Topics on Fast NMPC ...

The control updating period issue ...

Mazen Alamir

GIPSA-Lab, CNRS, University of Grenoble, France.

Outline

- Fast NMPC
- The Control Updating Period Paradigm
- The Proposed Updating Scheme
- Fast Gradient Related Considerations
- Numerical Investigation
- Conclusion & future works

Classical NMPC Quick reminder

Dynamical Systems

$$\dot{x} = f(x, u) \qquad \rightarrow \text{sampling time } \tau$$

Control parameterization (over some prediction horizon)

$$\mathcal{U}_{pwc}(p) := \left(u^{(1)}(p), \ldots, u^{(N)}(p)\right) \in \mathbb{U} \subset \mathbb{R}^{N \cdot n_u}$$

Open-Loop optimization problem

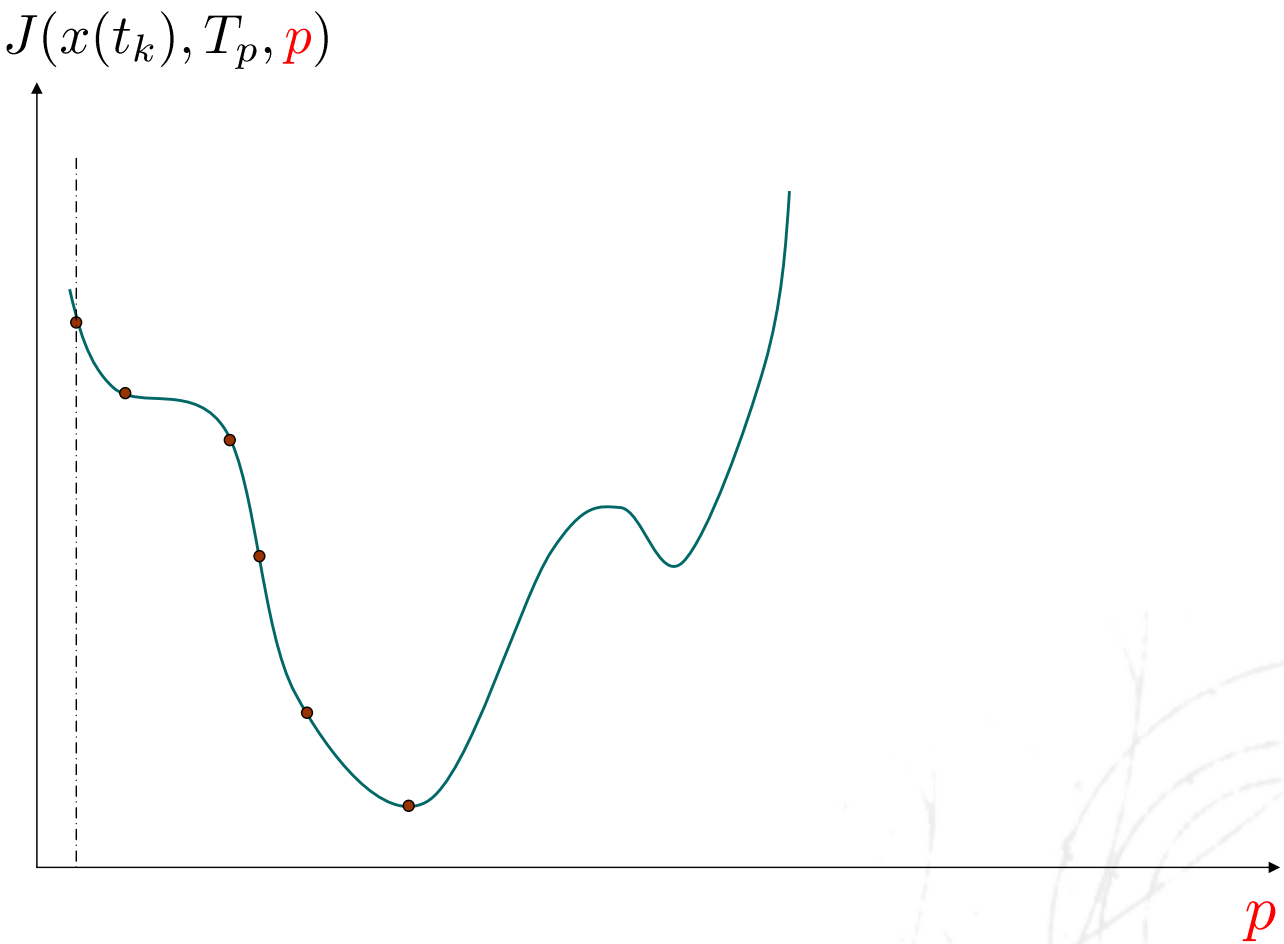
$$\mathcal{P}(x) \quad : \quad \hat{p}(x) := \arg\min_{p} \; J(p, x) \quad \text{under } C(p, x) \leq 0$$

Ideal NMPC feedback

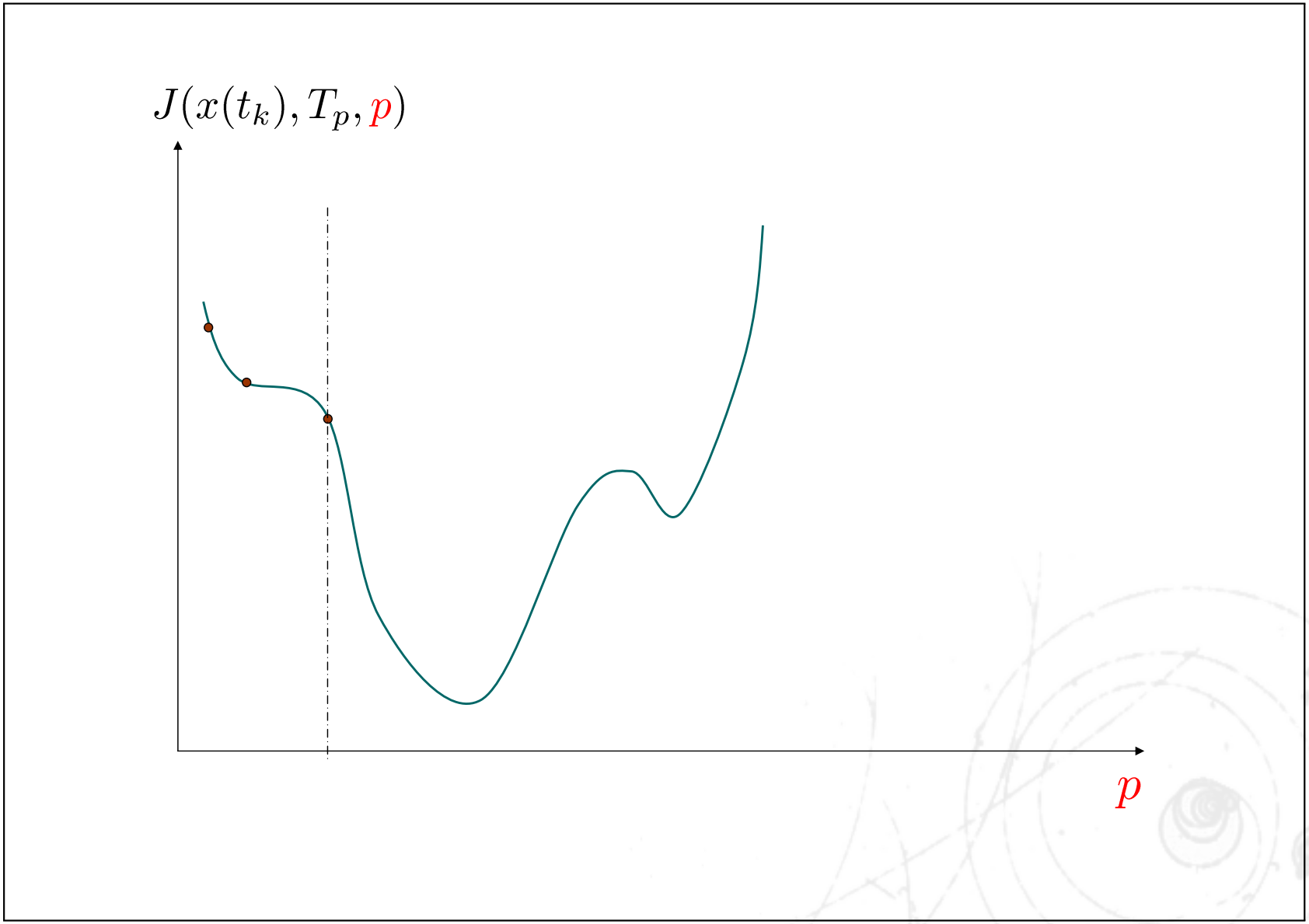
$$u(k) := u^{(1)}\left(\hat{p}(x(k))\right)$$

This suppose that the computation of $\hat{p}(x)$ can be done in a fraction of the sampling period τ. We call Fast systems those systems requiring sampling periods for which this is not true.

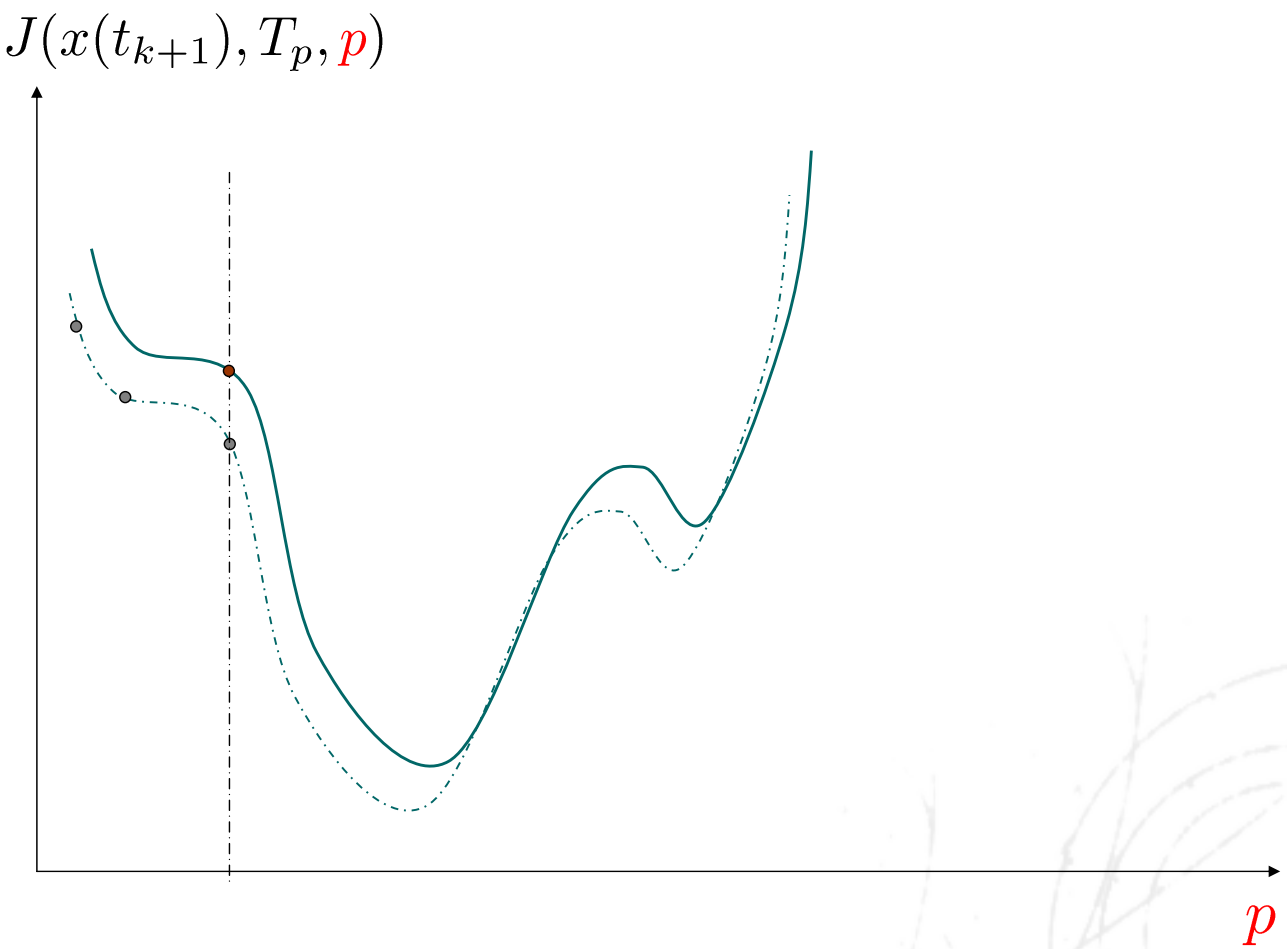
Real-Time NMPC The principle



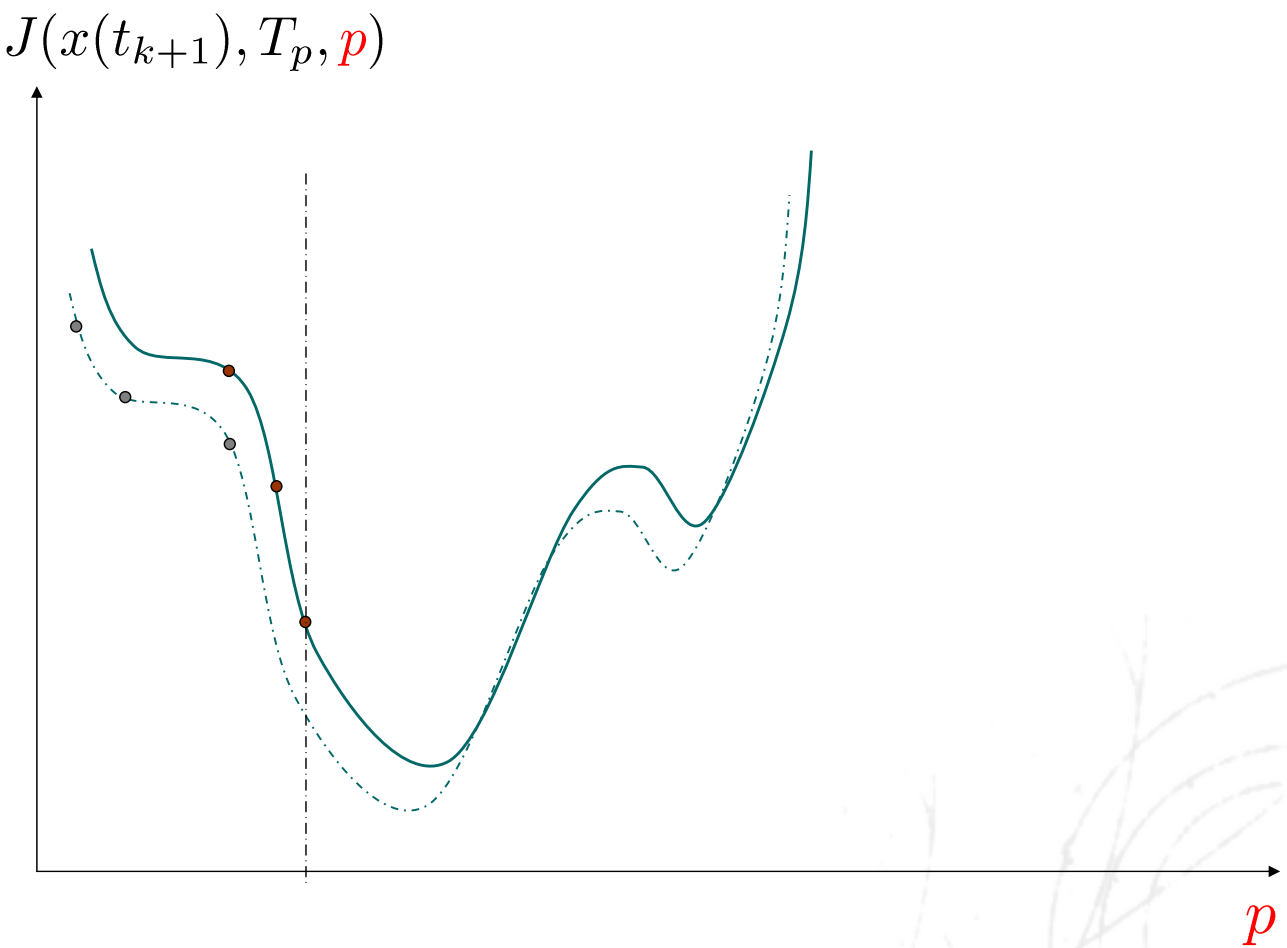
Real-Time NMPC The principle



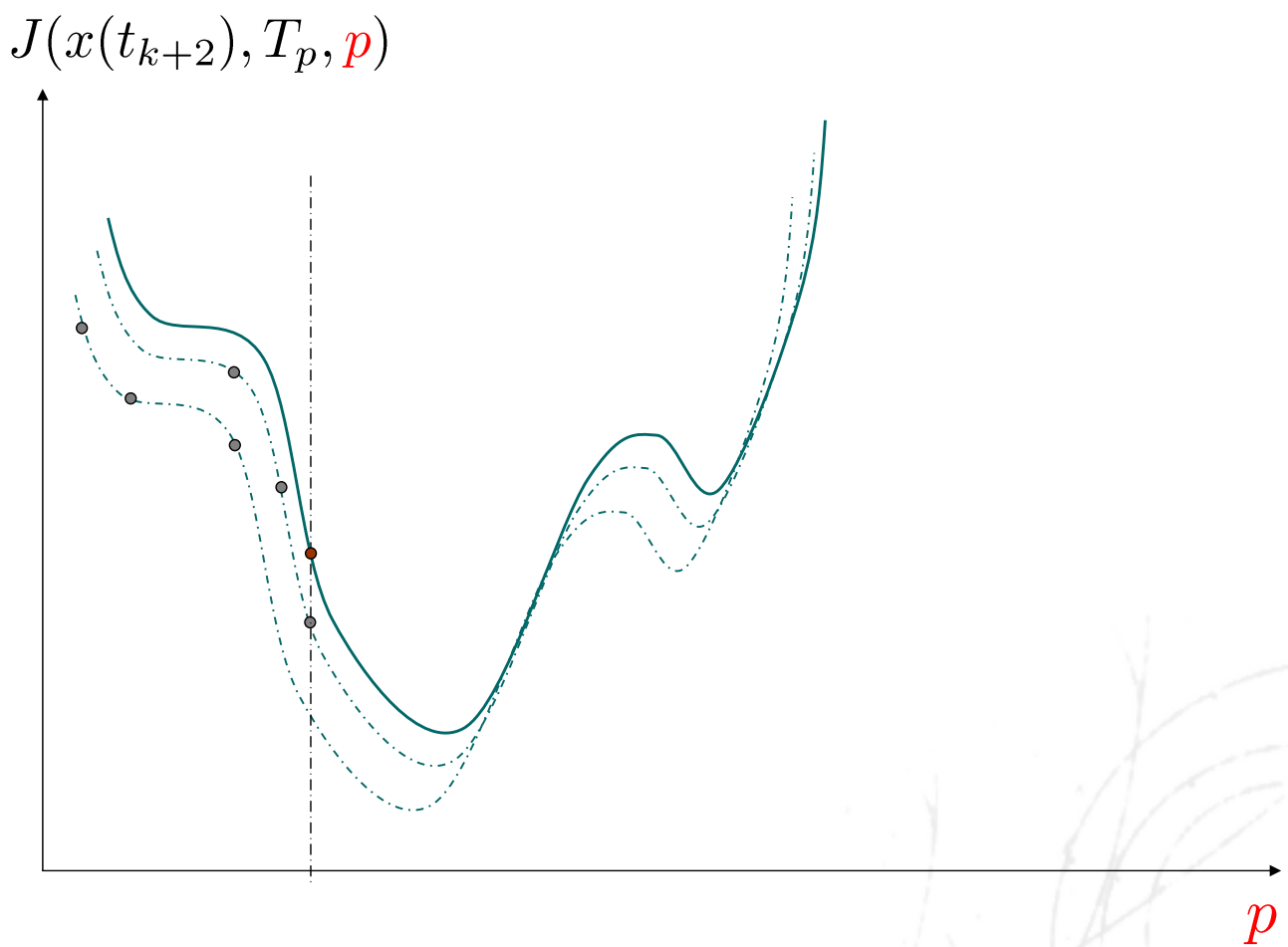
Real-Time NMPC The principle



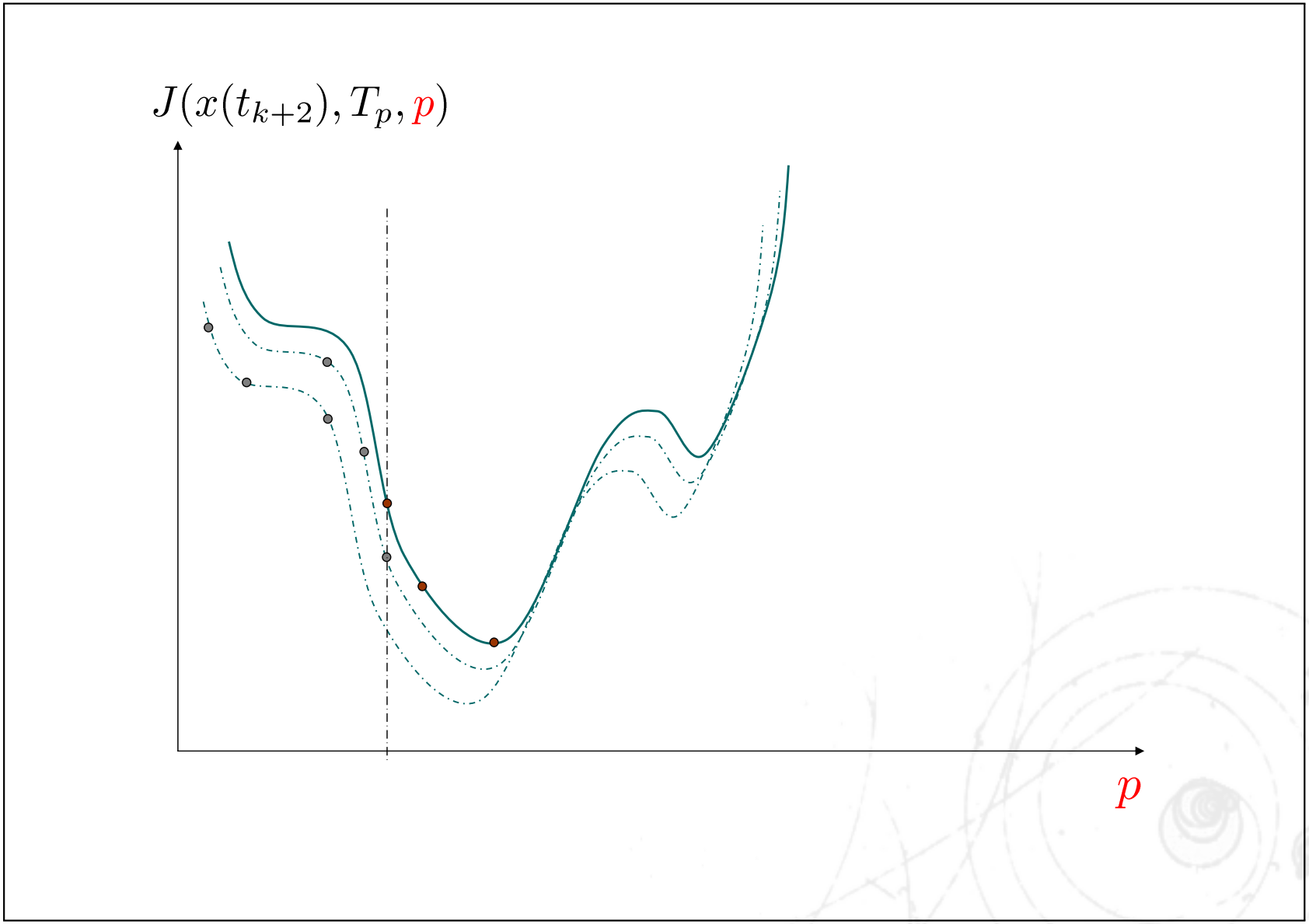
Real-Time NMPC The principle



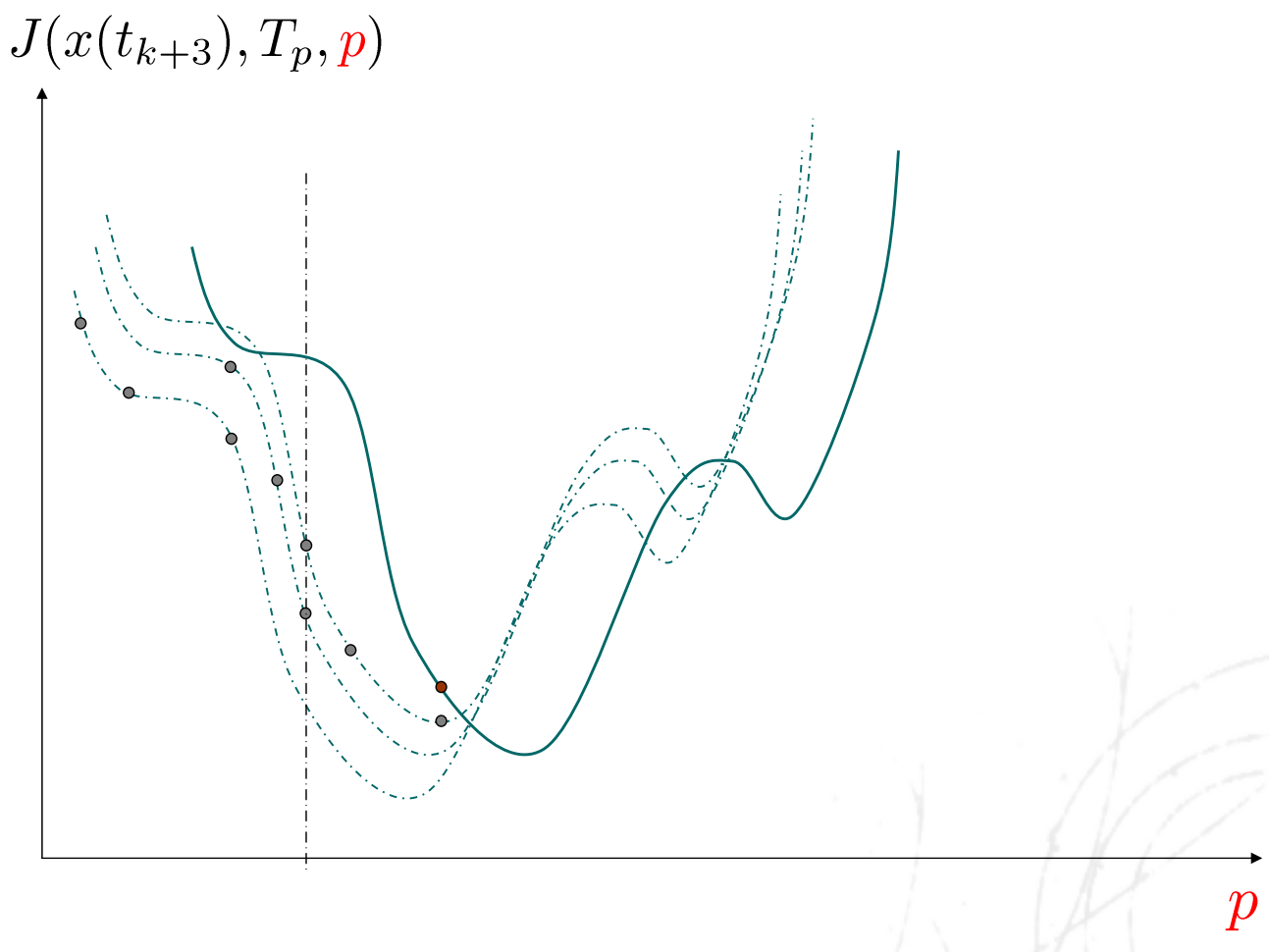
Real-Time NMPC The principle



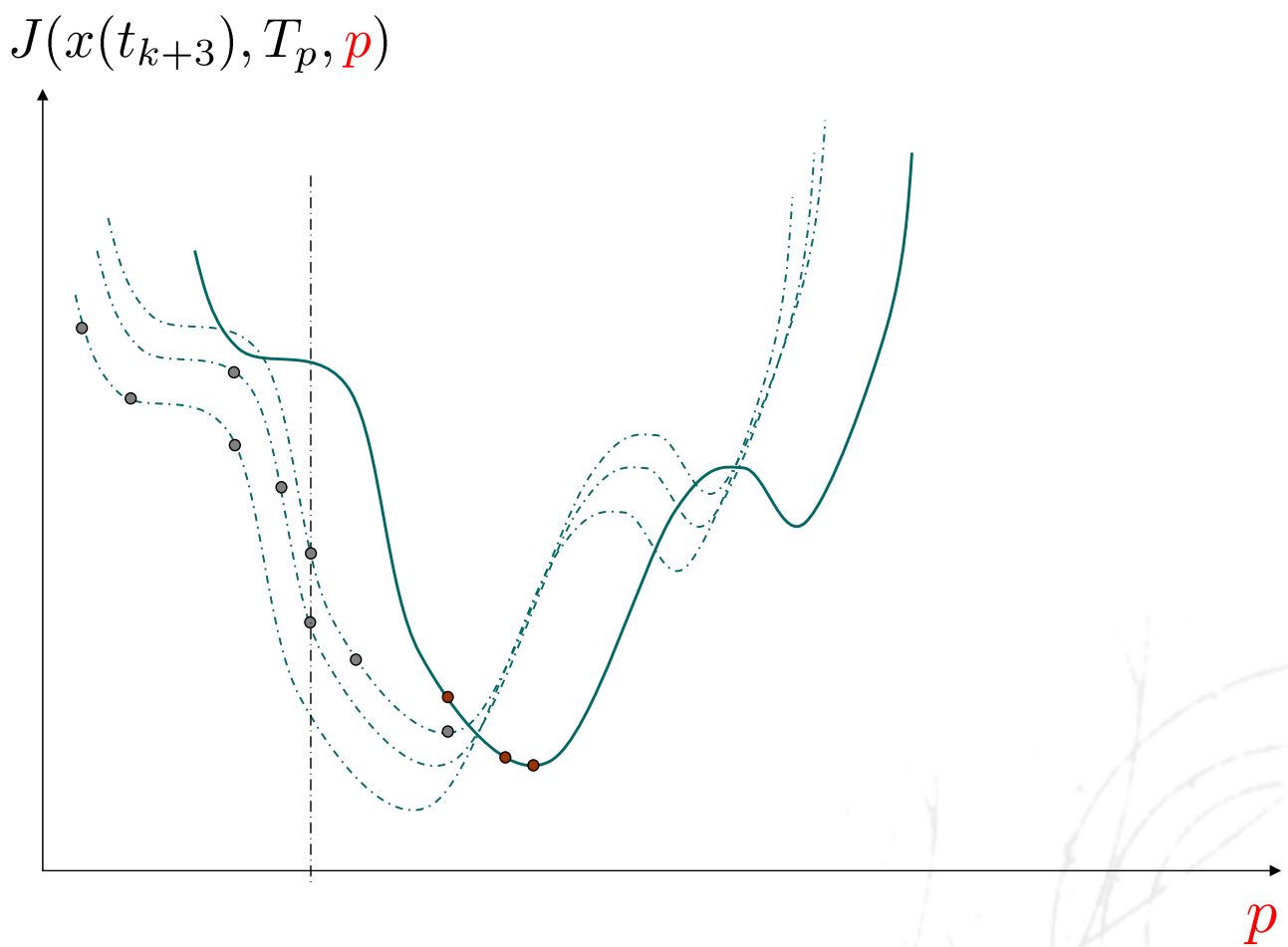
Real-Time NMPC The principle



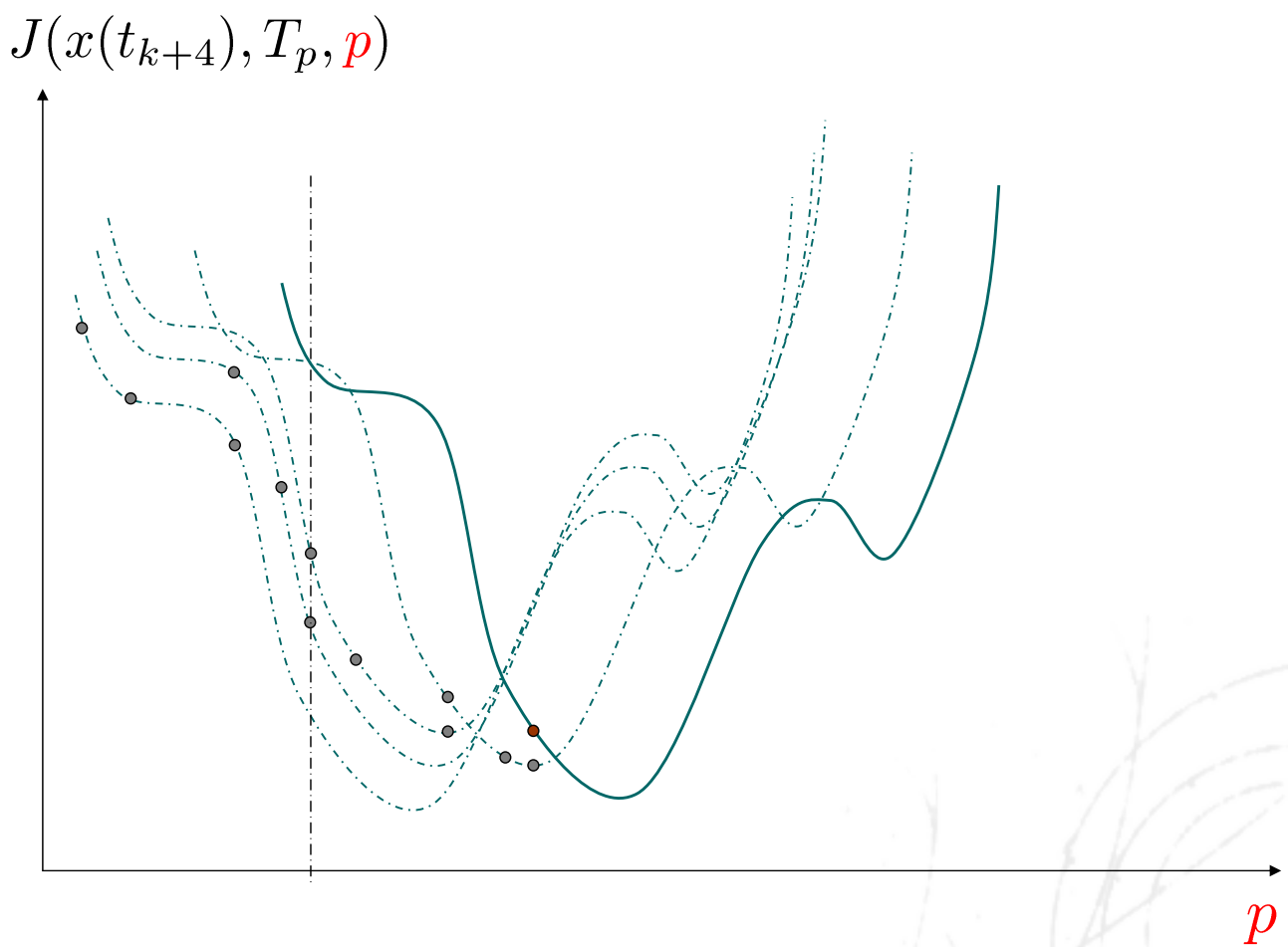
Real-Time NMPC The principle



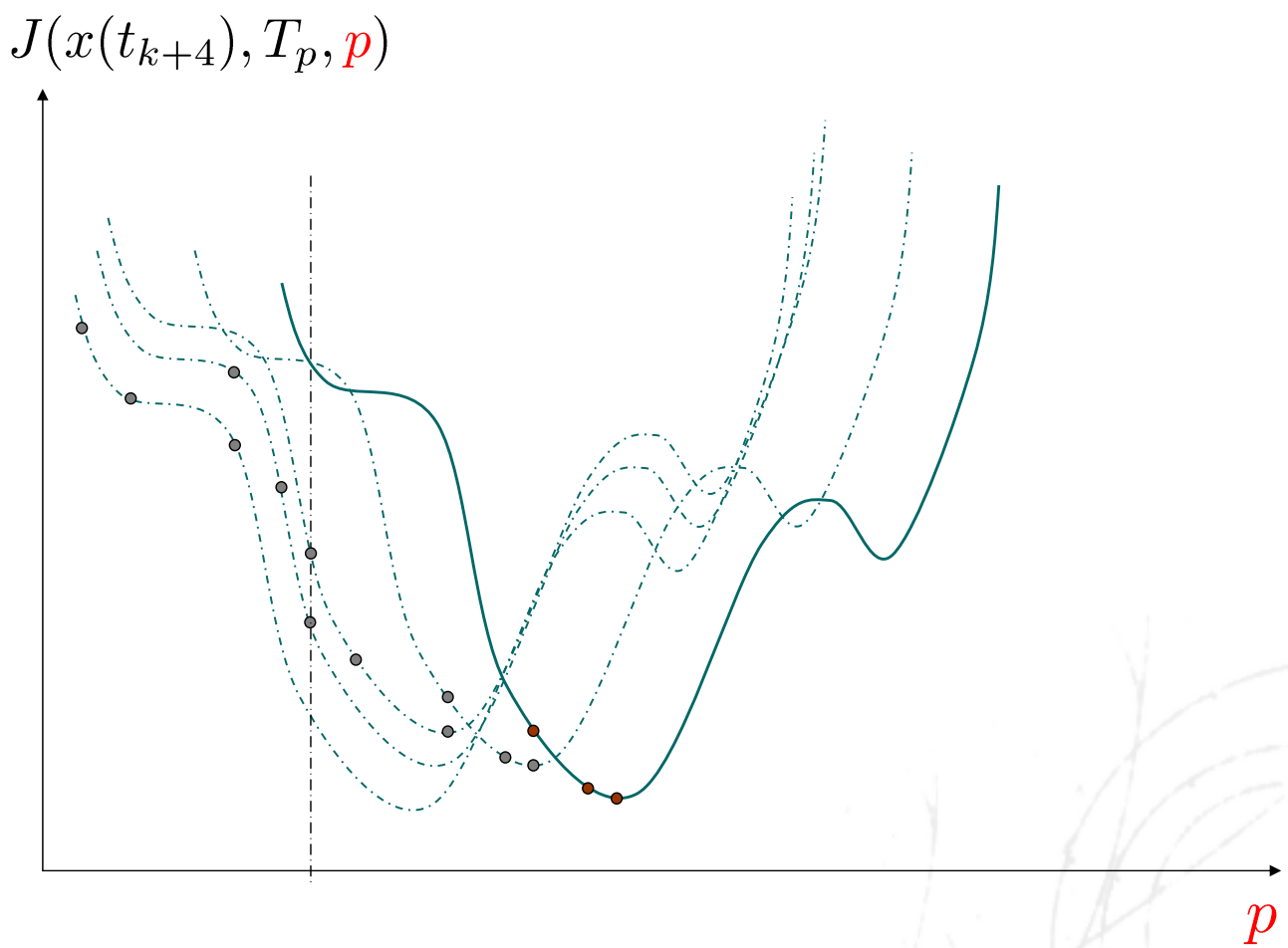
Real-Time NMPC The principle



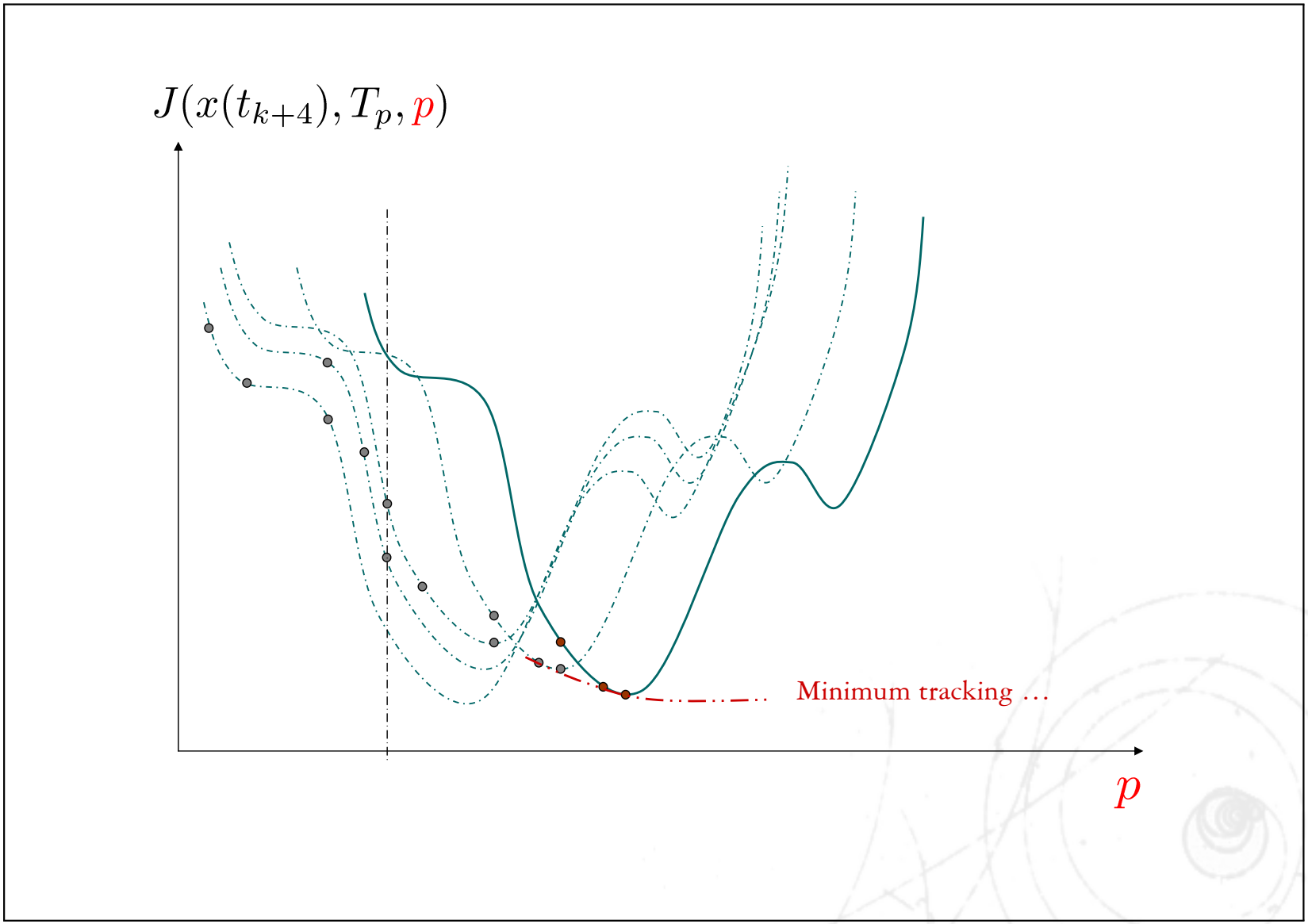
Real-Time NMPC The principle



Real-Time NMPC The principle



Real-Time NMPC The principle



Real-Time NMPC A bit of history . . .

PERGAMON

Control Engineering Practice 9 (2001) 107–116

CONTROL ENGINEERING PRACTICE

www.elsevier.com/locate/conengprac

Nonlinear receding horizon sub-optimal guidance law for the minimum interception time problem

Mazen Alamir*

Laboratoire d'Automatique de Grenoble, UMR 5528, CNRS-INPG-UJF, BP 46 Domaine Universitaire, Saint Martin d'Hères, France

Received 8 September 1999; accepted 18 November 1999

Page 5 : "In the *distributed* Newton Method, the Newton iterations are distributed over the successive sampling periods. **One step at each sampling period."**

Real-Time NMPC A bit of history ...

M. Alamir, Control Engineering Practice, **2001**.
T. Ohtsuka, Automatica, **2004**.
M. Diehl et al. SIAM J. Control and Optimization, **2005**.
IFAC Workshop on NMPC for Fast Systems, Grenoble, **2006**.

M. Alamir, Springer, **2006**.

Embedding numerical libraries is an issue in applications for memory, certification and reliability reasons.

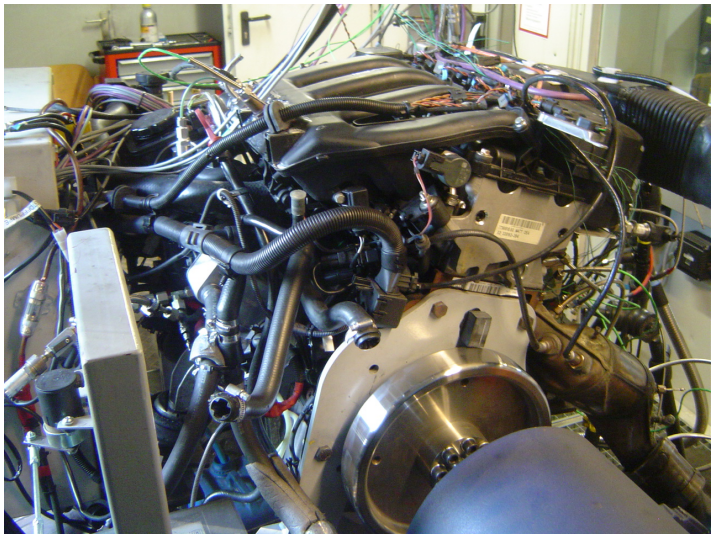
D. DeHaan, M. Guay, Springer LNCIS, **2007**.
Y. Wang, S. Boyd, IEEE CST, **2010**.
B. Houska et al. OCAM, **2010**.
...

Fast Gradient (Y. Nesterov, 1983) related NMPC implementation :

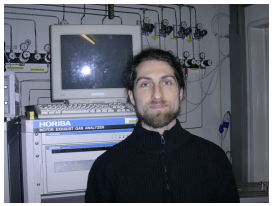
P. Zometa et al. ACC, **2012**
A. Bemporad, P. Patrinos, IFAC NMPC**2012**
C. Jones et al., IFAC NMPC**2012**
...

Examples Diesel engine

- BMW M47D Diesel Engine (Linz)
- Sampling time $\tau = 50$ ms
- Prediction Horizon $30 \cdot \tau$
- Observation Horizon $N_O = 10\tau$
- Solver SQP/Trust region
- Number of iteration $q = 30$
- NL model, 11 states, 2 controls




André Murilo

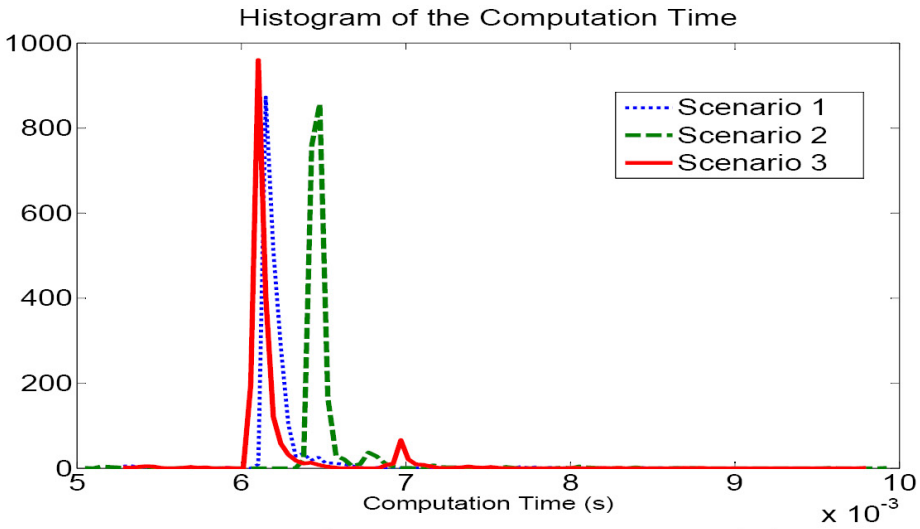

R. Fŭrhapter


Peter Ortner

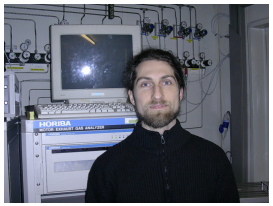
Alamir et al. Automotive MPC: Models, Methods & Applications, Linz, Austria, 2009

Examples Diesel engine

- BMW M47D Diesel Engine (Linz)
- Sampling time $\tau = 50$ ms
- Prediction Horizon $30 \cdot \tau$
- Observation Horizon $N_O = 10\tau$
- Solver SQP/Trust region
- Number of iteration $q = 30$
- NL model, 11 states, 2 controls




André Murilo

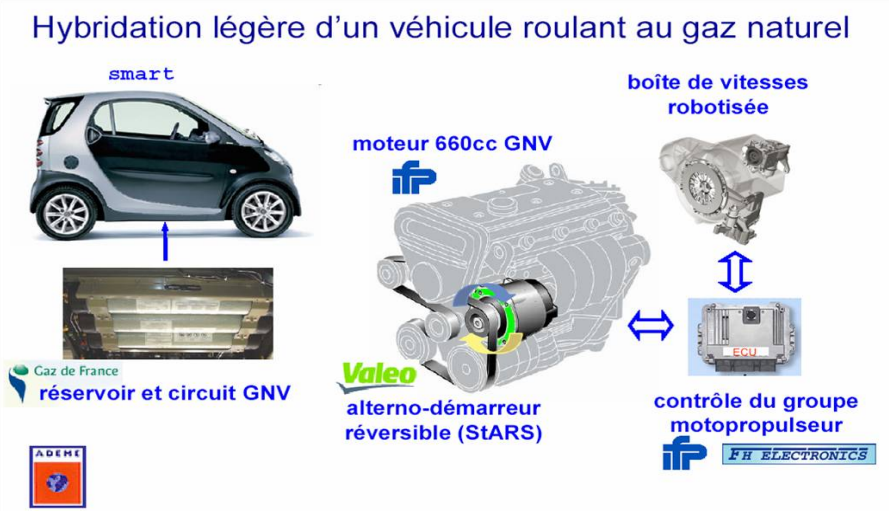

R. Fűrhapter


Peter Ortner

Alamir et al. Automotive MPC: Models, Methods & Applications, Linz, Austria, 2009

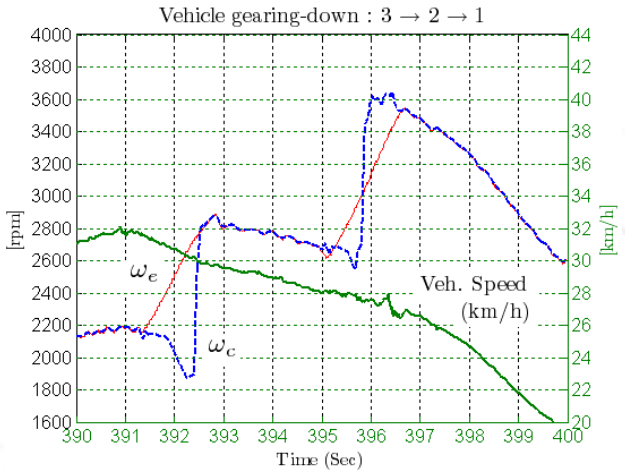
Examples AMT (Automated Manual Transmission)

- Only 0.3 Mb for control software
- Base sampling time $1\ ms$
- Preparation $+$ 1 iteration $\approx 50\mu s$
- Number of iteration $q = 21$



Rachid Amari

Paolo Tona

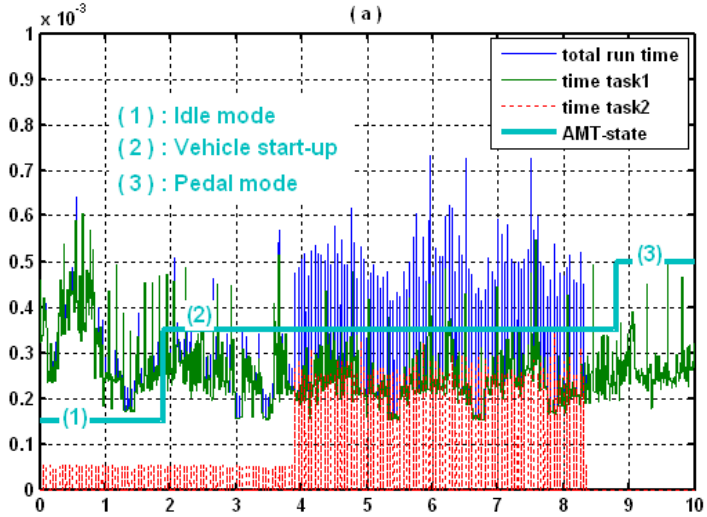


Amari et al. IFAC WC, Seoul, 2008

Alamir et al. Automotive MPC: Models, Methods & Applications, Linz, Austria, 2009

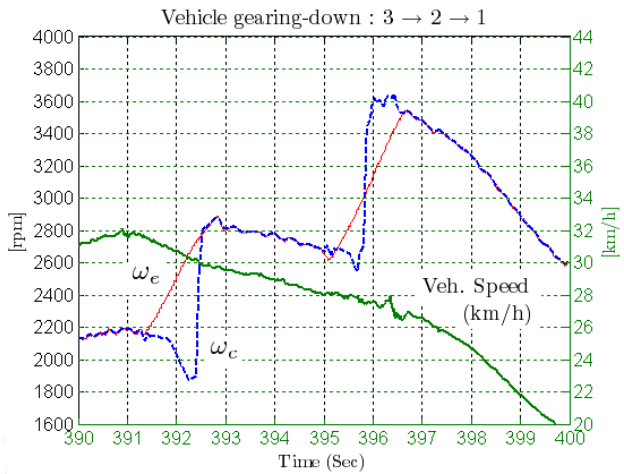
Examples AMT (Automated Manual Transmission)

- Only 0.3 Mb for control software
- Base sampling time $1\ ms$
- Preparation $+$ 1 iteration $\approx 50\mu s$
- Number of iteration $q = 21$



Rachid Amari

Paolo Tona



Amari et al. IFAC WC, Seoul, 2008

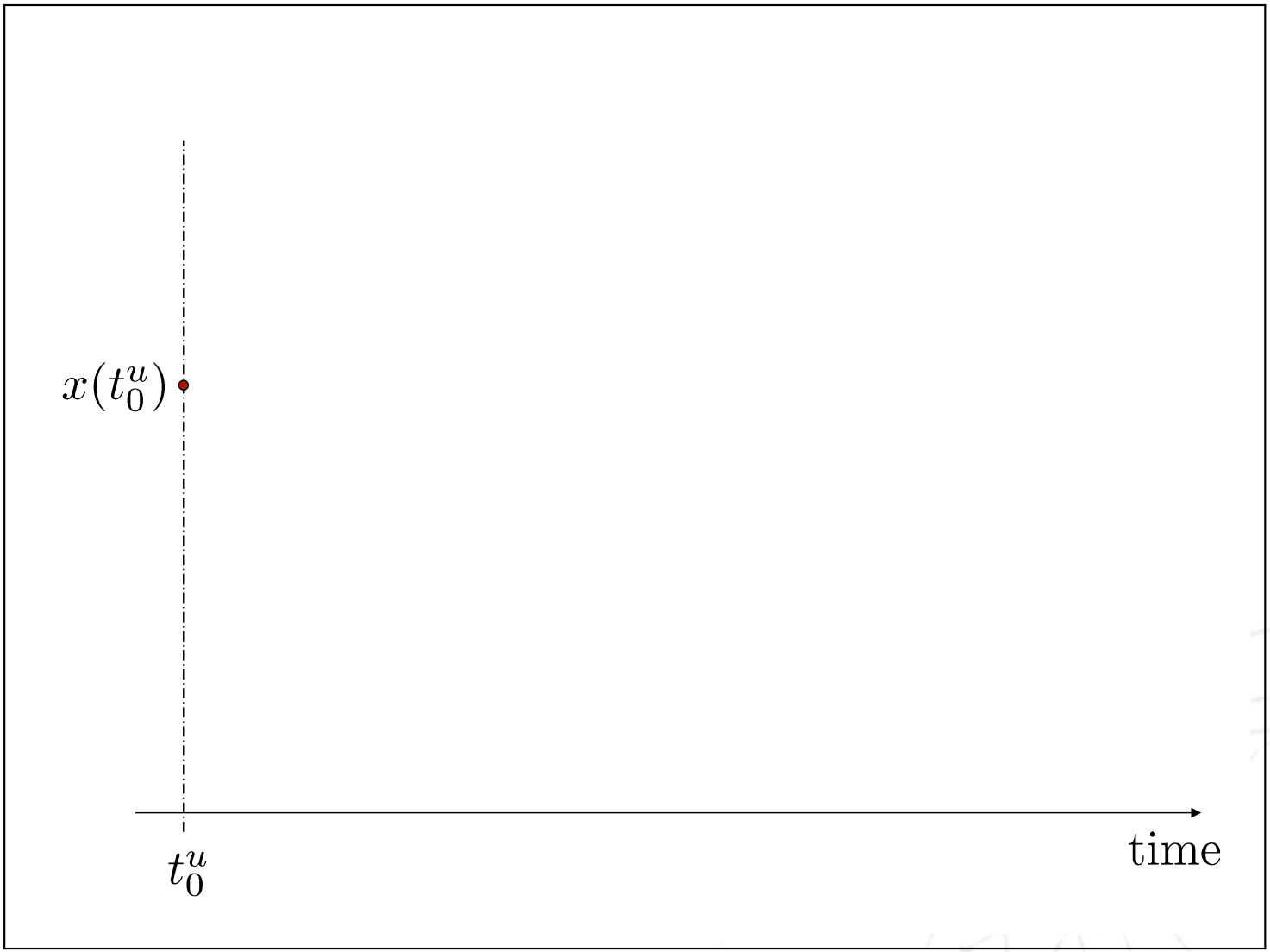
Alamir et al. Automotive MPC: Models, Methods & Applications, Linz, Austria, 2009

Examples Twin Pendulum

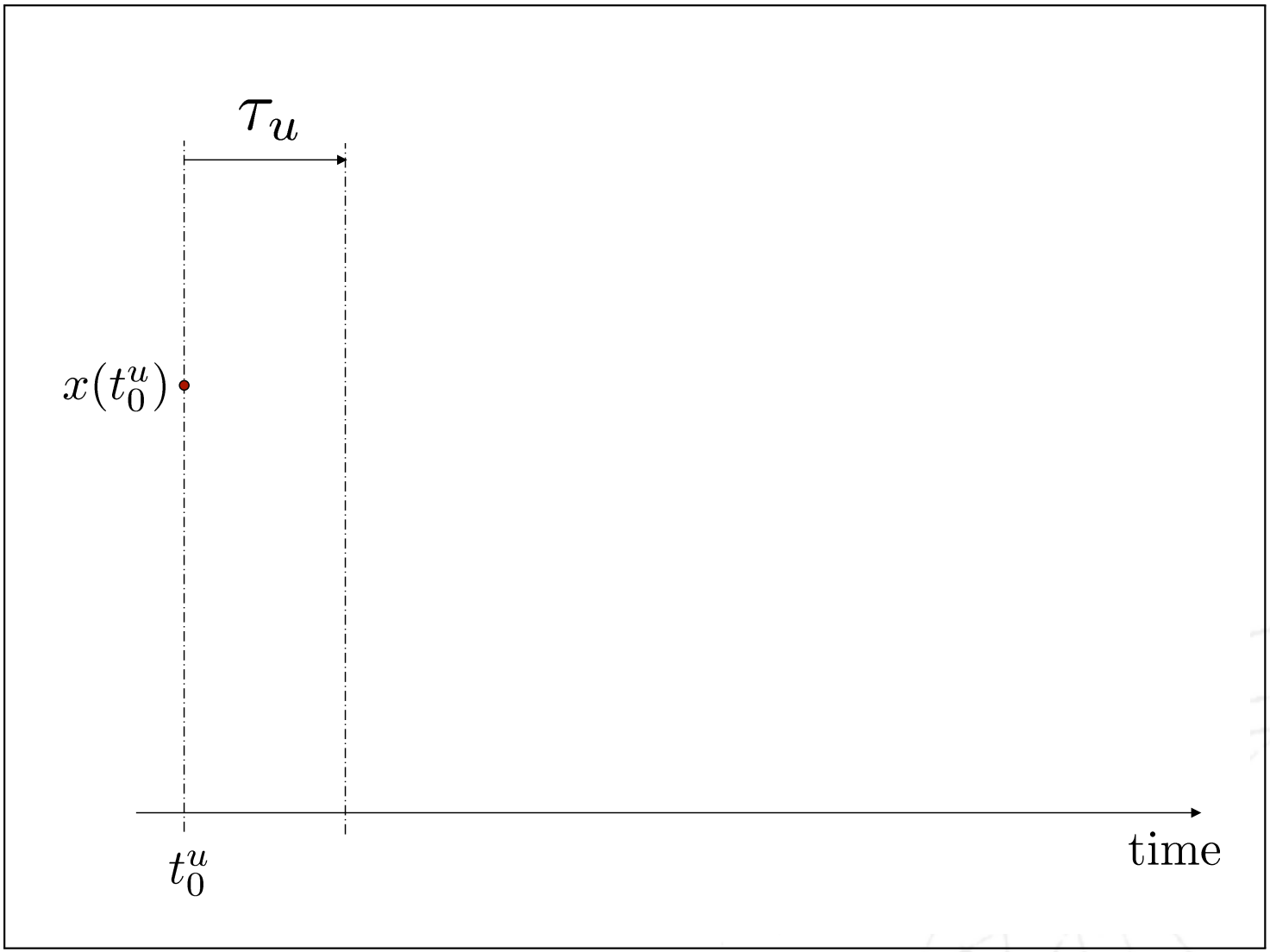
- NL system, 6 state, 1 control
- Highly unstable
- Sampling period $200\ ms$
- Number of iterations $q = 20$.

M. Alamir, A. Murilo, Automatica, **2008**.

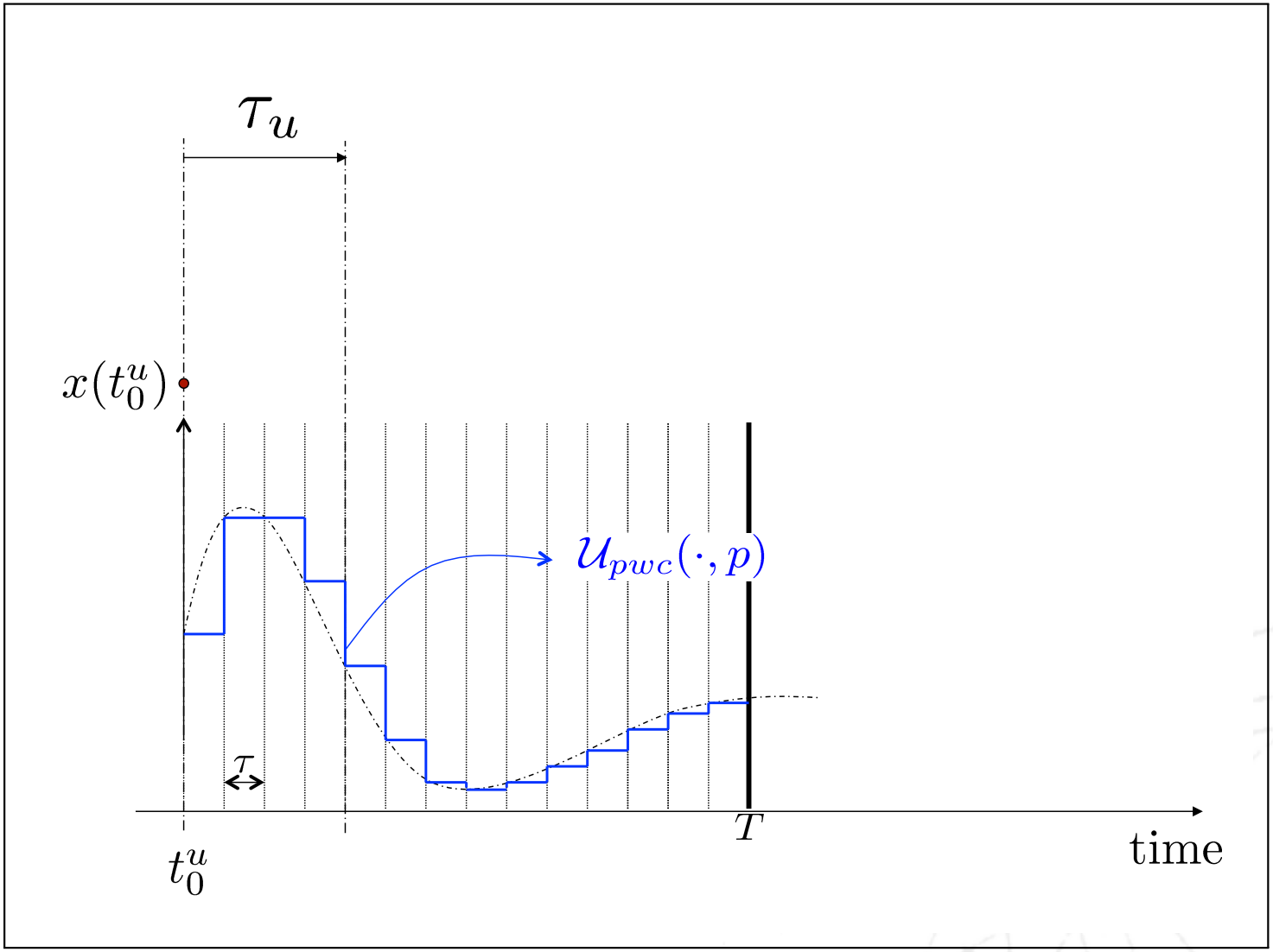
Real-Time NMPC More precisely



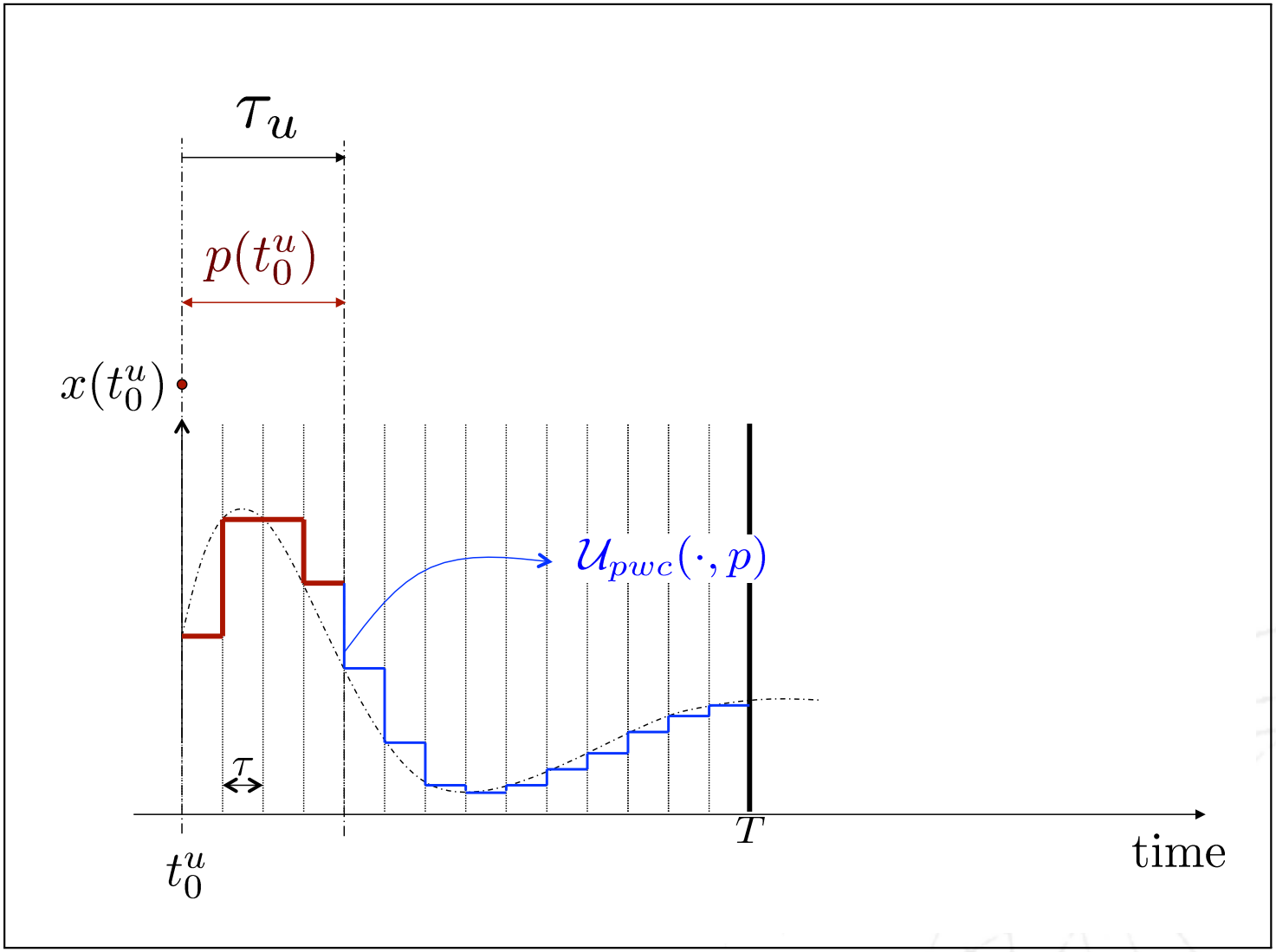
Real-Time NMPC More precisely



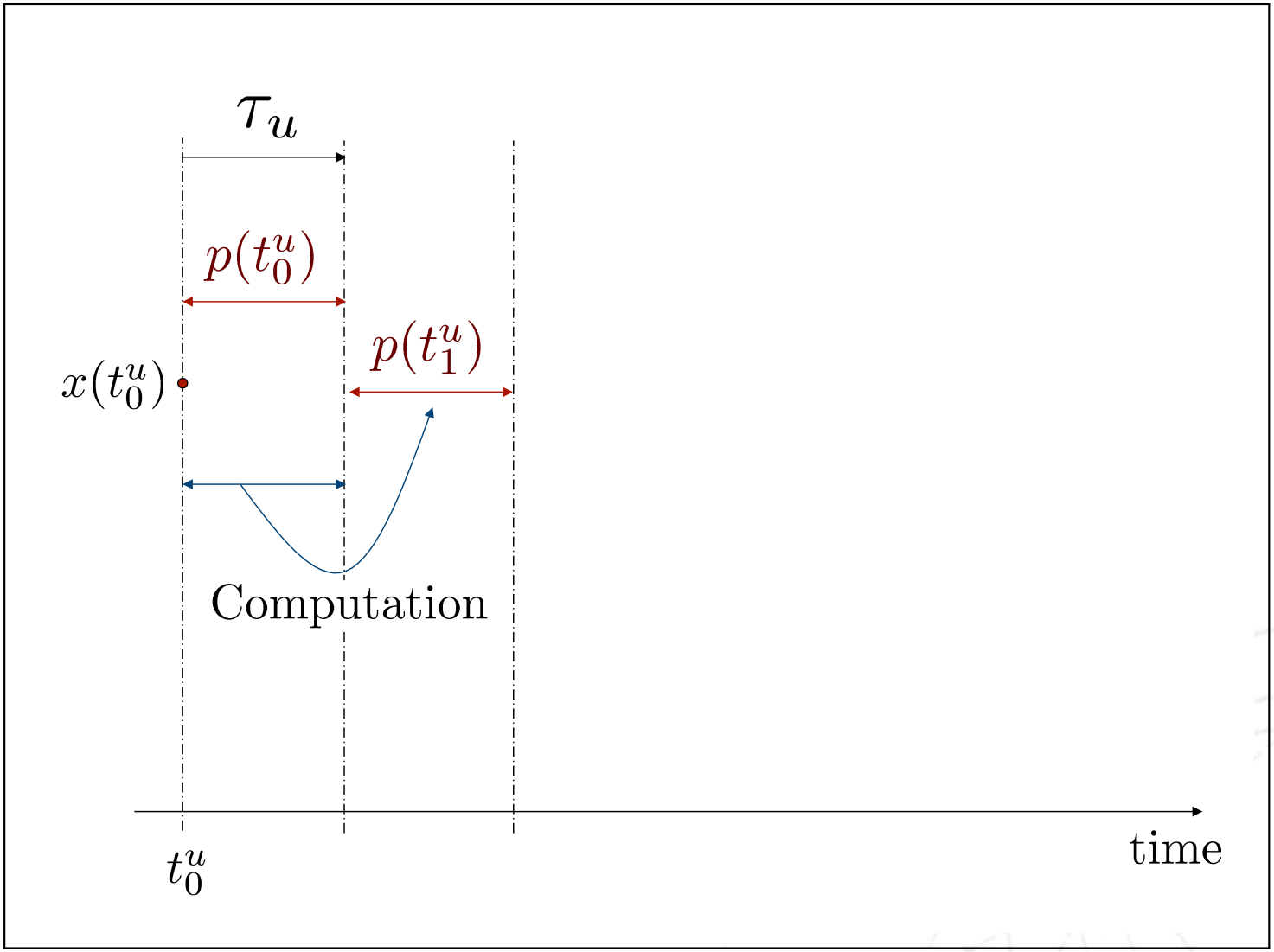
Real-Time NMPC More precisely



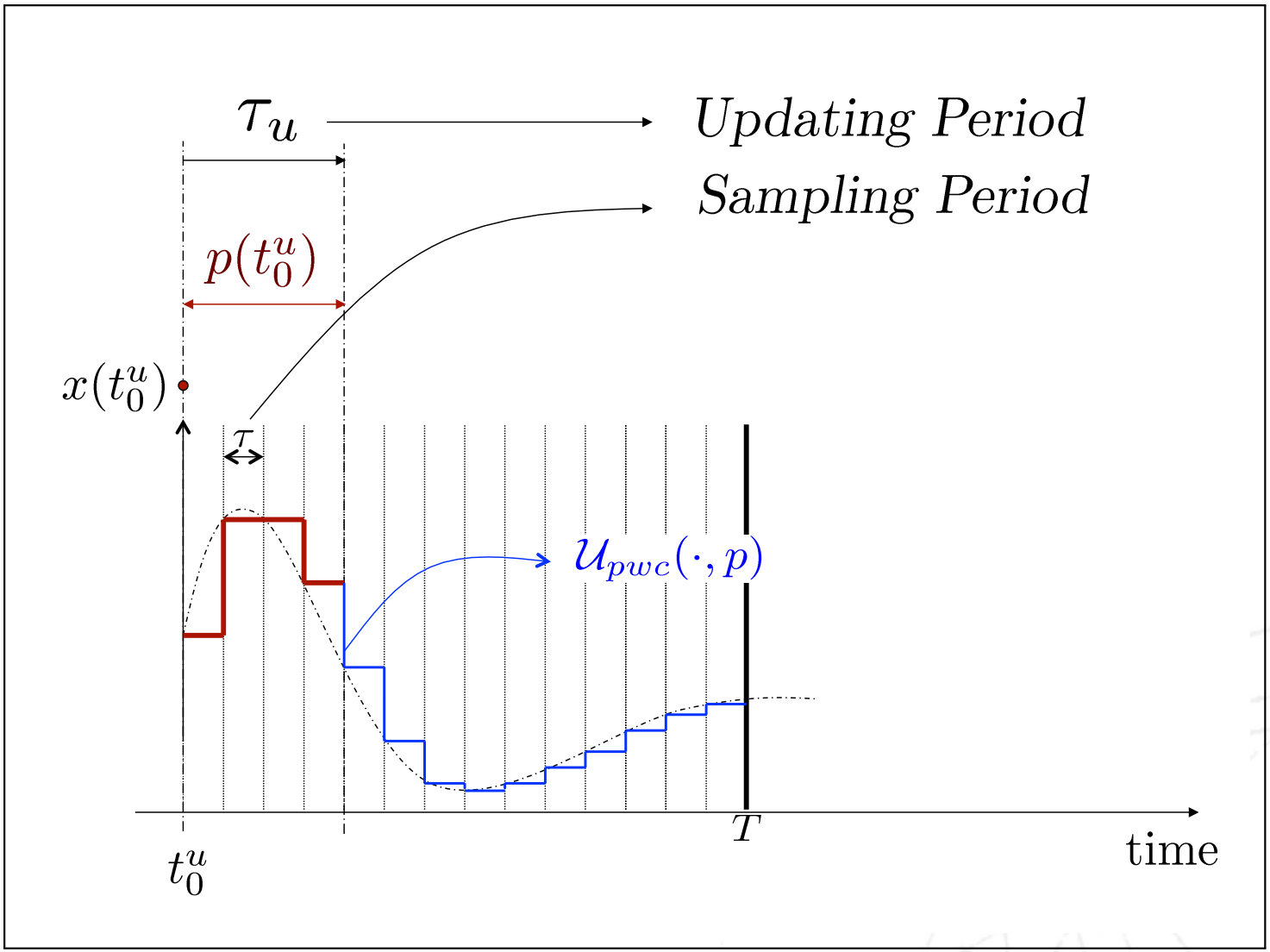
Real-Time NMPC More precisely



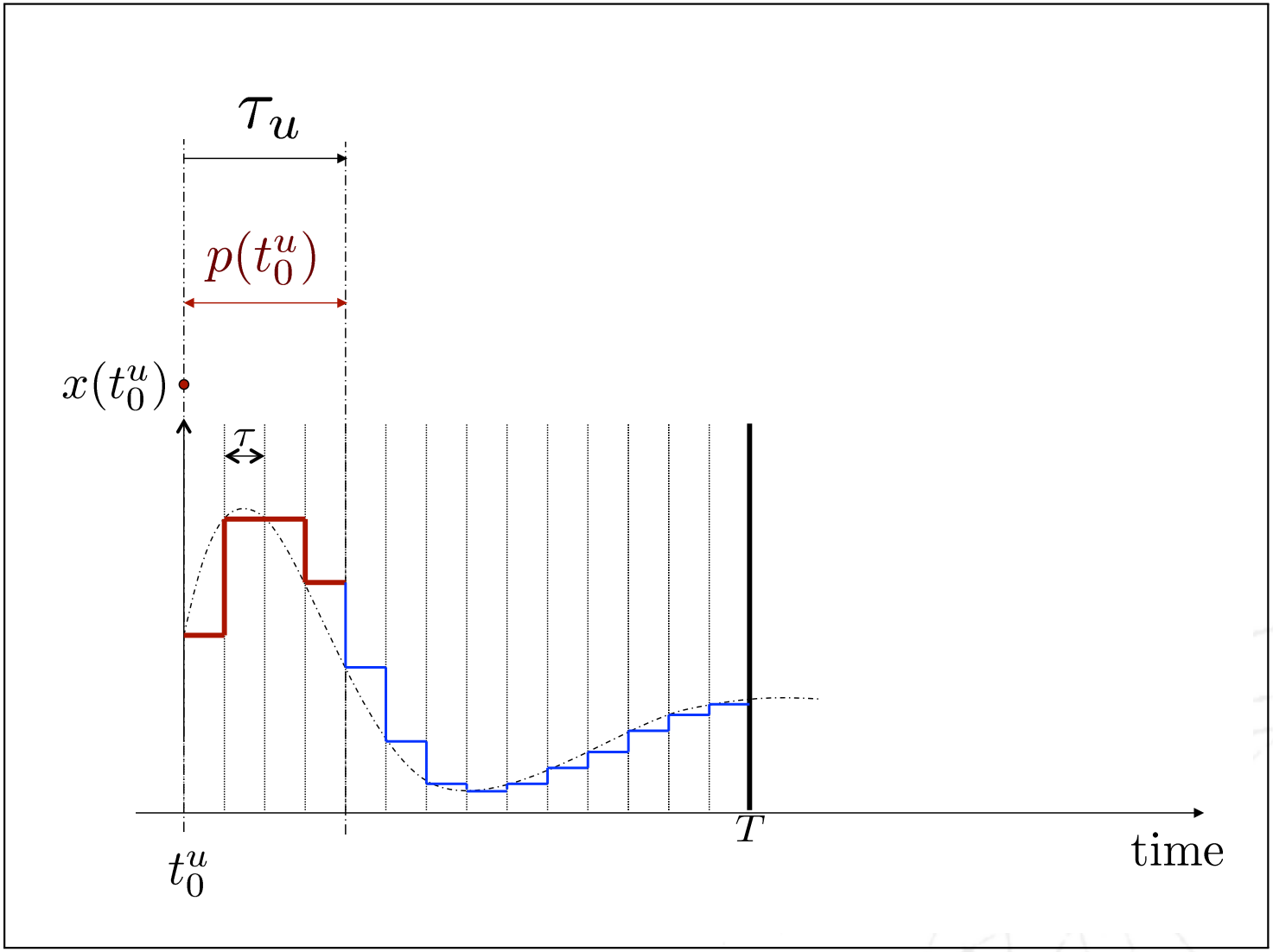
Real-Time NMPC More precisely



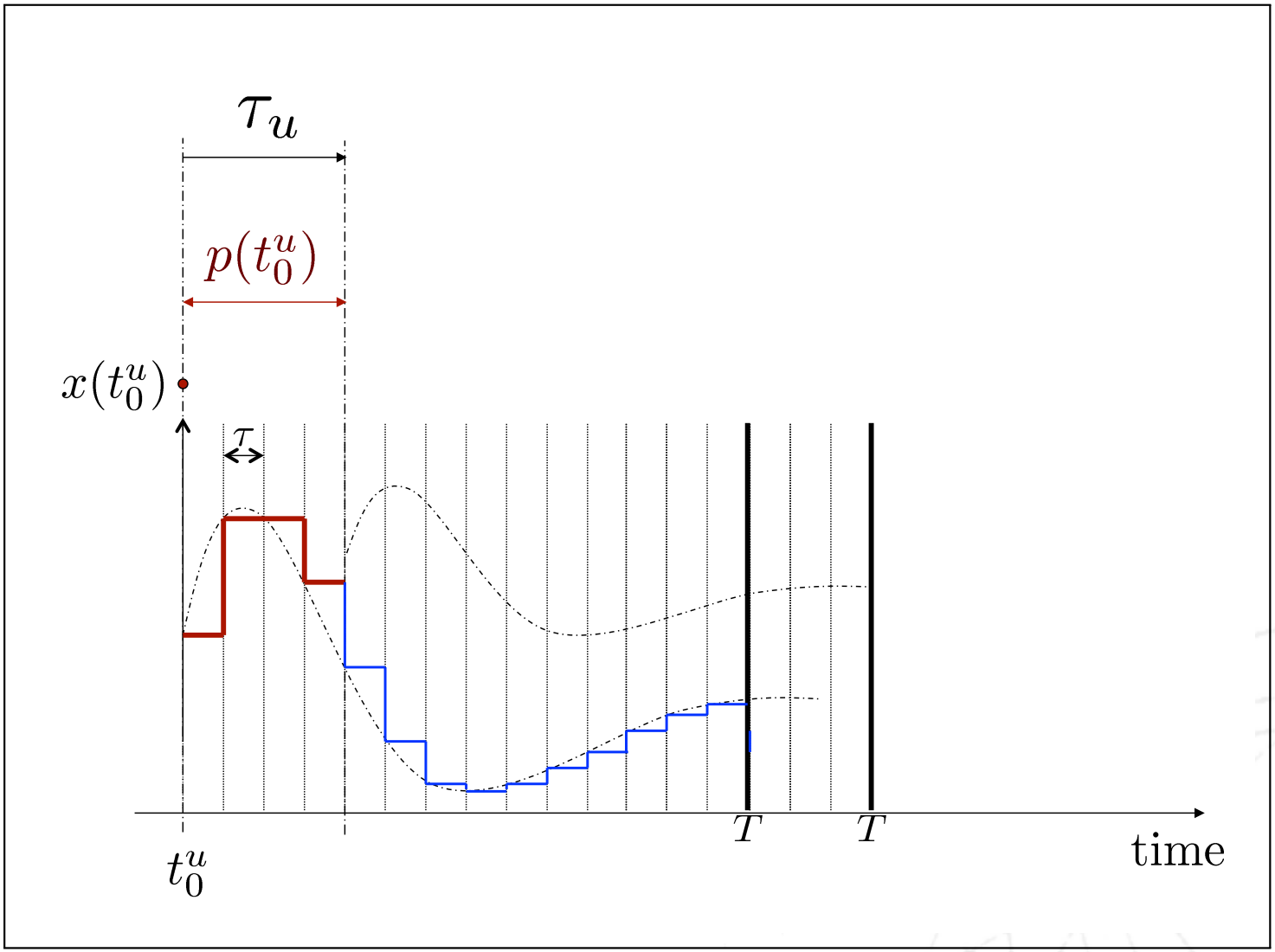
Real-Time NMPC More precisely



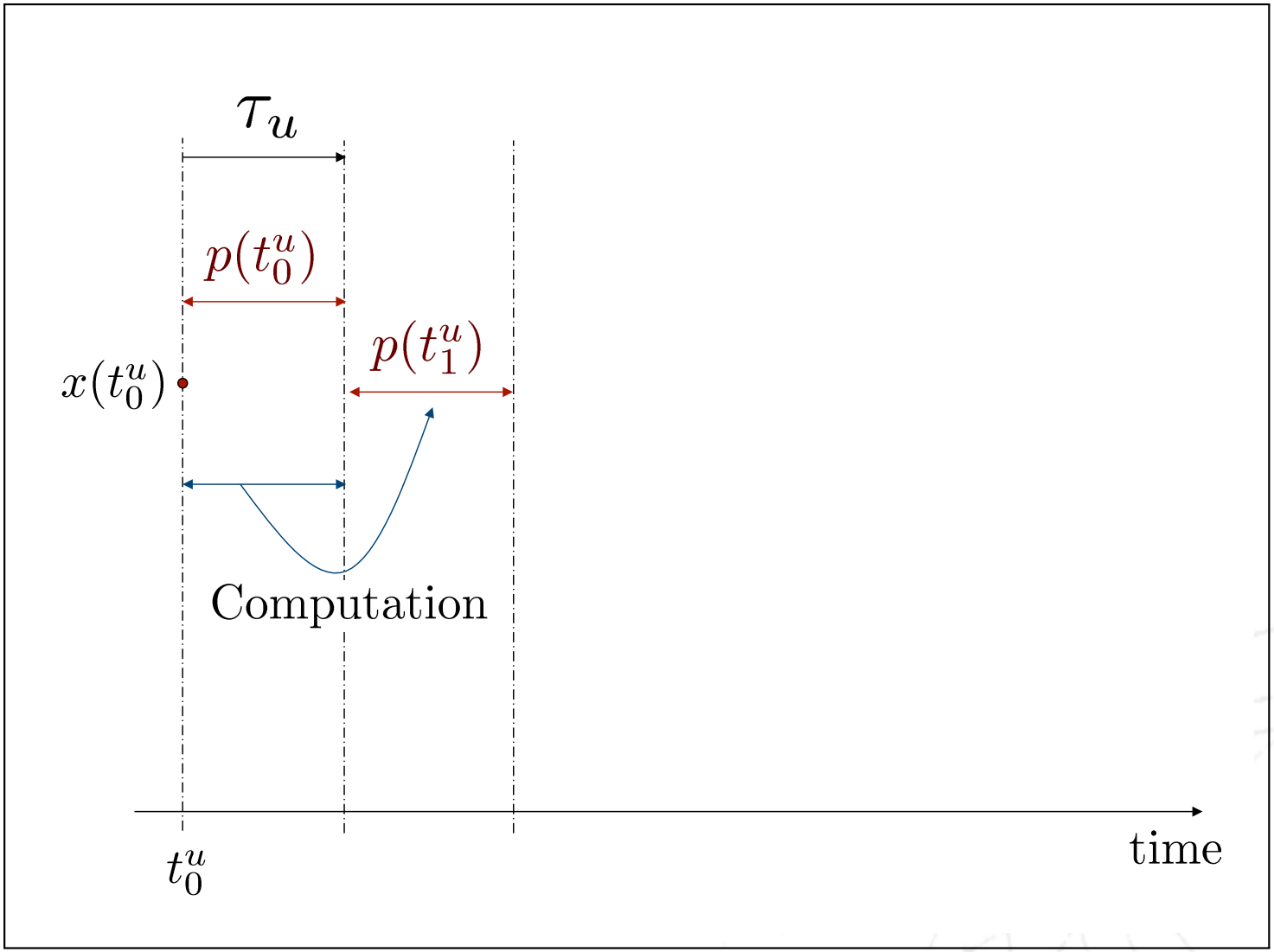
Real-Time NMPC More precisely



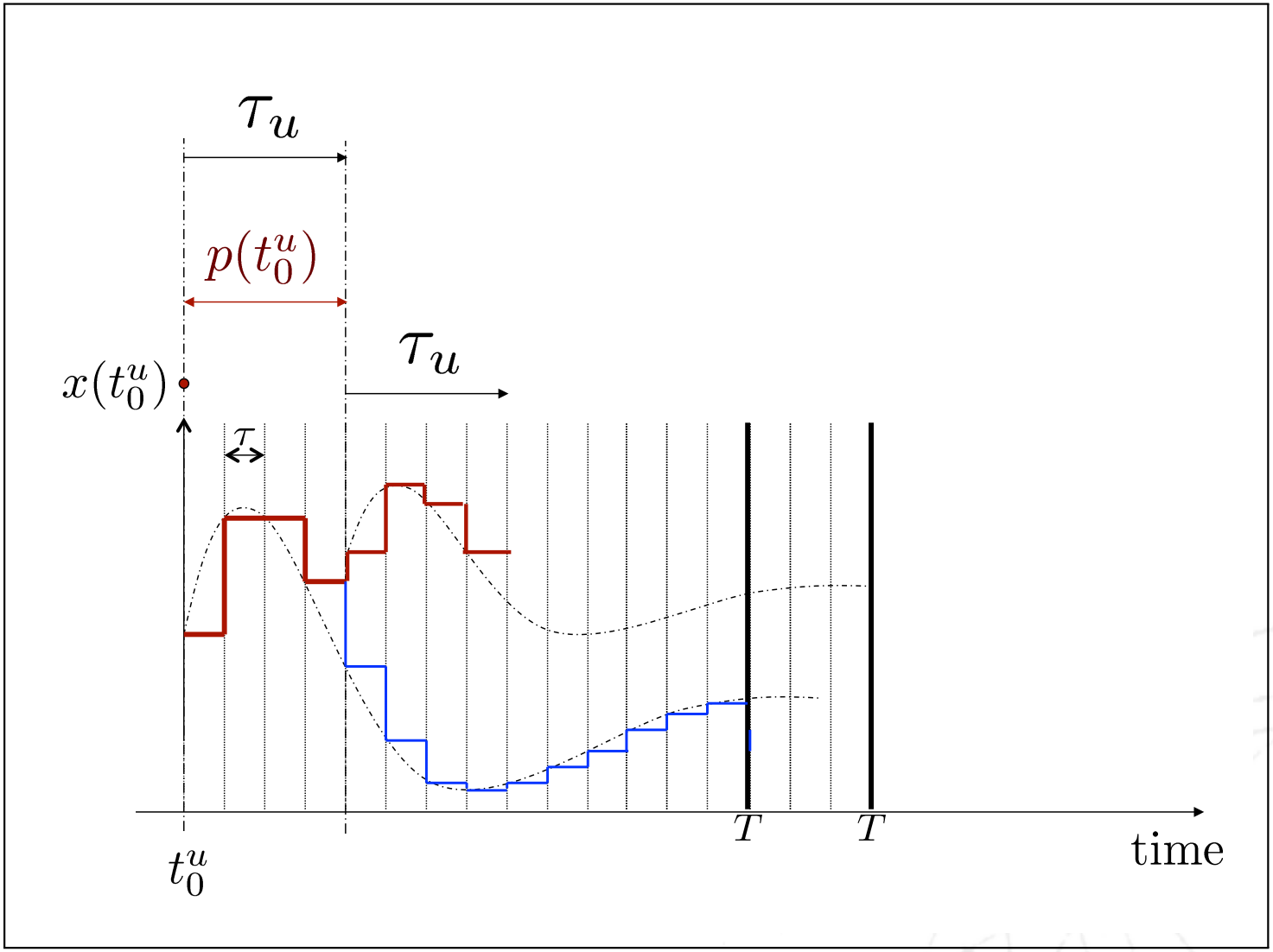
Real-Time NMPC More precisely



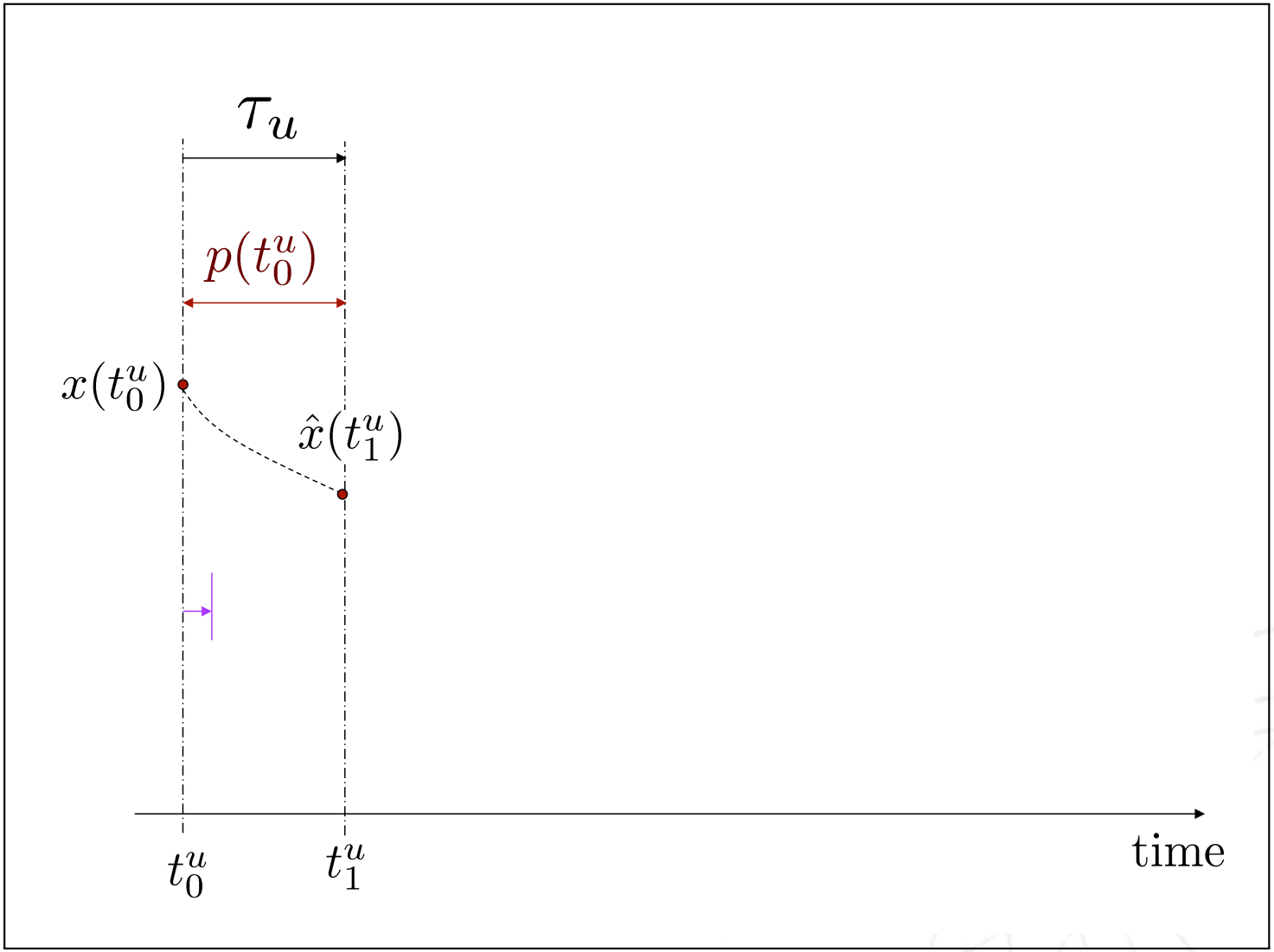
Real-Time NMPC More precisely



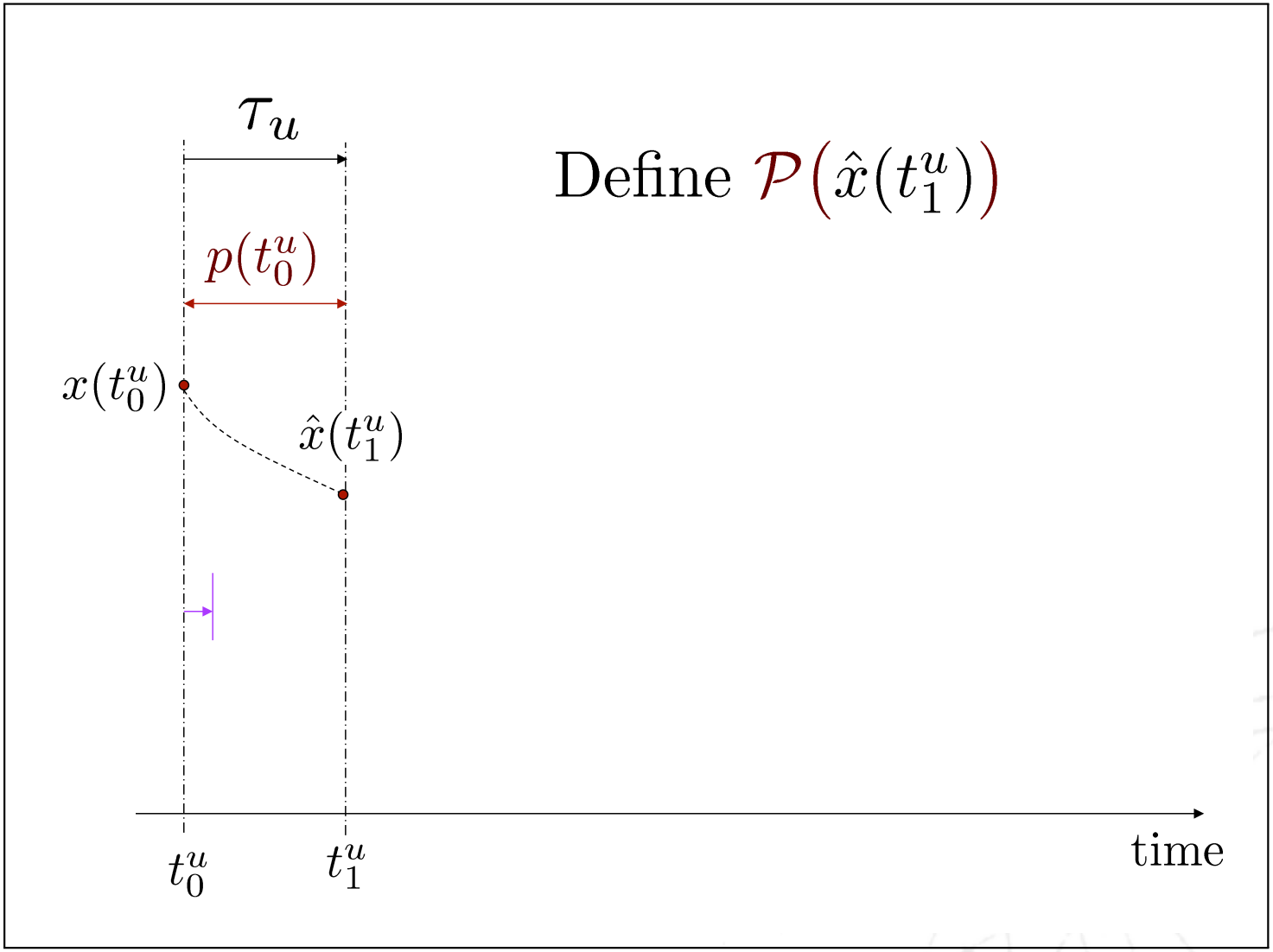
Real-Time NMPC More precisely



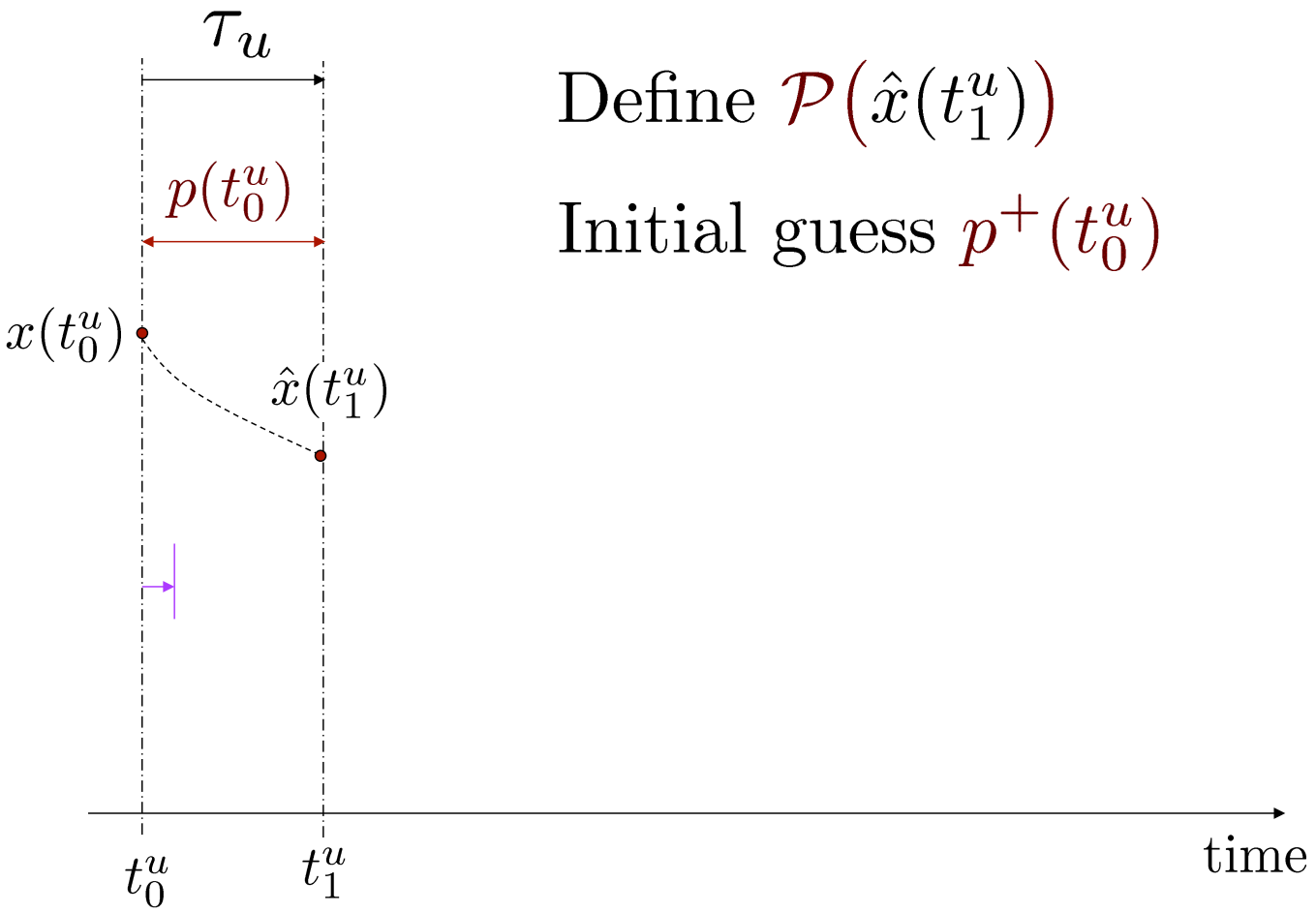
Real-Time NMPC More precisely



Real-Time NMPC More precisely



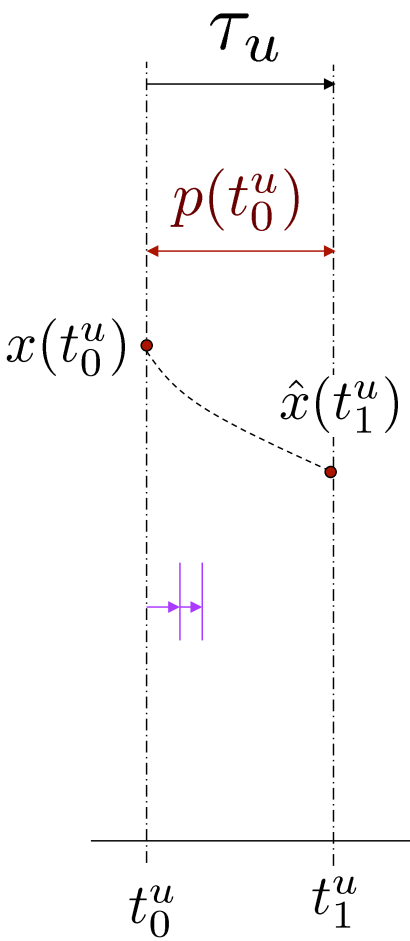
Real-Time NMPC More precisely



Define $\mathcal{P}\left(\hat{x}(t_1^u)\right)$

Initial guess $p^+(t_0^u)$

Real-Time NMPC More precisely

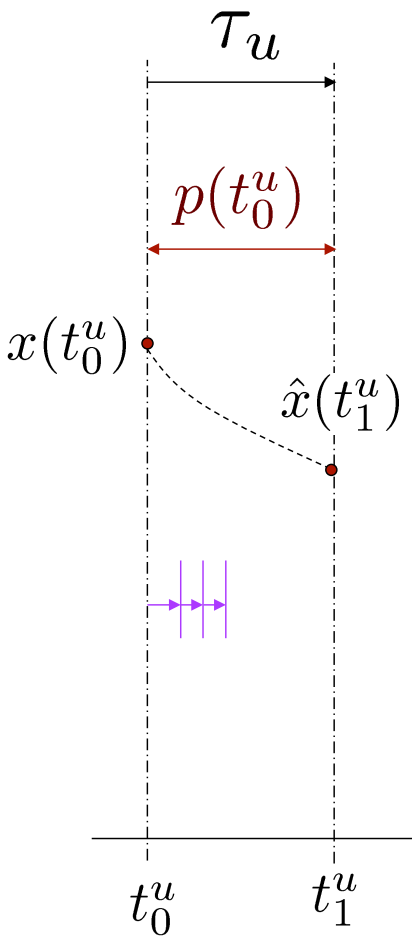


Define $\mathcal{P}\left(\hat{x}(t_1^u)\right)$

Initial guess $p^+(t_0^u)$

$$\mathcal{S}\left(p^+(t_0^u), \hat{x}(t_1^u)\right)$$

Real-Time NMPC More precisely

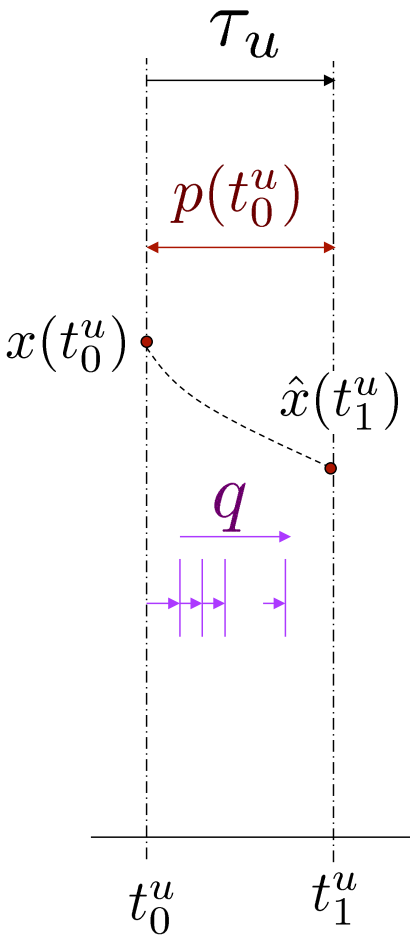


Define $\mathcal{P}\left(\hat{x}(t_1^u)\right)$

Initial guess $p^+(t_0^u)$

$$\mathcal{S} \circ \mathcal{S}\left(p^+(t_0^u), \hat{x}(t_1^u)\right)$$

Real-Time NMPC More precisely

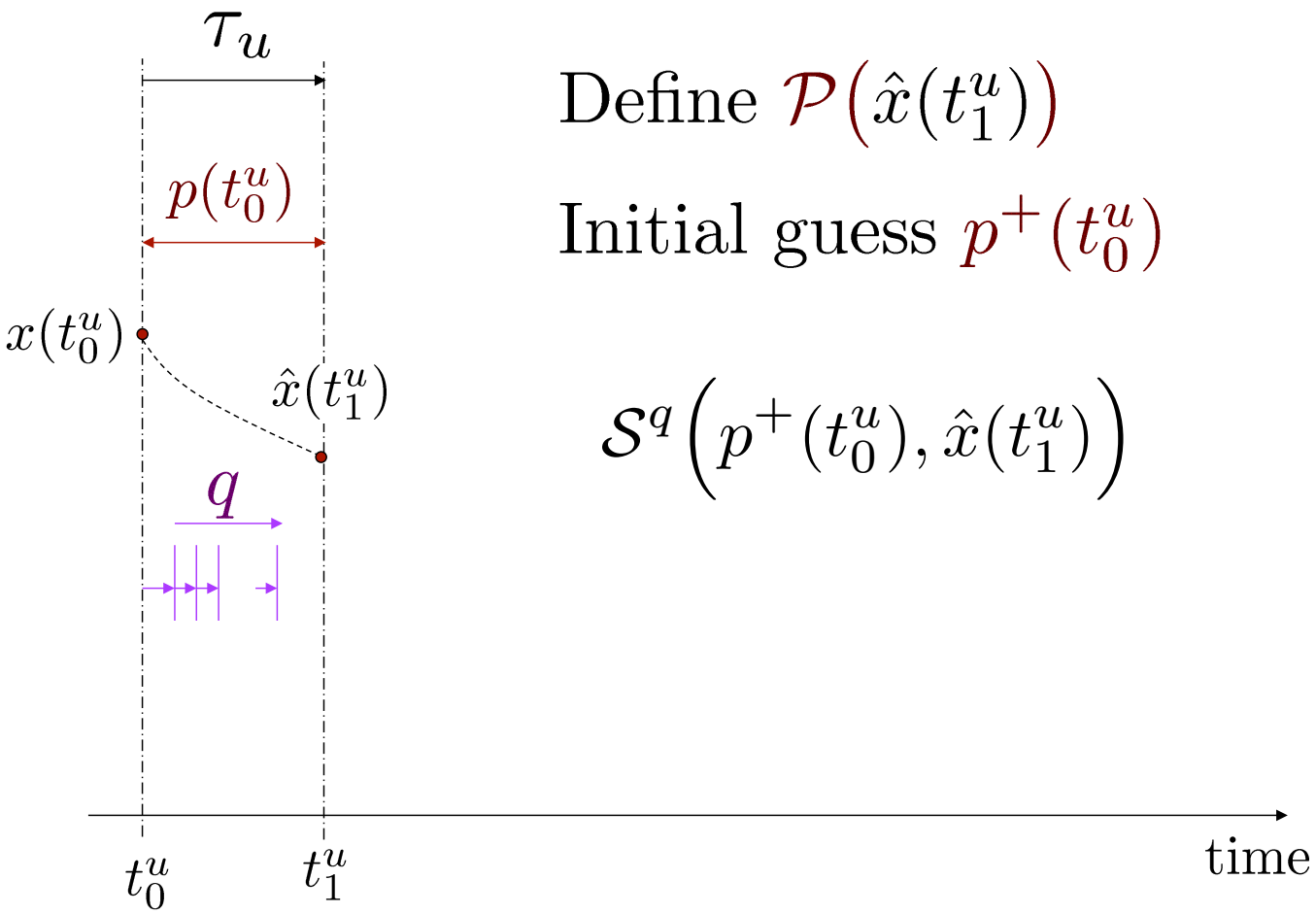


Define $\mathcal{P}\left(\hat{x}(t_1^u)\right)$

Initial guess $p^+(t_0^u)$

$$\mathcal{S} \circ \ldots \mathcal{S} \circ \mathcal{S}\left(p^+(t_0^u), \hat{x}(t_1^u)\right)$$

Real-Time NMPC More precisely

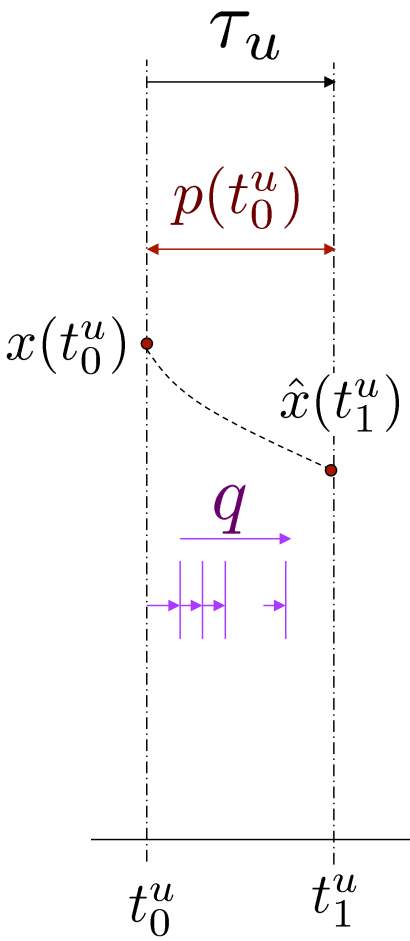


Define $\mathcal{P}\left(\hat{x}(t_1^u)\right)$

Initial guess $p^+(t_0^u)$

$$\mathcal{S}^q\left(p^+(t_0^u), \hat{x}(t_1^u)\right)$$

Real-Time NMPC More precisely

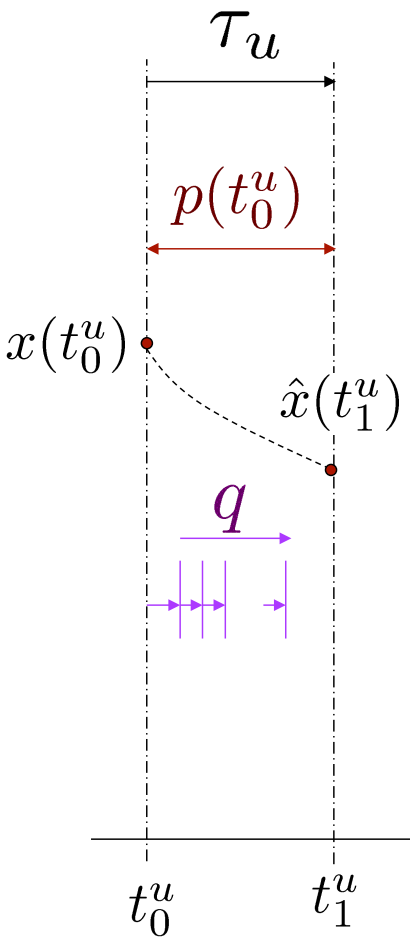


Define $\mathcal{P}\left(\hat{x}(t_1^u)\right)$

Initial guess $p^+(t_0^u)$

$$\mathcal{S}^q\left(p^+(t_0^u), \hat{x}(t_1^u)\right) \rightarrow p(t_k^u)$$

Real-Time NMPC More precisely



Define $\mathcal{P}\left(\hat{x}(t_1^u)\right)$

Initial guess $p^+(t_0^u)$

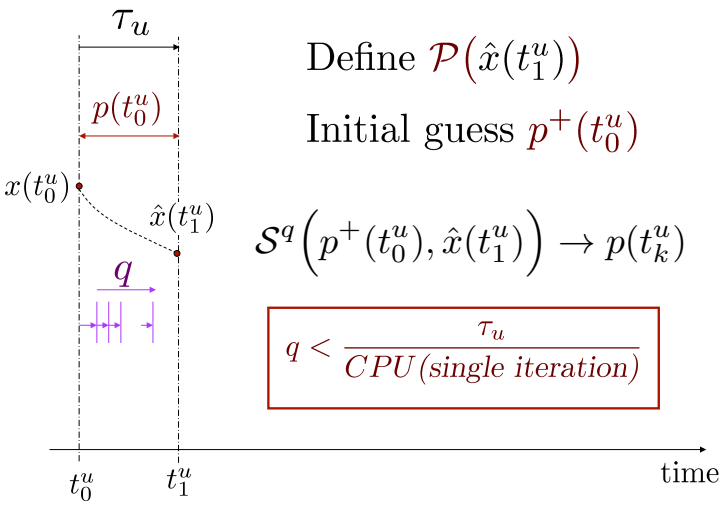
$$\mathcal{S}^q\left(p^+(t_0^u), \hat{x}(t_1^u)\right) \rightarrow p(t_k^u)$$

$$q < \frac{\tau_u}{CPU\,(single\ iteration)}$$

Outline

- Fast NMPC
- **The Control Updating Period Paradigm**
- The Proposed Updating Scheme
- Fast Gradient Related Considerations
- Numerical Investigation
- Conclusion & future works

The Control Updating Period Paradigm



Define $\mathcal{P}\left(\hat{x}(t_1^u)\right)$

Initial guess $p^+(t_0^u)$

$$\mathcal{S}^q\left(p^+(t_0^u), \hat{x}(t_1^u)\right) \rightarrow p(t_k^u)$$

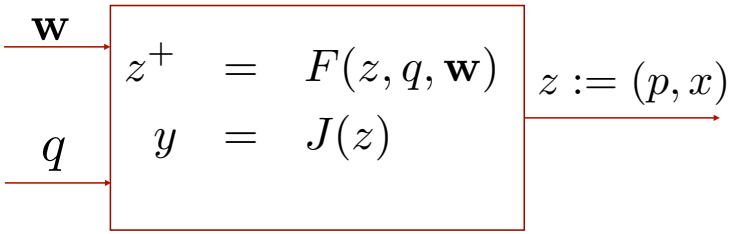
$$q < \frac{\tau_u}{CPU\,(single\ iteration)}$$

This results in the extended dynamic system:

$$\begin{aligned} x(t_{k+1}^u) &= X^r\left(q\tau, x(t_k^u), p(t_k^u), \mathbf{w}\right) \\ p(t_{k+1}^u) &= \mathcal{S}^q\left(p^+(t_k^u), \underbrace{X(q\tau, x(t_k^u), p(t_k^u))}_{\hat{x}(t_{k+1}^u)}\right) \end{aligned}$$

The Control Updating Period Paradigm

Given $\mathcal{S}$

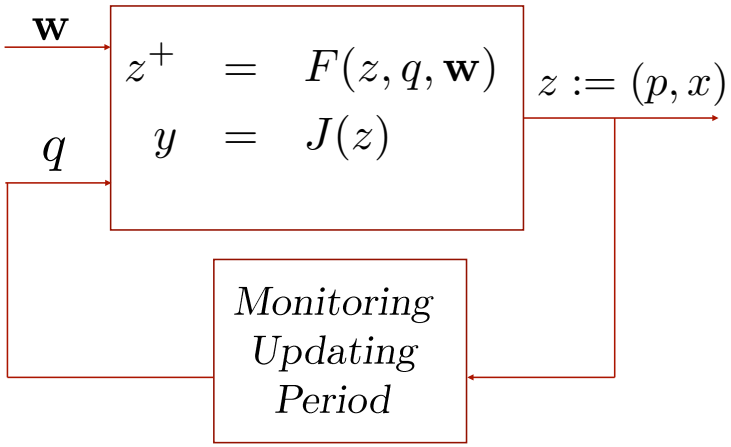


This results in the extended dynamic system:

$$
\begin{aligned}
x(t^u_{k+1}) &= X^r\left(q\tau, x(t^u_k), p(t^u_k), \mathbf{w}\right) \\
p(t^u_{k+1}) &= \mathcal{S}^q\Big(p^+(t^u_k), \underbrace{X(q\tau, x(t^u_k), p(t^u_k))}_{\hat{x}(t^u_{k+1})}\Big)
\end{aligned}
$$

The Control Updating Period Paradigm

Given $\mathcal{S}$

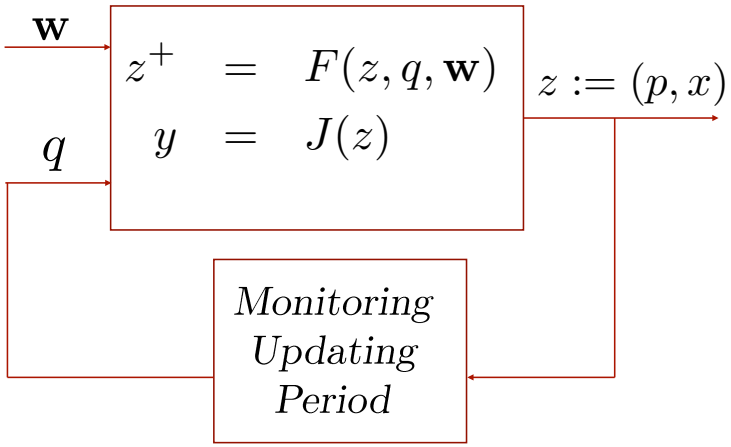


This results in the extended dynamic system:

$$
\begin{aligned}
x(t^u_{k+1}) &= X^r\left(q\tau, x(t^u_k), p(t^u_k), \mathbf{w}\right) \\
p(t^u_{k+1}) &= \mathcal{S}^q\Big(p^+(t^u_k), \underbrace{X(q\tau, x(t^u_k), p(t^u_k))}_{\hat{x}(t^u_{k+1})}\Big)
\end{aligned}
$$

The Control Updating Period Paradigm

Given $\mathcal{S}$



Problem Statement

Given a solver $\mathcal{S}$, propose a concrete (on-line) feedback:

$$q(t_k^u) = K(z(t_k^u), \dots)$$

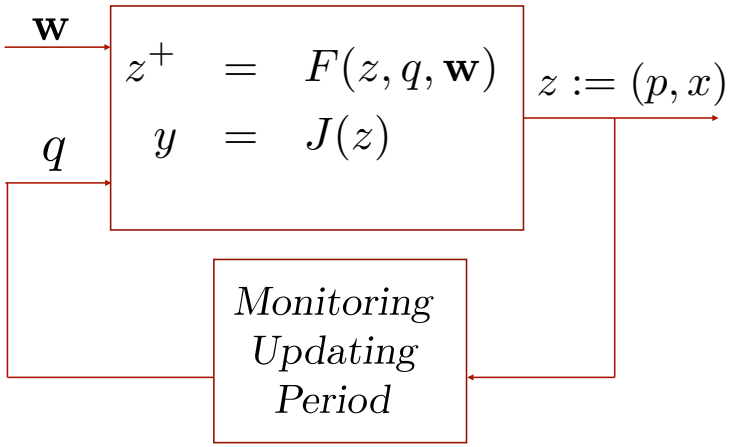
steering y to its minimum value.

This results in the extended dynamic system:

$$\begin{aligned} x(t_{k+1}^u) &= X^r\left(q\tau, x(t_k^u), p(t_k^u), \mathbf{w}\right) \\ p(t_{k+1}^u) &= \mathcal{S}^q\Big(p^+(t_k^u), \underbrace{X(q\tau, x(t_k^u), p(t_k^u))}_{\hat{x}(t_{k+1}^u)}\Big) \end{aligned}$$

The Control Updating Period Paradigm

Given $\mathcal{S}$



Problem Statement

Given a solver $\mathcal{S}$, propose a concrete (on-line) feedback:

$$q(t_k^u) = K(z(t_k^u), \dots)$$

steering y to its minimum value.

NOTA The computation time needed for this feedback must be negligible.

This results in the extended dynamic system:

$$\begin{aligned} x(t_{k+1}^u) &= X^r\left(q\tau, x(t_k^u), p(t_k^u), \mathbf{w}\right) \\ p(t_{k+1}^u) &= \mathcal{S}^q\Big(p^+(t_k^u), \underbrace{X(q\tau, x(t_k^u), p(t_k^u))}_{\hat{x}(t_{k+1}^u)}\Big) \end{aligned}$$

A Small Gain Paradigm

Preliminary Assumption

There is a positive $\underline{J} > 0$ such that for all (p, x) of interest, the inequality:

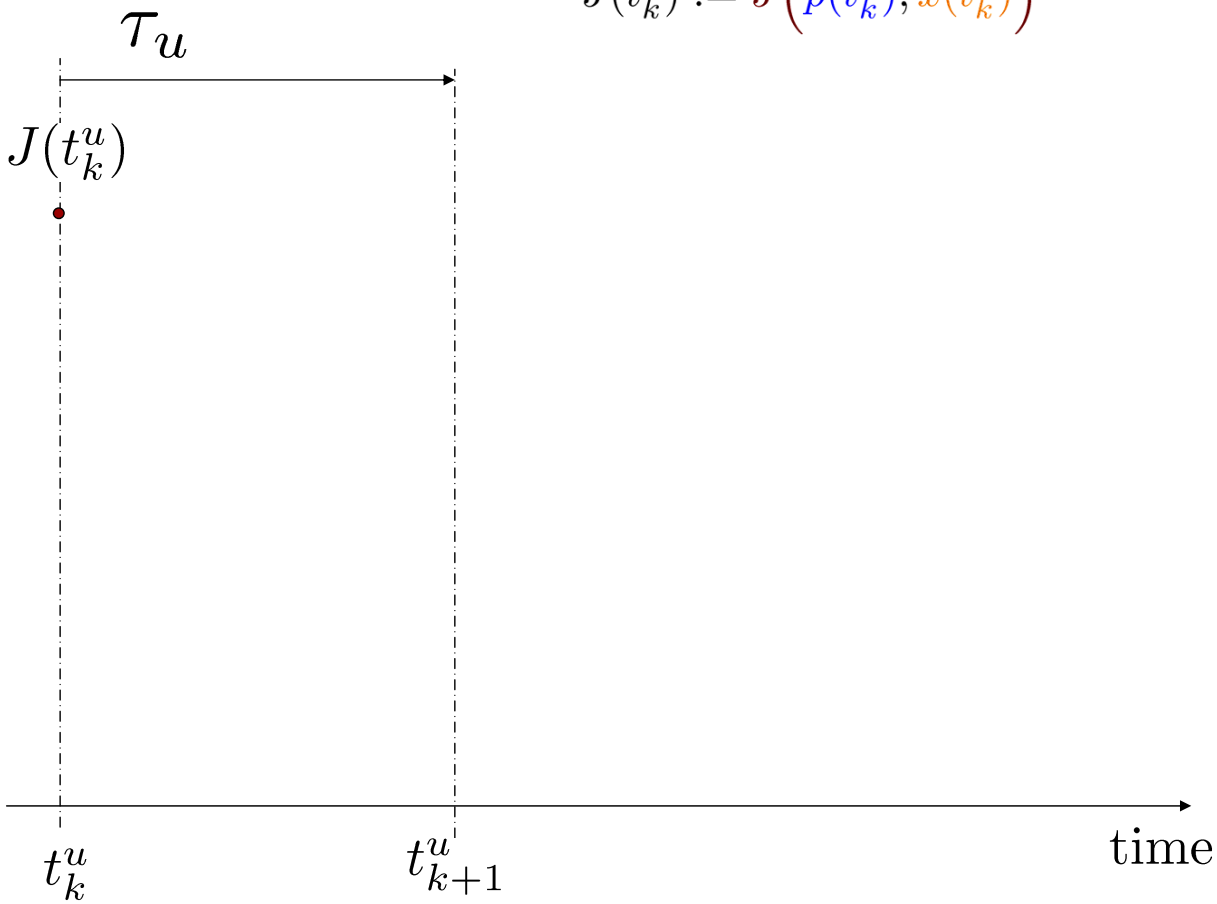
$$J(p, x) \geq \underline{J} > 0$$

is satisfied

This can always be satisfied by adding sufficiently high constant to the cost function's definition.

A Small Gain Paradigm

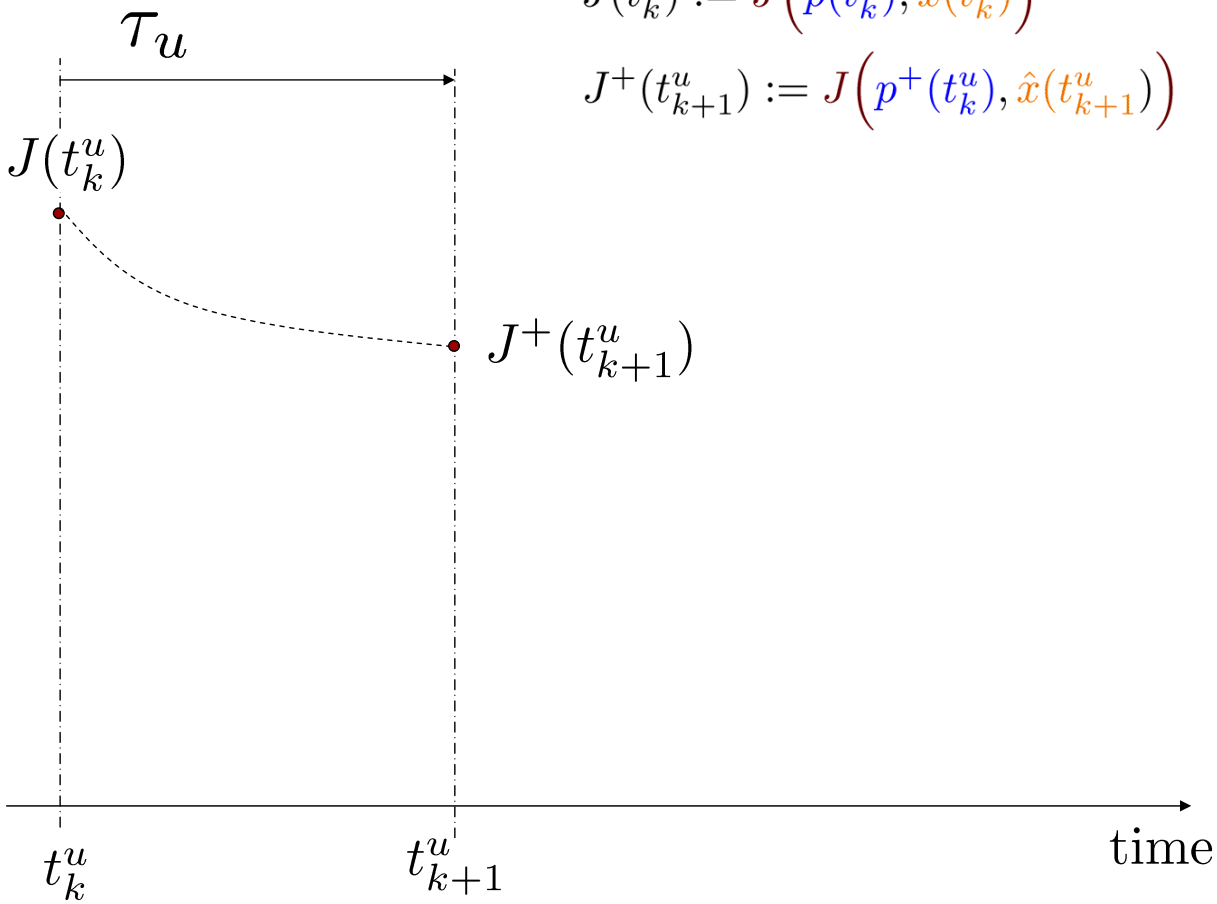
$$J(t_k^u) := J\Big(p(t_k^u), x(t_k^u)\Big)$$



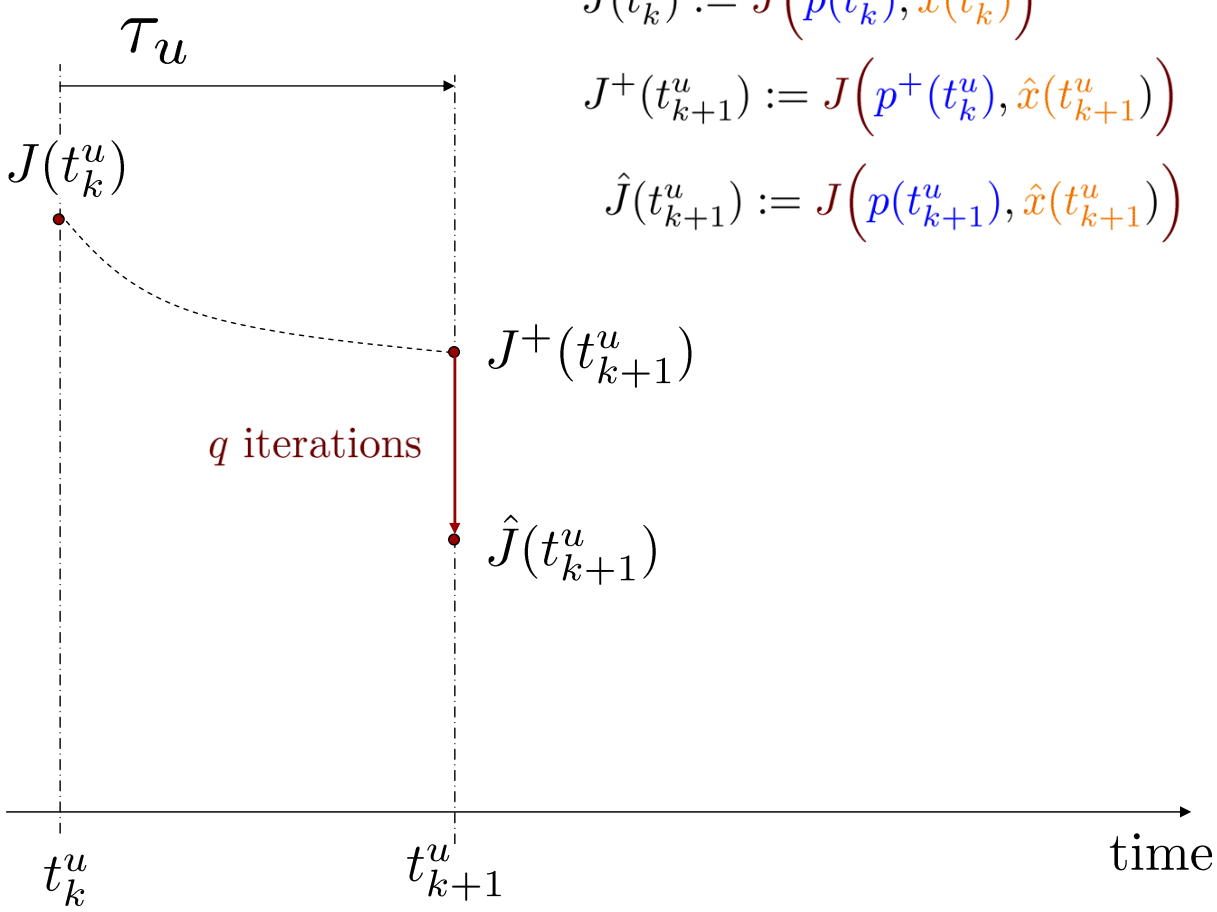
A Small Gain Paradigm

$$J(t_k^u) := J\Big(p(t_k^u), x(t_k^u)\Big)$$

$$J^+(t_{k+1}^u) := J\Big(p^+(t_k^u), \hat{x}(t_{k+1}^u)\Big)$$



A Small Gain Paradigm

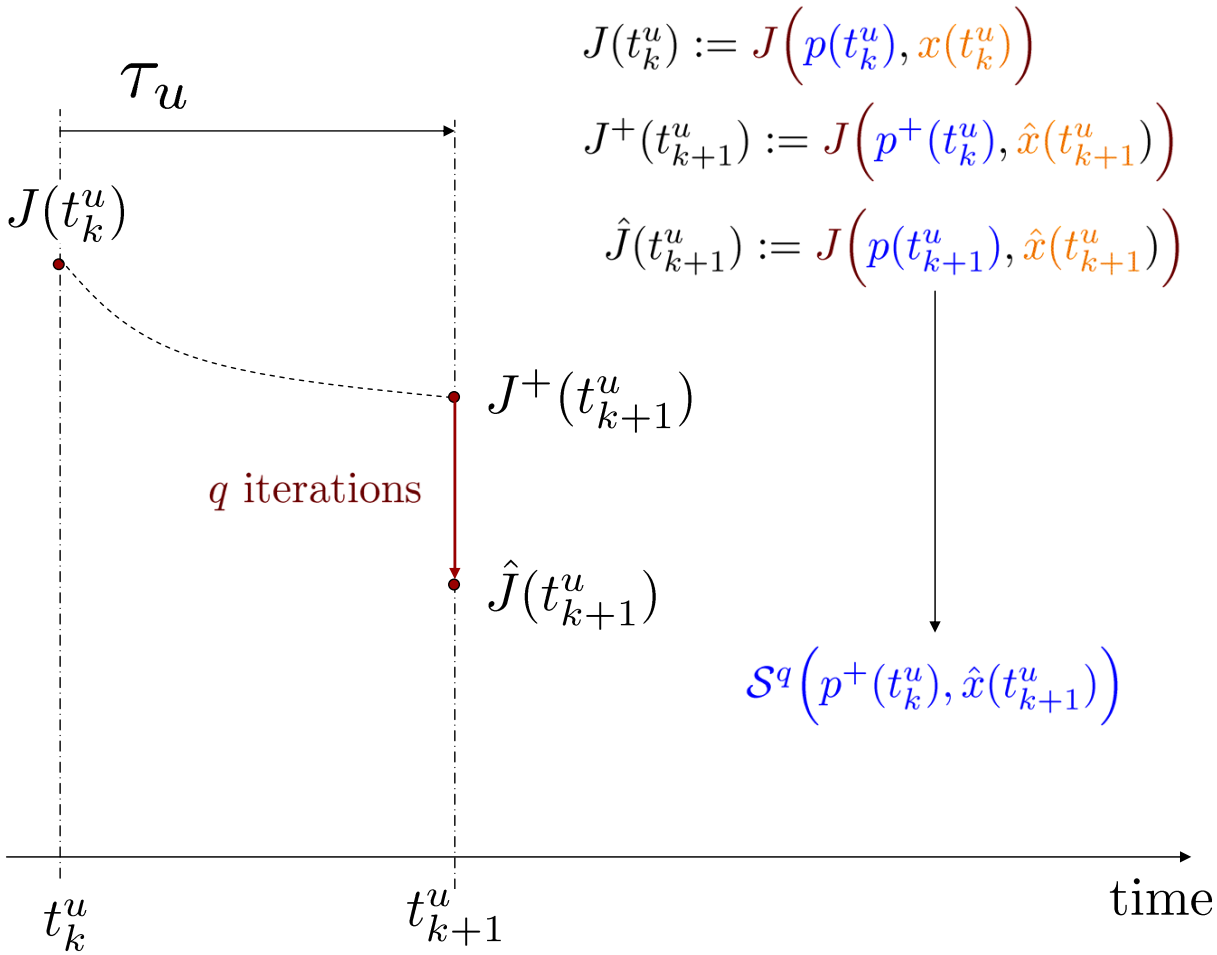


$$J(t_k^u) := J\Big(p(t_k^u), x(t_k^u)\Big)$$

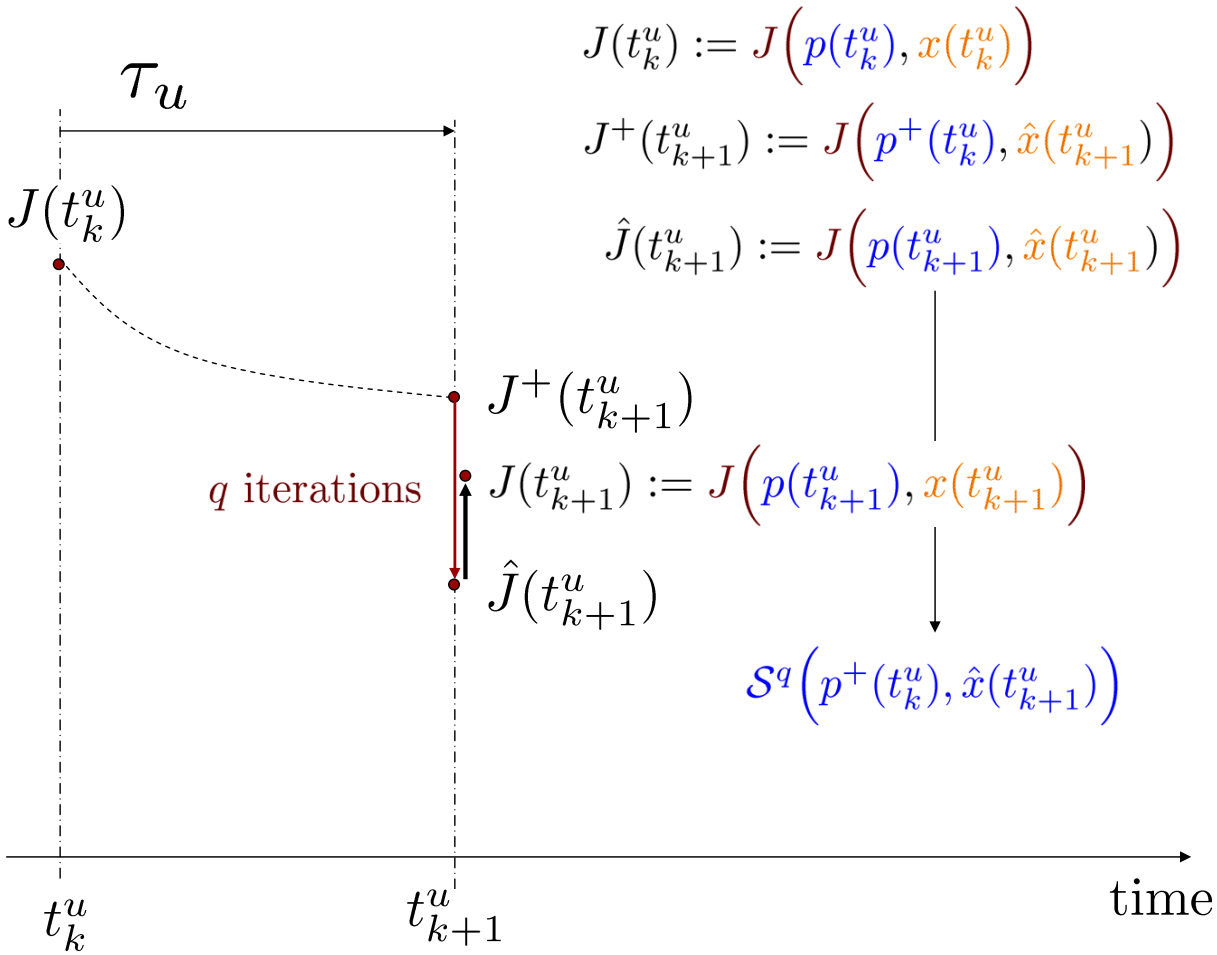
$$J^+(t_{k+1}^u) := J\Big(p^+(t_k^u), \hat{x}(t_{k+1}^u)\Big)$$

$$\hat{J}(t_{k+1}^u) := J\Big(p(t_{k+1}^u), \hat{x}(t_{k+1}^u)\Big)$$

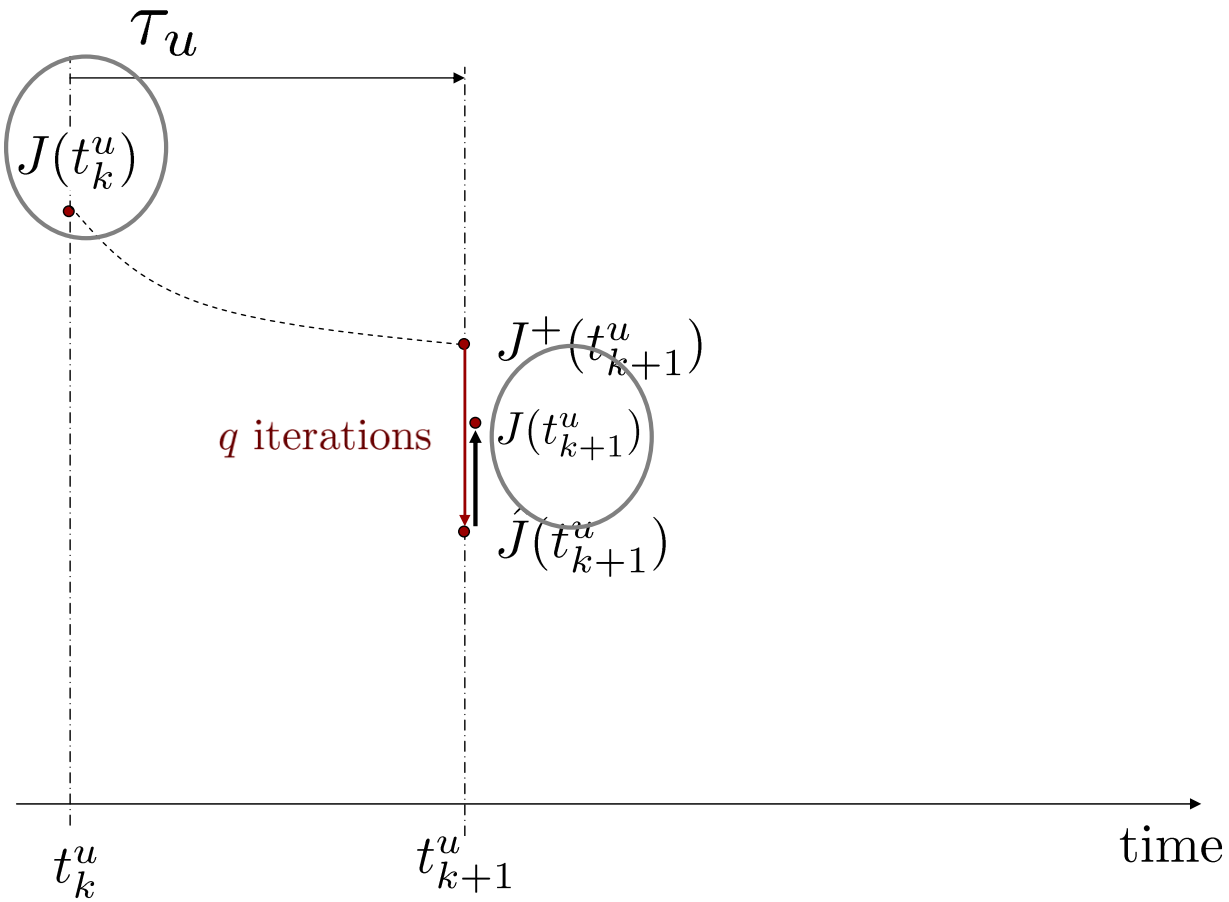
A Small Gain Paradigm



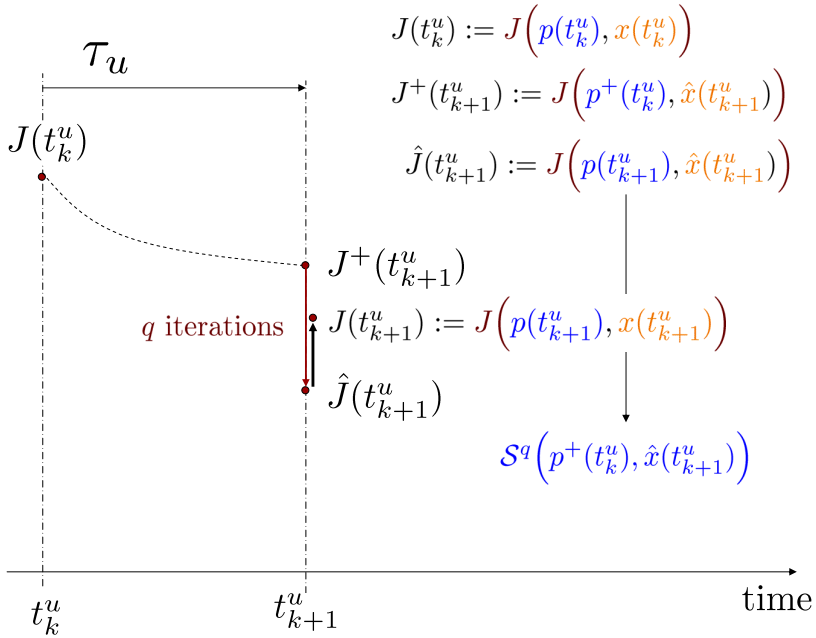
A Small Gain Paradigm



A Small Gain Paradigm

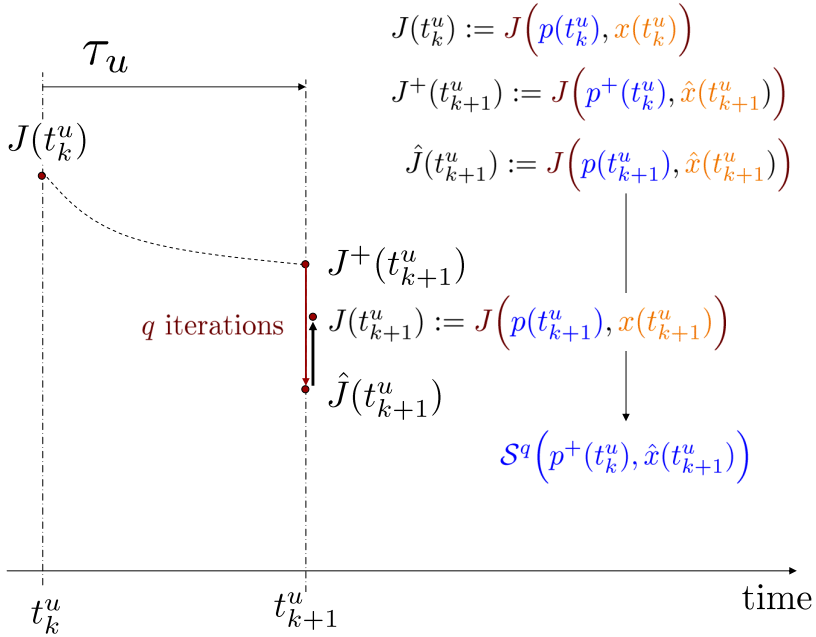


A Small Gain Paradigm



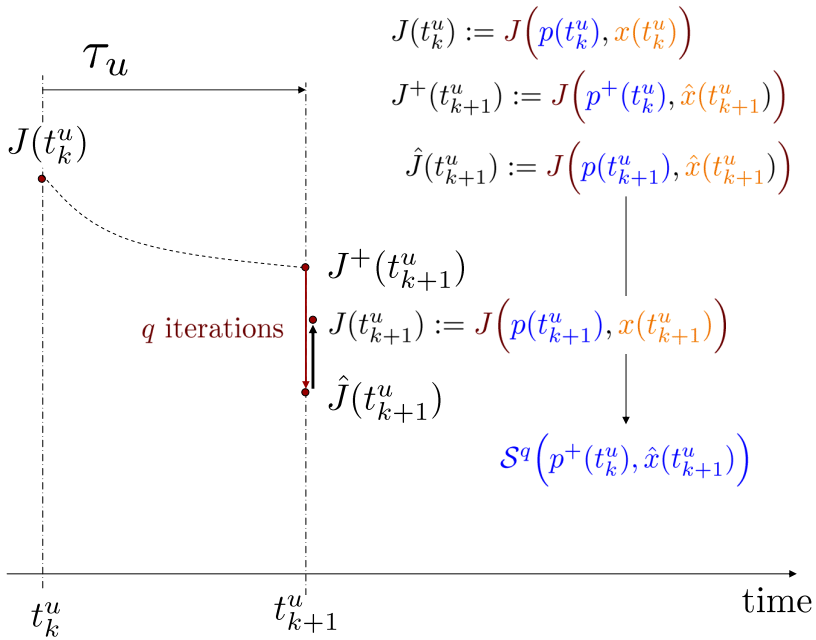
$$\frac{J(t_{k+1}^u)}{J(t_k^u)} = \left[\frac{J(t_{k+1}^u)}{\hat{J}(t_{k+1}^u)}\right] \times \left[\frac{\hat{J}(t_{k+1}^u)}{J^+(t_{k+1}^u)}\right] \times \left[\frac{J^+(t_{k+1}^u)}{J(t_k^u)}\right]$$

A Small Gain Paradigm



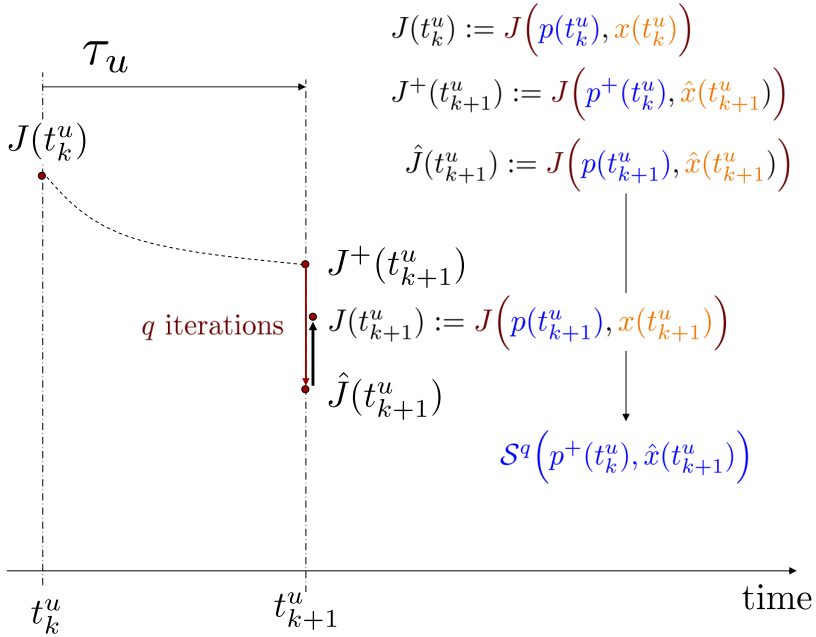
$$\frac{J(t_{k+1}^u)}{J(t_k^u)} = \left[\frac{J(t_{k+1}^u)}{\hat{J}(t_{k+1}^u)}\right] \times \underbrace{\left[\frac{\hat{J}(t_{k+1}^u)}{J^+(t_{k+1}^u)}\right]}_{E_k(q(t_k^u))} \times \left[\frac{J^+(t_{k+1}^u)}{J(t_k^u)}\right]$$

A Small Gain Paradigm



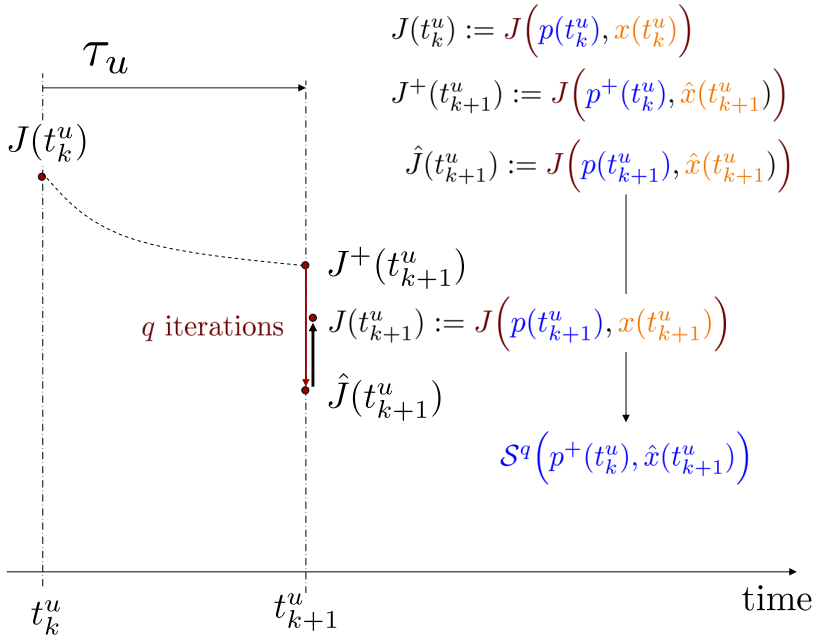
$$\frac{J(t_{k+1}^u)}{J(t_k^u)} = \underbrace{\left[\frac{J(t_{k+1}^u)}{\hat{J}(t_{k+1}^u)}\right]}_{\text{Uncertainty}} \times \underbrace{\left[\frac{\hat{J}(t_{k+1}^u)}{J^+(t_{k+1}^u)}\right]}_{E_k(q(t_k^u))} \times \left[\frac{J^+(t_{k+1}^u)}{J(t_k^u)}\right]$$

A Small Gain Paradigm



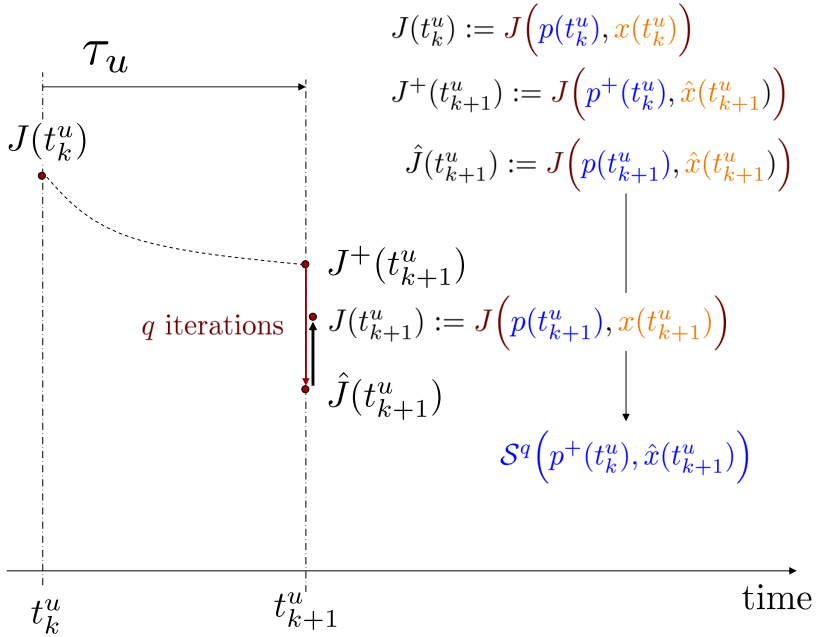
$$\frac{J(t_{k+1}^u)}{J(t_k^u)} = \underbrace{\left[\frac{J(t_{k+1}^u)}{\hat{J}(t_{k+1}^u)}\right]}_{\text{Uncertainty}} \times \underbrace{\left[\frac{\hat{J}(t_{k+1}^u)}{J^+(t_{k+1}^u)}\right]}_{E_k(q(t_k^u))} \times \underbrace{\left[\frac{J^+(t_{k+1}^u)}{J(t_k^u)}\right]}_{\text{Parametrization}}$$

A Small Gain Paradigm



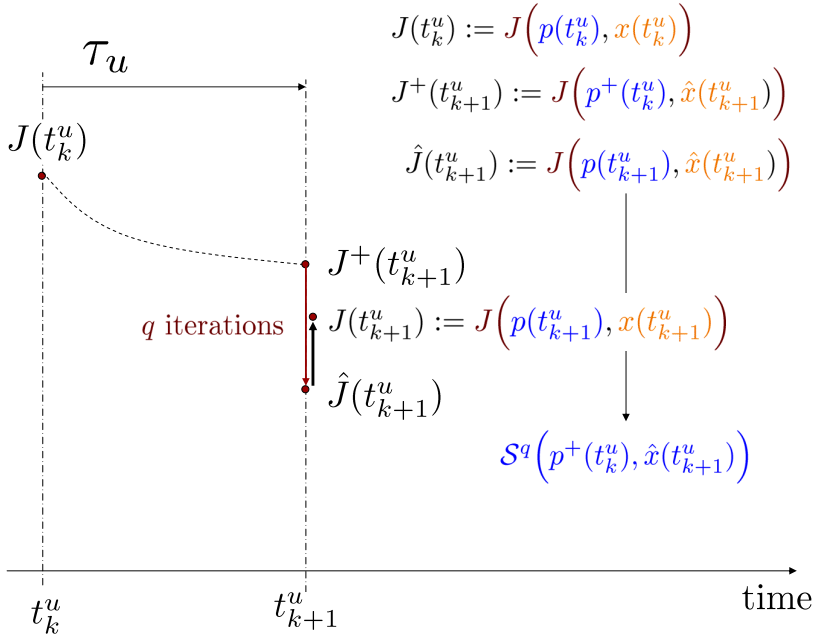
$$\frac{J(t_{k+1}^u)}{J(t_k^u)} = \underbrace{\left[\frac{\hat{J}(t_{k+1}^u)}{J^+(t_{k+1}^u)}\right]}_{E_k(q(t_k^u))} \times \underbrace{\left[\frac{J(t_{k+1}^u)}{\hat{J}(t_{k+1}^u)}\right]}_{\text{Uncertainty}} \times \underbrace{\left[\frac{J^+(t_{k+1}^u)}{J(t_k^u)}\right]}_{\text{Parametrization}}$$

A Small Gain Paradigm



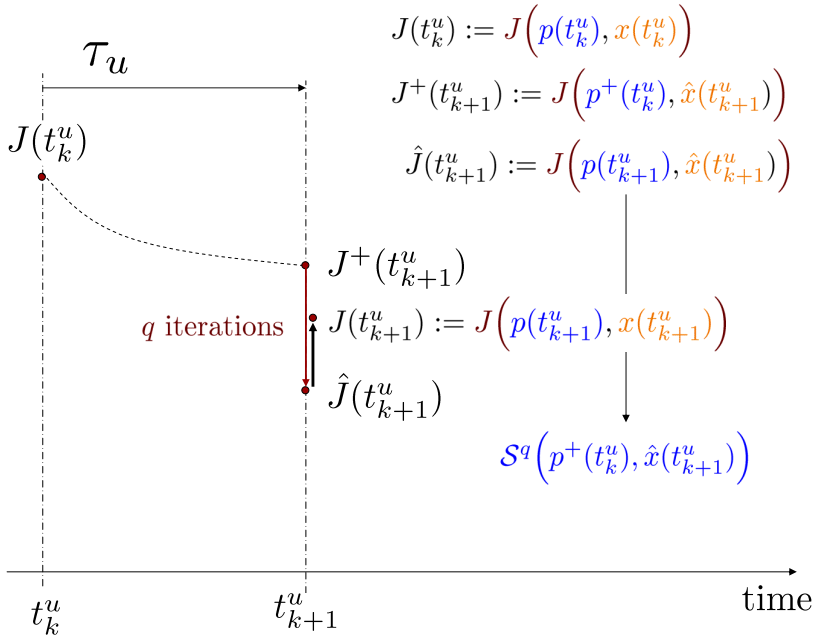
$$\frac{J(t_{k+1}^u)}{J(t_k^u)} = \underbrace{\left[\frac{\hat{J}(t_{k+1}^u)}{J^+(t_{k+1}^u)}\right]}_{E_k(q(t_k^u))} \times \underbrace{\left[\frac{J(t_{k+1}^u)}{\hat{J}(t_{k+1}^u)}\right] \times \left[\frac{J^+(t_{k+1}^u)}{J(t_k^u)}\right]}_{D_k(q(t_k^u))}$$

A Small Gain Paradigm



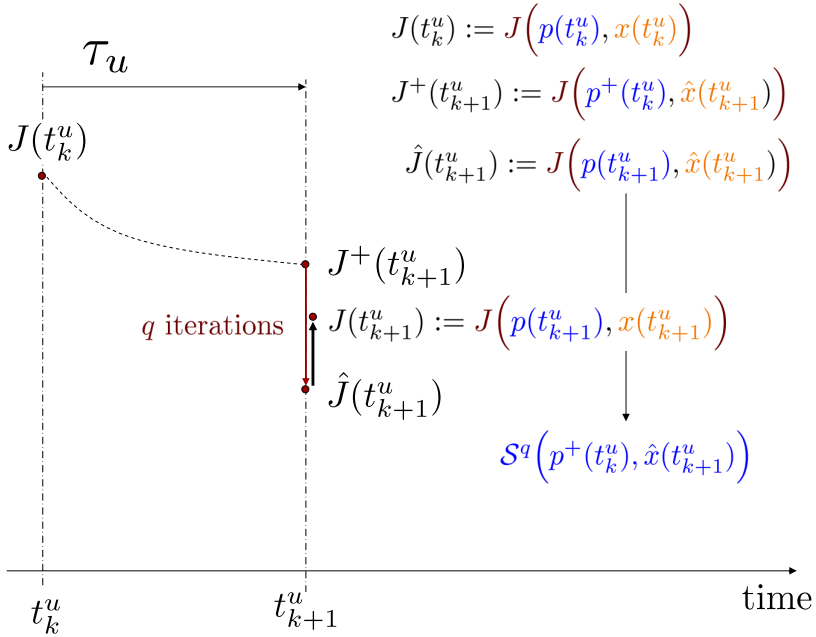
$$\frac{J(t_{k+1}^u)}{J(t_k^u)} = \underbrace{\left[\frac{\hat{J}(t_{k+1}^u)}{J^+(t_{k+1}^u)}\right]}_{E_k(q(t_k^u))} \times \underbrace{\left[\frac{J(t_{k+1}^u)}{\hat{J}(t_{k+1}^u)}\right] \times \left[\frac{J^+(t_{k+1}^u)}{J(t_k^u)}\right]}_{D_k(q(t_k^u))}$$

A Small Gain Paradigm



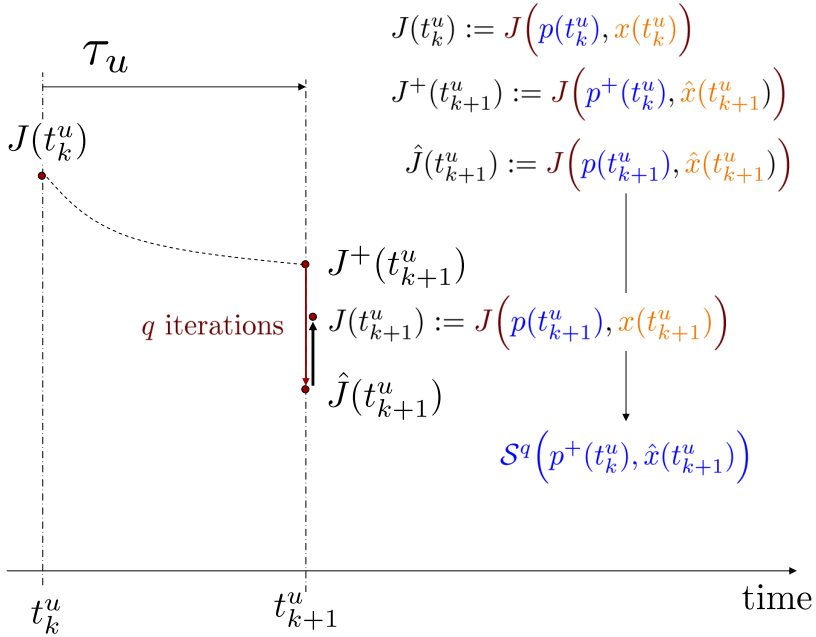
$$J(t_{k+1}^u) = \left[E_k(q(t_k^u))\right] \cdot \left[D_k(q(t_k^u))\right] \cdot J(t_k^u)$$

A Small Gain Paradigm



$$
\begin{aligned}
J(t_{k+1}^u) &= \left[E_k(q(t_k^u))\right] \cdot \left[D_k(q(t_k^u))\right] \cdot J(t_k^u) \\
&= \left[K_k(q(t_k^u))\right] \cdot J(t_k^u)
\end{aligned}
$$

A Small Gain Paradigm

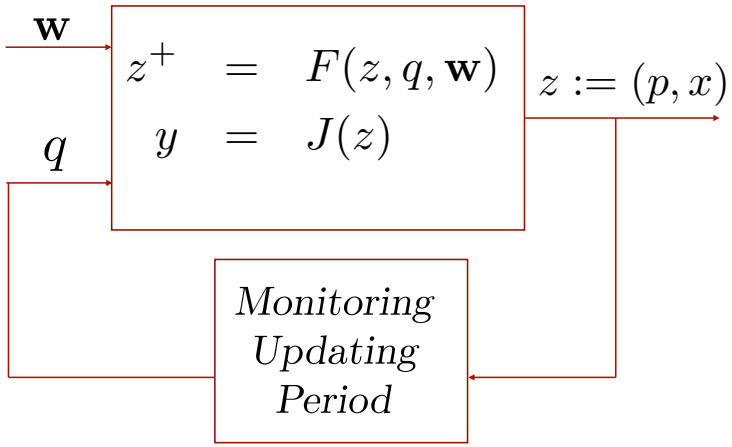


$$
\begin{aligned}
J(t_{k+1}^u) &= \Big[E_k(q(t_k^u))\Big] \cdot \Big[D_k(q(t_k^u))\Big] \cdot J(t_k^u) \\
&= \Big[K_k(q(t_k^u))\Big] \cdot J(t_k^u)
\end{aligned}
$$

NOTA: If $K_k < 1$, the settling time is $t_r(q(t_k^u)) \sim \dfrac{q(t_k^u)}{|\log\Big(K_k(q(t_k^u))\Big)|}$

The Ideal Updating Scheme Principle

Given $\mathcal{S}$



Problem Statement

Given a solver $\mathcal{S}$, propose a concrete (on-line) feedback:

$$q(t_k^u) = K(z(t_k^u), \dots)$$

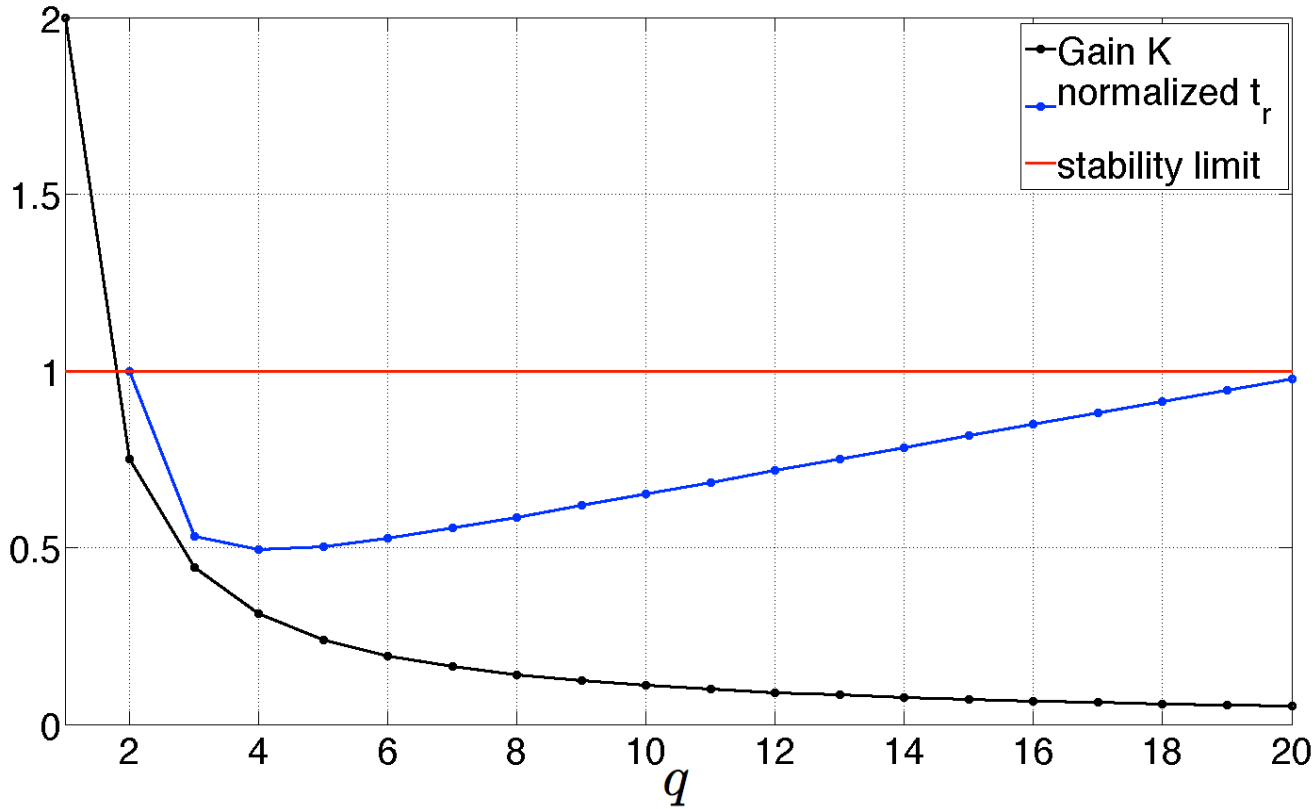
steering y to its minimum value.

$$q(t_{k+1}^u) := \begin{cases} \arg \min\limits_{q \in \{1, q_{max}\}} \dfrac{q}{|\log(K_k(q))|} & \text{if } K_k(q(t_k^u)) < 1 \\ \arg \min\limits_{q \in \{1, q_{max}\}} K_k(q) & \text{otherwise} \end{cases}$$

A one-step scalar predictive control . . . !!

Numerical Investigation Small gain paradigm illustration

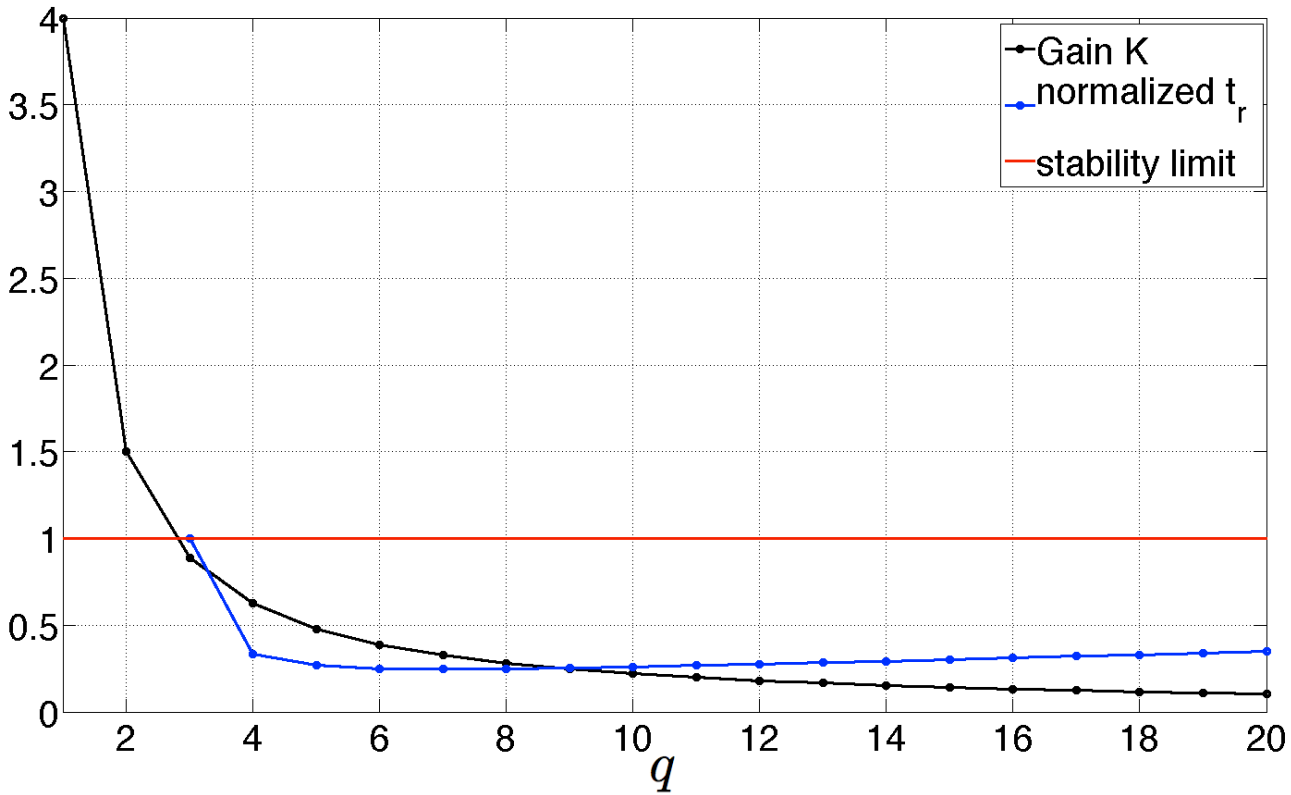
$$E(q) := \frac{1}{q^2} \quad ; \quad D(q) = (1 + 1 \cdot q)$$



$\Rightarrow$ Contraction needs $q \geq 2$, optimal $q = 4$

Numerical Investigation Small gain paradigm illustration

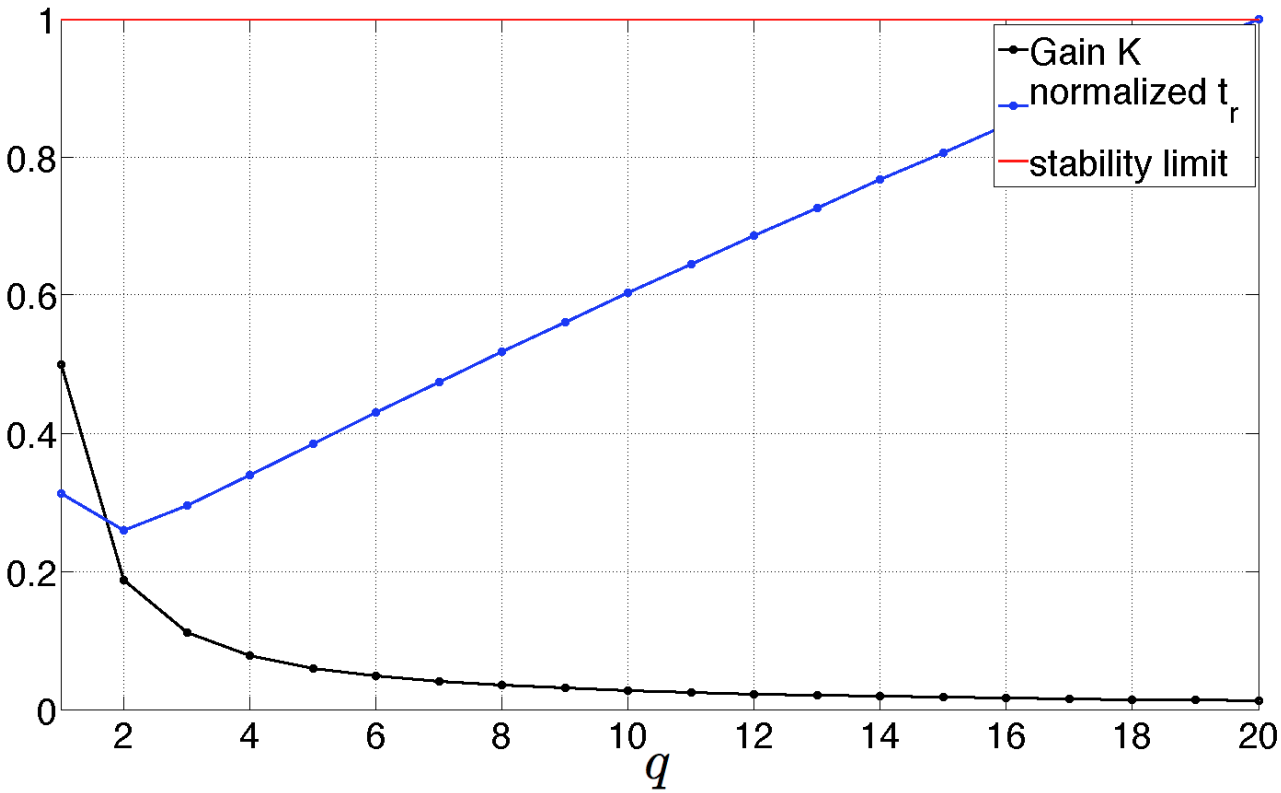
$$E(q) := \frac{2}{q^2} \quad ; \quad D(q) = (1 + 1 \cdot q)$$



$\Rightarrow$ Contraction needs $q \geq 3$, optimal $q = 6$

Numerical Investigation Small gain paradigm illustration

$$E(q) := \frac{0.25}{q^2} \quad ; \quad D(q) = (1 + 1 \cdot q)$$



$\Rightarrow$ Contraction needs $q \geq 1$, optimal $q = 2$

Outline

- Fast NMPC
- The Control Updating Period Paradigm
- **The Proposed Updating Scheme**
- Fast Gradient Related Considerations
- Numerical Investigation
- Conclusion & future works

The Ideal Updating Scheme Discussion ...

$$q(t^u_{k+1}) := \begin{cases} \arg \min\limits_{q \in \{1, q_{max}\}} \dfrac{q}{|\log(K_k(q))|} & \text{if } K_k(q(t^u_k)) < 1 \\ \arg \min\limits_{q \in \{1, q_{max}\}} K_k(q) & \text{otherwise} \end{cases}$$

Recall that:

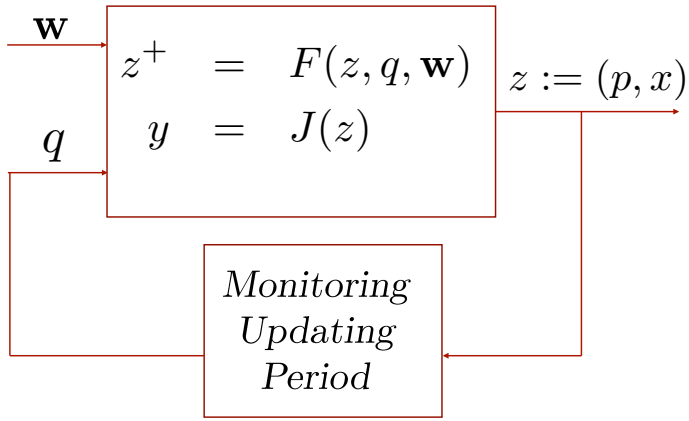
$$K_k(q) := E_k(q) \times D_k(q)$$

- $E_k(q)$ is badly known
- On-line identification based on a parametric structure:

  $$E_k(q) = \mathcal{E}(q, p_E(k))$$

  [Alamir, Springer, 2008]
- Critical when $q(t^u_k)$ is close to 1.

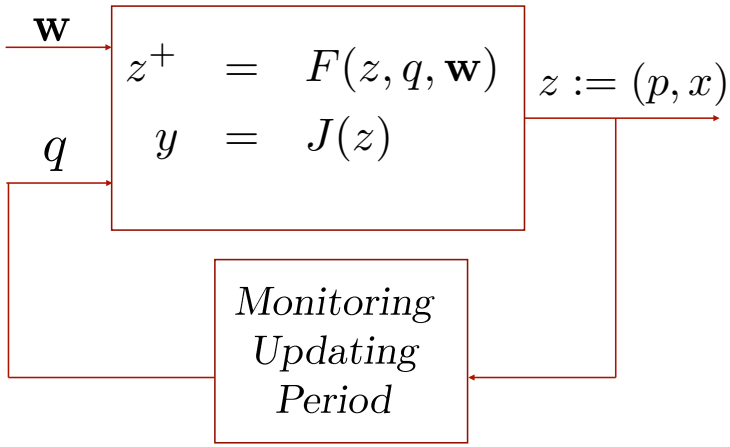
Given $\mathcal{S}$



The Ideal Updating Scheme

Idea ...

Given $\mathcal{S}$



Problem Statement

Given a solver $\mathcal{S}$, propose a concrete (on-line) feedback:

$$q(t_k^u) = K(z(t_k^u), \dots)$$

stabilizing $y = 0$.

$$q(t_{k+1}^u) := \arg \min_{q \in \{1, q_{max}\}} \begin{cases} \dfrac{q}{|\log(K_k(q))|} & \text{if } K_k(q(t_k^u)) < 1 \\ K_k(q) & \text{otherwise} \end{cases}$$

A one-step scalar predictive control . . . !!

→ Distribute the optimization over time . . . AGAIN !

Updating Algorithm

$$q(t_{k+1}^u) := \arg \min_{q \in \{1, q_{max}\}} \begin{cases} \dfrac{q}{|\log(K_k(q))|} & \text{if } K_k(q(t_k^u)) < 1 \\ K_k(q) & \text{otherwise} \end{cases}$$

Algorithm 1 Updating rule $q(t_{k+1}^u) = U(q(t_k^u), t_k^u)$

1: **If** $(K_k \geq 1)$ **then**

2: $\quad \Gamma \leftarrow \dfrac{\Delta K_k}{\Delta q}(q(t_k^u))$

3: **Else**

4: $\quad \Gamma \leftarrow \dfrac{\Delta(q/|\log(K_k(q))|)}{\Delta q} = \dfrac{-\log(K_k) + \dfrac{q}{K_k} \times \dfrac{\Delta K_k}{\Delta q}(q(t_k^u)}{[\log(K_k)]^2}$

5: **End If**

6: $q(t_{k+1}^u) \leftarrow \max\Big\{2, \min\big\{q_{max}, q(t_k^u) - \delta \cdot sign(\Gamma)\big\}\Big\}$

Updating Algorithm

$$q(t_{k+1}^u) := \arg \min_{q \in \{1, q_{max}\}} \begin{cases} \dfrac{q}{|\log(K_k(q))|} & \text{if } K_k(q(t_k^u)) < 1 \\ K_k(q) & \text{otherwise} \end{cases}$$

Algorithm 4 Updating rule $q(t_{k+1}^u) = U(q(t_k^u), t_k^u)$

1: **If** $(K_k \geq 1)$ **then**

2: $\quad \Gamma \leftarrow \dfrac{\Delta K_k}{\Delta q}(q(t_k^u))$

3: **Else**

4: $\quad \Gamma \leftarrow \dfrac{\Delta(q/|\log(K_k(q))|)}{\Delta q} = \dfrac{-\log(K_k) + \dfrac{q}{K_k} \times \dfrac{\Delta K_k}{\Delta q}(q(t_k^u)}{[\log(K_k)]^2}$

5: **End If**

6: $q(t_{k+1}^u) \leftarrow \max\Big\{2, \min\big\{q_{max}, q(t_k^u) - \delta \cdot sign(\Gamma)\big\}\Big\}$

Need to compute $K_k(q(t_k^u))$ and $\dfrac{\Delta K_k}{\Delta q}(q(t_k^u))$

Sensitivity Computation E_k-related quantities

Recall that:

$$E_k(q) := \frac{J(p^{(q)}, \hat{x}(t^u_{k+1}))}{J(p^{(0)}, \hat{x}(t^u_{k+1}))} \quad ; \quad p^{(0)} := p^+(t^u_k)$$

therefore:

$$\frac{\Delta E_k}{\Delta q}(q(t^u_k)) \approx \frac{J(p^{(q(t^u_k))}, \hat{x}(t^u_{k+1})) - J(p^{(q(t^u_k)-1)}, \hat{x}(t^u_{k+1}))}{J(p^{(0)}, \hat{x}(t^u_{k+1}))}$$

$\rightarrow$ q need to be greater than 2.

$\rightarrow$ $E_k(q(t^u_k))$ and $\frac{\Delta E_k}{\Delta q}(q(t^u_k))$ are computable from available computer data at the cost of **2 divisions and 1 addition**

Sensitivity Computation D_k-related quantities

Consider the simple uncertainty model:

$$D_k(q) = 1 + [\alpha_D] \cdot q$$

recall that one has by definition:

$$D_k(q(t_k^u)) := \frac{J_{k+1} \times J_k^+}{J_k \times \hat{J}_{k+1}} = \frac{1}{E_k} \frac{J_{k+1}}{J_k}$$

$$\Rightarrow \quad \alpha_D := \frac{\Delta D_k}{\Delta q}(q(t_k^u)) \approx \frac{1}{q(t_k^u)} \times \left[\frac{1}{E_k} \frac{J_{k+1}}{J_k} - 1\right]$$

$\rightarrow D_k(q(t_k^u))$ and $\frac{\Delta D_k}{\Delta q}(q(t_k^u))$ are computable from available computer data at the cost of **2 divisions and 1 multiplication**

$$\rightarrow \frac{\Delta K_k}{\Delta q} = E_k \frac{\Delta D_k}{\Delta q} + D_k \frac{\Delta E_k}{\Delta q}$$

Convergence

Convergence depends on the issue of a COMPETITION between:

1. A DECREASE in $t_r(q)$ due to the quantized gradient descent
2. A *potential* INCREASE due to change between t_k^u and t_{k+1}^u.

Does $t_r(q)$ show a unique local minimum on $[2, q_{max}]$?

This is more likely to be true if $E_k(\cdot)$ is strictly monotonically decreasing.

What about $E_k(\cdot)$ in the case of fast gradient . . . ?

Outline

- Fast NMPC
- The Control Updating Period Paradigm
- The Proposed Updating Scheme
- **Fast Gradient Related Considerations**
- Numerical Investigation
- Conclusion & future works

Fast gradient descent Definition, [Y. Nesterov, 1983]

Algorithm 2 Fast gradient iterations $\bar{p} = \mathcal{S}^{(q)}(p, x)$

1: Initialization. $p^{(0)} \leftarrow p$, $r \leftarrow p$
2: **for** $i = 1, q$ **do**
3: $\quad p^{(i)} \leftarrow P_C\big(r - \frac{1}{L}[\nabla J(r, x))]\big)$
4: $\quad r \leftarrow p^{(i)} + c(p^{(i)} - p^{(i-1)})$
5: **end for**
6: $\bar{p} \leftarrow p^{(q)}$

- $\forall (x, p_1, p_2)\ ,\ \|\nabla J(p_2, x) - \nabla J(p_1, x)\| \leq L\|p_2 - p_1\|$
- $c \in [0, 1]$:
  - $c = 0 \rightarrow$ pure (slow) gradient descent
  - For quadratic problems $c = \dfrac{\sqrt{\lambda_{max}(H)} - \sqrt{\lambda_{min}(H)}}{\sqrt{\lambda_{max}(H)} + \sqrt{\lambda_{min}(H)}}$
- $P_C(\cdot)$: projection on the admissible set.

Fast gradient descent Definition, [Y. Nesterov, 1983]

Algorithm 2 Fast gradient iterations $\bar{p} = \mathcal{S}^{(q)}(p, x)$

1: Initialization. $p^{(0)} \leftarrow p$, $r \leftarrow p$
2: **for** $i = 1, q$ **do**
3: $\quad p^{(i)} \leftarrow P_C\big(r - \frac{1}{L}[\nabla J(r, x))]\big)$
4: $\quad r \leftarrow p^{(i)} + c(p^{(i)} - p^{(i-1)})$
5: **end for**
6: $\bar{p} \leftarrow p^{(q)}$

- $J(p^{(k)}, x) - J^*(x) \leq -\dfrac{\alpha(\|x - x^*\|)}{(k+2)^2}$ (optimal c)
- $r \leftarrow P_C(r - (\nabla J)/L) + \ldots$ [r is somehow an integrator state]
- Induced oscillations for high values of $c \in [0, 1]$.

B. O'Donoghue and A. Candes. Adaptive restart for accelerated gradient schemes. arxiv:1204.3982. April 2012.

- **Efficiency map $E_k(\cdot)$ of original Fast Gradient is not monotonic**

Fast gradient descent Definition, [Y. Nesterov, 1983]

Algorithm 3 Fast gradient iterations $\bar{p} = \mathcal{S}^{(q)}(p, x)$ with constant restarting strategy

1: Initialization. $p^{(0)} \leftarrow p$, $r \leftarrow p$, $s \leftarrow 0$
2: **for** $i = 1, q$ **do**
3: $\quad s \leftarrow s + 1$
4: $\quad p^{(i)} \leftarrow P_C\big(r - \frac{1}{L}[\nabla J(r, x))]\big)$
5: $\quad r \leftarrow p^{(i)} + c(p^{(i)} - p^{(i-1)})$
6: $\quad$ **if** $(s = s_{max})$ **then** $r \leftarrow p^{(i)}$, $s = 0$ **End if**
7: **end for**
8: $\bar{p} \leftarrow p^{(q)}$

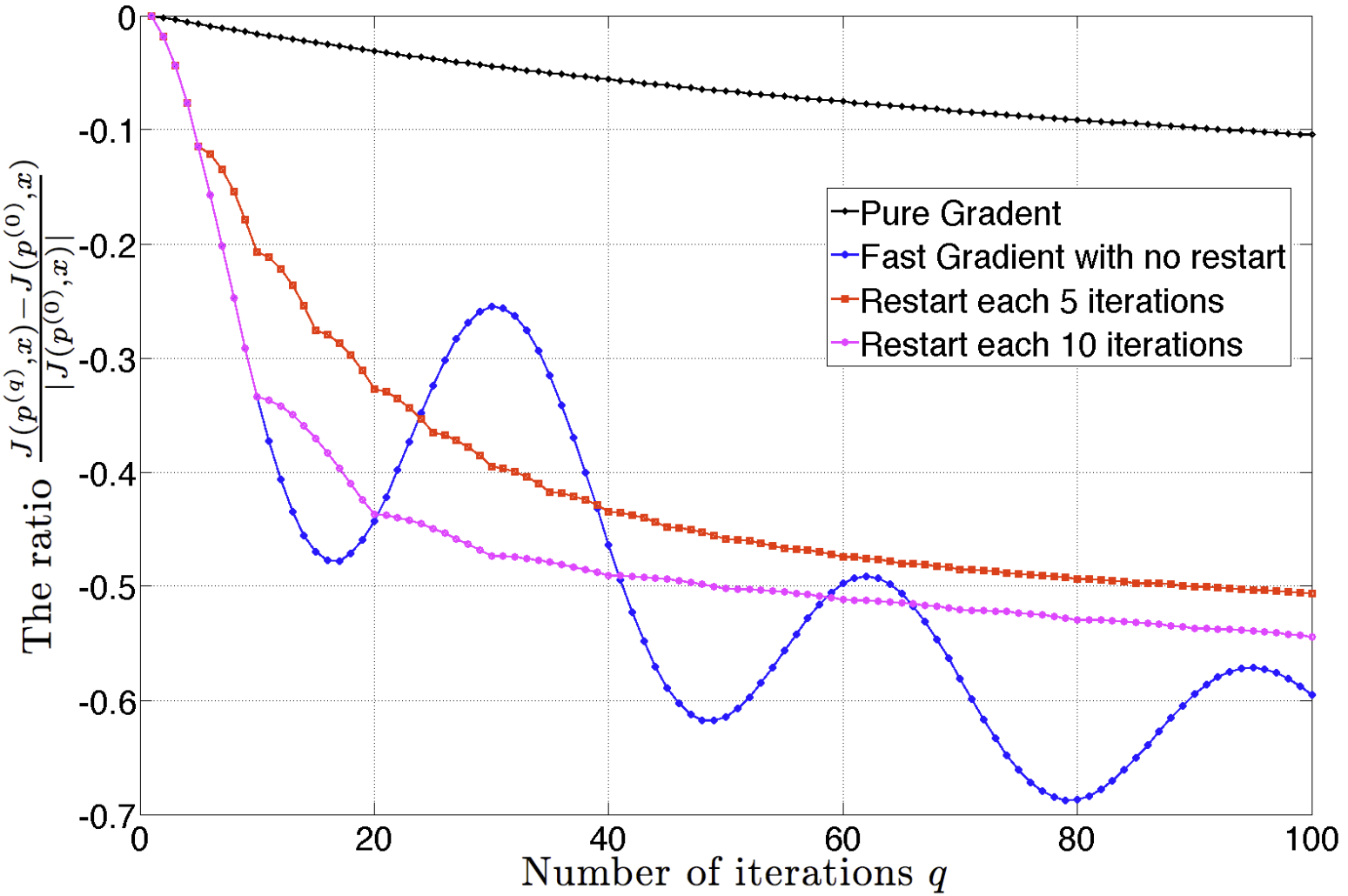
- $J(p^{(k)}, x) - J^*(x) \leq -\dfrac{\alpha(\|x - x^*\|)}{(k+2)^2}$ (optimal c)
- $r \leftarrow P_C(r - (\nabla J)/L) + \ldots$ [r is somehow an integrator state]
- Induced oscillations for high values of $c \in [0, 1]$.

B. O'Donoghue and A. Candes. Adaptive restart for accelerated gradient schemes. arxiv:1204.3982. April 2012.

- **Efficiency map $E_k(\cdot)$ of original Fast Gradient is not monotonic**

Fast Gradient with Restart Mechanism

Typical behavior of successive cost function values under pure gradient and fast gradient descent method without and with restarting mechanism



Outline

- Fast NMPC
- The Control Updating Period Paradigm
- The Proposed Updating Scheme
- Fast Gradient Related Considerations
- **Numerical Investigation**
- Conclusion & future works

Consider the following triple integrator system

$$\dot{x}_1 = x_2 \; ; \; \dot{x}_2 = x_3 \; ; \; \dot{x}_3 = u \quad ; \quad |u| \leq 1$$

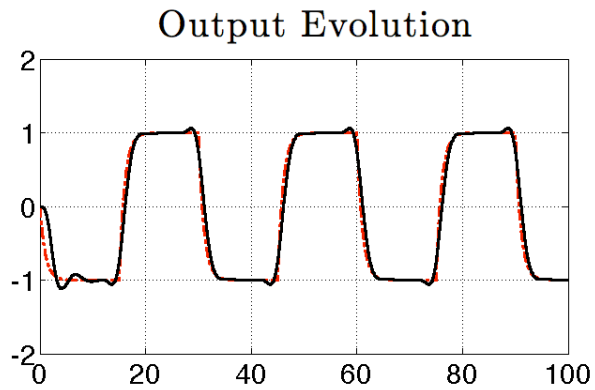
Consider MPC scheme based on the following cost function:

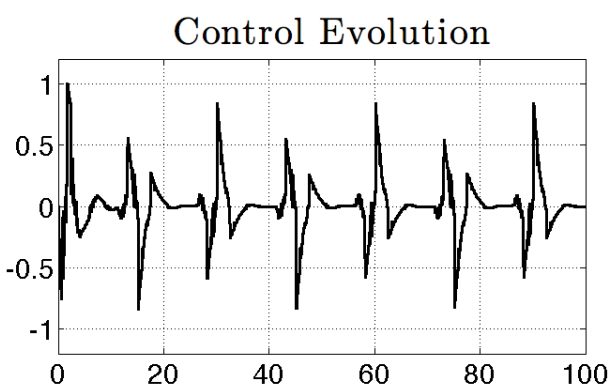
$$J = \sum_{k=1}^{N} \|y(k) - y^{ref}(k)\|_Q^2 + \|u(k)\|_R^2$$

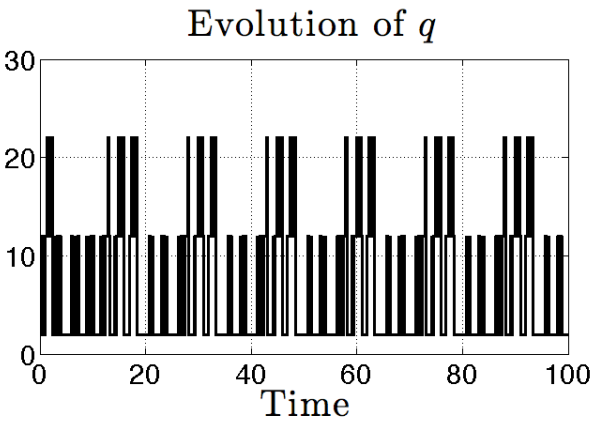
- Standard p.w.c parametrization
- $\tau = 0.02$, $Q = 100$, $R = 1$
- Restarting parameter $s_{max} = 8$
- Gradient step for q, $\delta = 10$
- Prediction horizon $N \in \{100, 200\}$
- $q_{max} = 100$

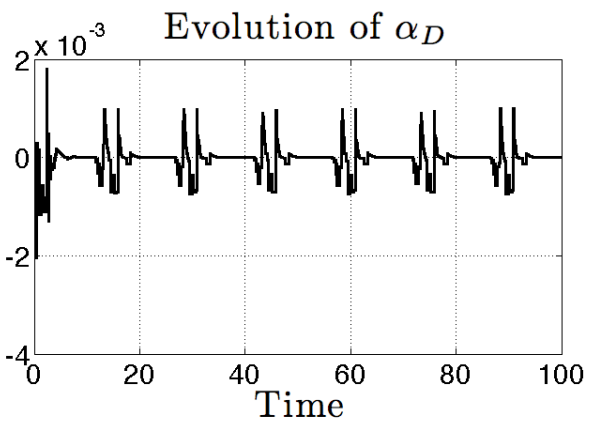
Illustrative example MPC-Controlled system with adaptive control updating period

- Adapt. ON
- $q(0) = 2$
- $N = 200$
- $\delta = 10$



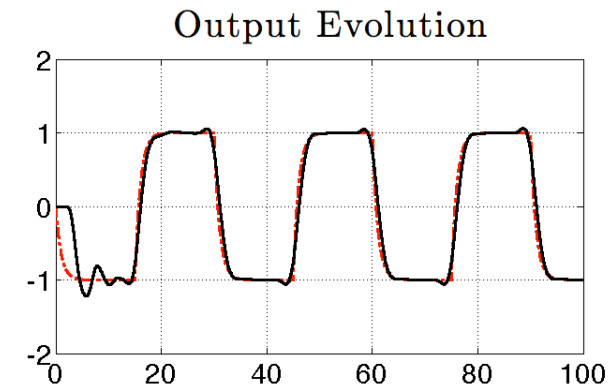


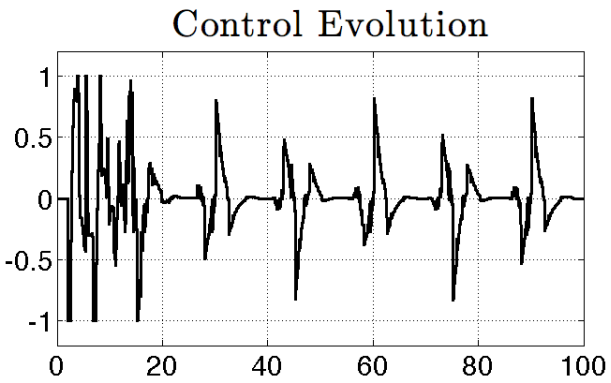


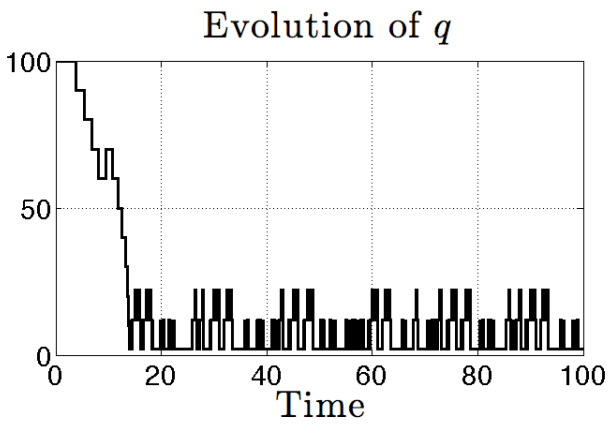


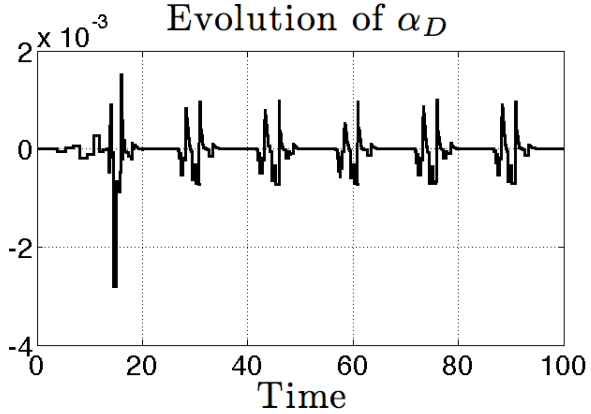
Illustrative example MPC-Controlled system with adaptive control updating period

- Adapt. ON
- $q(0) = 100$
- $N = 200$
- $\delta = 10$



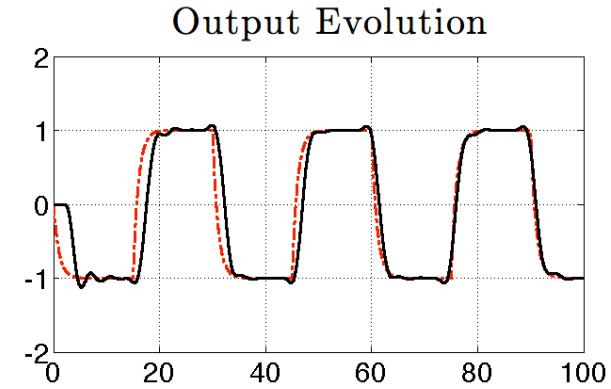


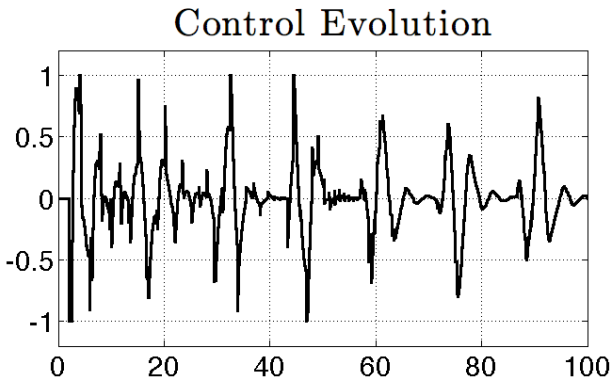


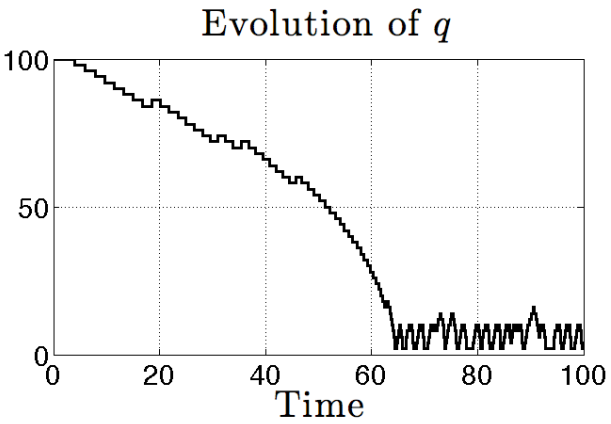


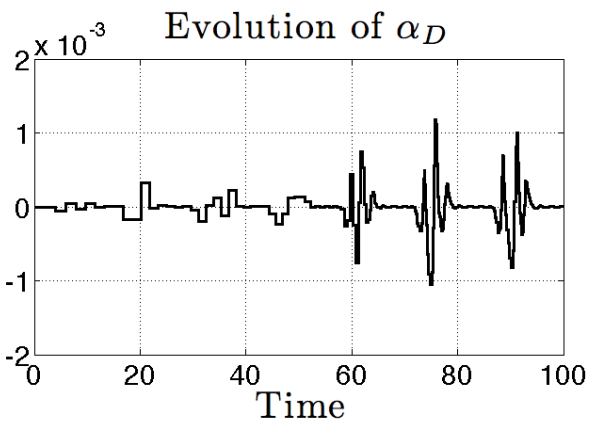
Illustrative example MPC-Controlled system with adaptive control updating period

- Adapt. ON
- $q(0) = 2$
- $N = 200$
- $\delta = 2$



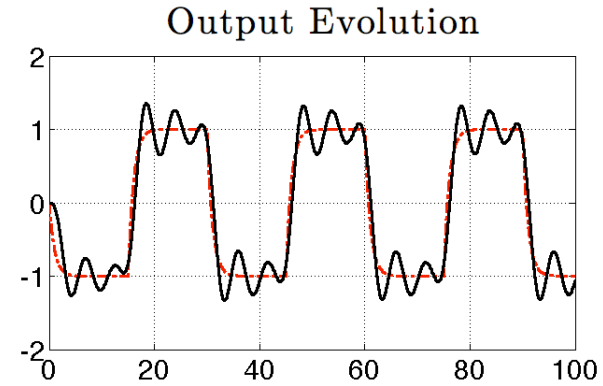


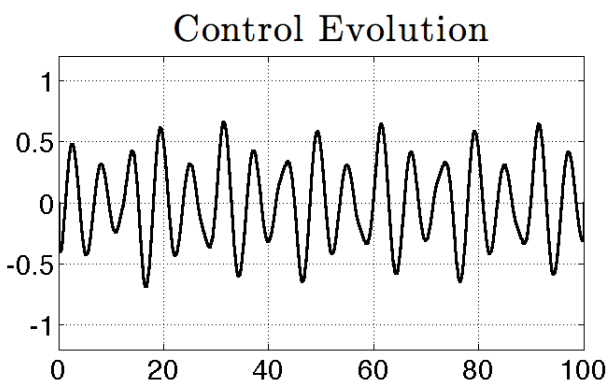


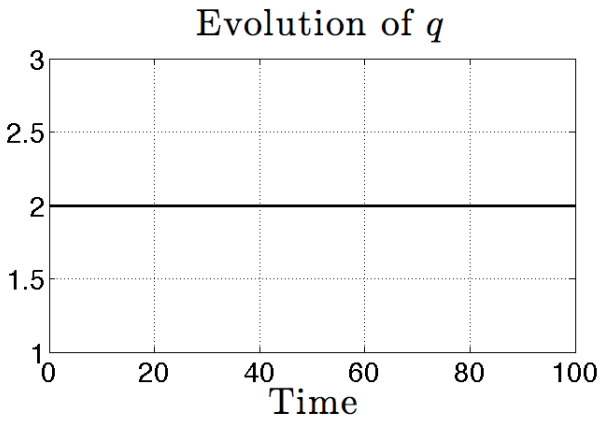


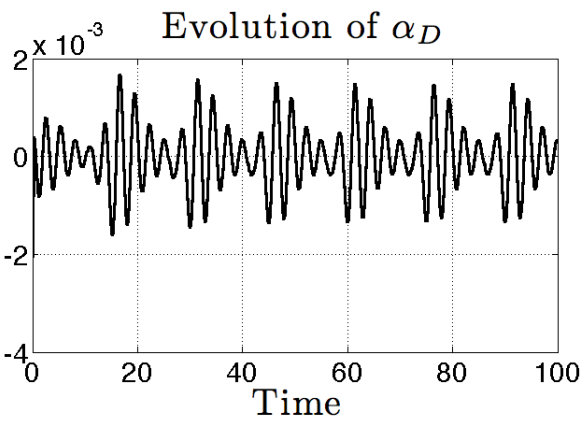
Illustrative example MPC-Controlled system with adaptive control updating period

- Adapt. OFF
- $q \equiv 2$
- $N = 200$
- $\delta = 10$



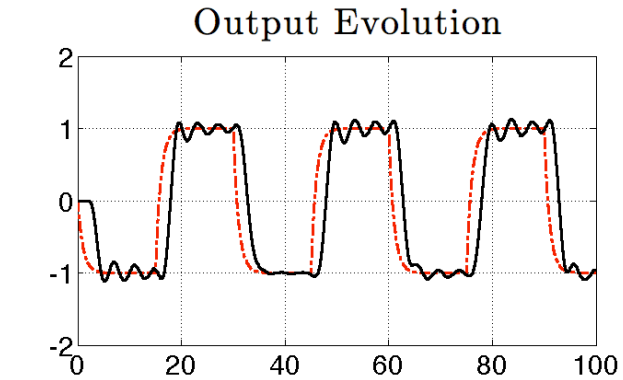


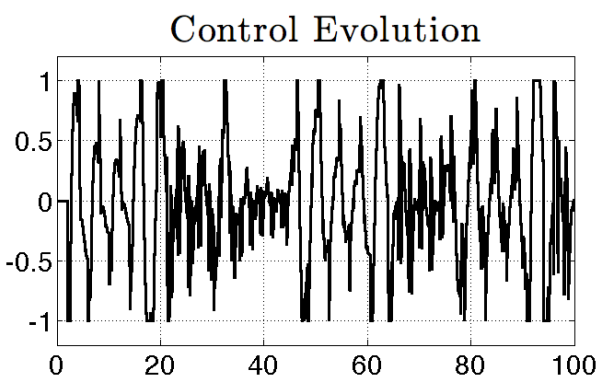


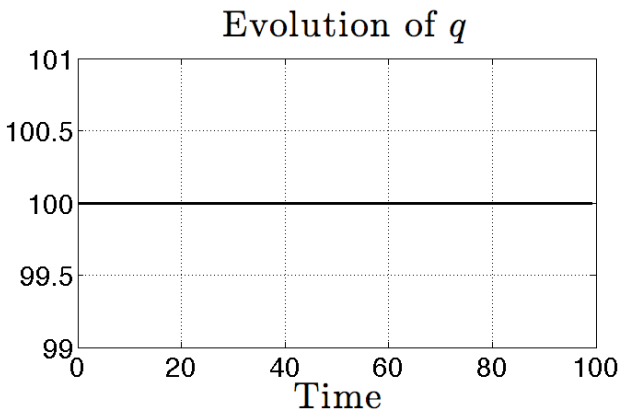


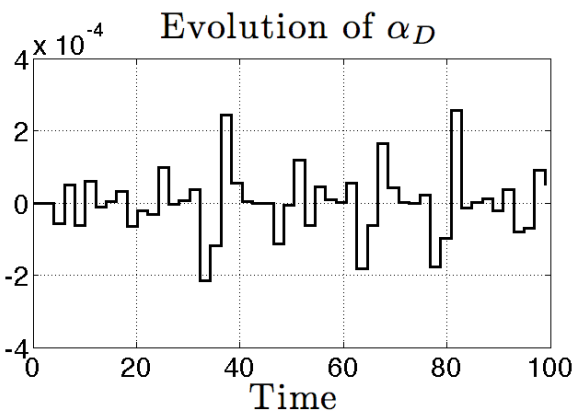
Illustrative example MPC-Controlled system with adaptive control updating period

- Adapt. OFF
- $q \equiv 100$
- $N = 200$
- $\delta = 10$



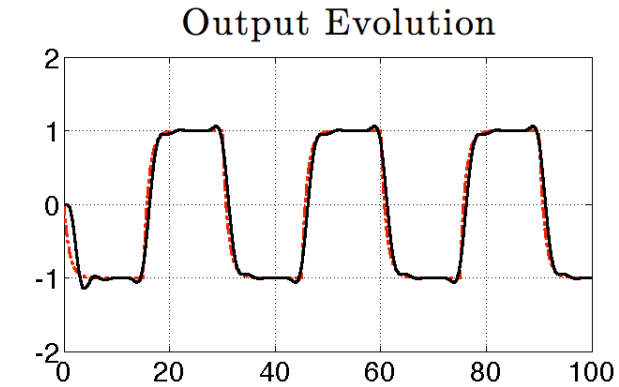


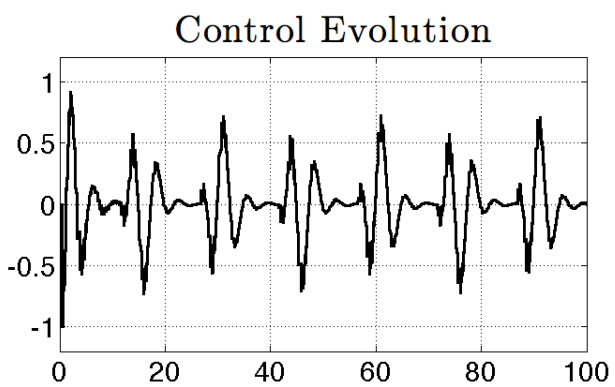


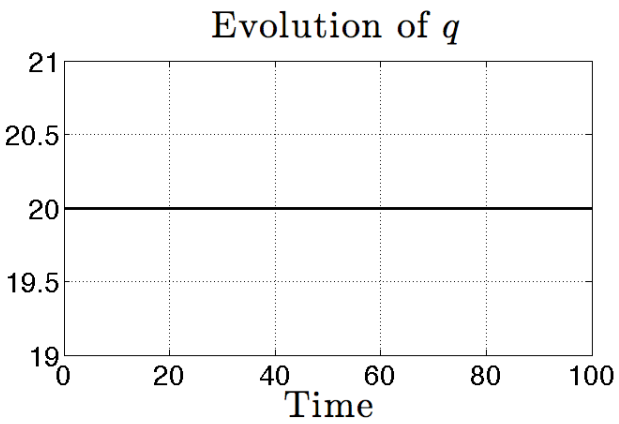


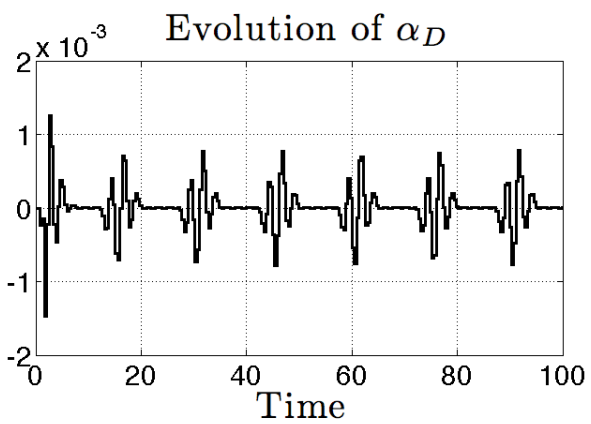
Illustrative example MPC-Controlled system with adaptive control updating period

- Adapt. OFF
- $q \equiv 20$
- $N = 200$
- $\delta = 10$



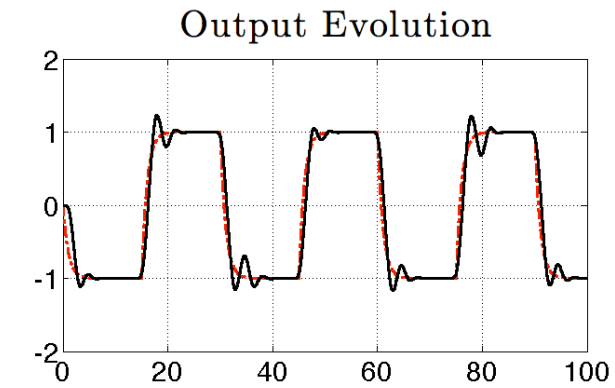


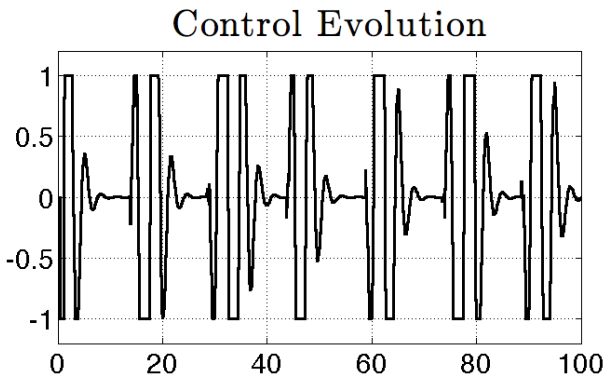


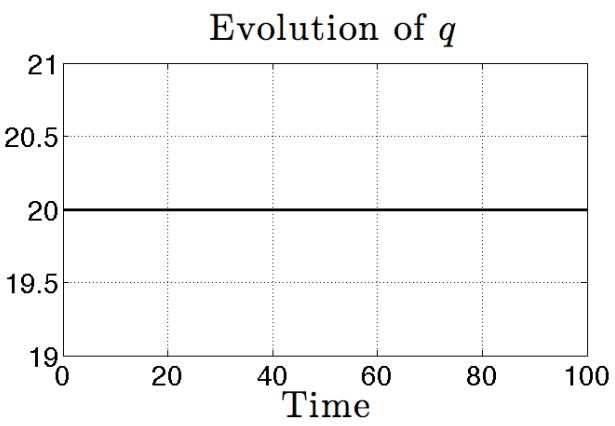


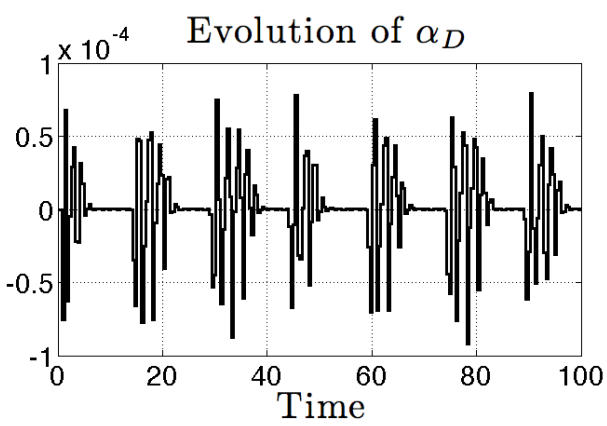
Illustrative example MPC-Controlled system with adaptive control updating period

- Adapt. OFF
- $q \equiv 20$
- $N = 100$
- $\delta = 10$



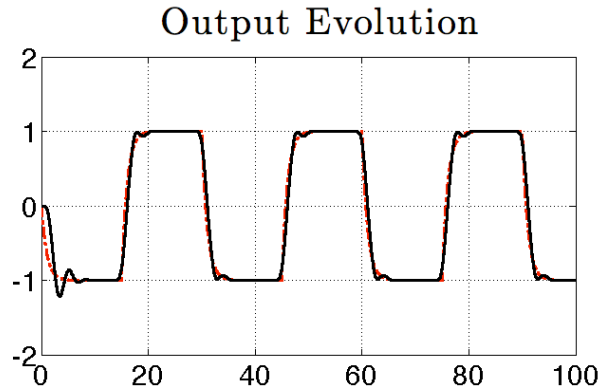


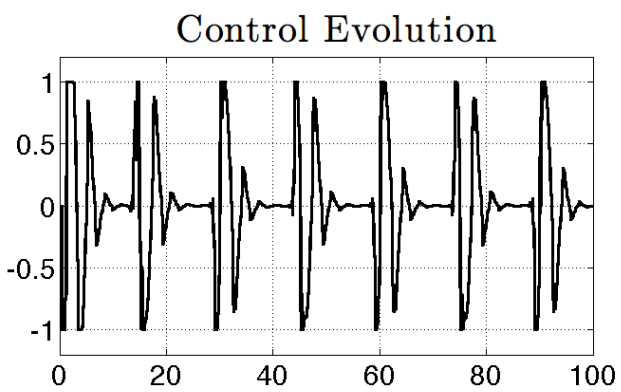


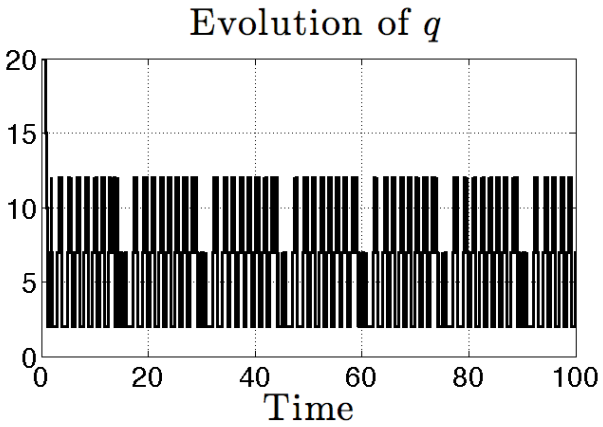


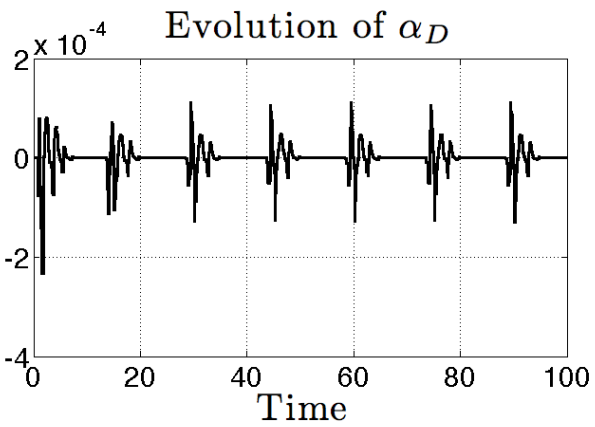
Illustrative example MPC-Controlled system with adaptive control updating period

- Adapt. ON
- $q(0) = 20$
- $N = 100$
- $\delta = 10$









Conclusion & Future Work

- Simple and computationally cheap solution for adapting control updating rate in distributed-in-time NMPC.
- General adaptation layer that can be added to your favorite descent method.
- Two tuning parameters (δ and s_{max})
- Avoid worst case choice of updating period [Allow less control CPU use].

- Dual formulation for Moving-Horizon Observers
- Potential use in a task management under computational resource sharing context.
- Interaction with event-based control and sensing paradigm.

For more details

http://arxiv.1209.4922