Nonlinear Model Predictive Control for Fast Systems

Mazen Alamir

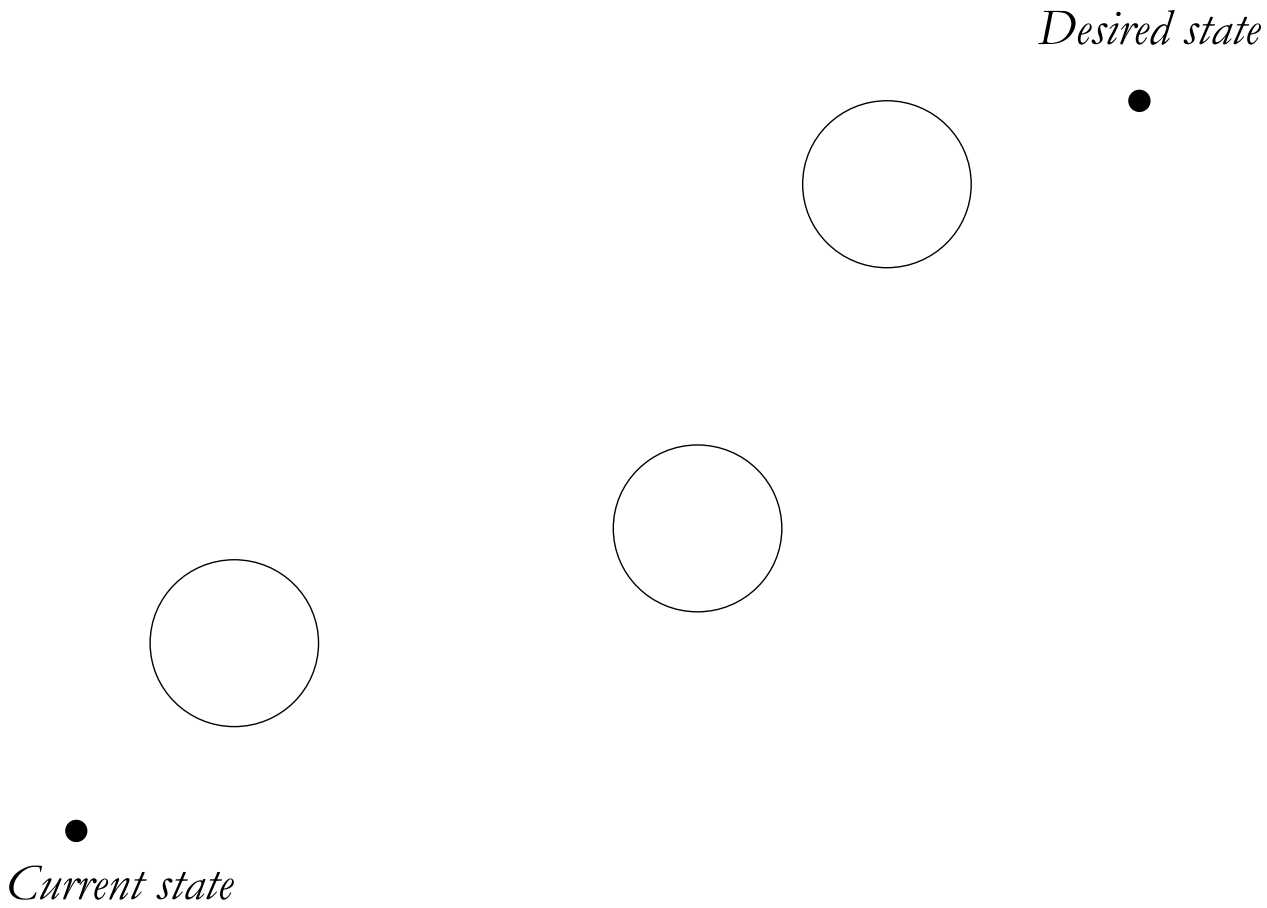
CNRS / Gipsa-lab / Département d'Automatique

Equipe Systèmes Non Linéaires et Complexité (SYSCO)

JNRR'07

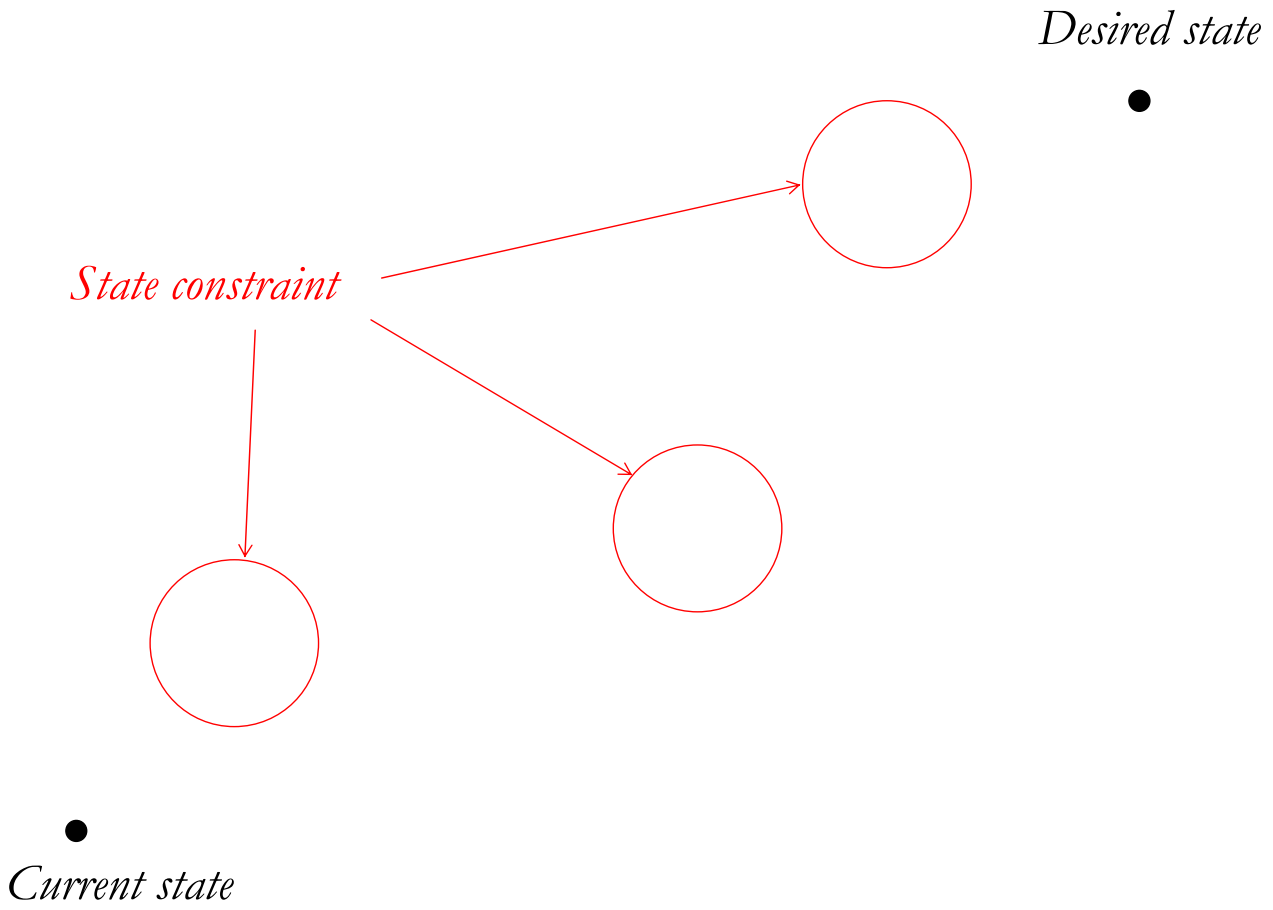
NMPC : You are already using it . . . !

NMPC : You are already using it . . . !

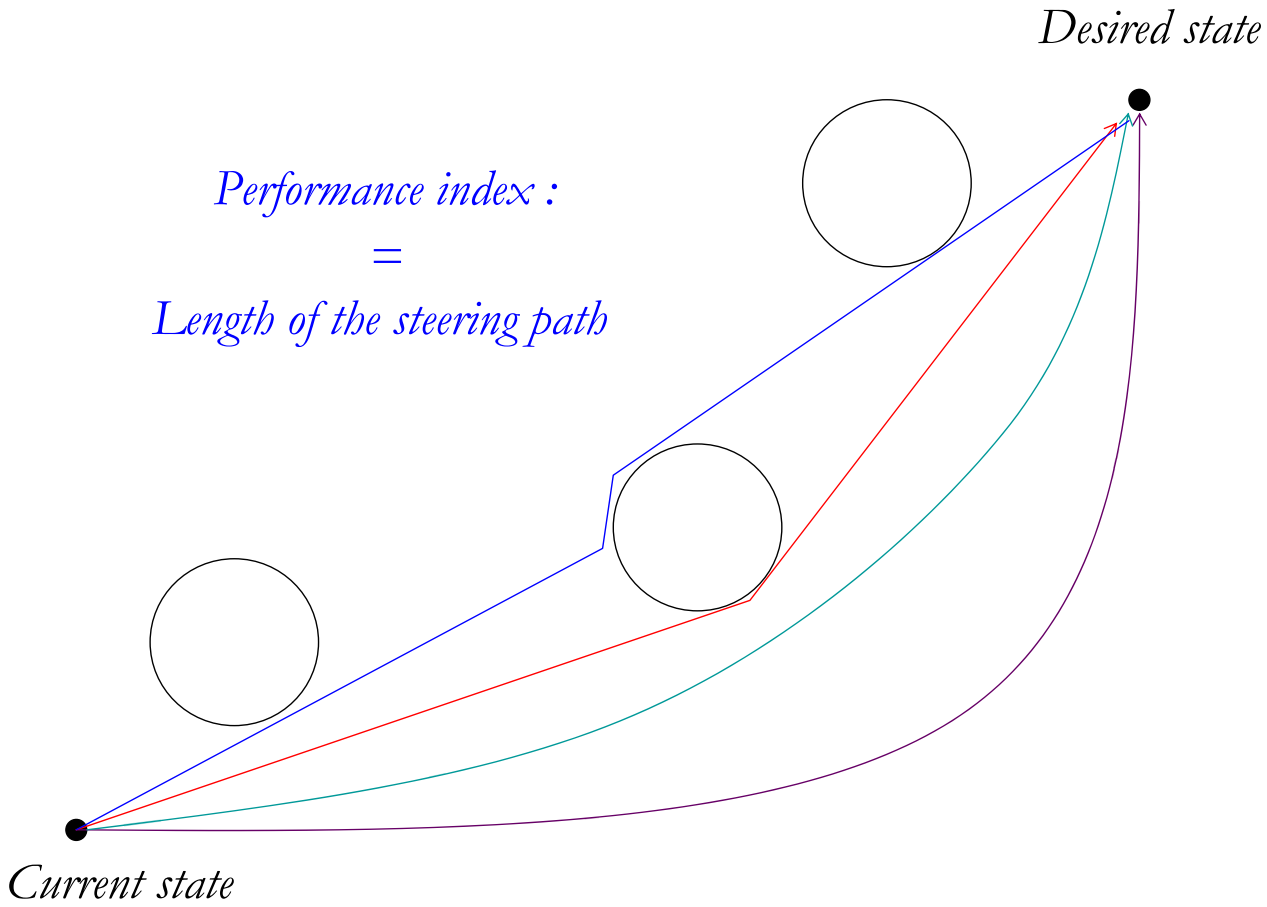


gipsa-lab

NMPC : You are already using it . . . !

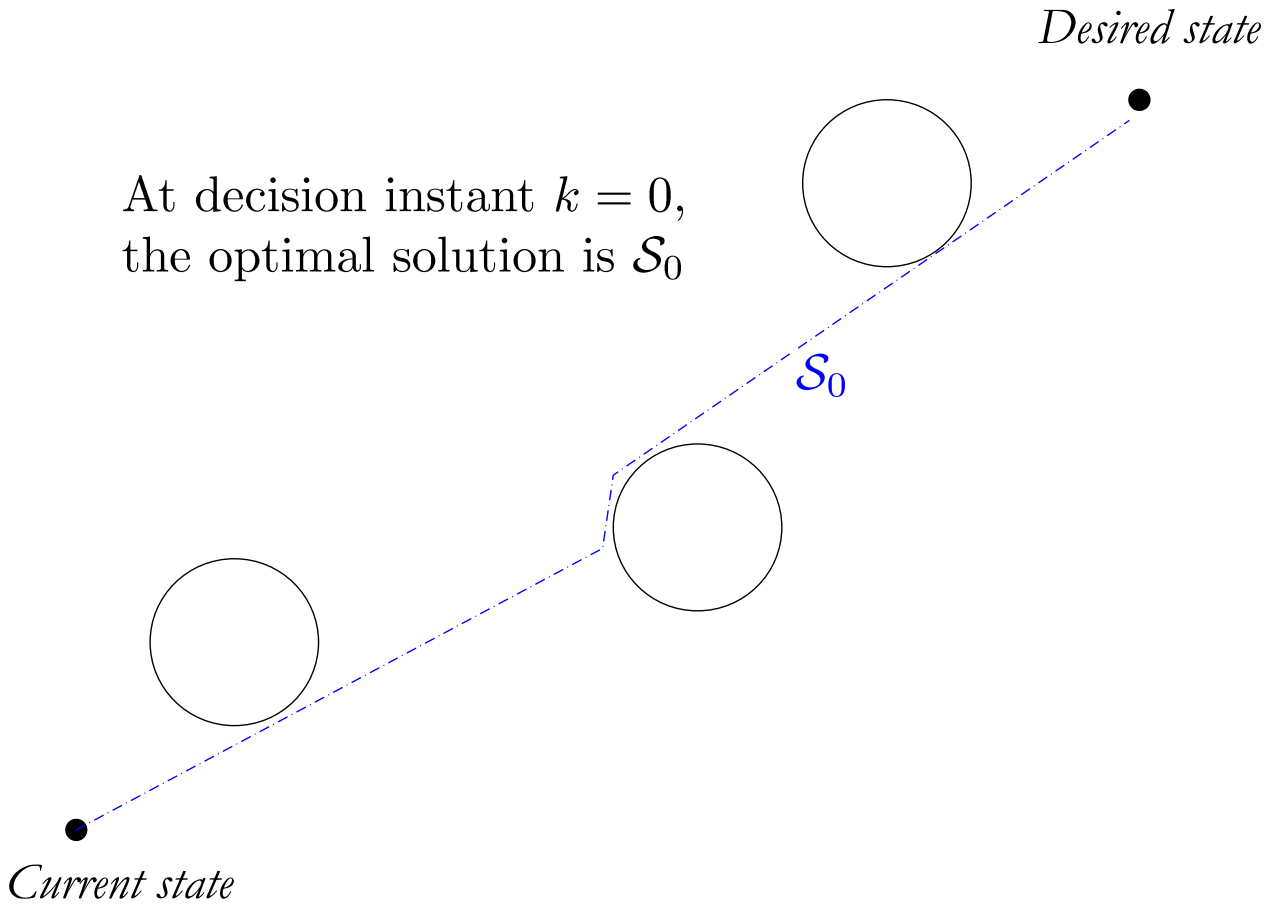


NMPC : You are already using it . . . !

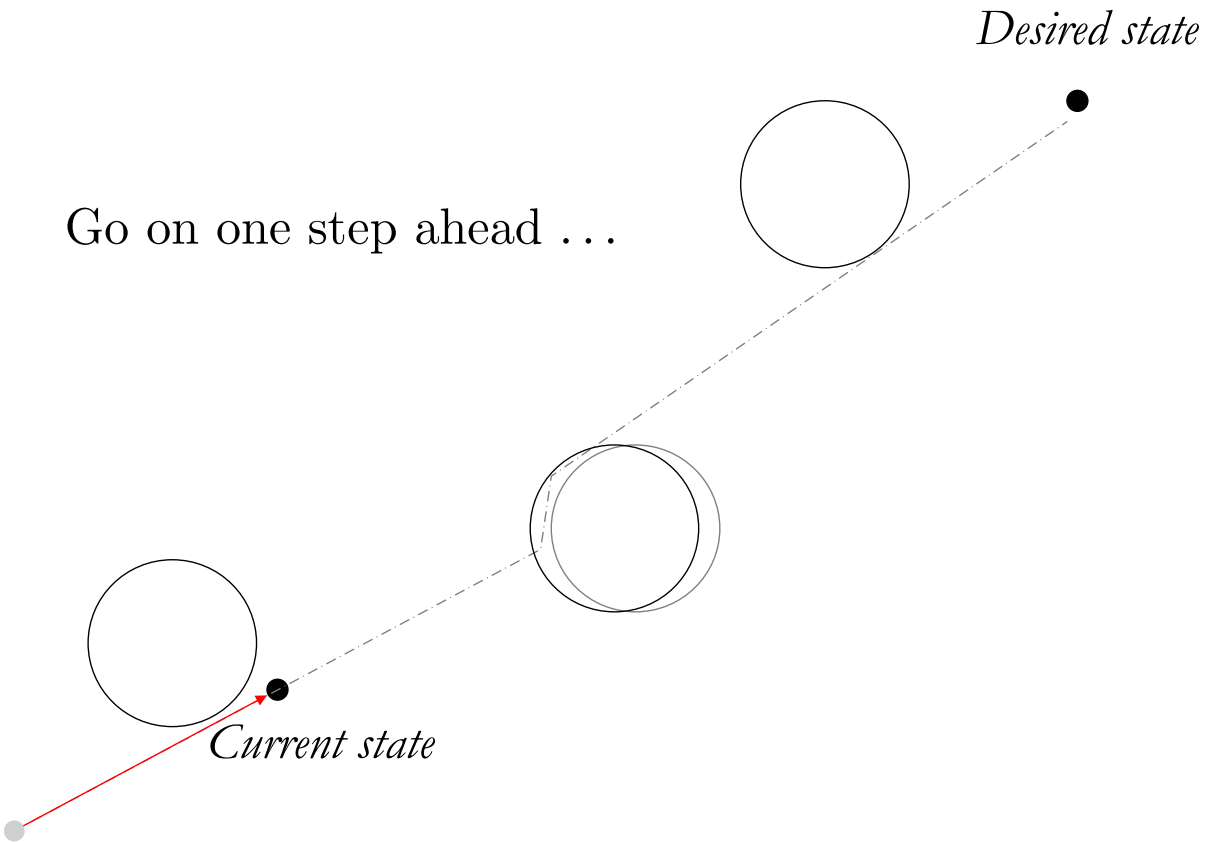


gipsa-lab

NMPC : You are already using it . . . !

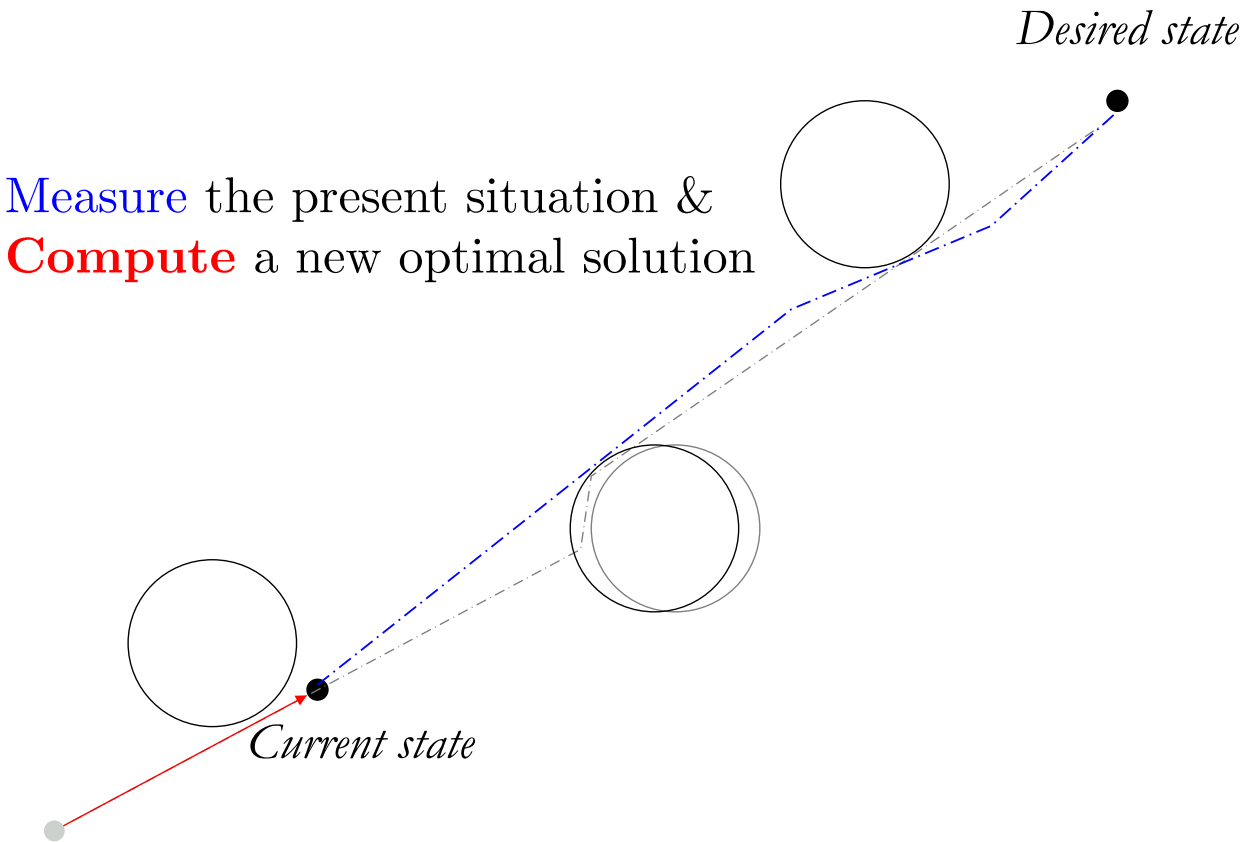


NMPC : You are already using it . . . !

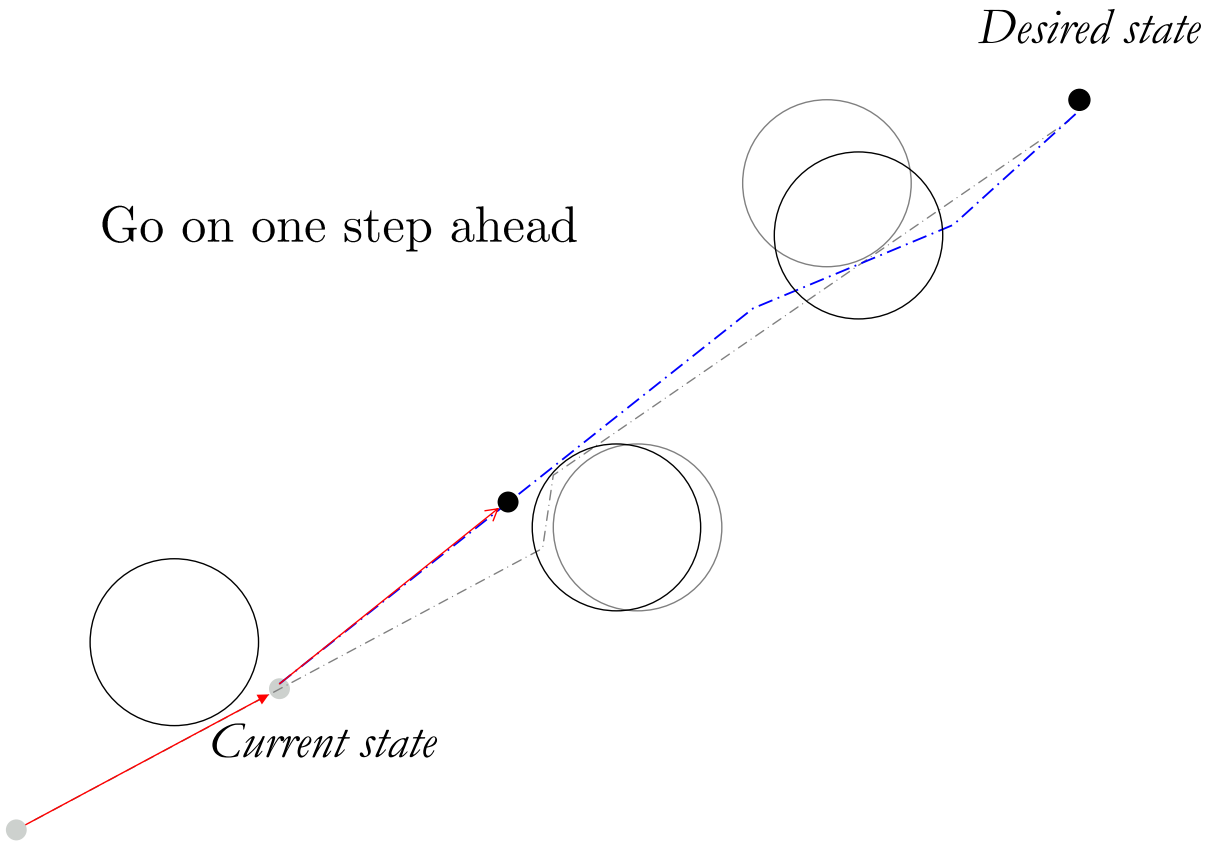


gipsa-lab

NMPC : You are already using it . . . !

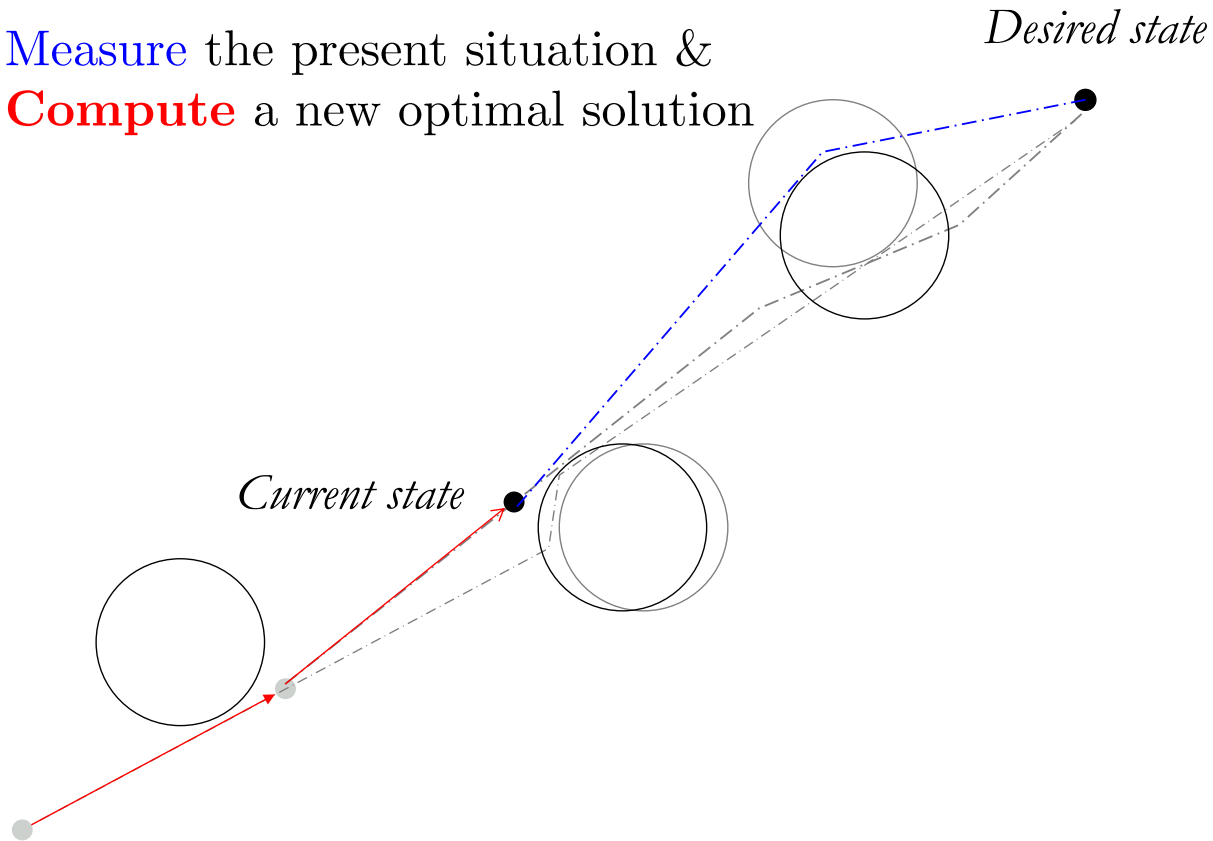


NMPC : You are already using it . . . !

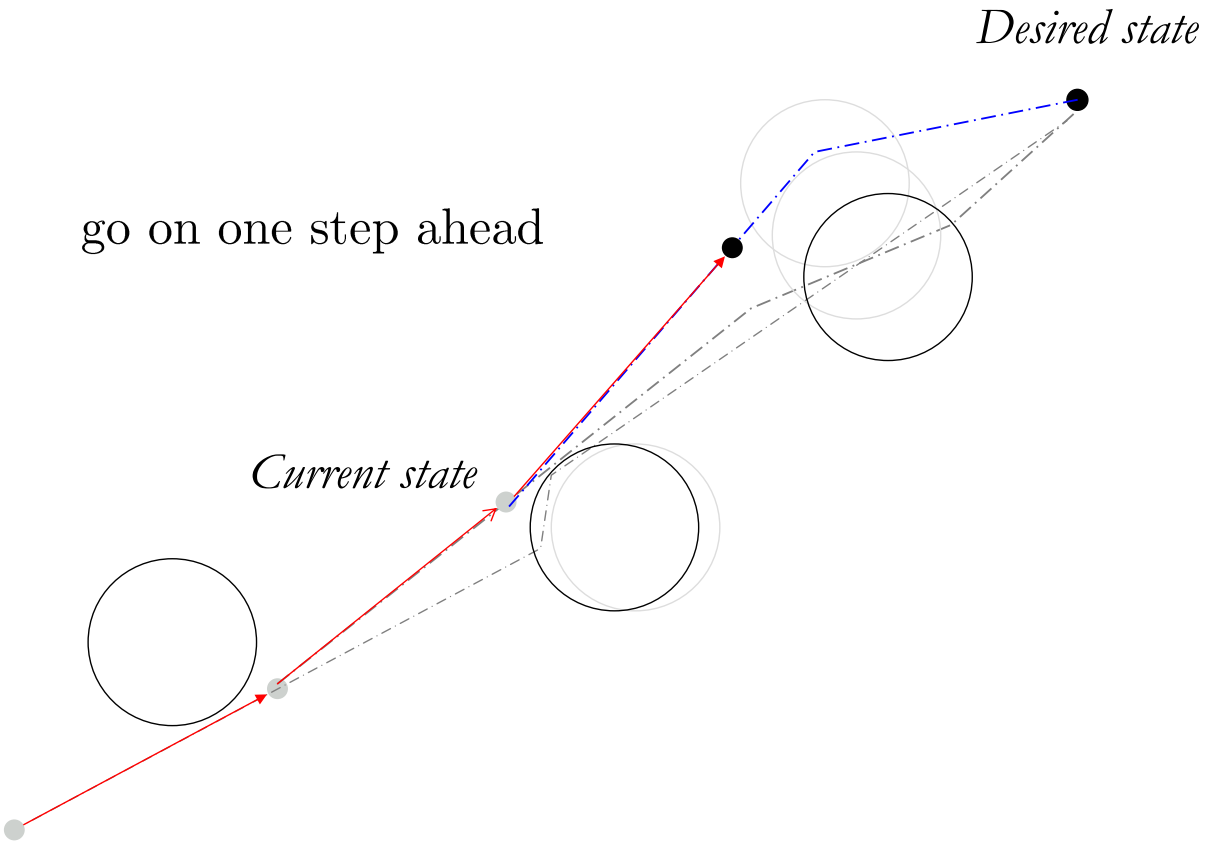


gipsa-lab

NMPC : You are already using it . . . !

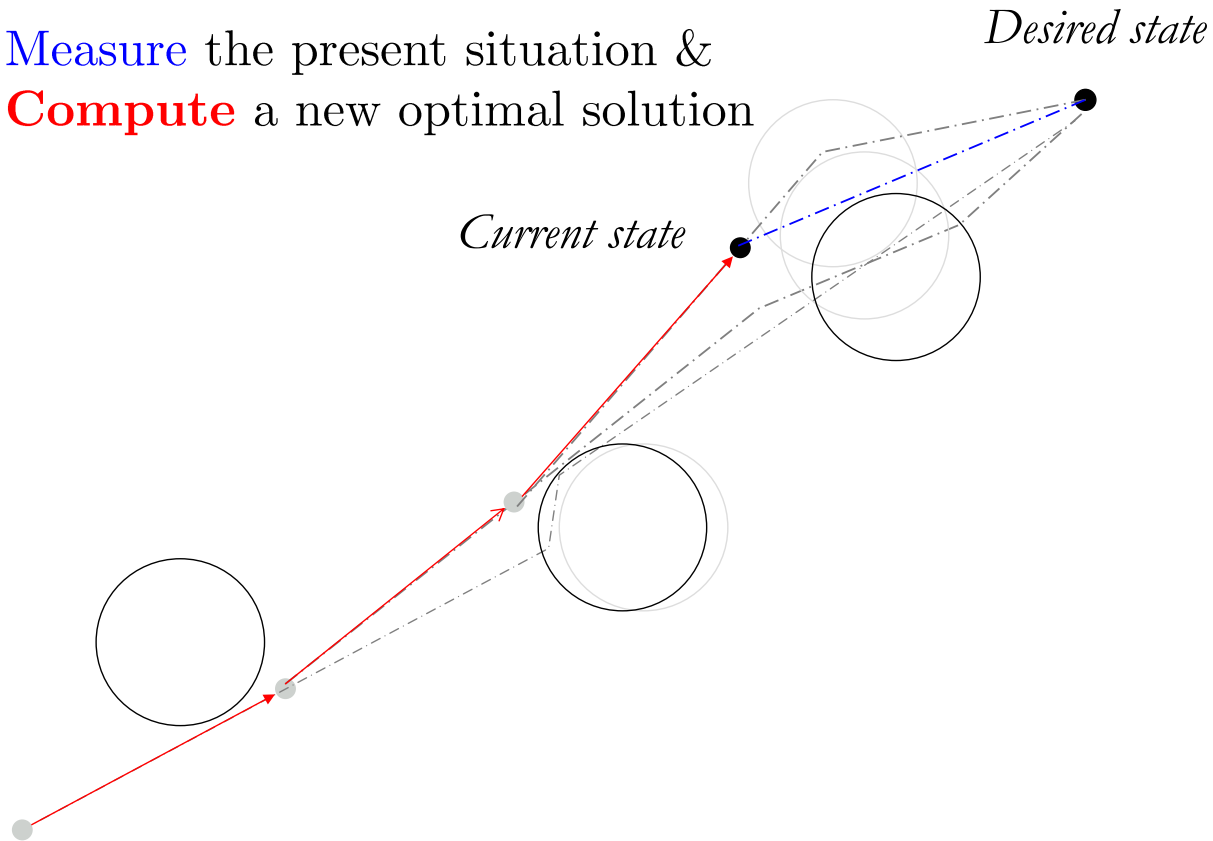


NMPC : You are already using it . . . !

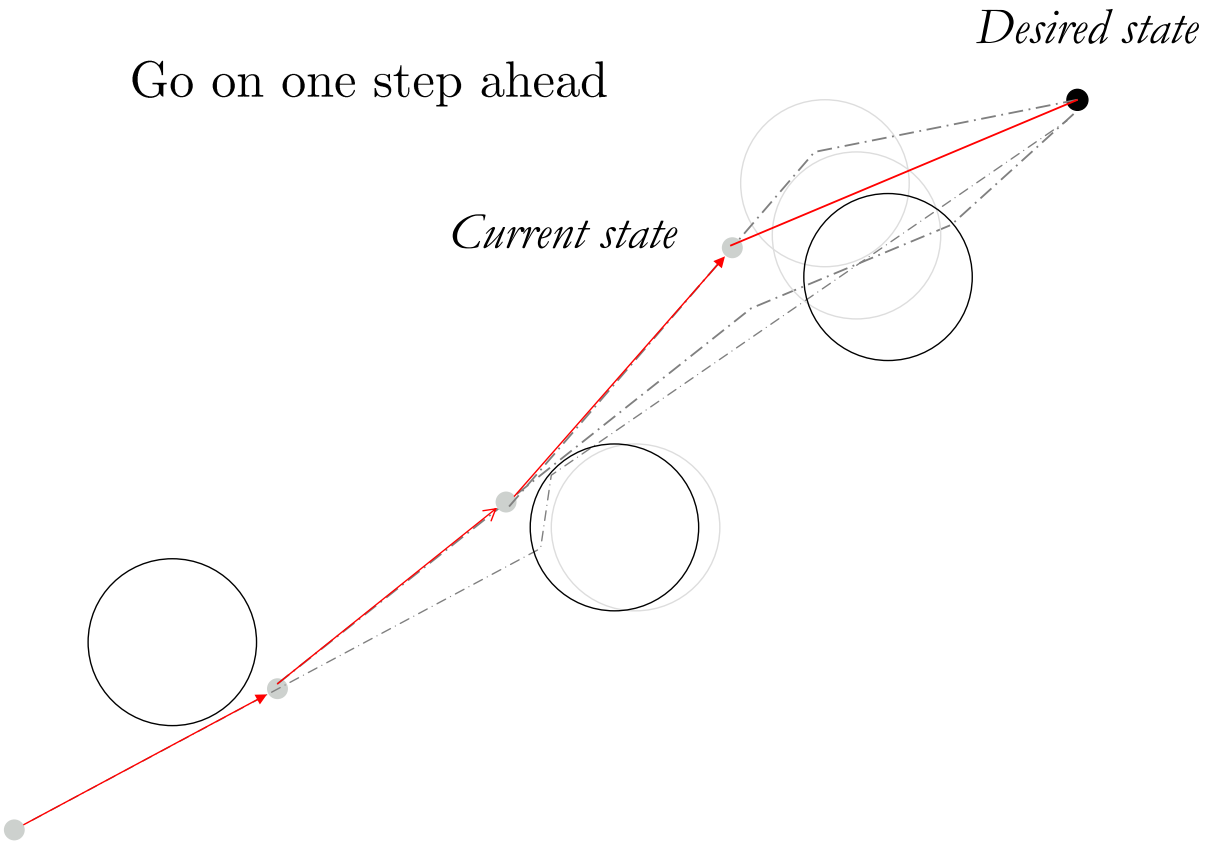


gipsa-lab

NMPC : You are already using it . . . !

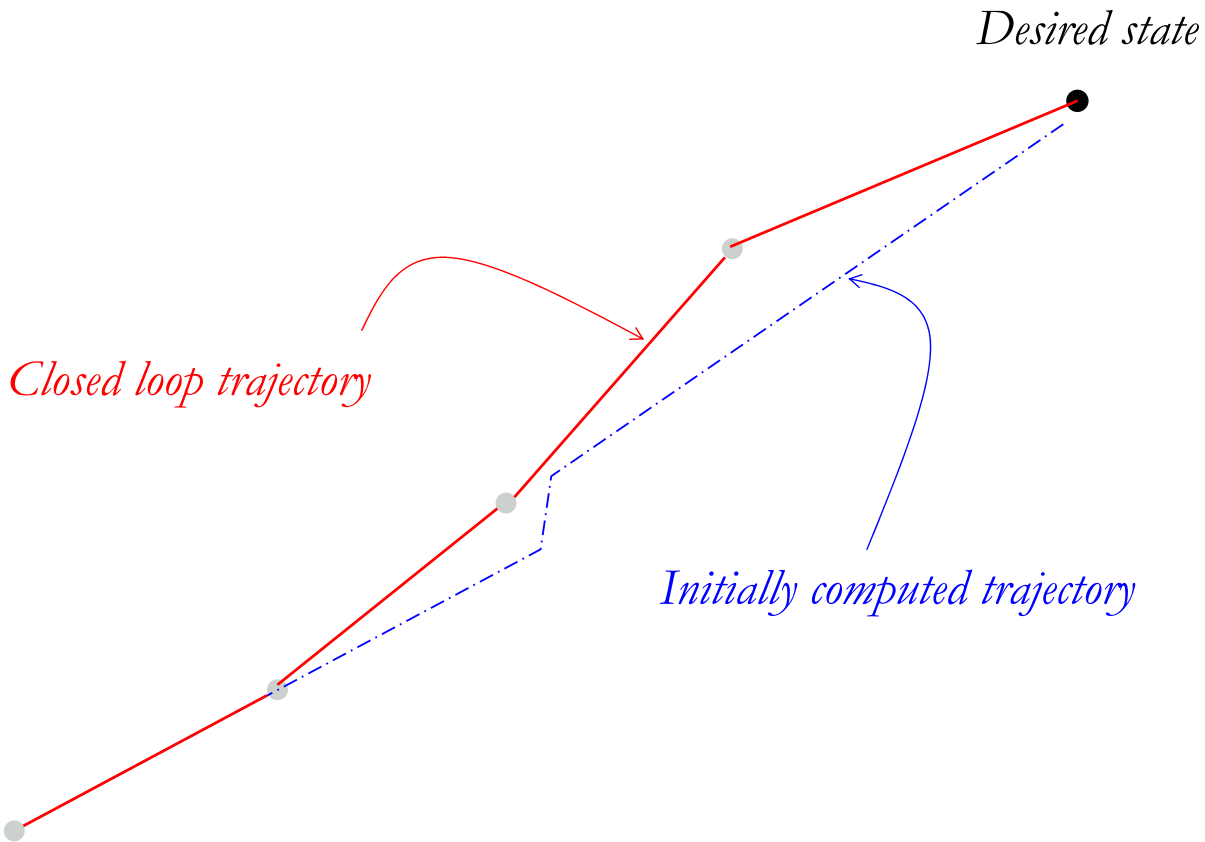


NMPC : You are already using it . . . !



gipsa-lab

NMPC : You are already using it . . . !



More formally

Consider a dynamical system :

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

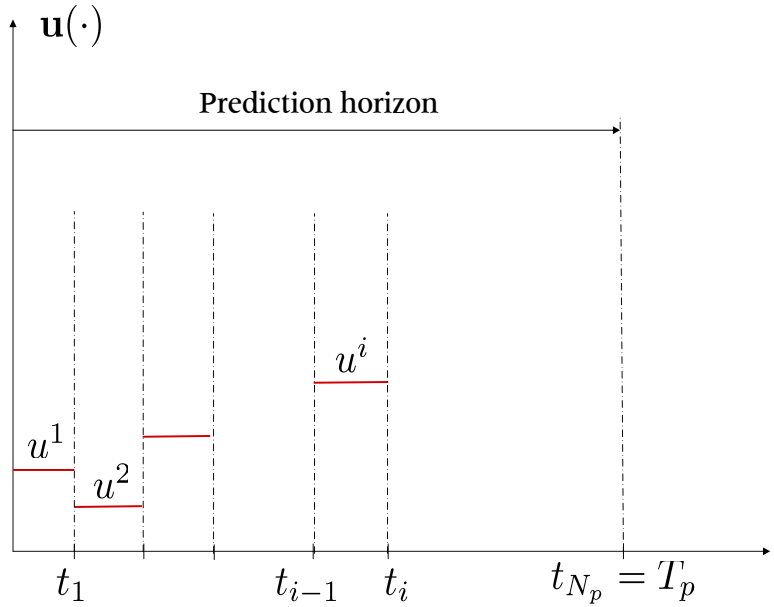
More formally

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

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

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

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

gipsa-lab

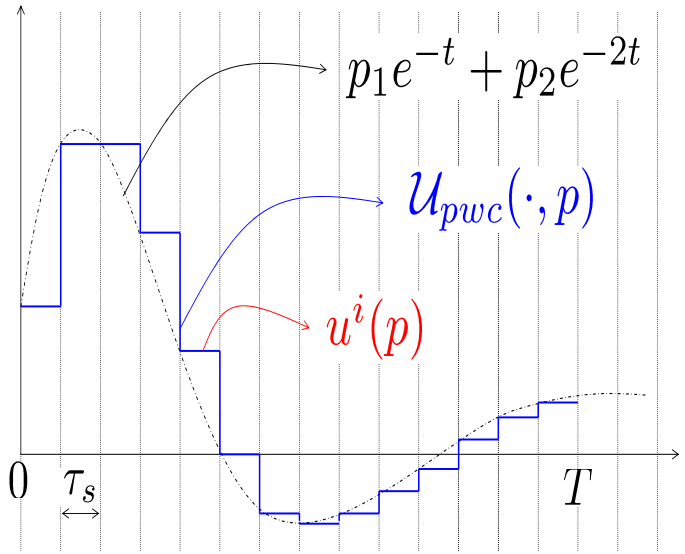
More formally

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 :

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



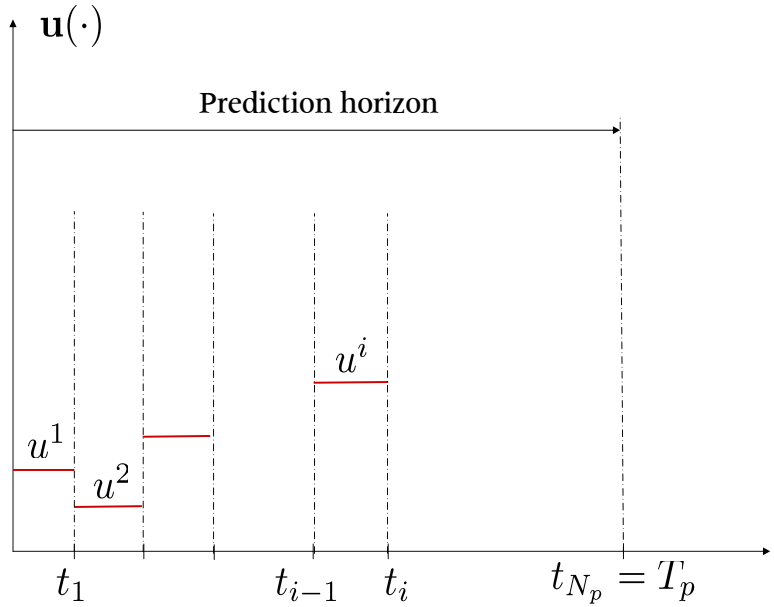
More formally

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

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

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

Advantages

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

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.

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)

gipsa-lab

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 $x^{(1)} = (1, 2)$.

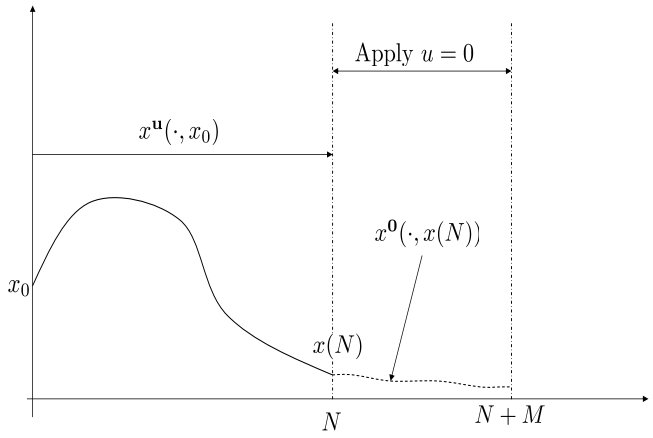
Consider the cost function

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

where

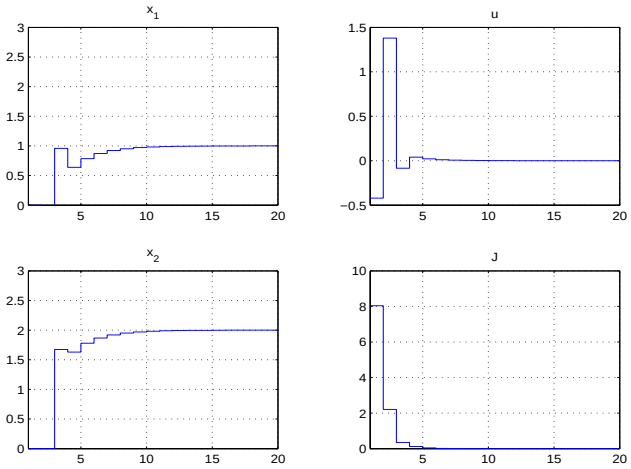
$$L(x, u) := \|x - x^{(1)}\|^2 + ru^2$$

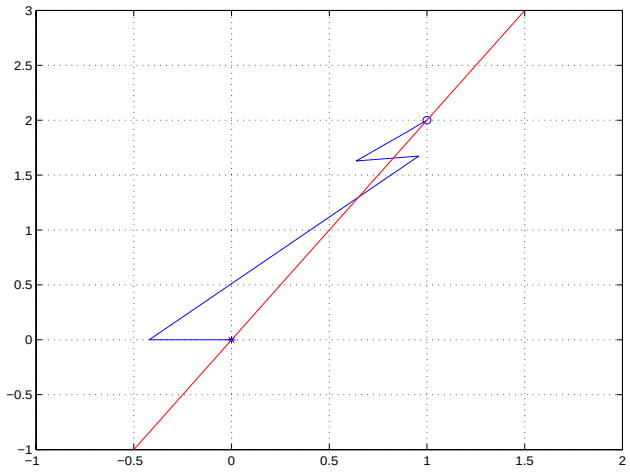
$$F(x) = \sum_{i=1}^{M-N} x^{\mathbf{0}}(i; x)$$



The stability is an issue !

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





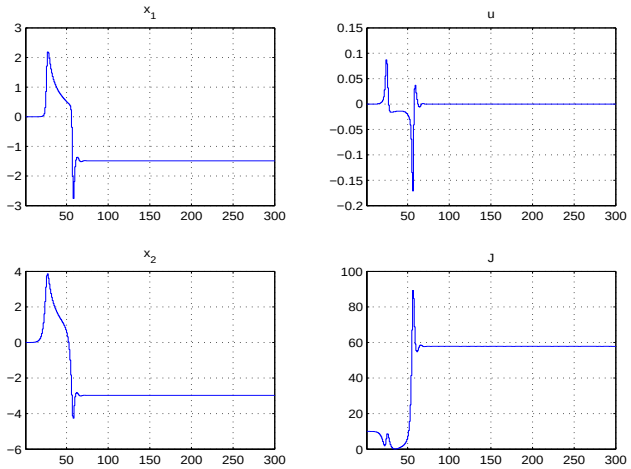
The stability is an issue !

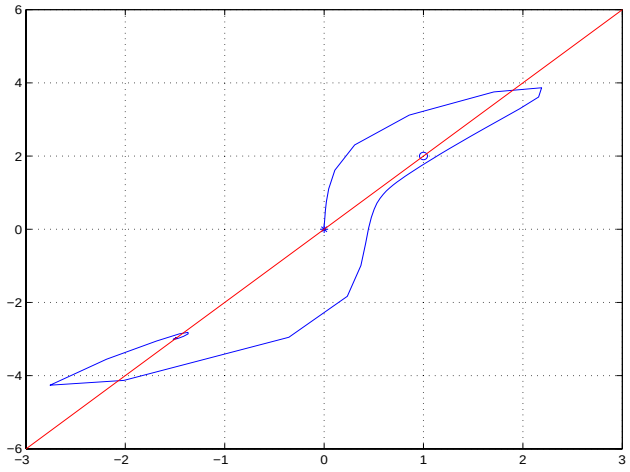
Let us assume that one looks for solution such that $u \leq 0.1$

Assume that for that reason one takes

$$r = 160$$

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

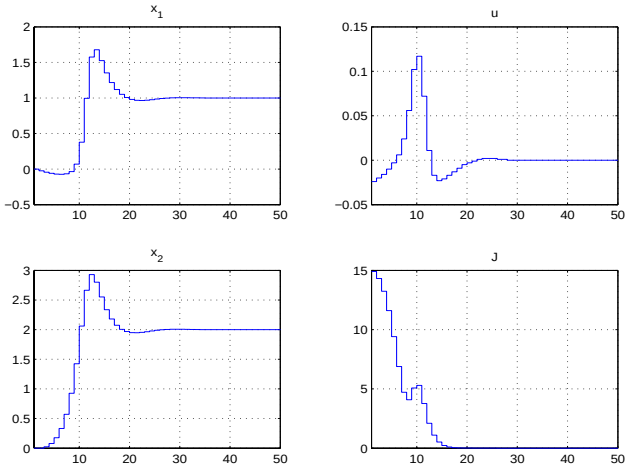


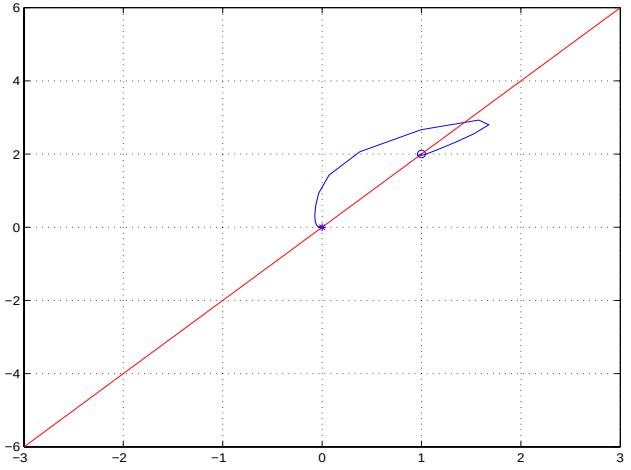


The stability is an issue !

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

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





The stability is an issue !

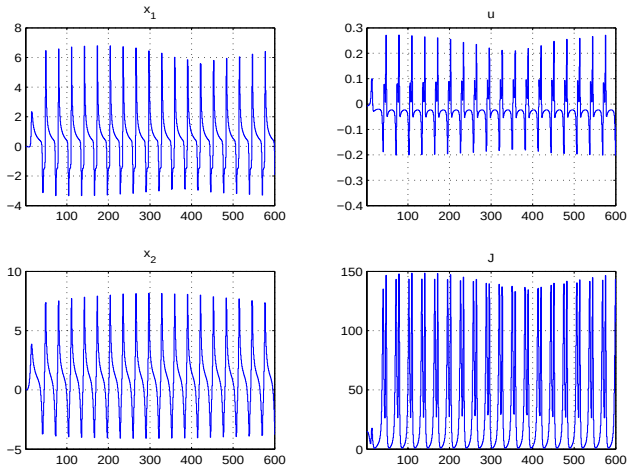
Increasing M enabled the target state to be reached.

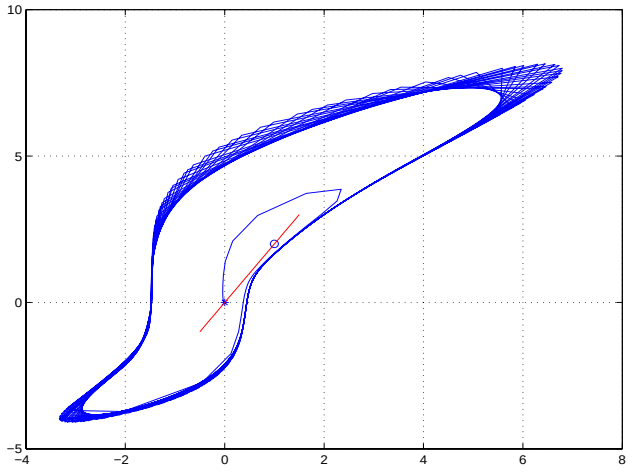
However, the control is again above 0.1,

So let us increase r again by taking

$$r = 500$$

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





The stability is an issue !

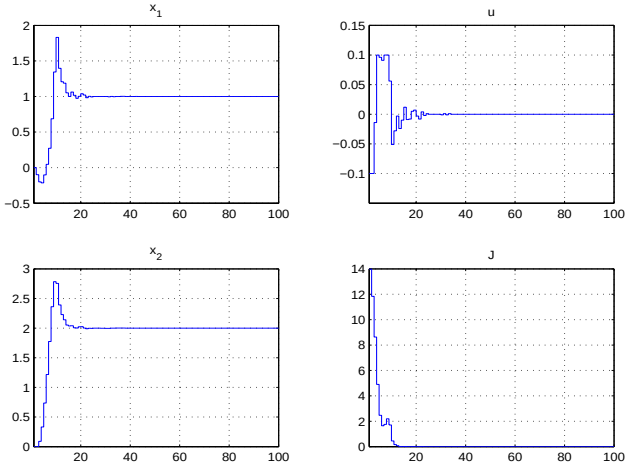
Let us explicitly impose the constraint

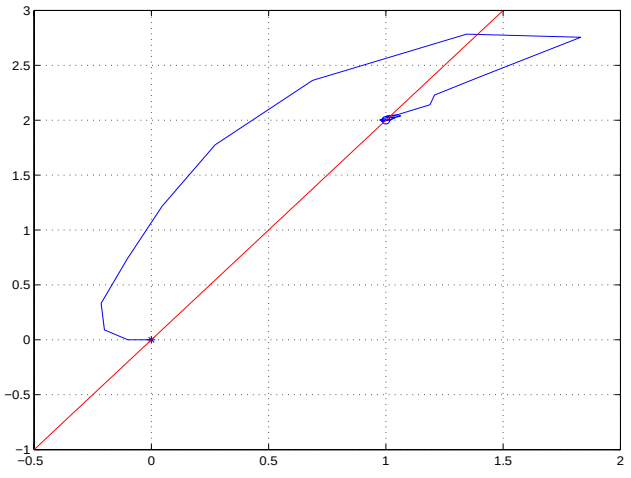
$$\mathbb{U} = [-0.1, 0.1]$$

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 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

- In the paper (and related references)
  - Classical formulations with guaranteed stability
  - Contractive formulations
  - Under ideal *optimizer*

The concept of distributed-on-time optimization

NMPC

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

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

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

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

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

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

More precisely

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

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

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

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

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

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

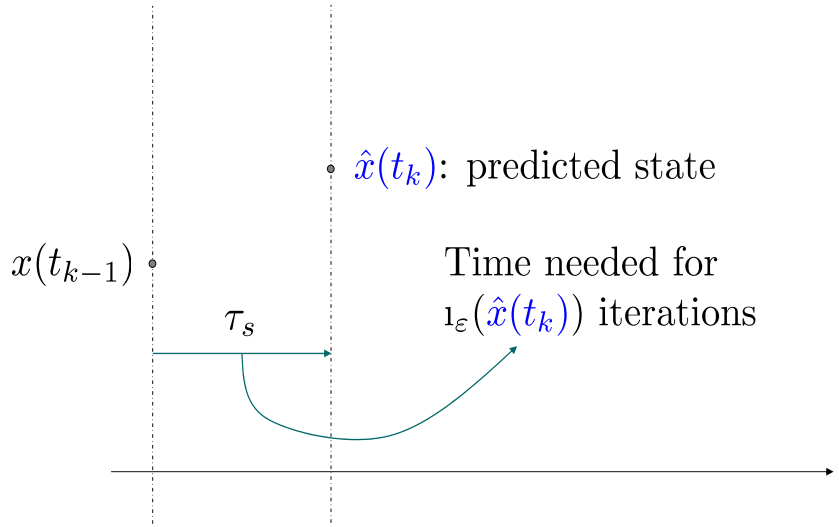
More precisely

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

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 $x(t_k)$.



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

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

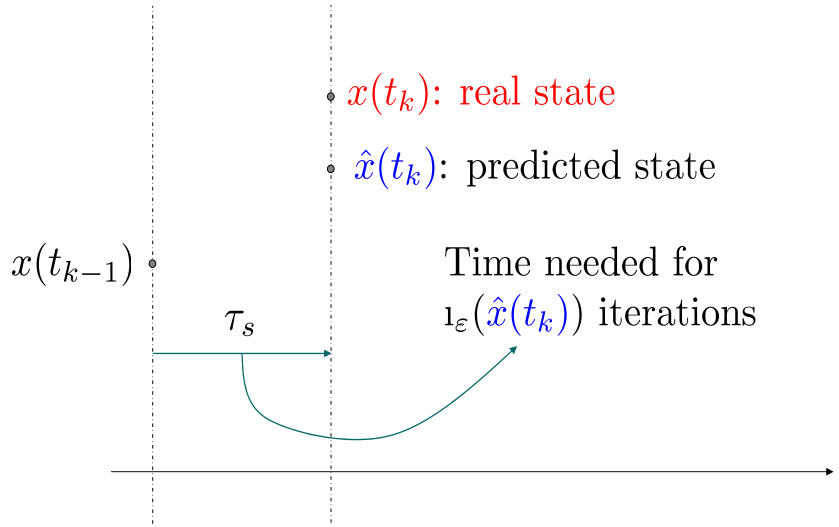
More precisely

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

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 $x(t_k)$.



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

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

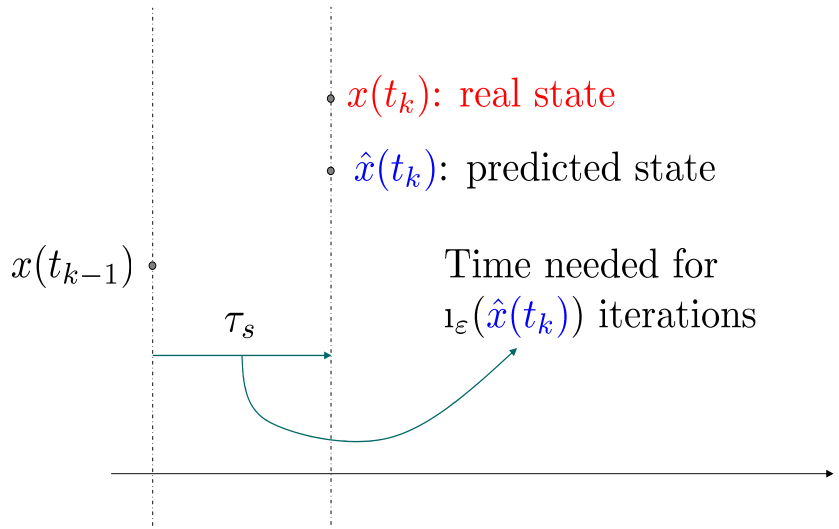
More precisely

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

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 $x(t_k)$.



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

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

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

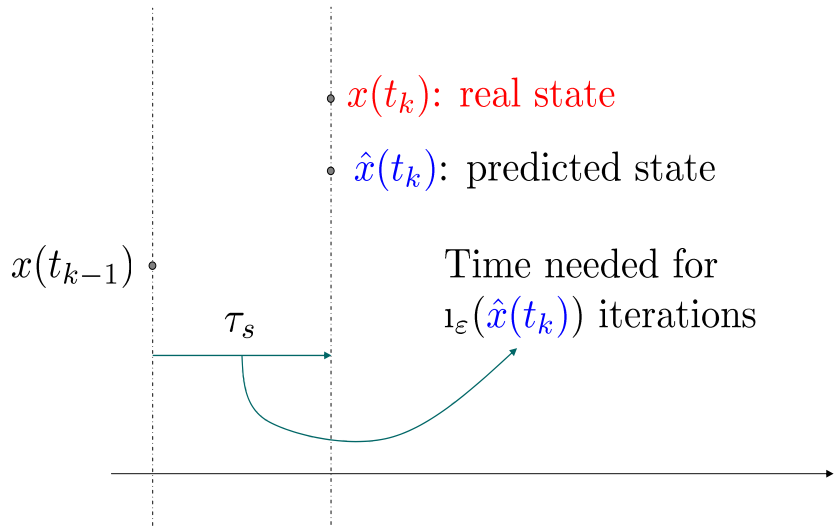
More precisely

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

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 :

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

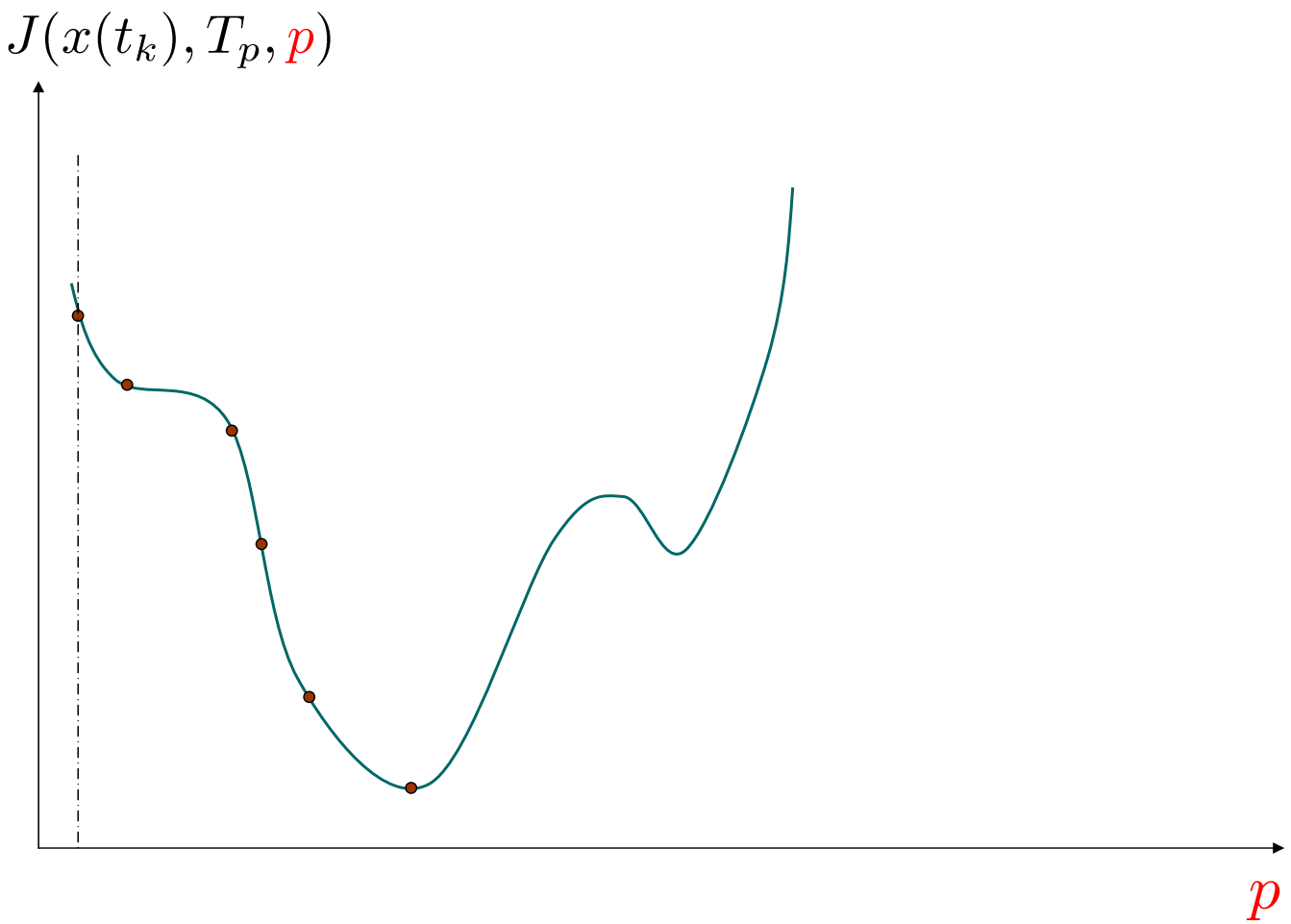
- p^+ is defined by the translatability of the control parametrization.
- $\hat{x}(t_{k+1})$ is the predicted state.

A more realistic approach

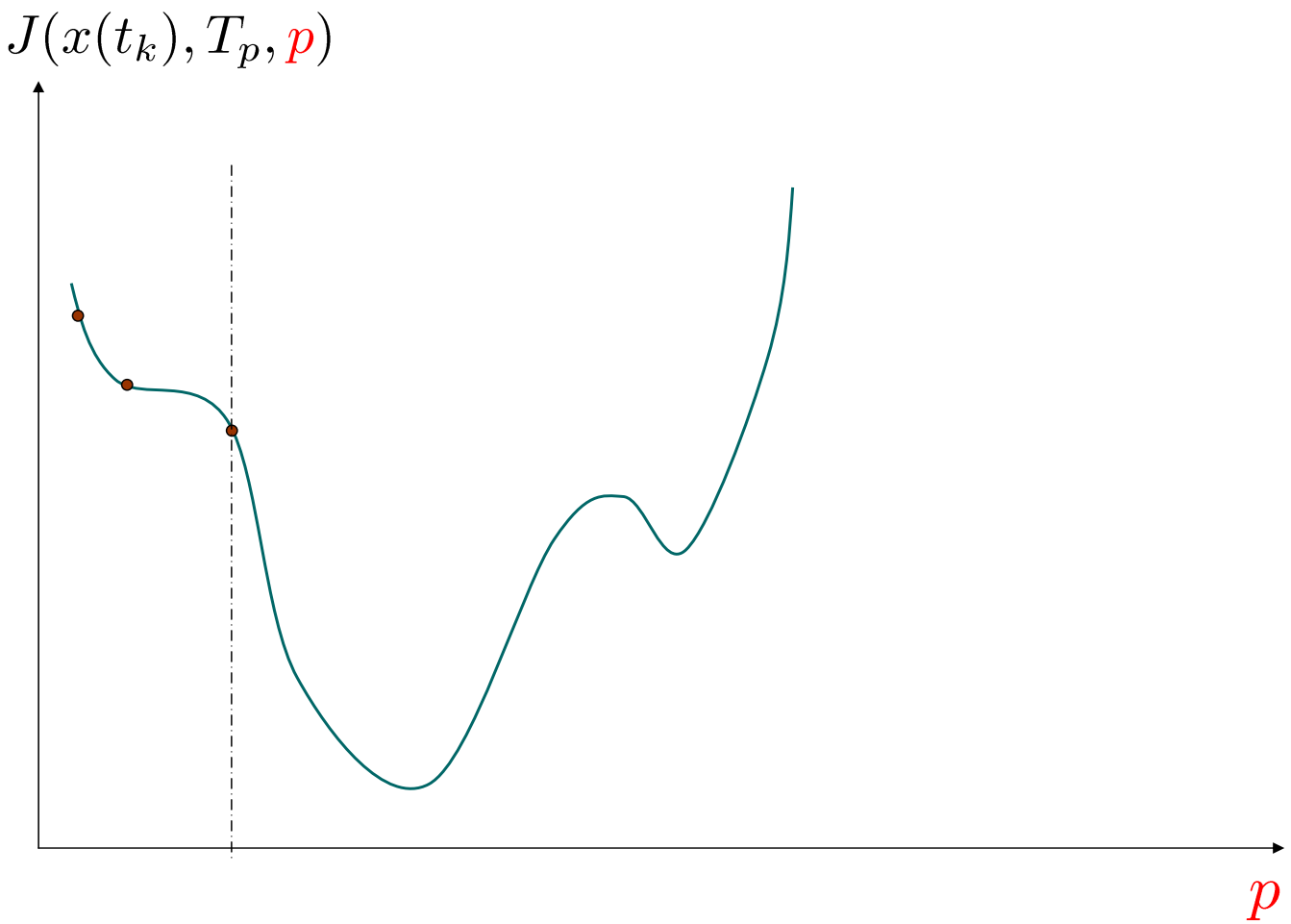
- Choose a sampling period τ_s
- Deduce a maximum number of iterations $q \in \mathbb{N}$ [compatible with τ_s]
- Distribute the optimization over the system real life-time :

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

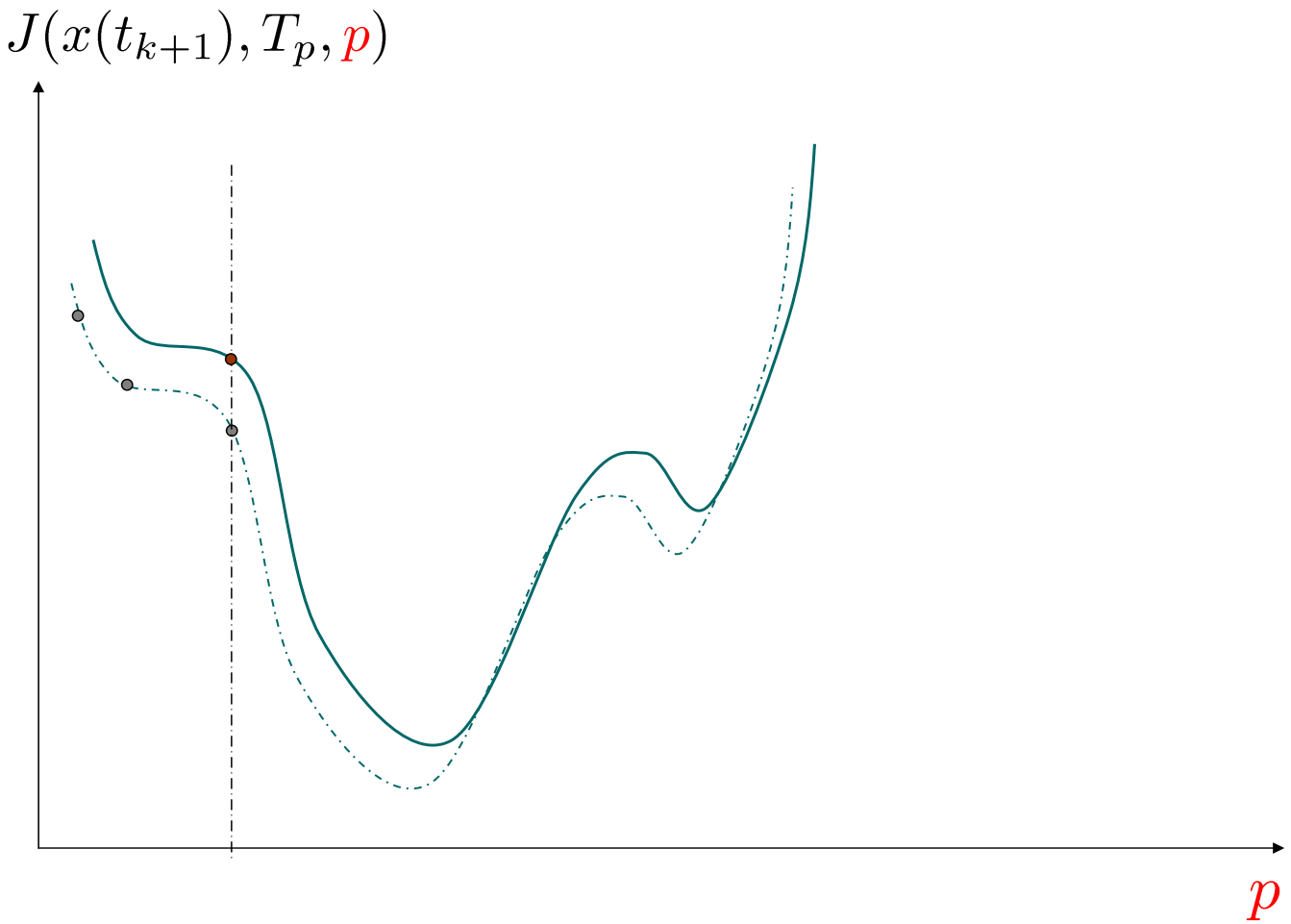
- p^+ is defined by the translatability of the control parametrization.
- $\hat{x}(t_{k+1})$ is the predicted state.
- Stability is a more involved issue than in classical formulations



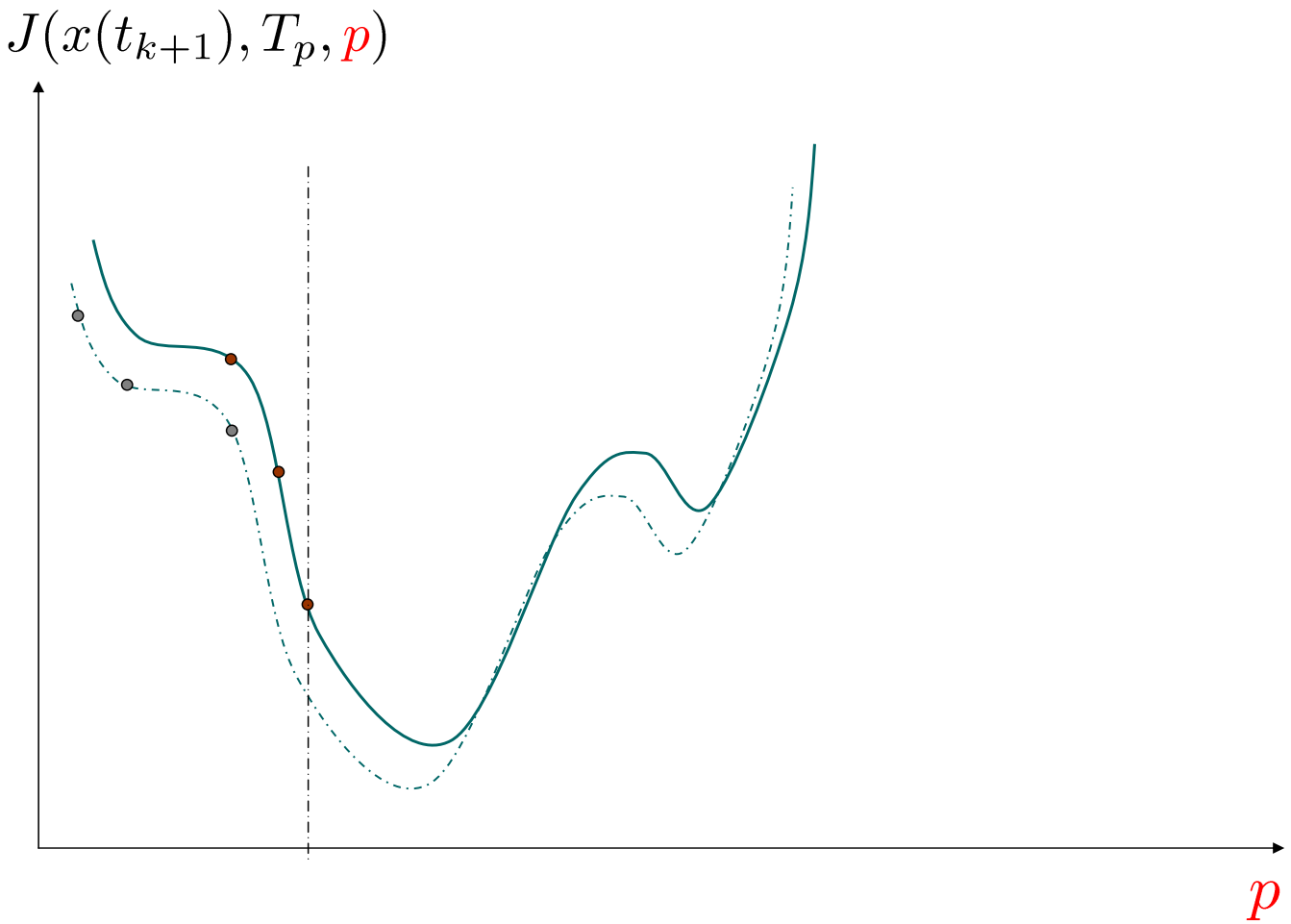
Case $q = 2$



Case $q = 2$

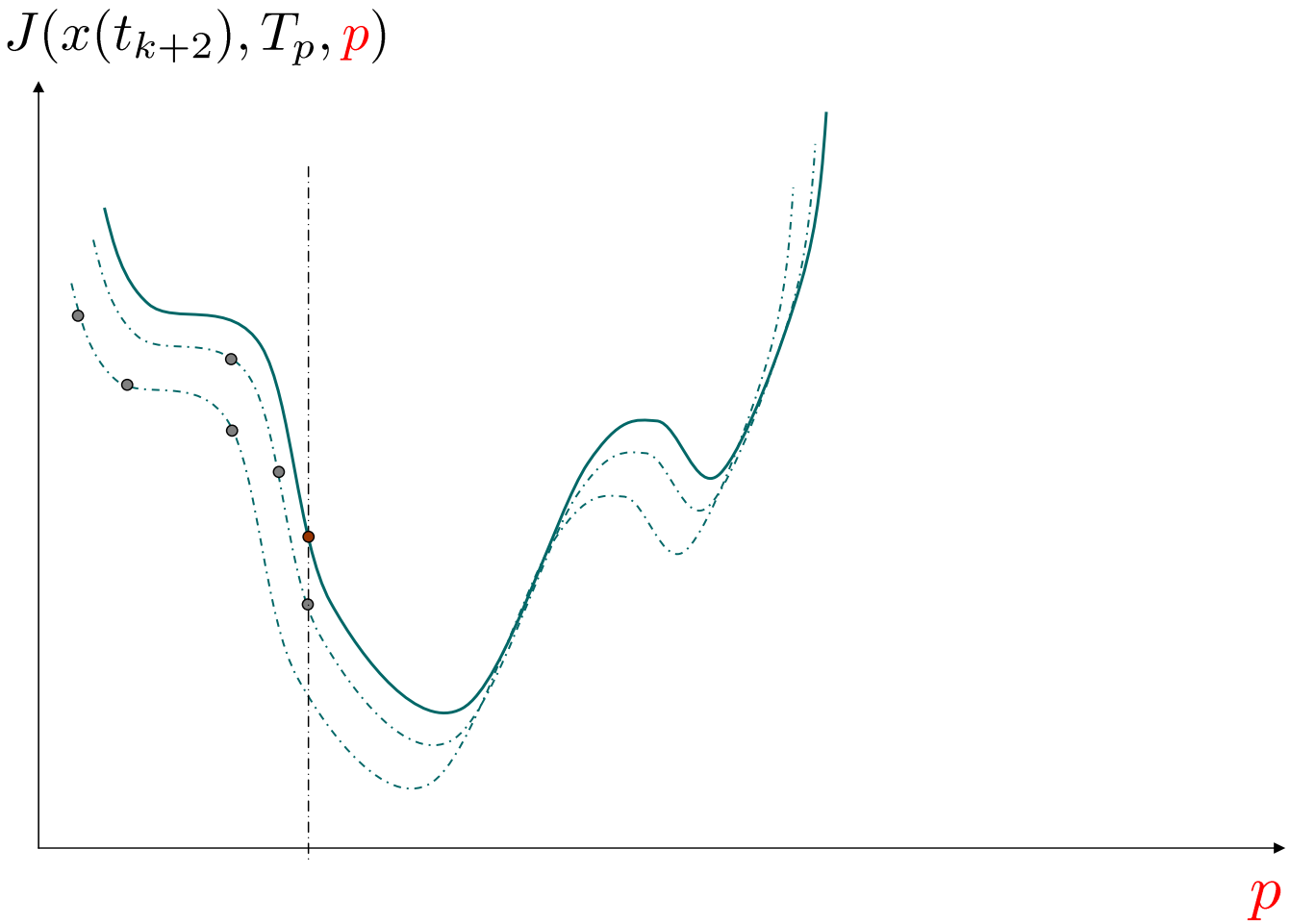


Case $q = 2$

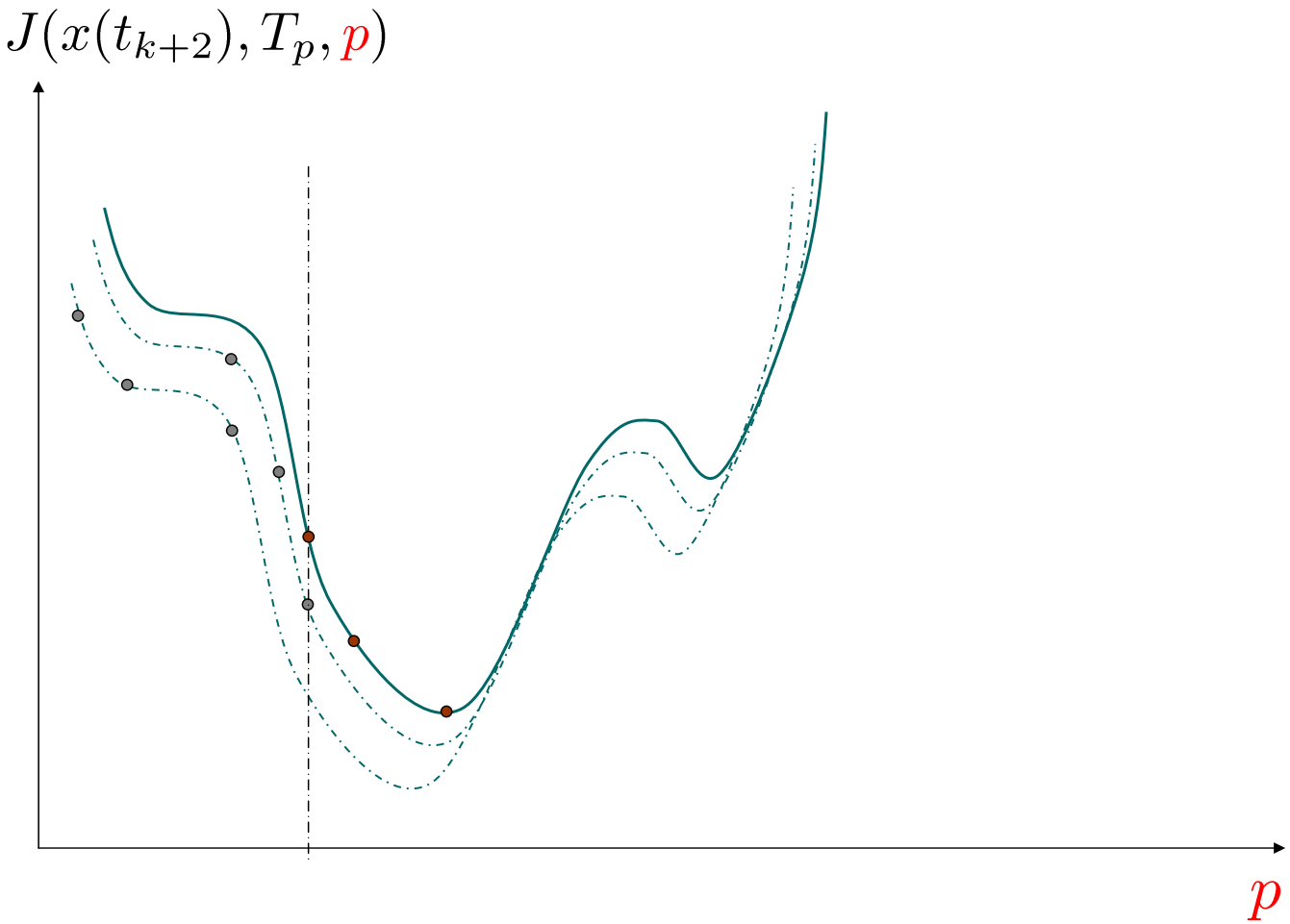


Case $q = 2$

gipsa-lab

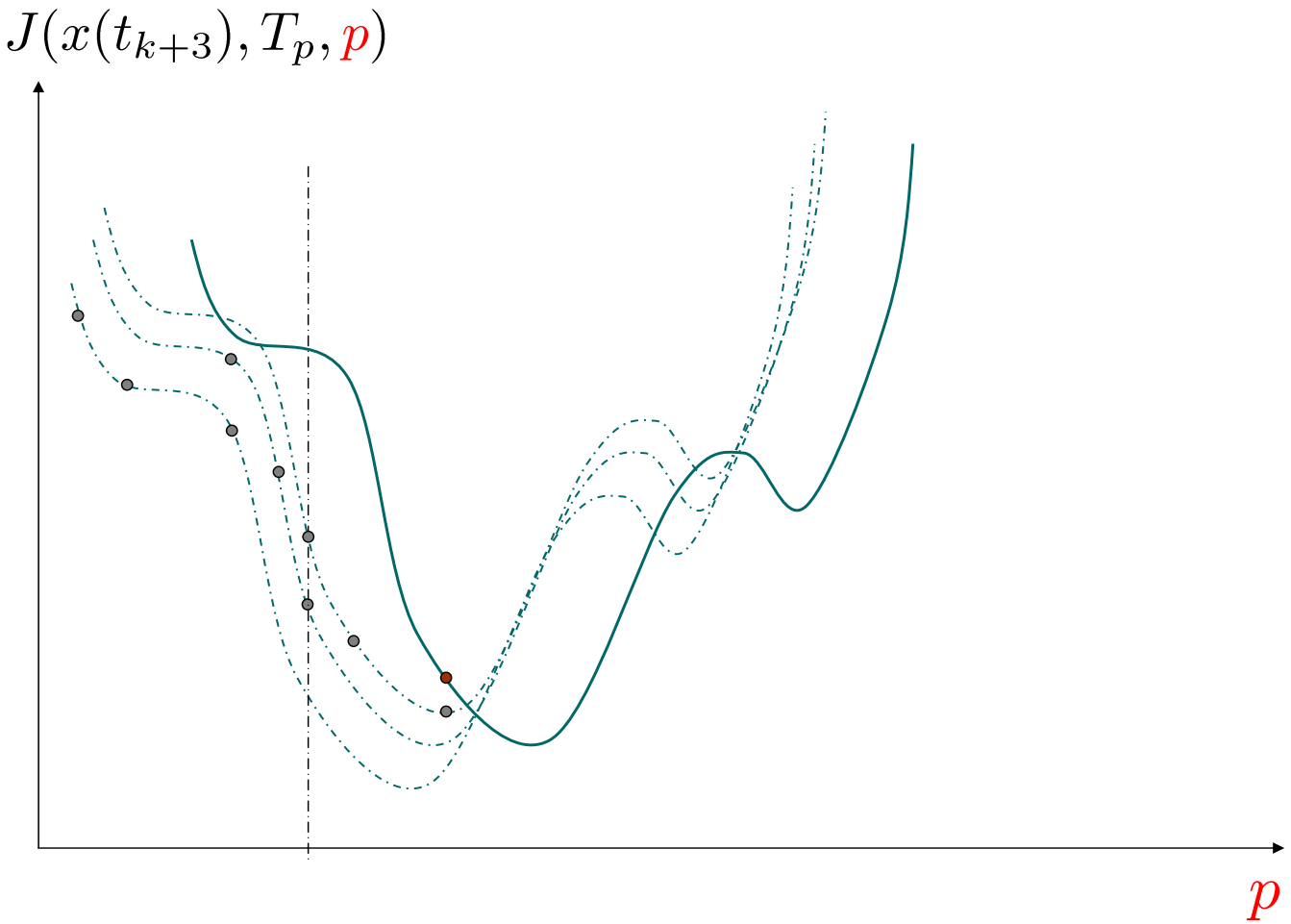


Case $q = 2$

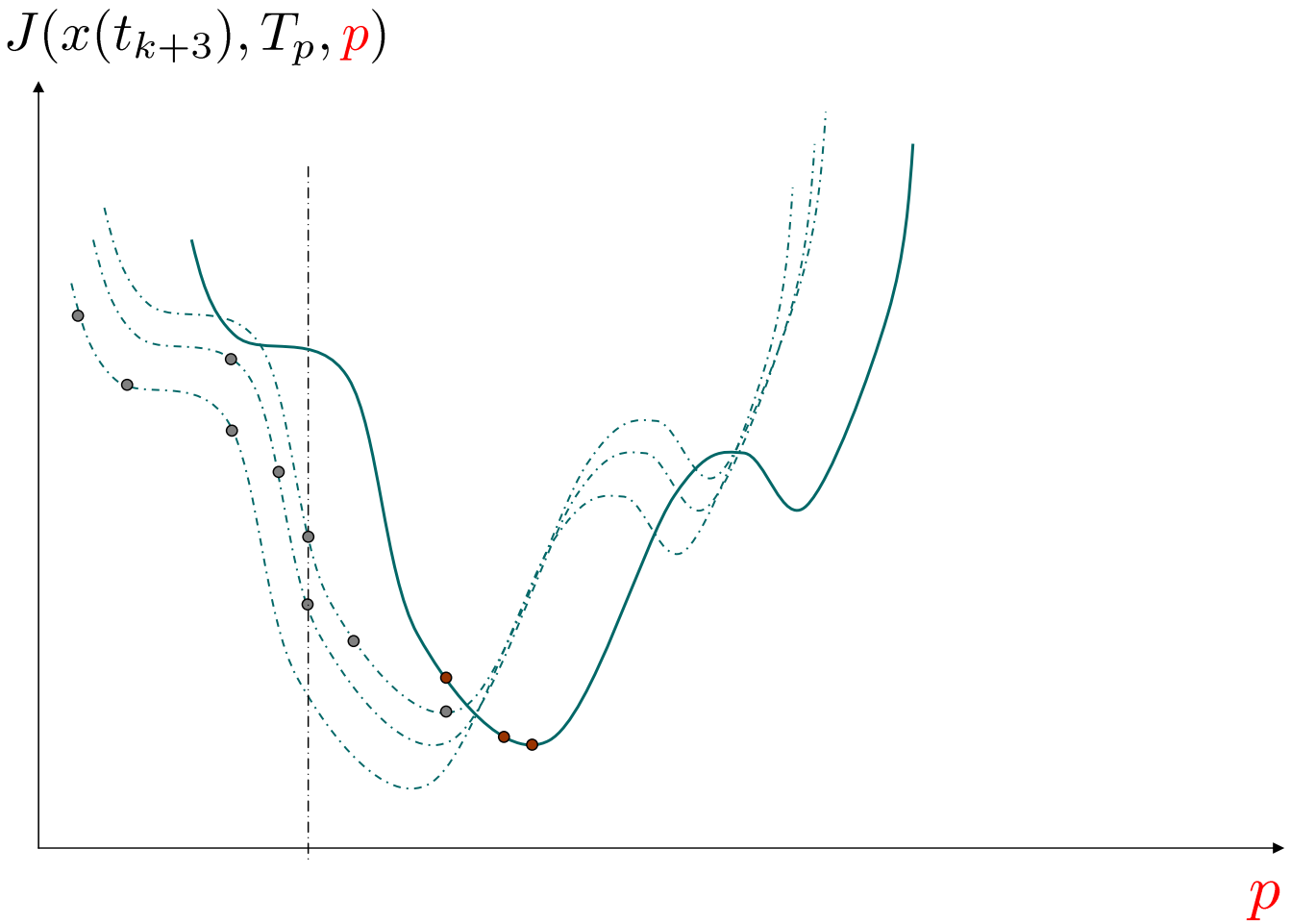


Case $q = 2$

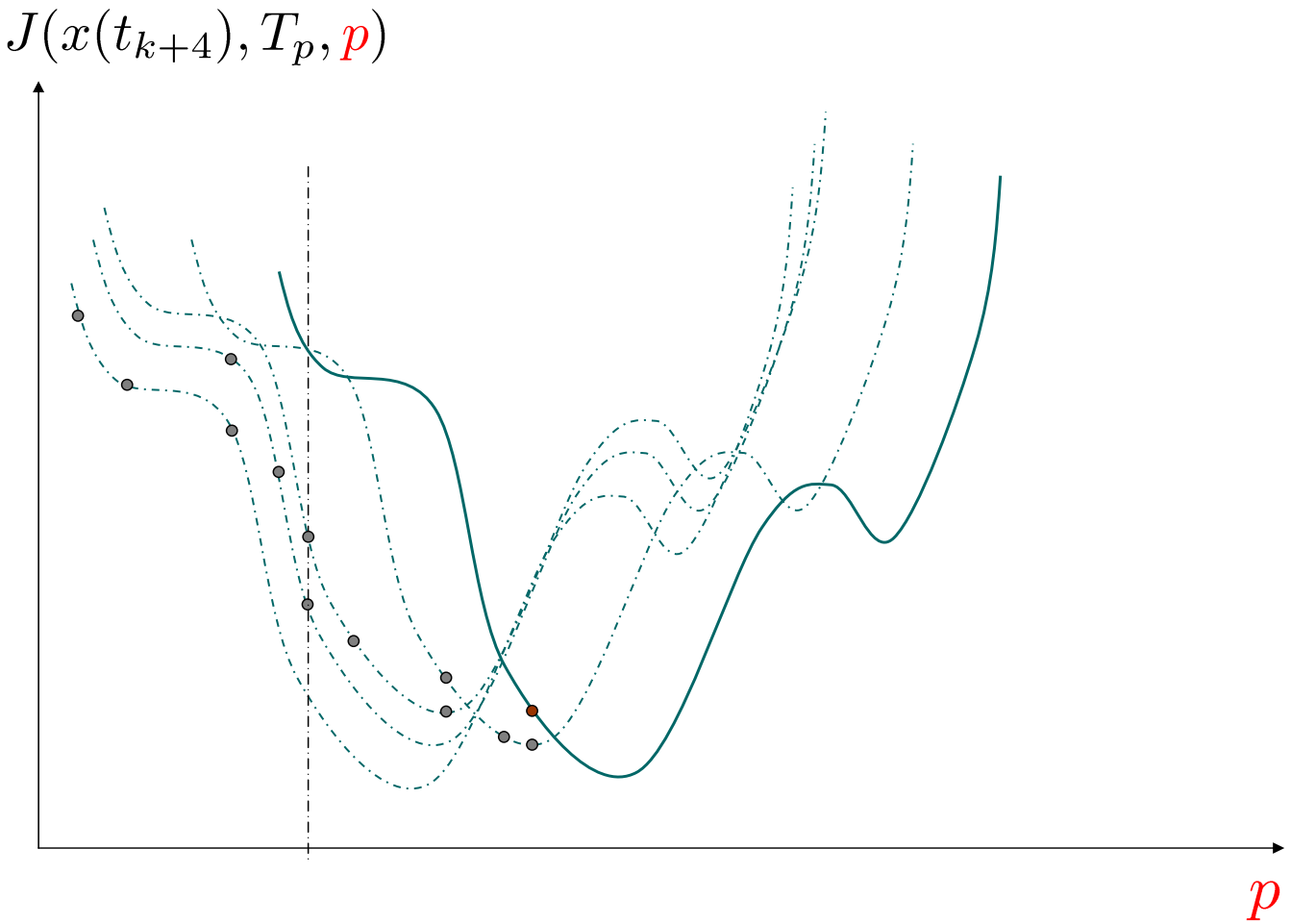
gipsa-lab



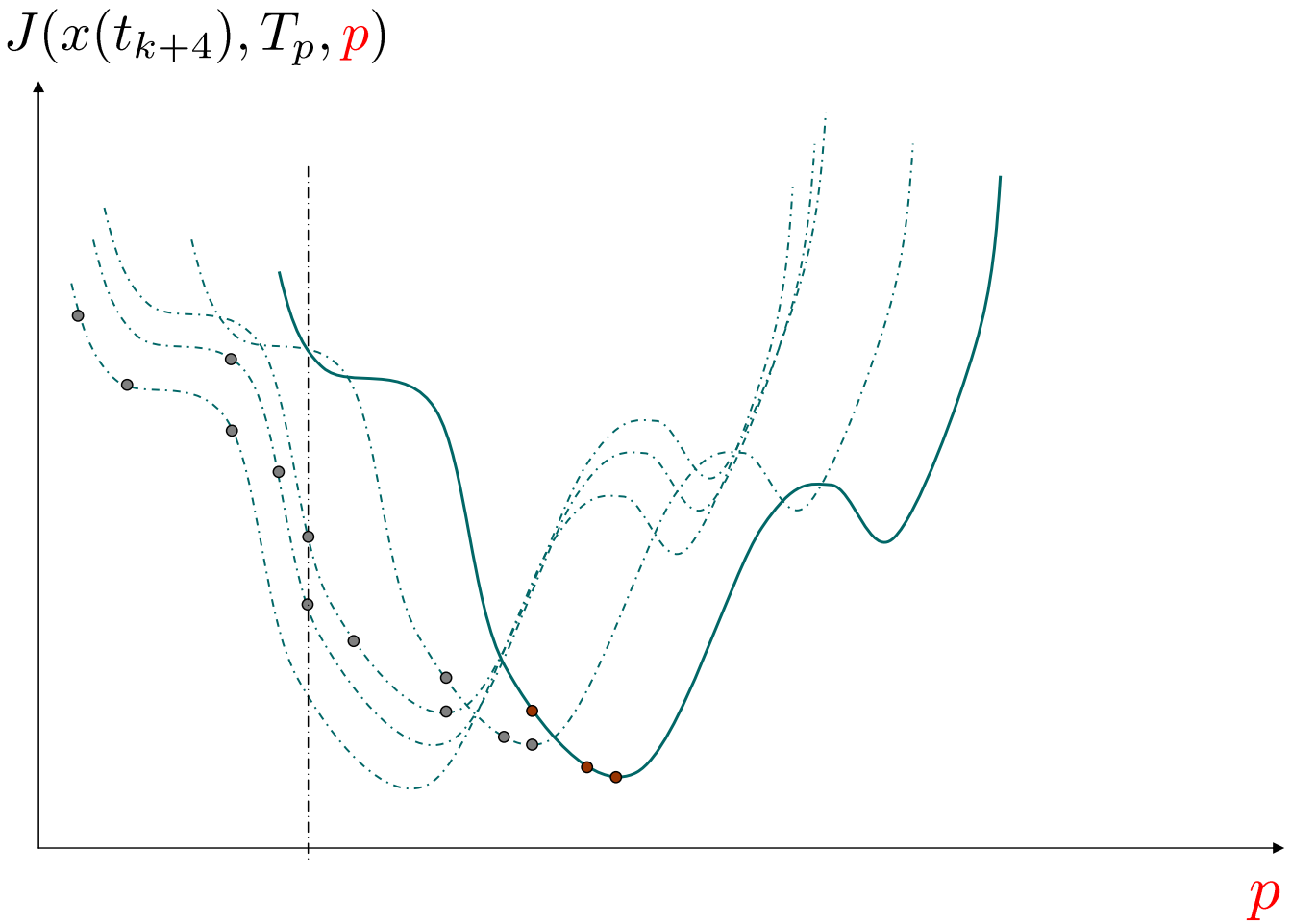
Case $q = 2$



Case $q = 2$

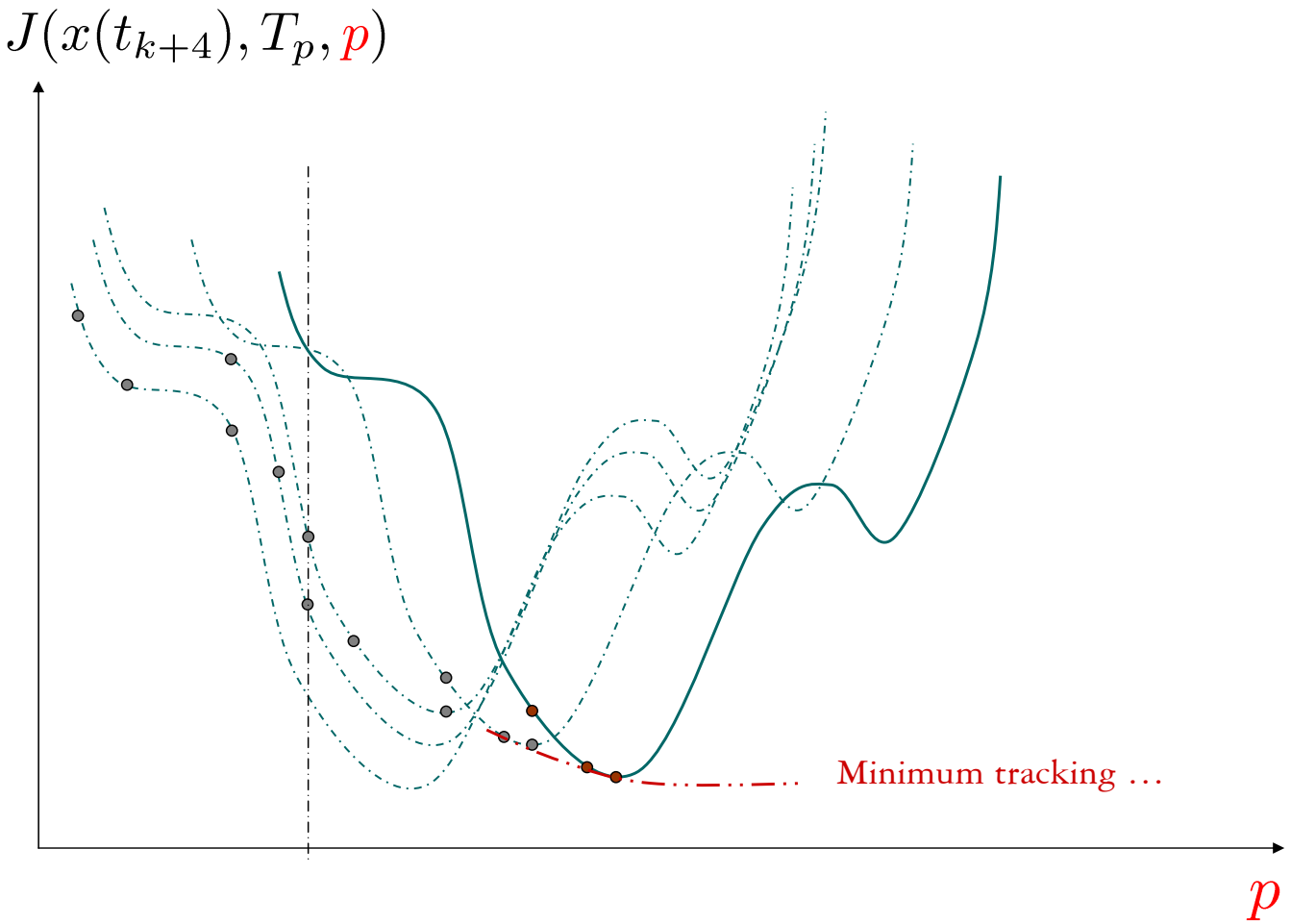


Case $q = 2$

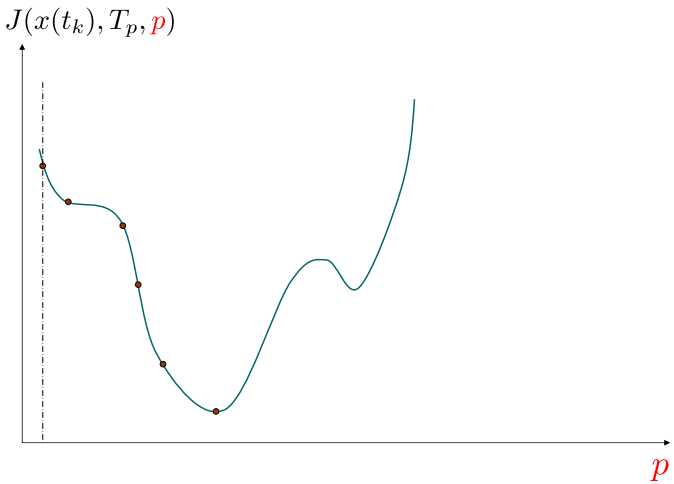


Case $q = 2$

gipsa-lab

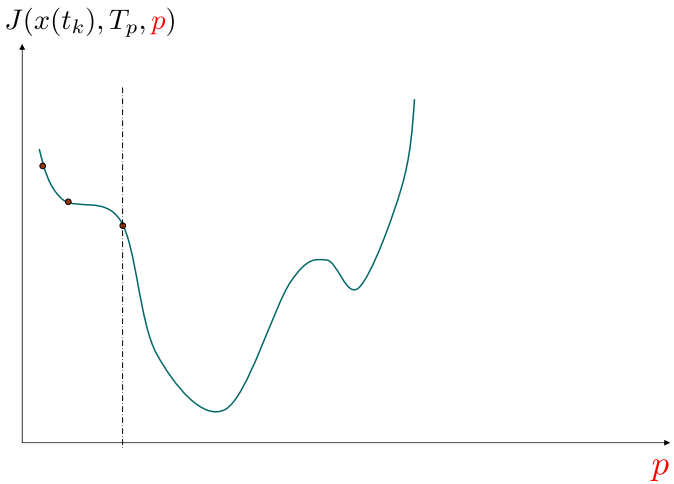


Case $q = 2$



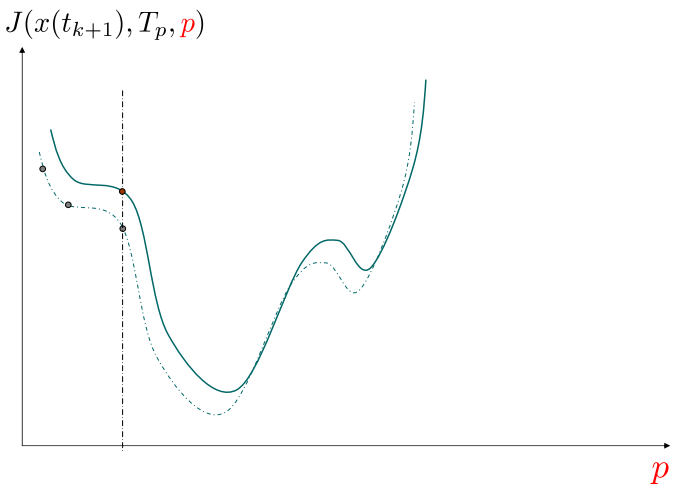
This may work provided that

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



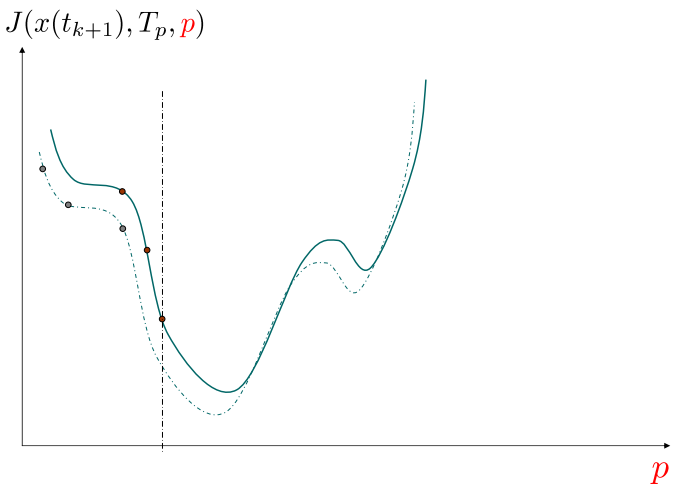
This may work provided that

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



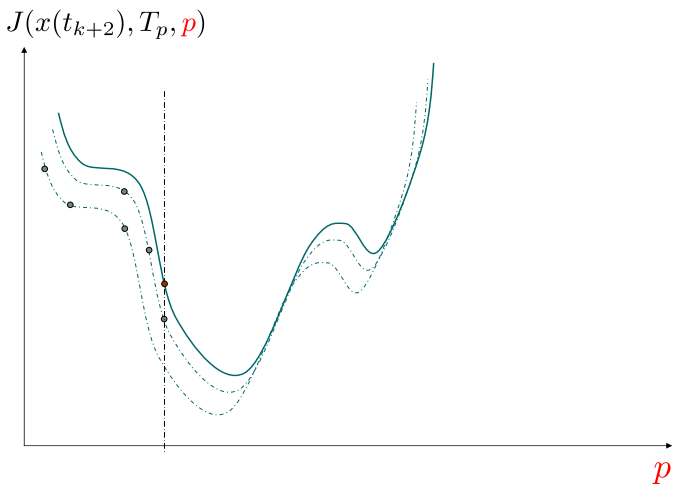
This may work provided that

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



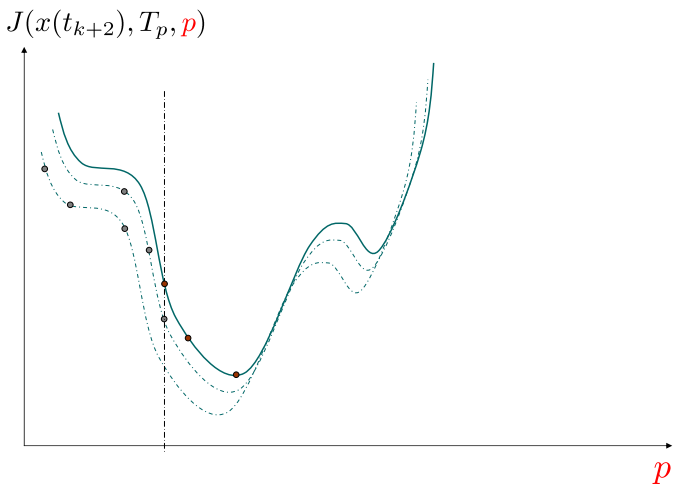
This may work provided that

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



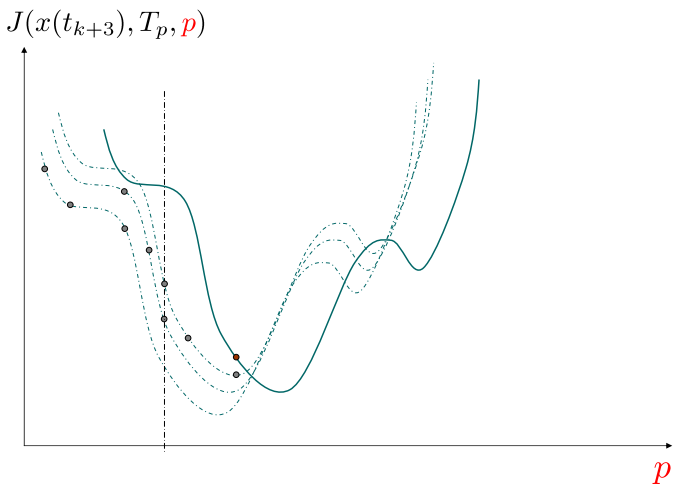
This may work provided that

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



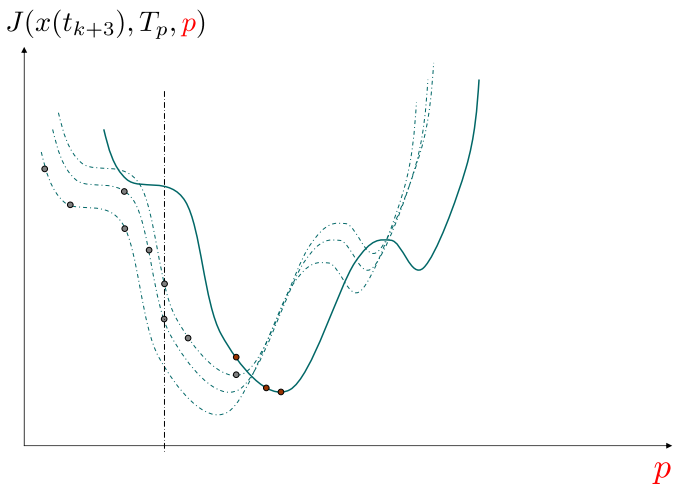
This may work provided that

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



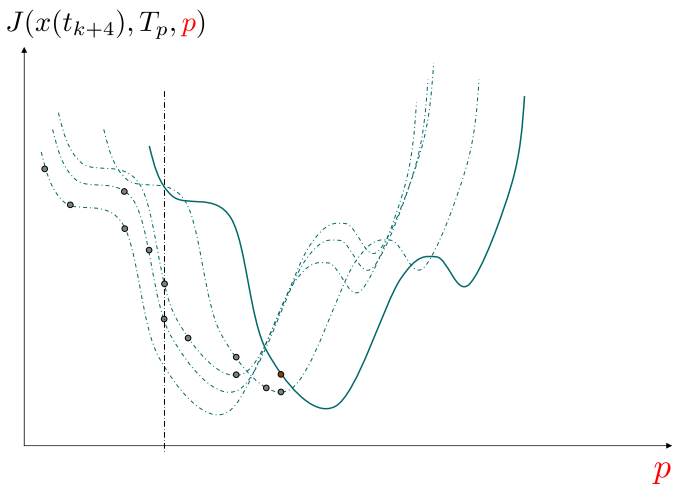
This may work provided that

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



This may work provided that

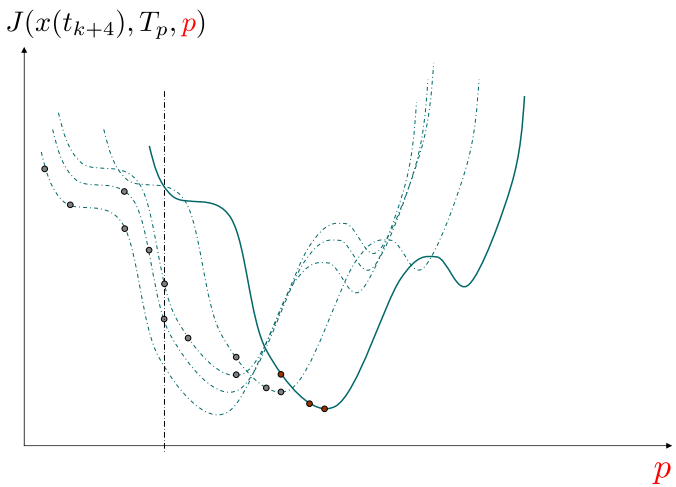
1. $\tau_s \ll$ characteristic time of the system
2. The iteration is efficient
3. No finite escape time in the transient phase
4. The original RH formulation is stabilizing



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

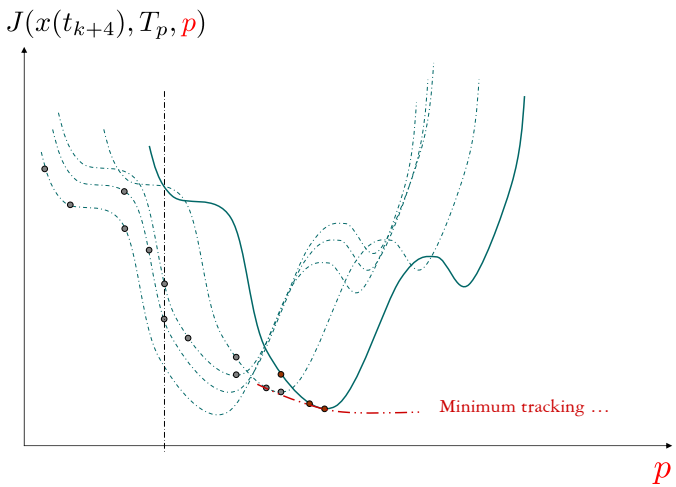
gipsa-lab



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



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

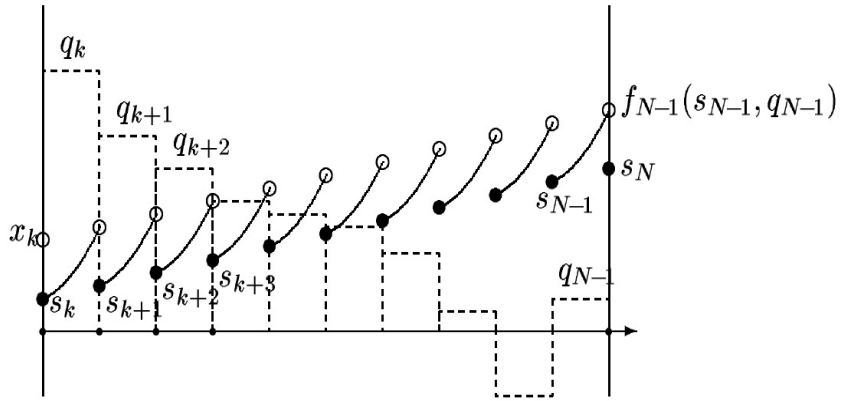
The iterative process $\mathcal{S}$

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

1. Multiple shooting approach [Diehl et al. 2005]
2. Continuation (homotopy) approach [Ohtsuka 2004]
3. Parametric approach

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

$$x_k - s_k = 0$$

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

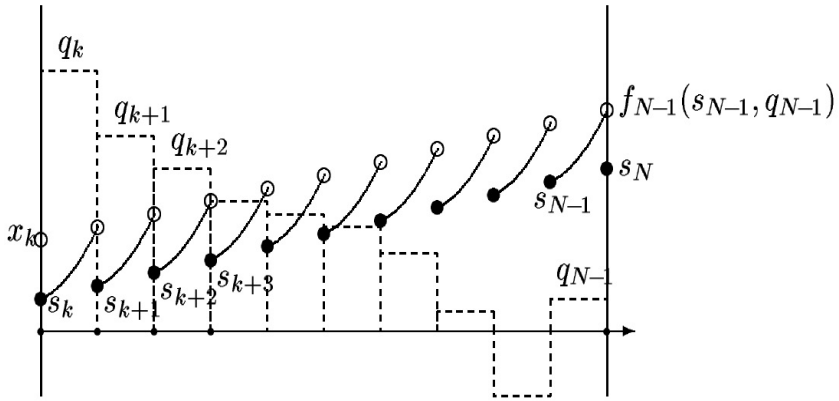
gipsa-lab

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

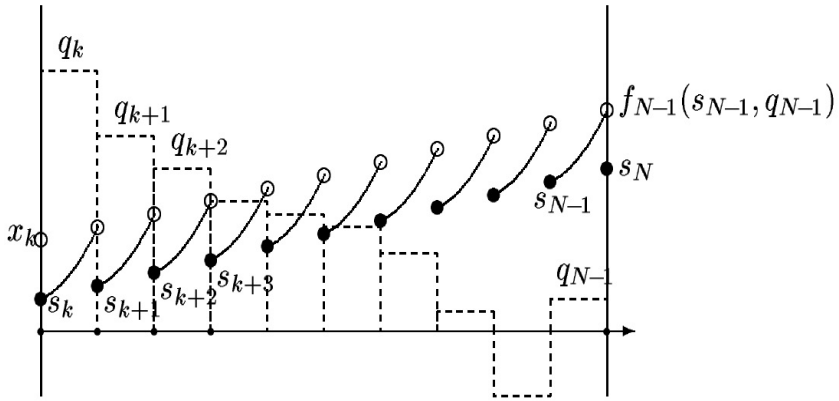
[Diehl et al. 2005]

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]

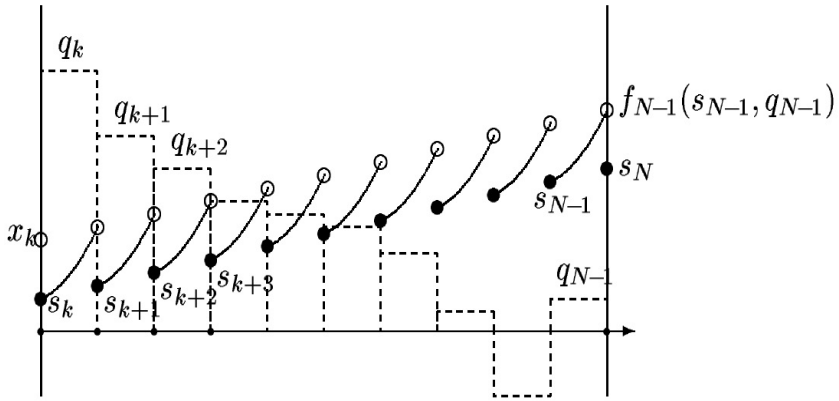
Karush-Kuhn-Tucker optimality condition :

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

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

$$x_k - s_k = 0$$

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

gipsa-lab

The multiple shooting approach

[Diehl et al. 2005]

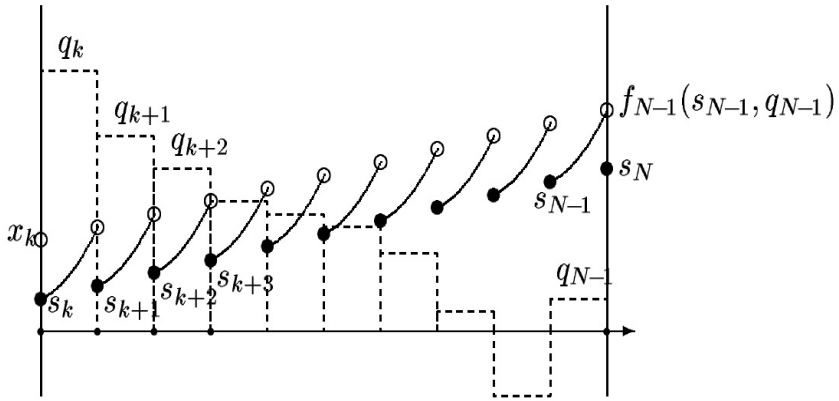
Karush-Kuhn-Tucker optimality condition :

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

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]

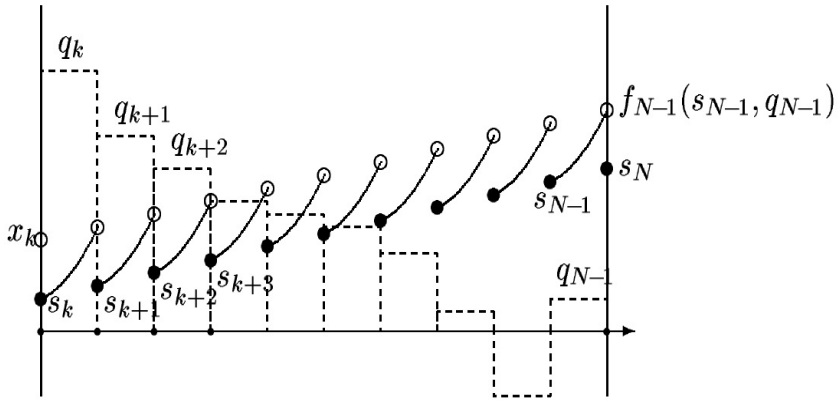
Karush-Kuhn-Tucker optimality condition :

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

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 :

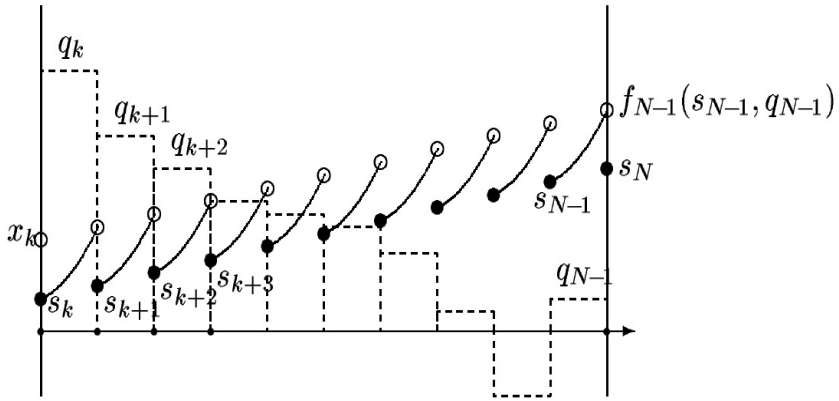
$$\nabla_y \mathcal{L}^k(y) = 0$$

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

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

gipsa-lab

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

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

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

gipsa-lab

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.

$\downarrow$

Complex CL behavior

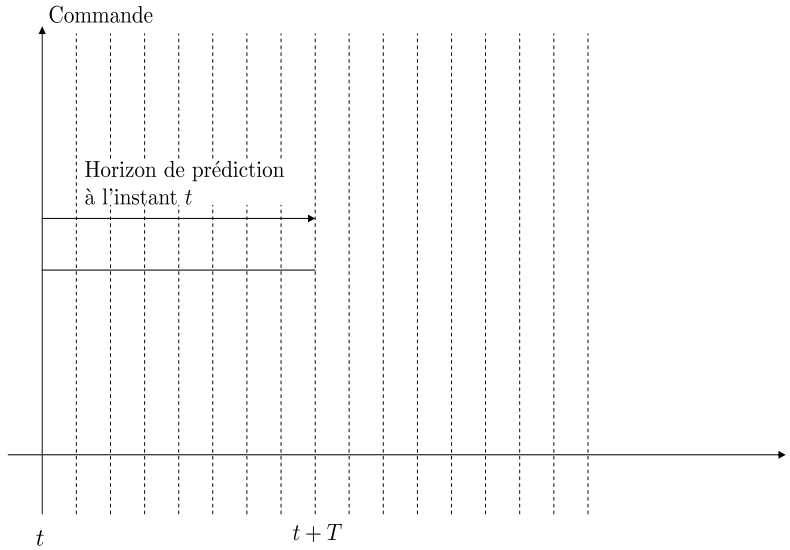
The parametric approach

Fact 1

Simple OL param.

$\downarrow$

Complex CL behavior



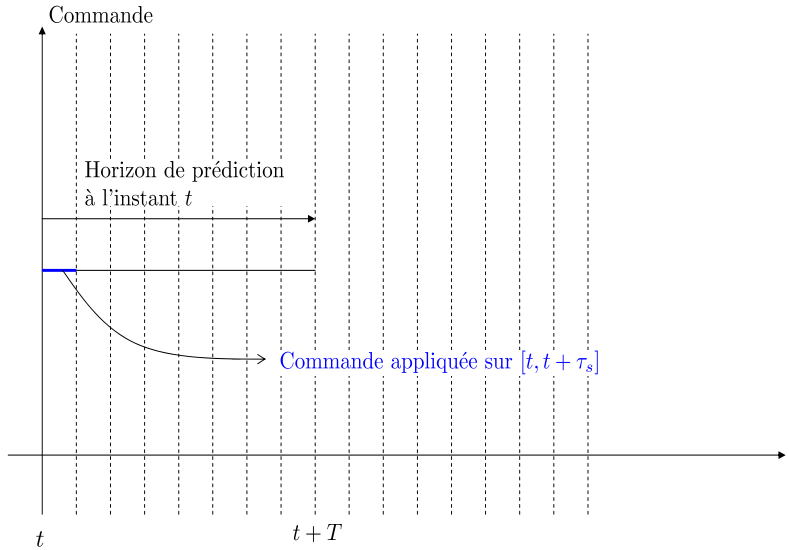
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



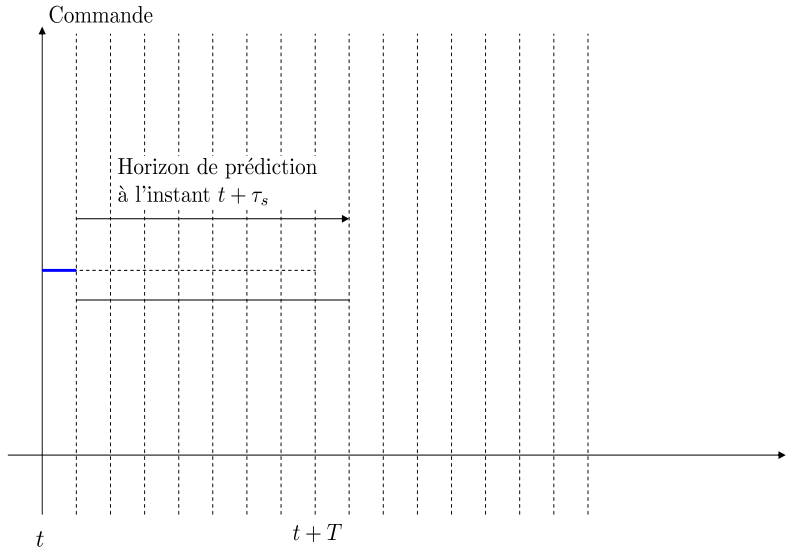
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



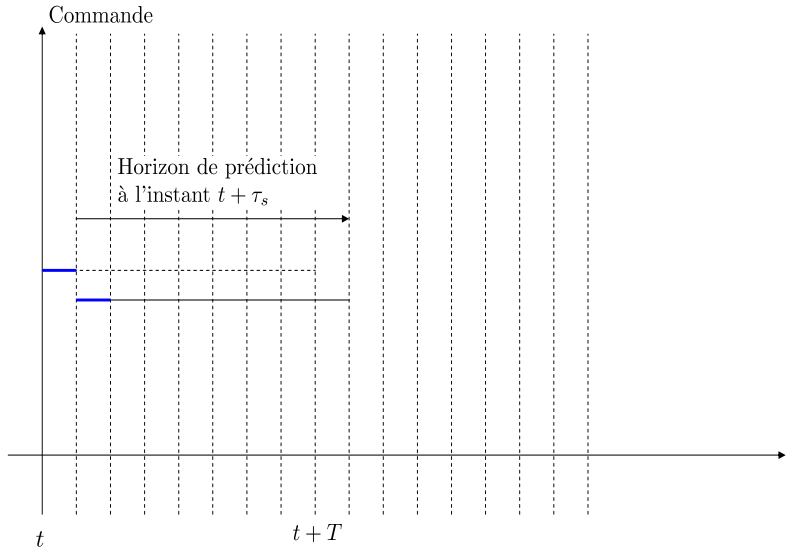
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



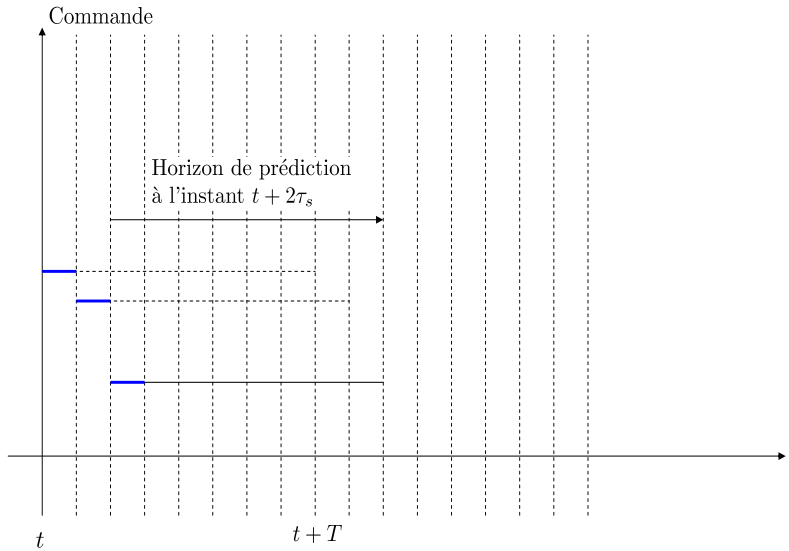
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



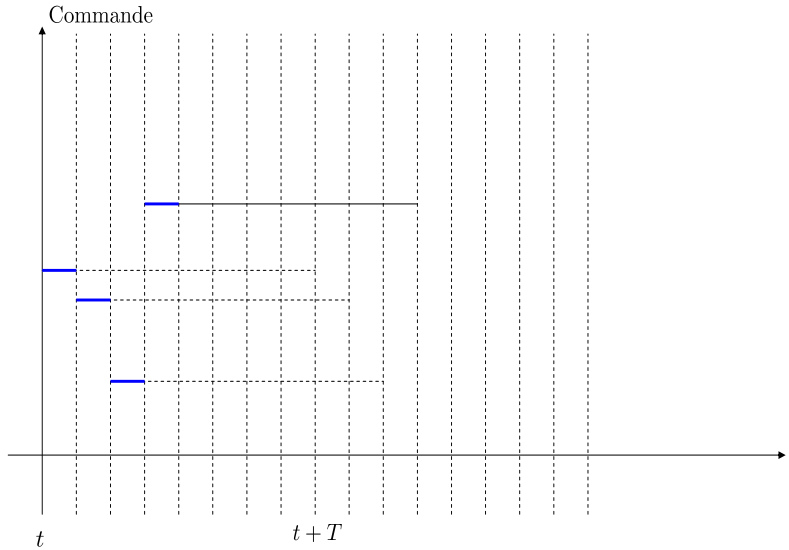
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



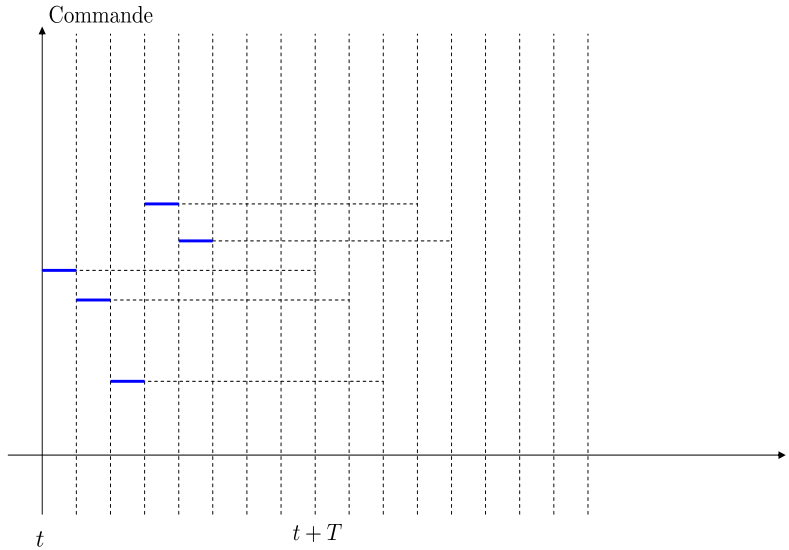
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



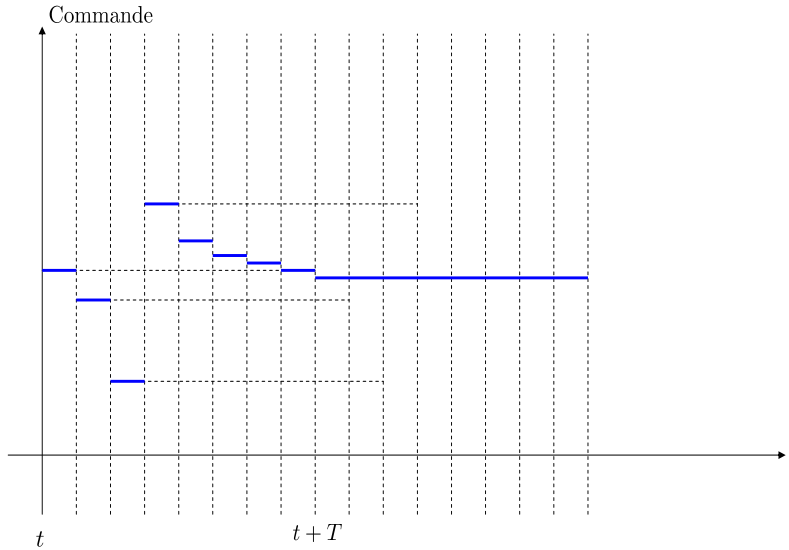
The parametric approach

Fact 1

Simple OL param.

↓

Complex CL behavior



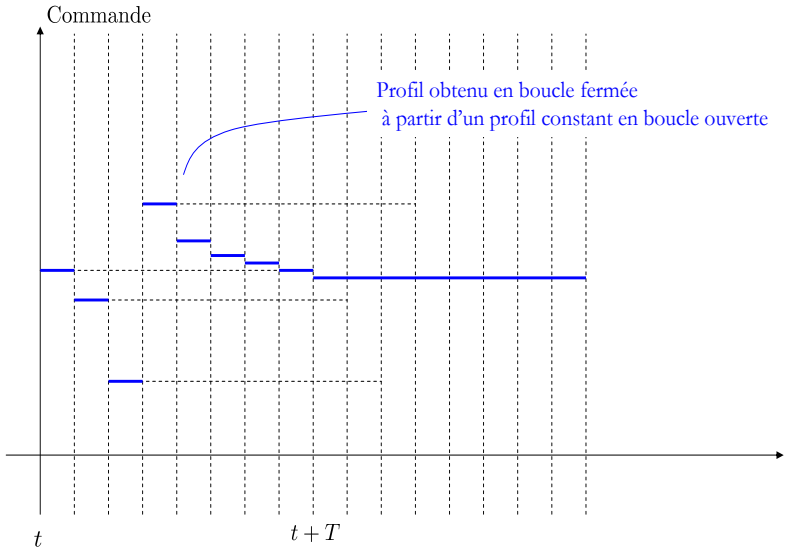
The parametric approach

Fact 1

Simple OL param.

$\downarrow$

Complex CL behavior



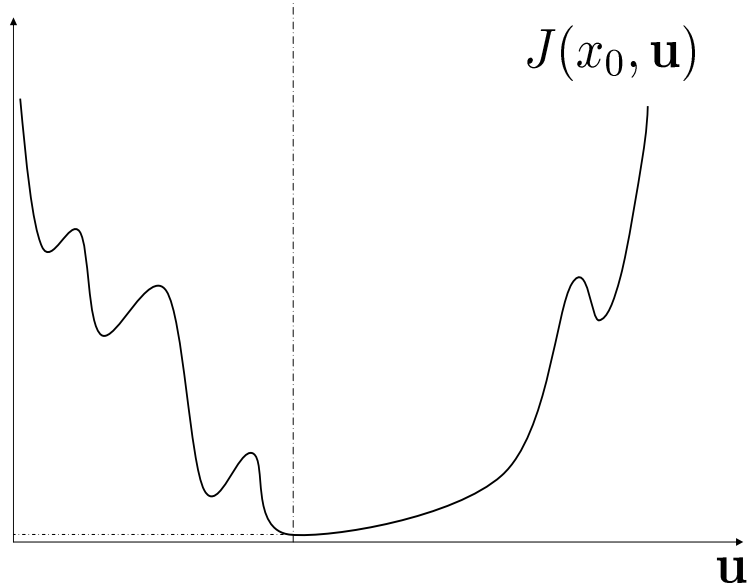
The parametric approach

Fact 2

Solve exactly a sub-optimal formulation

could be better than

Solve loosely the exact problem



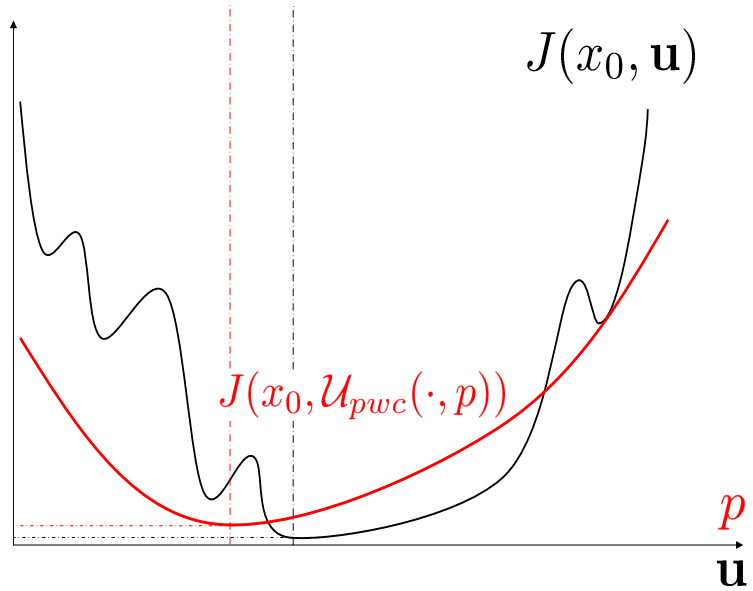
The parametric approach

Fact 2

Solve exactly a sub-optimal formulation

could be better than

Solve loosely the exact problem



gipsa-lab

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

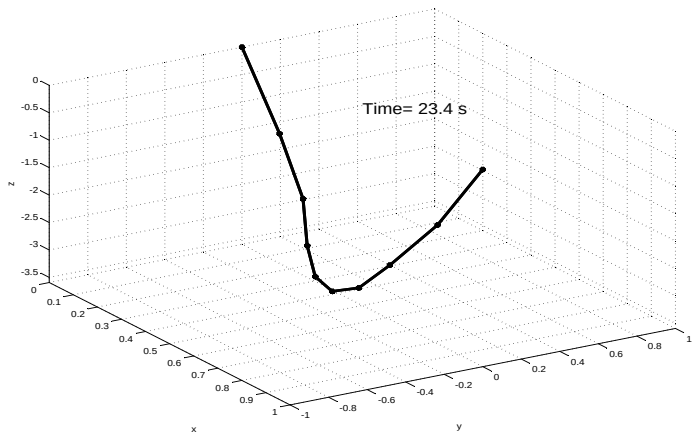
Render efficient

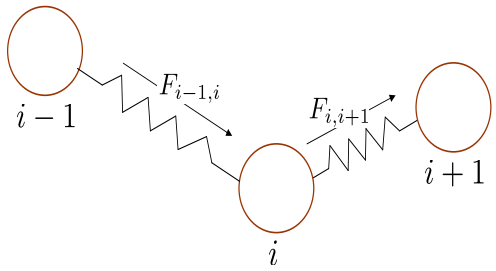
A family of simple, non smooth and memoryless algorithms that would be out of scope for large scale problems

Problème	n_p	temps de calcul (s)
Interception en temps minimal	1	18×10^{-3}
Stabilisation des systèmes chaînés	1	$< 10^{-3}$
Stabilisation d'un avion à décollage vertical	2	$< 10^{-3}$
Simple pendule inversé avec chariot	1	< 0.1
Double pendule inversé avec chariot	2	< 0.25
Stabilisation de satellite en mode défaillant	2	< 0.01

Non damped elastic chain

Non damped elastic chain



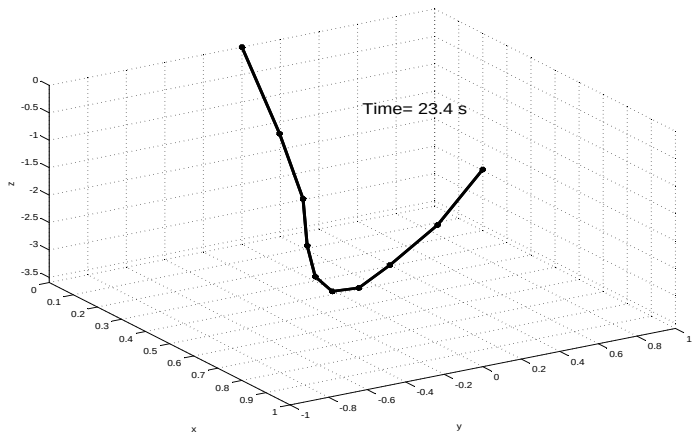


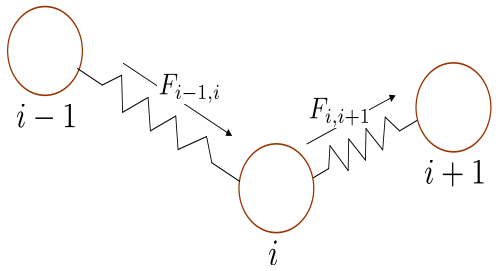
$$\dot{x}_i = \frac{1}{m}\Big(F_{i,i+1} - F_{i-1,i}\Big) + g \quad ; \quad i = 1, \ldots, N$$

$$\dot{x}_{N+1} = u \qquad \dot{x}_i \in \mathbb{R}^3$$

$$F_{i,i+1} = D\left(1 - \frac{L}{\|x_{i+1} - x_i\|}\right)(x_{i+1} - x_i)$$

Non damped elastic chain





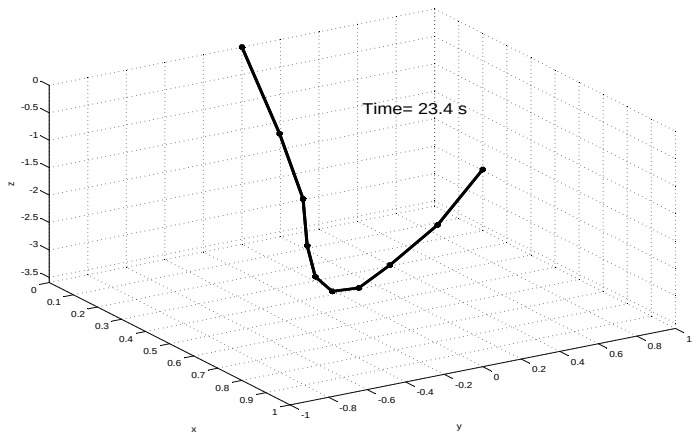
$$\dot{x}_i = \frac{1}{m}\Big(F_{i,i+1} - F_{i-1,i}\Big) + g \quad ; \quad i = 1, \ldots, N$$

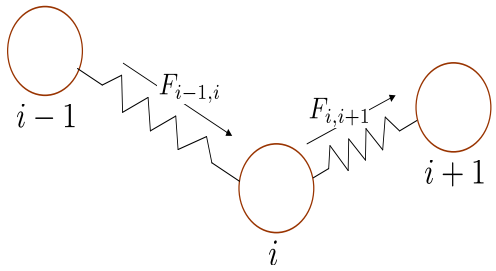
$$\dot{x}_{N+1} = u \qquad \dot{x}_i \in \mathbb{R}^3$$

$$F_{i,i+1} = D\left(1 - \frac{L}{\|x_{i+1} - x_i\|}\right)(x_{i+1} - x_i)$$

- 10 balls $\longrightarrow$
  - $(9 \times 3 \times 2) - 3 = 51$ states
  - 3 Controls

Non damped elastic chain





$$\dot{x}_i = \frac{1}{m}\Big(F_{i,i+1} - F_{i-1,i}\Big) + g \quad ; \quad i = 1, \ldots, N$$

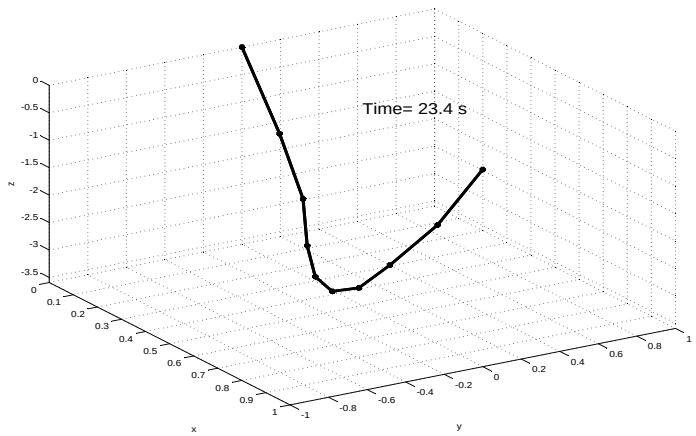
$$\dot{x}_{N+1} = u \qquad \dot{x}_i \in \mathbb{R}^3$$

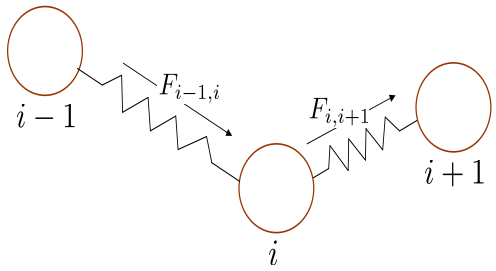
$$F_{i,i+1} = D\left(1 - \frac{L}{\|x_{i+1} - x_i\|}\right)(x_{i+1} - x_i)$$

- 10 balls $\longrightarrow$
  - $(9 \times 3 \times 2) - 3 = 51$ states
  - 3 Controls
- Saturation constraint

$$u \in [-u_{max}, +u_{max}]^3$$

Non damped elastic chain





$$\dot{x}_i = \frac{1}{m}\Big(F_{i,i+1} - F_{i-1,i}\Big) + g \quad ; \quad i = 1, \ldots, N$$

$$\dot{x}_{N+1} = u \qquad \dot{x}_i \in \mathbb{R}^3$$

$$F_{i,i+1} = D\left(1 - \frac{L}{\|x_{i+1} - x_i\|}\right)(x_{i+1} - x_i)$$

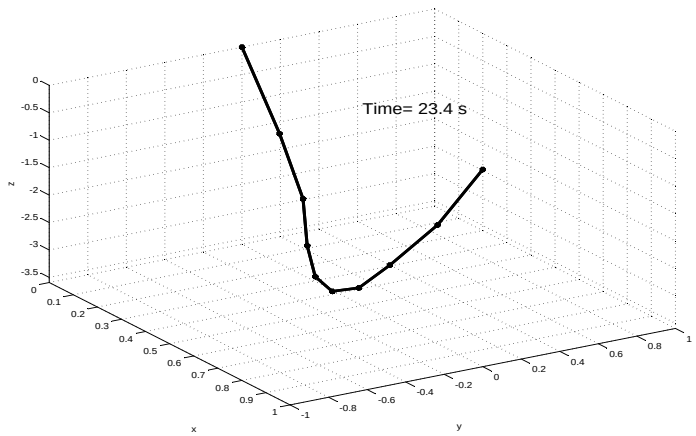
- 10 balls $\longrightarrow$
  - $(9 \times 3 \times 2) - 3 = 51$ states
  - 3 Controls
- Saturation constraint

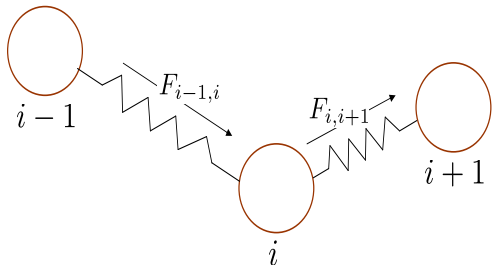
  $u \in [-u_{max}, +u_{max}]^3$
- State constraints

  $P_{N+1} \in P_f + [-\Delta P_{max}, +\Delta P_{max}]^3$
  $P_{N+1} \longrightarrow P_f$

gipsa-lab

Non damped elastic chain





$$\dot{x}_i = \frac{1}{m}\Big(F_{i,i+1} - F_{i-1,i}\Big) + g \quad ; \quad i = 1, \ldots, N$$

$$\dot{x}_{N+1} = u \qquad \dot{x}_i \in \mathbb{R}^3$$

$$F_{i,i+1} = D\left(1 - \frac{L}{\|x_{i+1} - x_i\|}\right)(x_{i+1} - x_i)$$

- 10 balls $\longrightarrow$
  - $(9 \times 3 \times 2) - 3 = 51$ states
  - 3 Controls
- Saturation constraint

$$u \in [-u_{max}, +u_{max}]^3$$

- State constraints

$$P_{N+1} \in P_f + [-\Delta P_{max}, +\Delta P_{max}]^3$$
$$P_{N+1} \longrightarrow P_f$$

- The cost function

$$J(x_0, T_p, \mathbf{u}) := \int_0^{T_p} \Big[\sum_{i=1}^{N_b} \dot{x}_i^2(\tau)\Big] d\tau$$

gipsa-lab

Control Parametrization

Use the parameterized explicit feedback :

$$u = Sat_{-u_{max}}^{+u_{max}} \left(-\lambda \left[P_{N+1} - P_{N+1}^{d}(p) \right] \right)$$

Control Parametrization

Use the parameterized explicit feedback :

$$u = Sat_{-u_{max}}^{+u_{max}} \left(-\lambda \left[P_{N+1} - P_{N+1}^{d}(p) \right] \right)$$

where the *desired position* $P_{N+1}^{d}(p)$ is given by :

$$P_{N+1}^{d}(p) := P_f + \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} e^{-\lambda_0 t} + \begin{pmatrix} p_4 \\ p_5 \\ p_6 \end{pmatrix} e^{-2\lambda_0 t} + \dots$$

where $p_i \in [-\Delta P_{max}, +\Delta P_{max}]$

Control Parametrization

Use the parameterized explicit feedback :

$$u = Sat_{-u_{max}}^{+u_{max}} \left(-\lambda \left[P_{N+1} - P_{N+1}^{d}(p) \right] \right)$$

where the *desired position* $P_{N+1}^{d}(p)$ is given by :

$$P_{N+1}^{d}(p) := P_f + \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} e^{-\lambda_0 t} + \begin{pmatrix} p_4 \\ p_5 \\ p_6 \end{pmatrix} e^{-2\lambda_0 t} + \dots$$

where $p_i \in [-\Delta P_{max}, +\Delta P_{max}]$

$\longrightarrow$ The cost function becomes $J(x_0, T_p, p)$

The iteration process $\mathcal{S}$

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

At iteration (i),

- Q-approximation of the scalar function $J(p_{k_i})$ together with a trust region updating. $\rightarrow$ compute dp_{k_i}
- Perform the update $p^{(i)} \leftarrow p^{(i)} + dp_{k_i}$ when appropriate
- L-approximation of the scalar function $J(p^{(i)} + \alpha G^{(i)})$ where

$$G^{(i)} = f(dp_1^{(i_1)}, dp_2^{(i_2)}, \ldots, dp_{n_p}^{(i_{n_p})}, \beta)$$

- Perform as many successful steps as possible along along $G^{(i)}$
- $i \leftarrow i+1$, $k_i \leftarrow \text{successor}(k_i)$

gipsa-lab

The iteration process $\mathcal{S}$

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

At iteration (i),

- Q-approximation of the scalar function $J(p_{k_i})$ together with a trust region updating. $\rightarrow$ compute dp_{k_i}
- Perform the update $p^{(i)} \leftarrow p^{(i)} + dp_{k_i}$ when appropriate
- L-approximation of the scalar function $J(p^{(i)} + \alpha G^{(i)})$ where

$$G^{(i)} = f(dp_1^{(i_1)}, dp_2^{(i_2)}, \ldots, dp_{n_p}^{(i_{n_p})}, \beta)$$

- Perform as many successful steps as possible along along $G^{(i)}$
- $i \leftarrow i+1$, $k_i \leftarrow \text{successor}(k_i)$

gipsa-lab

The iteration process $\mathcal{S}$

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

At iteration (i),

- Q-approximation of the scalar function $J(p_{k_i})$ together with a trust region updating. $\rightarrow$ compute dp_{k_i}
- Perform the update $p^{(i)} \leftarrow p^{(i)} + dp_{k_i}$ when appropriate
- L-approximation of the scalar function $J(p^{(i)} + \alpha G^{(i)})$ where

$$G^{(i)} = f(dp_1^{(i_1)}, dp_2^{(i_2)}, \ldots, dp_{n_p}^{(i_{n_p})}, \beta)$$

- Perform as many successful steps as possible along along $G^{(i)}$
- $i \leftarrow i+1$, $k_i \leftarrow \text{successor}(k_i)$

gipsa-lab

The iteration process $\mathcal{S}$

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

At iteration (i),

- Q-approximation of the scalar function $J(p_{k_i})$ together with a trust region updating. $\rightarrow$ compute dp_{k_i}
- Perform the update $p^{(i)} \leftarrow p^{(i)} + dp_{k_i}$ when appropriate
- L-approximation of the scalar function $J(p^{(i)} + \alpha G^{(i)})$ where

$$G^{(i)} = f(dp_1^{(i_1)}, dp_2^{(i_2)}, \ldots, dp_{n_p}^{(i_{n_p})}, \beta)$$

- Perform as many successful steps as possible along along $G^{(i)}$
- $i \leftarrow i+1$, $k_i \leftarrow \text{successor}(k_i)$

gipsa-lab

The iteration process $\mathcal{S}$

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

At iteration (i),

- Q-approximation of the scalar function $J(p_{k_i})$ together with a trust region updating. $\rightarrow$ compute dp_{k_i}
- Perform the update $p^{(i)} \leftarrow p^{(i)} + dp_{k_i}$ when appropriate
- L-approximation of the scalar function $J(p^{(i)} + \alpha G^{(i)})$ where

$$G^{(i)} = f(dp_1^{(i_1)}, dp_2^{(i_2)}, \ldots, dp_{n_p}^{(i_{n_p})}, \beta)$$

- Perform as many successful steps as possible along along $G^{(i)}$
- $i \leftarrow i+1$, $k_i \leftarrow \text{successor}(k_i)$

gipsa-lab

The iteration process $\mathcal{S}$

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

At iteration (i),

- Q-approximation of the scalar function $J(p_{k_i})$ together with a trust region updating. $\rightarrow$ compute dp_{k_i}
- Perform the update $p^{(i)} \leftarrow p^{(i)} + dp_{k_i}$ when appropriate
- L-approximation of the scalar function $J(p^{(i)} + \alpha G^{(i)})$ where

$$G^{(i)} = f(dp_1^{(i_1)}, dp_2^{(i_2)}, \ldots, dp_{n_p}^{(i_{n_p})}, \beta)$$

- Perform as many successful steps as possible along along $G^{(i)}$
- $i \leftarrow i+1$, $k_i \leftarrow \text{successor}(k_i)$

The iteration process $\mathcal{S}$

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

At iteration (i),

- Q-approximation of the scalar function $J(p_{k_i})$ together with a trust region updating. $\rightarrow$ compute dp_{k_i}
- Perform the update $p^{(i)} \leftarrow p^{(i)} + dp_{k_i}$ when appropriate
- L-approximation of the scalar function $J(p^{(i)} + \alpha G^{(i)})$ where

$$G^{(i)} = f(dp_1^{(i_1)}, dp_2^{(i_2)}, \ldots, dp_{n_p}^{(i_{n_p})}, \beta)$$

- Perform as many successful steps as possible along along $G^{(i)}$
- $i \leftarrow i+1$, $k_i \leftarrow \text{successor}(k_i)$

gipsa-lab

The iteration process $\mathcal{S}$

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

At iteration (i),

- Q-approximation of the scalar function $J(p_{k_i})$ together with a trust region updating. $\rightarrow$ compute dp_{k_i}
- Perform the update $p^{(i)} \leftarrow p^{(i)} + dp_{k_i}$ when appropriate
- L-approximation of the scalar function $J(p^{(i)} + \alpha G^{(i)})$ where

$$G^{(i)} = f(dp_1^{(i_1)}, dp_2^{(i_2)}, \ldots, dp_{n_p}^{(i_{n_p})}, \beta)$$

- Perform as many successful steps as possible along along $G^{(i)}$
- $i \leftarrow i+1$, $k_i \leftarrow \text{successor}(k_i)$

gipsa-lab

The iteration process $\mathcal{S}$

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

- Each Q-approximation : 3 cost function evaluations
- Each L-approximation along the gradient : 1 cost function evaluation

q is the number of allowed cost function evaluations during a sampling period.

Scenario description

1. Sampling period for control $\tau_s = 0.2$ *sec*
2. Platform PC-P4, 3.2GHz, 1Go RAM
3. Initial disturbance :

$$u = Sat_{-u_{max}}^{+u_{max}} \left(-\lambda \left[P_{N+1} - P_{N+1}^d(p) \right] \right) \quad ; \quad P_{N+1}^d(p) := P_f + \begin{pmatrix} 0.5 \\ 4 \\ 4 \end{pmatrix} e^{-\lambda_0 t}$$

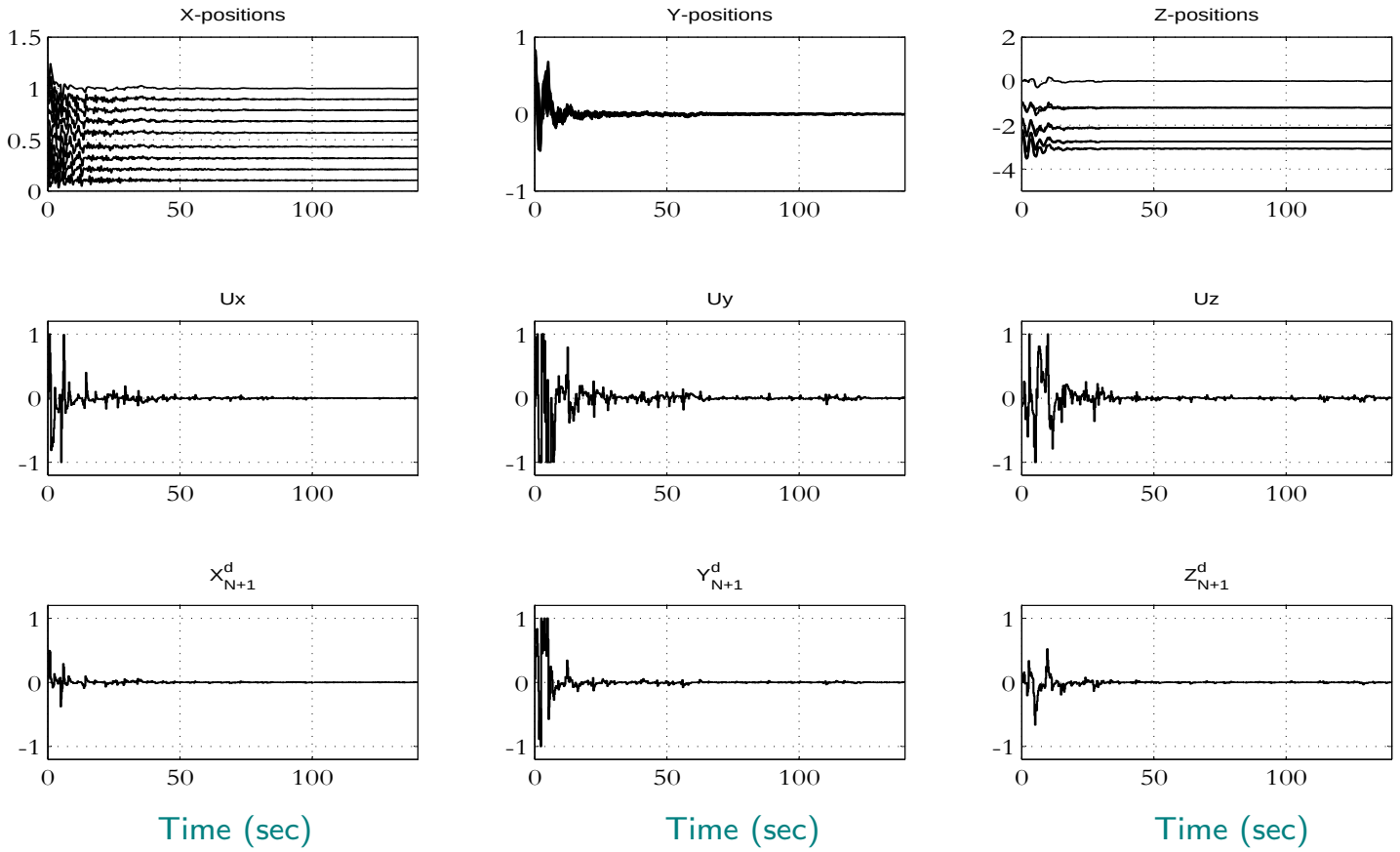
   activation of the controller 5 *sec* later.

4. Prediction horizon $T_p = 5$ *sec* or $T_p = 8$ *sec*.
5. Number or parameters $n_p \in \{3, 6, 9\}$
6. $\lambda = 1$ and $\lambda_0 = 2$
7. Maximal excursion of the end ball $\Delta P_{max} \in \{0.2m, 1m\}$

Ball positions

Control inputs

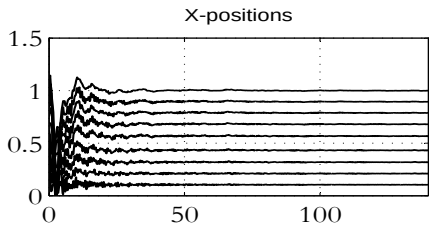
P^d_{N+1}

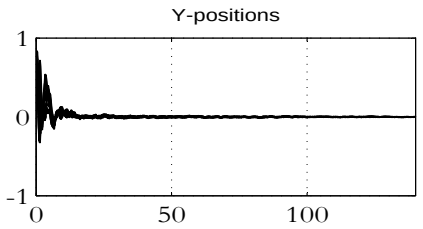


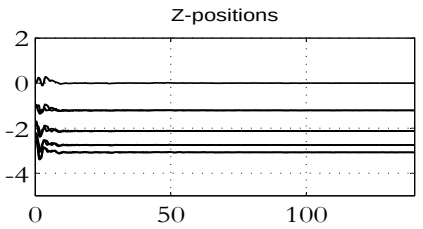
$\Delta P_{max} = 1 \quad ; \quad n_p = 3 \quad ; \quad q = 10$

gipsa-lab

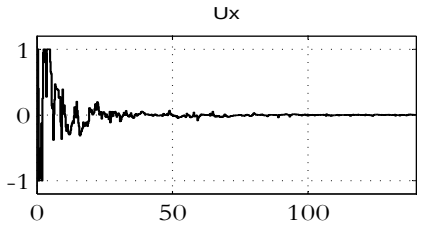
Ball positions

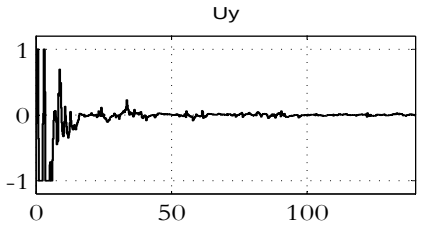


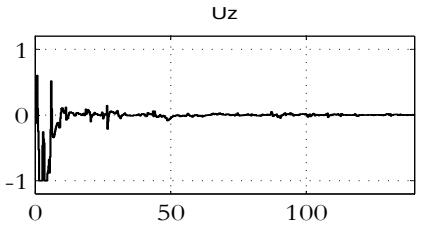




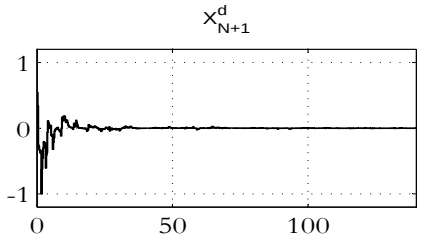
Control inputs

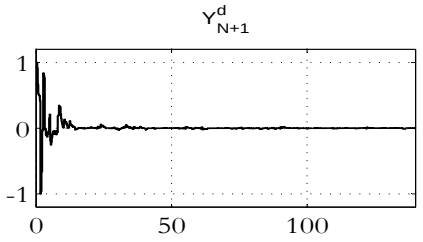


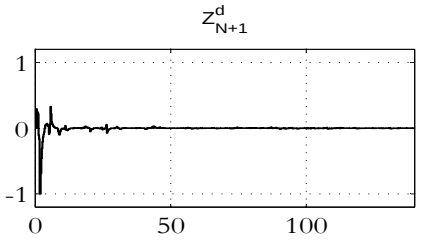




P^d_{N+1}



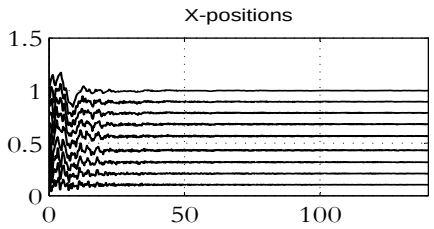


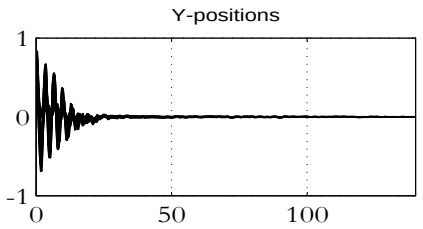


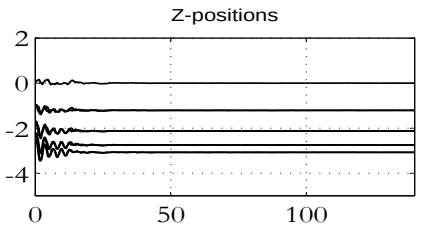
$\Delta P_{max} = 1 \quad ; \quad n_p = 6 \quad ; \quad q = 10$

gipsa-lab

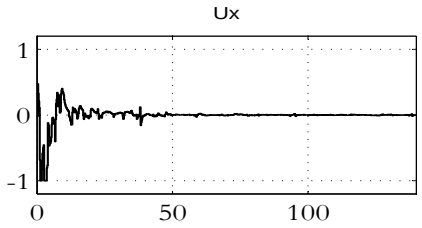
Ball positions

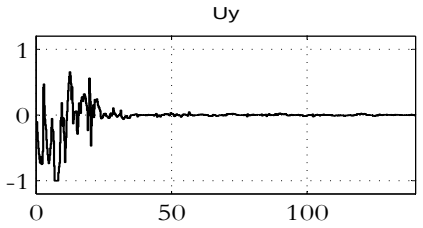


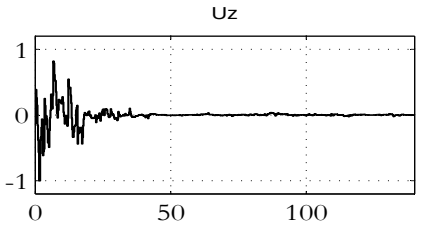




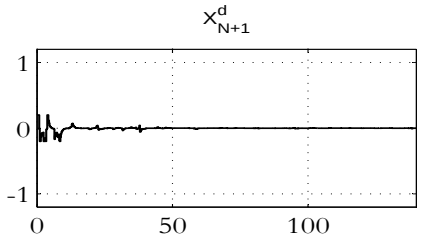
Control inputs

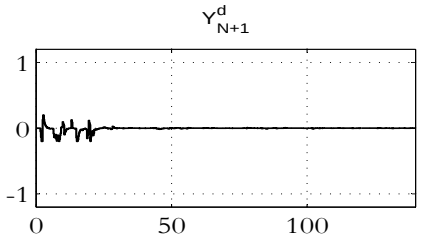


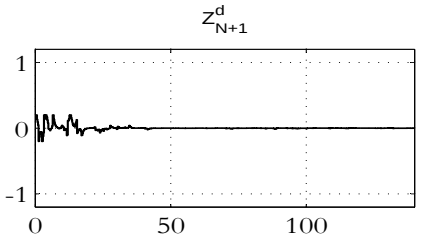




P^d_{N+1}



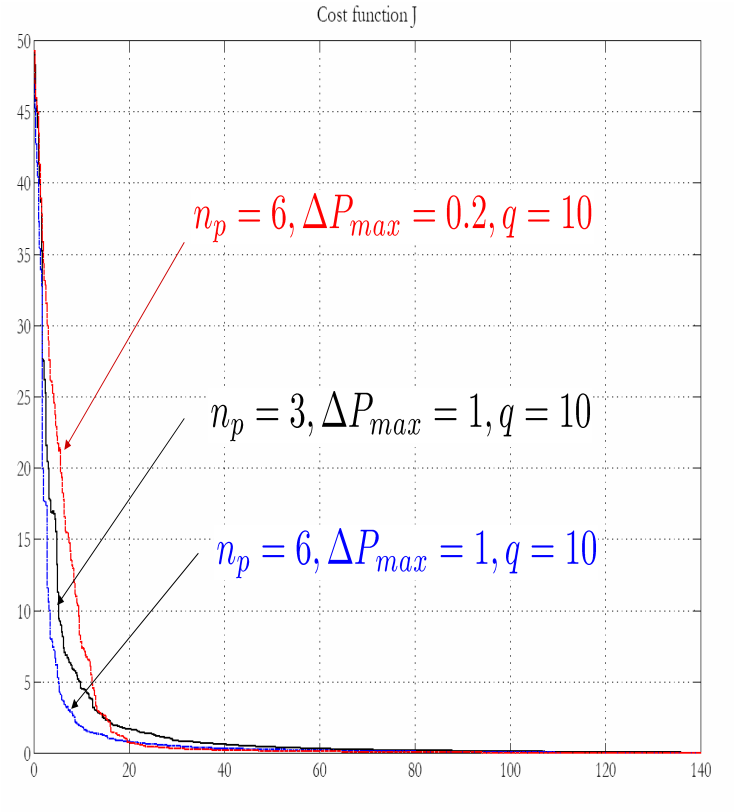


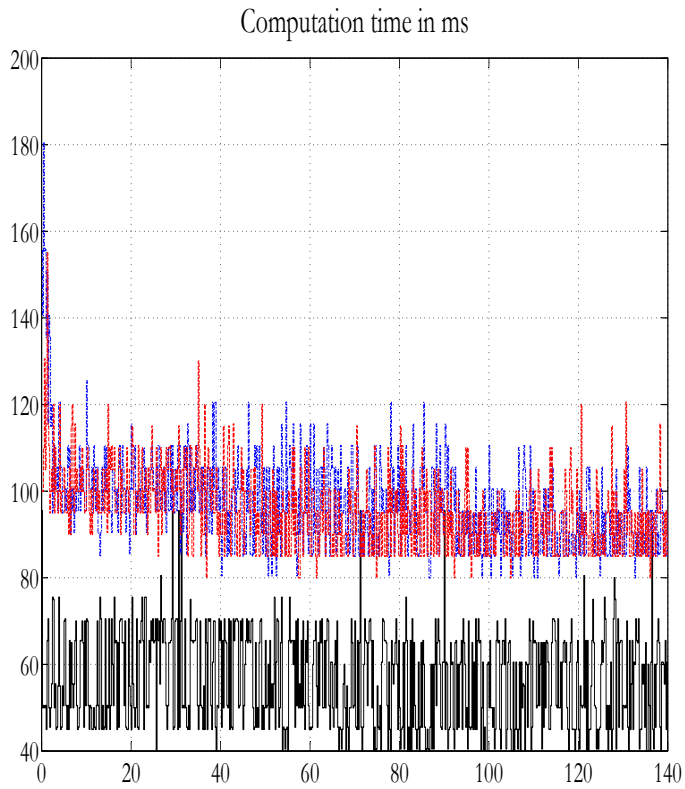


$\Delta P_{max} = 0.2 \quad ; \quad n_p = 6 \quad ; \quad q = 10$

gipsa-lab

Evolution of $J = \sum_{i=1}^{N_b} \dot{x}_i^2$ in closed-loop / Computation time (ms)

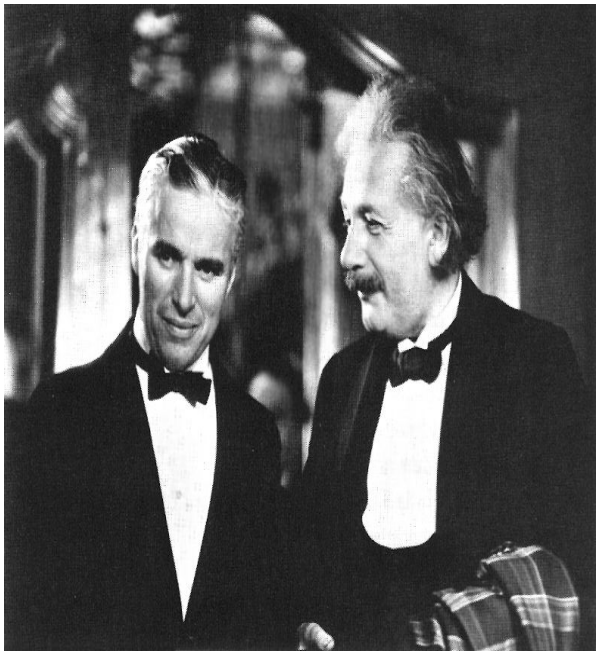




gipsa-lab

Animations with $n_p = 6, q = 10$

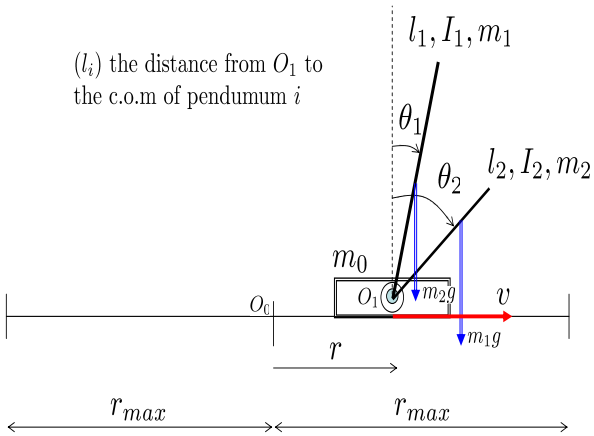
$\Delta P_{max} = 1$



$\Delta P_{max} = 0.2$

gipsa-lab

State constrained Swing-up and stabilization of a twin-pendulum system



[M. Alamir & A. Murilo Automatica 2007]

- Constrained State Feedback
- Swing-up & Stabilization
- Saturations on control & State
- Fast Constrained NMPC
- Sampling period 0.1 *sec*

gipsa-lab

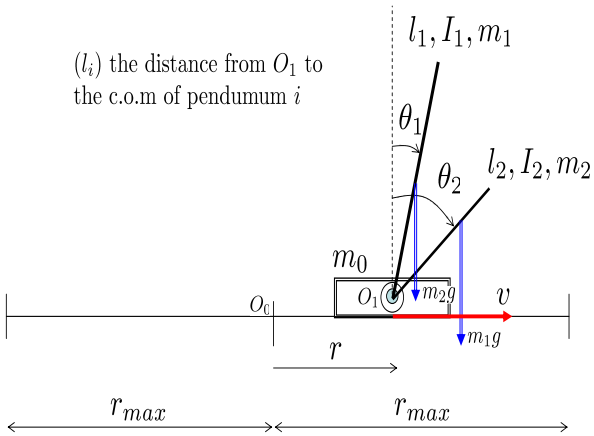
State constrained Swing-up and stabilization of a twin-pendulum system

[M. Alamir & A. Murilo Automatica 2007]

- Constrained State Feedback
- Swing-up & Stabilization
- Saturations on control & State
- Fast Constrained NMPC
- Sampling period 0.1 *sec*

gipsa-lab

State constrained Swing-up and stabilization of a twin-pendulum system



[M. Alamir & A. Murilo Automatica 2007]

- Constrained State Feedback
- Swing-up & Stabilization
- Saturations on control & State
- Fast Constrained NMPC
- Sampling period 0.1 *sec*

gipsa-lab

Preliminary experimental result

- $\dot{T}_j = u$
- $T_j \in [T_j^{min}, T_j^{max}]$
- $u \in [-\bar{u}, +\bar{u}]$

$$p := \begin{pmatrix} \frac{w_1}{\bar{u}} \\ \frac{w_2}{\bar{u}} \\ \frac{t_1}{t_f} \end{pmatrix}^T \in [-1, +1]^2 \times [0, 1]$$