Stability and robustness of nonlinear predictive control without stabilizing terminal constraints

Lars Grüne, Anders Rantzer, Nils Altmüller, Thomas Jahn, Jürgen Pannek, Karl Worthmann

Stability and robustness of nonlinear predictive control without stabilizing terminal constraints

Lars Grüne

Mathematisches Institut, Universität Bayreuth

in collaboration with

Anders Rantzer (Lund)
Nils Altmüller (Bayreuth), Thomas Jahn (Bayreuth),
Jürgen Pannek (Perth), Karl Worthmann (Bayreuth)

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

SADCO Kick-Off Meeting, Paris, March 2011

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: feedback stabilization

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 feedback stabilization via infinite horizon optimal control:

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

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

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 feedback stabilization via infinite horizon optimal control:

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

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

possibly subject to state/control constraints

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

Basic moving horizon MPC concept:

Model predictive control

Basic moving horizon MPC concept:

At each time instant n solve for the current state $x = x(n)$

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

Model predictive control

Basic moving horizon MPC concept:

At each time instant n solve for the current state $x = x(n)$

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

$\rightsquigarrow$ optimal trajectory $\quad x^{opt}(0), \ldots, x^{opt}(N-1)$

Model predictive control

Basic moving horizon MPC concept:

At each time instant n solve for the current state $x = x(n)$

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

$\rightsquigarrow$ optimal trajectory $\quad x^{opt}(0), \ldots, x^{opt}(N-1)$

with optimal control $u^{opt}(0), \ldots, u^{opt}(N-1)$

Model predictive control

Basic moving horizon MPC concept:

At each time instant n solve for the current state $x = x(n)$

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

$\rightsquigarrow$ optimal trajectory $x^{opt}(0), \ldots, x^{opt}(N-1)$

with optimal control $u^{opt}(0), \ldots, u^{opt}(N-1)$

$\rightsquigarrow$ MPC feedback law $F_N(x(n)) := u^{opt}(0)$

Model predictive control

Basic moving horizon MPC concept:

At each time instant n solve for the current state $x = x(n)$

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

$\rightsquigarrow$ optimal trajectory $\quad x^{opt}(0), \dots, x^{opt}(N-1)$

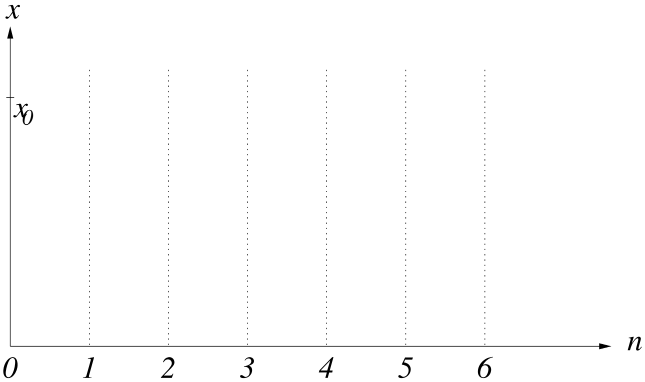
with optimal control $u^{opt}(0), \dots, u^{opt}(N-1)$

$\rightsquigarrow$ MPC feedback law $F_N(x(n)) := u^{opt}(0)$

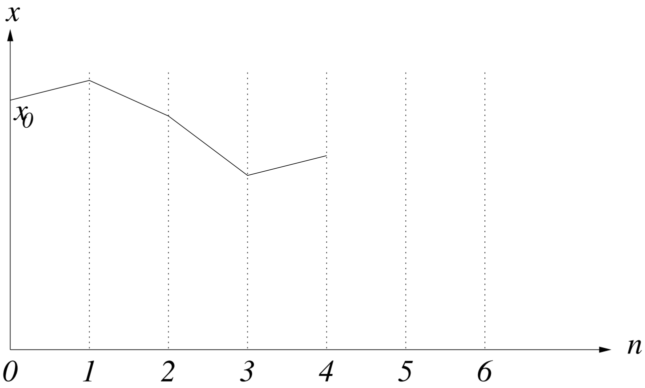
$\rightsquigarrow$ feedback controlled system ("closed loop")

$$x(n+1) = f(x(n), F_N(x(n))) = f(x^{opt}(0), u^{opt}(0)) = x^{opt}(1)$$

MPC from the trajectory point of view

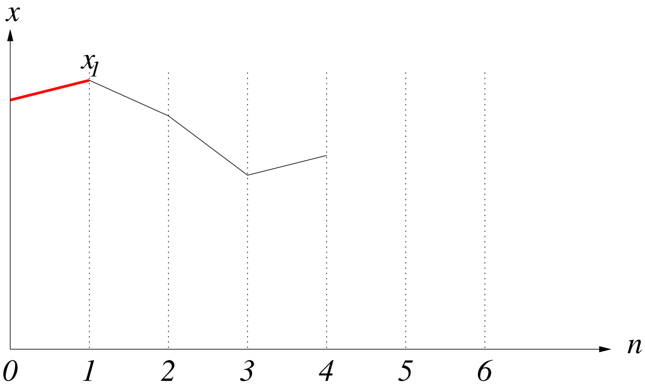


MPC from the trajectory point of view



black = predictions (open loop optimization)

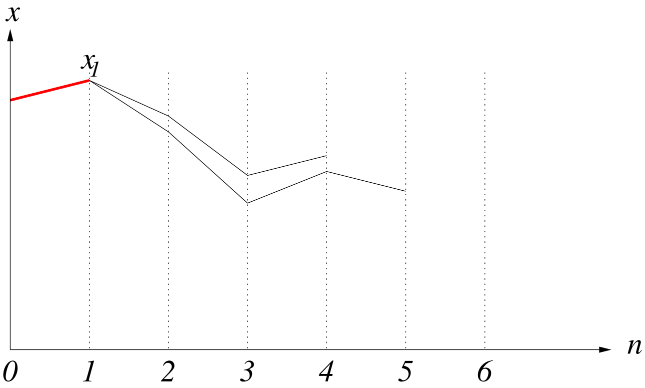
MPC from the trajectory point of view



black = predictions (open loop optimization)

red = MPC closed loop $x(n+1) = f(x(n), F_N(x(n)))$

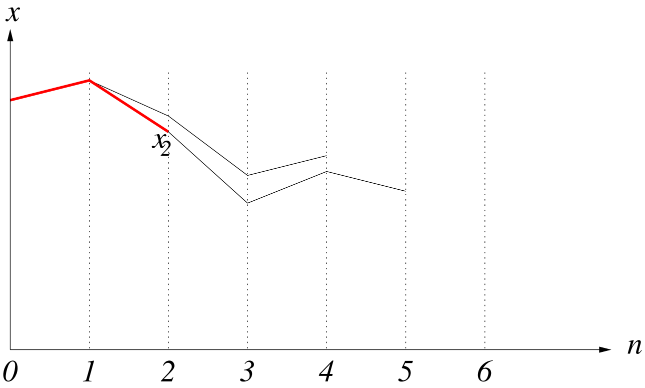
MPC from the trajectory point of view



black = predictions (open loop optimization)

red = MPC closed loop $x(n+1) = f(x(n), F_N(x(n)))$

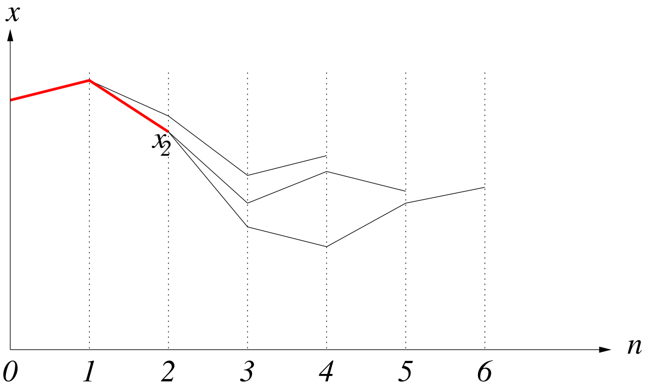
MPC from the trajectory point of view



black = predictions (open loop optimization)

red = MPC closed loop $\quad x(n+1) = f(x(n), F_N(x(n)))$

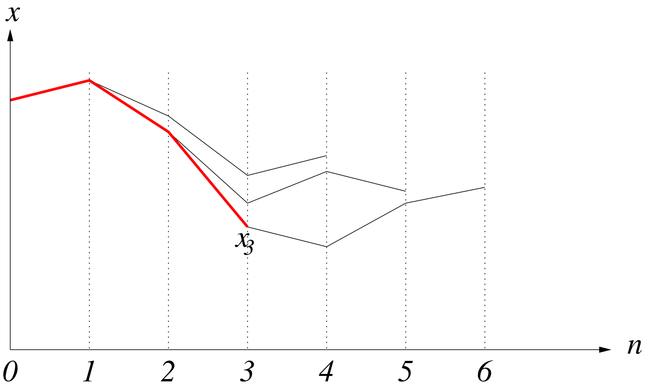
MPC from the trajectory point of view



black = predictions (open loop optimization)

red = MPC closed loop $\quad x(n+1) = f(x(n), F_N(x(n)))$

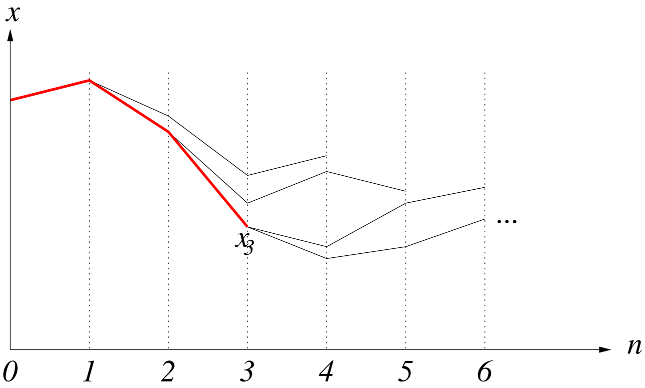
MPC from the trajectory point of view



black = predictions (open loop optimization)

red = MPC closed loop $\quad x(n+1) = f(x(n), F_N(x(n)))$

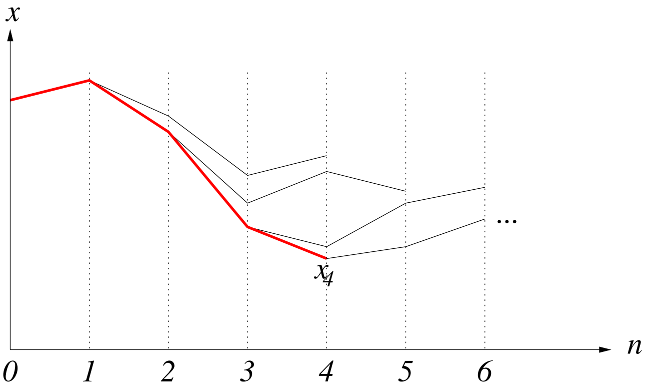
MPC from the trajectory point of view



black = predictions (open loop optimization)

red = MPC closed loop $\quad x(n+1) = f(x(n), F_N(x(n)))$

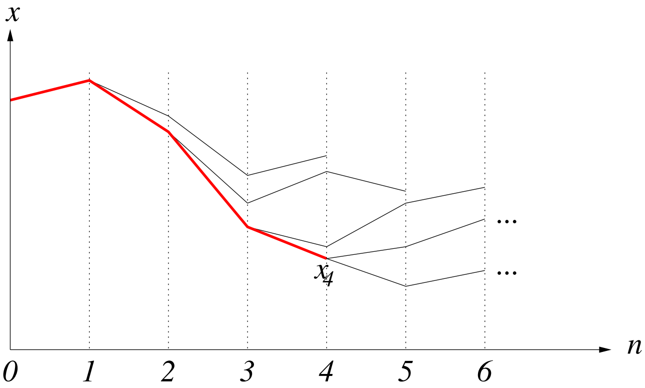
MPC from the trajectory point of view



black = predictions (open loop optimization)

red = MPC closed loop $\quad x(n+1) = f(x(n), F_N(x(n)))$

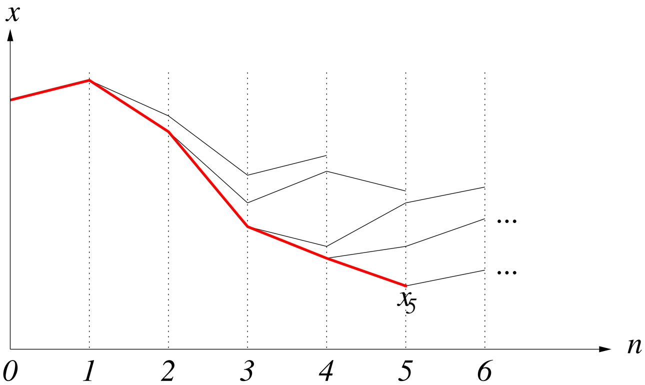
MPC from the trajectory point of view



black = predictions (open loop optimization)

red = MPC closed loop $\quad x(n+1) = f(x(n), F_N(x(n)))$

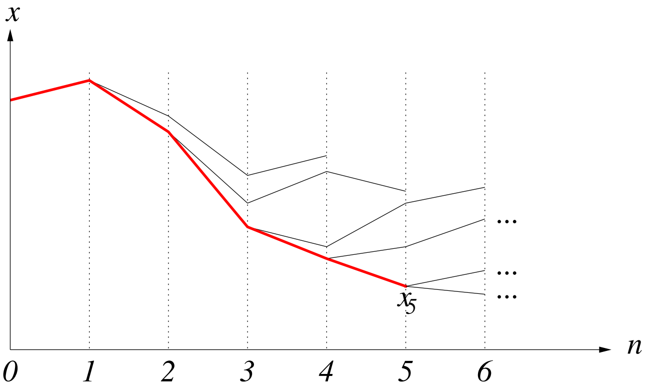
MPC from the trajectory point of view



black = predictions (open loop optimization)

red = MPC closed loop $\quad x(n+1) = f(x(n), F_N(x(n)))$

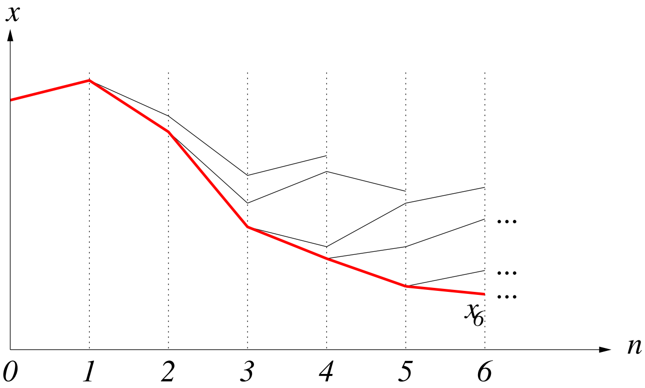
MPC from the trajectory point of view



black = predictions (open loop optimization)

red = MPC closed loop $\quad x(n+1) = f(x(n), F_N(x(n)))$

MPC from the trajectory point of view



black = predictions (open loop optimization)

red = MPC closed loop $\quad x(n+1) = f(x(n), F_N(x(n)))$

MPC: Questions

Questions:

- When does MPC stabilize the system?

MPC: Questions

Questions:

- When does MPC stabilize the system?
- How good is the MPC Feedback law compared to the infinite horizon optimal solution?

MPC: Questions

Questions:

- When does MPC stabilize the system?
- How good is the MPC Feedback law compared to the infinite horizon optimal solution?
- How robust is the MPC Feedback law with respect to perturbations?

MPC: Questions

Questions:

- When does MPC stabilize the system?
- How good is the MPC Feedback law compared to the infinite horizon optimal solution?
- How robust is the MPC Feedback law with respect to perturbations?
- How can we reduce the computational effort?

MPC: Questions

Questions:

- When does MPC stabilize the system?
- How good is the MPC Feedback law compared to the infinite horizon optimal solution?
- How robust is the MPC Feedback law with respect to perturbations?
- How can we reduce the computational effort?

Stability can be ensured by including additional "stabilizing" terminal constraints to the finite horizon problem. Here we consider problems without such stabilizing constraints.

MPC: Questions

Questions:

- When does MPC stabilize the system?
- How good is the MPC Feedback law compared to the infinite horizon optimal solution?
- How robust is the MPC Feedback law with respect to perturbations?
- How can we reduce the computational effort?

Stability can be ensured by including additional "stabilizing" terminal constraints to the finite horizon problem. Here we consider problems without such stabilizing constraints.

Without such constraints, stability is known to hold for "sufficiently large optimization horizon N"
[Alamir/Bornard '95, Jadbabaie/Hauser '05, Grimm et al. '05]

MPC: Questions

Questions:

- When does MPC stabilize the system?
- How good is the MPC Feedback law compared to the infinite horizon optimal solution?
- How robust is the MPC Feedback law with respect to perturbations?
- How can we reduce the computational effort?

Stability can be ensured by including additional "stabilizing" terminal constraints to the finite horizon problem. Here we consider problems without such stabilizing constraints.

Without such constraints, stability is known to hold for "sufficiently large optimization horizon N"
[Alamir/Bornard '95, Jadbabaie/Hauser '05, Grimm et al. '05]

How large is "sufficiently large"?

Estimating N

For obtaining a quantitative estimate we need quantitative information.

Estimating N

For obtaining a quantitative estimate we need quantitative information.

A suitable condition is "exponential controllability through ℓ":

> there exist real numbers $C > 0$, $\sigma \in (0,1)$ such that for each $x(0) \in X$ there is $u(\cdot)$ with
>
> $$\ell(x(n), u(n)) \leq C\sigma^n \ell^*(x(0))$$
>
> with $\ell^*(x) = \min_u \ell(x,u)$

Stability conditions

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

Stability conditions

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

Define $\alpha := 1 - \dfrac{(\gamma_N - 1)\prod_{i=2}^{N}(\gamma_i - 1)}{\prod_{i=2}^{N}\gamma_i - \prod_{i=2}^{N}(\gamma_i - 1)}$ with $\gamma_i = \sum_{k=0}^{i-1} C\sigma^k$

Stability conditions

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

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

Theorem: If $\alpha > 0$, then the MPC feedback F_N stabilizes all C, σ-exponentially controllable systems and we get $J_\infty(x, F_N) \leq \inf_{u \in U^\infty} J_\infty(x, u)/\alpha$

Stability conditions

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

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

Theorem: If $\alpha > 0$, then the MPC feedback F_N stabilizes all C, σ-exponentially controllable systems and we get $J_\infty(x, F_N) \leq \inf_{u \in U^\infty} J_\infty(x, u)/\alpha$

If $\alpha < 0$ then there exists a C, σ-exponentially controllable system, which is not stabilized by F_N

Stability conditions

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

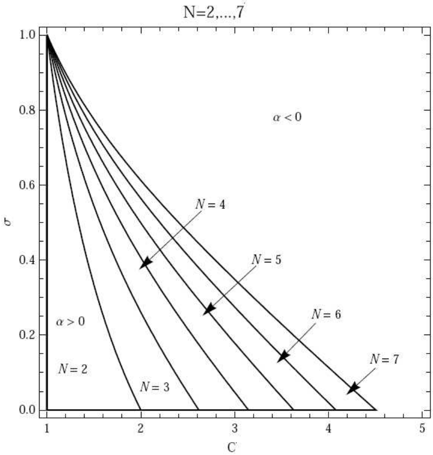
Define $\alpha := 1 - \dfrac{(\gamma_N - 1)\prod_{i=2}^{N}(\gamma_i - 1)}{\prod_{i=2}^{N}\gamma_i - \prod_{i=2}^{N}(\gamma_i - 1)}$ with $\gamma_i = \sum_{k=0}^{i-1} C\sigma^k$

Theorem: If $\alpha > 0$, then the MPC feedback F_N stabilizes all C, σ-exponentially controllable systems and we get
$J_\infty(x, F_N) \le \inf_{u \in U^\infty} J_\infty(x, u)/\alpha$

If $\alpha < 0$ then there exists a C, σ-exponentially controllable system, which is not stabilized by F_N

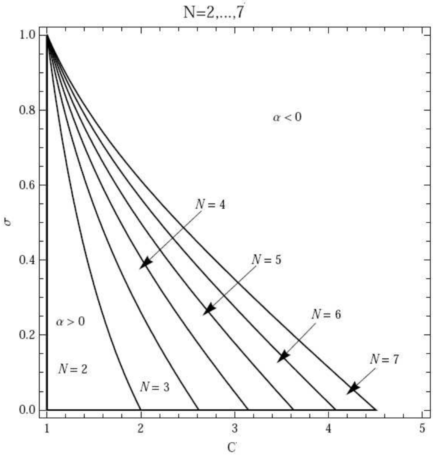
Moreover, $\alpha \to 1$ as $N \to \infty$

Stability chart for C and σ



(Figure: Harald Voit)

Stability chart for C and σ



(Figure: Harald Voit)

Conclusion: try to reduce C, e.g., by choosing ℓ appropriately

A PDE example

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 $y = y(t, x)$

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

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

and distributed control $u : \mathbb{R} \times \Omega \to \mathbb{R}$

A PDE example

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 $y = y(t, x)$

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

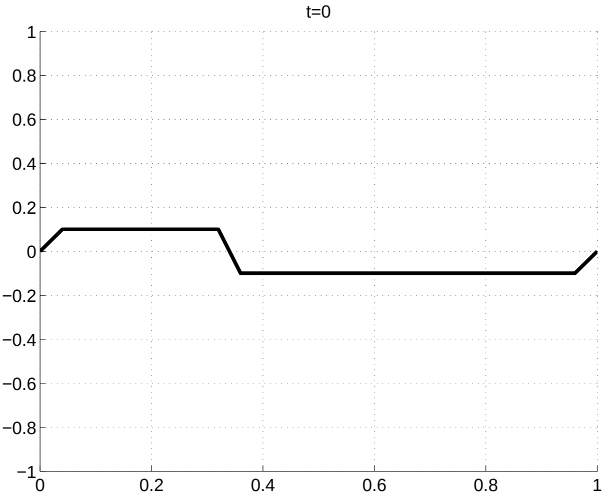
parameters $\nu = 0.1$ and $\mu = 10$

and distributed control $u : \mathbb{R} \times \Omega \to \mathbb{R}$

Discrete time system: $y(n) = y(nT, \cdot)$ for some $T > 0$

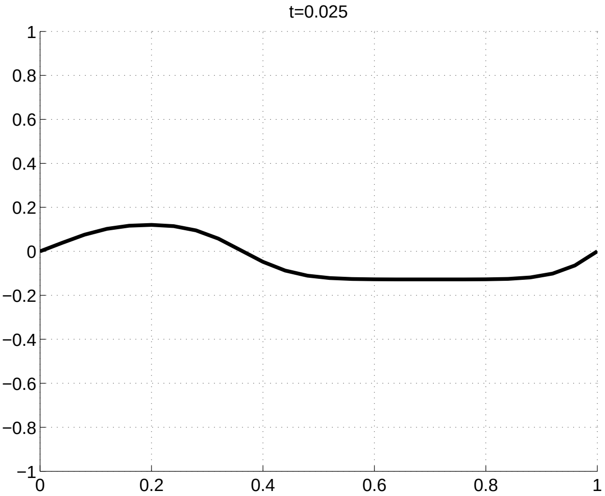
("sampled data system with sampling time T")

The uncontrolled PDE



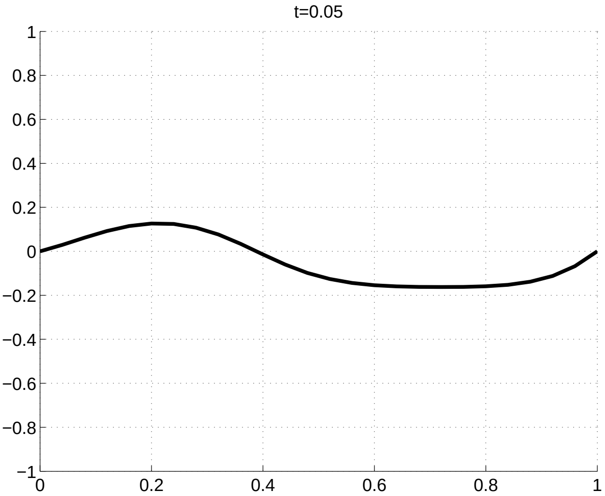
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



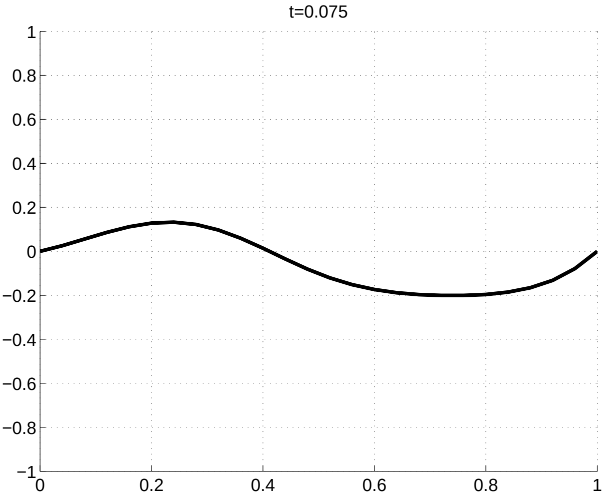
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



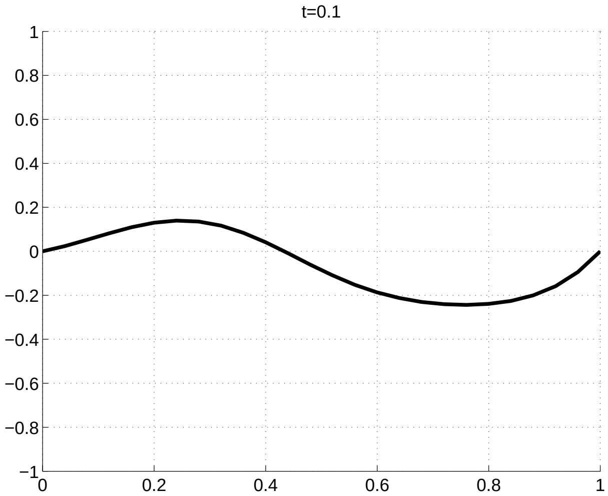
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



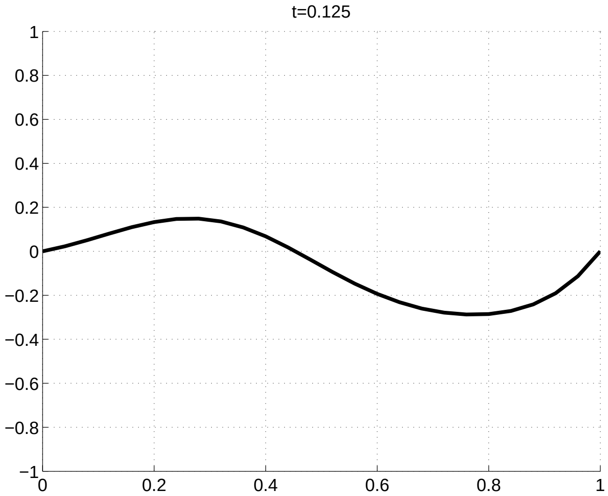
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



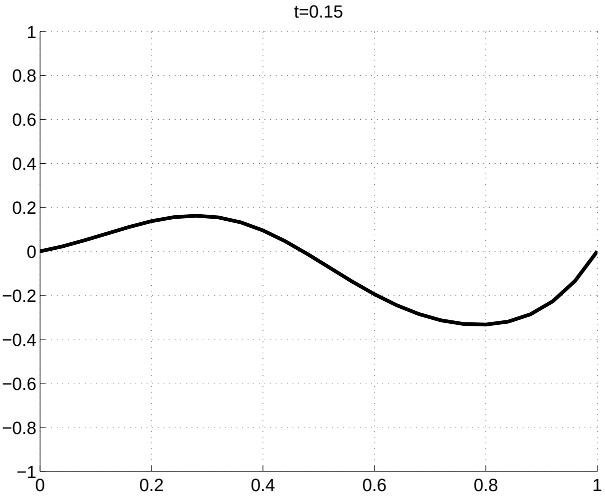
uncontrolled $(u \equiv 0)$

The uncontrolled PDE



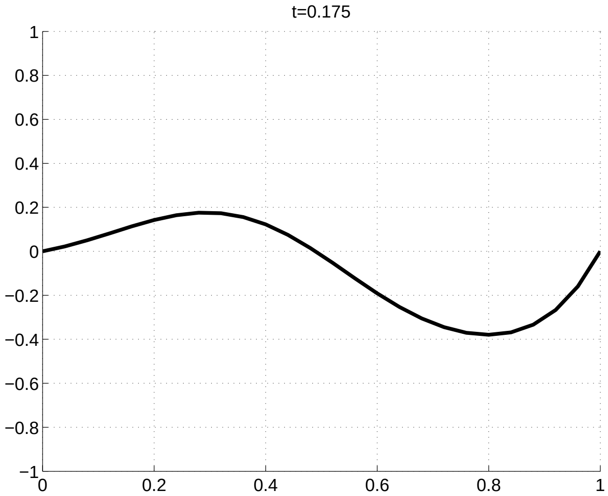
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



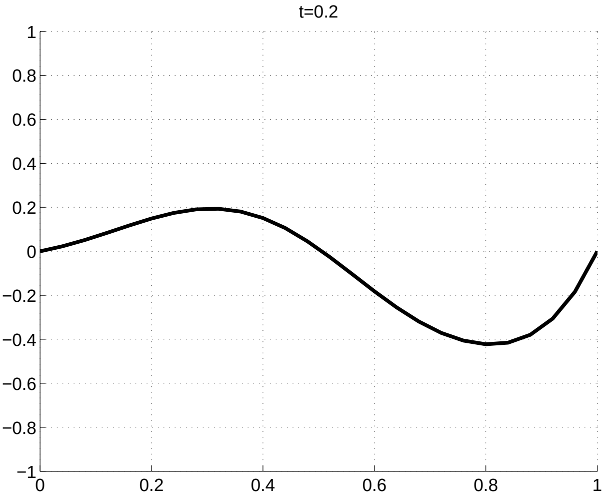
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



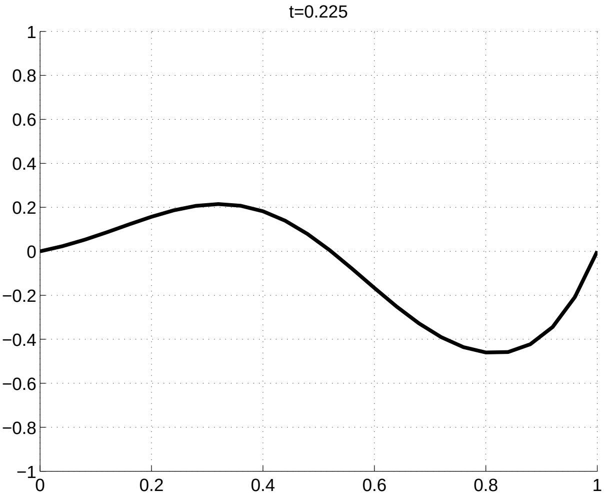
uncontrolled $(u \equiv 0)$

The uncontrolled PDE



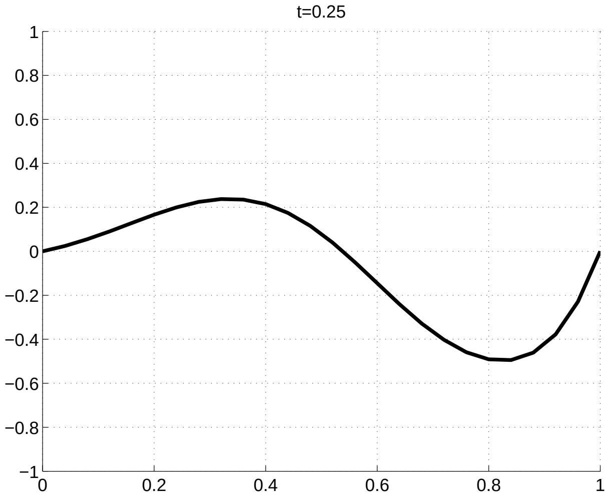
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



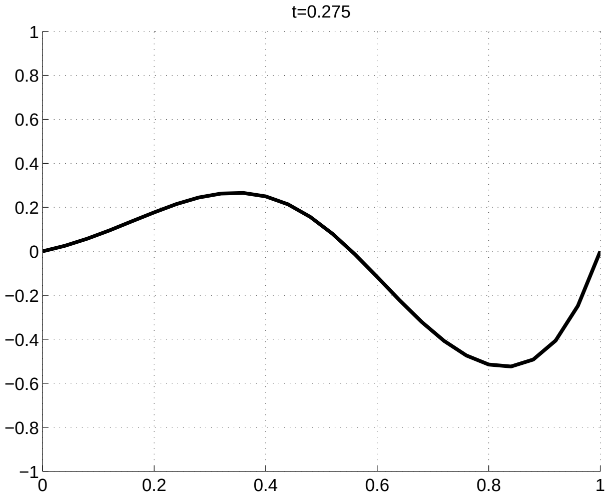
uncontrolled $(u \equiv 0)$

The uncontrolled PDE



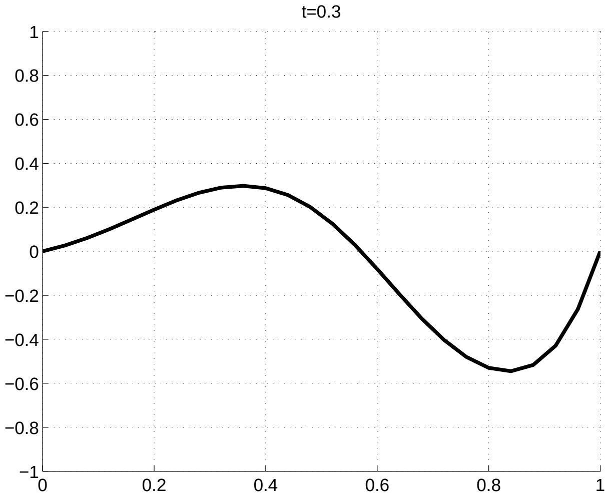
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



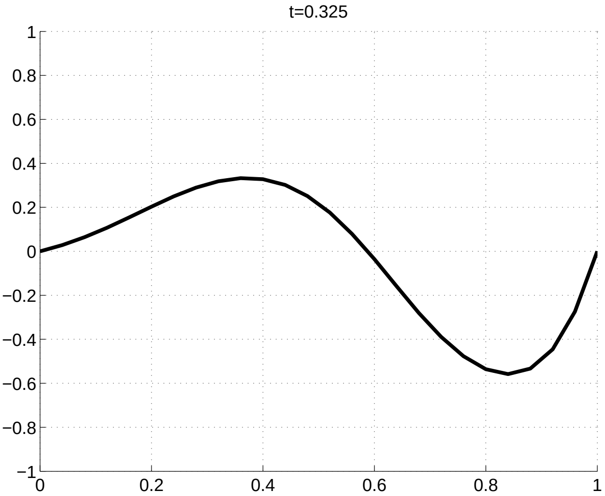
uncontrolled $(u \equiv 0)$

The uncontrolled PDE



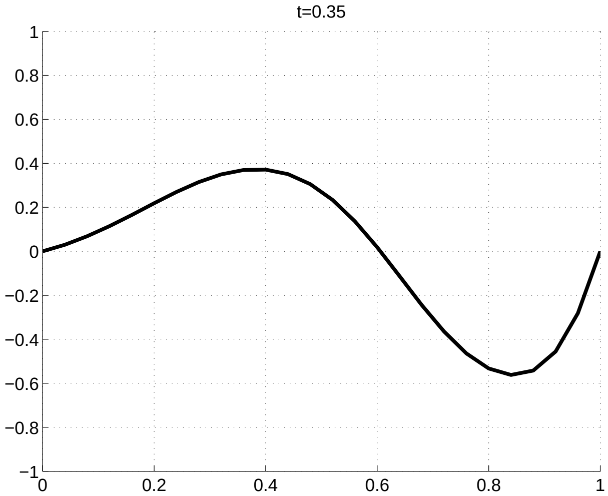
uncontrolled $(u \equiv 0)$

The uncontrolled PDE



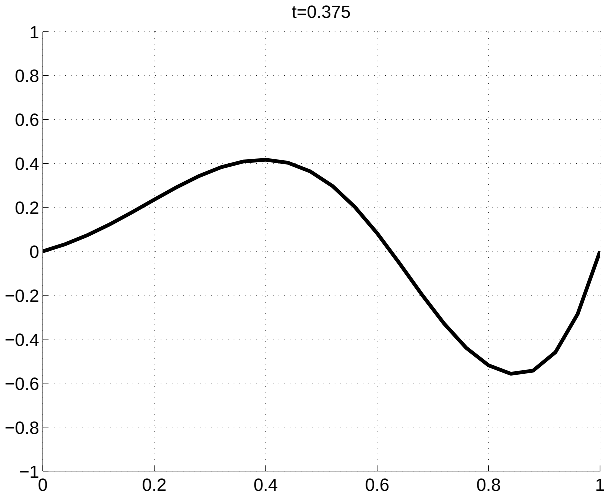
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



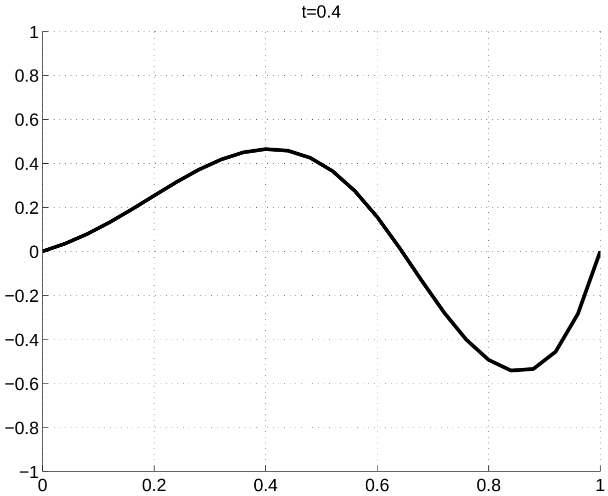
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



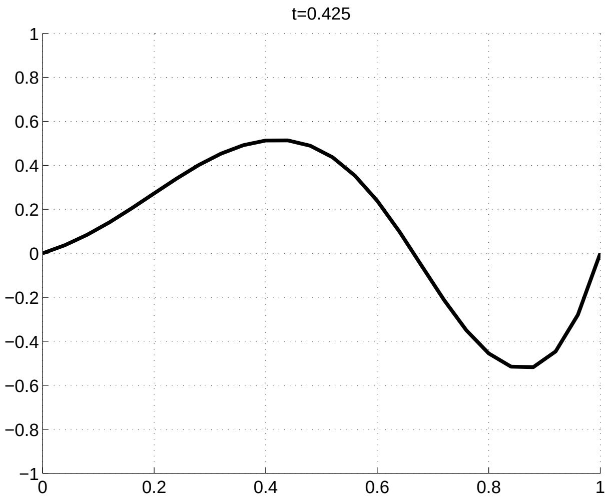
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



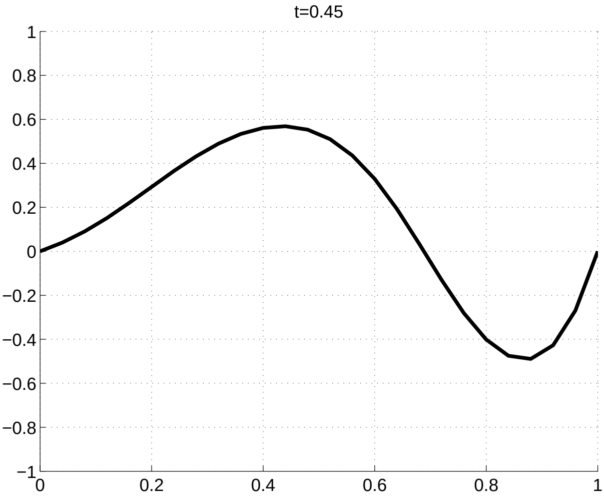
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



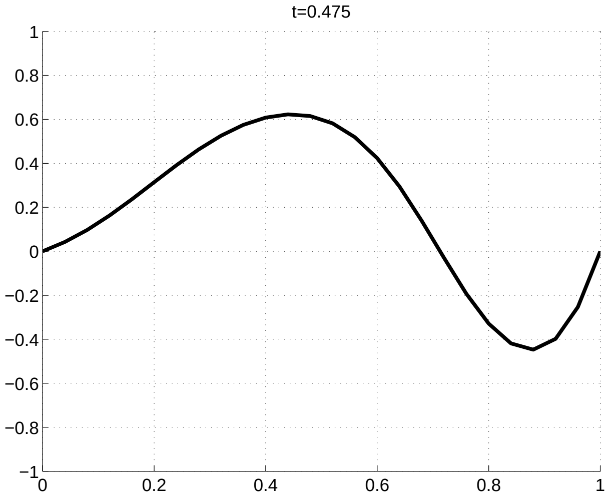
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



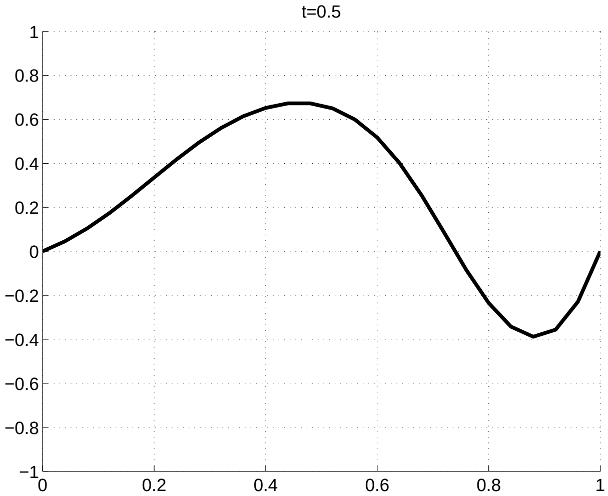
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



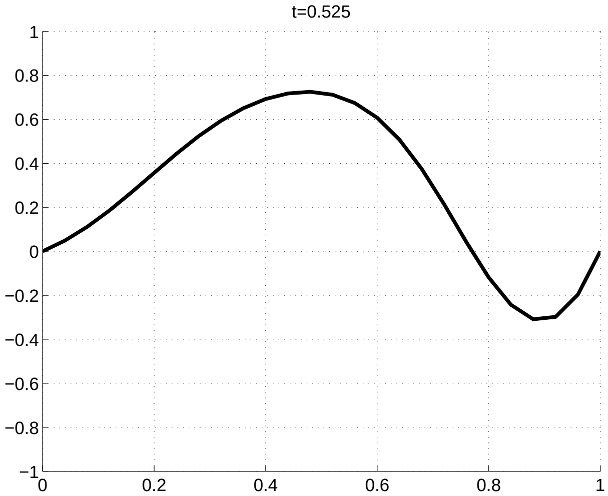
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



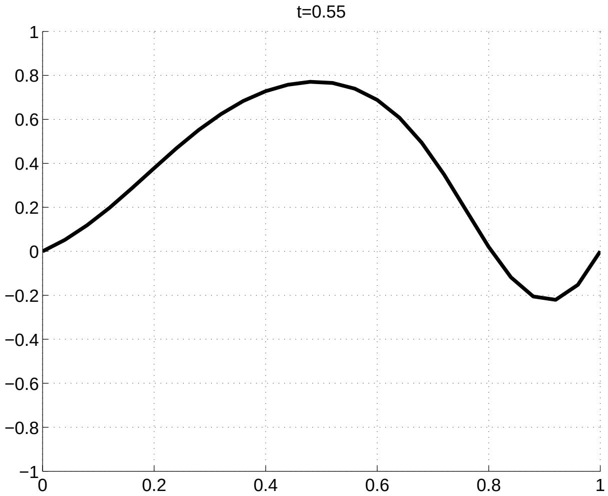
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



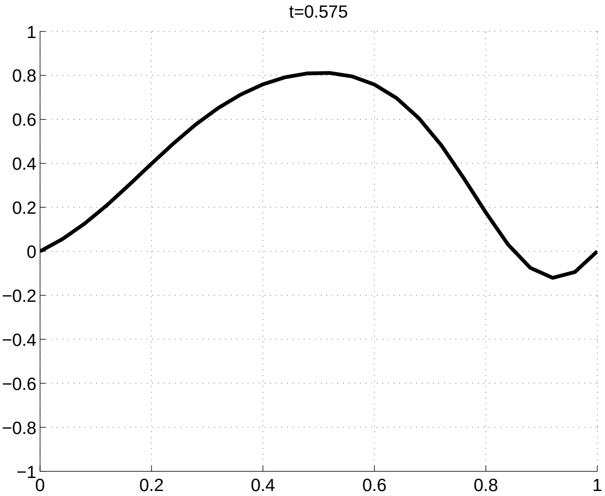
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



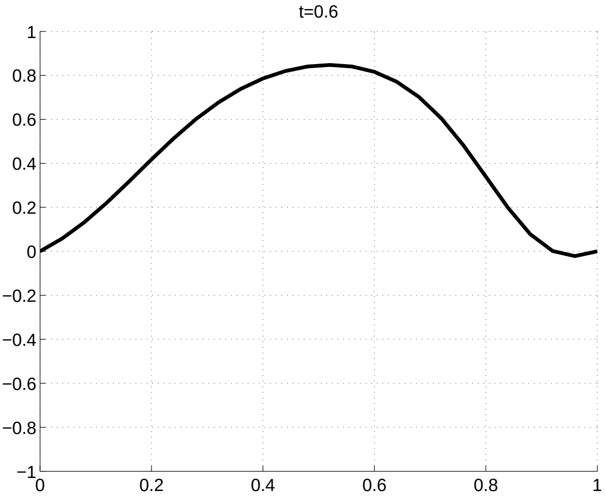
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



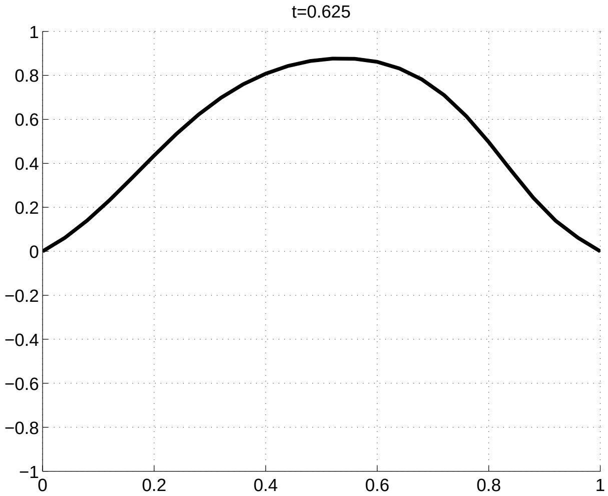
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



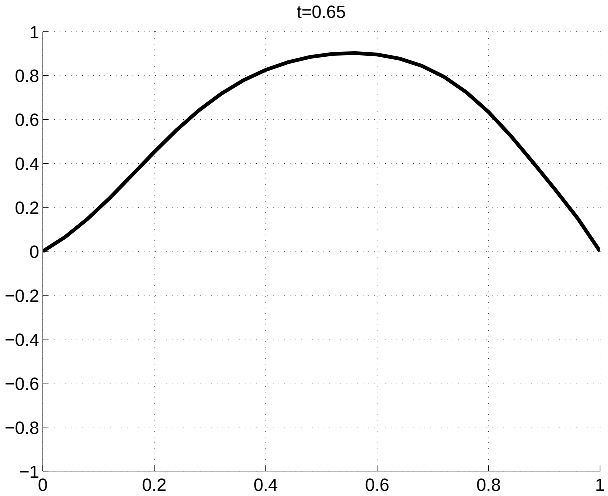
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



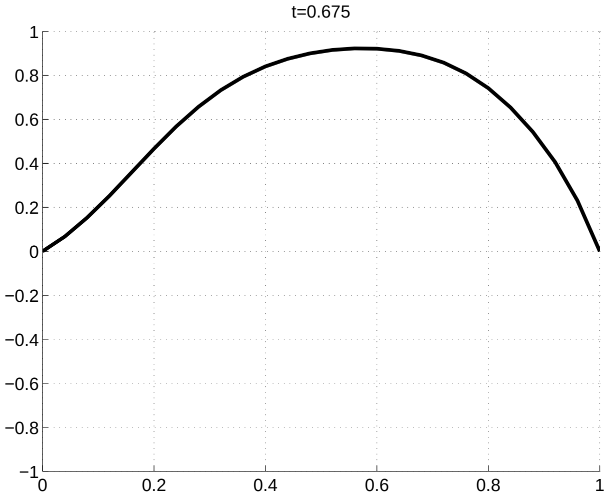
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



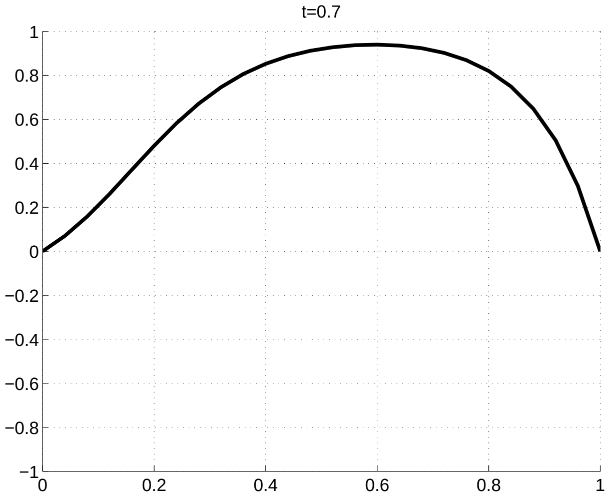
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



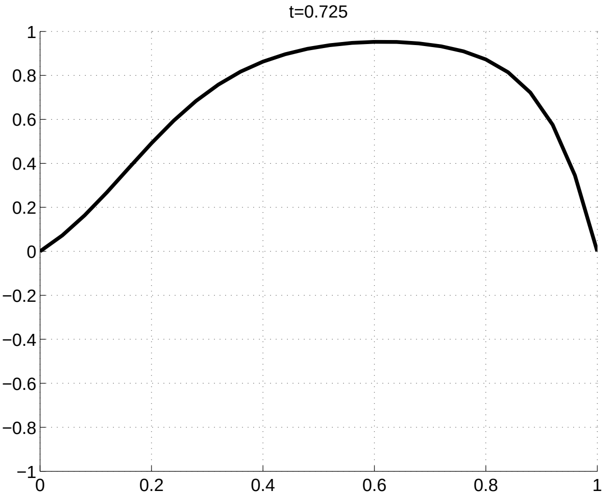
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



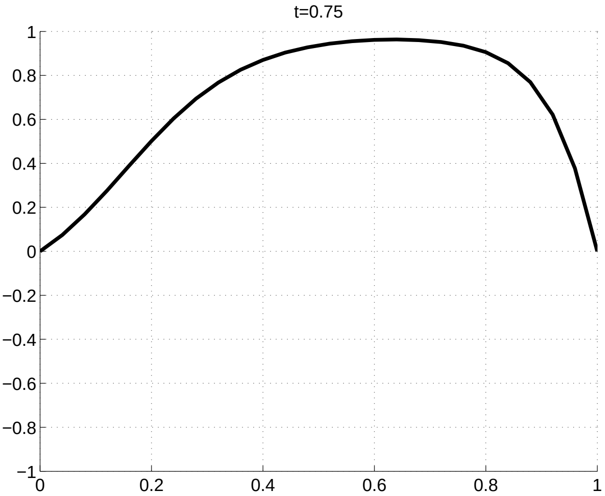
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



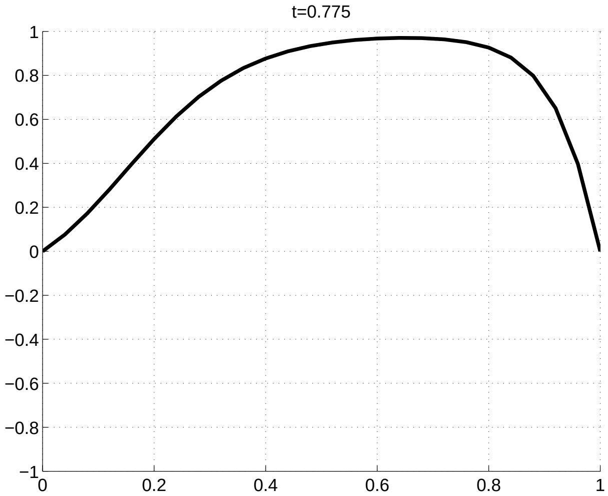
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



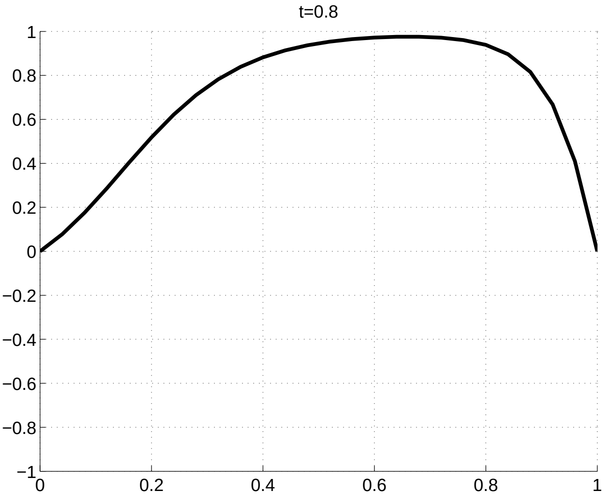
uncontrolled $(u \equiv 0)$

The uncontrolled PDE



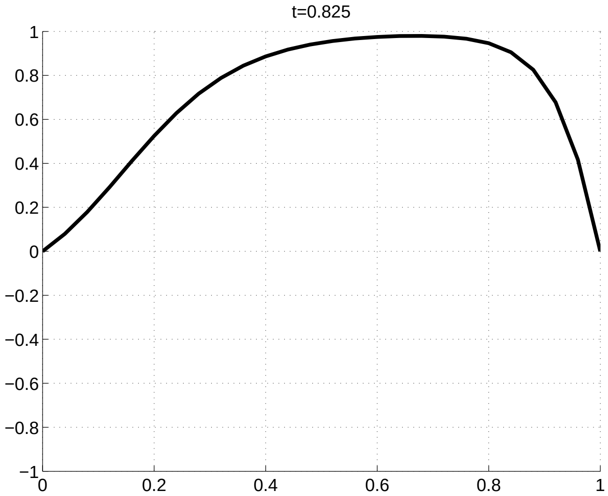
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



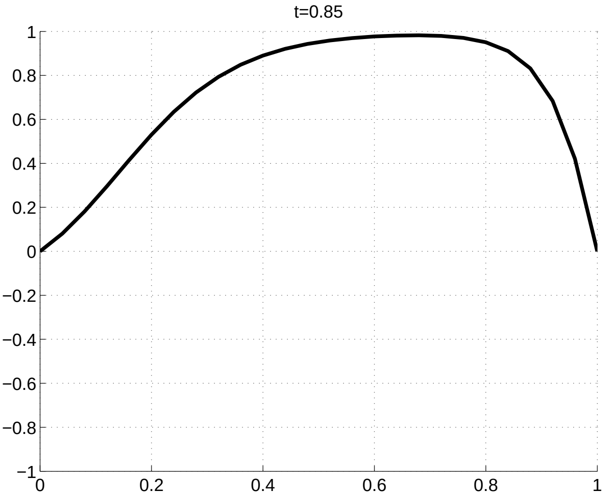
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



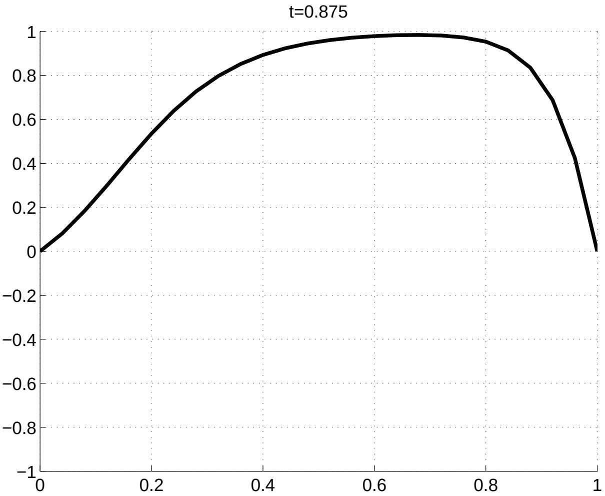
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



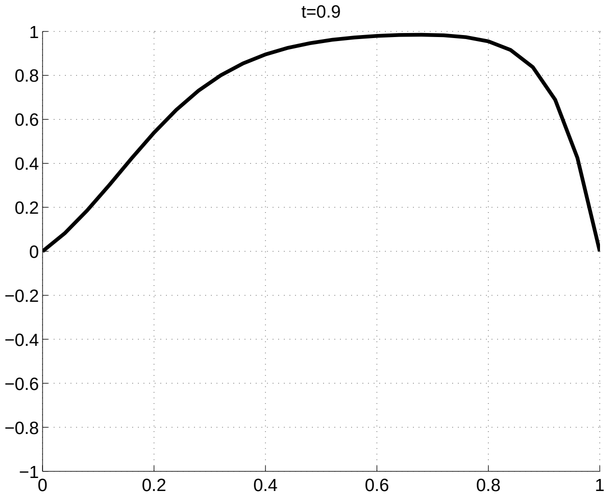
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



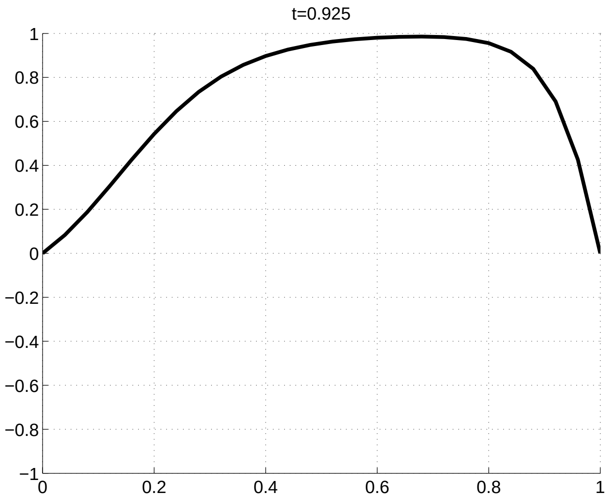
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



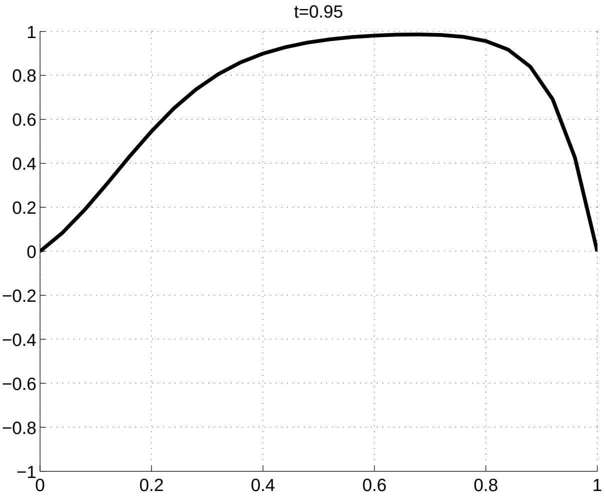
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



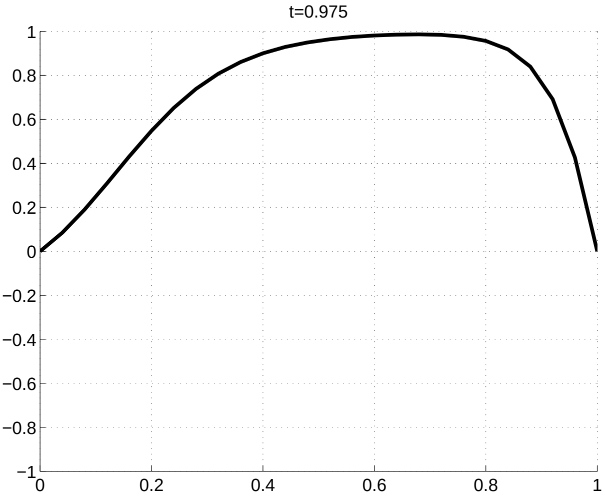
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



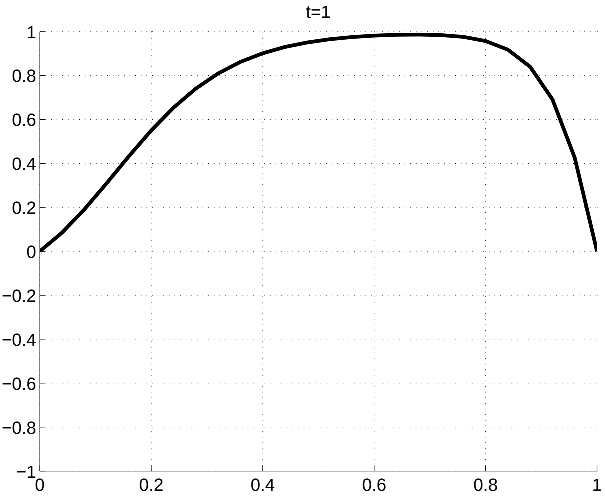
uncontrolled ($u \equiv 0$)

The uncontrolled PDE



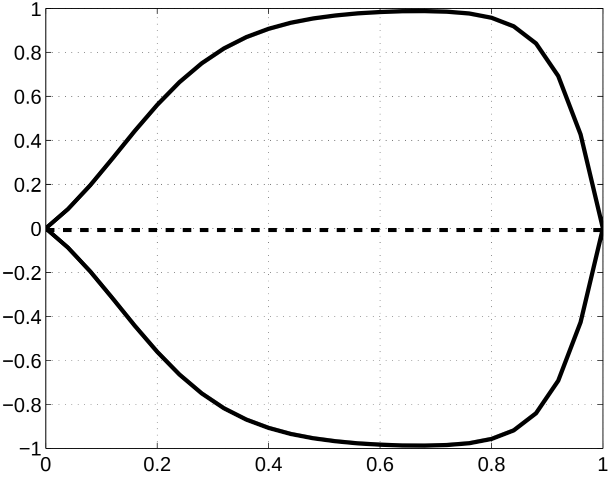
uncontrolled ($u \equiv 0$)

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 system $y(n)$ 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 system $y(n)$ at $y \equiv 0$

For $y \approx 0$ the control u must compensate for $y_x \rightsquigarrow u \approx -y_x$

MPC for the PDE example

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

Goal: stabilize the sampled data system $y(n)$ at $y \equiv 0$

For $y \approx 0$ the control u must compensate for $y_x \rightsquigarrow u \approx -y_x$

This observation and a little computation reveals:

For the (usual) quadratic L^2 cost

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

the constant C is much larger than for the quadratic H^1 cost

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

MPC for the PDE example

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

Goal: stabilize the sampled data system $y(n)$ at $y \equiv 0$

For $y \approx 0$ the control u must compensate for $y_x \rightsquigarrow u \approx -y_x$

This observation and a little computation reveals:

For the (usual) quadratic L^2 cost

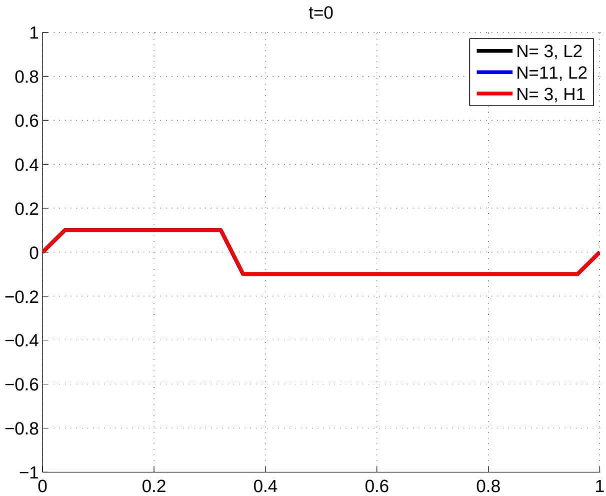
$$\ell(y(n), u(n)) = \|y(n)\|_{L^2}^2 + \lambda \|u(n)\|_{L^2}^2$$

the constant C is much larger than for the quadratic H^1 cost

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

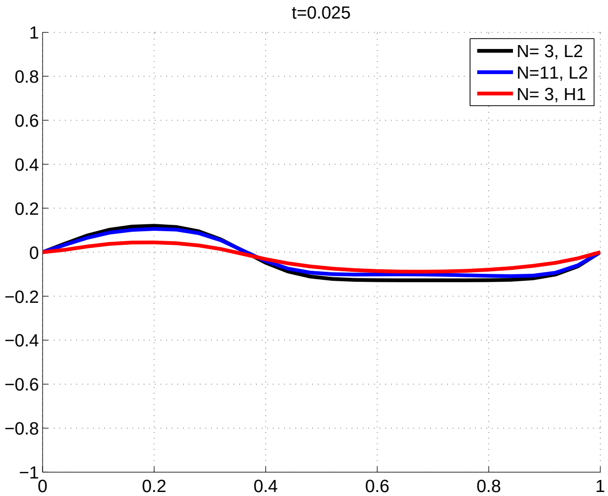
$\rightsquigarrow$ H^1 should perform better that L^2

MPC with L_2 vs. H_1 cost



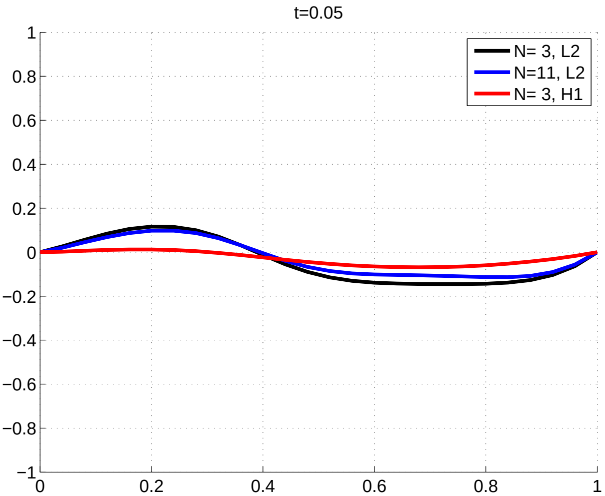
MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

MPC with L_2 vs. H_1 cost



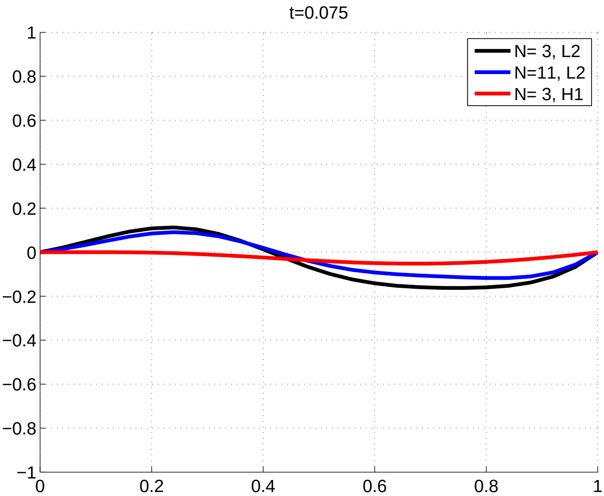
MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

MPC with L_2 vs. H_1 cost



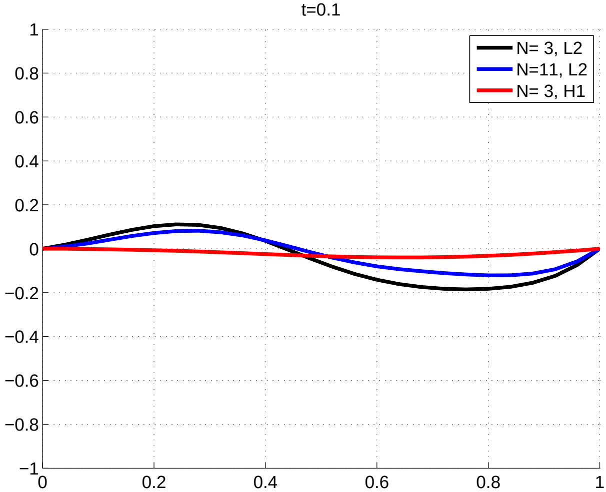
MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

MPC with L_2 vs. H_1 cost



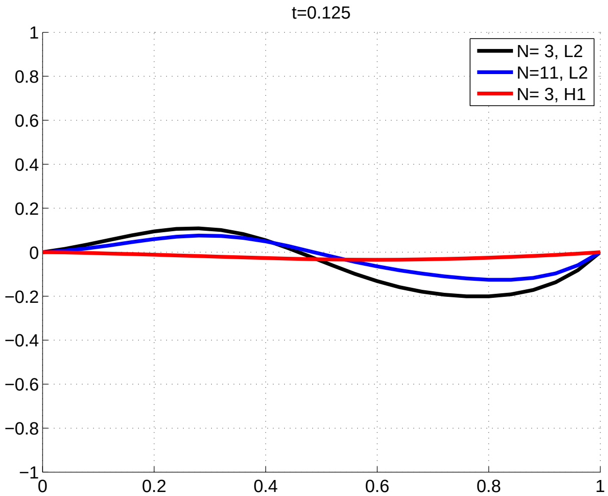
MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

MPC with L_2 vs. H_1 cost



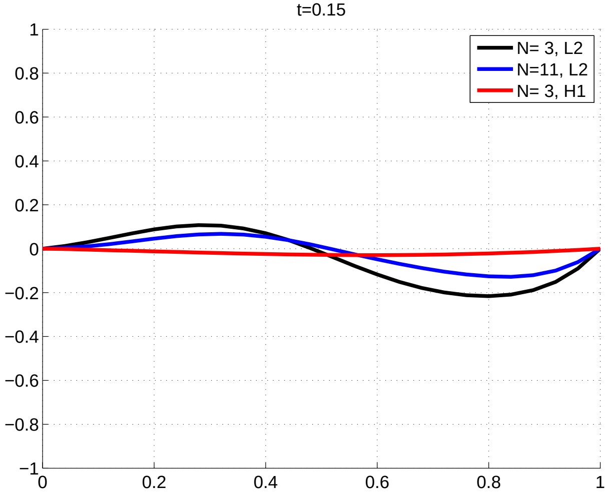
MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

MPC with L_2 vs. H_1 cost



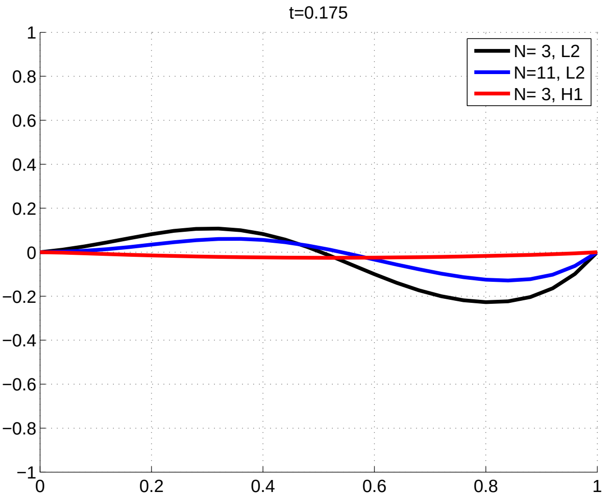
MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

MPC with L_2 vs. H_1 cost



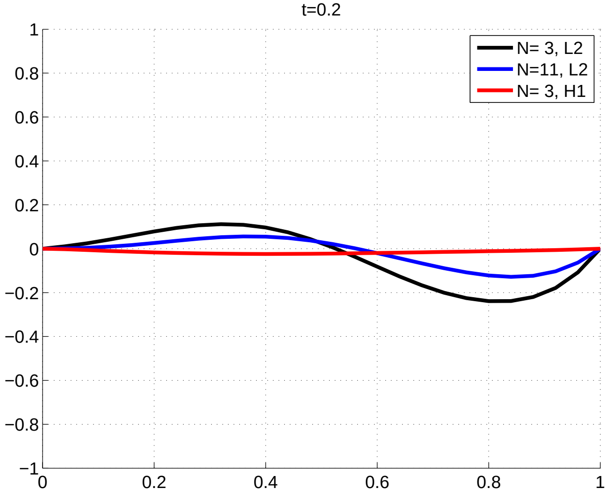
MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

MPC with L_2 vs. H_1 cost



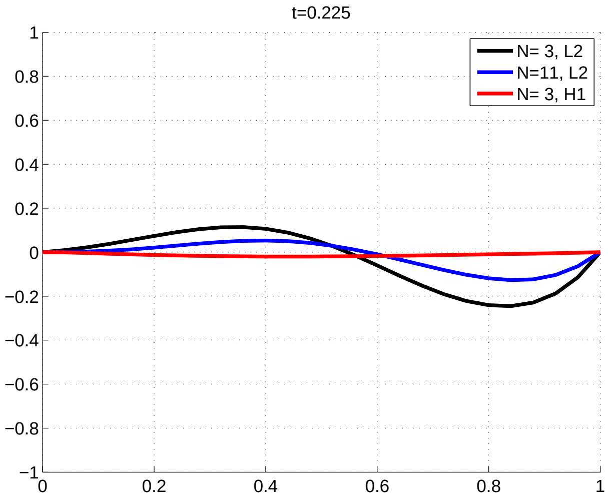
MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

MPC with L_2 vs. H_1 cost



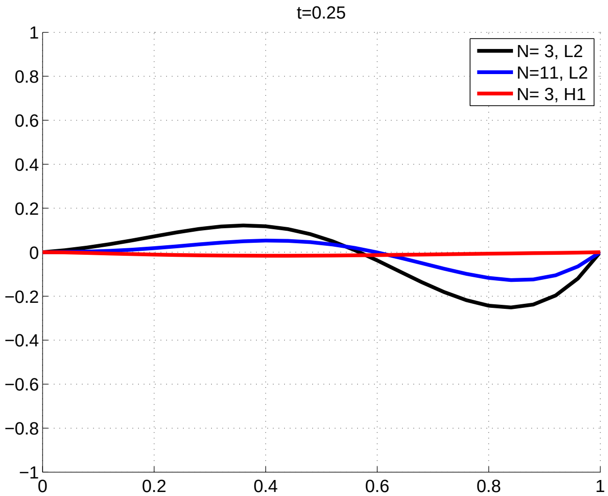
MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

MPC with L_2 vs. H_1 cost



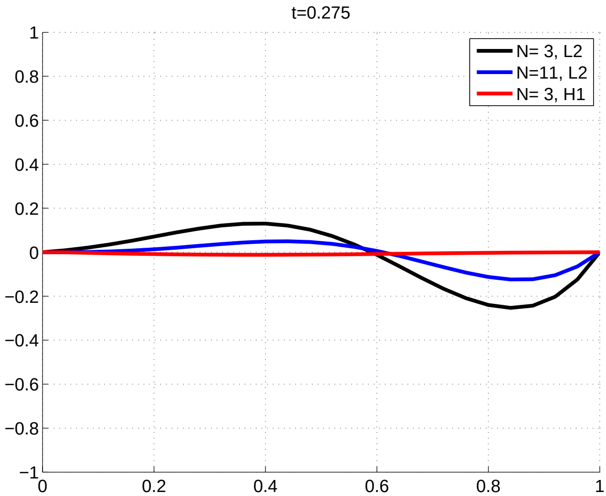
MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

MPC with L_2 vs. H_1 cost



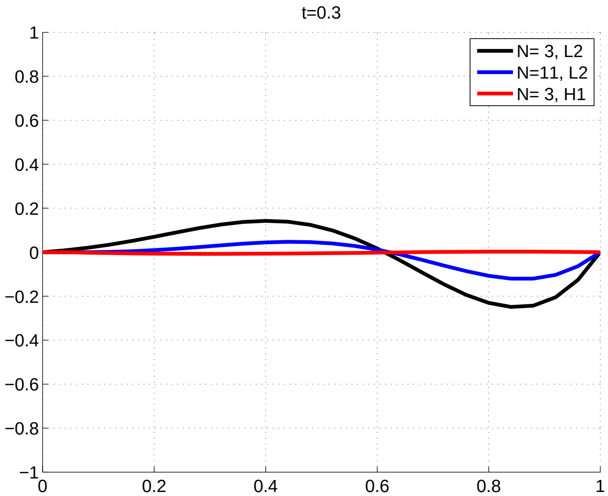
MPC with L_2 and H_1 cost, $\lambda = 0.1$, sampling time $T = 0.025$

MPC with L_2 vs. H_1 cost



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

MPC with L_2 vs. H_1 cost



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

Boundary Control

Now we change our PDE from distributed to (Dirichlet-) boundary control, i.e.

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

with

domain $\Omega = [0,1]$

solution $y = y(t,x)$

boundary conditions $y(t,0) = u_0(t)$, $y(t,1) = u_1(t)$

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

Boundary Control

Now we change our PDE from distributed to (Dirichlet-) boundary control, i.e.

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

with

domain $\Omega = [0,1]$

solution $y = y(t,x)$

boundary conditions $y(t,0) = u_0(t)$, $y(t,1) = u_1(t)$

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

with boundary control, stability can only be achieved via large gradients in the transient phase

Boundary Control

Now we change our PDE from distributed to (Dirichlet-) boundary control, i.e.

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

with

domain $\Omega = [0,1]$

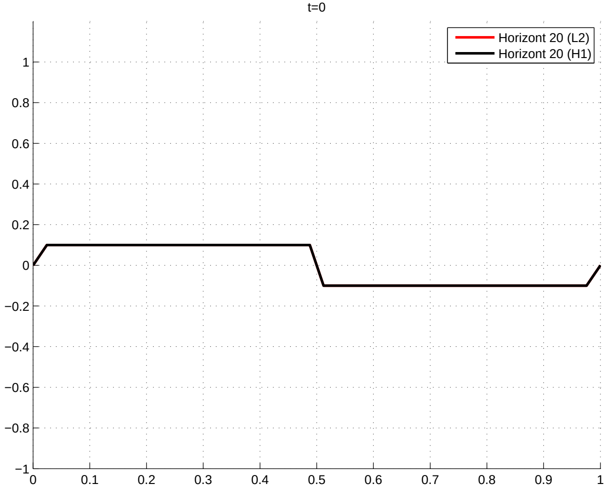
solution $y = y(t,x)$

boundary conditions $y(t,0) = u_0(t)$, $y(t,1) = u_1(t)$

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

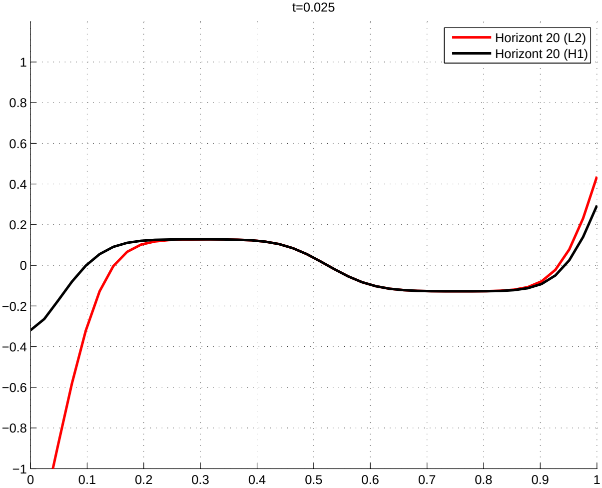
with boundary control, stability can only be achieved via large gradients in the transient phase
$\rightsquigarrow$ L^2 should perform better that H^1

Boundary control, L_2 vs. H_1, $N = 20$



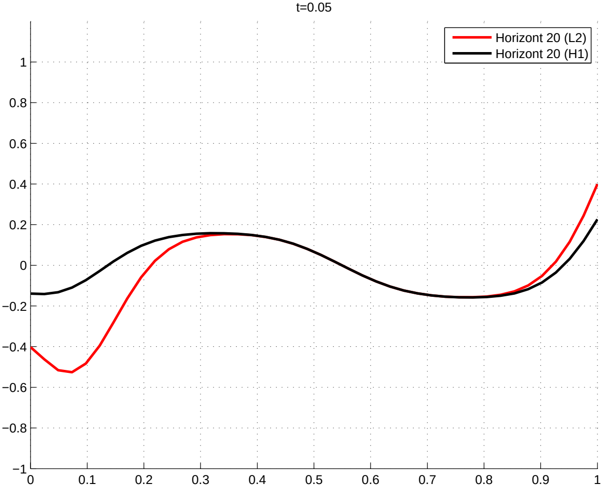
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



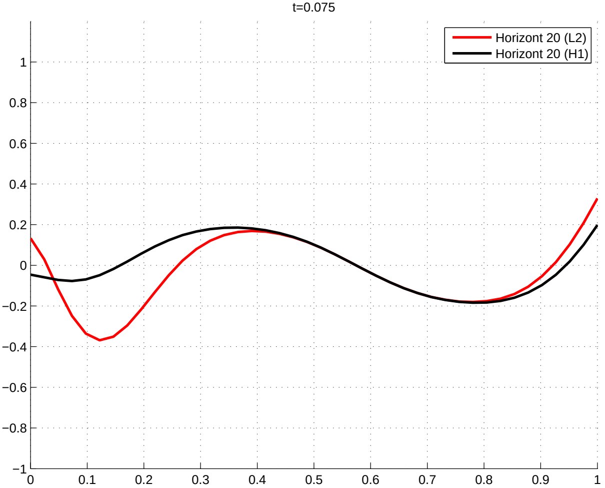
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



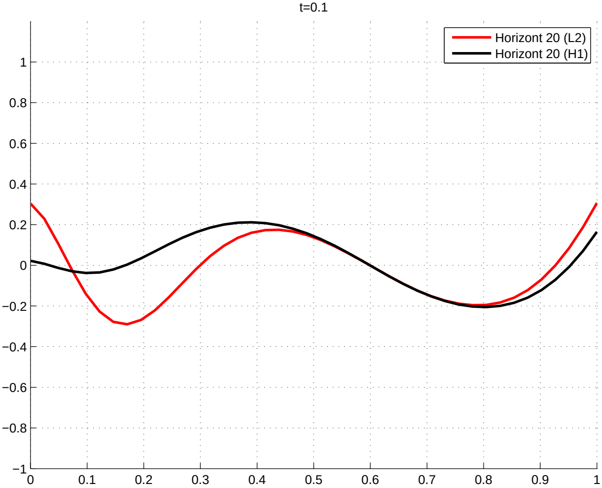
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



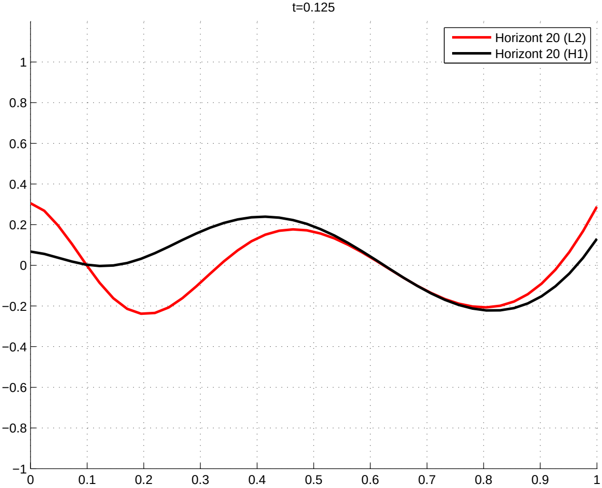
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



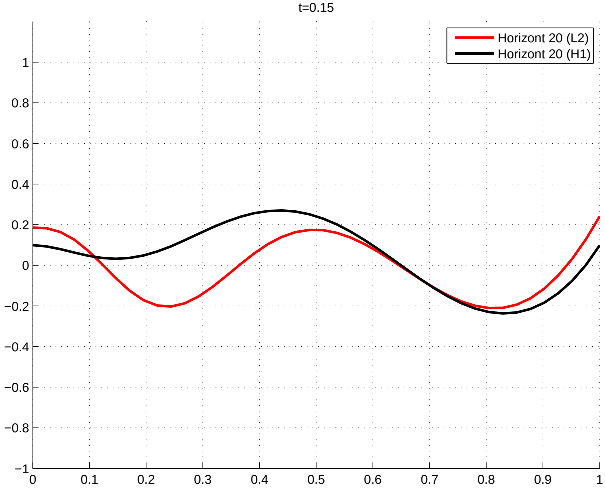
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



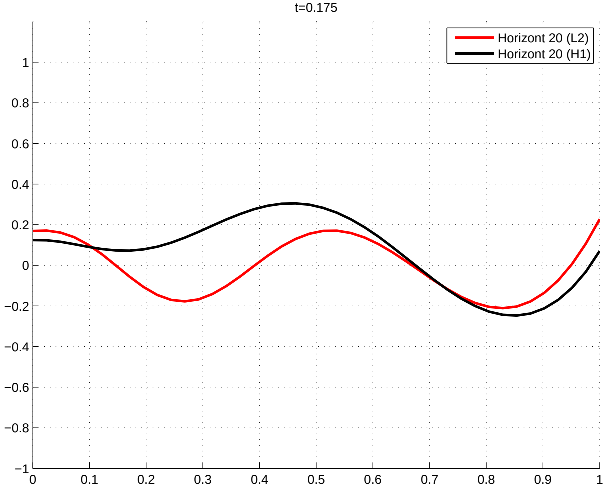
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



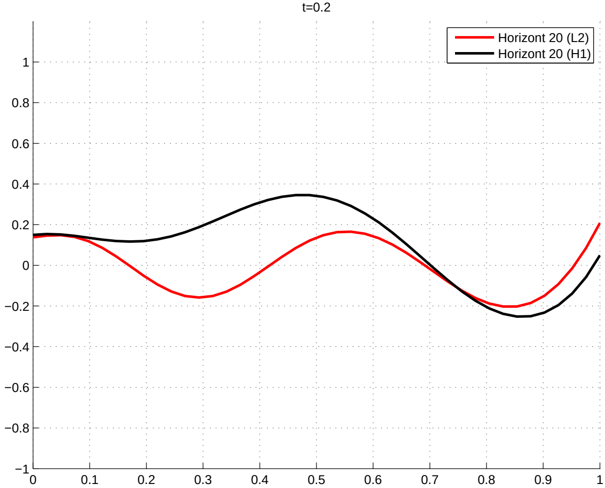
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



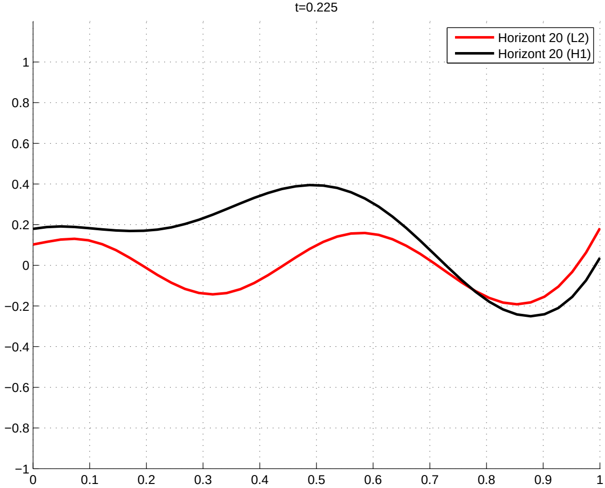
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



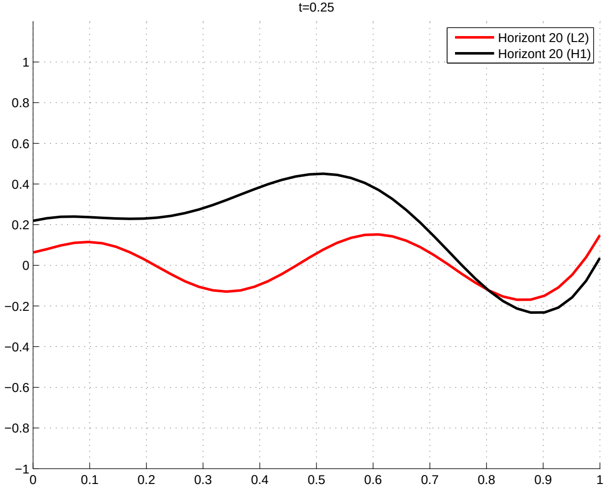
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



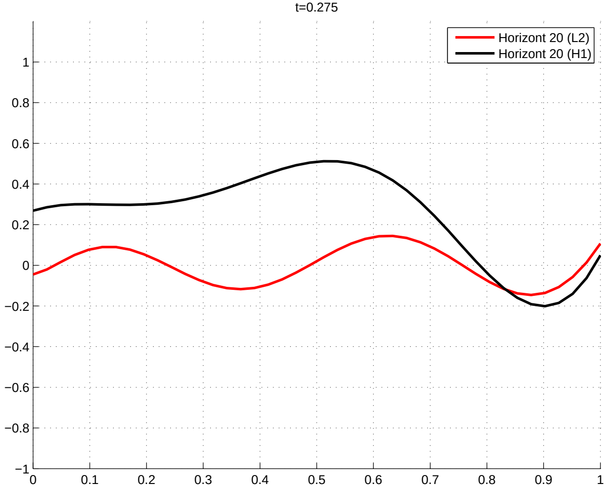
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



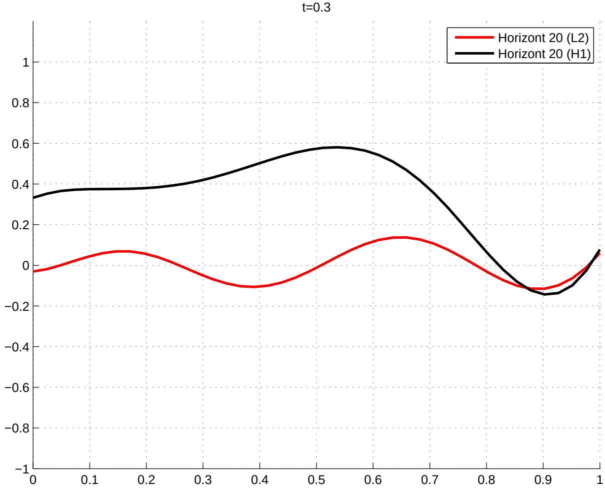
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



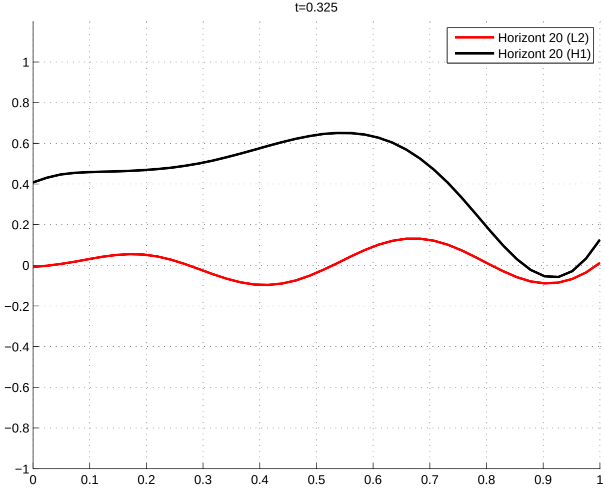
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



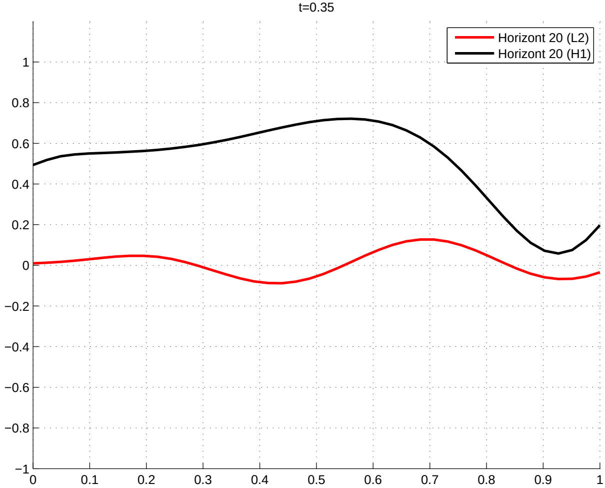
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



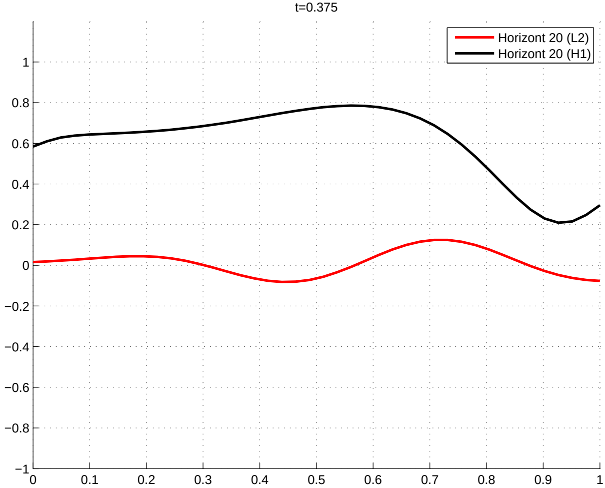
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



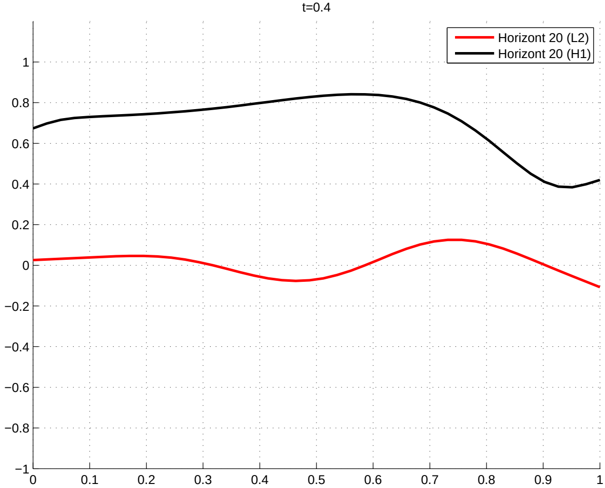
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



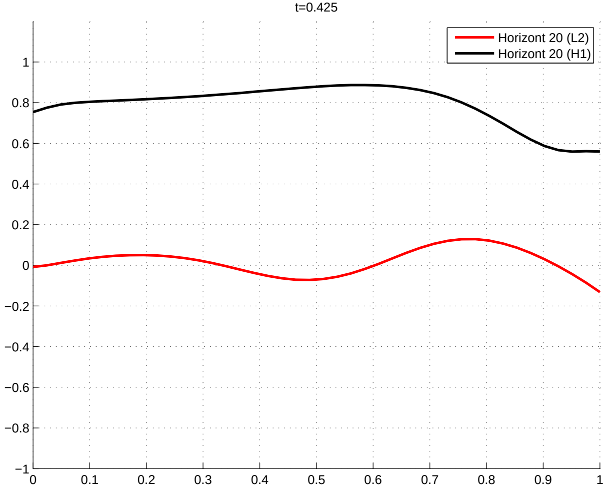
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



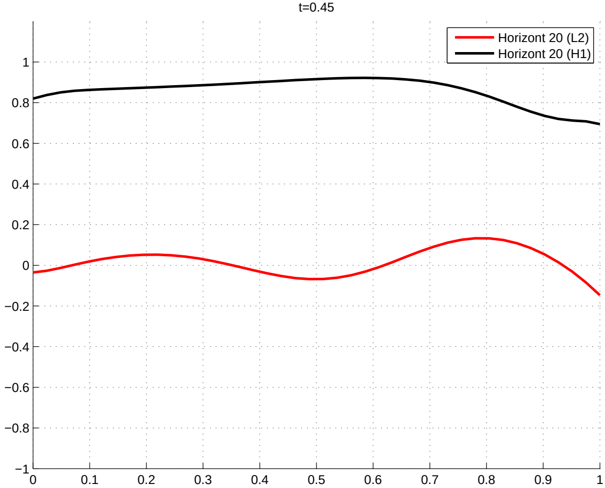
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



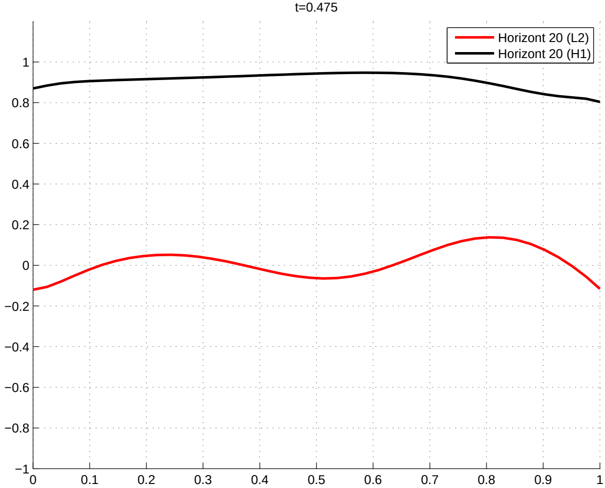
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



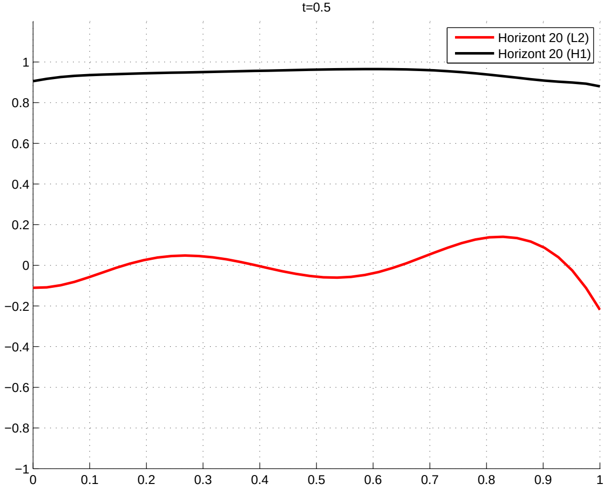
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



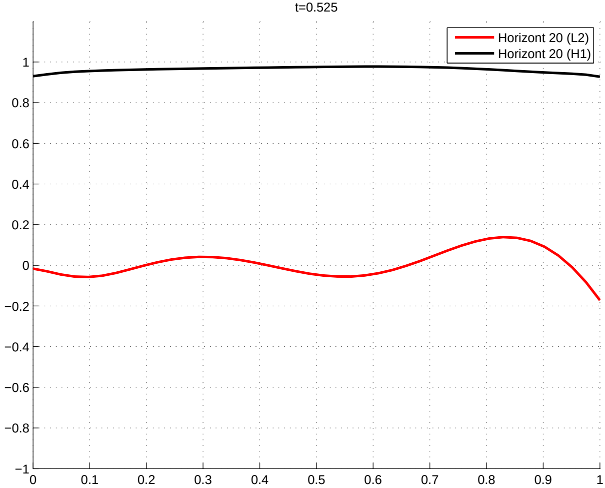
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



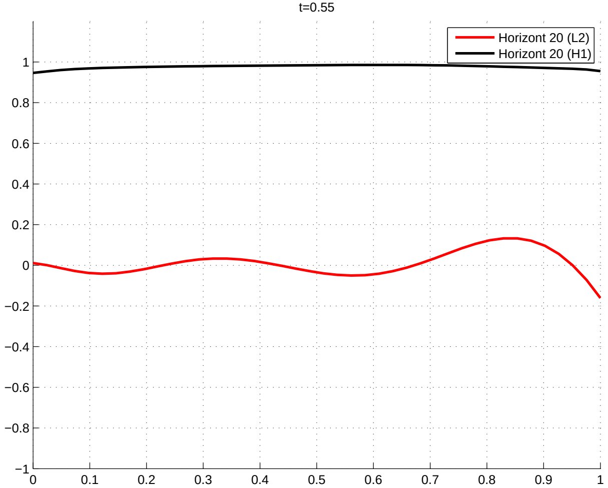
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



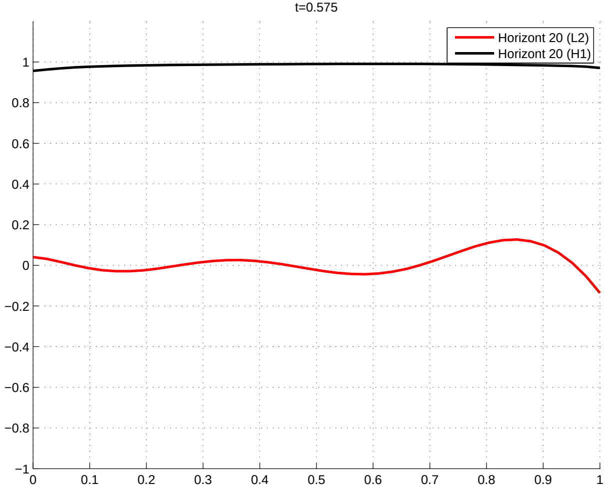
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



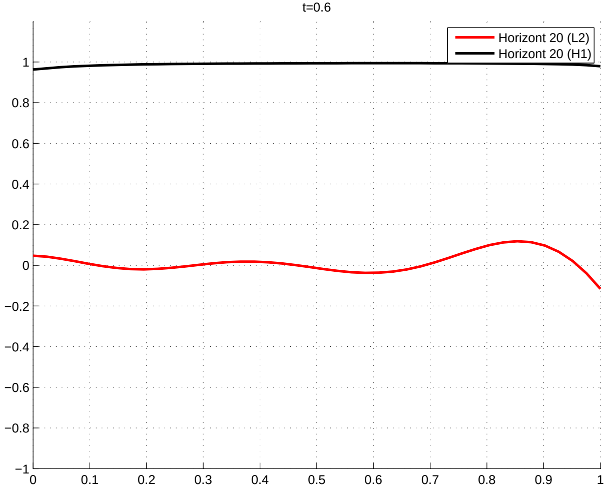
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



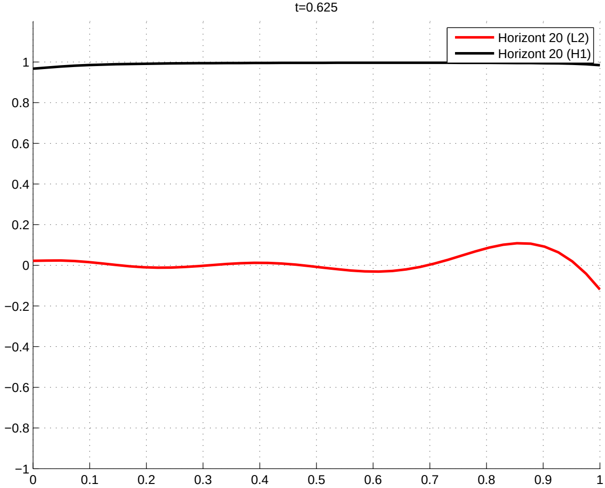
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



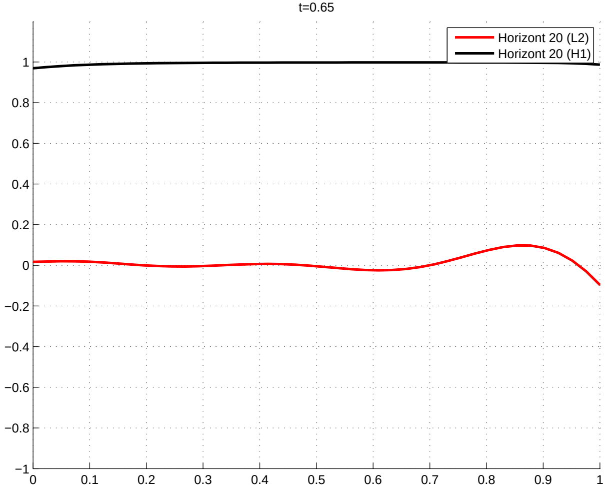
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



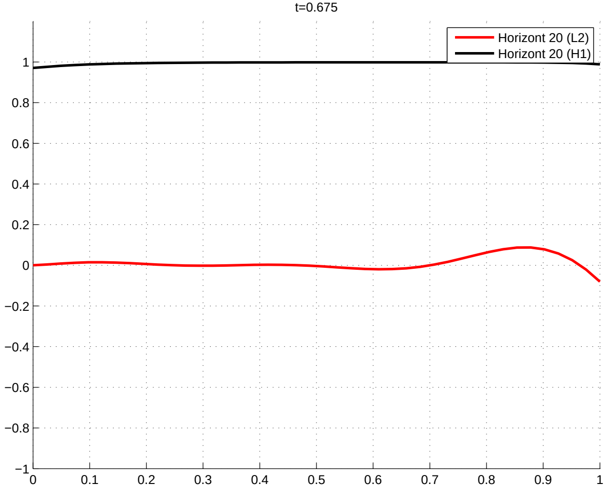
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



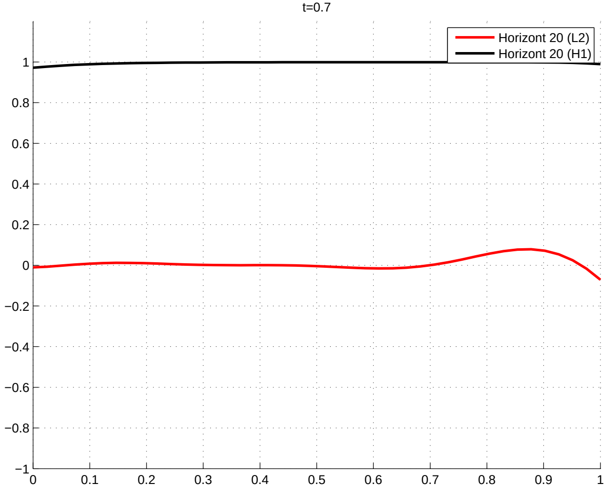
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



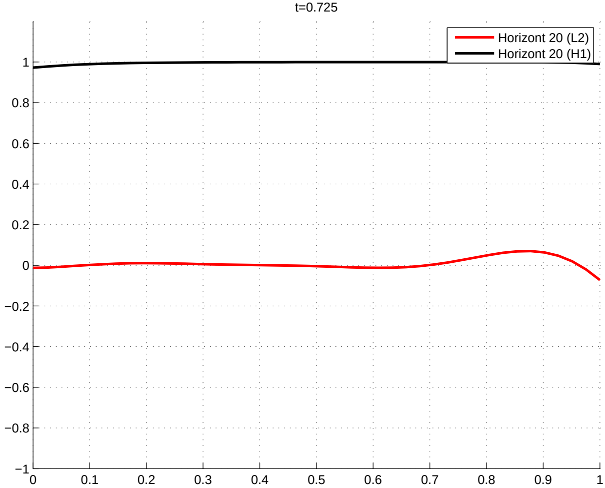
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



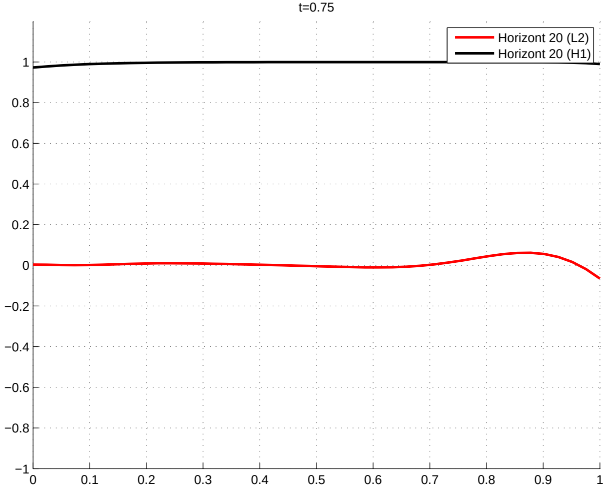
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



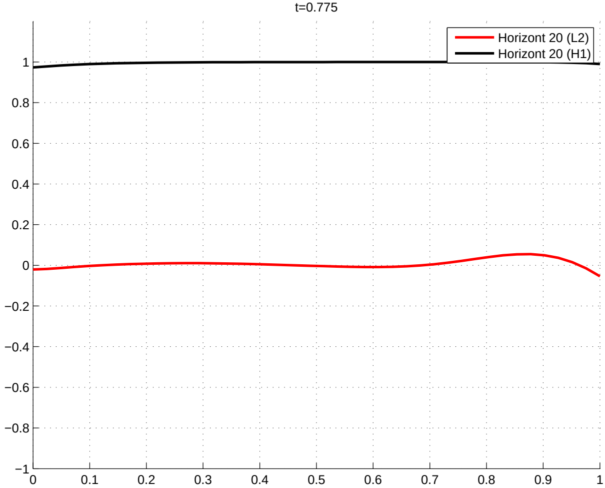
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



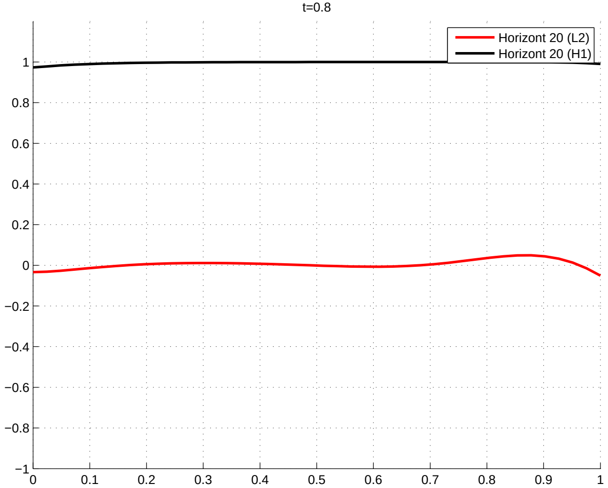
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



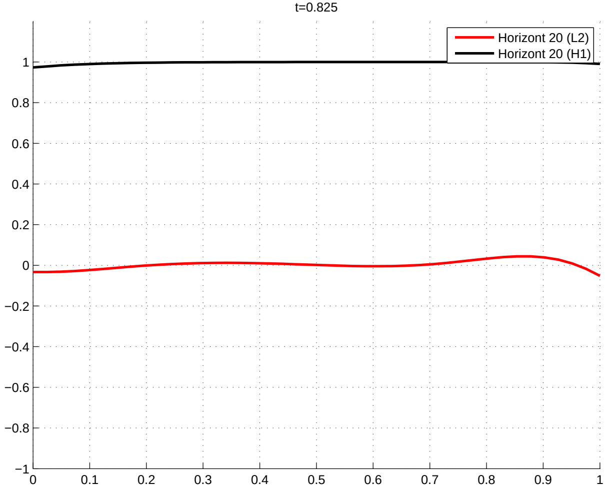
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



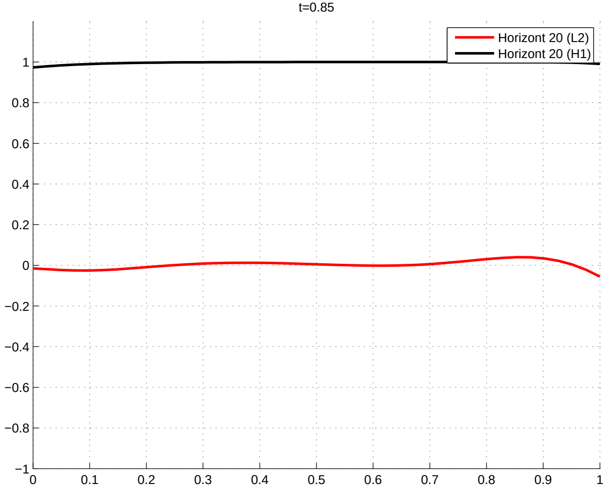
Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Boundary control, L_2 vs. H_1, $N = 20$



Boundary control, $\lambda = 0.001$, sampling time $T = 0.025$

Robustness

Usually, the model used for optimization

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

does not exactly match the real system

Robustness

Usually, the model used for optimization

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

does not exactly match the real system

This mismatch can, e.g., be modelled by an additive perturbation

$$x_{real}(n+1) = f(x_{real}(n), u(n)) + d(n)$$

Robustness

Usually, the model used for optimization

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

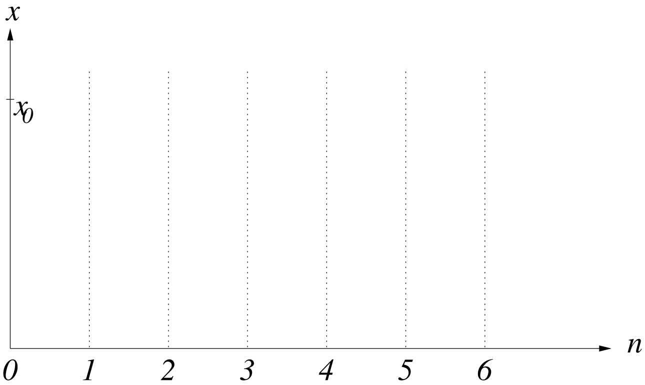
does not exactly match the real system

This mismatch can, e.g., be modelled by an additive perturbation

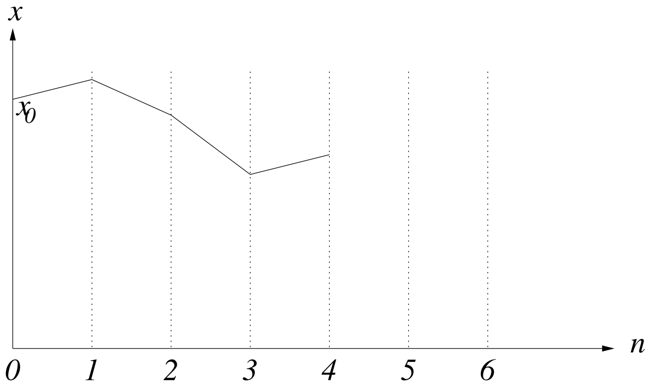
$$x_{real}(n+1) = f(x_{real}(n), u(n)) + d(n)$$

Robustness $:\Leftrightarrow$ the system still approaches/stays within a neighborhood of the stable equilibrium for small $d(n)$

Perturbations in MPC scheme

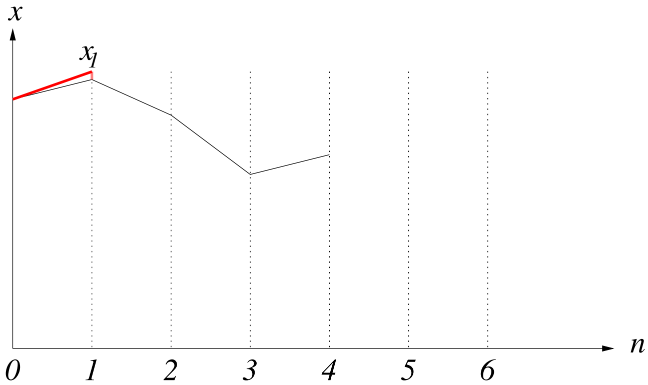


Perturbations in MPC scheme



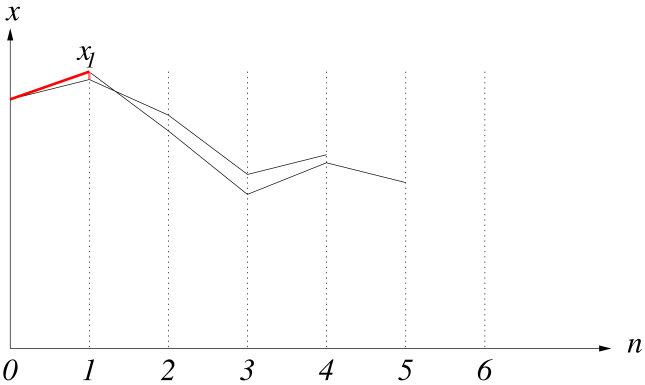
black = predictions (open loop optimization)

Perturbations in MPC scheme



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

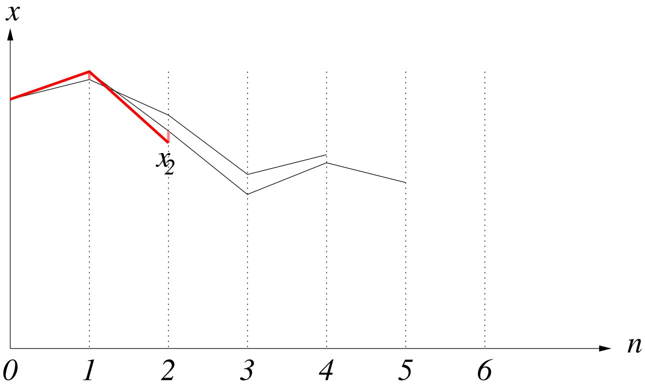
Perturbations in MPC scheme



black $=$ predictions (open loop optimization)

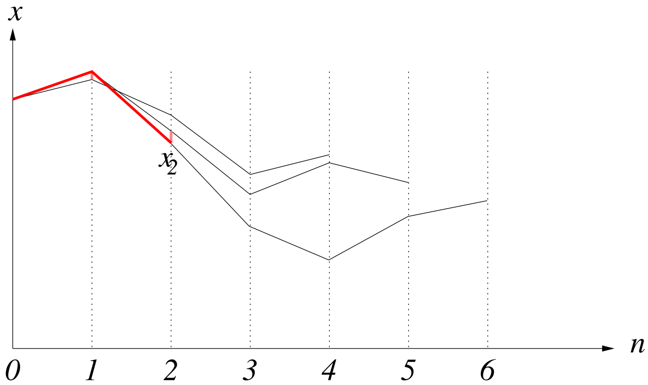
red $=$ perturbed MPC closed loop

Perturbations in MPC scheme



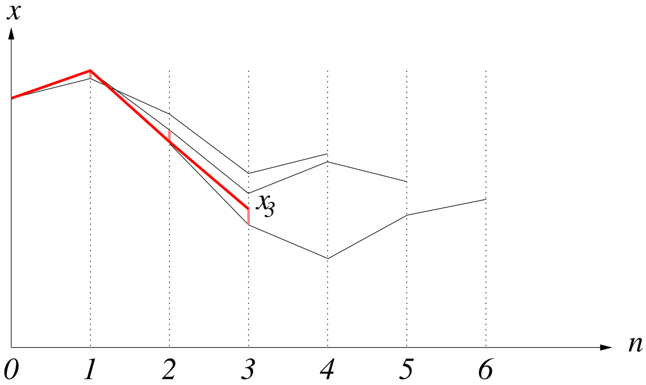
black = predictions (open loop optimization)
red = perturbed MPC closed loop

Perturbations in MPC scheme



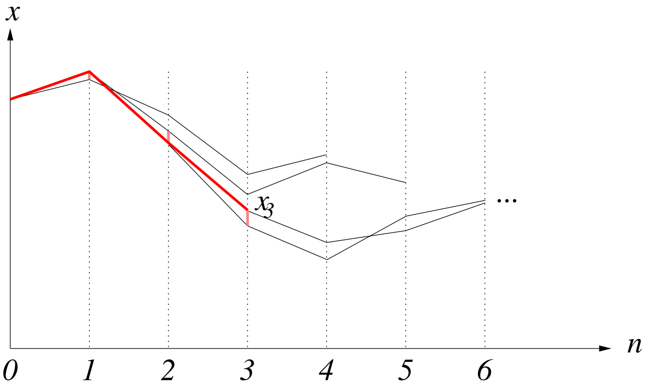
black = predictions (open loop optimization)
red = perturbed MPC closed loop

Perturbations in MPC scheme



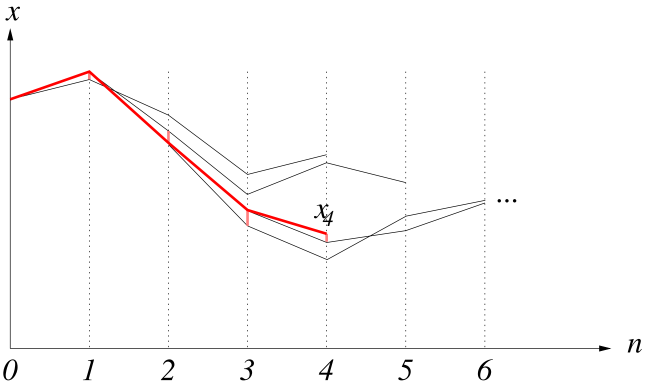
black = predictions (open loop optimization)
red = perturbed MPC closed loop

Perturbations in MPC scheme



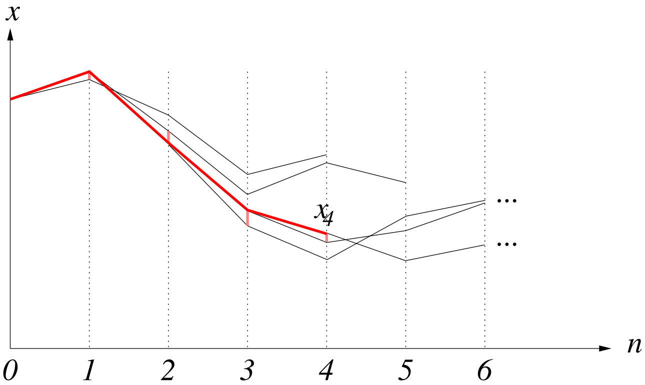
black = predictions (open loop optimization)
red = perturbed MPC closed loop

Perturbations in MPC scheme



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

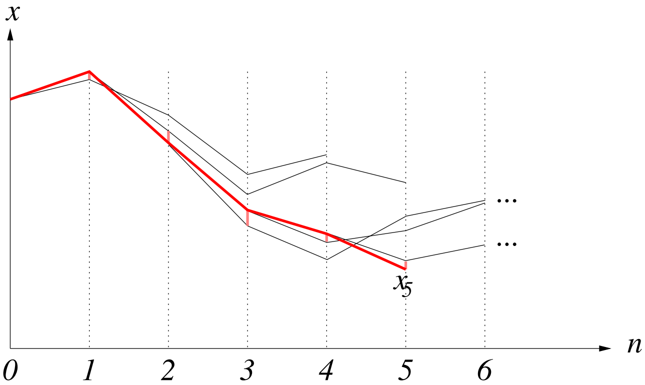
Perturbations in MPC scheme



black = predictions (open loop optimization)

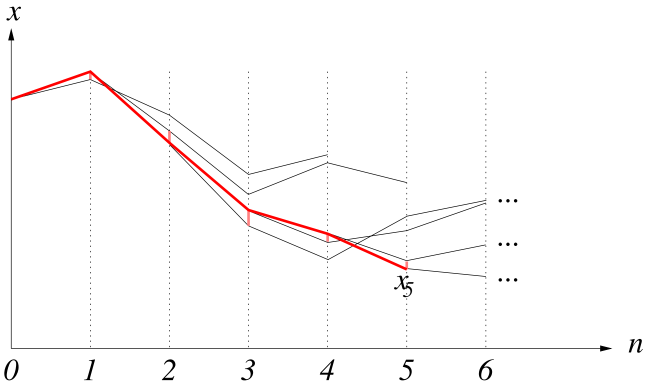
red = perturbed MPC closed loop

Perturbations in MPC scheme



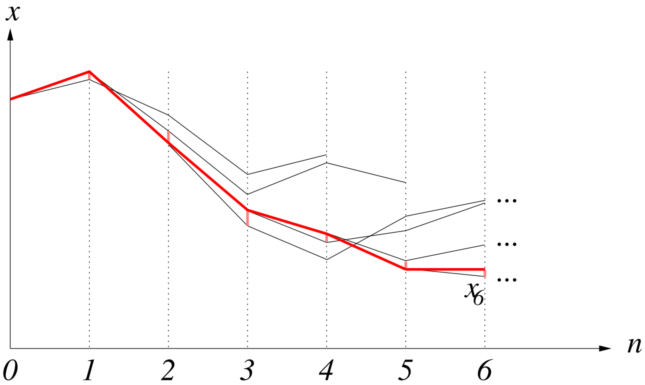
black = predictions (open loop optimization)
red = perturbed MPC closed loop

Perturbations in MPC scheme



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

Perturbations in MPC scheme



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

Robustness

Robustness can be ensured, e.g., by

- (uniform) continuity of the optimal value function $V_N(x) = \inf_u J_N(x,u)$, which serves as a Lyapunov function [De Nicolao/Magni/Scattolini '96; Nešić/Teel/Kokotović '99; Gr./Pannek '11]

Robustness

Robustness can be ensured, e.g., by

- (uniform) continuity of the optimal value function $V_N(x) = \inf_u J_N(x,u)$, which serves as a Lyapunov function [De Nicolao/Magni/Scattolini '96; Nešić/Teel/Kokotović '99; Gr./Pannek '11]

  (may not hold in presence of state constraints)

Robustness

Robustness can be ensured, e.g., by

- (uniform) continuity of the optimal value function $V_N(x) = \inf_u J_N(x,u)$, which serves as a Lyapunov function [De Nicolao/Magni/Scattolini '96; Nešić/Teel/Kokotović '99; Gr./Pannek '11]

  (may not hold in presence of state constraints)

- a specific construction of tightening state constraints [Michalska/Mayne '93; Limón/Alamo/Camacho '02; Grimm et al. '07; Gr./Pannek '11]

Robustness

Robustness can be ensured, e.g., by

- (uniform) continuity of the optimal value function $V_N(x) = \inf_u J_N(x,u)$, which serves as a Lyapunov function [De Nicolao/Magni/Scattolini '96; Nešić/Teel/Kokotović '99; Gr./Pannek '11]

  (may not hold in presence of state constraints)

- a specific construction of tightening state constraints [Michalska/Mayne '93; Limón/Alamo/Camacho '02; Grimm et al. '07; Gr./Pannek '11]

In the latter case, stability and robustness analysis must be carried out in an integrated way

Reducing the computational load

Back to the unperturbed case:

The computationally most expensive part of an MPC controller is the optimization

Reducing the computational load

Back to the unperturbed case:

The computationally most expensive part of an MPC controller is the optimization

Many approaches exist for increasing the efficiency of the optimization algorithm, see, e.g. [Diehl et al. '01ff.]

Reducing the computational load

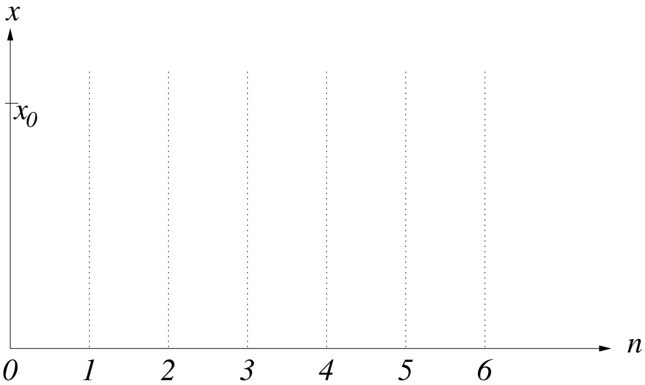
Back to the unperturbed case:

The computationally most expensive part of an MPC controller is the optimization

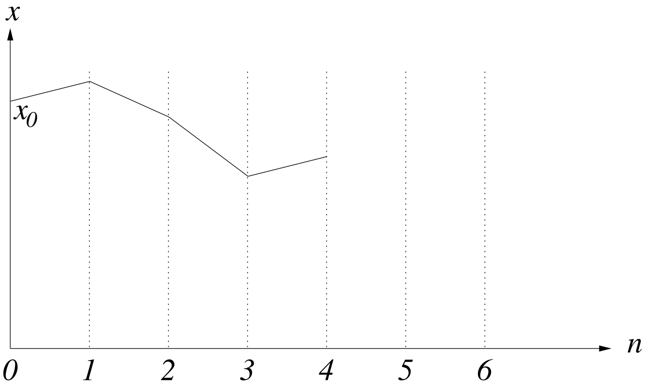
Many approaches exist for increasing the efficiency of the optimization algorithm, see, e.g. [Diehl et al. '01ff.]

A more systems theoretic approach: perform re-optimization less often

Schematic illustration of the idea

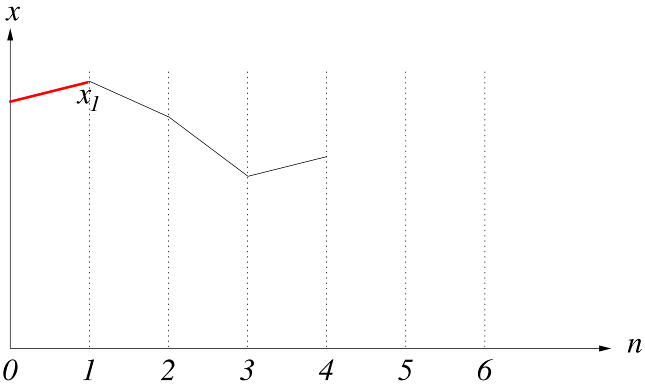


Schematic illustration of the idea



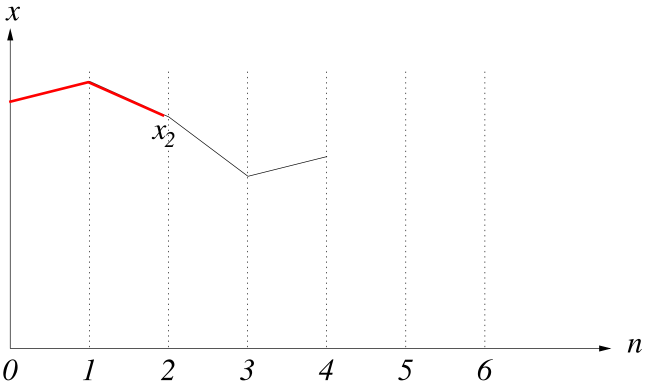
black = predictions (open loop optimization)

Schematic illustration of the idea



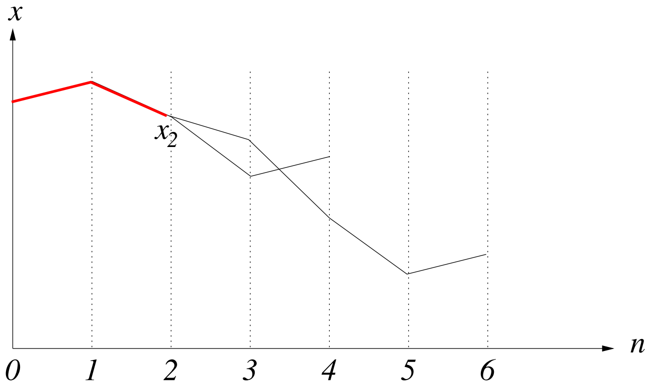
black = predictions (open loop optimization)
red = MPC closed loop

Schematic illustration of the idea



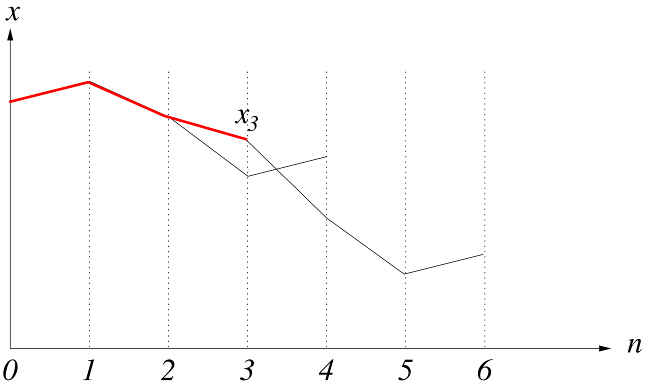
black $=$ predictions (open loop optimization)
red $=$ MPC closed loop

Schematic illustration of the idea



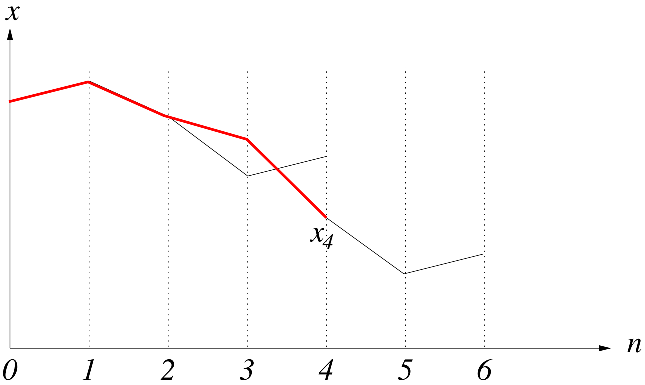
black $=$ predictions (open loop optimization)
red $=$ MPC closed loop

Schematic illustration of the idea



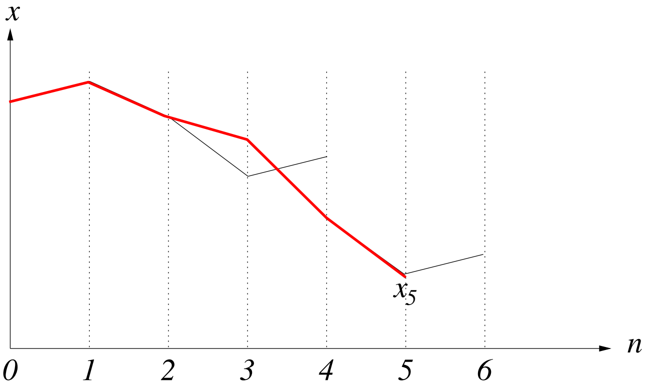
black = predictions (open loop optimization)
red = MPC closed loop

Schematic illustration of the idea



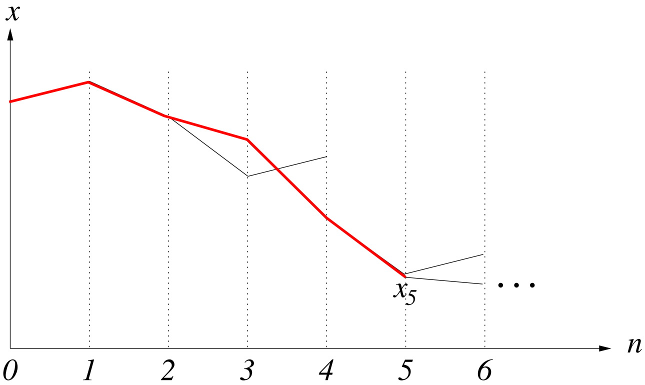
black $=$ predictions (open loop optimization)
red $=$ MPC closed loop

Schematic illustration of the idea



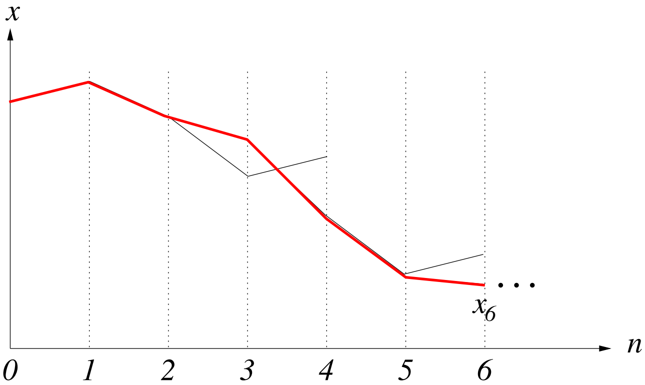
black = predictions (open loop optimization)
red = MPC closed loop

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

Stability analysis

Denote the by m_j the number of elements used from the j-th control sequence, called the "control horizon"

Stability analysis

Denote the by m_j the number of elements used from the j-th control sequence, called the "control horizon"

Then the stability and performance analysis extends to time-varying control horizons if we use $\alpha = \min_{m_j} \alpha(m_j)$ where

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

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

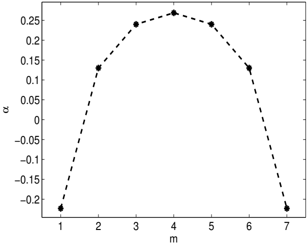
Property of $\alpha(m)$

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

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

and

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



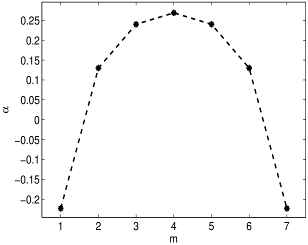
Property of $\alpha(m)$

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

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

and

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



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 $m_i \in \{1, \ldots, N-1\}$

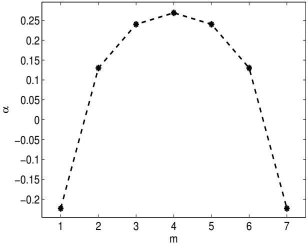
Property of $\alpha(m)$

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

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

and

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



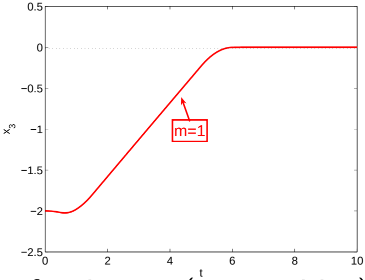
Corollary: If N is such that all C, σ-exponentially controllable systems are stabilized with "classical" MPC ($m = 1$), then they are stabilized for arbitrary varying control horizons $m_i \in \{1, \ldots, N-1\}$

How does $\alpha(m)$ look like for a single system?

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

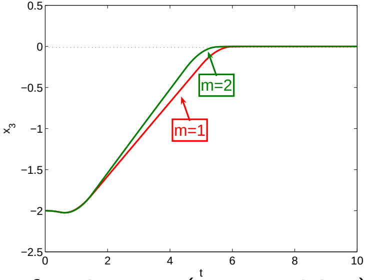


x_3 component of trajectory (cart position) for different m

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

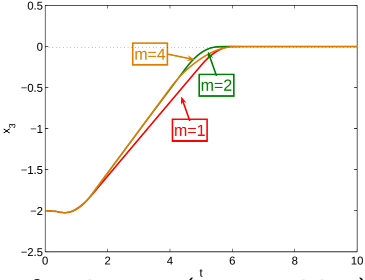


x_3 component of trajectory (cart position) for different m

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

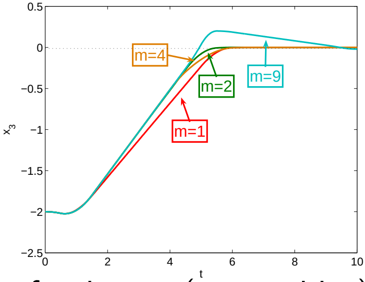


x_3 component of trajectory (cart position) for different m

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

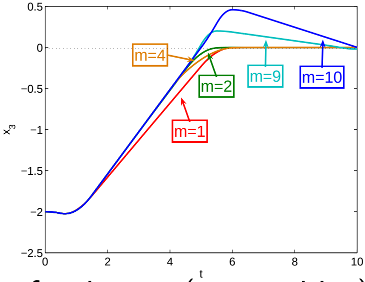


x_3 component of trajectory (cart position) for different m

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

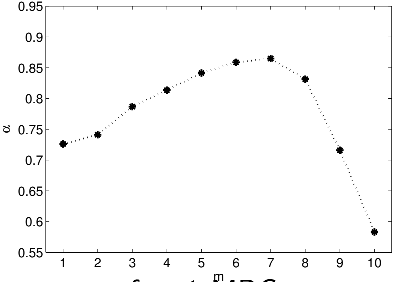


x_3 component of trajectory (cart position) for different m

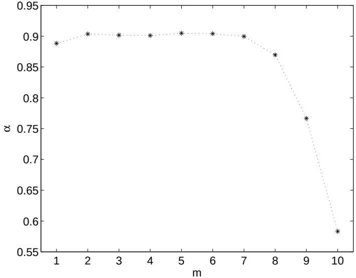
Example: linearized inverted pendulum

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

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



α after 1 MPC step



α at time $n = 20$

Discussion of the approach

Conclusion:

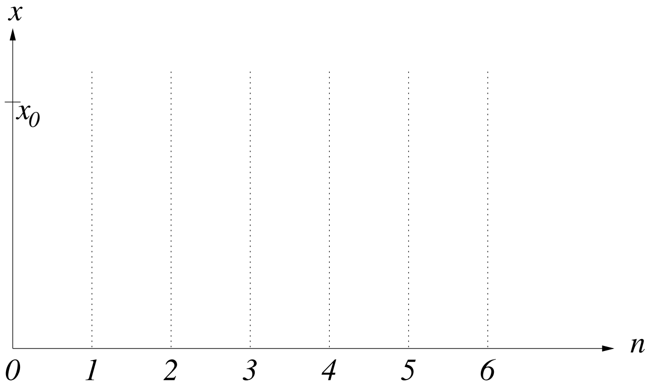
- longer control horizons can be used without affecting the nominal (=unperturbed) stability and performance

Discussion of the approach

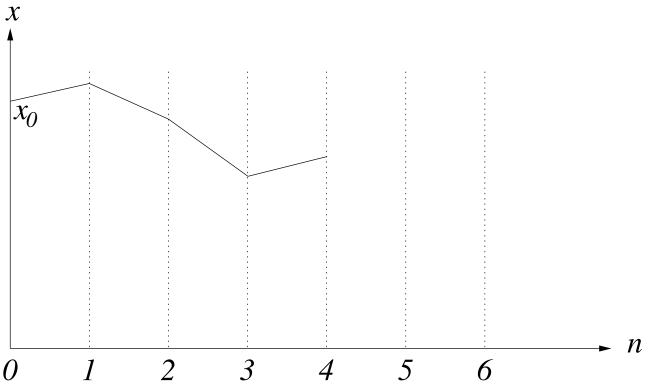
Conclusion:

- longer control horizons can be used without affecting the nominal (=unperturbed) stability and performance
- but: longer control horizons may reduce robustness

Problem of the approach: less robustness

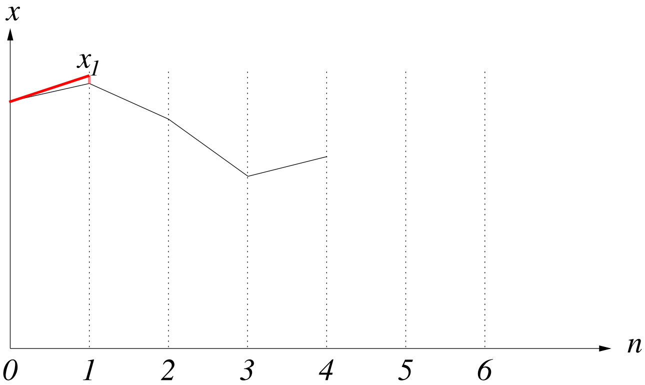


Problem of the approach: less robustness



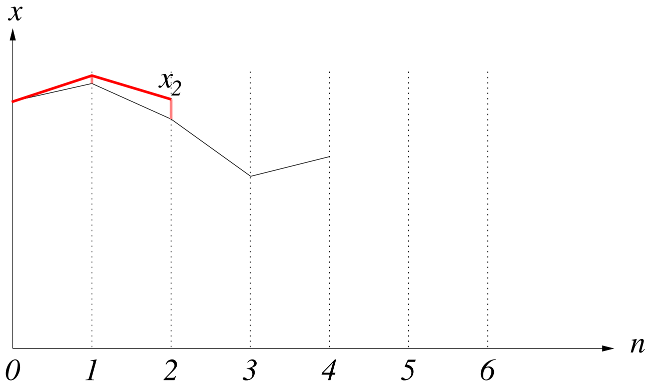
black = predictions (open loop optimization)

Problem of the approach: less robustness



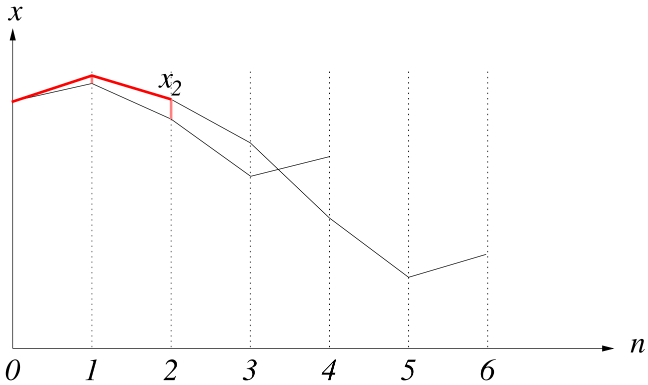
black = predictions (open loop optimization)
red = perturbed MPC closed loop

Problem of the approach: less robustness



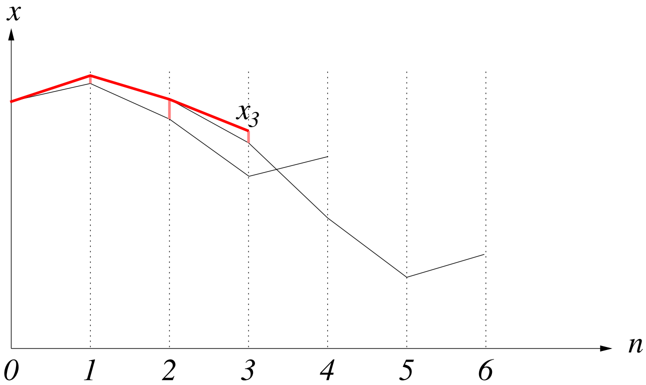
black $=$ predictions (open loop optimization)
red $=$ perturbed MPC closed loop

Problem of the approach: less robustness



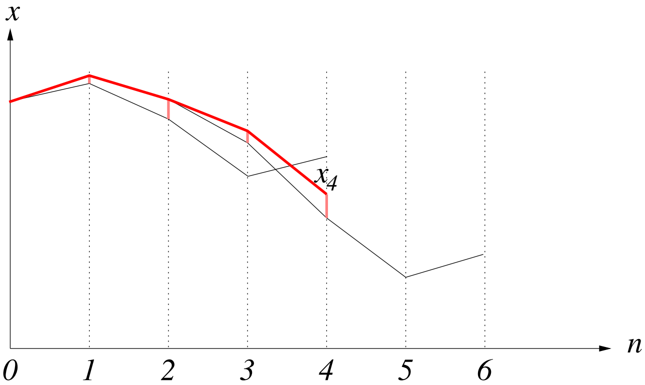
black = predictions (open loop optimization)
red = perturbed MPC closed loop

Problem of the approach: less robustness



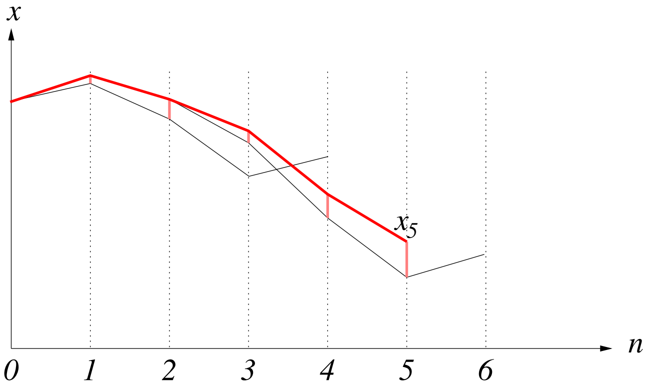
black = predictions (open loop optimization)
red = perturbed MPC closed loop

Problem of the approach: less robustness



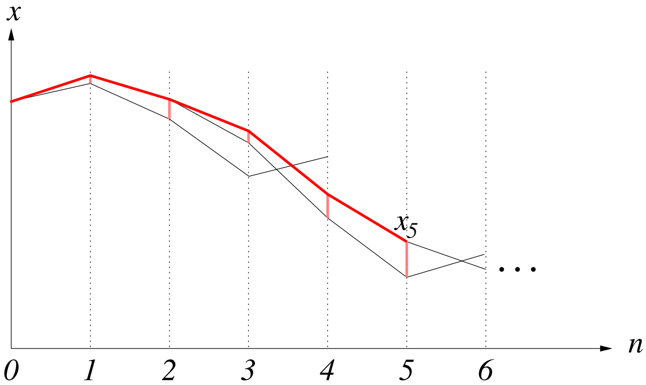
black = predictions (open loop optimization)
red = perturbed MPC closed loop

Problem of the approach: less robustness



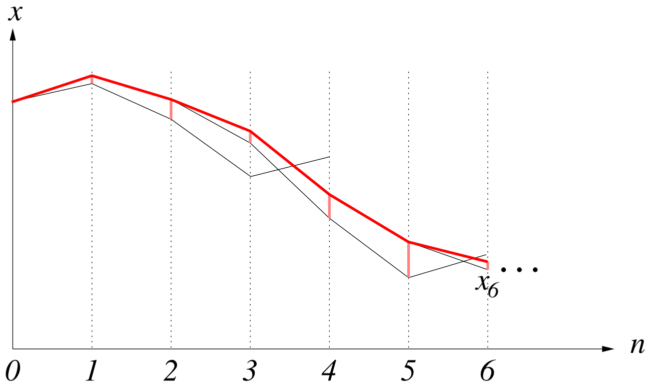
black = predictions (open loop optimization)
red = perturbed MPC closed loop

Problem of the approach: less robustness



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

Problem of the approach: less robustness



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

Discussion of the approach

Conclusion:

- longer control horizons can be used without affecting the nominal (=unperturbed) stability and performance
- but: longer control horizons may reduce robustness

Discussion of the approach

Conclusion:

- longer control horizons can be used without affecting the nominal (=unperturbed) stability and performance
- but: longer control horizons may reduce robustness

Remedy:

- use sensitivity based techniques to update the “tails” of the optimal control sequences
- perform an integrated robustness and stability analysis

This will be the starting point for SADCO Task 3.3

Summary and outlook

- we developed a stability and guaranteed performance analysis method for MPC schemes

Summary and outlook

- we developed a stability and guaranteed performance analysis method for MPC schemes
- with this method we can compute optimization horizon bounds N under controllability assumptions

Summary and outlook

- we developed a stability and guaranteed performance analysis method for MPC schemes
- with this method we can compute optimization horizon bounds N under controllability assumptions
- the approach can be coupled with robust MPC variants

Summary and outlook

- we developed a stability and guaranteed performance analysis method for MPC schemes
- with this method we can compute optimization horizon bounds N under controllability assumptions
- the approach can be coupled with robust MPC variants
- the method can be extended to analyzing varying control horizons $m_i \in \{1, \dots, M\}$

Summary and outlook

- we developed a stability and guaranteed performance analysis method for MPC schemes
- with this method we can compute optimization horizon bounds N under controllability assumptions
- the approach can be coupled with robust MPC variants
- the method can be extended to analyzing varying control horizons $m_i \in \{1, \ldots, M\}$
- main conclusion: larger and varying control horizons can be used without losing (nominal) stability and performance

Summary and outlook

- we developed a stability and guaranteed performance analysis method for MPC schemes
- with this method we can compute optimization horizon bounds N under controllability assumptions
- the approach can be coupled with robust MPC variants
- the method can be extended to analyzing varying control horizons $m_i \in \{1, \dots, M\}$
- main conclusion: larger and varying control horizons can be used without losing (nominal) stability and performance
- However, longer control horizons may reduce robustness

Summary and outlook

- we developed a stability and guaranteed performance analysis method for MPC schemes
- with this method we can compute optimization horizon bounds N under controllability assumptions
- the approach can be coupled with robust MPC variants
- the method can be extended to analyzing varying control horizons $m_i \in \{1, \ldots, M\}$
- main conclusion: larger and varying control horizons can be used without losing (nominal) stability and performance
- However, longer control horizons may reduce robustness
- tasks in SADCO project:

Summary and outlook

- we developed a stability and guaranteed performance analysis method for MPC schemes
- with this method we can compute optimization horizon bounds N under controllability assumptions
- the approach can be coupled with robust MPC variants
- the method can be extended to analyzing varying control horizons $m_i \in \{1, \ldots, M\}$
- main conclusion: larger and varying control horizons can be used without losing (nominal) stability and performance
- However, longer control horizons may reduce robustness
- tasks in SADCO project:
  - improve robustness using sensitivity techniques

Summary and outlook

- we developed a stability and guaranteed performance analysis method for MPC schemes
- with this method we can compute optimization horizon bounds N under controllability assumptions
- the approach can be coupled with robust MPC variants
- the method can be extended to analyzing varying control horizons $m_i \in \{1, \dots, M\}$
- main conclusion: larger and varying control horizons can be used without losing (nominal) stability and performance
- However, longer control horizons may reduce robustness
- tasks in SADCO project:
    - improve robustness using sensitivity techniques
    - integrated stability and robustness analysis