique linear system of the form $G\Delta\underline{u} \leq b$.

SOLVE at each iteration (k)

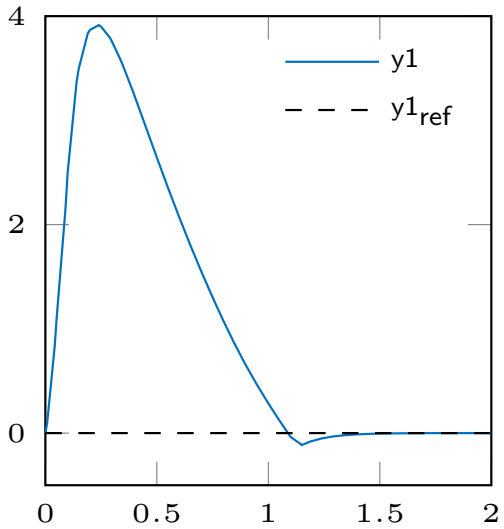
$$\min_{\Delta\underline{u}} \quad f(x_0, \Delta\underline{u}) = 0.5\Delta\underline{u}'H\Delta\underline{u} + \Delta\underline{u}'h$$
$$\text{s.t.} \quad s(x) = G\Delta\underline{u} \leq b$$

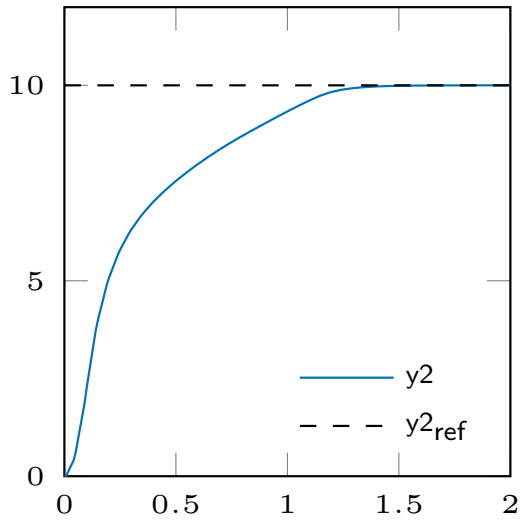
and apply only the first element of $\underline{u}(k) = \underline{u}(k-1) + \Delta\underline{u}(k)$

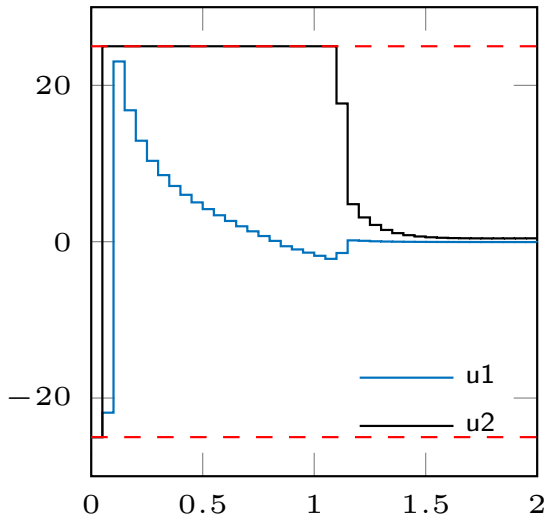
AFTI F16 - example

$N_p = 10,\ N_u = 2,\ Q = [10, 10],\ R = [0.01, 0.01],\ u_1, u_2 \in \{-25°, +25°\}$

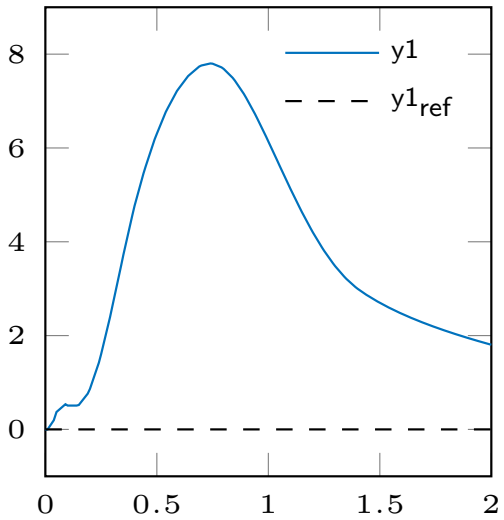
- constrained

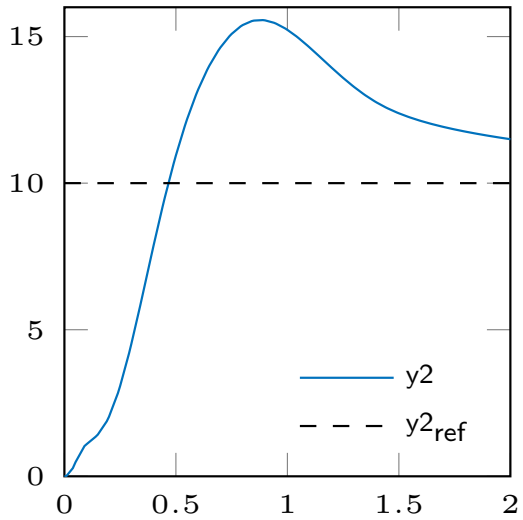


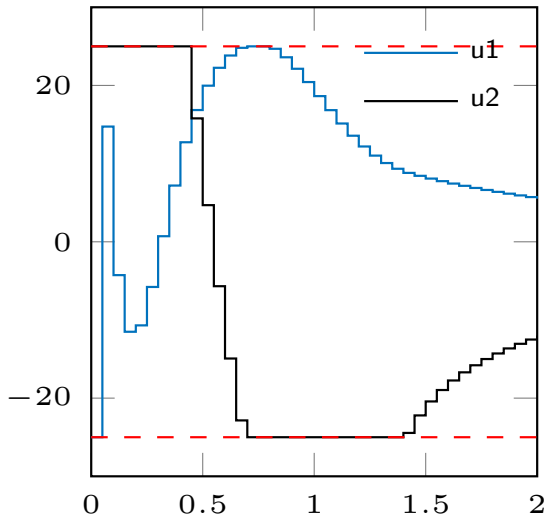




- unconstrained + saturation



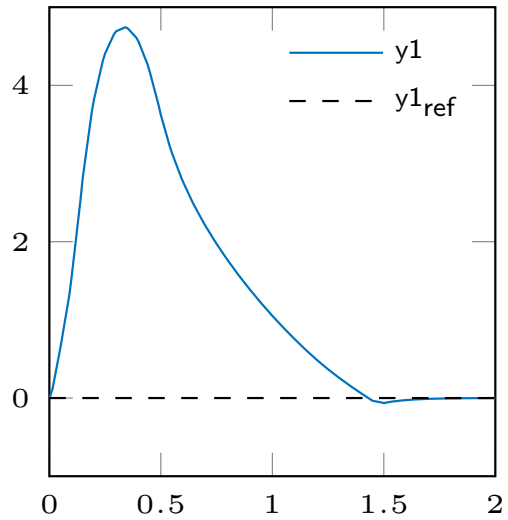


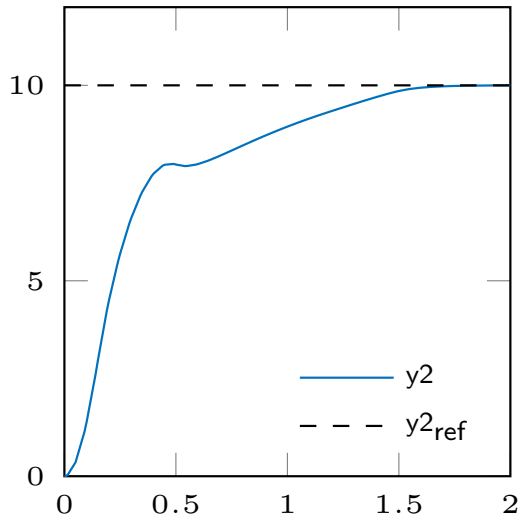


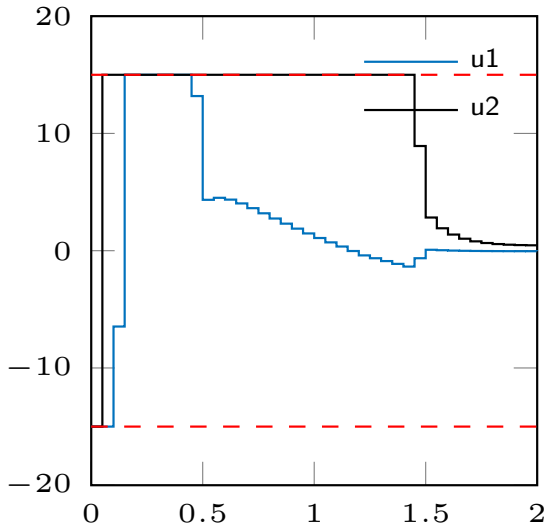
AFTI F16 - example

$N_p = 10,\ N_u = 2,\ Q = [10, 10],\ R = [0.01, 0.01],\ u_1, u_2 \in \{-15°, +15°\}$

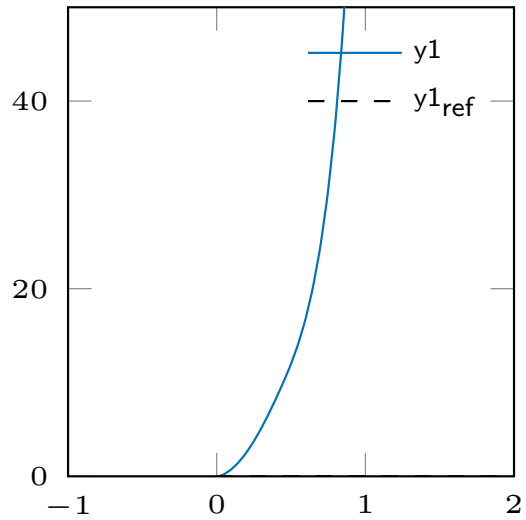
- constrained

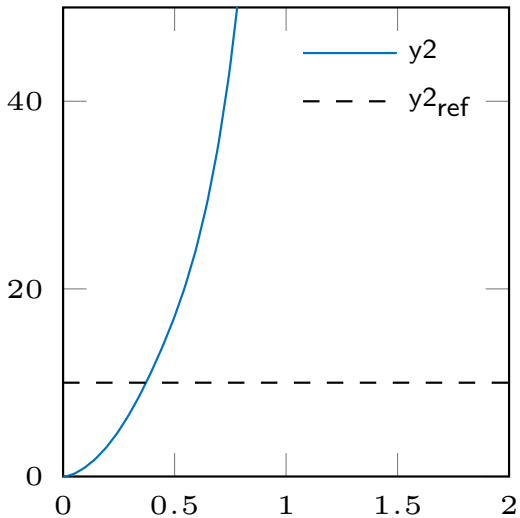


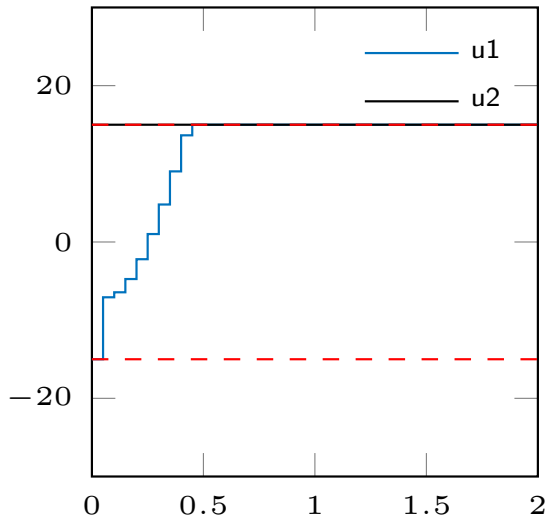




- unconstrained + saturation unstable!







Hard and Soft Constraints

- Input Constraints: they can always be enforced
- Output Constraints: could lead to **infeasibility** (the problem has no solution!) due to:
  - Model Uncertainties
  - Disturbances

Idea

Classify the constraints in:

- **Hard**: no violations are allowed at any time
- **Soft**: temporary violations of the constraints might be tolerated

Usually hard constraints include actuators limits or safety constraints; output bounds or performance constraints are usually imposed softened.

Hard and Soft Constraints

Output constraints relaxation:

$$
\begin{aligned}
\min_{\Delta \underline{\boldsymbol{u}}} \quad & 0.5 \Delta \underline{\boldsymbol{u}}' H \Delta \underline{\boldsymbol{u}} + \Delta \underline{\boldsymbol{u}}' h + \rho_\epsilon \epsilon^2 \\
\text{s.t.} \quad & u_{\min} \leq u_{k+i|k} \leq u_{\max}, \\
& \Delta u_{\min} \leq \Delta u_{k+i|k} \leq \Delta u_{\max}, \\
& y_{\min} - \epsilon \leq y_{k+i|k} \leq y_{\max} + \epsilon, \\
& \Delta u_{k+N_u+j|k} = 0, \\
& i = 0, 1, \ldots, N_p - 1 \\
& j = 0, 1, \ldots, N - N_u - 1
\end{aligned}
$$

- $\rho_\epsilon \gg Q, R$

The problem is transformed into

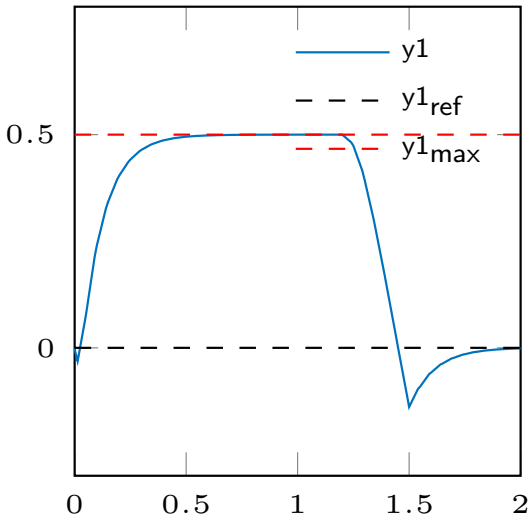
$$
\begin{aligned}
\min_{z} \quad & 0.5 z' H z + h' z \\
\text{s.t.} \quad & G z \leq b
\end{aligned}
$$

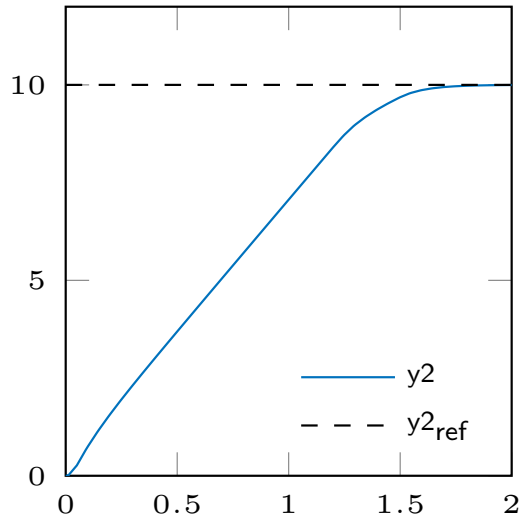
with $z = [\Delta \underline{\boldsymbol{u}}, \epsilon]'$

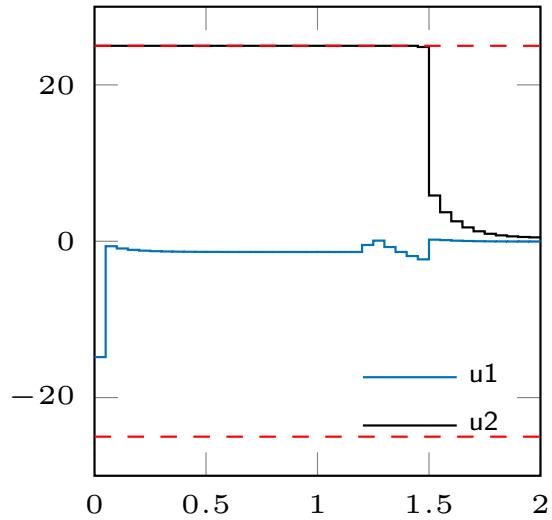
AFTI F16 - example

$N_p = 10$, $N_u = 2$, $Q = [10, 10]$, $R = [0.01, 0.01]$, $u_1, u_2 \in \{-15°, +15°\}$, $y_1 \in \{-0.5°, 0.5°\}$

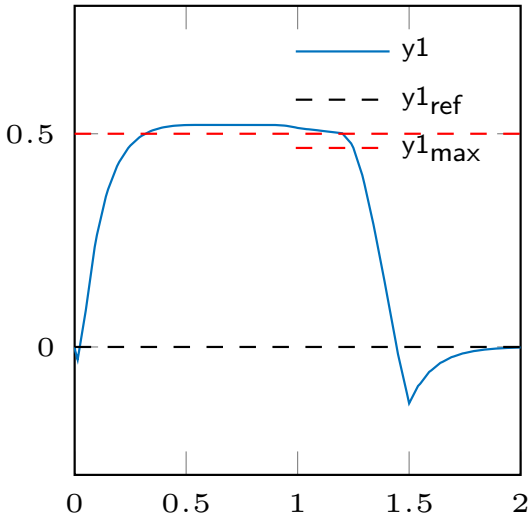
- hard output constraints

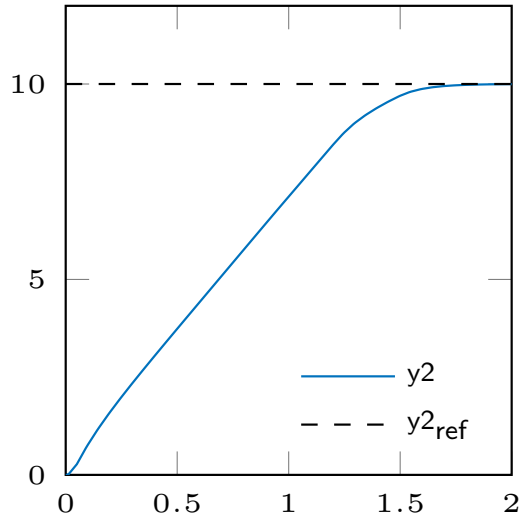


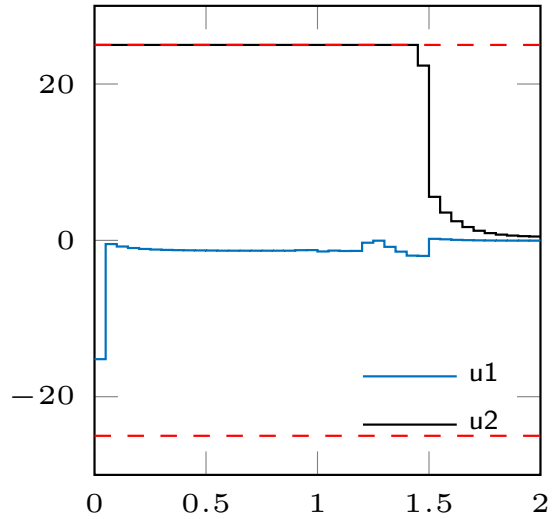




- soft output constraints







Outline

Quadratic Programming

Linear MPC can be casted into Quadratic Programming

$$\begin{aligned} \min_{z} \quad & 0.5 z' H z + h' z \\ \text{s.t.} \quad & Gz \leq b \end{aligned}$$

In the **unconstrained case**, the explicit solution is directly applicable
$z^* = -H^{-1} h$

. . . in constrained case an **optimization problem must be solved**.

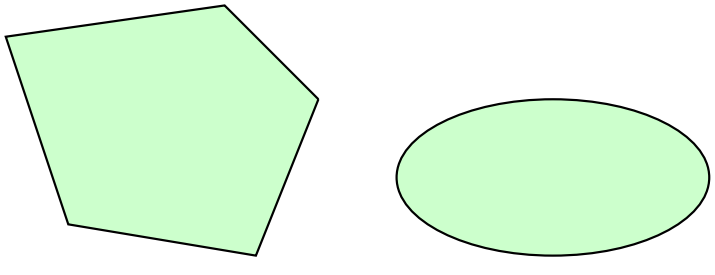
Generalities of **optimization problems**:

- difficult to solve
- long computation time
- solution could not be found

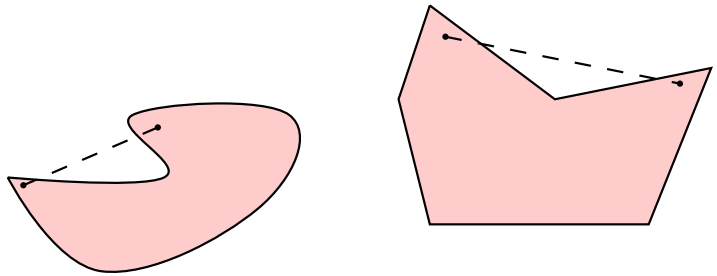
Exceptions where the solution is easy to find, and the process can be fast

- linear programming (LP)
- quadratic programming (QP)
- **convex optimization**

Convex Sets



Convex Sets



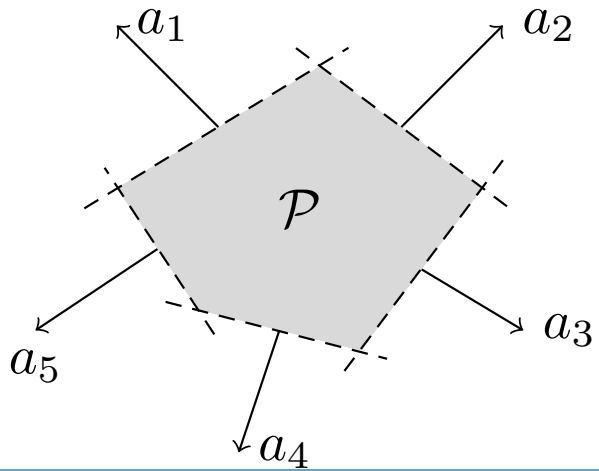
Non Convex Sets

A set $S \in X$ is convex if

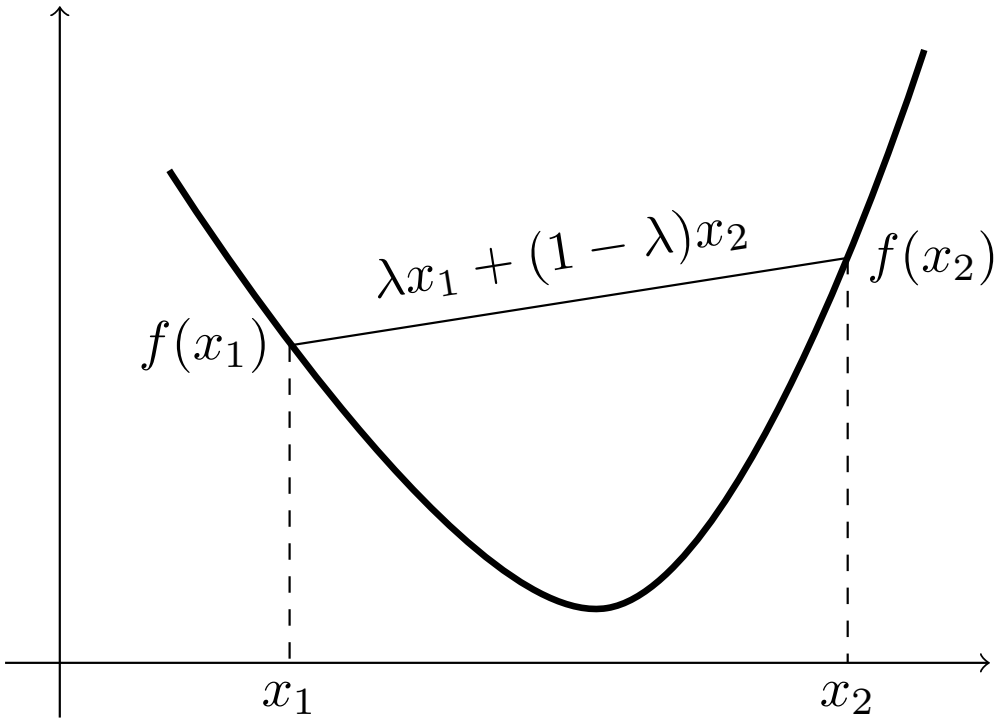
$$\forall x_1, x_2 \in S \Rightarrow \lambda x_1 + (1-\lambda)x_2 \in S,\ \lambda \in [0,1]$$

Polyhedra: solution set of a finite number of linear equalities and inequalities:

$$\mathcal{P} = \{x | a_i' x \leq b_i,\ i = 1, \ldots, n,\ c_j' x = d_j,\ j = 1, \ldots, m\}$$



Convex Function



A function $f : S \to X$ is convex if S is convex and

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

Convex Optimization

$$\begin{aligned} \min_{z} \quad & f_0(z) \\ \text{s.t.} \quad & f_i(z) \leq 0, \ i = 1, \ldots, m \\ & Gz = b \end{aligned}$$

where f_0 and f_i must be convex.

Why Convex Optimization?

- every locally optimal solution is globally optimal (uniqueness)
- fast algorithms exist for convex optimization

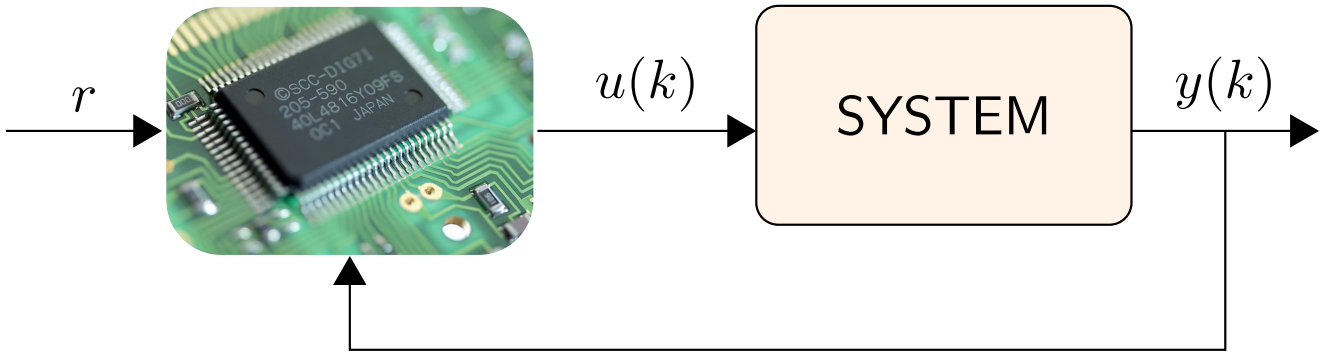
$\rightarrow$Quadratic programming is a particular case of convex optimization

$$\begin{aligned} \min_{z} \quad & 0.5 z' H z + h' z \\ \text{s.t.} \quad & g_i' z \leq b_i, \ i = 1, \ldots, m \end{aligned}$$

Outline

Embedded MPC

- the controlled process requires a simple formulation
  - the model has few variables (< 10)
  - the control horizon is reasonably short (< 10)
- the **time complexity is a big issue**
  - the system is fast (μs-ms scale)
  - a cheap board is used (MHz)
  - a limited memory is available (< 500kB)



The concept of "embedded" is going to vary, according to the power of cheap boards that is constantly growing

Embedded MPC

In MPC the computational burden is **very high** compared to other standard and advanced control, as...
...**a model based optimizer must be embed into the control board!**

Reliable and efficient solvers exist to solve QPs

Requirements for an embedded solver:

- **fast**: the control action must be calculated within the sampling interval
- **simple hardware**: the code must be suitable for simple architectures (e.g. fixed point and single precision)
- **simple code**: the code must be simple enough to be verifiable and certifiable
- **little memory**: both the solver code and the data needed must fit into the low memory of a cheap board

Embedded MPC

The time required for the solution can be divided into

Building

Time required to construct a **condensed form** from the problem formulation.

- In time-invariant applications the Hessian H, and the constraints matrix G can be pre-calculated **offline**

Solution of the optimization problem

Time needed to solve the constrained QP problem, given a expected *feasibility* and *optimality tolerance*

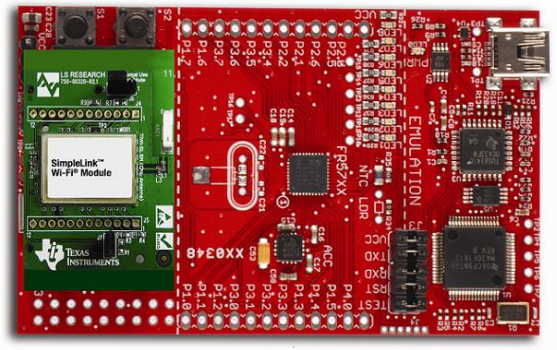
Commonly used QP solvers:

- active-set methods
- interior point methods
- gradient or Newton based methods

Embedded Solver - example

MSP-EXP430FR5739 Experimenter Board
(Wireless Sensors Network Applications)

- 16KB FRAM / 1KB SRAM
- 16-Bit RISC Architecture up to 24-MHz
- fixed point arithmetics
- 2x Timer$_A$ Blocks, 3x Timer$_B$ Block
- 1x USCI (UART/SPI/IrDA/I2C) Blocks, 16Ch 10-Bit ADC12$_B$, 16Ch Comp$_D$, 32 I/Os



	FRAM	EEPROM	Flash
Time to write 64 bytes to memory	$1.6\mu s$	$2200\mu s$	$6400\mu s$
Time to read 64 bytes to memory	$1.6\mu s$	$4.5\mu s$	$4.5\mu s$
Number of write cycles	100 trillion	$500000s$	100000
Voltage needed to write	$1.5V$	10 to $14V$	10 to $14V$

Convex Optimization Algorithms

Embedded Solver - example

The board has been tested with convex optimization algorithms **simple to code**

- dual gradient projection [GPD] (converge proof for fixed point)
- fast dual gradient projection [GPAD]
- alternating direction method of multipliers [ADMM]
- parallel quadratic programming [PQP]

Random generated QP problems have been used to test the algorithms; different sizes of QP have been tested considering

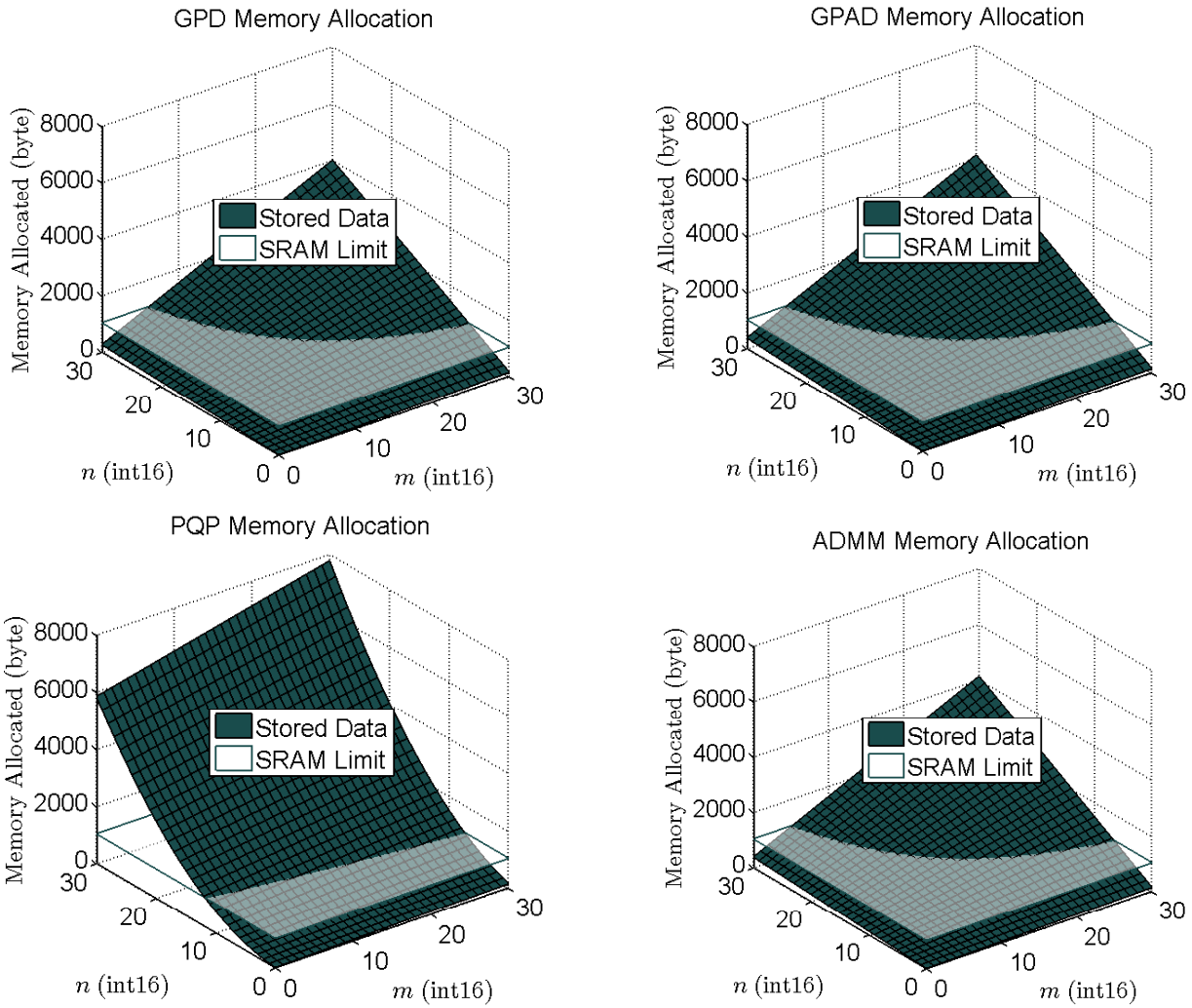
$$\begin{aligned} \min_{z} \quad & 0.5 z' H z + h' z \\ \text{s.t.} \quad & Gz \leq b \end{aligned}$$

with $H \in \Re^{n \times n}$, $G \in \Re^{m \times n}$ and $n \in [2; 10]$, $m \in [4; 20]$

Memory Allocation

Embedded Solver - example

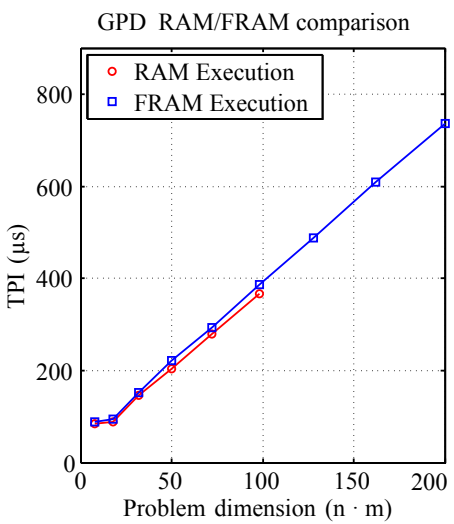
For embedded application, a static memory allocation is preferable

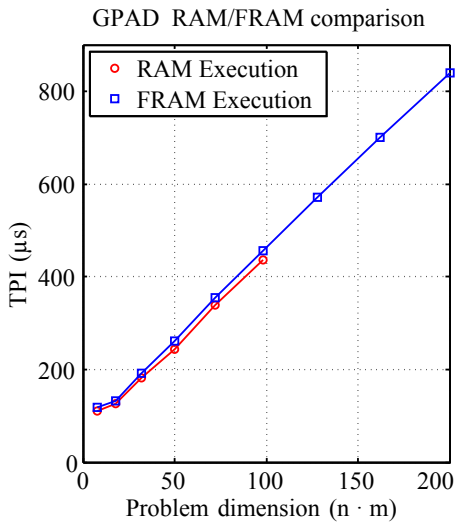


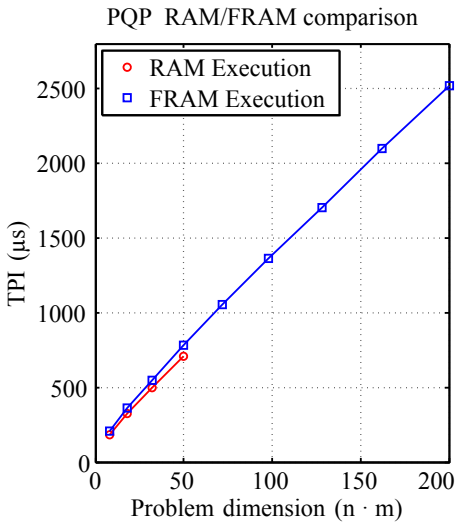
FRAM vs SRAM Benchmark

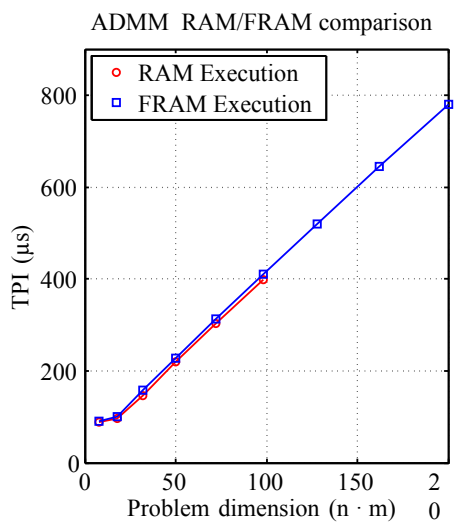
Embedded Solver - example

Evaluation of single Time Per Iteration (TPI)





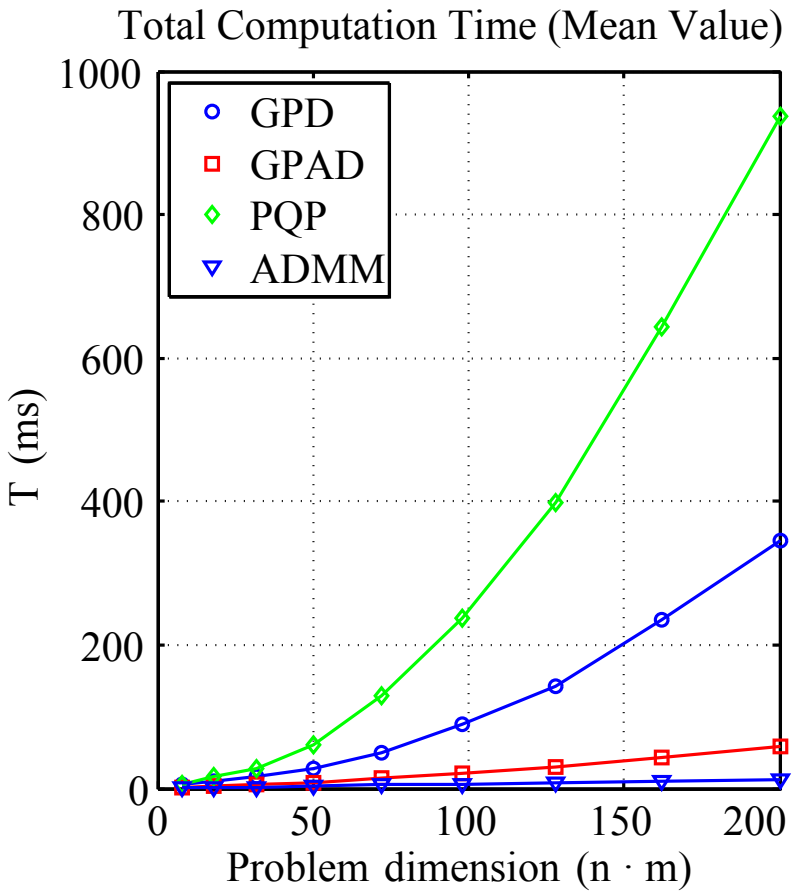


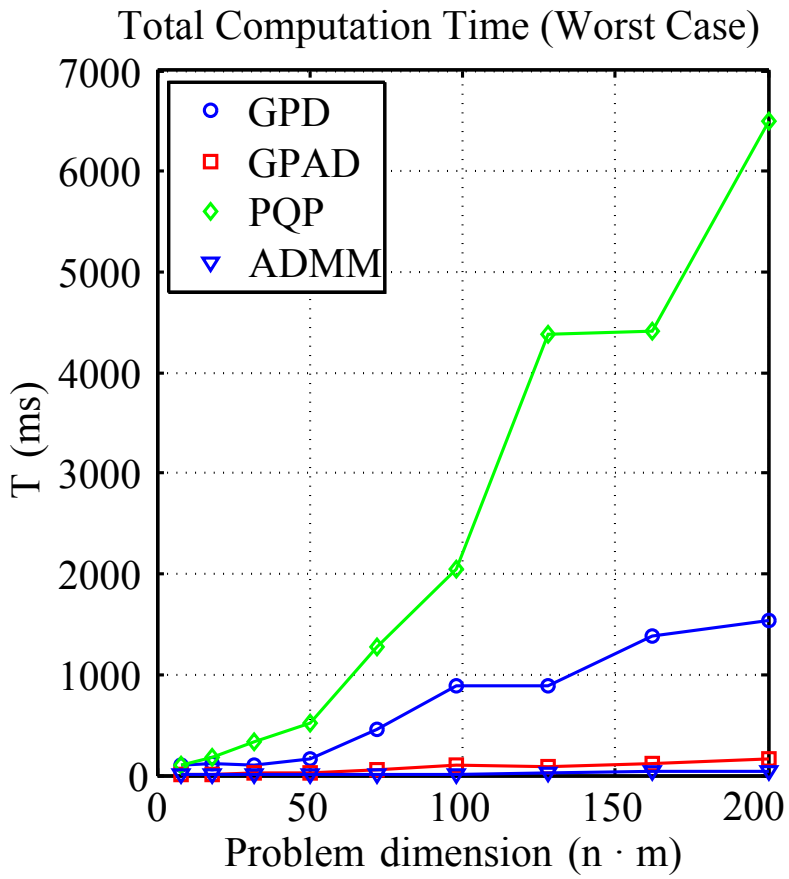


FRAM memory doesn't worse the TPI in a substantially way!

Solvers Benchmark

Embedded Solver - example





Cimini, G. and Bemporad, A. and Ippoliti , G. and Longhi , S. *"Explicit sensorless model predictive control of synchronous buck converter"*, 2014 6th European Embedded Design in Education and Research Conference (EDERC), pp.187-191, Sept 2014

Outline

Introduction

MPC in electrical drives and power conversion is, mainly, motivated by two facts:

- mathematical models of these systems are relatively well known
- several constraints are needed for safety reasons

The very big issue is the computational effort $\rightarrow$ power converters and electric drives are systems were the sampling time is $1/0.1\mu s$

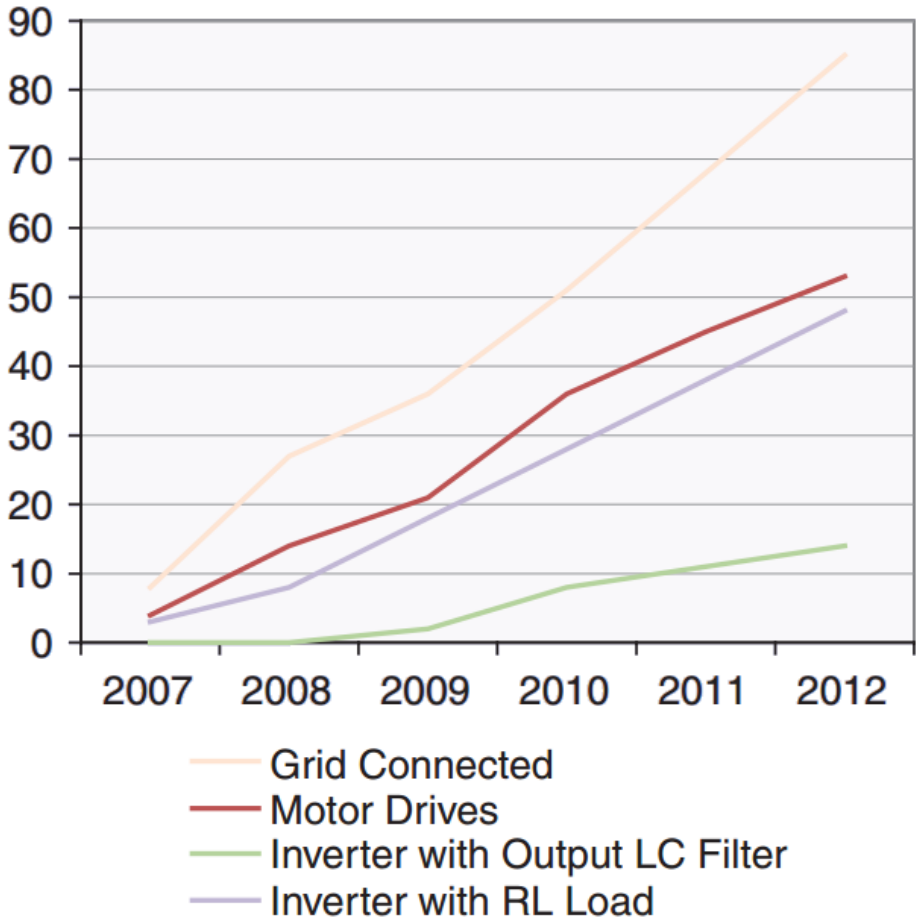
It is often solved considering **1-step optimization**

Some references:

- *Predictive Control of Power Converters and Electrical Drives*, J. Rodriguez, P. Cortes, Wiley, 2012
- *Model Predictive Control: A Review of Its Applications in Power Electronics*, Vazquez, S.; Leon, J.I.; Franquelo, L.G.; Rodriguez, J.; Young, H.A.; Marquez, A.; Zanchetta, P., IEEE Industrial Electronics Magazine, vol.8, no.1, pp.16,31, March 2014

Introduction - Growing interest!

Figure: The research works of MPC for PWM power converters published in IEEE conferences and journals from 2007 to 2012: the cumulative analysis for each application category



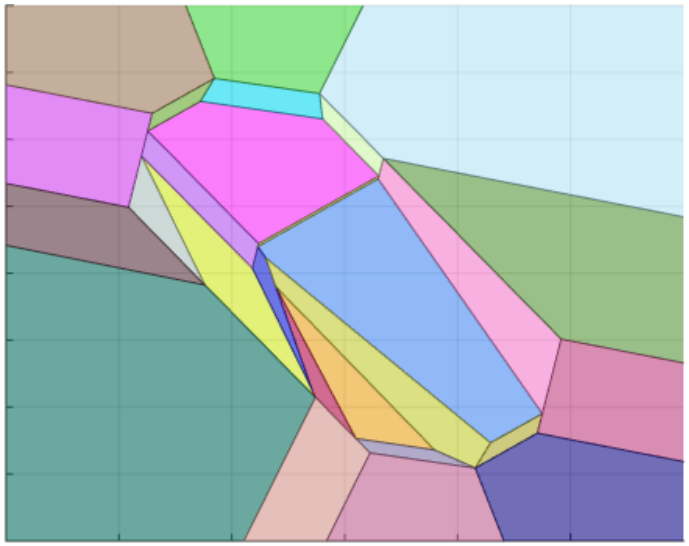
Existing solutions - Explicit MPC

$$\min_{z} \; 0.5z'Hz + z'Fx_0$$
$$\text{s.t.} \; Gz \leq s + Sx_0$$

continuous, piecewise affine problem

The optimization problem is pre-solved **offline** via multiparametric Quadratic Programming (mpQP)

- ✓ it is an easy-to-implement PieceWise Affine (PWA) function of the parameters
- ✓ fast binary search tree algorithm can be used
- ✗ it is applicable only to small problems
- ✗ high memory requirements

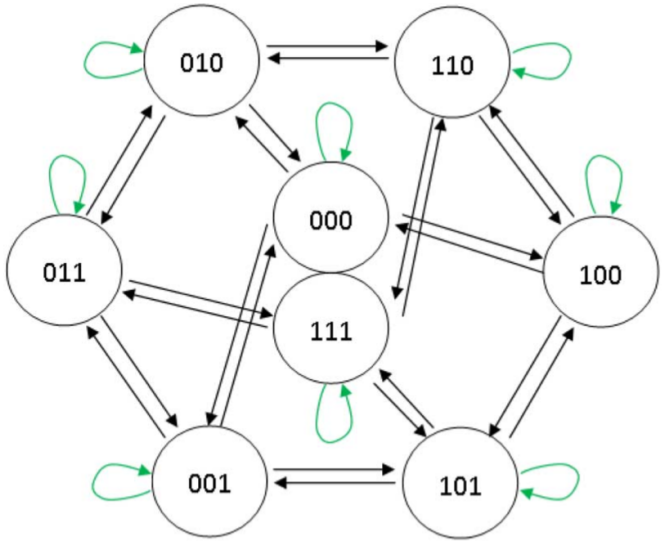


[a] Bolognani, S.; Bolognani, S.; Peretti, L.; Zigliotto, M., *"Design and Implementation of Model Predictive Control for Electrical Motor Drives"*, IEEE Transactions on Industrial Electronics, vol.56, n.6, pp.1925-1936, June 2009

Existing solutions - Finite Control Set

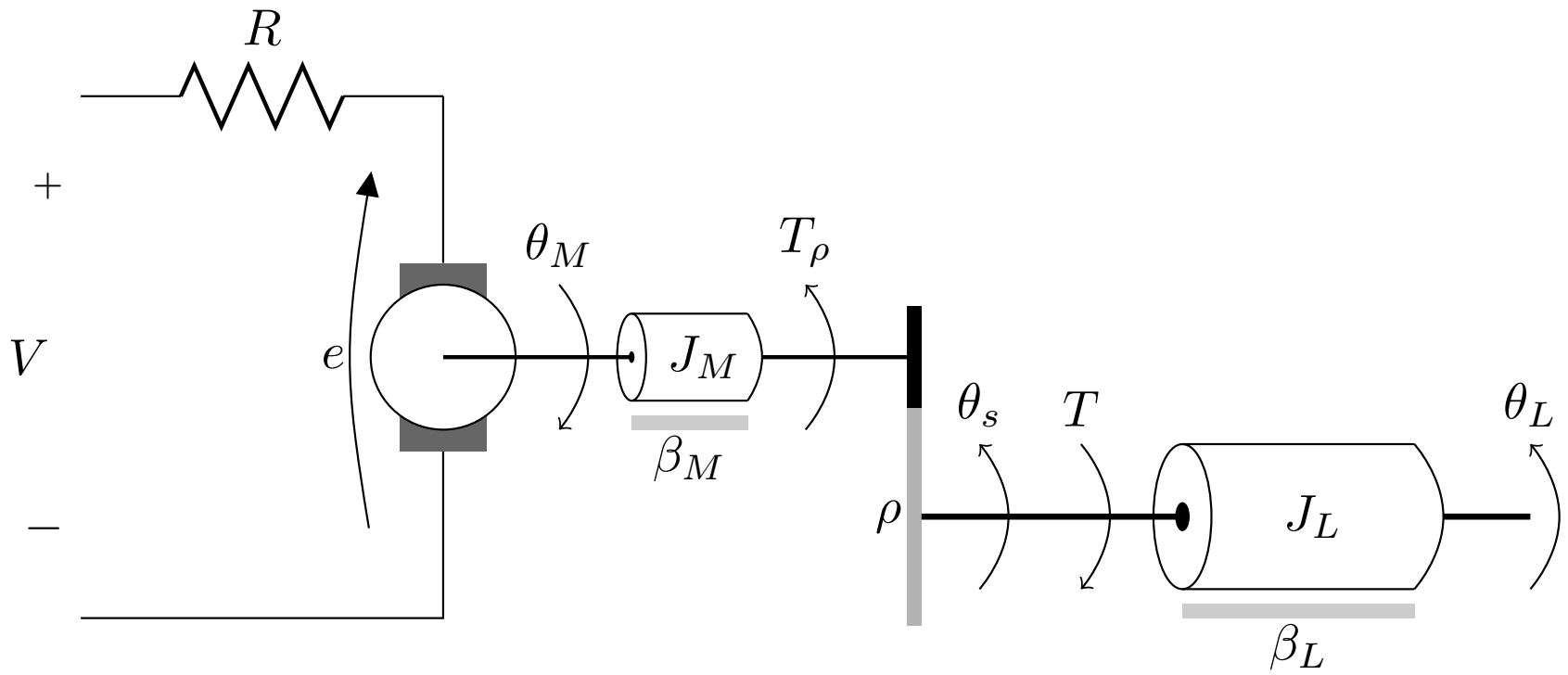
Exploits the **discrete nature** of the system by manipulating the inverter switch positions directly

- ✓ MPC is solved on line, by enumeration
- ✓ very fast with short prediction horizon
- ✗ computational load scales exponentially with the horizon
- ✗ high sampling frequency, not decoupled from control frequency
- ✗ variable frequency



[b] Rodriguez, J. and Kazmierkowski, M.P. and Espinoza, J.R. and Zanchetta, P. and Abu-Rub, H. and Young, H.A and Rojas, C.A. *"State of the Art of Finite Control Set Model Predictive Control in Power Electronics"*, IEEE Transactions on Industrial Informatics, vol.9, n.2, pp.1003-1016, May 2013

DC Servomechanism



V	Applied voltage
R	Armature resistance
T	Load torque
ω_L, ω_M	Load/Shaft angular velocity
k_θ	Torsional rigidity
k_T	Motor constant
J_M, J_L	Motor/Load inertia
ρ	Gear ratio
β_M, β_L	Motor/Load viscous friction

$$\dot{\hat{\omega}}_L = -\frac{k_\theta}{J_L}\left(\theta_L - \frac{\theta_M}{\rho}\right) - \frac{\beta_L}{J_L}\omega_L$$

$$\dot{\hat{\omega}}_M = \left(\frac{V - k_T\omega_M}{R}\right) - \frac{\beta_M\omega_M}{J_M} + \frac{k_\theta}{\rho J_M}\left(\theta_L - \frac{\theta_M}{\rho}\right)$$

DC Servomechanism

Defining the state variables as $x = \begin{bmatrix} \theta_L & \omega_L & \theta_M & \omega_M \end{bmatrix}$ the linear model is:

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

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

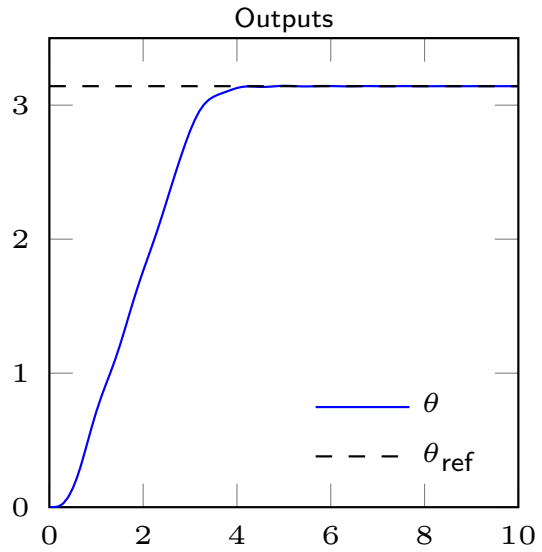
$$T = \begin{bmatrix} k_\theta & 0 & \frac{k_\theta}{\rho} & 0 \end{bmatrix} x$$

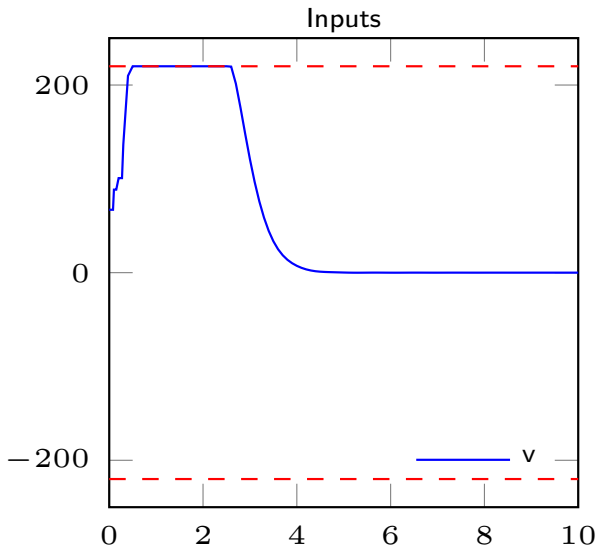
- Inputs:
  - Applied boltage V
- Outputs:
  - Load's angular position
  - Load torque **(not measured!)**

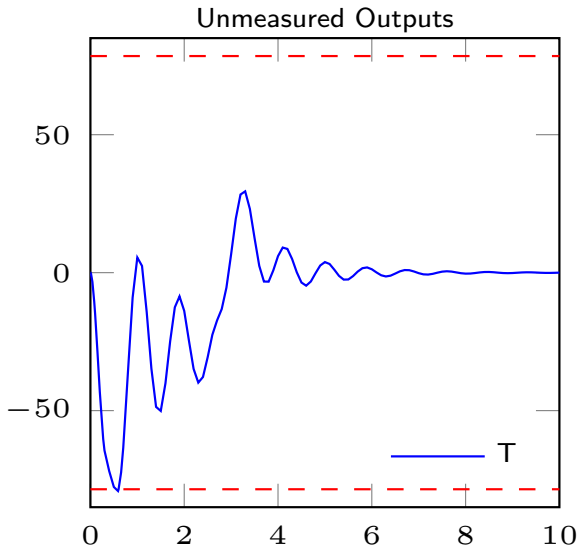
The unmeasured torque must be constrained, due to shaft's finite shear strength

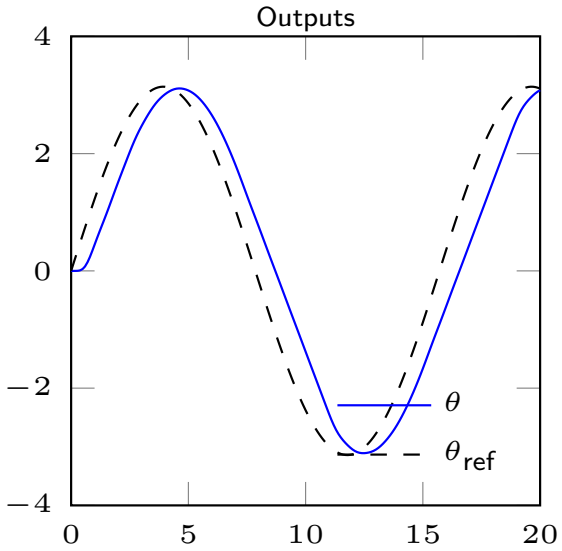
DC Servomechanism

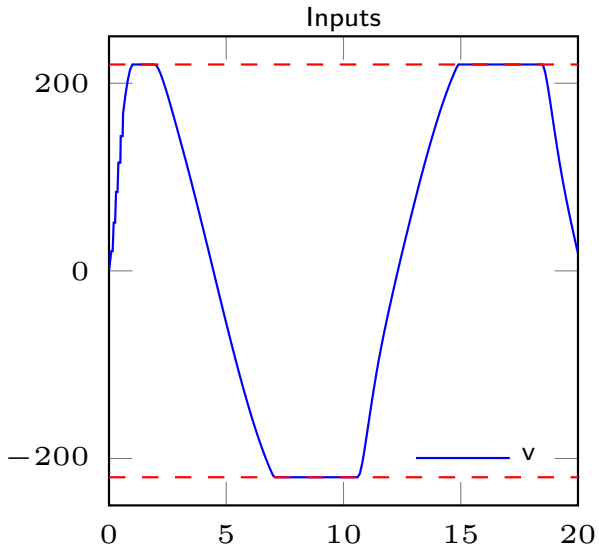
$N_p = 10,\ N_u = 2,\ Q = [10, 0],\ R = 0.05,\ V \in \{-220, 220\},\ T \in \{-78.5398, 78.5398\}$

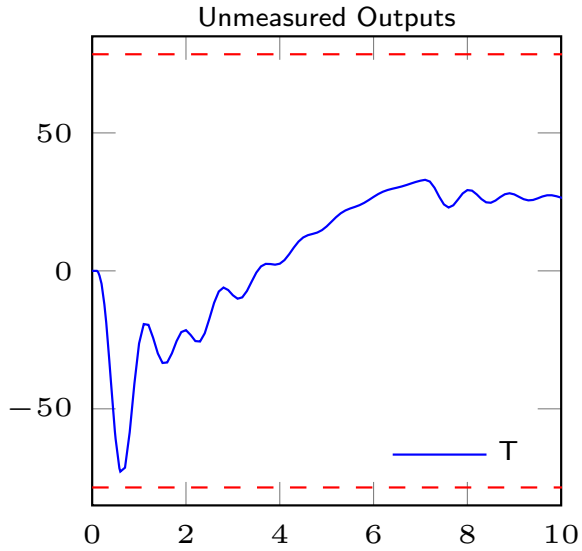












Synchronous Buck Converter

Switching Mode Power Supplies (SMPS) are power converters that regulate the output voltage with the help of

- two semiconductors (a **diode** and a **transistor**)
- storage elements (**capacitors** and **inductors**)

SMPS are mostly employed in those applications requiring strict limits on size and weight together with high performances and efficiency.

The standard categories of SMPS are:

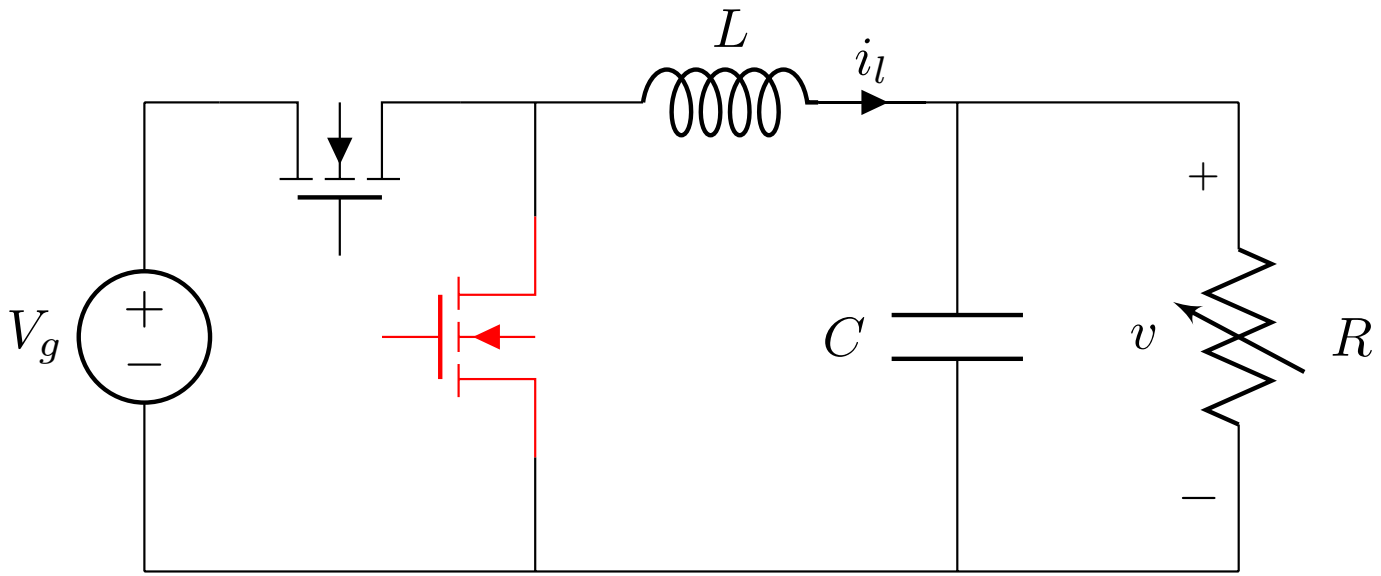
- buck $\rightarrow$ step down the output voltage
- buck $\rightarrow$ step up the output voltage
- buck-boost $\rightarrow$ step down or step up the output voltage

The objective of the control is to regulate the output voltage at a desired level, despite the input voltage and load disturbances

Synchronous Buck Converter

In synchronous converters the free-wheeling diode is replaced by another switch

- the master switch (black), is piloted by the input control
- the auxiliary switch (red), is piloted by the same signal $180°$ out-of-phase (it has the same role of a diode)

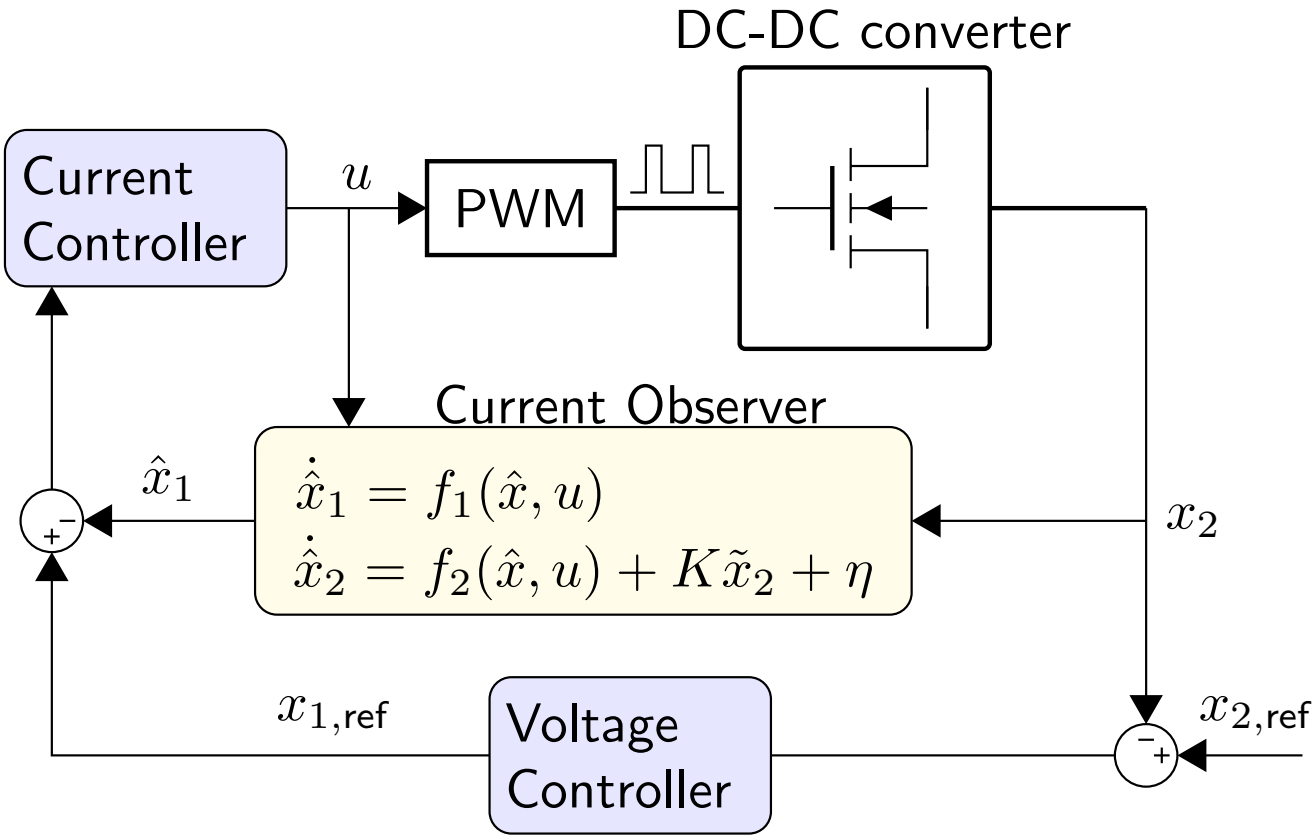


Why synchronous conversion?

- the lower resistance from drain to source helps reducing the voltage
- optimizes the overall power conversion

Synchronous Buck Converter

Sensorless Current Mode Control



The state variables are:

- inductor current x_1
- load voltage x_2

The control input u is the duty-cycle of the PWM which drives the MOSFETs

Synchronous Buck Converter

Sensorless Current Mode Control

Current Mode Control is employed to **improve the performance** of the standard approach (voltage mode control)

An additional current measurement is needed $\rightarrow$ there is the possibility to use an observer, instead of a sensor

Why Sensorless Control?

- eliminates the sensor cost
- avoid possible sensor offset
- decrease the sensitivity to noise
- guarantees the same performance of sensed controller

Synchronous Buck Converter

The mathematical model is obtained trough the Kirchhoff laws:

$$\langle \dot{x}_1(t) \rangle = \frac{u(t)V_g - (R_{DS_{on}} + R_L)\langle x_1(t) \rangle - \langle x_2(t) \rangle}{L}$$

$$\langle \dot{x}_2(t) \rangle = \frac{\langle x_1(t) \rangle - \dfrac{\langle x_2(t) \rangle}{R}}{C}$$

where $\langle \cdot \rangle$ indicates an **averaged variable** over a PWM period.

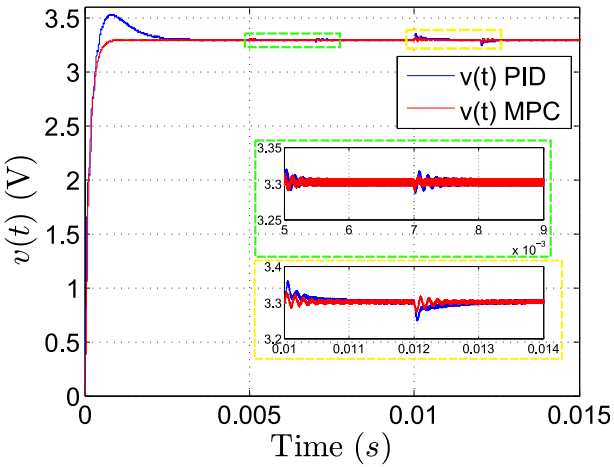
The linear model is given by

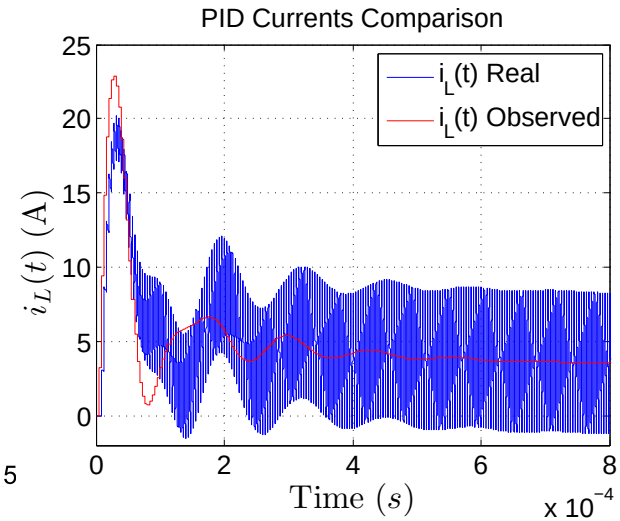
$$\dot{x}(t) = \begin{bmatrix} -\dfrac{R_{DS_{on}} + R_L}{L} & -1 \\ 1 & -\dfrac{1}{R} \end{bmatrix} x(t) + \begin{bmatrix} V_g \\ 0 \end{bmatrix} u(t)$$

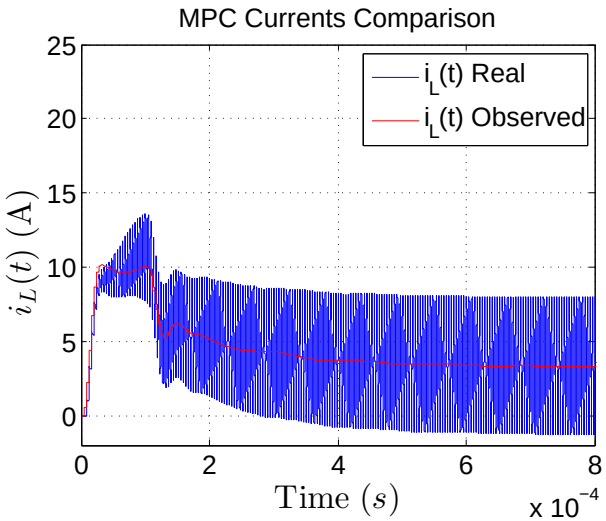
$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} x$$

Synchronous Buck Converter

$N_p = 4,\ N_u = 2,\ Q = 10,\ R = 0.05,\ u \in \{0, 1\},\ x_1 \in \{-\infty, 10\}$







Cimini, G. and Ippoliti, G. and Orlando, G. and Pirro, M. *"Explicit sensorless model predictive control of synchronous buck converter"*, 2013 International Conference on Renewable Energy Research and Applications (ICRERA), pp.1200-1205, Oct 2013