Nonlinear Nonlinear Model Predictive Control

Theory & Implementation

Mazen Alamir

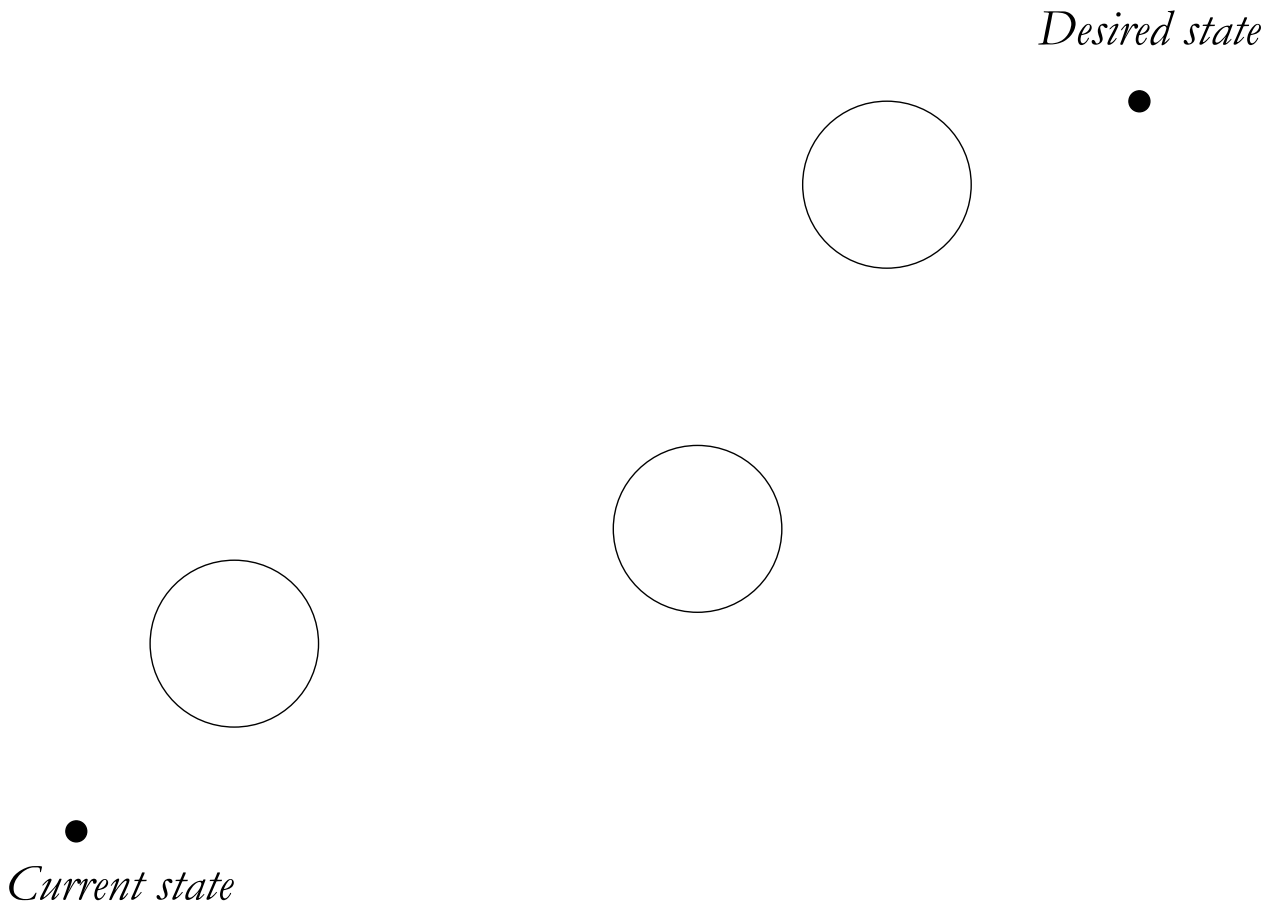
University of Grenoble, France,

CNRS / Gipsa-lab / Control Systems Department

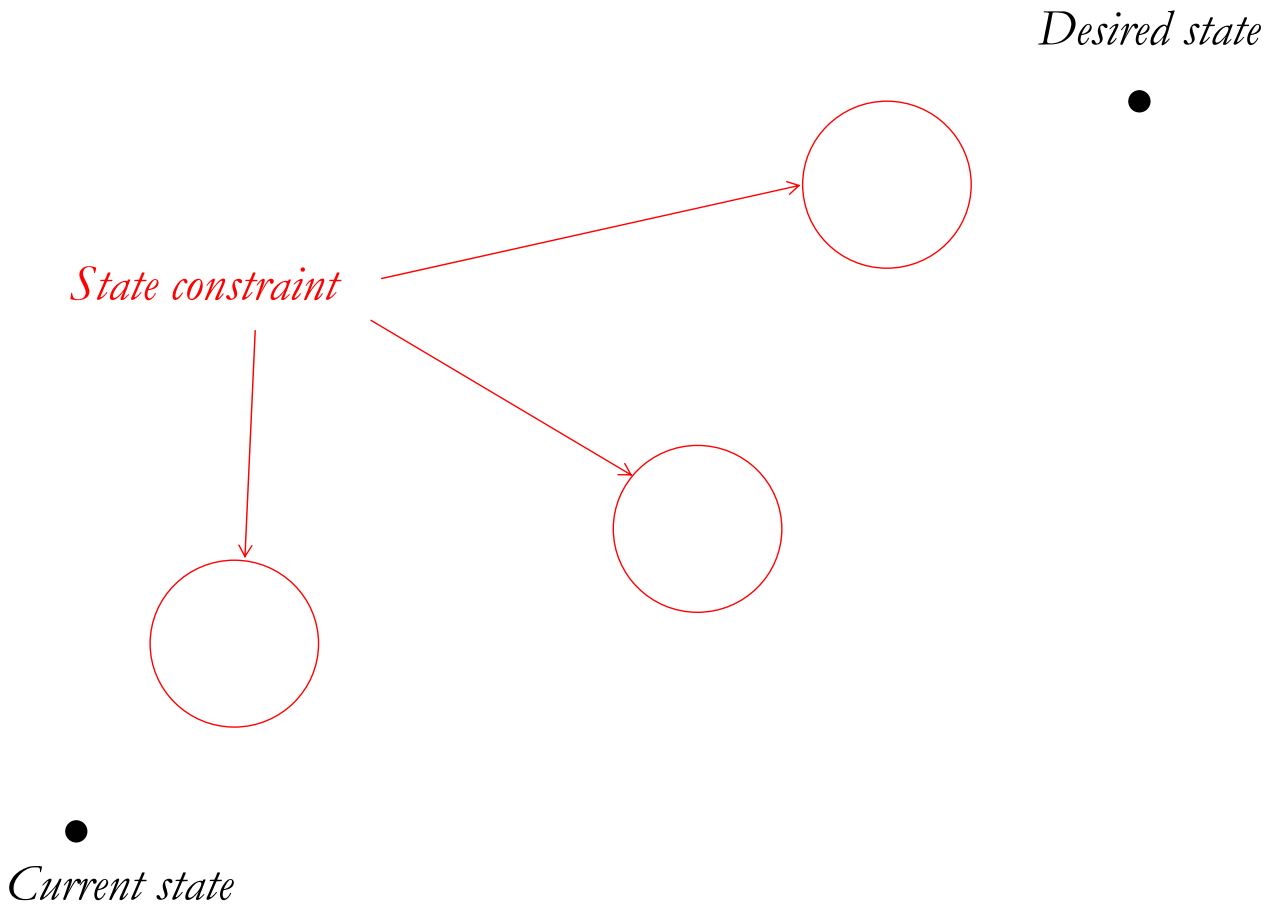
(Journée CCT-CNES, Toulouse, 15 Janvier 2009)

NMPC : You are already using it . . . !

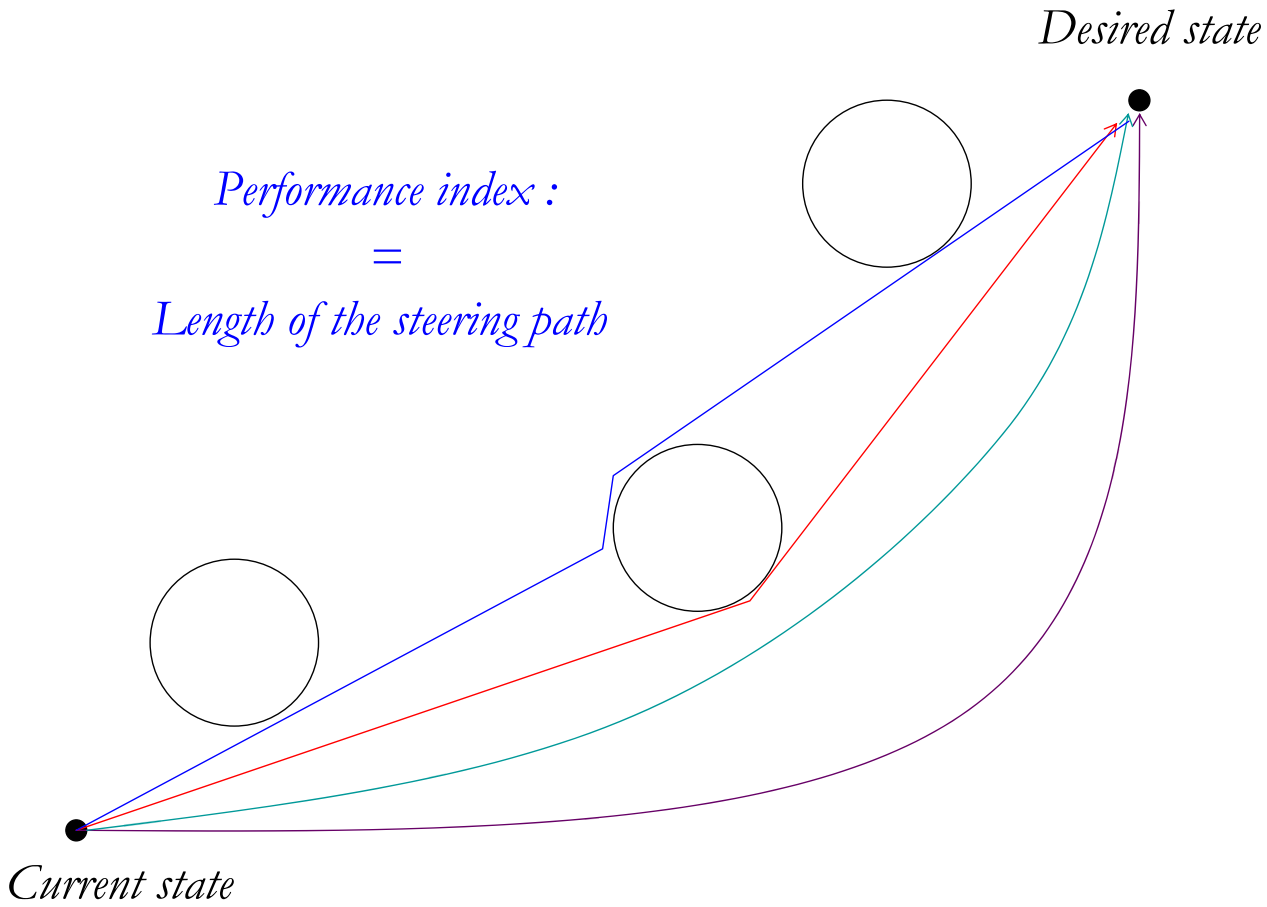
NMPC : You are already using it . . . !



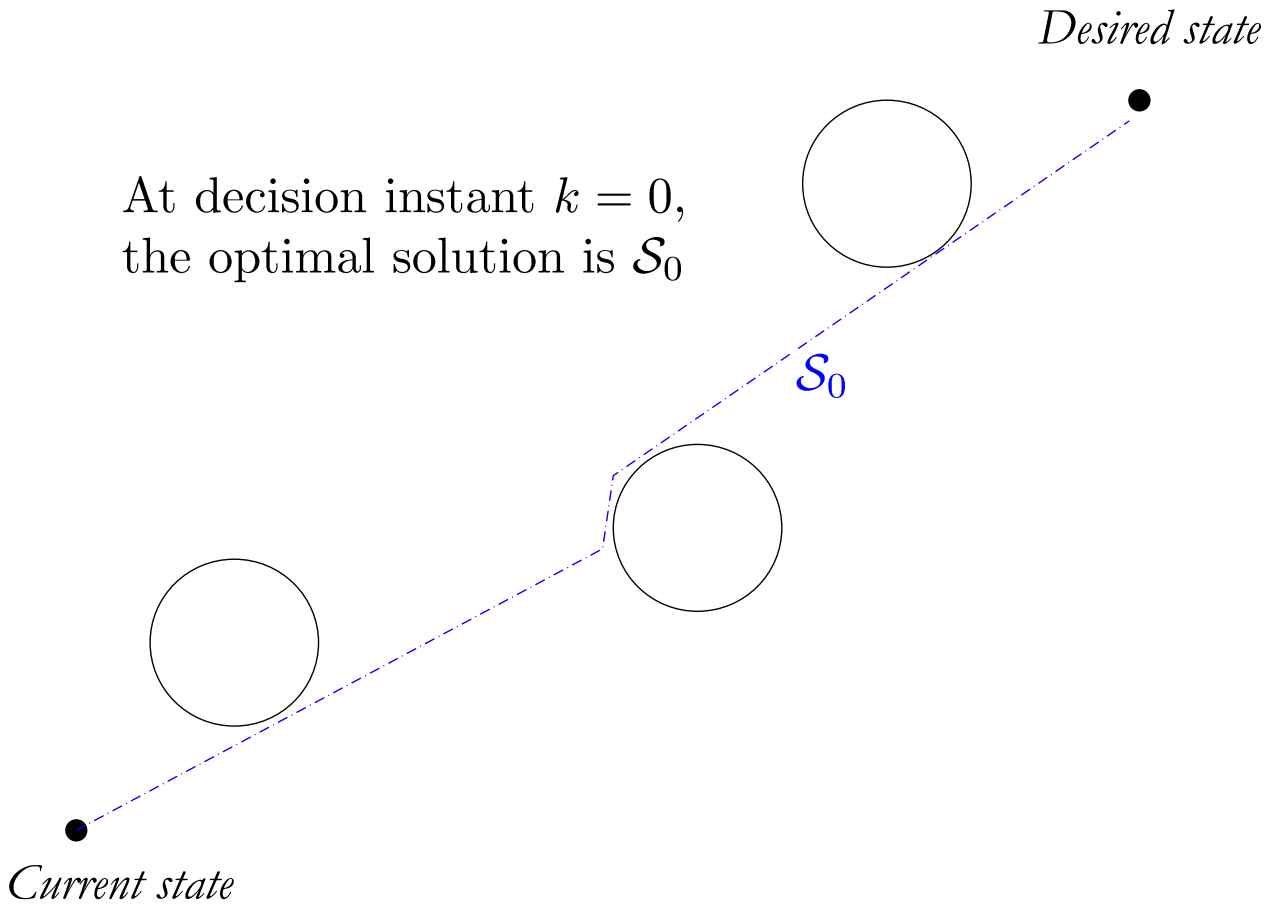
NMPC : You are already using it . . . !



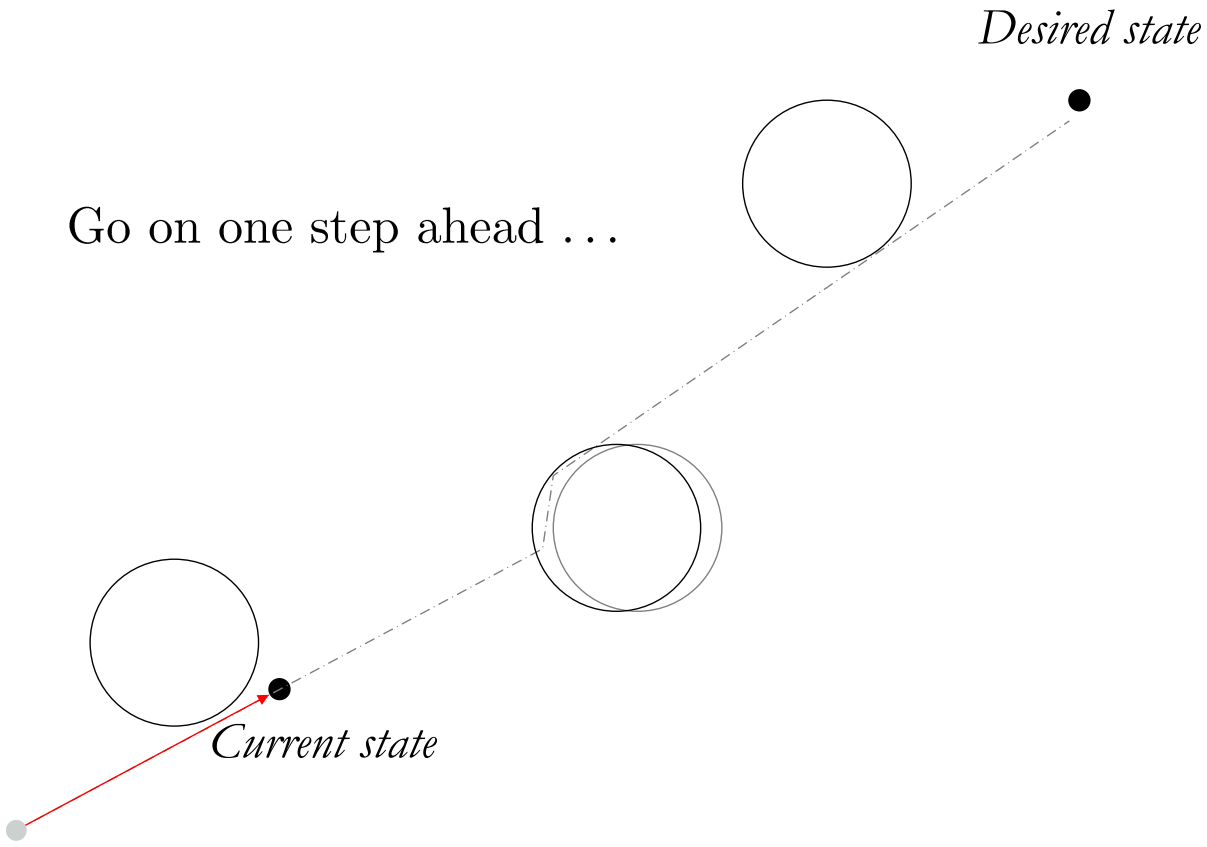
NMPC : You are already using it . . . !



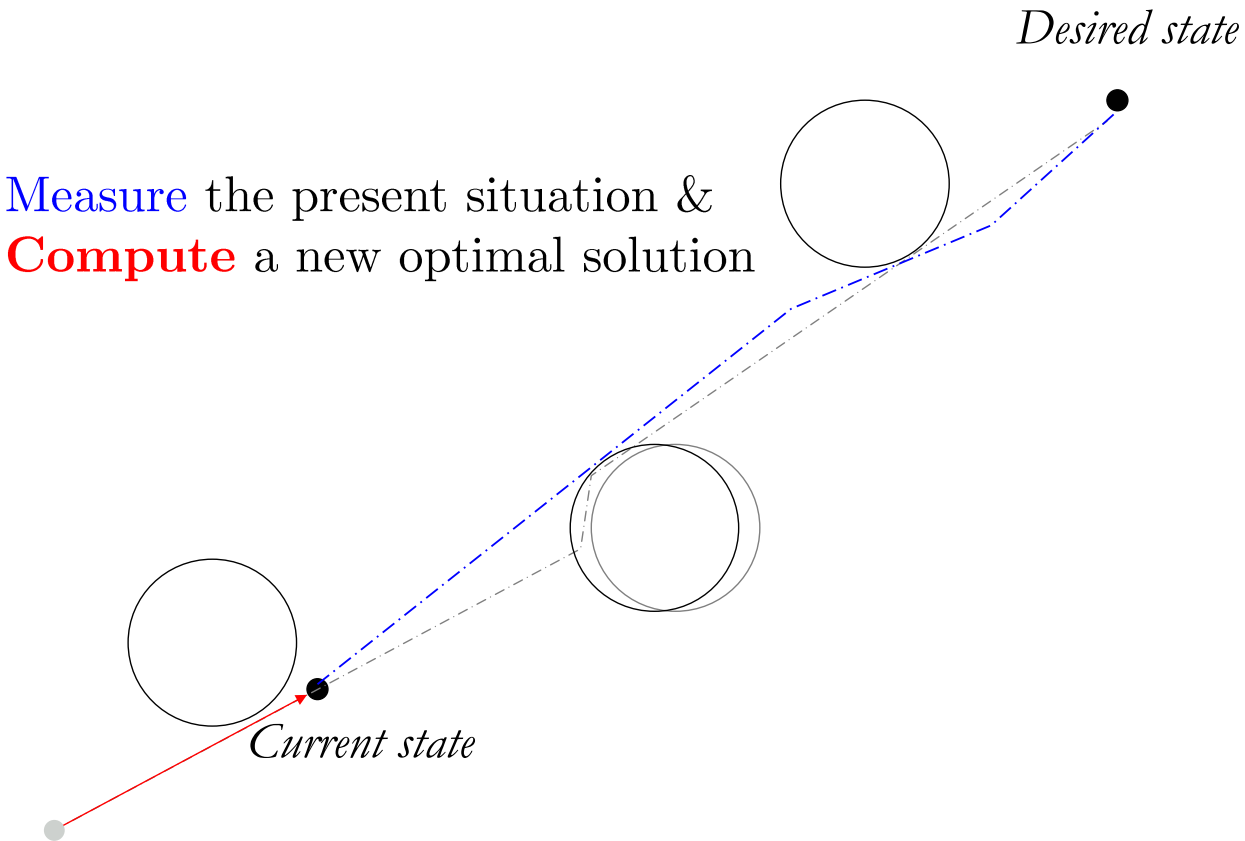
NMPC : You are already using it . . . !



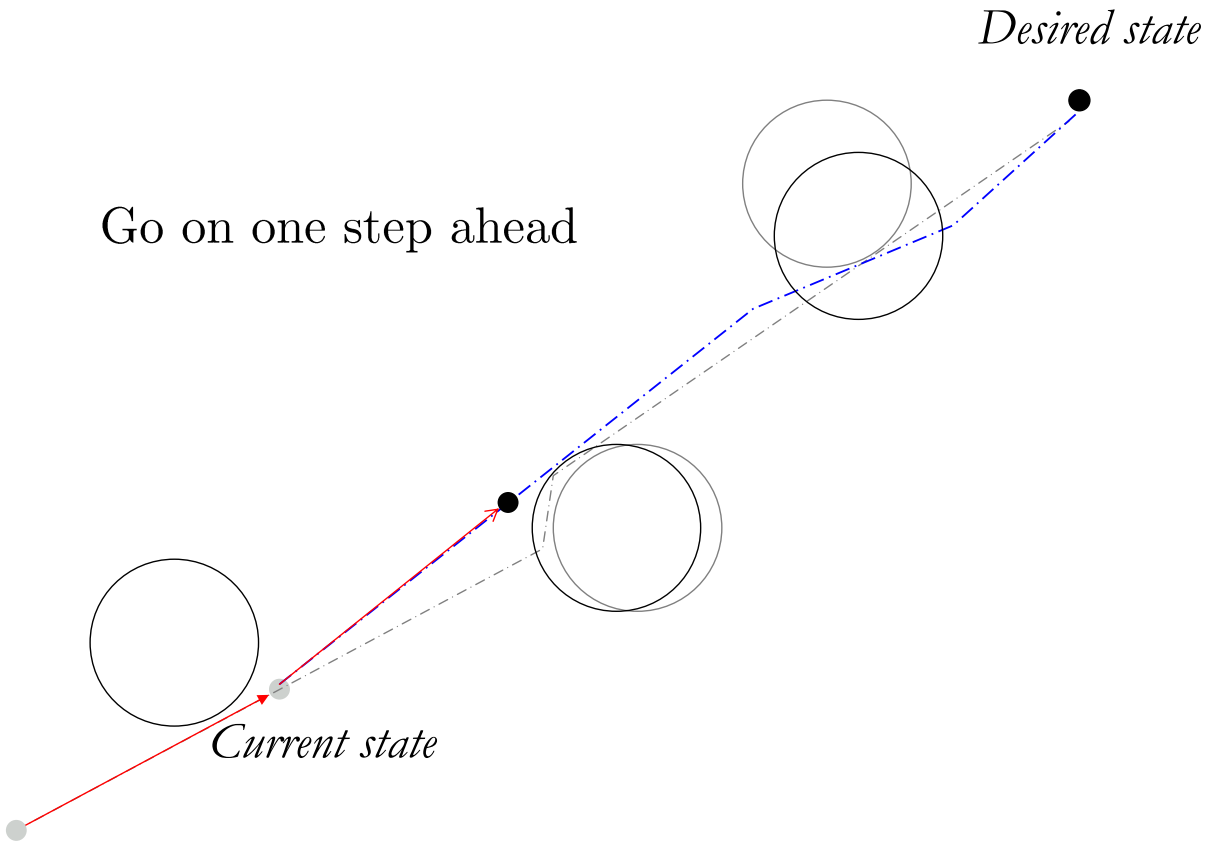
NMPC : You are already using it . . . !



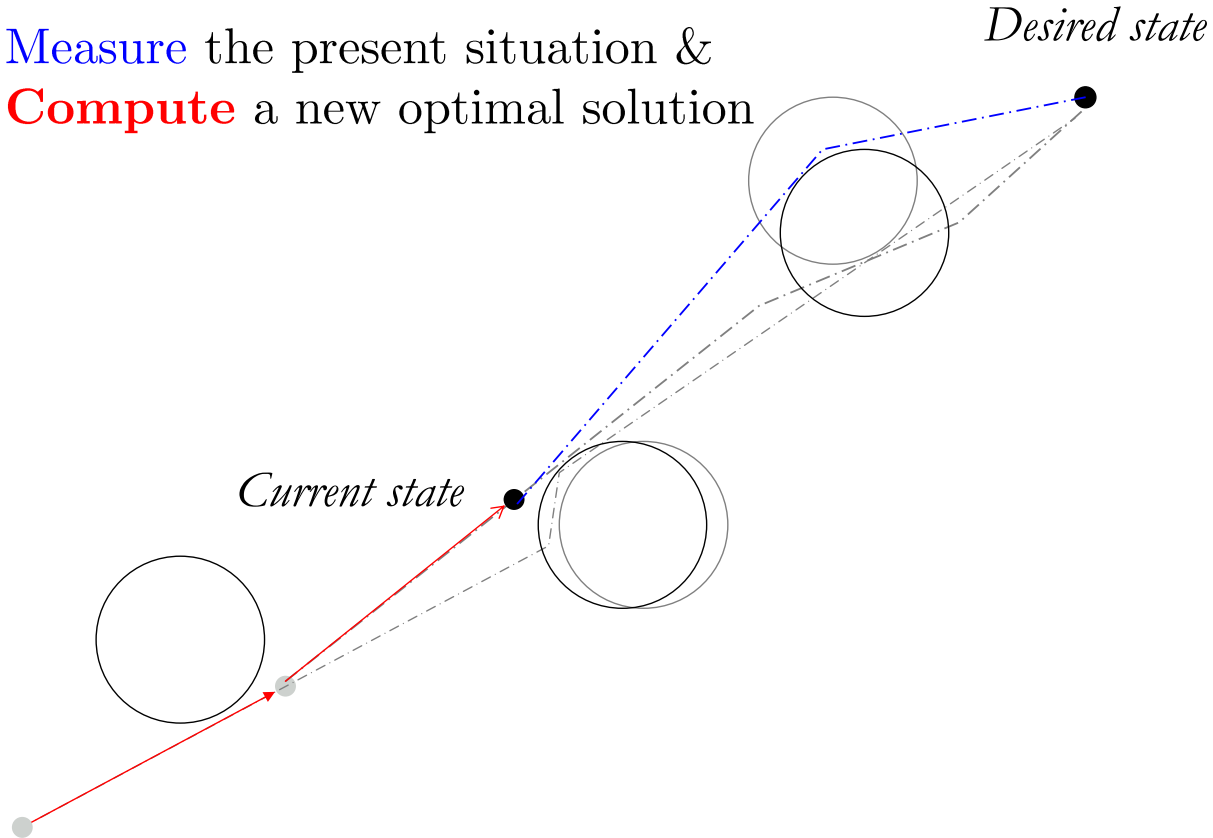
NMPC : You are already using it . . . !



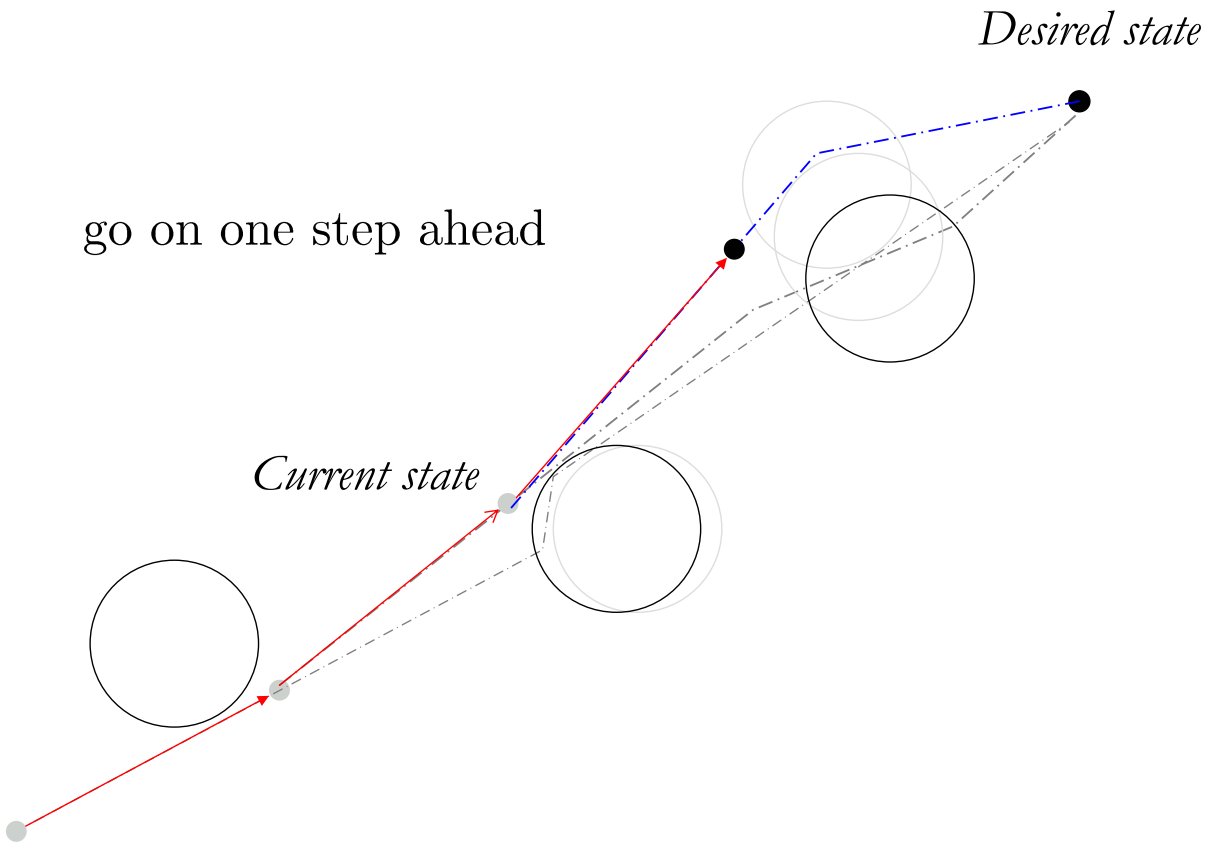
NMPC : You are already using it . . . !



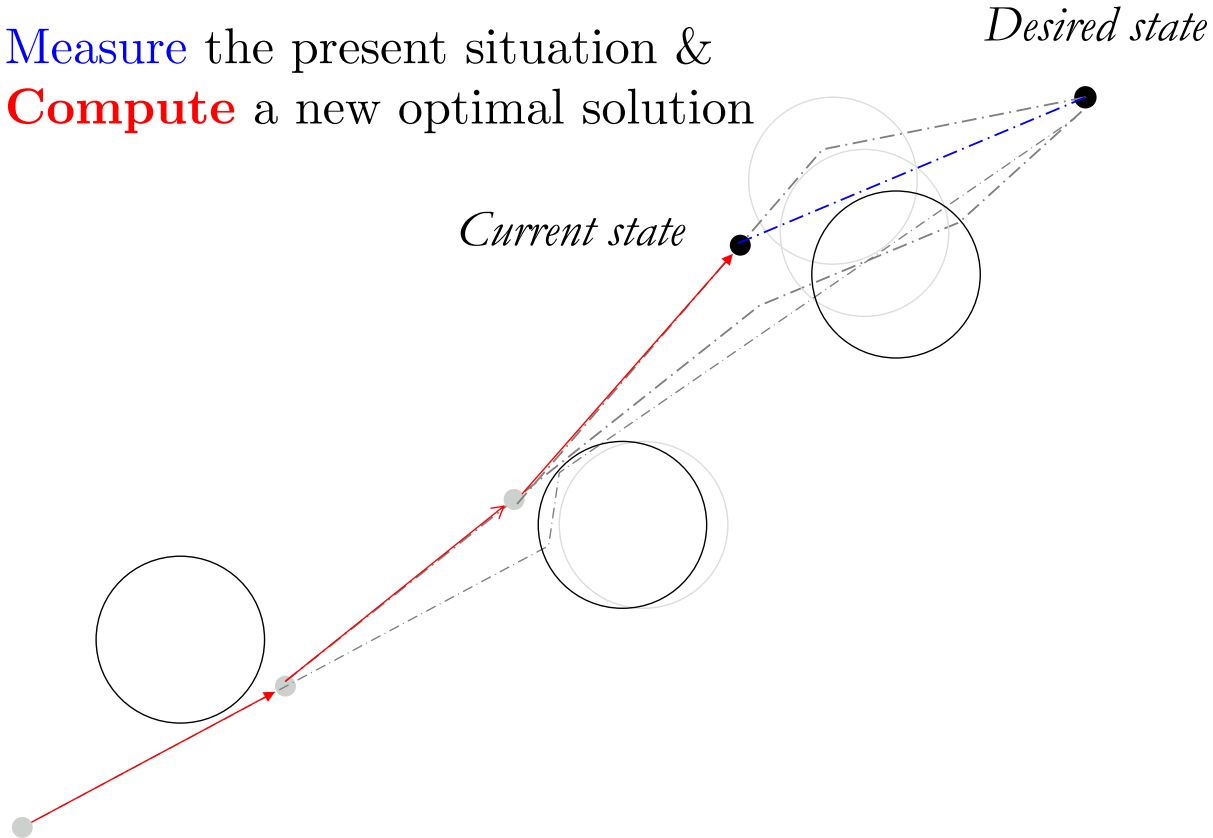
NMPC : You are already using it . . . !



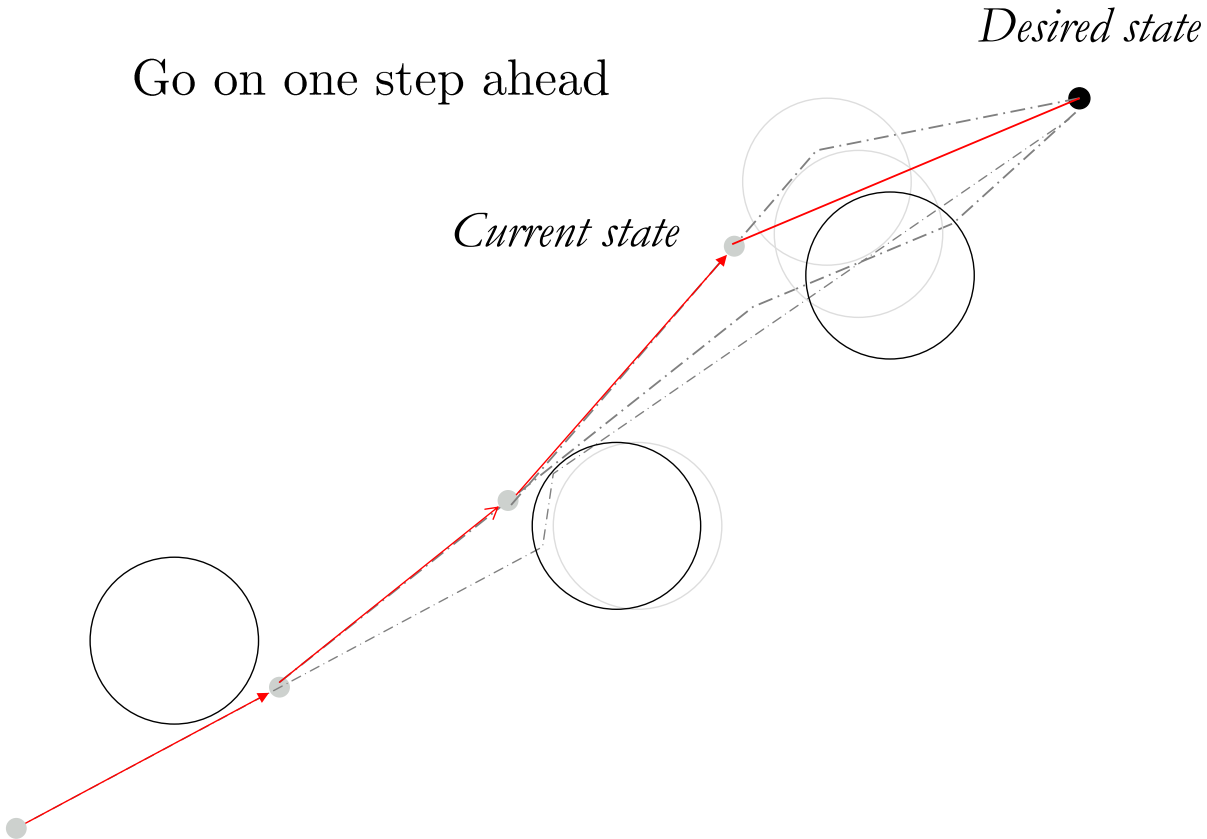
NMPC : You are already using it . . . !



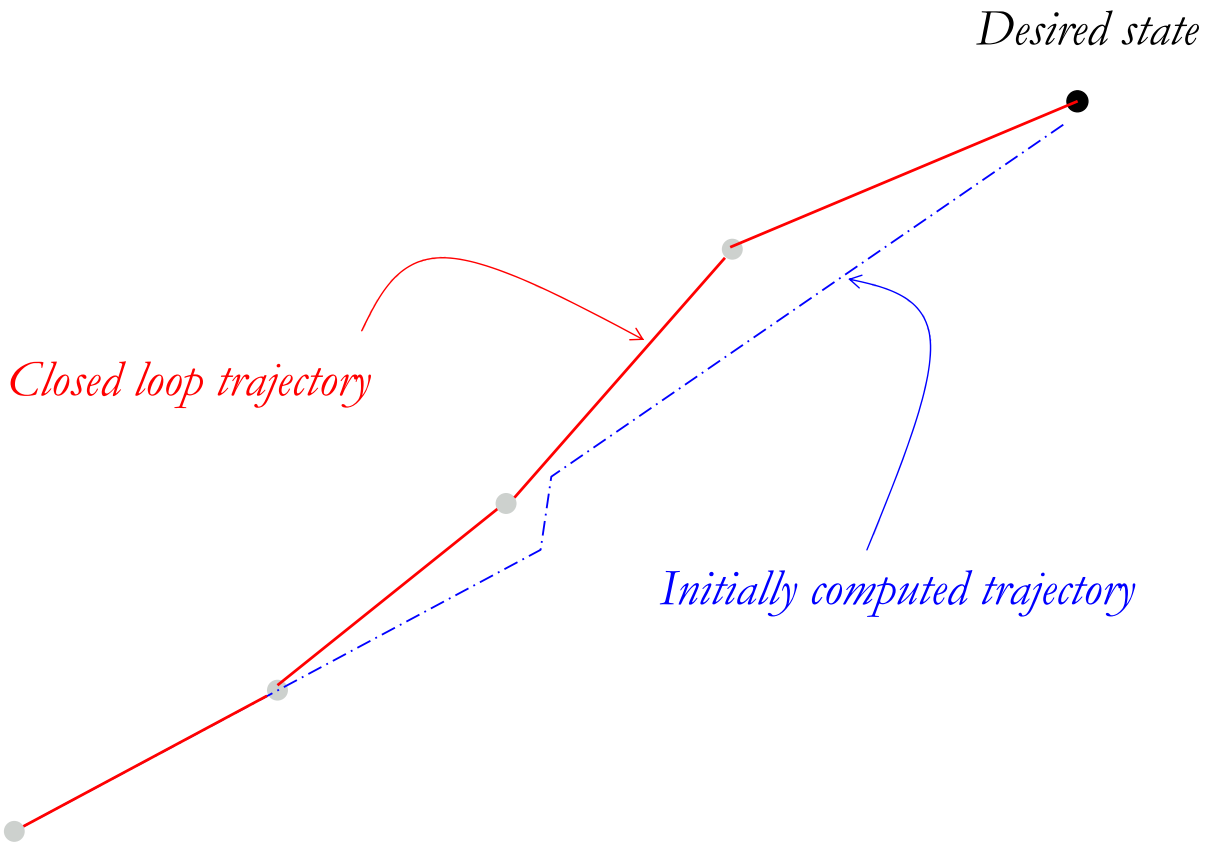
NMPC : You are already using it . . . !



NMPC : You are already using it . . . !



NMPC : You are already using it . . . !



More formally

Consider a dynamical system :

$$x(t) = X(t, x_0, \mathbf{u}) \quad x \in \mathbb{R}^n \quad \mathbf{u} \in \mathbb{U}^{[0,T]}$$

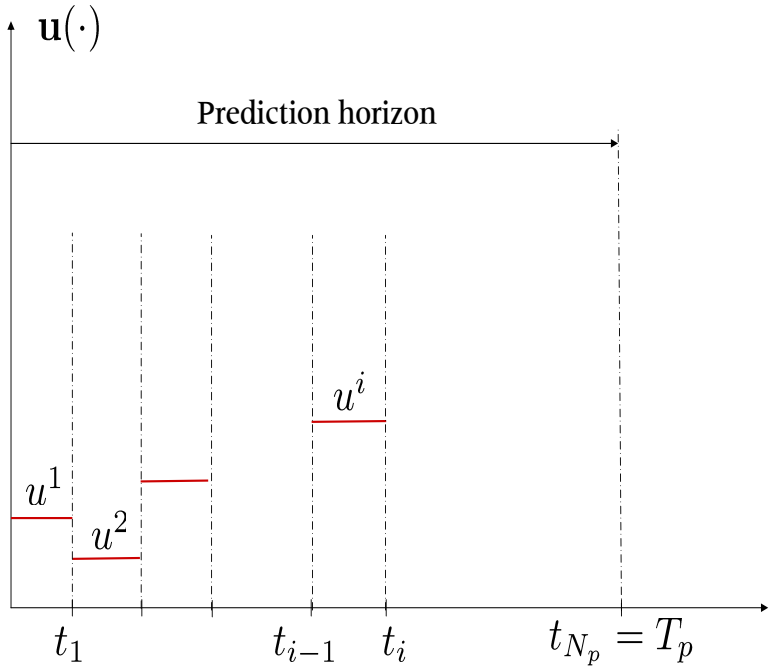
More formally

Consider a dynamical system :

$$x(t) = X(t, x_0, \mathbf{u}) \quad x \in \mathbb{R}^n \quad \mathbf{u} \in \mathbb{U}^{[0,T]}$$

control parametrization :

$$\mathbf{u}(t) \quad = \quad u^{(k)}(p) \quad ; \quad t \in [t_{k-1}, t_k]$$



More formally

Consider a dynamical system :

$$x(t) = X(t, x_0, \mathbf{u}) \quad x \in \mathbb{R}^n \quad \mathbf{u} \in \mathbb{U}^{[0,T]}$$

control parametrization :

$$\mathbf{u}(t) \quad = \quad u^{(k)}(p) \quad ; \quad t \in [t_{k-1}, t_k]$$

More generally

Any map

$$C : \mathbb{P} \rightarrow \mathbb{U}^N$$
$$C(p) = \begin{pmatrix} u^1(p) & \dots & u^N(p) \end{pmatrix}$$

defines a $\mathbb{P}$-parametrized piecewise constant control profile

$$\mathbf{u} = \mathcal{U}_{pwc}(\cdot, p)$$

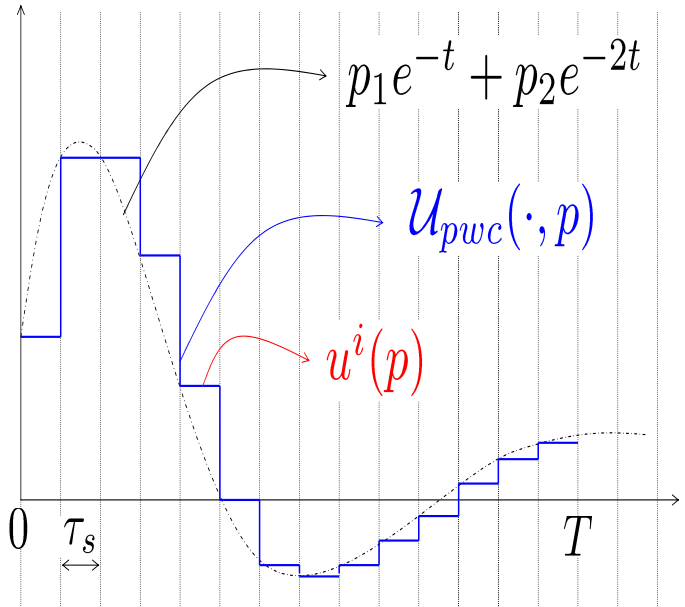
More formally

Consider a dynamical system :

$$x(t) = X(t, x_0, \mathbf{u}) \quad x \in \mathbb{R}^n \quad \mathbf{u} \in \mathbb{U}^{[0,T]}$$

control parametrization :

$$\mathbf{u}(t) \quad = \quad u^{(k)}(p) \quad ; \quad t \in [t_{k-1}, t_k]$$



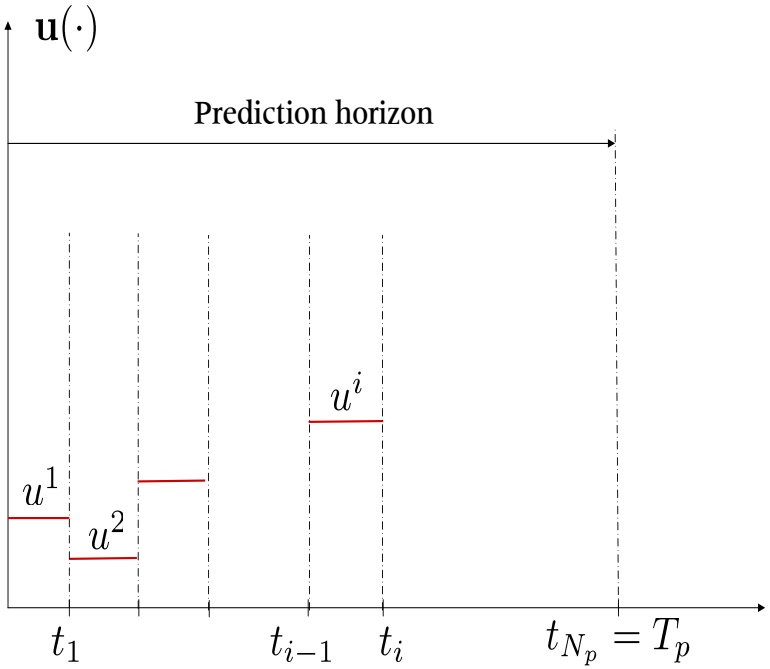
More formally

Consider a dynamical system :

$$x(t) = X(t, x_0, \mathbf{u}) \quad x \in \mathbb{R}^n \quad \mathbf{u} \in \mathbb{U}^{[0,T]}$$

control parametrization :

$$\mathbf{u}(t) \quad = \quad u^{(k)}(p) \quad ; \quad t \in [t_{k-1}, t_k]$$



NMPC

$$K := u^{(1)} \circ \hat{p} : \mathbb{R}^n \longrightarrow \mathbb{U}$$

$$\hat{p}(x(t_j)) := \arg\min_{p \in \mathbb{P}} \Big[J(x(t_j), p) \Big] \quad \text{under} \quad C(x(t_j), p) \leq 0$$

gipsa-lab

NMPC : Interesting . . .But . . .

NMPC : Interesting . . .But . . .

Advantages

1. No structural assumption on the model
2. Expressing optimality and trade-off
3. Systematic constraint handling

NMPC : Interesting . . .But . . .

Advantages

1. No structural assumption on the model
2. Expressing optimality and trade-off
3. Systematic constraint handling

Price to pay

Solving ON-LINE non convex optimization problems.

MPC is almost a technology . . . (but not NMPC)

Non-exhaustive MPC Vendor List

- ABB
- ACT
- Adaptics
- Adaptive Resources
- Adersa Home Page
- Aspen Technology
- Aurel Systems Inc.
- Batch CAD
- Bonner and Moore
- Brainwave
- C.F. Picou and Associates
- Chemstations
- Comdale Technologies
- Control Arts Inc.
- Control Consulting Inc.
- Control Dynamics Homepage
- Controlsoft Incorporated
- Cybosoft
- DOT Products
- Trieber Controls
- Yokogawa APC
- US Process Control L.L.C.
- Eldridge Engineering Inc.
- Elsag Bailey
- Envision Systems Inc.
- Gensym
- Enterprise Control Technologies
- Fantoft Process Group
- MATHWORKS
- Honeywell
- Hyprotech
- Inferential Control Company
- IntellOpt
- Knowledgescape
- MDC Technology
- Neuralware
- Nexus Engineering
- Objectspace
- Optimal Control Research
- Pavilion Technologies
- Predictive Control Ltd.
- Process System Consultants
- RSI
- Simulation and Advanced Controls Inc.
- Simtech
- Texas Controls Inc.

Typically linear applications and tools

R. Findeisen: NMPC for Fast Nonlinear Systems (Workshop CDC 2006, San Diego)

The stability is an issue !

Consider the dynamical system

$$\begin{aligned} x_1^+ &= x_1 + (1 + x_2^2)u \\ x_2^+ &= \frac{3}{2}x_2 - x_1 e^u \end{aligned}$$

The stability is an issue !

Consider the dynamical system

$$\begin{aligned} x_1^+ &= x_1 + (1 + x_2^2)u \\ x_2^+ &= \frac{3}{2}x_2 - x_1 e^u \end{aligned}$$

- ✓ Open-loop instable.
- ✓ Set of equilibrium states

$$\mathcal{E}_{st} = \left\{ x^{(\alpha)} := \begin{pmatrix} \alpha \\ 2\alpha \end{pmatrix} \quad ; \quad \alpha \in \mathbb{R} \right\}$$

Control objective starting at $x^{(0)} = (0, 0)$, stabilize the system around $x^{(1)} = (1, 2)$.

The stability is an issue !

Consider the dynamical system

$$\begin{aligned} x_1^+ &= x_1 + (1 + x_2^2)u \\ x_2^+ &= \frac{3}{2}x_2 - x_1 e^u \end{aligned}$$

- ✓ Open-loop instable.
- ✓ Set of equilibrium states

$$\mathcal{E}_{st} = \left\{ x^{(\alpha)} := \begin{pmatrix} \alpha \\ 2\alpha \end{pmatrix} \quad ; \quad \alpha \in \mathbb{R} \right\}$$

Control objective starting at $x^{(0)} = (0, 0)$, stabilize the system around $x^{(1)} = (1, 2)$.

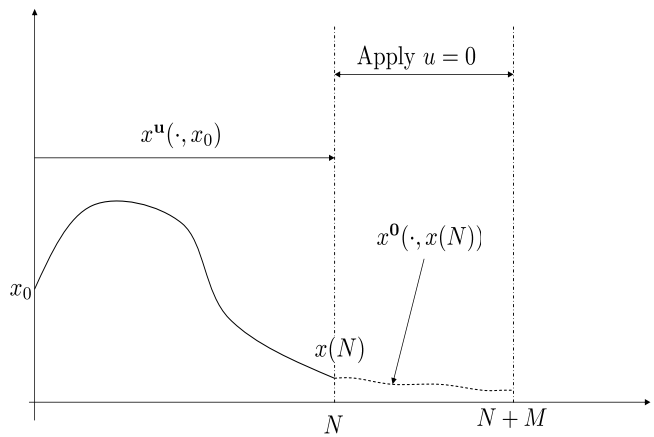
Consider the cost function

$$J(x, \mathbf{u}) := F(x(N)) + \sum_{i=0}^{N} L(x(i), u(i))$$

where

$$L(x, u) := \|x - x^{(1)}\|^2 + ru^2$$

$$F(x) = \sum_{i=1}^{M-N} x^{\mathbf{0}}(i; x)$$

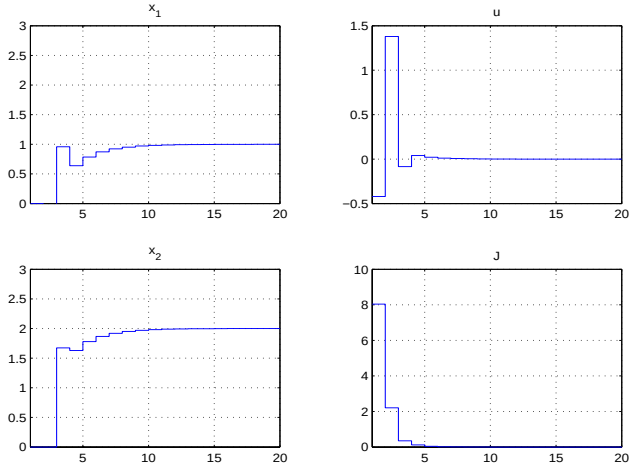


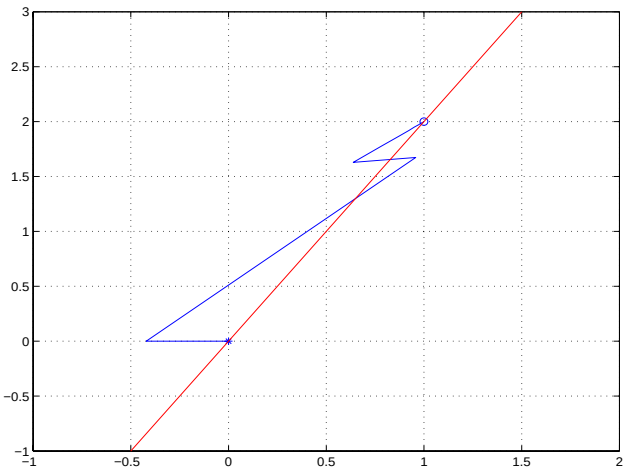
The stability is an issue !

Test 1 : $N = 2,\ M = 2,\ r = 1$

The stability is an issue !

Test 1 : $N = 2,\ M = 2,\ r = 1$





The stability is an issue !

Let us assume that one looks for solution such that $u \leq 0.1$

Assume that for that reason one takes

$$r = 160$$

$N = 2,\ M = 2,\ r = 160$

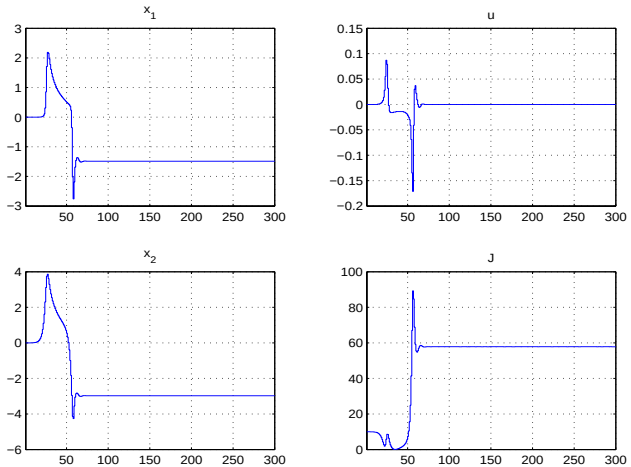
The stability is an issue !

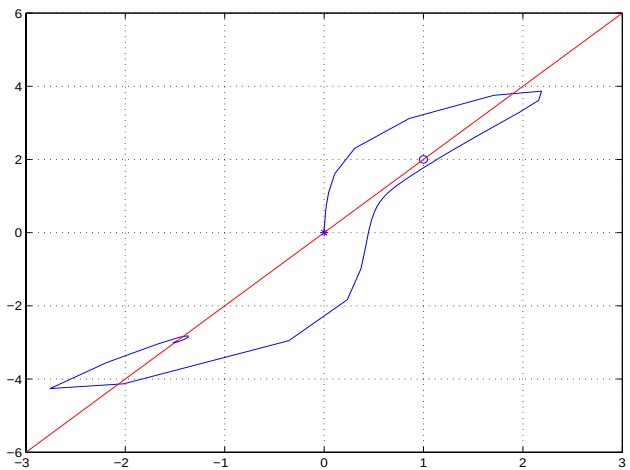
Let us assume that one looks for solution such that $u \leq 0.1$

Assume that for that reason one takes

$$r = 160$$

$N = 2,\ M = 2,\ r = 160$





gipsa-lab

The stability is an issue !

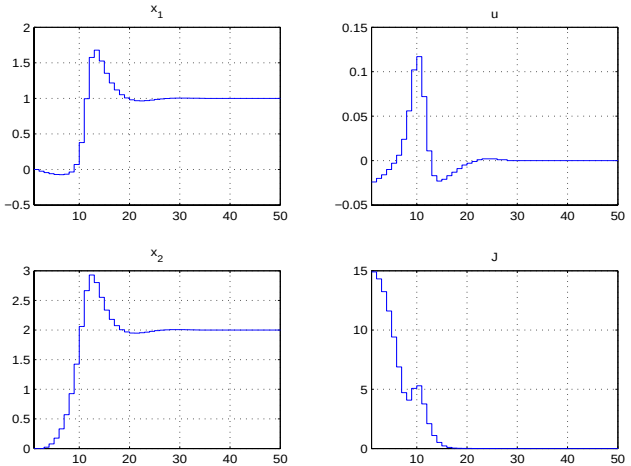
Let us take $M = 3$ (instead of 2)

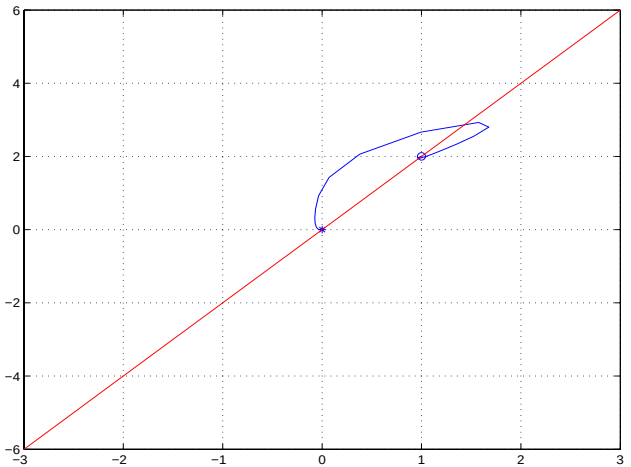
$N = 2$, $M = 3$, $r = 160$

The stability is an issue !

Let us take $M = 3$ (instead of 2)

$N = 2$, $M = 3$, $r = 160$





The stability is an issue !

Increasing M enabled the target state to be reached.

However, the control is again above 0.1,

So let us increase r again by taking

$$r = 500$$

$N = 2$, $M = 3$, $r = 500$

The stability is an issue !

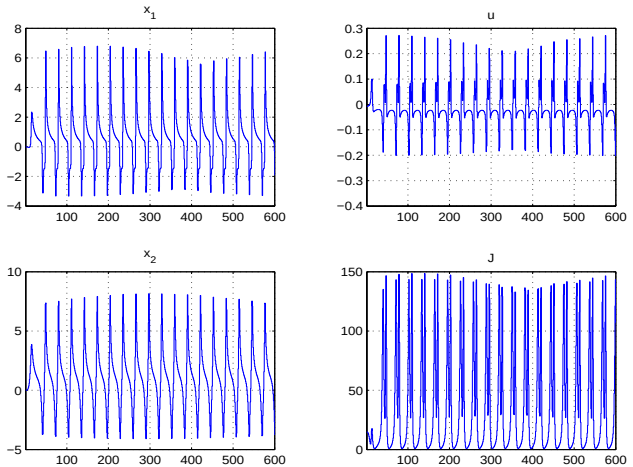
Increasing M enabled the target state to be reached.

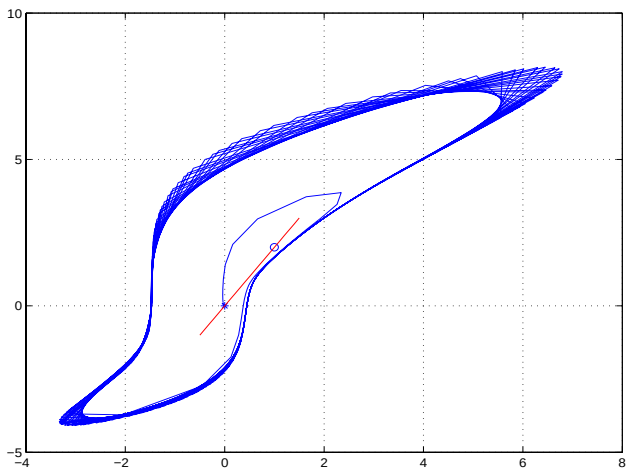
However, the control is again above 0.1,

So let us increase r again by taking

$$r = 500$$

$N = 2,\ M = 3,\ r = 500$





The stability is an issue !

Let us explicitly impose the constraint

$$\mathbb{U} = [-0.1, 0.1]$$

and test it with the configuration

$$r = 1 \quad ; \quad N = M = 3$$

$N = 3$, $M = 3$, $r = 1$

explicit constraint $|u| \leq 0.1$

The stability is an issue !

Let us explicitly impose the constraint

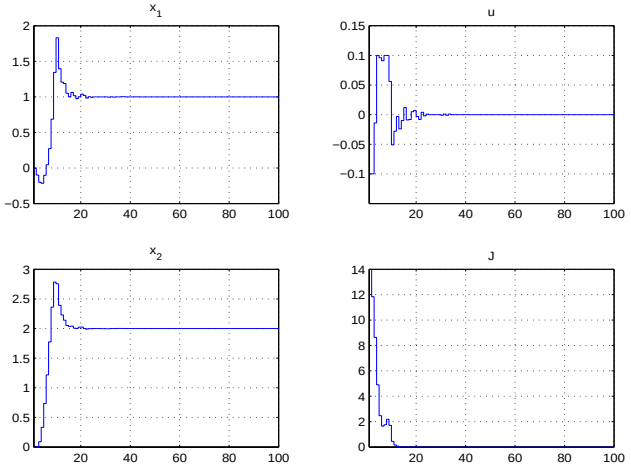
$$\mathbb{U} = [-0.1, 0.1]$$

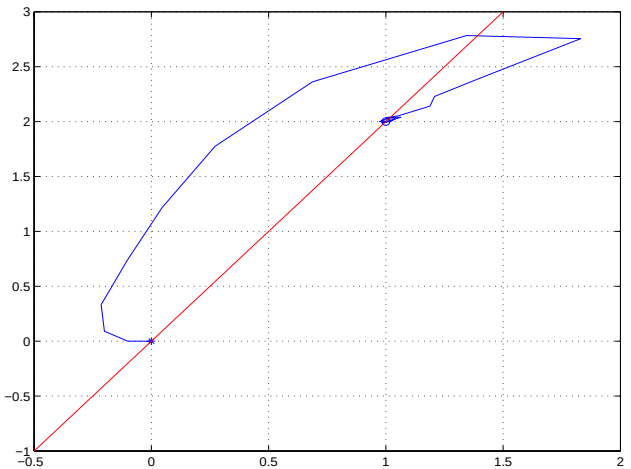
and test it with the configuration

$$r = 1 \quad ; \quad N = M = 3$$

$N = 3$, $M = 3$, $r = 1$

explicit constraint $|u| \leq 0.1$





gipsa-lab

The example shows that

- Nonlinear Model Predictive Control is a "*generic*" solution.
  (Who remember the system's equations ? ! !)

The example shows that

- Nonlinear Model Predictive Control is a "*generic*" solution.
  (Who remember the system's equations ? ! !)
- Easy handling of constraints

The example shows that

- Nonlinear Model Predictive Control is a "*generic*" solution.
  (Who remember the system's equations ? ! !)
- Easy handling of constraints

- The stability depends on the choice of
  - The cost function
  - The constraints
  - The control parametrization

The example shows that

- Nonlinear Model Predictive Control is a "*generic*" solution.
  (Who remember the system's equations ? ! !)
- Easy handling of constraints

- The stability depends on the choice of
  - The cost function
  - The constraints
  - The control parametrization

- For a detailed study of the stability conditions

  Mayne,D.Q., Rawlings,J.B., Rao,C.V., et al , Constrained model predictive control : stability and optimality, Automatica, 2000, Vol : 36, Pages : 789 - 814

The concept of distributed-on-time optimization

NMPC

$$K := u^{(1)} \circ \hat{p} : \mathbb{R}^n \rightarrow \mathbb{U}$$

$$\hat{p}(x(t_j)) := \arg\min_{p \in \mathbb{P}} \Big[J(x(t_j), p) \Big] \quad \text{under} \quad C(x(t_j), p) \leq 0$$

Real-Time Implementation Dilemma

The computation of $\hat{p}(x(t_k))$ is done using some iterations

$$\begin{aligned} p^{(i+1)} &= \mathcal{S}(p^{(i)}, x(t_k)) \\ p^{(0)} &= p_0 \in \mathbb{P} \end{aligned}$$

Real-Time Implementation Dilemma

The computation of $\hat{p}(x(t_k))$ is done using some iterations

$$\begin{aligned} p^{(i+1)} &= \mathcal{S}(p^{(i)}, x(t_k)) \\ p^{(0)} &= p_0 \in \mathbb{P} \end{aligned}$$

More precisely

$$\hat{p}(x(t_k)) = \lim_{i \to \infty} p^{(i)}$$

Real-Time Implementation Dilemma

The computation of $\hat{p}(x(t_k))$ is done using some iterations

$$\begin{aligned} p^{(i+1)} &= \mathcal{S}(p^{(i)}, x(t_k)) \\ p^{(0)} &= p_0 \in \mathbb{P} \end{aligned}$$

More precisely

$$\hat{p}(x(t_k)) = \lim_{i \to \infty} p^{(i)}$$

or using some stopping $\varepsilon > 0$:

$$\hat{p}(x(t_k)) = p^{(\imath_\varepsilon(x(t_k)))}$$

where $\imath_\varepsilon(x(t_k)) \in \mathbb{N}$ is the number of iterations that generally depend on $x(t_k)$.

Real-Time Implementation Dilemma

The computation of $\hat{p}(x(t_k))$ is done using some iterations

$$\begin{aligned} p^{(i+1)} &= \mathcal{S}(p^{(i)}, x(t_k)) \\ p^{(0)} &= p_0 \in \mathbb{P} \end{aligned}$$

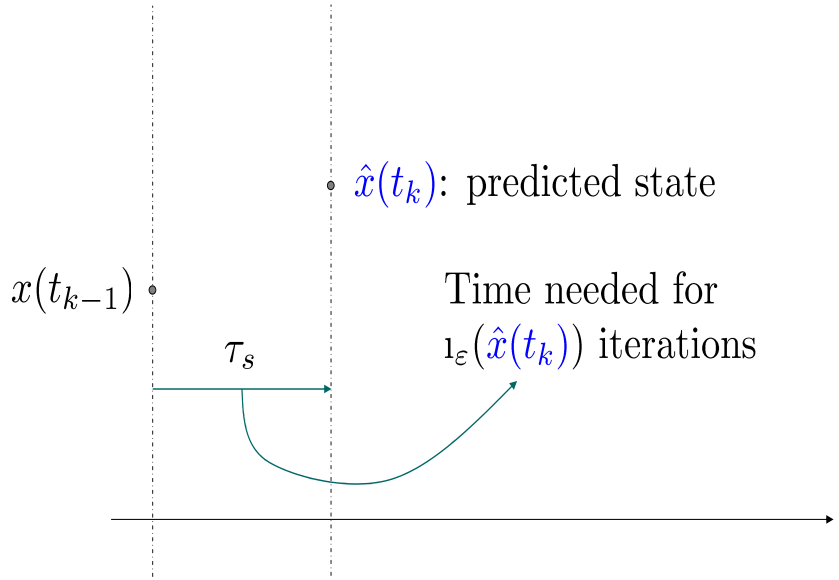
More precisely

$$\hat{p}(x(t_k)) = \lim_{i \to \infty} p^{(i)}$$

or using some stopping $\varepsilon > 0$:

$$\hat{p}(x(t_k)) = p^{(\imath_\varepsilon(x(t_k)))}$$

where $\imath_\varepsilon(x(t_k)) \in \mathbb{N}$ is the number of iterations that generally depend on $x(t_k)$.



Real-Time Implementation Dilemma

The computation of $\hat{p}(x(t_k))$ is done using some iterations

$$\begin{aligned} p^{(i+1)} &= \mathcal{S}(p^{(i)}, x(t_k)) \\ p^{(0)} &= p_0 \in \mathbb{P} \end{aligned}$$

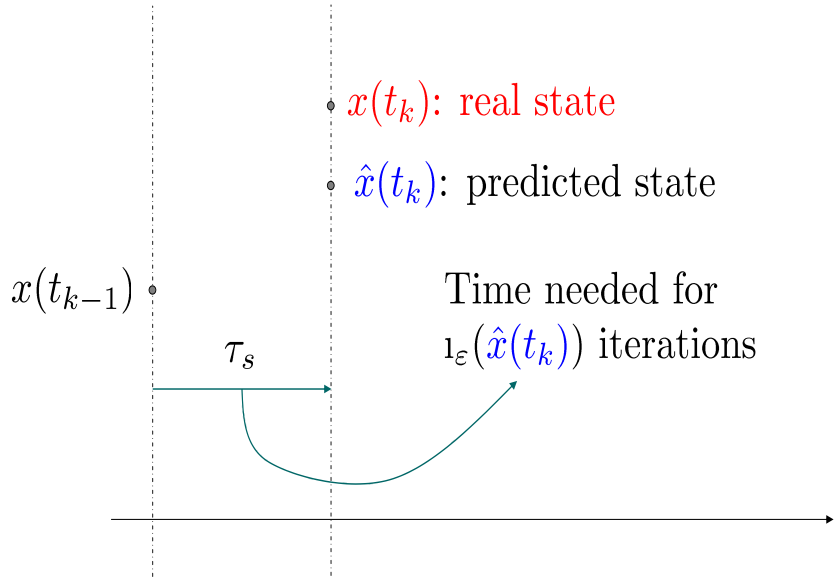
More precisely

$$\hat{p}(x(t_k)) = \lim_{i \to \infty} p^{(i)}$$

or using some stopping $\varepsilon > 0$:

$$\hat{p}(x(t_k)) = p^{(\imath_\varepsilon(x(t_k)))}$$

where $\imath_\varepsilon(x(t_k)) \in \mathbb{N}$ is the number of iterations that generally depend on $x(t_k)$.



gipsa-lab

Real-Time Implementation Dilemma

The computation of $\hat{p}(x(t_k))$ is done using some iterations

$$\begin{aligned} p^{(i+1)} &= \mathcal{S}(p^{(i)}, x(t_k)) \\ p^{(0)} &= p_0 \in \mathbb{P} \end{aligned}$$

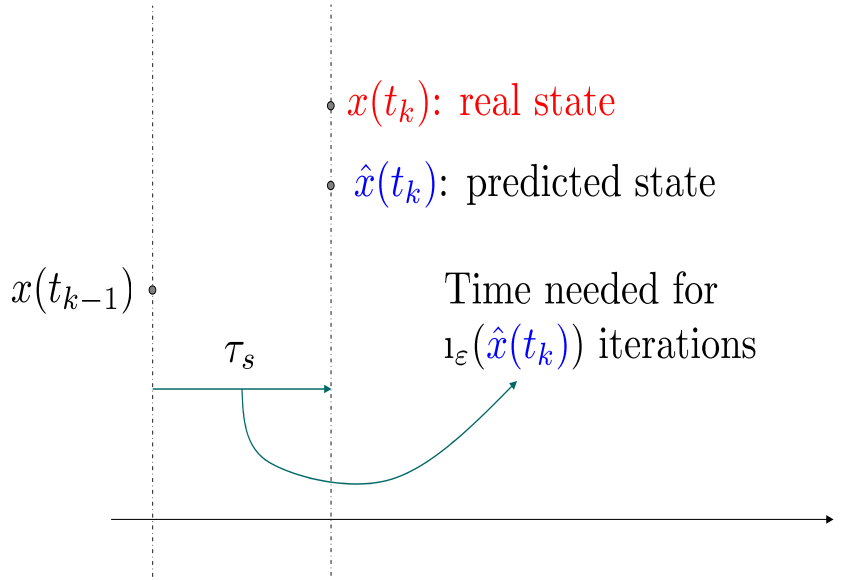
More precisely

$$\hat{p}(x(t_k)) = \lim_{i \to \infty} p^{(i)}$$

or using some stopping $\varepsilon > 0$:

$$\hat{p}(x(t_k)) = p^{(\imath_\varepsilon(x(t_k)))}$$

where $\imath_\varepsilon(x(t_k)) \in \mathbb{N}$ is the number of iterations that generally depend on $x(t_k)$.



- $\tau_s \geq \imath_\varepsilon(x(t_k)) \times \tau_{\text{iteraton}}$

Real-Time Implementation Dilemma

The computation of $\hat{p}(x(t_k))$ is done using some iterations

$$\begin{aligned} p^{(i+1)} &= \mathcal{S}(p^{(i)}, x(t_k)) \\ p^{(0)} &= p_0 \in \mathbb{P} \end{aligned}$$

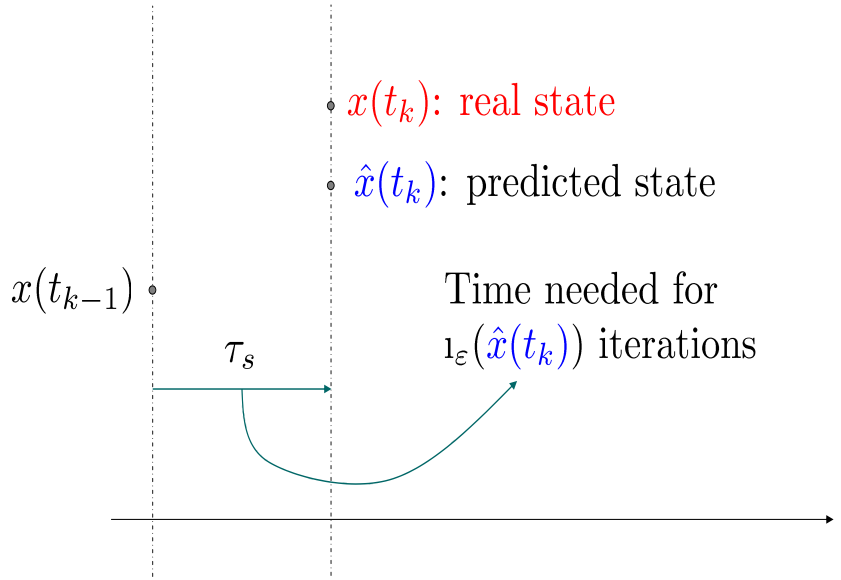
More precisely

$$\hat{p}(x(t_k)) = \lim_{i \to \infty} p^{(i)}$$

or using some stopping $\varepsilon > 0$:

$$\hat{p}(x(t_k)) = p^{(\imath_\varepsilon(x(t_k)))}$$

where $\imath_\varepsilon(x(t_k)) \in \mathbb{N}$ is the number of iterations that generally depend on $x(t_k)$.



- $\tau_s \geq \imath_\varepsilon(x(t_k)) \times \tau_{\text{iteraton}}$
- $\|x(t_k) - \hat{x}(t_k)\|$ increase with τ_s due to model uncertainties & disturbances

A more realistic approach

- Choose a sampling period τ_s

A more realistic approach

- Choose a sampling period τ_s
- Deduce a maximum number of iterations $q \in \mathbb{N}$ [compatible with τ_s]

A more realistic approach

- Choose a sampling period τ_s
- Deduce a maximum number of iterations $q \in \mathbb{N}$ [compatible with τ_s]
- Distribute the optimization over the system real life-time :

$$\begin{aligned} x(t_{k+1}) &= X(\tau_s, x(t_k), u^1(p(t_k))) \\ p(t_{k+1}) &= \mathcal{S}^{(q)}(p^+(t_k), \hat{x}(t_{k+1})) \end{aligned}$$

A more realistic approach

- Choose a sampling period τ_s
- Deduce a maximum number of iterations $q \in \mathbb{N}$ [compatible with τ_s]
- Distribute the optimization over the system real life-time :

$$\begin{aligned} x(t_{k+1}) &= X(\tau_s, x(t_k), u^1(p(t_k))) \\ p(t_{k+1}) &= \mathcal{S}^{(q)}(p^+(t_k), \hat{x}(t_{k+1})) \end{aligned}$$

- p^+ is defined by the translatability of the control parametrization.

A more realistic approach

- Choose a sampling period τ_s
- Deduce a maximum number of iterations $q \in \mathbb{N}$ [compatible with τ_s]
- Distribute the optimization over the system real life-time :

$$\begin{aligned} x(t_{k+1}) &= X(\tau_s, x(t_k), u^1(p(t_k))) \\ p(t_{k+1}) &= \mathcal{S}^{(q)}(p^+(t_k), \hat{x}(t_{k+1})) \end{aligned}$$

- p^+ is defined by the translatability of the control parametrization.
- $\hat{x}(t_{k+1})$ is the predicted state.

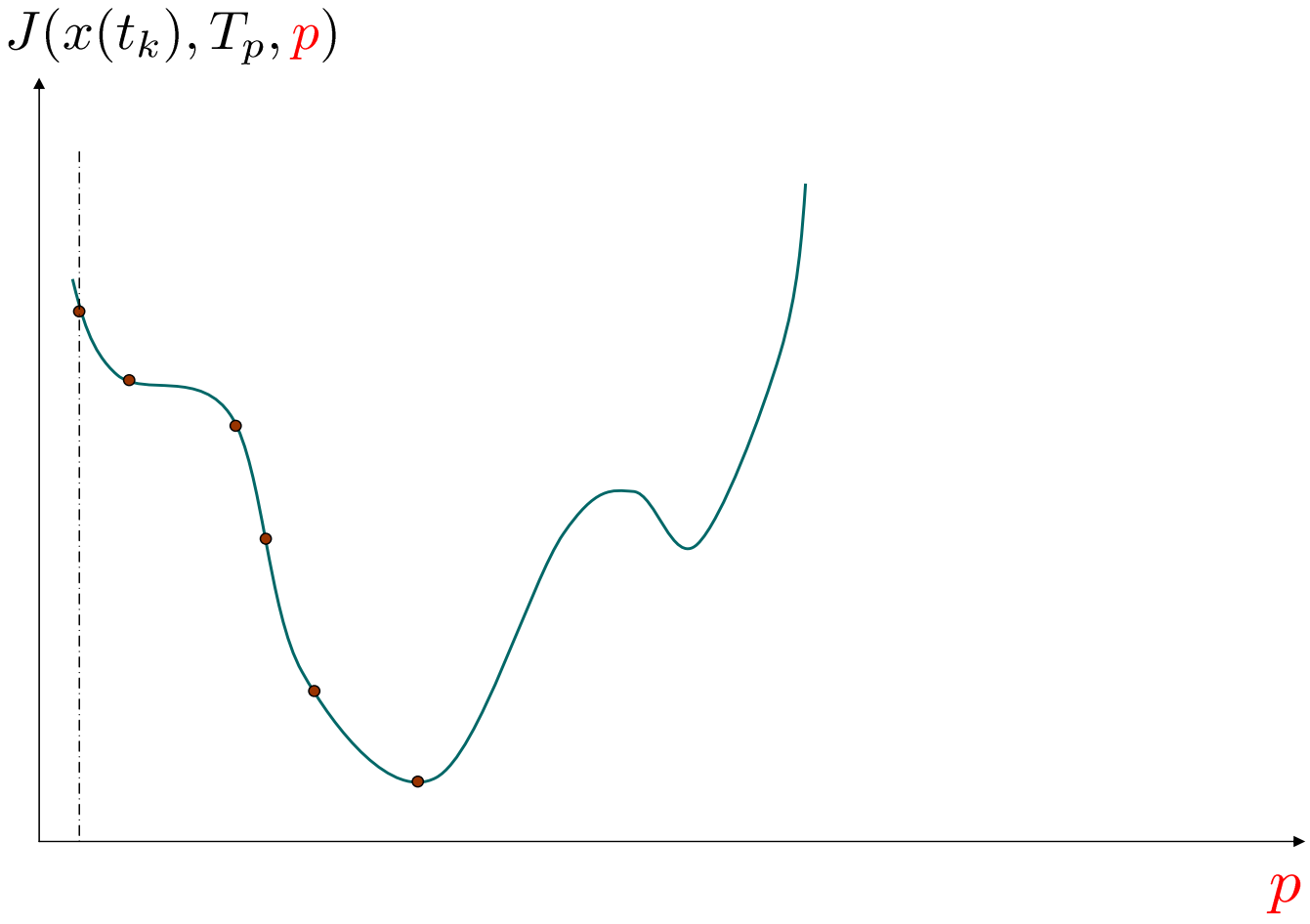
A more realistic approach

- Choose a sampling period τ_s
- Deduce a maximum number of iterations $q \in \mathbb{N}$ [compatible with τ_s]
- Distribute the optimization over the system real life-time :

$$\begin{aligned} x(t_{k+1}) &= X(\tau_s, x(t_k), u^1(p(t_k))) \\ p(t_{k+1}) &= \mathcal{S}^{(q)}(p^+(t_k), \hat{x}(t_{k+1})) \end{aligned}$$

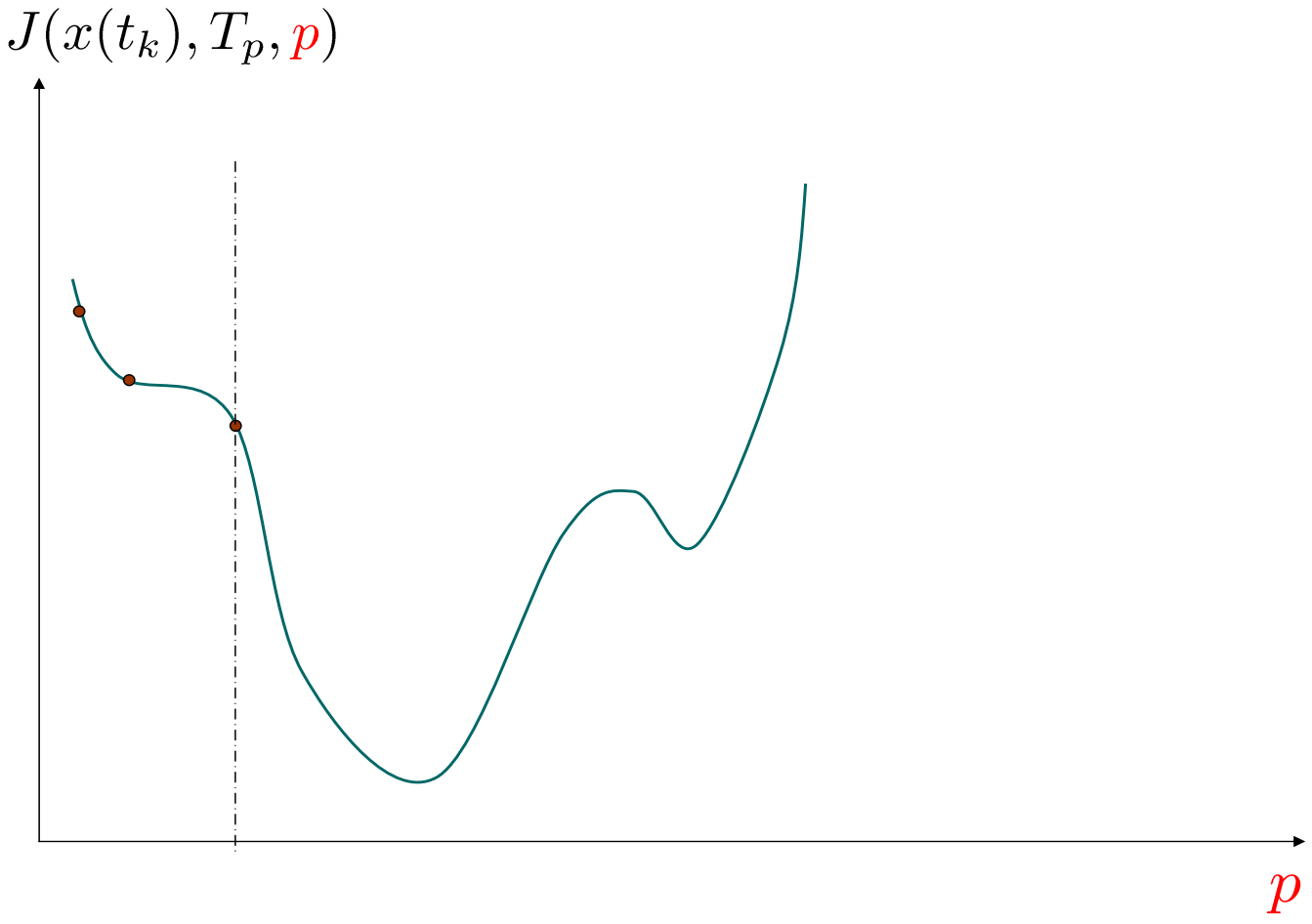
- p^+ is defined by the translatability of the control parametrization.
- $\hat{x}(t_{k+1})$ is the predicted state.
- Stability is a more involved issue than in classical formulations

Distributing the optimization over the system real-life time



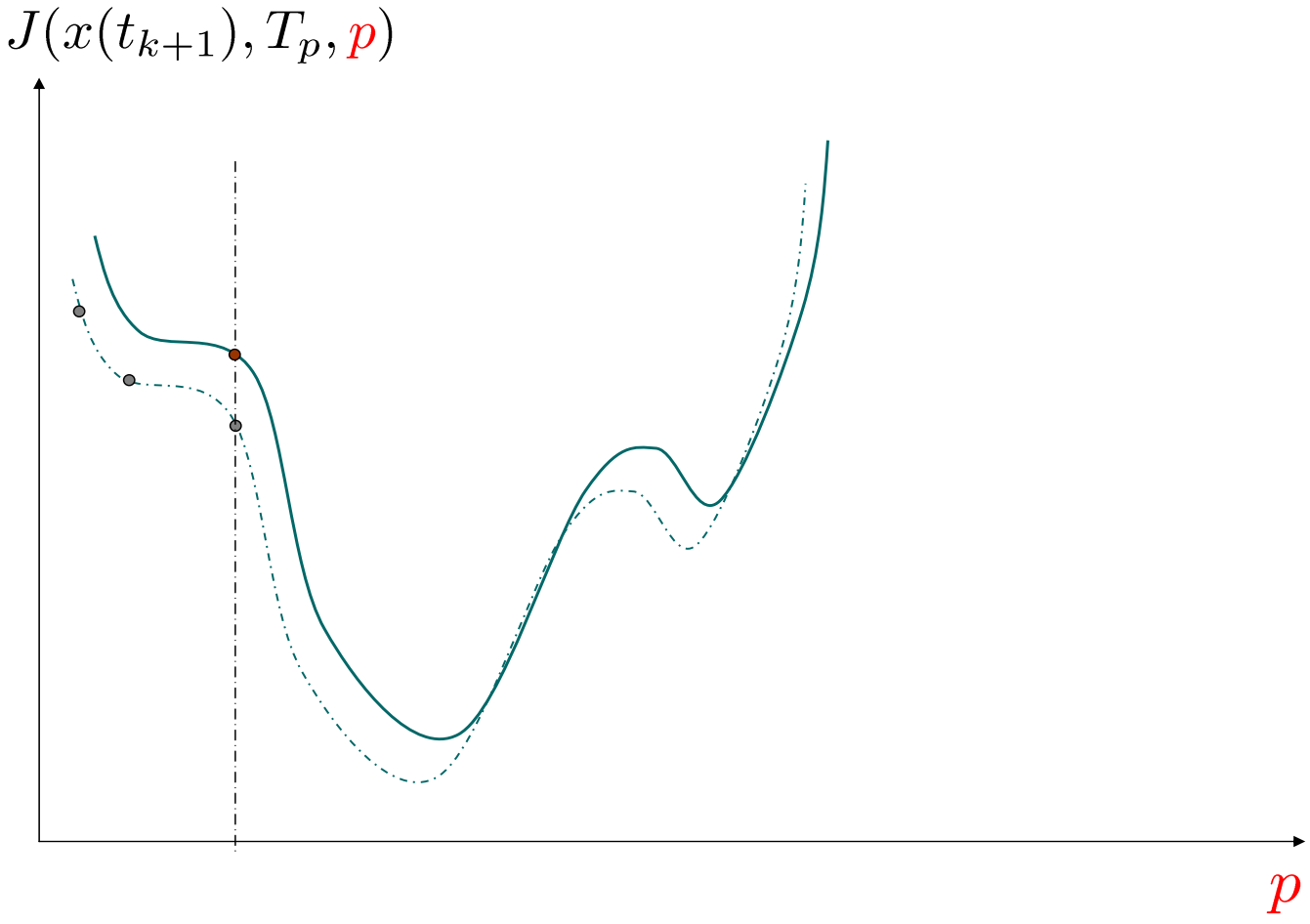
Case $q = 2$

Distributing the optimization over the system real-life time



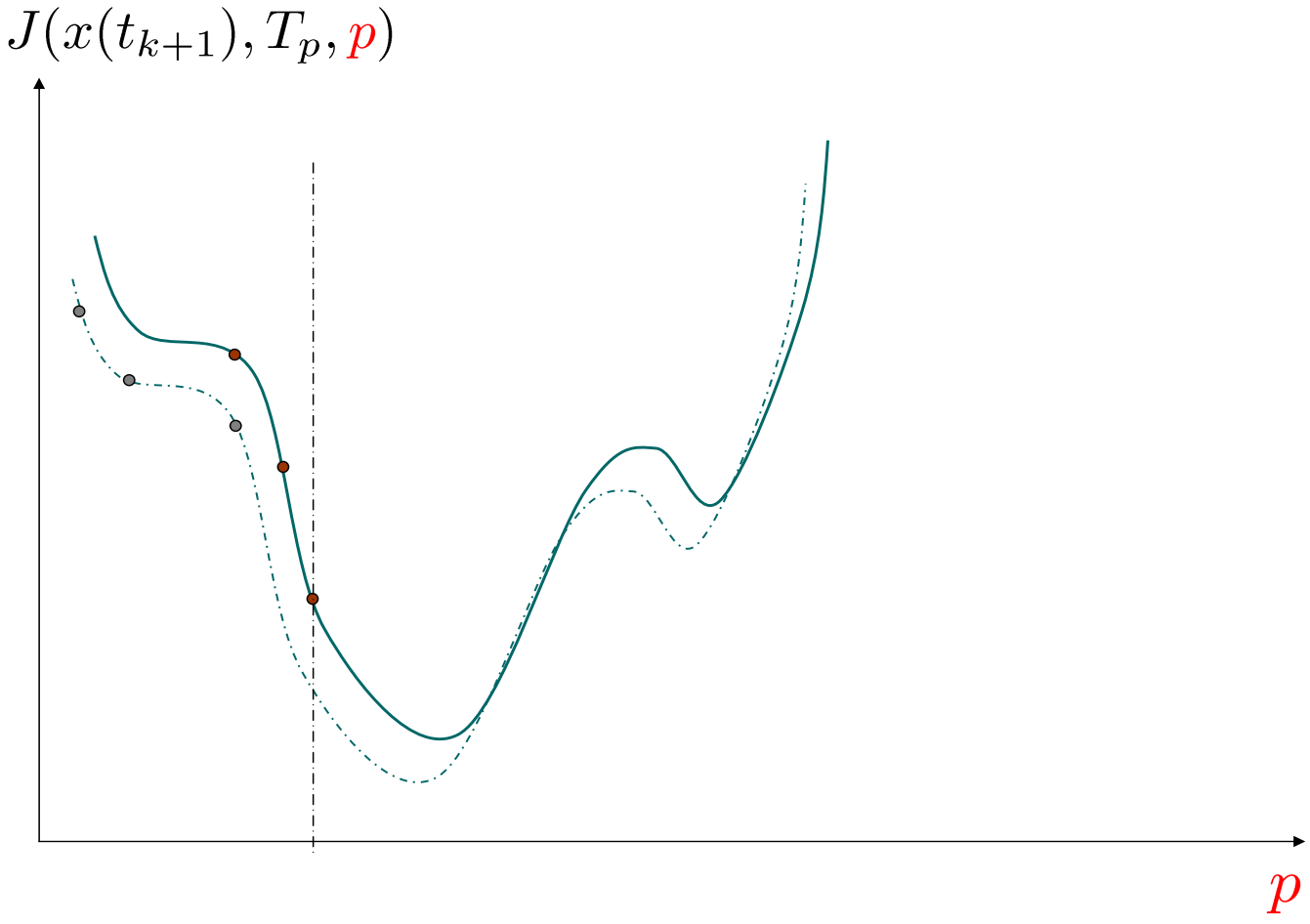
Case $q = 2$

Distributing the optimization over the system real-life time



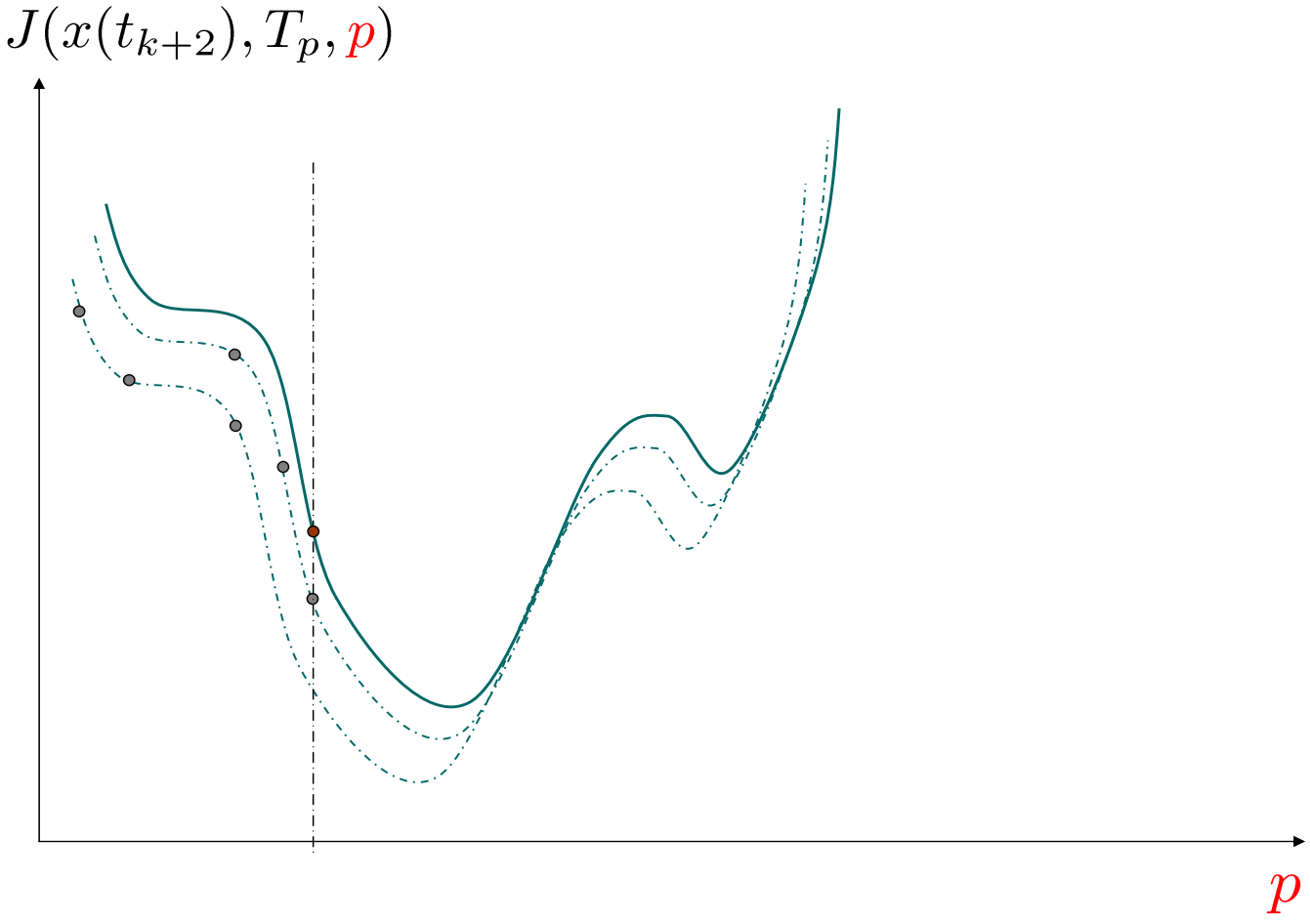
Case $q = 2$

Distributing the optimization over the system real-life time



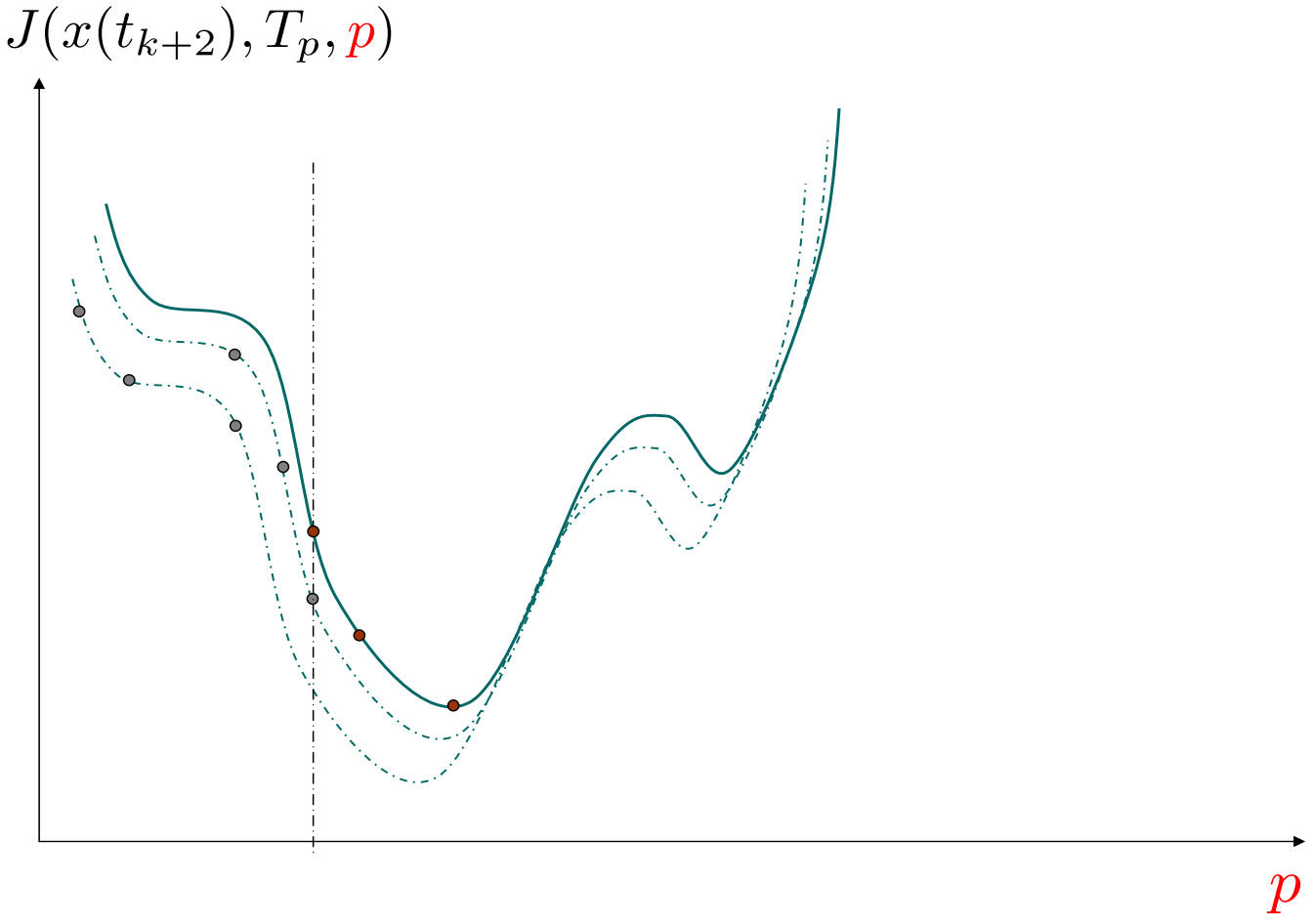
Case $q = 2$

Distributing the optimization over the system real-life time



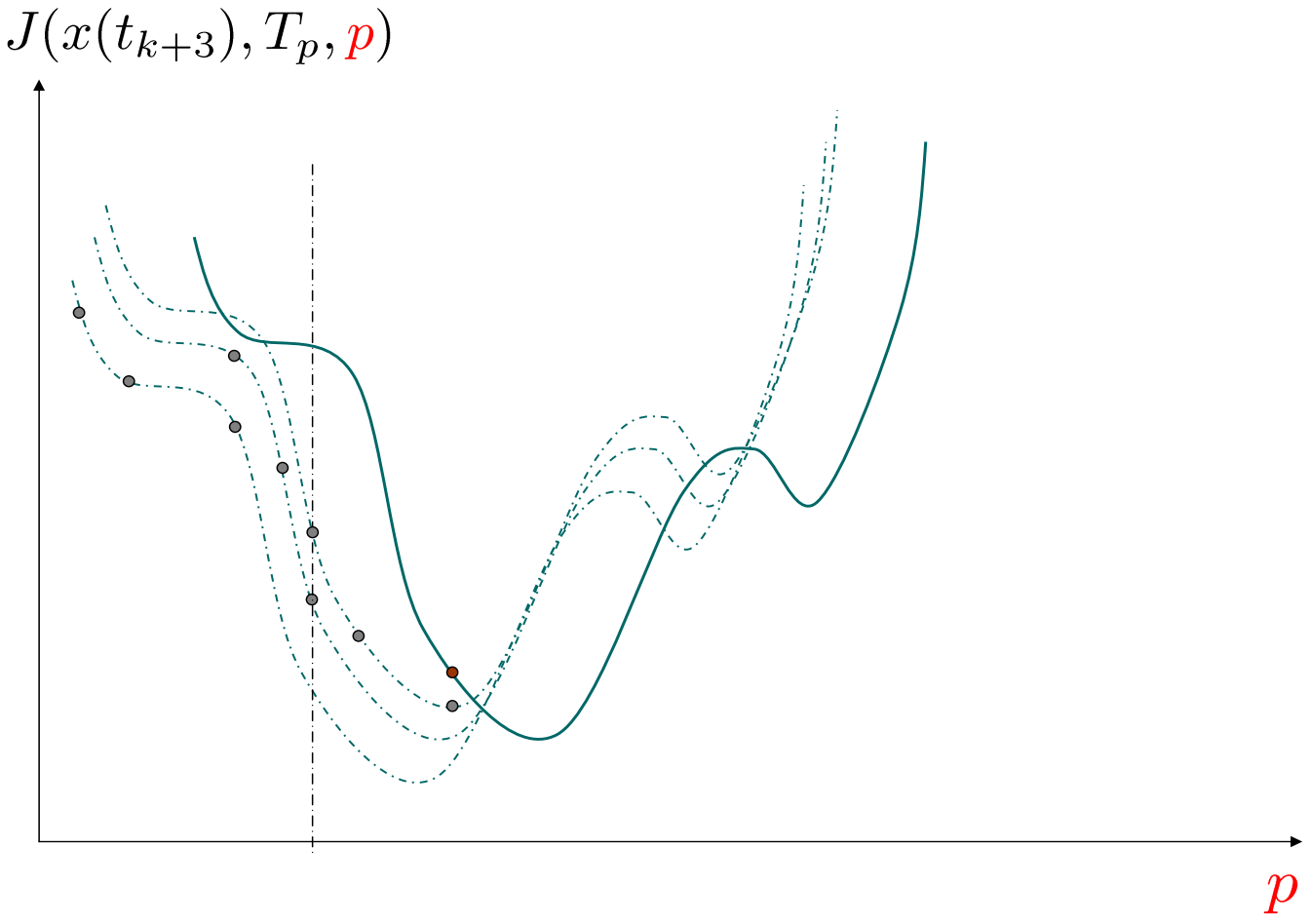
Case $q = 2$

Distributing the optimization over the system real-life time



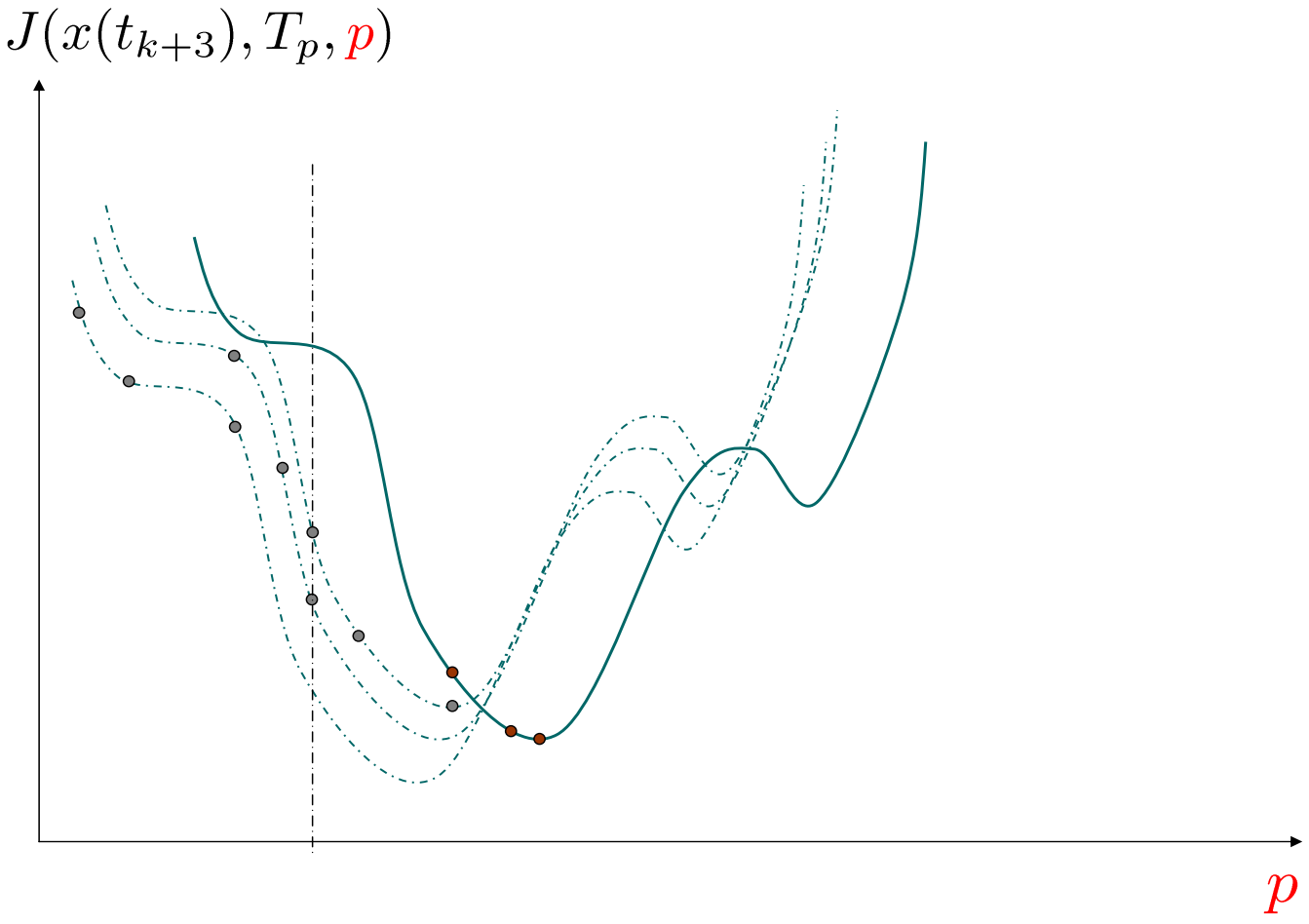
Case $q = 2$

Distributing the optimization over the system real-life time



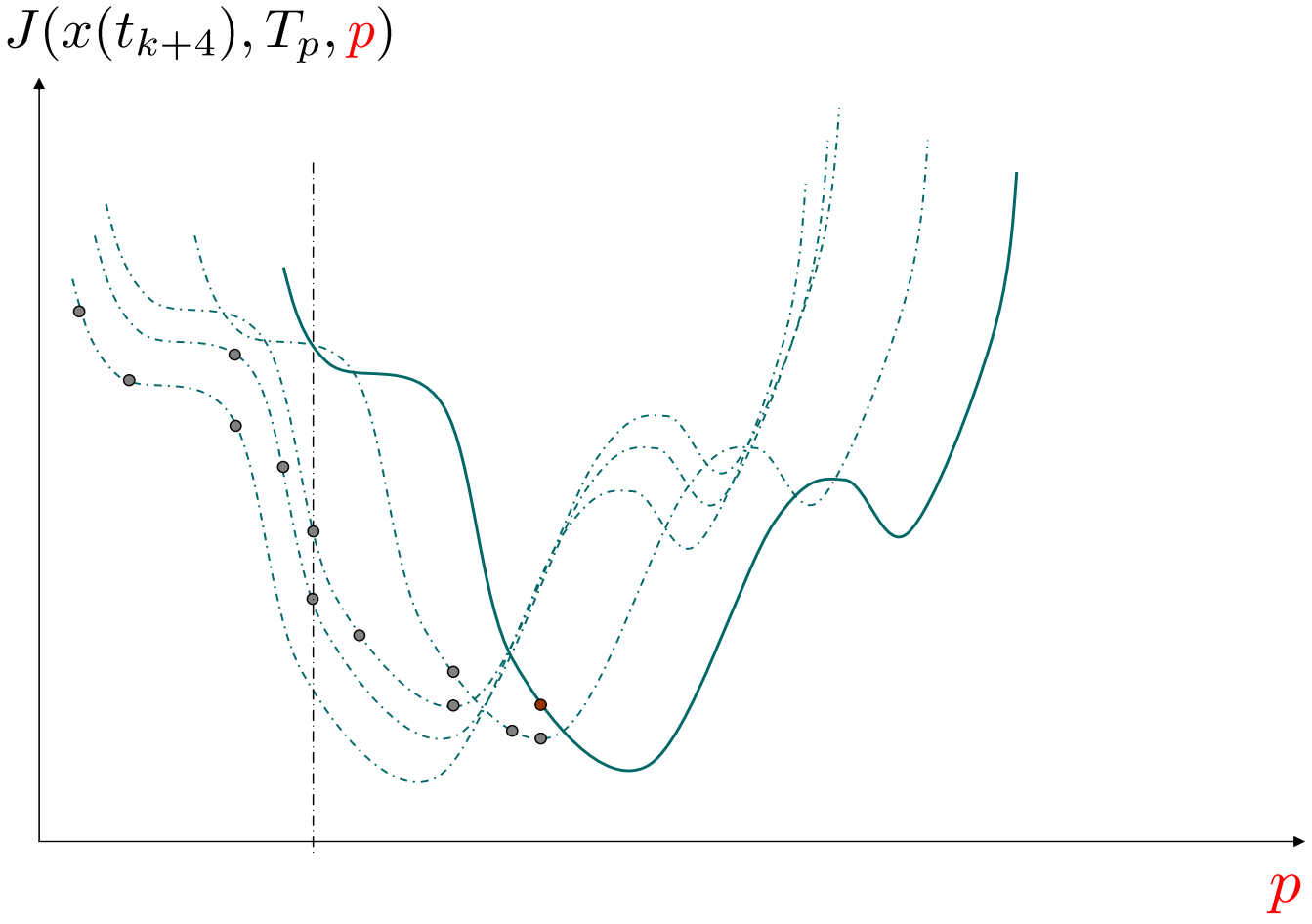
Case $q = 2$

Distributing the optimization over the system real-life time



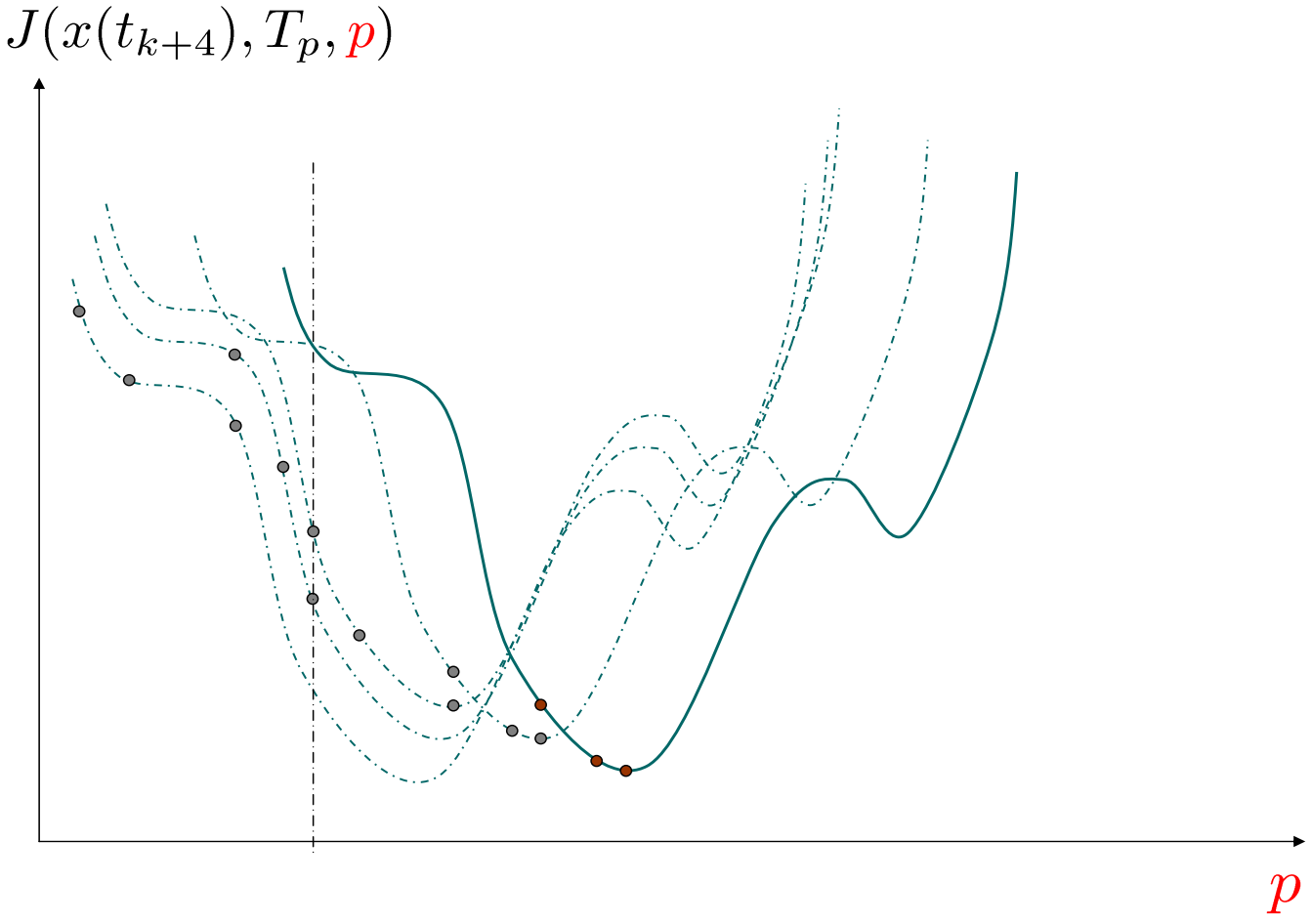
Case $q = 2$

Distributing the optimization over the system real-life time



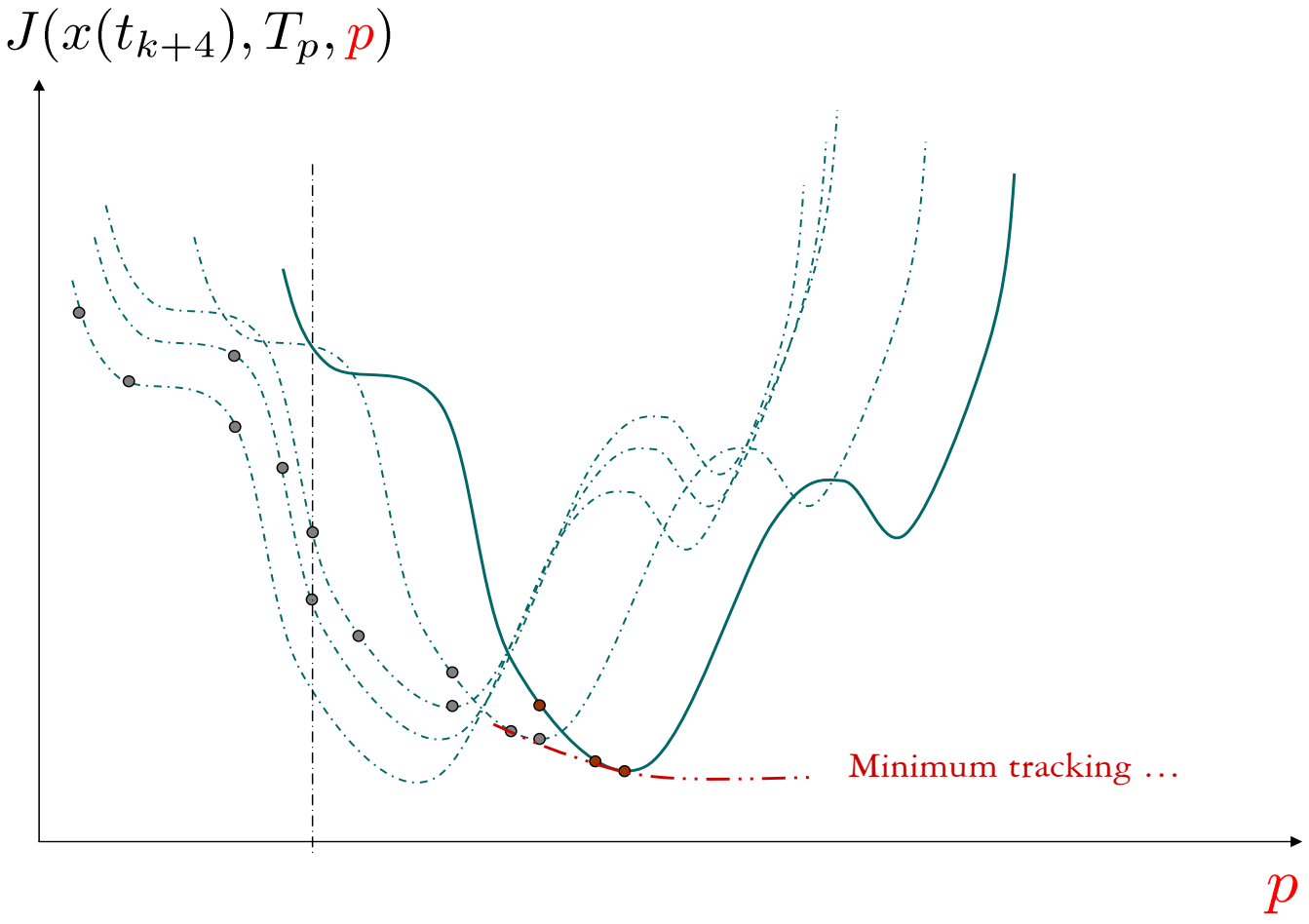
Case $q = 2$

Distributing the optimization over the system real-life time



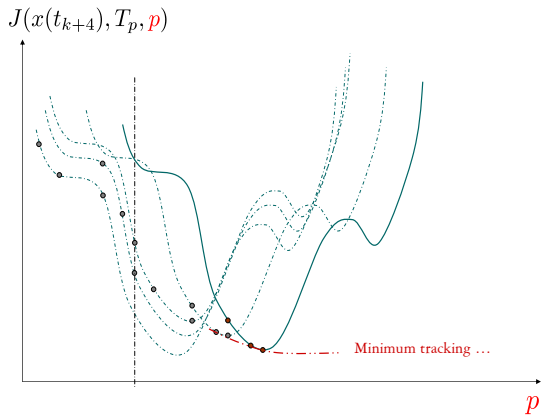
Case $q = 2$

Distributing the optimization over the system real-life time



Case $q = 2$

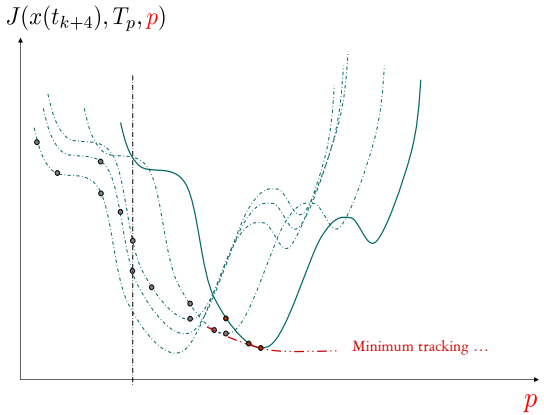
Distributing the optimization over the system real-life time



This may work provided that

1. $\tau_s \ll$ characteristic time of the system
2. The iteration is efficient
3. No finite escape time in the transient phase
4. The original RH formulation is stabilizing

Distributing the optimization over the system real-life time



[M.Alamir, A Framework for updating control period monitoring in real-time NMPC schemes International Workshop on Assessment and Future Directions of Nonlinear Model Predictive Control, Pavia, Italy, (2008).]

This may work provided that

1. $\tau_s \ll$ characteristic time of the system
2. The iteration is efficient
3. No finite escape time in the transient phase
4. The original RH formulation is stabilizing

gipsa-lab

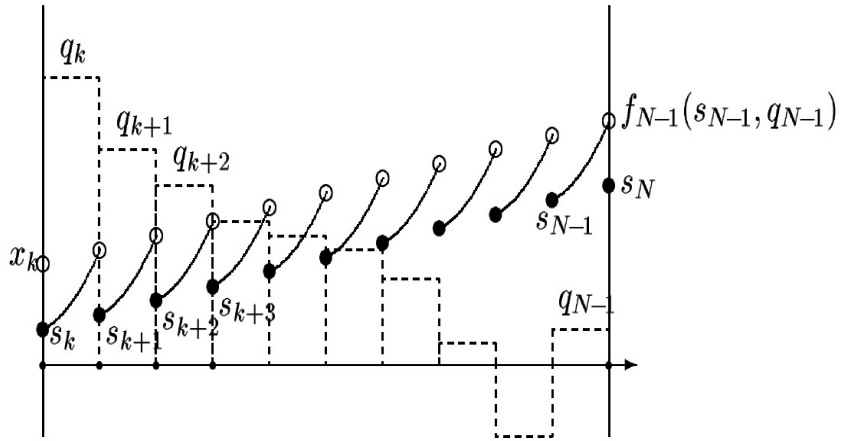
Main Alternatives

The iterative process $\mathcal{S}$

$$\begin{aligned} p^{(i+1)} &= \mathcal{S}(p^{(i)}, x(t_k)) \\ p^{(0)} &= p_0 \in \mathbb{P} \end{aligned}$$

Multiple shooting approach	[Diehl et al. 2005]
Continuation approach	[Ohtsuka 2004]
Parametric approach	[Alamir 2006]

The multiple shooting approach [Diehl et al. 2005]



Unknown

$$y = (\lambda_k, s_k, u_k, \ldots, \lambda_{k+N}, s_{k+N}, u_{k+N})^T$$

Constraints

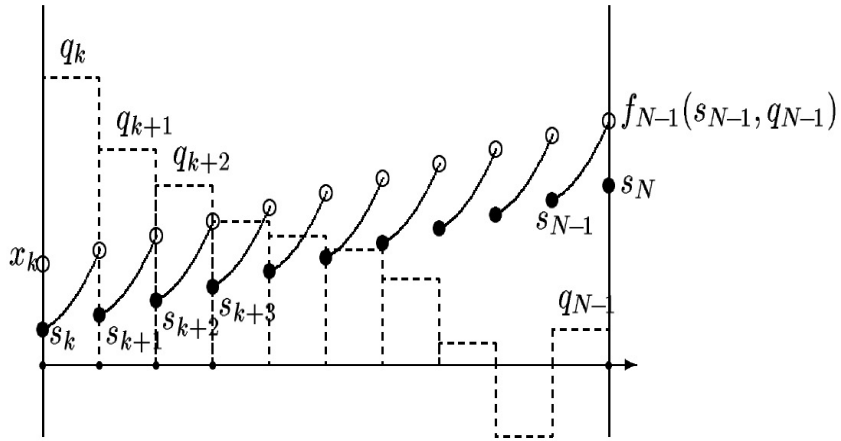
$$x_k - s_k = 0$$

$$f(s_i, u_i) - s_{i+1} = 0$$

The multiple shooting approach [Diehl et al. 2005]

Karush-Kuhn-Tucker optimality condition :

$$\nabla_y \mathcal{L}^k(y) = 0$$



Unknown

$$y = (\lambda_k, s_k, u_k, \ldots, \lambda_{k+N}, s_{k+N}, u_{k+N})^T$$

Constraints

$$x_k - s_k = 0$$

$$f(s_i, u_i) - s_{i+1} = 0$$

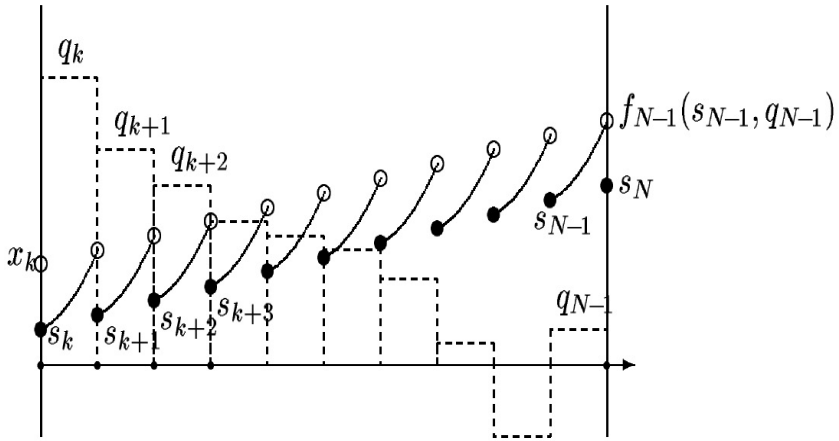
The multiple shooting approach [Diehl et al. 2005]

Karush-Kuhn-Tucker optimality condition :

$$\nabla_y \mathcal{L}^k(y) = 0$$

At each instant

$$\nabla_y \mathcal{L}^k(y) + J^k(y) \cdot \Delta y = 0$$



Unknown

$$y = (\lambda_k, s_k, u_k, \ldots, \lambda_{k+N}, s_{k+N}, u_{k+N})^T$$

Constraints

$$x_k - s_k = 0$$

$$f(s_i, u_i) - s_{i+1} = 0$$

The multiple shooting approach [Diehl et al. 2005]

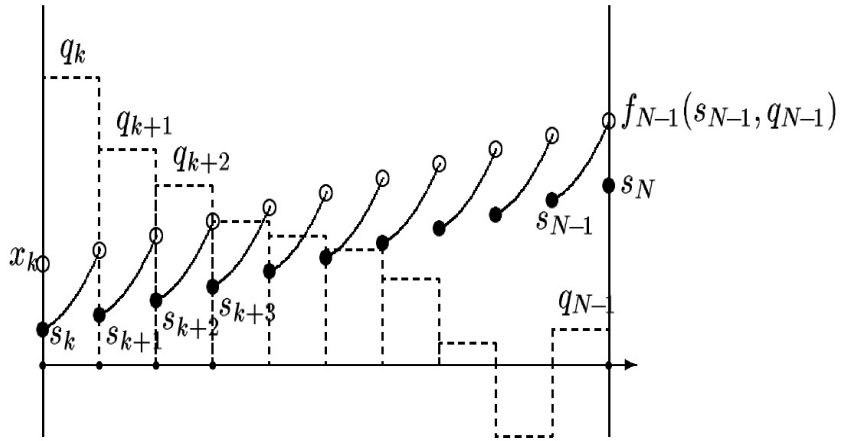
Karush-Kuhn-Tucker optimality condition :

$$\nabla_y \mathcal{L}^k(y) = 0$$

At each instant

$$\nabla_y \mathcal{L}^k(y) + J^k(y) \cdot \Delta y = 0$$

- $J^k(y)$ independent of $x(k)$



Unknown

$$y = (\lambda_k, s_k, u_k, \ldots, \lambda_{k+N}, s_{k+N}, u_{k+N})^T$$

Constraints

$$x_k - s_k = 0$$

$$f(s_i, u_i) - s_{i+1} = 0$$

The multiple shooting approach

[Diehl et al. 2005]

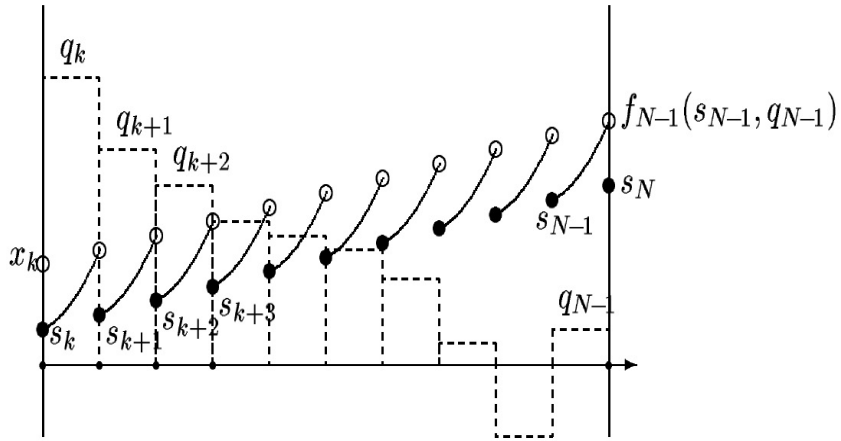
Karush-Kuhn-Tucker optimality condition :

$$\nabla_y \mathcal{L}^k(y) = 0$$

At each instant

$$\nabla_y \mathcal{L}^k(y) + J^k(y) \cdot \Delta y = 0$$

- $J^k(y)$ independent of $x(k)$
- transient violation of feasibility



Unknown

$$y = (\lambda_k, s_k, u_k, \ldots, \lambda_{k+N}, s_{k+N}, u_{k+N})^T$$

Constraints

$$x_k - s_k = 0$$

$$f(s_i, u_i) - s_{i+1} = 0$$

The multiple shooting approach

[Diehl et al. 2005]

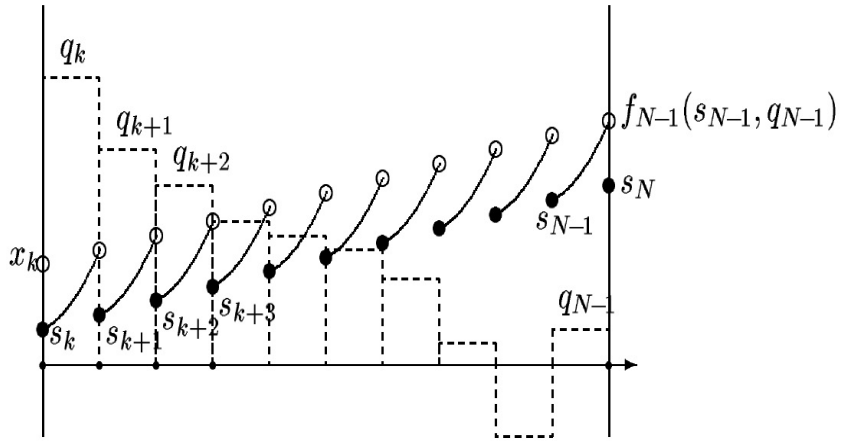
Karush-Kuhn-Tucker optimality condition :

$$\nabla_y \mathcal{L}^k(y) = 0$$

At each instant

$$\nabla_y \mathcal{L}^k(y) + J^k(y) \cdot \Delta y = 0$$

- $J^k(y)$ independent of $x(k)$
- transient violation of feasibility
- less risk of instability



Unknown

$$y = (\lambda_k, s_k, u_k, \ldots, \lambda_{k+N}, s_{k+N}, u_{k+N})^T$$

Constraints

$$x_k - s_k = 0$$

$$f(s_i, u_i) - s_{i+1} = 0$$

The multiple shooting approach

[Diehl et al. 2005]

Karush-Kuhn-Tucker optimality condition :

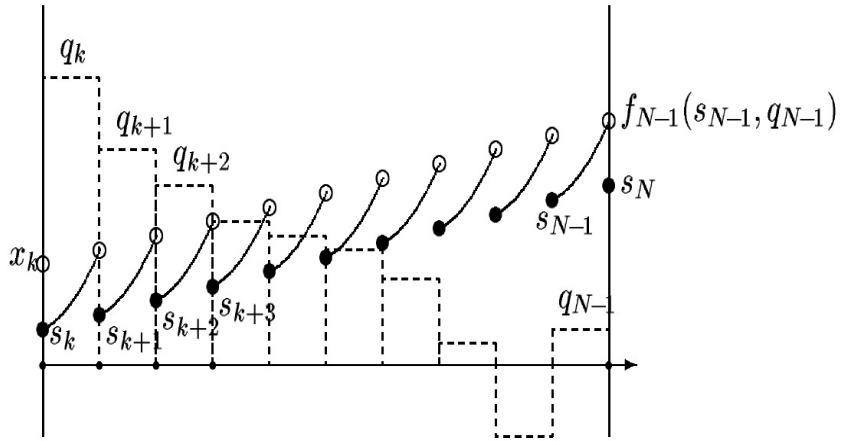
$$\nabla_y \mathcal{L}^k(y) = 0$$

At each instant

$$\nabla_y \mathcal{L}^k(y) + J^k(y) \cdot \Delta y = 0$$

- $J^k(y)$ independent of $x(k)$
- transient violation of feasibility
- less risk of instability

Available software MUSCODII



Unknown

$$y = (\lambda_k, s_k, u_k, \ldots, \lambda_{k+N}, s_{k+N}, u_{k+N})^T$$

Constraints

$$x_k - s_k = 0$$

$$f(s_i, u_i) - s_{i+1} = 0$$

gipsa-lab

The continuation approach

[Ohtsuka et al. 2004]

Hamiltonian system

$$\dot{x} = f(x, u)$$
$$\dot{\lambda} = -H_x(x, u, \lambda, \mu)$$

Hamiltonian

$$H(x, \lambda, \mu, u) =$$
$$L(x, u) + \lambda^T f(x, u) + \mu^T C(x, u)$$

Unknown

$$U := \left(u_0^T, \mu_0^T, \ldots, u_{N-1}^T, \mu_{N-1}^T\right)^T$$

The continuation approach

[Ohtsuka et al. 2004]

Optimality condition

$$F(U(t), x(t)) = 0$$

Hamiltonian system

$$\dot{x} = f(x, u)$$
$$\dot{\lambda} = -H_x(x, u, \lambda, \mu)$$

Hamiltonian

$$H(x, \lambda, \mu, u) =$$
$$L(x, u) + \lambda^T f(x, u) + \mu^T C(x, u)$$

Unknown

$$U := \left(u_0^T, \mu_0^T, \ldots, u_{N-1}^T, \mu_{N-1}^T\right)^T$$

The continuation approach

[Ohtsuka et al. 2004]

Optimality condition

$$F(U(t), x(t)) = 0$$

Impose the dynamic

$$\dot{F}(U, x) = A_s \cdot F(U, x)$$

By imposing

$$\dot{U} = F_U^{-1}\left(A_s F - F_x \dot{x}\right)$$

Hamiltonian system

$$\dot{x} = f(x, u)$$
$$\dot{\lambda} = -H_x(x, u, \lambda, \mu)$$

Hamiltonian

$$H(x, \lambda, \mu, u) =$$
$$L(x, u) + \lambda^T f(x, u) + \mu^T C(x, u)$$

Unknown

$$U := \left(u_0^T, \mu_0^T, \ldots, u_{N-1}^T, \mu_{N-1}^T\right)^T$$

The continuation approach

[Ohtsuka et al. 2004]

Optimality condition

$$F(U(t), x(t)) = 0$$

Impose the dynamic

$$\dot{F}(U, x) = A_s \cdot F(U, x)$$

By imposing

$$\dot{U} = F_U^{-1}\left(A_s F - F_x \dot{x}\right)$$

Krylov subspace approach

$$r = F_U \dot{U} + F_x \dot{x} - A_s F$$

Hamiltonian system

$$\dot{x} = f(x, u)$$
$$\dot{\lambda} = -H_x(x, u, \lambda, \mu)$$

Hamiltonian

$$H(x, \lambda, \mu, u) =$$
$$L(x, u) + \lambda^T f(x, u) + \mu^T C(x, u)$$

Unknown

$$U := \left(u_0^T, \mu_0^T, \ldots, u_{N-1}^T, \mu_{N-1}^T\right)^T$$

The continuation approach

[Ohtsuka et al. 2004]

Optimality condition

$$F(U(t), x(t)) = 0$$

Impose the dynamic

$$\dot{F}(U, x) = A_s \cdot F(U, x)$$

By imposing

$$\dot{U} = F_U^{-1}\left(A_s F - F_x \dot{x}\right)$$

Krylov subspace approach

$$r = F_U \dot{U} + F_x \dot{x} - A_s F$$

Hamiltonian system

$$\dot{x} = f(x, u)$$
$$\dot{\lambda} = -H_x(x, u, \lambda, \mu)$$

Hamiltonian

$$H(x, \lambda, \mu, u) = L(x, u) + \lambda^T f(x, u) + \mu^T C(x, u)$$

Unknown

$$U := \left(u_0^T, \mu_0^T, \ldots, u_{N-1}^T, \mu_{N-1}^T\right)^T$$

$$F_U W + F_x w \approx \frac{F(U + hW, x + hw) - F(U, x)}{h}$$

The continuation approach

[Ohtsuka et al. 2004]

Optimality condition

$$F(U(t), x(t)) = 0$$

Impose the dynamic

$$\dot{F}(U, x) = A_s \cdot F(U, x)$$

By imposing

$$\dot{U} = F_U^{-1}\left(A_s F - F_x \dot{x}\right)$$

Krylov subspace approach

$$r = F_U \dot{U} + F_x \dot{x} - A_s F$$

$$F_U W + F_x w \approx \frac{F(U + hW, x + hw) - F(U, x)}{h}$$

- Need initialization
- Need differentiability

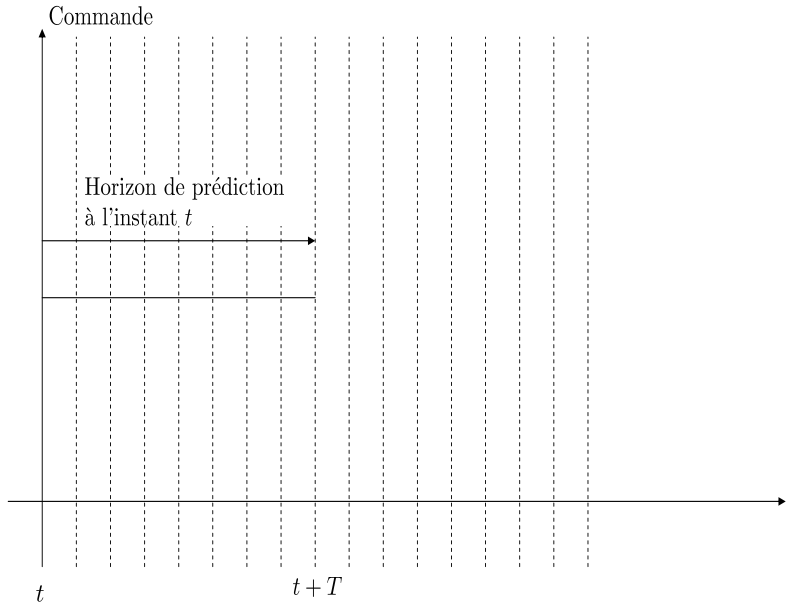
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



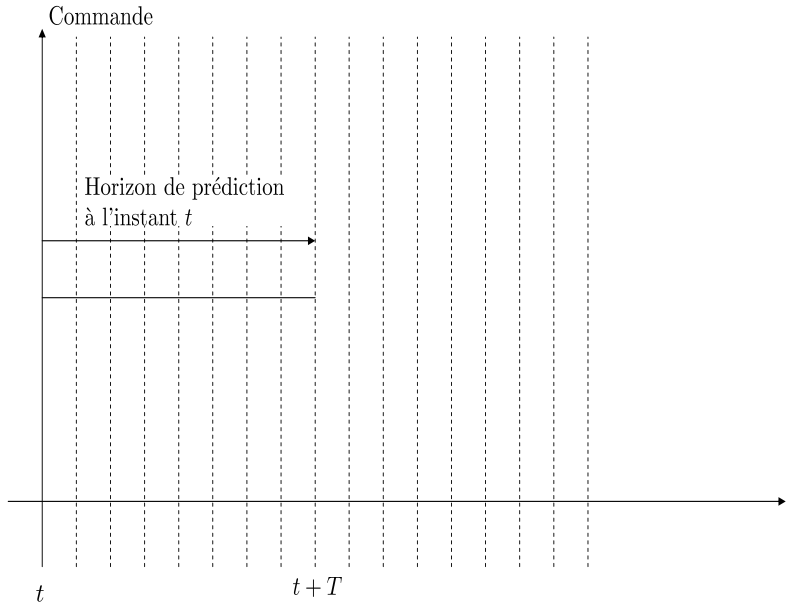
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



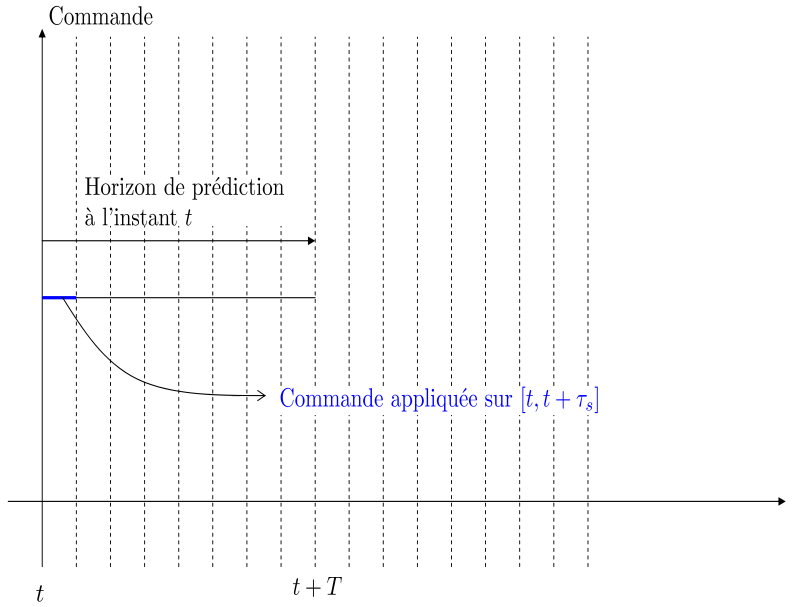
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



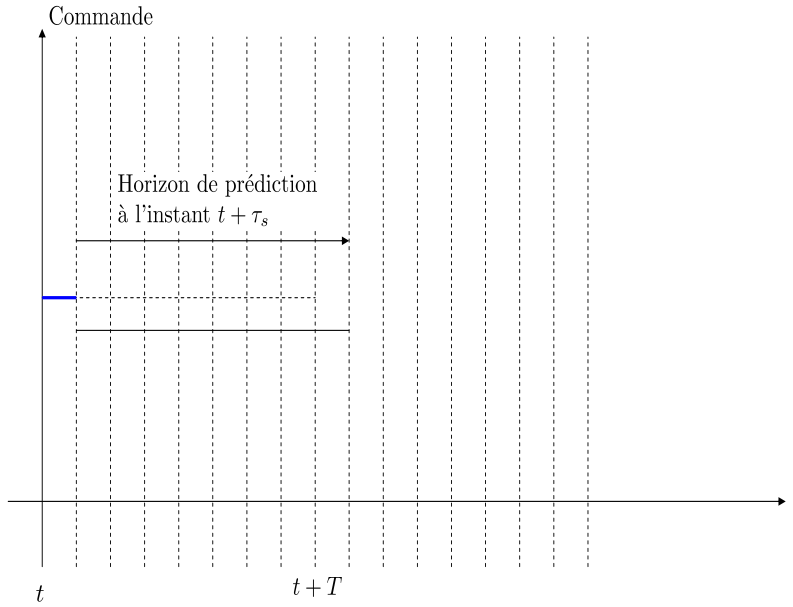
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



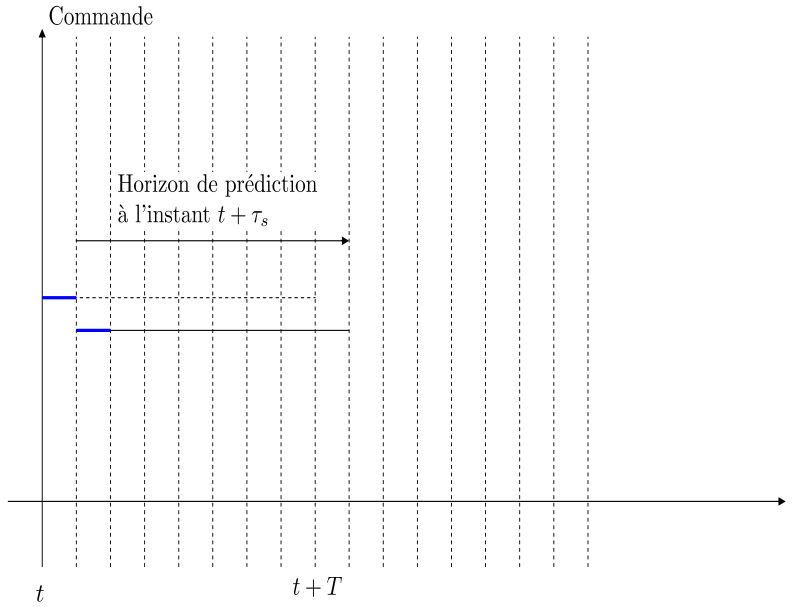
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



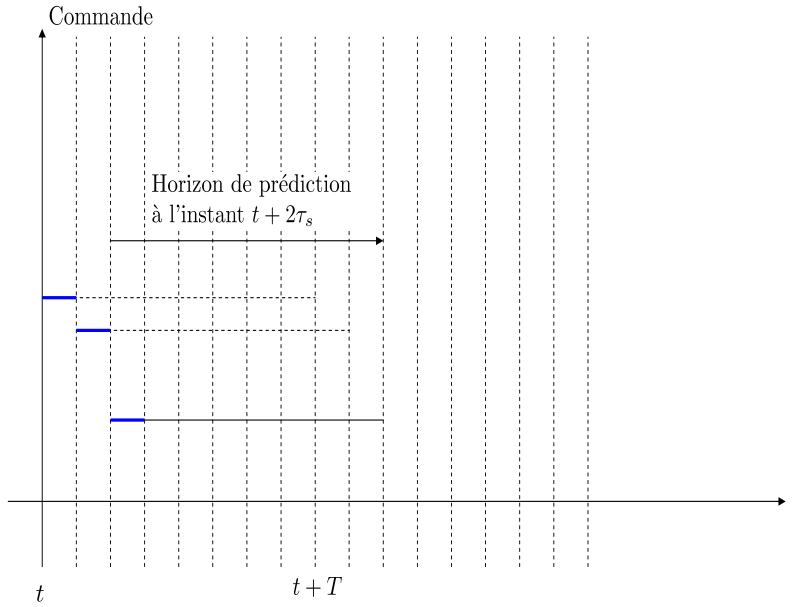
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



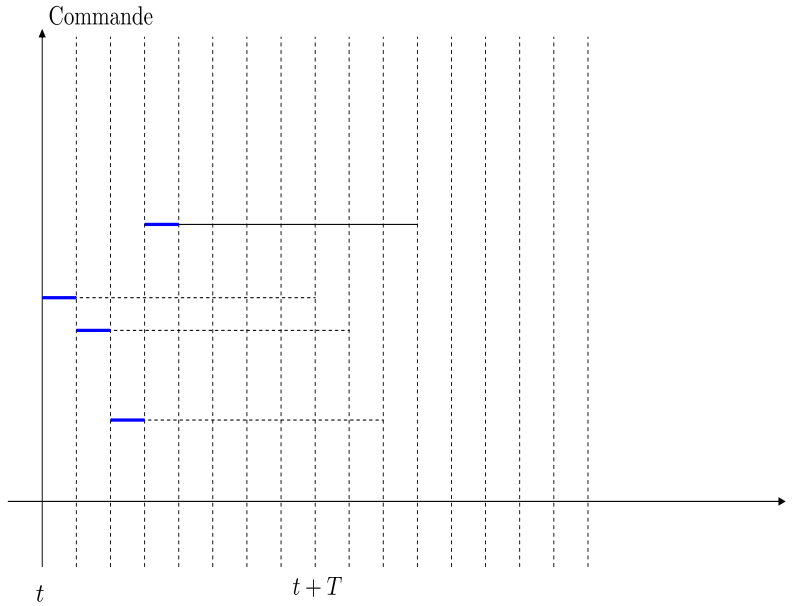
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



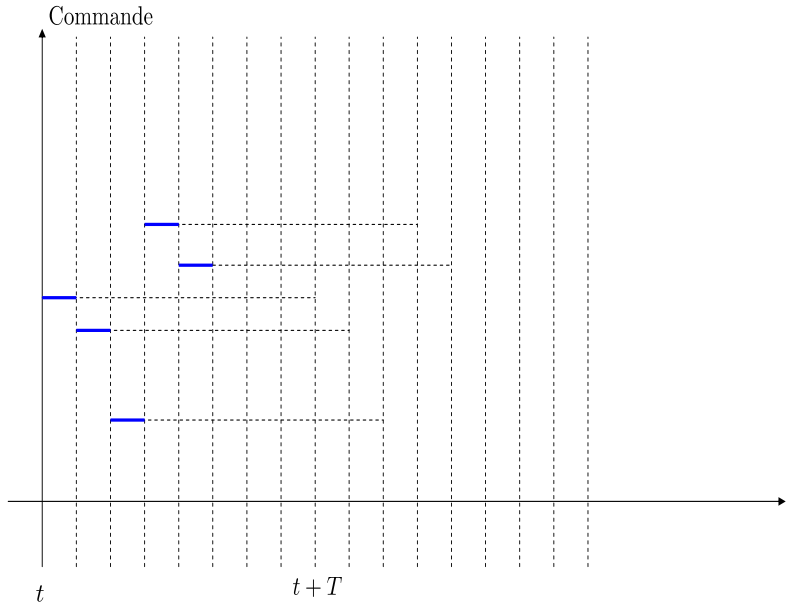
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



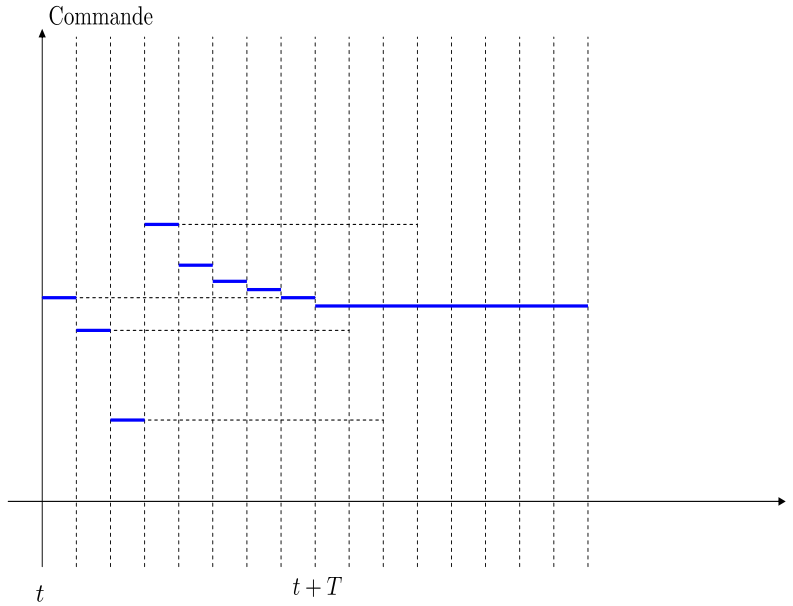
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



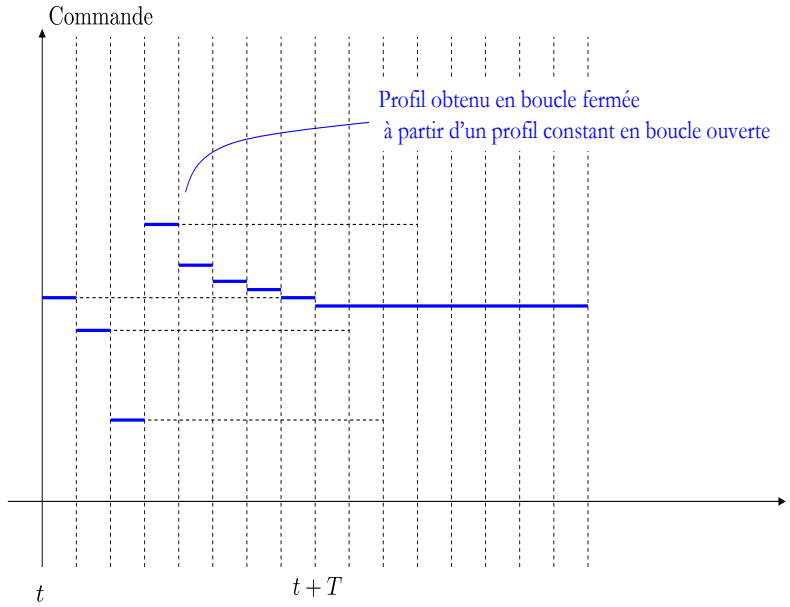
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



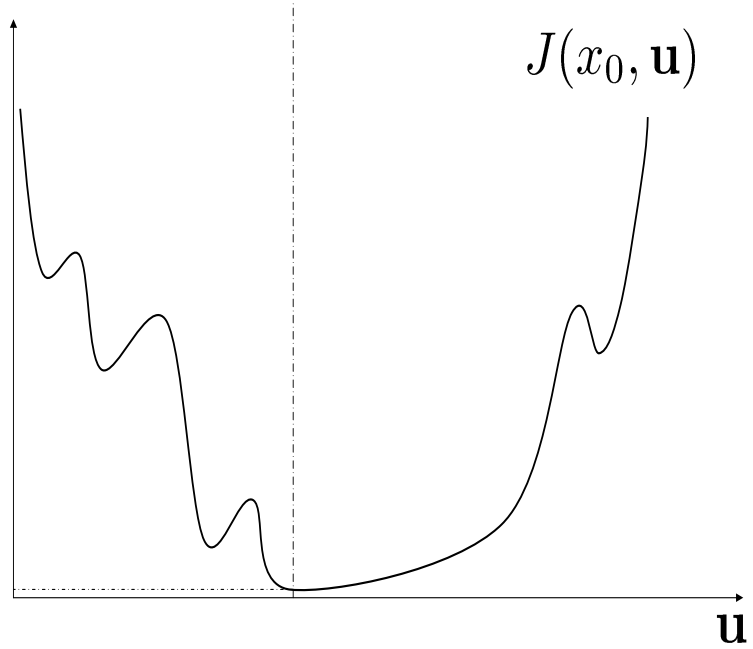
The parametric approach

Fact 2

Solve exactly a sub-optimal formulation

could be better than

Solve loosely the exact problem



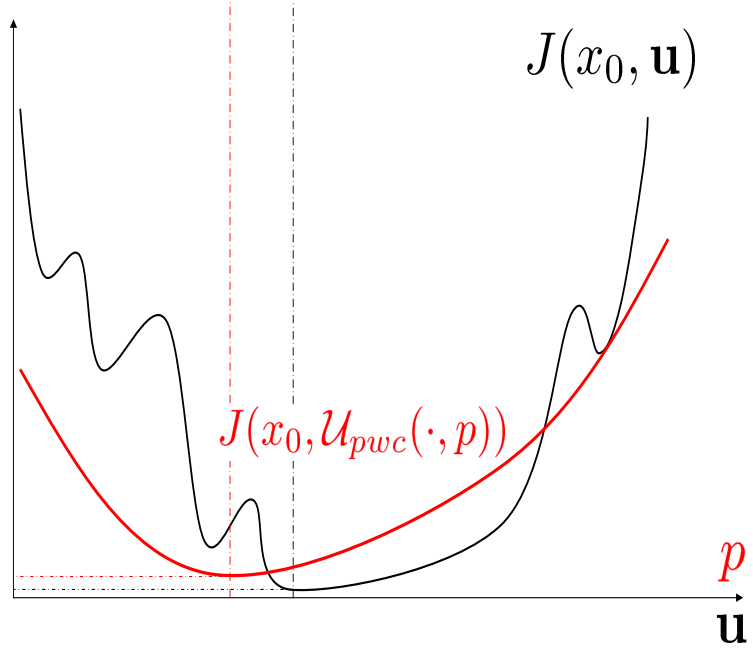
The parametric approach

Fact 2

Solve exactly a sub-optimal formulation

could be better than

Solve loosely the exact problem



The parametric approach

Fact 3

Constraints can often be embedded in the control parametrization

↓

Optimization improves performance under guaranteed stability and constraints fulfilment

The parametric approach

Fact 4

Reduced dimensional and well posed optimization problems

Render efficient

A family of simple, non smooth and memoryless algorithms that would be out of scope for large scale problems

Problem	n_p	CPU (ms)
Min Intercep. Time	1	2.0
Chained Syst. Stab	1	1.0
PVTOL Stab.	2	1.0
Twin Pendulum	1	5.0
Failure Satellite	2	10.0
AMT Hybrid Veh.	1	1.0
Diesel Air Path	2	10.0

The parametric approach

Fact 4

Reduced dimensional and well posed optimization problems

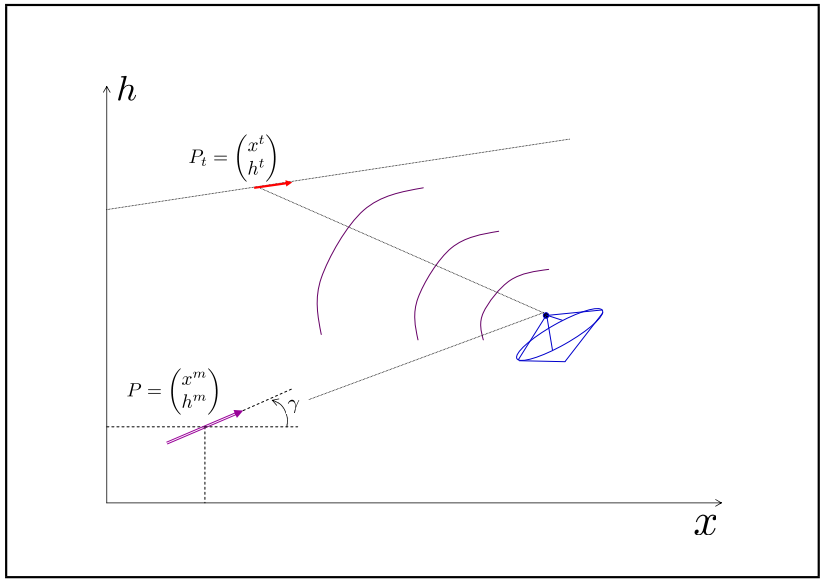
Render efficient

A family of simple, non smooth and memoryless algorithms that would be out of scope for large scale problems

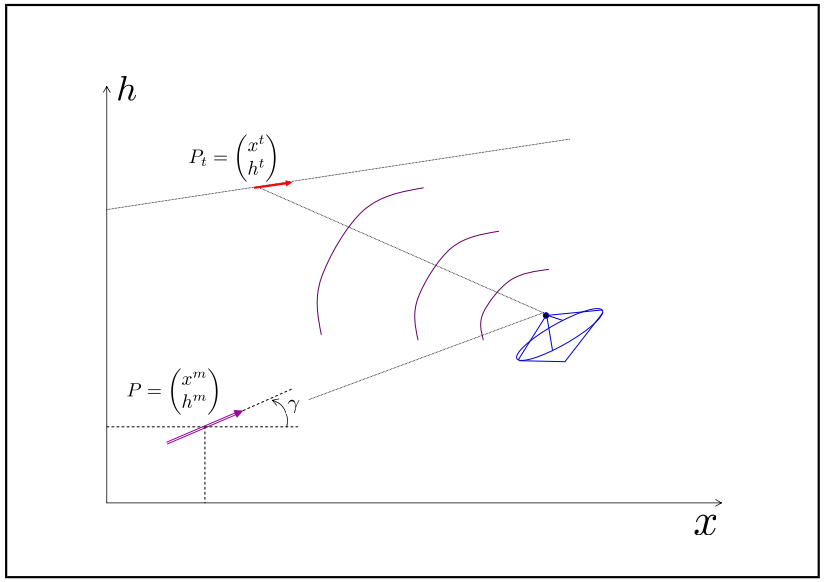
Problem	n_p	CPU (ms)
Min Intercep. Time	1	2.0
Chained Syst. Stab	1	1.0
PVTOL Stab.	2	1.0
Twin Pendulum	1	5.0
Failure Satellite	2	10.0
AMT Hybrid Veh.	1	1.0
Diesel Air Path	2	10.0

[Alamir, M. Stabilization of Nonlinear Systems Using Receding-Horizon Control Schemes : A Parametrized Approach for Fast Systems. Lecture Notes in Control and Information Sciences, Springer, London, ISBN 1-84628-470-8 (2006)]

The missile/target interception problem



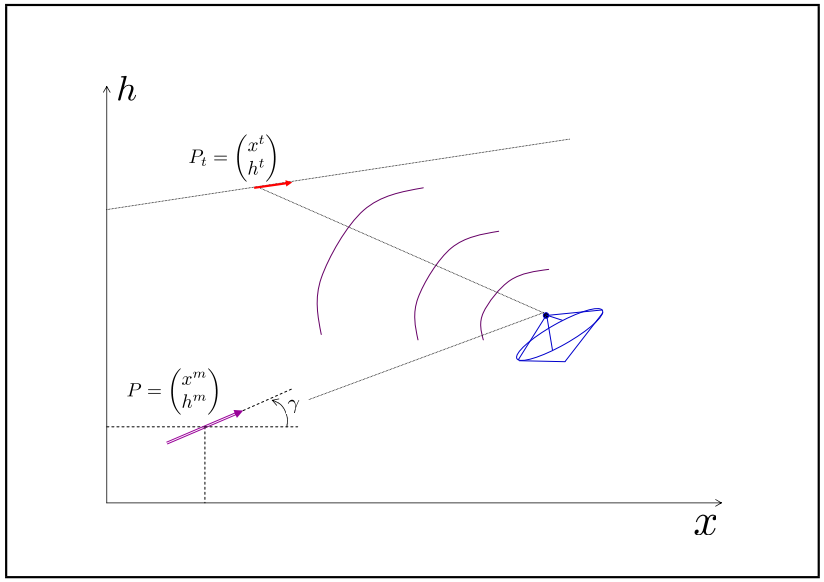
The missile/target interception problem



Missile equations

$$
\begin{aligned}
\dot{v} &= (T\cos\alpha - D(h,\alpha))/m - g\sin\alpha \\
\dot{\gamma} &= (L(h,\alpha) + T\sin\alpha)/(mv) - \frac{g}{v}\cos\gamma \\
\dot{x} &= v\cos\gamma \\
\dot{h} &= v\sin\gamma
\end{aligned}
$$

The missile/target interception problem



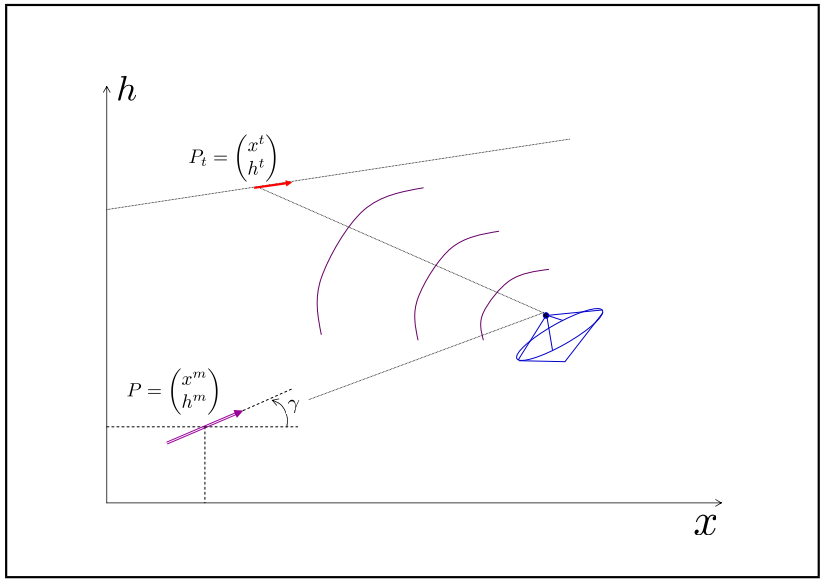
where

$$
\begin{aligned}
L &= \frac{1}{2}\rho(h)v^2 S C_{L_\alpha}(\alpha - \alpha_0) \\
D &= \frac{1}{2}\rho(h)v^2 S\left[C_{D_0} + k C_{L_\alpha}^2(\alpha - \alpha_0)^2\right] \\
\rho &= \rho_0 \exp(-\kappa h/h_0)
\end{aligned}
$$

Missile equations

$$
\begin{aligned}
\dot{v} &= (T\cos\alpha - D(h,\alpha))/m - g\sin\alpha \\
\dot{\gamma} &= (L(h,\alpha) + T\sin\alpha)/(mv) - \frac{g}{v}\cos\gamma \\
\dot{x} &= v\cos\gamma \\
\dot{h} &= v\sin\gamma
\end{aligned}
$$

The missile/target interception problem



where

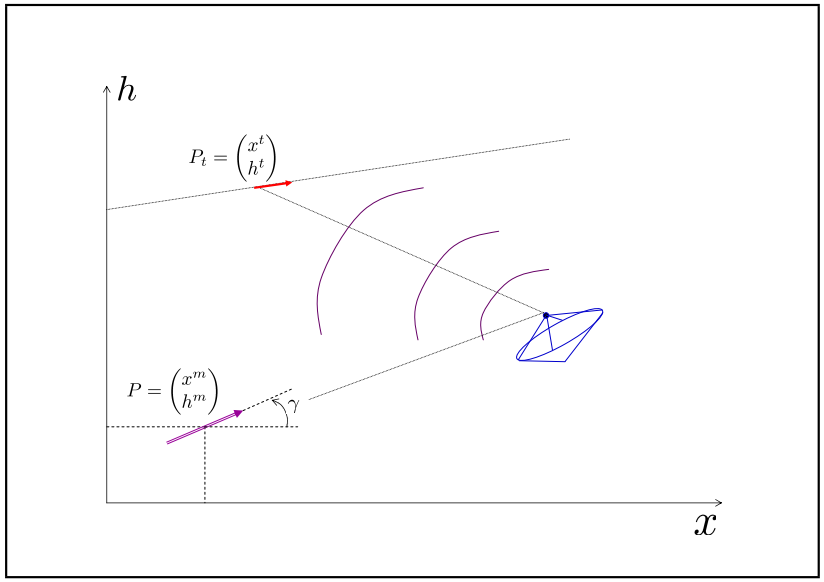
$$
\begin{aligned}
L &= \frac{1}{2}\rho(h)v^2 S C_{L_\alpha}(\alpha - \alpha_0) \\
D &= \frac{1}{2}\rho(h)v^2 S\left[C_{D_0} + k C_{L_\alpha}^2(\alpha - \alpha_0)^2\right] \\
\rho &= \rho_0 \exp(-\kappa h/h_0)
\end{aligned}
$$

$M(v,h)$	C_{L_α}	C_{D_0}	k
0.2	9.31	0.242	0.110
0.8	7.65	0.211	0.135
0.93	7.73	0.255	0.134
1.05	9.74	0.406	0.108
1.3	10.03	0.444	0.108
1.6	9.28	0.370	0.116
2.4	9.02	0.254	0.120
3.5	8.02	0.190	0.134
5.0	7.16	0.149	0.153

Missile equations

$$
\begin{aligned}
\dot{v} &= (T\cos\alpha - D(h,\alpha))/m - g\sin\alpha \\
\dot{\gamma} &= (L(h,\alpha) + T\sin\alpha)/(mv) - \frac{g}{v}\cos\gamma \\
\dot{x} &= v\cos\gamma \\
\dot{h} &= v\sin\gamma
\end{aligned}
$$

The missile/target interception problem

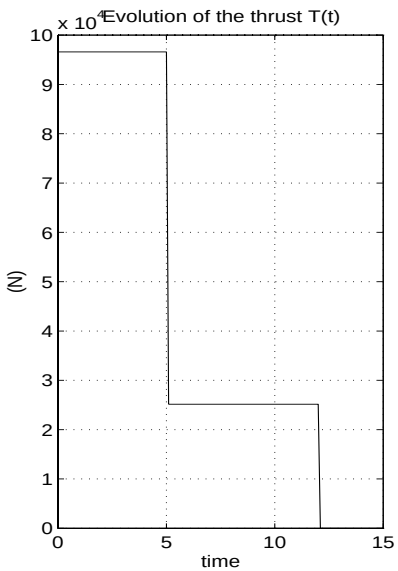


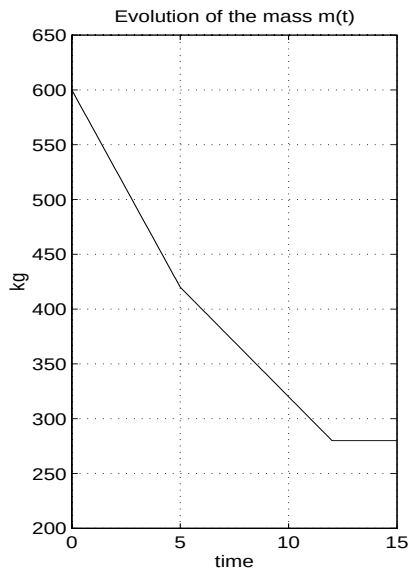
where

$$
\begin{aligned}
L &= \frac{1}{2}\rho(h)v^2 S C_{L_\alpha}(\alpha - \alpha_0) \\
D &= \frac{1}{2}\rho(h)v^2 S\left[C_{D_0} + kC_{L_\alpha}^2(\alpha - \alpha_0)^2\right] \\
\rho &= \rho_0 \exp(-\kappa h/h_0)
\end{aligned}
$$

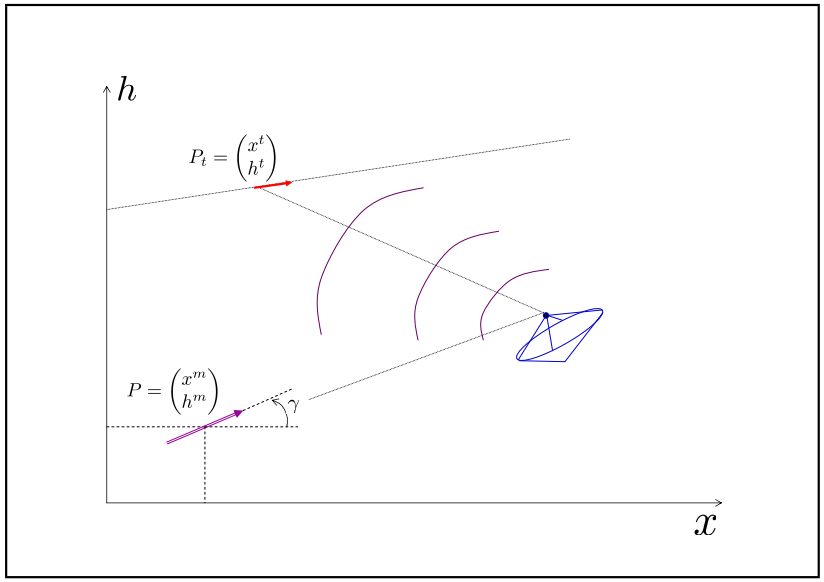
Missile equations

$$
\begin{aligned}
\dot{v} &= (T\cos\alpha - D(h,\alpha))/m - g\sin\alpha \\
\dot{\gamma} &= (L(h,\alpha) + T\sin\alpha)/(mv) - \frac{g}{v}\cos\gamma \\
\dot{x} &= v\cos\gamma \\
\dot{h} &= v\sin\gamma
\end{aligned}
$$





The missile/target interception problem



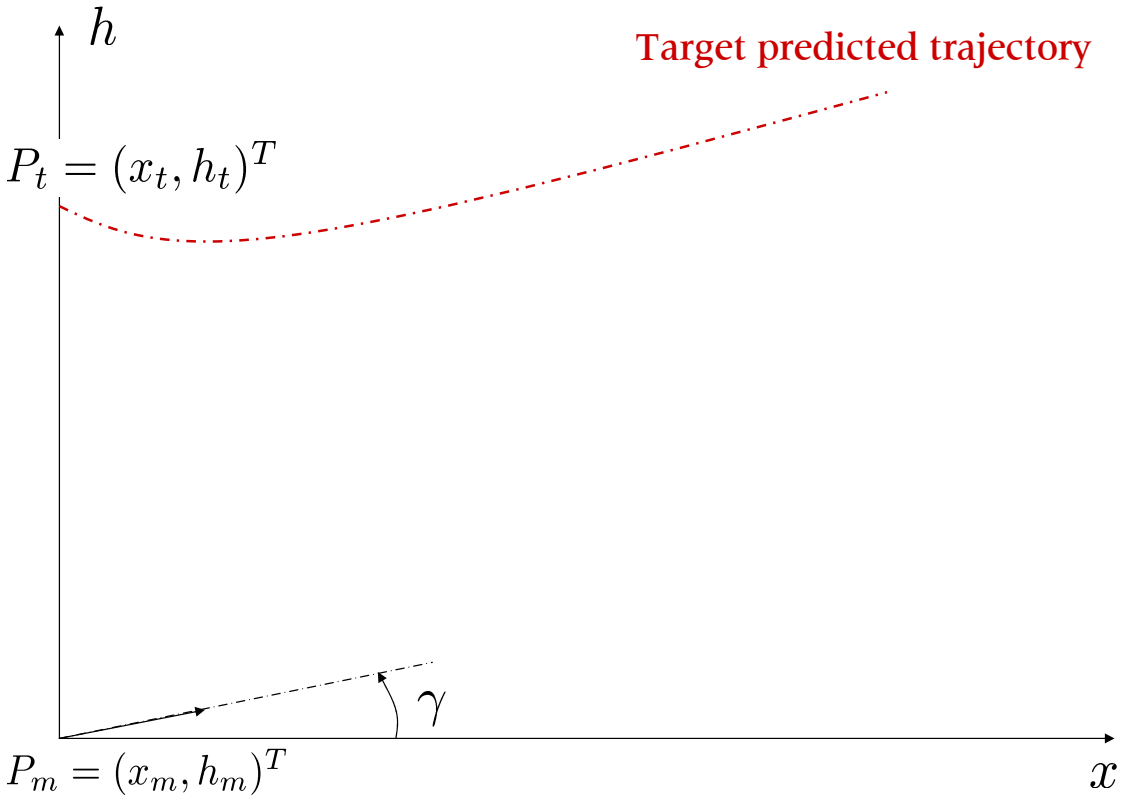
Missile equations

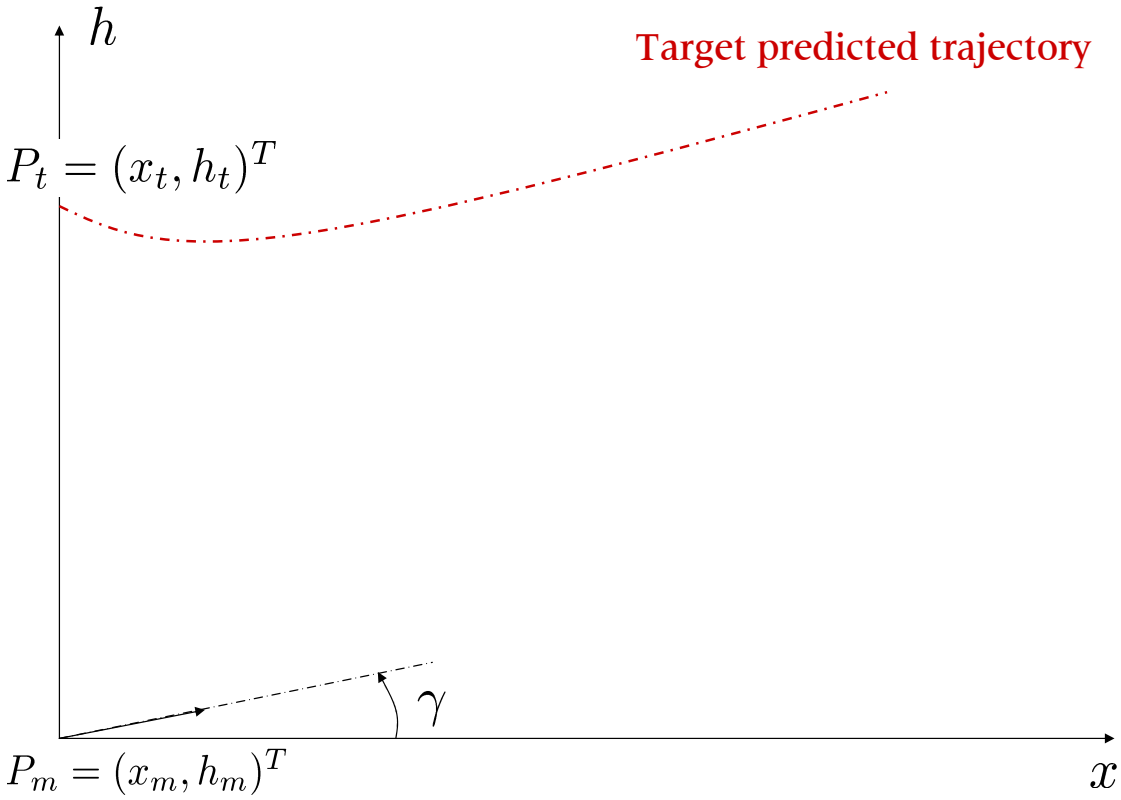
$$
\begin{aligned}
\dot{v} &= (T\cos\alpha - D(h,\alpha))/m - g\sin\alpha, \\
\dot{\gamma} &= (L(h,\alpha) + T\sin\alpha)/(mv) - \frac{g}{v}\cos\gamma, \\
\dot{x} &= v\cos\gamma, \\
\dot{h} &= v\sin\gamma,
\end{aligned}
$$

- Highly nonlinear
- Constrained
- free final time optimal control problem
- Model uncertainties
- Uncertainties on target's movement

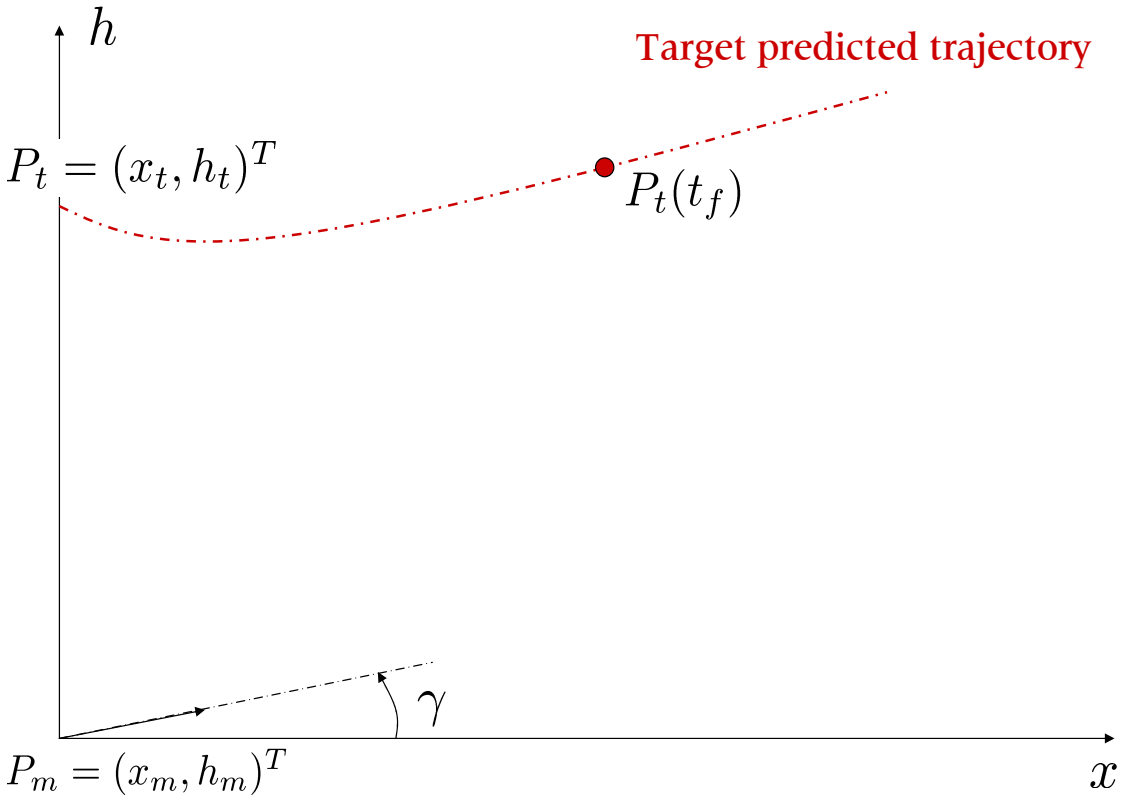
NMPC with scalar control parametrization

gipsa-lab

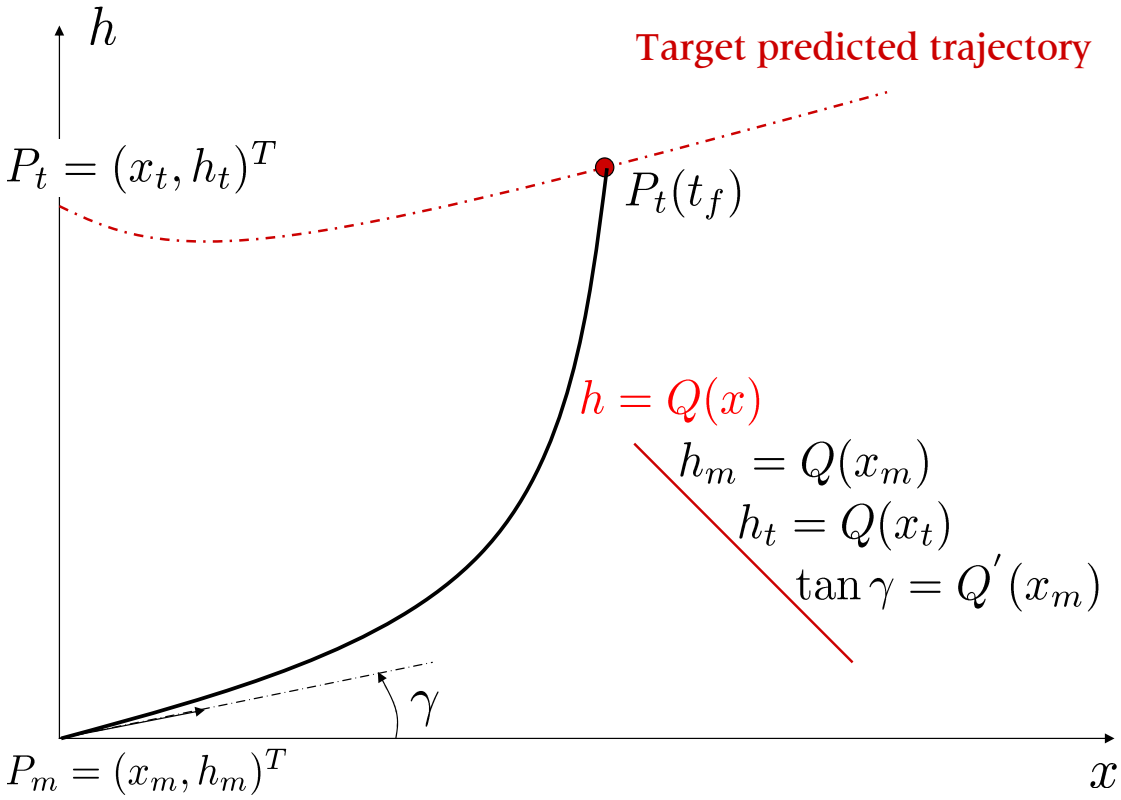

h
Target predicted trajectory
$P_t = (x_t, h_t)^T$
γ
$P_m = (x_m, h_m)^T$
x



Assume that the interception time is $t_f > 0$

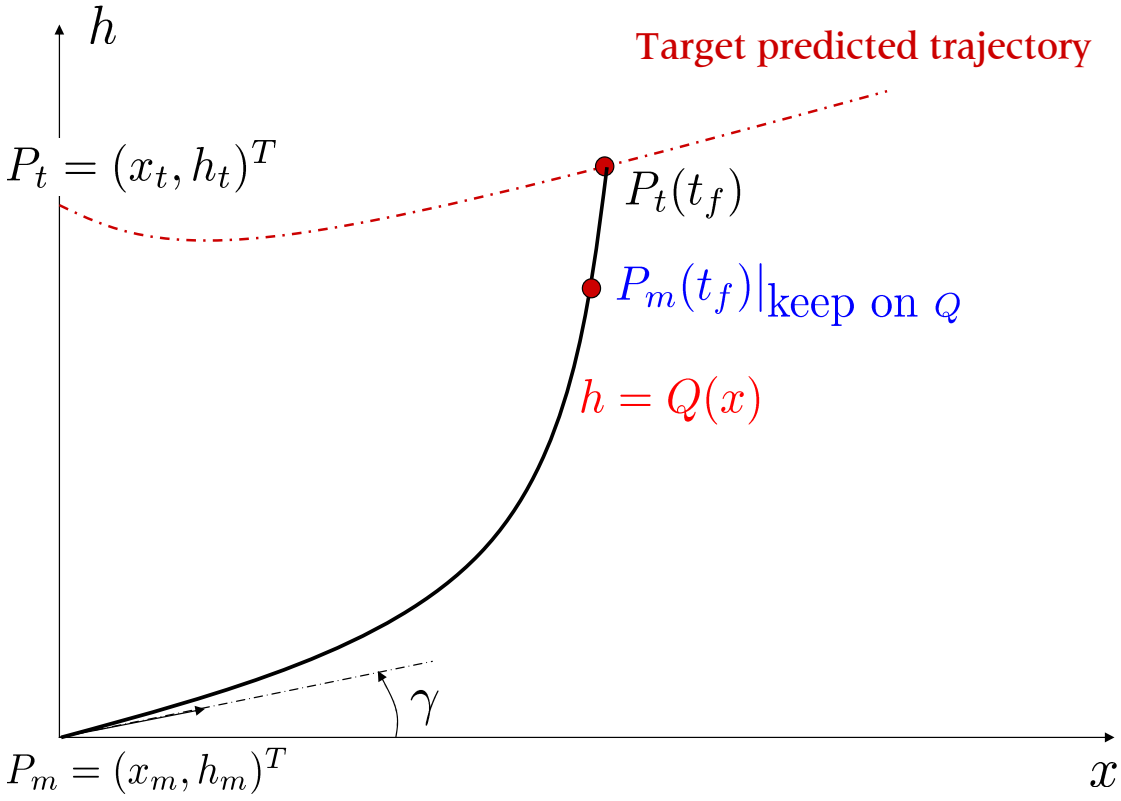


Assume that the interception time is $t_f > 0$



Assume that the interception time is $t_f > 0$

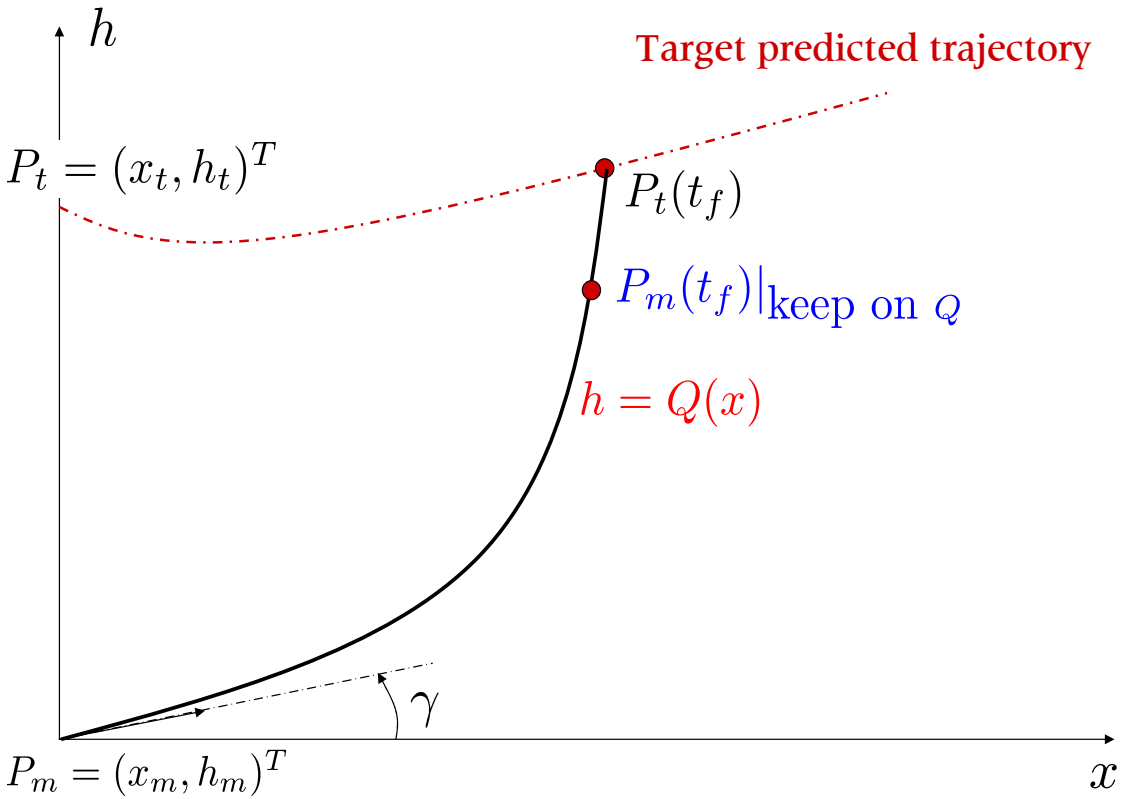
Consider the Parabola $Q(x)$ joining P_m to $P_t(t_f)$



Assume that the interception time is $t_f > 0$

Consider the Parabola $Q(x)$ joining P_m to $P_t(t_f)$

For arbitrary t_f, $P_m(t_f)|_{\text{keep on } Q} \neq P_t(t_f)$



The on-line optimization problem for RHC at decision instant t_k is

$$\min_{t_f} \quad \Big[\beta(t_f, X(t_k))\Big] = \Big\| P_m(t_f, X(t_k))_{|\text{keep on } Q} - P_t(t_f) \Big\|$$

$$\min_{t_f} \quad \left[\beta(t_f, X(t_k))\right] = \left\| P_m(t_f, X(t_k))_{|\text{keep on } Q} - P_t(t_f) \right\|$$

$$\min_{t_f} \quad \left[\beta(t_f, X(t_k))\right] = \left\| P_m(t_f, X(t_k))_{|\text{keep on } Q} - P_t(t_f) \right\|$$

Feedback 1

Solve exactly

$$\beta(t_f, X(t_k)) = 0$$

by dichotomy

$$\min_{t_f} \quad \Big[\beta(t_f, X(t_k))\Big] = \Big\| P_m(t_f, X(t_k))_{|\text{keep on } Q} - P_t(t_f) \Big\|$$

Feedback 1

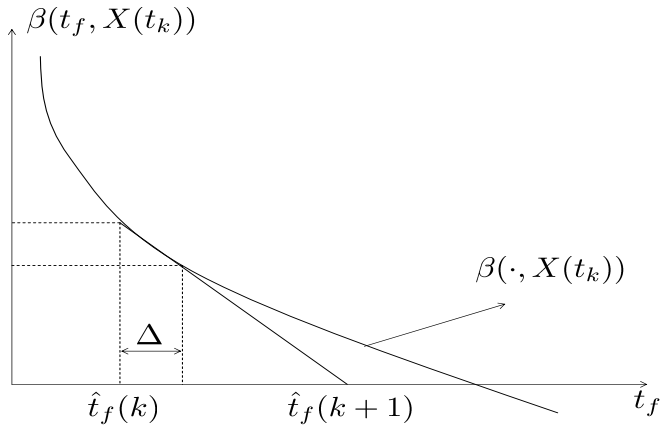
Solve exactly

$$\beta(t_f, X(t_k)) = 0$$

by dichotomy

Feedback 2

Real-Time gradient iteration



$$\min_{t_f} \quad \left[\beta(t_f, X(t_k))\right] = \left\| P_m(t_f, X(t_k))_{|\text{keep on } Q} - P_t(t_f) \right\|$$

Feedback 1

Solve exactly

$$\beta(t_f, X(t_k)) = 0$$

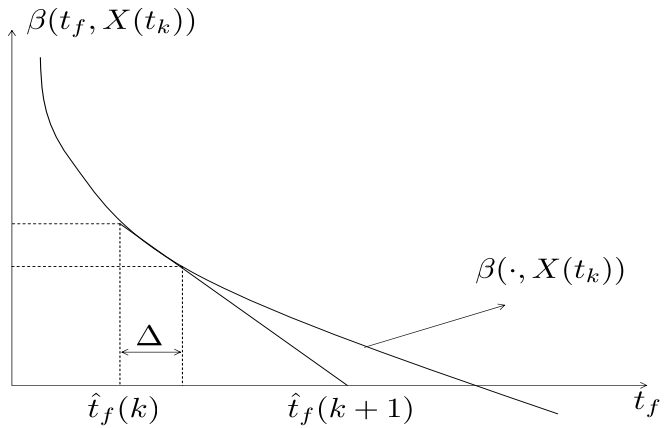
by dichotomy

Feedback 1	Feedback2
18 ms	1 ms

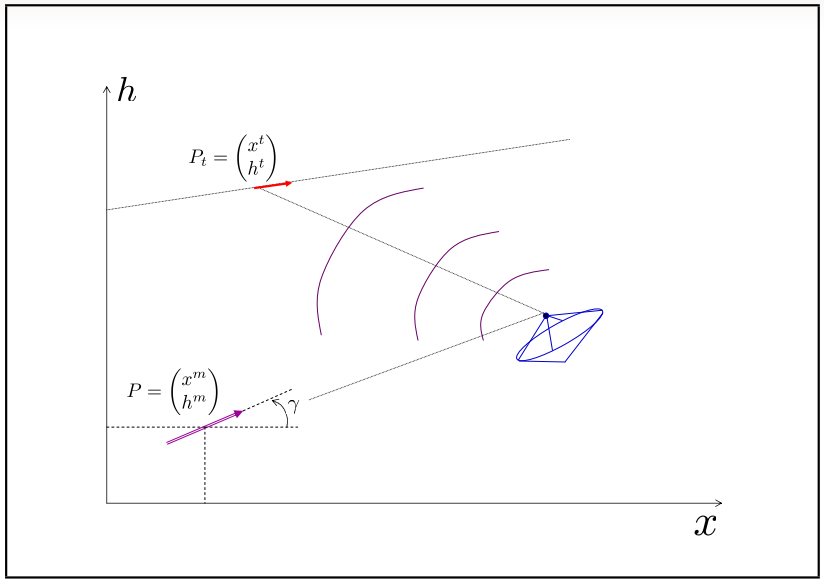
Execution time on PC-Pentium II 1.3GHz

Feedback 2

Real-Time gradient iteration



gipsa-lab



where

$$
\begin{aligned}
L &= \frac{1}{2}\rho(h)v^2 S C_{L_\alpha}(\alpha - \alpha_0) \\
D &= \frac{1}{2}\rho(h)v^2 S\left[C_{D_0} + k C_{L_\alpha}^2(\alpha - \alpha_0)^2\right] \\
\rho &= \rho_0 \exp(-\kappa h/h_0)
\end{aligned}
$$

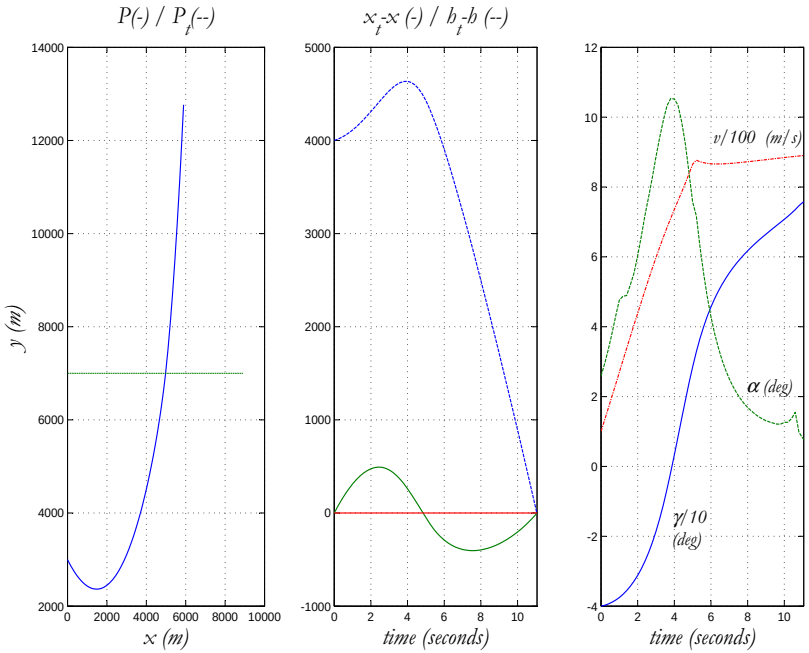
Missile equations

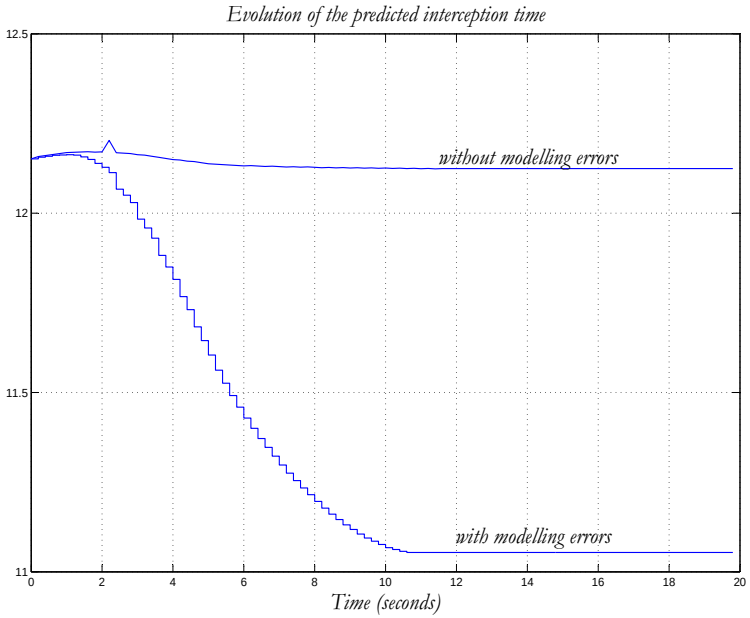
$$
\begin{aligned}
\dot{v} &= (T\cos\alpha - D(h,\alpha))/m - g\sin\alpha, \\
\dot{\gamma} &= (L(h,\alpha) + T\sin\alpha)/(mv) - \frac{g}{v}\cos\gamma, \\
\dot{x} &= v\cos\gamma, \\
\dot{h} &= v\sin\gamma,
\end{aligned}
$$

Description of uncertainties

$$
\begin{aligned}
C'_{D_0} &:= \lambda_1 C_{D_0} \\
k' &= \lambda_2 k \\
C'_{L_\alpha} &= \lambda_3 C_{L_\alpha},
\end{aligned}
$$

Simulations

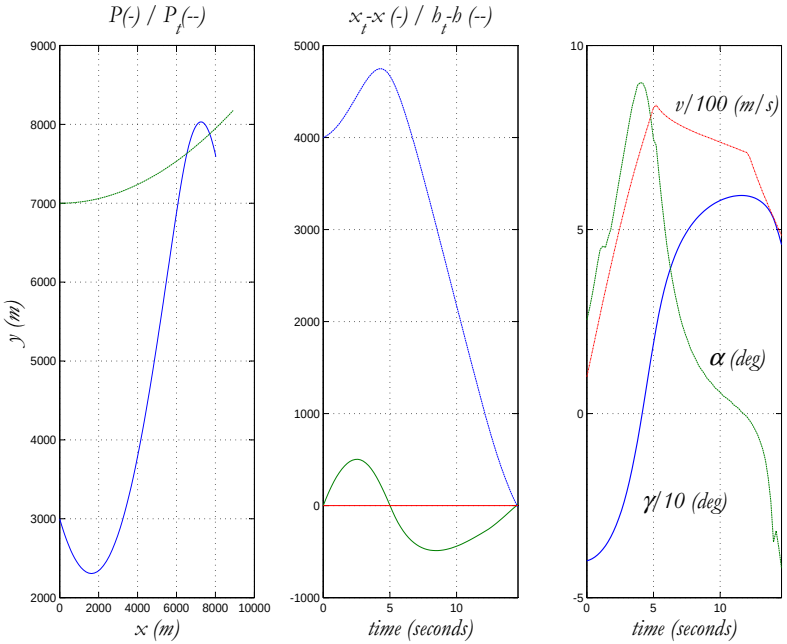


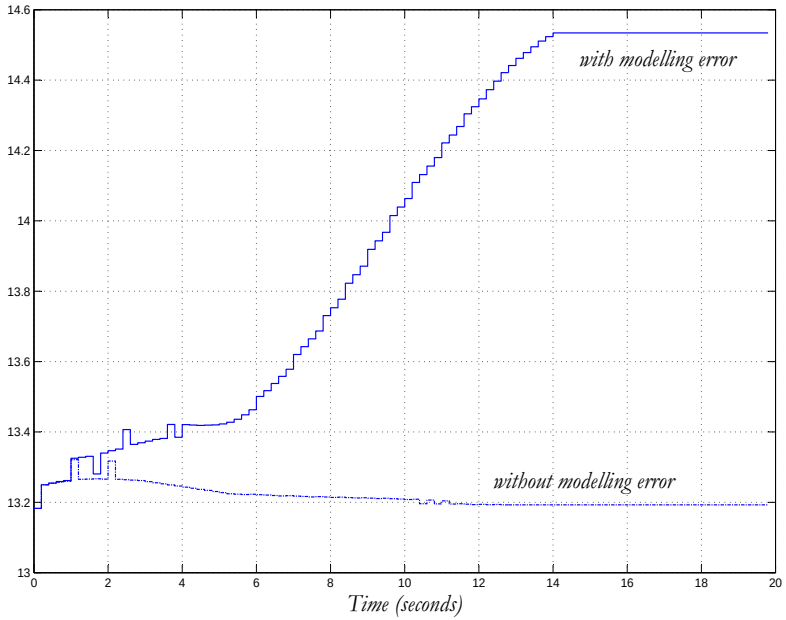


Closed-loop behavior under Feedback 1 and uncertainties given by

$$\lambda_1 = 0.8; \quad \lambda_2 = 0.6; \quad \lambda_3 = 1.25$$

Simulations

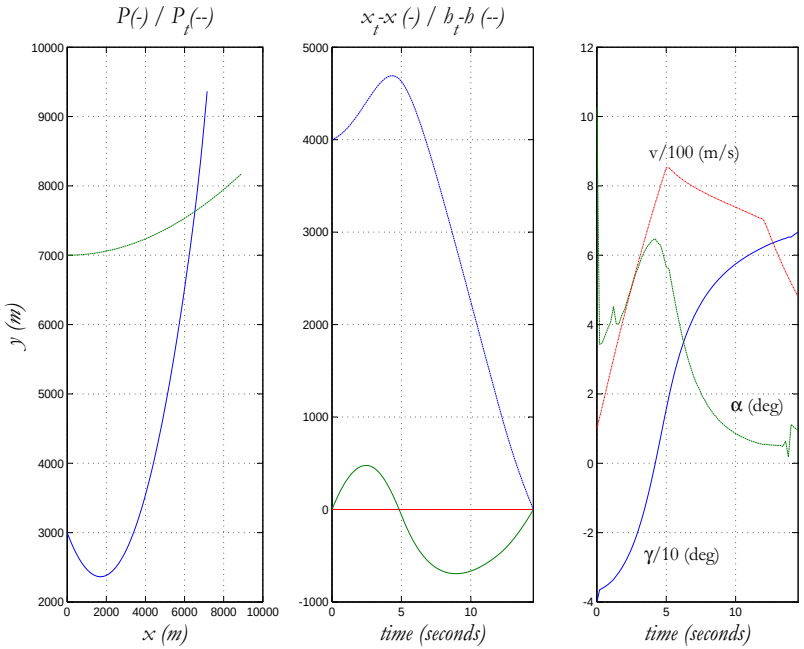


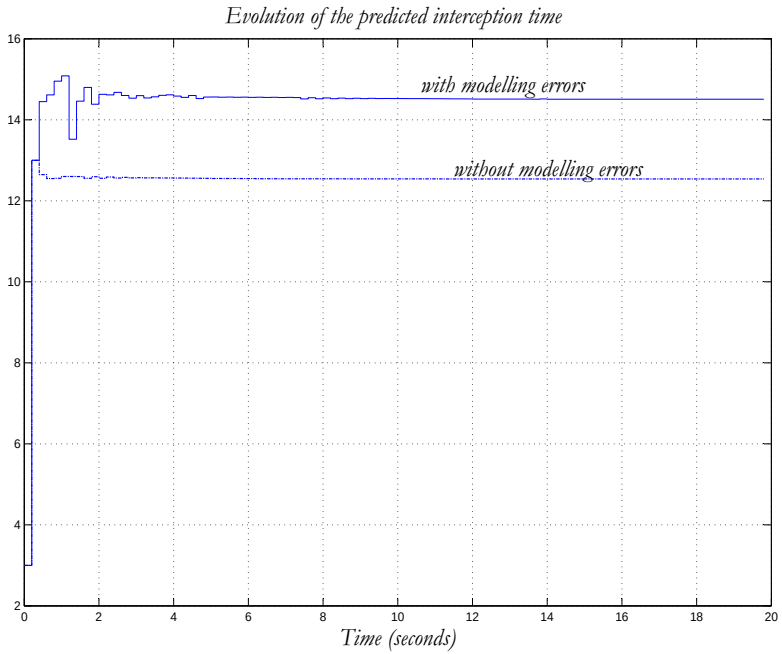


Closed-loop behavior under Feedback 1 and uncertainties given by

$$\lambda_1 = 1.2; \quad \lambda_2 = 0.6; \quad \lambda_3 = 1.25$$

Simulations





Closed-loop behavior under Feedback 2 and uncertainties given by

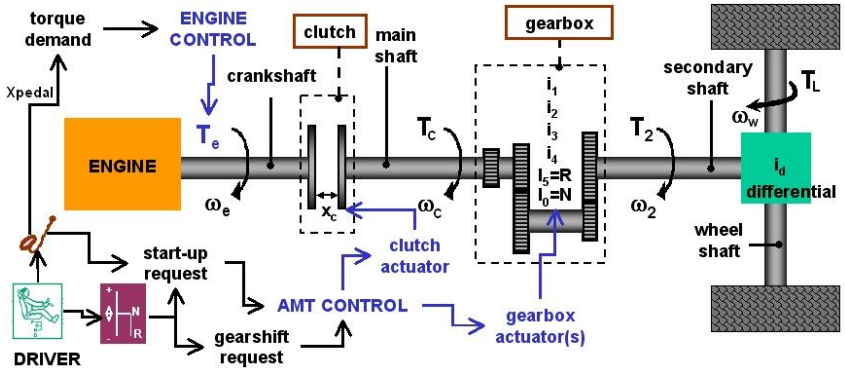
$$\lambda_1 = 1.2; \quad \lambda_2 = 0.6; \quad \lambda_3 = 1.25$$

[see Alamir, M. *Nonlinear Receding Horizon sub-optimal Guidance Law for minimum interception time problem*, Control Engineering Practice 9, No.1, 107-116 (2001)]

The Automated Manual Transmission

Trade-off between

- Manual : efficient, low weight, low cost
- Automated : better comfort / high cost & consumption

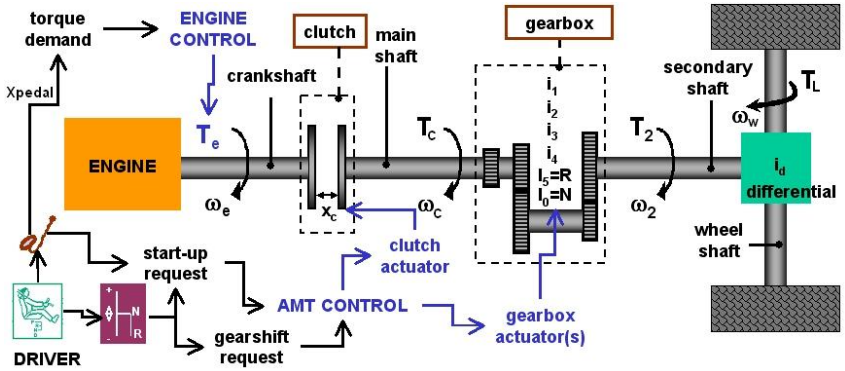


gipsa-lab

The Automated Manual Transmission

Trade-off between

- Manual : efficient, low weight, low cost
- Automated : better comfort / high cost & consumption



$$J_e\dot{\omega}_e = T_e - sign(\omega_{sl}) \cdot T_c(x_c)$$

$$[J_c + J_{eq}(i_g, i_d)]\dot{\omega}_c = sign(\omega_{sl}) \cdot T_c(x_c) - \frac{1}{i_g i_d}\left[k_{tw}\theta_{cw} + \beta_{tw}\left(\frac{\omega_c}{i_g i_d} - \omega_w\right)\right]$$

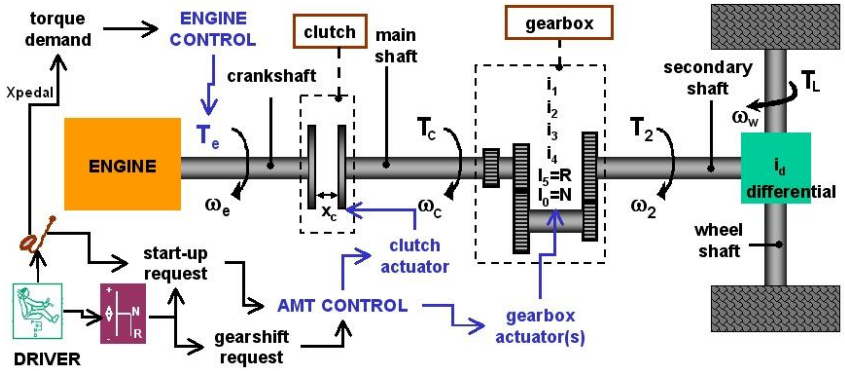
$$J_w\dot{\omega}_w = k_{tw}\theta_{cw} + \beta_{tw}\left(\frac{\omega_c}{i_g i_d} - \omega_w\right) - T_L(\omega_w)$$

$$\dot{\theta}_{cw} = \frac{\omega_c}{i_g i_d} - \omega_w$$

The Automated Manual Transmission

Trade-off between

- Manual : efficient, low weight, low cost
- Automated : better comfort / high cost & consumption

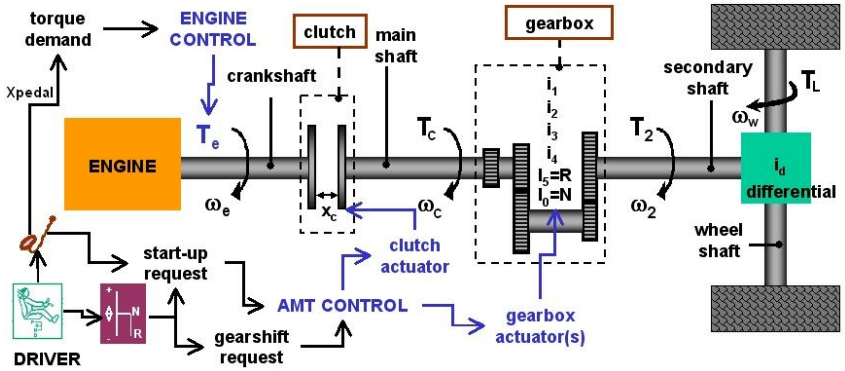


Control objective :

The Automated Manual Transmission

Trade-off between

- Manual : efficient, low weight, low cost
- Automated : better comfort / high cost & consumption



Control objective :

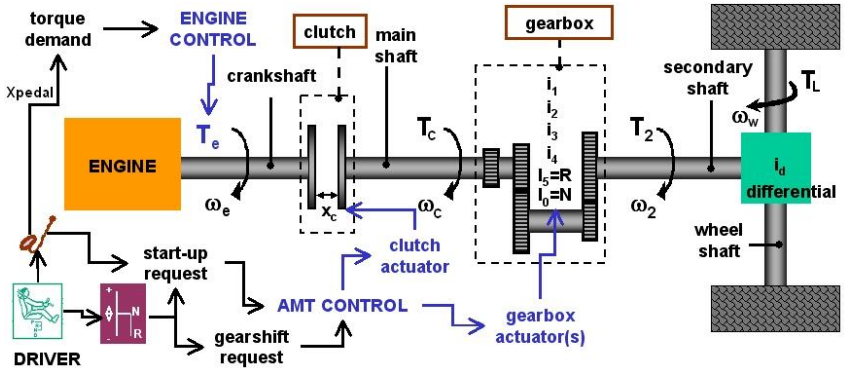
- Smooth transitions for $\omega_{sl} \longrightarrow 0$

  (this is intimately linked to the dynamic of the slip velocity $\omega_{sl} = \omega_e - \omega_c$)

The Automated Manual Transmission

Trade-off between

- Manual : efficient, low weight, low cost
- Automated : better comfort / high cost & consumption



Control objective :

- Smooth transitions for $\omega_{sl} \longrightarrow 0$

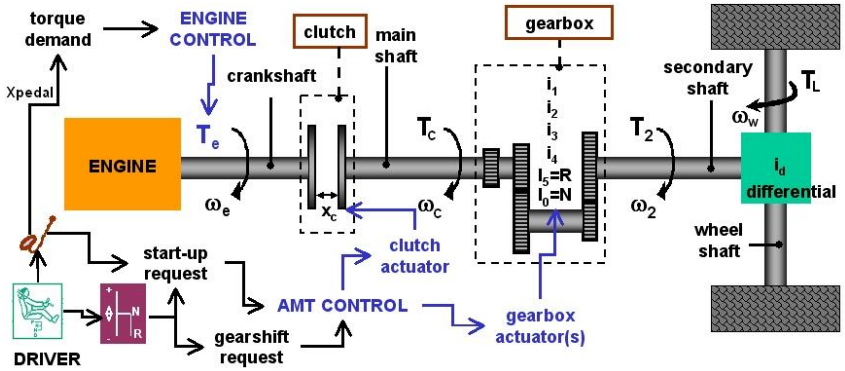
  (this is intimately linked to the dynamic of the slip velocity $\omega_{sl} = \omega_e - \omega_c$)
- Transparency

  The torque must be somehow related to the pedal's position

The Automated Manual Transmission

Trade-off between

- Manual : efficient, low weight, low cost
- Automated : better comfort / high cost & consumption



Control objective :

- Smooth transitions for $\omega_{sl} \longrightarrow 0$

  (this is intimately linked to the dynamic of the slip velocity $\omega_{sl} = \omega_e - \omega_c$)
- Transparency

  The torque must be somehow related to the pedal's position
- Control of the engine velocity :

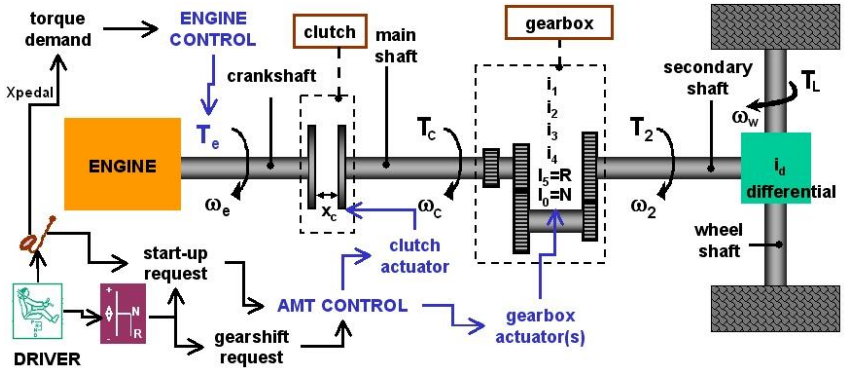
$$\omega_e^{ref} = max\left\{\omega_e^0, \mathcal{T}^{-1}\left(T_e^d(X_{pedal}, \omega_e)\right)\right\}$$

gipsa-lab

The Automated Manual Transmission

Trade-off between

- Manual : efficient, low weight, low cost
- Automated : better comfort / high cost & consumption



Constraints :

- Torque saturation

$$\begin{aligned} T_e &\in [T_e^{min}, T_e^{max}(\omega_e)] \\ T_c &\in [T_c^{min}, T_c^{max}(\omega_e)] \end{aligned}$$

- Torque variation rate saturation

$$\begin{aligned} \dot{T}_e &\in [\dot{T}_e^{min}, \dot{T}_e^{max}] \\ \dot{T}_c &\in [\dot{T}_c^{min}, \dot{T}_c^{max}] \end{aligned}$$

Some remarks

- Need for a simplified model

Some remarks

- Need for a simplified model
- A conventional constrained linear quadratic with the required prediction horizon cannot be solved within the allowable sampling period (50 ms).

Some remarks

- Need for a simplified model
- A conventional constrained linear quadratic with the required prediction horizon cannot be solved within the allowable sampling period (50 ms).
- Existing works suggested a first time period of open-loop control.

Some remarks

- Need for a simplified model
- A conventional constrained linear quadratic with the required prediction horizon cannot be solved within the allowable sampling period (50 ms).
- Existing works suggested a first time period of open-loop control.
- PWA related approaches are still not real-time compatible.

Some remarks

- Need for a simplified model
- A conventional constrained linear quadratic with the required prediction horizon cannot be solved within the allowable sampling period (50 ms).
- Existing works suggested a first time period of open-loop control.
- PWA related approaches are still not real-time compatible.
- The MPC stage uses the torque set-points T_e^{sp} and T_c^{sp} as control variables, namely :

$$u := \left(T_e^{sp}, T_c^{sp}\right)^T$$

Some remarks

- Need for a simplified model
- A conventional constrained linear quadratic with the required prediction horizon cannot be solved within the allowable sampling period (50 ms).
- Existing works suggested a first time period of open-loop control.
- PWA related approaches are still not real-time compatible.
- The MPC stage uses the torque set-points T_e^{sp} and T_c^{sp} as control variables, namely :

$$u := \left(T_e^{sp}, T_c^{sp}\right)^T$$

- Therefore, the closed-loop has to cope with the torque control imperfections & simplified modeling error

Simplified Model for Control

Simplified Model for Control

$$J_e\dot{\omega}_e = T_e - sign(\omega_{sl}) \cdot T_c(x_c)$$

$$[J_c + J_{eq}(i_g, i_d)]\dot{\omega}_c = sign(\omega_{sl}) \cdot T_c(x_c) - \frac{1}{i_g i_d}\left[k_{tw}\theta_{cw} + \beta_{tw}\left(\frac{\omega_c}{i_g i_d} - \omega_w\right)\right]$$

$$J_w\dot{\omega}_w = k_{tw}\theta_{cw} + \beta_{tw}\left(\frac{\omega_c}{i_g i_d} - \omega_w\right) - T_L(\omega_w)$$

$$\dot{\theta}_{cw} = \frac{\omega_c}{i_g i_d} - \omega_w$$

Simplified Model for Control

$$J_e\dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \delta_e$$

$$[J_c + J_{eq}(i_g, i_d)]\dot{\omega}_c = sign(\omega_{sl}) \cdot T_c(x_c) - \frac{1}{i_g i_d}\left[k_{tw}\theta_{cw} + \beta_{tw}\left(\frac{\omega_c}{i_g i_d} - \omega_w\right)\right]$$

$$J_w\dot{\omega}_w = k_{tw}\theta_{cw} + \beta_{tw}\left(\frac{\omega_c}{i_g i_d} - \omega_w\right) - T_L(\omega_w)$$

$$\dot{\theta}_{cw} = \frac{\omega_c}{i_g i_d} - \omega_w$$

Simplified Model for Control

$$J_e\dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \delta_e$$

$$[J_c + J_{eq}(i_g, i_d)]\dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \delta_c$$

$$J_w\dot{\omega}_w = k_{tw}\theta_{cw} + \beta_{tw}\left(\frac{\omega_c}{i_g i_d} - \omega_w\right) - T_L(\omega_w)$$

$$\dot{\theta}_{cw} = \frac{\omega_c}{i_g i_d} - \omega_w$$

Simplified Model for Control

$$J_e \dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \delta_e$$

$$[J_c + J_{eq}(i_g, i_d)]\dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \delta_c$$

Simplified Model for Control

$$J_e \dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \delta_e$$

$$[J_c + J_{eq}(i_g, i_d)]\dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \delta_c$$

- δ_e copes with tracking error on T_e^{sp} and T_c^{sp}.

Simplified Model for Control

$$J_e \dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \delta_e$$

$$[J_c + J_{eq}(i_g, i_d)]\dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \delta_c$$

- δ_e copes with tracking error on T_e^{sp} and T_c^{sp}.
- δ_c copes with
  1. tracking error on T_c^{sp}
  2. non explicitly represented dynamics
  3. unknown load T_L

Simplified Model for Control

$$J_e \dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \delta_e$$

$$[J_c + J_{eq}(i_g, i_d)]\dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \delta_c$$

- δ_e copes with tracking error on T_e^{sp} and T_c^{sp}.
- δ_c copes with
  1. tracking error on T_c^{sp}
  2. non explicitly represented dynamics
  3. unknown load T_L
- Over time windows where ω_{sl} keeps a constant sign, the system is linear.

Simplified Model for Control

$$J_e \dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \delta_e$$

$$[J_c + J_{eq}(i_g, i_d)]\dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \delta_c$$

- δ_e copes with tracking error on T_e^{sp} and T_c^{sp}.
- δ_c copes with
  1. tracking error on T_c^{sp}
  2. non explicitly represented dynamics
  3. unknown load T_L
- Over time windows where ω_{sl} keeps a constant sign, the system is linear.
- Since ω_e and ω_c are measured, observers can be designed for δ_e and δ_c

Simplified Model for Control

$$J_e \dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \delta_e$$

$$[J_c + J_{eq}(i_g, i_d)]\dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \delta_c$$

- δ_e copes with tracking error on T_e^{sp} and T_c^{sp}.
- δ_c copes with
  1. tracking error on T_c^{sp}
  2. non explicitly represented dynamics
  3. unknown load T_L
- Over time windows where ω_{sl} keeps a constant sign, the system is linear.
- Since ω_e and ω_c are measured, observers can be designed for δ_e and δ_c
- MPC design uses observer generated $\hat{\delta}_e$ and $\hat{\delta}_c$ with constant dynamics.

The Parameterized NMPC scheme

Model for prediction

$$J_e \dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \hat{\delta}_e$$
$$[J_c + J_{eq}] \dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \hat{\delta}_c$$

The Parameterized NMPC scheme

Model for prediction

$$J_e \dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \hat{\delta}_e$$
$$[J_c + J_{eq}]\dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \hat{\delta}_c$$

- Consider a prediction horizon $N_p \cdot \tau_s$

The Parameterized NMPC scheme

Model for prediction

$$J_e\dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \hat{\delta}_e$$
$$[J_c + J_{eq}]\dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \hat{\delta}_c$$

- Consider a prediction horizon $N_p \cdot \tau_s$
- Consider the reference on ω_e : $\omega_e^{ref} = max\left\{\omega_e^0, \mathcal{T}^{-1}\left(T_e^d(X_{pedal}, \omega_e)\right)\right\}$

The Parameterized NMPC scheme

Model for prediction

$$J_e\dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \hat{\delta}_e$$
$$[J_c + J_{eq}]\dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \hat{\delta}_c$$

- Consider a prediction horizon $N_p \cdot \tau_s$
- Consider the reference on ω_e : $\omega_e^{ref} = max\left\{\omega_e^0, \mathcal{T}^{-1}\left(T_e^d(X_{pedal}, \omega_e)\right)\right\}$
- Consider the reference on the slipping velocity $\omega_{sl} = \omega_e - \omega_c$ defined at each sampling instant $k \cdot \tau_s$:

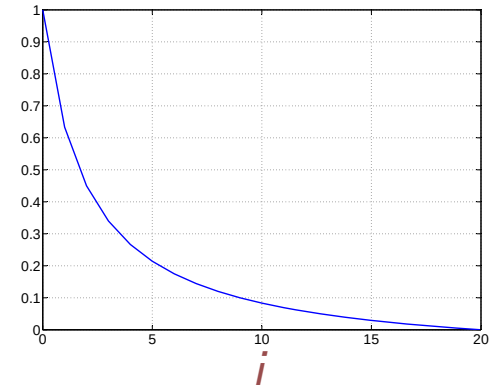
$$\omega_{sl}^{ref}(k+i, p) = \frac{1 - i/p}{1 + \lambda \cdot i/p} \cdot \omega_{sl}(k)$$

The Parameterized NMPC scheme

Model for prediction

$$J_e\dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \hat{\delta}_e$$
$$[J_c + J_{eq}]\dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \hat{\delta}_c$$

$\frac{1-i/p}{1+\lambda \cdot i/p}$, $N_p = 20$, $p = N_p$



- Consider a prediction horizon $N_p \cdot \tau_s$
- Consider the reference on ω_e : $\omega_e^{ref} = max\left\{\omega_e^0, \mathcal{T}^{-1}\left(T_e^d(X_{pedal}, \omega_e)\right)\right\}$
- Consider the reference on the slipping velocity $\omega_{sl} = \omega_e - \omega_c$ defined at each sampling instant $k \cdot \tau_s$:

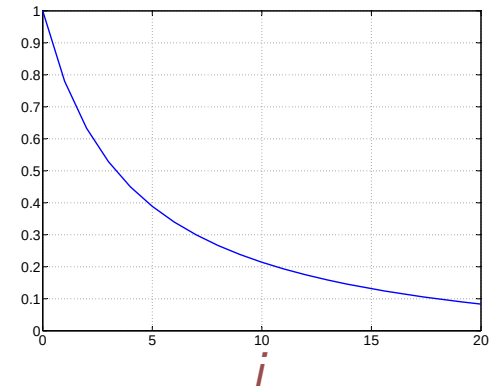
$$\omega_{sl}^{ref}(k+i, p) = \frac{1 - i/p}{1 + \lambda \cdot i/p} \cdot \omega_{sl}(k)$$

The Parameterized NMPC scheme

Model for prediction

$$J_e\dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \hat{\delta}_e$$
$$[J_c + J_{eq}]\dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \hat{\delta}_c$$

$\frac{1-i/p}{1+\lambda \cdot i/p}$, $N_p = 20$, $p = 2N_p$



- Consider a prediction horizon $N_p \cdot \tau_s$
- Consider the reference on ω_e : $\omega_e^{ref} = max\left\{\omega_e^0, \mathcal{T}^{-1}\left(T_e^d(X_{pedal}, \omega_e)\right)\right\}$
- Consider the reference on the slipping velocity $\omega_{sl} = \omega_e - \omega_c$ defined at each sampling instant $k \cdot \tau_s$:

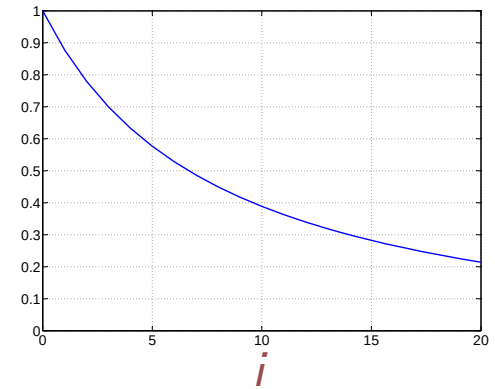
$$\omega_{sl}^{ref}(k+i, p) = \frac{1 - i/p}{1 + \lambda \cdot i/p} \cdot \omega_{sl}(k)$$

The Parameterized NMPC scheme

Model for prediction

$$J_e\dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \hat{\delta}_e$$
$$[J_c + J_{eq}]\dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \hat{\delta}_c$$

$\frac{1-i/p}{1+\lambda \cdot i/p}$, $N_p = 20$, $p = 3N_p$



- Consider a prediction horizon $N_p \cdot \tau_s$
- Consider the reference on ω_e : $\omega_e^{ref} = max\left\{\omega_e^0, \mathcal{T}^{-1}\left(T_e^d(X_{pedal}, \omega_e)\right)\right\}$
- Consider the reference on the slipping velocity $\omega_{sl} = \omega_e - \omega_c$ defined at each sampling instant $k \cdot \tau_s$:

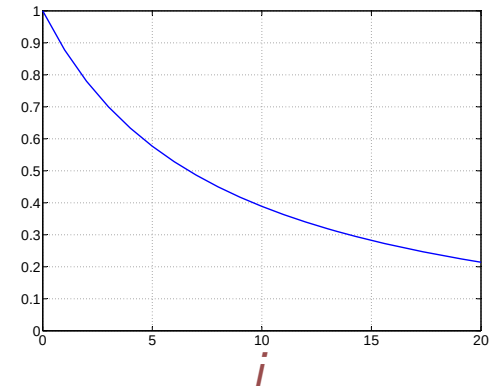
$$\omega_{sl}^{ref}(k+i, p) = \frac{1 - i/p}{1 + \lambda \cdot i/p} \cdot \omega_{sl}(k)$$

The Parameterized NMPC scheme

Model for prediction

$$J_e\dot{\omega}_e = u_1 - sign(\omega_{sl}) \cdot u_2 + \hat{\delta}_e$$
$$[J_c + J_{eq}]\dot{\omega}_c = sign(\omega_{sl}) \cdot u_2 - \hat{\delta}_c$$

$\frac{1-i/p}{1+\lambda \cdot i/p}$, $N_p = 20$, $p = 3N_p$



- Consider a prediction horizon $N_p \cdot \tau_s$
- Consider the reference on ω_e : $\omega_e^{ref} = max\left\{\omega_e^0, \mathcal{T}^{-1}\left(T_e^d(X_{pedal}, \omega_e)\right)\right\}$
- Consider the reference on the slipping velocity $\omega_{sl} = \omega_e - \omega_c$ defined at each sampling instant $k \cdot \tau_s$:

$$\omega_{sl}^{ref}(k+i, p) = \frac{1 - i/p}{1 + \lambda \cdot i/p} \cdot \omega_{sl}(k)$$

- p monitors the speed of convergence of ω_{sl} to 0

The Parameterized NMPC scheme

Given X_{pedal}, for each choice of $p \in [N_p, \infty]$, a piecewise constant control profile can be defined according to the following unconstrained problem :

$$\mathcal{U}_{pwc}(p,\omega) := arg\min_{\mathbf{u}} \Omega(\mathbf{u},p,\omega) := \sum_{i=1}^{N_p} \left\| \begin{pmatrix} \omega_{sl}(k+i) - \omega_{sl}^{ref}(k+i,p) \\ \omega_e(k+i) - \omega_e^{ref}(k+i) \end{pmatrix} \right\|_Q^2$$

The Parameterized NMPC scheme

Given X_{pedal}, for each choice of $p \in [N_p, \infty]$, a piecewise constant control profile can be defined according to the following unconstrained problem :

$$\mathcal{U}_{pwc}(p,\omega) := arg\min_{\mathbf{u}} \Omega(\mathbf{u},p,\omega) := \sum_{i=1}^{N_p} \left\| \begin{pmatrix} \omega_{sl}(k+i) - \omega_{sl}^{ref}(k+i,p) \\ \omega_e(k+i) - \omega_e^{ref}(k+i) \end{pmatrix} \right\|_Q^2$$

Recall that :

$$\begin{aligned} \omega_e^{ref}(k+i) &= max\left\{\omega_e^0, \mathcal{T}^{-1}\left(T_e^d(X_{pedal}, \omega_e(k+i))\right)\right\} \\ \omega_{sl}^{ref}(k+i,p) &= \frac{1 - i/p}{1 + \lambda \cdot i/p} \cdot \omega_{sl}(k) \end{aligned}$$

The Parameterized NMPC scheme

Given X_{pedal}, for each choice of $p \in [N_p, \infty]$, a piecewise constant control profile can be defined according to the following unconstrained problem :

$$\mathcal{U}_{pwc}(p,\omega) := arg\min_{\mathbf{u}} \Omega(\mathbf{u},p,\omega) := \sum_{i=1}^{N_p} \left\| \begin{pmatrix} \omega_{sl}(k+i) - \omega_{sl}^{ref}(k+i,p) \\ \omega_e(k+i) - \omega_e^{ref}(k+i) \end{pmatrix} \right\|_Q^2$$

Recall that :

$$\begin{aligned} \omega_e^{ref}(k+i) &= max\left\{\omega_e^0, \mathcal{T}^{-1}\left(T_e^d(X_{pedal}, \omega_e(k+i))\right)\right\} \\ \omega_{sl}^{ref}(k+i,p) &= \frac{1 - i/p}{1 + \lambda \cdot i/p} \cdot \omega_{sl}(k) \end{aligned}$$

- For each p and X_{pedal}, $\mathcal{U}_{pwc}(p,\omega)$ can be computed analytically

The Parameterized NMPC scheme

Given X_{pedal}, for each choice of $p \in [N_p, \infty]$, a piecewise constant control profile can be defined according to the following unconstrained problem :

$$\mathcal{U}_{pwc}(p,\omega) := arg\min_{\mathbf{u}} \Omega(\mathbf{u},p,\omega) := \sum_{i=1}^{N_p} \left\| \begin{pmatrix} \omega_{sl}(k+i) - \omega_{sl}^{ref}(k+i,p) \\ \omega_e(k+i) - \omega_e^{ref}(k+i) \end{pmatrix} \right\|_Q^2$$

Recall that :

$$\begin{aligned} \omega_e^{ref}(k+i) &= max\left\{\omega_e^0, \mathcal{T}^{-1}\left(T_e^d(X_{pedal}, \omega_e(k+i))\right)\right\} \\ \omega_{sl}^{ref}(k+i,p) &= \frac{1 - i/p}{1 + \lambda \cdot i/p} \cdot \omega_{sl}(k) \end{aligned}$$

- For each p and X_{pedal}, $\mathcal{U}_{pwc}(p,\omega)$ can be computed analytically
- For sufficiently high p (slow convergence), constraints can be fulfilled

The Parameterized NMPC scheme

Given X_{pedal}, for each choice of $p \in [N_p, \infty]$, a piecewise constant control profile can be defined according to the following unconstrained problem :

$$\mathcal{U}_{pwc}(p,\omega) := arg\min_{\mathbf{u}} \Omega(\mathbf{u},p,\omega) := \sum_{i=1}^{N_p} \left\| \begin{pmatrix} \omega_{sl}(k+i) - \omega_{sl}^{ref}(k+i,p) \\ \omega_e(k+i) - \omega_e^{ref}(k+i) \end{pmatrix} \right\|_Q^2$$

Recall that :

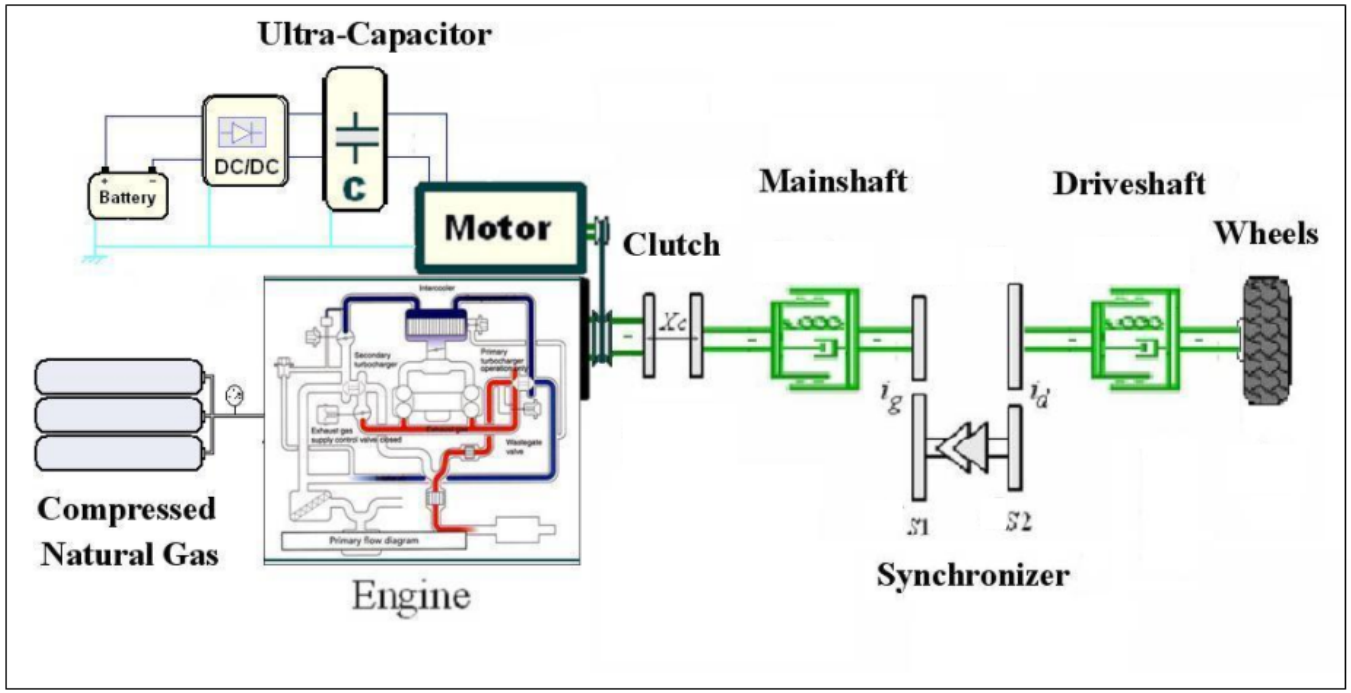
$$\begin{aligned} \omega_e^{ref}(k+i) &= max\left\{\omega_e^0, \mathcal{T}^{-1}\left(T_e^d(X_{pedal}, \omega_e(k+i))\right)\right\} \\ \omega_{sl}^{ref}(k+i,p) &= \frac{1-i/p}{1+\lambda \cdot i/p} \cdot \omega_{sl}(k) \end{aligned}$$

- For each p and X_{pedal}, $\mathcal{U}_{pwc}(p,\omega)$ can be computed analytically
- For sufficiently high p (slow convergence), constraints can be fulfilled

A scalar NMPC with the optimization problem :

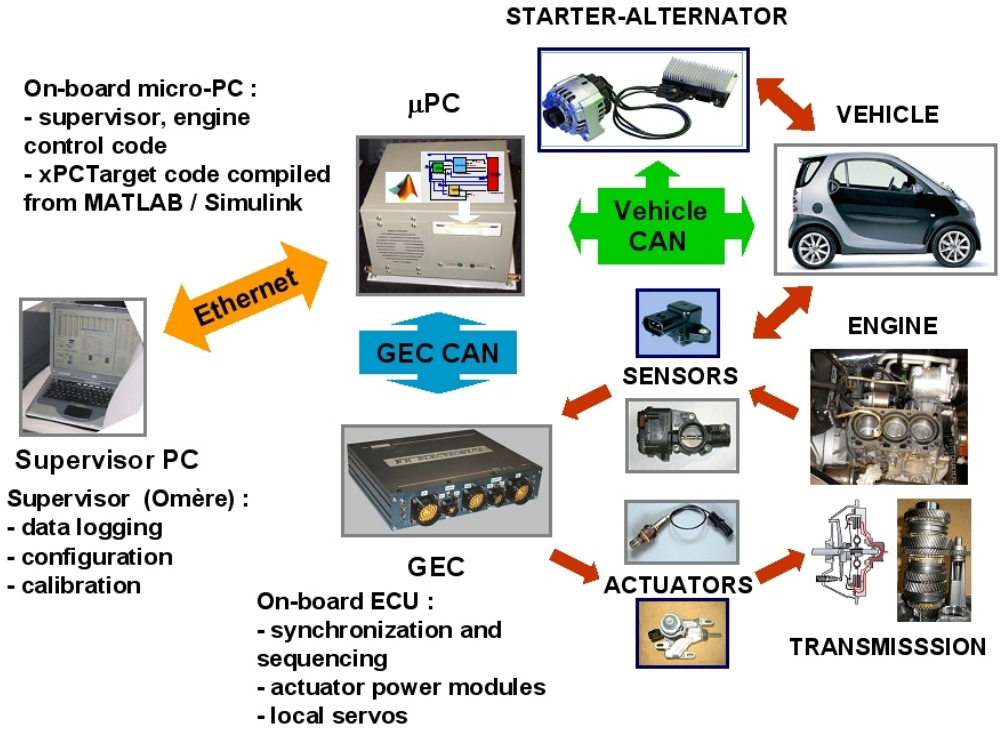
$$\min_{p \in [N_p, N_p^{max}]} J(p,\omega) = |p - t_f(X_{pedal})| \quad \text{under saturation constraints} \quad [C(p,\omega) \leq 0]$$

The IFP's SMART demo architecture (1)



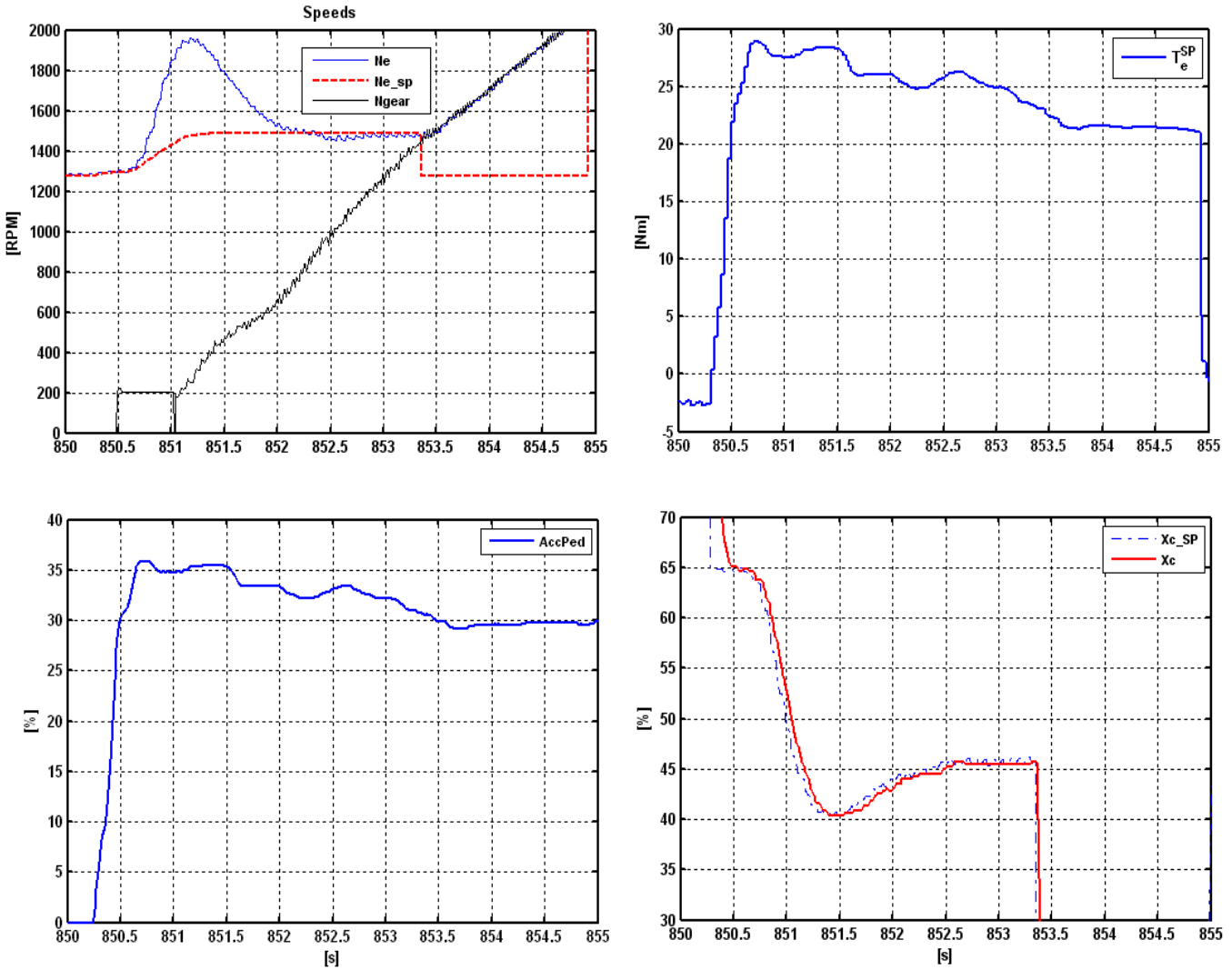
The mild-Hybrid powertrain of the VEHEGAN demo-car

The IFP's SMART demo architecture (2)



VEHEGAN on-board Control System

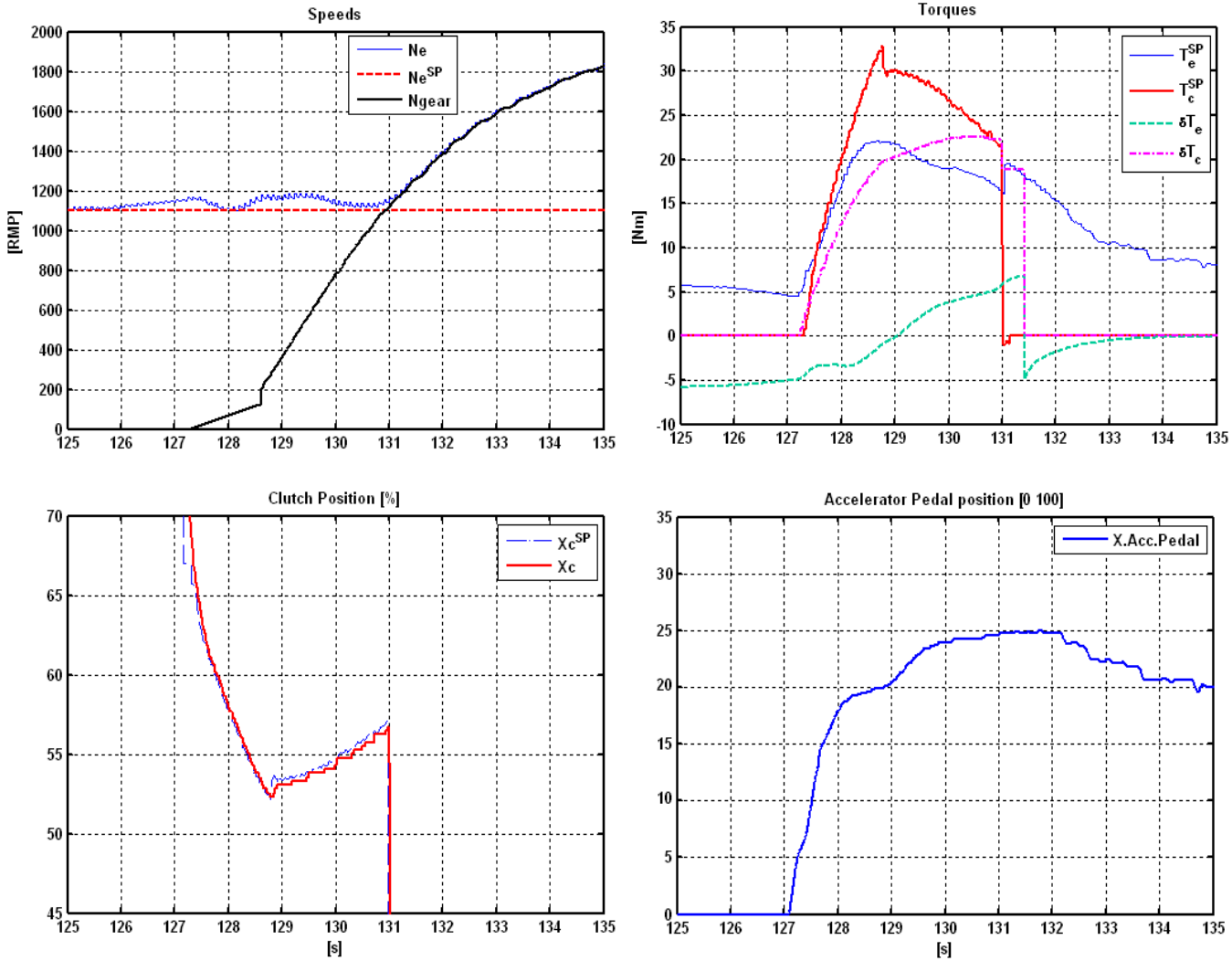
Comparison with pre-existing PID Controller (Fast start-up)



PID Controller

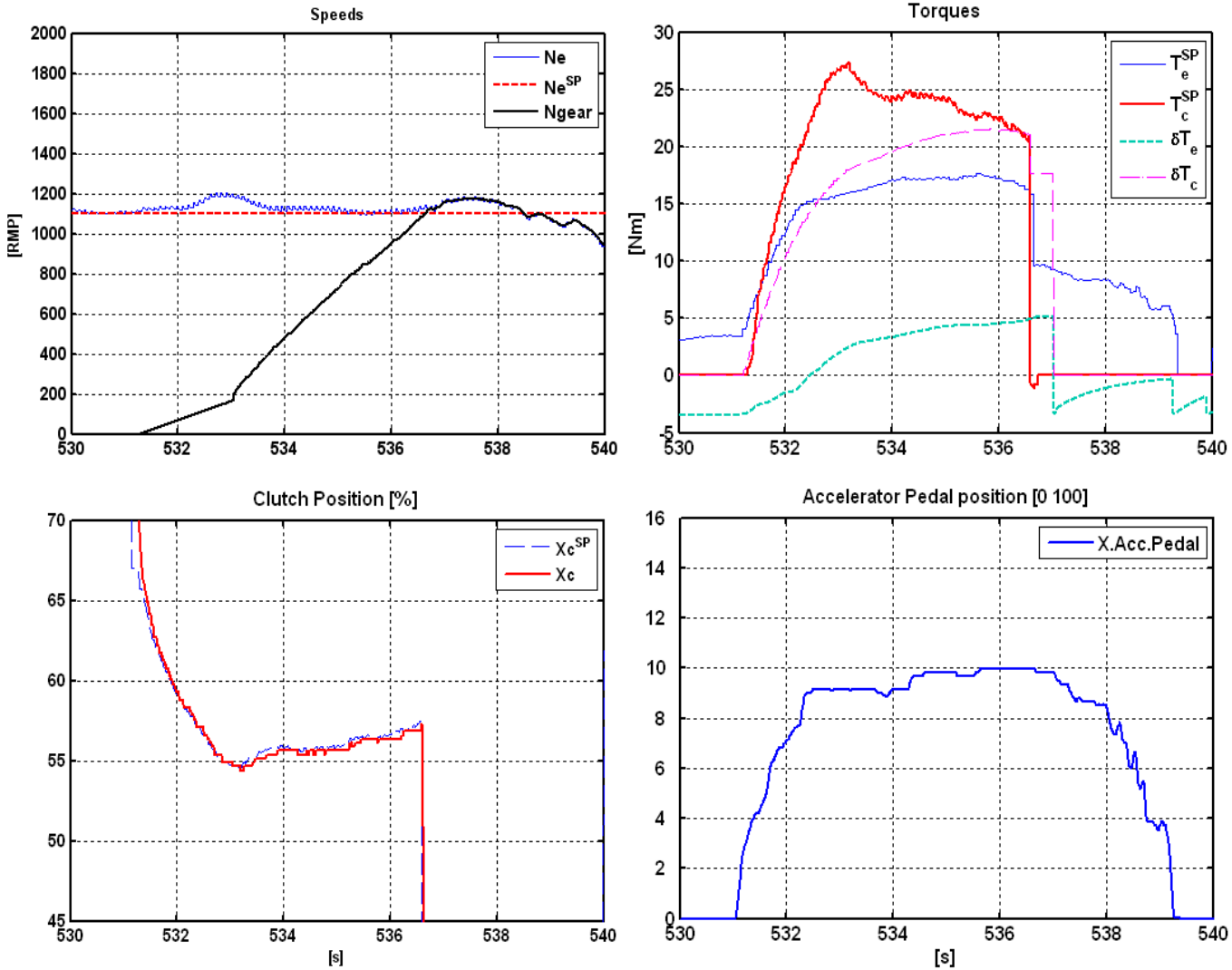
gipsa-lab

Comparison with pre-existing PID Controller (Fast start-up)



Parameterized NMPC

Comparison between slow and fast start-up

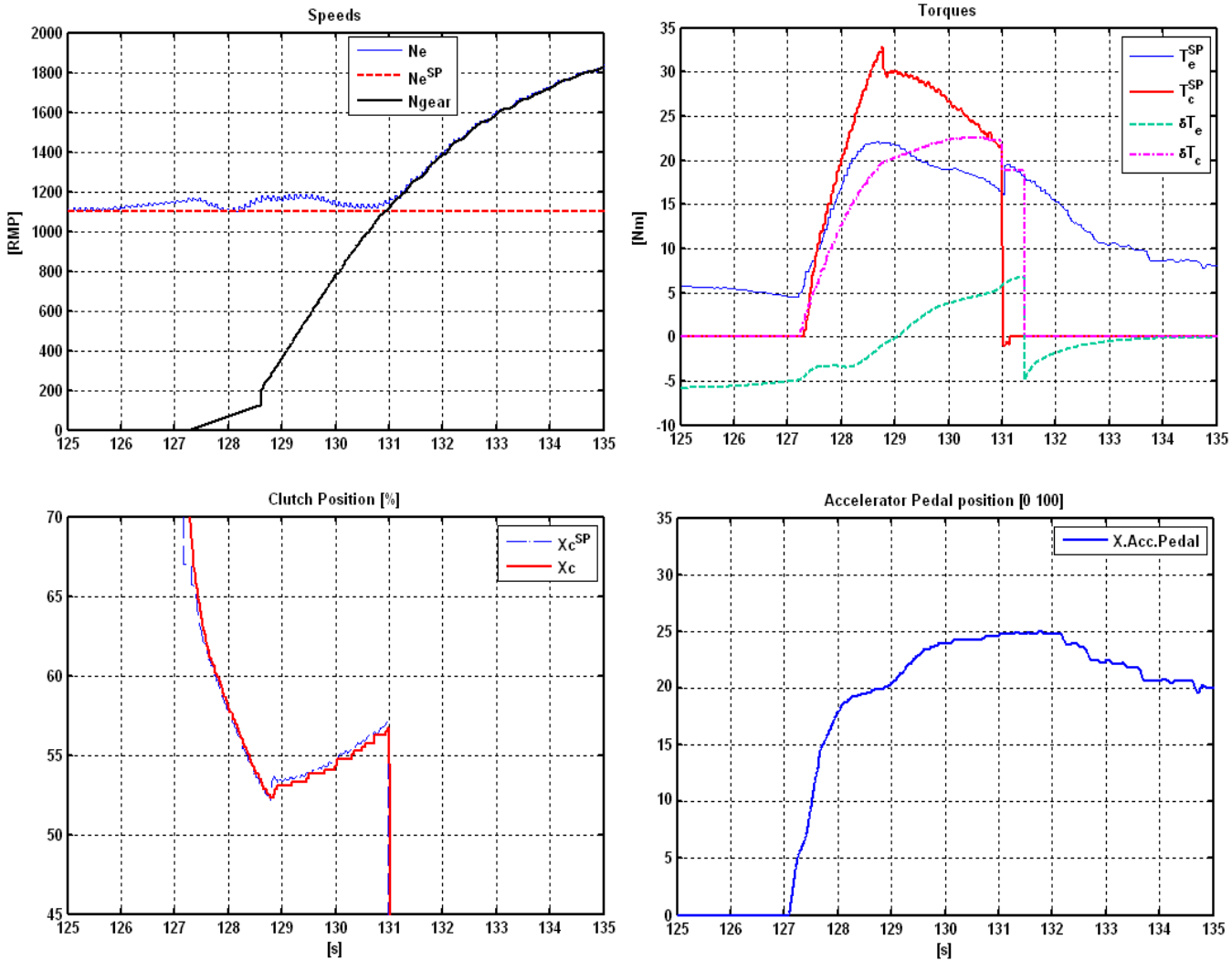


Parameterized NMPC - slow start-up

(The MPC scheme controls both the engine idle speed and the start-up phase)

gipsa-lab

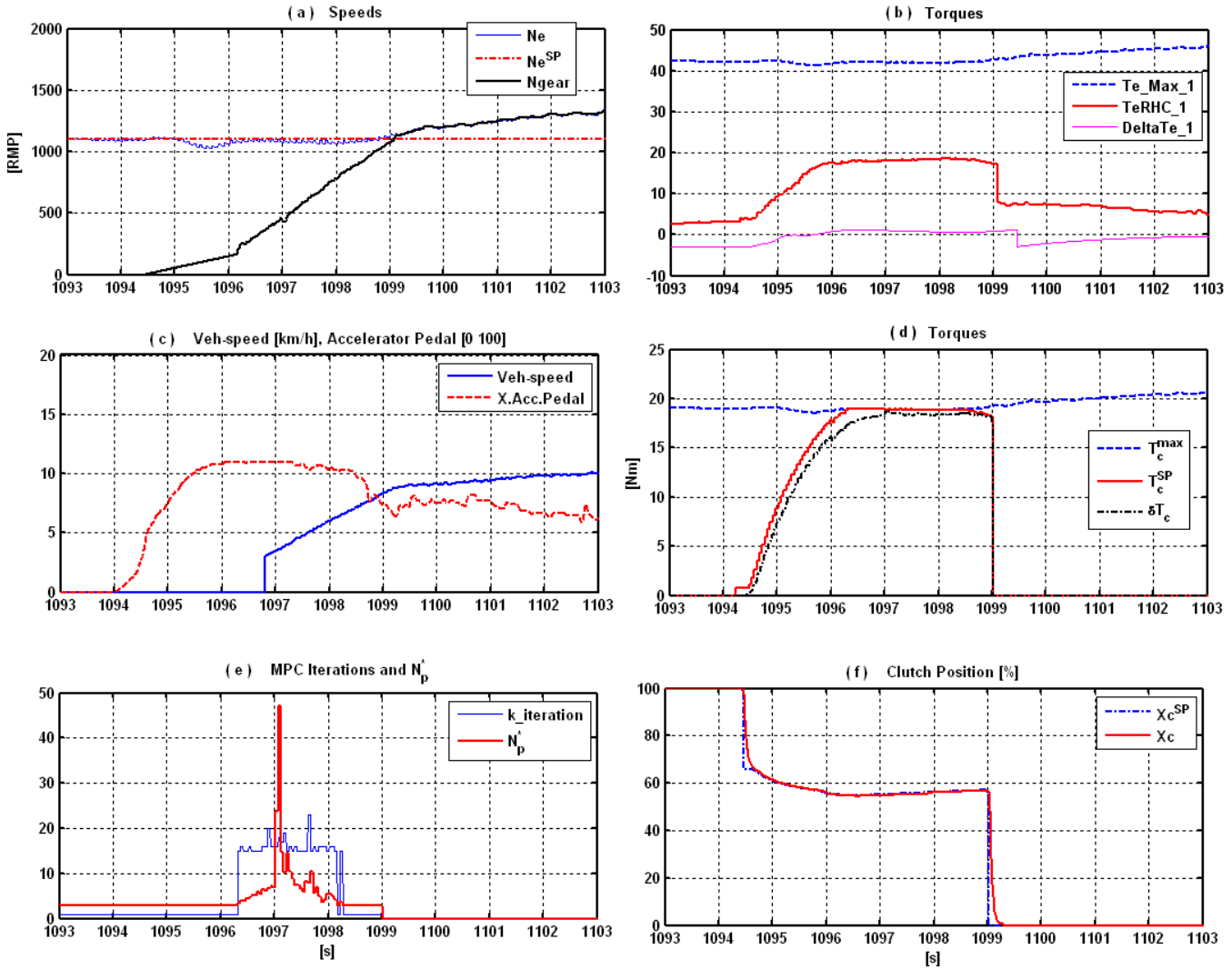
Comparison between slow and fast start-up



Parameterized NMPC - fast start-up

(The MPC scheme controls both the engine idle speed and the start-up phase)

Behavior Under Saturation



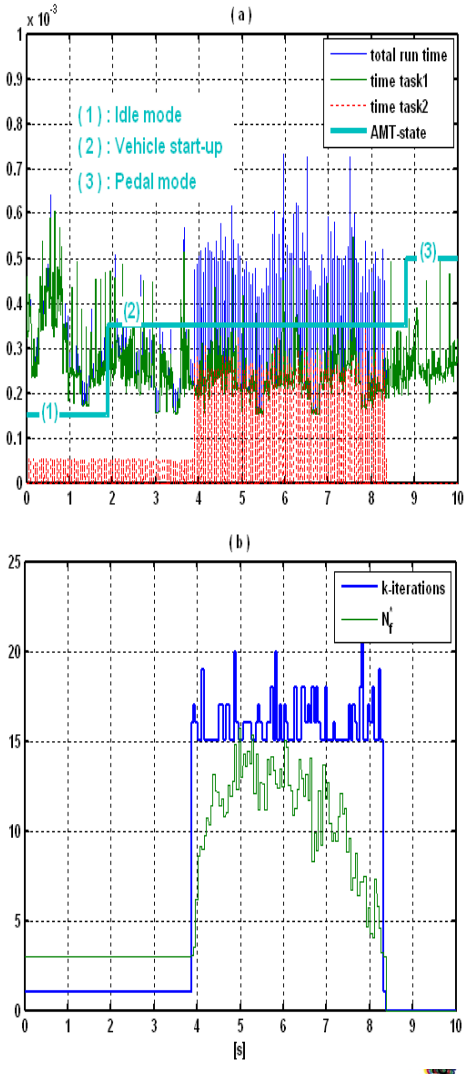
Parameterized NMPC - Actuators saturation

(Note the closed-loop evolution of the parameter $p =: N_p^*$)

gipsa-lab

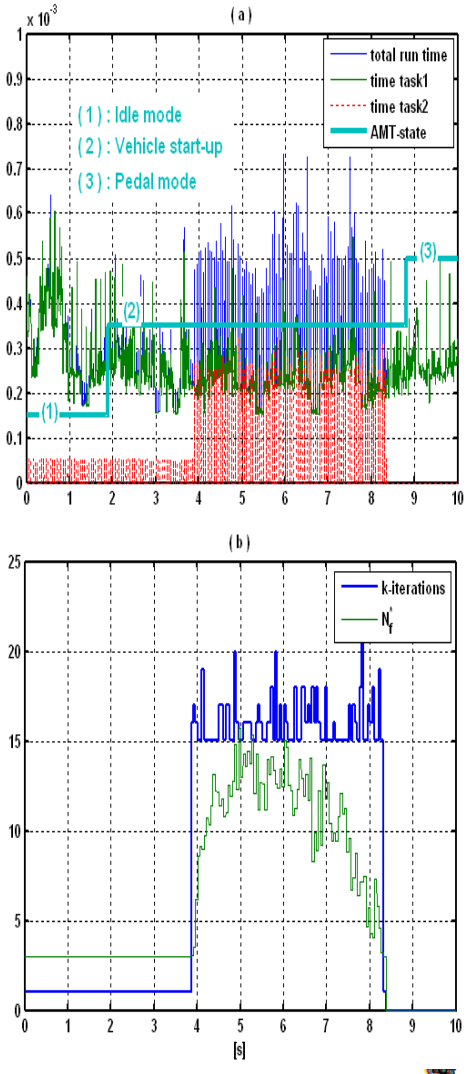
Some Real-Time Implementation Issues

- Matlab/Simulink, xPC Target



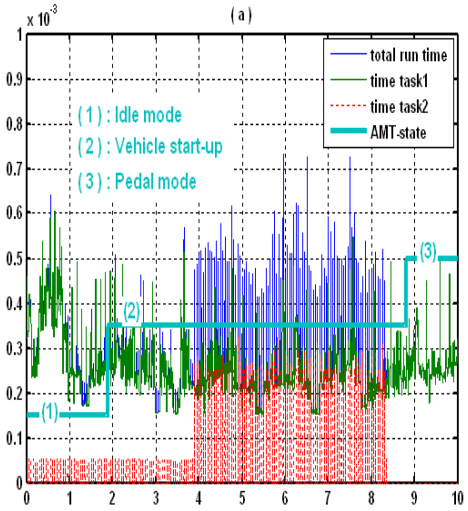
Some Real-Time Implementation Issues

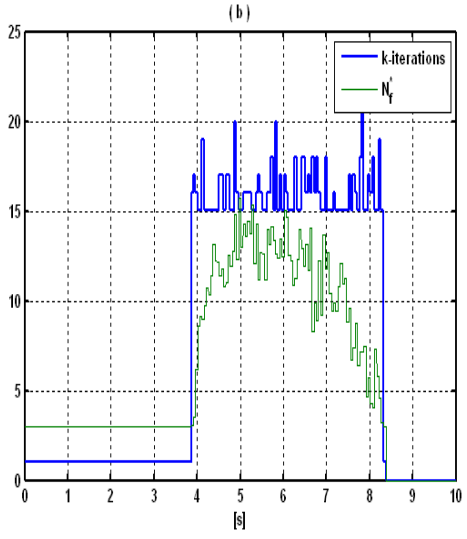
- Matlab/Simulink, xPC Target
- Pentium III, 256 Mb (RAM)



Some Real-Time Implementation Issues

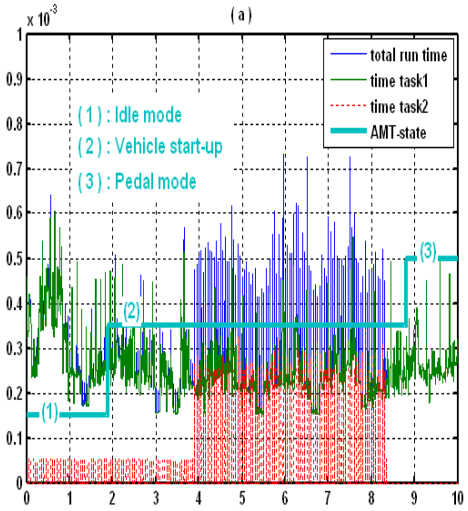
- Matlab/Simulink, xPC Target
- Pentium III, 256 Mb (RAM)
- Only 0.3Mb for Control Software

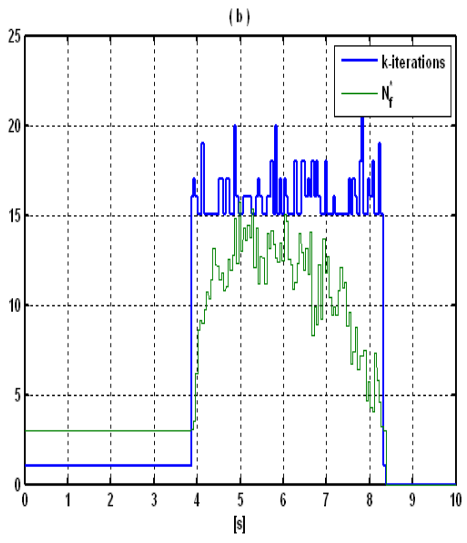




Some Real-Time Implementation Issues

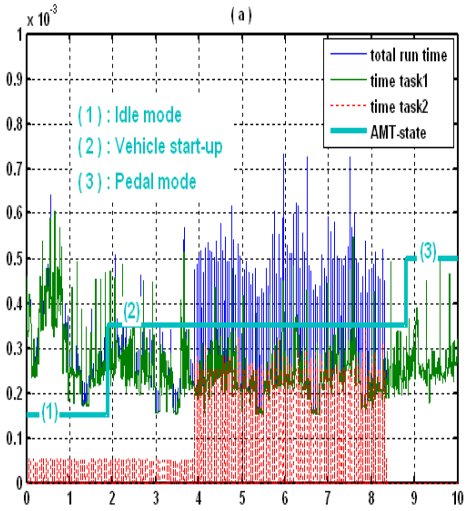
- Matlab/Simulink, xPC Target
- Pentium III, 256 Mb (RAM)
- Only 0.3Mb for Control Software
- Base sampling time 1 ms

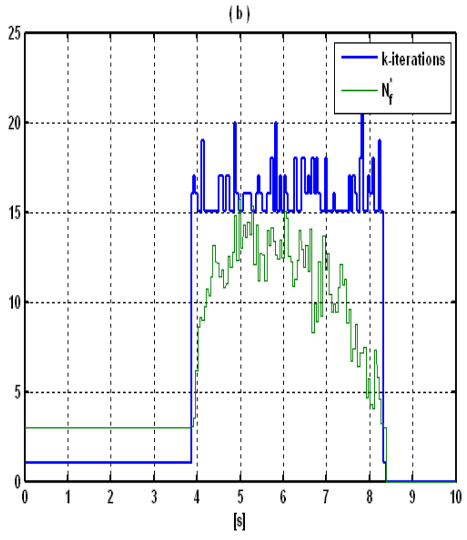




Some Real-Time Implementation Issues

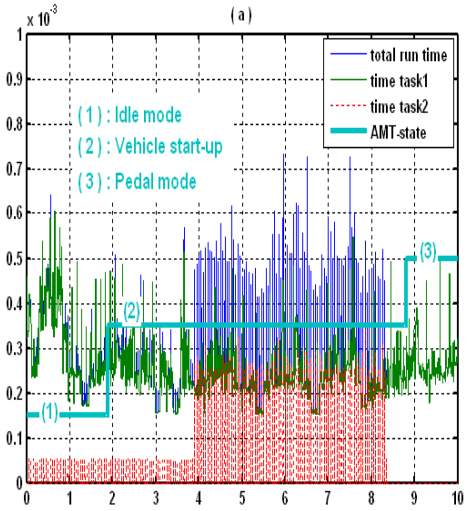
- Matlab/Simulink, xPC Target
- Pentium III, 256 Mb (RAM)
- Only 0.3Mb for Control Software
- Base sampling time 1 ms
- Control blocks are executed synchronously with cylinder top dead center

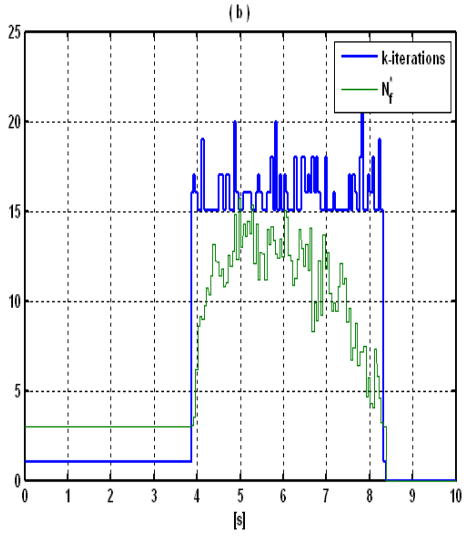




Some Real-Time Implementation Issues

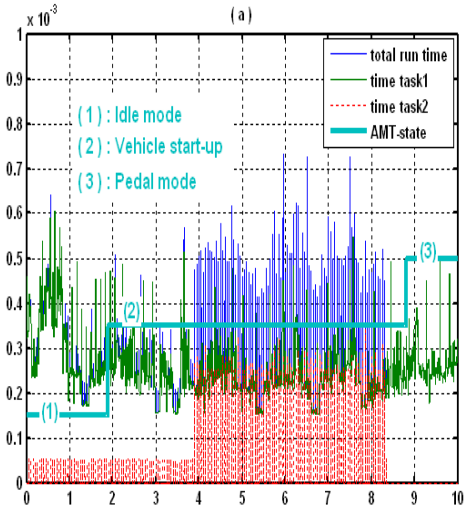
- Matlab/Simulink, xPC Target
- Pentium III, 256 Mb (RAM)
- Only 0.3Mb for Control Software
- Base sampling time 1 ms
- Control blocks are executed synchronously with cylinder top dead center
- variable sampling time $\in [6\ ms, 50\ ms]$

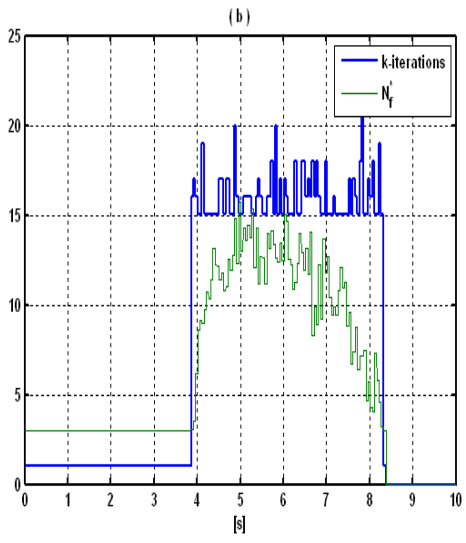




Some Real-Time Implementation Issues

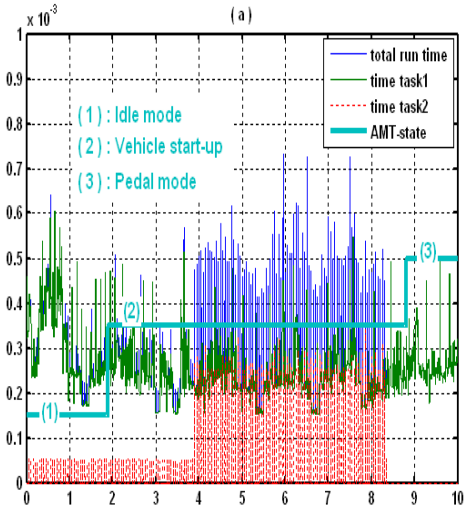
- Matlab/Simulink, xPC Target
- Pentium III, 256 Mb (RAM)
- Only 0.3Mb for Control Software
- Base sampling time 1 ms
- Control blocks are executed synchronously with cylinder top dead center
- variable sampling time $\in [6\ ms, 50\ ms]$
- transmission control blocks are executed at 10 ms

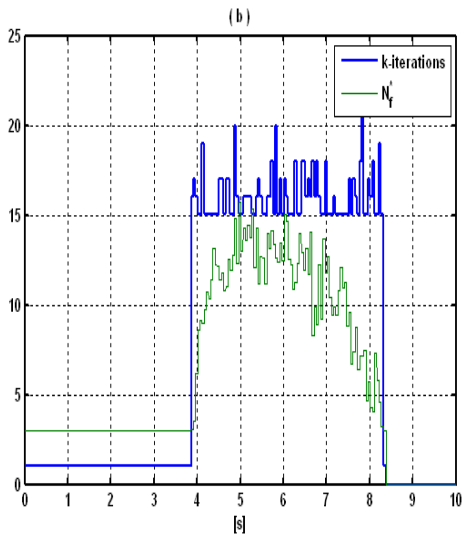




Some Real-Time Implementation Issues

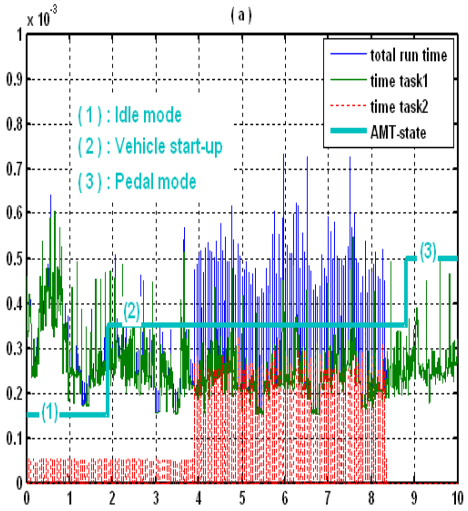
- Matlab/Simulink, xPC Target
- Pentium III, 256 Mb (RAM)
- Only 0.3Mb for Control Software
- Base sampling time 1 ms
- Control blocks are executed synchronously with cylinder top dead center
- variable sampling time $\in [6\ ms, 50\ ms]$
- transmission control blocks are executed at 10 ms
- Solution for p by dichotomy

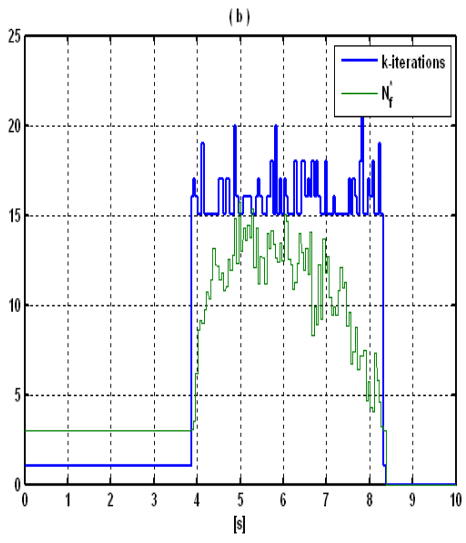




Some Real-Time Implementation Issues

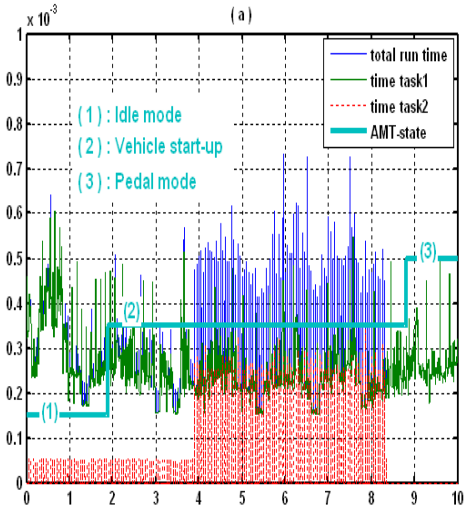
- Matlab/Simulink, xPC Target
- Pentium III, 256 Mb (RAM)
- Only 0.3Mb for Control Software
- Base sampling time 1 ms
- Control blocks are executed synchronously with cylinder top dead center
- variable sampling time $\in [6\ ms, 50\ ms]$
- transmission control blocks are executed at 10 ms
- Solution for p by dichotomy
- Multi-tasks config (control : task2, other task1)

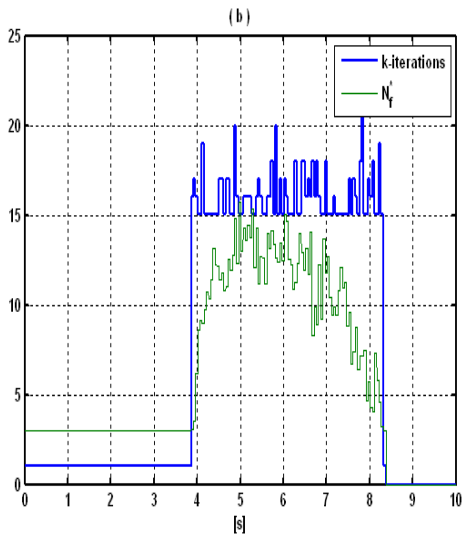




Some Real-Time Implementation Issues

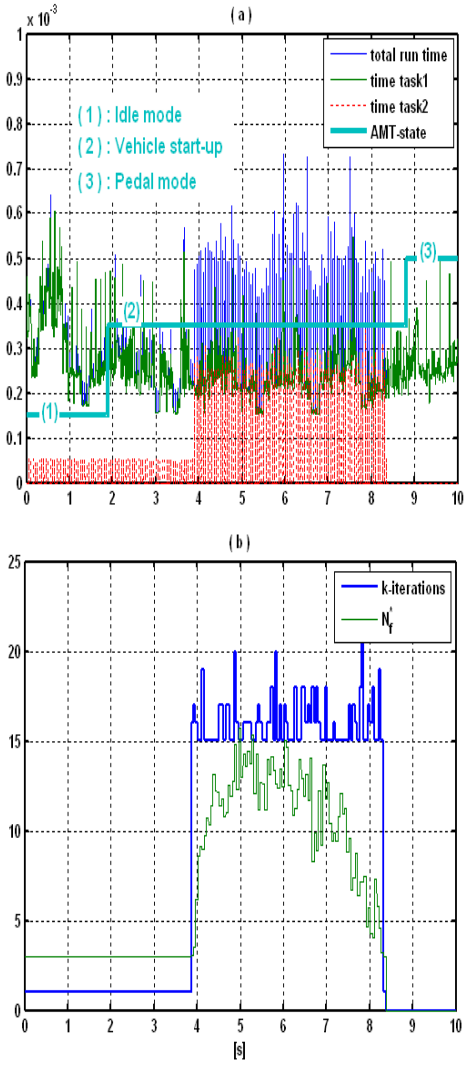
- Matlab/Simulink, xPC Target
- Pentium III, 256 Mb (RAM)
- Only 0.3Mb for Control Software
- Base sampling time 1 ms
- Control blocks are executed synchronously with cylinder top dead center
- variable sampling time $\in [6\ ms, 50\ ms]$
- transmission control blocks are executed at 10 ms
- Solution for p by dichotomy
- Multi-tasks config (control : task2, other task1)
- Time for preparation + 1 iteration $\approx 0.05\ ms$





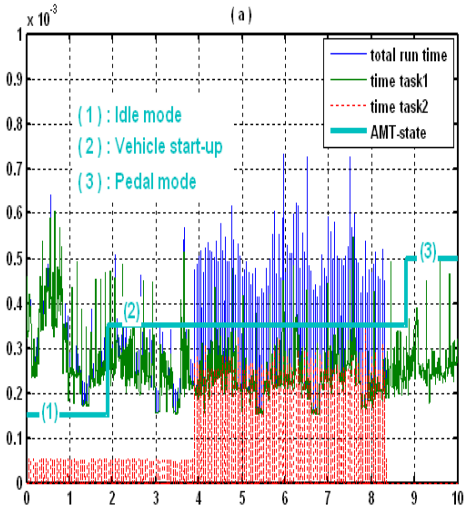
Some Real-Time Implementation Issues

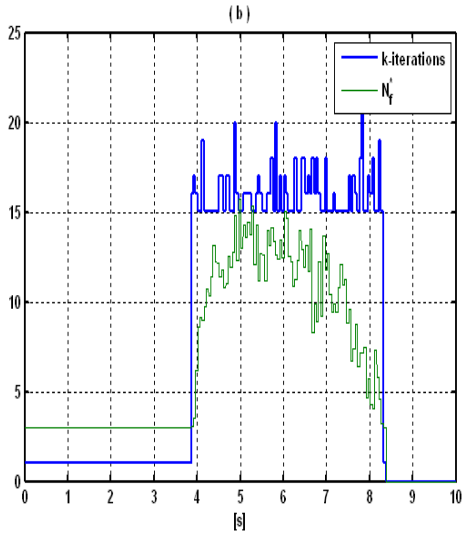
- Matlab/Simulink, xPC Target
- Pentium III, 256 Mb (RAM)
- Only 0.3Mb for Control Software
- Base sampling time 1 ms
- Control blocks are executed synchronously with cylinder top dead center
- variable sampling time $\in [6\ ms, 50\ ms]$
- transmission control blocks are executed at 10 ms
- Solution for p by dichotomy
- Multi-tasks config (control : task2, other task1)
- Time for preparation + 1 iteration $\approx 0.05\ ms$
- maximum (21 iterations) during saturation



Some Real-Time Implementation Issues

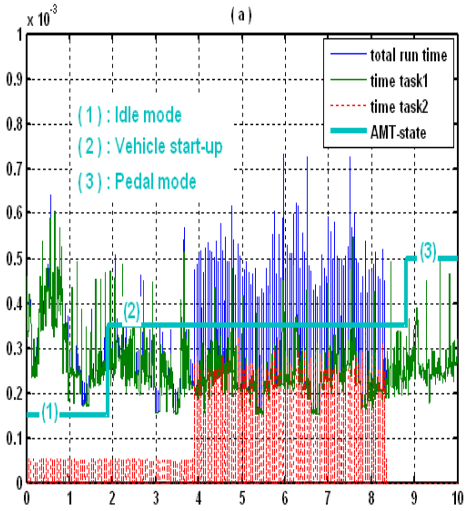
- Matlab/Simulink, xPC Target
- Pentium III, 256 Mb (RAM)
- Only 0.3Mb for Control Software
- Base sampling time 1 ms
- Control blocks are executed synchronously with cylinder top dead center
- variable sampling time $\in [6\ ms, 50\ ms]$
- transmission control blocks are executed at 10 ms
- Solution for p by dichotomy
- Multi-tasks config (control : task2, other task1)
- Time for preparation + 1 iteration $\approx 0.05\ ms$
- maximum (21 iterations) during saturation
- Control is computed each 50 ms

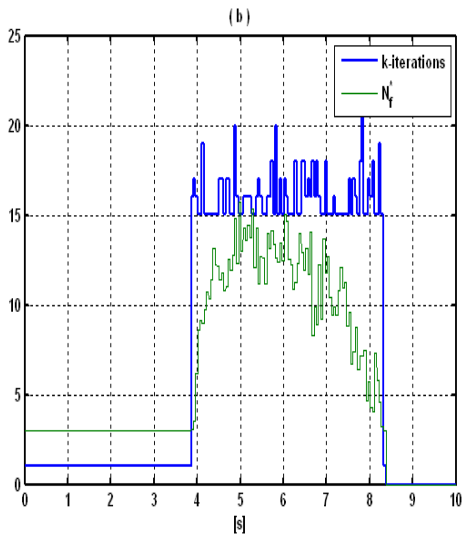




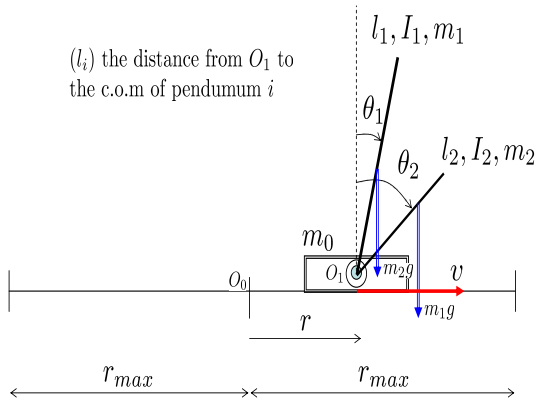
Some Real-Time Implementation Issues

- Matlab/Simulink, xPC Target
- Pentium III, 256 Mb (RAM)
- Only 0.3Mb for Control Software
- Base sampling time 1 ms
- Control blocks are executed synchronously with cylinder top dead center
- variable sampling time $\in [6\ ms, 50\ ms]$
- transmission control blocks are executed at 10 ms
- Solution for p by dichotomy
- Multi-tasks config (control : task2, other task1)
- Time for preparation + 1 iteration $\approx 0.05\ ms$
- maximum (21 iterations) during saturation
- Control is computed each 50 ms
- Sampling can be drastically reduced ($< 1\ ms$)





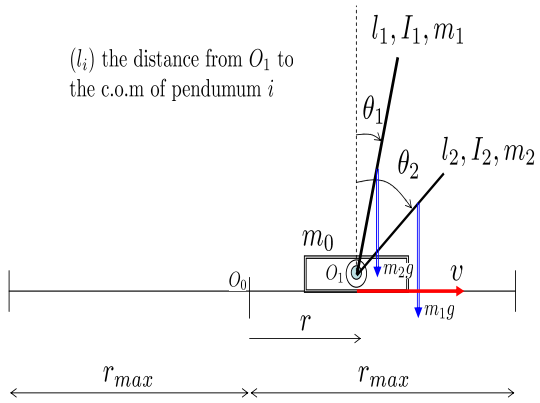
The twin pendulum Example

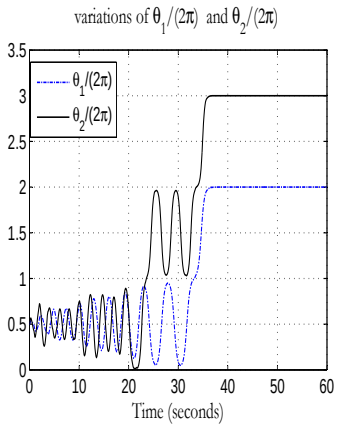


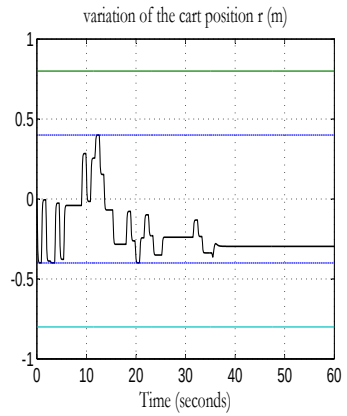
- Dual mode contractive scheme
- 3 integrations / sampling period [$card(\mathbb{P} = 3)$]
- Control and state constraints

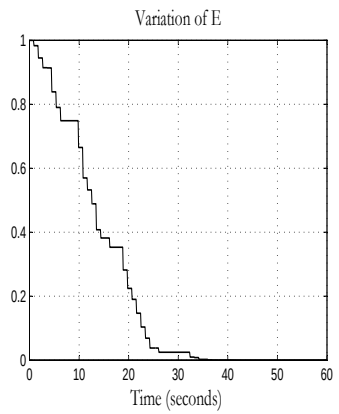
[Alamir, M. and Murilo, A. Swing-up and stabilization of a Twin-Pendulum under state and control constraints by fast NMPC scheme. Automatica, Vol. 44, pp. 1319-1324, (2008).]

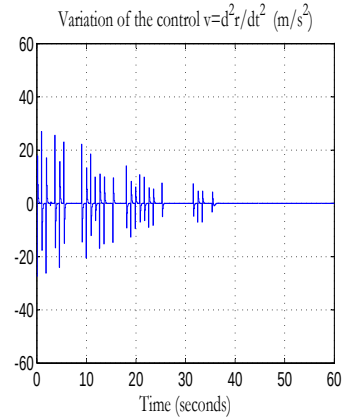
The twin pendulum Example







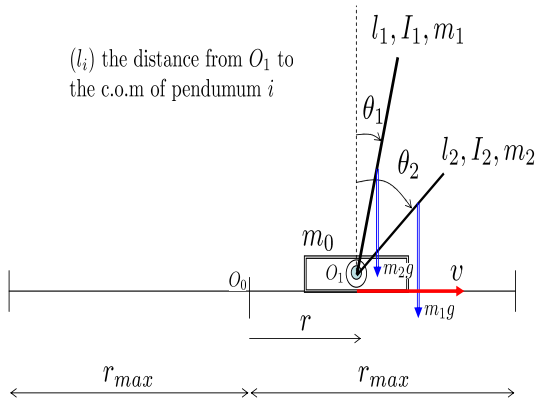


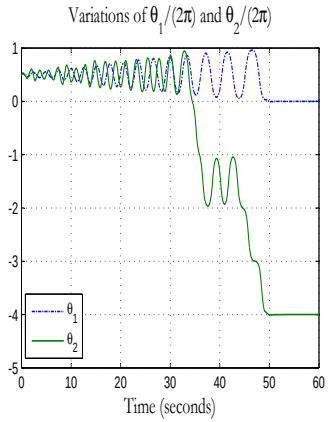


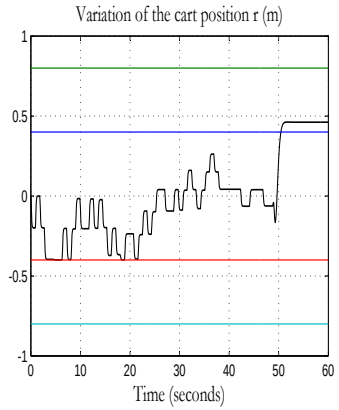
- Dual mode contractive scheme
- 3 integrations / sampling period [$card(\mathbb{P} = 3)$]
- Control and state constraints

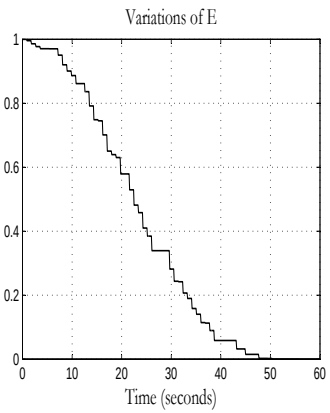
[Alamir, M. and Murilo, A. Swing-up and stabilization of a Twin-Pendulum under state and control constraints by fast NMPC scheme. Automatica, Vol. 44, pp. 1319-1324, (2008).]

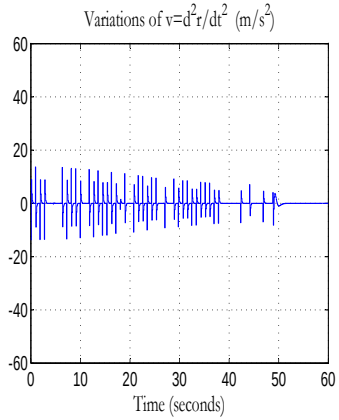
The twin pendulum Example









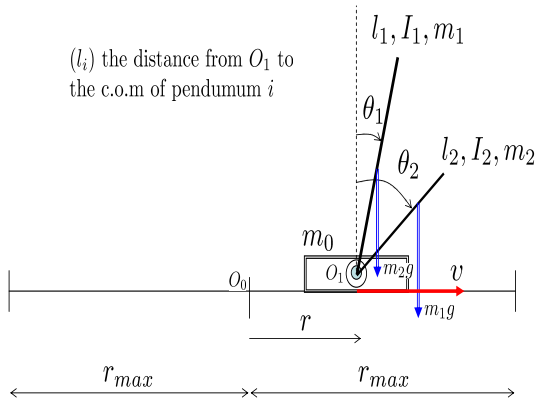


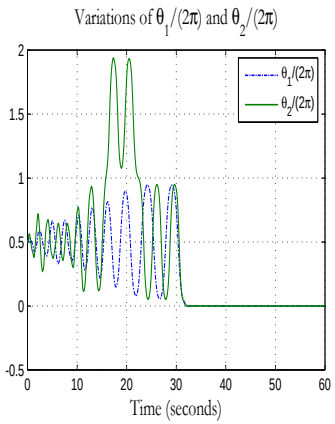
- Dual mode contractive scheme
- 3 integrations / sampling period [$card(\mathbb{P} = 3)$]
- Control and state constraints

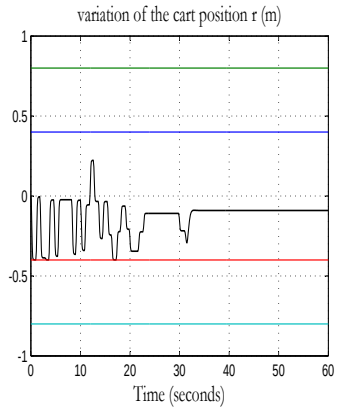
[Alamir, M. and Murilo, A. Swing-up and stabilization of a Twin-Pendulum under state and control constraints by fast NMPC scheme. Automatica, Vol. 44, pp. 1319-1324, (2008).]

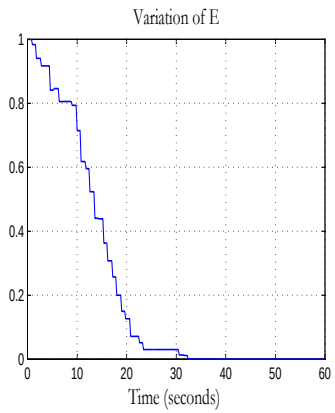
gipsa-lab

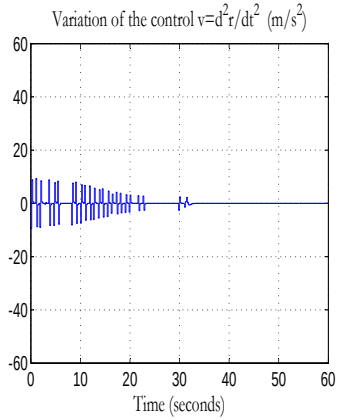
The twin pendulum Example









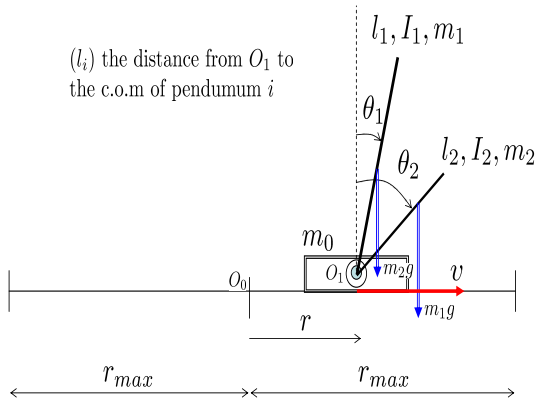


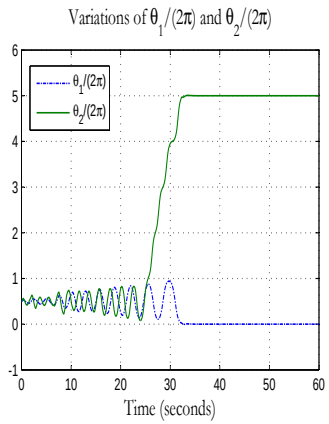
- Dual mode contractive scheme
- 3 integrations / sampling period [$card(\mathbb{P} = 3)$]
- Control and state constraints

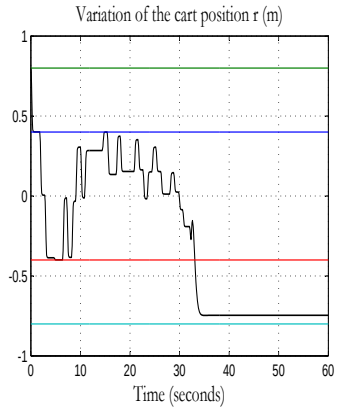
[Alamir, M. and Murilo, A. Swing-up and stabilization of a Twin-Pendulum under state and control constraints by fast NMPC scheme. Automatica, Vol. 44, pp. 1319-1324, (2008).]

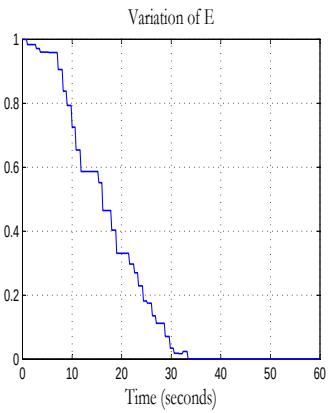
gipsa-lab

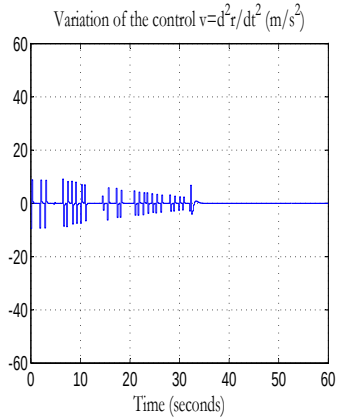
The twin pendulum Example









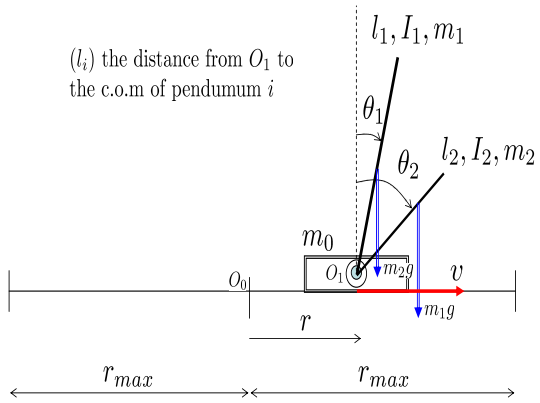


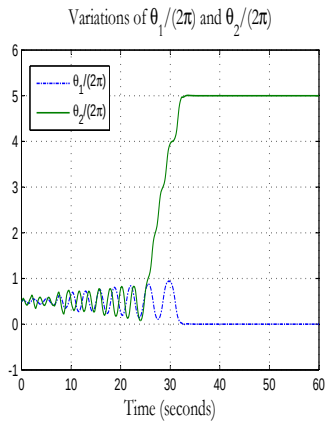
- Dual mode contractive scheme
- 3 integrations / sampling period [$card(\mathbb{P} = 3)$]
- Control and state constraints

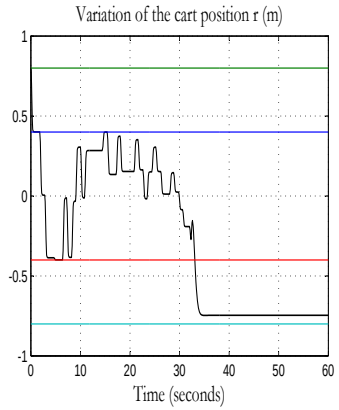
[Alamir, M. and Murilo, A. Swing-up and stabilization of a Twin-Pendulum under state and control constraints by fast NMPC scheme. Automatica, Vol. 44, pp. 1319-1324, (2008).]

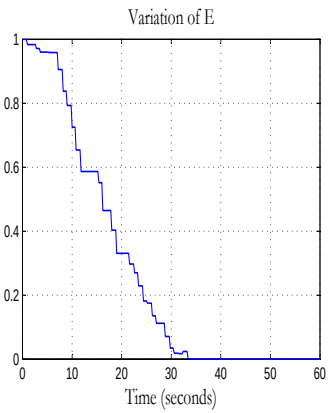
gipsa-lab

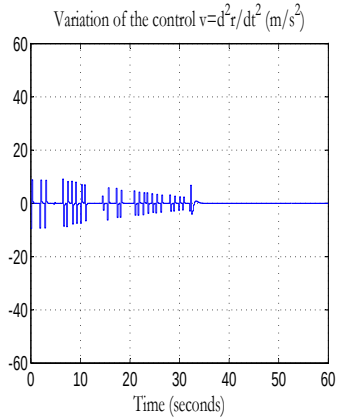
The twin pendulum Example









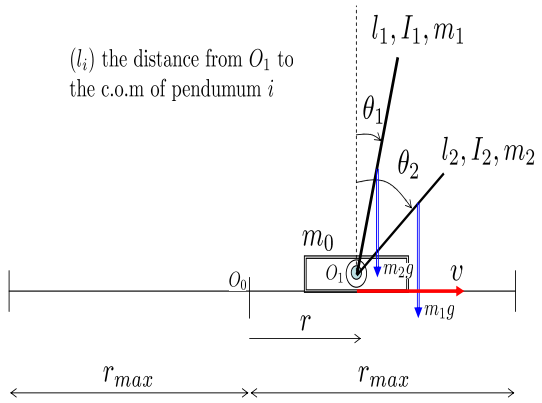


- Dual mode contractive scheme
- 3 integrations / sampling period [$card(\mathbb{P} = 3)$]
- Control and state constraints

[Alamir, M. and Murilo, A. Swing-up and stabilization of a Twin-Pendulum under state and control constraints by fast NMPC scheme. Automatica, Vol. 44, pp. 1319-1324, (2008).]

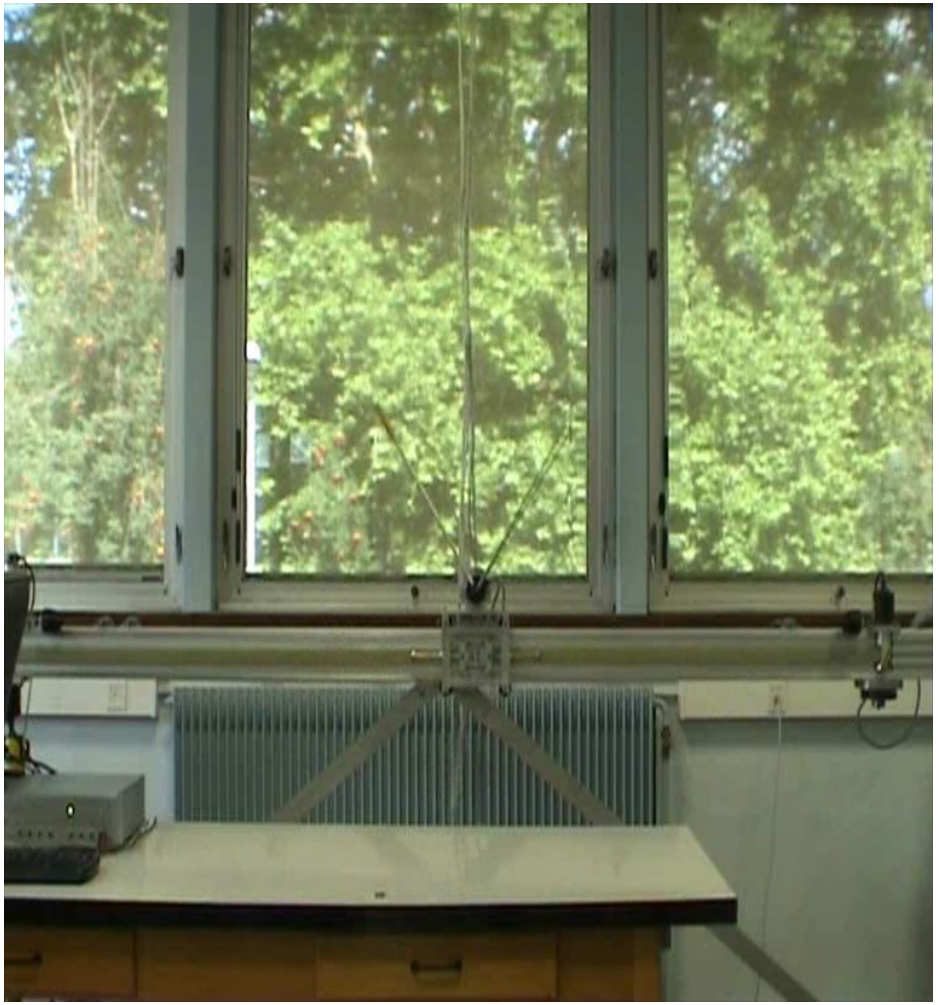
gipsa-lab

The twin pendulum Example



- Dual mode contractive scheme
- 3 integrations / sampling period [$card(\mathbb{P} = 3)$]
- Control and state constraints

[Alamir, M. and Murilo, A. Swing-up and stabilization of a Twin-Pendulum under state and control constraints by fast NMPC scheme. Automatica, Vol. 44, pp. 1319-1324, (2008).]



Conclusion

- Few problems resist to an NMPC-based solution

Conclusion

- Few problems resist to an NMPC-based solution
- The major obstacle is the computation time

Conclusion

- Few problems resist to an NMPC-based solution
- The major obstacle is the computation time
- Fast system oriented solution is an active research topics

Conclusion

- Few problems resist to an NMPC-based solution
- The major obstacle is the computation time
- Fast system oriented solution is an active research topics
- Efficient solution need problem dedicated parametrization

Conclusion

- Few problems resist to an NMPC-based solution
- The major obstacle is the computation time
- Fast system oriented solution is an active research topics
- Efficient solution need problem dedicated parametrization
- Space applications are definitely inside the feasibility domain of NMPC

Sufficient Conditions for Stability (but not necessary)

Necessary conditions for well defined and stable NMPC scheme

$$\mathcal{P}(x) \; : \; \min_{\mathbf{u}} \left\{ V(x,\mathbf{u}) \quad | \quad \mathbf{u} \in \mathcal{U}(x) \right\} \quad ; \quad x^+ = f(x,u)$$

$$\mathcal{U}(x) := \left\{ \mathbf{u} \in \mathbb{U}^N \mid \forall i \quad x^{\mathbf{u}}(i;x) \in \mathbb{X} \quad \text{and} \quad x^{\mathbf{u}}(N,x) \in X_f \right\}$$

$$V(x,\mathbf{u}) = F(x(N;x)) + \sum_{i=0}^{N} L(x^{\mathbf{u}}(i;x), u(i))$$

Sufficient Conditions for Stability (but not necessary)

Necessary conditions for well defined and stable NMPC scheme

$$\mathcal{P}(x) \; : \; \min_{\mathbf{u}} \left\{ V(x,\mathbf{u}) \quad | \quad \mathbf{u} \in \mathcal{U}(x) \right\} \quad ; \quad x^+ = f(x,u)$$

$$\mathcal{U}(x) := \left\{ \mathbf{u} \in \mathbb{U}^N \mid \forall i \quad x^{\mathbf{u}}(i;x) \in \mathbb{X} \quad \text{and} \quad x^{\mathbf{u}}(N,x) \in X_f \right\}$$

$$V(x,\mathbf{u}) = F(x(N;x)) + \sum_{i=0}^{N} L(x^{\mathbf{u}}(i;x), u(i))$$

Condition 1 : Continuity

The applications f, F, L are continuous in their arguments.

Sufficient Conditions for Stability (but not necessary)

Necessary conditions for well defined and stable NMPC scheme

$$\mathcal{P}(x) \; : \; \min_{\mathbf{u}} \left\{ V(x,\mathbf{u}) \quad | \quad \mathbf{u} \in \mathcal{U}(x) \right\} \quad ; \quad x^+ = f(x,u)$$

$$\mathcal{U}(x) := \left\{ \mathbf{u} \in \mathbb{U}^N \mid \forall i \quad x^{\mathbf{u}}(i;x) \in \mathbb{X} \quad \text{and} \quad x^{\mathbf{u}}(N,x) \in X_f \right\}$$

$$V(x,\mathbf{u}) = F(x(N;x)) + \sum_{i=0}^{N} L(x^{\mathbf{u}}(i;x), u(i))$$

Condition 2 : Compactness

- ✓ $\mathbb{U}$ is compact
- ✓ $\mathbb{X}$ and $X_f \subset \mathbb{X}$ are closed.

Sufficient Conditions for Stability (but not necessary)

Necessary conditions for well defined and stable NMPC scheme

$$\mathcal{P}(x) \; : \; \min_{\mathbf{u}} \left\{ V(x,\mathbf{u}) \quad | \quad \mathbf{u} \in \mathcal{U}(x) \right\} \quad ; \quad x^+ = f(x,u)$$

$$\mathcal{U}(x) := \left\{ \mathbf{u} \in \mathbb{U}^N \mid \forall i \quad x^{\mathbf{u}}(i;x) \in \mathbb{X} \quad \text{and} \quad x^{\mathbf{u}}(N,x) \in X_f \right\}$$

$$V(x,\mathbf{u}) = F(x(N;x)) + \sum_{i=0}^{N} L(x^{\mathbf{u}}(i;x), u(i))$$

Condition 3 : Detectability

The integral cost L must be such that

$$\left\{ L(x,u) \to 0 \right\} \quad \Rightarrow \quad \left\{ x \to 0 \right\}$$

Sufficient Conditions for Stability (but not necessary)

Necessary conditions for well defined and stable NMPC scheme

$$\mathcal{P}(x) \; : \; \min_{\mathbf{u}} \left\{ V(x,\mathbf{u}) \quad | \quad \mathbf{u} \in \mathcal{U}(x) \right\} \quad ; \quad x^+ = f(x,u)$$

$$\mathcal{U}(x) := \left\{ \mathbf{u} \in \mathbb{U}^N \mid \forall i \quad x^{\mathbf{u}}(i;x) \in \mathbb{X} \quad \text{and} \quad x^{\mathbf{u}}(N,x) \in X_f \right\}$$

$$V(x,\mathbf{u}) = F(x(N;x)) + \sum_{i=0}^{N} L(x^{\mathbf{u}}(i;x), u(i))$$

Condition 4 : X_f is positively invariant under some local $\kappa^f(\cdot)$

For all $\xi \in X_f$, there exists an admissible control $\kappa^f(\xi)$ such that :

$$f(\xi, \kappa^f(\xi)) \in X_f$$

psa-lab

Sufficient Conditions for Stability (but not necessary)

Necessary conditions for well defined and stable NMPC scheme

$$\mathcal{P}(x) \; : \; \min_{\mathbf{u}} \left\{ V(x,\mathbf{u}) \quad | \quad \mathbf{u} \in \mathcal{U}(x) \right\} \quad ; \quad x^+ = f(x,u)$$

$$\mathcal{U}(x) := \left\{ \mathbf{u} \in \mathbb{U}^N \mid \forall i \quad x^{\mathbf{u}}(i;x) \in \mathbb{X} \quad \text{and} \quad x^{\mathbf{u}}(N,x) \in X_f \right\}$$

$$V(x,\mathbf{u}) = F(x(N;x)) + \sum_{i=0}^{N} L(x^{\mathbf{u}}(i;x), u(i))$$

Condition 5 : The terminal cost $F(\cdot)$ is a Lyapunov under $\kappa^f(\cdot)$

For all $\xi \in X_f$, the admissible control $\kappa^f(\xi)$ is such that :

$$F(f(\xi, \kappa^f(\xi)) - F(\xi) \leq -L(\xi, \kappa^f(\xi))$$

psa-lab

Sufficient Conditions for Stability (but not necessary)

Necessary conditions for well defined and stable NMPC scheme

$$\mathcal{P}(x) \; : \; \min_{\mathbf{u}} \left\{ V(x,\mathbf{u}) \quad | \quad \mathbf{u} \in \mathcal{U}(x) \right\} \quad ; \quad x^+ = f(x,u)$$

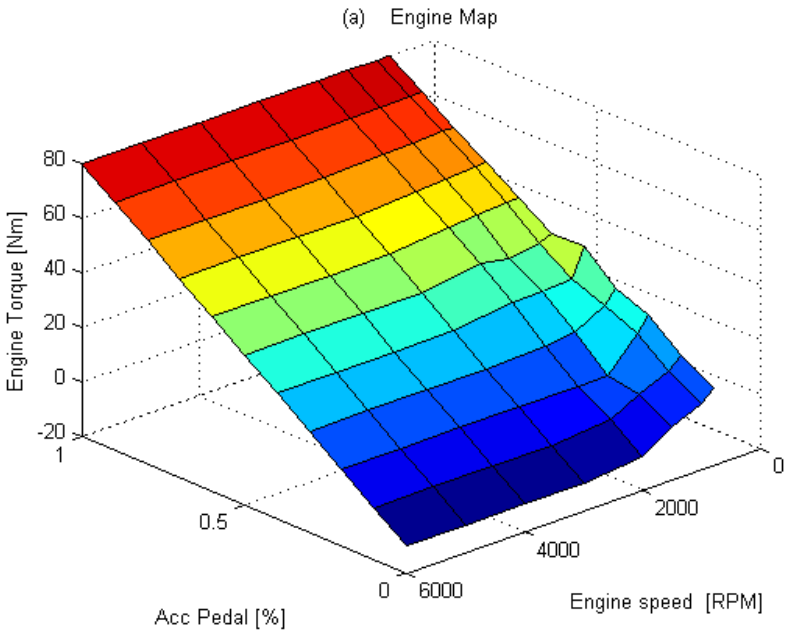
$$\mathcal{U}(x) := \left\{ \mathbf{u} \in \mathbb{U}^N \mid \forall i \quad x^{\mathbf{u}}(i;x) \in \mathbb{X} \quad \text{and} \quad x^{\mathbf{u}}(N,x) \in X_f \right\}$$

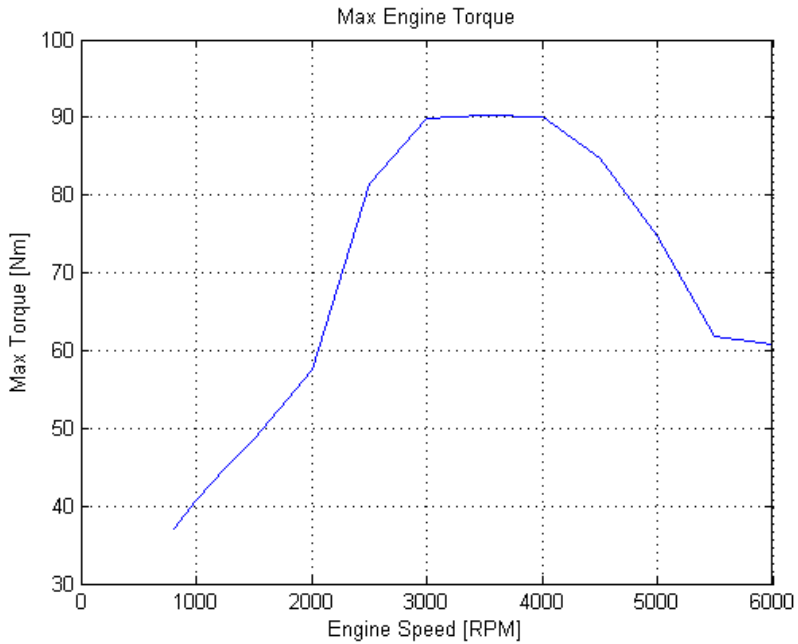
$$V(x,\mathbf{u}) = F(x(N;x)) + \sum_{i=0}^{N} L(x^{\mathbf{u}}(i;x), u(i))$$

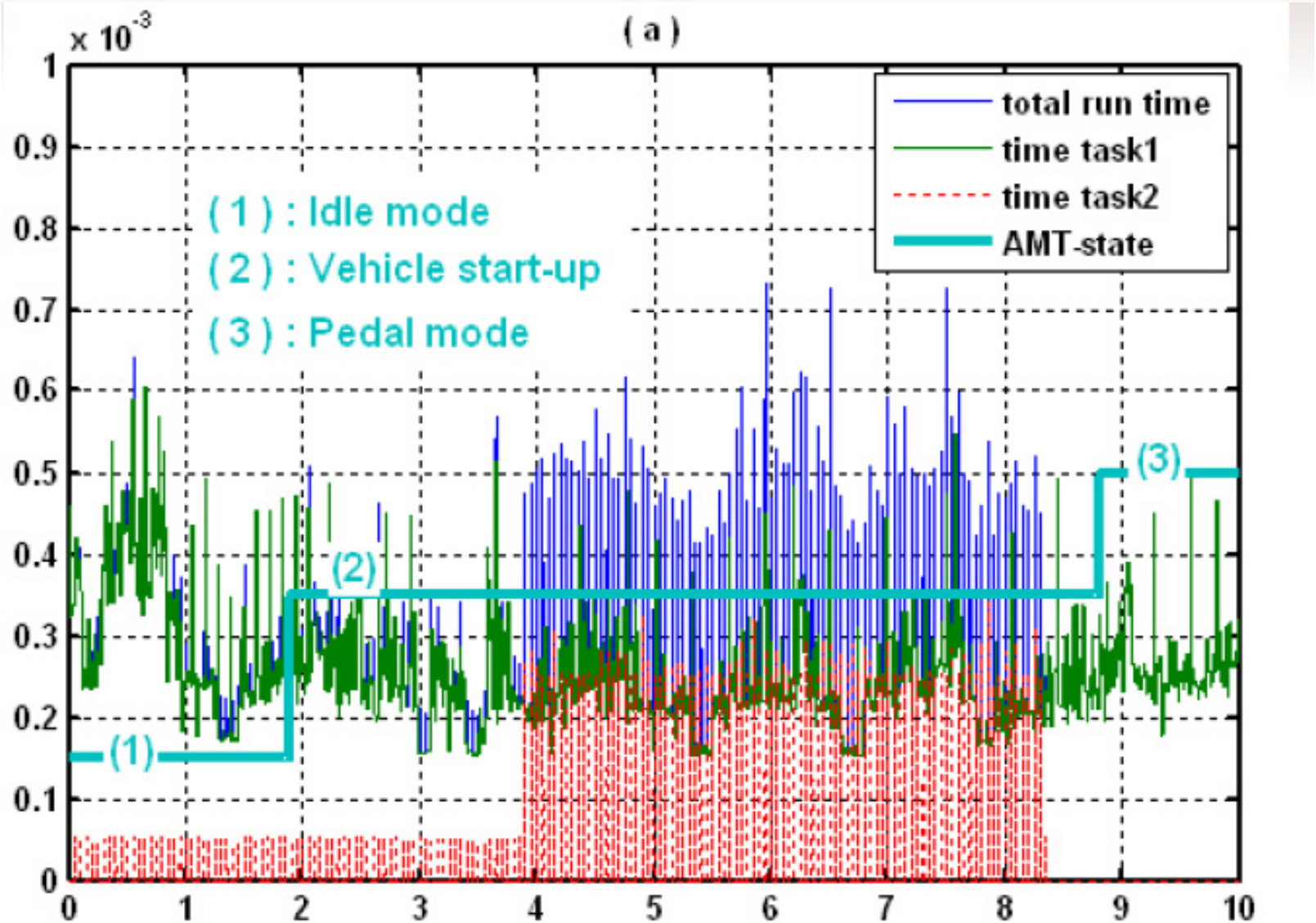
Condition 6 : Feasibility

There is at least a sequence that meets the constraints (in particular $x(0) \in X_N$ (the subset of state steerable in N steps to X_f with bounded controls in $\mathbb{U}$).

The IFP's SMART demo maps






(a)
x 10^-3
total run time
time task1
time task2
AMT-state
(1) : Idle mode
(2) : Vehicle start-up
(3) : Pedal mode
(1)
(2)
(3)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
1
2
3
4
5
6
7
8
9
10