Suboptimality estimates for NMPC schemes

Lars Grüne

Mathematisches Institut, Universität Bayreuth

in collaboration with

Anders Rantzer (Lund), Jürgen Pannek, Karl Worthmann (Bayreuth)

supported by DFG priority research program 1305
"Control theory for digitally networked dynamical systems"

First LCCC workshop, Lund, May 28-29, 2009

Setup

We consider nonlinear discrete time control systems

$$x(n+1) = f(x(n), u(n))$$

with $x(n) \in X$, $u(n) \in U$, $\quad X$, U arbitrary metric spaces

Setup

We consider nonlinear discrete time control systems

$$x(n+1) = f(x(n), u(n))$$

with $x(n) \in X$, $u(n) \in U$, $\quad X$, U arbitrary metric spaces

Problem: Optimal stabilization via infinite horizon optimal control:

For a running cost $\ell : X \times U \to \mathbb{R}_0^+$ solve

$$\text{minimize} \;\; J_\infty(x,u) = \sum_{n=0}^{\infty} \ell(x(n), u(n))$$

Model predictive control

Direct solution of the problem is numerically hard

Alternative method: model predictive control (MPC)

Model predictive control

Direct solution of the problem is numerically hard

Alternative method: model predictive control (MPC)

Idea: replace the original problem

$$\text{minimize } J_\infty(x,u) = \sum_{n=0}^{\infty} \ell(x(n),u(n))$$

by the iterative (online) solution of finite horizon problems

$$\text{minimize } J_N(x,u) = \sum_{n=0}^{N-1} \ell(x(n),u(n))$$

Model predictive control

Direct solution of the problem is numerically hard

Alternative method: model predictive control (MPC)

Idea: replace the original problem

$$\text{minimize } J_\infty(x,u) = \sum_{n=0}^{\infty} \ell(x(n), u(n))$$

by the iterative (online) solution of finite horizon problems

$$\text{minimize } J_N(x,u) = \sum_{n=0}^{N-1} \ell(x(n), u(n))$$

We obtain a feedback law F_N by a moving horizon technique

Model predictive control

At each time instant τ solve for the current state x_τ

$$\text{minimize } J_N(x_\tau, u) = \sum_{n=0}^{N-1} \ell(x(n), u(t)), \quad x(0) = x_\tau$$

$\rightsquigarrow$ optimal control $u^*(0), \ldots, u^*(N-1)$

Model predictive control

At each time instant τ solve for the current state x_τ

$$\text{minimize } J_N(x_\tau, u) = \sum_{n=0}^{N-1} \ell(x(n), u(t)), \quad x(0) = x_\tau$$

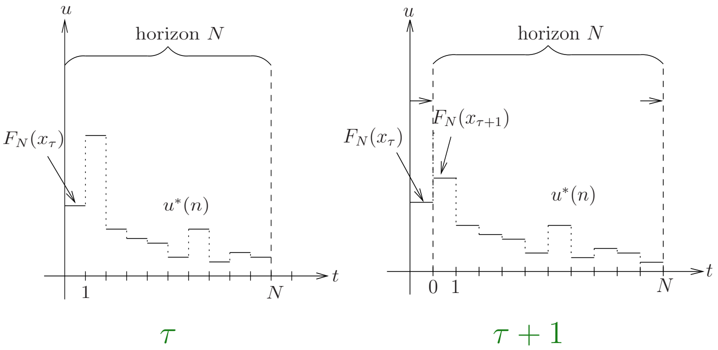
$\rightsquigarrow$ optimal control $u^*(0), \ldots, u^*(N-1)$ $\rightsquigarrow$ set $F_N(x_\tau) := u^*(0)$

Model predictive control

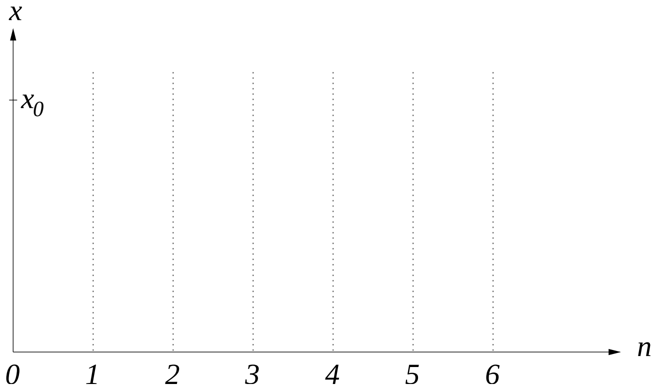
At each time instant τ solve for the current state x_τ

$$\text{minimize } J_N(x_\tau, u) = \sum_{n=0}^{N-1} \ell(x(n), u(t)), \quad x(0) = x_\tau$$

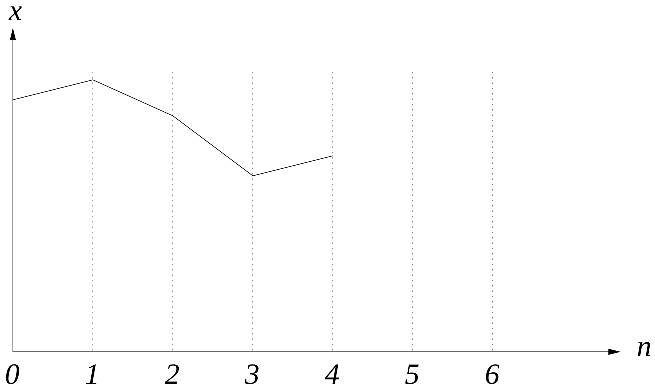
$\rightsquigarrow$ optimal control $u^*(0), \ldots, u^*(N-1)$ $\rightsquigarrow$ set $F_N(x_\tau) := u^*(0)$



MPC from the trajectory point of view

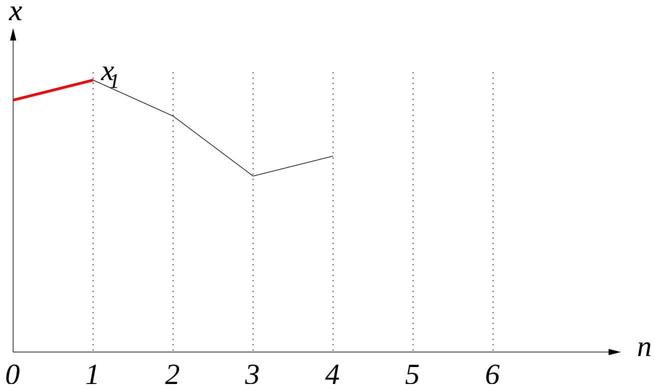


MPC from the trajectory point of view



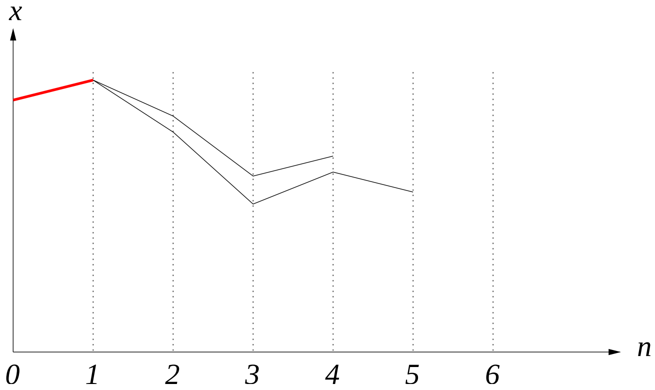
black = predictions (open loop optimization)

MPC from the trajectory point of view



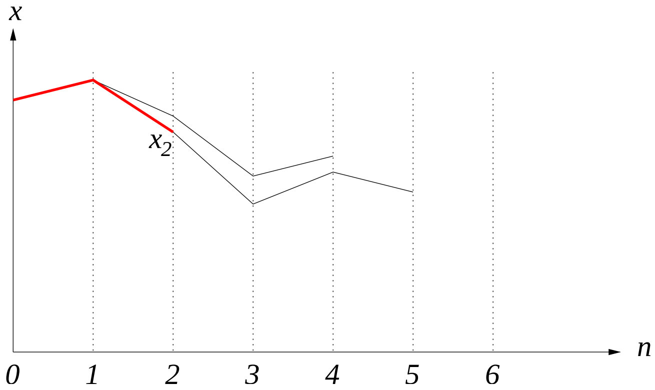
black = predictions (open loop optimization)
red = MPC closed loop

MPC from the trajectory point of view



black = predictions (open loop optimization)
red = MPC closed loop

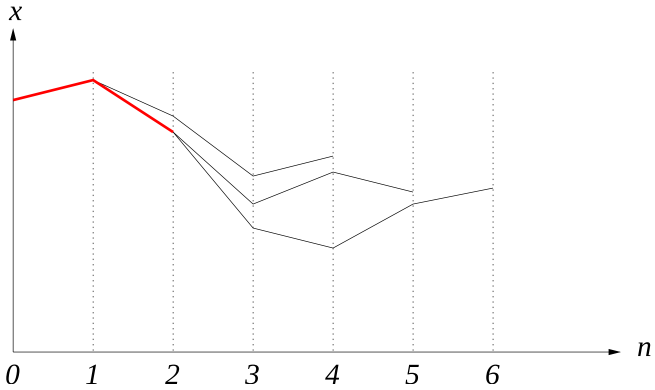
MPC from the trajectory point of view



black = predictions (open loop optimization)

red = MPC closed loop

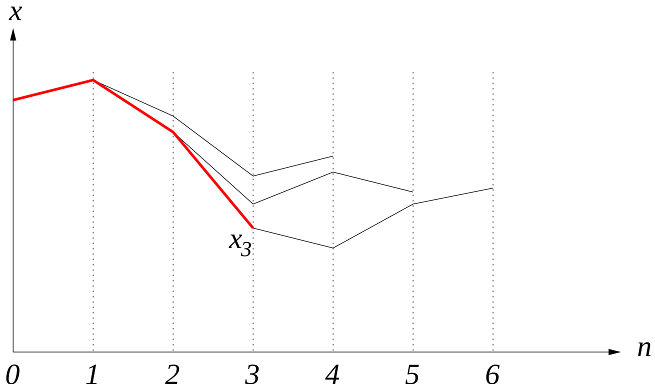
MPC from the trajectory point of view



black = predictions (open loop optimization)

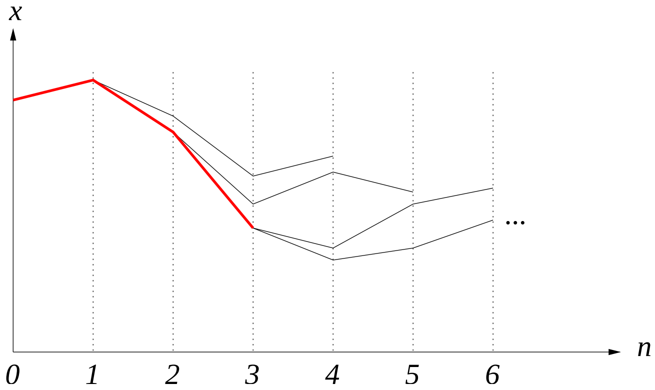
red = MPC closed loop

MPC from the trajectory point of view



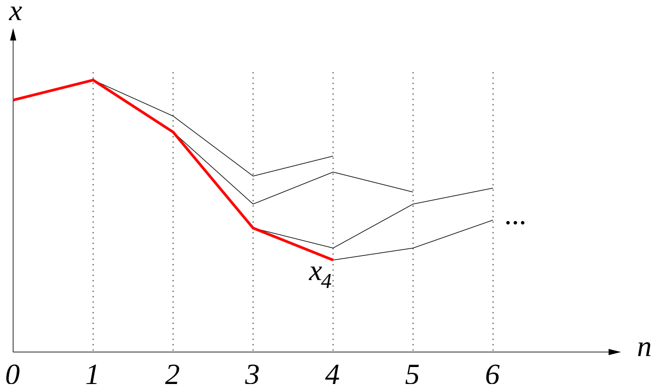
black = predictions (open loop optimization)
red = MPC closed loop

MPC from the trajectory point of view



black = predictions (open loop optimization)
red = MPC closed loop

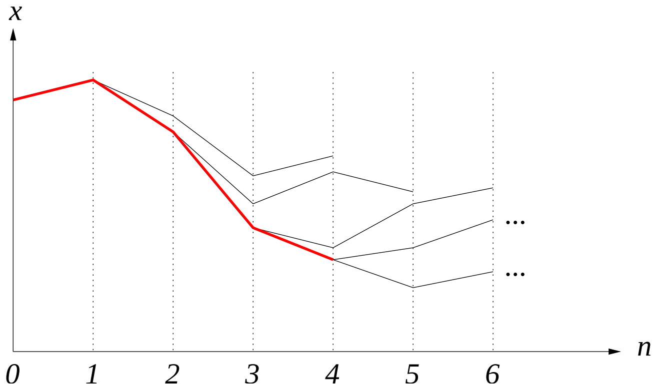
MPC from the trajectory point of view



black = predictions (open loop optimization)

red = MPC closed loop

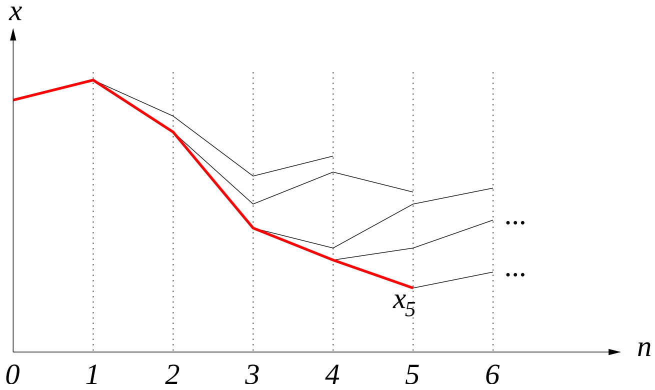
MPC from the trajectory point of view



black = predictions (open loop optimization)

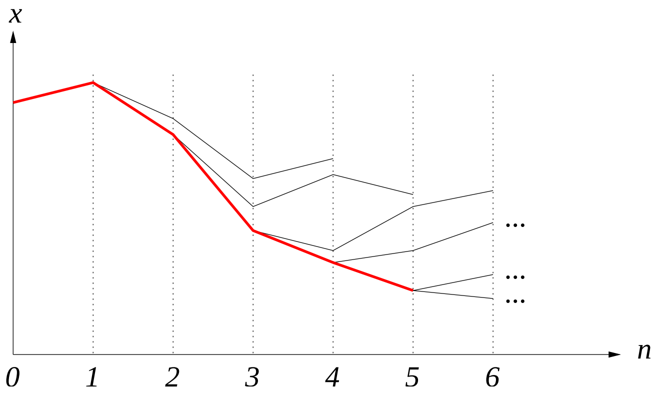
red = MPC closed loop

MPC from the trajectory point of view



black = predictions (open loop optimization)
red = MPC closed loop

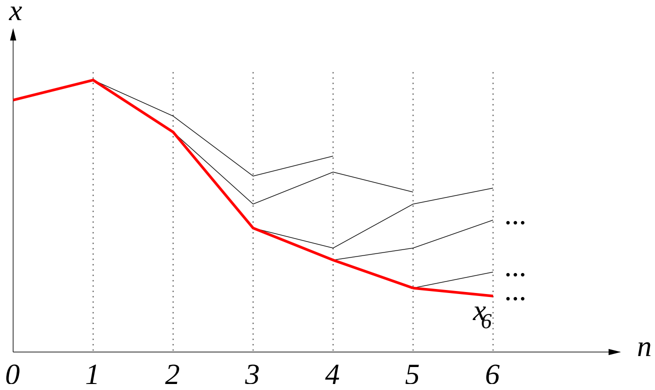
MPC from the trajectory point of view



black = predictions (open loop optimization)

red = MPC closed loop

MPC from the trajectory point of view



black = predictions (open loop optimization)
red = MPC closed loop

Relaxed dynamic programming inequality

Questions:

- When does MPC stabilize the system?

Relaxed dynamic programming inequality

Questions:

- When does MPC stabilize the system?
- How good is the MPC Feedback law?

Relaxed dynamic programming inequality

Questions:

- When does MPC stabilize the system?
- How good is the MPC Feedback law?

Define optimal value functions

$$V_N(x) := \inf_u J_N(x,u)$$
$$V_\infty(x) := \inf_u J_\infty(x,u)$$

Theorem: If there exists $\alpha \in (0,1]$ such that the relaxed dynamic programming inequality

$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha \ell(x, F_N(x))$$

holds for all x, then asymptotic stability follows (with V_N as Lyapunov function)

Relaxed dynamic programming inequality

Questions:

- When does MPC stabilize the system?
- How good is the MPC Feedback law?

Define optimal value functions

$$V_N(x) := \inf_u J_N(x,u)$$
$$V_\infty(x) := \inf_u J_\infty(x,u)$$

Theorem: If there exists $\alpha \in (0,1]$ such that the relaxed dynamic programming inequality

$$V_N(f(x,F_N(x))) \leq V_N(x) - \alpha\ell(x,F_N(x))$$

holds for all x, then asymptotic stability follows (with V_N as Lyapunov function) and we get the suboptimality estimate

$$J_\infty(x,F_N) \leq V_N(x)/\alpha \leq V_\infty(x)/\alpha$$

Relaxed dynamic programming inequality

Questions:

- When does MPC stabilize the system?
- How good is the MPC Feedback law?

Define optimal value functions

$$V_N(x) := \inf_u J_N(x,u)$$
$$V_\infty(x) := \inf_u J_\infty(x,u)$$

Theorem: If there exists $\alpha \in (0,1]$ such that the relaxed dynamic programming inequality

$$V_N(f(x,F_N(x))) \leq V_N(x) - \alpha \ell(x,F_N(x))$$

holds for all x, then asymptotic stability follows (with V_N as Lyapunov function) and we get the suboptimality estimate

$$J_\infty(x,F_N) \leq V_N(x)/\alpha \leq V_\infty(x)/\alpha$$

Note: The last inequality does not hold for MPC schemes with stabilizing terminal constraints

Computing α

Goal: Compute α in the relaxed DP inequality

$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha\ell(x, F_N(x))$$

Computing α

Goal: Compute α in the relaxed DP inequality

$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha\ell(x, F_N(x))$$

Different ways to compute α:

UNIVERSITÄT BAYREUTH

Computing α

Goal: Compute α in the relaxed DP inequality

$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha\ell(x, F_N(x))$$

Different ways to compute α:

- by computing V_N (see also [Shamma/Xiong '97])

Computing α

Goal: Compute α in the relaxed DP inequality

$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha\ell(x, F_N(x))$$

Different ways to compute α:

- by computing V_N (see also [Shamma/Xiong '97])
- by imposing suitable bounds on V_N [Gr./Rantzer '08]

Computing α

Goal: Compute α in the relaxed DP inequality

$$V_N(f(x, F_N(x))) \le V_N(x) - \alpha\ell(x, F_N(x))$$

Different ways to compute α:

- by computing V_N (see also [Shamma/Xiong '97])
- by imposing suitable bounds on V_N [Gr./Rantzer '08]
- using controllability properties [Gr. '09]

Computing α

Goal: Compute α in the relaxed DP inequality

$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha\ell(x, F_N(x))$$

Different ways to compute α:

- by computing V_N (see also [Shamma/Xiong '97])
- by imposing suitable bounds on V_N [Gr./Rantzer '08]
- using controllability properties [Gr. '09]
- online along the NMPC closed loop trajectory [Gr./Pannek '09]

Computing α

Goal: Compute α in the relaxed DP inequality

$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha\ell(x, F_N(x))$$

Different ways to compute α:

- by computing V_N (see also [Shamma/Xiong '97])
- by imposing suitable bounds on V_N [Gr./Rantzer '08]
- using controllability properties [Gr. '09] $\longleftarrow$
- online along the NMPC closed loop trajectory [Gr./Pannek '09]

Computing α

Goal: Compute α in the relaxed dynamic programming inequality

$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha\ell(x, F_N(x))$$

under the C, σ-exponential controllability property (with respect to the running cost l):

Computing α

Goal: Compute α in the relaxed dynamic programming inequality

$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha\ell(x, F_N(x))$$

under the C, σ-exponential controllability property (with respect to the running cost l):

For each $x(0)$ there exists a control sequence $u(\cdot)$ such that

$$\ell(x(n), u(n)) \leq C\sigma^n \ell^*(x(0))$$

holds for all $n \in \mathbb{N}_0$, where $\ell^*(x(0)) = \inf_{u \in U} \ell(x(0), u)$

Suboptimality and stability condition

C, σ-exponential controllability: $\ell(x(n),u(n)) \leq C\sigma^n \ell^*(x(0))$

Suboptimality and stability condition

C, σ-exponential controllability: $\ell(x(n),u(n)) \le C\sigma^n \ell^*(x(0))$

Theorem: The value

$$\alpha(C,\sigma) = 1 - \frac{(\gamma_N - 1)\prod_{i=2}^{N}(\gamma_i - 1)}{\prod_{i=2}^{N}\gamma_i - \prod_{i=2}^{N}(\gamma_i - 1)} \quad \text{with} \quad \gamma_i = \sum_{k=0}^{i-1} C\sigma^k$$

is the maximal α in the relaxed dynamic programming inequality over all C, σ-exponentially controllable systems.

Suboptimality and stability condition

C, σ-exponential controllability: $\ell(x(n),u(n)) \le C\sigma^n \ell^*(x(0))$

Theorem: The value

$$\alpha(C,\sigma) = 1 - \frac{(\gamma_N - 1)\prod_{i=2}^{N}(\gamma_i - 1)}{\prod_{i=2}^{N}\gamma_i - \prod_{i=2}^{N}(\gamma_i - 1)} \quad \text{with} \quad \gamma_i = \sum_{k=0}^{i-1} C\sigma^k$$

is the maximal α in the relaxed dynamic programming inequality over all C, σ-exponentially controllable systems.

In particular: If $\alpha(C,\sigma) > 0$, then the MPC feedback F_N stabilizes all C, σ-exponentially controllable systems and we get $J_\infty(x,F_N) \le V_\infty(x)/\alpha(C,\sigma)$.

Suboptimality and stability condition

C, σ-exponential controllability: $\ell(x(n), u(n)) \leq C\sigma^n \ell^*(x(0))$

Theorem: The value

$$\alpha(C,\sigma) = 1 - \frac{(\gamma_N - 1)\prod_{i=2}^{N}(\gamma_i - 1)}{\prod_{i=2}^{N}\gamma_i - \prod_{i=2}^{N}(\gamma_i - 1)} \quad \text{with} \quad \gamma_i = \sum_{k=0}^{i-1} C\sigma^k$$

is the maximal α in the relaxed dynamic programming inequality over all C, σ-exponentially controllable systems.

In particular: If $\alpha(C,\sigma) > 0$, then the MPC feedback F_N stabilizes all C, σ-exponentially controllable systems and we get $J_\infty(x, F_N) \leq V_\infty(x)/\alpha(C,\sigma)$.

Furthermore: If $\alpha(C,\sigma) < 0$ then there exists a C, σ-exponentially controllable system, which is not stabilized by F_N.

UNIVERSITÄT BAYREUTH

Idea of proof

We want to compute α such that for all x:

$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha\ell(x, F_N(x))$$

Idea of proof

We want to compute α such that for all x:

$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha \ell(x, F_N(x))$$

- the controllability property induces upper bounds on (sums of) the cost $\ell(x^*(n), u^*(n))$ along the optimal trajectory corresponding to $V_N(x)$

UNIVERSITÄT BAYREUTH

Idea of proof

We want to compute α such that for all x:

$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha \ell(x, F_N(x))$$

- the controllability property induces upper bounds on (sums of) the cost $\ell(x^*(n), u^*(n))$ along the optimal trajectory corresponding to $V_N(x)$
- the bounds on the $\ell(x^*(n), u^*(n))$ and the controllability condition induce upper bounds on $V_N(f(x, F_N(x)))$

UNIVERSITÄT BAYREUTH

Idea of proof

We want to compute α such that for all x:

$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha\ell(x, F_N(x))$$

- the controllability property induces upper bounds on (sums of) the cost $\ell(x^*(n), u^*(n))$ along the optimal trajectory corresponding to $V_N(x)$
- the bounds on the $\ell(x^*(n), u^*(n))$ and the controllability condition induce upper bounds on $V_N(f(x, F_N(x)))$
- combining these bounds leads to a linear program for α

Idea of proof

We want to compute α such that for all x:

$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha\ell(x, F_N(x))$$

- the controllability property induces upper bounds on (sums of) the cost $\ell(x^*(n), u^*(n))$ along the optimal trajectory corresponding to $V_N(x)$
- the bounds on the $\ell(x^*(n), u^*(n))$ and the controllability condition induce upper bounds on $V_N(f(x, F_N(x)))$
- combining these bounds leads to a linear program for α
- this linear program is explicitly solvable

Idea of proof

We want to compute α such that for all x:

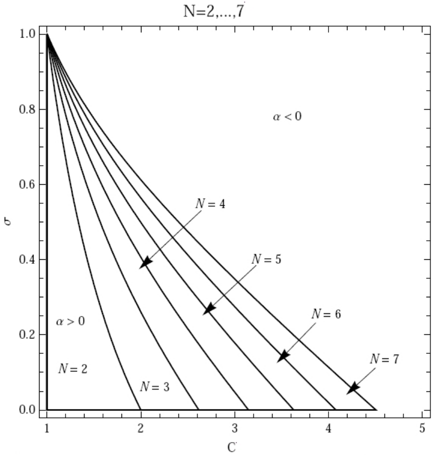
$$V_N(f(x, F_N(x))) \leq V_N(x) - \alpha\ell(x, F_N(x))$$

- the controllability property induces upper bounds on (sums of) the cost $\ell(x^*(n), u^*(n))$ along the optimal trajectory corresponding to $V_N(x)$
- the bounds on the $\ell(x^*(n), u^*(n))$ and the controllability condition induce upper bounds on $V_N(f(x, F_N(x)))$
- combining these bounds leads to a linear program for α
- this linear program is explicitly solvable
- the converse statement for $\alpha(C, \sigma) < 0$ is obtained by explicit construction of a counterexample

UNIVERSITÄT BAYREUTH

Stability chart for C and σ

Minimal horizon N for stable NMPC depending on C and σ

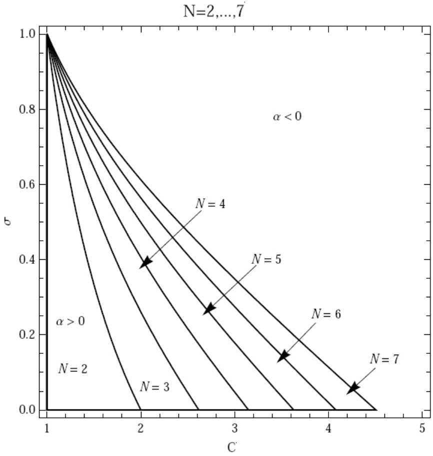


(Figure: Harald Voit)

UNIVERSITÄT BAYREUTH

Stability chart for C and σ

Minimal horizon N for stable NMPC depending on C and σ



(Figure: Harald Voit)

Conclusion: good performance can be expected for small overshoot C

Example

In general, quantitative values for C and σ (or analogous parameters in alternative controllability assumptions) are difficult if not impossible to obtain

Example

In general, quantitative values for C and σ (or analogous parameters in alternative controllability assumptions) are difficult if not impossible to obtain

However, our results are still useful if only qualitative information is known

Example

In general, quantitative values for C and σ (or analogous parameters in alternative controllability assumptions) are difficult if not impossible to obtain

However, our results are still useful if only qualitative information is known

We illustrate this with the 1d controlled PDE

$$y_t = y_x + \nu y_{xx} + \mu y(y+1)(1-y) + u$$

with

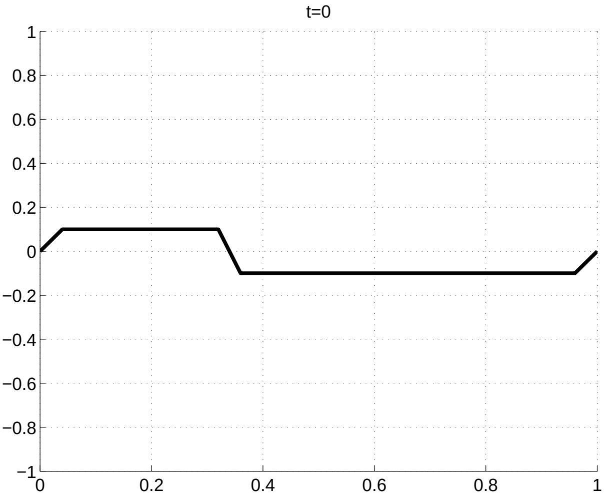
domain $\Omega = [0, 1]$

solution $y = y(t, x)$ and distributed control $u = u(t, x)$

boundary conditions $y(t, 0) = y(t, 1) = 0$

parameters $\nu = 0.1$ and $\mu = 10$

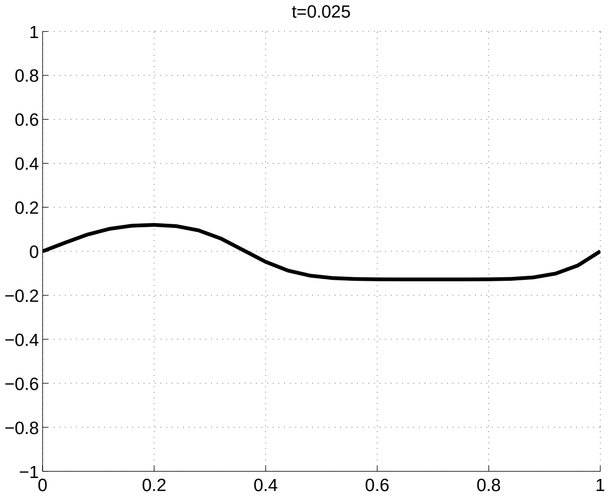
The uncontrolled PDE



uncontrolled ($u \equiv 0$)

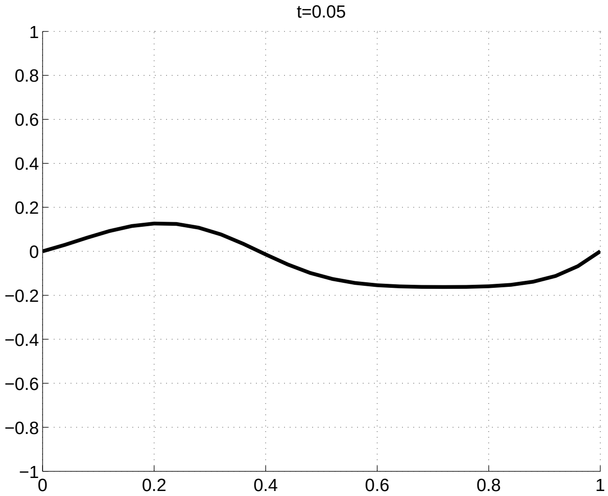
UNIVERSITÄT BAYREUTH

The uncontrolled PDE



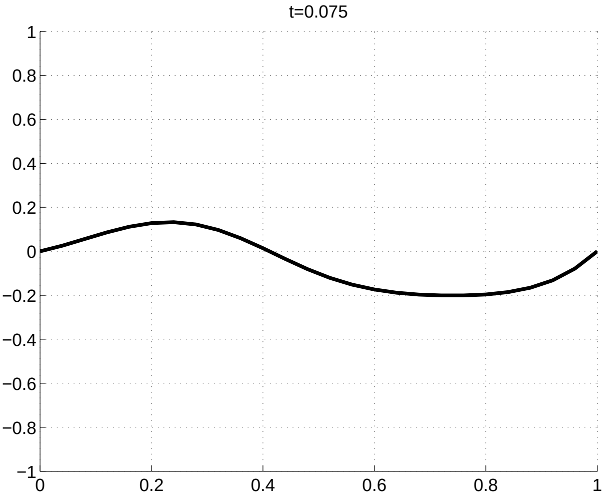
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



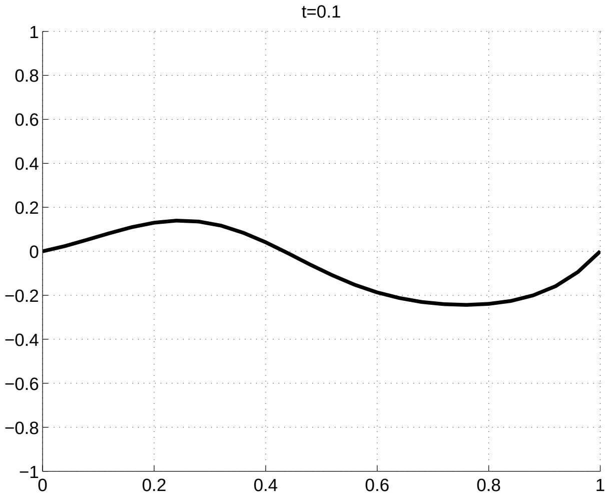
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



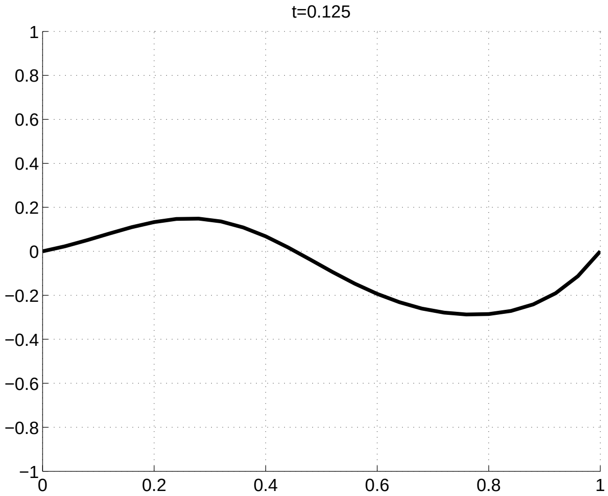
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



uncontrolled ($u \equiv 0$)

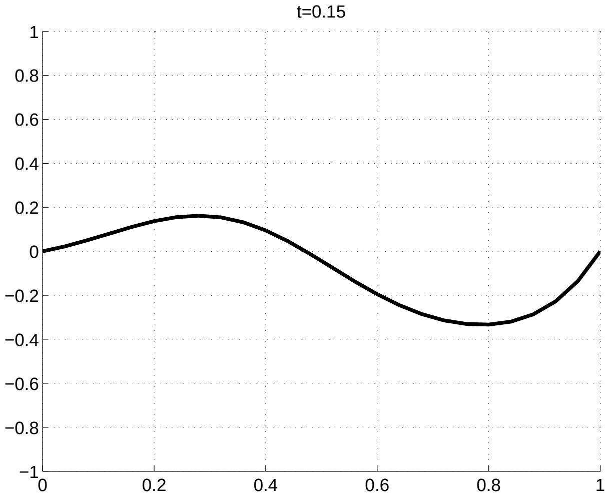
The uncontrolled PDE



uncontrolled ($u \equiv 0$)

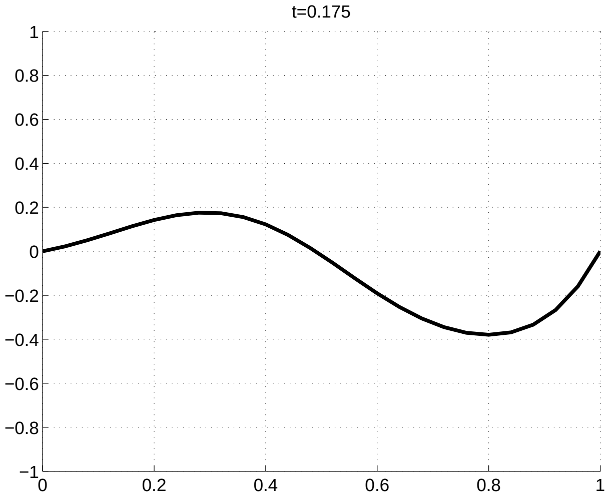
UNIVERSITÄT BAYREUTH

The uncontrolled PDE



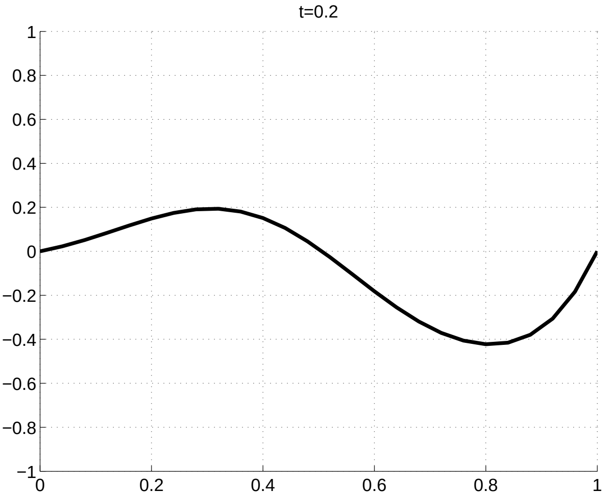
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



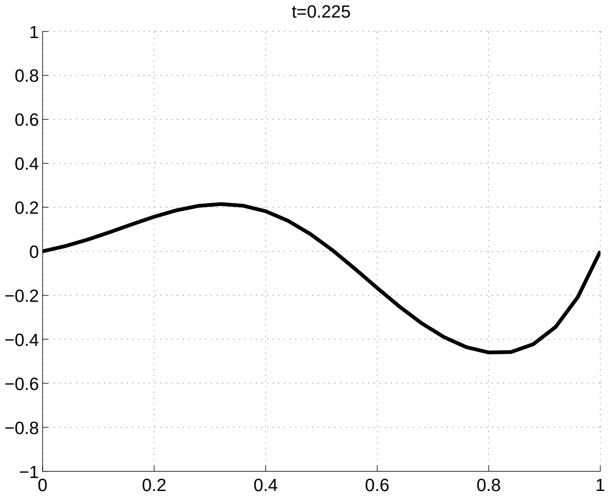
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



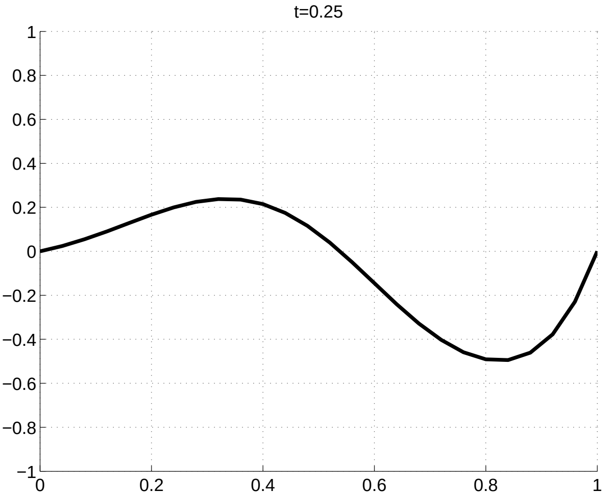
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



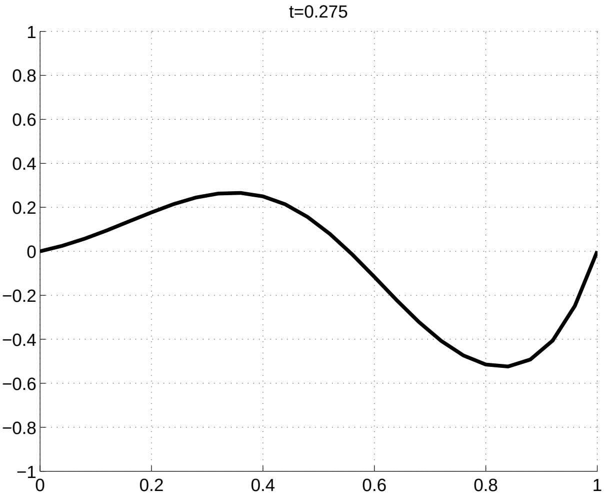
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



uncontrolled ($u \equiv 0$)

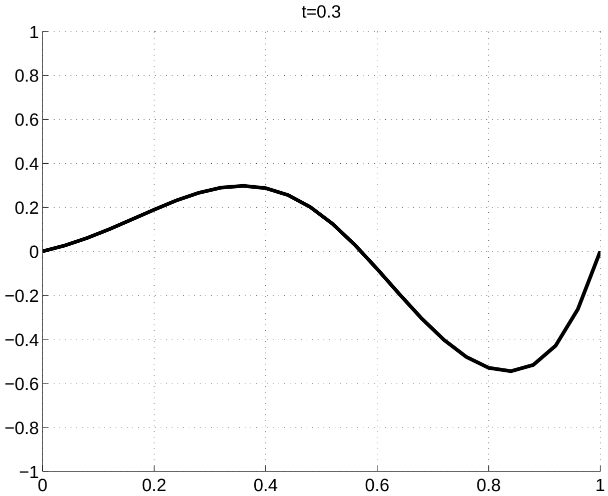
The uncontrolled PDE



uncontrolled ($u \equiv 0$)

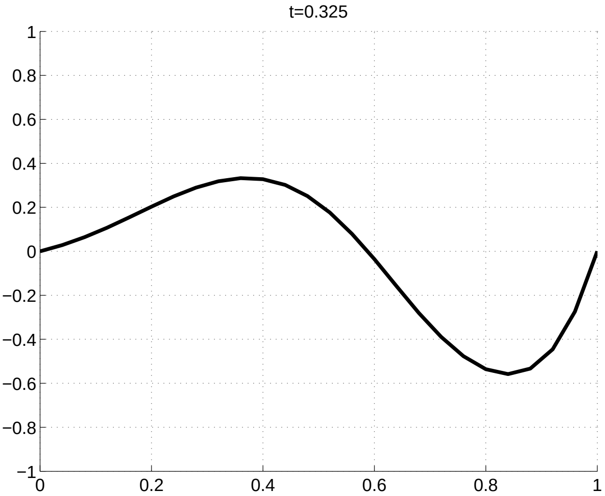
UNIVERSITÄT BAYREUTH

The uncontrolled PDE



uncontrolled ($u \equiv 0$)

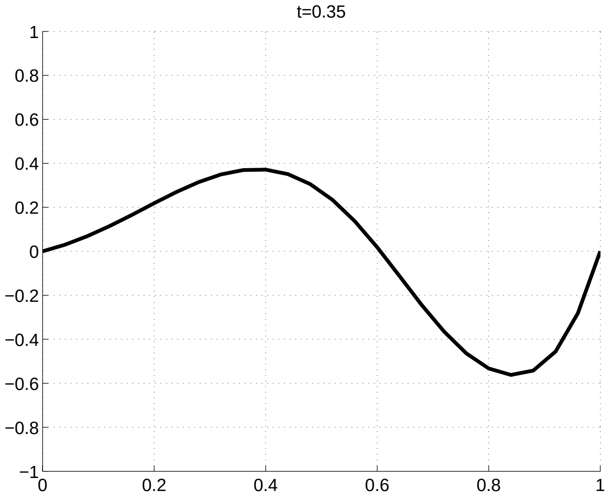
The uncontrolled PDE



uncontrolled ($u \equiv 0$)

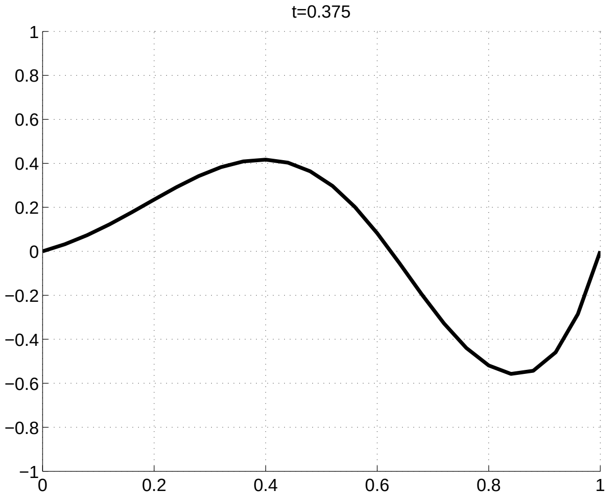
UNIVERSITÄT BAYREUTH

The uncontrolled PDE



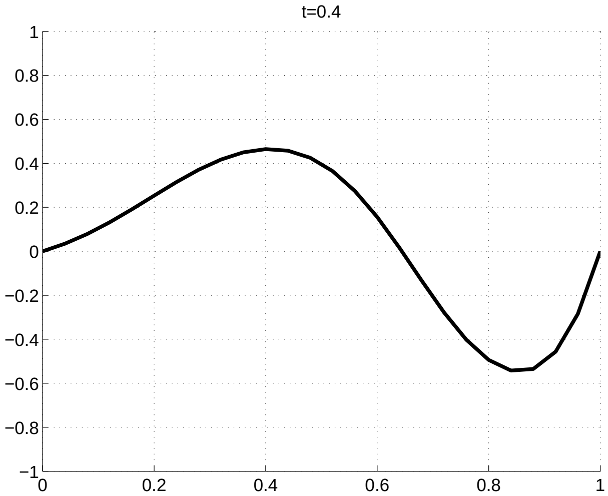
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



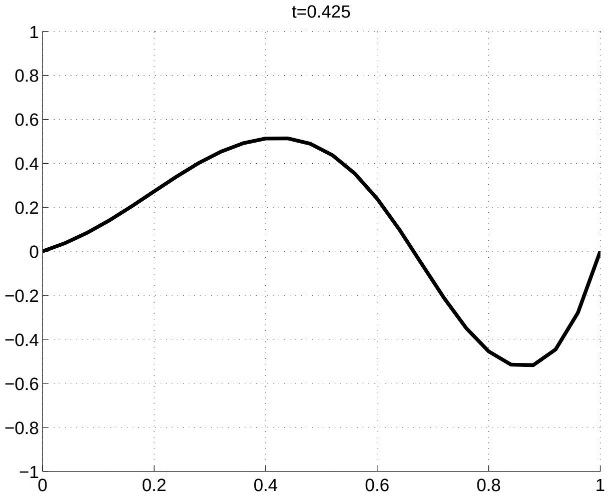
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



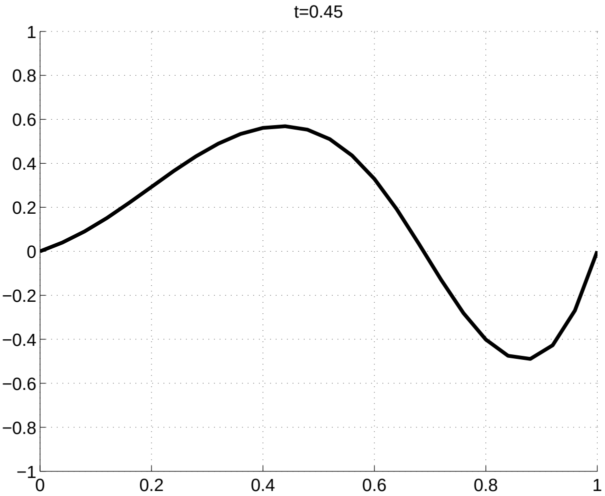
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



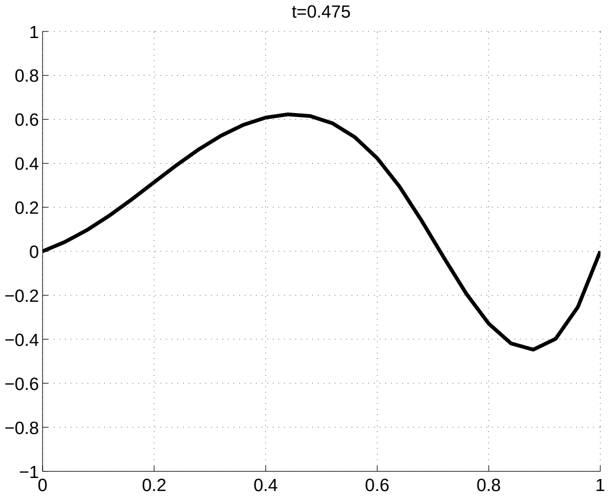
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



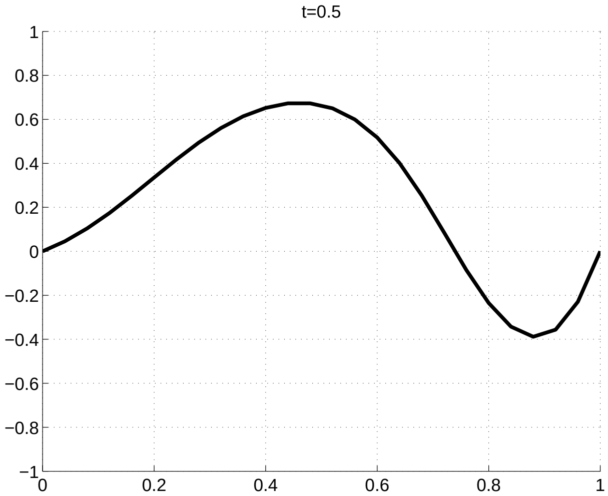
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



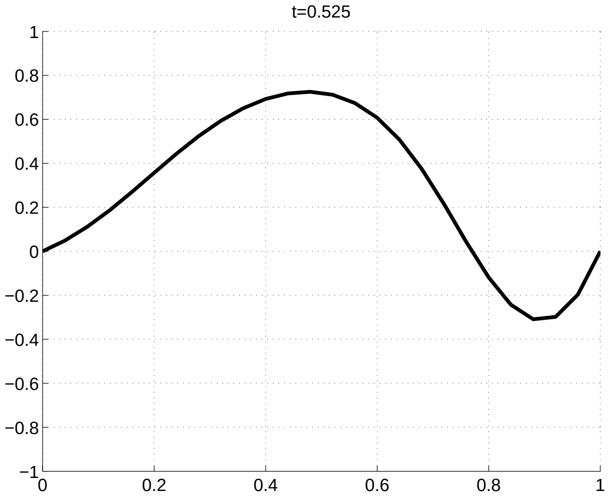
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



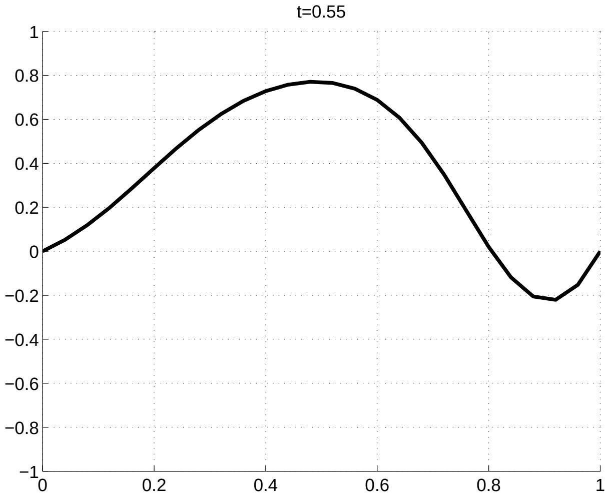
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



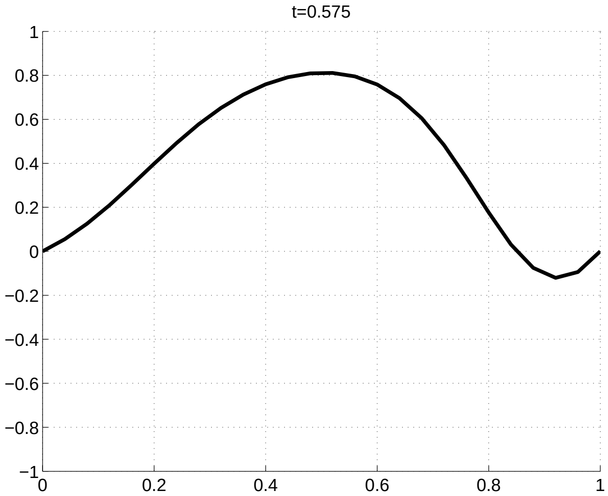
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



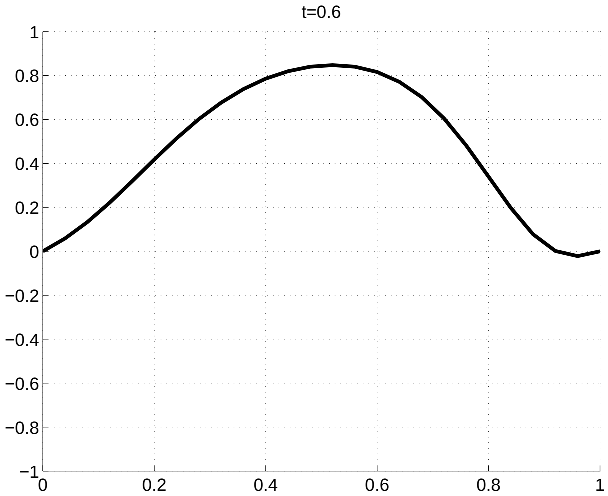
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



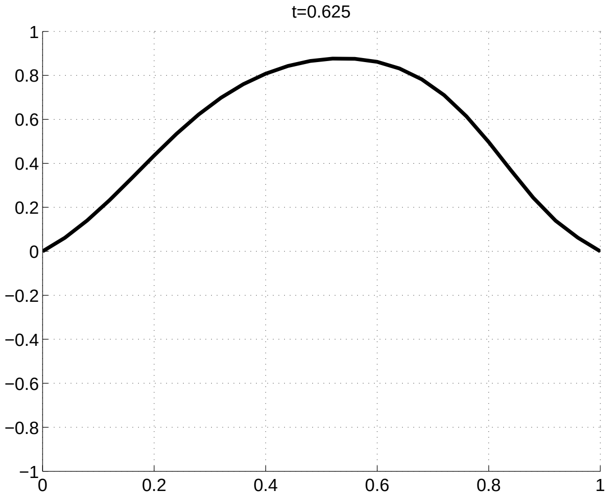
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



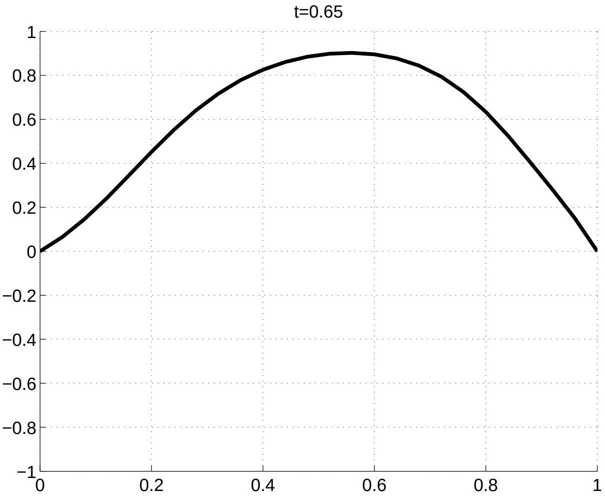
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



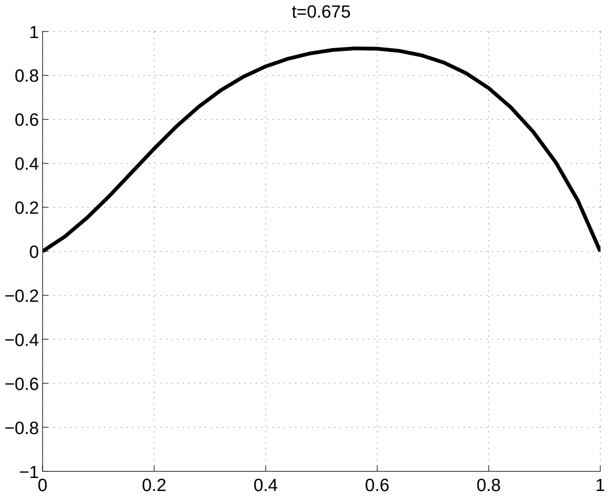
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



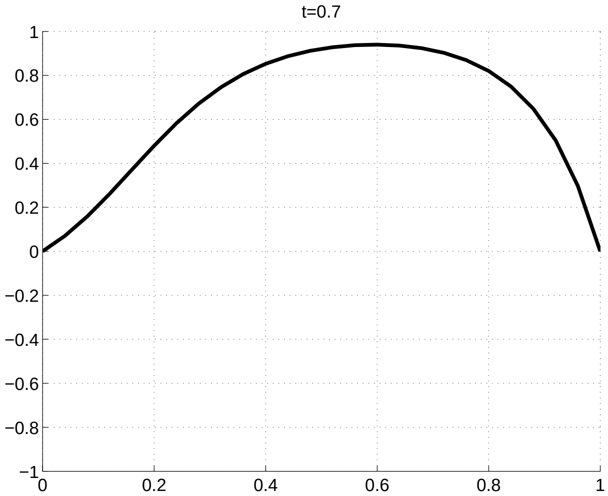
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



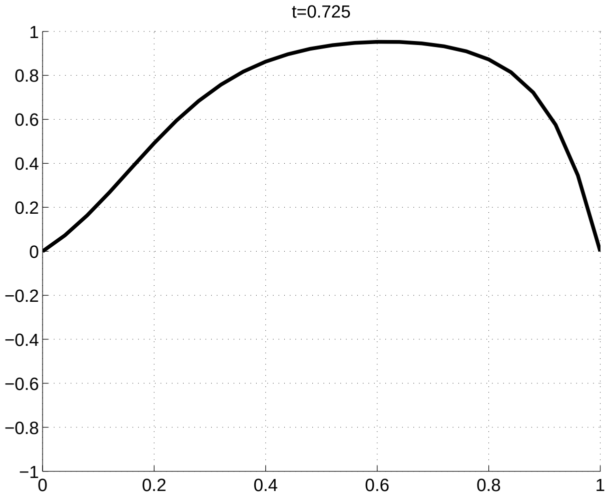
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



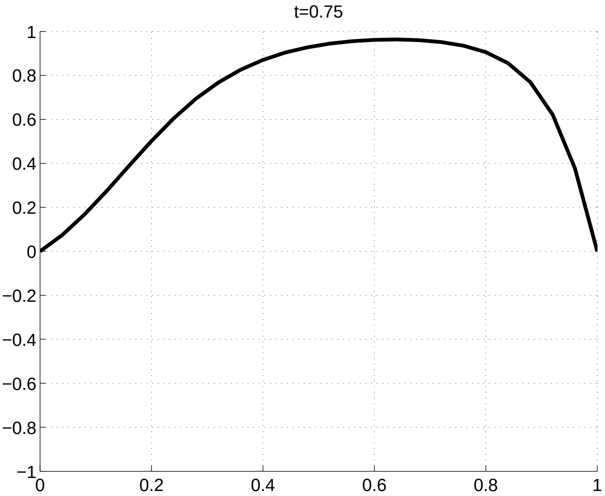
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



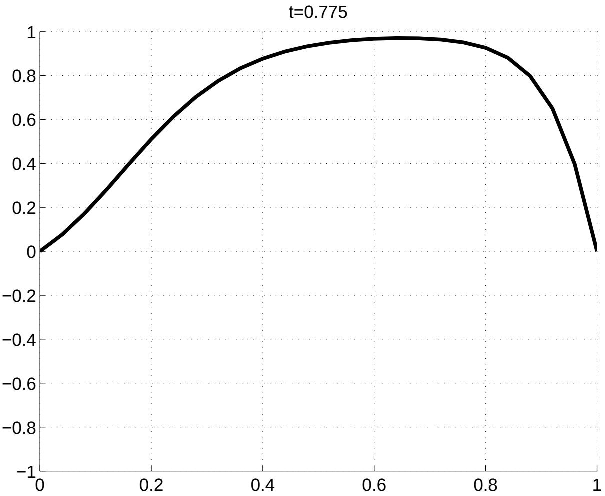
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



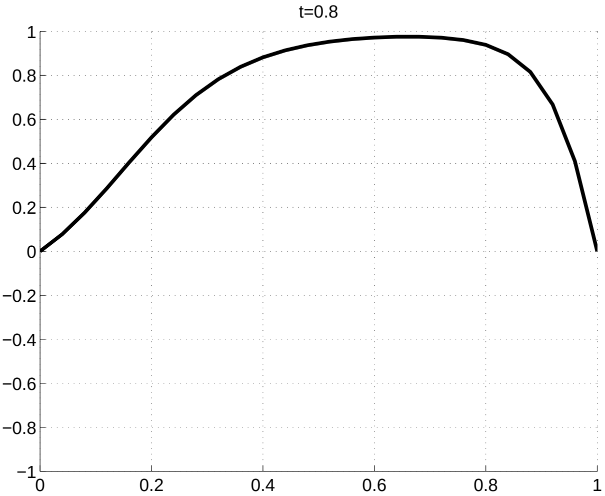
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



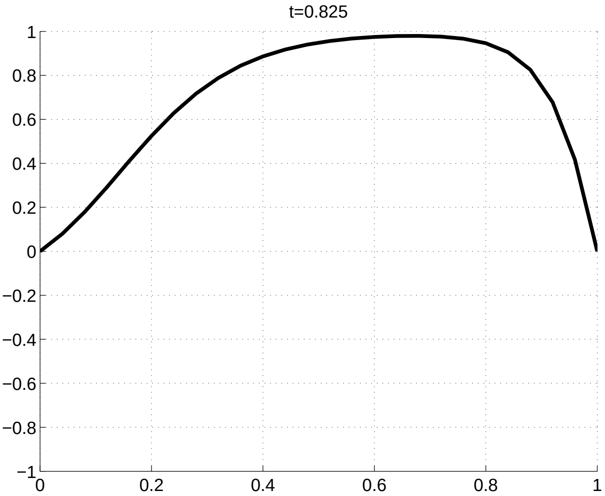
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



uncontrolled ($u \equiv 0$)

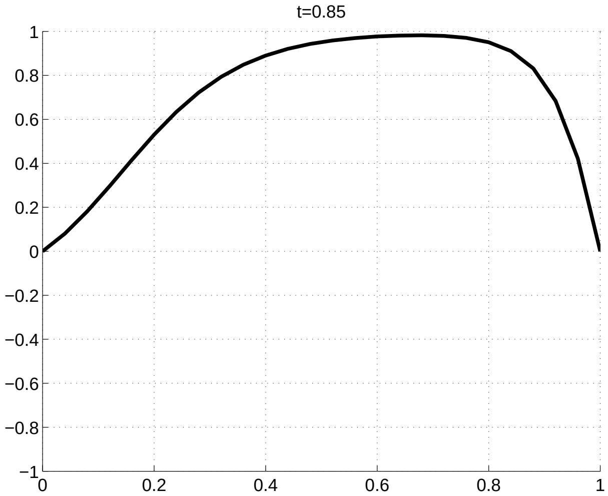
The uncontrolled PDE



uncontrolled ($u \equiv 0$)

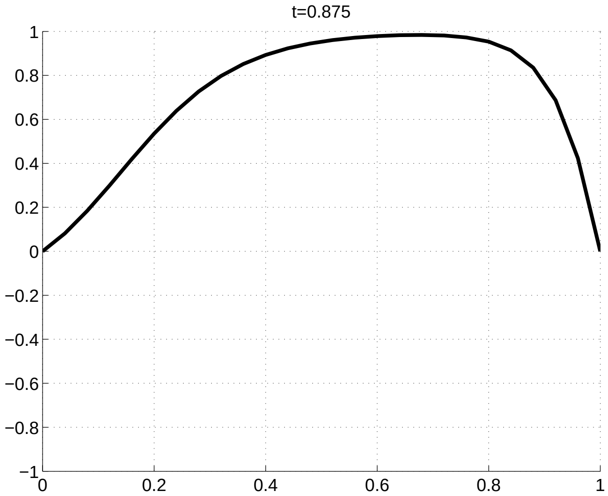
UNIVERSITÄT BAYREUTH

The uncontrolled PDE



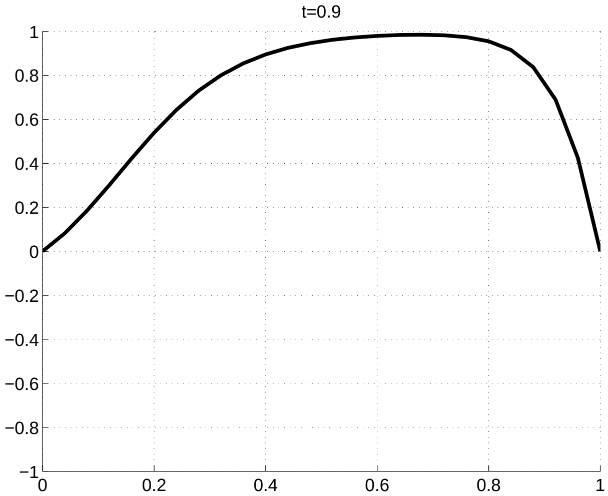
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



uncontrolled ($u \equiv 0$)

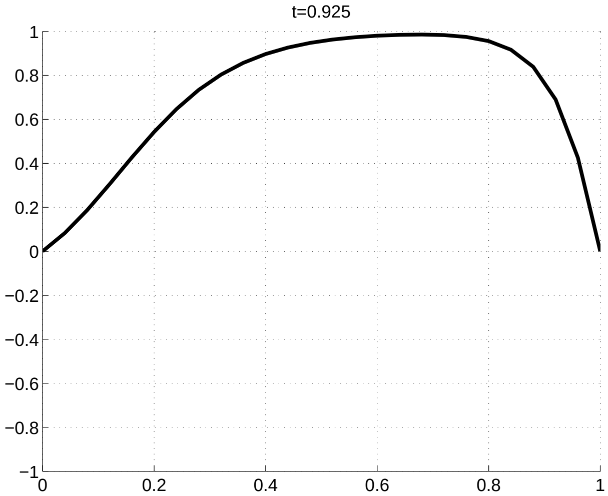
The uncontrolled PDE



uncontrolled $(u \equiv 0)$

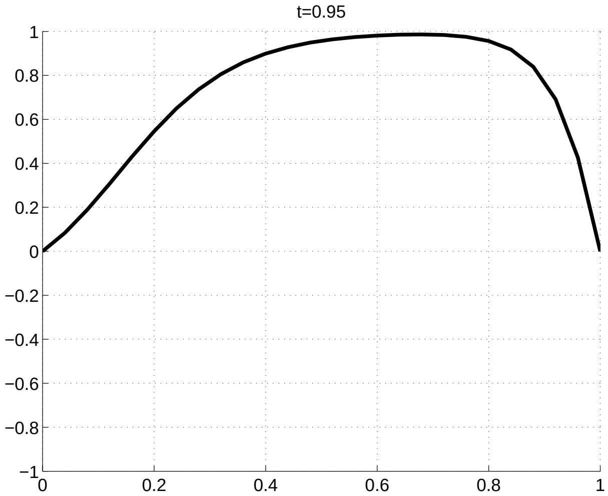
UNIVERSITÄT BAYREUTH

The uncontrolled PDE



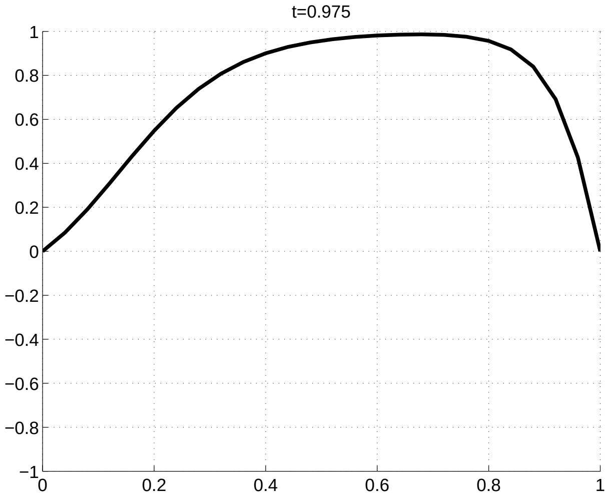
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



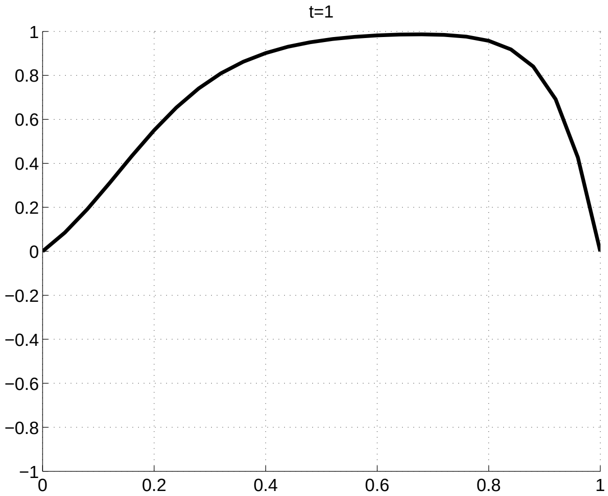
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



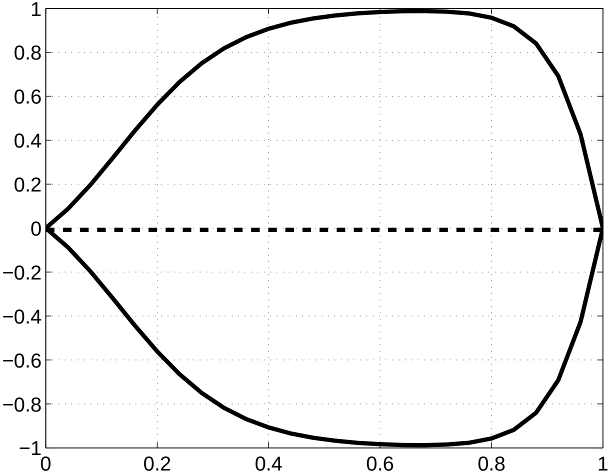
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



uncontrolled ($u \equiv 0$)

The uncontrolled PDE



all equilibrium solutions

MPC for the PDE example

$$y_t = y_x + \nu y_{xx} + \mu y(y+1)(1-y) + u$$

MPC for the PDE example

$$y_t = y_x + \nu y_{xx} + \mu y(y+1)(1-y) + u$$

Goal: stabilize the sampled data solution $y(n, \cdot)$ at $y \equiv 0$

MPC for the PDE example

$$y_t = y_x + \nu y_{xx} + \mu y(y+1)(1-y) + u$$

Goal: stabilize the sampled data solution $y(n,\cdot)$ at $y \equiv 0$

Usual cost: quadratic L^2 cost

$$\ell(y(n,\cdot), u(n,\cdot)) = \|y(n,\cdot)\|_{L^2}^2 + \lambda \|u(n,\cdot)\|_{L^2}^2$$

MPC for the PDE example

$$y_t = y_x + \nu y_{xx} + \mu y(y+1)(1-y) + u$$

Goal: stabilize the sampled data solution $y(n,\cdot)$ at $y \equiv 0$

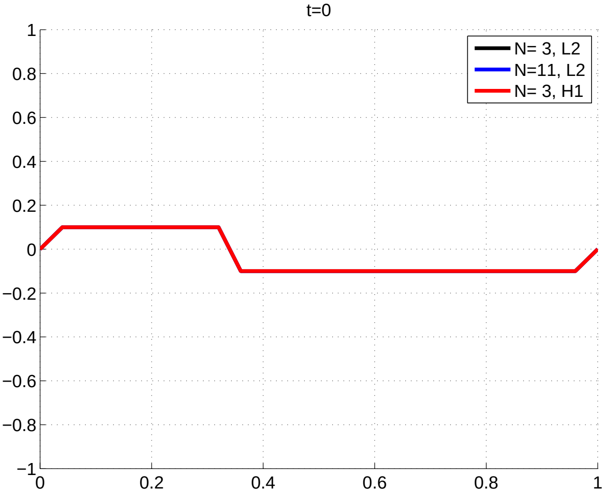
Usual cost: quadratic L^2 cost

$$\ell(y(n,\cdot),u(n,\cdot)) = \|y(n,\cdot)\|_{L^2}^2 + \lambda\|u(n,\cdot)\|_{L^2}^2$$

A small calculation reveals that for this L^2 cost the overshoot C is much larger than for the H^1 cost

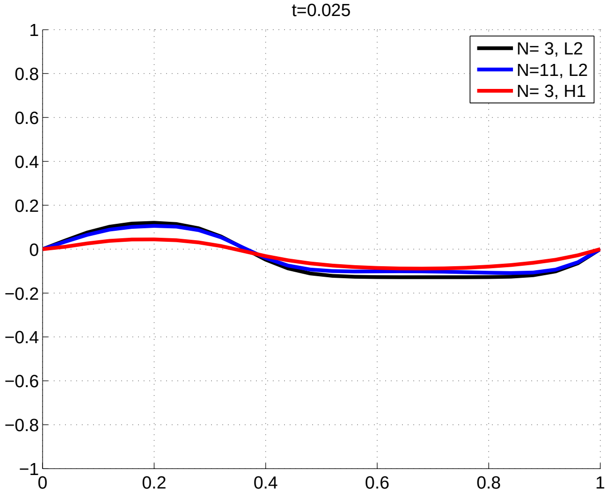
$$\ell(y(n,\cdot),u(n,\cdot)) = \underbrace{\|y(n,\cdot)\|_{L^2}^2 + \|y_x(n,\cdot)\|_{L^2}^2}_{=\|y(n,\cdot)\|_{H^1}^2} + \lambda\|u(n,\cdot)\|_{L^2}^2.$$

MPC with L_2 vs. H_1 cost



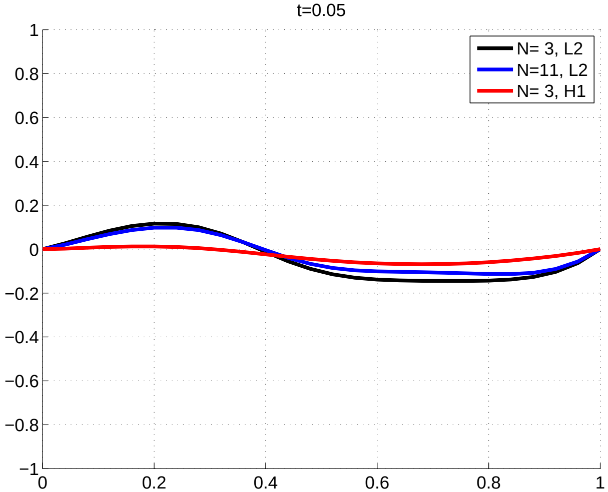
MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

MPC with L_2 vs. H_1 cost



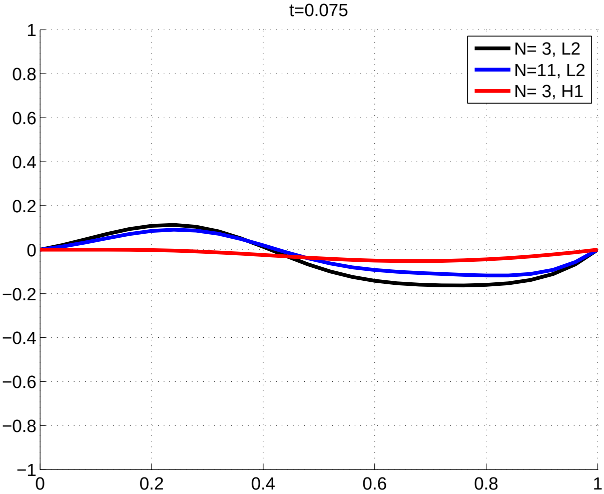
MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

MPC with L_2 vs. H_1 cost



MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

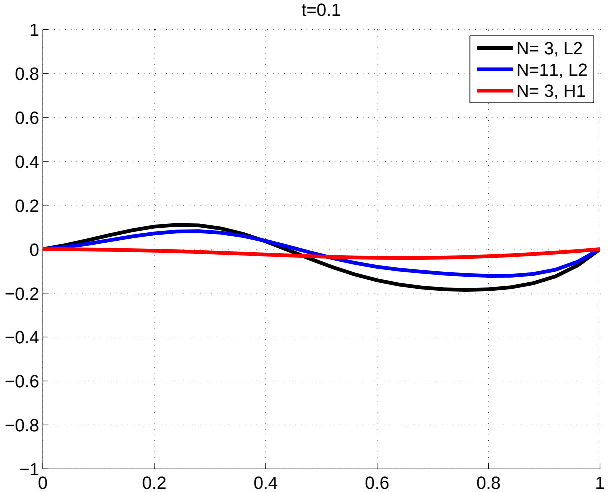
MPC with L_2 vs. H_1 cost



MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

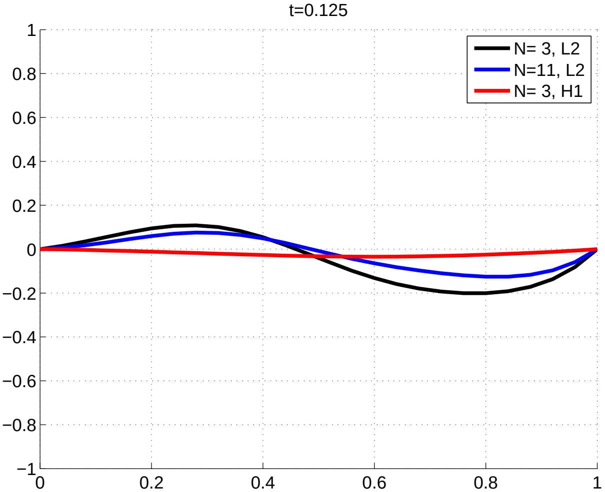
UNIVERSITÄT BAYREUTH

MPC with L_2 vs. H_1 cost



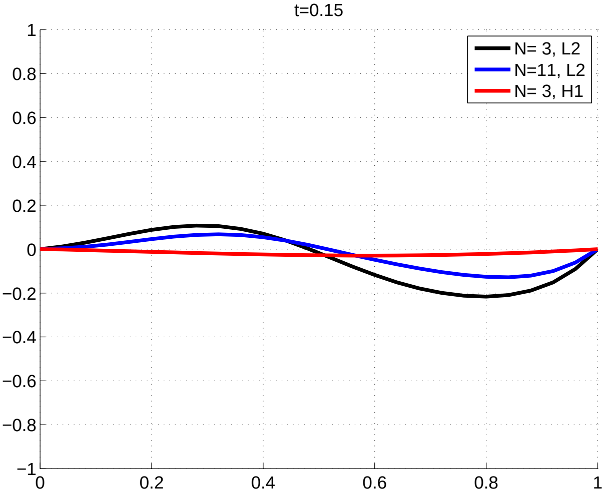
MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

MPC with L_2 vs. H_1 cost



MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

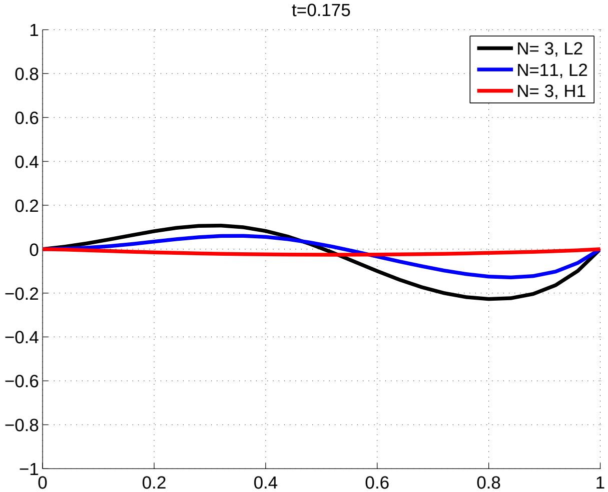
MPC with L_2 vs. H_1 cost



MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

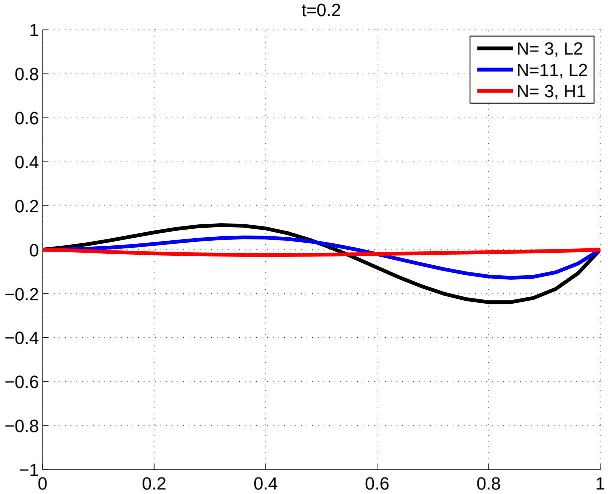
UNIVERSITÄT BAYREUTH

MPC with L_2 vs. H_1 cost



MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

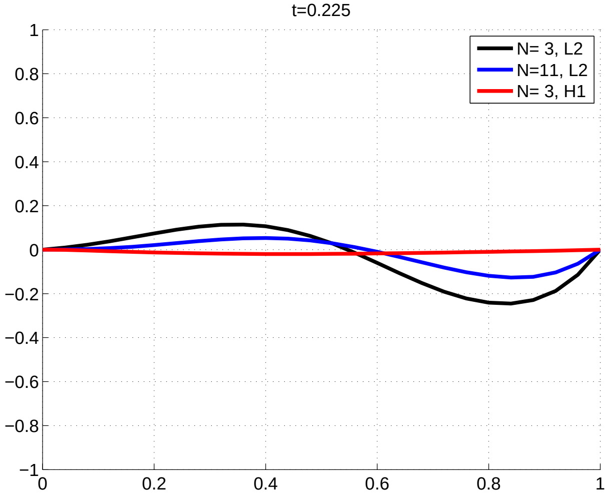
MPC with L_2 vs. H_1 cost



MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

UNIVERSITÄT BAYREUTH

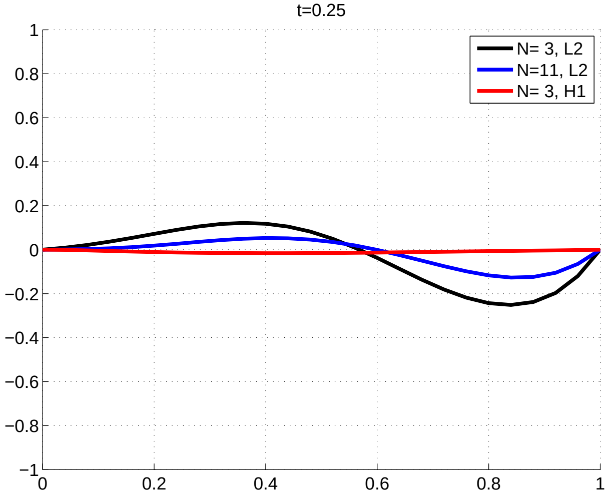
MPC with L_2 vs. H_1 cost



MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

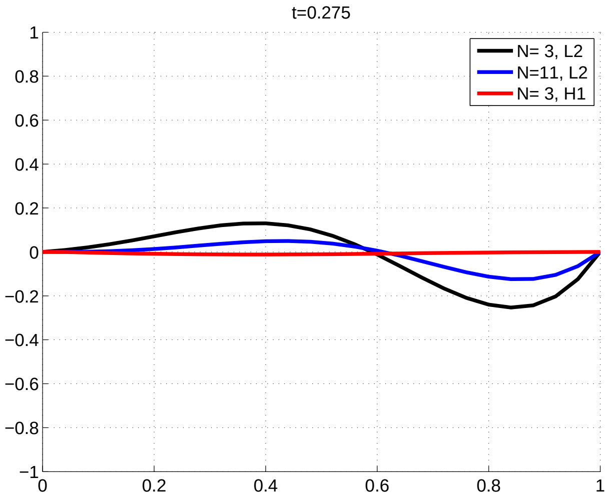
UNIVERSITÄT BAYREUTH

MPC with L_2 vs. H_1 cost



MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

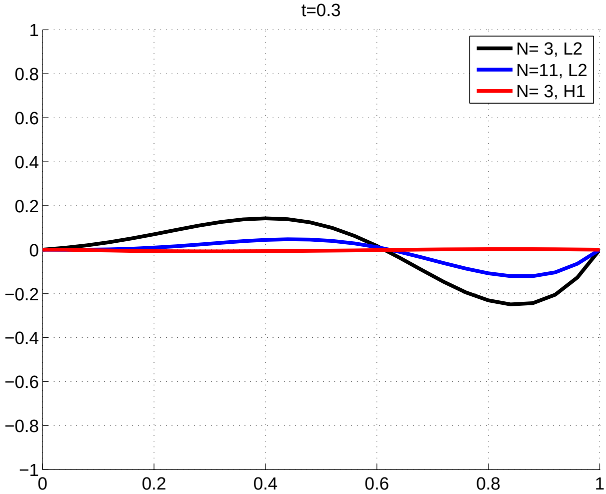
MPC with L_2 vs. H_1 cost



MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

UNIVERSITÄT BAYREUTH

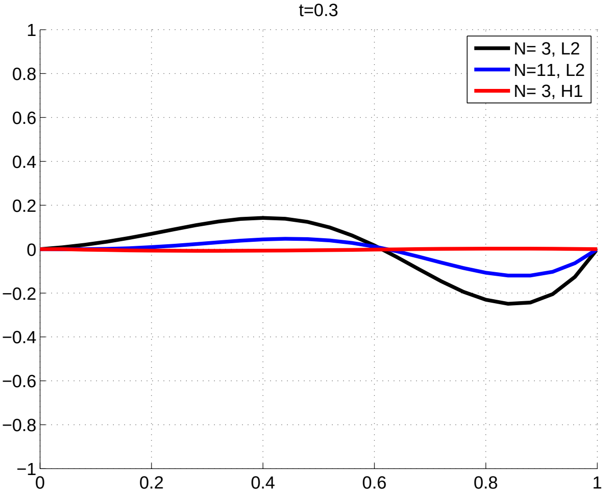
MPC with L_2 vs. H_1 cost



MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

UNIVERSITÄT BAYREUTH

MPC with L_2 vs. H_1 cost



MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

similar results can be obtained for the wave equation

Networked MPC

We now consider an MPC controller implemented via a network

UNIVERSITÄT BAYREUTH

Networked MPC

We now consider an MPC controller implemented via a network

For simplicity of exposition, we neglect

- computation and transmission delay
- disturbance and prediction error

Networked MPC

We now consider an MPC controller implemented via a network

For simplicity of exposition, we neglect

- computation and transmission delay
- disturbance and prediction error

Both can be added to our analysis, cf. [Gr./Pannek/ Worthmann '09] and [Findeisen/Gr., in preparation]

Networked MPC

We now consider an MPC controller implemented via a network

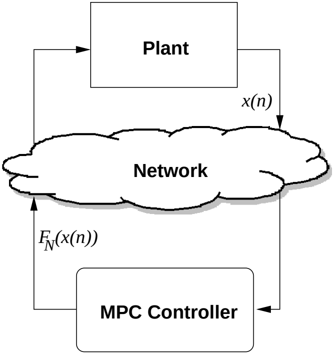
For simplicity of exposition, we neglect

- computation and transmission delay
- disturbance and prediction error

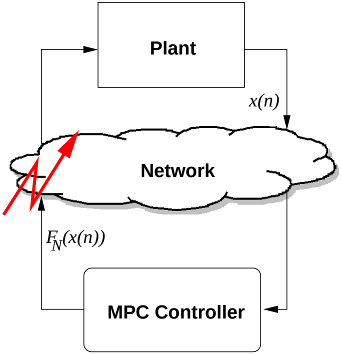
Both can be added to our analysis, cf. [Gr./Pannek/Worthmann '09] and [Findeisen/Gr., in preparation]

Here, we only consider packet loss

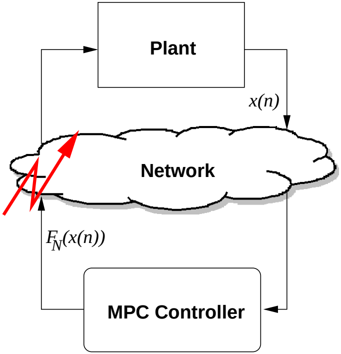
Packet loss



Packet loss

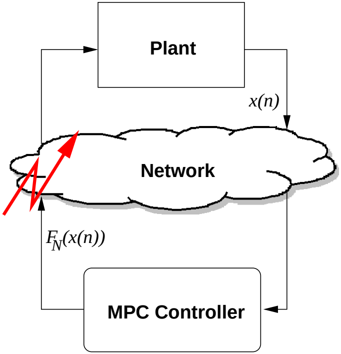


Packet loss



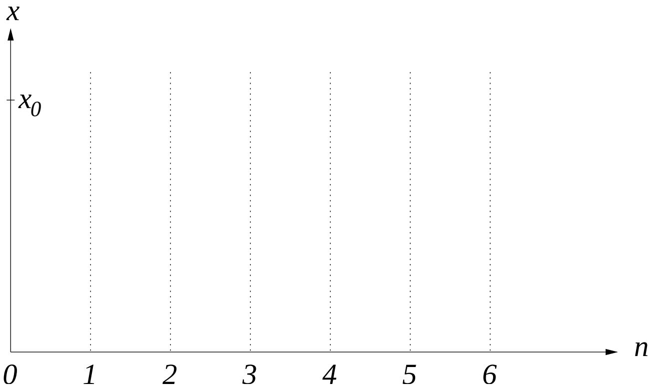
Idea: • send several values of optimal open loop control sequence (instead of just the first value)

Packet loss

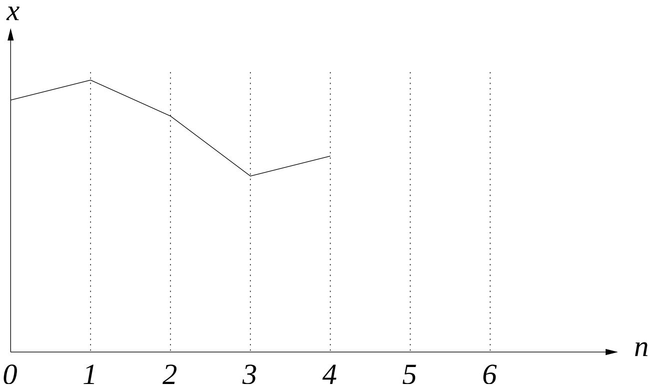


Idea:
- send several values of optimal open loop control sequence (instead of just the first value)
- use these values until next values arrive

Schematic illustration of the idea

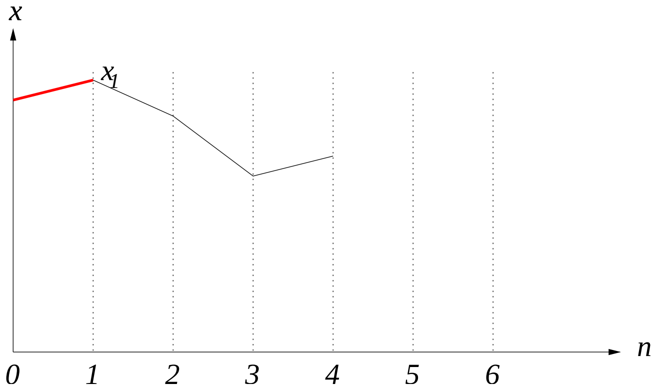


Schematic illustration of the idea



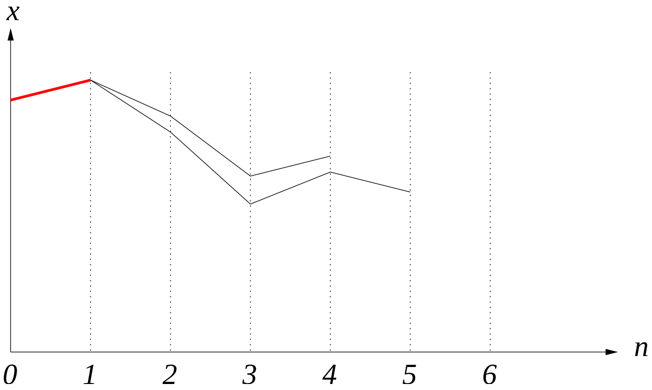
black = predictions (open loop optimization)

Schematic illustration of the idea



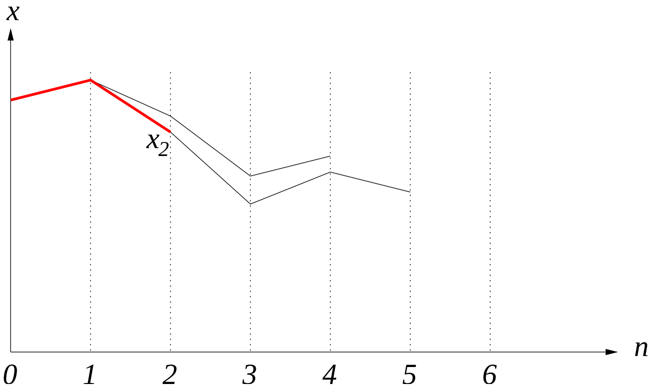
black = predictions (open loop optimization)
red = MPC closed loop

Schematic illustration of the idea



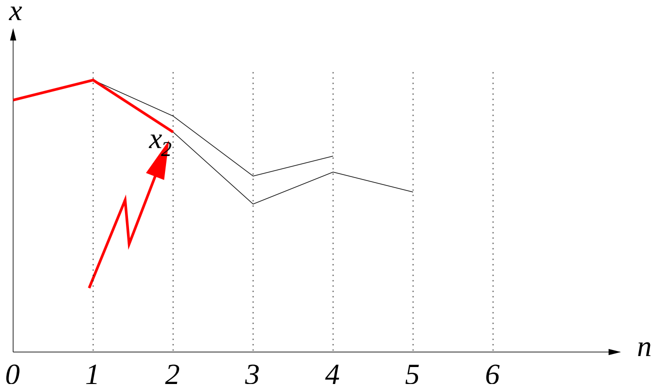
black = predictions (open loop optimization)
red = MPC closed loop

Schematic illustration of the idea



black = predictions (open loop optimization)
red = MPC closed loop

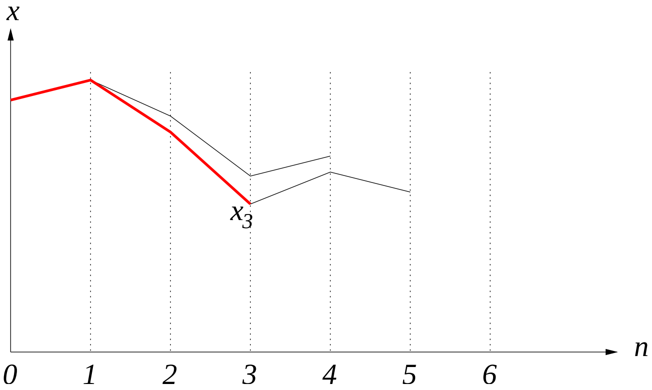
Schematic illustration of the idea



black = predictions (open loop optimization)

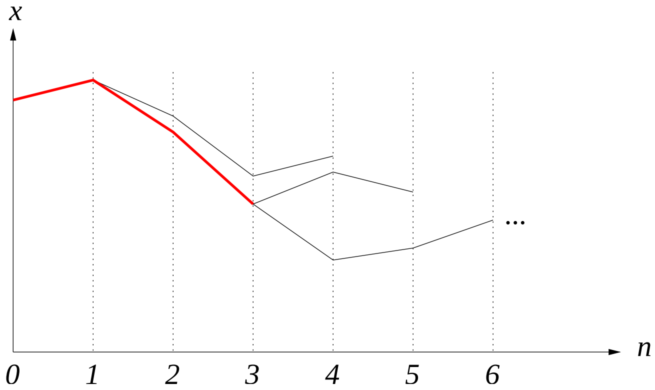
red = MPC closed loop

Schematic illustration of the idea



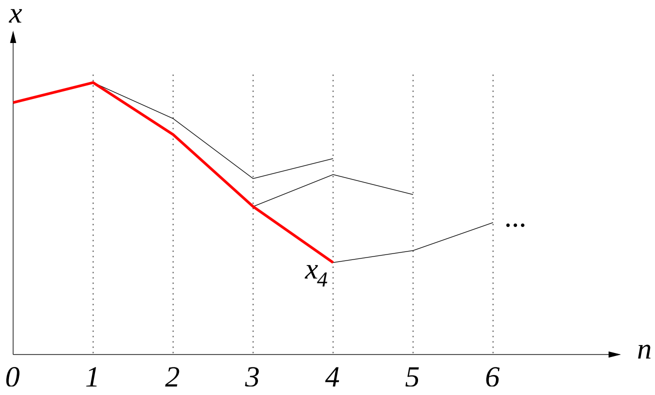
black = predictions (open loop optimization)
red = MPC closed loop

Schematic illustration of the idea



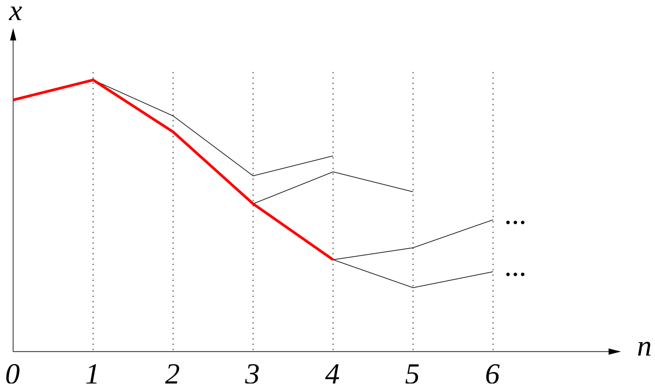
black = predictions (open loop optimization)
red = MPC closed loop

Schematic illustration of the idea



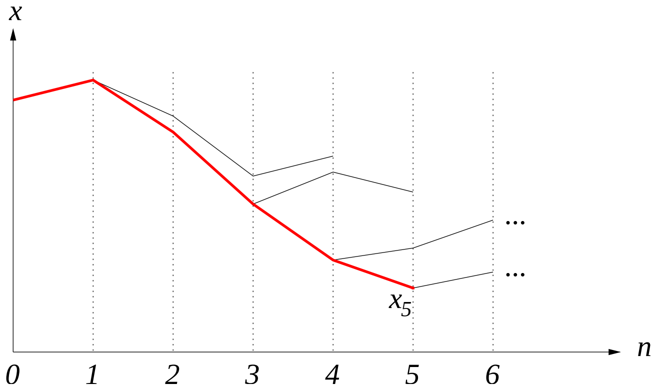
black = predictions (open loop optimization)
red = MPC closed loop

Schematic illustration of the idea



black = predictions (open loop optimization)
red = MPC closed loop

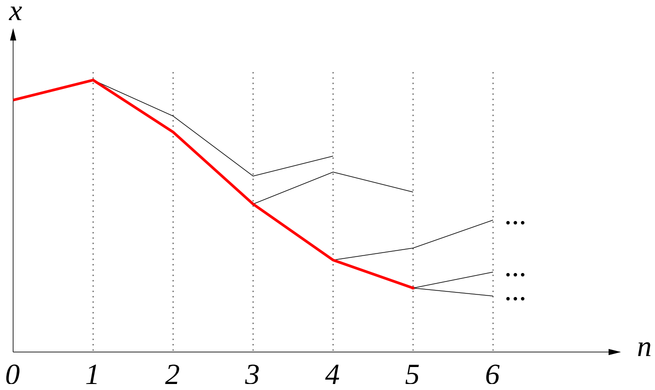
Schematic illustration of the idea



black = predictions (open loop optimization)

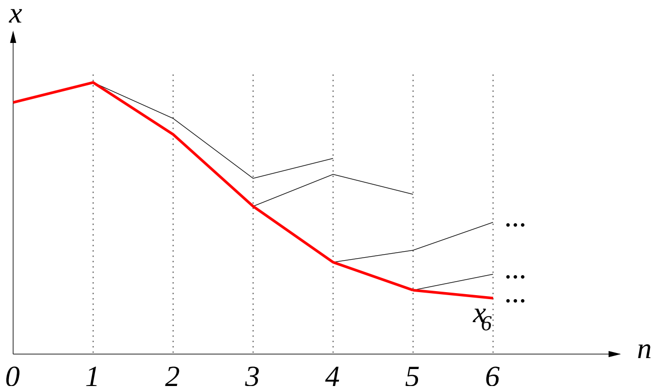
red = MPC closed loop

Schematic illustration of the idea



black = predictions (open loop optimization)
red = MPC closed loop

Schematic illustration of the idea



black = predictions (open loop optimization)

red = MPC closed loop

Rigorous formulation

Denote (successful) transmission times by n_i, $i = 1, 2, \ldots$

UNIVERSITÄT BAYREUTH

Rigorous formulation

Denote (successful) transmission times by n_i, $i = 1, 2, \dots$

Define a buffer length $M \in \mathbb{N}$, $M \leq N - 1$

Rigorous formulation

Denote (successful) transmission times by n_i, $i = 1, 2, \ldots$

Define a buffer length $M \in \mathbb{N}$, $M \leq N - 1$

At each transmission time n_i, the plant receives the feedback control sequence

$$F_N(x(n_i), k) = u^*(k), \quad k = 0, 1, 2, \ldots, M - 1$$

and implements

$$F_N(x_{n_i}, 0),\ F_N(x_{n_i}, 1),\ \ldots,\ F_N(x_{n_i}, m_i - 1)$$

on the control horizon $m_i = n_{i+1} - n_i \leq M$

Rigorous formulation

Denote (successful) transmission times by n_i, $i = 1, 2, \ldots$

Define a buffer length $M \in \mathbb{N}$, $M \leq N - 1$

At each transmission time n_i, the plant receives the feedback control sequence

$$F_N(x(n_i), k) = u^*(k), \quad k = 0, 1, 2, \ldots, M - 1$$

and implements

$$F_N(x_{n_i}, 0),\ F_N(x_{n_i}, 1),\ \ldots,\ F_N(x_{n_i}, m_i - 1)$$

on the control horizon $m_i = n_{i+1} - n_i \leq M$

Note m_i is unknown at time n_i

Stability theorem

Theorem: If there exists $\alpha \in (0,1]$ such that the relaxed dynamic programming inequality

$$V_N(x(m,x_0,u^*)) \le V_N(x) - \alpha \sum_{k=0}^{m-1} \ell(x(m,x_0,u^*),u^*(m))$$

holds for all $m = 1,\ldots,M$, then asymptotic stability follows for the MPC closed loop with arbitrary transmission times n_i, $i \in \mathbb{N}$, satisfying $n_{i+1} - n_i \ge M$.

UNIVERSITÄT BAYREUTH

Stability theorem

Theorem: If there exists $\alpha \in (0,1]$ such that the relaxed dynamic programming inequality

$$V_N(x(m,x_0,u^*)) \le V_N(x) - \alpha \sum_{k=0}^{m-1} \ell(x(m,x_0,u^*),u^*(m))$$

holds for all $m = 1,\ldots,M$, then asymptotic stability follows for the MPC closed loop with arbitrary transmission times n_i, $i \in \mathbb{N}$, satisfying $n_{i+1} - n_i \ge M$.

Furthermore, V_N is Lyapunov function at the transmission times n_i and we get the suboptimality estimate

$$J_\infty(x,F_N) \le V_\infty(x)/\alpha$$

UNIVERSITÄT BAYREUTH

Computation of α

Again, for C, σ-exponentially controllable systems, this $\alpha = \alpha(C, \sigma, N, m)$ can be explicitly computed:

$$\alpha = 1 - \frac{\prod_{i=m+1}^{N} (\gamma_i - 1) \prod_{i=N-m+1}^{N} (\gamma_i - 1)}{\left(\prod_{i=m+1}^{N} \gamma_i - \prod_{i=m+1}^{N} (\gamma_i - 1)\right)\left(\prod_{i=N-m+1}^{N} \gamma_i - \prod_{i=N-m+1}^{N} (\gamma_i - 1)\right)}$$

with $\gamma_i = \sum_{k=0}^{i-1} C\sigma^k$

UNIVERSITÄT BAYREUTH

Computation of α

Again, for C, σ-exponentially controllable systems, this $\alpha = \alpha(C, \sigma, N, m)$ can be explicitly computed:

$$\alpha = 1 - \frac{\prod_{i=m+1}^{N} (\gamma_i - 1) \prod_{i=N-m+1}^{N} (\gamma_i - 1)}{\left(\prod_{i=m+1}^{N} \gamma_i - \prod_{i=m+1}^{N} (\gamma_i - 1)\right)\left(\prod_{i=N-m+1}^{N} \gamma_i - \prod_{i=N-m+1}^{N} (\gamma_i - 1)\right)}$$

with $\gamma_i = \sum_{k=0}^{i-1} C\sigma^k$

Note: at first, this yields a stability criterion for each fixed control horizon m only.

Computation of α

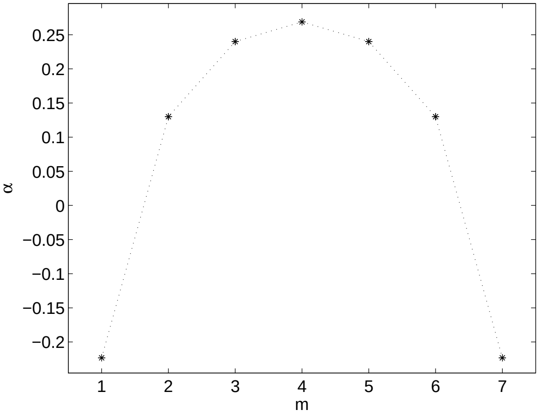
Again, for C, σ-exponentially controllable systems, this $\alpha = \alpha(C, \sigma, N, m)$ can be explicitly computed:

$$\alpha = 1 - \frac{\prod_{i=m+1}^{N} (\gamma_i - 1) \prod_{i=N-m+1}^{N} (\gamma_i - 1)}{\left(\prod_{i=m+1}^{N} \gamma_i - \prod_{i=m+1}^{N} (\gamma_i - 1)\right) \left(\prod_{i=N-m+1}^{N} \gamma_i - \prod_{i=N-m+1}^{N} (\gamma_i - 1)\right)}$$

with $\gamma_i = \sum_{k=0}^{i-1} C\sigma^k$

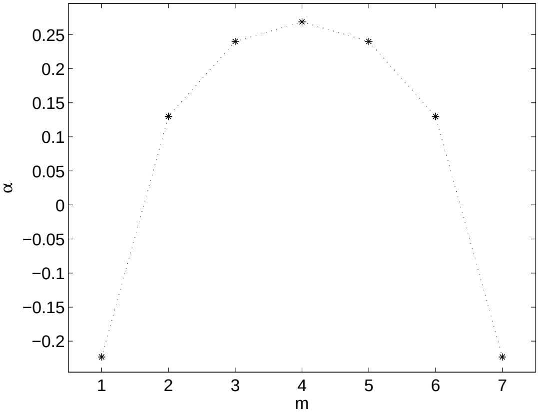
Note: at first, this yields a stability criterion for each fixed control horizon m only. However, since we get a common Lyapunov function V_N, stability carries over to varying control horizon m_i

Example



$\alpha(C, \sigma, N, m)$ for $C = 2$, $\sigma = 0.68$, $N = 8$, $m = 1, \ldots, 7$

Example



$\alpha(C,\sigma,N,m)$ for $C=2$, $\sigma=0.68$, $N=8$, $m=1,\ldots,7$

This symmetry and monotonicity is not a coincidence

Properties of α

Theorem: The values $\alpha(N,m)$ satisfy

$$\alpha(N,m) = \alpha(N,N-m), \;\; m = 1,\ldots,N-1$$

and

$$\alpha(N,m) \le \alpha(N,m+1), \;\; m = 1,\ldots \lceil N/2 \rceil$$

Properties of α

Theorem: The values $\alpha(N,m)$ satisfy

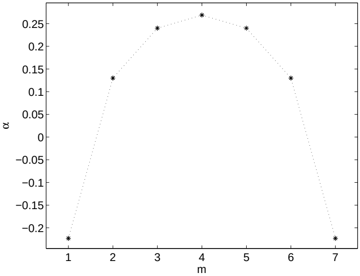
$$\alpha(N,m) = \alpha(N, N-m), \;\; m = 1, \ldots, N-1$$

and

$$\alpha(N,m) \le \alpha(N, m+1), \;\; m = 1, \ldots \lceil N/2 \rceil$$

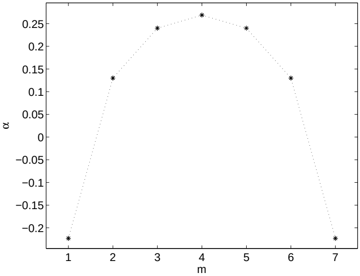
Corollary: If N is such that all C, σ-exponentially controllable systems are stabilized with "classical" MPC ($m = 1$), then they are stabilized for arbitrary varying control horizons $m_i \in \{1, \ldots, N-1\}$

Monotonicity in simulation examples



The monotonicity means that when enlarging the control horizon the closed loop performance first improves and then becomes worse, again

Monotonicity in simulation examples



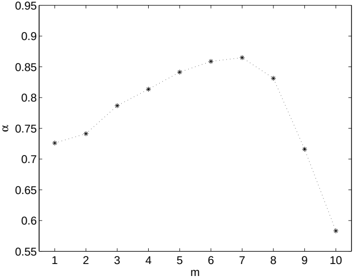
The monotonicity means that when enlarging the control horizon the closed loop performance first improves and then becomes worse, again

While we cannot exactly recover the symmetry in simulation examples, we can observe this monotonicity

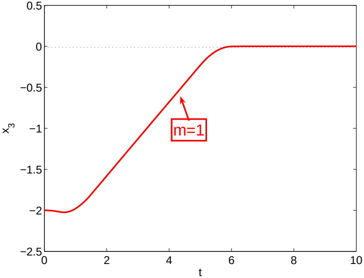
Example: linearized inverted pendulum

$$\dot{x} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ g & -k & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} u, \qquad x_0 = \begin{pmatrix} 0 \\ 0 \\ -2 \\ 0 \end{pmatrix}$$

sampling time $T = 0.5$, $\ell(x,u) = 2\|x\|_1 + 4\|u\|_1$, $N = 11$



α after 1 MPC step

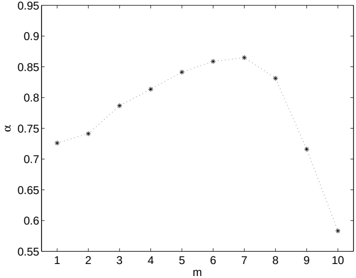


x_3 component of trajectory

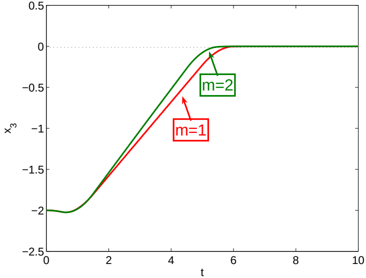
Example: linearized inverted pendulum

$$\dot{x} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ g & -k & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} u, \qquad x_0 = \begin{pmatrix} 0 \\ 0 \\ -2 \\ 0 \end{pmatrix}$$

sampling time $T = 0.5$, $\ell(x,u) = 2\|x\|_1 + 4\|u\|_1$, $N = 11$



α after 1 MPC step



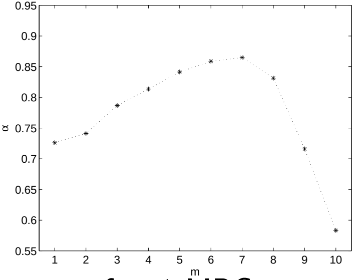
x_3 component of trajectory

UNIVERSITÄT BAYREUTH

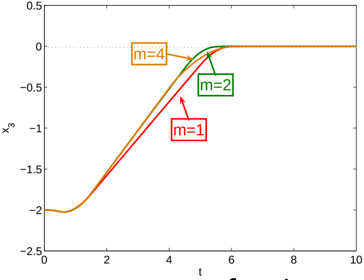
Example: linearized inverted pendulum

$$\dot{x} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ g & -k & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} u, \qquad x_0 = \begin{pmatrix} 0 \\ 0 \\ -2 \\ 0 \end{pmatrix}$$

sampling time $T = 0.5$, $\ell(x,u) = 2\|x\|_1 + 4\|u\|_1$, $N = 11$



α after 1 MPC step

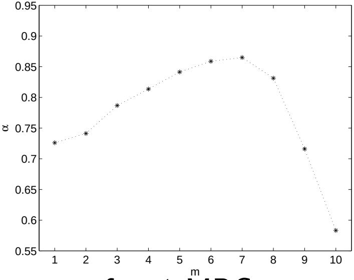


x_3 component of trajectory

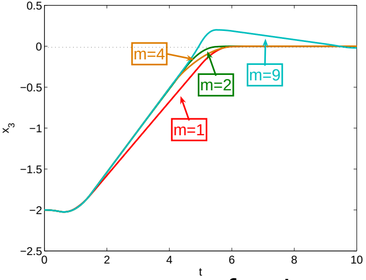
Example: linearized inverted pendulum

$$\dot{x} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ g & -k & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} u, \qquad x_0 = \begin{pmatrix} 0 \\ 0 \\ -2 \\ 0 \end{pmatrix}$$

sampling time $T = 0.5$, $\ell(x,u) = 2\|x\|_1 + 4\|u\|_1$, $N = 11$



α after 1 MPC step



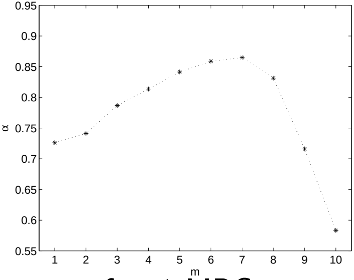
x_3 component of trajectory

UNIVERSITÄT BAYREUTH

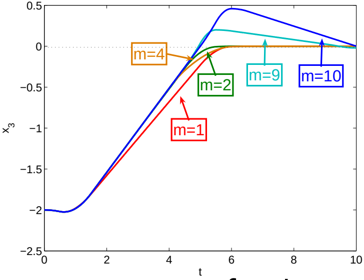
Example: linearized inverted pendulum

$$\dot{x} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ g & -k & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} u, \qquad x_0 = \begin{pmatrix} 0 \\ 0 \\ -2 \\ 0 \end{pmatrix}$$

sampling time $T = 0.5$, $\ell(x,u) = 2\|x\|_1 + 4\|u\|_1$, $N = 11$



α after 1 MPC step



x_3 component of trajectory

UNIVERSITÄT BAYREUTH

Relaxed dynamic programming in distributed MPC — some ideas

In distributed MPC schemes, the online optimization is carried out locally in the subsystems

Relaxed dynamic programming in distributed MPC — some ideas

In distributed MPC schemes, the online optimization is carried out locally in the subsystems

To this end

- the optimization in each time instant has to be localized, i.e., split up

Relaxed dynamic programming in distributed MPC — some ideas

In distributed MPC schemes, the online optimization is carried out locally in the subsystems

To this end

- the optimization in each time instant has to be localized, i.e., split up
- some communication between the subsystems is necessary, often restricted to communication with neighbors

Relaxed dynamic programming in distributed MPC — some ideas

In distributed MPC schemes, the online optimization is carried out locally in the subsystems

To this end

- the optimization in each time instant has to be localized, i.e., split up
- some communication between the subsystems is necessary, often restricted to communication with neighbors

Structural properties, e.g., whether the subsystems are physically coupled or only coupled via the optimization objective play an important role for the design of the scheme

Communication during optimization

If we assume sufficiently fast communication, then subsystems may exchange information during the iterative optimization in each time step

UNIVERSITÄT BAYREUTH

Communication during optimization

If we assume sufficiently fast communication, then subsystems may exchange information during the iterative optimization in each time step

Suitable decoupling of the problem leads to the central optimum (whereas naïve decoupling may only lead to a Nash equilibrium, cf. [Venkat/Rawlings/Wright '05])

Communication during optimization

If we assume sufficiently fast communication, then subsystems may exchange information during the iterative optimization in each time step

Suitable decoupling of the problem leads to the central optimum (whereas naïve decoupling may only lead to a Nash equilibrium, cf. [Venkat/Rawlings/Wright '05])

Optimization iteration is slowed down by communication
$\rightsquigarrow$ termination before convergence may be necessary

Communication during optimization

If we assume sufficiently fast communication, then subsystems may exchange information during the iterative optimization in each time step

Suitable decoupling of the problem leads to the central optimum (whereas naïve decoupling may only lead to a Nash equilibrium, cf. [Venkat/Rawlings/Wright '05])

Optimization iteration is slowed down by communication
$\rightsquigarrow$ termination before convergence may be necessary

Ideas: Use a relaxed dynamic programming type condition

$$\widetilde{V}_N(f(x, \widetilde{F}_N(x))) \leq \widetilde{V}_N(x) - \bar{\alpha}\ell(x, \widetilde{F}_N(x))$$

for some predefined $\bar{\alpha} \in (0, 1)$ as termination criterion.

Communication during optimization

If we assume sufficiently fast communication, then subsystems may exchange information during the iterative optimization in each time step

Suitable decoupling of the problem leads to the central optimum (whereas naïve decoupling may only lead to a Nash equilibrium, cf. [Venkat/Rawlings/Wright '05])

Optimization iteration is slowed down by communication
$\rightsquigarrow$ termination before convergence may be necessary

Ideas: Use a relaxed dynamic programming type condition

$$\widetilde{V}_N(f(x, \widetilde{F}_N(x))) \leq \widetilde{V}_N(x) - \bar{\alpha}\ell(x, \widetilde{F}_N(x))$$

for some predefined $\bar{\alpha} \in (0, 1)$ as termination criterion.
Design the optimzation such that this condition is satisfied after few iterations.

No communication during optimization

For a multi-vehicle formation problem, [Dunbar/Murray '06] showed stability for an NMPC scheme in which only the open loop optimized predicted trajectories are communicated (once in each time step)

No communication during optimization

For a multi-vehicle formation problem, [Dunbar/Murray '06] showed stability for an NMPC scheme in which only the open loop optimized predicted trajectories are communicated (once in each time step)

The stability proof heavily relies on

- stabilizing terminal constraints

No communication during optimization

For a multi-vehicle formation problem, [Dunbar/Murray '06] showed stability for an NMPC scheme in which only the open loop optimized predicted trajectories are communicated (once in each time step)

The stability proof heavily relies on

- stabilizing terminal constraints
- additional (severe) constraints on the predicted trajectory in the optimization

No communication during optimization

For a multi-vehicle formation problem, [Dunbar/Murray '06] showed stability for an NMPC scheme in which only the open loop optimized predicted trajectories are communicated (once in each time step)

The stability proof heavily relies on

- stabilizing terminal constraints
- additional (severe) constraints on the predicted trajectory in the optimization

In addition, since there is no information exchange during the optimization, only some kind of Nash equilibrium (instead of a central optimum) can be expected

No communication during optimization

For a multi-vehicle formation problem, [Dunbar/Murray '06] showed stability for an NMPC scheme in which only the open loop optimized predicted trajectories are communicated (once in each time step)

The stability proof heavily relies on

- stabilizing terminal constraints
- additional (severe) constraints on the predicted trajectory in the optimization

In addition, since there is no information exchange during the optimization, only some kind of Nash equilibrium (instead of a central optimum) can be expected

Idea: Replace the constraints by suitable "decentralized" controllability conditions under which stability and — if possible — suboptimality can be shown with our techniques

Asynchronous Optimization

So far, we implicitly assumed that optimization in the the subsystems is performed synchronously

However, for various reasons it may be preferable that different subsystems optimize at different time instances

UNIVERSITÄT BAYREUTH

Asynchronous Optimization

So far, we implicitly assumed that optimization in the the subsystems is performed synchronously

However, for various reasons it may be preferable that different subsystems optimize at different time instances

This is similar to the networked setting, but now the times n_i when switching from one open loop sequence to another are different in each subsystem

Asynchronous Optimization

So far, we implicitly assumed that optimization in the the subsystems is performed synchronously

However, for various reasons it may be preferable that different subsystems optimize at different time instances

This is similar to the networked setting, but now the times n_i when switching from one open loop sequence to another are different in each subsystem

Idea: use similar techniques as in the networked setting in order to analyze stability and suboptimality

Summary and Conclusions

- based on a simple relaxed dynamic programming inequality we developed a stability and guaranteed performance analysis method for MPC schemes without having to impose stabilizing terminal constraints

Summary and Conclusions

- based on a simple relaxed dynamic programming inequality we developed a stability and guaranteed performance analysis method for MPC schemes without having to impose stabilizing terminal constraints
- with this method we can compute optimization horizon bounds N via explicit analytical formulas

Summary and Conclusions

- based on a simple relaxed dynamic programming inequality we developed a stability and guaranteed performance analysis method for MPC schemes without having to impose stabilizing terminal constraints
- with this method we can compute optimization horizon bounds N via explicit analytical formulas
- these analytic results allow for qualitative statements even if only rough quantitative information is available

Summary and Conclusions

- based on a simple relaxed dynamic programming inequality we developed a stability and guaranteed performance analysis method for MPC schemes without having to impose stabilizing terminal constraints
- with this method we can compute optimization horizon bounds N via explicit analytical formulas
- these analytic results allow for qualitative statements even if only rough quantitative information is available
- the method can be extended to varying control horizons $m_i \in \{1, \ldots, M\}$ and shows that larger and varying control horizons can be used without losing (nominal) stability and performance

UNIVERSITÄT BAYREUTH

Summary and Conclusions

- based on a simple relaxed dynamic programming inequality we developed a stability and guaranteed performance analysis method for MPC schemes without having to impose stabilizing terminal constraints
- with this method we can compute optimization horizon bounds N via explicit analytical formulas
- these analytic results allow for qualitative statements even if only rough quantitative information is available
- the method can be extended to varying control horizons $m_i \in \{1, \ldots, M\}$ and shows that larger and varying control horizons can be used without losing (nominal) stability and performance
- we expect various applications in distributed MPC, both analytic and algorithmic

Summary and Conclusions

- based on a simple relaxed dynamic programming inequality we developed a stability and guaranteed performance analysis method for MPC schemes without having to impose stabilizing terminal constraints
- with this method we can compute optimization horizon bounds N via explicit analytical formulas
- these analytic results allow for qualitative statements even if only rough quantitative information is available
- the method can be extended to varying control horizons $m_i \in \{1, \ldots, M\}$ and shows that larger and varying control horizons can be used without losing (nominal) stability and performance
- we expect various applications in distributed MPC, both analytic and algorithmic