2016 IEEE 55th Conference on Decision and Control (CDC)
ARIA Resort & Casino
December 12-14, 2016, Las Vegas, USA

An Efficient Implementation of Partial Condensing for Nonlinear Model Predictive Control

Gianluca Frison[1,2], Dimitris Kouzoupis[1], John Bagterp Jørgensen[2], Moritz Diehl[1]

Abstract— **Partial (or block) condensing is a recently proposed technique to reformulate a Model Predictive Control (MPC) problem into a form more suitable for structure-exploiting Quadratic Programming (QP) solvers. It trades off horizon length for input vector size, and this degree of freedom can be employed to find the best problem size for the QP solver at hand. This paper proposes a Hessian condensing algorithm particularly well suited for partial condensing, where a state component is retained as an optimization variable at each stage of the partially condensed MPC problem. The optimal input-horizon trade-off is investigated from a theoretical point of view (based on algorithms flop count) as well as by benchmarking (in practice, the performance of linear algebra routines for different matrix sizes plays a key role). Partial condensing can also be seen as a technique to replace many operations on small matrices with fewer operations on larger matrices, where linear algebra routines perform better. Therefore, in case of small-scale MPC problems, partial condensing can greatly improve performance beyond the flop count reduction.**

I. Introduction

One classical way of solving linear Model Predictive Control (MPC) problems is to use the state-space equation to remove the state variables from the problem formulation. This technique is known as condensing and it reformulates the large and sparse Hessian of the MPC problem into a small and dense one, that is generally factorized using a Cholesky factorization at a computational cost of $\mathcal{O}(N^3)$ flops (where N is the horizon length of the MPC problem). Alternatively, the backward Riccati recursion [12], [14], [15] or the Schur complement method [4], [17] can be employed to solve the sparse formulation of the linear MPC problem at a computational cost of $\mathcal{O}(N)$ flops. Despite the favorable scalability in N of the methods operating on the sparse formulation, condensing can be competitive also in a Nonlinear MPC (NMPC) framework [16], especially after the recent development of efficient Hessian condensing algorithms [1], [3], [7], [11].

In [2], the author makes redundant the question of whether the sparse or the condensed formulation is preferable, by proposing techniques to control the level of sparsity in MPC problems, trading off horizon length for input vector size. This degree of freedom in the formulation of the MPC problem can be employed to find the optimal level of sparsity given the Quadratic Programming (QP) solver at hand, e.g. the one minimizing the flop count. In particular, the horizon length can be reduced at the expense of a larger input vector size (partial condensing), or conversely the input vector size can be reduced at the expense of a larger horizon length (sequential update). For the Riccati recursion solver considered in [2], it is found that if roughly $n_x > n_u$ (where n_x is the state vector size and n_u is the input vector size) a decrease in the horizon length leads to a reduction in the flop count, while if roughly $n_x < n_u$ an increase in the horizon length leads to a reduction in the flop count.

The authors in [13] investigate the use of partial condensing (named block condensing therein) in the framework of NMPC. When employed in conjunction with the dual Newton strategy, as implemented in the software qpDUNES [5], partial condensing may bring further computational benefits beyond the flop count reduction, due to the decrease of the required number of outer Newton iterations [13].

In the current paper, only the partial condensing case is considered. In the solution of linear MPC problems (and more generally linear Optimal Control Problems (OCP)), the partial condensing procedure can be performed off-line, and therefore the efficiency of the partial condensing algorithm itself is of limited interest. However, in the NMPC framework, the partial condensing algorithm is performed on-line, at each Sequential Quadratic Programming (SQP) iteration. Therefore, in this paper a condensing procedure specially tailored to partial condensing is proposed. This algorithm takes into account the fact that a state component is retained as an optimization variable at each stage of the partially condensed OCP and it exploits the lack of terminal cost in the OCP sub-problems being condensed. This is investigated in Section IV, while Sections II and III recall the linear OCP and the partial condensing idea.

A Riccati-based Interior Point Method (IPM) is employed for the solution of linear OCPs. An efficient implementation of the backward Riccati recursion is described in [10]. The optimal horizon length N_p for the partially condensed OCP is investigated both from a theoretical point of view and by benchmarking. In the latter case, the performance of linear algebra routines for different matrix sizes plays a key role. The linear algebra routines are implemented using the high-performance techniques proposed in [8], [9]. This is investigated in Sections V and VI.

Finally, partial condensing reformulates the state bounds of the original OCP into general mixed linear constraints that depend on all inputs and initial state of each condensed stage. In an IPM these constraints are computationally much more expensive than bounds. In OCP formulations with many state bounds, this extra cost can destroy the computational

[1] Department of Microsystems Engineering, University of Freiburg {gianluca.frison , dimitris.kouzoupis , moritz.diehl} at imtek.uni-freiburg.de.
[2] Department of Applied Mathematics and Computer Science, Technical University of Denmark {giaf , jbjo} at dtu.dk.

advantages due to partial condensing. However, since each state depends on the inputs only at previous stages, the constraint matrix is block triangular. If this structure is exploited, these constraints can be processed much cheaper than general constraints, in case of OCPs with many states and long horizon. This is investigated in Section VII.

II. Problem Formulation

In this paper, linear OCPs are assumed to be QPs in the special form

$$\begin{aligned}\min_{u,x} \quad & \phi = \sum_{n=0}^{N-1} \frac{1}{2}\begin{bmatrix}u_n\\x_n\\1\end{bmatrix}^T \begin{bmatrix}R_n & S_n & r_n\\ S_n^T & Q_n & q_n\\ r_n^T & q_n^T & 0\end{bmatrix}\begin{bmatrix}u_n\\x_n\\1\end{bmatrix} + \\ & \quad + \frac{1}{2}\begin{bmatrix}x_N\\1\end{bmatrix}^T\begin{bmatrix}Q_N & q_N\\ q_N^T & 0\end{bmatrix}\begin{bmatrix}x_N\\1\end{bmatrix}\\ s.t. \quad & x_0 = \hat{x}_0 \\ & x_{n+1} = A_n x_n + B_n u_n + b_n\\ & u_n^{\mathrm{l}} \le u_n \le u_n^{\mathrm{u}}\\ & x_n^{\mathrm{l}} \le x_n \le x_n^{\mathrm{u}}\\ & d_n^{\mathrm{l}} \le C_n x_n + D_n u_n \le d_n^{\mathrm{u}}\end{aligned} \tag{1}$$

The cost function has linear and quadratic terms in both u_n and x_n for $n = 0, \dots, N-1$, plus the terminal cost on x_N at the last stage. The dynamics are described by a state-space equation. There are bounds on inputs and states at all stages, plus general constraints at all stages.

If the constraint on the initial value of x_0 is retained, this is a MPC problem. If conversely it is dropped, this is in the form of a Moving Horizon Estimation (MHE) problem.

III. Idea and Notation

This section recalls the idea of partial condensing through an example and introduces the notation used in the remainder of the paper. Let us consider an OCP with horizon length $N = 6$, and let us define the partition of the state and input vectors such that the states and inputs of $N_c = 3$ consecutive stages are grouped together into blocks (i.e. groups of consecutive stages).

$$\bar{x} = \left[\begin{array}{c}x_0\\x_1\\x_2\\\hline x_3\\x_4\\x_5\\\hline x_6\end{array}\right] \doteq \begin{bmatrix}\bar{x}_0\\\bar{x}_3\\x_6\end{bmatrix}, \quad \bar{u} = \left[\begin{array}{c}u_0\\u_1\\u_2\\\hline u_3\\u_4\\u_5\end{array}\right] \doteq \begin{bmatrix}\bar{u}_0\\\bar{u}_3\end{bmatrix}.$$

These can be seen as the state and input vector of an OCP with horizon length $N_p = N/N_c = 2$. The cost function matrices and vectors can be partitioned accordingly,

$$\bar{Q} = \begin{bmatrix}\bar{Q}_0 & 0 & 0\\ 0 & \bar{Q}_3 & 0\\ 0 & 0 & Q_6\end{bmatrix}, \quad \bar{q} = \begin{bmatrix}\bar{q}_0\\\bar{q}_3\\q_6\end{bmatrix},$$

$$\bar{S} = \begin{bmatrix}\bar{S}_0 & 0 & 0\\ 0 & \bar{S}_3 & 0\end{bmatrix}, \quad \bar{R} = \begin{bmatrix}\bar{R}_0 & 0\\ 0 & \bar{R}_3\end{bmatrix}, \quad \bar{r} = \begin{bmatrix}\bar{r}_0\\\bar{r}_3\end{bmatrix}.$$

Therefore the cost function can be rewritten as

$$\begin{aligned}\phi &= \sum_{i=0}^{N_p-1}\frac{1}{2}\begin{bmatrix}\bar{u}_k\\\bar{x}_k\end{bmatrix}^T\begin{bmatrix}\bar{R}_k & \bar{S}_k\\ \bar{S}_k^T & \bar{Q}_k\end{bmatrix}\begin{bmatrix}\bar{u}_k\\\bar{x}_k\end{bmatrix} + \begin{bmatrix}\bar{r}_k\\\bar{q}_k\end{bmatrix}^T\begin{bmatrix}\bar{u}_k\\\bar{x}_k\end{bmatrix} + \phi_N\\ &= \sum_{i=0}^{N_p-1}\bar{\phi}_k + \phi_N\end{aligned}$$

where $k := N_c i$ and ϕ_N is the terminal cost.

The idea of partial condensing is to remove all but the first state component in each block by means of condensing. A block of N_c stages is condensed into a single stage (that motivates the name block condensing). If the original OCP is an MPC problem, 1 out of the N_p sub-problems are in MPC form (i.e. the initial state is fixed) and $N_p - 1$ are in MHE form (i.e. the initial state is an optimization variable). In the case that the original OCP is an MHE problem, all N_p sub-problems are in the MHE form.

Given an horizon of $N_c = 3$ within each block, the state vector $\bar{x}_k$ can be computed as function of the initial state x_k and the inputs $\bar{u}_k$ of the block, as

$$\bar{x}_k = \Gamma_{x,k}x_k + \Gamma_{u,k}\bar{u}_k + \Gamma_{b,k} \tag{2}$$

where

$$\Gamma_{x,k} = \begin{bmatrix}I\\A_k\\A_{k+1}A_k\end{bmatrix}, \quad \Gamma_{u,k} = \begin{bmatrix}0 & 0 & 0\\ B_k & 0 & 0\\ A_{k+1}B_k & B_{k+1} & 0\end{bmatrix},$$
$$\Gamma_{b,k} = \begin{bmatrix}0\\b_k\\A_{k+1}b_k + b_{k+1}\end{bmatrix}. \tag{3}$$

The initial state of the following block, $x_{k+N_c} = x_{k+3}$, can be computed as a function of the initial state x_k and the inputs $\bar{u}_k$ of the current block as

$$x_{k+N_c} = \bar{A}_k x_k + \bar{B}_k\bar{u}_k + \bar{b}_k, \tag{4}$$

where the matrices

$$\begin{aligned}\bar{A}_k &= A_{k+2}A_{k+1}A_k\\ \bar{B}_k &= \begin{bmatrix}A_{k+2}A_{k+1}B_k & A_{k+2}B_{k+1} & B_{k+2}\end{bmatrix}\\ \bar{b}_k &= A_{k+2}(A_{k+1}b_k + b_{k+1}) + b_{k+2}\end{aligned} \tag{5}$$

can be seen as an extra block-row of the matrices $\Gamma_{x,k}$, $\Gamma_{u,k}$ and $\Gamma_{b,k}$. Equation (4) is the state-space equation of a linear OCP with horizon $N_p = 2$, that has the same state vector size n_x and a larger input vector size $N_c n_u$. By using (2), $\bar{\phi}_k$ can be rewritten as function of the initial state x_k and the inputs $\bar{u}_k$ of each block

$$\bar{\phi}_k = \frac{1}{2}\begin{bmatrix}\bar{u}_k\\x_k\end{bmatrix}^T\begin{bmatrix}H_{R,k} & H_{S,k}\\ H_{S,k}^T & H_{Q,k}\end{bmatrix}\begin{bmatrix}\bar{u}_k\\x_k\end{bmatrix} + \begin{bmatrix}g_{r,k}\\g_{q,k}\end{bmatrix}^T\begin{bmatrix}\bar{u}_k\\x_k\end{bmatrix}$$

where

$$\begin{aligned}H_{R,k} &= \Gamma_{u,k}^T\bar{Q}_k\Gamma_{u,k} + \Gamma_{u,k}^T\bar{S}_k^T + \bar{S}_k\Gamma_{u,k} + \bar{R}_k\\ H_{S,k} &= \Gamma_{u,k}^T\bar{Q}_k\Gamma_{x,k} + \bar{S}_k\Gamma_{x,k}\\ H_{Q,k} &= \Gamma_{x,k}^T\bar{Q}_k\Gamma_{x,k}\\ g_{r,k} &= \Gamma_{u,k}^T\bar{Q}_k\Gamma_{b,k} + \bar{S}_k\Gamma_{b,k} + \Gamma_{u,k}^T\bar{q}_k + \bar{r}_k\\ g_{q,k} &= \Gamma_{x,k}^T\bar{Q}_k\Gamma_{b,k} + \Gamma_{x,k}^T\bar{q}_k\end{aligned}$$

About constraints, input bounds are unaffected by the partial condensing procedure, while both general constraints and state bounds get in the form of general constraints.

By inserting (2) into the expression for a general constraint

$$\bar{d}_k^{\mathrm{l}} \leq \bar{C}_k \bar{x}_k + \bar{D}_k \bar{u}_k \leq \bar{d}_k^{\mathrm{u}}$$

the result is still a general constraint

$$\bar{d}_k^{\mathrm{l}} - \bar{C}_k \Gamma_{b,k} \leq \bar{C}_k \Gamma_{x,k} x_k + (\bar{D}_k + \bar{C}_k \Gamma_{u,k}) \bar{u}_k \leq \bar{d}_k^{\mathrm{u}} - \bar{C}_k \Gamma_{b,k}.$$

By inserting (2) into the expression for a state bound

$$\bar{x}_k^{\mathrm{l}} \leq \bar{x}_k \leq \bar{x}_k^{\mathrm{u}}$$

the result is no longer a state bound, but rather a general constraint

$$\bar{x}_k^{\mathrm{l}} - \Gamma_{b,k} \leq \Gamma_{x,k} x_k + \Gamma_{u,k} \bar{u}_k \leq \bar{x}_k^{\mathrm{u}} - \Gamma_{b,k}.$$

However, the matrix $\Gamma_{u,k}$ is block-triangular, and this can be exploited in tailored algorithms, as shown in Section VII.

In the following, only input bounds and state bounds of the original OCP are considered.

IV. Partial Condensing Algorithms

As already noticed, the expression for the state-space system matrices $\bar{A}_k$, $\bar{B}_k$ and $\bar{b}_k$ can be obtained considering them as an extra block-row in the matrices $\Gamma_{x,k}$, $\Gamma_{u,k}$ and $\Gamma_{b,k}$ respectively. The algorithm to compute all these matrices can be easily obtained by inspecting the expressions in (3) and (5), and it has a computational cost of about $2N_c n_x^3 + N_c^2 n_x^2 n_u$ flops.

In principle, any Hessian condensing algorithm can be employed to obtain the expressions for the cost function matrices of the partially condensed OCP, by condensing the N_p sub-problems. However, since most of these sub-problems are in the MHE form, one should employ an algorithm performing well in the case where the initial state is an optimization variable.

In [8], three different Hessian condensing algorithms are compared. In the MHE case, an algorithm proposed therein is found to be the best choice for nearly all problem sizes. The algorithm employs a recursion that has analogies with the backward Riccati recursion to simultaneously compute all quantities needed in the construction of the condensed Hessian matrix and gradient vector

$$H_k = \begin{bmatrix} H_{R,k} & H_{S,k} \\ H_{S,k}^T & H_{Q,k} \end{bmatrix}, \quad g_k = \begin{bmatrix} g_{r,k} \\ g_{q,k} \end{bmatrix},$$

for the cost function of the condensed OCP. Furthermore, the algorithm can be tailored to the case of terminal cost equal to zero (i.e. $Q_{k+N_c} = 0$ and $q_{k+N_c} = 0$) which nearly is a reduction by one stage of the horizon length N_c.

The algorithm is summarized in Algorithm 1, where (to keep the notation lighter) the stages of the OCP sub-problem are indicated as ranging between 0 to N_c, and the index k is dropped. If N_c stages are condensed, the computational cost of the algorithm is of about $N_c^2 n_x n_u^2 + \frac{7}{3} N_c n_x^3$ flops. In case of state bounds in the original OCP, the constraint matrices relative to x_k and $\bar{u}_k$ ($\Gamma_{x,k}$ and $\Gamma_{u,k}$, respectively) are available at no additional computational cost.

Algorithm 1 Computation of H_Q, H_S, H_R, g_q, g_r, assuming $Q_N = 0$ and $q_N = 0$

```
Require:
    $\Gamma_x$, $\Gamma_u$, $\Gamma_b$
1:  $\begin{bmatrix} D_{N_c-1} & & \\ M_{N_c-1}^T & P_{N_c-1} & \\ m_{N_c-1}^T & p_{N_c-1}^T & * \end{bmatrix} \leftarrow \begin{bmatrix} R_{N_c-1} & & \\ S_{N_c-1}^T & Q_{N_c-1} & \\ r_{N_c-1}^T & q_{N_c-1}^T & * \end{bmatrix}$
2:  $H_R[N_c-1, N_c-1] \leftarrow D_{N_c-1}$
3:  $H_R[N_c-1, 0:N_c-2] \leftarrow M_{N_c-1} \cdot \Gamma_u[N_c-1, 0:N_c-2]$
4:  $H_S[N_c-1] \leftarrow M_{N_c-1} \cdot \Gamma_x[N_c-1]$
5:  $g_r[N_c-1] \leftarrow m_{N_c-1} + M_{N_c-1} \cdot \Gamma_b[N_c-1]$
6:  for $i \leftarrow N_c-2, \ldots, 0$ do
7:      $\begin{bmatrix} \mathcal{L}_{00} & \\ \mathcal{L}_{10} & * \end{bmatrix} \leftarrow \begin{bmatrix} P_{i+1} & \\ p_{i+1}^T & * \end{bmatrix}^{1/2}$
8:      $\mathcal{A}^T \mathcal{L} \leftarrow \begin{bmatrix} B_i^T \\ A_i^T \\ b_i^T \end{bmatrix} \cdot \mathcal{L}_{00} + \begin{bmatrix} \\ \\ \mathcal{L}_{10} \end{bmatrix}$
9:      $\begin{bmatrix} D_i & & \\ M_i^T & P_i & \\ m_i^T & p_i^T & * \end{bmatrix} \leftarrow \begin{bmatrix} R_i & & \\ S_i^T & Q_i & \\ r_i^T & q_i^T & * \end{bmatrix} + (\mathcal{A}^T \mathcal{L}) \cdot (\mathcal{A}^T \mathcal{L})^T$
10:     $H_R[i,i] \leftarrow D_i$
11:     $H_R[i, 0:i-1] \leftarrow M_i \cdot \Gamma_u[i, 0:i-1]$
12:     $H_S[i] \leftarrow M_i \cdot \Gamma_x[i]$
13:     $g_r[i] \leftarrow m_i + M_i \cdot \Gamma_b[i]$
14: end for
15: $H_Q \leftarrow P_0$
16: $g_q \leftarrow p_0$
```

V. Choice of N_p: theoretical considerations

This section investigates the choice of the optimal N_p based on the flop count of algorithms. The backward Riccati recursion in [10] and an iteration of the Riccati-based IPM in [9] are considered, and their computational cost is modeled considering only terms cubic in the number of stage variables n_x and n_u. Both algorithms are implemented such that the number of states and inputs can vary stage-wise: therefore N_p is not constrained to be an integer divisor of N.

A. Backward Riccati recursion case

The flop count of the Riccati recursion in [10] is

$$\left(\tfrac{1}{3} n_x^3\right) + (N-1)\left(\tfrac{7}{3} n_x^3 + 4 n_x^2 n_u + 2 n_x n_u^2 + \tfrac{1}{3} n_u^3\right) + \\ + \left(n_x^2 n_u + n_x n_u^2 + \tfrac{1}{3} n_u^3\right) \quad (6)$$

where terms due to the last, middle and first stages are grouped together.

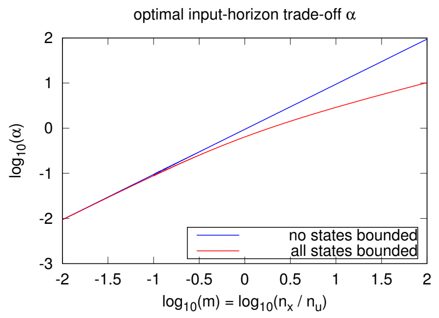
Assumption 1: The flop count can be approximated as

$$N\left(\tfrac{7}{3} n_x^3 + 4 n_x^2 n_u + 2 n_x n_u^2 + \tfrac{1}{3} n_u^3\right) = \\ = n_x^3 N\left(\tfrac{1}{3m^3} + \tfrac{2}{m^2} + \tfrac{4}{m} + \tfrac{7}{3}\right)$$

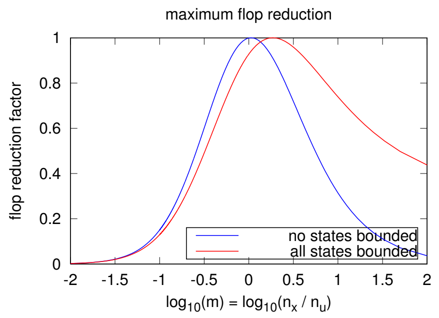
(i.e. assuming that the first and last stages together count for a middle stage) where

$$m = \frac{n_x}{n_u}$$

is the ratio between the number of states and inputs.



(a) Optimal α.



(b) Maximum flop reduction factor $f(m)$.

Fig. 1: Optimal input-horizon trade-off α and maximum flop reduction factor as function of $m = \frac{n_x}{n_u}$ for an IPM iteration, for the extreme case of no and all states bounded. The Riccati recursion alone corresponds to the no-states-bounded case.

Assumption 2: The input vector size and the horizon length can be traded off continuously. This means that the horizon length in the partially condensed OCP can be chosen as $N_p(\alpha) = \frac{N}{\alpha}$ for some $\alpha \in \mathbb{R}$, $\alpha > 0$, and consequently the input vector size in the partially condensed OCP is αn_u (i.e. α is the continuous counterpart of the block size N_c).

As a consequence, the computational cost to solve the partially condensed OCP using the backward Riccati recursion is of

$$n_x^3 \frac{N}{\alpha} \left(\tfrac{1}{3m^3}\alpha^3 + \tfrac{2}{m^2}\alpha^2 + \tfrac{4}{m}\alpha + \tfrac{7}{3} \right) \tag{7}$$

flops, and it is a function of α. Notice that $\alpha > 1$ corresponds to a reduction of the horizon length (the case considered in this paper), while $\alpha < 1$ corresponds to an increase of the horizon length (the case not considered). The minimizer of the function can be found by setting its derivative to zero. For $\alpha > 0$, the minimum of this function is attained at

$$\alpha = 0.94224m \tag{8}$$

and it is a (linear) function of the ratio m only, depicted in blue for $0.01 < m < 100$ in Figure 1a. Since $\alpha \approx m$, the maximum flop reduction is obtained by reducing the horizon length for $m < 1$ and by increasing it for $m > 1$.

The approximate maximum flop reduction factor is

$$f(m) = \frac{n_x^3 \frac{N}{\alpha} \left(\frac{1}{3}\frac{1}{m^3}\alpha^3 + 2\frac{1}{m^2}\alpha^2 + 4\frac{1}{m}\alpha + \frac{7}{3} \right)}{n_x^3 N \left(\frac{1}{3}\frac{1}{m^3} + 2\frac{1}{m^2} + 4\frac{1}{m} + \frac{7}{3} \right)} = \\ = \frac{8.6568m^2}{\left(\frac{7}{3}m^3 + 4m^2 + 2m + \frac{1}{3} \right)} \tag{9}$$

and it is a function of m only.

The function $f(m)$ is plotted in blue for $0.01 < m < 100$ in Figure 1b. Note that, for $m \approx 1$ (i.e. $n_x \approx n_u$) there is no flop reduction (i.e. the OCP has already the optimal sparsity level). The flop reduction gets significant only if n_x and n_u are of rather different size.

In practice, the best value of N_p is also affected by the value of N, since a small value of N limits the number of possible choices for the actual (integer) N_p.

B. IPM iteration case

In case there are only bounds and no general constraints in an OCP, the backward Riccati recursion is the computationally most expensive step at each iteration of an IPM. However, in case of general constraints, their contribution to the Hessian update at each IPM iteration is also a costly operation. In the general case, it has a computational cost of

$$(n_x^2 n_g) + (N-1)(n_x + n_u)^2 n_g + (n_u^2 n_g)$$

flops, where n_g is the number of general constraints at each OCP stage.

In the partially condensed OCP, the state bounds at the last stage remain in the form of state bounds, while at the intermediate stages they have the constraint matrix

$$\begin{bmatrix} D_n \\ C_n \end{bmatrix} = \begin{bmatrix} \Gamma_{u,n} \\ \Gamma_{x,n} \end{bmatrix}$$

that has size $(\alpha n_u + n_x) \times (\alpha n_g)$. At the first stage, the constraint matrix is in the same form in the MHE case, while it reduces to $D_0 = \Gamma_{u,0}$ in the MPC case. In the worst case (all states are bounded), it holds $n_g = n_x$, and therefore the computational cost of the Hessian update at each IPM iteration is of about (using Assumptions 1 and 2)

$$n_x^3 \frac{N}{\alpha} \left(\tfrac{1}{m^2}\alpha^3 + \tfrac{2}{m}\alpha^2 + \alpha \right) \tag{10}$$

flops. The approximate total computational cost per IPM iteration is given by the sum of (7) and (10), i.e.

$$n_x^3 \frac{N}{\alpha} \left((\tfrac{1}{3m^3} + \tfrac{1}{m^2})\alpha^3 + (\tfrac{2}{m^2} + \tfrac{2}{m})\alpha^2 + (\tfrac{4}{m} + 1)\alpha + \tfrac{7}{3} \right) \tag{11}$$

flops. This expression is more complex than the one in (7) and therefore it is not possible to obtain simple expressions for the optimal α value and the maximum flop reduction factor similar to (8) and (9). However, for each value of m, the flop count expression (11) can be minimized numerically, and the results are plotted in red in Figures 1a and 1b. Figure 1a shows that the optimal α is no longer a linear function of m, and for large values of m the optimal block size is smaller than in the backward Riccati recursion case. Figure 1b shows that, in case of $m > 1$ (i.e. $n_x > n_u$), the maximum flop reduction that can be attained is much lower than in the backward Riccati recursion case. E.g. for $m = 100$ (i.e. $n_x = 100n_u$), the backward Riccati recursion has a flop reduction factor of 0.036 (or about 27x speed-up), while the IPM iteration has a flop reduction factor of 0.44 (or slightly more than 2x speed-up) if all states are bounded.

This shows that an analysis based on the backward Riccati recursion alone may not accurately describe the behavior of a Riccati-based IPM, in case there are many state bounds in the original OCP formulation. In case not all states are bounded in the original OCP formulation, the attainable flop reduction is in between the two extremes, and interpolates linearly with the number of state bounds per OCP stage.

VI. Choice of N_p: linear algebra performance

This section investigates how the optimal choice of N_p is affected by the performance of the linear algebra routines.

The computational performance of linear algebra routines generally depends on implementation choices and matrix sizes. In particular, for linear algebra routines in HPMPC [6], [9], the performance increases quickly for matrix sizes up to about 20-50, and then it stabilizes close to the peak throughput for matrices of size up to about 300-400, before slightly decreasing due to lack of blocking for cache. The exact intervals depend on the Instruction Set Architecture (ISA) and on the specific linear algebra routine [8]. Linear algebra routines can attain only a small fraction of the peak throughput for very small matrices, affecting solver performance in case of small-scale OCPs. Partial condensing is therefore expected to further improve performance beyond the flop reduction, due to the use of larger matrices.

Figure 2 investigates this through an example. Two unconstrained MPC problems are considered, one small with $n_x = 4$ and $n_u = 1$, and one large with $n_x = 40$ and $n_u = 10$. Both problems have the same horizon length $N = 20$, and the same ratio $m = {}^{n_x}\!/_{n_u} = 4$. Employing a backward Riccati recursion, the formula (8) gives an optimal (real) value of N_p equal to 5.3, and the formula (9) a flop reduction factor of 0.625.

In figure 2a it is plotted the flop count for the small MPC problem for different values of N_p when Assumptions 1 and 2 are dropped. The picture for the large MPC problem would be identical, with the y axes scaled exactly by $1000 = 10^3$ (the computational cost in (6) is cubic in n_x and n_u). The optimal value of N_p is equal to 4, for a flop count reduction of 0.559. Starting from an N_p value equal to $N = 20$ on the right side of the plots, the flop count decreases steadily as N_p decreases until it reaches a minimum. For smaller values of N_p, the flop count quickly increases, and for $N_p = 1$ it is larger than the one of the original MPC problem with $N = 20$. Figures 2b and2c plot the running times. The red curves are the results when the linear algebra routines are provided by the C99 target ISA in HPMPC (optimal `dgemm` kernel size of 4×4). For the large scale problem, the plot in Figure 2c closely resembles the figures from the flop count. However, for the small scale problem, the plot in Figure 2b is rather different: the best value for N_p is 2, and for $N_p = 1$ the solution time is much lower than the original MPC problem. This is due to the fact that for such a small problem size, there is a big advantage in replacing many operations on very small matrices (smaller than the `dgemm` kernel size) with fewer operations with larger matrices, where linear algebra routines can attain a better performance (even if the flop count increases). This trend gets even more clear for the blue curves, where the linear algebra routines are provided by the AVX2 target ISA in HPMPC (optimal `dgemm` kernel size ok 12×4).

These plots also show that, for small-scale OCPs, the availability of powerful ISAs makes little difference on the solution time of algorithms (Figure 2b), while it makes big difference for larger-scale OCPs (Figure 2c). The use of partial condensing can help to better exploit powerful ISAs also for small-scale OCPs.

VII. Efficient Handling of State Bounds

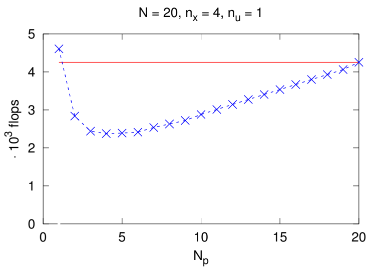
Partial condensing can be employed as a preparation step before the call to a Riccati-based IPM solver in order to change the size of the linear MPC problem. Two IPM versions are tested. The one has not been tailored to take advantage of the special structure of the matrix Γ_u, and therefore it is processed as the constraint matrix of general constraints. The other has been tailored to exploit the block-triangular structure of the matrix Γ_u.

The test problem is the mass-spring problem [17] with $N = 20$, $n_x = 20$ and $n_u = 2$. In the original MPC problem, all states are bounded: this is the worst case for the application of partial condensing. The value of $m = {}^{n_x}\!/_{n_u}$ is equal to 10, which in case all states are bounded gives an optimal value of α of 2.9 (i.e. optimal (real) $N_p \approx 6.9$) and a flop reduction factor of about 0.74. As a comparison, in case there are no state bounds the same value of m gives an optimal value of α equal to 9.4 (i.e. optimal (real) $N_p \approx 2.1$) and a flop reduction factor of about 0.31 (i.e. a much larger flop reduction).

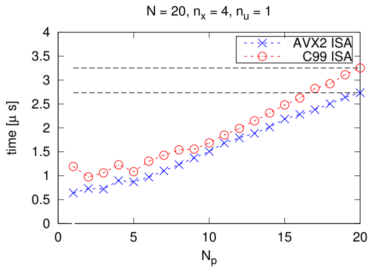
The plot in Figure 3a shows the actual flop count for an IPM iteration, computed dropping Assumptions 1 and 2. The minimum flop count is attained for $N_p = 6$, and the flop reduction factor is 0.64. Figure 3b shows the solution time when the state bounds for the original OCP are processed as general constraints for the partially condensed OCP. The cost of performing partial condensing on-line (as e.g. in an NMPC framework) is small and increases steadily as N_p decreases. The solution time for 10 IPM iterations resembles the flop count picture, with the best N_p equal to 4 for a solution time reduction factor of 0.72. If the partial condensing cost is added, the best N_p is equal to 5 for a smaller solution time reduction factor of 0.81.

Figure 3c shows the solution time when the block-triangular structure of Γ_u is exploited while processing the constraint matrix associated with the partial condensing of state bounds: for small values of N_p there is clearly an improvement. However, for this specific problem size, there is very little improvement in the solution time reduction for the optimal N_p, that is equal to 4 for a solution time reduction factor equal to 0.78.

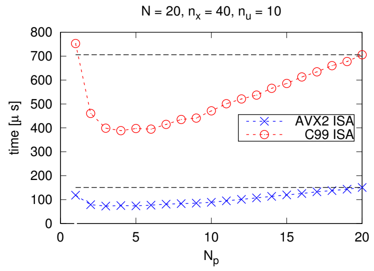
Therefore, the exploitation of the block-triangular structure of the matrix Γ_u is advantageous only if the block size N_c is large, and this is the case if both m and N are large, i.e. for an OCP with many more states than inputs and a long horizon.



(a) Small-scale, flops.

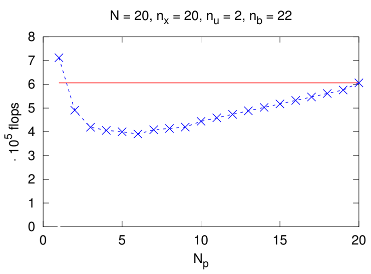


(b) Small-scale, time.

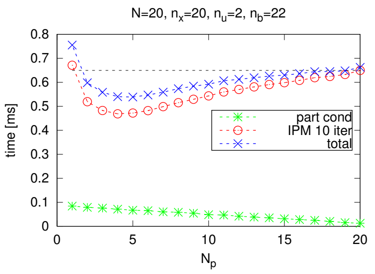


(c) Large-scale, time.

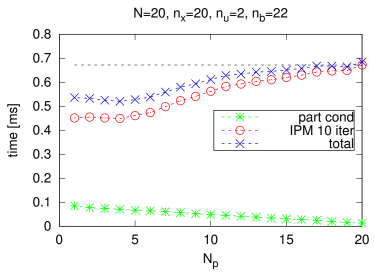
Fig. 2: Flop count / solution time for the reformulations of a small and a large unconstrained MPC problem obtained using partial condensing for different values of N_p ($N_p = 20$ is fully sparse, $N_p = 1$ is fully condensed). The test processor is Intel Core i7 4800MQ, C99 and AVX2 ISAs in HPMPC.



(a) IPM, flops per iteration.



(b) IPM, time, general.



(c) IPM, time, tailored.

Fig. 3: Flop count / solution time for the reformulations of a MPC problem, obtained using partial condensing for different values of N_p. The MPC problem is solved using a Riccati-based IPM, with state bound processed as general constraints (center) or as tailored constraints for partial condensing (right). Test processor is Intel Core i7 3520M, AVX ISA in HPMPC.

VIII. Conclusion

This paper investigates the use of partial condensing as a technique to speed-up the solution of linear OCPs in an NMPC framework by using a Riccati-based IPM. Thanks to the use of a Hessian condensing algorithm specially tailored to partial condensing, the partial condensing algorithm itself accounts for a small fraction of the total solution time. For small-scale OCPs, partial condensing decreases the solution times well beyond the flop reduction, replacing many operations on small matrices with few operations on large matrices (linear algebra performs better as the matrix size increases). In this framework, partial condensing allows to better exploit hardware throughput also for small-scale OCPs.

References

[1] J. Andersson. *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, Faculty of Engineering Science, K.U. Leuven, Heverlee, Belgium, 2013.

[2] D. Axehill. Controlling the level of sparsity in MPC. *Systems & Control Letters*, 76:1–7, 2015.

[3] D. Axehill and M. Morari. An alternative use of the Riccati recursion for efficient optimization. *Systems & Control Letters*, 61:37–40, 2012.

[4] A. Domahidi, A. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones. Efficient interior point methods for multistage problems arising in receding horizon control. In *IEEE Conference on Decision and Control (CDC)*, pages 668 – 674, Maui, HI, USA, December 2012.

[5] J. V. Frasch, S. Sager, and M. Diehl. A parallel quadratic programming method for dynamic optimization problems. *Mathematical Programming Computations*, 7(3):289–329, 2015.

[6] G. Frison. HPMPC. https://github.com/giaf/hpmpc.git.

[7] G. Frison. Numerical methods for model predictive control. Master's thesis, Department of Informatics and Mathematical Modelling, Technical University of Denmark, Kgs. Lyngby, Denmark, 2012.

[8] G. Frison. *Algorithms and Methods for High-Performance Model Predictive Control*. PhD thesis, Department of Applied Mathematics and Computer Science, Technical University of Denmark, Kgs. Lyngby, Denmark, 2016.

[9] G. Frison, H. H. B. Sørensen, B. Dammann, and J. B. Jørgensen. High-performance small-scale solvers for linear model predictive control. In *IEEE European Control Conference*, pages 128–133. IEEE, 2014.

[10] G. Frison and J. B. Jørgensen. Efficient implementation of the Riccati recursion for solving linear-quadratic control problems. In *IEEE Multi-conference on Systems and Control*, pages 1117–1122. IEEE, 2013.

[11] G. Frison and J. B. Jørgensen. A fast condensing method for solution of linear-quadratic control problems. In *IEEE Conference on Decision and Control*, pages 7715–7720. IEEE, 2013.

[12] J. B. Jørgensen, J. B. Rawlings, and S. B. Jørgensen. Numerical methods for large-scale moving horizon estimation and control. In *Int. Symposium on Dynamics and Control Process Systems (DYCOPS)*, volume 7, 2004.

[13] D. Kouzoupis, R. Quirynen, J. V. Frasch, and M. Diehl. Block condensing for fast nonlinear MPC with the dual Newton strategy. In *5th IFAC Conference on Nonlinear Model Predictive Control*, 2015.

[14] C. V. Rao, S. J. Wright, and J. B. Rawlings. Application of interior-point methods to model predictive control. *Journal of optimization theory and applications*, 99:723–757, 1998.

[15] M. C. Steinbach. *A structured interior point SQP method for nonlinear optimal control problems*. Springer, 1994.

[16] M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl. Auto-generated algorithms for nonlinear model predictive control on long and on short horizons. In *IEEE Conference on Decision and Control*, pages 5113–5118. IEEE, 2013.

[17] Y. Wang and S. Boyd. Fast model predictive control using online optimization. In *IFAC World Congress*, pages 6974 – 6997, Seoul, July 2008.