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:

\[
\hat{u}(k) = (\hat{u}^{(0)}(k)^T, \ldots, \hat{u}^{(N_p-1)}(k)^T)^T \in \mathbb{U} \subset \mathbb{R}^{N_p \cdot m}
\]

that minimizes some cost function \( J(x(k), u) \) and applies \( \hat{u}^{(0)}(k) \) 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:

\[ \hat{u}(k) = (\hat{u}^{(0)}(k)^T, \ldots, \hat{u}^{(N_p-1)}(k)^T)^T \in U \subset \mathbb{R}^{N_p \cdot m} \]

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

Here, a particular parameterized control is used since:

\[ U := \left\{ u \in \mathbb{R}^{N_p \cdot m} \mid u^{(i)} = U(i, p) \quad \text{where} \ p \in \mathbb{P} \right\} \]
Consider the dynamic system:

\[ x(k + 1) = f(x(k), u(k)) \]

Ideal NMPC computes an optimal sequence:

\[ \hat{u}(k) = (\hat{u}^{(0)}(k)^T, \ldots, \hat{u}^{(N_p-1)}(k)^T)^T \in U \subset \mathbb{R}^{N_p \cdot m} \]

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

Here, a particular parameterized control is used since:

\[ U := \left\{ u \in \mathbb{R}^{N_p \cdot m} \mid u^{(i)} = U(i, p) \text{ where } p \in P \right\} \]
Consider the dynamic system:

\[ x(k + 1) = f(x(k), u(k)) \]

Ideal NMPC computes an optimal sequence:

\[ \hat{u}(k) = (\hat{u}^{(0)}(k)^T, \ldots, \hat{u}^{(N_p - 1)}(k)^T)^T \in U \subset \mathbb{R}^{N_p \cdot m} \]

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

Here, a particular parameterized control is used since:

\[
U := \left\{ u \in \mathbb{R}^{N_p \cdot m} \mid u^{(i)} = U(i, p) \text{ where } p \in P \right\}
\]

\[
\bullet \quad J(x(k), u) \longrightarrow J(x(k), p)
\]
Consider the dynamic system:

\[ x(k + 1) = f(x(k), u(k)) \]

Ideal NMPC computes an optimal sequence:

\[ \hat{u}(k) = (\hat{u}^{(0)}(k)^T, \ldots, \hat{u}^{(N_p-1)}(k)^T)^T \in U \subset \mathbb{R}^{N_p \cdot m} \]

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

Here, a particular parameterized control is used since:

\[ U := \left\{ u \in \mathbb{R}^{N_p \cdot m} \mid u^{(i)} = U(i, p) \text{ where } p \in P \right\} \]

- \( J(x(k), u) \longrightarrow J(x(k), p) \)
- 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:

\[ \hat{u}(k) = (\hat{u}^{(0)}(k))^T, \ldots, (\hat{u}^{(N_p-1)}(k))^T \in U \subset \mathbb{R}^{N_p \cdot m} \]

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

Here, a particular parameterized control is used since:

\[ U := \{ u \in \mathbb{R}^{N_p \cdot m} \mid u^{(i)} = U(i, p) \text{ where } p \in \mathbb{P} \} \]

- \( J(x(k), u) \rightarrow J(x(k), p) \)
- NMPC looks for an optimal parameter vector \( \hat{p}(x(k)) \)
- The control \( U(0, \hat{p}(x(k))) \) is applied over \([k, k + 1]\)
Consider the dynamic system:

\[ x(k + 1) = f(x(k), u(k)) \]

Ideal NMPC computes an optimal sequence:

\[ \hat{u}(k) = (\hat{u}^{(0)}(k)^T, \ldots, \hat{u}^{(N_p-1)}(k)^T)^T \in U \subset \mathbb{R}^{N_p \cdot m} \]

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

Here, a particular parameterized control is used since:

\[ U := \left\{ u \in \mathbb{R}^{N_p \cdot m} \mid u^{(i)} = U(i, p) \quad \text{where} \quad p \in \mathbb{P} \right\} \]

- \( J(x(k), u) \rightarrow J(x(k), p) \)
- NMPC looks for an optimal parameter vector \( \hat{p}(x(k)) \)
- The control \( U(0, \hat{p}(x(k))) \) 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:

\[ \hat{u}(k) = (\hat{u}^{(0)}(k)^T, \ldots, \hat{u}^{(N_p-1)}(k)^T)^T \in U \subset \mathbb{R}^{N_p \cdot m} \]

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

Here, a particular parameterized control is used since:

\[ U := \left\{ u \in \mathbb{R}^{N_p \cdot m} \mid u^{(i)} = \mathcal{U}(i, p) \text{ where } p \in \mathbb{P} \right\} \]

- \( J(x(k), u) \rightarrow J(x(k), p) \)
- 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 $\hat{p}(x(k))$. 

Unfortunately, only a finite number of iterations $q$ can be performed according to:

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

where $\theta$ 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:

\[
\begin{align*}
x(k + 1) &= f(x(k), U(0, p(k))) \\
p(k + 1) &= S(p(k), x(k), \theta(k)) \\
\theta(k + 1) &= D(p(k), x(k), \theta(k))
\end{align*}
\]

where $\theta$ 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{align*}
  x(k + 1) &= f(x(k), U(0, p(k))) \\
  p(k + 1) &= S(p(k), x(k), \theta(k)) \\
  \theta(k + 1) &= D(p(k), x(k), \theta(k))
\end{align*}

where $\theta$ 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:

\[
\begin{align*}
x(k+1) &= f(x(k), U(0, p(k))) \\
p(k+1) &= S(p(k), x(k), \theta(k)) \\
\theta(k+1) &= D(p(k), x(k), \theta(k))
\end{align*}
\]

where $\theta$ 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:

\[
\begin{align*}
    x(k + 1) &= f(x(k), U(0, p(k))) \\
    p(k + 1) &= S(p(k), x(k), \theta(k)) \\
    \theta(k + 1) &= D(p(k), x(k), \theta(k))
\end{align*}
\]

where $\theta$ 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:

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

where $\theta$ 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:

\[
\begin{align*}
x(k + 1) & = f(x(k), U(0, p(k))) \\
p(k + 1) & = S(p(k), x(k), \theta(k)) \\
\theta(k + 1) & = D(p(k), x(k), \theta(k))
\end{align*}
\]

where $\theta$ 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:

\[
\begin{align*}
    x(k+1) &= f(x(k), U(0, p(k))) \\
    p(k+1) &= S(p(k), x(k), \theta(k)) \\
    \theta(k+1) &= D(p(k), x(k), \theta(k))
\end{align*}
\]

where \( \theta \) 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:

$$x(k + 1) = f(x(k), U(0, p(k)))$$
$$p(k + 1) = S(p(k), x(k), \theta(k))$$
$$\theta(k + 1) = D(p(k), x(k), \theta(k))$$

where $\theta$ 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:

\[
\begin{align*}
    x(k+1) &= f(x(k), U(0, p(k))) \\
    p(k+1) &= S(p(k), x(k), \theta(k)) \\
    \theta(k+1) &= D(p(k), x(k), \theta(k))
\end{align*}
\]

where \( \theta \) 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:

\[
\begin{align*}
  x(k+1) &= f(x(k), U(0, p(k))) \\
  p(k+1) &= S(p(k), x(k), \theta(k)) \\
  \theta(k+1) &= D(p(k), x(k), \theta(k))
\end{align*}
\]

where $\theta$ 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:

\[
\begin{align*}
    x(k+1) &= f(x(k), U(0, p(k))) \\
    p(k+1) &= S(p(k), x(k), \theta(k)) \\
    \theta(k+1) &= D(p(k), x(k), \theta(k))
\end{align*}
\]

where $\theta$ 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:

\[
\begin{align*}
  x(k+1) &= f(x(k), U(0, p(k))) \\
  p(k+1) &= S(p(k), x(k), \theta(k)) \\
  \theta(k+1) &= D(p(k), x(k), \theta(k))
\end{align*}
\]

where $\theta$ 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:

$$x(k+1) = f(x(k), U(0, p(k)))$$

$$p(k+1) = S(p(k), x(k), \theta(k))$$

$$\theta(k+1) = D(p(k), x(k), \theta(k))$$

where $\theta$ is some internal variable expressing past knowledge.

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:

$$
x(k + 1) = f(x(k), U(0, p(k)))
$$

$$
p(k + 1) = S(p(k), x(k), \theta(k))
$$

$$
\theta(k + 1) = D(p(k), x(k), \theta(k))
$$

where $\theta$ is some internal variable expressing past knowledge.

---

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

T. Ohtsuka/Automatica, 40 (2004)


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


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

Unfortunately, only a \textbf{finite number of iterations} $q$ can be performed according to:

$$x(k+1) = f(x(k), U(0, p(k)))$$

$$p(k+1) = S(p(k), x(k), \theta(k))$$

$$\theta(k+1) = D(p(k), x(k), \theta(k))$$

where $\theta$ is some internal variable expressing past knowledge.

Existing non parameterized formulations:

\begin{itemize}
  \item High dimensional decision variable
  \item Need for highly involved softwares
  \item High memory storage, etc.
\end{itemize}

---

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

T. Ohtsuka/Automatica, 40 (2004)


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


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

Unfortunately, only a finite number of iterations $q$ can be performed according to:

\[
\begin{align*}
x(k+1) &= f(x(k), U(0, p(k))) \\
p(k+1) &= S(p(k), x(k), \theta(k)) \\
\theta(k+1) &= D(p(k), x(k), \theta(k))
\end{align*}
\]

where $\theta$ 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.

This contribution

Propose a specific instantiation of $\theta$, $S$ and $D$ 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:

\[ U(i, p) := Sat[u_{min}, u_{max}](U^{unc}(i, p)) \]

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

→ There is no constraints-related feasibility issue
→ The constrained underlying optimization problem is given by:

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

→ 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:

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

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

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:

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

Let us define by induction:

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

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

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:

\[
\forall i \in \mathbb{N} \quad \mathcal{U}(i+1, p) = \mathcal{U}(i, p^+) \]

Let us define by induction:

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

and using

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

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

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:

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

Let us define by induction:

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

and using

$$x^+ = f(x, U(0, p))$$

one can define the induction rule:

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

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

$x$
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 $\{J(z^i)\}^r_{i=0}$.

$\triangle$
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 \( \{ J(z^{i+}) \}_{i=0}^{r} \).
Use the successive data obtained during the system life-time to \textit{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:

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

<table>
<thead>
<tr>
<th>$n$</th>
<th>$n_p$</th>
<th>$n_q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>45</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>91</td>
</tr>
</tbody>
</table>
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:

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

For instance,

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

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

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

$\rho_k$ 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 \( A(0) \), use Fact1 to generate the \((r + 1) \cdot \text{card}(A(0))\) identification data:

\[
D(0) := \left\{ \mathbb{Z}(0), \{J(z)\}_{z \in \mathbb{Z}(0)} \right\}
\]

\[
\mathbb{Z}(0) := \left\{ (\hat{x}(1), p)^i \mid (p, i) \in A(0) \times \{0, \ldots, r\} \right\}
\]

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

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

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

This defines a linear system $[A_1]q = B_1 \in \mathbb{R}^{(r+1) \cdot \text{card}(\mathbb{A}(1))} \rightarrow q(2)$.
- Given $x(1)$ and $q(1)$, $p(1)$ is obtained by a small QP solution.
- For each element in $\mathbb{A}(1)$, use Fact1 to generate the $(r + 1) \cdot \text{card}(\mathbb{A}(1))$ identification data:

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

  $$\mathbb{Z}(1) := \left\{ (\hat{x}(2), p)^i+ | (p, i) \in \mathbb{A}(1) \times \{0, \ldots, r\} \right\}$$

  This defines a linear system $[A_1]q = B_1 \in \mathbb{R}^{(r+1) \cdot \text{card}(\mathbb{A}(1))} \rightarrow q(2)$
Given $x(2)$ and $q(2)$, $p(2)$ is obtained by a small QP solution.

For each elements in $A(2)$, use Fact1 to generate the $(r + 1) \cdot \text{card}(A(2))$ identification data:

- $D(2) := \{\mathbb{Z}(2), \{J(z)\}_{z \in \mathbb{Z}(2)}\}$
- $\mathbb{Z}(2) := \{\hat{x}(3), p\}^{i+} | (p, i) \in A(2) \times \{0, \ldots, r\}$

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

Successive blocks 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 \( \sigma \).
Moreover,

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

\[ \text{card}(A(k)) = n_{\text{max}} \]

where \( n_{\text{max}} \) 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:

$$\theta(k) := \left\{ D(k), \ldots, D(k - N_m + 1) \right\}$$

in which

$$D(k) := \left\{ \mathbb{Z}(k), \left\{ J(z) \right\}_{z \in \mathbb{Z}(k)} \right\}$$
During the sampling period $[k, k + 1]$,

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

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

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

   $$ \rightarrow \text{solve a linear system of dimension } n_{ls} := n_{max}(r + 1) + n_q $$

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

\[
\begin{align*}
\ddot{y} &= -u_1 \sin \theta + \epsilon u_2 \cos \theta \\
\dot{z} &= u_1 \cos \theta + \epsilon u_2 \sin \theta - 1 \\
\dot{\theta} &= u_2
\end{align*}
\]

Constraints

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

Objective Stabilize \(x^d = (y^d, z^d, 0, 0, 0, 0)^T\)
System Model

\[ \ddot{y} = -u_1 \sin \theta + \epsilon u_2 \cos \theta \]
\[ \ddot{z} = u_1 \cos \theta + \epsilon u_2 \sin \theta - 1 \]
\[ \ddot{\theta} = u_2 \]

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

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

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

\[
\begin{align*}
\ddot{y} &= -u_1 \sin \theta + \epsilon u_2 \cos \theta \\
\ddot{z} &= u_1 \cos \theta + \epsilon u_2 \sin \theta - 1 \\
\dot{\theta} &= u_2
\end{align*}
\]

Constraints

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

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

Translatability

\[
p^+ = \text{diag}(e^{-\lambda\tau_s}, e^{-3\lambda\tau_s}, e^{-6\lambda\tau_s}, e^{-\lambda\tau_s}, e^{-3\lambda\tau_s}, e^{-6\lambda\tau_s}) \cdot p
\]
System Model

\[
\begin{align*}
\ddot{y} &= -u_1 \sin \theta + \epsilon u_2 \cos \theta \\
\dot{z} &= u_1 \cos \theta + \epsilon u_2 \sin \theta - 1 \\
\dot{\theta} &= u_2
\end{align*}
\]

Constraints

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

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

Saturation on $p$

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

\[
\begin{align*}
\ddot{y} &= -u_1 \sin \theta + \epsilon u_2 \cos \theta \\
\ddot{z} &= u_1 \cos \theta + \epsilon u_2 \sin \theta - 1 \\
\dot{\theta} &= u_2
\end{align*}
\]

Constraints

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

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

Cost Function

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

\[
\begin{align*}
\ddot{y} &= -u_1 \sin \theta + \epsilon u_2 \cos \theta \\
\ddot{z} &= u_1 \cos \theta + \epsilon u_2 \sin \theta - 1 \\
\ddot{\theta} &= u_2
\end{align*}
\]

Constraints

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

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

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

\[
\bigcup_{k=0}^{N_m} \mathbb{A}(k) = \{e_i\}_i^{6} = 1
\]

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.
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**