To celebrate the 80th birthday of Prof. David Mayne

A Parallel Formulation for Predictive Control with Nonuniform Hold Constraints☆

Stefano Longo[a,*], Eric C. Kerrigann[a,b], Keck Voon Ling[c], George A. Constantinides[a]

[a]*Department of Electrical and Electronic Engineering, Imperial College London, London, SW7 2AZ, UK*
[b]*Department of Aeronautics, Imperial College London, London, SW7 2AZ, UK*
[c]*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, 639798*

Abstract

This paper investigates the use of parallel computing architectures (multi-core, FPGA, GPU) to solve, at each sampling instant, a constrained optimal control problem. A set of approximated (hence smaller) problems are solved simultaneously and the solution of the one with lower open-loop cost is implemented. The approximation consists of the inclusion of additional hold constraints, which effectively reduce the number of steps in the prediction. Since smaller problems are solved, and these are solved in parallel, the computational delay is reduced and faster sampling becomes an option. The proposed method can outperform, in terms of closed-loop cost, a standard receding horizon control formulation because higher sampling rates can improve performance, even if suboptimal solutions are considered. Feasibility and stability can be guaranteed by an appropriate selection of the hold constraints.

Keywords: Predictive Control, Lyapunov stability

1. Introduction

The frustrating drawback of Model Predictive Control (MPC) is the computation time required to solve the online optimization problem within the sampling interval. This issue precludes the application of MPC techniques for systems with fast dynamics or, more generally, for systems where the platform running the control algorithm is not sufficiently powerful. A number of solutions, categorizable in two groups have been proposed. In the first one, an effort is made to find ways

☆This research has been supported by the EPSRC grant numbers EP/G031576/1, EP/F041004/1, EP/I020357/1, EP/I012036/1 and the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement number FP7-ICT-2009-4 248940.

*Corresponding author. Tel: +44 (0)20 7594 6118; fax: +44 (0)20 7581 4419.

Email addresses: s.longo@imperial.ac.uk (Stefano Longo), e.kerrigan@imperial.ac.uk (Eric C. Kerrigann), g.constantinides@imperial.ac.uk (Keck Voon Ling), ekvling@ntu.edu.sg (George A. Constantinides)

to shift a large amount of calculation offline so that in the online process the problem is reduced to simply fetching the pre-calculated values from look-up tables (see Bemporad et al. (2002)). This is known as 'explicit MPC' and only works reliably for systems with few states and inputs. In the second one, an effort is made into speeding up the online calculation by either developing faster algorithms that exploit the special structure of the optimization problem (e.g. Wang and Boyd (2010); Rao et al. (1998)) or by reducing the problem size by somehow approximating the solution. In fact, faster algorithm implementations and reduction of the problem size could be considered together, but this possibility does not yet seem to have been investigated in any systematic way.

An example of faster implementation by approximation are the practically and widely used move blocking strategies (see Maciejowski (2002); Tøndel and Johansen (2002); Qin and Badgewell (2003); Cagienard et al. (2007)). In move blocking, 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 blocking strategies, despite working well in practice, lack feasibility and/or stability proofs (which are normally enforced by terminal constraints, see Mayne et al. (2000)). In Cagienard et al. (2007) it is shown that it is possible to guarantee feasibility and stability by using a suitably-defined dynamic strategy. Recently, in Oldewurtel et al. (2009); Gondhalekar et al. (2009); Gondhalekar and Imura (2010), a generalized approach for enforcing feasibility of move blocking MPC laws that are least-restrictive has been studied.

The speed of the online optimization is highly dependent on the algorithm used to solve the Quadratic Programming (QP) problem. The so-called 'non-condensed' approach, for example, leaves the predicted states as variables to be found by the QP solver (see Wright (1993); Rao et al. (1998); Maciejowski (2002)). Although this approach increases the number of variables, it gives the QP matrices a particular banded structure that is desirable for fast optimization (see Wright (1993, 1997)). Such characteristics motivate the use of strategies that predicts only a smaller number of states and inputs. This can be seen as 'down-sampling' the prediction which, for a given time horizon, reduces the number of steps in the horizon and therefore reduces the size of the QP problem. In other words, the plant is sampled with period h but the online optimization is solved for a plant sampled every $m_i h$ seconds, where $m_i \in \mathbb{N}$ does not need to be the same for all the prediction steps i. The result is an MPC problem with Nonuniform Hold Constraints (NHC) similar to the one recently investigated in Pannocchia et al. (2010) (although in Pannocchia et al. (2010) the step size is optimized online). It is easy to realize that an MPC law with NHC is equivalent to move blocking MPC. We prefer the former definition rather than the latter, because it allows us to formulate the problem in a more natural way.

Alongside the aforementioned approaches for faster MPC implementation, there is a newly emerging concept that tackles the speeding up process from a completely new angle. A large optimization problem is divided into smaller subproblems that can be solved sequentially (see Ling et al. (Feb 2010)) or simultaneously (see Ling et al. (2011)) by exploiting multi-core processors or modern platforms like FPGAs or GPUs that inherently allow for parallelism (see Constantinides (2009); Jerez et al. (2011)). For example, in an FPGA, algorithms can be efficiently pipelined in order to continually use hardware blocks (such as adders and multipliers used for vector dot products). This increase in the number of operations performed in a given time interval by pipelining is not due to an increase in hardware resources, but is a result of its efficient usage.

In this paper, we propose an algorithm to solve a constrained LQR problem that combines the strategy of NHC and the opportunity provided by parallel computing platforms. For a Receding Horizon Control (RHC) implementation, the contributions of this paper are both theoretical and practical and are: i) 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 closed-loop performance and/or reduction of computation time when compared to standard RHC or RHC with NHC.

We call the proposed formulation *parallel NHC-RHC*. Parallel NHC-RHC is inspired by multiplexed MPC (see Ling et al. (Feb 2010)) and its parallel variant called channel-hopping MPC (see Ling et al. (2011)). Whereas multiplexed MPC and channel-hopping MPC are applicable to n_u-input systems with $n_u > 1$, our formulation is applicable even if $n_u = 1$. The advantages can be interpreted in two ways. One is that if the processor is not fast enough to execute a standard RHC algorithm, then it might be capable to do so with parallel NHC-RHC without compromising the feasibility and stability guarantees. The other one is that parallel NHC-RHC allows for the selection of shorter sampling intervals, since the online optimization problem is smaller and faster to solve. In other words, we have the freedom to trade suboptimality with sampling period and since one works against the other (in terms of performance), we can search for an optimum closed-loop performance. Also, it is well-known that faster sampling gives better disturbance rejection.

This paper is organized as follows. In Section 2 we define the standard time-invariant constrained LQR problem. In Section 3 we show how to set up the NHC control problems that will be solved online and in parallel. The idea is that 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. We carry 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 Yuz et al. (2005); Pannocchia et al. (2010). 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 period. In Section 4 we prove that, if the solution to a set of NHC control problems are implemented in a receding horizon fashion, feasibility and stability of parallel NHC-RHC is guaranteed by an appropriate choice of the hold constraints. Section 5 gives some remarks on performance. The relation between the sampling period and the closed-loop cost can be easily calculated for unconstrained problems from the solution of the Riccati equation. For the constrained case this relation is not straightforward and this is investigated using simulation examples, which are given in Section 6.

2. Preliminaries

Consider the following continuous-time LTI plant model

$$\dot{x}(t) = A_c x(t) + B_c u(t), \tag{1}$$

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 sampling period h such that

$$u(t) = u(kh), \quad \forall t \in [kh, (k+1)h), \tag{2}$$

and k is the sampling instant. The system in (1) can be sampled with a periodic interval h giving the sampled-data system, for constant input u,

$$x(kh+h) = A_h x(kh) + B_h u(kh), \quad k \in \mathbb{N}_0, \tag{3}$$

where $x(kh) \in \mathbb{R}^{n_x}$, $u(kh) \in \mathbb{R}^{n_u}$, $A_h := e^{A_c h}$ 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 $2\pi/h$ (Chen and Francis, 1995, 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 (necessary and sufficient conditions can be found in Kimura (1990)). If this condition is applied only to the unstable eigenvalues of A_c, then stabilizability is preserved after discretization.

Associated with the system in (1) consider the continuous-time, finite horizon LQ problem defined by the cost function

$$J_c(x(\cdot), u(\cdot)) := x(T)' P_c x(T) + \int_0^T \Big[x(t)' Q_{1,c} x(t) + u(t)' Q_{2,c} u(t) \Big] dt, \tag{4}$$

where $Q_{1,c} = Q_{1,c}' \geq 0$, $Q_{2,c} = Q_{2,c}' > 0$, $(Q_{1,c}^{1/2}, A)$ detectable $P_c = P_c' > 0$ and T is the prediction horizon. These matrices are given to define a controller for an ideal closed-loop performance of the continuous-time model. The equivalent sampled-data cost function is

$$J_h(x(\cdot), u(\cdot)) := x(Nh)' P_h x(Nh) + \sum_{k=0}^{N-1} \begin{bmatrix} x(kh) \\ u(kh) \end{bmatrix}' \underbrace{\begin{bmatrix} Q_1 & Q_{12} \\ Q_{12}' & Q_2 \end{bmatrix}}_{Q_h} \begin{bmatrix} x(kh) \\ u(kh) \end{bmatrix}, \tag{5}$$

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

$$N := \left\lceil \frac{T}{h} \right\rceil, \tag{6}$$

$\lceil \cdot \rceil$ is the ceiling function and expressions for Q_h and P_h can be found in Åström and Wittenmark (1997). Matrices Q_2 and P_h are positive definite. With the assumption that the input $u(t)$ is constant over the interval $t \in [kh, k(h+1))$ and the states $x(kh)$ are sampled at time instant $t = kh$ without computational delays, there is no approximation error between (4) and (5) (see Åström and Wittenmark (1997)). The computational delay can be explicitly taken into account by, for example, appropriate augmentation of the plant matrices (Maciejowski, 2002, Section 2.5).

For the constrained LQR problem we assume full state feedback and we suppose that constraints exist on the input moves and on the states. They are represented by $(u(kh), x(kh)) \in \mathbb{Z}_h \subseteq \mathbb{R}^{n_u} \times \mathbb{R}^{n_x}$ where, in order to solve a QP, we assume that $\mathbb{Z}_h$ is a polyhedral set with the origin in its interior. The set $\mathbb{Z}_h$ has to be a function of the sampling period h, as shown in Yuz et al. (2005), to guarantee state constraint satisfaction between sample instants. Let $x(kh) := x$ be the state measured at sample instant k and x_i, where $i \in \mathbb{N}_0$, be the predicted state at $k+i$ for the given

initial state x, and an input sequence $(u_0, u_1 \ldots, u_{i-1})$. Also, let the set $\mathcal{X}(K_h)$ be a constraint-admissible positively invariant set for the system in (3) subject to control $u = K_h x$ (Cagienard et al., 2007, Definition 1), where K_h is a pre-specified stabilizing feedback controller such that

$$x \in \mathcal{X}(K_h) \Rightarrow \big[(K_h x, x) \in \mathbb{Z}_h \quad \text{and} \quad (A_h + B_h K_h)x \in \mathcal{X}(K_h)\big]. \tag{7}$$

Let the vector with the predicted input moves and predicted states be

$$\boldsymbol{u} := \begin{bmatrix} u_0' & u_1' & \ldots & u_{N-1}' \end{bmatrix}', \quad \boldsymbol{u} \in \mathbb{R}^{Nn_u} \tag{8}$$

and

$$\boldsymbol{x} := \begin{bmatrix} x_1' & x_2' & \ldots & x_N' \end{bmatrix}', \quad \boldsymbol{x} \in \mathbb{R}^{Nn_x}, \tag{9}$$

respectively, then the *standard time-invariant constrained LQR* problem is defined as

$$\begin{aligned} \bar{J}_h^*(x) := \min_{\boldsymbol{u},\boldsymbol{x}} & \left\{ x_N' P_h x_N + \sum_{i=0}^{N-1} \begin{bmatrix} x_i \\ u_i \end{bmatrix}' Q_h \begin{bmatrix} x_i \\ u_i \end{bmatrix} \right\}, \\ \text{s.t.} \quad & (x_i, u_i) \in \mathbb{Z}_h, \quad x_N \in \mathcal{X}(K_h), \\ & x_{i+1} = A_h x_i + B_h u_i, \quad x_0 = x, \\ & \forall i \in \{0, 1, \ldots, N-1\}. \end{aligned} \tag{10}$$

The terminal set constraint $x_N \in \mathcal{X}(K_h)$ is needed so that recursive feasibility is achieved for a receding horizon implementation (see Mayne et al. (2000); Cagienard et al. (2007)). The unique optimizer to the control problem above is defined as $\boldsymbol{u}^*(x)$. The subscript h in the plant and weighting matrices indicates that those matrices are derived from a discretization of a continuous-time problem (although they could also be identified from data).

3. Solution by NHC and parallel formulation

The idea we propose for the speeding up of the computation of the solution of the standard constrained LQR problem in (10) is to define a set of approximated (hence faster to solve) problems and solve them simultaneously using the emerging parallel computing technology. The input sequence of the problem resulting in the lowest open-loop cost will be the solution to the constrained LQR problem. The approximation consists in coarsely sampling the prediction horizon by imposing hold constraints that are larger than the sampling period h. The computation time is reduced because the QPs are smaller and solved in parallel.

3.1. Nonuniform Hold Constraints (NHC)

The NHC scheme we are adopting describes a control system where the controller samples the states x and updates the inputs u at a certain period h but it performs the prediction using step sizes that are integer multiples of h, as represented diagrammatically in Figure 1. This can be formalized as follows. Let us first define the *sequence of holds* to be

$$\mathcal{M} := (m_0, m_1, \ldots, m_M), \tag{11}$$

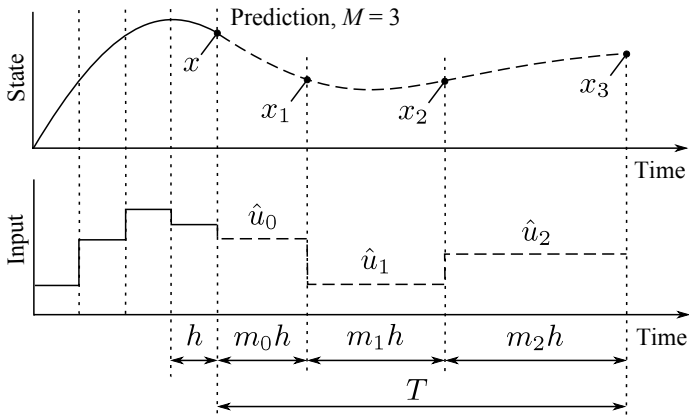

Figure 1: Schematic representation of the NHC scheme. The prediction is performed for M steps that are multiple of the sampling period h.

where $m_i \in \mathbb{N}$, $i = 0, 1, \ldots, M$ are the holds and M is the number of steps in the horizon such that, if the horizon is sought to be of length T, then

$$\sum_{i=0}^{M} m_i h = T. \tag{12}$$

The transition from the problem in (10) to the one with NHC involves the inclusion in (10) of the hold constraints

$$u_{q_j} = u_{q_j+1} = \ldots = u_{q_j+m_j-1} = \hat{u}_j, \quad j = 0, 1, \ldots, M-1, \tag{13}$$

where

$$q_j := \begin{cases} 0 & \text{for} \quad j = 0 \\ \sum_{l=0}^{j-1} m_l & \text{for} \quad j > 0 \end{cases}. \tag{14}$$

This is written more compactly in the following *NHC problem*, denoted as $\mathrm{P}(x, \mathcal{M})$, and defined as

$$\begin{aligned} \mathrm{P}(x, \mathcal{M}): \quad J_h^*(x, \mathcal{M}) := \min_{\hat{\boldsymbol{u}}, \hat{\boldsymbol{x}}} & \left\{ \hat{x}_M' P_{m_M h} \hat{x}_M + \sum_{i=0}^{M-1} \begin{bmatrix} \hat{x}_i \\ \hat{u}_i \end{bmatrix}' Q_{m_i h} \begin{bmatrix} \hat{x}_i \\ \hat{u}_i \end{bmatrix} \right\}, \\ \text{s.t.} \quad & (\hat{x}_i, \hat{u}_i) \in \mathbb{Z}_{m_i h}, \quad \hat{x}_M \in \mathcal{X}(K_{m_M h}), \\ & \hat{x}_{i+1} = A_{m_i h} \hat{x}_i + B_{m_i h} \hat{u}_i, \quad \hat{x}_0 = x, \\ & \forall i \in \{0, 1, \ldots, M-1\}, \end{aligned} \tag{15}$$

where $\hat{\boldsymbol{u}} := \begin{bmatrix} \hat{u}_0' & \hat{u}_1' & \ldots & \hat{u}_{M-1}' \end{bmatrix}'$ and $\hat{\boldsymbol{x}} := \begin{bmatrix} \hat{x}_1' & \hat{x}_2' & \ldots & \hat{x}_M' \end{bmatrix}'$ are the coarse vectors of predicted input moves and predicted states, respectively. Note that the state constraints are enforced only at the discrete intervals given by $\mathcal{M}$, hence we restrict our constraints to a suitably-defined $\mathbb{Z}_{m_i h} \subseteq \mathbb{Z}_h$ (see Yuz et al. (2005)) to ensure inter-sample constraint satisfaction. Note that (15) is a time-varying constrained LQR problem.

The NHC formulation in (15) implies that, for the formulation of the QP, the predicted states and instants where the constraints are enforced coincide with the time instants specified by $\mathcal{M}$. This assumption can be relaxed. For instance, the problem in (15) could be restated in terms of $\mathcal{M}$ as well as two other sequences that define the time instants of the states that have to be included in the prediction and the time instants of constraint enforcement. This setup gives more flexibility for 'tuning' the control problem in relation to the plant and the QP implementation. Such a complication will not be considered here.

NHC schemes offer a degree of flexibility that can significantly reduce the QP problem size. For example, the theoretical computational cost for a condensed implementation of the problem in (10) is $O(N^3 n_u^3)$ (see Rao et al. (1998); Maciejowski (2002)). The NHC implemented as in (15) effectively reduces the number of steps in the prediction from N to $M < N$ which translates, in this case, to a cubic improvement in computational cost without having to shorten the prediction horizon T.

3.2. Parallel formulation

Now, consider any hardware platform that allows parallel computation (as the aim here is to demonstrate the potential advantage of such a parallel implementation). This can be multi-core processors, FPGAs, GPUs or even independent industrial PCs or microprocessor boards. In this platform, more than one problem $\mathrm{P}(x, \mathcal{M})$ can be solved in parallel. Let $\mathcal{S} := \{\mathcal{M}_s | s = 0, 1, \ldots, S-1\}$ be a set of S different sequences of holds. Let $\hat{\boldsymbol{u}}_s^*(x)$ and $J_h^*(\cdot, \mathcal{M}_s)$ be the optimizer and the value function, respectively, of $\mathrm{P}(x, \mathcal{M})$ with $\mathcal{M} = \mathcal{M}_s$. 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, see Constantinides (2009); Jerez et al. (2011)). In the parallel formulation with NHC we propose S problems, each associated with a sequence from the set $\mathcal{S}$, are solved simultaneously at each sampling instant. The resulting optimal input sequence is defined as $\hat{\boldsymbol{u}}_{s^*(x)}^*(x)$, where

$$s^*(x) := \underset{s}{\operatorname{argmin}} \ \{J_h^*(x, \mathcal{M}_s) \mid \mathcal{M}_s \in \mathcal{S}\}. \tag{16}$$

In other words, the optimal input sequence is the one that results in the smallest predicted open-loop cost among S optimization problems (also shown diagrammatically in Figure 2). Note that the parallel formulation is an improvement to sequential (non-parallel) NHC because the open-loop cost can only be equal or better than the case where only one NHC problem is solved. Such an improvement comes from solving parallel problems and is achieved from the hardware usage in the aforementioned platforms.

If the solution to the parallel optimal control problem defined above is implemented in a receding horizon fashion we will refer to it as parallel NHC-RHC. In the next section we will show that feasibility and stability of parallel NHC-RHC can be guaranteed by an appropriate selection of $\mathcal{S}$.

4. Feasibility and stability of a receding horizon implementation

In standard RHC, if $x \mapsto x' P_h x$ is a local Lyapunov function for the closed-loop system with controller gain K_h then stability can be guaranteed (see Mayne et al. (2000)). The controller K_h

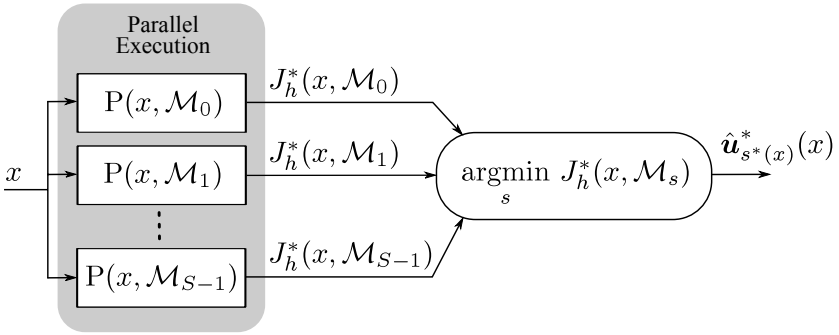

Figure 2: Schematic implementation of the algorithm. S problems with NHCs are solved in parallel but only the solution of the one resulting in the smallest predicted cost is used for implementation.

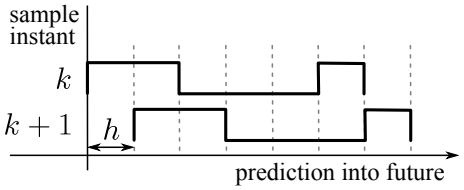

Figure 3: An NHC formulation where at time k the predicted inputs are allowed to change every $(2h, 3h, h)$. In the shifted sequence of predicted inputs (sample instant $k+1$) the points where the inputs are allowed to change do not coincide with the sequence at sample instant k.

can be chosen to be $K_h = -(B_h' P_h B_h + Q_2)^{-1}(B_h' P_h A_h + Q_{12}')$ where P_h is the discrete-time equivalent of P_c, and P_c is the solution of the continuous-time algebraic Riccati equation

$$A_c' P_c + P_c A_c - P_c B_c Q_{2,c}^{-1} B_c' P_c + Q_{1,c} = 0 \tag{17}$$

(see (4), (5) and Åström and Wittenmark (1997)). Both feasibility and stability can be proven by considering a shifted vector of input moves, at time $k+1$,

$$\tilde{\boldsymbol{u}}(x) := \left[\ (u_1^*(x))' \quad (u_2^*(x))' \quad \cdots \quad (u_{N-1}^*(x))' \quad (K_h x_N^*(x))'\ \right]', \tag{18}$$

where $x_N^*(x)$ is the predicted state at time $k+N$ given x and $\boldsymbol{u}^*(x)$ at time k. 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 $\bar{J}_h^*(\cdot)$ from (10) is a Lyapunov function (due to the choice of P_h), stability is also guaranteed (see Mayne et al. (2000)).

It is well-known (see Cagienard et al. (2007)) that, for NHC problems, the shifting argument does not generally apply because the points where the inputs u are held (according to the constraints in (13)) in the shifted vector $\tilde{\boldsymbol{u}}(x)$ do not correspond to the points where the inputs are held in the non-shifted vector $\boldsymbol{u}^*(x)$. For example, Figure 3 shows a situation of an NHC formulation where $\mathcal{M} = (2, 3, 1)$, i.e. the predicted input is held for $2h$, $3h$ and h or, equivalently, the following constraints apply: $u_0 = u_1$, $u_2 = u_3 = u_4$. Clearly, at time $k+1$, the points in the prediction horizon where the inputs are held do not coincide to the ones at time k and the shifting argument collapses.

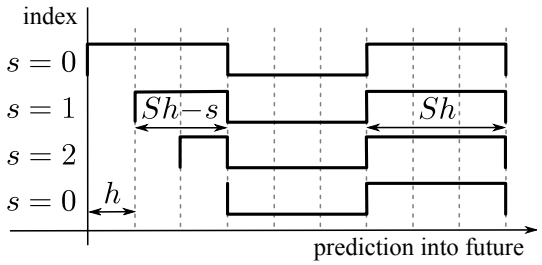

Figure 4: The shifting argument for recursive feasibility for parallel NHC-RHC can be re-established by selecting particular sequences of hold constraints. This is because it is always possible to choose the solution of a problem in which the points where the inputs at future sample instants are held coincide with the ones of the previous shifted solution.

We now show that, by considering a particular set $\mathcal{S}$, it is possible to re-establish the shifting argument for parallel NHC-RHC. The key idea is to have $\mathcal{S}$ such that, for a sequence $(a_0, a_1, \ldots, a_M) \in \mathcal{S}$, it is always possible to find a sequence $(b_0, b_1, \ldots, b_M) \in \mathcal{S}$ such that, if $a_0 > 1$, then $b_0 = a_0 - 1, b_1 = a_1, b_2 = a_2, \cdots, b_M = a_M$ and if $a_0 = 1$, then $b_0 = a_1, b_1 = a_2, \cdots, b_{M-1} = a_M$. For the case where $a_0 = 1$, b_M depends on the choice of the terminal constraint and weight as will be shown later for a particular case. This ensures that, if at time step k a sequence from $\mathcal{S}$ is used, at time step $k+1$ it is always possible to find another sequence in $\mathcal{S}$ in which the instants where the inputs are held coincide with the instants where the inputs are held in shifted vector of input moves at time k. An example of such a set is $\{(2,3,1),(1,3,1),(3,1,2),(2,1,2),(1,1,2),(1,2,3)\}$.

To avoid unnecessary complications with notation, we consider only a special simple case. Let $\mathcal{S} := \{\mathcal{M}_s | s = 0, 1, \ldots, S-1\}$ where

$$\mathcal{M}_s := \big(\underbrace{S-s, S, S, \ldots, S}_{M}\big), \quad s = 0, 1, \ldots, S-1 \tag{19}$$

and $S \in \mathbb{N}_0$. Note that, for $S = 1$, this formulation is actually a standard RHC for a time-invariant constrained LQR problem. Also note that the prediction horizon T of the NHC problems becomes a function of s according to $T(s) := MSh - sh$. This is illustrated, for an example with $S = 3$, in Figure 4.

The parallel NHC-RHC control law is defined as

$$\hat{\kappa}(x) := \begin{bmatrix} I_{n_u} & 0 \end{bmatrix} \hat{\boldsymbol{u}}^*_{s^*(x)}(x), \tag{20}$$

where $s^*(x)$ is given in (16) and $\mathcal{M}_s$ is defined as in (19).

Proposition 4.1. *For non-pathological sampling frequencies, if the receding horizon control law in* (20) *is implemented, then the origin of the closed-loop system* (1) *in feedback with* (20) *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 $\mathrm{P}(x, \mathcal{M}_s)$ 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

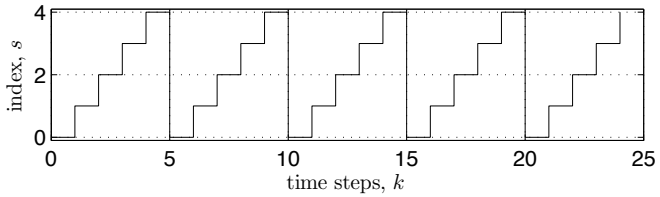

Figure 5: An example of a fixed pattern that guarantees feasibility (and stability) for $S = 5$ if a parallel implementation is not available (s is the index of $\mathcal{M}_s$ in (19)).

to obtain a feasible solution at time $k + 1$, i.e. the time instants where the inputs are held must coincide for the prediction obtained at k and the prediction obtained at $k+1$. Assume that, at time k, the sequence of holds $(S-i, S, S, \ldots, S)$ is used. In order to ensure that at the next time step the predicted input moves are held at the same points in time we must use a sequence of holds which is the previous one but with the first hold instant decreased by one, i.e. $(S-i-1, S, S, \ldots, S)$. If $i = S-1$, then we can use $(S, S, S, \ldots, S)$. In the sequence $\mathcal{M}_s$ in (19), the holds are all equal to each other, apart from the first one, which depends on s and can take a value from $\{1, 2, \ldots, S\}$. Hence, it is always possible to find a sequence of holds which guarantees that, at the next time step, the predicted inputs are held at the same time instants.

If problem $\mathrm{P}(x, \mathcal{M}_i)$ 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 problem $\mathrm{P}(x, \mathcal{M}_{i+1})$ (or $\mathrm{P}(x, \mathcal{M}_0)$ if $i = S-1$), 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 (15) is P_{Sh} (i.e. a discretization of P_c with sampling period Sh) and $x \mapsto x' P_{Sh} x$ is a local Lyapunov function. Hence, the value function $x \mapsto J_h^*(x, \mathcal{M}_{s^*(x)})$ in (15) can be used as a Lyapunov function (see Mayne et al. (2000)). ■

Remark 4.1. *Note that to prove recursive feasibility (and stability), it is sufficient to solve a single problem* $\mathrm{P}(x, \mathcal{M}_s)$*, at each sampling instant, with* s *following the pattern* $0, 1, \ldots, S-1, 0, 1, \ldots$*, as shown in Figure 5 for* $S = 5$*. However, in the interval from* $\mathrm{P}(x, \mathcal{M}_0)$ *to* $\mathrm{P}(x\mathcal{M}_{S-1})$ *the prediction horizon is not receding. This fixed pattern is actually a simplified version of the one proposed in Cagienard et al. (2007).*

Remark 4.2. *Figure 2 and* (15) *suggest that parallel NHC-RHC can be implemented as parallel copies of* (15)*, each parameterized with* $\mathcal{M}_s$*,* $s = 0, 1, \ldots, S-1$*. Arising from Remark 4.1, we could also view parallel NHC-RHC, conceptually, as parallel copies of* (15)*, of which sequences of holds follow the pattern*

$$\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_{S-1}, \mathcal{M}_0, \ldots, \tag{21}$$

for the first problem,

$$\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{S-1}, \mathcal{M}_0, \mathcal{M}_1, \ldots, \tag{22}$$

for the second problem and so on. Viewed in this way, recursive feasibility and stability of parallel NHC-RHC can be established by the following reasoning: since each NHC problem is recursively feasible and stable, and the parallel NHC-RHC algorithm picks the lowest cost among them, parallel NHC-RHC will also be recursively feasible and stable. It is also clear from this reasoning that feasibility and stability can be achieved with $\tilde{S} \leq S$ NHC problems as described above, running in parallel with $\mathcal{M}_s$ following periodic patterns as the ones in (21) *and* (22)*.*

5. Closed-loop performance

The effective reduction of the number of prediction horizon steps from N to M results in a smaller optimization problem, which may become computationally realizable if the original problem was not realizable. Since the online optimization problem is now faster to solve, the plant's output can be sampled at a higher rate to improve the closed-loop performance and its disturbance rejection properties. Hence, with parallel NHC-RHC, 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 (6), 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 parallel NHC-RHC is more significant than its increase due to a larger N. In general, this depends on how the problem is formulated and the method used to solve the QP problems.

6. Illustrative example

In this section we show, with simulation examples, the advantages of parallel NHC-RHC when compared to a standard RHC implementation (without the additional hold constraints) and a sequential (non-parallel) NHC one. We use as a benchmark an unstable oscillator described by

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} 5 & 15 \\ -15 & 5 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 50 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t), \end{aligned} \tag{23}$$

having input constraints $-1 \leq u \leq 1$. No state constraints are included so that the results are not complicated by the problem of guaranteeing state constraint satisfaction between sampling instants (as discussed in Section 3). The associated performance measure is given by the LQ cost in (4) with matrices $Q_{1,c} = 1$, $Q_{2,c} = 0.1$ and P_c as the solution of (17). To satisfy our feasibility and stability conditions, the terminal weight has been set to P_{Sh}, (i.e. a discretization of P_c for a sampling period Sh) and the maximum positively invariant set is given by $\mathcal{X}(K_{Sh})$ (this can be computed with the algorithm in Gilbert and Tan (1991)).

We let $\hat{T} = 0.3$s be a desired prediction time horizon and calculate the actual horizon as $T = Nh$, where

$$N := S \left\lceil \frac{\hat{T}}{Sh} \right\rceil . \tag{24}$$

Note that standard RHC corresponds to an NHC implementation with $S = 1$ and, in this case, the definitions for N in (24) and (6) are identical. The reason for calculating T this way is to ensure that the time horizon is an integer multiple of h and $T \geq \hat{T}$ always. The horizon $\hat{T}$ (and consequently N and M) is chosen long enough so that $x_N \in \mathcal{X}(K_{Sh})$ at the solution of all the NHC problems.

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

$$F := \sum_{k=0}^{N_{\text{sim}}-1} \begin{bmatrix} x(kh) \\ u(kh) \end{bmatrix}' Q_h \begin{bmatrix} x(kh) \\ u(kh) \end{bmatrix}, \tag{25}$$

where Q_h is the performance matrix as in (5) and $N_{\text{sim}} = 1$s. The computational delay has been catered for as in (Maciejowski, 2002, Section 2.5) and the input as been initialized with $u(0) = 0.3$. As a measure of computational effort, we use the time required by MATLAB (with the toolbox YALMIP, see Löfberg (2004)) to solve the QP problem(s) 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 computation time considered is the longest (worst case) among them. Note that the performance of parallel NHC-RHC can be enhanced by solving the QP with different solvers that exploit the particular structure of the problem. This could be the subject of future investigation. For the moment it is sufficient to show that the computation time is reduced by the proposed implementation even when using a general purpose solver.

6.1. Closed-loop cost versus computation time

We now compare the closed-loop performance of the above system for a range of sampling periods and for the following different implementations.

- The proposed parallel NHC-RHC characterized by the control law given in (20). We solve (in parallel) $S = 5$ NHC problems $\mathrm{P}(x, \mathcal{M}_s)$, $s = 0, 1, 2, 3, 4$ (defined in (15)) where $\mathcal{M}_s$ is given in (19). Recursive feasibility is guaranteed because of Proposition 4.1.

- The standard RHC characterized by the control law

$$\kappa(x) := \begin{bmatrix} I_{n_u} & 0 \end{bmatrix} \boldsymbol{u}^*(x), \tag{26}$$

  where $\boldsymbol{u}^*(x)$ is the unique optimizer to the problem in (10).

- A sequential NHC-RHC. For fairness of comparison we enforce recursive feasibility also to this implementation. This is achieved, in accordance with Remark 4.1, by the control law

$$\tilde{\kappa}(x) := \begin{bmatrix} I_{n_u} & 0 \end{bmatrix} \hat{\boldsymbol{u}}^*_{\tilde{s}(k)}(x), \tag{27}$$

  where

$$k \mapsto \tilde{s}(k) = (0, 1, 2, 3, 4, 0, 1, \ldots), \tag{28}$$

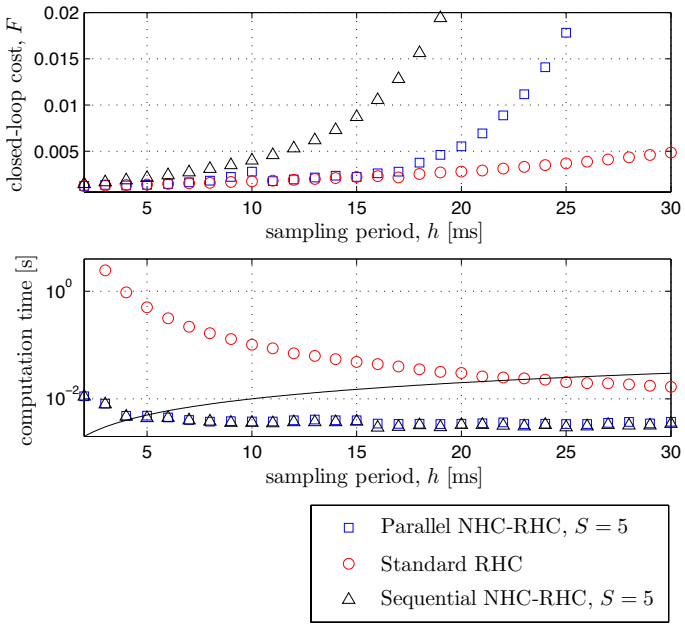

Figure 6: Performance comparison of three different implementations. There is a limit (solid line, bottom plot) on how fast we can sample, based on the computation time, which determines the best achievable closed-loop cost. Only the points below the line are possible.

(thus we solve $\mathrm{P}(x, \mathcal{M}_s)$ with $s = \tilde{s}(k)$). Only one problem (but not the same one) is solved at each sampling instant. This sequential NHC-RHC implementation is similar to the ones considered in Cagienard et al. (2007) where the receding horizon controller is a dynamic state feedback controller. Note that in the interval from $\tilde{s}(0)$ to $\tilde{s}(4)$ the prediction horizon is not receding (Remark 4.1), but decreasing.

The top plot of Figure 6 shows the variation of the cost F as the sampling period increases. The bottom plot shows the computation time (the worst case among the parallel problems for parallel NHC-RHC) required to solve the QP problem(s). The computation was carried out on a quad core machine with a 2.66 GHz CPU, 16 GB of RAM and running MATLAB version 7.4.0.287.

The top plot of Figure 6 indicates that the cost of parallel NHC-RHC is very close the one of standard RHC for small sampling periods h but, as h increases, these costs diverge. Nevertheless, by slightly reducing h, it is possible for parallel NHC-RHC to achieve a performance equal to (or better than) the one of standard RHC. Sampling faster, however, gives less time to solve the QP problems. The bottom plot of Figure 6 shows that this is not an issue, since in parallel NHC-RHC the computation time required to solve the QP problems is always sufficiently small. Indeed, there is a bound by how fast we can sample, which is given by the fact that sampling (normally) cannot be faster than the computation time. This bound is shown by the line in the bottom plot of Figure 6. We can see that sampling faster than approximately 24ms is not possible for standard RHC, while it is possible to sample as fast as 5ms for parallel NHC-RHC and sequential NHC-RHC.

Figure 6 also suggests that it is possible to achieve better closed-loop performance with parallel NHC-RHC than standard RHC when constraints on available computational time are considered.

	Shortest possible sampling period h [ms]	Best achievable performance F
Parallel NHC-RHC	5	0.0014
Standard RHC	24	0.0035
Sequential NHC-RHC	5	0.0022

Table 1: Comparison of achievable closed-loop performance.

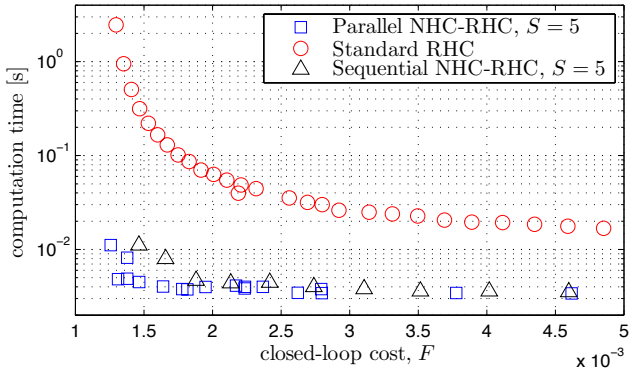

Figure 7: Pareto-optimal tradeoff between computation time required to solve the optimization problem and closed-loop cost. For a given computational power, parallel NHC-RHC has a lower cost than the other implementations. Likewise, a desired closed-loop cost can be achieved with less computational power by using parallel NHC-RHC.

For this particular case, we use the bottom plot of Figure 6 to see how fast we can sample for each implementation and then we use the top plot to see what performance we can achieve at that sampling rate. Table 1 shows the lowest cost F achievable for each implementation and it is clear that the best performance is the one of parallel NHC-RHC.

For an infinitely powerful processor, standard RHC is always the best implementation. When the computation time becomes a constraint (which is always the case), the proposed method approximates the solution but the gained computational speedup can be used to sample faster and recover the performance lost. In the example above, parallel NHC-RHC performed better than standard RHC because the loss due to approximation was less than the gain due to faster sampling.

It is worth noting that the reason why the computation times for parallel NHC-RHC and sequential NHC-RHC are almost identical is because the QPs solved have the same size. However, the closed-loop cost of sequential NHC-RHC for increasing h deteriorates more quickly. This is due to the index $\tilde{s}(k)$ being fixed for the sequential implementation, in contrast to parallel NHC-RHC where $s^*(x)$ is a function of the current state.

Finally, the results of Figure 6 have also been presented in Figure 7 as a scatter plot to show the Pareto-optimal tradeoff of the closed-loop cost and computation time. From this plot it is possible

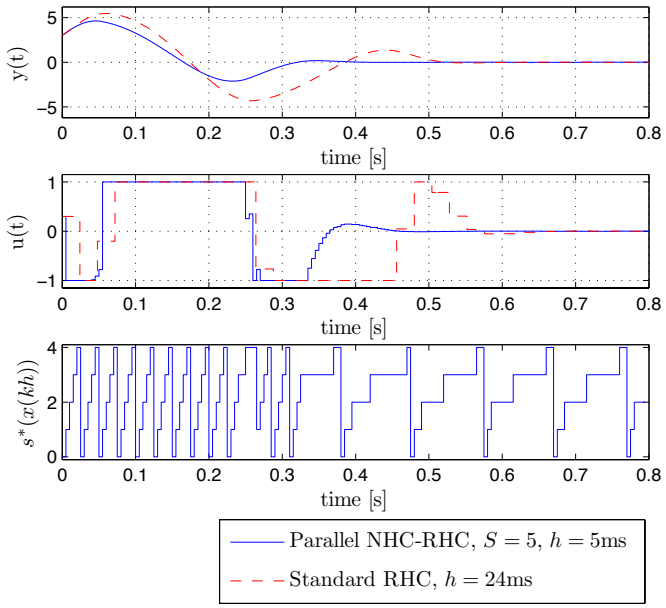

Figure 8: Performance comparison between parallel NHC-RHC and standard RHC. The sampling period h used for each one is the shortest possible one which is constrained by the computation time required to solve the optimization problem.

to determine: i) what cost can be achieved for each implementation for a given computation time and ii) how much computation time is needed for each implementation to achieve a given cost. In both cases, parallel NHC-RHC offers the best compromise.

6.2. Closed-loop performance

The previous results indicate that choosing shorter sampling periods for parallel NHC-RHC is possible because the QP problems can be solved faster than for standard RHC. We now compare parallel NHC-RHC and standard RHC by showing the evolution of the trajectories. They are both implemented at their shortest possible sampling periods (see Figure 6 and Table 1). This reflects the practical situation where the sampling rate is constrained by the computation time required to solve the QP problem(s) given the available computational resources. The simulation trajectories, plotted in Figure 8, clearly indicate that a better performance is achieved with parallel NHC-RHC. This suggests that, despite finding sub-optimal solutions, the ability to sample faster can recover or improve the closed-loop performance, which explains why blocking schemes are successfully applied in practice (see Qin and Badgewell (2003)).

The bottom plot of Figure 8 shows the evolution of the optimal index $s^*(x)$, which determines what sequence of holds $\hat{\mathcal{M}}_{s^*(x)}$ is used (see (16)). It is interesting to notice that, when the input constraints are active, $s^*(x)$ evolves following almost exactly the pattern of $0, 1, 2, 3, 4, 0, 1, \ldots$, i.e. the one that sufficiently guarantees recursive feasibility according to Remark 4.1. However, if the input constraints are not active, different patterns are used by the algorithm as terminal constraint satisfaction becomes easier.

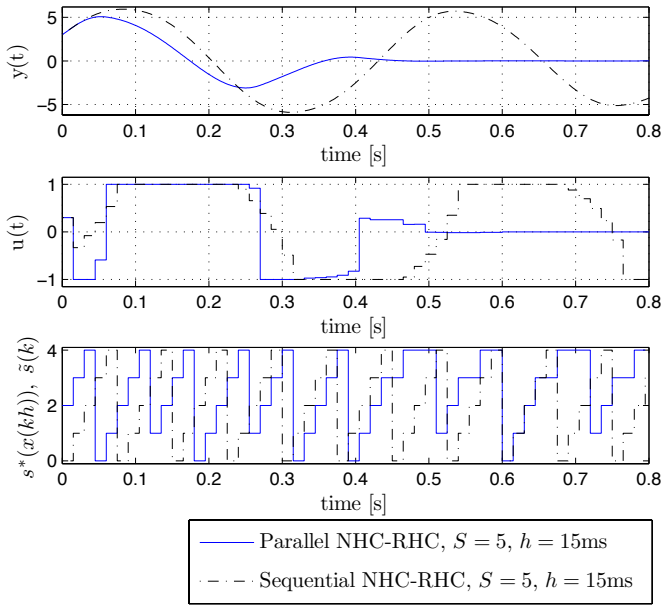

Figure 9: Performance comparison between parallel NHC-RHC and sequential NHC-RHC. For a given sampling period h parallel NHC-RHC performs better because $s^*(x)$ (solid line) is a function of the current state.

Figure 9 shows a comparison between parallel NHC-RHC and sequential NHC-RHC, both implemented with same sampling period and same S. Both implementations are identical apart from the fact that for parallel NHC-RHC, 5 problems are solved in parallel and the solution of the best one is used, while for sequential NCH a fixed pattern of single problems are solved. The closed-loop behavior of parallel NHC-RHC is more desirable also in this case. The ability to choose $s^*(x)$ optimally online (shown in the bottom plot of Figure 9) is the reason why parallel NHC-RHC performs better than sequential NHC-RHC for equal sampling periods.

7. Conclusions

With this paper, we proposed a method to reduce the computation time of the online optimization problem required by RHC. This is achieved by first approximating the solution of a standard time-invariant constrained LQR problem and then exploiting the parallelism offered by new computing architectures. 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 size of the optimization problems is smaller compared to the one of a standard RHC, such problems can be solved in less time, allowing for faster sampling. The approximation is achieved by adding hold constraints and performing the prediction at steps that are larger than the actual system's sampling interval. The parallelism is exploited by solving a number of differently defined problems simultaneously and then choosing the solution of the one that resulted in the lowest open-loop cost. An example showed how the proposed implementation

(called parallel NHC-RHC) can outperform standard RHC (and similar sequential implementations) in terms of closed-loop performance. Future work could include the implementation of parallel NHC-RHC in hardware such as an FPGA.

References

Åström, K., Wittenmark, B.. Computer-Controlled Systems. 3^{rd} ed. Prentice-Hall, 1997.

Bemporad, A., Morari, M., Dua, V., Pistikopolous, E.N.. The explicit linear quadratic regulator for constrained systems. Automatica 2002;38(1):3–20.

Cagienard, R., Grieder, P., Kerrigan, E.C., Morari, M.. Move blocking strategies in receding horizon control. J Process Control 2007;17(6):563–570.

Chen, T., Francis, B.. Optimal Sampled-data Control Systems. Springer, London, 1995.

Constantinides, G.A.. Tutorial paper: Parallel architectures for model predictive control. In: Proc. of the European Control Conference 2009. Budapest, HU; 2009. p. 138–143.

Gilbert, E.G., Tan, K.T.. Linear systems with state and control constraints: the theory and applications of maximal output admissible sets. IEEE Transactions on Automatic Control 1991;36(9):1008–1020.

Gondhalekar, R., Imura, J.. Least-restrictive move-blocking model predictive control. Automatica 2010;46(7):1234–1240.

Gondhalekar, R., Imura, J., Kashima, K.. Controlled invariant feasibility - a general approach to enforcing strong feasibility in MPC applied to move-blocking. Automatica 2009;45(12):2869–2875.

Jerez, J.L., Constantinides, G.A., Kerrigan, E.C., Ling, K.V.. Parallel MPC for real-time FPGA-based implementation. In: Proc. 18^{th} IFAC World Congress. Milan, IT; 2011. .

Kimura, M.. Preservation of stabilizability of a continuous time-invariant linear system after discretization. Int J Systems Science 1990;21(1):65–91.

Ling, K.V., Maciejowski, J.M., Guo, J., Siva, E.. Channel-hopping model predictive control. In: Proc. 18^{th} IFAC World Congress. Milan, IT; 2011. .

Ling, K.V., Maciejowski, J.M., Richards, A., Wu, B.F.. Multiplexed Model Predictive Control. Technical Report; Cambridge University Engineering Department, CUED/F-INFENG/TR.657; Feb 2010.

Löfberg, J.. YALMIP : A toolbox for modeling and optimization in MATLAB. In: Proc. CACSD Conference. Taipei, Taiwan; 2004. .

Maciejowski, J.M.. Predictive control with constraints. Prentice-Hall, 2002.

Mayne, D.Q., Rawlings, J.B., Rao, C.V., Scokaert, P.O.M.. Constrained model predictive control: Stability and optimality. Automatica 2000;36(6):789–814.

Oldewurtel, F., Gondhalekar, R., Jones, C.N., Morari, M.. 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. p. 7381–7386.

Pannocchia, G., Rawlings, J.B., Mayne, D.Q., Marquardt, W.. On computing solutions to the continuous time constrained linear quadratic regulator. IEEE Trans on Automatic Control 2010;55(9):2192–2198.

Qin, S.J., Badgewell, T.A.. A survey of industrial model predictive control technology. Control Engineering Practice 2003;11(7):733–764.

Rao, C.V., Wright, S.J., Rawlings, J.B.. Application of interior-point methods to model predictive control. J Optimization Theory and Applications 1998;99(3):723–757.

Tøndel, P., Johansen, T.A.. Complexity reduction in explicit linear model predictive control. In: Proc. 15^{th} IFAC World Congress. Spain; 2002. .

Wang, Y., Boyd, S.. Fast model predictive control using online optimization. IEEE Transactions on Control System Technology 2010;18(2):267–278.

Wright, S.J.. Interior point methods for optimal control of discrete time systems. J Optimization Theory and Applications 1993;77(1):161–187.

Wright, S.J.. Applying new optimization algorithms to model predictive control. In: J.C. Kantor, C.E. Garcia, B. Carnahan (Eds.), 5^{th} Int. Conf. Chemical Process Control, CACHE, AIChE. Lake Tahoe, CA; 1997. p. 147–155.

Yuz, J., Goodwin, G., Feuer, A., De Doná, J.. Control of constrained linear systems using fast sampling rates. Systems & Control Letters 2005;54(10):981–990.