2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

Parallel Move Blocking Model Predictive Control

Stefano Longo, Eric C. Kerrigan, Keck Voon Ling and George A. Constantinides

Abstract— **This paper proposes the use of parallel computing architectures (multi-core, FPGA, GPU) to implement a parallel move blocking Model Predictive Control (MPC) algorithm where multiple, but smaller optimization problems are solved simultaneously. Since these problems are solved in parallel, the computational delay is reduced when compared to standard MPC. This allows for faster sampling that can outperform, in terms of closed-loop cost, a standard MPC formulation. Feasibility and stability are guaranteed by an appropriate selection of so-called blocking matrices.**

I. Introduction

The frustrating drawback of Model Predictive Control (MPC) is the computational time required to solve the online optimization problem within the sampling period. This precludes the application of MPC techniques for systems where the platform running the control algorithm is not sufficiently powerful. Among the proposed solutions for faster MPC implementation, there is the practically and widely used Move Blocking (MB) strategy [1]–[4]. In MB, the predicted control trajectory is forced to remain constant over some steps, hence the degrees of freedom are reduced by fixing some optimization variables. Many MB strategies, despite working well in practice, lack feasibility and/or stability proofs (which are normally enforced by terminal constraints [5]). In [4] it is shown that it is possible to guarantee feasibility and stability by using a time-varying blocking strategy. Recently, in [6]–[8], a generalized approach for enforcing feasibility of MB MPC laws that are least-restrictive has been studied.

Besides the established approaches for faster MPC applications, there is a newly emerging concept that tackles the speeding up process from a completely new angle. Instead of solving one large optimization problem, this is divided into smaller subproblems that are solved sequentially [9] or simultaneously [10] by exploiting multi-core processors or modern platforms like FPGAs or GPUs that inherently allow for parallelism [11], [12]. For example, in an FPGA, algorithms can be efficiently pipelined in order that hardware blocks (such as adders or multipliers used for vector dot products) are continually used. This increase in the number of operations performed by pipelining is not due to an increase in hardware resources but is a result of its efficient usage.

In this paper, we propose the Parallel Move Blocking (PMB) algorithm, which is a way to combine the strategy of MB and the opportunity provided by parallel computing platforms. The contributions of the paper are both theoretical and practical and are: i) a feasibility and stability conditions that can be satisfied a priori (i.e. a feasible input sequence at a particular time instant implies a feasible input sequence for the next time step ad infinitum); ii) a potential improvement in performance when compared to a 'standard' MB implementation because, from the set of solutions of different MB problems, we can always choose the one with the smallest predicted cost.

PMB is inspired by multiplexed MPC [9] and its parallel variant called channel-hopping MPC [10]. Whereas multiplexed MPC and channel-hopping MPC are applicable to n_u-input systems ($n_u > 1$), PMB is applicable even if $n_u = 1$. The applicability of PMB can be interpreted in two ways. One is that if the processor is not fast enough to execute a standard MPC algorithm, then it might be capable to do so with PMB without compromising the feasibility and stability guarantees. The other one is that PMB allows for the selection of shorter sampling times, since the online optimization is smaller and faster to solve (it is well-known that faster sampling gives better disturbance rejection). In other words, we can trade optimality with sampling time.

We first show how to set up the MB problems that will be solved online and in parallel. The idea is that the first input from the sequence of inputs that results in the lowest open-loop cost is used and the problems are solved again at the next sampling instant. The feasibility and stability of the proposed scheme is guaranteed by appropriate choices of 'blocking matrices'. We carried out our design in the sampled-data framework, from a continuous-time plant with an associated quadratic, finite horizon cost function in the same spirit as [13]. We find the equivalent discrete-time representation of the plant and the cost and set up the optimal control problem in terms of the sampling time. The relation between sampling time and closed-loop cost for constrained problems is not straightforward and this is investigated using simulation examples.

II. Parallel Move Blocking MPC

Consider the following discrete-time LTI plant model

$$x(k+1) = A_h x(k) + B_h u(k), \quad k \in \mathbb{N}_0, \tag{1}$$

where $x(k) \in \mathbb{R}^{n_x}$, $u(k) \in \mathbb{R}^{n_u}$ and (A_h, B_h) is a stabilizable pair. This plant can be identified from data or obtained from a discretization of a continuous-time plant with sampling period h. We assume full state feedback and we suppose that constraints exist on the input moves and on the states. They are represented by $u(k) \in \mathbb{U} \subseteq \mathbb{R}^{n_u}$ and $x(k) \in \mathbb{X}_h \subseteq \mathbb{R}^{n_x}$, respectively, where $\mathbb{U}$ and $\mathbb{X}_h$ are compact polyhedral sets with the origin in their interior. The set $\mathbb{X}_h$ has to be a function of the sampling period h, as shown in [13], to guarantee constraint satisfaction between sample instants. Let $x(k)$ or x be the state measured at k and x_i, $i \in \mathbb{N}_0$, be the predicted state at $k+i$ for the given

S. Longo, E. C. Kerrigan and G. A. Constantinides are with the Department of Electrical and Electronic Engineering (E. C. Kerrigan is also with the Department of Aeronautics), Imperial College, London, SW7 2AZ, UK {s.longo, e.kerrigan, g.constantinides}@imperial.ac.uk

K. V. Ling is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, 639798 ekvling@ntu.edu.sg

state $x(k)$, input sequence $(u_0, u_1, \ldots, u_{i-1})$ and prediction horizon $N \in \mathbb{N}$. Also, let the set $\mathcal{X}(K_h)$ be the maximum constraint-admissible positively invariant set for the system in (1) subject to control $u(k) = K_h x(k)$ [4, Definition 1], where K_h is a pre-specified stabilizing feedback controller such that $x(k) \in \mathcal{X}(K_h)$ implies $K_h x(k) \in \mathbb{U}$ and $(A_h + B_h K_h)x(k) \in \mathcal{X}(K_h)$. The vector with the predicted input moves is defined as

$$\boldsymbol{u} := \begin{bmatrix} u_0' & u_1' & \cdots & u_{N-1}' \end{bmatrix}', \quad \boldsymbol{u} \in \mathbb{R}^{Nn_u}. \tag{2}$$

The *standard MPC* problem is defined as

$$\begin{aligned} \bar{J}_h^*(x) &:= \min_{\boldsymbol{u}} \left\{ x_N' P_h x_N + \sum_{i=0}^{N-1} \begin{bmatrix} x_i \\ u_i \end{bmatrix}' Q_h \begin{bmatrix} x_i \\ u_i \end{bmatrix} \right\}, \\ \text{s.t.} \quad & x_i \in \mathbb{X}_h, \quad u_i \in \mathbb{U}, \quad x_N \in \mathcal{X}(K_h), \\ & x_{i+1} = A_h x_i + B_h u_i, \quad x_0 = x, \\ & \forall i \in \{0, 1, \ldots, N-1\}, \end{aligned} \tag{3}$$

where Q_h and P_h are weighting matrices that will be defined in Section II-A. The terminal set constraint $x_N \in \mathcal{X}(K_h)$ is needed so that recursive feasibility is achieved [4], [5]. The unique optimizer to the control problem above is defined as $\boldsymbol{u}^*(x)$.

We now convert the standard problem in (3) into one with input blocking. We do so by fixing some inputs to be constant over a number of time steps. Let $M \in \{0,1\}^{N \times \hat{N}}$ be a binary matrix, where $\hat{N} < N$, and let $\boldsymbol{u} = (M \otimes I_{n_u})\hat{\boldsymbol{u}}$. Matrix M will be called the 'blocking matrix' [4]. Vector $\boldsymbol{u} \in \mathbb{R}^{Nn_u}$ is the full degree of freedom vector of control moves while $\hat{\boldsymbol{u}} := \begin{bmatrix} \hat{u}_0' & \hat{u}_1' & \cdots & \hat{u}_{\hat{N}-1}' \end{bmatrix}'$, $\hat{\boldsymbol{u}} \in \mathbb{R}^{\hat{N}n_u}$ is the reduced one ($\otimes$ is the Kronecker product). The control problem becomes

$$\begin{aligned} &\mathcal{P}(M): \\ &J_h^*(x, M) := \min_{\hat{\boldsymbol{u}}} \left\{ x_N' P_h x_N + \sum_{i=0}^{N-1} \begin{bmatrix} x_i \\ u_i \end{bmatrix}' Q_h \begin{bmatrix} x_i \\ u_i \end{bmatrix} \right\}, \\ &\text{s.t.} \quad x_i \in \mathbb{X}_h, \quad u_i \in \mathbb{U}, \quad x_N \in \mathcal{X}(K_h), \\ &\quad \boldsymbol{u} = (M \otimes I_{n_u})\hat{\boldsymbol{u}}, \\ &\quad x_{i+1} = A_h x_i + B_h u_i, \quad x_0 = x, \\ &\quad \forall i \in \{0, 1, \ldots, N-1\}. \end{aligned} \tag{4}$$

Problem $\mathcal{P}(M)$ in (4) will be denoted as the *standard MB* problem.

Now consider a hardware platform that allows parallel computation. This can be multi-core processors, FPGAs, GPUs or even independent industrial PCs or microprocessor boards (as the aim here is to demonstrate the potential advantage of such a parallel implementation). In this platform, more than one problem $\mathcal{P}(M)$ can be solved in parallel. Let $\{M_s | s = 0, 1, \ldots, S-1\}$ be a set of S different blocking matrices. Let $\hat{\boldsymbol{u}}_s^*(x)$ and $J_h^*(\cdot, M_s)$ be the optimizer and the value function, respectively, of $\mathcal{P}(M)$ with $M = M_s$. S is the number of problems that can be solved in parallel in a particular hardware implementation and it will be a limitation of the hardware (e.g. in FPGAs it would depend on the number of problems that can be pipelined [11]). In the PMB scheme proposed, S problems, each associated with blocking

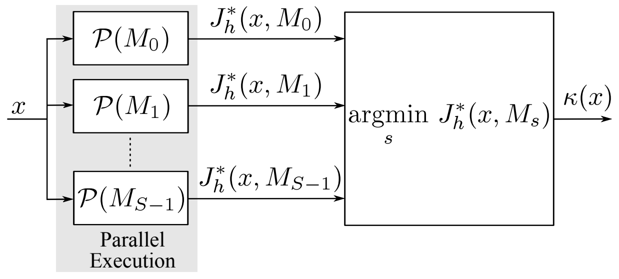

Fig. 1. Implementation of the PMB algorithm.

matrix $M_0, M_1, \ldots, M_{S-1}$, are solved simultaneously at each sampling instant. The PMB MPC law is given by

$$\kappa(x) := \begin{bmatrix} I_{n_u} & 0 & \cdots & 0 \end{bmatrix} \hat{\boldsymbol{u}}_{s^*(x)}^*(x), \tag{5}$$

where $s^*(x)$ is the index of the blocking matrix $M_{s^*(x)}$ such that

$$s^*(x) := \operatorname*{argmin}_s \; J_h^*(x, M_s). \tag{6}$$

In other words, the applied input move is the one that results in the smallest predicted cost among S optimization problems (also shown diagrammatically in Figure 1). This is indeed an improvement to standard MB because the open-loop cost can only be equal or better than the case where only one MB problem is solved. Such an improvement comes from solving parallel problems and is achieved from the efficient hardware usage in the aforementioned platforms.

A. Sampled-data formulation

It is desirable for our implementation to formulate the problem as a sampled-data one. Consider the continuous-time LTI system

$$\dot{x}(t) = A_c x(t) + B_c u(t), \tag{7}$$

where $x(t) \in \mathbb{R}^{n_x}$, $u(t) \in \mathbb{R}^{n_u}$ and (A_c, B_c) is a stabilizable pair. The control input $u(t)$ is a signal created by a sample-and-hold element for a constant sampling period h such that $u(t) = u(kh)$ for all $t \in [kh, (k+1)h)$ and k is the sampling instant (the sampling period h is removed as an argument where obvious). The system in (7) can be sampled with a periodic interval h giving the sampled-data system, for constant input u, in (1) where $A_h = e^{A_c h}$ and $B_h = \int_0^h e^{A_c \tau} d\tau B_c$. If the sampling frequency is non-pathological (i.e. A_c does not have two eigenvalues with equal real parts and imaginary part that differ by an integral multiple of $2\pi/h$, [14, Theorem 3.2.1]), then (A_c, B_c) controllable implies (A_h, B_h) reachable. This is a sufficient condition for preservation of reachability after discretization. If this condition is applied only to the unstable eigenvalues of A_c, then stabilizability is preserved after discretization.

Associated with the system in (7) consider the continuous-time, finite horizon LQ problem defined by the cost function

$$J_c := x(T_f)' P_c x(T_f) + \int_0^{T_f} \Big[x(t)' Q_{1c} x(t) + u(t)' Q_{2c} u(t) \Big] dt, \tag{8}$$

where T_f is the time horizon, $Q_{1c} = Q_{1c}' \geq 0$, $Q_{2c} = Q_{2c}' > 0$, $(Q_{1c}^{1/2}, A)$ detectable and $P_c = P_c' > 0$. These matrices

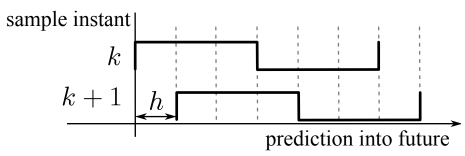

Fig. 2. In the shifted blocked sequence of predicted inputs the points where the inputs are allowed to change do not coincide.

are given to define a controller for an ideal closed-loop performance of the continuous-time model. The equivalent sampled-data cost function is

$$J_h := x(N)'P_h x(N) + \sum_{k=0}^{N-1} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}' \underbrace{\begin{bmatrix} Q_1 & Q_{12} \\ Q_{12}' & Q_2 \end{bmatrix}}_{Q_h} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}, \tag{9}$$

where N is the number of samples for the predicted horizon that, for the moment, is defined as

$$N := \left\lceil \frac{T_f}{h} \right\rceil, \tag{10}$$

$\lceil\cdot\rceil$ is the ceiling function and expressions for Q_h and P_h can be found in [15]. Matrices Q_2 and P_h are positive definite. With the assumption that the input $u(t)$ is constant over the interval $t \in [kh, (h+1)k)$ and the states $x(k)$ are sampled at the time instant $t = kh$ without computational delays, there is no approximation error between (8) and (9) [15].

III. Feasibility and stability

We will prove here that feasibility and stability can be guaranteed by particular selections of blocking matrices. In standard MPC, it is well-known that if $x \mapsto x'P_h x$ is a local Lyapunov function for the closed-loop system with controller gain K_h then stability is guaranteed [5]. The controller K_h can be chosen to be $K_h = -(B_h'P_hB_h + Q_2)^{-1}(B_h'P_hA_h + Q_{12}')$ where P_h is the discrete-equivalent of P_c, and P_c is the solution of the continuous-time algebraic Riccati equation

$$A_c'P_c + P_cA_c - P_cB_cQ_{2,c}^{-1}B_c'P_c + Q_{1,c} = 0 \tag{11}$$

(see (8), (9) and [15]). Both feasibility and stability can be proved by considering a shifted vector of input moves, at time $k+1$,

$$\tilde{\boldsymbol{u}}(x) := \begin{bmatrix}(u_1^*(x))' & \cdots & (u_{N-1}^*(x))' & (K_hx_N^*(x))'\end{bmatrix}', \tag{12}$$

where $x_N^*(x)$ is the predicted state at time $k+N$. The shifted vector $\tilde{\boldsymbol{u}}(x)$ is guaranteed to be feasible at time $k+1$ because of the constraint $x_N \in \mathcal{X}(K_h)$ and that $\mathcal{X}(K_h)$ is an invariant and constraint-admissible set. Furthermore, since the value function $\bar{J}_h^*(\cdot)$ is a Lyapunov function (due to the choice of P_h), stability is also guaranteed (this is usually called the 'shifting argument' [5]).

It is well-known [4] that, for standard MB, the shifting argument does not generally apply because, in the shifted version of $\hat{\boldsymbol{u}}$, the points in $\hat{\boldsymbol{u}}$ where the inputs u are allowed to change do not correspond to the non-shifted vector anymore. For example, Figure 2 shows a situation of a blocking scheme where the inputs are allowed to change every $3h$. Clearly, at the next sampling step, the points in the prediction horizon where the inputs change do not coincide and the shifting argument collapses.

We now show that a particular selection of blocking matrices and a redefinition of the control problem can re-establish the shifting argument for PMB. Consider a standard MB problem where the predicted input moves u_i are blocked at regular intervals and $\hat{S} \leq S$, $\hat{S} \in \mathbb{N}_0$ indicates for how many steps a predicted input is blocked. For example, if $\hat{S} = 2$ then the vector of blocked predicted inputs moves will be

$$\boldsymbol{u} := \begin{bmatrix}\hat{u}_0' & \hat{u}_0' & \hat{u}_1' & \hat{u}_1' & \cdots & \hat{u}_{\hat{N}-1}'\hat{u}_{\hat{N}-1}'\end{bmatrix}'. \tag{13}$$

Notice that for $\hat{S} = 1$ the scheme is actually the standard MPC scheme in (3). Let $\hat{T}_f$ be the desired prediction horizon and define

$$\hat{N} := \left\lceil \frac{\hat{T}_f}{\hat{S}h} \right\rceil. \tag{14}$$

At this point we redefine the number of steps in the discrete horizon as

$$N := \hat{S}\hat{N}. \tag{15}$$

(For the standard MPC, i.e. for $\hat{S} = 1$, the definitions for N in (15) and (10) are identical.) Moreover, we let $T_f := Nh$ to ensure that the time horizon is an integer multiple of h and $T_f \geq \hat{T}_f$ always.

Finally, define $\hat{S}$ blocking matrices as

$$M_s := \begin{bmatrix} \mathbf{1}_{\hat{S}-s} & 0 \\ 0 & I_{\hat{N}-1} \otimes \mathbf{1}_{\hat{S}} \end{bmatrix}, \quad s = 0, 1, \ldots, \hat{S}-1, \tag{16}$$

where $\mathbf{1}_n$ is the column vector of ones of length n. For $s = 0$, M_s is the blocking matrix of the original standard MB problem while for $0 < s \leq \hat{S} - 1$, M_s is a different 'truncated' blocking matrix. The blocking matrices as defined in (16) are always 'admissible' according to [4, Definition 3].

The final step, before presenting the result, is to define the control problems for PMB as

$$\begin{aligned} \hat{J}_h^*(x, M, \hat{S}) := \min_{\hat{\boldsymbol{u}}} & \left\{ x_N'P_{\hat{S}h}x_N + \sum_{i=0}^{N-1} \begin{bmatrix} x_i \\ u_i \end{bmatrix}' Q_h \begin{bmatrix} x_i \\ u_i \end{bmatrix} \right\}, \\ \text{s.t.} \quad & x_i \in \mathbb{X}_h, \quad u_i \in \mathbb{U}, \quad x_N \in \mathcal{X}(K_{\hat{S}h}), \\ & \boldsymbol{u} = (M \otimes I_{n_u})\hat{\boldsymbol{u}}, \\ & x_{i+1} = A_hx_i + B_hu_i, \quad x_0 = x, \\ & \forall i \in \{0, 1, \ldots, N-1\}. \end{aligned} \tag{17}$$

The difference between (4) and (17) is in the terminal weight and terminal set constraint: P_h and $\mathcal{X}(K_h)$ in (4) becomes $P_{\hat{S}h}$ and $\mathcal{X}(K_{\hat{S}h})$ in (17). The control law is defined as

$$\hat{\kappa}(x) := \begin{bmatrix} I_{n_u} & 0 & \cdots & 0 \end{bmatrix} \hat{\boldsymbol{u}}_{\hat{s}^*(x)}^*(x) \tag{18}$$

where $\hat{s}^*(x)$ is the index of the blocking matrix $M_{\hat{s}^*(x)}$ such that

$$\hat{s}^*(x) := \operatorname*{argmin}_s \hat{J}_h^*(x, M_s, \hat{S}). \tag{19}$$

Proposition 3.1: For non-pathological sampling frequencies, if the PMB scheme described above is implemented by

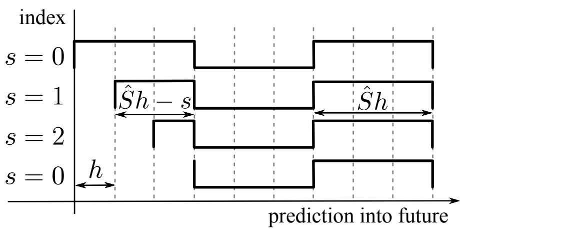

Fig. 3. By solving specific parallel problems it is possible to re-establish the shifting argument for recursive feasibility.

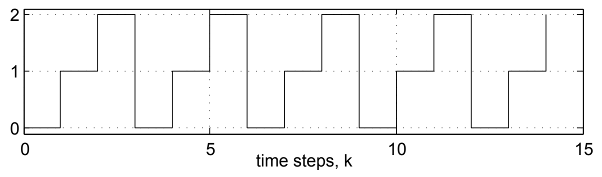

Fig. 4. An example of a fixed pattern that guarantees feasibility (and stability) for $\hat{S}=3$ if a parallel implementation is not available (s is the index of M_s).

solving $\hat{S}$ problems (17) in parallel using blocking matrices $M=M_s$, then the origin of the closed-loop system (7) in feedback with (18) is an asymptotically stable equilibrium point. The region of attraction is equal to the set of states for which there exists an s such that $\mathcal{P}(M_s)$ has a solution.

Proof. First of all consider that (A_h, B_h) is stabilizable if h is non-pathological. To prove feasibility we need to show that the vector of predicted input moves obtained at time k can be used to obtain a feasible solution at time $k+1$. From the definition of the blocking matrices in (16) we can see that the vector of blocked input moves $(M_{\hat{S}-1}\otimes I_{n_u})\hat{\boldsymbol{u}}$ is a truncated version of the vector of blocked input moves $(M_{\hat{S}-2}\otimes I_{n_u})\hat{\boldsymbol{u}}$ which is a truncated version of $(M_{\hat{S}-3}\otimes I_{n_u})\hat{\boldsymbol{u}}$ and so on, up to $(M_0\otimes I_{n_u})\hat{\boldsymbol{u}}$. The input moves $(M_0\otimes I_{n_u})\hat{\boldsymbol{u}}$ can also be considered as a truncated version of $(M_{\hat{S}-1}\otimes I_{n_u})\hat{\boldsymbol{u}}$ with the addition, at the end, of a tail which is another degree of freedom of $\hat{S}$ blocked inputs. This is illustrated, for an example with $\hat{S}=3$, in Figure 3. If the problem for M_s is feasible at time k then it will also be feasible at time $k+1$ because the controller, at the next time step, can choose the solution of the problem with blocking matrix M_{s+1} (or M_0 if $s=\hat{S}-1$), which is the one that guarantees satisfaction of all the constraints at time $k+1$. If the controller chooses the solution of another problem because it resulted in a smaller cost, then this solution must be feasible. Hence, at least one feasible solution is always available, which proves recursive feasibility.

The argument for stability follows naturally. The chosen terminal weight in the problem in (17) is $P_{\hat{S}h}$ (i.e. a discretization of P_c with sampling time $\hat{S}h$) and $x\mapsto x'P_{\hat{S}h}x$ is a local Lyapunov function. Hence, the value function $\hat{J}_h^*(\cdot, M, \hat{S})$ in (17) can be used as a Lyapunov function [5]. ∎

Remark 3.1: Note that to prove recursive feasibility (and stability), it is sufficient to solve a single problem, at each sampling instant, with blocking matrix M following the pattern $M_0, M_1, \ldots, M_{\hat{S}-1}, M_0, M_1, \ldots$, as shown in Figure 4 for $\hat{S}=3$. However, in the interval from M_0 to $M_{\hat{S}-1}$ the prediction horizon is not receding. This fixed scheme is actually a simplified version of the one proposed in [4].

IV. Performance

The degree of freedom reduction of the input moves from N to $\hat{N}$ results in a smaller optimization problem that may become computationally realizable if the original problem was not realizable. On the other hand, since the online optimization problem is now faster to solve, a shorter sampling period might be chosen to improve the system closed-loop performance and its disturbance rejection properties. Hence, with PMB, we reduce the optimization problem size at the expense of closed-loop performance, but then regain performance by reducing the sampling period. However, because of (10), smaller sampling periods also result in a potentially larger number of horizon steps N. It will be shown by simulation that, in terms of computational speed, the reduction in problem size achieved by PMB is more significant than its increase due to a larger N.

V. Illustrative example

In this section we show, with simulation examples, the potential of PMB when compared to standard MB and standard MPC. We use as a benchmark an unstable oscillator described by (7) with matrices

$$A_c=\begin{bmatrix}0.1 & 0.3\\ -0.3 & 0.1\end{bmatrix},\quad B_c=\begin{bmatrix}0\\1\end{bmatrix}, \tag{20}$$

output $y(t)=[1\ 0]x(t)$ and input constraints $-1\le u\le 1$. No state constraints are included so that the results are decoupled form the problem of guaranteeing constraint satisfaction in-between sampling instants. The associated performance measure is given by the LQ cost in (8) with matrices $Q_{1c}=1$, $Q_{2c}=0.1$ and P_c as the solution of (11). To satisfy our feasibility and stability conditions, the terminal weight has been set to $P_{\hat{S}h}$, (i.e. a discretization of P_c for a sampling time $\hat{S}h$) and the maximum positively invariant set is given by $\mathcal{X}(K_{\hat{S}h})$. The desired prediction horizon is $\hat{T}_f=15$s and therefore $T_f=Nh$ following (14) and (15) (T_f is large enough to ensure that the state at the end of the horizon is in the terminal set).

To evaluate the performance of the different schemes, the system is let to settle to the origin from the initial conditions $x(0)=\begin{bmatrix}3 & 3\end{bmatrix}'$ (the input constraints are active). Furthermore, a closed-loop cost is defined as

$$F:=\sum_{k=0}^{N_{\text{sim}}-1}\begin{bmatrix}x(k)\\u(k)\end{bmatrix}' Q_h\begin{bmatrix}x(k)\\u(k)\end{bmatrix}, \tag{21}$$

where Q_h is the performance matrix as in (9) and $N_{\text{sim}}=40$s. As a measure of computational effort, we use the time required by MATLAB (with the toolbox YALMIP [16]) to solve the Quadratic Programming (QP) problem using the built-in solver `quadprog`. For the parallel implementation, since the simulation does not run in real-time, the QP problems, at each sampling instant, are solved sequentially and the computational time considered is the longest (worst case) among them. Note that the performance of PMB can be enhanced by solving the QP with different solvers that exploit the particular structure of the problem. This will

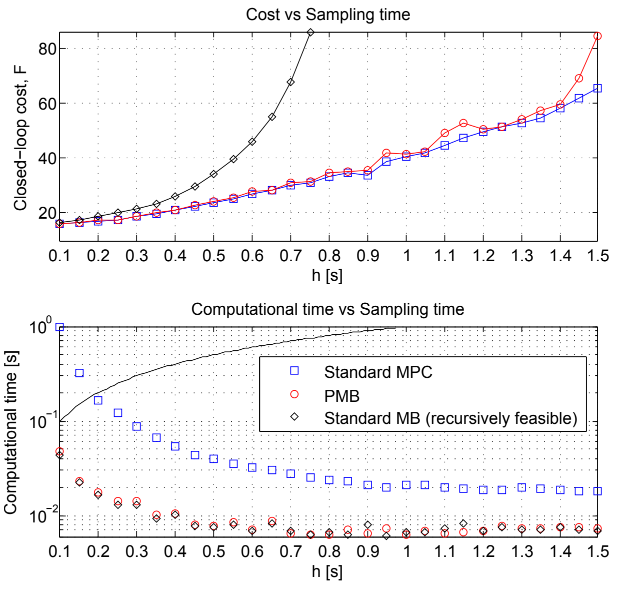

Fig. 5. Performance comparison.

be the subject of future investigation. For the moment it is sufficient to show that the computational time is reduced by the proposed scheme even when using a general purpose solver.

A. *Closed-loop cost and computational time*

In Figure 5 we compare three MPC schemes. The first one is the standard MPC defined in (3) (i.e. the sequence of predicted inputs is allowed to change every step). The second one is the proposed PMB scheme where we parallelize $\hat{S} = 5$ problems, each with a blocking matrix given by (16). Recursive feasibility is guaranteed because of Proposition 3.1. The last one is the standard MB scheme defined in (4). For fairness of comparison we implement the standard MB scheme in a way that recursive feasibility is also guaranteed. This is done, in accordance with Remark 3.1, by using a periodic sequence of blocking matrices given by (16) with $s = \tilde{s}(k)$ where

$$k \mapsto \tilde{s}(k) = (0, 1, 2, 3, 4, 0, 1, \ldots). \tag{22}$$

Although different blocking matrices are used, only one problem is solved at each sampling instant, hence this is a time-varying standard MB scheme. The top plot of Figure 5 shows the variation of the cost F as the sampling time increases. The bottom plot shows the computational time (the worst case for PMB) required to solve the QP problem. The computation was performed on a desktop PC with a 2.83 GHz CPU, 8 GB of RAM and running MATLAB's version 7.10.0.499.

The top plot of Figure 5 indicates that the cost of PMB is always slightly higher than the one for standard MPC but, by a small reduction of the sampling time, it is possible for PMB to achieve a performance equal to (or better than) the one of standard MPC. Sampling faster, however, gives less time to solve the QP problems. The bottom plot of Figure 5 shows that this is not an issue since in PMB the computational time required to solve the QP problems is always sufficiently small. In fact, we could interpret these results in a different

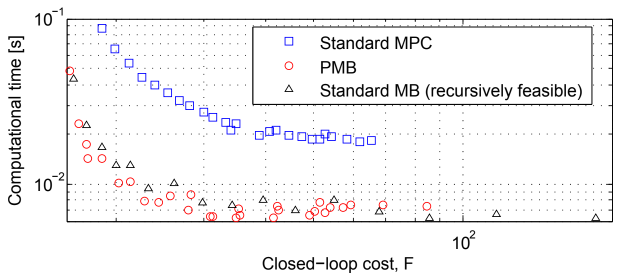

Fig. 6. Pareto-optimal tradeoff between computation time and cost.

way by assuming that there is a bound by how fast we can sample. In this particular example, the theoretical bound, shown by the line in the bottom plot of Figure 5, indicates that sampling faster than approximately 0.2s is not possible for standard MPC (while it is possible for the other two schemes) because the time to solve the QP problem is longer than the sampling time.

Let us assume now that there is a computational time bound of 0.02s by which we have to solve the QP problem. According to Figure 5, the fastest we can sample with such a bound is $h = 0.95$s for standard MPC and $h = 0.2$s for PMB. With these sampling periods, the value of the cost F for PMB is lower than the one of standard MPC, hence PMB can give us a better performance.

It is worth mentioning that the computational time for standard MB is almost identical to the one of PMB because the QP problems have the same size (this is shown in Figure 5). However, the closed-loop cost of standard MB deteriorates quickly as h increases. This is due to the sequence of blocking matrices being fixed (and not a function of the current state), as will be shown later.

The results of Figure 5 have also been presented in Figure 6 as a scatter plot to show the Pareto-optimal tradeoff of the closed-loop cost and computational time. From this plot it is possible to determine: i) what cost can be achieved for each scheme for a given computational power and ii) how much computational power is needed for each scheme to achieve a given cost.

B. *Closed-loop performance*

Choosing shorter sampling periods for PMB is possible because the QP problems can be solved faster than for standard MPC. We now show that, for a given bound on the available computational time it is possible to achieve better performance with PMB. We consider, as before, the hypothetical case where there is a computational time bound of 0.02s that does not allow sampling periods $h < 0.95$s for standard MPC (see Figure 5). If PMB is implemented, however, the sampling periods that are not allowed are $h < 0.2$s. The simulation results for standard MPC with $h = 0.95$s and PMB with $h = 0.2$s are shown in Figure 7. Clearly, the PMB scheme performs better for the given computational time bound.

The bottom plot of Figure 7 shows the evolution of the optimal index s^* which determines the blocking matrix in (16). It is interesting to notice that, every time the input constraints are active, s^* evolves following exactly the pattern of $\tilde{s}(k)$ in (22) i.e. the one that sufficiently guarantees recursive

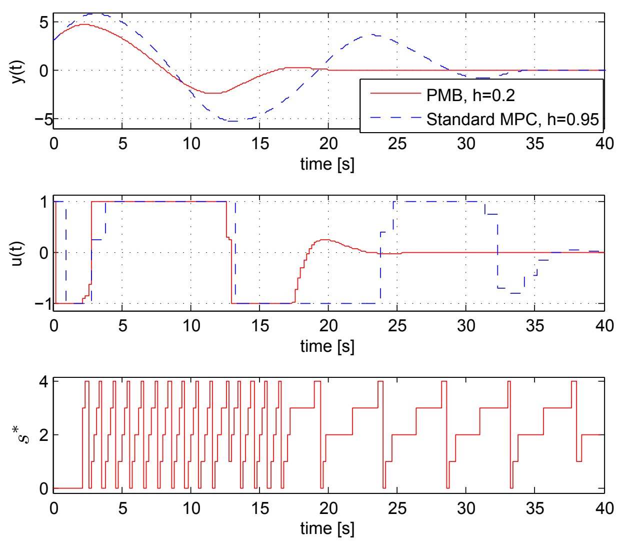

Fig. 7. Performance comparison between PMB and standard MPC for a given computational time bound of 0.02s.

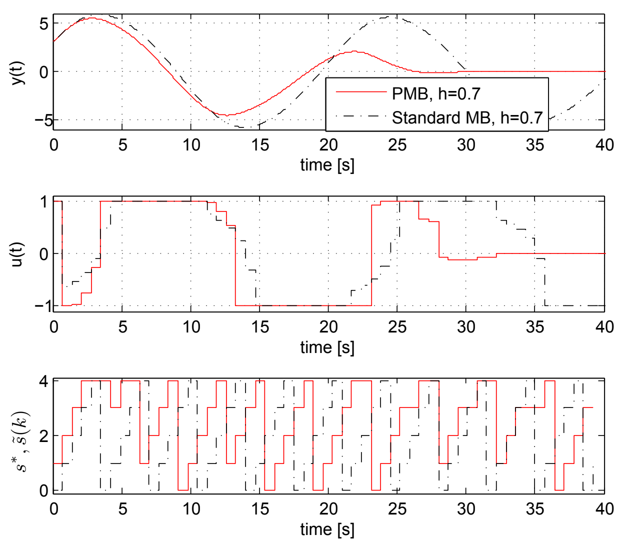

Fig. 8. Performance comparison between PMB and standard MB for a given computational time bound of 0.007s.

feasibility according to Remark 3.1. However, if the input constraints are not active, different patterns are used by the algorithm as terminal constraint satisfaction becomes easier.

Let us now consider the case where the computational time bound is reduced to 0.007s. From Figure 5, it is clear that it is not possible to implement the standard MPC. However it is possible to implement the standard MB and PMB with a sampling time of $h = 0.7$s. The simulation results, given in Figure 8, show that, also in this case, PMB performs better than standard MB. This can be explained by inspecting the evolution of the index s shown in the bottom plot of Figure 8. For standard MB, s is fixed and it is given by $\tilde{s}(k)$ (22). However, for PMB, s is the optimal index s^* (i.e. the index of the problem with the lowest open-loop cost) determined online. The results show that s^* does not always follow the fixed pattern of $\tilde{s}(k)$. The ability to choose s optimally online is the reason why PMB performs better than standard MB for equal sampling times.

VI. Conclusions

We have proposed a method to exploit parallel computing architectures in order to reduce the computational burden of solving the online optimization problem required by MPC. The theoretical advantage is that recursive feasibility (and thus stability) can be guaranteed a priori by an appropriate selection of problems to be solved in parallel. The practical advantage is that, since the optimization problems are smaller compared to standard MPC, they can be solved in less time, allowing for faster sampling. Some examples showed how the proposed PMB scheme can outperform standard MPC and standard MB in terms of closed-loop performance. Future work could include the implementation of PMB and NPS schemes in hardware on an FPGA.

VII. Acknowledgments

This research has been supported by the EPSRC grant number EP/G031576/1, EP/F041004/1 and the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement number FP7-ICT-2009-4 248940.

References

[1] J. M. Maciejowski, *Predictive control with constraints*. Prentice-Hall, 2002.

[2] P. Tøndel and T. A. Johansen, "Complexity reduction in explicit linear model predictive control," in *Proc.* 15^{th} *IFAC World Congress*, Spain, 2002.

[3] S. J. Qin and T. A. Badgewell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.

[4] R. Cagienard, P. Grieder, E. C. Kerrigan, and M. Morari, "Move blocking strategies in receding horizon control," *J. Process Control*, vol. 17, no. 6, pp. 563–570, 2007.

[5] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[6] F. Oldewurtel, R. Gondhalekar, C. N. Jones, and M. Morari, "Blocking parameterizations for improving the computational tractability of affine disturbance feedback MPC problems," in *Proc. Joint* 48^{th} *IEEE Conference on Decision and Control and* 28^{th} *Chinese Control Conference*, P.R. China, 2009, pp. 7381–7386.

[7] R. Gondhalekar, J. Imura, and K. Kashima, "Controlled invariant feasibility - a general approach to enforcing strong feasibility in MPC applied to move-blocking," *Automatica*, vol. 45, no. 12, pp. 2869–2875, 2009.

[8] R. Gondhalekar and J. Imura, "Least-restrictive move-blocking model predictive control," *Automatica*, vol. 46, no. 7, pp. 1234–1240, 2010.

[9] K. V. Ling, J. M. Maciejowski, A. Richards, and B. F. Wu, "Multiplexed model predictive control," Cambridge University Engineering Department, CUED/F-INFENG/TR.657, Tech. Rep., Feb 2010.

[10] K. V. Ling, J. M. Maciejowski, J. Guo, and E. Siva, "Channel-hopping model predictive control," in *Proc.* 18^{th} *IFAC World Congress*, Milan, IT, 2011.

[11] G. A. Constantinides, "Tutorial paper: Parallel architectures for model predictive control," in *Proc. of the European Control Conference 2009*, Budapest, HU, 2009, pp. 138–143.

[12] J. L. Jerez, G. A. Constantinides, E. C. Kerrigan, and K. V. Ling, "Parallel MPC for real-time FPGA-based implementation," in *Proc.* 18^{th} *IFAC World Congress*, Milan, IT, 2011.

[13] J. Yuz, G. Goodwin, A. Feuer, and J. De Doná, "Control of constrained linear systems using fast sampling rates," *Systems & Control Letters*, vol. 54, no. 10, pp. 981–990, 2005.

[14] T. Chen and B. Francis, *Optimal Sampled-data Control Systems*. Springer, London, 1995.

[15] K. Åström and B. Wittenmark, *Computer-Controlled Systems*, 3^{rd} ed. Prentice-Hall, 1997.

[16] J. Löfberg, "YALMIP : A toolbox for modeling and optimization in MATLAB," in *Proc. CACSD Conference*, Taipei, Taiwan, 2004. [Online]. Available: http://users.isy.liu.se/johanl/yalmip