A Framework for Real-Time Implementation of Low Dimensional Parameterized NMPC

Mazen Alamir

French National Research Center (CNRS)
Gipsa-lab, Control Systems Department.
University of Grenoble, France

Consider the dynamic system:

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

Ideal NMPC computes an optimal sequence:

$$\hat{\mathbf{u}}(k) = \left(\hat{\mathbf{u}}^{(0)}(k)^T, \dots, \hat{\mathbf{u}}^{(N_p-1)}(k)^T\right)^T \in \mathbb{U} \subset \mathbb{R}^{N_p \cdot m}$$

that minimizes some cost function $J(x(k), \mathbf{u})$ and applies $\hat{\mathbf{u}}^{(0)}(k)$ during the sampling period $[k, k+1]$.

Consider the dynamic system:

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

Ideal NMPC computes an optimal sequence:

$$\hat{\mathbf{u}}(k) = \left(\hat{\mathbf{u}}^{(0)}(k)^T, \dots, \hat{\mathbf{u}}^{(N_p-1)}(k)^T\right)^T \in \mathbb{U} \subset \mathbb{R}^{N_p \cdot m}$$

that minimizes some cost function $J(x(k), \mathbf{u})$ and applies $\hat{\mathbf{u}}^{(0)}(k)$ during the sampling period $[k, k+1]$.

Here, a particular parameterized control is used since:

$$\mathbb{U} := \left\{\mathbf{u} \in \mathbb{R}^{N_p \cdot m} \quad | \quad \mathbf{u}^{(i)} = \mathcal{U}(i, p) \quad \text{where } p \in \mathbb{P}\right\}$$

Consider the dynamic system:

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

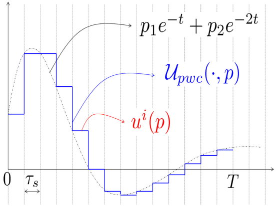
Ideal NMPC computes an optimal sequence:

$$\hat{\mathbf{u}}(k) = \left(\hat{\mathbf{u}}^{(0)}(k)^T, \dots, \hat{\mathbf{u}}^{(N_p-1)}(k)^T\right)^T \in \mathbb{U} \subset \mathbb{R}^{N_p \cdot m}$$

that minimizes some cost function $J(x(k), \mathbf{u})$ and applies $\hat{\mathbf{u}}^{(0)}(k)$ during the sampling period $[k, k+1]$.

Here, a particular parameterized control is used since:

$$\mathbb{U} := \left\{ \mathbf{u} \in \mathbb{R}^{N_p \cdot m} \quad | \quad \mathbf{u}^{(i)} = \mathcal{U}(i, p) \quad \text{where } p \in \mathbb{P} \right\}$$



Consider the dynamic system:

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

Ideal NMPC computes an optimal sequence:

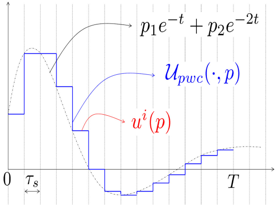
$$\hat{\mathbf{u}}(k) = \left(\hat{\mathbf{u}}^{(0)}(k)^T, \dots, \hat{\mathbf{u}}^{(N_p-1)}(k)^T\right)^T \in \mathbb{U} \subset \mathbb{R}^{N_p \cdot m}$$

that minimizes some cost function $J(x(k), \mathbf{u})$ and applies $\hat{\mathbf{u}}^{(0)}(k)$ during the sampling period $[k, k+1]$.

Here, a particular parameterized control is used since:

$$\mathbb{U} := \left\{\mathbf{u} \in \mathbb{R}^{N_p \cdot m} \quad | \quad \mathbf{u}^{(i)} = \mathcal{U}(i,p) \quad \text{where } p \in \mathbb{P}\right\}$$

- $J(x(k), \mathbf{u}) \longrightarrow J(x(k), p)$



Consider the dynamic system:

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

Ideal NMPC computes an optimal sequence:

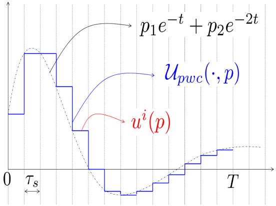
$$\hat{\mathbf{u}}(k) = \left(\hat{\mathbf{u}}^{(0)}(k)^T, \dots, \hat{\mathbf{u}}^{(N_p-1)}(k)^T\right)^T \in \mathbb{U} \subset \mathbb{R}^{N_p \cdot m}$$

that minimizes some cost function $J(x(k), \mathbf{u})$ and applies $\hat{\mathbf{u}}^{(0)}(k)$ during the sampling period $[k, k+1]$.

Here, a particular parameterized control is used since:

$$\mathbb{U} := \left\{\mathbf{u} \in \mathbb{R}^{N_p \cdot m} \quad | \quad \mathbf{u}^{(i)} = \mathcal{U}(i, p) \quad \text{where } p \in \mathbb{P}\right\}$$

- $J(x(k), \mathbf{u}) \longrightarrow J(x(k), p)$
- NMPC looks for an optimal parameter vector $\hat{p}(x(k))$



Consider the dynamic system:

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

Ideal NMPC computes an optimal sequence:

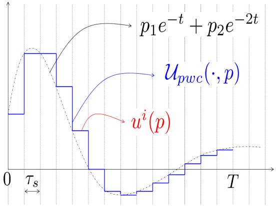
$$\hat{\mathbf{u}}(k) = \left(\hat{\mathbf{u}}^{(0)}(k)^T, \dots, \hat{\mathbf{u}}^{(N_p-1)}(k)^T\right)^T \in \mathbb{U} \subset \mathbb{R}^{N_p \cdot m}$$

that minimizes some cost function $J(x(k), \mathbf{u})$ and applies $\hat{\mathbf{u}}^{(0)}(k)$ during the sampling period $[k, k+1]$.

Here, a particular parameterized control is used since:

$$\mathbb{U} := \left\{\mathbf{u} \in \mathbb{R}^{N_p \cdot m} \quad | \quad \mathbf{u}^{(i)} = \mathcal{U}(i, p) \quad \text{where } p \in \mathbb{P}\right\}$$

- $J(x(k), \mathbf{u}) \longrightarrow J(x(k), p)$
- NMPC looks for an optimal parameter vector $\hat{p}(x(k))$
- The control $\mathcal{U}(0, \hat{p}(x(k))$ is applied over $[k, k+1]$



Consider the dynamic system:

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

Ideal NMPC computes an optimal sequence:

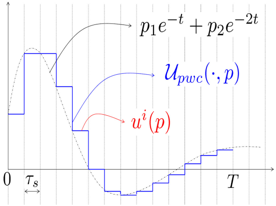
$$\hat{\mathbf{u}}(k) = \left(\hat{\mathbf{u}}^{(0)}(k)^T, \dots, \hat{\mathbf{u}}^{(N_p-1)}(k)^T\right)^T \in \mathbb{U} \subset \mathbb{R}^{N_p \cdot m}$$

that minimizes some cost function $J(x(k), \mathbf{u})$ and applies $\hat{\mathbf{u}}^{(0)}(k)$ during the sampling period $[k, k+1]$.

Here, a particular parameterized control is used since:

$$\mathbb{U} := \left\{\mathbf{u} \in \mathbb{R}^{N_p \cdot m} \quad | \quad \mathbf{u}^{(i)} = \mathcal{U}(i, p) \quad \text{where } p \in \mathbb{P}\right\}$$

- $J(x(k), \mathbf{u}) \longrightarrow J(x(k), p)$
- NMPC looks for an optimal parameter vector $\hat{p}(x(k))$
- The control $\mathcal{U}(0, \hat{p}(x(k))$ is applied over $[k, k+1]$
- A static state feedback



Consider the dynamic system:

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

Ideal NMPC computes an optimal sequence:

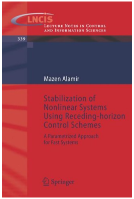
$$\hat{\mathbf{u}}(k) = \left(\hat{\mathbf{u}}^{(0)}(k)^T, \dots, \hat{\mathbf{u}}^{(N_p-1)}(k)^T\right)^T \in \mathbb{U} \subset \mathbb{R}^{N_p \cdot m}$$

that minimizes some cost function $J(x(k), \mathbf{u})$ and applies $\hat{\mathbf{u}}^{(0)}(k)$ during the sampling period $[k, k+1]$.

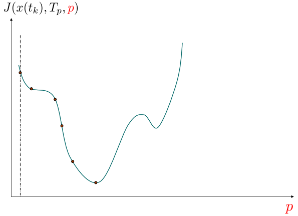
Here, a particular parameterized control is used since:

$$\mathbb{U} := \left\{\mathbf{u} \in \mathbb{R}^{N_p \cdot m} \quad | \quad \mathbf{u}^{(i)} = \mathcal{U}(i, p) \quad \text{where } p \in \mathbb{P}\right\}$$

- $J(x(k), \mathbf{u}) \longrightarrow J(x(k), p)$
- NMPC looks for an optimal parameter vector $\hat{p}(x(k))$
- The control $\mathcal{U}(0, \hat{p}(x(k))$ is applied over $[k, k+1]$
- A static state feedback



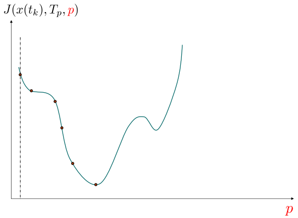
- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$



- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$
\begin{aligned}
x(k+1) &= f\Big(x(k), \mathcal{U}(0, p(k))\Big) \\
p(k+1) &= \mathcal{S}\Big(p(k), x(k), \theta(k)\Big) \\
\theta(k+1) &= \mathcal{D}\Big(p(k), x(k), \theta(k)\Big)
\end{aligned}
$$

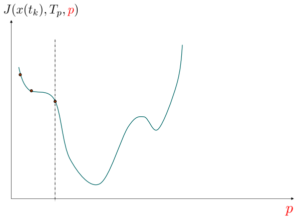
where θ is some internal variable expressing past knowledge.



- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$\begin{aligned} x(k+1) &= f\big(x(k), \mathcal{U}(0, p(k))\big) \\ p(k+1) &= \mathcal{S}\big(p(k), x(k), \theta(k)\big) \\ \theta(k+1) &= \mathcal{D}\big(p(k), x(k), \theta(k)\big) \end{aligned}$$

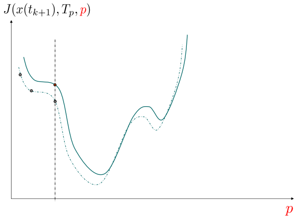
where θ is some internal variable expressing past knowledge.



- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$
\begin{aligned}
x(k+1) &= f\Big(x(k), \mathcal{U}(0, p(k))\Big) \\
p(k+1) &= \mathcal{S}\Big(p(k), x(k), \theta(k)\Big) \\
\theta(k+1) &= \mathcal{D}\Big(p(k), x(k), \theta(k)\Big)
\end{aligned}
$$

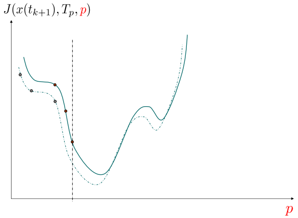
where θ is some internal variable expressing past knowledge.



- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$\begin{aligned} x(k+1) &= f\big(x(k), \mathcal{U}(0, p(k))\big) \\ p(k+1) &= \mathcal{S}\big(p(k), x(k), \theta(k)\big) \\ \theta(k+1) &= \mathcal{D}\big(p(k), x(k), \theta(k)\big) \end{aligned}$$

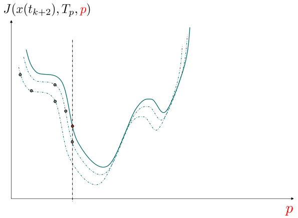
where θ is some internal variable expressing past knowledge.



- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$\begin{aligned} x(k+1) &= f\big(x(k), \mathcal{U}(0, p(k))\big) \\ p(k+1) &= \mathcal{S}\big(p(k), x(k), \theta(k)\big) \\ \theta(k+1) &= \mathcal{D}\big(p(k), x(k), \theta(k)\big) \end{aligned}$$

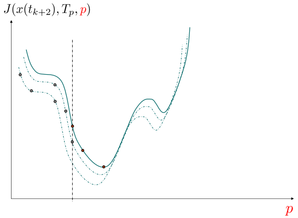
where θ is some internal variable expressing past knowledge.



- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$
\begin{aligned}
x(k+1) &= f\big(x(k), \mathcal{U}(0, p(k))\big) \\
p(k+1) &= \mathcal{S}\big(p(k), x(k), \theta(k)\big) \\
\theta(k+1) &= \mathcal{D}\big(p(k), x(k), \theta(k)\big)
\end{aligned}
$$

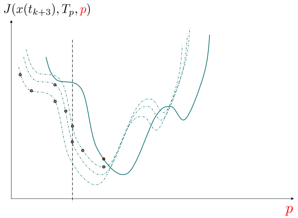
where θ is some internal variable expressing past knowledge.



- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$\begin{aligned} x(k+1) &= f\big(x(k),\mathcal{U}(0,p(k))\big) \\ p(k+1) &= \mathcal{S}\big(p(k),x(k),\theta(k)\big) \\ \theta(k+1) &= \mathcal{D}\big(p(k),x(k),\theta(k)\big) \end{aligned}$$

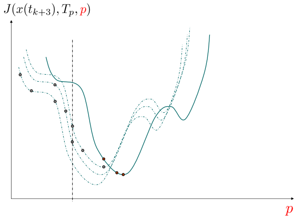
where θ is some internal variable expressing past knowledge.



- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$
\begin{aligned}
x(k+1) &= f\Big(x(k), \mathcal{U}(0, p(k))\Big) \\
p(k+1) &= \mathcal{S}\Big(p(k), x(k), \theta(k)\Big) \\
\theta(k+1) &= \mathcal{D}\Big(p(k), x(k), \theta(k)\Big)
\end{aligned}
$$

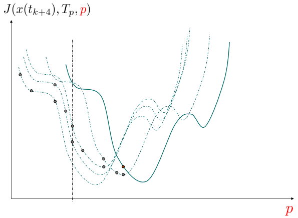
where θ is some internal variable expressing past knowledge.



- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$
\begin{aligned}
x(k+1) &= f\big(x(k), \mathcal{U}(0, p(k))\big) \\
p(k+1) &= \mathcal{S}\big(p(k), x(k), \theta(k)\big) \\
\theta(k+1) &= \mathcal{D}\big(p(k), x(k), \theta(k)\big)
\end{aligned}
$$

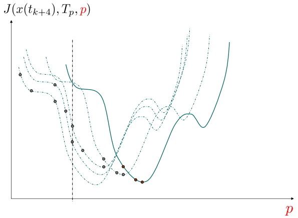
where θ is some internal variable expressing past knowledge.



- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$
\begin{aligned}
x(k+1) &= f\big(x(k), \mathcal{U}(0, p(k))\big) \\
p(k+1) &= \mathcal{S}\big(p(k), x(k), \theta(k)\big) \\
\theta(k+1) &= \mathcal{D}\big(p(k), x(k), \theta(k)\big)
\end{aligned}
$$

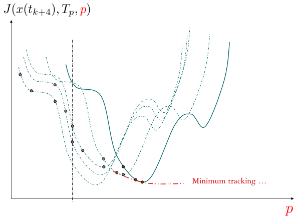
where θ is some internal variable expressing past knowledge.



- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$\begin{aligned} x(k+1) &= f\Big(x(k), \mathcal{U}(0, p(k))\Big) \\ p(k+1) &= \mathcal{S}\Big(p(k), x(k), \theta(k)\Big) \\ \theta(k+1) &= \mathcal{D}\Big(p(k), x(k), \theta(k)\Big) \end{aligned}$$

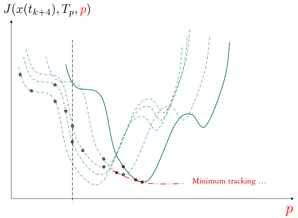
where θ is some internal variable expressing past knowledge.



- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$
\begin{aligned}
x(k+1) &= f\big(x(k), \mathcal{U}(0, p(k))\big) \\
p(k+1) &= \mathcal{S}\big(p(k), x(k), \theta(k)\big) \\
\theta(k+1) &= \mathcal{D}\big(p(k), x(k), \theta(k)\big)
\end{aligned}
$$

where θ is some internal variable expressing past knowledge.

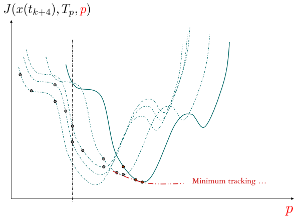


M. Alamir/Control Eng. Practice, 9 (2001)

- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$\begin{aligned} x(k+1) &= f\big(x(k), \mathcal{U}(0, p(k))\big) \\ p(k+1) &= \mathcal{S}\big(p(k), x(k), \theta(k)\big) \\ \theta(k+1) &= \mathcal{D}\big(p(k), x(k), \theta(k)\big) \end{aligned}$$

where θ is some internal variable expressing past knowledge.



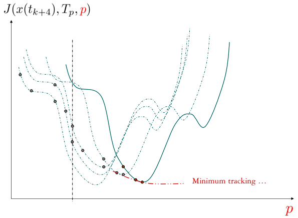
M. Alamir/Control Eng. Practice, 9 (2001)

Newton method is used instead of dichotomy. In the "*distributed*" Newton method, the Newton method iterations are distributed over the successive sampling periods. One step is performed at each sampling period.

- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$\begin{aligned} x(k+1) &= f\big(x(k), \mathcal{U}(0, p(k))\big) \\ p(k+1) &= \mathcal{S}\big(p(k), x(k), \theta(k)\big) \\ \theta(k+1) &= \mathcal{D}\big(p(k), x(k), \theta(k)\big) \end{aligned}$$

where θ is some internal variable expressing past knowledge.



M. Alamir/Control Eng. Practice, 9 (2001)

T. Ohtsuka/Automatica, 40 (2004)

M. Diehl et al./SIAM Contr. Opt.,43 (2005)

D. DeHaan et al./IEEE TAC, 52 (2007)

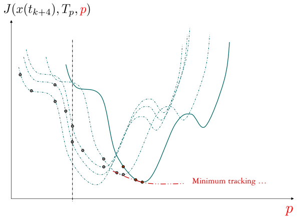
V. M. Zavala et al./Int. J. Robust. Nonlinear Contr., 18 (2008)

...

- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$\begin{aligned} x(k+1) &= f\Big(x(k), \mathcal{U}(0, p(k))\Big) \\ p(k+1) &= \mathcal{S}\Big(p(k), x(k), \theta(k)\Big) \\ \theta(k+1) &= \mathcal{D}\Big(p(k), x(k), \theta(k)\Big) \end{aligned}$$

where θ is some internal variable expressing past knowledge.



M. Alamir/Control Eng. Practice, 9 (2001)

T. Ohtsuka/Automatica, 40 (2004)

M. Diehl et al./SIAM Contr. Opt.,43 (2005)

D. DeHaan et al./IEEE TAC, 52 (2007)

V. M. Zavala et al./Int. J. Robust. Nonlinear Contr., 18 (2008)

...

Existing non parameterized formulations:

1. High dimensional decision variable
2. Need for highly involved softwares
3. High memory storage, etc.

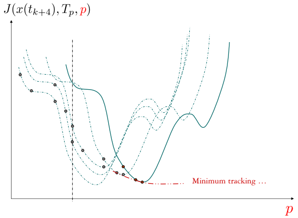
- The ideal scheme assumes instantaneous computation of $\hat{p}(x(k))$
- Unfortunately, only a finite number of iterations q can be performed according to:

$$\begin{aligned} x(k+1) &= f\big(x(k),\mathcal{U}(0,p(k))\big) \\ p(k+1) &= \mathcal{S}\big(p(k),x(k),\theta(k)\big) \\ \theta(k+1) &= \mathcal{D}\big(p(k),x(k),\theta(k)\big) \end{aligned}$$

where θ is some internal variable expressing past knowledge.

Existing non parameterized formulations:

1. High dimensional decision variable
2. Need for highly involved softwares
3. High memory storage, etc.



M. Alamir/Control Eng. Practice, 9 (2001)

This contribution

Propose a specific instantiation of θ, $\mathcal{S}$ and $\mathcal{D}$ that is compatible with real-time implementation of parameterized NMPC

It is assumed that the problem constraints are handled through:

1. The control parametrization
2. The inclusion of high penalty term (interior point method)

Example:

$$\mathcal{U}(i,p) := Sat_{[u_{min},u_{max}]}\Big(\mathcal{U}^{unc}(i,p)\Big)$$

$$J(x,p) := J^{unc}(x,p) + \rho \cdot \phi(x,p)$$

→ There is no constraints-related feasibility issue

→ The constrained underlying optimization problem is given by:

$$\min_{p\in\mathbb{P}} \quad J(x,p) \quad \text{where } \mathbb{P} \text{ is a hyper cube}$$

→ It is assumed that the formulation is stabilizing in the ideal case

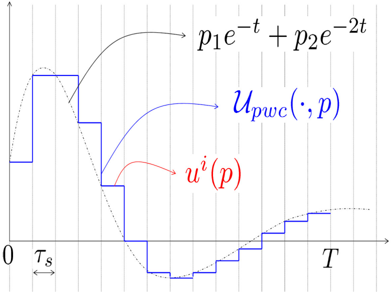
→ This paper is not concerned with a new stabilizing formulation

→ It concerns real-time implementability

Assumption 1

For all $p \in \mathbb{P}$, there is a parameter value $p^+ \in \mathbb{P}$ such that:

$$\forall i \in \mathbb{N} \qquad \mathcal{U}(i+1, p) = \mathcal{U}(i, p^+)$$



$$p^+ = \begin{pmatrix} e^{-\tau_s} & 0 \\ 0 & e^{-2\tau_s} \end{pmatrix} p$$

The translated version of a control trajectory belongs to the parameterized set

[See Alamir, Springer, 2006]

Assumption 1

For all $p \in \mathbb{P}$, there is a parameter value $p^+ \in \mathbb{P}$ such that:

$$\forall i \in \mathbb{N} \qquad \mathcal{U}(i+1, p) = \mathcal{U}(i, p^+)$$

Let us define by induction:

$$p^{0+} = p \quad ; \quad p^{i+} = \left[p^{(i-1)+}\right]^+$$

$$p^+ = \begin{pmatrix} e^{-\tau_s} & 0 \\ 0 & e^{-2\tau_s} \end{pmatrix} p$$

The translated version of a control trajectory belongs to the parameterized set

[See Alamir, Springer, 2006]

Assumption 1

For all $p \in \mathbb{P}$, there is a parameter value $p^+ \in \mathbb{P}$ such that:

$$\forall i \in \mathbb{N} \qquad \mathcal{U}(i+1, p) = \mathcal{U}(i, p^+)$$

Let us define by induction:

$$p^{0+} = p \quad ; \quad p^{i+} = \left[p^{(i-1)+}\right]^+$$

and using

$$x^+ = f\big(x, \mathcal{U}(0,p)\big)$$

$$p^+ = \begin{pmatrix} e^{-\tau_s} & 0 \\ 0 & e^{-2\tau_s} \end{pmatrix} p$$

The translated version of a control trajectory belongs to the parameterized set

[See Alamir, Springer, 2006]

Assumption 1

For all $p \in \mathbb{P}$, there is a parameter value $p^+ \in \mathbb{P}$ such that:

$$\forall i \in \mathbb{N} \qquad \mathcal{U}(i+1,p) = \mathcal{U}(i,p^+)$$

Let us define by induction:

$$p^{0+} = p \quad ; \quad p^{i+} = \left[p^{(i-1)+}\right]^+$$

and using

$$x^+ = f\big(x, \mathcal{U}(0,p)\big)$$

one can define the induction rule:

$$z^{0+} = z \quad ; \quad z^{i+} = \left[z^{(i-1)+}\right]^+$$

with $z := (x,p)$

$$p^+ = \begin{pmatrix} e^{-\tau_s} & 0 \\ 0 & e^{-2\tau_s} \end{pmatrix} p$$

The translated version of a control trajectory belongs to the parameterized set

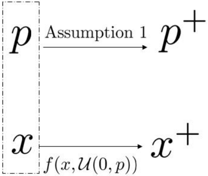
[See Alamir, Springer, 2006]

p

x

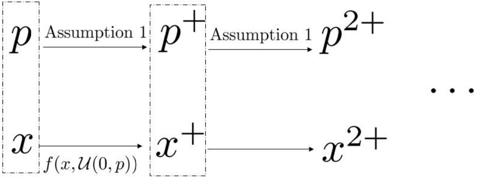
$$p \xrightarrow{\text{Assumption 1}} p^+$$

$$x$$


p
Assumption 1
p^+
x
$f(x, \mathcal{U}(0,p))$
x^+

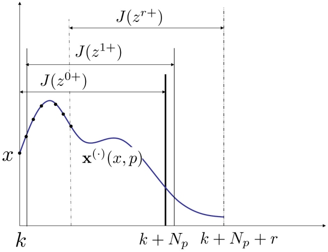
$$p \xrightarrow{\text{Assumption 1}} p^{+} \xrightarrow{\text{Assumption 1}} p^{2+}$$

$$x \xrightarrow[f(x,\mathcal{U}(0,p))]{} x^{+}$$


p
Assumption 1
p^+
Assumption 1
p^{2+}
x
$f(x, \mathcal{U}(0,p))$
x^+
x^{2+}
$\cdots$

$$
\begin{tikzcd}[column sep=huge, row sep=large]
p \arrow[r, "\text{Assumption 1}"] & p^{+} \arrow[r, "\text{Assumption 1}"] & p^{2+} & \\
 & & & \cdots \\
x \arrow[r, "{f(x,\mathcal{U}(0,p))}"'] & x^{+} \arrow[r] & x^{2+} & \\
z \arrow[r] & z^{+} \arrow[r] & z^{2+} &
\end{tikzcd}
$$

$$
\begin{tikzcd}
p \arrow[r, "\text{Assumption 1}"] & p^{+} \arrow[r, "\text{Assumption 1}"] & p^{2+} & \cdots \\
x \arrow[r, "f(x,\mathcal{U}(0,p))"'] & x^{+} \arrow[r] & x^{2+} & \\
z \arrow[r] & z^{+} \arrow[r] & z^{2+} &
\end{tikzcd}
$$



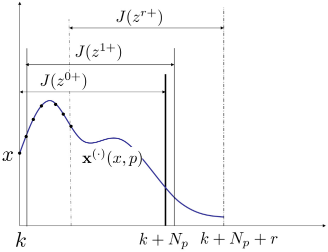
- Recall that $J(x,p) = J(z)$
- We have the following fact:

Fact 1

Given a pair $z = (p,x)$, the computation of the corresponding system's trajectory over a horizon of length $(N_p + r)$ provides $(r+1)$ realizations of the function $J(z)$, namely $\{J(z^{i+})\}_{i=0}^{r}$. $\triangle$

For instance,

If $N_p = 20$ and $r = 9$, one obtains 10 realizations at the price of 50% extra-simulations.



- Recall that $J(x,p) = J(z)$
- We have the following fact:

Fact 1

Given a pair $z = (p,x)$, the computation of the corresponding system's trajectory over a horizon of length $(N_p + r)$ provides $(r+1)$ realizations of the function $J(z)$, namely $\{J(z^{i+})\}_{i=0}^{r}$. $\triangle$

Use the successive data obtained during the system life-time to dynamically identify a quadratic approximation of $J(\cdot)$:

$$\hat{J}_k(z) = J_k^* + L_k^T z + z^T Q_k z =: \Phi(z) \cdot q(k)$$

where:

$$q(k) \in \mathbb{R}^{n_q} \quad ; \quad n_q := (1+n_z)(1+n_z/2) \quad ; \quad n_z = n + n_p$$

Use the successive data obtained during the system life-time to dynamically identify a quadratic approximation of $J(\cdot)$:

$$\hat{J}_k(z) = J_k^* + L_k^T z + z^T Q_k z =: \Phi(z) \cdot q(k)$$

where:

$$q(k) \in \mathbb{R}^{n_q} \quad ; \quad n_q := (1 + n_z)(1 + n_z/2) \quad ; \quad n_z = n + n_p$$

For instance,

n	n_p	n_q
2	2	15
3	2	21
5	2	36
5	3	45
8	4	91

Use the successive data obtained during the system life-time to dynamically identify a quadratic approximation of $J(\cdot)$:

$$\hat{J}_k(z) = J_k^* + L_k^T z + z^T Q_k z =: \Phi(z) \cdot q(k)$$

where:

$$q(k) \in \mathbb{R}^{n_q} \quad ; \quad n_q := (1+n_z)(1+n_z/2) \quad ; \quad n_z = n + n_p$$

For instance,

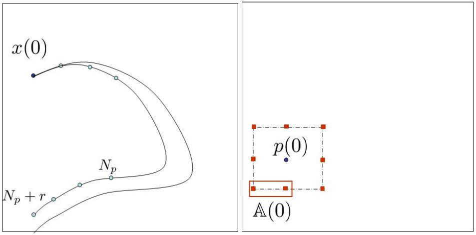
n	n_p	n_q
2	2	15
3	2	21
5	2	36
5	3	45
8	4	91

Given $x(k)$, $p(k-1)$ and $q(k)$, the updating law for p is obtained by solving:

$$p(k) := arg \min_{p \in \mathbb{P} \cap \left\{p(k-1) + \mathbb{B}(\rho_k)\right\}} \Big[\Phi(x(k), p)\Big] \cdot q(k)$$

which is a QP problem of dimension n_p with $2n_p$ constraints.

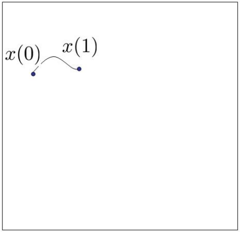
ρ_k is a trust region coefficient which is updated classically (increased after a success and decreased after a failure)

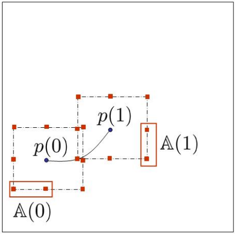


- Let $x(0)$, $p(0)$ and $q(0)$ be given, apply $p(0)$ during $[0, 1]$,
- For each elements in $\mathbb{A}(0)$, use Fact1 to generate the $(r+1) \cdot card(\mathbb{A}(0))$ identification data:

$$\begin{aligned} D(0) &:= \Big\{\mathbb{Z}(0), \{J(z)\}_{z\in\mathbb{Z}(0)}\Big\} \\ \mathbb{Z}(0) &:= \Big\{(\hat{x}(1), p)^{i+} \mid (p, i) \in \mathbb{A}(0) \times \{0, \dots, r\}\Big\} \end{aligned}$$

This defines a linear system $[A_0]q = B_0 \in \mathbb{R}^{(r+1)\cdot card(\mathbb{A}(0))} \quad \rightarrow q(1)$

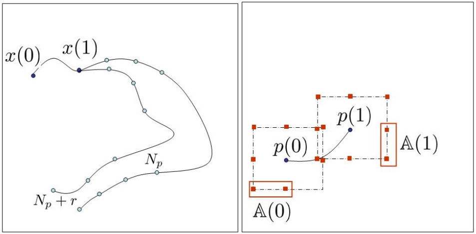




- Given $x(1)$ and $q(1)$, $p(1)$ is obtained by a small QP solution
- For each elements in $\mathbb{A}(1)$, use Fact1 to generate the $(r+1)\cdot card(\mathbb{A}(1))$ identification data:

$$\begin{aligned} D(1) &:= \Big\{\mathbb{Z}(1), \{J(z)\}_{z\in\mathbb{Z}(1)}\Big\} \\ \mathbb{Z}(1) &:= \Big\{(\hat{x}(2), p)^{i+} \mid (p,i) \in \mathbb{A}(1) \times \{0,\dots,r\}\Big\} \end{aligned}$$

This defines a linear system $[A_1]q = B_1 \in \mathbb{R}^{(r+1)\cdot card(\mathbb{A}(1))} \quad \rightarrow q(2)$

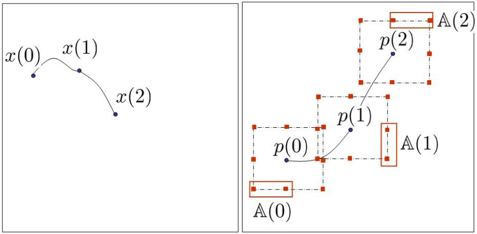


- Given $x(1)$ and $q(1)$, $p(1)$ is obtained by a small QP solution
- For each elements in $\mathbb{A}(1)$, use Fact1 to generate the $(r+1)\cdot card(\mathbb{A}(1))$ identification data:

$$D(1) := \Big\{\mathbb{Z}(1), \{J(z)\}_{z\in\mathbb{Z}(1)}\Big\}$$

$$\mathbb{Z}(1) := \Big\{(\hat{x}(2), p)^{i+} \mid (p,i) \in \mathbb{A}(1)\times\{0,\dots,r\}\Big\}$$

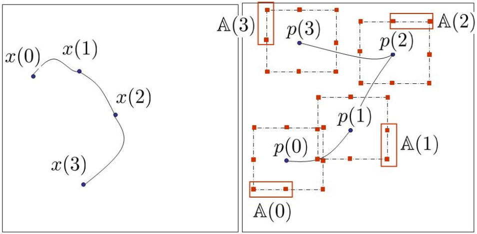
This defines a linear system $[A_1]q = B_1 \in \mathbb{R}^{(r+1)\cdot card(\mathbb{A}(1))} \quad \rightarrow q(2)$



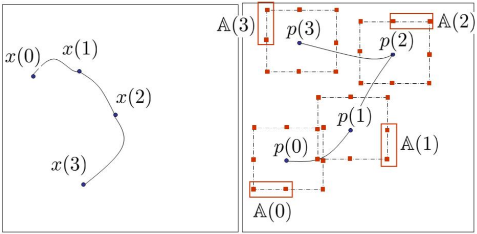
- Given $x(2)$ and $q(2)$, $p(2)$ is obtained by a small QP solution
- For each elements in $\mathbb{A}(2)$, use Fact1 to generate the $(r+1)\cdot card(\mathbb{A}(2))$ identification data:

$$\begin{aligned} D(2) &:= \Big\{\mathbb{Z}(2), \big\{J(z)\big\}_{z\in\mathbb{Z}(2)}\Big\} \\ \mathbb{Z}(2) &:= \Big\{(\hat{x}(3),p)^{i+} \mid (p,i)\in \mathbb{A}(2)\times\{0,\dots,r\}\Big\} \end{aligned}$$

This defines a linear system $[A_2]q = B_2 \in \mathbb{R}^{(r+1)\cdot card(\mathbb{A}(2))} \quad \rightarrow q(3)$



$x(0)$
$x(1)$
$x(2)$
$x(3)$
$\mathbb{A}(3)$
$p(3)$
$p(2)$
$\mathbb{A}(2)$
$p(1)$
$\mathbb{A}(1)$
$p(0)$
$\mathbb{A}(0)$

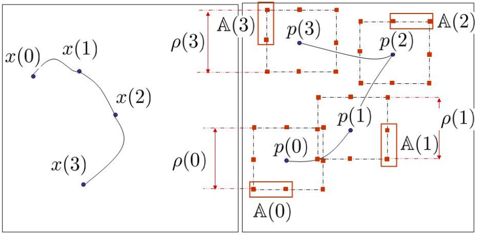


As a matter of fact,

Successive blocs of linear subsystems

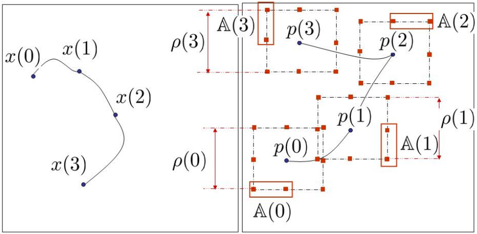
$$A_k q = b_k$$

arrive at each sampling period that can be used to obtain a recursive least squares estimation of $q(k)$ with some forgetting factor σ.



Moreover,

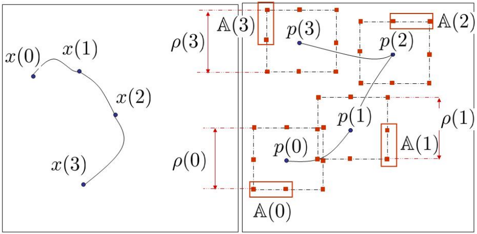
the updated trust region diameter $\rho(k)$ is used to define the hyper cubes of of increments on p



Note that the following choice is made

$$card(\mathbb{A}(k)) = n_{max}$$

where n_{max} is the maximum number of system integrations that can be performed within a single sampling period.



The internal variable $\theta(k)$ invoked in the general setting is nothing but:

$$\theta(k) := \Big\{D(k), \dots, D(k - N_m + 1)\Big\}$$

in which

$$D(k) := \left\{\mathbb{Z}(k), \Big\{J(z)\Big\}_{z \in \mathbb{Z}(k)}\right\}$$

During the sampling period $[k, k+1]$,

1. Solve the n_p-dimensional QP with $2n_p$ saturation constraints:

$$p(k) := arg \min_{p \in \mathbb{P} \cap \left\{ p(k-1) + \mathbb{B}(\rho_k) \right\}} \left[\Phi(x(k), p) \right] \cdot q(k)$$

2. Simulate the system $n_{max} + 1$ times over a prediction horizon of length $N_p + r$ to obtain the new bloc of linear subsystem $[A_k]q = B_k$

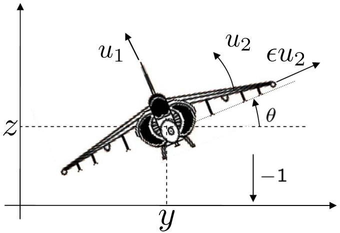
3. Update $q(k)$ using a recursive least squares step
   [$\rightarrow$ solve a linear system of dimension $n_{ls} := n_{max}(r+1) + n_q$]

In the next PVTOL example:

$$n_p = 6 \quad , \quad n_{max} \in \{1, 2, 3\} \quad , \quad r \in \{0, 5\} \quad , \quad n_q = 66$$

System Model

$$\begin{aligned}\ddot{y} &= -u_1 \sin\theta + \epsilon u_2 \cos\theta \\ \ddot{z} &= u_1 \cos\theta + \epsilon u_2 \sin\theta - 1 \\ \ddot{\theta} &= u_2\end{aligned}$$



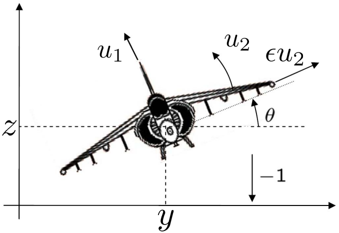
Constraints

$$u_1 \in [0, u_1^{max} = 1 + \gamma]$$
$$u_2 \in [-u_2^{max}, +u_2^{max}]$$

Objective Stabilize $x^d = (y^d, z^d, 0, 0, 0, 0)^T$

System Model

$$\begin{aligned}\ddot{y} &= -u_1 \sin\theta + \epsilon u_2 \cos\theta \\ \ddot{z} &= u_1 \cos\theta + \epsilon u_2 \sin\theta - 1 \\ \ddot{\theta} &= u_2\end{aligned}$$



Constraints

$$u_1 \in [0, u_1^{max} = 1 + \gamma]$$
$$u_2 \in [-u_2^{max}, +u_2^{max}]$$

Objective Stabilize $x^d = (y^d, z^d, 0, 0, 0, 0)^T$

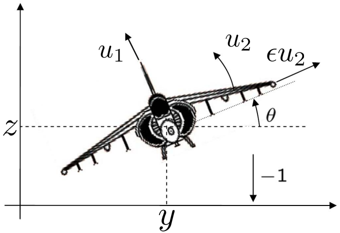
Control parametrization

$$\mathcal{U}(i,p,x) = Sat_{[0,u_1^{max}]\times[-u_2^{max},+u_2^{max}]}\Big[\mathcal{U}^{unc}(i,p,x)\Big]$$

$$\mathcal{U}^{unc}(i,p,x) := \begin{pmatrix} p_1 e^{-\lambda(i\tau_s)} + p_2 e^{-3\lambda(i\tau_s)} + p_3 e^{-6\lambda(i\tau_s)} \\ p_4 e^{-\lambda(i\tau_s)} + p_5 e^{-3\lambda(i\tau_s)} + p_6 e^{-6\lambda(i\tau_s)} \end{pmatrix}$$

System Model

$$\begin{aligned}\ddot{y} &= -u_1 \sin\theta + \epsilon u_2 \cos\theta \\ \ddot{z} &= u_1 \cos\theta + \epsilon u_2 \sin\theta - 1 \\ \ddot{\theta} &= u_2\end{aligned}$$



Constraints

$$u_1 \in [0, u_1^{max} = 1 + \gamma]$$
$$u_2 \in [-u_2^{max}, +u_2^{max}]$$

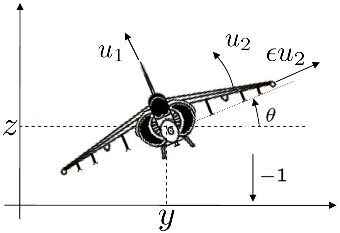
Objective Stabilize $x^d = (y^d, z^d, 0, 0, 0, 0)^T$

Translatability

$$p^+ = diag\big(e^{-\lambda\tau_s}, e^{-3\lambda\tau_s}, e^{-6\lambda\tau_s}, e^{-\lambda\tau_s}, e^{-3\lambda\tau_s}, e^{-6\lambda\tau_s}\big) \cdot p$$

System Model

$$\begin{aligned}\ddot{y} &= -u_1 \sin\theta + \epsilon u_2 \cos\theta \\ \ddot{z} &= u_1 \cos\theta + \epsilon u_2 \sin\theta - 1 \\ \ddot{\theta} &= u_2\end{aligned}$$



Constraints

$$u_1 \in [0, u_1^{max} = 1 + \gamma]$$
$$u_2 \in [-u_2^{max}, +u_2^{max}]$$

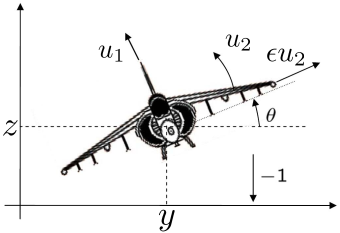
Objective Stabilize $x^d = (y^d, z^d, 0, 0, 0, 0)^T$

Saturation on p

$$\mathbb{P} := [-u_1^{max}, u_1^{max}]^3 \times [-u_2^{max}, u_2^{max}]^3$$

System Model

$$\begin{aligned}\ddot{y} &= -u_1 \sin\theta + \epsilon u_2 \cos\theta \\ \ddot{z} &= u_1 \cos\theta + \epsilon u_2 \sin\theta - 1 \\ \ddot{\theta} &= u_2\end{aligned}$$



Constraints

$$u_1 \in [0, u_1^{max} = 1 + \gamma]$$
$$u_2 \in [-u_2^{max}, +u_2^{max}]$$

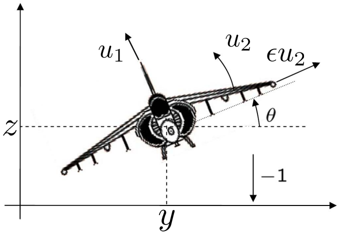
Objective Stabilize $x^d = (y^d, z^d, 0, 0, 0, 0)^T$

Cost Function

$$J(x,p) := \sum_{i=0}^{N_p} \left[\|\mathbf{x}^{(i)}(x,p) - x^d\|_{Q_x}^2 + \|\mathcal{U}(i,p,x) - u^d\|_{R_u}^2 \right]$$

System Model

$$\begin{aligned} \ddot{y} &= -u_1 \sin\theta + \epsilon u_2 \cos\theta \\ \ddot{z} &= u_1 \cos\theta + \epsilon u_2 \sin\theta - 1 \\ \ddot{\theta} &= u_2 \end{aligned}$$



Constraints

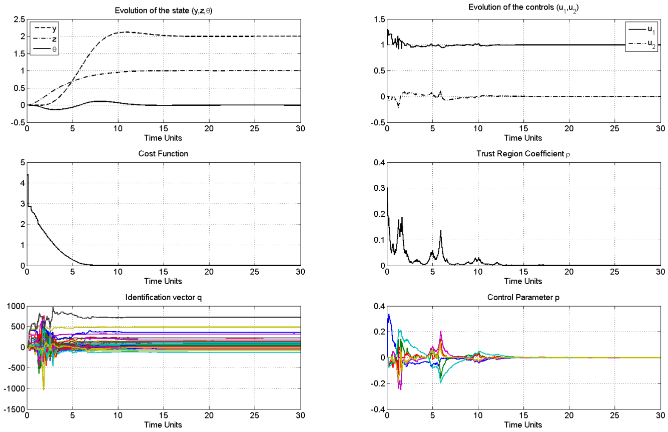
$$\begin{aligned} &u_1 \in [0, u_1^{max} = 1+\gamma] \\ &u_2 \in [-u_2^{max}, +u_2^{max}] \end{aligned}$$

Objective Stabilize $x^d = (y^d, z^d, 0, 0, 0, 0)^T$

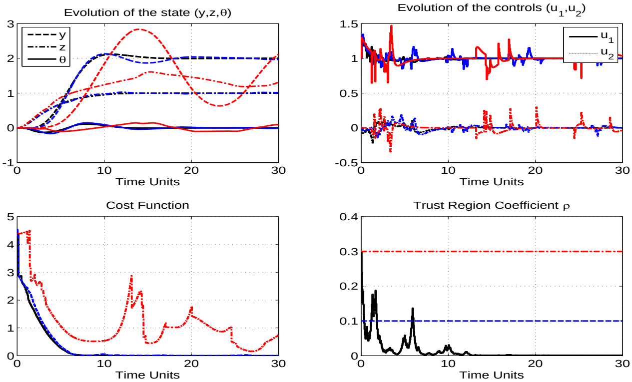
The $\mathbb{A}(k)$ subsets

$$\cup_{k=0}^{N_m} \mathbb{A}(k) = \{e_i\}_{i=1}^{6}$$

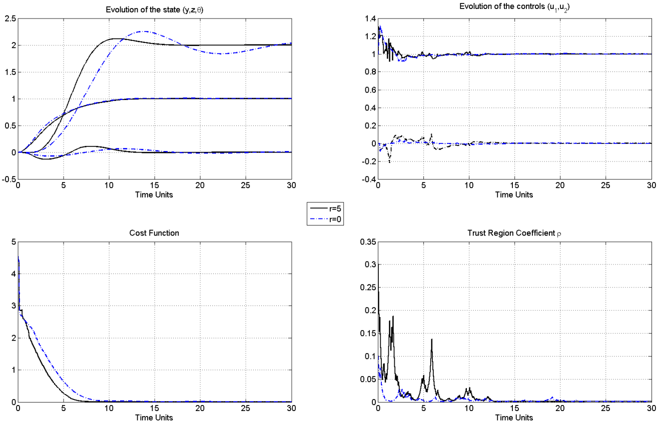
with $N_m \times n_{max} = 6$



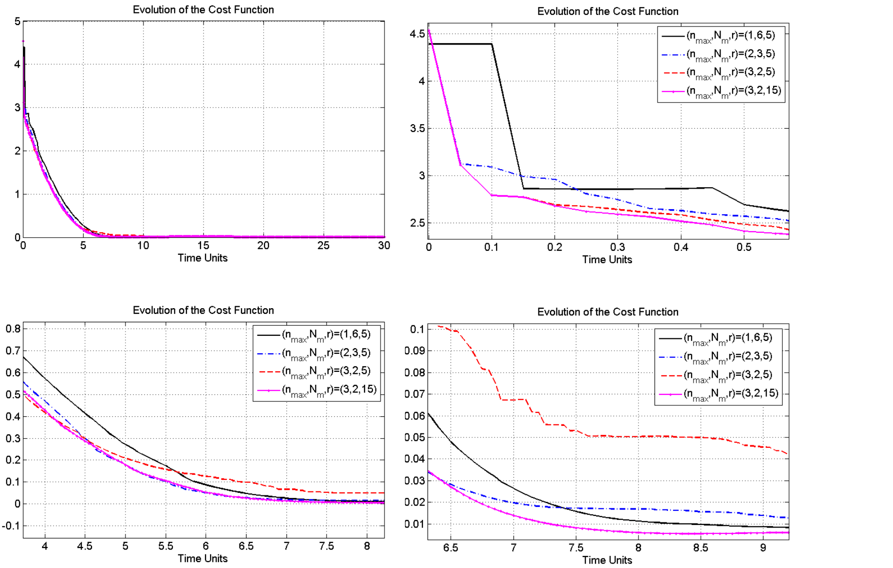
Reference scenario: $N_p = 150$, $n_{max} = 1$, $N_m = 6$ and $r = 5$.



Relevance of the Trust region mechanism: $\rho = 0.1$, $\rho = 0.3$



Relevance of r: $r = 5$ vs $r = 0$. Recall that $N_p = 150$ meaning that the use of $r = 5$ represents 3.3% of extra computation.


Evolution of the Cost Function
Time Units
Evolution of the Cost Function
(n_max,N_m,r)=(1,6,5)
(n_max,N_m,r)=(2,3,5)
(n_max,N_m,r)=(3,2,5)
(n_max,N_m,r)=(3,2,15)
Time Units
Evolution of the Cost Function
(n_max,N_m,r)=(1,6,5)
(n_max,N_m,r)=(2,3,5)
(n_max,N_m,r)=(3,2,5)
(n_max,N_m,r)=(3,2,15)
Time Units
Evolution of the Cost Function
(n_max,N_m,r)=(1,6,5)
(n_max,N_m,r)=(2,3,5)
(n_max,N_m,r)=(3,2,5)
(n_max,N_m,r)=(3,2,15)
Time Units

Conclusion

- Framework for real-time implementation of parameterized NMPC
- Not suitable for high dimensional state vectors ($n \leq 10$)
- No use of numerical differentiation
- Heavily exploit the translatability property of the control parametrization
- Use only heavily certified QP solvers
- Should enables sampling time less than few mili-seconds