pyMPC Documentation

Marco Forgione

March 8, 2021

1 Mathematical formulation

1.1 Dynamical system

We consider a linear, discrete-time dynamical system with n_u inputs and n_x states:

$$x_{k+1} = Ax_k + Bu_k, \tag{1}$$

with $A \in \mathbb{R}^{n_x \times n_x}$ and $B \in \mathbb{R}^{n_x \times n_u}$.

1.2 MPC cost function

The Model Predictive Control (MPC) problem solved by pyMPC is:

$$\arg\min_{U} \overbrace{\frac{1}{2}\big(x_N - x_{\text{ref}}\big)^\top Q_{x_N}\big(x_N - x_{\text{ref}}\big)}^{=J_{Q_{x_N}}} + \overbrace{\frac{1}{2}\sum_{k=0}^{N_p-1}\big(x_k - x_{\text{ref}}\big)^\top Q_x\big(x_k - x_{\text{ref}}\big)}^{J_{Q_x}} +$$
$$+ \overbrace{\frac{1}{2}\sum_{k=0}^{N_p-1}\big(u_k - u_{\text{ref}}\big)^\top Q_u\big(u_k - u_{\text{ref}}\big)}^{J_u} + \overbrace{\frac{1}{2}\sum_{k=0}^{N_p-1}\Delta u_k^\top Q_{\Delta u}\Delta u_k}^{J_{\Delta u}} \tag{2a}$$

subject to :

$$x_{k+1} = Ax_k + Bu_k \tag{2b}$$
$$u_{\min} \leq u_k \leq u_{\max} \tag{2c}$$
$$x_{\min} \leq x_k \leq x_{\max} \tag{2d}$$
$$\Delta u_{\min} \leq \Delta u_k \leq \Delta u_{\max} \tag{2e}$$
$$x_0 = \bar{x} \tag{2f}$$
$$u_{-1} = \bar{u}, \tag{2g}$$

where $\Delta u_k = u_k - u_{k-1}$ and the optimization variables are the elements of the input sequence:

$$U = \left\{ u_0,\ u_1,\ \ldots,\ u_{N_p-1} \right\}. \tag{3}$$

According to the numerical solution strategy, the elements of the state sequence:

$$X = \left\{ x_0,\ x_1,\ \ldots,\ x_{N_p} \right\} \tag{4}$$

may be either included as optimization variables of the Quadratic Programming (QP) problem (in the sparse formulation), or eliminated (in the condensed formulation). The current implementation of pyMPC is based on the sparse formulation and utilizes the QP solver OSQP.

1.3 Notation

Bolt symbols are used to denote column vectors containing the stacked elements of quantities of interest over the prediction horizon:

$$\mathbf{u}^\top = \begin{bmatrix} u_0^\top & u_1^\top & \ldots & u_{N_p-1}^\top \end{bmatrix}^\top, \tag{5a}$$

$$\mathbf{x}^\top = \begin{bmatrix} x_0^\top & x_1^\top & \ldots & x_{N_p}^\top \end{bmatrix}^\top, \tag{5b}$$

$$\mathbf{x}_{\text{ref}}^\top = \begin{bmatrix} x_{\text{ref}}^\top & x_{\text{ref}}^\top & \ldots & x_{\text{ref}}^\top \end{bmatrix}^\top, \tag{5c}$$

$$\mathbf{u}_{\text{ref}}^\top = \begin{bmatrix} u_{\text{ref}}^\top & u_{\text{ref}}^\top & \ldots & u_{\text{ref}}^\top \end{bmatrix}^\top, \tag{5d}$$

$$\mathbf{x}_{\text{min}}^\top = \begin{bmatrix} x_{\text{min}}^\top & x_{\text{min}}^\top & \ldots & x_{\text{min}}^\top \end{bmatrix}^\top, \tag{5e}$$

$$\mathbf{x}_{\text{max}}^\top = \begin{bmatrix} x_{\text{max}}^\top & x_{\text{max}}^\top & \ldots & x_{\text{max}}^\top \end{bmatrix}^\top, \tag{5f}$$

$$\mathbf{u}_{\text{min}}^\top = \begin{bmatrix} u_{\text{min}}^\top & u_{\text{min}}^\top & \ldots & u_{\text{min}}^\top \end{bmatrix}^\top, \tag{5g}$$

$$\mathbf{u}_{\text{max}}^\top = \begin{bmatrix} u_{\text{max}}^\top & u_{\text{max}}^\top & \ldots & u_{\text{max}}^\top \end{bmatrix}^\top, \tag{5h}$$

$$\Delta\mathbf{u}_{\text{min}}^\top = \begin{bmatrix} \Delta u_{\text{min}}^\top & \Delta u_{\text{min}}^\top & \ldots & \Delta u_{\text{min}}^\top \end{bmatrix}^\top, \tag{5i}$$

$$\Delta\mathbf{u}_{\text{max}}^\top = \begin{bmatrix} \Delta u_{\text{max}}^\top & \Delta u_{\text{max}}^\top & \ldots & \Delta u_{\text{max}}^\top \end{bmatrix}^\top. \tag{5j}$$

Note that we consider constant references and constraints over the prediction horizon for notation simplicity. The extension to time-varying references and constraints is straightforward.

1.4 Receding horizon implementation

In a typical implementation, the MPC input is applied in *receding horizon*. At each time step i, problem (2) is solved with $x_0 = x[i],\ u_{-1} = u[i-1]$ and

an optimal input sequence $u_0, \ldots, u_{N_p}$ is obtained. The first element of this sequence u_0 is the control input that is actually applied at time instant i. At time instant $i+1$, a new state $x[i+1]$ is measured (or estimated), and the process is iterated.

Thus, formally, the MPC control law is a (static) function of the current state and the previous input:

$$u_{MPC} = K(x[i], u[i-1]). \tag{6}$$

Note that this function also depends on the references x_{ref} and u_{ref} and on the system matrices A and B.

2 Quadratic Programming Formulation

Available Quadratic Programming (QP) solvers such as OSQP and QPOases expect a problem having a standard form[1] such as:

$$\min \frac{1}{2}\mathbf{z}^\top \mathbf{P}\mathbf{z} + \mathbf{p}^\top \mathbf{z} \tag{7a}$$

subject to

$$l \leq \mathbf{A}\mathbf{z} \leq u. \tag{7b}$$

Thus, to implement the MPC controller using a standard QP solver, we need to re-write the MPC optimization problem (2) in form (7).

3 Sparse QP Formulation

In the *sparse* QP formulation of the MPC problem, the decision variables to be optimized are both $\mathbf{u}$ and $\mathbf{x}$. The dynamic constraints given by the model equations (2b) are included explicitly as equality constraints on the state variables $\mathbf{x}$.

The resulting QP problem is high-dimensional, sparse, and highly structured. Certain QP solvers can take advantage of sparsity and structure of the QP problem and perform well with this formulation, despite its high dimensionality.

[1] The exact formulation is solver-dependent, but the different formulations are substantially equivalent.

3.1 Cost function

3.1.1 Terms in Q_x and Q_{x_N}

By direct inspection, the non-constant terms of the cost function in Q_x and Q_{x_N} are:

$$J_{Q_x} = \frac{1}{2}\begin{bmatrix} x_0^\top & x_1^\top & \dots & x_{N_p}^\top \end{bmatrix}^\top \overbrace{\text{blkdiag}(Q_x, Q_x, \dots, Q_{x_N})}^{=\mathcal{Q}_x} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N_p} \end{bmatrix} + \\ + \overbrace{\begin{bmatrix} -{x_{\text{ref}}}^\top Q_x & -{x_{\text{ref}}}^\top Q_x & \dots & -{x_{\text{ref}}}^\top Q_{x_N} \end{bmatrix}}^{p_{Q_x}} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N_p} \end{bmatrix}^\top . \quad (8)$$

More compactly, this is equivalent to:

$$J_{Q_x} = \frac{1}{2}\mathbf{x}^\top \mathcal{Q}_x \mathbf{x} - \mathbf{x}_{\text{ref}}^\top \mathcal{Q}_x \mathbf{x} = \frac{1}{2}\mathbf{x}^\top \mathcal{Q}_x \mathbf{x} + p_{Q_x}^\top \mathbf{x}. \quad (9)$$

3.1.2 Terms in Q_u

Similarly, for the term J_{Q_u}:

$$J_{Q_u} = \frac{1}{2}\begin{bmatrix} u_0^\top & u_1^\top & \dots & u_{N_p-1}^\top \end{bmatrix} \overbrace{\text{blkdiag}(Q_u, Q_u, \dots, Q_u)}^{\mathcal{Q}_u} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N_p-1} \end{bmatrix} + \\ + \overbrace{\begin{bmatrix} -{u_{\text{ref}}}^\top Q_u & -{u_{\text{ref}}}^\top Q_u & \dots & -{u_{\text{ref}}}^\top Q_u \end{bmatrix}}^{p_{Q_u}} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N_p-1} \end{bmatrix} \quad (10)$$

More compactly, this is equivalent to:

$$J_{Q_u} = \frac{1}{2}\mathbf{u}^\top \mathcal{Q}_u \mathbf{u} - \mathbf{u}_{\text{ref}}^\top \mathcal{Q}_u \mathbf{u} = \frac{1}{2}\mathbf{u}^\top \mathcal{Q}_u \mathbf{u} + p_{Q_u} \mathbf{u}. \quad (11)$$

3.1.3 Terms in $Q_{\Delta u}$

As for the terms in $Q_{\Delta u}$, we have instead:

$$J_{\Delta u} = \frac{1}{2}\begin{bmatrix} u_0 & u_1 & \dots & u_{N_p-1} \end{bmatrix}^\top \overbrace{\begin{bmatrix} 2Q_{\Delta u} & -Q_{\Delta u} & 0 & \dots & \dots & 0 \\ -Q_{\Delta u} & 2Q_{\Delta u} & -Q_{\Delta u} & 0 & \dots & 0 \\ 0 & -Q_{\Delta u} & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & -Q_{\Delta u} & 2Q_{\Delta u} & -Q_{\Delta u} \\ 0 & 0 & 0 & 0 & -Q_{\Delta u} & Q_{\Delta u} \end{bmatrix}}^{\mathcal{Q}_{\Delta u}} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N_p-1} \end{bmatrix}^\top - u_{-1}^\top Q_{\Delta u} u_0 \quad (12)$$

More compactly, this is equivalent to:

$$J_{\Delta u} = \frac{1}{2}\mathbf{u}^\top \mathcal{Q}_{\Delta u}\mathbf{u} - u_{-1}^\top Q_{\Delta u} u_0$$

It is convenient to write the above expression using stacked vectors only:

$$J_{\Delta u} = \frac{1}{2}\mathbf{u}^\top \mathcal{Q}_{\Delta u}\mathbf{u} + \overbrace{\begin{bmatrix} -{u_{-1}}^\top Q_{\Delta u} & 0 & \dots & 0 \end{bmatrix}}^{p_{Q_{\Delta u}}} \mathbf{u} \quad (13)$$

3.2 Constraints

3.2.1 Linear dynamics

Let us consider the linear equality constraints (2b) representing the system dynamics. These can be written in matrix form as:

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N_p-1} \\ x_{N_p} \end{bmatrix} = \overbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ A_d & 0 & \dots & 0 \\ 0 & A_d & \dots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \dots & A_d \end{bmatrix}}^{=\mathcal{A}} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N_p-1} \\ x_{N_p} \end{bmatrix} + \overbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ B_d & 0 & \dots & 0 \\ 0 & B_d & \dots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \dots & B_d \end{bmatrix}}^{=\mathcal{B}} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N_p-2} \\ u_{N_p-1} \end{bmatrix} + \overbrace{\begin{bmatrix} \bar{x} \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}}^{=\mathcal{C}}. \quad (14)$$

Thus, we get a set of linear equality constraints representing the system dynamics (2b). These constraints can be written as

$$\begin{bmatrix} (\mathcal{A} - I) & \mathcal{B} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} = -\mathcal{C}. \quad (15)$$

3.2.2 Variable bounds: x and u

The bounds on x and u are readily implemented as:

$$\begin{bmatrix}\mathbf{x}_{\min}\\ \mathbf{u}_{\min}\end{bmatrix} \leq \begin{bmatrix}I & 0\\ 0 & I\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} \leq \begin{bmatrix}\mathbf{x}_{\max}\\ \mathbf{u}_{\max}\end{bmatrix}. \tag{16}$$

3.2.3 Input increment bound: Δu

The input increment bound for all the time steps may be written as:

$$\begin{bmatrix}u_{-1}+\Delta u_{\min}\\ \Delta u_{\min}\\ \vdots\\ \Delta u_{\min}\end{bmatrix} \leq \overbrace{\begin{bmatrix}I & 0 & \dots & \dots & 0 & 0\\ -I & I & 0 & \dots & 0 & 0\\ 0 & -I & I & \dots & 0 & 0\\ \vdots & & & & & \\ 0 & 0 & \dots & 0 & -I & I\end{bmatrix}}^{=\Delta} \begin{bmatrix}u_0\\ u_1\\ \vdots\\ u_{N_c-1}\end{bmatrix} \leq \begin{bmatrix}u_{-1}+\Delta u_{\max}\\ \Delta u_{\max}\\ \vdots\\ \Delta u_{\max},\end{bmatrix} \tag{17}$$

where we have defined the matrix Δ performing the finite difference operation on $\mathbf{u}$.

3.3 Soft constraints

Bounds on x may result in an unfeasible QP problem! A common solution is to transform the hard constraints in x into soft constraints by means of *slack variables* $\boldsymbol{\epsilon}$. In the current implementation, there are as many slack variables as state variables, i.e. $\boldsymbol{\epsilon} \in \mathbb{R}^{N_p n_x \times 1}$. We use the constraint:

$$\begin{bmatrix}x_{\min}\\ u_{\min}\end{bmatrix} \leq \begin{bmatrix}I & 0 & I\\ 0 & I & 0\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\\ \boldsymbol{\epsilon}\end{bmatrix} \leq \begin{bmatrix}x_{\max}\\ u_{\max}\end{bmatrix}. \tag{18}$$

In order to penalize constraint violation, we have a penalty term in the cost function:

$$J_\epsilon = \frac{1}{2}\boldsymbol{\epsilon}^\top \mathcal{Q}_\epsilon \boldsymbol{\epsilon}, \tag{19}$$

where

$$\mathcal{Q}_\epsilon = \text{blkdiag}(Q_\epsilon, Q_\epsilon, \dots Q_\epsilon) \tag{20}$$

and Q_ϵ is σI_{n_x}, with σ a "large" constant (e.g. $\sigma = 10^4$).

3.4 Control Horizon

Sometimes, we may want to use a control horizon $N_c < N_p$ instead of the standard $N_c = N_p$. In this case, the input considered is constant for $N_c \geq N_p$.

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N_p-1} \\ x_{N_p} \end{bmatrix} = \overbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ A_d & 0 & \dots & 0 \\ 0 & A_d & \dots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \dots & A_d \end{bmatrix}}^{=\mathcal{A}} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N_p-1} \\ x_{N_p} \end{bmatrix} + \overbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ B_d & 0 & \dots & 0 \\ 0 & B_d & \dots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \dots & B_d \\ 0 & 0 & \dots & \vdots \\ 0 & 0 & \dots & B_d \end{bmatrix}}^{=\mathbb{B}} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N_c-1} \\ \vdots \\ u_{N_c-1} \end{bmatrix} + \overbrace{\begin{bmatrix} \bar{x} \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}}^{=\mathcal{C}} \tag{21}$$

The contributions J_{Q_u} of the cost function also changes:

$$J_{Q_u} = \frac{1}{2} \begin{bmatrix} u_0^\top & u_1^\top & \dots & u_{N_p-1}^\top \end{bmatrix} \operatorname{blkdiag}\Big(Q_u, Q_u, \dots, (N_p - N_c + 1) Q_u \Big) \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N_p-1} \end{bmatrix} +$$

$$+ \begin{bmatrix} -{u_{\text{ref}}}^\top Q_u & -{u_{\text{ref}}}^\top Q_u & \dots & -(N_p - N_c + 1) {u_{\text{ref}}}^\top Q_u \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N_p-1} \end{bmatrix} \tag{22}$$

Instead, $J_\Delta u$ does not change (because the input is constant for $k \geq N_c$!

3.5 Overall formulation

The overall sparse QP formulation is thus

$$\min \frac{1}{2} \mathbf{z}^\top \mathbf{P} \mathbf{z} + \mathbf{p}^\top \mathbf{z} \tag{23a}$$

subject to

$$l \leq \mathbf{A} \mathbf{z} \leq u, \tag{23b}$$

where

$$\mathbf{z}^\top = \begin{bmatrix}\mathbf{x}^\top & \mathbf{u}^\top & \boldsymbol{\epsilon}^\top\end{bmatrix} \tag{24a}$$

$$\mathbf{P} = \text{blkdiag}(\mathcal{Q}_x, \mathcal{Q}_u + \mathcal{Q}_{\Delta u}, \mathcal{Q}_\epsilon) \tag{24b}$$

$$\mathbf{p}^\top = \begin{bmatrix}p_{Q_x}^\top, p_{Q_u}^\top + p_{\Delta_u}^\top, 0\end{bmatrix} \tag{24c}$$

$$\mathbf{A} = \begin{bmatrix}(\mathcal{A} - I) & \mathcal{B} & 0 \\ I & 0 & I \\ 0 & I & 0 \\ 0 & \Delta & 0\end{bmatrix} \tag{24d}$$

$$l^\top = \begin{bmatrix}-\mathcal{C}^\top, \mathbf{x}_{\min}^\top, \mathbf{u}_{\min}^\top, \Delta\mathbf{u}_{\min}^\top\end{bmatrix}^\top \tag{24e}$$

$$u^\top = \begin{bmatrix}-\mathcal{C}^\top, \mathbf{x}_{\max}^\top, \mathbf{u}_{\max}^\top, \Delta\mathbf{u}_{\max}^\top\end{bmatrix}^\top. \tag{24f}$$

3.6 Implementation details

Exploiting sparsity The sparse QP formulation of MPC may be convenient when it is used in combination with a QP solver that is able to exploit sparsity and structure of the problem. The OSQP solver seems to be a good candidate. The sparse QP matrices have to be implemented in a sparse format (e.g., using the `scipy.sparse` API in Python).

Minimal update of the QP matrices The terms $\mathcal{C}$ in (24e), (24f) and $p_{\Delta u}$ in (24c) have to be modified at each iterationas as they depend on the initial state x_0 and the previously applied input u_{-1}, respectively. In the case of time-varying system matrices A_d and B_d, the terms $\mathcal{A}$ and $\mathcal{B}$ in (24d) also change and have to be updated. All the other terms do not change for the different iterations of the MPC. It is important to perform a minimal on-line update of the system matrices to save computational time.

Kronecker product Several matrices appearing in the expressions above may be more easily implemented using the *kronecker product*. For instance, the $\mathcal{A}$ is equivalent to:

$$\mathcal{A} = \text{kron}\left(\begin{bmatrix}0 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \dots & 1\end{bmatrix}, A_d\right) \tag{25}$$

In Python code, we obtain the sparse matrix $\mathcal{A}$ with the syntax:

```
A_cal = sparse.kron(sparse.eye(N, k=-1), Ad).
```

Similarly, we would obtain the dense version of $\mathcal{A}$ with the syntax:

```
A_cal = np.kron(np.eye(N, k=-1), Ad).
```

4 Condensed QP formulation

In the *condensed* QP formulation, the input sequence vector $\mathbf{u}$ are the only optimization variables. The state variables are eliminated by applying the so-called Lagrange equations:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N_p-1} \\ x_{N_p} \end{bmatrix} = \overbrace{\begin{bmatrix} A_d \\ A_d^2 \\ \vdots \\ A_d^{N_p-1} \\ A_d^{N_p} \end{bmatrix}}^{=\mathcal{A}} x_0 + \overbrace{\begin{bmatrix} B_d & 0 & 0 & \dots & 0 & 0 \\ A_dB_d & B_d & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & 0 & 0 \\ A_d^{N_p-2}B_d & A_d^{N_p-3}B_d & A_d^{N_p-4}B_d & \dots & B_d & 0 \\ A_d^{N_p-1}B_d & A_d^{N_p-2}B_d & A_d^{N_p-3}B_d & \dots & A_dB_d & B_d \end{bmatrix}}^{=\mathcal{B}} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N_p-1} \end{bmatrix}. \tag{26}$$

In vector notation, this is simply:

$$\mathbf{x} = \mathcal{A}x_0 + \mathcal{B}\mathbf{u} \tag{27}$$

The resulting QP problem is lower-dimensional than the one obtained with the sparse formulation (Section 3), as $\mathbf{x}$ is no longer a free decision variable. However, sparsity and structural properties of the resulting QP problem are generally lost.

4.1 Cost function

The elements of the cost function may be written compactly as:

$$J_{Q_x} = \frac{1}{2}\left(\mathcal{A}x_0 + \mathcal{B}\mathbf{u} - \mathbf{x}_{\text{ref}}\right)^\top \mathcal{Q}_x \left(\mathcal{A}x_0 + \mathcal{B}\mathbf{u} - \mathbf{x}_{\text{ref}}\right) \tag{28}$$

$$J_{Q_u} = \frac{1}{2}(\mathbf{u} - \mathbf{u}_{\text{ref}})^\top \mathcal{Q}_u (\mathbf{u} - \mathbf{u}_{\text{ref}}) \tag{29}$$

$$J_{Q_{\Delta u}} = \frac{1}{2}\mathbf{u}^\top \mathcal{Q}_{\Delta u}\mathbf{u} + \begin{bmatrix} -u_{-1}{}^\top Q_{\Delta u} & 0 & \dots & 0 \end{bmatrix} \mathbf{u} \tag{30}$$

Summing up and expanding terms:

$$J = \frac{1}{2}\mathbf{u}^\top \mathcal{B}^\top \mathcal{Q}_x \mathcal{B}\mathbf{u} + \frac{1}{2}(\mathcal{A}x_0 - \mathbf{x}_{\text{ref}})^\top \mathcal{Q}_x (\mathcal{A}x_0 - \mathbf{x}_{\text{ref}}) + (\mathcal{A}x_0 - \mathbf{x}_{\text{ref}})^\top \mathcal{Q}_x \mathcal{B}\mathbf{u} + \\ + \frac{1}{2}\mathbf{u}^\top \mathcal{Q}_u \mathbf{u} + \frac{1}{2}{\mathbf{u}_{\text{ref}}}^\top \mathcal{Q}_u \mathbf{u}_{\text{ref}} + -{\mathbf{u}_{\text{ref}}}^\top \mathcal{Q}_u \mathbf{u} + \\ + \frac{1}{2}\mathbf{u}^\top \mathcal{Q}_{\Delta u}\mathbf{u} + \begin{bmatrix} -{u_{-1}}^\top Q_{\Delta u} & 0 & \dots & 0 \end{bmatrix} \mathbf{u} \quad (31)$$

Neglecting constant terms and collecting:

$$J = C + \frac{1}{2}\mathbf{u}^\top \overbrace{\left(\mathcal{B}^\top \mathcal{Q}_x \mathcal{B} + \mathcal{Q}_u + \mathcal{Q}_{\Delta u}\right)}^{=\mathbf{P}} \mathbf{u} + \\ \overbrace{\left[(\mathcal{A}x_o - \mathbf{x}_{\text{ref}})^\top \mathcal{Q}_x \mathcal{B} - {\mathbf{u}_{\text{ref}}}^\top \mathcal{Q}_u - \begin{bmatrix} {u_{-1}}^\top Q_{\Delta u} & 0 & \dots & 0 \end{bmatrix} \right)}^{=\mathbf{p}^\top} \mathbf{u} \quad (32)$$

Thus, we have

$$\mathbf{p} = \mathcal{B}^\top \mathcal{Q}_x (\mathcal{A}x_0 - \mathbf{x}_{\text{ref}}) - \mathcal{Q}_u \mathbf{u}_{\text{ref}} + \begin{bmatrix} -Q_{\Delta u} u_{-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (33)$$

expanding

$$\mathbf{p} = \overbrace{\mathcal{B}^\top \mathcal{Q}_x \mathcal{A}}^{=p_{x_0}} x_0 \; + \; \overbrace{-\mathcal{B}^\top \mathcal{Q}_x}^{p_{\mathbf{x}_{\text{ref}}}} \mathbf{x}_{\text{ref}} \; + \; \overbrace{-\mathcal{Q}_u}^{=p_{\mathbf{u}_{\text{ref}}}} \mathbf{u}_{\text{ref}} + \overbrace{\begin{bmatrix} -Q_{\Delta u} \\ 0 \\ \vdots \\ 0 \end{bmatrix}}^{=p_{u_{-1}}} u_{-1} \quad (34)$$

4.2 Control horizon

In this case, we define:

$$\mathbf{u}^\top = \begin{bmatrix} u_0^\top & u_1^\top & \dots & u_{N_c-1}^\top \end{bmatrix}^\top . \quad (35)$$

We can exploit the equation:

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N_p-1} \end{bmatrix} = \overbrace{\begin{bmatrix} I & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \dots & I \\ 0 & 0 & \dots & \vdots \\ 0 & 0 & \dots & I \end{bmatrix}}^{=S} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N_c-1} \end{bmatrix} = S\mathbf{u} \tag{36}$$

Then, the state variables satisfy:

$$\mathbf{x} = \mathcal{A}x_0 + \mathcal{B}S\mathbf{u}. \tag{37}$$

Then, the cost function term J_{Q_x} can then be computed as in (28), by re-defining $\mathcal{B} = S\mathcal{B}$. The term J_{Q_u} should also be adjusted by re-defining $\mathcal{Q}_u = \text{blkdiag}\,(Q_u, Q_u, \dots (N_p - N_c + 1)Q_u)$, as the last element of the input sequence in the control horizon has to be weighted for $(N_p - N_c + 1)$ time steps.

4.3 Implementation details

By adopting the condensed MPC formulation, we obtain a lower-dimensional QP problem as the state variables are eliminated. However, the sparsity properties of the resulting QP problem are generally lost. In particular, the matrix $\mathbf{P}$ is in general fully populated.

In practice, we expect good performance from the condensed formulation for relatively small MPC problems, and using solvers optimized for dense QP problems. A good candidate may be QPOases.

4.4 The unconstrained case

By dropping the constraints on u Δu, and x, the condensed formulation allows writing the solution of the MPC problem in closed-form. In fact, for an unconstrained QP problem, the optimal value of $\mathbf{u}$ is:

$$\mathbf{u}^{\text{opt}} = -\mathbf{P}^{-1}\mathbf{p}. \tag{38}$$

Expanding, we obtain in our case:

$$\mathbf{u}^{\text{opt}} = k_{x_0}x_0 + k_{\mathbf{x}_{\text{ref}}}\mathbf{x}_{\text{ref}} + k_{\mathbf{u}_{\text{ref}}}\mathbf{u}_{\text{ref}}k_{u_{-1}}u_{-1} \tag{39}$$

with:

$$k_{x_0} = -\mathbf{P}^{-1} p_{x_0} \tag{40a}$$
$$k_{\mathbf{x}_{\text{ref}}} = -\mathbf{P}^{-1} p_{\mathbf{x}_{\text{ref}}} \tag{40b}$$
$$k_{\mathbf{u}_{\text{ref}}} = -\mathbf{P}^{-1} p_{\mathbf{u}_{\text{ref}}} \tag{40c}$$
$$k_{u_{-1}} = -\mathbf{P}^{-1} p_{u_{-1}}. \tag{40d}$$

Thus, the control law is a simple linear function of x_0, $\mathbf{x}_{\text{ref}}$, $\mathbf{u}_{\text{ref}}$, and u_{-1}.

For constant system matrices A and B, the QP matrices $\mathbf{P}$ and $\mathbf{p}$ are also fixed. It is then convenient to compute the inverse of $\mathbf{P}$ off-line and use formulas (40a) for real-time computations. In the case of time-varying system matrices, the QP matrices change at each iteration of MPC. Then, it is perhaps more efficient and numerically stable to obtain the solution of (38) by using standard routines for solving linear systems, instead of computing the inverse of $\mathbf{P}$ on-line.