# Multiplexed Model Predictive Control *


Keck Voon Ling
School of Electrical and Electronics Engineering
Nanyang Technological University
Singapore, 639798
ekvling@ntu.edu.sg

Jan Maciejowski
Department of Engineering
University of Cambridge
United Kingdom

Arthur Richards
Department of Aerospace Engineering
University of Bristol
United Kingdom

Bing Fang Wu
School of Electrical and Electronics Engineering
Nanyang Technological University
Singapore


16 February 2010


**Abstract**

This paper proposes a form of MPC in which the control variables are moved asynchronously. This contrasts with most MIMO control schemes, which assume that all variables are updated simultaneously. MPC outperforms other control strategies through its ability to deal with constraints. This requires on-line optimization, hence computational complexity can become an issue when applying MPC to complex systems with fast response times. The multiplexed MPC scheme described in this paper solves the MPC problem for each subsystem sequentially, and updates subsystem controls as soon as the solution is available, thus distributing the control moves over a complete update cycle. The resulting computational speed-up allows faster response to disturbances, which may result in improved performance, despite finding sub-optimal solutions to the original problem.



**Keywords:** Predictive control, distributed control, multivariable control, periodic systems, constrained control.



*This work is supported by A*STAR project "Model Predictive Control on a Chip" (Ref: 052-118-0059). Much of the material in this paper was previously presented at the IFAC World Congress 2005, and at the European Control Conference 2007. KV Ling would also like to acknowledge the support of Tan Chin Tuan Fellowship and of Pembroke College, Cambridge.

# 1 Introduction

## 1.1 The basic idea

Model Predictive Control (MPC) has become an established control technology in the petrochemical industry, and its use is currently being pioneered in an increasingly wide range of process industries [23, 34]. It is also being proposed for a range of higher bandwidth applications, such as ships [22], aerospace [21, 24], and road vehicles [20]. This paper is concerned with facilitating applications of MPC in which computational complexity, in particular computation time, is likely to be an issue. One can foresee that applications to embedded systems, with the MPC algorithm implemented in a chip or an FPGA [5, 13, 14, 17], are likely to run up against this problem.

MPC operates by solving an optimization problem on-line, in real time, to determine a plan for future operation. Only an initial portion of that plan is implemented, and the process is repeated, re-planning when new information becomes available. Since numerical optimization naturally handles hard constraints, MPC offers good performance while operating close to constraint boundaries [18]. Solving a numerical optimization can be a complex problem, and for situations in which computation is limited, the time to find the solution can be the limiting factor in the choice of the update interval. Most MPC theory to date, and as far as we know all implementations, assumes that all the control inputs are updated at the same instant. Suppose that a given MPC control problem can be solved in not less than $T$ seconds, so that the smallest possible update interval is $T$. The computational complexity of typical MPC problems, including time requirements, tends to vary as $O((m \times N_u)^3)$, where $m$ is the number of control inputs and $N_u$ is the horizon length. We propose to use MPC to update only one control variable at a time, but to exploit the reduced complexity to update successive inputs at intervals smaller than $T$, typically $T/m$. After $m$ updates a fresh cycle of updates begins, so that each whole cycle of updates repeats with cycle time $T$. We call this scheme *multiplexed MPC*, or MMPC. We assume that fresh measurements of the plant state are available at these reduced update intervals $T/m$. The main motivation for this scheme is the belief that in many cases the approximation involved in updating only one input at a time will be outweighed — as regards performance benefits — by the more rapid response to disturbances, which this scheme makes possible. It is often the case that "do something sooner" leads to better control than "do the optimal thing later". Fig. 1 shows the pattern of input moves in the MMPC scheme with $m = 3$, compared with the conventional scheme in which the three input moves are synchronized. We will refer to conventional MPC as *Synchronized MPC*, or SMPC, in the rest of this paper.

The scheme which we investigate here is close to common industrial practice in complex plants, where it is often impossible to update all the control inputs simultaneously, because of their sheer number, and the limitations of the communications channels between the controller and the actuators.

In addition to treating the 'nominal' MMPC case, in which the model is assumed to represent the plant perfectly, we extend MMPC to guarantee robust constraint satisfaction and feasibility of all optimizations despite the action of unknown but bounded

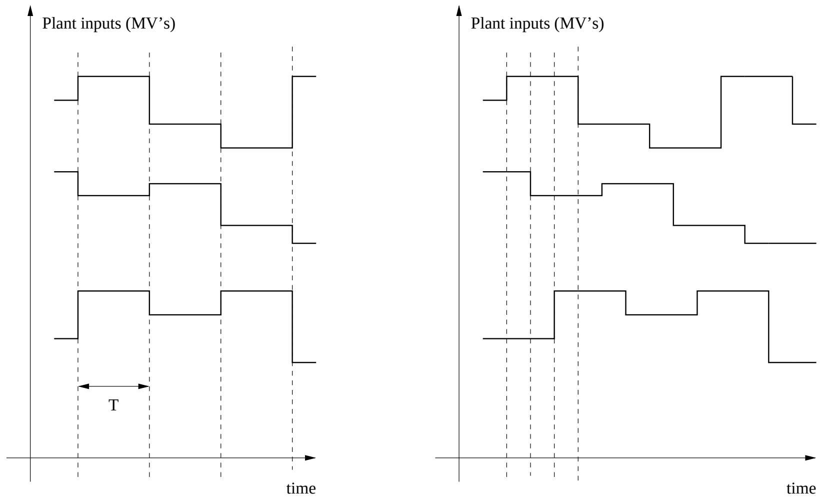



Figure 1: Patterns of input moves for conventional 'synchronized' MPC (left), and for the Multiplexed MPC (right) introduced in this paper.

disturbances. These are key issues in MPC: performance benefits are achieved by operating close to constraint boundaries, but when the state evolution no longer matches the predictions, constraint violation and infeasibility can result. Many methods have been developed to endow conventional synchronous MPC with robustness [19, 1]. For use with MMPC, we have adopted the *constraint tightening* approach [11, 8, 27, 25], in which the constraints of the optimization are modified to retain a margin for future feedback action. Since only the constraint limits are modified, the computational complexity remains the same as for the equivalent nominal MPC. Constraint tightening is therefore well-suited to MMPC, which is aimed at computation-limited applications.

Various generalizations of our scheme are possible. For example, subsets of control inputs might be updated simultaneously, perhaps all the inputs in each subset being associated with one subunit. The assumption of equal intervals between the updates of plant inputs is not essential to the MMPC idea. Any pattern of update intervals can be supported, providing that it repeats in subsequent update cycles. A further generalization, albeit involving a significantly harder problem, would be not to update each control input in a fixed sequence, but to decide in real time which input (if any) needs updating most urgently — one could call this *just-in-time MPC*.

### 1.2 Related Work

MMPC is related to distributed MPC (DMPC) [7], both dividing the optimisation into smaller sub-problems. Several works have been published which propose 'distributed MPC' in the sense that subsets of control inputs are updated by means of an MPC algorithm. But these usually assume that several sets of such computations are performed in parallel, on the basis of local measurements only, and that all the control inputs are then updated simultaneously. In some applications, such as formation flying of unmanned vehicles [10], it is assumed that the state vectors of subunits (vehicles) are distinct, and that coupling between subunits occurs only through constraints and performance measures. In [33] five different MPC-based schemes are proposed, of which four are distributed or decentralized MPC schemes of some kind. Their schemes 4 and 5 are the closest to our multiplexed scheme. In these schemes an MPC solution is solved iteratively for each control input, but it is assumed that no new sensor information arrives during the iteration, and that all the control inputs are updated simultaneously when the iterations have been completed.

Various approaches to robust DMPC have been investigated, including worst-case predictions [12], retention of "emergency" plans [31, 6], invariant "tube" predictions [32] and constraint tightening [26]. Work on DMPC has typically focussed on spatially distributed systems with some structure in the system, *e.g.* teams of vehicles with decoupled dynamics. In contrast, our new robust MMPC makes no assumptions on the overall system structure, and considers temporal distribution, breaking the optimisation down into a sequence of smaller problems, potentially on the same processor.

In [2] a similar scheme to ours is proposed, but it is assumed that a limitation occurs on network bandwidth rather than on central computing resources. Hence optimal trajectories are computed for all the plant inputs, but these are communicated to the plant one input (or one group of inputs) at a time — with the optimization taking this communication restriction into account. If the communication sequence is fixed and periodic then the scheme proposed in [2] is essentially the same as a version of MMPC to which we previously referred as 'scheme 1', except that we allowed constraints on inputs and states [16]. [2] also considers the case that a feedback law is fixed for each input (or group of inputs), that the inputs are updated according to some periodic scheme, and that a heuristic is used to determine (online) the best point in the period for a given state; this gives a heuristic version of 'just-in-time control' as defined above, though not really MPC any longer, since the feedback law is assumed to be predetermined. We emphasize that the driving factor behind the development of MMPC is operation in a processor-limited environment, motivating decomposition of the optimisation to reduce computational delays. Therefore we have not considered the impact of communication limits: indeed, in many applications of MMPC, the computation may take place serially on a single processor, and thus communication is not a concern.

In [30] MPC is considered with opposite assumptions to ours on update rates. There the plant inputs are considered to be updated relatively frequently, compared with the rate at which output measurements become available. This is in contrast to MMPC, in which the plant outputs are assumed to be measured relatively frequently, compared with the rate at which inputs are updated. It is remarked in [30] that the predictive control law which results (with the specific assumptions made there) is periodic, the period being the ratio of the input update rate to the output measurement rate (assuming this is an integer). A similar observation is central to the development in our section 3.

An alternative strategy for speeding up the computations involved in MPC is 'explicit MPC', which involves off-line precomputation of the 'pieces' of the piecewise-affine controller which is the optimal solution [20]. But that is not feasible if the number of 'pieces' required is excessively large, or if the constraints or the plant model change relatively frequently.

MMPC was introduced by us in [16]. Robust MMPC was first described in [28]. In [29] our MMPC idea was applied (by others) to the control of an aircraft engine. In [15] an experimental evaluation of MMPC is reported.

### 1.3 Structure of the paper

The rest of this paper is organized as follows. In Section 2 a formulation of MMPC is presented in detail. Section 3 establishes the nominal stability of MMPC with this formulation. Section 4 then derives a formula for the value of the cost function attained by MMPC. Section 5 develops a more elaborate formulation of MMPC, with the objective of guaranteeing robust feasibility, and establishes an appropriate theorem. Section 6 gives numerical simulation examples and compare the performance of MMPC with SMPC for cases with significant plant uncertainty, represented by unknown but bounded disturbances. Finally, concluding remarks are given in Section 7.

## 2 Problem formulation

### 2.1 Preliminary

We consider the following discrete-time linear plant model in state-space form, with state vector $x_k \in \mathbb{R}^n$ and $m$ (scalar) inputs $u_{1,k}, \ldots, u_{m,k}$:

$$x_{k+1} = Ax_k + \sum_{j=1}^{m} B_j \Delta u_{j,k} \tag{1}$$

where each $B_j$ is a column vector and $\Delta u_{j,k} = u_{j,k} - u_{j,k-1}$. (This could be generalized to the case where $B_j \in \mathbb{R}^{n \times p_j}$ and $\Delta u_{j,k} \in \mathbb{R}^{p_j}$, with $\sum_j p_j$ inputs.) We assume that $(A, [B_1, \ldots, B_m])$ is stabilizable. For ease of notation, when we drop the index $j$, we mean the complete $B$ matrix and the input vector so that the system (1) may be written as

$$x_{k+1} = Ax_k + B\Delta u_k$$

We assume that at time step $k$ the complete state vector $x_k$ is known exactly from measurements. We will consider only the regulation problem in detail, but tracking problems, especially those with non-zero constant references, can be easily transformed into equivalent regulation problems [3, sec.3.3].

Multiplexed MPC, at discrete-time index $k$, changes only plant input $\Delta u_{\sigma(k),k}$, where $\sigma(k)$ is an indexing function which identifies the input channel to be moved at each step, and is defined as:

$$\sigma(k) = (k \bmod m) + 1 \tag{2}$$

(We assume, without loss of generality, that we update input 1 at time index 0.) The asynchronous nature of the multiplexed control moves, as illustrated in Fig. 1, is captured by the constraint

$$\Delta u_{j,k} = 0 \text{ if } j \neq \sigma(k). \tag{3}$$

It is then possible to rewrite the system dynamics (1) as a linear periodically time-varying single-input system:

$$x_{k+1} = Ax_k + B_{\sigma(k)}\Delta\tilde{u}_k \tag{4}$$

where $\Delta\tilde{u}_k = \Delta u_{\sigma(k),k}$. From this point onwards, we use this periodic description of the plant so that we can draw on known results for periodic time-varying systems.

**Remark 1** *Some of the generalizations to which we alluded in section 1.1 could be treated by redefining the sequencing function $\sigma(\cdot)$ appropriately. For example, for a particular 3-input system, updating the inputs in the sequence $(1, 2, 1, 3, 1, 2, 1, 3, \ldots)$, thus updating one of the inputs twice as often as the others, could be represented in this way.*

The unique advantage of MPC, compared with other control strategies, is its capacity to take account of constraints in a systematic manner. As usual in MPC, we will suppose that constraints may exist on the input moves, $\Delta u_k \in \mathbb{U}_{\sigma(k)}$, and on states, $x_k \in \mathbb{X}$, where $\mathbb{X}$ and $\mathbb{U}_{\sigma(k)}$ are compact polyhedral sets containing the origin in their interior. Note that the control move set depends on the time, since the channel to be moved differs from step to step. If constraints on the actual control inputs $u$ are required, then $u$ must appear in the augmented state $x$, and those constraints can be incorporated in the state constraint set $\mathbb{X}$.

Let $N = (N_u - 1)m + 1$ where $N_u$ is the control horizon, a design parameter which will later be used to denote the number of control moves to be optimized *per input channel* of the original system (1). The $N$-step prediction model at time $k$ for the system described by (4) is

$$\vec{X}_{k+1|k} = \Phi x_{k|k} + G_{\sigma(k)}\Delta\vec{U}_{k|k} \tag{5}$$

where

$$\vec{X}_{k+1|k} = \begin{bmatrix} x_{k+1|k} \\ x_{k+2|k} \\ \vdots \\ x_{k+N|k} \end{bmatrix}, \qquad \Delta\vec{U}_{k|k} = \begin{bmatrix} \Delta\tilde{u}_{k|k} \\ \Delta\tilde{u}_{k+1|k} \\ \vdots \\ \Delta\tilde{u}_{k+N-1|k} \end{bmatrix}, \qquad \Phi = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix},$$

$$G_{\sigma(k)} = \begin{bmatrix} B_{\sigma(k)} & 0 & \ldots & 0 \\ AB_{\sigma(k)} & B_{\sigma(k+1)} & \ldots & 0 \\ \vdots & & \ddots & \\ A^{N-1}B_{\sigma(k)} & \ldots & AB_{\sigma(k+N-2)} & B_{\sigma(k+N-1)}) \end{bmatrix} \tag{6}$$

$\Delta\tilde{u}_{k+i|k}$ denotes the prediction made at time $k$ of a control move to be executed at time $k+i$, and $x_{k+i|k}$ denotes the corresponding prediction of $x_{k+i}$, made at time $k$.

## 2.2 The MMPC Algorithm

In the following, $K_{\sigma(k)}$ denotes a pre-specified stabilizing linear periodic state-feedback controller of (4); $\big(\mathcal{X}_I(K_{\sigma(k)})\big)$ denotes a sequence of sets in which none of the constraints is active, and which satisfies the 'periodic invariance' condition for the linear periodic system (4) when the feedback controller

$$\Delta\tilde{u}_k = -K_{\sigma(k)} x_k \tag{7}$$

is applied, namely

$$x_k \in \mathcal{X}_I(K_{\sigma(k)}) \Rightarrow -K_{\sigma(k)} x_k \in \mathbb{U}_{\sigma(k)} \text{ and } (A - B_{\sigma(k)} K_{\sigma(k)}) x_k \in \mathcal{X}_I(K_{\sigma(k+1)})$$

and of course $\mathcal{X}_I(K_{\sigma(k)}) \subseteq \mathbb{X}$, for $\sigma(k) = 1, \ldots, m$.

Some assumptions must be made about those inputs which have already been planned but which have not yet been executed. We will assume that all such planned decisions are known to the controller, and that it assumes that they will be executed as planned, i.e.,

$$\Delta\tilde{u}_{k+i|k} = \Delta\tilde{u}_{k+i|k-1}, \quad i \neq 0, m, 2m, \ldots \tag{8}$$

(In fact, new decisions will be made at time $k+i$ in the light of new measurements.)

MMPC solves the following finite-time constrained linear periodic control problem:

$$\begin{array}{lll} \mathcal{P}_{\sigma(k)}(x_k): & \text{Minimise} & J_k = F_{\sigma(k)}(x_{k+N|k}) + \sum_{i=0}^{N-1} \left( \|x_{k+i|k}\|_q^2 + \|\Delta\tilde{u}_{k+i|k}\|_r^2 \right) \\ & \text{wrt} & \Delta\tilde{u}_{k+i|k}, \quad (i = 0, m, 2m, \ldots, N-1) \\ & \text{s.t.} & \Delta\tilde{u}_{k+i|k} \in \mathbb{U}_{\sigma(k+i)}, \quad (i = 0, \ldots, N-1) \\ & & x_{k+i|k} \in \mathbb{X}, \quad (i = 1, \ldots, N-1) \\ & & x_{k+N|k} \in \mathcal{X}_I(K_{\sigma(k)}) \\ & & x_{k+i+1|k} = A x_{k+i|k} + B_{\sigma(k+i)} \Delta\tilde{u}_{k+i|k} \\ & & \Delta\tilde{u}_{k+i|k} = \Delta\tilde{u}_{k+i|k-1}, \quad (i \neq 0, m, \ldots, N-1) \end{array} \tag{9}$$

where $F_{\sigma(k)}(x_{k+N|k}) \geq 0$ is a suitably chosen terminal cost.

We denote the resulting optimizing control sequence as $\Delta\mathbf{u}^o(x_k)$. Only the first control $\Delta\tilde{u}_k^o$ in $\Delta\mathbf{u}^o(x_k)$ is applied to the system at time $k$, so that we apply the predictive control in the usual receding-horizon manner.

In MMPC, there are essentially $m$ MPC controllers, operating in sequence, in a cyclic manner. They share information, however, in the sense that the complete plant state is available to each controller — although not at the same times — and the currently planned future moves of each controller are also available to all the others.

For clarity, we set out the following algorithm which defines 'nominal' MMPC (as contrasted with 'robust' MMPC which will be introduced in section 5):

**Algorithm 1 (Nominal MMPC)**

1. *Set* $k := k_0$. *Initialise by solving problem* (9)*, but optimising over all the variables* $\Delta\tilde{u}_{k+i|k}, i = 0, 1, \ldots, N-1$.

2. *Apply control move* $\Delta u_{\sigma(k),k} = \Delta\tilde{u}_{k|k}$

3. *Store planned moves* $\Delta\vec{u}_{k,m|k}$.

4. *Pause for one time step, increment $k$, obtain new measurement $x_k$.*
5. *Solve problem* (9)*.*
6. *Go to step 2.*

Note that Step 1 involves solving for inputs across all channels, not just channel $\sigma(k)$. This type of initialisation requirement is common in distributed MPC. Subsequent results do not depend on the optimality of this initial solution, only its feasibility.

# 3 Stability of MMPC

In this section we establish sufficient conditions under which the MMPC scheme gives closed-loop stability. We then apply standard results on optimal control of periodic systems to our plant written in the form of (4), assuming that all constraints are inactive, to propose a terminal cost $F(\cdot)$ which, when used in the MMPC algorithm introduced in section 2.2, ensures stability of the closed loop even when constraints are active.

**Theorem 1** *MMPC, obtained by implementing the nominal MMPC Algorithm 1, gives closed-loop stability if the problems are well-posed, and if the set of terminal costs $\{F_\sigma(\cdot)\}$ satisfies*

$$F_{\sigma^+}([A - B_{\sigma^+}K_{\sigma^+}]x) + \|x\|_q^2 + \|K_{\sigma^+}x\|_r^2 \leq F_\sigma(x) \quad \text{for} \quad \sigma = 1, \ldots, m. \tag{10}$$

*where $\sigma^+ = (\sigma \bmod m) + 1$, namely the cyclical successor value to $\sigma$.*

**Proof:**
The proof follows a standard argument for MPC stability proofs (see [19] for example), adapted to our setting. It depends on the constrained optimization being feasible at each step, and the feasibility at any particular time step depends on the details of the constrained optimization problem that is being solved. For the nominal case with a perfect model and in the absence of disturbances, if feasible solutions are obtained over an initial period, then feasibility is assured thereafter.

Let $\Delta\tilde{u}^*_{k+i|k}$ denote the optimal solution to (9) at time step $k$, for $i = 0, m, 2m, \ldots, N-1$, let $x^*_{k+i|k}$ denote the corresponding state sequence, for $i = 1, 2, \ldots, N$, and let $J^*_k$ be the corresponding value of the cost function $J_k$.

Then a candidate input sequence to be applied to the plant at the next time step, $k+1$, is

$$\begin{aligned}&\Big(\Delta\tilde{u}^*_{k+1|k+1-m}, \Delta\tilde{u}^*_{k+2|k+2-m}, \ldots, \Delta\tilde{u}^*_{k+m|k},\\ &\Delta\tilde{u}^*_{k+1+m|k+1-m}, \Delta\tilde{u}^*_{k+2+m|k+2-m}, \ldots, \Delta\tilde{u}^*_{k+2m|k}, \ldots,\\ &\Delta\tilde{u}^*_{k+1+(N_u-2)m|k+1-m}, \Delta\tilde{u}^*_{k+2+(N_u-2)m|k+2-m}, \ldots,\\ &\qquad\qquad\qquad\qquad \Delta\tilde{u}^*_{k+(N_u-1)m|k}, -K_{\sigma(k+1)}x^*_{k+N|k}\Big)\end{aligned} \tag{11}$$

(recall that $N-1 = (N_u-1)m$). The input sequence applied at time step $k$ is the same, except that the initial term $\Delta\tilde{u}^*_{k|k}$ is pre-pended to it, and that the final term

$K_{\sigma(k+1)}x^*_{k+N|k}$ is omitted. Let the cost obtained with the candidate solution (11) be $\tilde{J}_{k+1}$. Then

$$\tilde{J}_{k+1} - J^*_k = ||K_{\sigma(k+1)}x^*_{k+N|k}||^2_r + F_{\sigma(k+1)}\left((A - B_{\sigma(k+1)}K_{\sigma(k+1)})x^*_{k+N|k}\right) - ||\Delta\tilde{u}^*_{k|k}||^2_r - ||x^*_{k|k}||^2_q - F_{\sigma(k)}(x_{k+N|k}) + \|x_{k+N|k}\|^2_q \quad (12)$$

Hence $\tilde{J}_{k+1} - J^*_k \leq 0$ if (10) holds. Now optimisation at time step $k+1$ will result in a value function

$$J^*_{k+1} \leq \tilde{J}_{k+1} \quad (13)$$

and hence

$$J^*_{k+1} \leq J^*_k \quad (14)$$

if (10) holds.

But $J^*_k \geq 0$ for all $k$, hence $J^*_{k+1} - J^*_k \to 0$. But, from (12)–(14) we have that

$$J^*_{k+1} - J^*_k \leq -\|x^*_{k|k}\|^2_q - \|\Delta u^*_{k|k}\|^2_r \quad (15)$$

Hence $x^*_{k|k} \to 0$ (and $\Delta u^*_{k|k} \to 0$). But $x_{k|k} = x_k$, so $x_k \to 0$.

■

**Remark 2** *Note the implicit assumption that $N$ is chosen sufficiently large to ensure feasibility of the constrained optimisation problem posed. Also note the assumption in each planning optimisation that the linear state feedback law* (7) *is applied at* every *step after the end of the optimisation horizon.*

The following results on unconstrained infinite-time linear quadratic control of periodic systems are known [4]. Consider the plant (4) and the quadratic cost function

$$J_k = \sum_{i=0}^{\infty}\left(\|x_{k+i}\|^2_q + \|\Delta\tilde{u}_{k+i}\|^2_r\right) \quad (16)$$

Then this cost is minimised by finding $\bar{P}_i, i = 1, \ldots, m$, the Symmetric, Periodic and Positive Semidefinite (SPPS) solution of the following discrete-time periodic Riccati equation (DPRE)

$$P_k = A^T P_{k+1} A - A^T P_{k+1} B_{\sigma(k)} (B^T_{\sigma(k)} P_{k+1} B_{\sigma(k)} + r)^{-1} B^T_{\sigma(k)} P_{k+1} A + q \quad (17)$$

and setting

$$\Delta\tilde{u}_k = -K_{\sigma(k)} x_k \quad (18)$$

where

$$K_{\sigma(k)} = (B^T_{\sigma(k)} \bar{P}_{\sigma(k+1)} B_{\sigma(k)} + r)^{-1} B^T_{\sigma(k)} \bar{P}_{\sigma(k+1)} A \quad (19)$$

Furthermore, the resulting minimal value of $J_k$ is given by $J^*_k = x^T_k \bar{P}_{\sigma(k)} x_k$. Thus one way of choosing a suitable set of terminal costs to satisfy (10) is to set

$$F_{\sigma(k)}(x) = x^T \bar{P}_{\sigma(k+N)} x \quad (20)$$

which leads to

$$J^*_{k+1} - J^*_k = -\|x^*_{k|k}\|^2_q - \|\Delta u^*_{k|k}\|^2_r \quad (21)$$

**Remark 3** *The terminal cost* (20) *would be the optimal cost-to-go if at each step* $k$ *the optimisation was over future values of* all *input channels, rather than those in input channel* $\sigma(k)$ *only. A version of MMPC in which this is done was called 'scheme 1 MMPC' in our earlier paper [16]. The version presented in this paper was called 'scheme 2' in [16]. We no longer advocate 'scheme 1', as it does not give any reduction of computational complexity, compared with conventional SMPC.*

# 4 Cost of MMPC when constraints are inactive

Each solution of the optimisation problem (9) depends on the plans made in previous optimisations. Hence the optimal cost obtained with MMPC, even in the case that all constraints are inactive, is not given by (20). In this section we will introduce an augmented state which includes those existing plans that are not going to be modified by the current optimisation. This will allow us to obtain, in Theorem 2, an expression for the optimal cost of the same form as (20). This will provide an analysis tool for predicting and comparing the performance of various MMPC designs. In the process we will see that MMPC can be rewritten in a more familiar MPC form, but with a periodically time-varying (augmented state) model.

Note that a similar development could be used to compute the optimal MMPC cost if the set of active constraints was constant and known. The nature of the MMPC control law in such circumstances is also linear periodic. Consequently the MMPC control law in the presence of constraints is piecewise-linear-periodic; as in the standard 'explicit' MPC case, the 'pieces' correspond to regions of the state space in which the set of active constraints remains constant.

The development of this section, in particular Theorem 2, facilitates performance evaluation of MMPC in certain circumstances. For example, it is useful for evaluating the trade-off between the restricted optimisation performed by MMPC and the reduced update rate available with conventional SMPC.

## 4.1 Unconstrained MMPC as periodic state feedback

We introduce the following definitions, which gather together those variables which are optimised at each step by the MMPC algorithm:

$$\Delta\vec{u}_{k,i|k} = \begin{bmatrix} \Delta\tilde{u}_{k+i|k} \\ \Delta\tilde{u}_{k+m+i|k} \\ \vdots \\ \Delta\tilde{u}_{k+(N_u-2)m+i|k} \end{bmatrix} \tag{22}$$

for $i = 1, 2, \ldots, m$.

$$\Delta\vec{u}_{0|k} = \begin{bmatrix} \Delta\tilde{u}_{k|k} \\ \Delta\tilde{u}_{k+m|k} \\ \vdots \\ \Delta\tilde{u}_{k+(N_u-1)m|k} \end{bmatrix} = \begin{bmatrix} \Delta\tilde{u}_{k|k} \\ \Delta\vec{u}_{k,m|k} \end{bmatrix} \tag{23}$$

Recall that $N = (N_u - 1)m + 1$ where $N_u$ is the control horizon, a design parameter which denotes the number of control moves to be optimized per input channel of the original system (1). By grouping the predicted control signals into $m$ vectors, the prediction model (5) can be re-written as

$$\vec{X}_{k+1|k} = \Phi x_{k|k} + g_1^{\sigma(k)} \Delta\vec{u}_{0|k} + g_2^{\sigma(k)} \Delta\vec{u}_{k,1|k} + \ldots + g_m^{\sigma(k)} \Delta\vec{u}_{k,m-1|k} \tag{24}$$

where $\Delta\vec{u}_{k,i|k}$ and $\Delta\vec{u}_{0|k}$ are as defined in (22) and (23), respectively, and $g_i^{\sigma(k)}$, $(i = 1, \ldots, m)$ are matrices whose columns are columns of the $G_{\sigma(k)}$ matrix (6), namely, $g_1^{\sigma(k)}$ is the matrix whose columns are columns $1, 1+m, \ldots, 1+(N_u-1)m$ of the matrix $G_{\sigma(k)}$, while $g_i^{\sigma(k)}$, $(i = 2, \ldots, m)$ contains columns $i, i+m, \ldots, i+(N_u-2)m$ columns of $G_{\sigma(k)}$.

In MMPC only $\Delta\vec{u}_{0|k}$ is taken as the decision variable at time $k$, and appropriate assumptions are made about $\Delta\vec{u}_{k,i|k}$, $i = 1, \ldots, m-1$. Note that the length of $\Delta\vec{u}_{0|k}$ is $N_u$ while the length of $\Delta\vec{u}_{k,i|k}$ for $i = 1, \ldots, m-1$ is $N_u - 1$. When $N_u = 1$, $\Delta\vec{u}_{k,i|k}$, $i = 1, \ldots, m-1$, become zero-length vectors. In MMPC we assume that the $\Delta\vec{u}_{k,i|k}, i = 1, \ldots, m-1$ are those inputs which have already been planned in previous steps but have not yet been executed, namely

$$\Delta\vec{u}_{k,i|k} = \Delta\vec{u}_{k-1,i+1|k-1}, \qquad (i = 1, \ldots, m-1). \tag{25}$$

We define the vector $\Delta\vec{u}^p_{k|k-1}$ which holds the previously planned but not yet executed control moves as

$$\Delta\vec{u}^p_{k|k-1} = \begin{bmatrix} \Delta\vec{u}_{k-1,2|k-1} \\ \Delta\vec{u}_{k-1,3|k-1} \\ \vdots \\ \Delta\vec{u}_{k-1,m|k-1} \end{bmatrix} \tag{26}$$

Thus, it can be deduced from (24) that, if no constraints are active, then the MMPC control law is a linear periodic state feedback:

$$\Delta\vec{u}_{0|k} = \tilde{K}_{\sigma(k)} \xi_k \tag{27}$$

where we have introduced the augmented state vector

$$\xi_k = \begin{bmatrix} x_k \\ \Delta\vec{u}^p_{k|k-1} \end{bmatrix} \tag{28}$$

### 4.2 A formula for the MMPC Cost

Using the augmented state vector introduced in (28), the dynamics of the plant operating under MMPC can be expressed as

$$\xi_{k+1} = \tilde{A}\xi_k + \tilde{B}_{\sigma(k)} \Delta\vec{u}_{k,0|k} \tag{29}$$

where

$$\tilde{A} = \begin{bmatrix} A & 0 \\ 0 & A_u \end{bmatrix} \quad \tilde{B}_{\sigma(k)} = \begin{bmatrix} B_{\sigma(k)} & 0 \\ 0 & B_u \end{bmatrix}$$

$$A_u = \begin{bmatrix} 0 & I & 0 & \cdots & 0 \\ 0 & 0 & I & & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & & \vdots \\ 0 & & \cdots & \cdots & 0 \end{bmatrix} \qquad B_u = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ I \end{bmatrix}$$

The value of the quadratic cost (see also Remark 4)

$$J_k = \sum_{i=0}^{\infty} \left( \|\xi_{k+i+1}\|_{\tilde{q}}^2 + \|\Delta \vec{u}_{0|k+i}\|_{\tilde{r}}^2 \right) \tag{30}$$

when control law of the form (27) is applied, is given by the following theorem:

**Theorem 2** *The value of the cost* (30) *for the system* (29)*, when any stabilising linear periodic state feedback of the form* (27) *is applied, is given by*

$$\tilde{J}_k = J_{\xi,k} + J_{u,k} = \xi_k^T P_{\xi+u,\sigma(k)} \xi_k \tag{31}$$

*where* $J_{\xi,k} = \xi_k^T P_{\xi,\sigma(k)} \xi_k$ *and* $J_{u,k} = \xi_k^T P_{u,\sigma(k)} \xi_k$*, and* $P_{\xi,\sigma(k)}$*,* $P_{u,\sigma(k)}$ *and* $P_{\xi+u,\sigma(k)}$ *are, respectively, solutions of the following Lyapunov equations:*

$$P_{\xi,\sigma(k)} = \Psi_{\sigma(k)}^T P_{\xi,\sigma(k)} \Psi_{\sigma(k)} + \bar{\Phi}_{\sigma(k)}^T Q \bar{\Phi}_{\sigma(k)} \tag{32}$$

$$P_{u,\sigma(k)} = \Psi_{\sigma(k)}^T P_{u,\sigma(k)} \Psi_{\sigma(k)} + \bar{K}_{\sigma(k)}^T R \bar{K}_{\sigma(k)} \tag{33}$$

$$\begin{aligned} P_{\xi+u,\sigma(k)} &= \Psi_{\sigma(k)}^T P_{\xi+u,\sigma(k)} \Psi_{\sigma(k)} + \bar{\Phi}_{\sigma(k)}^T Q \bar{\Phi}_{\sigma(k)} \\ &\quad + \bar{K}_{\sigma(k)}^T R \bar{K}_{\sigma(k)} \end{aligned} \tag{34}$$

*where*

$$\begin{aligned} \Psi_{\sigma(k)} &= \tilde{\Phi}_{\sigma(k+m-1)} \cdots \tilde{\Phi}_{\sigma(k+1)} \tilde{\Phi}_{\sigma(k)} \\ \tilde{\Phi}_{\sigma(k)} &= \tilde{A} + \tilde{B}_{\sigma(k)} \tilde{K}_{\sigma(k)} \\ \bar{\Phi}_{\sigma(k)} &= \begin{bmatrix} \tilde{\Phi}_{\sigma(k)} \\ \tilde{\Phi}_{\sigma(k+1)} \tilde{\Phi}_{\sigma(k)} \\ \vdots \\ \tilde{\Phi}_{\sigma(k+m-1)} \cdots \tilde{\Phi}_{\sigma(k)} \end{bmatrix} \\ \bar{K}_{\sigma(k)} &= \begin{bmatrix} \tilde{K}_{\sigma(k)} \\ \tilde{K}_{\sigma(k+1)} \tilde{\Phi}_{\sigma(k)} \\ \vdots \\ \tilde{K}_{\sigma(k+m-1)} \tilde{\Phi}_{\sigma(k+m-2)} \cdots \tilde{\Phi}_{\sigma(k)} \end{bmatrix} \\ Q &= diag(\tilde{q},\ \tilde{q}, \cdots, \tilde{q}) \qquad and \\ R &= diag(\tilde{r},\ \tilde{r}, \cdots, \tilde{r}) \end{aligned}$$

**Proof:**
The closed-loop dynamics of the system (29), when stabilising linear periodic state feedback of the form (27) is applied, is given by

$$\xi_{k+1} = \tilde{A} \xi_k + \tilde{B}_{\sigma(k)} \Delta \vec{u}_{0|k} = (\tilde{A} + \tilde{B}_{\sigma(k)} \tilde{K}_{\sigma(k)}) \xi_k$$

or in general

$$\xi_{k+i+1} = \tilde{\Phi}_{\sigma(k+i)}\xi_{k+i}, \quad i = 0, 1, \ldots$$

where

$$\tilde{\Phi}_{\sigma(k+i)} = \tilde{A} + \tilde{B}_{\sigma(k+i)}\tilde{K}_{\sigma(k+i)}$$

Then

$$\begin{bmatrix} \xi_{k+jm+1} \\ \xi_{k+jm+2} \\ \vdots \\ \xi_{k+jm+m} \end{bmatrix} = \underbrace{\begin{bmatrix} \tilde{\Phi}_{\sigma(k)} \\ \tilde{\Phi}_{\sigma(k+1)}\tilde{\Phi}_{\sigma(k)} \\ \vdots \\ \tilde{\Phi}_{\sigma(k+m-1)}\cdots\tilde{\Phi}_{\sigma(k)} \end{bmatrix}}_{\bar{\Phi}_{\sigma(k)}}\xi_{k+jm}$$

and

$$\begin{aligned} \begin{bmatrix} \Delta\vec{u}_{0|k+jm} \\ \Delta\vec{u}_{0|k+jm+1} \\ \vdots \\ \Delta\vec{u}_{0|k+jm+m-1} \end{bmatrix} &= \begin{bmatrix} \tilde{K}_{\sigma(k)}\xi_{k+jm} \\ \tilde{K}_{\sigma(k+1)}\xi_{k+jm+1} \\ \vdots \\ \tilde{K}_{\sigma(k+m-1)}\xi_{k+jm+m-1} \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} \tilde{K}_{\sigma(k)} \\ \tilde{K}_{\sigma(k+1)}\tilde{\Phi}_{\sigma(k)} \\ \vdots \\ \tilde{K}_{\sigma(k+m-1)}\tilde{\Phi}_{\sigma(k+m-2)}\ldots\tilde{\Phi}_{\sigma(k)} \end{bmatrix}}_{\bar{K}_{\sigma(k)}}\xi_{k+jm} \end{aligned}$$

Thus,

$$\begin{aligned} J_{\xi,k} &= \sum_{i=0}^{\infty}\|\xi_{k+i+1}\|_{\tilde{q}}^2 \\ &= \sum_{j=0}^{\infty}\sum_{i=1}^{m}\|\xi_{k+jm+i}\|_{\tilde{q}}^2 \\ &= \sum_{j=0}^{\infty}\xi_{k+jm}^T\bar{\Phi}_{\sigma(k)}^T Q\bar{\Phi}_{\sigma(k)}\xi_{k+jm} \\ &= \xi_k^T[\sum_{j=0}^{\infty}(\Psi_{\sigma(k)}^j)^T[\bar{\Phi}_{\sigma(k)}^T Q\bar{\Phi}_{\sigma(k)}]\Psi_{\sigma(k)}^j]\xi_k \\ &= \xi_k^T P_{\xi,\sigma(k)}\xi_k \end{aligned}$$

where

$$\begin{aligned} Q &= diag(\tilde{q},\ \tilde{q}, \cdots, \tilde{q}) \\ \Psi_{\sigma(k)} &= \tilde{\Phi}_{\sigma(k+m-1)}\cdots\tilde{\Phi}_{\sigma(k+1)}\tilde{\Phi}_{\sigma(k)} \end{aligned}$$

and

$$P_{\xi,\sigma(k)} = \sum_{j=0}^{\infty}(\Psi^j_{\sigma(k)})^T[\tilde{\Phi}^T_{\sigma(k)}Q\tilde{\Phi}_{\sigma(k)}]\Psi^j_{\sigma(k)}$$

is a convergent series since the controller $\tilde{K}_{\sigma(k)}$ is stabilizing. Thus $P_{\xi,\sigma(k)}$ can be computed by solving the following Lyapunov equation

$$P_{\xi,\sigma(k)} = \Psi^T_{\sigma(k)}P_{\xi,\sigma(k)}\Psi_{\sigma(k)} + \bar{\Phi}^T_{\sigma(k)}Q\bar{\Phi}_{\sigma(k)}$$

Similarly, the sum of the control increments can be computed as

$$\begin{aligned}
J_{u,k} &= \sum_{i=0}^{\infty}\|\Delta\vec{u}_{0|k+i}\|^2_{\tilde{r}} = \sum_{j=0}^{\infty}\sum_{i=0}^{m-1}\|\Delta\vec{u}_{0|k+jm+i}\|^2_{\tilde{r}} = \sum_{j=0}^{\infty}\sum_{i=0}^{m-1}\|\bar{K}_{\sigma(k+i)}\xi_{k+jm+i}\|^2_{\tilde{r}} \\
&= \sum_{j=0}^{\infty}\xi^T_{k+jm}\bar{K}^T_{\sigma(k)}R\bar{K}_{\sigma(k)}\xi_{k+jm} \\
&= \xi^T_k\sum_{j=0}^{\infty}(\Psi^j_{\sigma(k)})^T[\bar{K}^T_{\sigma(k)}R\bar{K}_{\sigma(k)}]\Psi^j_{\sigma(k)}\xi_k \\
&= x^T_kP_{u,\sigma(k)}x_k
\end{aligned}$$

where

$$R = diag(\tilde{r},\ \tilde{r},\cdots,\tilde{r})$$

and

$$P_{u,\sigma(k)} = \sum_{i=0}^{\infty}(\Psi^i_{\sigma(k)})^T[\bar{K}^T_{\sigma(k)}R\bar{K}_{\sigma(k)}]\Psi^i_{\sigma(k)}$$

which can be computed by solving the Lyapunov equation

$$P_{u,\sigma(k)} = \Psi^T_{\sigma(k)}P_{u,\sigma(k)}\Psi_{\sigma(k)} + \bar{K}^T_{\sigma(k)}R\bar{K}_{\sigma(k)}$$

Finally, let

$$\begin{aligned}
P_{\xi+u,\sigma(k)} &= P_{\xi,\sigma(k)} + P_{u,\sigma(k)} \\
&= \sum_{j=0}^{\infty}(\Psi^j_{\sigma(k)})^T[\bar{\Phi}^T_{\sigma(k)}Q\bar{\Phi}_{\sigma(k)}]\Psi^j_{\sigma(k)} + \sum_{i=0}^{\infty}(\Psi^i_{\sigma(k)})^T[\bar{K}^T_{\sigma(k)}R\bar{K}_{\sigma(k)}]\Psi^i_{\sigma(k)}
\end{aligned}$$

and it is clear that $P_{\xi+u,\sigma(k)}$ can be computed as the solution of the Lyapunov equation

$$P_{\xi+u,\sigma(k)} = \Psi^T_{\sigma(k)}P_{\xi+u,\sigma(k)}\Psi_{\sigma(k)} + \bar{\Phi}^T_{\sigma(k)}Q\bar{\Phi}_{\sigma(k)} + \bar{K}^T_{\sigma(k)}R\bar{K}_{\sigma(k)}$$

■

The cost (16), when the MMPC control law of the form (27) is applied, is given by (30) with

$$\tilde{q} = \begin{bmatrix} q & 0 \\ 0 & 0 \end{bmatrix}, \quad \tilde{r} = \begin{bmatrix} r & 0 \\ 0 & 0 \end{bmatrix} \tag{35}$$

and the corresponding initial conditions hold on $\xi_k$; for example, at the beginning of an MMPC run it may be appropriate to set

$$\xi_k = \begin{bmatrix} x_k^T & (\Delta \vec{u}^p_{k|k-1})^T \end{bmatrix}^T = \begin{bmatrix} x_k^T & 0 \end{bmatrix}^T$$

**Remark 4** *Note that the cost defined in* (30) *differs in the first term from that defined in* (9)*. That is, in* (30) *we do not include any contribution from $x_{k|k}$, since that is fixed and cannot be influenced by the optimisation at time step $k$.*

**Remark 5** *It is seen that the system* (29) *is linear and periodic, while the cost* (30) *is quadratic with constant coefficients. Thus the optimal control law can be obtained from the theory given in [4], and is of the form* (27)*. This suggests yet another method to compute the terminal cost $F_{\sigma(k)}(x_{k+N|k})$ to ensure nominal stability of MMPC in addition to that presented in Theorem 1. A family of MMPC designs may be obtained by optimising the cost function* (30) *subject to the system* (29) *by choosing appropriate $\tilde{q}$ and $\tilde{r}$ matrices.*

**Remark 6** *If the second part of $\xi_k$, namely $\Delta \vec{u}^p_{k|k-1}$, were included in the optimisation, so that the optimal cost became a function of $x_k$ only, then the optimal cost, and the optimal solution, would be the same as that obtained with 'scheme 1' in our earlier papers, namely it would correspond to the cost resulting from allowing each 'agent' to optimise* all *future inputs rather than just 'its own' input.*

# 5 Robust MMPC

This section develops a robust version of MMPC. Uncertainty is introduced into the plant model as a bounded disturbance. The constraints which appear in the MMPC algorithm are then modified so that robust feasibility can be guaranteed, providing that it is achieved initially.

The plant dynamics (1) are now extended to include an unmeasured but bounded disturbance $w_k$:

$$x_{k+1} = Ax_k + \sum_{j=1}^{m} B_j \Delta u_{j,k} + Ew_k. \tag{36}$$

where $w_k$ satisifies

$$w_k \in \mathcal{W} \ \forall k \tag{37}$$

and $\mathcal{W}$ is a known, bounded set containing the origin.

As explained in Section 2, the system dynamics (36) can be re-written as a periodic linear system

$$x_{k+1} = Ax_k + B_{\sigma(k)}\Delta\tilde{u}_k + Ew_k \tag{38}$$

For robust MMPC we solve the following finite-time constrained linear periodic control problem, which we denote $\mathcal{P}_{\sigma(k)}(x_{k|k}, \Delta\vec{u}^p_{k|k-1}, w_{k-1})$:

$$\begin{array}{ll}
\text{Minimize} & J_k = F_{\sigma(k)}(x_{k+N|k}) + \sum_{i=0}^{N-1}\left(\|x_{k+i|k}\|_q^2 + \|\tilde{u}_{k+i|k}\|_r^2\right) \\
\text{wrt} & \Delta\tilde{u}_{k+i|k} \quad (i = 0, m, 2m, \ldots, N-1) \\
\text{s.t.} & \Delta\tilde{u}_{k+i|k} \in \mathcal{U}_{i,\sigma(k)}, \quad (i = 0, 1, \ldots, N-1) \\
& x_{k+i|k} \in \mathcal{X}_{i,\sigma(k)}, \quad (i = 1, 2, \ldots, N-1) \\
& x_{k+N|k} \in \mathcal{T}_{\sigma(k)} \\
& x_{k+i+1|k} = Ax_{k+i|k} + B_{\sigma(k+i)}\Delta\tilde{u}_{k+i|k}, \quad (i = 0, 1, \ldots) \\
& \Delta\tilde{u}_{k+i|k} = \Delta\tilde{u}_{k+i|k-1} + M_{i,\sigma(k)}Ew_{k-1}, \ \forall i \neq jm
\end{array} \tag{39}$$

Note some differences from (9). The predicted inputs and states are constrained to lie in sets $\mathcal{U}_{i,\sigma(k)}$ and $\mathcal{X}_{i,\sigma(k)}$ which depend on how far into the prediction horizon they are, as well as on $k$. The target set at the end of the horizon has been modified from $\mathcal{X}_{\mathcal{I}}(K_{\sigma(k)})$ to $\mathcal{T}_{\sigma(k)}$. Finally, the inputs which are not being optimised are assumed to be modified from their previously planned values by the feedback term $M_{i,\sigma(k)}Ew_{k-1}$; note that the value of $Ew_{k-1}$ can be inferred from data $\{u_{j-1}, x_j : j \leq k\}$.

The constraint sets $\mathcal{U}_{i,\sigma(k)}, \mathcal{X}_{i,\sigma(k)}$ and $\mathcal{T}_{\sigma(k)}$ are constructed to ensure robust feasibility, such that if some solution

$$\Delta\vec{U}^*_{k|k} = \left(\Delta\tilde{u}^*_{k|k},\ \Delta\tilde{u}^*_{k+1|k},\ \Delta\tilde{u}^*_{k+2|k},\ \ldots,\ \Delta\tilde{u}^*_{k+N-1|k}\right)^T \tag{40}$$

is feasible at some time $k$ then a candidate solution

$$\widehat{\Delta U}_{k+1} = \begin{pmatrix} \Delta\tilde{u}^*_{k+1|k} + M_{0,\sigma(k+1)}Ew_k \\ \vdots \\ \Delta\tilde{u}^*_{k+N-1|k} + M_{N-2,\sigma(k+1)}Ew_k \\ -K_{\sigma(k+1)}x^*_{k+N|k} + M_{N-1,\sigma(k+1)}Ew_k \end{pmatrix} \tag{41}$$

is feasible at time $k+1$ for all $w_k \in \mathcal{W}$. The designer chooses the feedback parameters $M_{i,\sigma(k)}$ and $K_{\sigma(k)}$ offline (as in [27], on which this development is based).

To achieve this robust feasibility property, the state constraints $x_k \in \mathbb{X}$ are tightened using a recursion

$$\mathcal{X}_{0,\sigma(k)} = \mathbb{X} \tag{42a}$$

$$\mathcal{X}_{i+1,\sigma(k)} = \mathcal{X}_{i,\sigma(k+1)} \sim L_{i,\sigma(k+1)}E\mathcal{W} \tag{42b}$$

where

$$L_{0,\sigma(k)} = I \tag{43a}$$

$$L_{i+1,\sigma(k)} = AL_{i,\sigma(k)} + B_{\sigma(k+i)}M_{i,\sigma(k)} \tag{43b}$$

for the chosen feedback policy $M_{i,\sigma(k)}$ and the "$\sim$" operator denotes the Pontryagin difference between two sets:

$$\mathcal{A} \sim \mathcal{B} = \{a \mid a + b \in \mathcal{A}\ \forall b \in \mathcal{B}\} \tag{44}$$

Similarly, the input move constraint sets $\Delta\tilde{u}_k \in \mathbb{U}_k$ are tightened using the following recursion

$$\mathcal{U}_{0,\sigma(k)} = \mathbb{U}_{\sigma(k)} \tag{45a}$$

$$\mathcal{U}_{i,\sigma(k)} = \mathcal{U}_{i-1,\sigma(k+1)} \sim M_{i-1,\sigma(k+1)} E\mathcal{W} \tag{45b}$$

The terminal sets $\mathcal{T}_{\sigma(k)}$ have the robust invariance properties that, if $x \in \mathcal{T}_{\sigma(k)}$ and $w \in \mathcal{W}$ then

$$\left(A - B_{\sigma(k+N)} K_{\sigma(k+1)}\right) x + \left[A L_{N-1,\sigma(k+1)} + B_{\sigma(k+N)} M_{N-1,\sigma(k+1)}\right] E w \in \mathcal{T}_{\sigma(k+1)} \tag{46a}$$

$$-K_{\sigma(k+1)} x \in \mathcal{U}_{N,\sigma(k)} \tag{46b}$$

and

$$\mathcal{T}_{\sigma(k)} \subseteq \mathcal{X}_{N,\sigma(k)}. \tag{46c}$$

The parameters $M_{i,\sigma(k)}$ and $K_{\sigma(k)}$ are chosen by the designer. The parameters $L_{i,\sigma(k)}$, which relate the control perturbations in (41) to the corresponding changes in the state predictions, are then fixed by (43). These settings determine the amount of constraint tightening applied in (42). Typically, to achieve a large feasible region, the control policy chosen should minimise the quantities limited by the constraints.

A restrictive but convenient choice of candidate policy is to select $M_{i,\sigma(k)}$, $i = 0, \ldots, N-2$ such that $L_{N,\sigma(k)} = 0\ \forall k$ and then set $M_{N-1,\sigma(k)} = 0$, $K_{\sigma(k)} = 0$ and $\mathcal{T}_{\sigma(k)} = \{0\}\ \forall k$.

The following algorithm defines robust MMPC. It uses notations defined in (1), (4) and (23). It is the same as Algorithm 1 except that problem (39) is solved instead of problem (9).

**Algorithm 2 (Robust MMPC)**

1. *Set $k := k_0$. Initialise by solving problem* (39)*, but optimising over all the variables $\Delta\tilde{u}_{k+i|k}, i = 0, 1, \ldots, N-1$.*
2. *Apply control move $\Delta u_{\sigma(k),k} = \Delta\tilde{u}_{k|k}$*
3. *Store planned moves $\Delta\vec{u}_{k,m|k}$.*
4. *Pause for one time step, increment $k$, obtain new measurement $x_k$.*
5. *Solve problem* (39)*.*
6. *Go to step 2.*

We will need the following result concerning the use of the $L_{i,\sigma(k)}$ matrices.

**Lemma 1** *Suppose that* $x_{k+1} = x_{k+1|k} + Ew_k$ *and*

$$\Delta\tilde{u}_{k+j|k+1} = \Delta\tilde{u}_{k+j|k} + M_{j-1,\sigma(k+1)}Ew_k, \quad (j = 1, 2, \ldots, i) \tag{47}$$

*Then*

$$x_{k+i|k+1} = x_{k+i|k} + L_{i-1,\sigma(k+1)}Ew_k, \quad (i = 1, 2, \ldots) \tag{48}$$

**Proof:** We prove the lemma by induction on $i$. Suppose the result is true for some $i$. Then

$$x_{k+i+1|k+1} = Ax_{k+i|k+1} + B_{\sigma(k+i)}\Delta\tilde{u}_{k+i|k+1} \tag{49}$$
$$= A[x_{k+i|k} + L_{i-1,\sigma(k+1)}Ew_k] + B_{\sigma(k+i)}\Delta\tilde{u}_{k+i|k+1} \tag{50}$$

But

$$x_{k+i+1|k} = Ax_{k+i|k} + B_{\sigma(k+i)}\Delta\tilde{u}_{k+i|k} \tag{51}$$

and, by assumption,

$$\Delta\tilde{u}_{k+i|k+1} = \Delta\tilde{u}_{k+i|k} + M_{i-1,\sigma(k+1)}Ew_k \tag{52}$$

so that

$$x_{k+i+1|k+1} = x_{k+i+1|k} + [AL_{i-1,\sigma(k+1)} + B_{\sigma(k+i)}M_{i-1,\sigma(k+1)}]Ew_k \tag{53}$$
$$= x_{k+i+1|k} + L_{i,\sigma(k+1)}Ew_k \quad \text{because of (43b)} \tag{54}$$

and hence the claimed result is true for $i + 1$.

Now consider $i = 1$: $x_{k+1|k+1} = x_{k+1} = x_{k+1|k} + Ew_k$, so the claimed result holds for $i = 1$, since $L_{0,\sigma(k+1)} = I$, by definition (43a).

Thus the result holds for $i \geq 1$. ∎

**Theorem 3** *If the system (36) is controlled using Algorithm 2 and the initial optimisation at time* $k = k_0$ *(*ie *step 1 of the algorithm) can be solved, and* $x_{k_0} \in \mathbb{X}$*, then* $(i)$ *the optimisation remains feasible and* $(ii)$ *the constraints* $x_k \in \mathbb{X}$ *and* $\Delta\tilde{u}_k \in \mathbb{U}_k$ *are satisfied for* $k > k_0$ *and for all disturbances satisfying (37).*

**Proof:** $(i)$ We will begin by showing that, by construction of the constraints in (42), feasibility at any time $k$ implies feasibility at time $k + 1$. In particular, we will demonstrate feasibility by establishing that the candidate solution (41) satisfies all the constraints of the optimisation. Therefore, feasibility at time $k = k_0$ implies feasibility at all future times $k > k_0$.

Assume that we have a feasible solution (40) at time $k$, and that $\Delta\tilde{u}^*_{k|k}$ is applied as input to the plant (38). This results in the next plant state being

$$x_{k+1} = Ax_k + B_{\sigma(k)}\Delta\tilde{u}^*_{k|k} + Ew_k = x_{k+1|k} + Ew_k \tag{55}$$

Thus from (41) and Lemma 1 we have

$$x_{k+i+1|k+1} = x_{k+i+1|k} + L_{i,\sigma(k+1)}Ew_k \tag{56a}$$
$$\Delta\tilde{u}_{k+i+1|k+1} = \Delta\tilde{u}_{k+i+1|k} + M_{i,\sigma(k+1)}Ew_k \ (i = 0, 1, \ldots, N-1) \tag{56b}$$

and since (40) was assumed feasible, we know $x_{k+i+1|k} \in \mathcal{X}_{i+1,\sigma(k)}$ and $\Delta\tilde{u}_{k+i+1|k} \in \mathcal{U}_{i+1,\sigma(k)}$. Combining this with (56), the definition of the Pontryagin difference (44) and the recursions (42b) and (45b), we know $x_{k+i+1|k+1} \in \mathcal{X}_{i,\sigma(k+1)}$ and $\Delta\tilde{u}_{k+i+1|k+1} \in \mathcal{U}_{i,\sigma(k+1)}$ for all $w_k \in \mathcal{W}$.

We also need to show that $x_{k+N+1|k+1} \in \mathcal{T}_{\sigma(k+1)}$ if the candidate solution (41) is applied. We have $x_{k+N|k} \in \mathcal{T}_{\sigma(k)}$ by the assumption of feasibility at time $k$.

$$x_{k+N+1|k+1} = Ax_{k+N|k+1} + B_{\sigma(k+N)}\Delta\tilde{u}_{k+N|k+1} \tag{57}$$

But, from (41),

$$\Delta\tilde{u}_{k+N|k+1} = -K_{\sigma(k+1)}x_{k+N|k} + M_{N-1,\sigma(k+1)}Ew_k \tag{58}$$

and, from Lemma 1 (since $\Delta\tilde{u}_{k+N|k} = -K_{\sigma(k+1)}x_{k+N|k}$),

$$x_{k+N|k+1} = x_{k+N|k} + L_{N-1,\sigma(k+1)}Ew_k \tag{59}$$

Hence, substituting (58) and (59) into (57) gives

$$\begin{aligned} x_{k+N+1|k+1} &= \left[A - B_{\sigma(k+N)}K_{\sigma(k+1)}\right]x_{k+N|k} + \\ &\quad + \left[AL_{N-1,\sigma(k+1)} + B_{\sigma(k+N)}M_{N-1,\sigma(k+1)}\right]Ew_k && (60) \\ &\in \mathcal{T}_{\sigma(k+1)} \quad \text{because of (46a).} && (61) \end{aligned}$$

Having established $x_{k+i+1|k} \in \mathcal{X}_{i+1,\sigma(k)}$, $\Delta\tilde{u}_{k+i+1|k} \in \mathcal{U}_{i+1,\sigma(k)}$ and $x_{k+N+1|k+1} \in \mathcal{T}_{\sigma(k+1)}$ for all $w_k \in \mathcal{W}$, the feasibility of the candidate solution has been proven, and thus the feasibility of the optimisation is proven.

$(ii)$ It remains to show that the state and input constraints are satisfied. Feasibility at all steps demands that $x_k = x_{k|k} \in \mathcal{X}_{0,\sigma(k)}$ which from (42a) implies $x_k \in \mathbb{X}$. Similarly, $\Delta\tilde{u}_k = \Delta\tilde{u}_{k|k} \in \mathcal{U}_{0,\sigma(k)}$ which from (45a) implies $\Delta\tilde{u}_k \in \mathbb{U}_k$.

■

# 6 Examples

This section demonstrates the potential benefits of MMPC by employing it in simulation for the control of three different example systems. In the first example we consider nominal MMPC; we show how the cost formula can be used to evaluate some of the design choices. In the second and third examples comparisons are made between the robust MMPC scheme and standard — but also robustified — "synchronous" MPC (SMPC). In these examples all simulations were performed on the same PC with a 3.2GHz Intel Pentium 4 processor and 1GB RAM. Matlab version 7.1 (R14, Service Pack 3) was employed, using Simulink to simulate the system dynamics and the "quadprog" optimisation function to solve the necessary quadratic programming (QP) problems. Computation times were measured using the Matlab profiler.

## 6.1 Nominal MMPC: Effects of $\mathbf{N_u}$ and updating sequence

In this section, numerical examples will be given to illustrate how the cost formula for MMPC can be used for evaluating the effect of various values of $N_u$, and of the updating sequence, on the closed-loop performance, when constraints are not active.

The cost formula for MMPC is calculated as (30) with (35) and initial condition of $\xi_k = \begin{bmatrix} x_k^T & 0 \end{bmatrix}^T$. Hence, only the upper-left $n \times n$ ($n$ is the dimension of $x_k$) sub-matrices are relevant in the cost computation. To be specific, the sub-matrices are $\hat{P}_{\sigma(k)}$, $\hat{P}_{x,\sigma(k)}$ and $\hat{P}_{u,\sigma(k)}$ as shown below

$$P_{\xi+u,\sigma(k)} = \begin{bmatrix} \hat{P}_{\sigma(k)} & \star \\ \star & \star \end{bmatrix}, \quad P_{\xi,\sigma(k)} = \begin{bmatrix} \hat{P}_{x,\sigma(k)} & \star \\ \star & \star \end{bmatrix},$$

$$P_{u,\sigma(k)} = \begin{bmatrix} \hat{P}_{u,\sigma(k)} & \star \\ \star & \star \end{bmatrix}$$

where $P_{\xi+u,\sigma(k)}$, $P_{\xi,\sigma(k)}$ and $P_{u,\sigma(k)}$ are defined in (34), (32) and (33), respectively, and $\star$ denotes a sub-matrix of compatible dimensions, which can be omitted from the cost computation.

Therefore, the quadratic cost of MMPC can be computed as

$$J_{\sigma(k)} = J_{x,\sigma(k)} + J_{u,\sigma(k)} = x_k^T \hat{P}_{\sigma(k)} x_k$$

where $J_{x,\sigma(k)} = x_k^T \hat{P}_{x,\sigma(k)} x_k$ and $J_{u,\sigma(k)} = x_k^T \hat{P}_{u,\sigma(k)} x_k$.

Now we have a way to compare different MMPC schemes, including differences in horizon lengths and update sequences. In other words, given two MMPC schemes, whose costs are $J_i = x_k^T \hat{P}_i x_k$ and $J_j = x_k^T \hat{P}_j x_k$, then

$$J_i - J_j = x_k^T (\hat{P}_i - \hat{P}_j) x_k$$

Hence analysis of the properties of the difference $\hat{P}_i - \hat{P}_j$ gives information on the relative merits of the two MMPC designs. For example, $\hat{P}_i - \hat{P}_j > 0$ indicates that design $j$ is better than design $i$ for all initial conditions $x_0$.

Consider the following two-input-two-output continuous-time plant

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \frac{1}{7s+1} & \frac{1}{3s+1} \\ \frac{2}{8s+1} & \frac{1}{4s+1} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}$$

We chose the sampling time to be $T = 1s$. For MMPC, the states were measured at $T/m = 0.5s$ with $u_1$ and $u_2$ alternatively applied at $0.5s$ intervals, but each held constant over a period of $T = 1s$, that is, $u_1$ is updated at times $(0, 1s, 2s, \ldots)$ and $u_2$ is updated at times $(0.5s, 1.5s, \ldots)$ (for the updating sequence of $u_1,\ u_2,\ u_1,\ u_2, \cdots,$). For all the results listed below, the tuning parameters for MMPC are: $q = I$ and $r = 1$.

The horizon length $N_u$ is an important tuning parameter for MPC in general, and remains so for MMPC. With the MMPC cost formula, we can compute the cost and predict the performance difference with different $N_u$. To illustrate, we generate a simulation scenario by adding a step input disturbance to the plant and see how the performance and cost vary with different $N_u$. This is done by replacing $\Delta u$ in (1) by $(\Delta u + \Delta d)$, namely modelling a *change* of input disturbance, so that an impulse on $\Delta d$ corresponds to a step disturbance. This then allows such a disturbance to be represented by the initial condition $x(0) = B\Delta d$, which in turn allows the use of formula (31). (The results were checked against numerical estimation of the cost accumulated during simulation.)

MMPC is a periodic control scheme; thus its performance depends on the time at which a disturbance occurs. More specifically, for the two-input plant considered here, depending on the time at which disturbances occur, $u_{1,k}$ may react first (ie control updating sequence $(u_{1,k}, u_{2,k}, u_{1,k+1}\ldots))$, or $u_{2,k}$ may react first (ie control updating sequence $(u_{2,k}, u_{1,k}, u_{2,k+1}, \ldots))$. This depends on whether $\sigma(k) = 1$ or $\sigma(k) = 2$. This section uses the cost formula (31) to compare the cost of MMPC for these two different updating sequences in a specific scenario.

Table 1 shows the eigenvalues of $\hat{P}_1 - \hat{P}_2$ as $N_u$ varies, where $\hat{P}_1$ represents the cost matrix when $\sigma(k) = 1$ (updating sequence $(u_{1,k}, u_{2,k}, \ldots)$) while $\hat{P}_2$ represents the cost matrix when $\sigma(k) = 2$ (updating sequence $(u_{2,k}, u_{1,k}, \ldots)$). From the table, it can be seen that $\hat{P}_1 - \hat{P}_2$ is indefinite, which means that one updating sequence is not definitely better than the other, but depends on the specific scenario — as expected. Fig. 2 compares the closed-loop performance between the two updating sequences when $N_u = 5$. The solid lines show the response to a step disturbance on each input when $u_1$ is the first input to react to it (disturbance occurs at step $k$ and $\sigma(k) = 1$), while the dashed lines show the response when $u_2$ is the first input to react to the disturbance. The input trajectories approximately interchange in the two cases, as do the output trajectories, so there is little to choose between the two as regards performance. The cost difference $x_k^T(\hat{P}_1 - \hat{P}_2)x_k$ in this case is 0.3599, which means that the second updating sequence is slightly better than the first for this particular disturbance, as judged by the cost function.

Table 1: MMPC: Eigenvalues of $(\hat{P}_1 - \hat{P}_2)$ with different $N_u$

| $N_u$ | Eigenvalues of $(\hat{P}_1 - \hat{P}_2)$ | | | | | |
|---|---|---|---|---|---|---|
| 1 | -9.4274 | -0.0069 | 0.0000 | 0.0000 | 0.0042 | 4.9050 |
| 2 | -11.3972 | -0.0008 | -0.0002 | 0.0001 | 0.0014 | 6.2401 |
| 3 | -15.9939 | -0.0190 | -0.0001 | 0.0001 | 0.0289 | 9.7714 |
| 4 | -21.9017 | -0.0438 | -0.0002 | 0.0001 | 0.0603 | 14.0665 |
| 5 | -28.3386 | -0.0710 | -0.0001 | 0.0000 | 0.0905 | 18.2002 |

### 6.2 Robust MMPC: Spring-Mass Example

This section considers the control of the simple mechanical system shown in Fig. 3. The system comprises four point masses moving in one dimension. Each has mass of five units and is connected to the adjacent masses by a spring of stiffness one unit.

Each controller minimizes control energy subject to a constraint on the the position of mass 1, shown as output $y$ in Fig. 3. Control energy is taken as $\int u(t)^T u(t)dt$

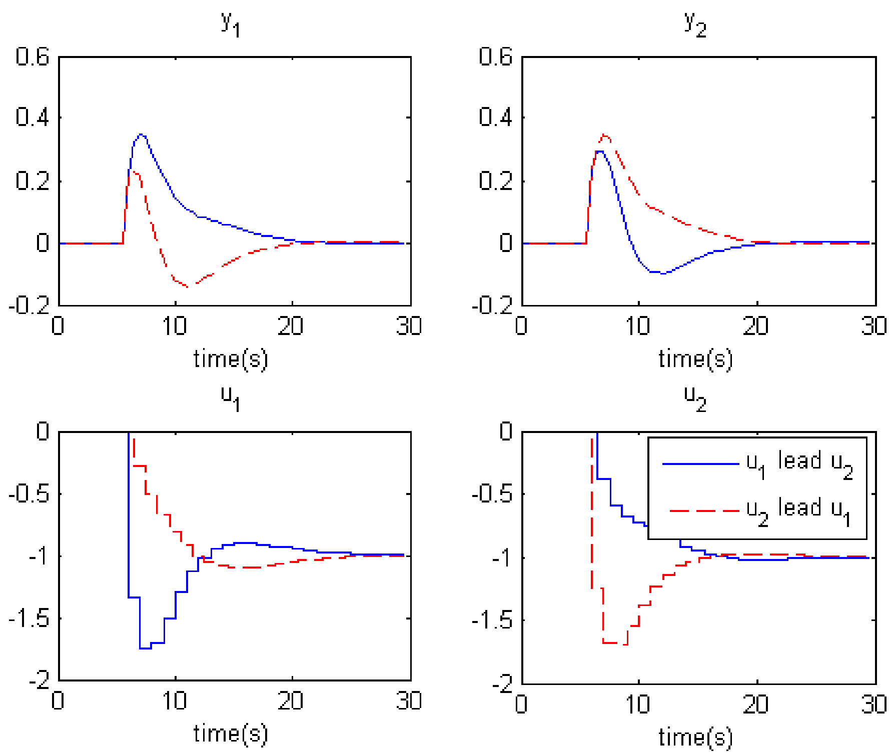



Figure 2: Effects of updating sequence, step input disturbance, $N_u = 5$, solid: $(u_{1,k}, u_{2,k}, \ldots)$, dashed:$(u_{2,k}, u_{1,k}, \ldots)$.

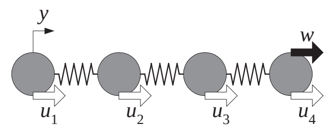



Figure 3: Spring-Mass Example System

over a 400s simulation. The inputs are the control moves $\Delta u_k$ applied to forces $u_i$ acting on each mass, and therefore the control force levels $u(t)$ are elements in an augmented state vector. All controllers were made robust to a disturbance force of up to 0.01 unit acting on mass 4. In the simulations, a disturbance pulse was applied to that mass of magnitude 0.01 from 50s to 200s.

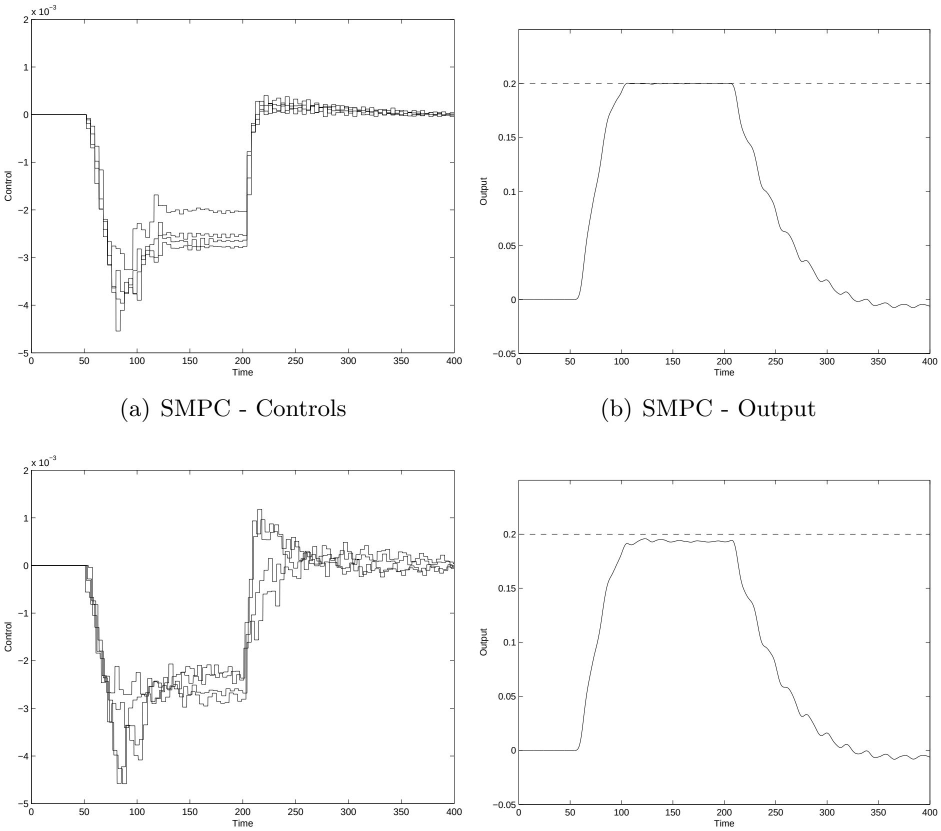
In the MMPC simulations, control moves were applied at intervals of one second, *i.e.* channel 1 moved at $t = t_1$ seconds, then channel 2 at $t = t_1 + 1$ seconds, and so on. In the comparison SMPC simulations, moves were made on all channels every four seconds, but to ensure fair comparison, the constraints were enforced at intervals of one second as in MMPC. Computation time is taken as the time spent in the "quadprog" function, totalled over all calls during the simulation.

Figure 4 shows the control input signals and the output signals for each of the two controllers considered, using a horizon of 120s in both cases. The asynchronous control moves can be seen in the control signal plots from the MMPC simulation. In both cases, the output signal runs tightly against the constraint (shown dashed) for the duration of the disturbance pulse. This is as expected, since the objective is to minimize control energy and therefore the controller makes use of all available flexibility in the output constraint. The output under MMPC is slightly further from the limit than under SMPC, possibly because that controller effectively solves a more constrained problem due to the reduced decision variable set. However, the effect is not significant.

To further illustrate the ability of the new robust MMPC to satisfy hard constraints despite disturbances, the simulation using MMPC was repeated using different constraint levels. The resulting output signals are shown in Figure 5. In every case, the signal goes right to its limit, but never beyond, and the optimisations remain feasible. These results illustrate that the constraints are active in these simulations and that the robust MMPC method does not introduce undue conservatism.

Table 2 compares detailed statistics from the results in Fig. 4. Observe that the performance, in terms of the control energy, is roughly the same for both controllers. However, MMPC is slightly faster than SMPC, since its sub-problems have only a quarter as many decision variables as SMPC. This illustrates the underlying premise of MMPC: it is faster to solve a sequence of four problems of 31 variables than one problem of 124.

To further explore the issue of scalability, the simulations from Fig. 4 using SMPC and MMPC were repeated with various horizon lengths. Figure 6 shows the variation of total computation time with horizon length for both controllers. With a very short horizon, SMPC is faster than MMPC. We hypothesize that this is due to overheads in the QP solver, such as set-up time, which dominate the solution time for small




(a) SMPC - Controls

(b) SMPC - Output

(c) MMPC - Controls

(d) MMPC - Output

Figure 4: Spring-Mass Example: Responses to Disturbance Pulse

problems and therefore penalise the more frequent optimisation calls of MMPC. However, as the horizon length increases, the computation time becomes dominated by the actual solution process and MMPC scales more favorably than SMPC.

### 6.3 Robust MMPC: Flight Dynamics Example

This section considers longitudinal control of an A-7A Corsair II aircraft. The dynamics model was taken from Example 6.1 in Ref. [9] and augmented to include a thrust input as well as the elevator input. Both inputs are constrained to $[-0.04, 0.04]$ and the constraints are made robust to input disturbances in the range $[-0.01, 0.01]$ on each channel. The simulation runs for 200s and a disturbance of 0.01 is applied to both channels from 20s to 120s. The planning horizon is 80s in all cases and the objective is to minimize $x_2^2$ where the state element $x_2$ corresponds to the velocity normal to the aircraft axis in the body frame.

Figure 7 shows the control and output signals from simulations using the two different controllers. SMPC executes moves on both channels at intervals of one second. MMPC performs a single move on alternating channels every half a second. Thus the total number of moves on each channel in each simulation is the same. Table 3 compares the results using the same metrics as in the previous section, except for

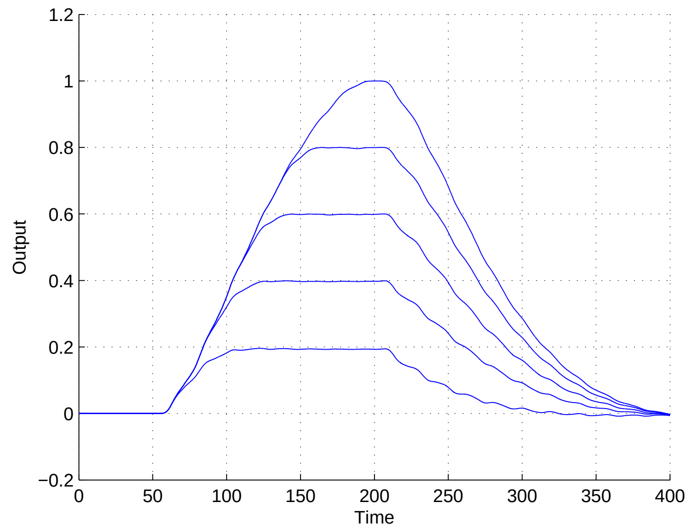



Figure 5: Spring-Mass Example: Outputs from MMPC for Constraint Settings 0.2, 0.4, 0.6, 0.8, and 1.0

Table 2: Spring-Mass Example: Results for Each Controller Rejecting Disturbance Pulse.

| Controller | SMPC | MMPC |
|---|---|---|
| $\int \mathbf{u}(t)^T \mathbf{u}(t) dt \times 1000$ | 4.312 | 4.320 |
| Computation Time (s) | 6.6 | 5.6 |
| N$^{\text{o.}}$ of QP Solutions | 100 | 400 |
| N$^{\text{o.}}$ of Decision Vars. per QP | 124 | 31 |

the performance which is here taken as the peak value of the normal velocity $\|x_2\|_\infty$. Unlike in the spring-mass example, there is significant variation in performance between the two controllers. The MMPC controller, with its faster response time, is able to mitigate the short period response more effectively than SMPC, which leaves a significant spike at the onset of the disturbance, indicating that in this case, it is better to respond to a disturbance quickly with one channel than slowly with both. MMPC also requires significantly less computational effort than SMPC for this example. Note that the computation times are approximately in accordance with the expected $O(\nu^3)$ behaviour, where $\nu$ is the number of decision variables: in this example SMPC has 80 decision variables, and 200 QP problems are solved during the simulation, whereas MMPC has 41 variables, and 400 QP problems are solved. $(200 \times 80^3) : (400 \times 41^3) = 3.7$, which is quite close to the ratio of computation times $42.25 : 9.15 = 4.6$.

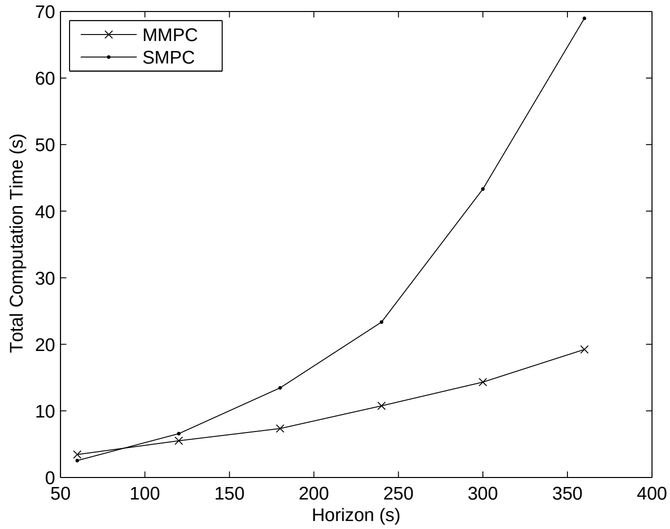



Figure 6: Spring-Mass Example: Variation of Computation Time with Horizon Length for SMPC and MMPC

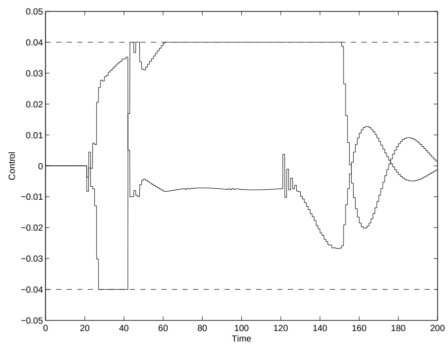
Table 3: Aircraft Example: Results for Each Controller Rejecting Disturbance Pulse.

| Controller | SMPC | MMPC |
|---|---|---|
| $\|x_2(t)\|_\infty$ | 8.53 | 6.49 |
| Computation Time | 42.25 | 9.15 |
| N$^{\text{o.}}$ of QP Solutions | 200 | 400 |
| N$^{\text{o.}}$ of Decision Vars. per QP | 80 | 41 |

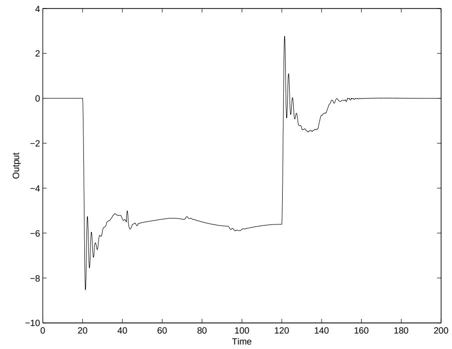
# 7 Conclusion

In this work a novel control scheme known as *Multiplexed* MPC was proposed, which is expected to be of practical benefit because it offers reduced computational complexity. Multiplexed model predictive control (MMPC) updates one input at a time, of a multi-input controlled plant. The motivation is to reduce the computational complexity of MPC, in order to allow reduced control update intervals. For some plants this leads to improved control, as a result of the controller being able to react to disturbances more quickly. MMPC scales well with increasing numbers of inputs, since the computational complexity depends only weakly on the number of inputs. The proposed MMPC scheme has been proved to be nominally stable. The nominal stability of a large class of other multiplexed MPC schemes follows by the same argument as we used in this paper.

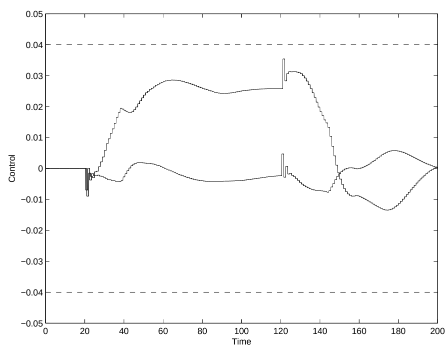
Some performance benefit over conventional MPC can be obtained as a result of faster reactions to disturbances, despite suboptimal solutions being obtained. This has been demonstrated by an example. However, the closed loop disturbance rejection performance under MMPC is time varying because of the periodic nature of the control scheme.



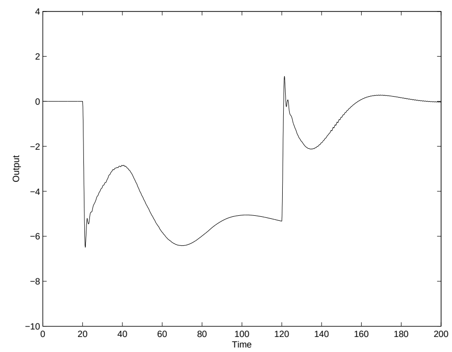

(a) SMPC - Controls




(b) SMPC - Output




(c) MMPC - Controls




(d) MMPC - Output

Figure 7: Aircraft Example: Responses to Disturbance Pulse

In this paper we have extended the basic MMPC idea to obtain robust feasibility and robust constraint satisfaction in the presence of unknown but bounded disturbances.

Simulation examples have demonstrated that our scheme succeeds in maintaining constraint satisfaction and feasibility despite the presence of disturbances. Furthermore, they have shown that performance improvements can indeed be obtained in some circumstances, compared with conventional MPC, they have indicated the kind of computational speed-up that can result from adoption of the MMPC scheme, and they have illustrated that these benefits are retained in circumstances where the constraints are active.

# References


[1] A. Bemporad and M. Morari. Robust model predictive control: A survey. *Robustness in Identification and Control*, 245:207–226, 1999.

[2] M.E-M. Ben Gaid, A. Çela, and Y. Hamam. Optimal integrated control and scheduling of networked control systems with communication constraints: application to a car suspension system. *IEEE Transactions on Control Systems Technology*, 14(4):776–787, July 2006.

[3] R.R. Bitmead, M. Gevers, and V. Wertz. *Applied Optimal Control: The Thinking Man's GPC.* Prentice-Hall, Englewood Cliffs, NJ, 1990.

[4] S. Bittanti, P. Colaneri, and G. De Nicolao. The difference periodic Riccati equation for the periodic prediction problem. *IEEE Transactions on Automatic Control,*, 33(8):706–712, August 1988.

[5] L.G. Bleris, P.D. Vouzis, M.G. Arnold, and M.V. Kothare. A co-processor FPGA platform for the implementation of real-time model predictive control. In *Proc. American Control Conference*, pages 1912–1917. Minneapolis, June 2006.

[6] F. Borrelli, T. Keviczky, and G. J. Balas. Collision-free UAV formation flight using decentralized optimization and invariant sets. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, December 2004.

[7] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, 2:44–52, 2002.

[8] L. Chisci, J. A. Rossiter, and G. Zappa. Systems with persistent disturbances: Predictive control with restrictive constraints. *Automatica*, 37(7):1019–1028, 2001.

[9] M. V. Cook. *Flight Dynamics Principles.* Butterworth Heinemann, 1997.

[10] W. B. Dunbar and R. M. Murray. Distributed receding horizon control for multi-vehicle formation stabilization. *Automatica*, 42(4):549–558, April 2006.

[11] J. R. Gossner, B. Kouvaritakis, and J. A. Rossiter. Stable generalized predictive control with constraints and bounded disturbances. *Automatica*, 33(4):551–568, 1997.

[12] D. Jia and B. Krogh. Min-max feedback model predictive control for distributed control with communication. In *Proceedings of American Control Conference*, pages 4507–45, 2002.

[13] T.A. Johanson, W. Jackson, R. Schreibar, and P. Tondel. Hardware architecture design for explict model predictive control. In *Proc. American Control Conference*, pages 1924–1929. Minneapolis, June 2006.

[14] G. Knagge, A. Wills, A. Mills, and B. Ninness. ASIC and FPGA implementation strategies for model predictive control. In *Proc. 10th European Control Conference*, Budapest, August 2009. European Union Control Association.

[15] K.V. Ling, W.K. Ho, B. Wu, A.G. Aribowo, Y. Feng, and H. Yan. Experimental evaluation of multiplexed MPC for semiconductor manufacturing. In *Proc. Asian Control Conference*, pages 1719–1722. Hong Kong, August 2009.

[16] K.V. Ling, J.M. Maciejowski, and B-F. Wu. Multiplexed model predictive control. In *Proc. 16th IFAC World Congress*, Prague, July 2005.

[17] K.V. Ling, S.P. Yue, and J.M. Maciejowski. An FPGA implementation of model predictive control. In *Proc. American Control Conference*, pages 1930–1935. Minneapolis, June 2006.

[18] J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2002.

[19] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: stability and optimality. *Automatica*, 36:789–814, 2000.

[20] M. Morari, M. Baotić, and F. Borrelli. Hybrid systems modeling and control. *European Journal of Control*, 9(2–3):177–189, 2003.

[21] R.M. Murray, J. Hauser, A. Jadbabie, M.B. Milam, N. Petit, W.B. Dunbar, and R. Franz. Online control customization via optimization-based control. In T. Samad and G. Balas, editors, *Software-Enabled Control*. IEEE Press and Wiley, 2003.

[22] T. Perez, G.C. Goodwin, and C.W. Tzeng. Model predictive rudder roll stabilization control for ships. In *Proc. 5th IFAC Conf. on Manoeuvring and Control of Marine Craft*. Aalborg, Denmark, January 2000.

[23] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764, 2003.

[24] A. Richards and J.P. How. Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility. In *Proc. American Control Conference*. Denver, June 2003.

[25] A. G. Richards. Robust model predictive control for time-varying systems. In *Proceedings of the joint 44th IEEE Conference on Decision and Control and European Control Conference*, 2005.

[26] A. G. Richards and J. P. How. A decentralized algorithm for robust constrained model predictive control. In *Proceedings of the American Control Conference*, 2004.

[27] A. G. Richards and J. P. How. Robust stable model predictive control with constraint tightening. In *Proceedings of American Control Conference*, 2006.

[28] A.G. Richards, K.V. Ling, and J.M. Maciejowski. Robust multiplexed model predictive control. In *Proc. European Control Conference*. Kos, Greece, July 2007.

[29] H. Richter, A. Singaraju, and J.S. Litt. Multiplexed predictive control of a large commercial turbofan engine. *AIAA Journal of Guidance, Control and Dynamics*, 31(2):273–281, March–April 2008. (DOI: 10.2514/1.30591).

[30] J.A. Rossiter, J. Sheng, T. Chen, and S.L. Shah. Interpretations of and options in dual-rate predictive control. *Journal of Process Control*, 15:135–148, 2005.

[31] T. Schouwenaars, J. P. How, and E. Feron. Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees. In *Proceedings of AIAA Guidance Navigation and Control Conference*, 2004.

[32] P. A. Trodden and A. G. Richards. Robust distributed model predictive control using tubes. In *Proceedings of the American Control Conference*, Minneapolis, June 2006.

[33] A.N. Venkat, J.B. Rawlings, and S.J. Wright. Plant-wide optimal control with decentralized MPC. In *Proc. 7th International Symposium on Dynamics and Control of Process Systems (DYCOPS)*. Cambridge, MA, July 2004.

[34] M.T. Virta, H. Wang, and J.C. Roberts. Developing a novel control method for cost effective papermaking at Mondi's Creams Mill. *Paper Technology*, 45(5):33–38, 2004.