Introduction to Model Predictive Control (MPC)

Oscar Mauricio Agudelo Mañozca

Bart De Moor

ESAT - KU Leuven

May 11th, 2017

Course :

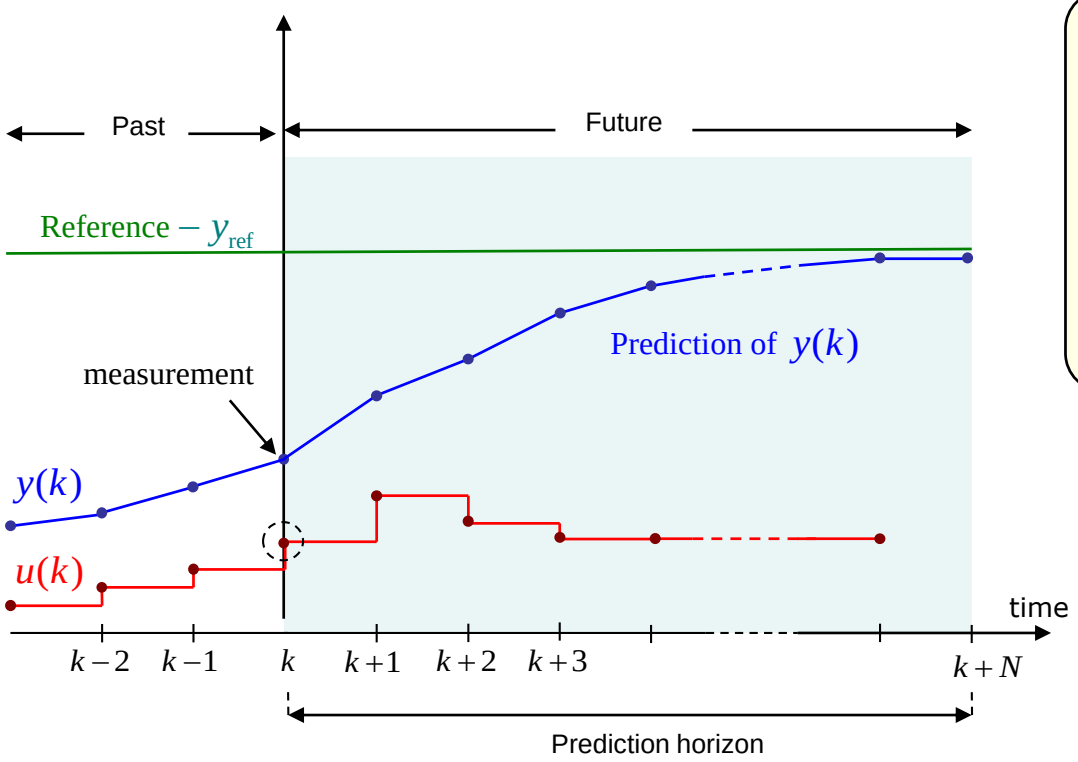
Computergestuurde regeltechniek

ESAT

Basic Concepts

Control method for handling input and state constraints within an optimal control setting.

Principle of predictive control



$$\min_{u(k),\ldots,u(k+N-1)} \sum_{i=1}^{N} \left(y_{\text{ref}} - y(k+i) \right)^2$$

subject to

- model of the process
- input constraints
- output / state constraints

Why to use MPC ?

- It handles multivariable interactions
- It handles input and state constraints
- It can push the plants to their limits of performance.
- It is easy to explain to operators and engineers

Some applications of MPC

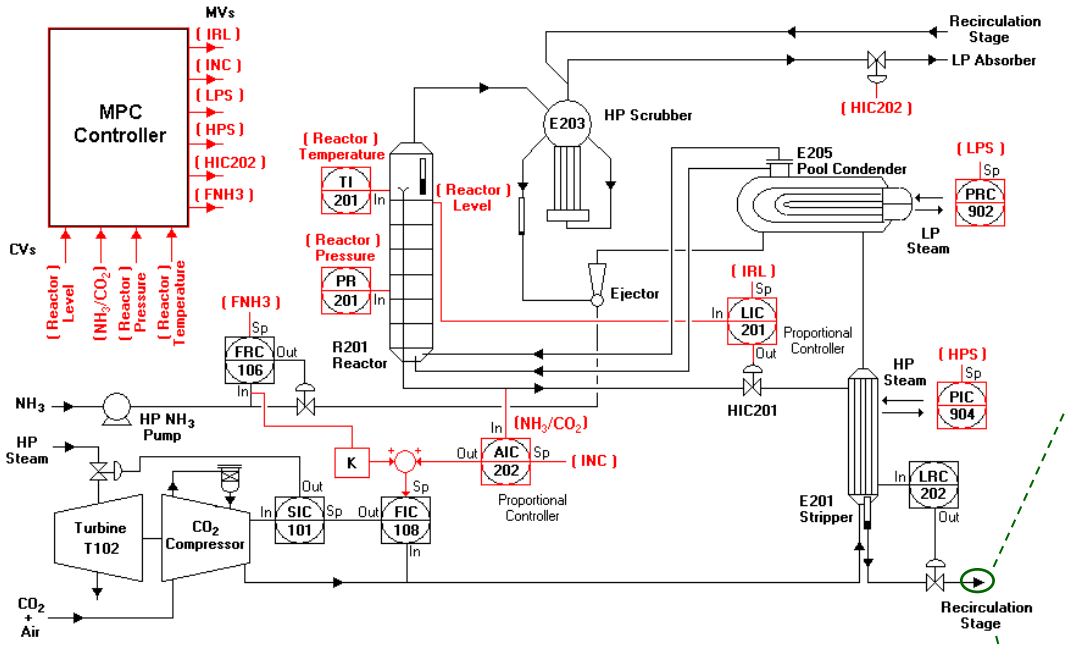
Control of synthesis section of a urea plant

MPC strategies have been used for stabilizing and maximizing the throughput of the synthesis section of a urea plant, while satisfying all the process constraints.

Urea plant of Yara at Brunsbüttel (Germany), where a MPC control system has been set by IPCOS

Some applications of MPC

Control of synthesis section of a urea plant



Reaction 1: Fast and Exothermic

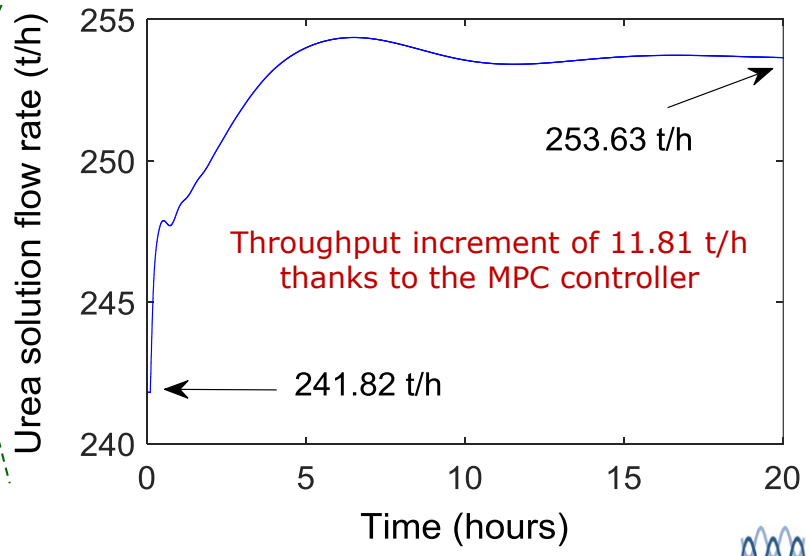
$$2NH_3 + CO_2 \rightarrow NH_2COONH_4$$

Ammonia | Carbon dioxide | Ammonium carbamate

Reaction 2: Slow and Endothermic

$$NH_2COONH_4 \Leftrightarrow NH_2CONH_2 + H_2O$$

Ammonium carbamate | Urea | Water



Results of a preliminary study done by

STADIUS IPCOS Protomation

Some applications of MPC

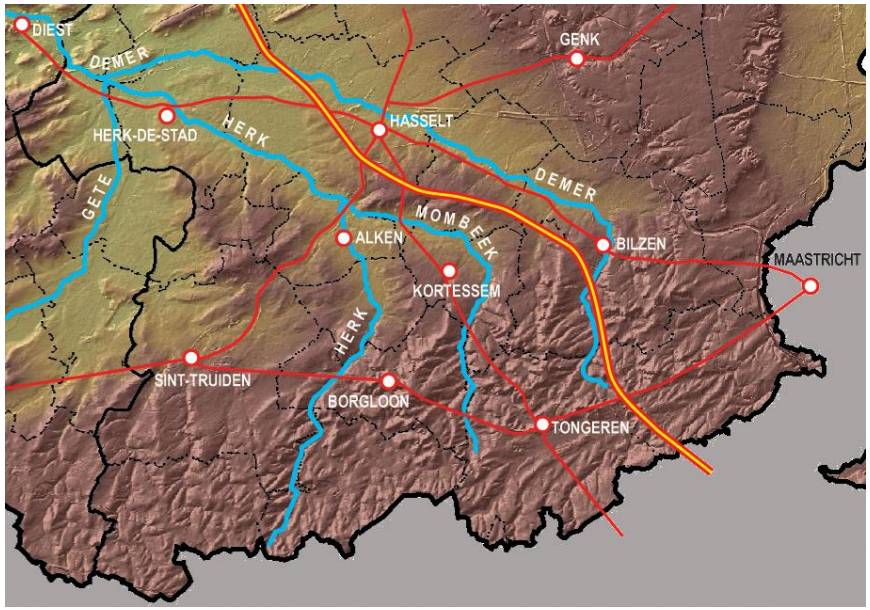
Flood Control: The Demer

A Nonlinear MPC control strategy has been implemented (2016) for avoiding future floodings of the Demer river in Belgium. Partners: STADIUS, Dept. Civil Engineering of KU Leuven, IPCOS, IMDC, Antea Group, and Cofely Fabricom.

The Demer in Hasselt

Flooding events due to heavy rainfall:
1905, 1926, 1965, 1966, 1993-1994, 1995, 1998, 2002 and 2010.

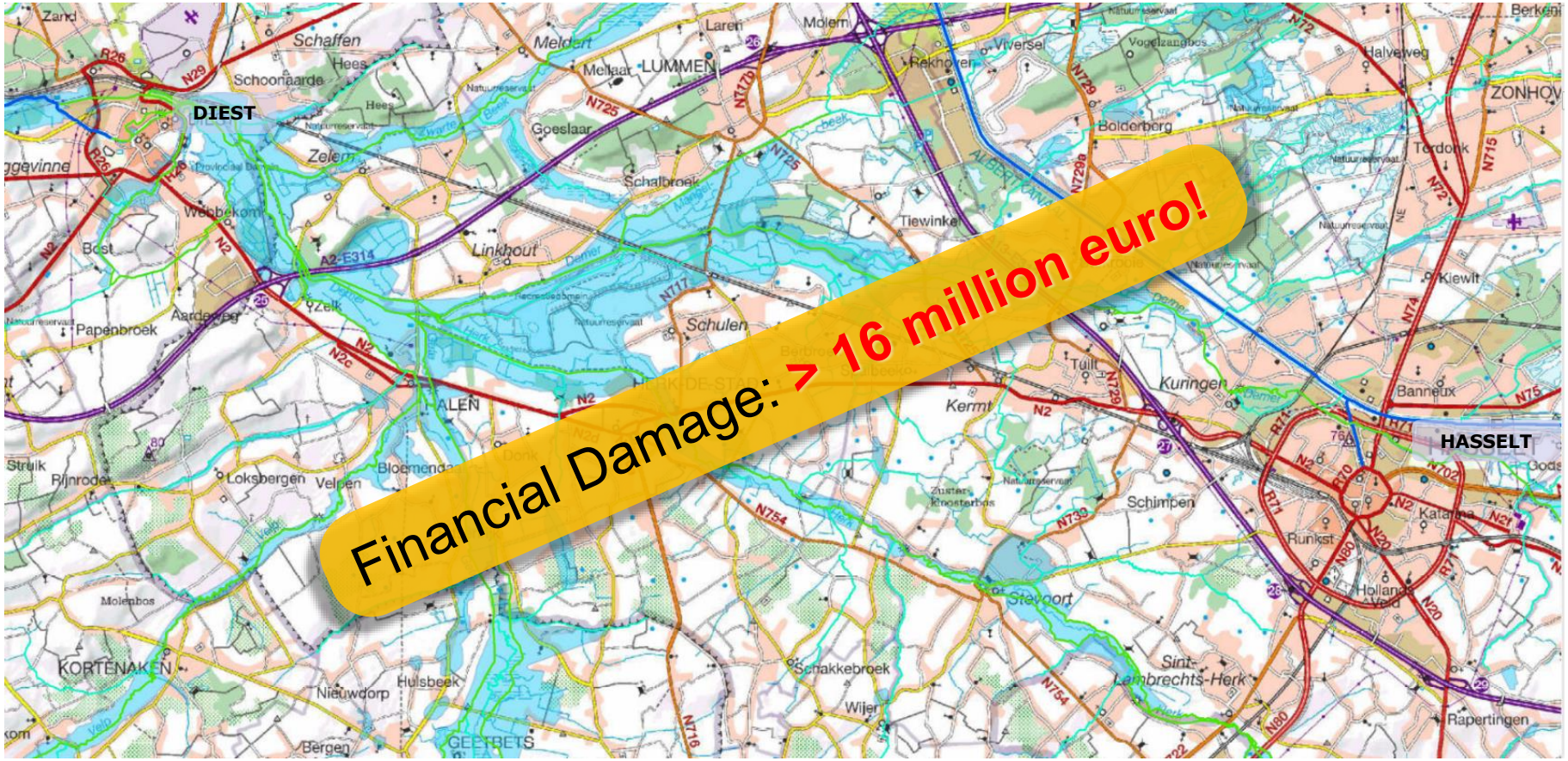
Control Strategy: PLC logic
(e.g., three-pos controller)



The Demer and its tributaries in the south of the province of Limburg

Some applications of MPC

Flood Control: The Demer

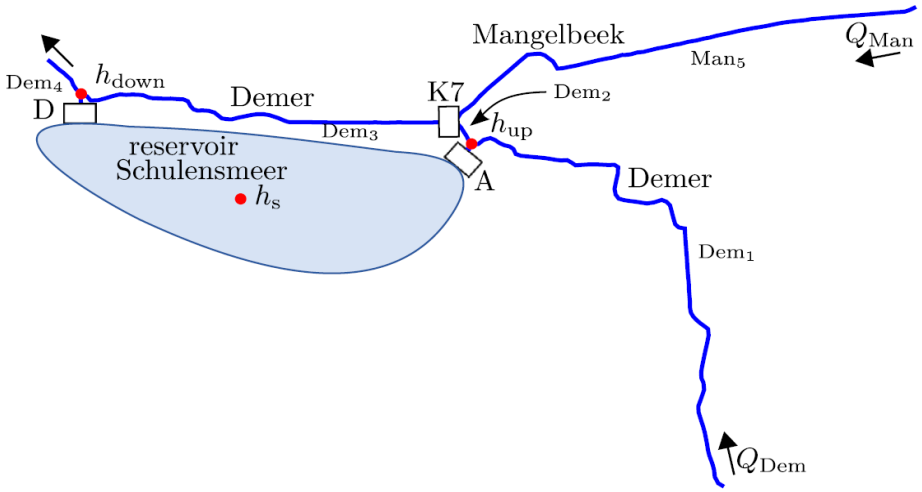


Flooded area during the flood event of 1998.
Control Strategy: PLC logic (e.g., three-position controller)

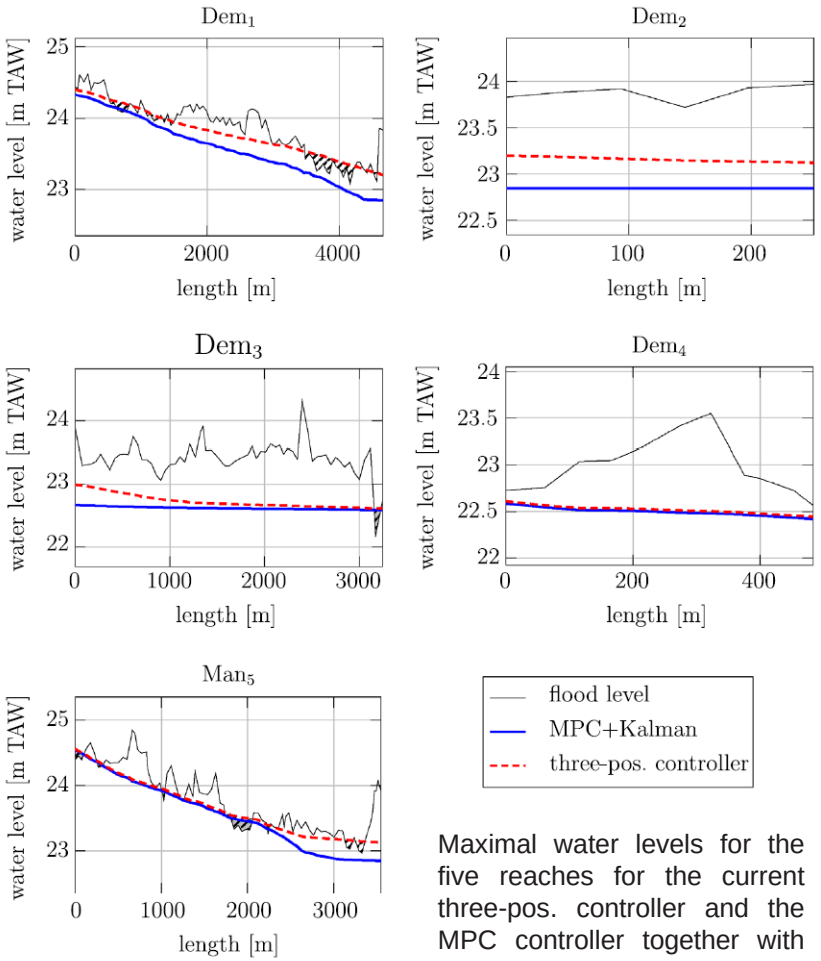
Some applications of MPC

Flood Control: The Demer

Upstream part of the Demer that is modelled and controlled in a preliminary study carried out by STADIUS



Notice: The MPC controller takes rain predictions into account!



Maximal water levels for the five reaches for the current three-pos. controller and the MPC controller together with their flood levels (Flood event 2002) .

Some applications of MPC

In addition MPC, has been used

- in all sort of petrochemical and chemical plants,
- in food processing,
- in automotive industry,
- in the control of tubular chemical reactors,
- in the normalization of the blood glucose level of critical ill patients,
- in power converters,
- for the control of power generating kites under changing wind conditions,
- in mechatronic systems (e.g., mobile robots),
- in power generation,
- in aerospace,
- in HVAC systems (building control)
- …

Basic Concepts

Kinds of MPC

- Linear MPC : it uses a linear model of the plant $\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k)$

  → Convex optimization problem

- Nonlinear MPC: it uses a nonlinear model of the plant $\mathbf{x}(k+1) = \mathbf{f}\left(\mathbf{x}(k), \mathbf{u}(k)\right)$

  → Non-convex optimization problem

Remark: Since linear MPC includes constraints, it is a non-linear control strategy !!!

Linear MPC formulation (Classical MPC)

$$\min_{\mathbf{x}_N, \mathbf{u}_N} \sum_{i=1}^{N} \left(\mathbf{x}(k+i) - \mathbf{x}_{\text{ref}}(k+i)\right)^T \mathbf{Q} \left(\mathbf{x}(k+i) - \mathbf{x}_{\text{ref}}(k+i)\right) + \sum_{i=0}^{N-1} \left(\mathbf{u}(k+i) - \mathbf{u}_{\text{ref}}(k+i)\right)^T \mathbf{R} \left(\mathbf{u}(k+i) - \mathbf{u}_{\text{ref}}(k+i)\right)$$

subject to

$$\mathbf{x}(k+1+i) = \mathbf{A}\mathbf{x}(k+i) + \mathbf{B}\mathbf{u}(k+i), \quad i = 0,1,\ldots,N-1, \quad \rightarrow \text{Model of the plant}$$

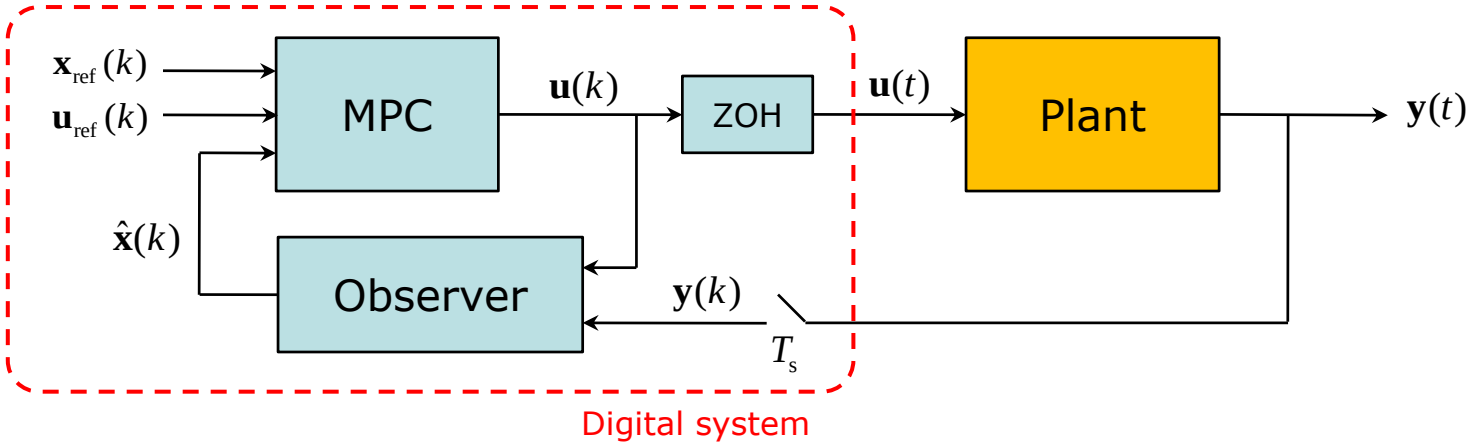
$$\left|\mathbf{u}(k+i)\right| \le \mathbf{u}_{\max}, \quad i = 0,1,\ldots,N-1, \quad \rightarrow \text{Input constraints}$$

$$\left|\mathbf{x}(k+i)\right| \le \mathbf{x}_{\max}, \quad i = 1,2,\ldots,N, \quad \rightarrow \text{State constraints}$$

with: $\mathbf{x}_N = \left[\mathbf{x}(k+1); \mathbf{x}(k+2); \ldots; \mathbf{x}(k+N)\right]$, $\mathbf{u}_N = \left[\mathbf{u}(k); \mathbf{u}(k+1); \ldots; \mathbf{u}(k+N-1)\right]$

MPC Algorithm

Typical MPC control Loop



MPC Algorithm

At every sampling time:

- Read the current state of the process, $\mathbf{x}(k)$
- Compute an optimal control sequence by solving the MPC optimization problem

$$\text{Solution} \rightarrow \mathbf{u}(k), \mathbf{u}(k+1), \mathbf{u}(k+2), \ldots, \mathbf{u}(k+N-1)$$

- Apply to the plant **ONLY** the first element of such a sequence $\rightarrow \mathbf{u}(k)$

LQR and Classical MPC

For simplicity, Let's assume that the references are set to zero.

LQR

$$\min_{\mathbf{x}_\infty, \mathbf{u}_\infty} \sum_{k=1}^{\infty} \mathbf{x}(k)^T \mathbf{Q}\, \mathbf{x}(k) + \mathbf{u}(k)^T \mathbf{R}\, \mathbf{u}(k)$$

subject to

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k), \qquad k = 0,1,\ldots,\infty$$

The optimal solution has the form: $\mathbf{u}(k) = -\mathbf{K}\mathbf{x}(k)$

PRO

- Explicit, Linear solution
- Low online computational burden

CON

- Constraints are not taken into account
- No predictive capacity

CLassical Linear MPC

$$\min_{\mathbf{x}_N, \mathbf{u}_N} \sum_{k=1}^{N} \mathbf{x}(k)^T \mathbf{Q}\, \mathbf{x}(k) + \sum_{k=0}^{N-1} \mathbf{u}(k)^T \mathbf{R}\, \mathbf{u}(k)$$

subject to

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k), \qquad k = 0,1,\ldots,N-1,$$
$$|\mathbf{x}(k)| \le \mathbf{x}_{\max}, \qquad k = 1,2,\ldots,N,$$
$$|\mathbf{u}(k)| \le \mathbf{u}_{\max}, \qquad k = 0,1,\ldots,N-1,$$

PRO

- Takes constraints into account
- Proactive behavior

CON

- High online computational burden
- No explicit solution
- feasibility? Stability?

If $N \to \infty$, and constraints are not considered ➔ the MPC and LQR give the same solution

MPC optimization problem – Implementation details

The following optimization problem,

$$\min_{\mathbf{x}_N,\mathbf{u}_N} \sum_{i=1}^{N} \left(\mathbf{x}(k+i)-\mathbf{x}_{\text{ref}}(k+i)\right)^T \mathbf{Q}\left(\mathbf{x}(k+i)-\mathbf{x}_{\text{ref}}(k+i)\right) + \sum_{i=0}^{N-1} \left(\mathbf{u}(k+i)-\mathbf{u}_{\text{ref}}(k+i)\right)^T \mathbf{R}\left(\mathbf{u}(k+i)-\mathbf{u}_{\text{ref}}(k+i)\right)$$

subject to

$$\begin{aligned} &\mathbf{x}(k+1+i) = \mathbf{A}\mathbf{x}(k+i)+\mathbf{B}\mathbf{u}(k+i), && i=0,1,\ldots,N-1,\\ &\left|\mathbf{u}(k+i)\right| \le \mathbf{u}_{\max}, && i=0,1,\ldots,N-1,\\ &\left|\mathbf{x}(k+i)\right| \le \mathbf{x}_{\max}, && i=1,2,\ldots,N, \end{aligned}$$

with: $\mathbf{x}_N \in \Re^{n_x \cdot N} = \left[\mathbf{x}(k+1);\mathbf{x}(k+2);\ldots;\mathbf{x}(k+N)\right]$, $\mathbf{u}_N \in \Re^{n_u \cdot N} = \left[\mathbf{u}(k);\mathbf{u}(k+1);\ldots;\mathbf{u}(k+N-1)\right]$

can be rewritten as a LCQP (Linearly Constrained Quadratic Program) problem in $\tilde{\mathbf{x}}$ as follows:

$$\min_{\tilde{\mathbf{x}}} \ \frac{1}{2}\tilde{\mathbf{x}}^T \mathbf{H}\tilde{\mathbf{x}} + \mathbf{f}^T\tilde{\mathbf{x}}$$

subject to

$$\begin{aligned} \mathbf{A}_e\tilde{\mathbf{x}} &= \mathbf{b}_e\\ \mathbf{A}_i\tilde{\mathbf{x}} &\le \mathbf{b}_i \end{aligned}$$

with:

$$\tilde{\mathbf{x}} \in \Re^{N\cdot(n_x+n_u)} = \left[\mathbf{x}_N;\mathbf{u}_N\right]$$

MPC optimization problem – Implementation Details

where

$$\mathbf{H} \in \Re^{\left(N\cdot(n_x+n_u)\right)\times\left(N\cdot(n_x+n_u)\right)} = 2\begin{bmatrix} \mathbf{Q} & & & \\ & \ddots & & \\ & & \mathbf{Q} & \\ & & & \mathbf{R} \\ & & & & \ddots \\ & & & & & \mathbf{R} \end{bmatrix}$$

(N times for $\mathbf{Q}$, N times for $\mathbf{R}$)

$$\mathbf{f} \in \Re^{\left(N\cdot(n_x+n_u)\right)} = -\mathbf{H}\begin{bmatrix} \mathbf{x}_{\text{ref}}(1) \\ \vdots \\ \mathbf{x}_{\text{ref}}(N) \\ \mathbf{u}_{\text{ref}}(0) \\ \vdots \\ \mathbf{u}_{\text{ref}}(N-1) \end{bmatrix}$$

$$\mathbf{A}_{\text{i}} \in \Re^{\left(2N\cdot(n_x+n_u)\right)\times\left(N\cdot(n_x+n_u)\right)} = \begin{bmatrix} \mathbf{I}_{n_x\cdot N} & \\ -\mathbf{I}_{n_x\cdot N} & \\ & \mathbf{I}_{n_u\cdot N} \\ & -\mathbf{I}_{n_u\cdot N} \end{bmatrix}$$

$$\mathbf{b}_{\text{i}} \in \Re^{2N\cdot(n_x+n_u)} = \begin{bmatrix} \mathbf{x}_{\max} \\ \vdots \\ \mathbf{x}_{\max} \\ \mathbf{u}_{\max} \\ \vdots \\ \mathbf{u}_{\max} \end{bmatrix}$$

($2N$ times for $\mathbf{x}_{\max}$, $2N$ times for $\mathbf{u}_{\max}$)

n_x = number of states, n_u = number of inputs, N = prediction horizon

MPC optimization problem – Implementation Details

$$\mathbf{A}_e \in \Re^{(N\cdot n_x)\times(N\cdot(n_x+n_u))} = \left[\begin{array}{ccccc|ccc} \mathbf{I}_{n_x} & & & & & -\mathbf{B} & & \\ -\mathbf{A} & \mathbf{I}_{n_x} & & & & & -\mathbf{B} & \\ & \ddots & \ddots & & & & & \ddots \\ & & & -\mathbf{A} & \mathbf{I}_{n_x} & & & -\mathbf{B} \end{array}\right]$$

(N times over the left block; N times over the right block)

$$\mathbf{b}_e \in \Re^{N\cdot n_x} = \begin{bmatrix} \mathbf{A}\mathbf{x}(k) \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}$$

n_x = number of states, n_u = number of inputs, N = prediction horizon

Remarks:

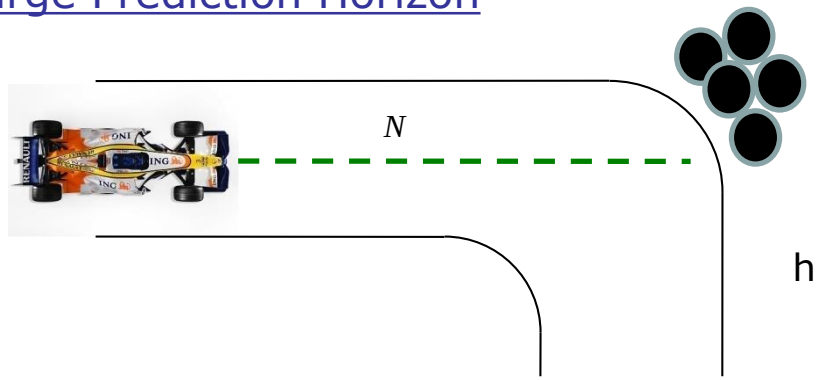
- The problem is convex if $\mathbf{Q}$ and $\mathbf{R}$ are positive semi-definite
- The hessian matrix $\mathbf{H}$, and the matrices $\mathbf{A}_e$ and $\mathbf{A}_i$ are sparse.
- Number of optimization variables: $N(n_x + n_u)$. This number can be reduced to $N\cdot n_u$ (Condensed form of the MPC) through elimination of the states $\mathbf{x}(k)$ but at the cost of sparsity !!!

Matlab function for solving the LCQP optimization problem

```
x_tilde = quadprog (H, f, Ai, bi , Ae, be)
```

MPC with terminal cost

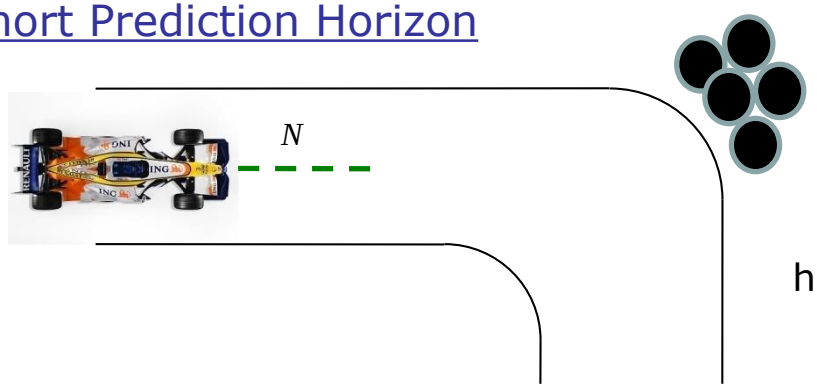
Large Prediction Horizon



What happens?

We have a winner !!!

Short Prediction Horizon



What happens?

We have an accident !!!

N should be large enough in order to keep the process under control

MPC with terminal cost

$$\min_{\mathbf{x}_N,\mathbf{u}_N} \sum_{i=1}^{N-1} \left(\mathbf{x}(k+i)-\mathbf{x}_{\text{ref}}(k+i)\right)^T \mathbf{Q}\left(\mathbf{x}(k+i)-\mathbf{x}_{\text{ref}}(k+i)\right) + \sum_{i=0}^{N-1} \left(\mathbf{u}(k+i)-\mathbf{u}_{\text{ref}}(k+i)\right)^T \mathbf{R}\left(\mathbf{u}(k+i)-\mathbf{u}_{\text{ref}}(k+i)\right)$$

$$+\left(\mathbf{x}(k+N)-\mathbf{x}_{\text{ref}}(k+N)\right)^T \mathbf{S}\left(\mathbf{x}(k+N)-\mathbf{x}_{\text{ref}}(k+N)\right)$$

subject to

$$\mathbf{x}(k+1+i) = \mathbf{A}\mathbf{x}(k+i)+\mathbf{B}\mathbf{u}(k+i), \qquad i=0,1,\ldots,N-1,$$

$$\left|\mathbf{u}(k+i)\right| \le \mathbf{u}_{\max}, \qquad i=0,1,\ldots,N-1,$$

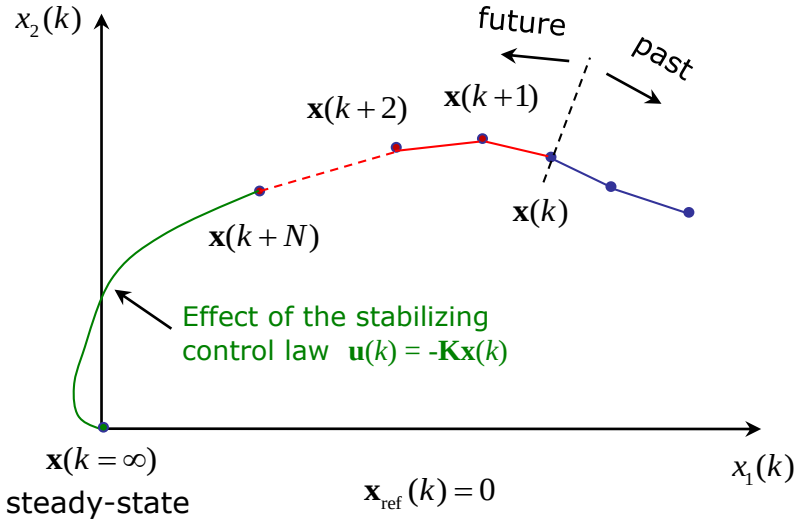
$$\left|\mathbf{x}(k+i)\right| \le \mathbf{x}_{\max}, \qquad i=1,2,\ldots,N,$$

Main goal of the terminal cost:

To include the terms for which $i \ge N$ in the cost function (To extend the prediction horizon to infinity)

What do we gain?

"Stability"



MPC with terminal cost

How to calculate S ?

By solving the discrete-time Riccati equation,

$$\mathbf{A}^T\mathbf{S}\mathbf{A}-\mathbf{S}-(\mathbf{A}^T\mathbf{S}\mathbf{B})(\mathbf{B}^T\mathbf{S}\mathbf{B}+\mathbf{R})^{-1}(\mathbf{B}^T\mathbf{S}\mathbf{A})+\mathbf{Q}=\mathbf{0}$$

where $\mathbf{u}(k) = -\mathbf{K}\mathbf{x}(k)$ is the stabilizing control law. $\mathbf{K}=\left(\mathbf{B}^T\mathbf{S}\mathbf{B}+\mathbf{R}\right)^{-1}\left(\mathbf{B}^T\mathbf{S}\mathbf{A}\right)$.

How to carry out this calculation in Matlab?

```
[K,S] = dlqr (A, B, Q, R )
```

Implementation details of the MPC with terminal constraint

The only change in the LCQP is the hessian matrix

$$\mathbf{H}\in\Re^{\left(N\cdot\left(n_x+n_u\right)\right)\times\left(N\cdot\left(n_x+n_u\right)\right)}=2\begin{bmatrix}\mathbf{Q} & & & & \\ & \ddots & & & \\ & & \mathbf{Q} & & \\ & & & \mathbf{S} & \\ & & & & \mathbf{R} & & \\ & & & & & \ddots & \\ & & & & & & \mathbf{R}\end{bmatrix}$$

(N - 1 times over the $\mathbf{Q}$ block; N times over the $\mathbf{R}$ block)

H0K03a : Advanced Process Control

Model-based Predictive Control 1 : Introduction

Bert Pluymers
Prof. Bart De Moor

Katholieke Universiteit Leuven, Belgium
Faculty of Engineering Sciences
Department of Electrical Engineering (ESAT)
Research Group SCD-SISTA

Overview

- **MPC 1** : Introduction
- **MPC 2** : Dynamic Optimization
- **MPC 3** : Stability
- **MPC 4** : Robustness

- **Industry Speaker** : Christiaan Moons (IPCOS)

(november 3rd)

bert.pluymers@esat.kuleuven.be

Overview

Lesson 1 : Introduction

- Motivating example
- MPC Paradigm
- History
- Mathematical Formulation
- MPC Basics

bert.pluymers@esat.kuleuven.be

Motivating Example

Consider a linear discrete-time state-space model

$$x_{k+1} = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} u_k, \qquad \forall k,$$

called a '*double integrator*'.

We want to design a state feedback controller

$$u_k = \kappa(x_k),$$

that stabilizes the system (i.e. steers it to x=[0; 0]) starting from x=[1; 0], without violating the imposed input constraints

$$|u_k| \leq 0.1, \qquad \forall k.$$

bert.pluymers@esat.kuleuven.be

Motivating Example

Furthermore, we want the controller to lead to a minimal control 'cost' defined as

$$\sum_{k=1}^{\infty} (x_k^T Q x_k) + (u_k^T R u_k),$$

with state and input weighting matrices

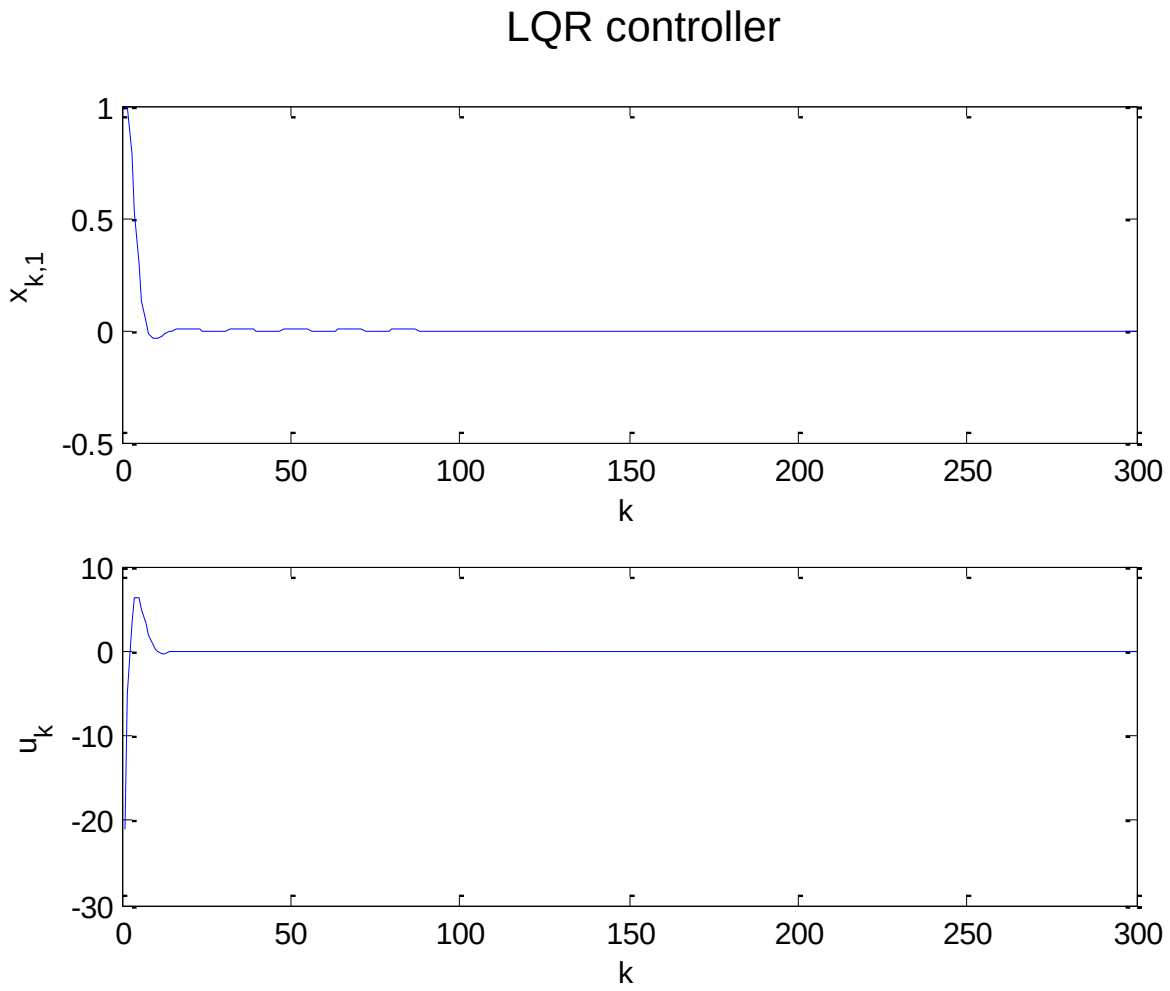
$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \qquad R = 0.01.$$

A straightforward candidate is the LQR controller, which has the form

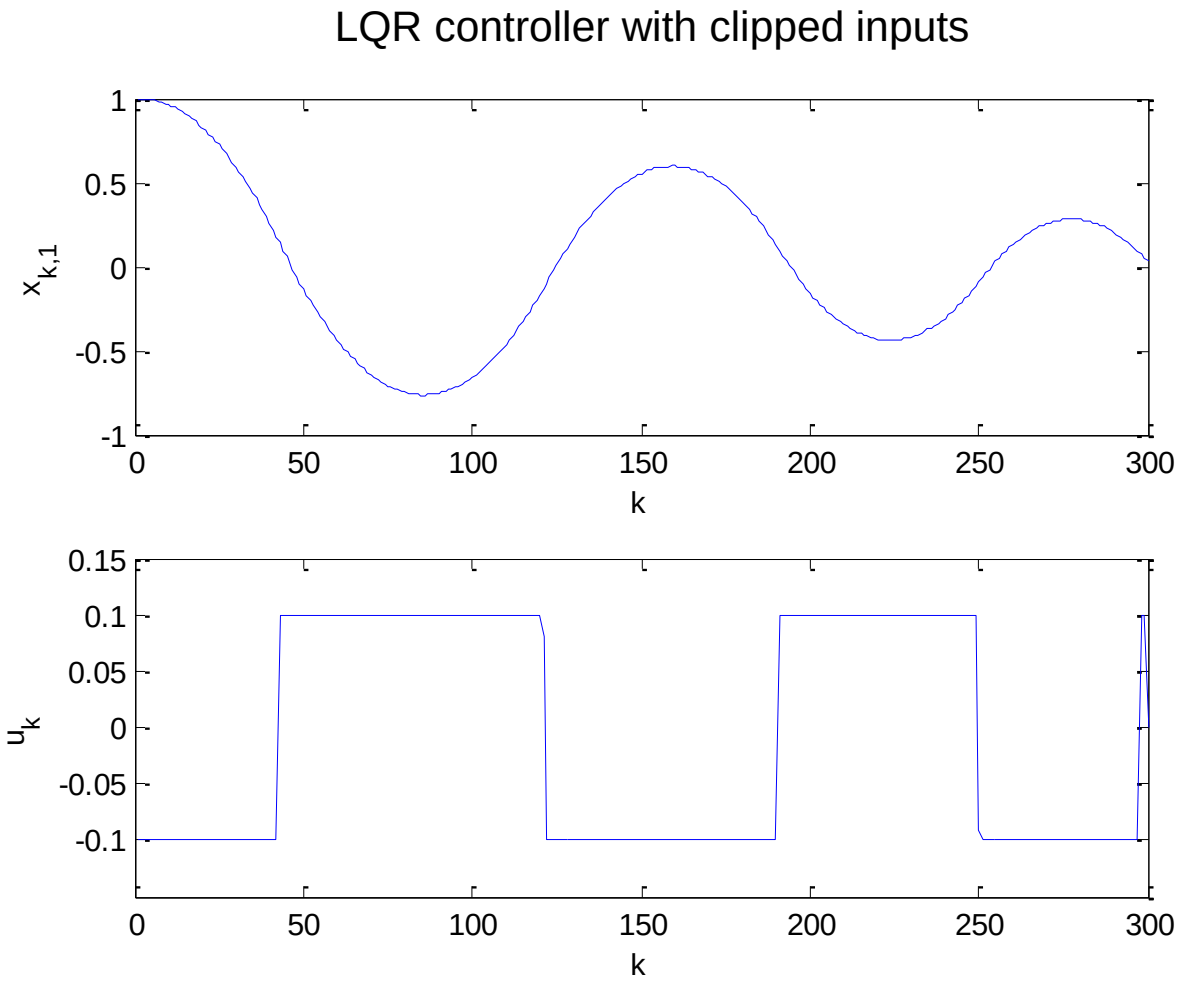
$$u_k = -K x_k.$$

bert.pluymers@esat.kuleuven.be

Motivating Example

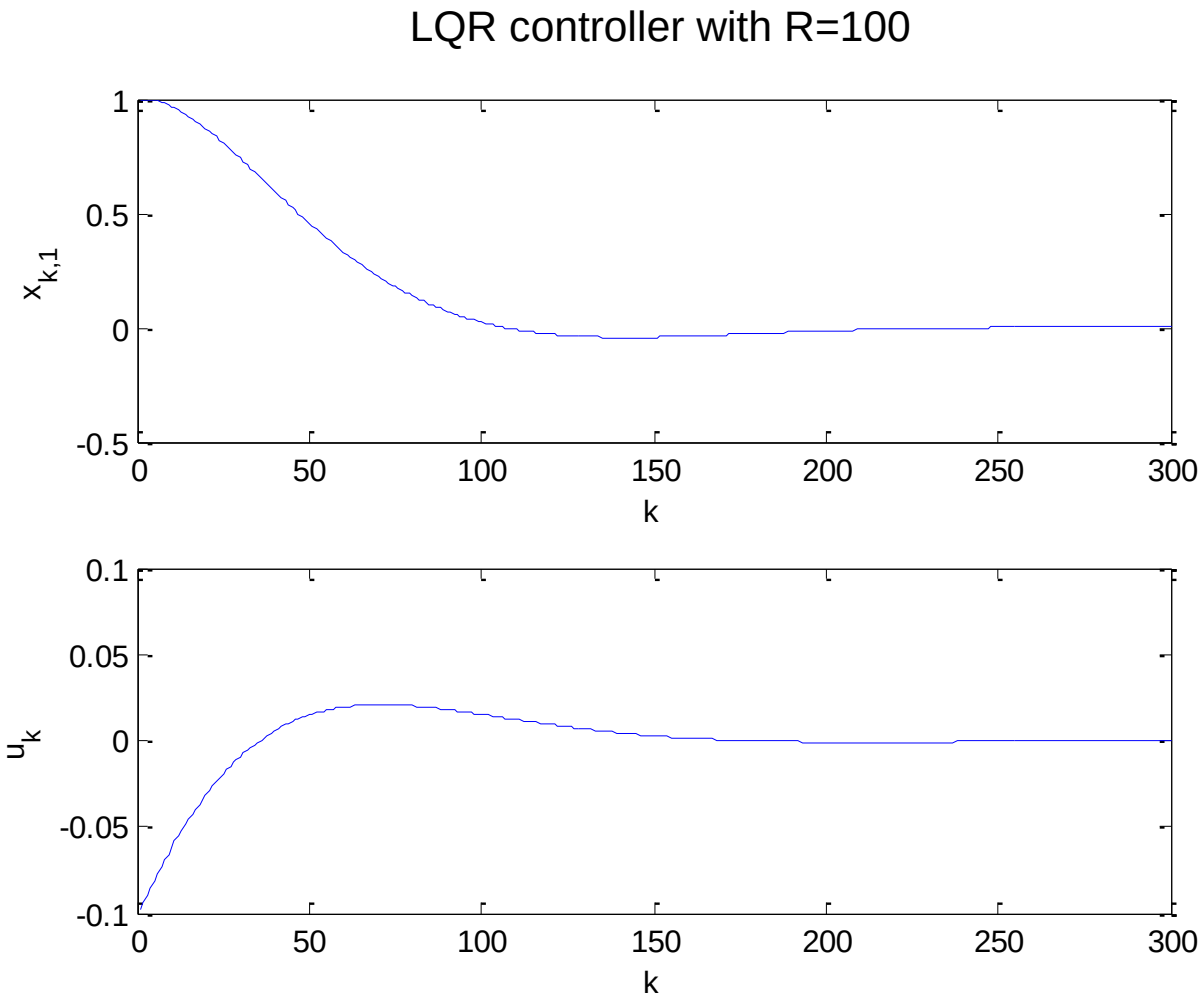


Motivating Example



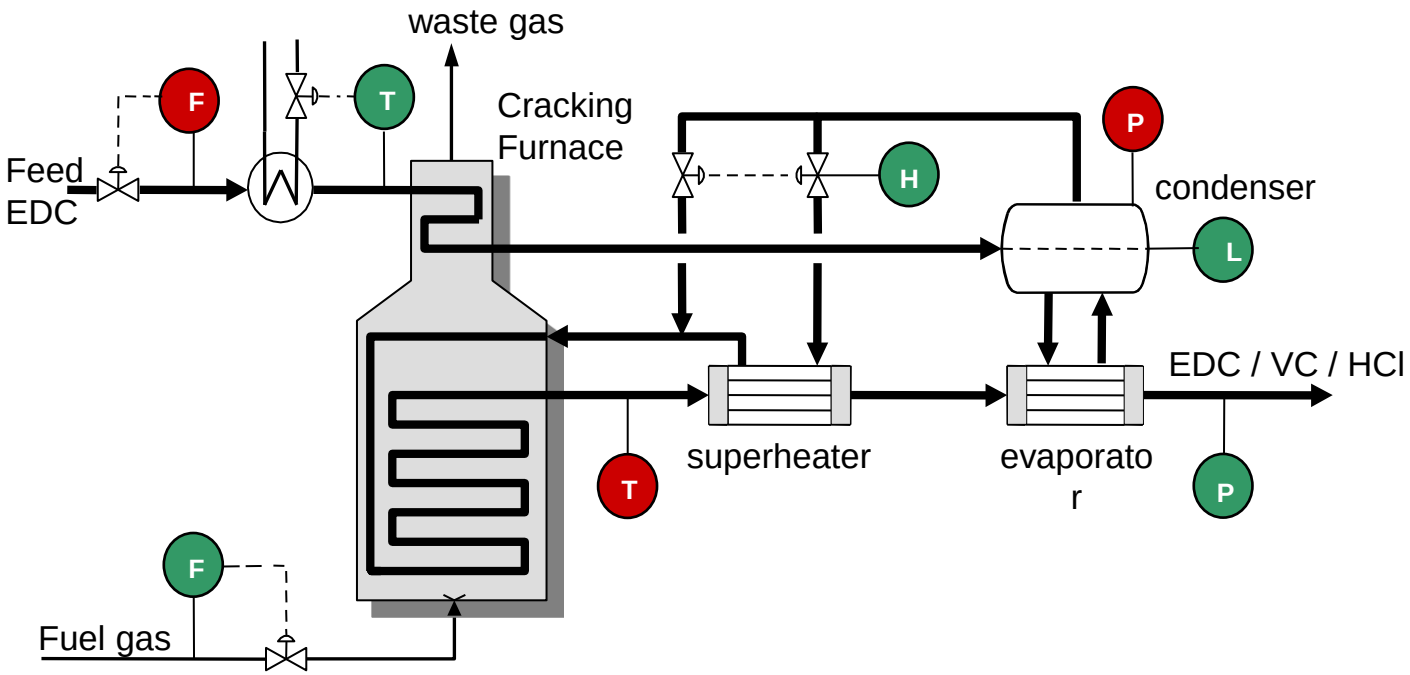
bert.pluymers@esat.kuleuven.be

Motivating Example



bert.pluymers@esat.kuleuven.be

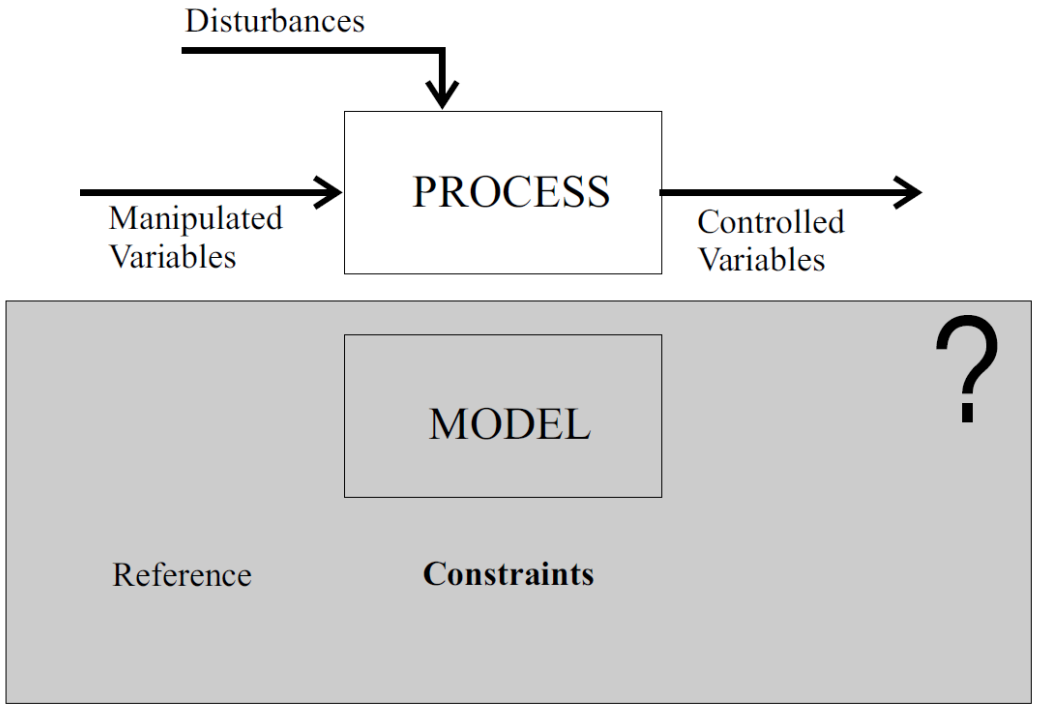
Motivating Example



Systematic way to deal with this issue… ?

bert.pluymers@esat.kuleuven.be

MPC Paradigm

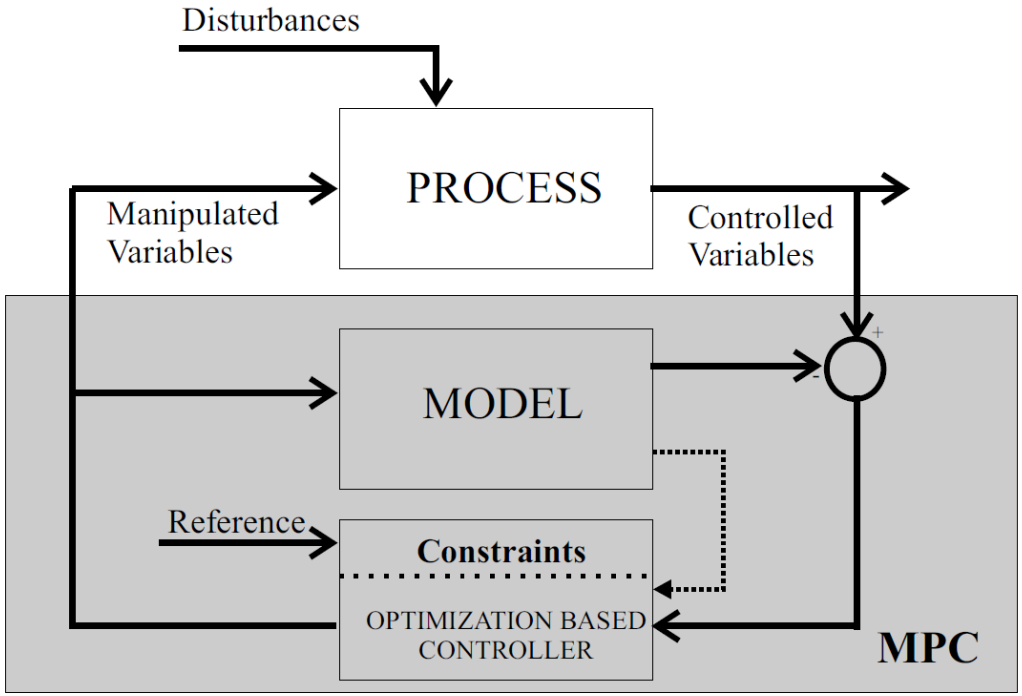


Process industry in ’70s : how to control a process ???

and… easy to understand (i.e. teach) and implement !

MPC Paradigm

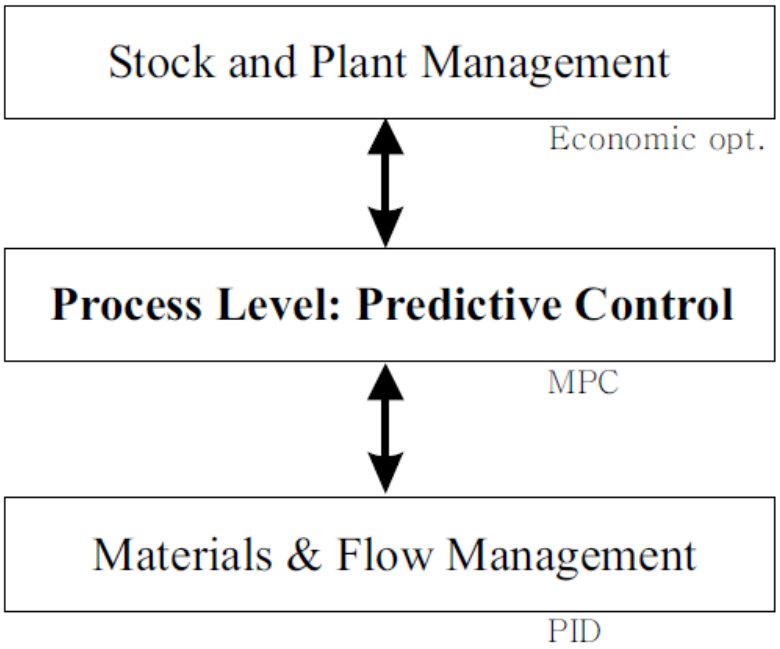
→ Modelbased Predictive Control (MPC)



- **Predictive** : use model to optimize future input sequence
- **Feedback** : incoming measurements used to compensate for inaccuracies in predictions and unmeasured disturbances

MPC Paradigm

MPC has earned its place in the control hierarchy…



- **Econ. Opt.** : optimize profits using market and plant information (~day)
- **MPC** : steer process to desired trajectory (~minute)
- **PID** : control flows, temp., press., … towards MPC setpoints (~second)

History

Before 1960's :

- only input/output models, i.e. transfer functions, FIR models

- Controllers :
  - heuristic (e.g. on/off controllers)
  - PID, lead/lag compensators, …
  - mostly SISO
  - MIMO case : input/output pairing, then SISO control

bert.pluymers@esat.kuleuven.be

History

Early 1960's : Rudolf Kalman

- Introduction of the State Space model :
  - notion of states as 'internal memory' of the system
  - states not always directly measurable : **'Kalman' Filter** !
  - afterwards LQR (as the dual of Kalman filtering)
  - LQG : LQR + Kalman filter

- But LQG no real succes in industry :
  - constraints not taken into account
  - only for linear models
  - only quadratic cost objectives
  - no model uncertainties

bert.pluymers@esat.kuleuven.be

History

During 1960's : 'Receding Horizon' concept

- **Propoi, A. I.** (1963). "*Use of linear programming methods for synthesizing sampled-data automatic systems*". Automatic Remote Control, 24(7), 837–844.

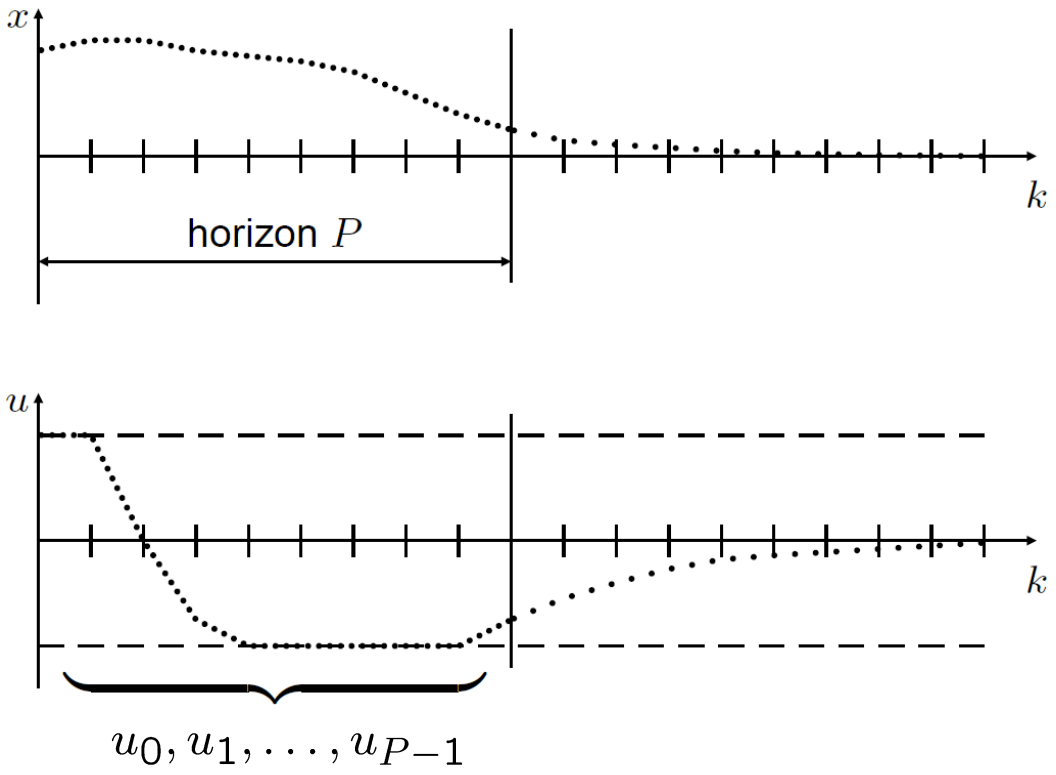
- **Lee, E. B., & Markus, L.** (1967). "*Foundations of optimal control theory*". New York: Wiley. :

> *"… One technique for obtaining a feedback controller synthesis from knowledge of open-loop controllers is to measure the current control process state and then compute very rapidly for the open-loop control function. The first portion of this function is then used during a short time interval, after which a new measurement of the function is computed for this new measurement. The procedure is then repeated. …"*

bert.pluymers@esat.kuleuven.be

History

During 1960's : 'Receding Horizon' concept



History

1970's : 1st generation MPC

- Extension of the LQR / LQG framework through combination with the 'receding horizon' concept
- IDCOM (Richalet et al., 1976) :
  - IR models
  - quadratic objective
  - input / output constraints
  - heuristic solution strategy
- DMC (Shell, 1973) :
  - SR models
  - quadratic objective
  - no constraints
  - solved as least-squares problem

bert.pluymers@esat.kuleuven.be

History

Early 1980's : 2nd generation MPC

- Improve the rather ad-hoc constraint handling of the 1st generation MPC algorithms
- QDMC (Shell, 1983) :
  - SR models
  - quadratic objective
  - linear constraints
  - solved as a quadratic program (QP)

Late 1980's : 3rd generation MPC

- IDCOM-M (Setpoint, 1988), SMOC (Shell, late 80's), …
  - Constraint prioritizing
  - Monitoring / Removal of ill-conditioning
  - fault-tolerance w.r.t. lost signals

bert.pluymers@esat.kuleuven.be

History

Mid 1990's : 4th generation MPC

- DMC-Plus (Honeywell Hi-Spec, '95), RMPCT (Aspen Tech, '96)
  - Graphical user interfaces
  - Explicit control objective hierarchy
  - Estimation of model uncertainty

Currently (in industry) still …

- … no guarantees for stability
- … often approximate optimization methods
- … not all support state-space models
- … no explicit use of model uncertainty in controller design

bert.pluymers@esat.kuleuven.be

MPC Formulation

Basic ingredients :

- **prediction model** to predict plant response to future input sequence
- (finite,) **sliding window** (receding horizon control)
- **parameterization** of future input sequence into finite number of parameters
  - discrete-time : inputs at discrete time steps
  - continuous-time : weighted sum of basis functions
- **optimization** of future input sequence
- **reference** trajectory
- **constraints**

MPC Formulation

First a word on notations

n_x	number of states
n_u	number of inputs
$x_k \in \mathbb{R}^{n_x \times 1}$	state vector at time k
$u_k \in \mathbb{R}^{n_u \times 1}$	input vector at time k
$\mathbf{x}_N \in \mathbb{R}^{(N \cdot n_x) \times 1}$	stacked vector of N future states so $\mathbf{x}_N = [x_1; x_2; \ldots; x_N]$
$\mathbf{u}_N \in \mathbb{R}^{(N \cdot n_u) \times 1}$	stacked vector of N future inputs so $\mathbf{u}_N = [u_0; u_1; \ldots; u_{N-1}]$
$x_{k+1} = f(x_k, u_k)$	system under consideration here
$f_P(\cdot, \cdot)$	plant equation
$f_M(\cdot, \cdot)$	model equation
$J_N(\mathbf{x}_N, \mathbf{u}_N)$	cost function for horizon of N
$\mathbf{x}_N^o$	optimal value of $\mathbf{x}_N$
$\mathbf{u}_N^o$	optimal value of $\mathbf{u}_N$
$J_N^o(x_0)$	optimal value of cost function

bert.pluymers@esat.kuleuven.be

MPC Formulation

Assumptions / simplifications :

- no plant-model mismatch : $f_P \equiv f_M$
- no disturbance inputs
- all states are measured

  (or estimation errors are negligible)
- no sensor noise

. . . seem trivial issues but form essential difficulties in applications . . .

Theoretical Formulation (cfr. CACSD)

given x_0, solve :

$$\left[\begin{array}{lll} \min_{\mathbf{x}_\infty, \mathbf{u}_\infty} & J_\infty(\mathbf{x}_\infty, \mathbf{u}_\infty) & \\ \text{s.t.} & u_k \in \mathcal{U}_k & k = 0, \ldots, \infty \\ & x_k \in \mathcal{X}_k & k = 1, \ldots, \infty \\ & x_{k+1} = f_M(x_k, u_k) & k = 0, \ldots, \infty \end{array}\right.$$

with $\mathcal{U}_k \subset \mathbb{R}^{n_u}$ and $\mathcal{X}_k \subset \mathbb{R}^{n_x}$ convex

- future window of length ∞
- impossible to solve, because . . .
  - infinite number of optimization variables
  - infinite number of inequality constraints
  - infinite number of equality constraints

bert.pluymers@esat.kuleuven.be

Formulation 1

given x_0, solve :

$$\left[\begin{array}{lll} \min_{\mathbf{x}_\infty, \mathbf{u}_\infty} & J_\infty(\mathbf{x}_\infty, \mathbf{u}_\infty) & \\ \text{s.t.} & u_k \in \mathcal{U}_k & k = 0, \ldots, \infty \\ & x_k \in \mathcal{X}_k & k = 1, \ldots, \infty \\ & x_{k+1} = f_M(x_k, u_k) & k = 0, \ldots, \infty \end{array}\right.$$

with $\mathcal{U}_k \subset \mathbb{R}^{n_u}$ and $\mathcal{X}_k \subset \mathbb{R}^{n_x}$ convex

- still future window of length ∞, BUT
  - quadratic cost function
  - no input or state constraints
  - linear model
- Optimal solution has the form $u_k = -Kx_k$
- Find K by solving Ricatti equation → **LQR controller**

bert.pluymers@esat.kuleuven.be

Formulation 1 : LQR

PRO

- explicit, linear solution
- low online computational complexity

CON

- constraints not taken into account
- linear model assumption
- only quadratic cost functions
- no predictive capacity

Formulation 2

given x_0, solve :

$$\left[\begin{array}{lll} \min_{\mathbf{x}_N, \mathbf{u}_N} & J_N(\mathbf{x}_N, \mathbf{u}_N) & \\ \text{s.t.} & u_k \in \mathcal{U}_k & k = 0, \ldots, N-1 \\ & x_k \in \mathcal{X}_k & k = 1, \ldots, N \\ & x_{k+1} = f_M(x_k, u_k) & k = 0, \ldots, N-1 \end{array}\right.$$

with $\mathcal{U}_k \subseteq \mathbb{R}^{n_u}$ and $\mathcal{X}_k \subseteq \mathbb{R}^{n_x}$ convex

- changes :
  - keep all constraints etc.
  - reduce horizon to length N
- solution obtainable through dynamic optimization
- only u_0 is applied, in order to obtain feedback at each k about x_k

bert.pluymers@esat.kuleuven.be

Formulation 2 : Classic MPC

PRO

- takes constraints into account
- proactive behaviour
- wide range of (convex) cost functions possible
- also for nonlinear models (but convex ?)

CON

- high online computational complexity
- no explicit solution
- feasibility ?
- stability ?
- robustness ?

- **In what follows, we will concentrate on this formulation.**

bert.pluymers@esat.kuleuven.be

Formulation 3

given x_0, solve :

$$\left[\begin{array}{lll} \min_K & J_\infty(\mathbf{x}_\infty, \mathbf{u}_\infty) = \sum_{k=0}^{\infty} \left(x_k^T Q x_k + u_k^T R u_k\right) & \\ \text{s.t.} & u_k \in \mathcal{U} & k = 0, \ldots, \infty \\ & x_k \in \mathcal{X} & k = 1, \ldots, \infty \\ & x_{k+1} = Ax_k + Bu_k & k = 0, \ldots, \infty \\ & u_k = Kx_k & k = 0, \ldots, \infty \end{array}\right.$$

with $\mathcal{U} \subseteq \mathbb{R}^{n_u}$ and $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ convex and with piece-wise linear boundaries.

- linear form of feedback law is enforced
- problem can be recast as a convex (LMI-based) optimization problem

more on this later . . .

---

bert.pluymers@esat.kuleuven.be

Summary

Formulation 1

- infinite horizon
- no constraints
- explicit solution
- → LQR

Formulation 2

- finite horizon
- constraints
- no explicit solution
- → Classic MPC

Formulation 3

- infinite horizon
- constraints
- explicit solution enforced
- → has elements of LQR ánd MPC

bert.pluymers@esat.kuleuven.be

Open vs. closed loop control

- **Open loop** : no state/output feedback : feedforward control
- **Closed loop** : state/output feedback : e.g. LQR

MPC is a mix of both :

- internally optimizing an **open loop** finite horizon control problem
- but at each k there is state feedback to compensate unmodelled dynamics and disturbance inputs → **closed loop** control paradigm.
- has implications on e.g stability analysis
- Is of essential importance in **Robust MPC,** more on this later…

bert.pluymers@esat.kuleuven.be

Standard MPC Algorithm

1. Assume current time = 0
2. Measure or estimate x_0 and solve for u_N and x_N :

$$\begin{aligned}
&\min_{\mathbf{x}_N,\mathbf{u}_N} \quad J_N(\mathbf{x}_N,\mathbf{u}_N) = \sum_{k=0}^{N-1} l(x_k, r_k, u_k) \\
&\qquad\qquad\qquad\qquad\qquad\qquad + F(x_N) \\
&\text{s.t.} \quad u_k \in \mathcal{U} \qquad\qquad k = 0,\ldots,N-1 \\
&\qquad\quad x_k \in \mathcal{X} \qquad\qquad k = 1,\ldots,N-1 \\
&\qquad\quad x_N \in \mathcal{X}_N \\
&\qquad\quad x_{k+1} = f_M(x_k, u_k) \quad k = 0,\ldots,N-1
\end{aligned}$$

3. Apply u^o_0 and go to step 1

Remarks :

- $F, \mathcal{X}_N$: Terminal state cost and constraint
- l : some kind of norm function
- Sliding Window

MPC design choices

1. prediction model

2. cost function

- norm $l(\cdot,\cdot,\cdot)$
- horizon N
- terminal state cost

3. constraints

- typical input/state constraints
- terminal constraint

Prediction Model

- **Input/Output or State-Space ?**
  - I/O restricted to stable, linear plants
  - Hence SS-models

- **Type of model determines class of MPC algorithm**
  - Linear model : Linear MPC
  - Non-linear model : Non-linear MPC (or NMPC)
  - Linear model with uncertainties : Robust MPC
  - **BUT** : MPC is always a non-linear feedback law due to the constraints

- **Type of model determines class of involved optimization problem**
  - Linear models lead to most efficiently solvable opt.-problems
  - Choose simplest model that fits the real plant ‘sufficiently well’

bert.pluymers@esat.kuleuven.be

Cost Function

General Cost Function:

$$J_N(\mathbf{x}_N, \mathbf{u}_N) = \sum_{k=0}^{N-1} l(x_k, r_k, u_k) + F(x_N)$$

Design Functions and Parameters:

1. $l(x, r, u)$
2. Horizon N
3. *F*(x)
4. Reference Trajectory

Cost Function

1. $\ell(x, r, u)$

- Typically sum of norm in $x - r$ and norm in u
- May be time depending (i.e. ℓ_k) (but usually isn't)
- **Most common:**
  - Quadratic: $(x_k - r_k)^T Q_k (x_k - r_k) + u_k^T R_k u_k$
  - Linear: $w_x |x_k - r_k| + w_u |u_k|$
  - Non-linear: e.g. grade change
  - Also common : J in terms of Δu_k

Cost Function

2. N

- **N increases: more control variables**
  - ▷ General rule: N big enough w.r.t. dynamics
    - ▷ for FIR, $N >$ order
    - ▷ for SS, $N >$ slowest mode
  - ▷ computational complexity !
- **Intuitively: better for stability :**
  - ▷ $N \to \infty \Rightarrow J_N \to J_\infty = J_{LQR}$ (for x_0 close to origin)
- **Most general case: 3 different horizons :**
  - ▷ Control Horizon N
  - ▷ Prediction Horizon P
  - ▷ Delay Horizon D
  - ▷ $J = \sum_D^P x_k^T Q x_k + \sum_0^{N-1} u_k^T R u_k$
  - ▷ most common : $D = 0, P = N$

bert.pluymers@esat.kuleuven.be

Cost Function

3. $F(x_N)$

- Called **Terminal State Cost**

- Main Goal:
  - ▷ 'include' $k \geq N$
  - ▷ $J_N \rightarrow J_\infty$
  - ▷ $\rightarrow$ Stability

- Comparable to $P > N$

  Example: Suppose $u_k \equiv 0, \quad k \geq N$

  More on this later (stability)

Constraints

General Constraint form:

- Input constraints: $u_k \in U_k, \quad k = 0, \ldots, N-1$
- State constraints: $x_k \in X_k, \quad k = 1, \ldots, N$

Remarks:

- U_k, X_k convex !
- Combinations can also occur, e.g. output constraints

Constraints

Typically:

- Input bounds: $|u_k| < u_b$
- Bounds on states: $|x_k| < x_b$
- Output bounds: $|y_k| < y_b$

  $\rightarrow |Cx_k + Du_k| < y_b$
- Rate of Change Constraints (ROC): $|\Delta u_k| < \Delta u_b$
- Terminal State Constraint: $x_N < x_{Nb}$

Notice:

- all **linear** constraints
- all **hard** constraints

  $\rightarrow$ Infeasibility may occur

Constraints

Soft Constraints

Introduced to overcome infeasibility

Method: Penalise in cost

$$x_k < x_b$$
$$\downarrow$$
$$x_k - \epsilon_{x_k} < x_b, \quad \epsilon_{x_k} \geq 0$$
$$\downarrow$$
$$J^{sc} = J + \rho_k \ell_\epsilon(\epsilon_k)$$

Constraints may be violated

Choose 1-norm or ∞-norm for ℓ_ϵ

- Used for non-important constraints
- Used in priority ranking

bert.pluymers@esat.kuleuven.be

Constraints

Priority Ranking

Principle: Hard constraints can be added until there is no degree of freedom left. All other constraints are imposed as soft constraints.

$\longrightarrow$ Method:

Use soft constraints with **exact penalty function** and use weights ρ to create a ranking

Terminal State Constraints

Goal: comparable to terminal state cost:

Bring the final state x_N to a value s.t. stability is guaranteed

More on this later , but intuitive example:

Choose X_N or x_{Nb}, in combination with $F(x_N)$, such that state constraints can't be violated for $k > N$

Reference Insertion

Problem :

- previous sections all assumed $x_{\text{ref}} = 0, u_{\text{ref}} = 0$
- in reality $x_{\text{ref}} \neq 0, u_{\text{ref}} \neq 0$ and possibly time-varying
- replacing x with $x - x_{\text{ref}}$ in all formulas not sufficient !!!

Solutions :

- static reference insertion
- dynamic reference insertion (*trajectory optimization*)

Reference Insertion

Static Reference Insertion

Method

- consider reference states $x_{k,\text{ref}}$ individually for each k (hence static)
- calculate $u_{k,\text{ref}}$ such that

$$x_{k,\text{ref}} = Ax_{k,\text{ref}} + Bu_{k,\text{ref}}$$

Properties

- similar to method using N_x, N_u discussed in CACSD
- numerically easy
- does not necessarily lead to a consistent trajectory (i.e. $x_{k+1,\text{ref}} \neq Ax_{k,\text{ref}} + Bu_{k,\text{ref}}$)

bert.pluymers@esat.kuleuven.be

Reference Insertion

Dynamic Reference Insertion

Method

- consider an entire window of reference states $x_{k,\text{ref}}, k = 0, \ldots, N'$, with typically $N' >> N$
- calculate $u_{k,\text{ref}}$ and $x'_{k,\text{ref}}$ as

$$\begin{bmatrix} \min_{\mathbf{x}'_{\text{ref}}, \mathbf{u}_{\text{ref}}} \ \text{cost}(\mathbf{x}_{\text{ref}}, \mathbf{x}'_{\text{ref}}, \mathbf{u}_{\text{ref}}) \\ \text{s.t.} \ \ x'_{k+1,\text{ref}} = Ax'_{k,\text{ref}} + Bu_{k,\text{ref}}, \ \ k = 0, \ldots, N' - 1 \end{bmatrix}$$

Properties

- numerically very demanding, but . . .
- . . . performed off-line, can hence be extended with non-linear cost function and non-linear plant model
- leads to consistent trajectories
  - improved control performance
  - ideal to combine with MPC using local linearization (= *delta-mode MPC*)

bert.pluymers@esat.kuleuven.be

Reference Insertion

Dynamic Reference Insertion

Method

- consider an entire window of reference states $x_{k,\text{ref}}, k = 0, \ldots, N'$, with typically $N' >> N$
- calculate $u_{k,\text{ref}}$ and $x'_{k,\text{ref}}$ as

$$\begin{bmatrix} \min_{\mathbf{x}'_{\text{ref}}, \mathbf{u}_{\text{ref}}} \ \text{cost}(\mathbf{x}_{\text{ref}}, \mathbf{x}'_{\text{ref}}, \mathbf{u}_{\text{ref}}) \\ \text{s.t.} \ \ x'_{k+1,\text{ref}} = Ax'_{k,\text{ref}} + Bu_{k,\text{ref}}, \ \ k = 0, \ldots, N' - 1 \end{bmatrix}$$

Properties

- numerically very demanding, but . . .
- . . . performed off-line, can hence be extended with non-linear cost function and non-linear plant model
- leads to consistent trajectories
  - improved control performance
  - ideal to combine with MPC using local linearization (= *delta-mode MPC*)

bert.pluymers@esat.kuleuven.be

Exercise Sessions

- **Ex. 1** : Optimization oriented
- **Ex. 2** : MPC oriented
- **Ex. 3** : real-life MPC/optimization problem

Evaluation

- (brief !) report (groups of 2)
- oral examination → insight !

References

[1] J. Maciejowski. *Predictive Control with Constraints*. Pearson Education (Prentice-Hall imprint), 2001.

[2] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

[3] S.J. Qin and T.A. Badgwell. An overview of industrial model predictive control technology. *AIChE Symposium Series 316*, 93:232–256, 1996.

[4] J. Richalet. Industrial applications of model based predictive control. *Automatica*, 29(5):1251–1274, 1993.

[5] J. A. Rossiter. *Model-Based Predictive Control : A Practical Approach*. CRC Press, 2004.

H0K03a : Advanced Process Control

Model-based Predictive Control 2 : Dynamic Optimization

Bert Pluymers
Prof. Bart De Moor

Katholieke Universiteit Leuven, Belgium
Faculty of Engineering Sciences
Department of Electrical Engineering (ESAT)
Research Group SCD-SISTA

Overview

Lesson 2 : Dynamic Optimization

- Optimization basics
- Convex Optimization
- Dynamic Optimization
- Optimization Algorithms

bert.pluymers@esat.kuleuven.be

Notation

General form :

$$\begin{bmatrix} \min_x & f(x) \\ \text{s.t.} & h(x) = 0 \\ & g(x) \leq 0 \end{bmatrix}$$

Legend :

- $x \in \mathbb{R}^n$: vector of optimization variables
- $f : \mathbb{R}^n \rightarrow \mathbb{R}$: objective function / cost function
- $h : \mathbb{R}^n \rightarrow \mathbb{R}^l$: equality constraints
- $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$: inequality constraints
- $x^\star$: solution to optimization problem
- $f^\star \equiv f(x^\star)$: optimal function value

bert.pluymers@esat.kuleuven.be

Gradient & Hessian

Gradient :

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

(points in direction of steepest ascent)

Hessian :

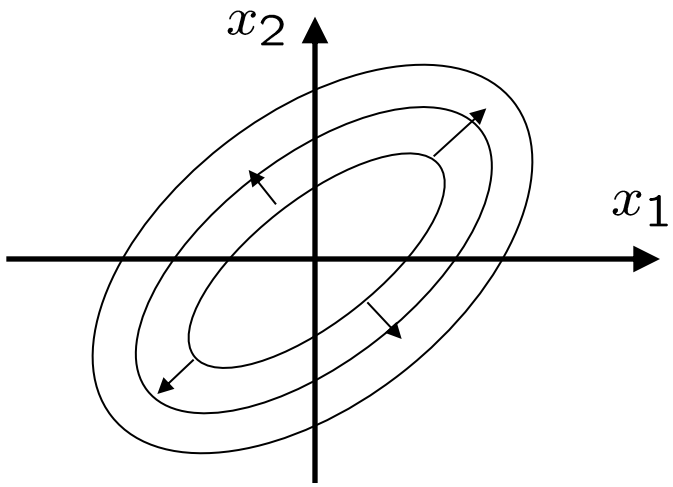
$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & & \\ \vdots & & \ddots & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & & & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

(gives information about local curvature of $f(x)$)

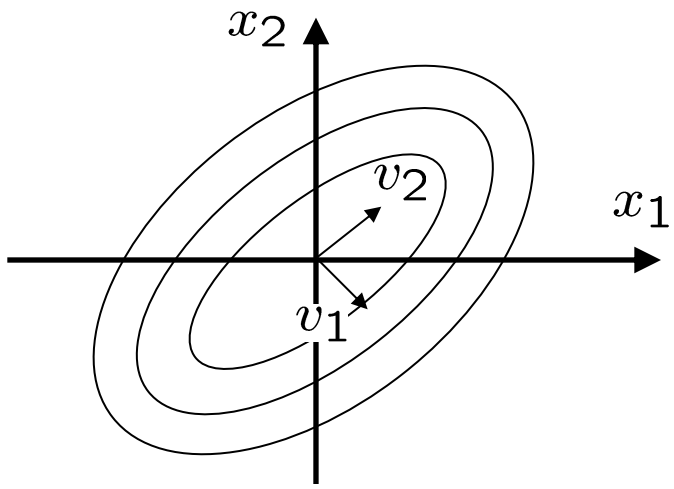
bert.pluymers@esat.kuleuven.be

Gradient & Hessian

Example :



Gradients for different x



Eigenvectors of hessian at the origin ($\lambda_1 > \lambda_2$)

Unconstrained Optimality Conditions

Necessary condition for optimality of $x^\star$

$$\nabla f(x^\star) = 0$$

Sufficient conditions for minimum

$$\nabla f(x^\star) = 0$$

$$\nabla^2 f(x^\star) \text{ positive definite}$$

Classification of optima :

$\nabla^2 f(x^\star)$ positive definite	*minimum*
$\nabla^2 f(x^\star)$ indefinite	*saddle point*
$\nabla^2 f(x^\star)$ negative definite	*maximum*

bert.pluymers@esat.kuleuven.be

Lagrange Multipliers

Introduction of *Lagrange multipliers* leads to *Lagrangian* :

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{l} \lambda_i g_i(x) + \sum_{i=1}^{m} \mu_i h_i(x)$$

with

λ_i Lagrange multipliers of the ineq. constraints
μ_i Lagrange multipliers of the eq. constriants

Constrained optimum can be found as

$$\begin{bmatrix} \max_{\lambda,\mu} \min_x & L(x, \lambda, \mu) \\ \text{s.t.} & \lambda \geq 0 \end{bmatrix}$$

Minimization over x but Maximization over λ, μ !!!!

bert.pluymers@esat.kuleuven.be

Lagrange Multipliers

Constrained optimum can be found as

$$\begin{bmatrix} \max_{\lambda,\mu} \min_{x} & L(x,\lambda,\mu) \\ \text{s.t.} & \lambda \geq 0 \end{bmatrix}$$

First-order optimality conditions in x

$$\nabla f(x^\star) + \sum_{i=1}^{l} \lambda_i \nabla g_i(x^\star) + \sum_{i=1}^{m} \mu_i \nabla h_i(x^\star) = 0$$

Gradient of $f(x)$

Gradient of ineq.

Gradient of eq.

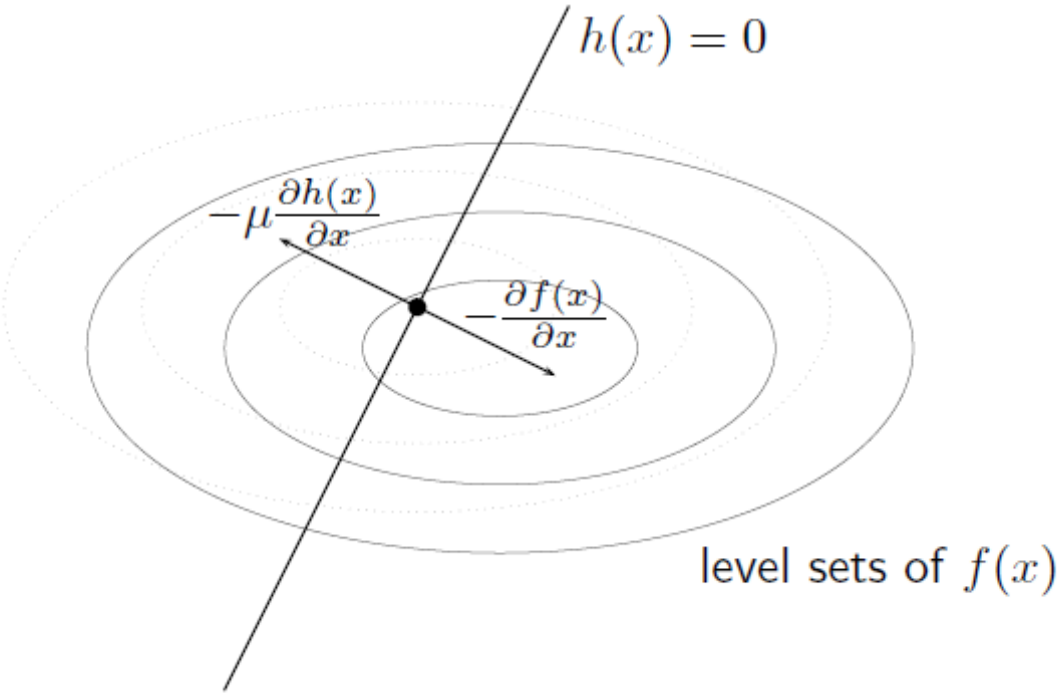
Interpretation ???

bert.pluymers@esat.kuleuven.be

Lagrange Multipliers

- **Intuitive** : additional terms act as additional 'forces' against the gradient of the objective function to keep the optimum *on* or *on the right side of* the constraints.

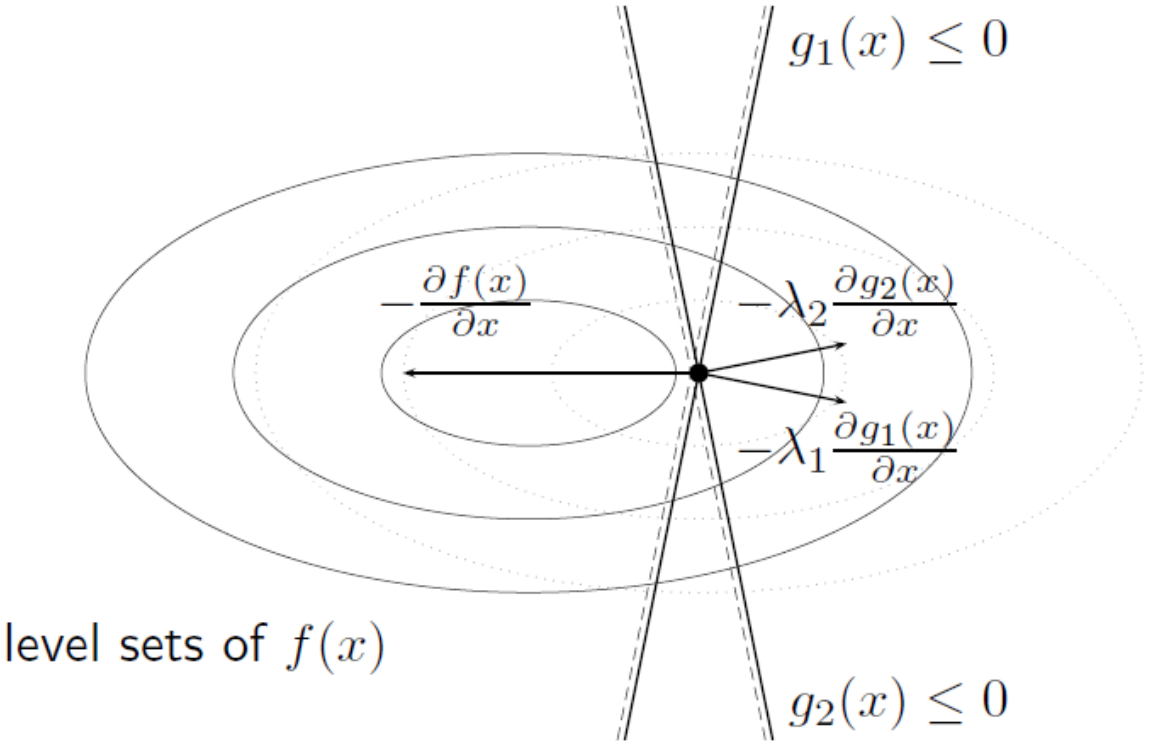
- **Example : equality constraints**



bert.pluymers@esat.kuleuven.be

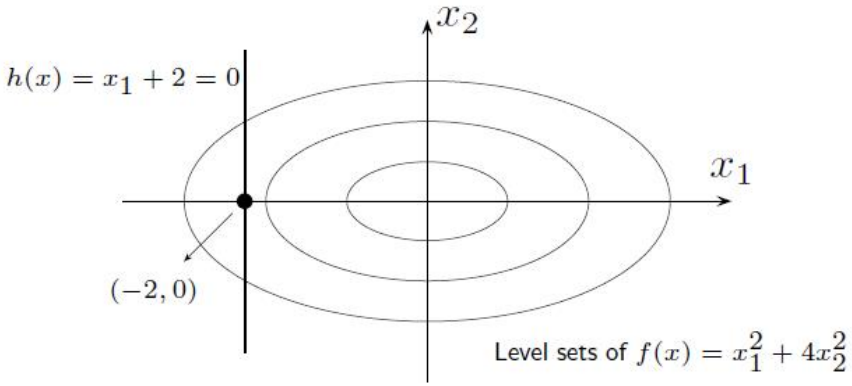
Lagrange Multipliers

- **Example : inequality constraints**

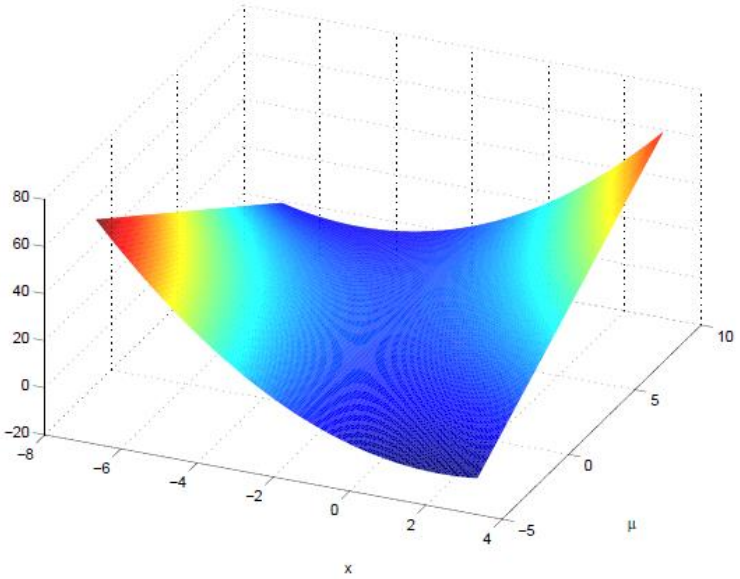


Lagrange Multipliers

Minimization $\leftrightarrow$ maximization :



Lagrangean for $x_2 = 0$



bert.pluymers@esat.kuleuven.be

Lagrange Duality

Consider

$$l(\lambda, \mu) = \inf_x L(x, \lambda, \mu)$$

leading to the *dual problem* :

$$\left[\begin{array}{ll} \max_{\lambda,\mu} & l(\lambda, \mu) \\ \text{s.t.} & \lambda \geq 0 \end{array}\right.$$

- Primal variable : x
- Dual variables : λ, μ

Karush-Kuhn-Tucker Conditions

From previous considerations we can now state necessary conditions for constrained optimality :

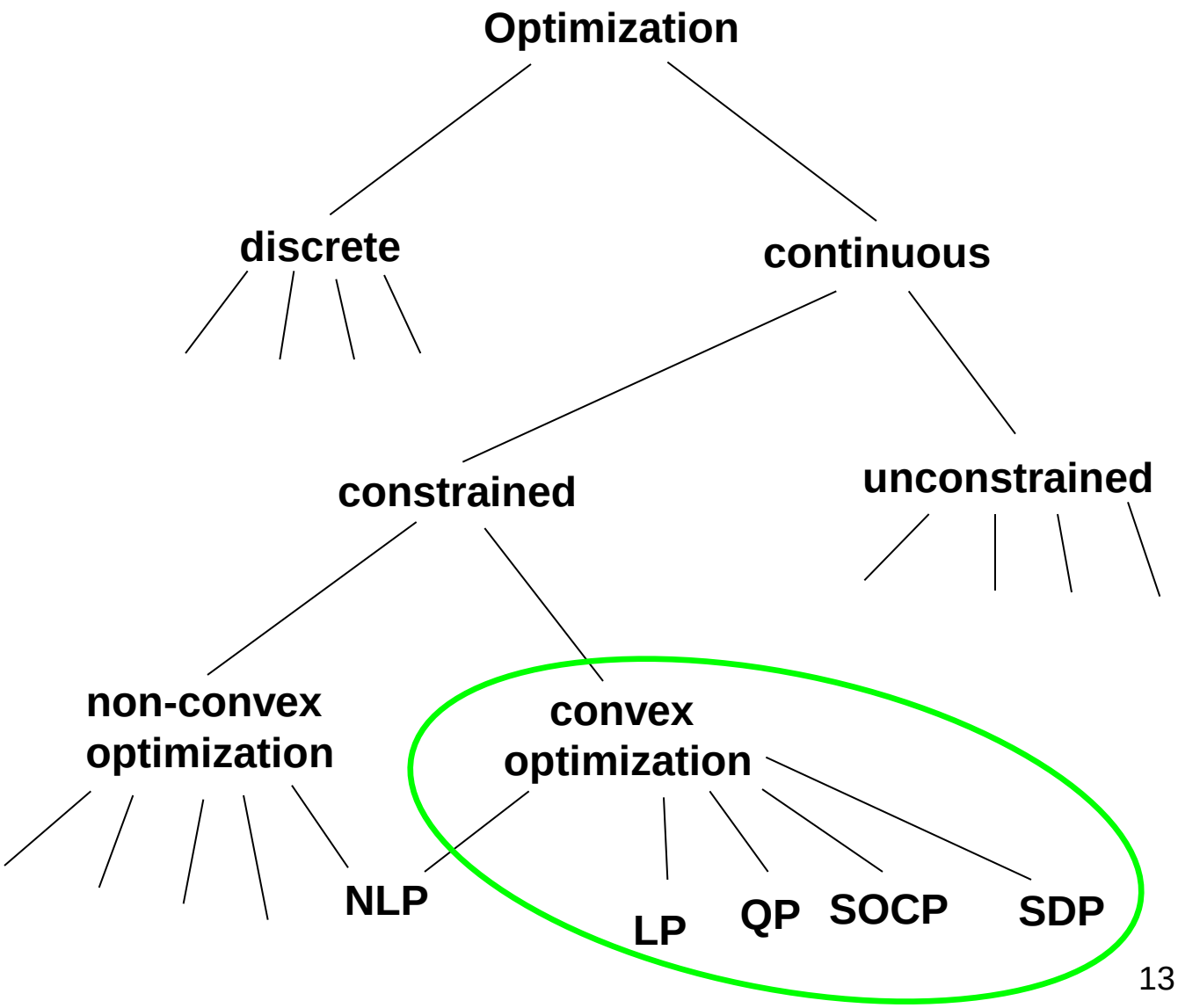
$$\nabla f(x^\star) + \sum_{i=1}^{l} \lambda_i^\star \nabla g_i(x^\star) + \sum_{i=1}^{m} \mu_i^\star \nabla h_i(x^\star) = 0,$$

$$\begin{aligned} h_i(x^\star) &= 0, && \forall i, \\ g_i(x^\star) &\leq 0, && \forall i, \\ \lambda_i^\star &\geq 0, && \forall i, \\ \lambda_i^\star g_i(x^\star) &= 0, && \forall i. \end{aligned}$$

These are called the **KKT conditions**.

bert.pluymers@esat.kuleuven.be

Optimization Tree



Convex Optimization

An optimization problem of the form

$$\begin{bmatrix} \min_x & f(x) \\ \text{s.t.} & h(x) = 0 \\ & g(x) \leq 0 \end{bmatrix}$$

is convex iff for any two *feasible* points x, y :

- $\lambda x + (1-\lambda)y$ is feasible $\forall \lambda \in [0,1]$
- $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)g(x), \forall \lambda \in [0,1]$

This is satisfied iff

- `the cost function is a convex function
- the equality constraints or linear or absent
- the inequality constraints define a convex region

bert.pluymers@esat.kuleuven.be

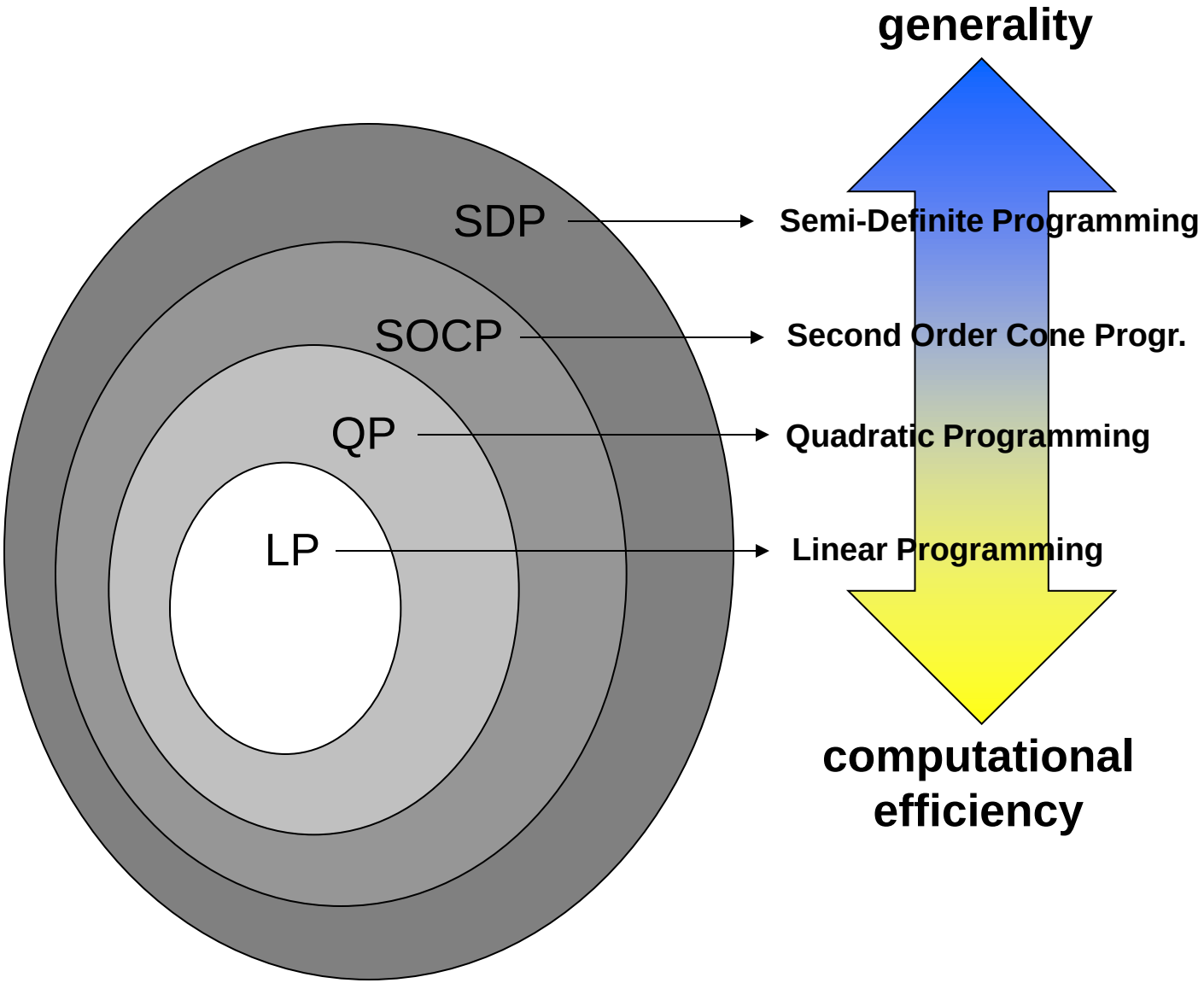
Convex Optimization

Importance of convexity :

- no local minima, one global optimum
- under certain conditions, primal and dual have same solution
- efficient solvers exist
  - polynomial worst-case execution time
  - guaranteed precision

bert.pluymers@esat.kuleuven.be

From LP to SDP



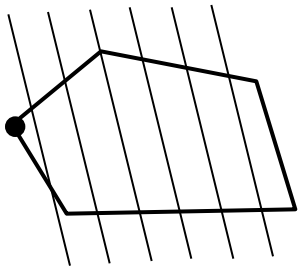
Linear Programming (LP)

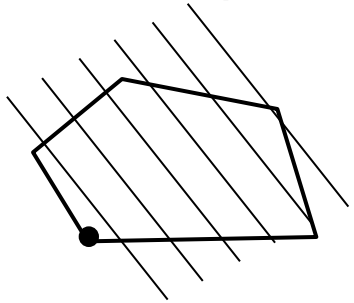
General form :

$$\left[\begin{array}{ll} \min_x & f^\top x \\ \text{s.t.} & A_\text{e} x = b_\text{e} \\ & A_\text{i} x \leq b_\text{i} \end{array}\right.$$

Remarks :

- always convex
- optimal solution always at a corner of ineq. constraints





- typically used in finance / economics / management

bert.pluymers@esat.kuleuven.be

Linear Programming (LP)

Eliminating equality constraints :

$$\left[\begin{array}{ll} \min_x & f^\top x \\ \text{s.t.} & A_\text{e} x = b_\text{e} \\ & A_\text{i} x \leq b_\text{i} \end{array}\right.$$

Reparametrize optimization vector :

$$x = Cy + d$$

with

$$y \in \mathbb{R}^{n-l}$$

$$d = A_\text{e}^\dagger b_\text{e}$$

$$A_\text{e} C = 0 \qquad \text{and } C \text{ full column rank}$$

Leading to

$$\left[\begin{array}{ll} \min_y & f^\top C y \\ \text{s.t.} & A_\text{i} C y \leq b_\text{i} - A_\text{i} d \end{array}\right.$$

bert.pluymers@esat.kuleuven.be

Quadratic Programming (QP)

General form :

$$\left[\begin{array}{ll} \min_x & x^\top H x + 2 f^\top x \\ \text{s.t.} & A_{\text{¡}} x \leq b_{\text{¡}} \end{array}\right.$$

Remarks :

- convex iff $H \succeq 0$
- LP is a special case of QP (imagine $H = 0$)
- Used in all domains of engineering

Second-Order Cone Programming (SOCP)

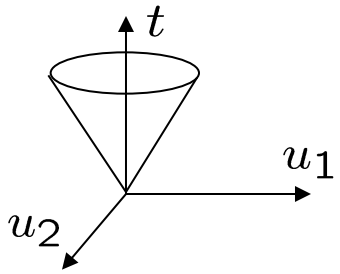
General form :

$$\begin{bmatrix} \min_x & f^\top x \\ \text{s.t.} & A_i x \leq b_i \\ & \|M_i x + n_i\| \leq c_i^\top x + d_i, \quad i = 1, \ldots, N, \end{bmatrix}$$

SOC constraint

Remarks :

- Always convex
- Second-Order, Ice-Cream, Lorentz cone $\|u\| \leq t,$:



- $\|u\| \equiv \sqrt{u^\top u}$
- Engineering applications with sum-of-squares, robust LP, robust QP

bert.pluymers@esat.kuleuven.be

Second-Order Cone Programming (SOCP)

QP as special case of SOCP :

$$\left[\begin{array}{ll} \min_x & x^\mathsf{T} H x + 2 f^\mathsf{T} x \\ \text{s.t.} & A_\mathsf{i} x \leq b_\mathsf{i} \end{array}\right.$$

Rewrite this as

$$\left[\begin{array}{ll} \min_x & \|H^{\frac{1}{2}} x + H^{-\frac{1}{2}} f\|^2 - f^\mathsf{T} H^{-1} f, \\ \text{s.t.} & A_\mathsf{i} x \leq b_\mathsf{i} \end{array}\right.$$

which is equivalent to

$$\left[\begin{array}{ll} \min_x & \|H^{\frac{1}{2}} x + H^{-\frac{1}{2}} f\|^2, \\ \text{s.t.} & A_\mathsf{i} x \leq b_\mathsf{i} \end{array}\right.$$

By introducing an additional variable t we get the SOCP

$$\left[\begin{array}{ll} \min_{x,t} & t \\ \text{s.t.} & \|H^{\frac{1}{2}} x + H^{-\frac{1}{2}} f\| \leq t \\ & A_\mathsf{i} x \leq b_\mathsf{i} \end{array}\right.$$

bert.pluymers@esat.kuleuven.be

Semi-Definite Programming (SDP)

General form :

$$\left[\begin{array}{ll} \min_x & f^\mathsf{T} x \\ \text{s.t.} & F(x) \succeq 0 \end{array}\right.$$

with

$$F(x) = F_0 + \sum_{i=1}^{n} x_i F_i, \qquad F_i = F_i^\mathsf{T} \in \mathbb{R}^{k \times k}$$

Remarks :

- $A \succeq 0$ means that A should be positive semi-definite
- $A \succeq B$ means that $A - B$ should be pos. semi-def.
- always convex :

$$F(x) \succeq 0, F(y) \succeq 0, \qquad F(\lambda x + (1-\lambda)y) \succeq 0, \ \forall \lambda \in [0,1]$$

- ineq. constraints called LMI's : **L**inear **M**atrix **I**neq.
- LMI's arise in many applications of Systems & Control Theory

bert.pluymers@esat.kuleuven.be

Semi-Definite Programming (SDP)

convexity :

$$F(x) \succeq 0, F(y) \succeq 0, \qquad F(\lambda x + (1-\lambda)y) \succeq 0, \ \forall \lambda \in [0,1]$$

Easily verified :

$$\begin{aligned} F(x) \succeq 0 \quad &\Rightarrow v^{\mathsf{T}} F(x) v \geq 0, \qquad \forall v \in \mathbb{R}^k, \\ F(y) \succeq 0 \quad &\Rightarrow v^{\mathsf{T}} F(y) v \geq 0, \qquad \forall v \in \mathbb{R}^k, \end{aligned}$$

hence

$$\begin{aligned} \lambda(v^{\mathsf{T}} F(x) v) + (1-\lambda)(v^{\mathsf{T}} F(y) v) \geq 0, \qquad & \forall v \in \mathbb{R}^k, \forall \lambda \in [0,1] \\ v^{\mathsf{T}}(\lambda F(x) + (1-\lambda) F(y)) v \geq 0, \qquad & \forall v \in \mathbb{R}^k, \forall \lambda \in [0,1] \\ v^{\mathsf{T}} F(\lambda x + (1-\lambda) y) v \geq 0, \qquad & \forall v \in \mathbb{R}^k, \forall \lambda \in [0,1] \end{aligned}$$

and therefore

$$F(\lambda x + (1-\lambda)y) \succeq 0, \ \forall \lambda \in [0,1]$$

which means that *LMI's are convex constraints*.

bert.pluymers@esat.kuleuven.be

Semi-Definite Programming (SDP)

Schur Complement :

$$\begin{bmatrix} A & B^T \\ B & C \end{bmatrix} \succeq 0$$

is equivalent with

$$A - B^T C^{-1} B \succeq 0, \qquad C \succeq 0$$

More general :

$$\begin{bmatrix} A & B_1^T & B_2^T \\ B_1 & C_1 & 0 \\ B_2 & 0 & C_2 \end{bmatrix} \succeq 0 \quad \Leftrightarrow \quad \begin{cases} A - B_1^T C_1^{-1} B_1 - B_2^T C_2^{-1} B_2 \succeq 0, \\ C_1 \succeq 0, \\ C_2 \succeq 0 \end{cases}$$

Remarks :

- Originally developed in a statistical framework
- Today widely used in S&C in order to reformulate problems involving eigenvalues as an LMI.

bert.pluymers@esat.kuleuven.be

Semi-Definite Programming (SDP)

SOCP as special case of SDP :

$$\left[\begin{array}{ll} \min_x & f^\top x \\ \text{s.t.} & \|M_i x + n_i\| \leq c_i^\top x + d_i, \quad i = 1, \ldots, N, \end{array}\right.$$

is equivalent with (by using Schur complement) :

$$\left[\begin{array}{ll} \min_x & f^\top x \\ \text{s.t.} & \left[\begin{array}{cc} (c_i^\top x + d_i) I & (M_i x + n_i) \\ (M_i x + n_i)^\top & c_i^\top x + d_i \end{array}\right] \succeq 0, \quad i = 1, \ldots, N, \end{array}\right.$$

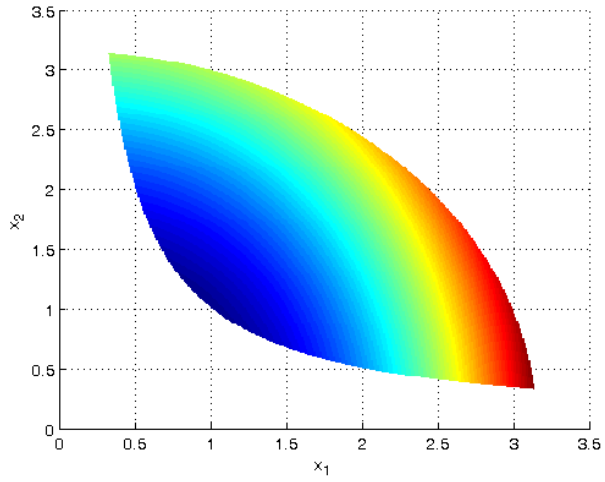
(*exercise : apply Schur complement to LMI and reconstruct SOC constraint*)

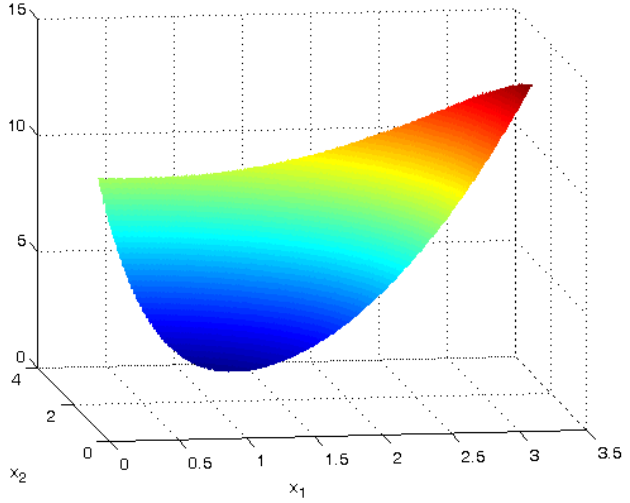
Convexity = SDP ?

Formulated as SDP	$\Rightarrow$	Convex optimization
Convex optimization	$\not\Rightarrow$	formulatable as SDP

Example :

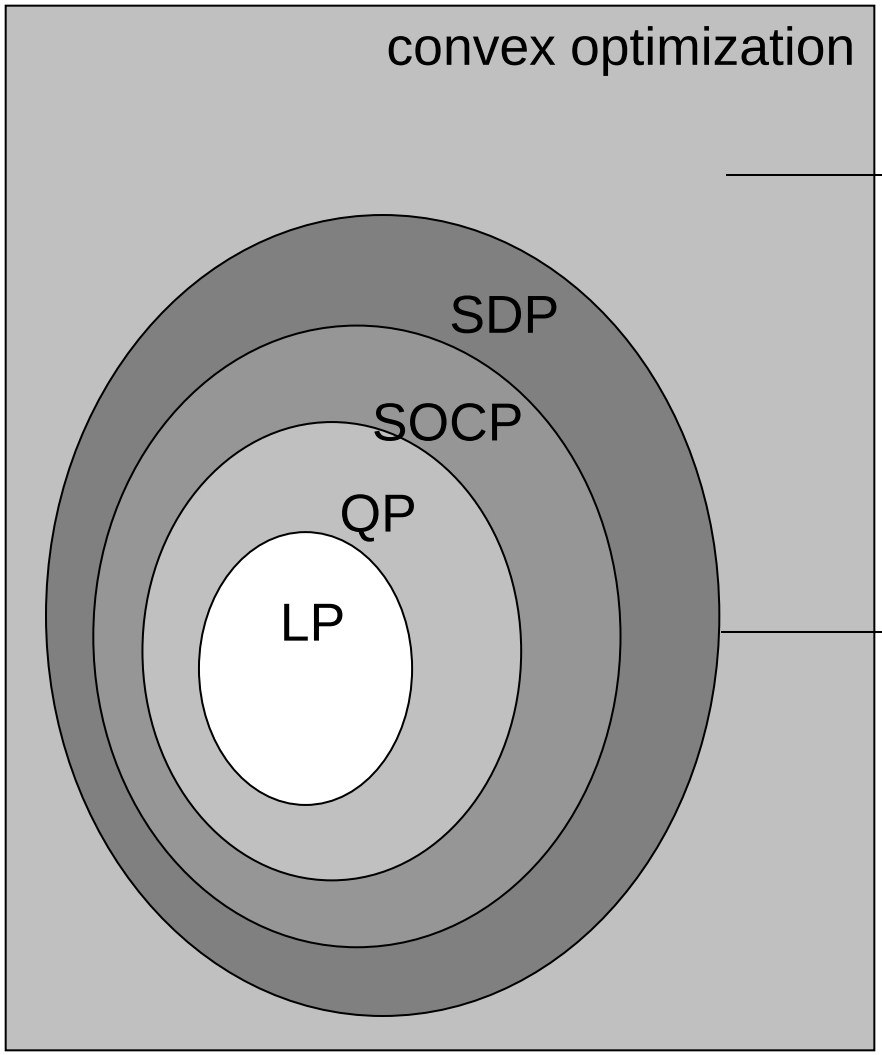
$$\begin{cases} \min_{x_1,x_2} & x_1^{2.34} + x_2^{1.89}, \\ \text{s.t.} & x_1 x_2 \geq 1, \\ & x_1^2 + x_2^2 \leq 10. \end{cases}$$





bert.pluymers@esat.kuleuven.be

Convexity ≠ SDP !

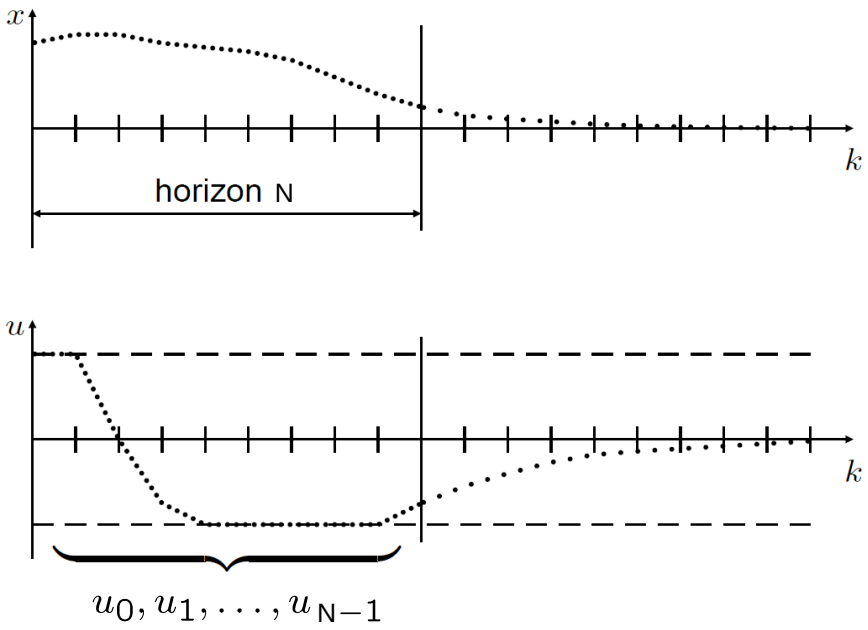


convexity difficult to exploit
(computationally)

structure easily exploitable
(many toolboxes available)
→ **significant efficiency gains !**

MPC Paradigm

- At every discrete time instant k , given information about the current system state x_k, calculate an 'optimal' input sequence over a finite time horizon :



- Apply the first input u_k to the real system
- Repeat at the next time instant $k+1$, using new state measurements / estimates.

Dynamic Programming

- Finding the optimal input sequence is done by means of ***Dynamic Programming***

- Definition* :

  "DP is a class of solution methods for solving sequential decision problems with a compositional cost structure"

- Invented by Richard Bellman (1920-1984) in 1953

bert.pluymers@esat.kuleuven.be

Dynamic Programming

Example 1 : The darts problem* :

"Obtain a score of 301 as fast as possible while beginning and ending in a double."

http://plus.maths.org/issue3/dynamic/

- **Decision** : next area towards which to throw the dart
- **Cost** : time

* *D. Kohler, Journal of the Operational Research Society*, 1982.

bert.pluymers@esat.kuleuven.be

Dynamic Programming

Example 2 : DNA sequence allignment :

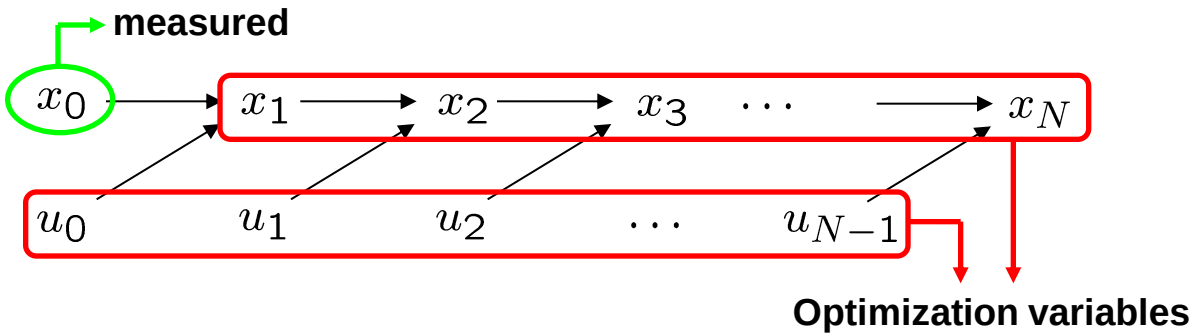
```
G A A T T C A G T T A    (sequence #1)
G G A T C G A            (sequence #2)
```

```
G  A  A  T  T  C  A  G  T  T  A
|     |     |  |     |        |
G  G  A  _  T  C  _  G  _  _  A
```

- **Decisions** : which nucleotides to match
- **Cost** : e.g. based on substitution / insertion prob.
- **Algorithms** : Baum/Welch, Waterman/Smith, …

Dynamic Programming in MPC

"*Series of sequential decisions*" :



Typical optimization problem :

$$
\begin{bmatrix}
\min_{\mathbf{x}_N, \mathbf{u}_N} & \sum_{k=1}^{N} \left(x_k^\mathsf{T} Q x_k \right) + \sum_{k=0}^{N-1} \left(u_k^\mathsf{T} R u_k \right), & \\
\text{s.t.} & A_u u_k \leq 1_\mathsf{v}, & k = 0, \ldots, N-1, \\
 & A_x x_k \leq 1_\mathsf{v}, & k = 1, \ldots, N, \\
 & x_{k+1} = A x_k + B u_k, & k = 0, \ldots, N-1,
\end{bmatrix}
$$

→ standard QP formulation ?

bert.pluymers@esat.kuleuven.be

Linear MPC as standard QP

Optimization vector :

$$x_{\text{opt}} = [u_0;\ \ldots\ ; u_{N-1};\ x_1;\ \ldots\ ; x_N]$$

Cost function :

$$x_{\text{opt}}^{\mathsf{T}} H x_{\text{opt}} + 2 f^{\mathsf{T}} x_{\text{opt}}$$

$$H = \begin{bmatrix} R & & & & & \\ & \ddots & & & & \\ & & R & & & \\ & & & Q & & \\ & & & & \ddots & \\ & & & & & Q \end{bmatrix}, \qquad f = 0.$$

For convexity $H \succeq 0$ hence $Q \succeq 0, R \succeq 0$.

bert.pluymers@esat.kuleuven.be

Linear MPC as standard QP

Equality constraints

$$A_{\mathrm{e}} x_{\mathrm{opt}} = b_{\mathrm{e}}$$

with

$$A_{\mathrm{e}} = \begin{bmatrix} -B & & I & & \\ & -B & -A & I & \\ & & \ddots & \ddots & \ddots \end{bmatrix}, \qquad b_{\mathrm{e}} = \begin{bmatrix} Ax_0 \\ 0 \\ \vdots \end{bmatrix}.$$

(u_0, u_1, x_1, x_2 mark the block columns)

Sparsity : many entries in A_{e} equal to 0

Linear MPC as standard QP

Inequality constraints

$$A_{\mathsf{i}} x_{\mathsf{opt}} = b_{\mathsf{i}}$$

with

$$A_{\mathsf{i}} = \begin{bmatrix} A_{\mathsf{u}} & & & & & \\ & \ddots & & & & \\ & & A_{\mathsf{u}} & & & \\ & & & A_{\mathsf{x}} & & \\ & & & & \ddots & \\ & & & & & A_{\mathsf{x}} \end{bmatrix}, \qquad b_{\mathsf{i}} = \begin{bmatrix} b_{\mathsf{u}} \\ \vdots \\ b_{\mathsf{u}} \\ b_{\mathsf{x}} \\ \vdots \\ b_{\mathsf{x}} \end{bmatrix}.$$

Sparsity : many entries in A_{e} equal to 0

bert.pluymers@esat.kuleuven.be

Alternatives

Typical Problems : LP

Linear model, linear constraints, linear objective function :

$$\left[\begin{array}{lll} \min_{\mathbf{x}_N,\mathbf{u}_N} & \sum_{k=1}^{N} w_x^T|x_k - r_k| + \sum_{k=0}^{N-1} w_u^T|u_k| & \\ \text{s.t.} & |u_k| \leq u_{\max} & k = 0,\ldots,N-1 \\ & |x_k| \leq x_{\max} & k = 1,\ldots,N-1 \\ & x_{k+1} = Ax_k + Bu_k & k = 0,\ldots,N-1 \end{array}\right.$$

- slightly faster to solve
- 'chattering' : optimal solution jumps around

bert.pluymers@esat.kuleuven.be

Alternatives

Typical Problems : NLP

Non-linear model, linear constraints, quadratic objective function :

$$\left[\begin{array}{lll} \min_{\mathbf{x}_N, \mathbf{u}_N} & \sum_{k=1}^{N} (x_k - r_k)^T Q (x_k - r_k) & \\ & + \sum_{k=0}^{N-1} u_k^T R u_k & \\ \text{s.t.} & |u_k| \leq u_{\max} & k = 0, \ldots, N-1 \\ & |x_k| \leq x_{\max} & k = 1, \ldots, N-1 \\ & x_{k+1} = f(x_k, u_k) & k = 0, \ldots, N-1 \end{array}\right.$$

Results in a general *non-linear program (NLP)*.

Non-convex optimization in general : to be avoided !!!

bert.pluymers@esat.kuleuven.be

MPC and optimization

Special requisites

- MPC = Real-time !
- scalability
  - → suitable for large-scale systems
    - → especially for MPC problems
  - → worst case complexity
- intermediate results
  - → what to do when deadline nears ?
- warm starts
  - → reuse information from previous time step
- exploit sparsity

bert.pluymers@esat.kuleuven.be

MPC and optimization

Special requisites : scaling of MPC

- # variables : $N(n_x + n_u)$ or Nn_u
- # constraints : $2N(n_x + n_u)$ (bounds)
- # active constraints : ???

Two dimensions :

1. variables, constraints ($\sim$ proportional)
   $\rightarrow$ constant for given model and given N
2. active constraints (between 0 and 0.5 # constraints)
   $\rightarrow$ varies each time step k

bert.pluymers@esat.kuleuven.be

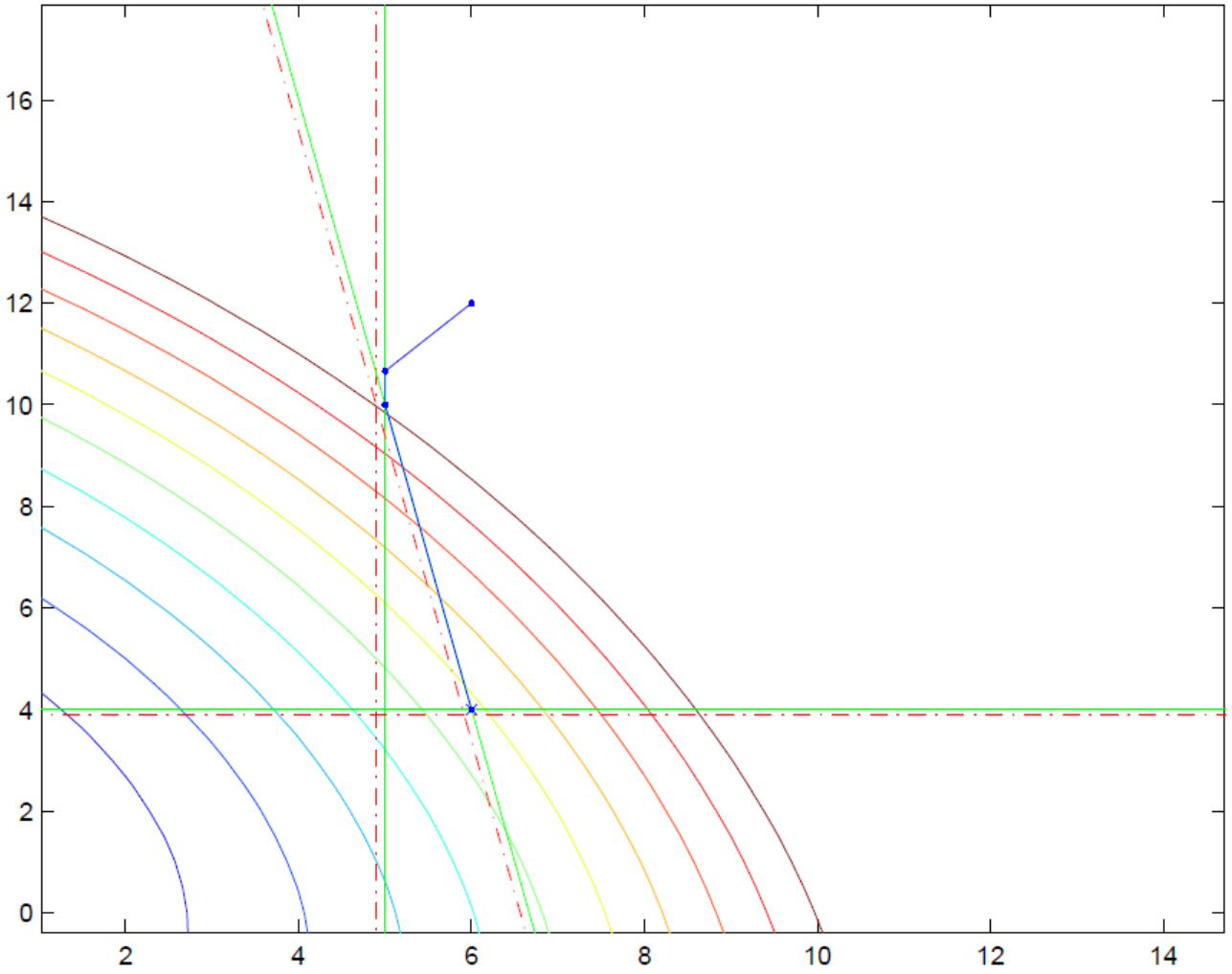
Active Set Methods

1. choose an **initial** *feasible* **point** x_0
2. make initial guess about the *active constraints* in the optimum
3. convert the active constraints (=active set, working set) to equality constraints
4. omit the other constraints
5. solve the resulting equality constrained LCQP
6. use the result as a search direction and proceed until a new constraint is encountered
7. update active set
8. until convergence, repeat from step 2

bert.pluymers@esat.kuleuven.be

Active Set Methods

- Overview
- Optimization Basics
- Convex Optimization
- Dynamic Optimization
- **Optimization Algorithms**



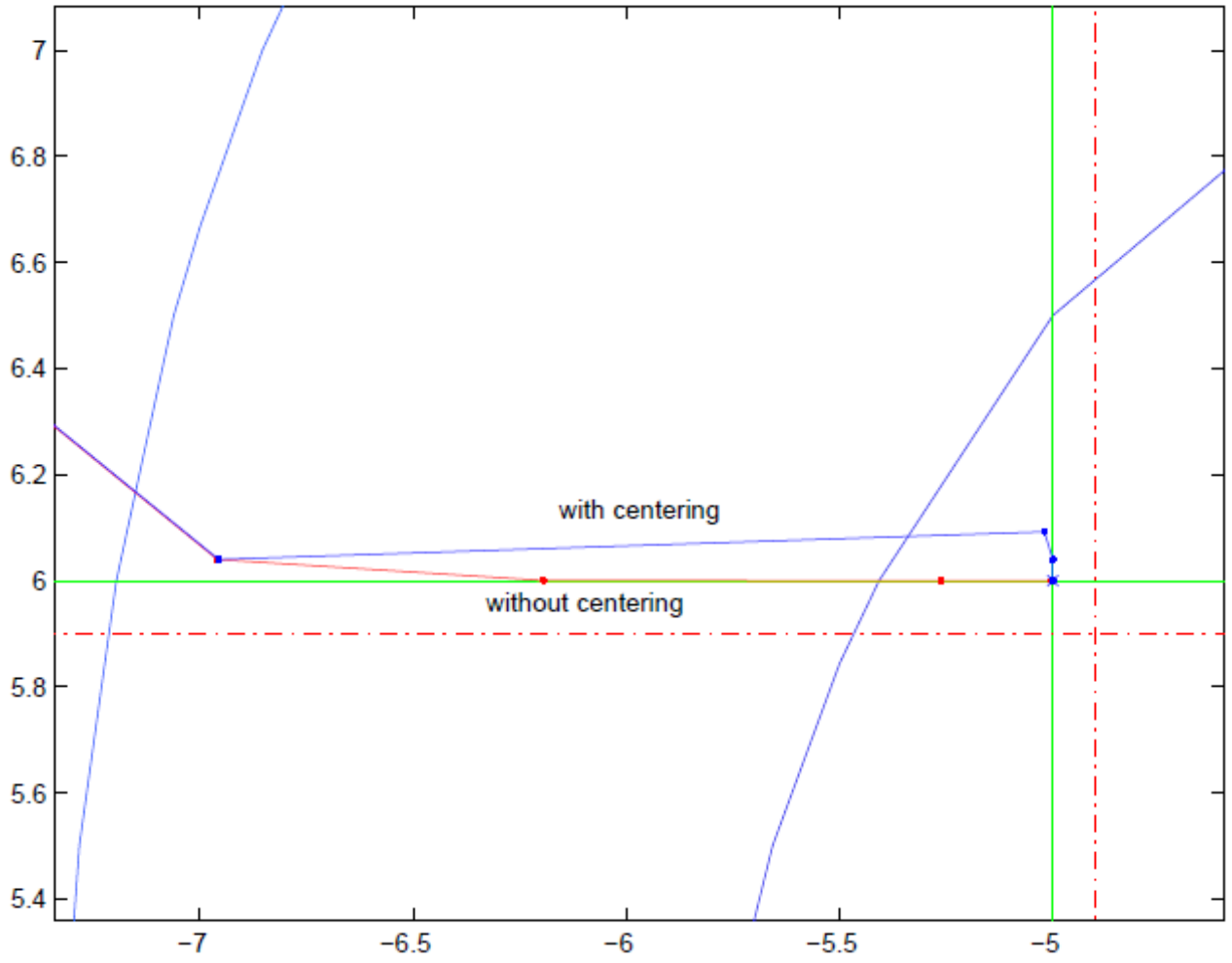
bert.pluymers@esat.kuleuven.be

Interior Point Methods

1. Choose initial point (x, λ_i, μ_i)
2. Linearize KKT conditions around current point
3. Solve Linearized KKT system to obtain *search direction*
4. Calculate step length such that $\lambda_i \geq 0, g_i(x) \leq 0$
5. Repeat from step 2, until convergence

Interior Point Methods

- Overview
- Optimization Basics
- Convex Optimization
- Dynamic Optimization
- **Optimization Algorithms**



bert.pluymers@esat.kuleuven.be

Comparison

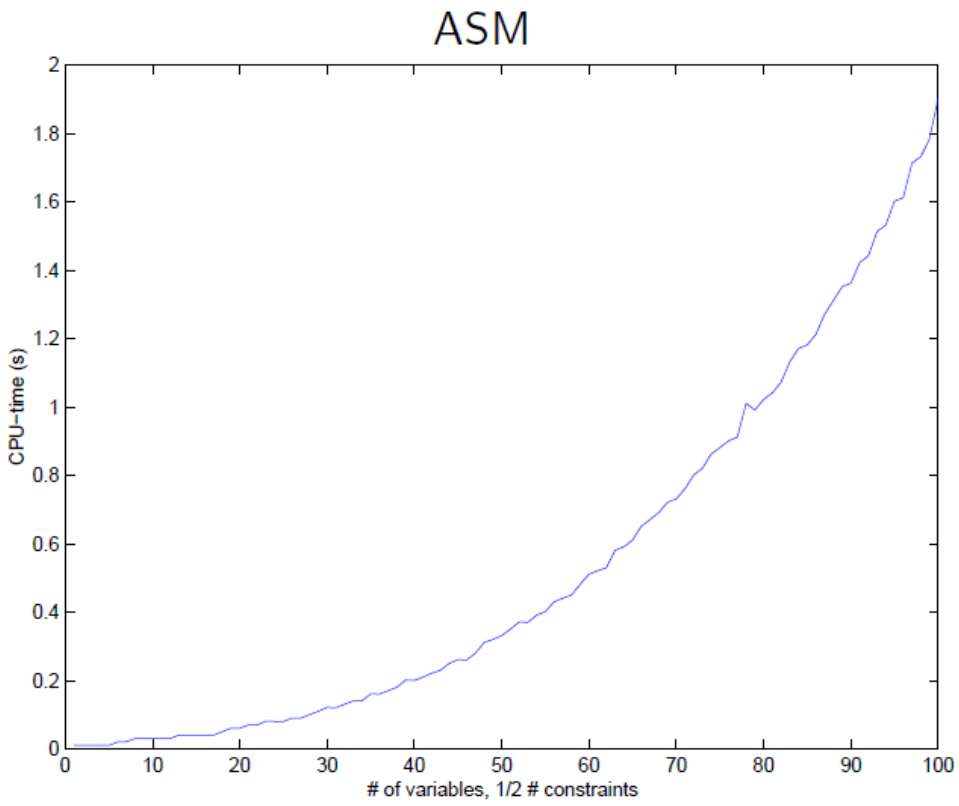
- *ASM allow hot start*
  - reuse of active set, factorizations, …
- *ASM has feasible intermediate results*
  - by construction
- *IPM can exploit sparsity*
  - solution of KKT system by sparse solver

In industry currently mostly ASM due to first two advantages, but IPM under consideration…

bert.pluymers@esat.kuleuven.be

Comparison

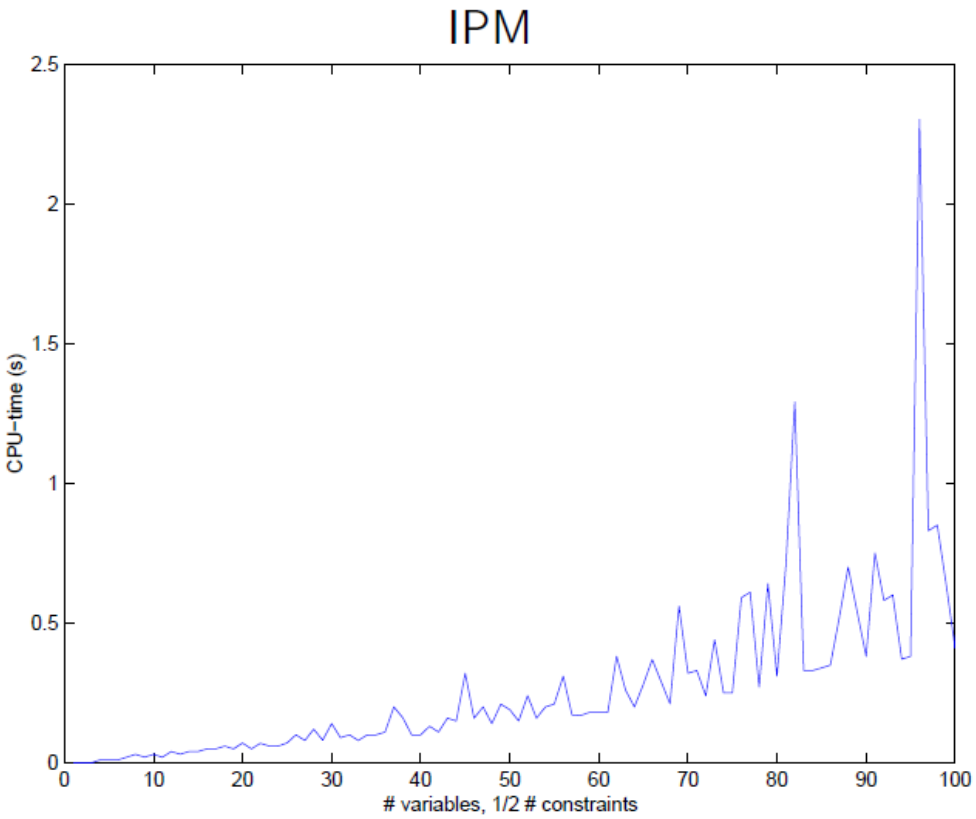
Case study : IPM & ASM complexity



For larger n approximately $O(n^3)$.

Comparison

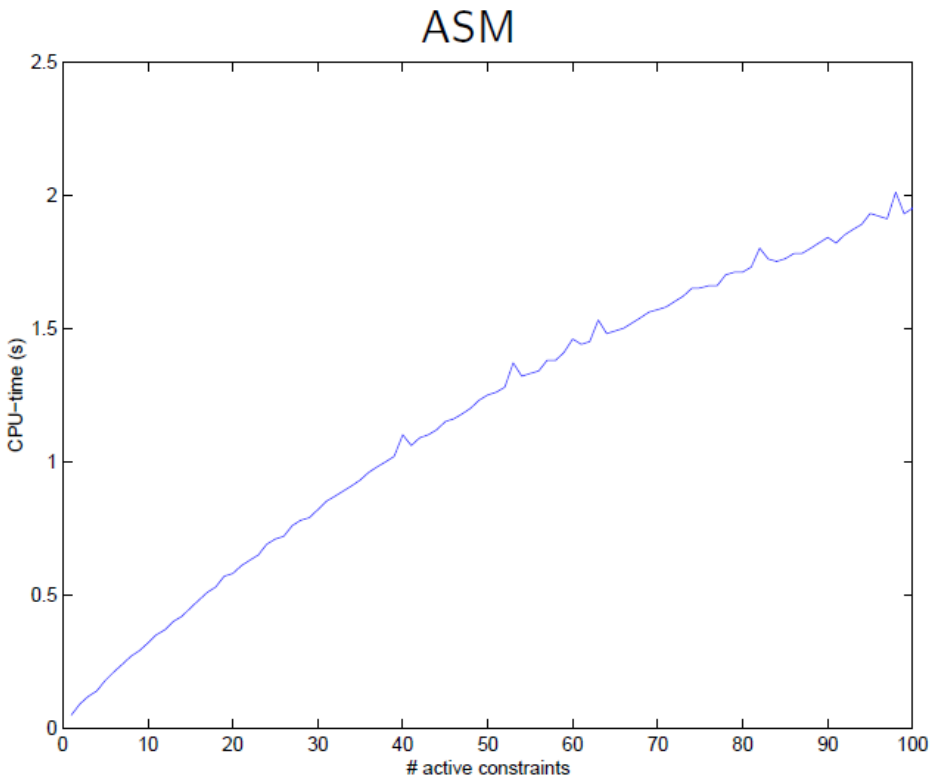
Case study : IPM & ASM complexity



Unclear from results, but theoretically $O(n^3)$.

Comparison

Case study : IPM & ASM complexity

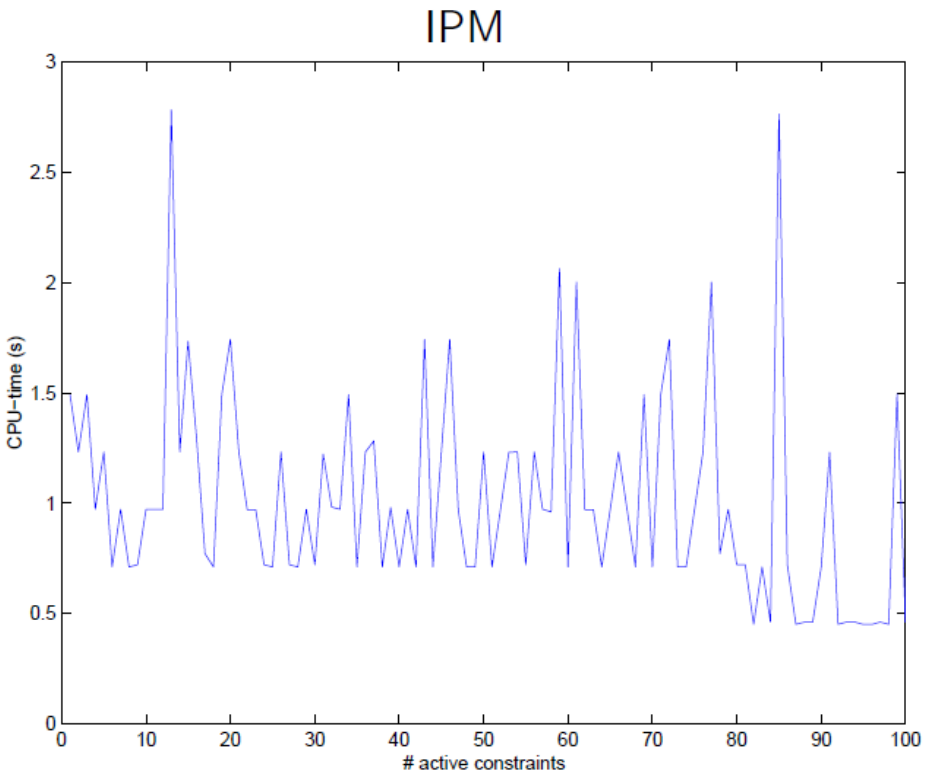


More or less linear ?

bert.pluymers@esat.kuleuven.be

Comparison

Case study : IPM & ASM complexity



O(1) !!!

bert.pluymers@esat.kuleuven.be

Conclusion

- convex optimization powerful tool !!!
- different optimization algorithm have pro's and con's
- try to avoid NLP's !!!!

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

= the bible!

Freely available at

`http://www.stanford.edu/~boyd/cvxbook.html`

H0K03a : Advanced Process Control

Model-based Predictive Control 3 : Stability

Bert Pluymers
Prof. Bart De Moor

Katholieke Universiteit Leuven, Belgium
Faculty of Engineering Sciences
Department of Electrical Engineering (ESAT)
Research Group SCD-SISTA

Overview

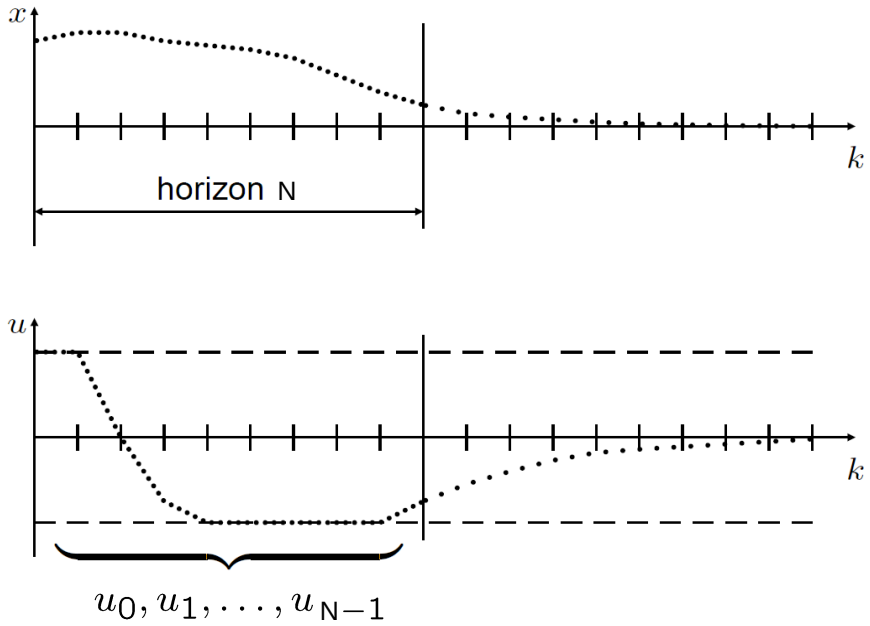
Lecture 3 : Stability

- Introduction
- Example
- Stability Theory
- Set Invariance
- Implementations

bert.pluymers@esat.kuleuven.be

MPC Paradigm

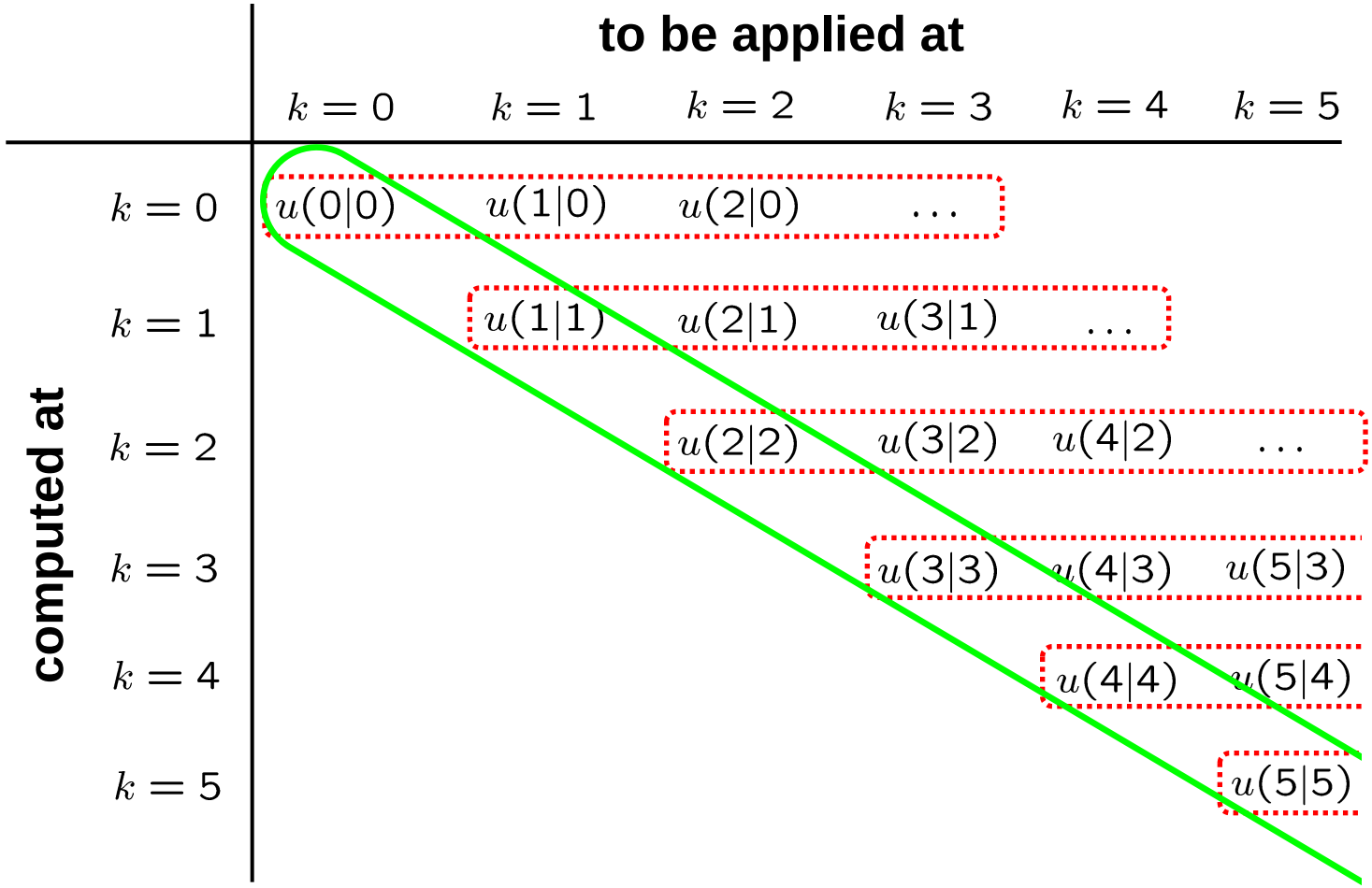
- At every discrete time instant k , given information about the current system state x_k, calculate an 'optimal' input sequence over a finite time horizon :



- Apply the first input u_k to the real system
- Repeat at the next time instant $k+1$, using new state measurements / estimates.

bert.pluymers@esat.kuleuven.be

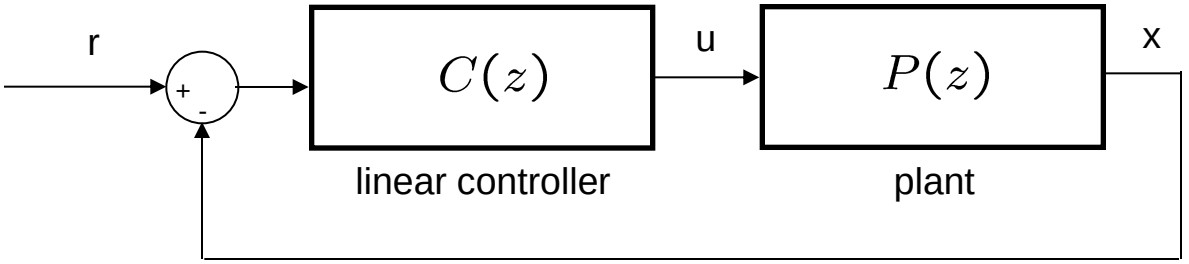
Optimality of input sequence



bert.pluymers@esat.kuleuven.be

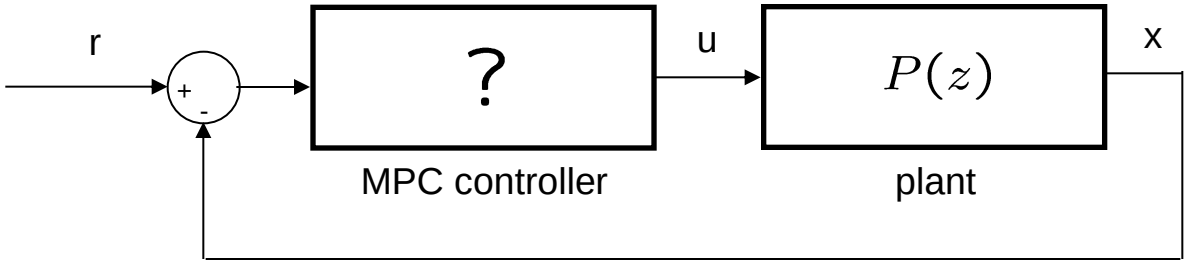
Stability Analysis

- **Classical Way :**



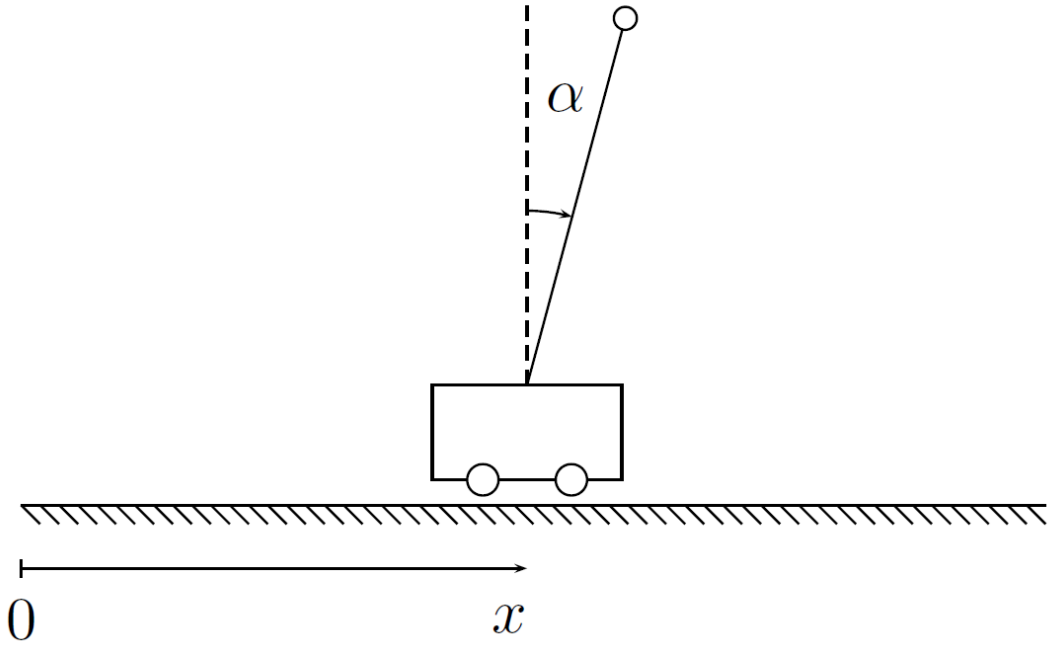
Analyse poles/zeros of $P(z)C(z)$ and associated transfer functions.

- **Modelbased Predictive Control :**



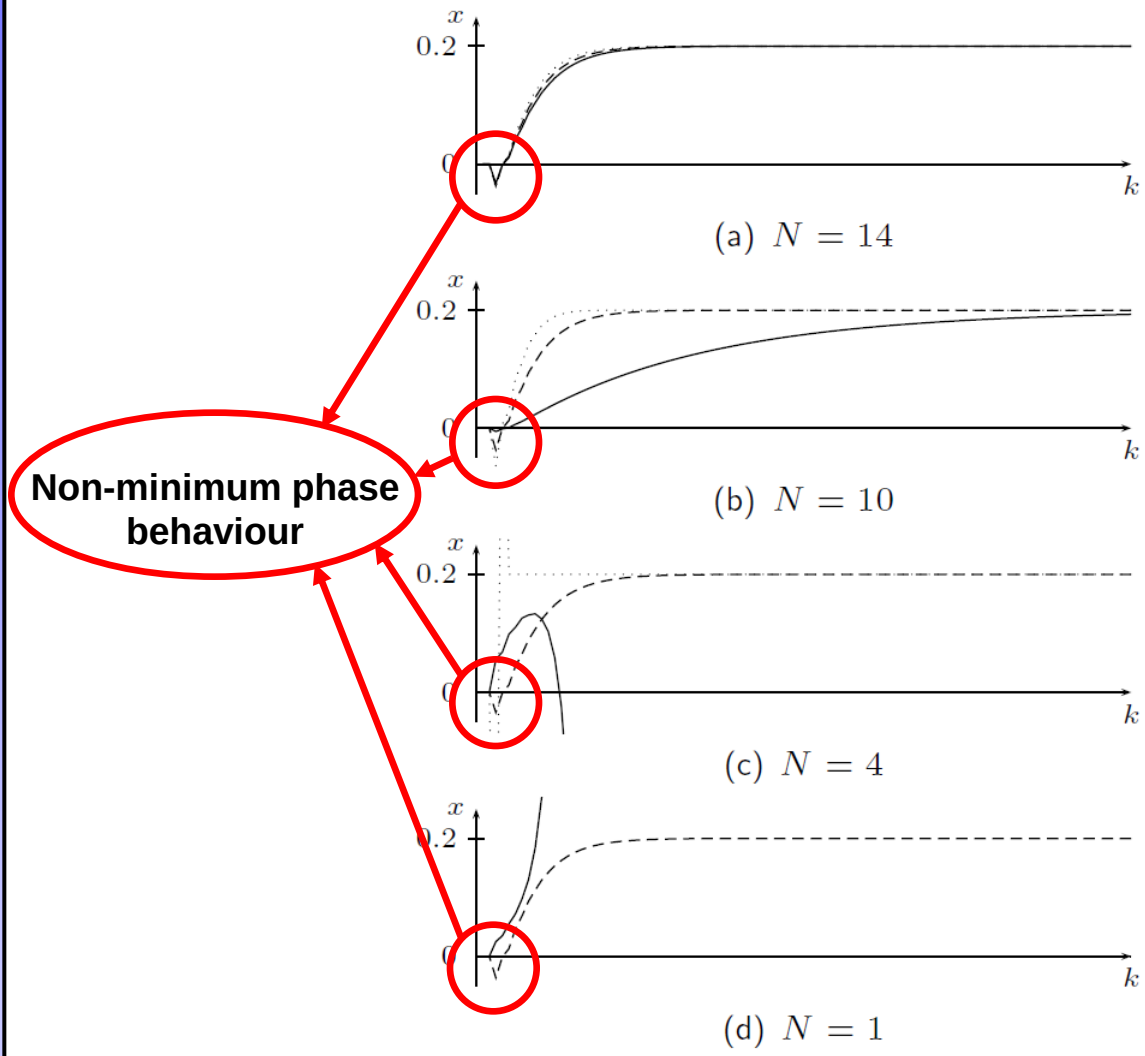
Lyapunov theory for stability.

Inverted Pendulum



- 1 input : V_{motor}
- 4 states : $x, \dot{x}, \alpha, \dot{\alpha}$
- open loop unstable system

Inverted Pendulum



- 4 different horizon lengths
- 3 different MPC variants (to be defined later)

Stability Theory

- Explicit vs. Optimization-based controller
- Transfer functions → Lyapunov theory

Stability is obtained / proven in 2 steps :

1. **Recursive feasibility**

   i.e. controller well-defined for all k

2. **Lyapunov function construction**

   i.e. trajectories converge to equilibrium

Stability Theory

Limited validity of MPC stability framework :

- only for *'stabilization'* problems :
  - initial state
  - system steered towards
  - no disturbances allowed (but extension possible)

- no general stability framework for *'tracking'* problems

bert.pluymers@esat.kuleuven.be

Stability Theory

Recursive Feasibility

If the optimization problem is feasible for time k *, then it is also feasible for time* $k+1$.

(*and hence for all* $k+i, i \geq 0$)

Feasible Region

The region in state space, defined by all states x *for which the MPC optimization problem is feasible.*

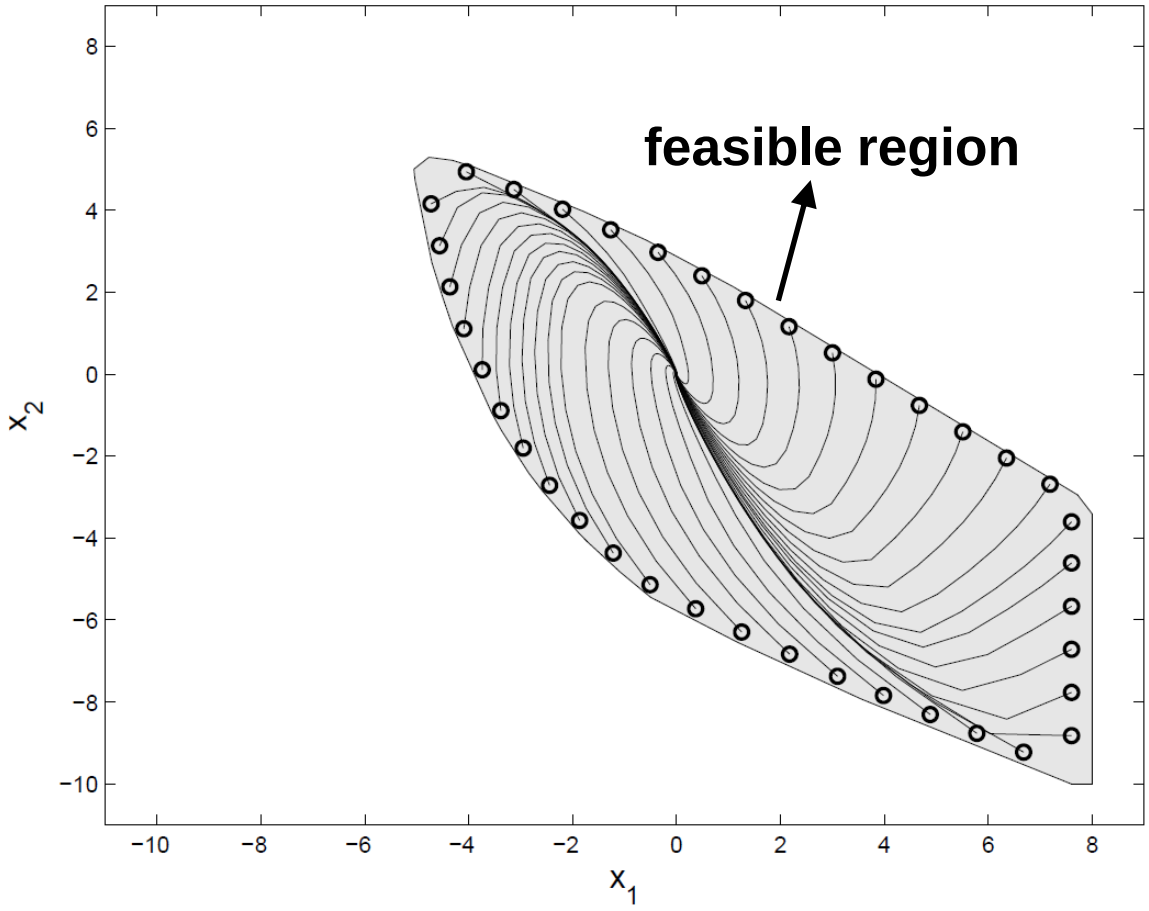
→ **Recursive feasibility proven** : all states within feasible region lead to trajectories for which the MPC-controller is feasible and hence well-defined.

---

bert.pluymers@esat.kuleuven.be

Stability Theory

→ **Recursive feasibility proven** : all states within feasible region lead to trajectories for which the MPC-controller is feasible and hence well-defined.



bert.pluymers@esat.kuleuven.be

MPC Stability Measures

Some new notations :

$x(k+i\|k)$	state at time $k+i$, predicted at time k
$x(k) \equiv x(k\|k)$	real state at time k
$\mathbf{x}_N(k)$	$= [x(k\|k); \ldots; x(k+N\|k)]$
$u(k+i\|k)$	input at time $k+i$, calculated at time k
$u(k) \equiv u(k\|k)$	real input at time k
$\mathbf{u}_N(k)$	$= [u(k\|k); \ldots; u(k+N-1\|k)]$

MPC Stability Measures

$$
\begin{aligned}
&J_N^o(x(k)) = \\
&\quad \min_{\mathbf{x}_N(k),\mathbf{u}_N(k)} \ \sum_{i=0}^{N-1} l(x(k+i|k),u(k+i|k)) \\
&\qquad\qquad + F(x(k+N|k)) \\
&\text{s.t.} \qquad x(k+i|k) \in \mathbb{X} \qquad\qquad i = 1 \dots N-1 \\
&\qquad\quad u(k+i|k) \in \mathbb{U} \qquad\qquad i = 0 \dots N-1 \\
&\qquad\quad x(k+N|k) \in \mathbb{X}_N \\
&\qquad\quad x(k+i+1|k) = f(x(k+i|k),u(k+i|k)) \\
&\qquad\qquad\qquad\qquad\qquad\qquad i = 0 \dots N-1 \\
&\qquad\quad x(k|k) = x_k \qquad\qquad \text{(measured)}
\end{aligned}
$$

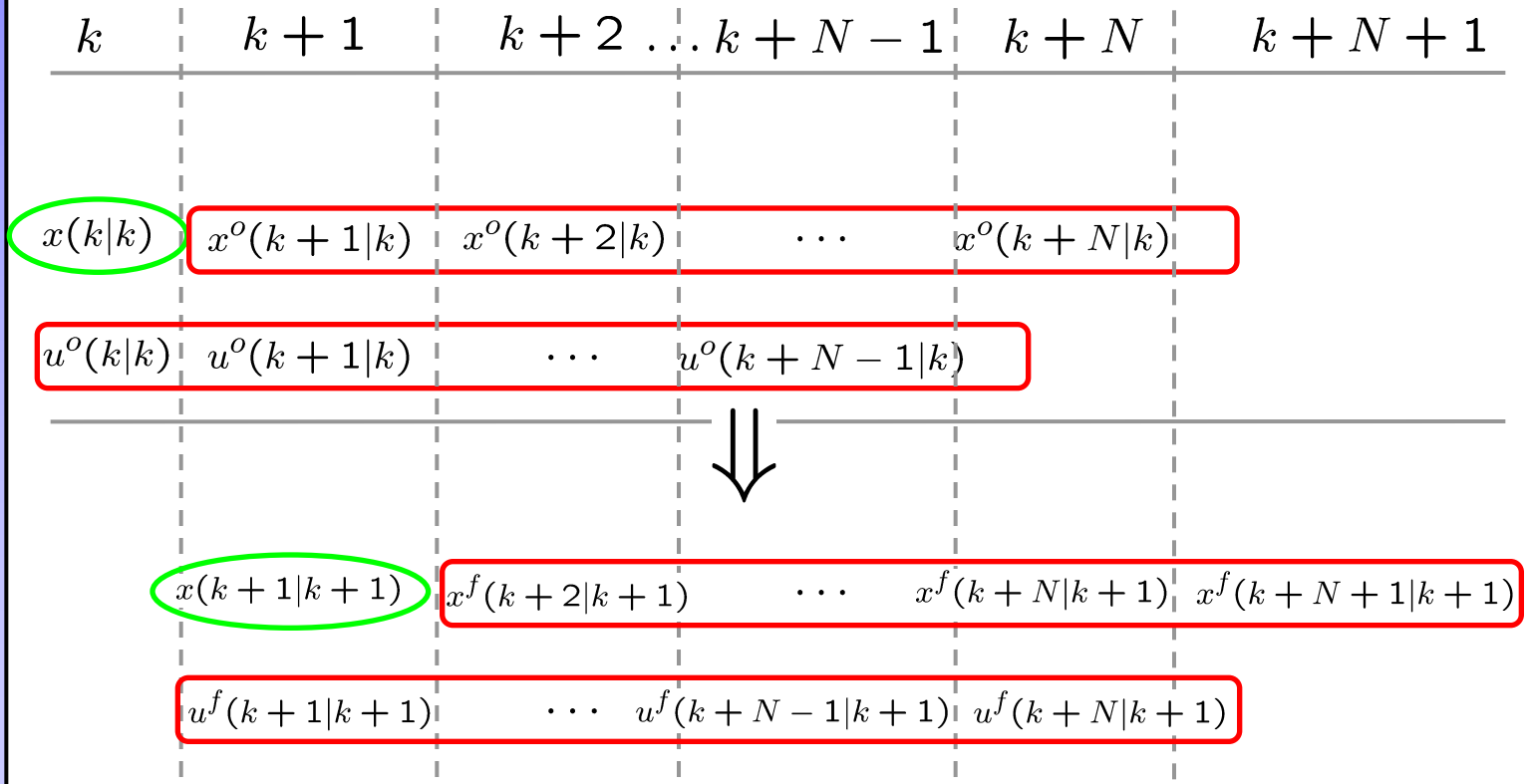
recursive feasibility (*terminal constraint*)

Lyapunov stability (*terminal cost*)

bert.pluymers@esat.kuleuven.be

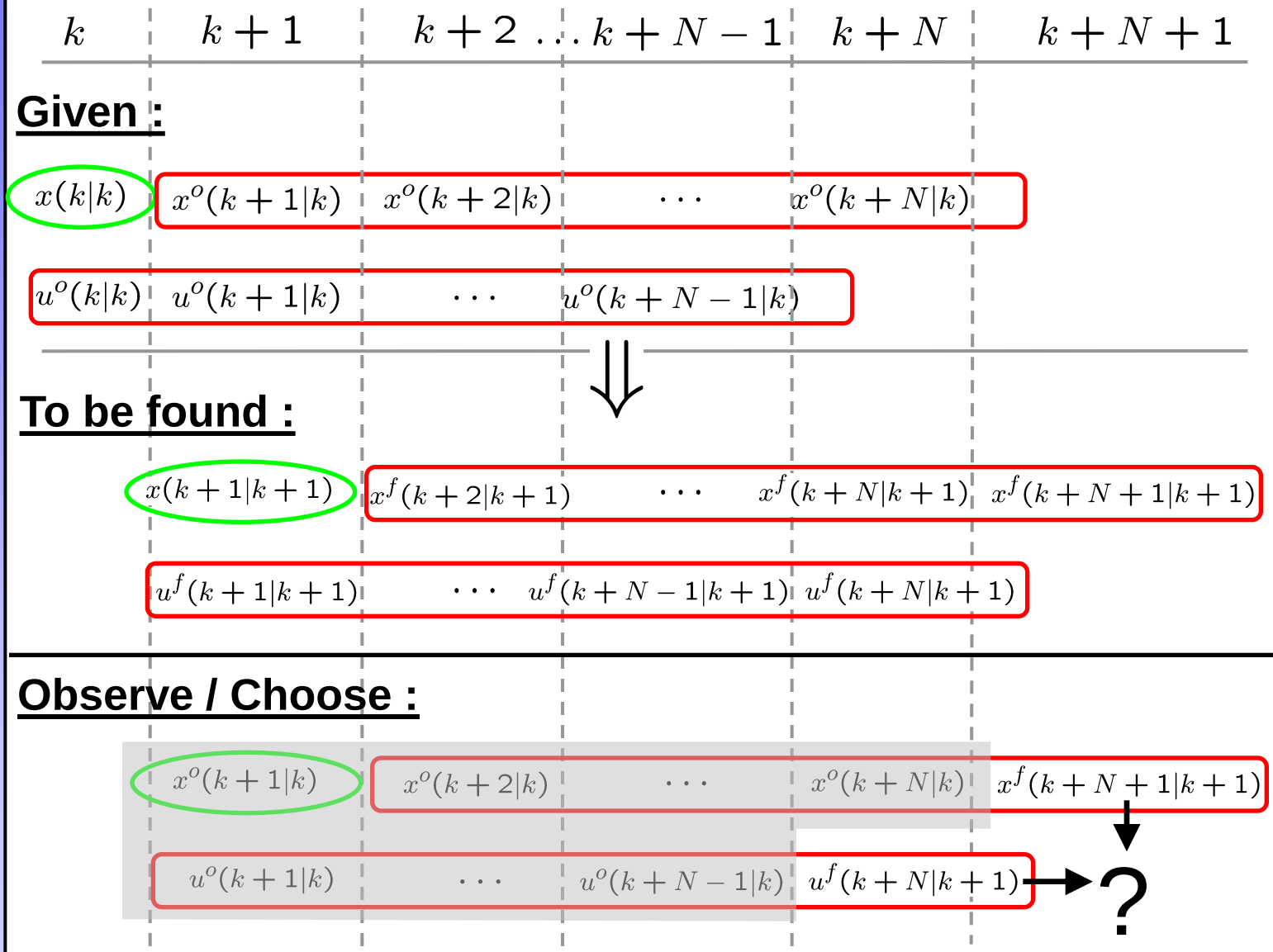
Step 1 : Recursive Feasibility

Problem : given the *optimal* (and hence feasible) solution to the optimization at time k, construct a *feasible* solution for the optimization at time $k+1$.



bert.pluymers@esat.kuleuven.be

Step 1 : Recursive Feasibility



Step 1 : Recursive Feasibility

Plant state at time $k+1$ predicted at time k :

$$x^o(k+1|k) = f_{\text{pred}}(x(k|k), u^o(k|k))$$

Real plant state at time $k+1$:

$$x(k+1|k+1) = f_{\text{plant}}(x(k|k), u(k))$$

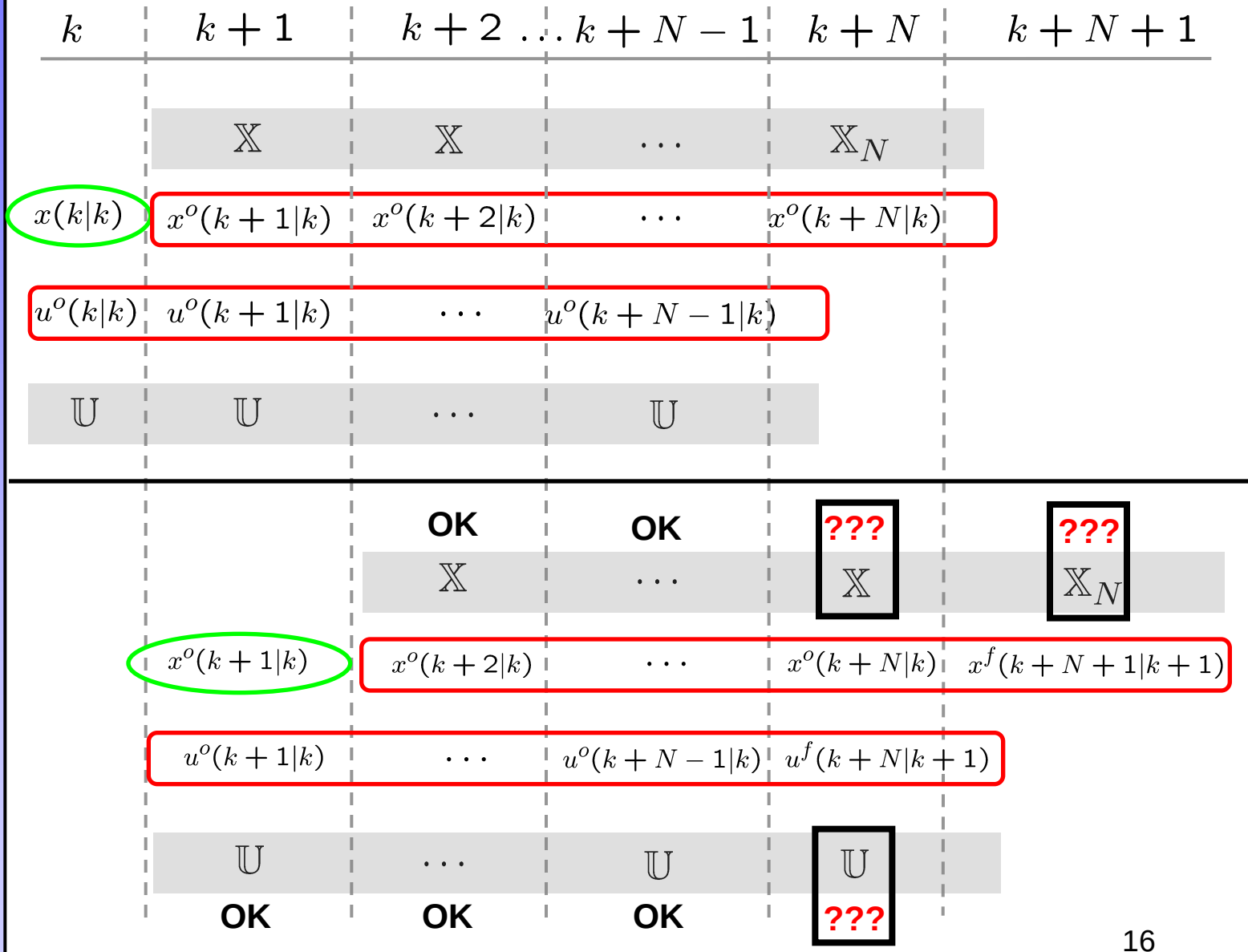
Assumption : No plant $\leftrightarrow$ model mismatch, i.e. $f_{\text{plant}} \equiv f_{\text{pred}}$

$$\Rightarrow \qquad x(k+1|k+1) \equiv x^o(k+1|k)$$

Hence, reusing the overlapping part of the input sequence will also result in an identical state sequence

bert.pluymers@esat.kuleuven.be

Step 1 : Recursive Feasibility



bert.pluymers@esat.kuleuven.be

Step 1 : Recursive Feasibility

Condition 1 : $x^o(k+N|k) \in \mathbb{X}_N \quad \Rightarrow \quad x^o(k+N|k) \in \mathbb{X}$

$\Rightarrow$ Satisfied if $\mathbb{X}_N \subseteq \mathbb{X}$

Condition 2 : $u^f(k+N|k+1) \in \mathbb{U}$

Condition 3 : $x^f(k+N+1|k+1) \in \mathbb{X}_N$

How to choose $u^f(k+N|k+1) \in \mathbb{U}$
and $x^f(k+N+1|k+1) \in \mathbb{X}_N$?

bert.pluymers@esat.kuleuven.be

Step 1 : Recursive Feasibility

How to choose $u^f(k+N|k+1)$ and $x^f(k+N+1|k+1)$?

Assume we know a locally stabilizing controller κ_N :

$$u_k = \kappa_N(x_k)$$

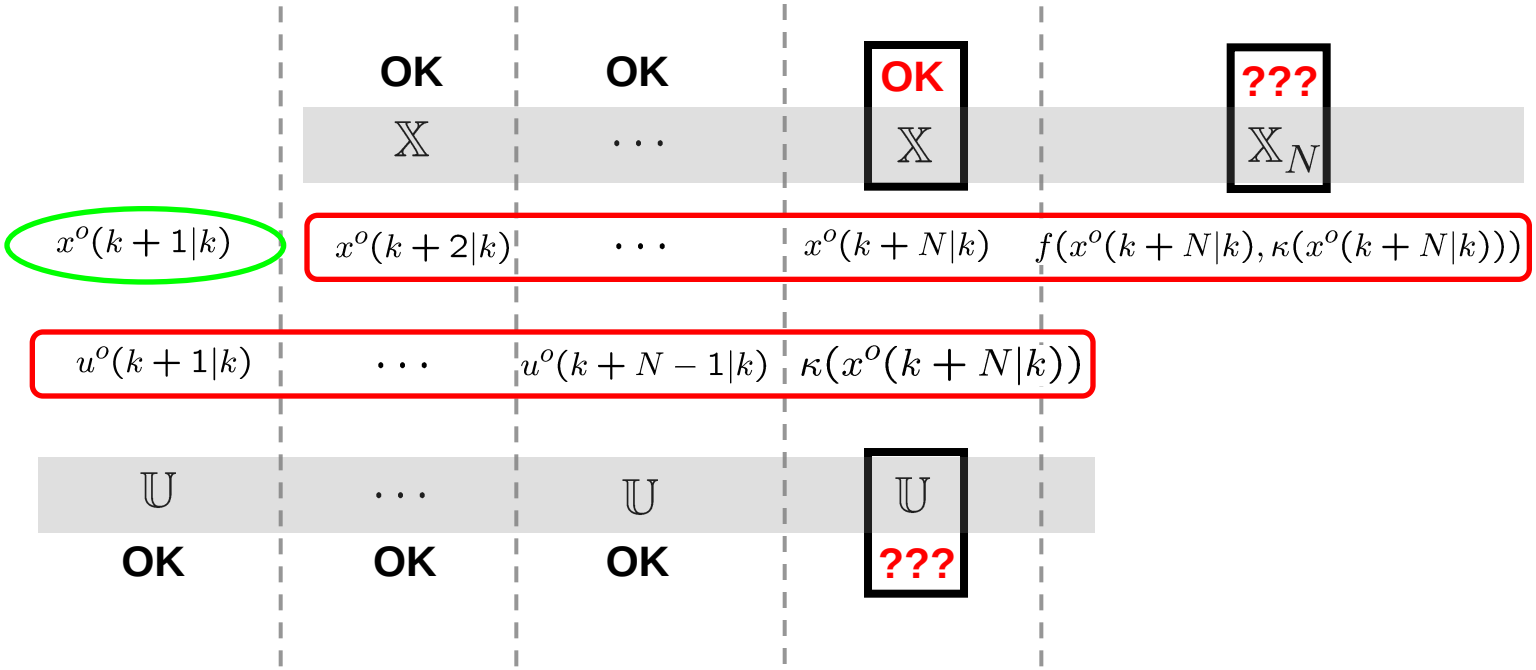
i.e. such that $x_{k+1} = f(x_k, \kappa_N(x_k))$ is locally stable.

Then choose

$$u^f(k+N|k+1) = \kappa(x^o(k+N|k))$$
$$x^f(k+N+1|k+1) = f(x^o(k+N|k), \kappa(x^o(k+N|k)))$$

bert.pluymers@esat.kuleuven.be

Step 1 : Recursive Feasibility



Condition 2 : $\kappa(x^o(k+N|k)) \in \mathbb{U}$

Condition 3 : $f(x^o(k+N|k), \kappa(x^o(k+N|k))) \in \mathbb{X}_N$

Step 1 : Recursive Feasibility

Condition 2 : $\kappa(x^o(k+N|k)) \in \mathbb{U}$

Condition 3 : $f(x^o(k+N|k), \kappa(x^o(k+N|k))) \in \mathbb{X}_N$

Since we know that $x^o(k+N|k)) \in \mathbb{X}_N$ ...

Condition 2 is satisfied if

$$\kappa_N(x) \in \mathbb{U}, \qquad \forall x \in \mathbb{X}_N$$

Condition 3 is satisfied if

$$f(x, \kappa_N(x)) \in \mathbb{X}_N, \qquad \forall x \in \mathbb{X}_N$$

Step 1 : Recursive Feasibility

Summary & Interpretation

Recursive feasibility is guaranteed if

1) $\mathbb{X}_N \subseteq \mathbb{X}$ → **Terminal constraint is feasible w.r.t state constraints**

2) $\kappa_N(x) \in \mathbb{U}, \quad \forall x \in \mathbb{X}_N$ → **Terminal constraint is feasible w.r.t input constraints**

3) $f(x, \kappa_N(x)) \in \mathbb{X}_N, \quad \forall x \in \mathbb{X}_N$ → **Terminal constraint is a *positive invariant set* w.r.t κ_N**

Step 2 : Lyapunov stability

- **Lyapunov stability :**

  If $\exists V(x)$ such that for some region $\mathbb{X}_f$ around 0

$$V(x_{\text{next}}) < V(x), \forall x \in \mathbb{X}_f \backslash 0,$$
$$V(0) = 0,$$

  then all trajectories starting within $\mathbb{X}_f$ asymptotically evolve towards 0.

- **In the MPC context :**
  - $\mathbb{X}_f$ is chosen as the feasible region,
  - $V(x)$ is chosen as the optimal cost value of the MPC optimization problem for the given $x \in \mathbb{X}_f$.

Under which conditions is $V(x)$ a Lyapunov function ???

bert.pluymers@esat.kuleuven.be

Step 2 : Lyapunov stability

Under which conditions is $V(x)$ a Lyapunov function ???

We have to prove that $V(x(k+1)) < V(x(k)), \forall x(k) \neq 0$

Or in other words that

$$J(x(k+1), \mathbf{x}_N^o(k+1), \mathbf{u}_N^o(k+1)) < J(x(k), \mathbf{x}_N^o(k), \mathbf{u}_N^o(k))$$

This is satisfied if

$$J(x(k+1), \mathbf{x}_N^f(k+1), \mathbf{u}_N^f(k+1)) < J(x(k), \mathbf{x}_N^o(k), \mathbf{u}_N^o(k))$$

bert.pluymers@esat.kuleuven.be

Step 2 : Lyapunov stability

Special relationship between the two cost expressions :

$$J(x(k+1), \mathbf{x}_N^f(k+1), \mathbf{u}_N^f(k+1)) =$$
$$J(x(k), \mathbf{x}_N^o(k), \mathbf{u}_N^o(k))$$
$$+ F_N(f(x^o(k+N|k), \kappa_N(x^o(k+N|k))))$$
$$- F_N(x^o(k+N|k))$$
$$+ l(x^o(k+N|k), \kappa_N(x^o(k+N|k)))$$
$$- l(x(k), u(k))$$

should be < 0

Satisfied if

$$F_N(x^o(k+N|k)) - F_N(f(x^o(k+N|k), \kappa_N(x^o(k+N|k)))) \geq l(x^o(k+N|k), \kappa_N(x^o(k+N|k)))$$

Condition 4 : $F_N(x) - F_N(x, \kappa_N(x)) \geq l(x, \kappa_N(x)), \quad \forall x \in \mathbb{X}_N$

Summary & Interpretation

Recursive feasibility and asymptotic stability is guaranteed if

1) $\mathbb{X}_N \subseteq \mathbb{X}$ → **Terminal constraint is feasible w.r.t state constraints**

2) $\kappa_N(x) \in \mathbb{U}, \quad \forall x \in \mathbb{X}_N$ → **Terminal constraint is feasible w.r.t input constraints**

3) $f(x, \kappa_N(x)) \in \mathbb{X}_N, \quad \forall x \in \mathbb{X}_N$ → **Terminal constraint is a *positive invariant set* w.r.t κ_N**

4) $F_N(x) - F_N(x, \kappa_N(x)) \geq l(x, \kappa_N(x)), \quad \forall x \in \mathbb{X}_N$

↳ **Lyapunov inequality**

i.e. $F_N(x)$ should 'overbound' cost of terminal controller

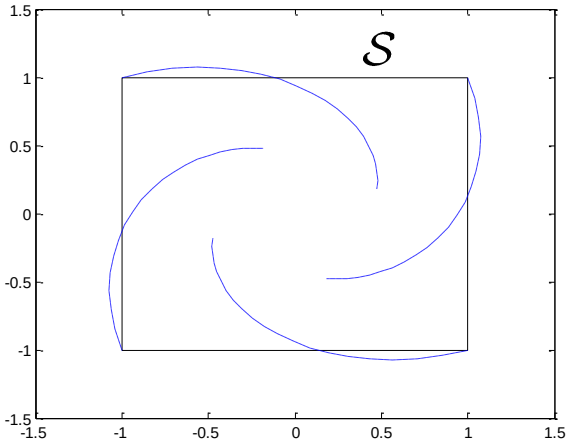
Iff the optimization problem is feasible at time $k = 0$!!!!!

- conditions are sufficient, but not necessary
- $\kappa_N(\cdot)$ is only used implicitly !

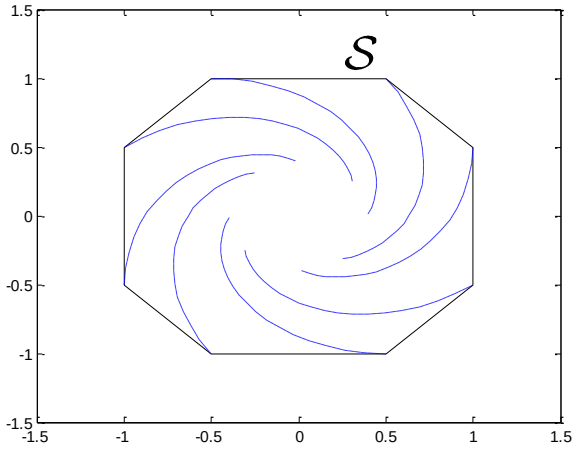
bert.pluymers@esat.kuleuven.be

Set Invariance

"Given an autonomous dynamical system, then a set $\mathcal{S}$ is (positive) invariant if it is guaranteed that if the current state lies within $\mathcal{S}$, all future states will also lie within $\mathcal{S}$."



not invariant

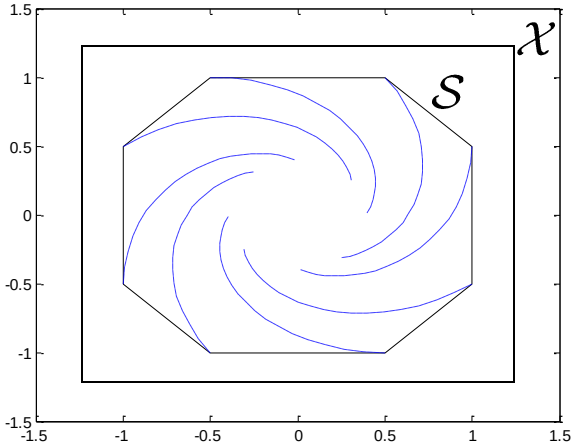


invariant

Set Invariance

- Useful tool for analysis of controllers for constrained systems
- Example :
  - linear system $x_{k+1} = f(x_k, u_k),$
  - linear controller $u_k = \kappa(x_k),$
  - state constraints $x_k \in \mathcal{X},$

$$\left.\begin{array}{l} x_{k+1} = f(x_k, u_k), \\ u_k = \kappa(x_k), \end{array}\right\} \quad x_{k+1} = f(x_k, \kappa(x_k)),$$



'feasible region' of closed loop system

bert.pluymers@esat.kuleuven.be

Set Invariance

Consider an autonomous time-invariant system as defined previously

$$x_{k+1} = f_{aut}(x_k),$$
$$A_x x_k \leq b_x, \qquad k = 0, \ldots, \infty.$$

A set $\mathcal{S}$ is …

… **invariant** iff $\quad f_{aut}(x) \in \mathcal{S}, \qquad \forall x \in \mathcal{S},$

… **feasible** iff $\quad \mathcal{S} \subseteq \mathcal{X} \triangleq \{x | A_x x \leq b_x\}.$

Problem :

Given an autonomous dynamical system subject to state constraints, find the feasible invariant set $\mathcal{S}$ of maximal size.

bert.pluymers@esat.kuleuven.be

Invariant sets for LTI systems

(Gilbert et al.,1991, IEEE TAC)

Given an L**TI system subject to linear constraints**

$$x_{k+1} = \Phi x_k, \qquad k = 0, \ldots, \infty,$$
$$A_x x_k \leq b_x, \qquad k = 0, \ldots, \infty,$$

then the largest size feasible invariant set can be found as

$$\mathcal{S} = \{x | A_x x \leq b_x, \quad A_x \Phi x \leq b_x, \quad \ldots, \quad A_x \Phi^n x \leq b_x\},$$

with n a finite integer.

- $\mathcal{S}$ is constructed by simple forward prediction
- $\mathcal{S}$ can be proven to be the largest feasible invariant set
- $\mathcal{S}$ is called the **Maximal Admissible Set (MAS)**

bert.pluymers@esat.kuleuven.be

Implementations

How to choose $\mathbb{X}_N, \kappa_N(\cdot), F_N(\cdot)$ such that conditions are satisfied ?

Different possibilities, depending of

- type of system (linear, non-linear)
- stability of the system
- presence of state constraints
- horizon length
- time constraints during design !

bert.pluymers@esat.kuleuven.be

Implementations

Terminal equality constraint :

- $\mathbb{X}_N = 0$
- $F_N(\cdot), \kappa(\cdot)$ irrelevant, thus omitted

All conditions trivially satisfied !

Implementations

Terminal equality constraint

PRO :

- straightforward method, no tuning required

CON :

- performance issues
- infeasibilities !!!

  $\rightarrow$ large N may be required

Implementations

Linear, stable, input-constrained, state-unconstrained systems, quadratic cost :

- $\mathbb{X}_N = \mathbb{X} = \mathbb{R}^{n_x}$
- $\kappa(x) = 0$
- $F_N(x) = x^T S x$ with $S = \sum_{i=0}^{\infty} (A^i)^T Q A^i$

$\rightarrow$ Choice of $\mathbb{X}_N$ automatically satisfies 1) and 3)
$\rightarrow$ Choice of $\kappa(\cdot)$ satisfies 2)
$\rightarrow$ $F_N(\cdot)$ *exactly* satisfies 4)

$\rightarrow$ similar to $M >> N$

bert.pluymers@esat.kuleuven.be

Implementations

Non-linear, constrained systems

$\longrightarrow$ Terminal cost AND Terminal constraint set

- Choose a locally stabilizing controller $\kappa(\cdot)$
- Use the value function $F(\cdot)$, associated with the controller as terminal cost $F_N(\cdot)$.
- Use a level set of $F(\cdot)$ as a terminal constraint set (small enough in order to satisfy 1) and 2))

No generic methods available to compute this for all types of systems, constraints and costs !!!

bert.pluymers@esat.kuleuven.be

Implementations

Linear, constrained systems, quadratic cost

$\rightarrow$ Things become much simpler !!!

- Calculate $\kappa(\cdot)$ as an LQR-controller with identical Q and R

  $\rightarrow$ guaranteed stabilizing *around origin*

- The value function reduces to $F(x) = x^T S x$, where S comes out of the DARE

  $\rightarrow$ Condition 4) automatically satisfied

- Use an *invariant set* of terminal closed loop system as X_N

  $\rightarrow$ more on this later

bert.pluymers@esat.kuleuven.be

Implementations

Linear, constrained systems, quadratic cost

$\rightarrow$ Terminal cost AND Terminal constraint set

PRO

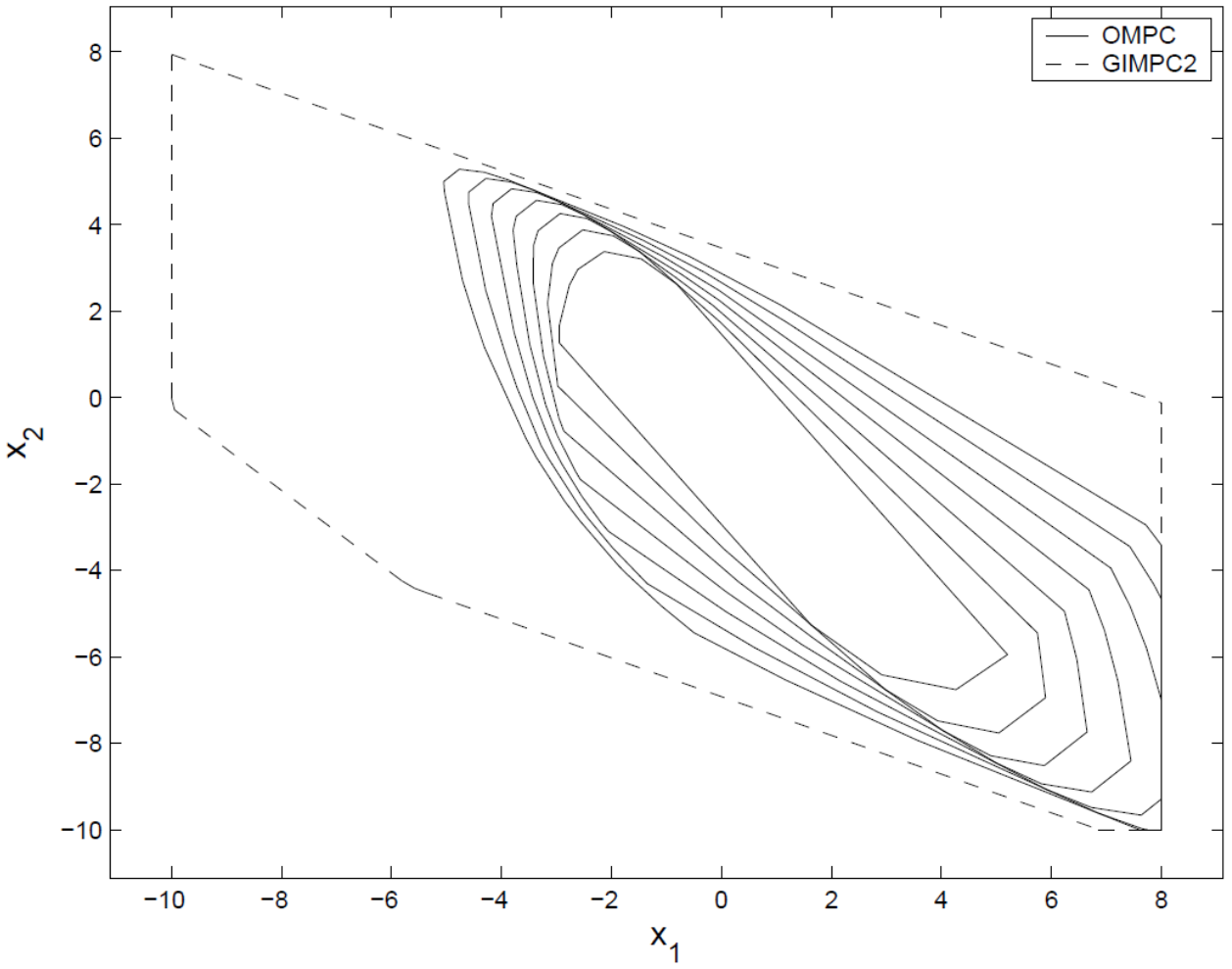
- improved performance
- also for unstable and constrained systems

CON

- initial infeasibilities may still occur
- X_N might turn out to be very small
  (e.g. if R ánd $\mathbb{U}$ small)

bert.pluymers@esat.kuleuven.be

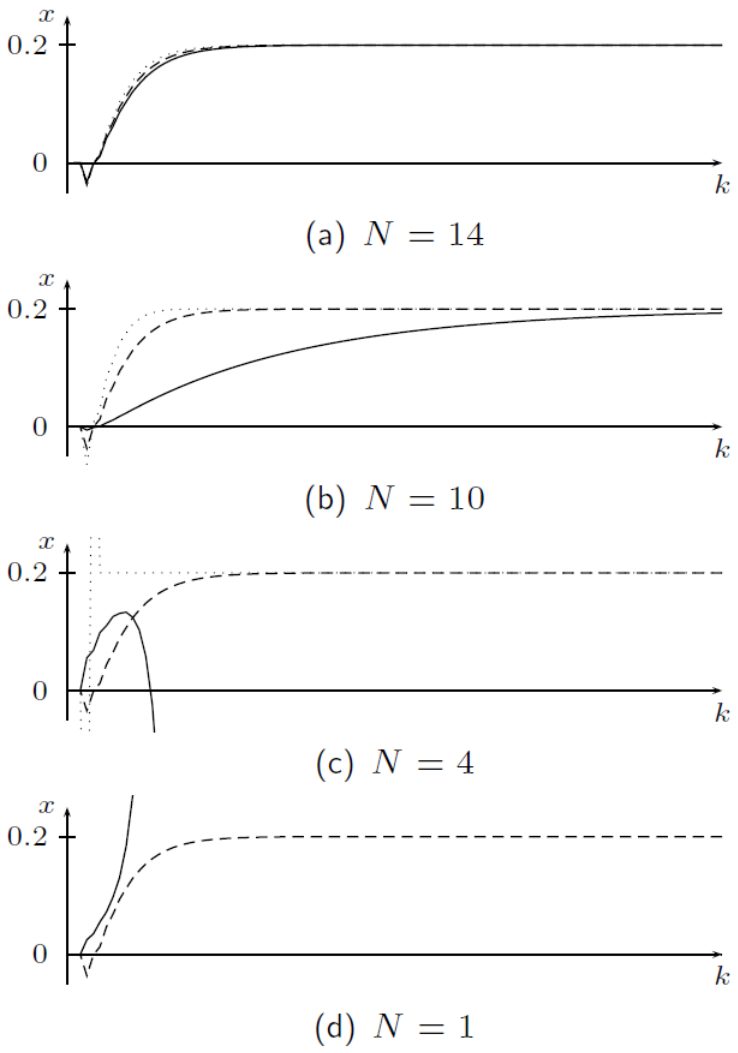
Implementations



Terminal constraint set determines feasible region

bert.pluymers@esat.kuleuven.be

Implementations



Solid : standard MPC, **dashed** : terminal cost, constraint, **dotted** : terminal equality constr.

Conclusion

- stability of standard MPC not guaranteed
- pole/zero analysis impossible
  - recursive feasibility
  - Lyapunov stability
- general stability framework for stabilization problems
- different implementations
- stability measures allow the use of shorter horizons

References

[1] F. Blanchini. Set invariance in control. *Automatica*, 35:1747–1767, 1999.

[2] E. G. Gilbert and K. T. Tan. Linear systems with state and control constraints: the theory and application of maximal output admissable sets. *IEEE Transactions on Automatic Control*, 36(9):1008–1020, 1991.

[3] E. Kerrigan. *Robust Constraint Satisfaction: Invariant Sets and Predictive Control*. PhD thesis, Cambridge, 2000.

[4] M. Kothare, V. Balakrishnan, and M. Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32:1361–1379, 1996.

[5] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

H0K03a : Advanced Process Control

Model-based Predictive Control 4 : Robustness

Bert Pluymers
Prof. Bart De Moor

Katholieke Universiteit Leuven, Belgium
Faculty of Engineering Sciences
Department of Electrical Engineering (ESAT)
Research Group SCD-SISTA

Overview

Lecture 4 : Robustness

- Example
- Robustness
- Robust MPC
- Conclusion

bert.pluymers@esat.kuleuven.be

Example

Linear state-space system of the form

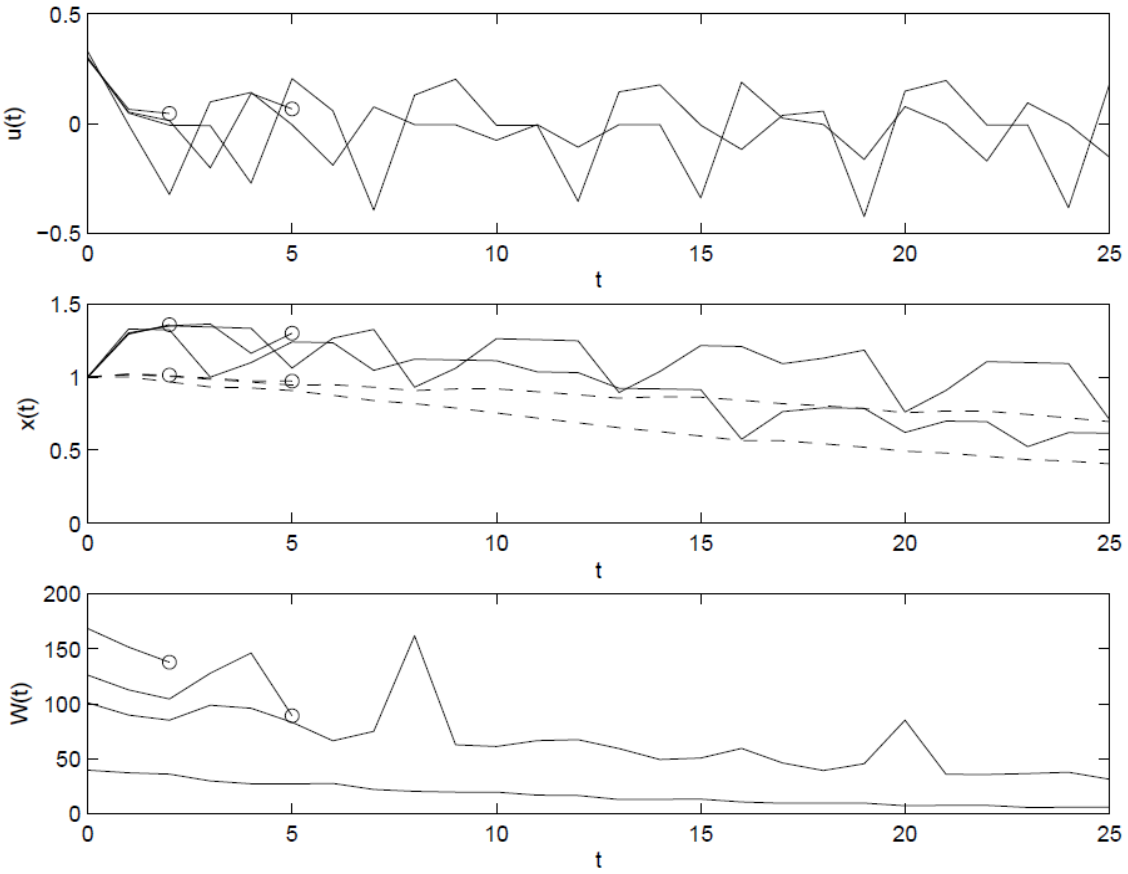
$$x_{k+1} = \begin{bmatrix} 1 & 0 \\ -0.1 - 0.2c_k & 1.1 + 0.2c_k \end{bmatrix} x_k + \begin{bmatrix} 1 - 0.8c_k \\ 0 \end{bmatrix} u_k.$$

with bounded parametric uncertainty $c_k \in [0, 1]$.

Aim : steer this system towards the origin from initial state $x_0 = [1; 1]$ without violating the constraint $-1 \leq u_k \leq 1$.

Example

Results for 4 different parameter settings :



- **Recursive feasibility ?**
- **Monotonicity of the cost ?**

Robustness

Robust with respect to what ?

- Disturbances

$$x_{k+1} = f_{\text{pred}}(x_k, u_k) + w_k$$

Cause predictions of
'nominal' MPC to be inaccurate

- Model uncertainty

$$\begin{aligned} x_{k+1} &= f_{\text{plant}}(x_k, u_k), \\ &= f_{\text{pred}}(x_k, u_k) + f_{\text{diff}}(x_k, u_k), \end{aligned}$$

bert.pluymers@esat.kuleuven.be

Robustness

Main aims :

- Keep recursive feasibility properties, despite model errors, disturbances
- Keep asymptotic stability (in the case without disturbances)

We need to have an idea about …

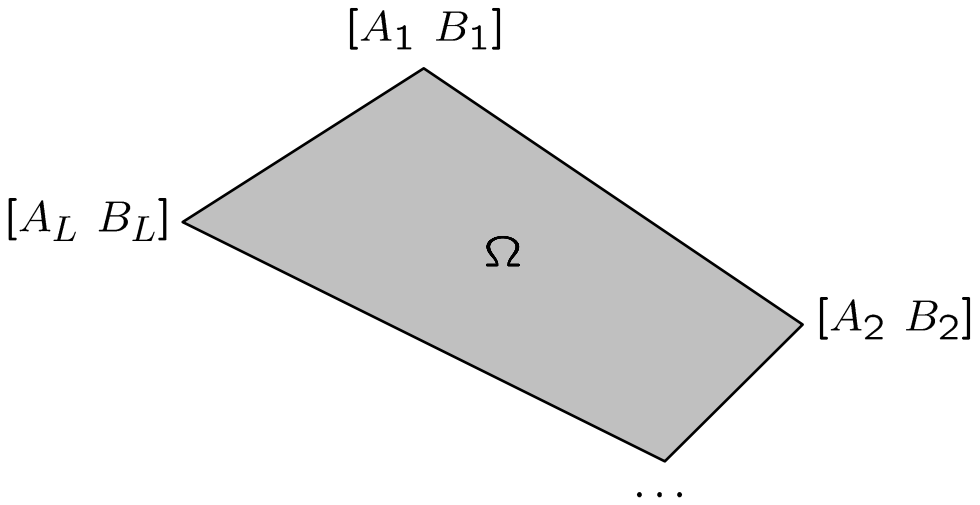
- the size of the model uncertainty
- the size of the disturbances

Uncertain Models

Linear Parameter-Varying state space models with polytopic uncertainty description

$$x_{k+1} = A(k)x_k + B(k)u_k,$$

$$[A(k)\ \ B(k)] \in \Omega \triangleq \mathrm{Co}\{[A_1\ \ B_1], \ldots, [A_L\ \ B_L]\}$$



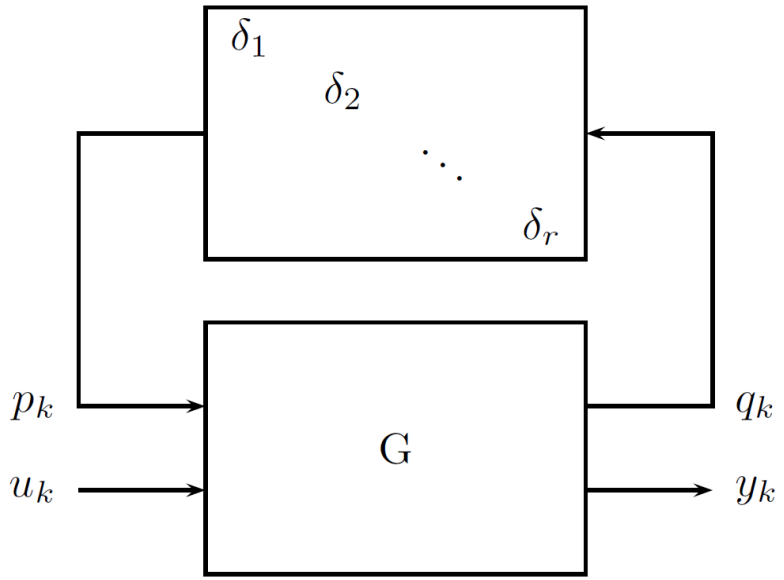
bert.pluymers@esat.kuleuven.be

Uncertain Models

Linear Parameter-Varying state space models with norm-bounded uncertainty description

$$x_{k+1} = A(k)x_k + B(k)u_k,$$

$$\begin{cases} A(k) = A_0 + B_p \Delta_k C_q \\ B(k) = B_0 + B_p \Delta_k D_{qu} \end{cases} \quad \text{with} \quad \|\Delta_k\| \leq 1$$



bert.pluymers@esat.kuleuven.be

Bounded Disturbances

- Typically bounded by a polytope : $w_k \in \mathcal{W}$
- Can be described in two ways
  - $\mathcal{W} = \text{Co}\{w_1, \ldots, w_n\}$
  - $\mathcal{W} = \{w | A_w w \leq 1_v\}$

- Trivial condition for well-posedness :

$$\mathcal{W} \subseteq \mathcal{X}$$

Robust MPC

Main aims :

- Keep recursive feasibility properties, despite model errors, disturbances
- Keep asymptotic stability (in the case without disturbances)

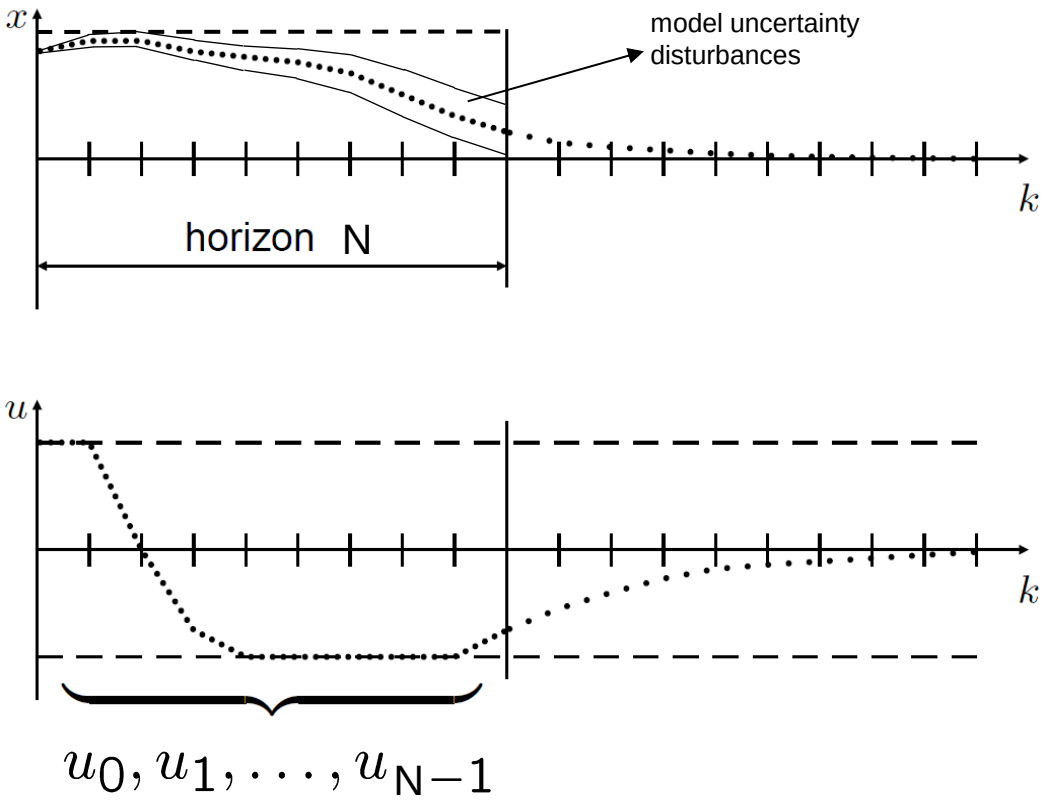
Necessary modifications :

- *Uncertain* predictions (e.g predictions with all models within uncertainty region)
  - worst-case constraint satisfaction over all predictions
  - worst-case cost over all predictions
- Terminal cost has to satisfy *multiple* Lyap. Ineq.
- Terminal constraint has to be a *robust* invariant set

bert.pluymers@esat.kuleuven.be

Robust MPC

Uncertain predictions :



bert.pluymers@esat.kuleuven.be

Uncertain Predictions

Step 1) Robust Constraint Satisfaction

$$x(k+i|k) \in \mathcal{X}, \qquad \begin{cases} \forall [A(k+j)\ B(k+j)] \in \Omega, & j = 0, \ldots, i-1, \\ i = 1, \ldots, N. \end{cases}$$

Observations :

- $x(k+i|k)$ depends linearly on $[A(k+j)\ B(k+j)], j = 0, \ldots, i-1,$
- Ω is a convex polytopic set
- $\mathcal{X}$ is a convex set

Result : Sufficient to impose constraint only for vert. of Ω :

$$x(k+i|k) \in \mathcal{X}, \qquad \begin{cases} \forall [A(k+j)\ B(k+j)] \in \{[A_1\ B_1], \ldots, [A_L\ B_L]\}, \\ \qquad\qquad j = 0, \ldots, i-1, \\ i = 1, \ldots, N. \end{cases}$$

Uncertain Predictions

LTI
(L=1)

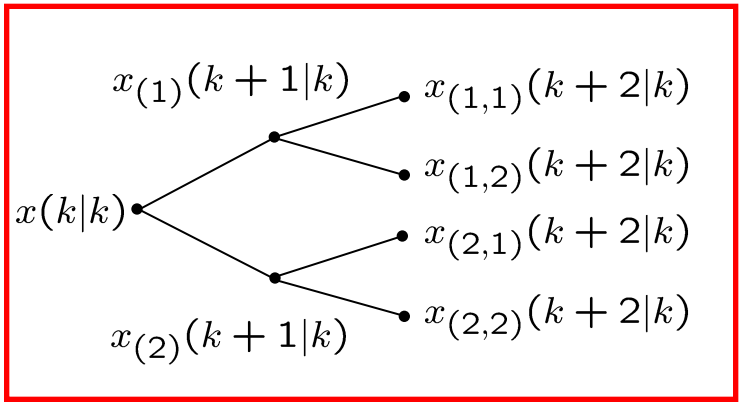
$$[A(k)\ \ B(k)] \equiv [A\ \ B]$$

$x(k|k)$ $x(k+1|k)$ $x(k+2|k)$

$u(k|k)$ $u(k+1|k)$

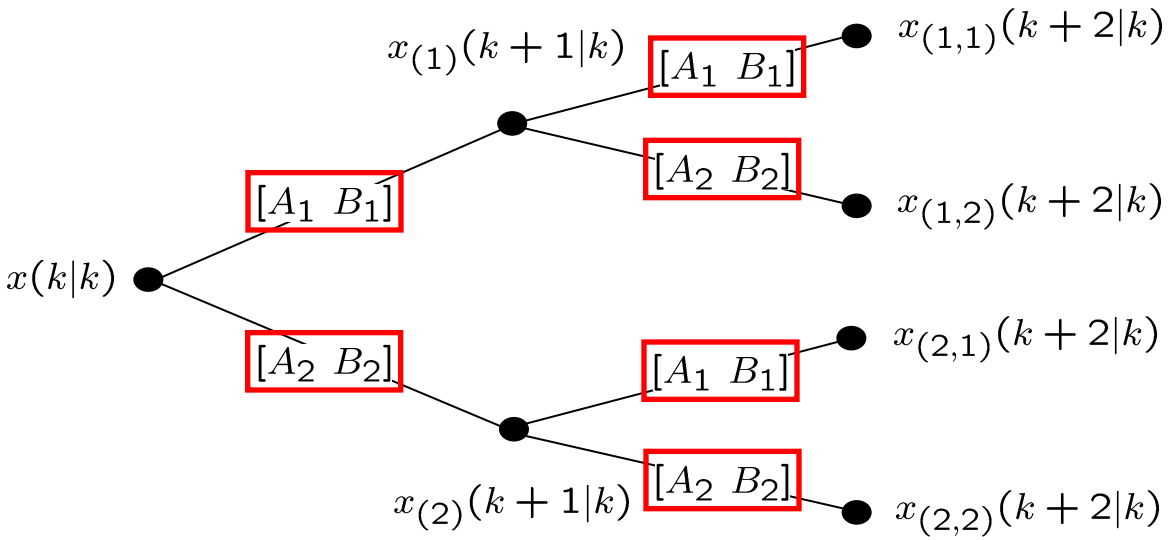
LPV
(L>1, e.g. 2)

$$[A(k)\ \ B(k)] \in \Omega$$



$u(k|k)$ $u(k+1|k)$

Uncertain Predictions



Impose state constraints on all nodes of state prediction tree

→ number of constraints increases expon. with incr. N !!!

Worst-Case Cost Objective

Step 2) Worst-Case cost minimization

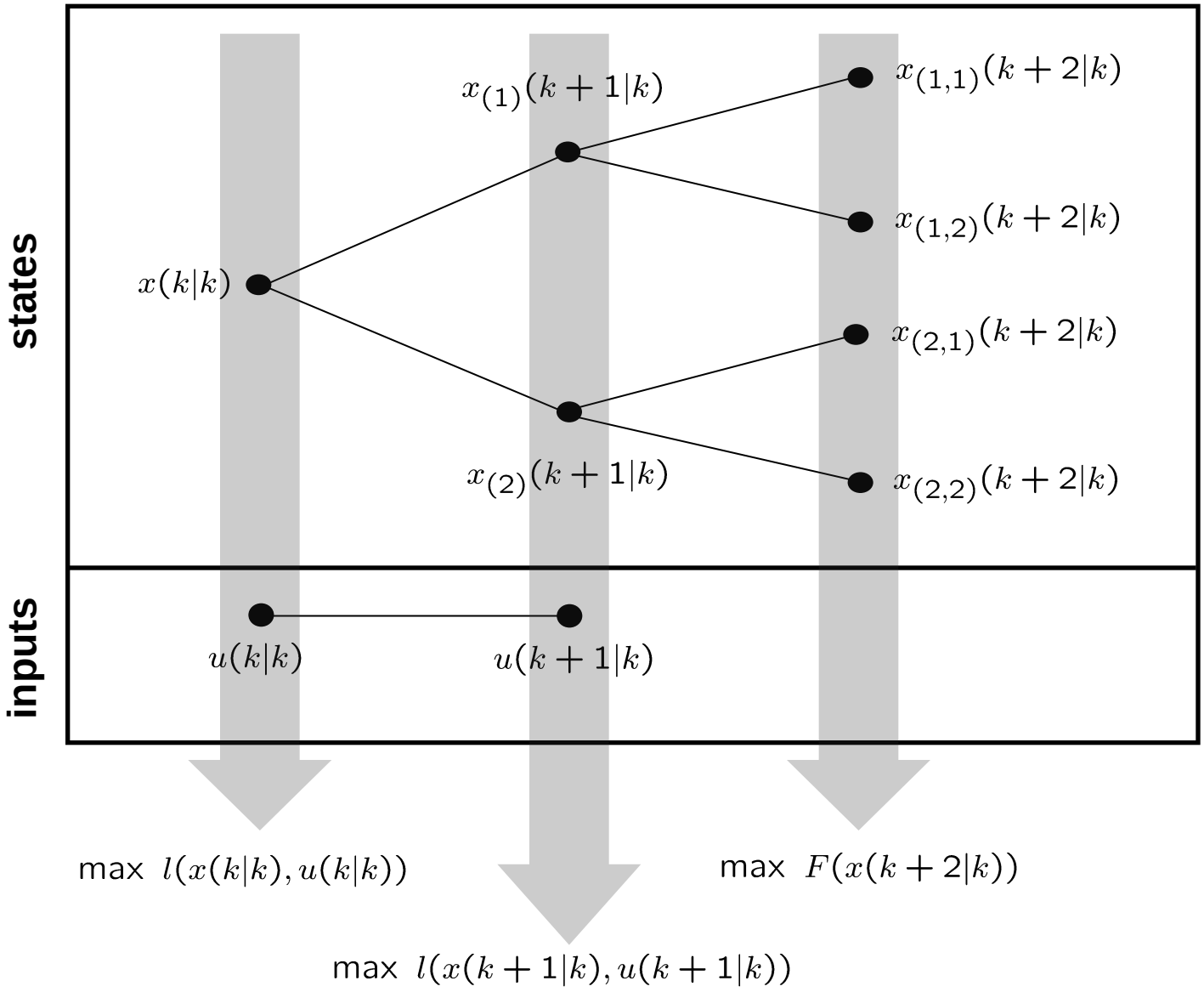
$$\min_{\mathbf{x}_N(k),\mathbf{u}_N(k)} \quad \max_{\substack{[A(k+j|k)\ \ B(k+j|k)]\in\Omega \\ j=0,\ldots,N-1}} \left(\sum_{i=0}^{N-1} l(x(k+i|k),u(k+i|k)) + F(x(k+N|k)) \right)$$

Observations :

- $x(k+i|k)$ depends linearly on $[A(k+j)\ \ B(k+j)], j=0,\ldots,i-1,$
- Ω is a convex polytopic set
- cost function typically convex function of $u(k+i|k), x(k+i|k)$

→ **Also for objective function sufficient to make predictions only with vertices of uncertainty polytope**

Worst-Case Cost Objective



Worst-Case Cost Objective

(1-norm)

$$\min_{\mathbf{x}_N(k),\mathbf{u}_N(k)} \quad \max_{\substack{[A(k+j|k)\ \ B(k+j|k)]\in\{[A_1\ \ B_1],\ldots,[A_L\ \ B_L]\} \\ j=0,\ldots,N-1}} \left(\sum_{i=0}^{N-1} (q\|x(k+i|k)\|_1 + r\|u(k+i|k)\|_1) + q_N\|x(k+N|k)\|_1 \right)$$

$$\Downarrow$$

$$\min_{\mathbf{x}_N(k),\mathbf{u}_N(k),\overline{x}_i,\overline{u}_i} \sum_{i=0}^{N-1} (q\overline{x}_i + r\overline{u}_i) + q_F\overline{x}_N$$

LP

$$\text{s.t.} \quad \|x_{(j_1,\ldots,j_i)}(k+i|k)\|_1 \le \overline{x}_i, \quad \begin{cases} j_{1,\ldots,i} = 1,\ldots,L, \\ i = 0,\ldots,N, \end{cases}$$

$$\|u(k+i|k)\|_1 \le \overline{u}_i, \quad i = 0,\ldots,N-1.$$

Worst-Case Cost Objective

(2-norm)

$$\min_{\mathbf{x}_N(k),\mathbf{u}_N(k)} \quad \max_{\substack{[A(k+j|k)\ \ B(k+j|k)]\in\{[A_1\ \ B_1],\ldots,[A_L\ \ B_L]\} \\ j=0,\ldots,N-1}} \left(\sum_{i=0}^{N-1} (\|x(k+i|k)\|_Q^2 + \|u(k+i|k)\|_R^2) + \|x(k+N|k)\|_{Q_N}^2 \right)$$

$$\Updownarrow$$

$$\min_{\mathbf{x}_N(k),\mathbf{u}_N(k),\overline{x}_i,\overline{u}_i} \sum_{i=0}^{N-1} (\overline{x}_i + \overline{u}_i) + \overline{x}_N$$

CVX ?

s.t.

$$\|x_{(j_1,\ldots,j_i)}(k+i|k)\|_Q^2 \le \overline{x}_i,$$

$$\|x_{(j_1,\ldots,j_N)}(k+N|k)\|_{Q_N}^2 \le \overline{x}_N,$$

$$\|u(k+i|k)\|_R^2 \le \overline{u}_i,$$

$$\begin{cases} j_{1,\ldots,i} = 1,\ldots,L, \\ i = 0,\ldots,N-1, \end{cases}$$

$$j_{1,\ldots,N} = 1,\ldots,L,$$

$$i = 0,\ldots,N-1.$$

bert.pluymers@esat.kuleuven.be

Worst-Case Cost Objective

(2-norm)

Constraints of the form :

$$\|T_x \mathbf{x}_N + T_u \mathbf{u}_N\|^2 \leq c,$$

CVX ?

$$\Updownarrow$$

$$\|2T_x \mathbf{x}_N + 2T_u \mathbf{u}_N\|^2 \leq 4c,$$

$$\Updownarrow$$

$$\|2T_x \mathbf{x}_N + 2T_u \mathbf{u}_N\|^2 + (c-1)^2 \leq 4c + (c-1)^2,$$

$$\Updownarrow$$

$$\|2T_x \mathbf{x}_N + 2T_u \mathbf{u}_N\|^2 + (c-1)^2 \leq (c+1)^2,$$

$$\Updownarrow$$

$$\left\| \begin{bmatrix} 2T_x \mathbf{x}_N + 2T_u \mathbf{u}_N \\ c-1 \end{bmatrix} \right\|^2 \leq (c+1)^2,$$

$$\Updownarrow$$

$$\left\| \begin{bmatrix} 2T_x \mathbf{x}_N + 2T_u \mathbf{u}_N \\ c-1 \end{bmatrix} \right\| \leq (c+1),$$

→ **SOC**

bert.pluymers@esat.kuleuven.be

Worst-Case Cost Objective

(2-norm)

$$\min_{\mathbf{x}_N(k),\mathbf{u}_N(k)} \quad \max_{\substack{[A(k+j|k)\ B(k+j|k)]\in\{[A_1\ B_1],\ldots,[A_L\ B_L]\} \\ j=0,\ldots,N-1}} \left(\sum_{i=0}^{N-1} (\|x(k+i|k)\|_Q^2 + \|u(k+i|k)\|_R^2) + \|x(k+N|k)\|_{Q_N}^2 \right)$$

$$\Updownarrow$$

$$\min_{\mathbf{x}_N(k),\mathbf{u}_N(k),\overline{x}_i,\overline{u}_i} \sum_{i=0}^{N-1} (\overline{x}_i + \overline{u}_i) + \overline{x}_N$$

SOCP

s.t.

$$\left\| \begin{bmatrix} Q^{\frac{1}{2}} x_{(j_1,\ldots,j_i)}(k+i|k) \\ \overline{x}_i - 1 \end{bmatrix} \right\| \leq \overline{x}_i + 1, \qquad \begin{cases} j_{1,\ldots,i} = 1,\ldots,L, \\ i = 0,\ldots,N-1, \end{cases}$$

$$\left\| \begin{bmatrix} Q_N^{\frac{1}{2}} x_{(j_1,\ldots,j_N)}(k+N|k) \\ \overline{x}_N - 1 \end{bmatrix} \right\| \leq \overline{x}_N + 1, \qquad j_{1,\ldots,N} = 1,\ldots,L,$$

$$\left\| \begin{bmatrix} R^{\frac{1}{2}} u(k+i|k) \\ \overline{u}_i - 1 \end{bmatrix} \right\| \leq \overline{u}_i + 1, \qquad i = 0,\ldots,N-1.$$

bert.pluymers@esat.kuleuven.be

Robust MPC

(2-norm)

By rewriting $x_{(j_1,\ldots,j_i)}(k+i|k) = G_{(j_1,\ldots,j_i)}\mathbf{u}_N + g_{(j_1,\ldots,j_i)}$ we now get

SOCP

$$\min_{\mathbf{u}_N(k),\overline{x}_i,\overline{u}_i} \sum_{i=0}^{N-1} (\overline{x}_i + \overline{u}_i) + \overline{x}_N$$

s.t.

$$\left\| \begin{bmatrix} Q^{\frac{1}{2}}(G_{(j_1,\ldots,j_i)}\mathbf{u}_N + g_{(j_1,\ldots,j_i)}) \\ \overline{x}_i - 1 \end{bmatrix} \right\| \leq \overline{x}_i + 1, \qquad \begin{cases} j_{1,\ldots,i} = 1,\ldots,L, \\ i = 0,\ldots,N-1, \end{cases}$$

$$\left\| \begin{bmatrix} Q_N^{\frac{1}{2}}(G_{(j_1,\ldots,j_N)}\mathbf{u}_N + g_{(j_1,\ldots,j_N)}) \\ \overline{x}_N - 1 \end{bmatrix} \right\| \leq \overline{x}_N + 1, \qquad j_{1,\ldots,N} = 1,\ldots,L,$$

$$\left\| \begin{bmatrix} R^{\frac{1}{2}}u(k+i|k) \\ \overline{u}_i - 1 \end{bmatrix} \right\| \leq \overline{u}_i + 1, \qquad i = 0,\ldots,N-1,$$

$$A_x(G_{(j_1,\ldots,j_i)}\mathbf{u}_N + g_{(j_1,\ldots,j_i)}) \leq b_x, \qquad \begin{cases} j_{1,\ldots,i} = 1,\ldots,L, \\ i = 0,\ldots,N-1, \end{cases}$$

$$A_{x_N}(G_{(j_1,\ldots,j_N)}\mathbf{u}_N + g_{(j_1,\ldots,j_N)}) \leq b_{x_N}, \qquad j_{1,\ldots,N} = 1,\ldots,L,$$

$$A_u u(k+i|k) \leq b_u, \qquad i = 0,\ldots,N-1.$$

Terminal constraint

Terminal cost

Robust Terminal Cost

"non-robust" stability condition for terminal cost:

$$F_N(x) - F_N(f(x, \kappa_N(x))) \geq l(x, \kappa_N(x)), \quad \forall x \in \mathbb{X}_N$$

In case of…

- LPV system with polytopic uncertainty
- linear feedback controller $u_k = \kappa_N(x_k) = -Kx_k$
- quadratic cost criterion $l(x, u) = x^T Q x + u^T R u$
- quadratic terminal cost $F_N(x) = x^T Q_N x$

… this becomes :

$$x^T Q_N x - x^T (A-BK)^T Q_N (A-BK) x \geq x^T Q x + x^T K^T R K x, \quad \forall x \in \mathbb{X}_N, \quad \forall [A\ B] \in \Omega$$

or equivalent :

$$Q_N - (A-BK)^T Q_N (A-BK) \succeq Q + K^T R K, \quad \forall [A\ B] \in \Omega$$

Robust Terminal Cost

Robust stability condition for terminal cost:

$$Q_N - (A - BK)^T Q_N (A - BK) \succeq Q + K^T R K, \qquad \forall [A \ B] \in \Omega$$

Observations :

- inequality is convex and linear in A and B (i.e. LMI in A, B)
- Ω is a convex polytopic set

Hence, inequality satisfied $\forall [A \ B] \in \Omega$ iff

$$Q_N - (A_i - B_i K)^T Q_N (A_i - B_i K) \succeq Q + K^T R K, \qquad i = 1, \ldots, L.$$

Robust Terminal Cost : Design

1. Find a robustly stabilizing controller

$$u_k = \kappa_N(x_k) = -Kx_k \quad :$$
$$lim_{k\to\infty}\|x_k\| = 0, \quad \forall [A(i)\ B(i)] \in \Omega, i \geq 0$$

2. Find a terminal cost satisfying

$$Q_N - (A_i - B_iK)^T Q_N (A_i - B_iK) \succeq Q + K^T R K, \qquad i = 1, \ldots, L.$$

by solving the following optimization problem :

$$\min_{Q_N} \sum_{i=1}^{n_x} c_i, \longrightarrow \text{Minimization of eigenvalues of } Q_N$$

$$\text{s.t.} \quad Q_N \preceq diag(c_1, \ldots, c_{n_x}),$$
$$Q_N - (A_i - B_iK)^T Q_N (A_i - B_iK) \succeq Q + K^T R K, \qquad i = 1, \ldots, L.$$

SDP

optimization variables

Robust Terminal Constraint

Reminder : nominal case

Recursive feasibility is guaranteed if

remain unchanged

1) $\mathbb{X}_N \subseteq \mathbb{X}$ → **Terminal constraint is feasible w.r.t state constraints**

2) $\kappa_N(x) \in \mathbb{U},$ $\forall x \in \mathbb{X}_N$ → **Terminal constraint is feasible w.r.t input constraints**

3) $f(x, \kappa_N(x)) \in \mathbb{X}_N,$ $\forall x \in \mathbb{X}_N$ → **Terminal constraint is a *positive invariant set* w.r.t κ_N**

Has to be modified in order to Model uncertainty into account → **Robust positive invariance**

Robust Terminal Constraint

Consider linear terminal controller $u_k = \kappa_N(x_k) = -Kx_k$,

then the resulting closed loop system is :

$$x_{k+1} = \Phi(k)x_k,$$

$$\Phi(k) \in \Omega' = \mathrm{Co}\{\Phi_1, \ldots, \Phi_L\}, \quad \Phi_i = (A_i - B_i K)$$

Robust positive invariance :

$$x \in \mathcal{S}, \quad \Rightarrow \quad \Phi x \in \mathcal{S}, \qquad \forall \Phi \in \Omega'$$

Again : sufficient to satisfy inclusion $\forall \Phi_i, i = 1, \ldots, L$

Robust Terminal Constraint

Reminder : invariant sets for LTI systems

Given an L**TI system subject to linear constraints**

$$x_{k+1} = \Phi x_k, \qquad k = 0, \ldots, \infty,$$
$$A_x x_k \leq b_x, \qquad k = 0, \ldots, \infty,$$

then the largest size feasible invariant set can be found as

$$\mathcal{S} = \{x | A_x x \leq b_x, \quad A_x \Phi x \leq b_x, \quad \ldots, \quad A_x \Phi^n x \leq b_x\},$$

with n a finite integer.

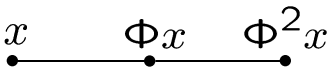
Comes down to making forward predictions using Φ

Robust Terminal Constraint

LTI
(L=1,n=2)

$$\Phi(k) \equiv \Phi$$

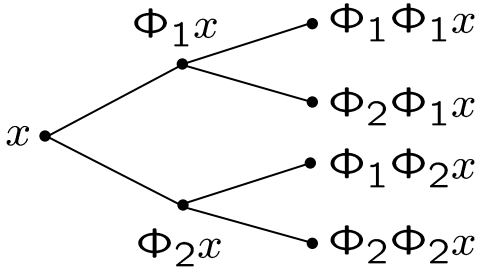
$$A_{\mathcal{S}} = \begin{bmatrix} A_x \\ A_x\Phi \\ A_x\Phi^2 \end{bmatrix}$$



LPV
(L>1, e.g. 2, n=2)

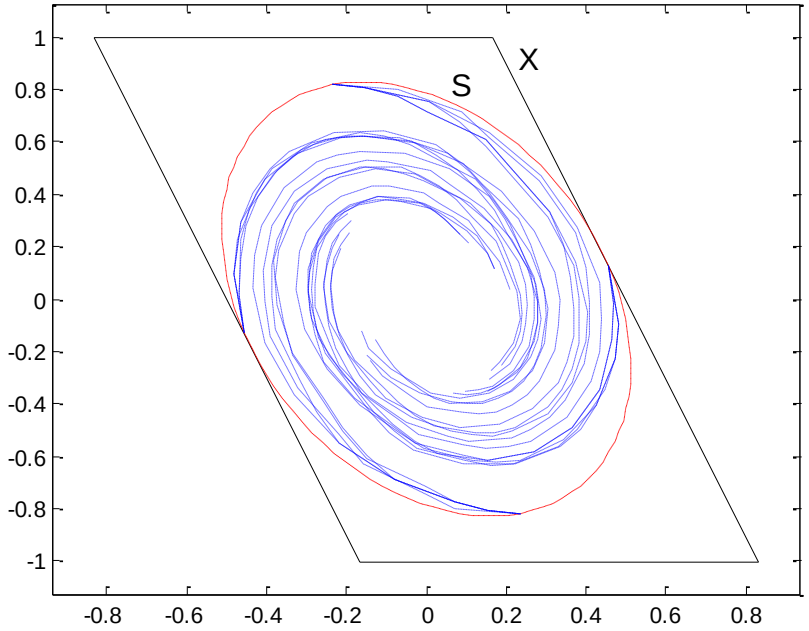
$$\Phi(k) \in \mathrm{Co}\{\Phi_1, \Phi_2\}$$

$$A_{\mathcal{S}} = \begin{bmatrix} A_x \\ A_x\Phi_1 \\ A_x\Phi_2 \\ A_x\Phi_1\Phi_1 \\ A_x\Phi_1\Phi_2 \\ A_x\Phi_2\Phi_1 \\ A_x\Phi_2\Phi_2 \end{bmatrix}$$



Ellipsoidal invariant sets for LPV systems

(Kothare et al.,1996, Automatica)



- Constructed by solving semi-definite program (SDP)
- Conservative with respect to constraints

Polyhedral invariant sets for LPV systems

Reformulate invariance condition :

A set $\mathcal{S}$ is invariant with respect to a system defined by Ω iff

$$\mathcal{S} \subseteq \mathcal{S}^{-},$$

with

$$\mathcal{S}^{-} \triangleq \{x | \Phi x \in \mathcal{S}, \forall \Phi \in \Omega\}.$$

Sufficient condition :

$$\begin{array}{c} x \in \mathcal{S}, \\ \Downarrow \\ x \in \mathcal{S}^{-}, \\ \Downarrow \\ \Phi x \in \mathcal{S}, \quad \forall \Phi \in \Omega. \end{array}$$

Also necessary condition

bert.pluymers@esat.kuleuven.be

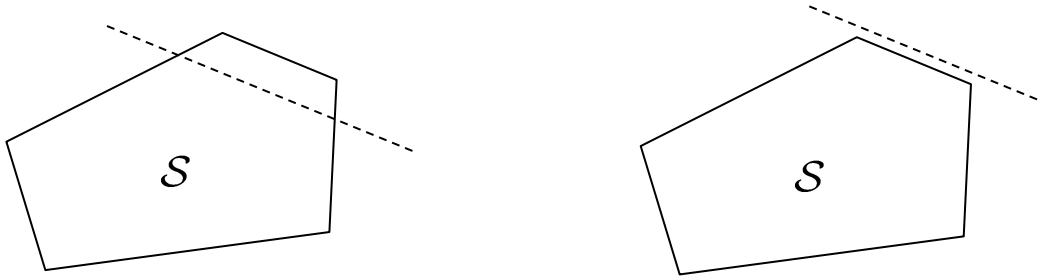
Polyhedral invariant sets for LPV systems

Algorithm :

- Initialize $\mathcal{S} := \mathcal{X}$.
- iteratively add constraints from $\mathcal{S}^-$ to $\mathcal{S}$ until $\mathcal{S} \subseteq \mathcal{S}^-$.

Advantages :

- in step 2 only 'significant' constraints are added to $\mathcal{S}$:



significant insignificant

Polyhedral invariant sets for LPV systems

Algorithm :

- Initialize $\mathcal{S} := \mathcal{X}$.
- iteratively add constraints from $\mathcal{S}^-$ to $\mathcal{S}$ until $\mathcal{S} \subseteq \mathcal{S}^-$.

Advantages :

- prediction tree never explicitly constructed
- given a polyhedral set $\mathcal{S} = \{x | A_{\mathcal{S}} x \leq b_{\mathcal{S}}\}$, it is straightforward to calculate $\mathcal{S}^-$:

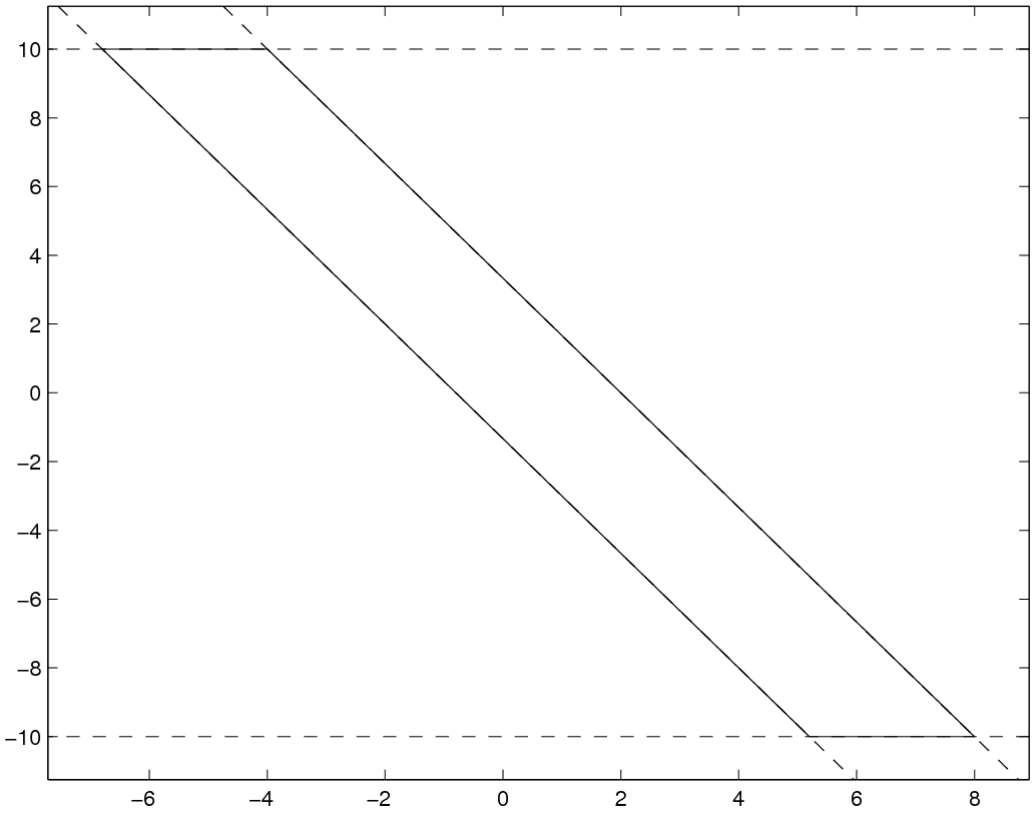
$$A_{\mathcal{S}^-} = \begin{bmatrix} A_{\mathcal{S}}\Phi_1 \\ A_{\mathcal{S}}\Phi_2 \\ \vdots \\ A_{\mathcal{S}}\Phi_L \end{bmatrix}, \qquad b_{\mathcal{S}^-} = \begin{bmatrix} b_{\mathcal{S}} \\ b_{\mathcal{S}} \\ \vdots \\ b_{\mathcal{S}} \end{bmatrix}.$$

bert.pluymers@esat.kuleuven.be

Polyhedral invariant sets for LPV systems

Example

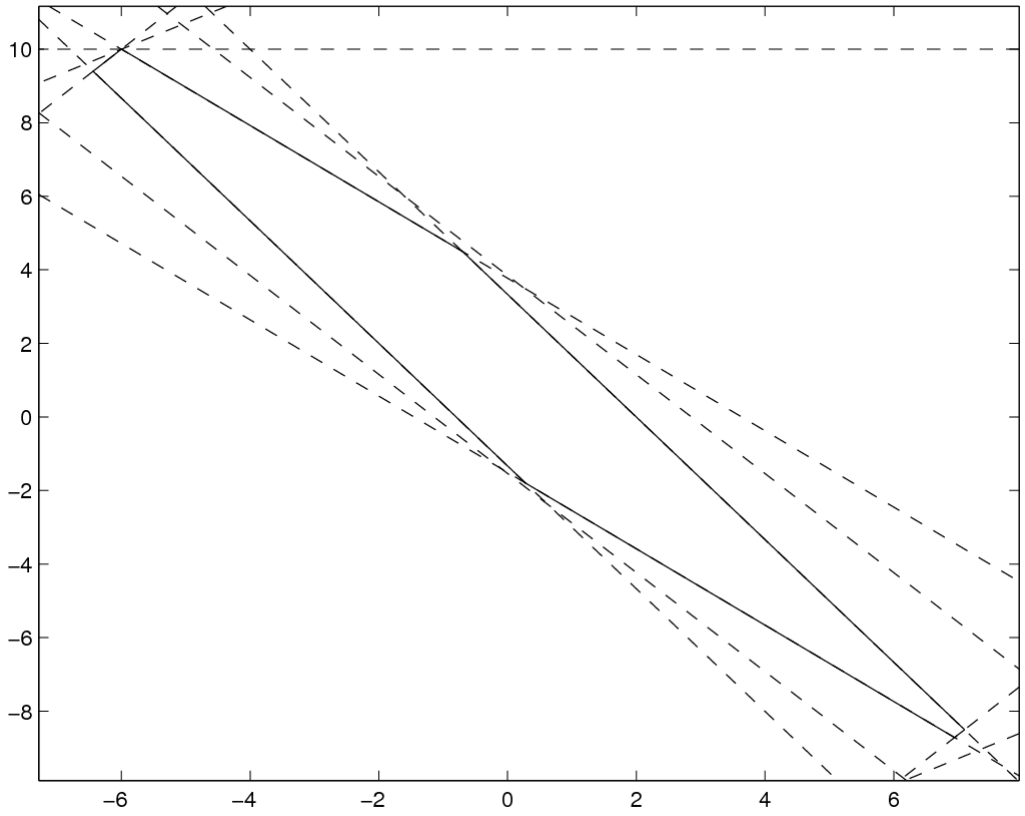
Initialization



Polyhedral invariant sets for LPV systems

Example

Iteration 10

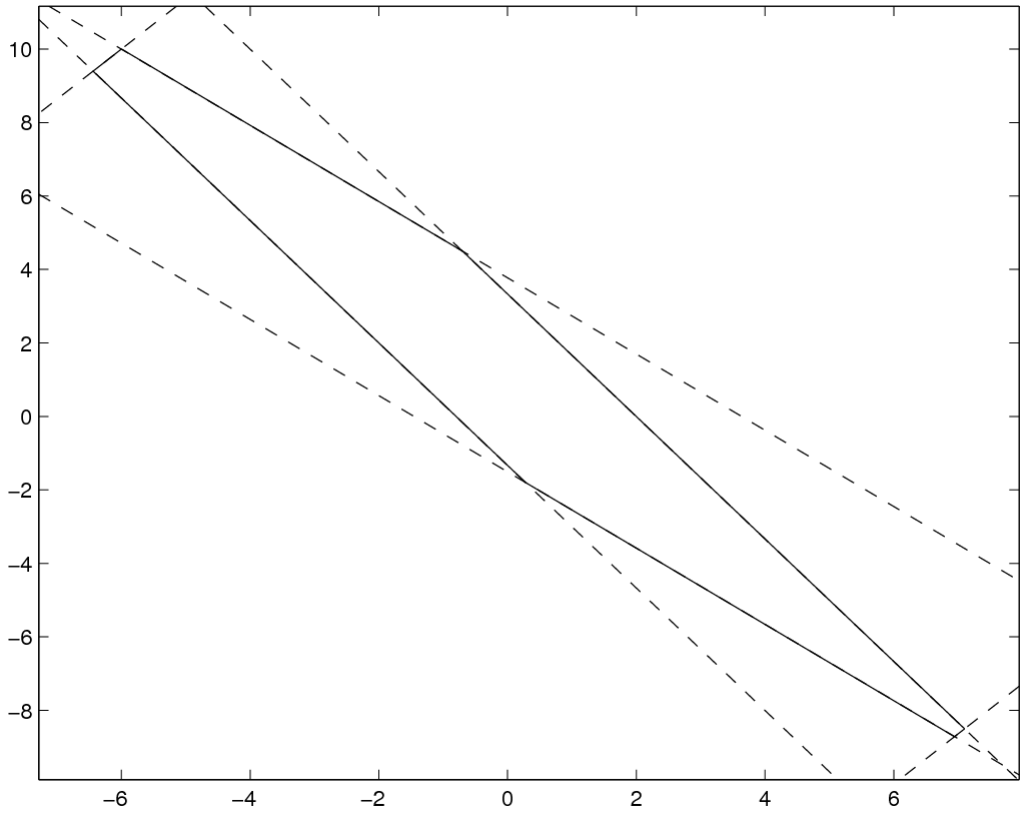


bert.pluymers@esat.kuleuven.be

Polyhedral invariant sets for LPV systems

Example

Iteration 10 + garbage collection

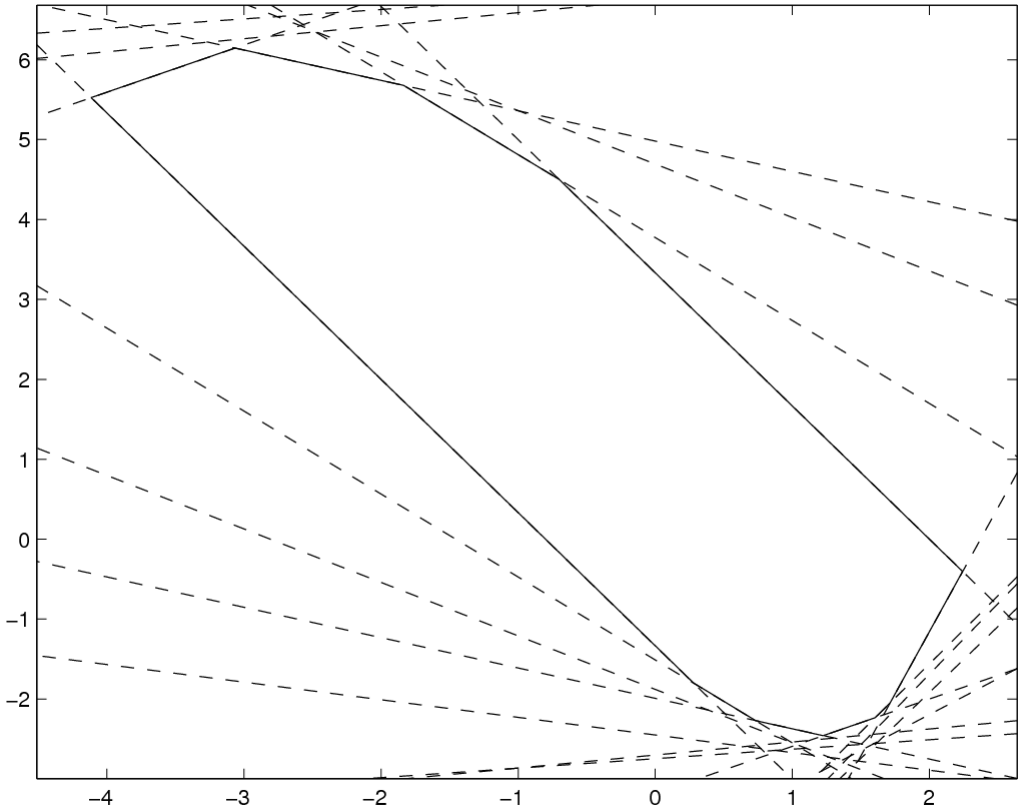


bert.pluymers@esat.kuleuven.be

Polyhedral invariant sets for LPV systems

Example

Iteration 20

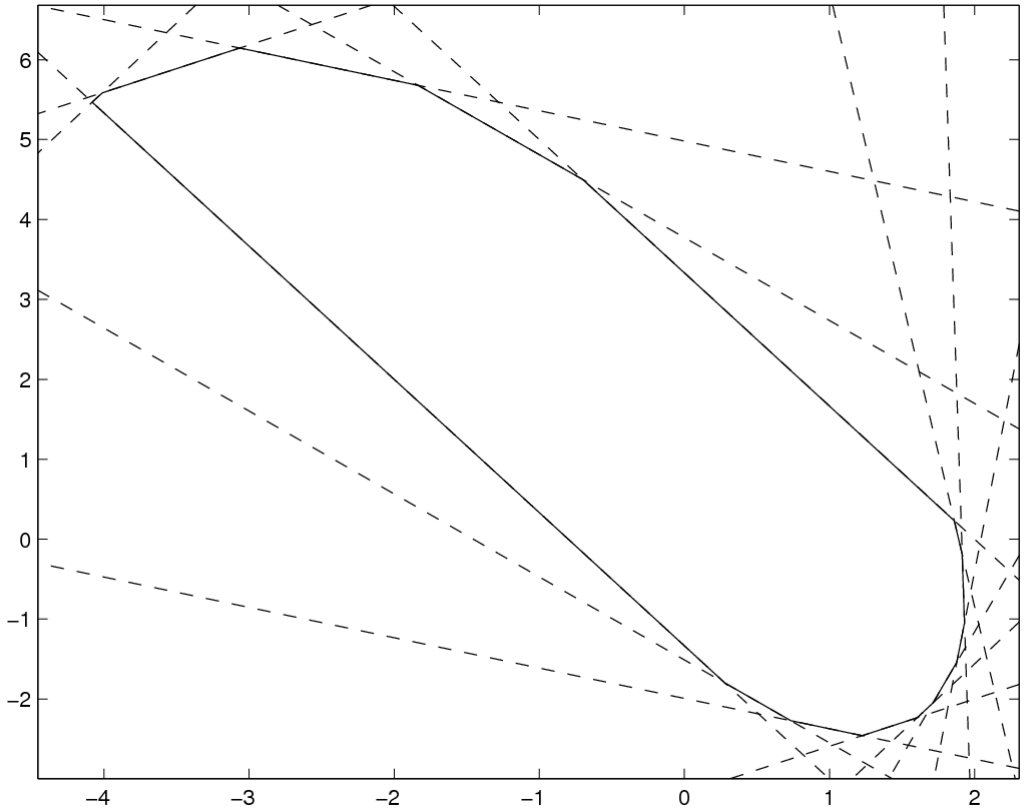


bert.pluymers@esat.kuleuven.be

Polyhedral invariant sets for LPV systems

Example

Final Result

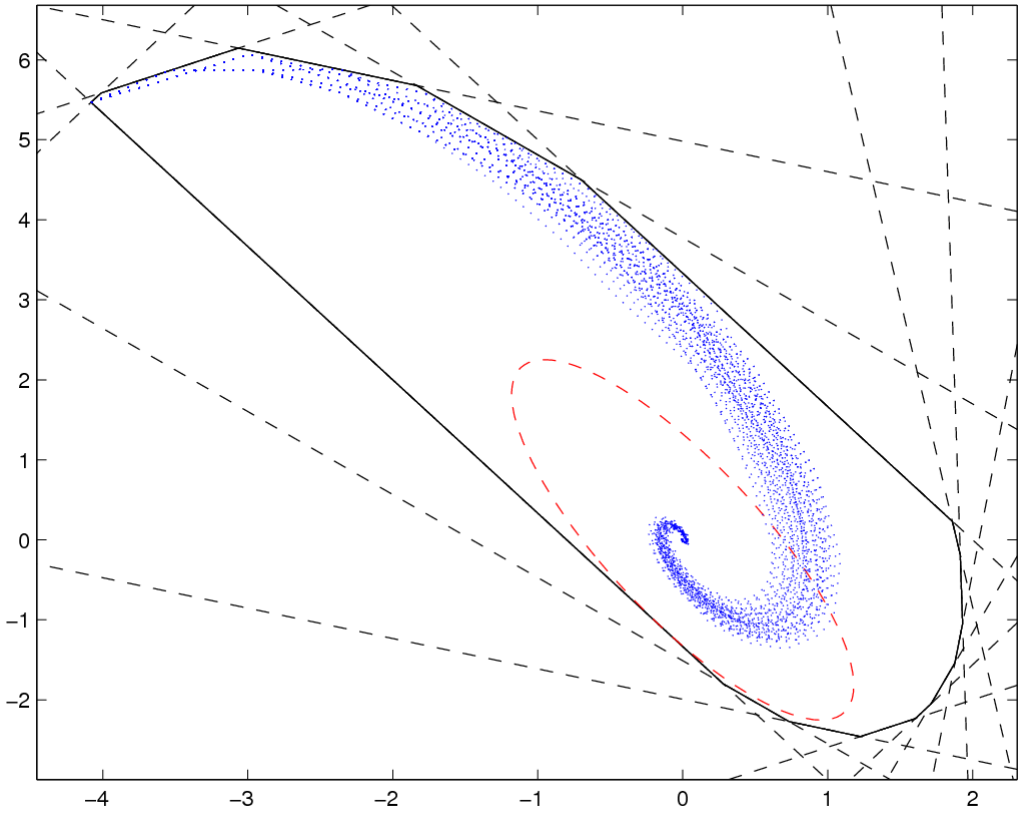


bert.pluymers@esat.kuleuven.be

Polyhedral invariant sets for LPV systems

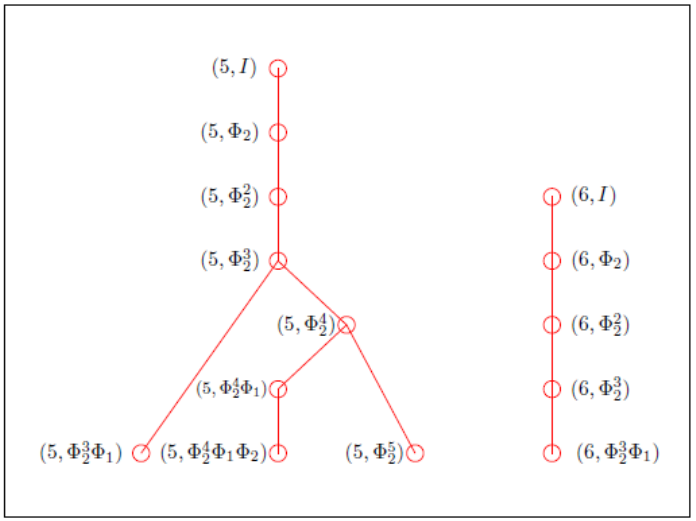
Example

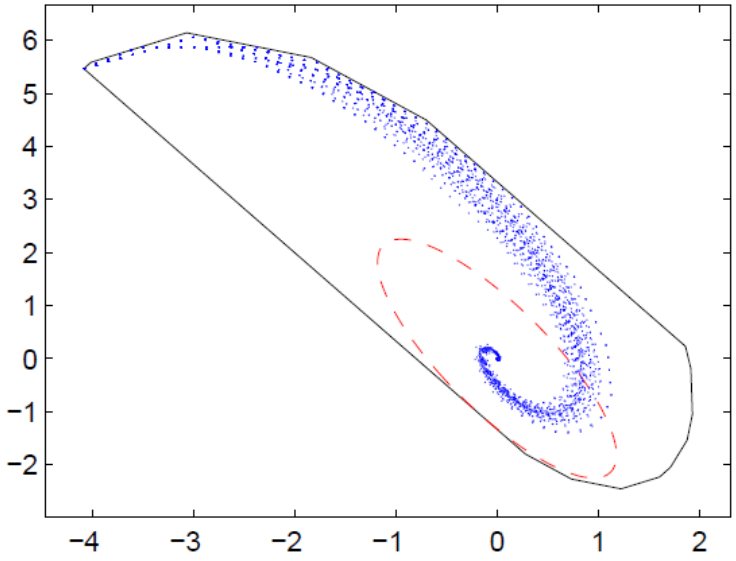
Final Result



Polyhedral invariant sets for LPV systems

Example

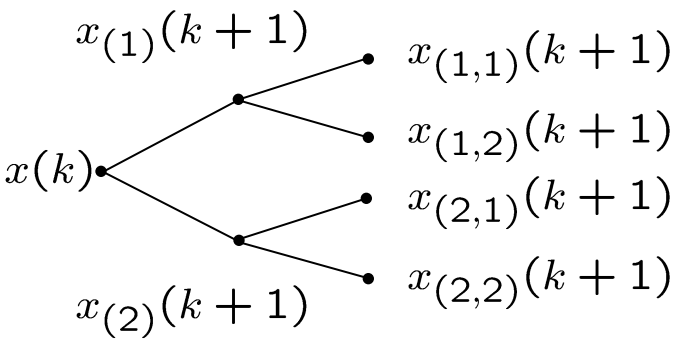


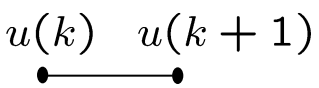


bert.pluymers@esat.kuleuven.be

Recursive feasibility, stability guarantee ?

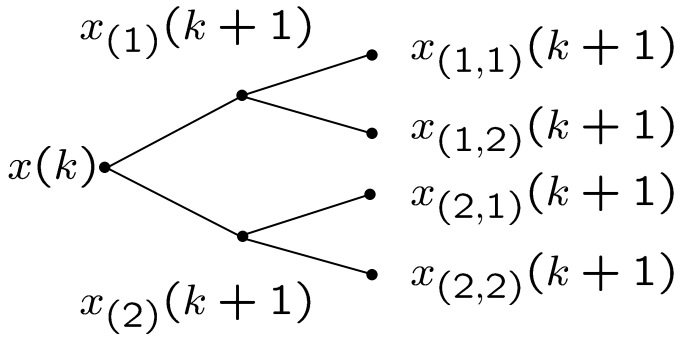
<u>Open</u> loop optimal input sequence

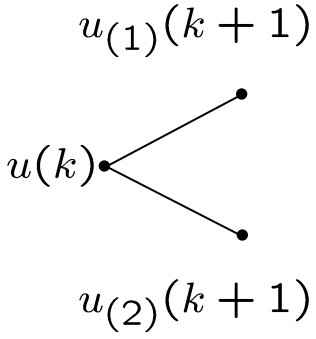




NO recursive feasibility !!!

<u>Closed</u> loop optimal input sequence

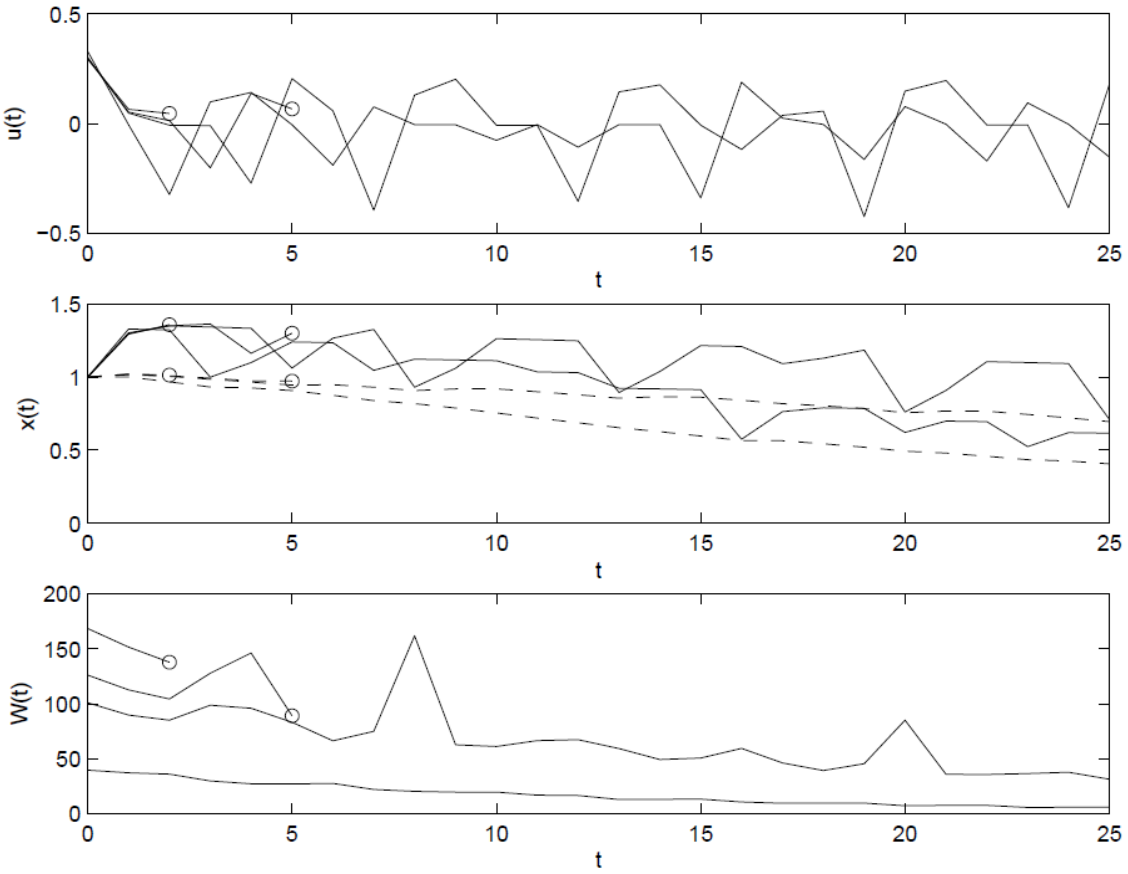




Recursive feasibility

Example revisited…

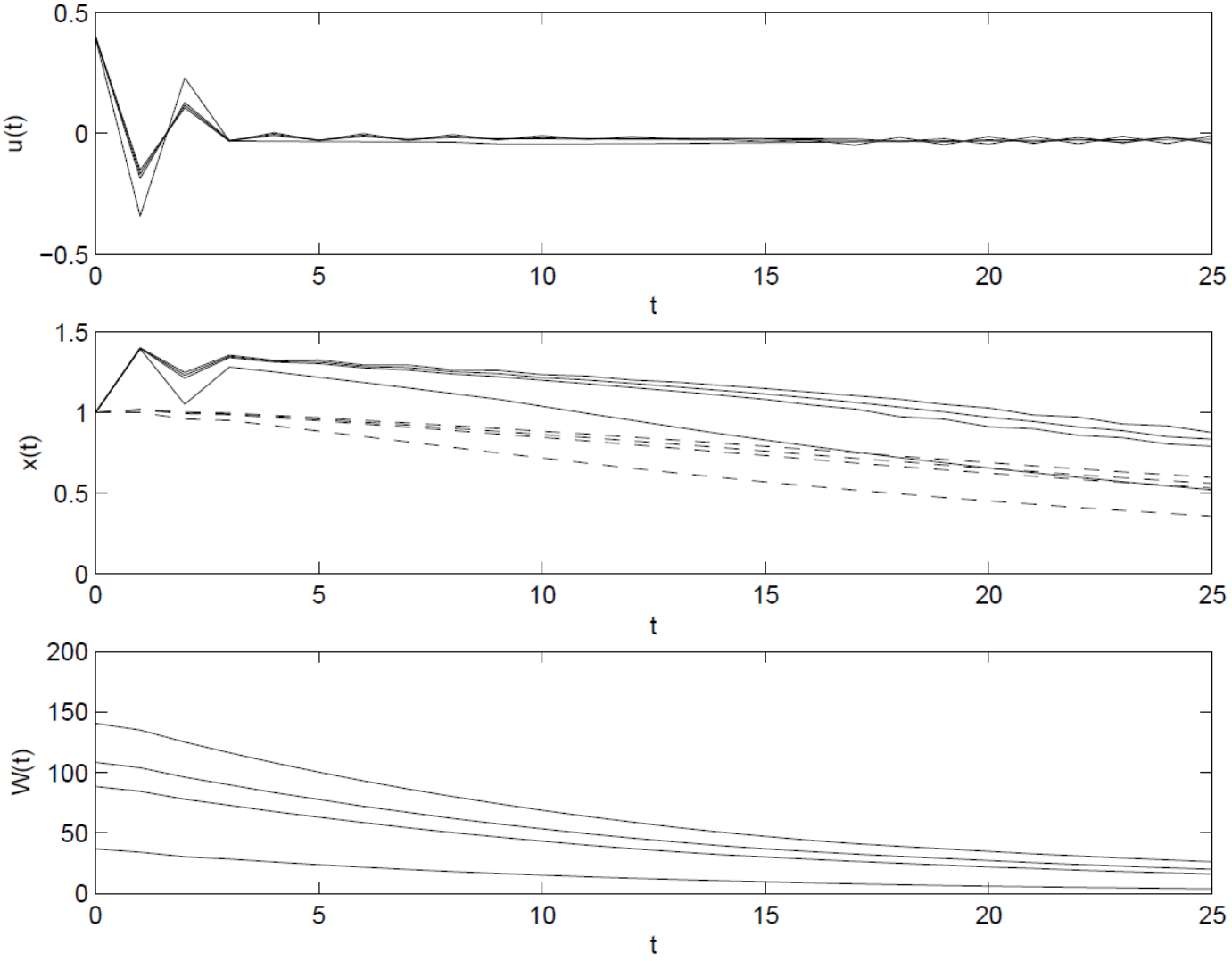
Results for 4 different parameter settings :



- **Recursive feasibility ?**
- **Monotonicity of the cost ?**

Example revisited…

- Example
- Robustness
- **Robust MPC**
- Conclusion



bert.pluymers@esat.kuleuven.be

Conclusion

- Robustness w.r.t a) bounded model uncertainty
  b) bounded disturbances
- necessary modifications :
  - worst-case constraints satisfaction
  - worst-case objective function
  - terminal cost
  - terminal constraint
  - “open-loop” vs. “closed-loop” predictions

  → **currently hot research topic !**
- convex optimization but problem size impractical

  → **currently hot research topic !**

bert.pluymers@esat.kuleuven.be