# Min–Max MPC based on a computationally efficient upper bound of the worst case cost

**Article** *in* Journal of Process Control · June 2006

DOI: 10.1016/j.jprocont.2005.07.005

CITATIONS
34

READS
128

**4 authors**, including:

Daniel R. Ramirez
University of Seville
**57** PUBLICATIONS **853** CITATIONS
SEE PROFILE

Teodoro Alamo
Universidad de Sevilla
**240** PUBLICATIONS **4,675** CITATIONS
SEE PROFILE

Eduardo F. Camacho
Universidad de Sevilla
**389** PUBLICATIONS **12,603** CITATIONS
SEE PROFILE

# Min-Max MPC based on a computationally efficient upper bound of the worst case cost*

D.R. Ramírez,† T. Alamo, E.F. Camacho and D. Muñoz de la Peña
Departamento de Ingeniería de Sistemas y Automática, Universidad de Sevilla
Escuela Superior de Ingenieros, Camino de los Descubrimientos s/n, 41092 Sevilla
Phone: +34 954487347, Fax: +34 954487340
email: {danirr,alamo,eduardo,davidmps}@cartuja.us.es

June 28, 2005

**Abstract**

Min-Max MPC (MMMPC) controllers [1] suffer from a great computational burden which limits their applicability in the industry. Sometimes upper bounds of the worst possible case of a performance index have been used to reduce the computational burden. This paper proposes a computationally efficient MMMPC control strategy in which the worst case cost is approximated by an upper bound based on a diagonalization scheme. The upper bound can be computed with $O(n^3)$ operations and using only simple matrix operations. This implies that the algorithm can be coded easily even in non mathematical oriented programming languages such as those found in industrial embedded control hardware. A simulation example is given in the paper.

**Keywords:** Predictive control, Minimax techniques,Uncertain linear systems.

*This work is an expanded and revised version of a paper to be presented in the 16th IFAC World Congress, Praha 2005.

†Corresponding author

# 1 Introduction

In Min-Max MPC controllers [2, 1], the value of the control signal to be applied is found by minimizing the worst case of a performance index (usually quadratic) which is in turn computed by maximizing over the possible expected values of disturbances and uncertainty. Solving these problems can be very time consuming as they are of the NP-hard kind [3, 4, 5]. Thus, the implementation of this type of control is very difficult leading to a lack of experimental results. Only a few applications to plants with slow dynamics [6] or complex simulated models [7] have been reported. For moderate fast dynamics the min-max problem can be solved numerically only when the number of extreme realizations of the uncertainty is relatively low. This is the case when the prediction horizon is small or when a complexity reduction strategy like that of [8] is used. When fast dynamics are to be controlled the min-max problem cannot be solved numerically, and approximate solutions have to be used [9, 10]. However, these techniques impose great rigidity in the controller parameters, as well as a certain degree of approximation error.

Recently, the MMMPC control law has proven to be piecewise affine when a quadratic [11] or 1-norm based criterion [12, 13] is used as the cost function. With these results, together with those obtained when multiparametric mathematical programming is applied [12], explicit forms of the control law can be built. However, the number of regions in which the state space has to be partitioned grows with the prediction horizon in a combinatorial explosion. Thus, storage requirements and searching time for the appropiate region can be very high for practical values of the prediction and control horizons. A search tree strategy has been proposed to reduce the searching time in the MPC context [14, 15]. If the process model or the controller tuning parameters change, however, the computation of the regions has to be done again. This field continues evolving and new and more elaborate robust predictive controllers based on multiparametric programming have appeared in [16, 17].

Often the computational burden issue is solved by using a bound of the worst case cost instead of computing it explicitly [18]. The upper bound can be computed by using LMI techniques such as in [19, 20, 21, 22, 23]. The LMI problems have a computational burden that cannot be neglected in certain applications. Moreover, the interior point methods used to solve the LMI depend on the initial solution and the time needed to converge is not always the same.

We consider in this paper linear systems with bounded additive uncertainties and a quadratic criterion. In this case, computing the worst case cost implies the solution of a quadratic maximization problem with a computational complexity that is exponential with the prediction horizon. In this paper an efficient upper bound of a quadratic maximization problem over a hypercube is presented. It is based on simple matrix operations that can be performed very efficiently and it is easy to implement in dedicated embedded control hardware. Within the MMMPC context it can be used as a substitute for the worst case cost for systems in which there is no computational power available to solve the LMI problems on line or in which it is difficult to implement an LMI solver. Moreover, in different simulation examples the performance of this strategy is very close to the one obtained using the LMIs and not very different from the one corresponding to the exact worst case cost. On the other hand, as illustrated by the simulations given in the paper, the computational burden of the proposed bound is much lower than that of the LMIs and the one needed to compute the exact maximum.

The paper is organized as follows: section 2 presents the MMMPC controller. Section 3 presents the efficient upper bound on the quadratic maximization problem and section 4 presents a performance analysis of the proposed bound compared to the LMI bound. A simulation example is given in section 5. Finally, section 6 presents the conclusions.

## 2 Min-Max MPC with bounded additive uncertainties

Consider the following state space model with bounded additive uncertainties [2]:

$$\begin{aligned} x(t+1) &= Ax(t)+Bu(t)+D\theta(t) \\ y(t) &= Cx(t) \end{aligned} \tag{1}$$

with $x(t) \in \mathbb{R}^{dimx}$, $u(t) \in \mathbb{R}^{dimu}$, $\theta(t) \in \{\theta \in \mathbb{R}^{dim\theta} : \|\theta\|_\infty \leq \theta_m\}$, $y(t) \in \mathbb{R}^{dimy}$. Here it will be assumed (without loss of generality) that $\theta_m = 1$. If this is not the case, matrix $D$ can be scaled to $\theta_m D$. Consider a sequence $\mathbf{u} = \left[u(t)^T \cdots u(t+N_u-1)^T\right]^T$ of values of the control signal over a control horizon $N_u$ and $\boldsymbol{\theta} = \left[\theta(t)^T \cdots \theta(t+N-1)^T\right]^T$ a sequence of future values of $\theta(t)$ over a prediction horizon $N$. Furthermore, let $J(\boldsymbol{\theta}, \mathbf{u}, x)$ be a quadratic performance index of the form:

$$J(\boldsymbol{\theta}, \mathbf{u}, x) = \sum_{j=0}^{N} x(t+j|t)^T Q_j x(t+j|t) + \sum_{j=0}^{N_u-1} u(t+j|t)^T R_j u(t+j|t) \tag{2}$$

where $x(t+j|t)$ is the prediction of the state for $t+j$ made at $t$ when the future values of the uncertainty are supposed to be given by the sequence $\boldsymbol{\theta} \in \Theta = \{\boldsymbol{\theta} \in \mathbb{R}^{N \cdot dim\theta} : \|\boldsymbol{\theta}\|_\infty \leq 1\}$. On the other hand $Q_j \in \mathbb{R}^{dimx \times dimx}$, $R_j \in \mathbb{R}^{dimu \times dimu}$ are symmetric positive definite matrices used as weighting parameters.

Min-Max MPC [1] is based on finding the control sequence $\mathbf{u}$ that minimizes $J(\boldsymbol{\theta}, \mathbf{u}, x)$ for the worst possible case of the predicted future evolution of the process state or output signal. This is accomplished through the solution of a min-max problem like:

$$\begin{aligned} \mathbf{u}^*(x) = \arg \quad & \min_{\mathbf{u} \in U} \quad J^*(\mathbf{u}, x) \\ & \text{s.t.} \quad L\mathbf{u} \leq c + Fx \end{aligned} \tag{3}$$

with

$$J^*(\mathbf{u}, x) = \max_{\boldsymbol{\theta} \in \Theta} \; J(\boldsymbol{\theta}, \mathbf{u}, x) \tag{4}$$

with $U \subseteq \mathbb{R}^{N_u \cdot dimu}$, $L \in \mathbb{R}^{nc \times (N_u \cdot dimu)}$, $F \in \mathbb{R}^{nc \times dimx}$ and $c \in \mathbb{R}^{nc}$, (where $nc$ is the number of constraints). The linear constraints in (3) impose the robust fullfilment of the constraints on $\mathbf{u}$

and $x$ [18]. As usual in all predictive control schemes, the solution of problem (3) is applied in a feedback manner using a receding horizon strategy. Note that the results presented in this paper are valid when using a semi-feedback approach [24, 25] in which the control input is given by $u(t) = -Kx(t) + v(t)$ where the feedback matrix $K$ is chosen to achieve some desired property such as nominal stability or LQR optimality without constraints. The MMMPC controller will compute the optimal sequence of correction control inputs $v(t)$. Rewritting the state equation of system (1) as

$$x(t+1) = A_{CL}x(t) + Bv(t) + D\theta(t) \tag{5}$$

it is clear that such semi-feedback MMMPC can be casted into problem (3) with $A_{CL} = (A - BK)$.

Note that the model is linear on $x$, $\mathbf{u}$, $\boldsymbol{\theta}$ and the cost is a quadratic criterion. Therefore the cost function can be rewritten as [2, 8]:

$$J(\boldsymbol{\theta}, \mathbf{u}, x) = \mathbf{u}^T M_{uu}\mathbf{u} + \boldsymbol{\theta}^T M_{\theta\theta}\boldsymbol{\theta} + 2\boldsymbol{\theta}^T M_{\theta u}\mathbf{u} + 2x^T M_{uf}^T\mathbf{u} + 2x^T M_{\theta f}^T\boldsymbol{\theta} + x^T M_{ff}x \tag{6}$$

where $M_{\theta\theta}$ is positive semidefinite. Therefore $J(\boldsymbol{\theta}, \mathbf{u}, x)$ is convex on $\boldsymbol{\theta}$. As $J^*(\mathbf{u}, x)$ corresponds to the maximization of a convex function on the hypercube $\Theta$, the maximum is attained at least at one of the vertices of $\Theta$ [26]. Thus, $J^*(\mathbf{u}, x)$ can be computed as

$$J^*(\mathbf{u}, x) = \max_{\boldsymbol{\theta} \in \text{vert}\{\Theta\}} J(\boldsymbol{\theta}, \mathbf{u}, x) \tag{7}$$

In order to find the value of $J^*(\mathbf{u}, x)$ it is necessary to evaluate the function for all the vertices of $\Theta$. Taking into account that the number of vertices is $2^{N \cdot dim\theta}$ it is clear that the problem can only be solved in real time for small values of the prediction horizon and small dimensions of $\theta$ (because this is a well known NP-hard problem).

A different strategy aimed to reduce the computational burden is proposed here. Instead of minimizing $J^*(\mathbf{u}, x)$ in (3) an upper bound of $J^*(\mathbf{u}, x)$ is minimized. As it is shown in the following sections this upper bound is computationally efficient as it is computed in polynomial time

(the computational cost is $O(n^3)$ instead of $O(2^n)$). The tradeoff is a slightly more conservative control law, but, as illustrated in section 5, the performance will not be very different to that obtained when computing $J^*(\mathbf{u},x)$ exactly.

# 3 Upper bound for the quadratic maximization problem

In this section a procedure to efficiently compute an upper bound of the worst case cost is given. It can be seen from (6) that

$$J^*(\mathbf{u},x) = \max_{\boldsymbol{\theta}\in \text{vert}\{\Theta\}} \boldsymbol{\theta}^T S \boldsymbol{\theta} + 2\boldsymbol{\theta}^T p(\mathbf{u},x) + r(\mathbf{u},x) \tag{8}$$

where

$$\begin{aligned} S &= M_{\theta\theta} \\ p(\mathbf{u},x) &= M_{\theta u}\mathbf{u} + M_{\theta f}x \qquad (9) \\ r(\mathbf{u},x) &= \mathbf{u}^T M_{uu}\mathbf{u} + 2x^T M_{uf}^T \mathbf{u} + x^T M_{ff} x \qquad (10) \end{aligned}$$

Therefore, the problem of computing $J^*(\mathbf{u},x)$ belongs to the following class of problems:

$$\gamma^* = \max_{\theta\in \text{vert}\{\Theta\}} \theta^T S\theta + 2\theta^T p + r \tag{11}$$

The following proposition introduces an equivalent quadratic maximization problem.

**Proposition 1** *Problem (11) is equivalent to the following augmented problem:*

$$\max_{\begin{bmatrix} \theta_e \\ \theta \end{bmatrix} \in vert\{\Theta_A\}} \begin{bmatrix} \theta_e \\ \theta \end{bmatrix}^T \begin{bmatrix} r & p^T \\ p & S \end{bmatrix} \begin{bmatrix} \theta_e \\ \theta \end{bmatrix} \tag{12}$$

*where $\theta_e \in \mathbb{R}$ and $\Theta_A$ is the augmented unitary hypercube, i.e.:*

$$\Theta_A = \left\{ \begin{bmatrix} \theta_e \\ \theta \end{bmatrix},\ \theta_e \in \mathbb{R},\ |\theta_e| \le 1,\ \theta \in \Theta \right\}.$$

**Proof:**

Problem (11) can be rewritten as :

$$\begin{aligned} \gamma^* &= \max_{\theta \in \text{vert}\{\Theta\}} \begin{bmatrix} 1 \\ \theta \end{bmatrix}^T \begin{bmatrix} r & p^T \\ p & S \end{bmatrix} \begin{bmatrix} 1 \\ \theta \end{bmatrix} \\ &= \max_{\theta \in \text{vert}\{\Theta\}} \begin{bmatrix} -1 \\ -\theta \end{bmatrix}^T \begin{bmatrix} r & p^T \\ p & S \end{bmatrix} \begin{bmatrix} -1 \\ -\theta \end{bmatrix} \end{aligned} \tag{13}$$

Note that if every vertex in $\text{vert}\{\Theta\}$ is multiplied by $-1$ then the resulting set of vertices does not change (that is, $\text{vert}\{\Theta\} = \{-\theta : \theta \in \text{vert}\{\Theta\}\}$). Thus,

$$\gamma^* = \max_{\theta \in \text{vert}\{\Theta\}} \begin{bmatrix} -1 \\ -\theta \end{bmatrix}^T \begin{bmatrix} r & p^T \\ p & S \end{bmatrix} \begin{bmatrix} -1 \\ -\theta \end{bmatrix} = \max_{\theta \in \text{vert}\{\Theta\}} \begin{bmatrix} -1 \\ \theta \end{bmatrix}^T \begin{bmatrix} r & p^T \\ p & S \end{bmatrix} \begin{bmatrix} -1 \\ \theta \end{bmatrix} \tag{14}$$

The augmented problem (12) is equivalent to

$$\max \left\{ \max_{\theta \in \text{vert}\{\Theta\}} \begin{bmatrix} 1 \\ \theta \end{bmatrix}^T \begin{bmatrix} r & p^T \\ p & S \end{bmatrix} \begin{bmatrix} 1 \\ \theta \end{bmatrix}, \max_{\theta \in \text{vert}\{\Theta\}} \begin{bmatrix} -1 \\ \theta \end{bmatrix}^T \begin{bmatrix} r & p^T \\ p & S \end{bmatrix} \begin{bmatrix} -1 \\ \theta \end{bmatrix} \right\}$$

which taking into account equations (13) and (14) is equal to

$$\max \{\gamma^*, \gamma^*\} = \gamma^*$$

therefore

$$\max_{\begin{bmatrix} \theta_e \\ \theta \end{bmatrix} \in \text{vert}\{\Theta_A\}} \begin{bmatrix} \theta_e \\ \theta \end{bmatrix}^T \begin{bmatrix} r & p^T \\ p & S \end{bmatrix} \begin{bmatrix} \theta_e \\ \theta \end{bmatrix} = \gamma^*$$

Thus, the augmented problem (12) provides the same maximum as problem (11) and this completes the proof. □

The augmented problem can be rewritten as:

$$\gamma^* = \max_{z \in \text{vert}\{\Theta_A\}} z^T H z \tag{15}$$

where $H \in \mathbb{R}^{n \times n}$. Now suppose that $T$ is a diagonal matrix such that $T \geq H$, then:

$$z^T H z \leq z^T T z = \sum_{i=1}^{n} T_{ii} z_i^2 \leq trace(T) \|z\|_\infty^2 \leq trace(T)$$

thus:

$$\gamma^* \leq \text{trace}(T)$$

Therefore a conservative upper bound of $\gamma^*$ can be found solving the following LMI problem:

$$\begin{aligned} \sigma^* = \quad & \min \quad \text{trace}(T) \\ & s.t. \\ & \qquad T \geq H \\ & \qquad T \text{ diagonal} \end{aligned} \tag{16}$$

If $H \geq 0$ this upper bound of $\gamma^*$ satisfies [27]:

$$\gamma^* \leq \sigma^* \leq \frac{\pi}{2}\gamma^*$$
$$\frac{2\sigma^*}{\pi} \leq \gamma^* \leq \sigma^*$$

This means that $\sigma^*$ provides both an upper and lower bound. Moreover, the conservativeness of the bound does not depend on the dimension of $H$. Thus, $\sigma^*$ is an appropiate bound to be used within a worst case MPC strategy as a conservative substitute of the worst case cost, i.e.:

$$\begin{aligned} J^*(\mathbf{u},x) \leq \quad & \min \quad \text{trace}(T) \\ & s.t. \\ & \qquad T \geq \begin{bmatrix} r(\mathbf{u},x) & p^T(\mathbf{u},x) \\ p(\mathbf{u},x) & S \end{bmatrix} \\ & \qquad T \text{ diagonal} \end{aligned} \tag{17}$$

However, solving the LMI problem (17) is computationally demanding enough to pose a problem when the sampling time is small and the dimension of $H$ is moderately high. Furthermore, its implementation can be challenging in industrial hardware where complex numerical libraries are seldom found. We propose another method to find a conservative upper bound of $J^*(\mathbf{u},x)$.

The goal here is to find the smallest (i.e. minimum trace) diagonal matrix $T$ such that $T \geq H$. The strategy is to obtain a diagonal matrix adding to $H$ $n-1$ semidefinite positive matrices of the form $v_i v_i^T$:

$$H + v_1 v_1^T + v_2 v_2^T + v_3 v_3^T + \cdots + v_{n-1} v_{n-1}^T = T$$

where $T$ is a diagonal matrix. Thus the problem is to find $v_i$, $i = 1, \cdots, n-1$ such that $T$ is diagonal and the conservativeness of the bound is kept as low as possible. Suppose that

$H = \begin{bmatrix} a & b^T \\ b & H_r \end{bmatrix}$, $a \in \mathbb{R}$ and that we want to add $v_1 v_1^T$ in such a way that:

$$\begin{bmatrix} a & b^T \\ b & H_r \end{bmatrix} + v_1 v_1^T = \begin{bmatrix} d & 0 \\ 0 & \hat{H}_r \end{bmatrix}, \quad d \in \mathbb{R} \tag{18}$$

Once $v_1$ is found, the process continues by choosing $v_2$ such that $\hat{H}_r$ is also partially diagonalized and so on. If $v_1$ is chosen to be $\begin{bmatrix} \alpha & e^T \end{bmatrix}^T$ then $v_1 v_1^T$ becomes:

$$\begin{bmatrix} \alpha \\ e \end{bmatrix} \begin{bmatrix} \alpha & e^T \end{bmatrix} = \begin{bmatrix} \alpha^2 & \alpha e^T \\ \alpha e & e e^T \end{bmatrix}$$

with $\alpha > 0$. This implies that $\alpha e = -b$ thus $e = \frac{-b}{\alpha}$, $d = a + \alpha^2$ and $\hat{H}_r = H_r + \frac{b b^T}{\alpha^2}$.

The parameter $\alpha$ should be chosen to minimize the error introduced by the diagonalization in the original augmented maximization problem. This error is:

$$z^T v_1 v_1^T z = z^T \begin{bmatrix} \alpha \\ -\frac{b}{\alpha} \end{bmatrix} \begin{bmatrix} \alpha & -\frac{b^T}{\alpha} \end{bmatrix} z$$

The error is maximum when $z$ turns out to be:

$$z^* = \operatorname{sign} \begin{bmatrix} \alpha \\ -\frac{b}{\alpha} \end{bmatrix}$$

(and also when it is of opposite sign). Taking into account that

$$\begin{bmatrix} \alpha & -\frac{b^T}{\alpha} \end{bmatrix} z^* = \left\| \begin{matrix} \alpha \\ -\frac{b}{\alpha} \end{matrix} \right\|_1$$

(where $\|x\|_1$ is the 1-norm equal to the sum of the absolute values of the components of $x$), the maximum error is

$$\left\| \begin{matrix} \alpha \\ -\frac{b}{\alpha} \end{matrix} \right\|_1^2$$

The value of $\alpha$ that minimizes the maximum error can easily be computed by finding the value that makes the derivative of

$$\left\| \begin{matrix} \alpha \\ -\frac{b}{\alpha} \end{matrix} \right\|_1 = \alpha + \frac{1}{\alpha} \|b\|_1$$

equal to zero. Such value is:

$$\alpha = \sqrt{\|b\|_1} \tag{19}$$

The procedure to compute the upper bound $\sigma_u$ is summarized in the following steps:

**Procedure 1** *Procedure to compute* $\sigma_u(H) \geq \max_{z \in vert\{\Theta_A\}} z^T H z$.

1. *Let* $T = H \in \mathbb{R}^{n \times n}$.

2. *for* $k = 1$ *to* $n-1$

3. *Let* $H_{sub} = [T_{ij}]$ *for* $i, j = k \cdots n$.

4. *Compute* $\alpha$ *for* $H_{sub} = \begin{bmatrix} a & b \\ b^T & H_r \end{bmatrix}$ *from (19).*

5. *Make* $v_k^T = \begin{bmatrix} \alpha & \frac{-b^T}{\alpha} \end{bmatrix}$.

6. *Make* $v_e^T = \begin{bmatrix} 0 & \cdots & 0 & v_k^T \end{bmatrix} \in \mathbb{R}^n$

7. *Update* $T$ *by making* $T = T + v_e v_e'$.

8. *endfor*

9. *Compute the upper bound from* $\sigma_u(H) = \sum_{i=1}^{n} T_{ii}$.

By construction $T \geq H$. Therefore, $\max_{z \in \text{vert}\{\Theta_A\}} z^T H z \leq \max_{z \in \text{vert}\{\Theta_A\}} z^T T z = \text{trace}(T) = \sigma_u(H)$. That is, $J^*(\mathbf{u}, x) = \max_{z \in \text{vert}\{\Theta_A\}} z^T \begin{bmatrix} r(\mathbf{u},x) & p^T(\mathbf{u},x) \\ p(\mathbf{u},x) & S \end{bmatrix} z \leq \sigma_u \left( \begin{bmatrix} r(\mathbf{u},x) & p^T(\mathbf{u},x) \\ p(\mathbf{u},x) & S \end{bmatrix} \right)$.

Note that only simple matrix operations are needed to compute the upper bound using the procedure given before. This implies that the algorithm can be coded easily even in non mathematical oriented programming languages such as those found in industrial embedded control hardware. This is relevant because a difficult implementation is a drawback when applying complex control strategies in the industry.

**Proposed strategy**

The proposed control strategy will be to apply the solution of

$$\begin{aligned} \mathbf{u}^*(x) = \arg \quad & \min_{\mathbf{u} \in U} \quad \sigma_u \left( \begin{bmatrix} r(\mathbf{u},x) & p^T(\mathbf{u},x) \\ p^T(\mathbf{u},x) & S \end{bmatrix} \right) \\ & \text{s.t.} \quad L\mathbf{u} \leq c + Fx \end{aligned} \tag{20}$$

in a receding horizon manner, where $p(\mathbf{u},x)$, $r(\mathbf{u},x)$ and $S$ are computed as in (9)-(10) and $\sigma_u$ is computed using procedure 1.

Note that procedure 1 runs for $n-1$ iterations each one with a computational complexity of $O(n^2)$, thus it is an $O(n^3)$ algorithm. The matrix dimension $n$ is in turn $(N \cdot dim\theta)+1$, therefore the number of necessary computations is roughly $c_1((N \cdot dim\theta)+1)^3$. Instead of minimizing a cost function that requires an exponential number of operations (roughly $c_2(2^{N \cdot dim\theta})$), here it is proposed to minimize an upper-bound of this cost function that can be evaluated in polynomial time. The overall computational burden will be much lower as illustrated in the example given in section 5.

**Remark 1** *Note that in procedure 1 the goal is to find a diagonal matrix $T$ such that $T \geq H \geq 0$. The maximization problem $\max_{z \in vert\{\Theta_A\}} z^T T z$ will then be solved by $\sigma_u = \sum_{i=1}^{n} T_{ii}$. Suppose that at a certain iteration of procedure 1 we have a partly diagonalizated matrix $T \geq H$ in which $H_{sub}$ of step 3 has all its elements nonnegative. Then $\max_{z \in vert\{\Theta_A\}} z^T T z = \|T\|_1 \geq \gamma^*$. Therefore, it is not necessary to continue the diagonalization of $T$ as $\max_{z \in vert\{\Theta_A\}} z^T T z$ is readily solved by $\sigma_u = \|T\|_1$.*

# 4 Performance analysis of the proposed bound

Here the accuracy of the proposed upper bound is discussed. The upper bound will be compared with the LMI bound and the 1-norm of the matrix $H$, which is itself a very rough upper bound:

$$\sigma_1 = \|H\|_1 = \sum_{i=1}^{N} \sum_{j=1}^{N} |H_{ij}|$$

The 1-norm will be equal to the maximum when all the elements of $z^*$ are nonnegative. In the following it will be shown that $\sigma_u \leq \sigma_1$.

Taking into account the block structure of $H$ given in (18) the 1-norm of $H$ can be computed as

$$\sigma_1 = \|H_r\|_1 + 2\|b\|_1 + |a|$$

On the other hand, the 1-norm after a diagonalization step can be computed as

$$\sigma_2 = \left\| \begin{array}{cc} a+\|b\|_1 & 0 \\ 0 & H_r + \frac{bb^T}{\|b\|_1} \end{array} \right\|_1 \leq |a| + \|b\|_1 + \|H_r\|_1 + \left\| \frac{bb^T}{\|b\|_1} \right\|_1 \tag{21}$$

Taking into account that $\left\| \frac{bb^T}{\|b\|_1} \right\|_1 = \|b\|_1$ it follows that

$$\sigma_2 \leq \sigma_1 \tag{22}$$

Thus, the diagonalization scheme proposed in section 3 provides a succession of improved upper bounds.

Now both the proposed bound and the 1-norm will be compared with the LMI bound. Consider figure 1 which shows the mean deviation of the proposed bound (solid plot) from the LMI bound (computed as $\left(\frac{\sigma_u}{\sigma^*} - 1\right) \times 100$) as a function of the dimension of $H$. For this comparison, a group of random positive definite matrices where the mean value of its non-diagonal elements is zero have been generated[1] (200 matrices for each dimension). It can be seen that the deviation from the LMI bound grows with matrix dimension, as it can be expected from the error introduced at each diagonalization step. Even though, it is noteworthy that for much of the range needed in control applications (up to $\dim\{H\} = 30$, which accounts for $1,073,741,824$ vertices in $\Theta$) the deviation from the LMI bound remains under $20\%$. Moreover, it can be seen that the 1-norm is always worse (more conservative) than the proposed bound.

Another interesting property is that, taking into account remark 1, the deviation of the proposed upper bound from the LMI bound depends on the mean value of the elements of $H_0$, with $H = H_0' H_0$. As illustrated by figure 2, the highest error is when the mean is around zero, quickly dropping to very small values when the mean goes to positive or negative values. In MMMPC control problems, as found by the authors through many simulations, the mean of the sum of the elements of matrix $H_0$ is generally neatly different from zero, although not all its entries share the same sign. This means that the bound accuracy will be close to that of the LMI bound.

[1] These matrices are generated subtracting two uniformly distributed random matrices created using the *Matlab* **rand** function. Then, every matrix is multiplied by itself transposed to make it positive semidefinite.

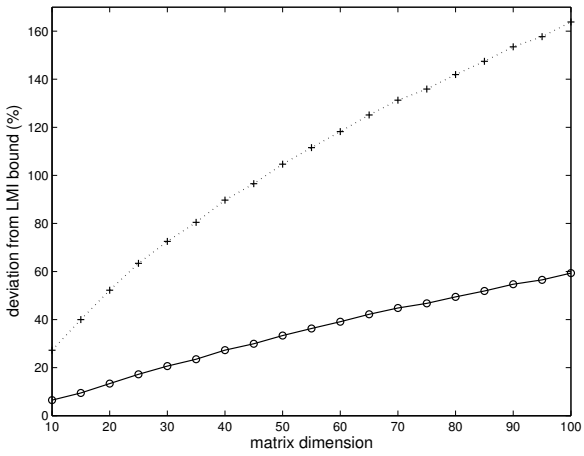


Figure 1: Mean deviation from the LMI bound plotted as a function of matrix dimension for a group of randomly generated matrices (see text for details). 1-norm dotted, proposed bound solid.

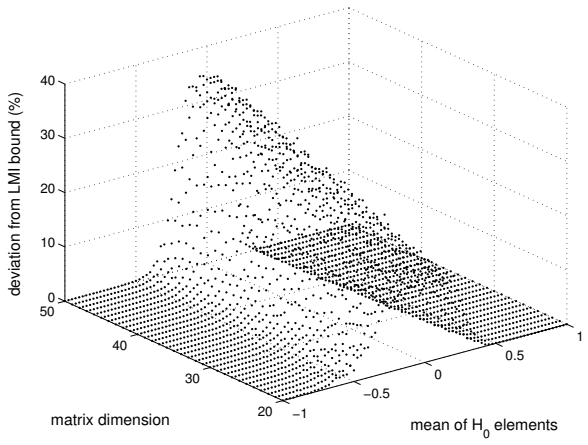


Figure 2: Mean deviation from the LMI bound plotted as a function of the mean of the elements of $H_0$ for a group of randomly generated matrices of dimension between 20 and 50.

## 4.1 Computational burden of the upper bound

The proposed bound has lower computational burden than the LMI bound. Figure 3 shows the speed-up factor (computed as $\frac{\text{flops}_{\text{LMI}}}{\text{flops}_{\text{proposed}}}$, where $\text{flops}_{\text{LMI}}$ and $\text{flops}_{\text{proposed}}$ are the flops needed to compute the LMI and proposed upper bounds respectively) for a group of random matrices with different mean and dimensions[2]. Remark 1 has been taking into account when computing the proposed bound. It can be seen that the proposed bound can be computed using many times less floating point operations[3] than the LMI bound. This can be exploited to apply worst case control for systems with fast dynamics or to use hardware with low computational power.

[2]To make this comparison as fair as possible, the LMI bound has been obtained using the solver by F. Rendl that can be downloaded from `http://www.math.uni-klu.ac.at/or/Software`. This solver, specific for problem (16), proved to be very much efficient than the standard solver provided with the LMI Toolbox of *Matlab*.

[3]The number of operations needed for computing each bound have been obtained using the *Matlab* **flops** function. The precision of the LMI solver was left to default.

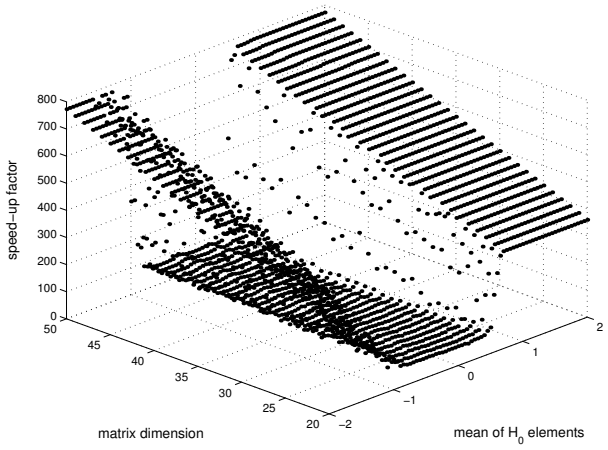


Figure 3: Speed-up factor (computed as $\frac{\text{flops}_{\text{LMI}}}{\text{flops}_{\text{proposed}}}$) between the proposed upper bound and the LMI bound for a group of randomly generated matrices with different mean and dimension.

It is noteworthy that as matrix dimension grows no clear trend is seen when the mean of $H_0$ is around zero. The computational cost for the proposed bound grows as $O(n^3)$ where $n$ is the matrix dimension. On the other hand, the interior point methods used for this class of LMI problems are also $O(n^3)$ algorithms, hence the relatively constant speed-up factor seen in figure 3 for zero mean matrices. Note however that the underlying constant in the number of operations needed by the LMI solver is bigger, leading to the speed-up factor shown. On the other hand, as illustrated in figure 3, the mean of the entries and the dimension of $H_0$, where $H = H_o' H_o$, affects the speed-up factor severely when this mean is different from zero. In this case, in the light of remark 1, the proposed upper bound is computed much faster than the LMI bound. In fact, when most of the entries of matrix $H_0$ (and therefore $H$) share the same sign, the speed-up factor between the proposed bound and the LMI bound grows progressively showing a linear trend. This can be interpreted as an evidence that for those matrices, the proposed bound tends to be $O(n^2)$ and not $O(n^3)$ which is the case when the mean is around zero.

# 5 Simulation example

Consider the two-tank network shown in figure 4. For this process, liquid streams flow into tanks 1 and 2 at respective volumetric rates $F_1$, and $F_2$; the outflow from each tank is assumed to be proportional to the respective liquid levels $h_1$ and $h_2$ in each tank. The liquid leaving tank 2 is split into two with a fraction F exiting, and the remainder R pumped back to the first tank. Thus, this is a two-input, two-output system, with the flow rates of the two inlet streams as the two inputs, and the liquid level in each tank as the two output variables.

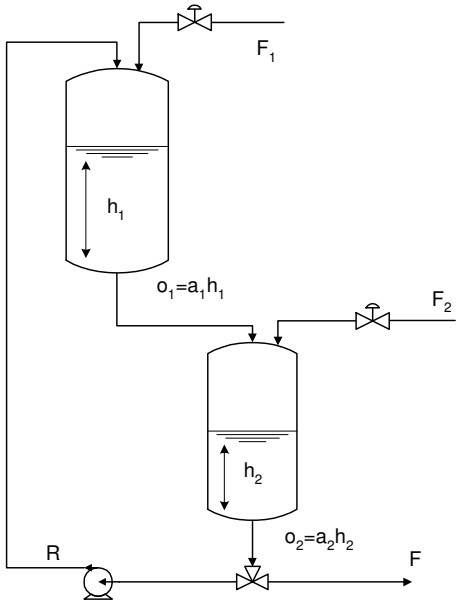


Figure 4: A two-tank network.

Let the section for tank 1 be 3 $m^2$ and that of the second tank 2 $m^2$. Moreover, assume that the constants of proportionality are identical and given as $a_1 = a_2 = 0.5$ $m^2 min^{-1}$ and that 40% of the amount of liquid leaving tank 2 is recycled back to tank 1. With these assumptions the following continuous time state-space model can be obtained (see chapter 20 of [28] for details):

$$\begin{aligned} \dot{x} &= \begin{bmatrix} -\frac{0.5}{3} & \frac{0.2}{3} \\ \frac{0.5}{2} & -\frac{0.5}{2} \end{bmatrix} x + \begin{bmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x \end{aligned} \tag{23}$$

A discrete time model has been obtained from (23) sampling at 0.2 minutes using a zero-order holder. Figure 5 shows the results of the proposed controller applied to the two-tank model. The set-point for the liquid level of each tank was 0.4 m and 0.5 m respectively. The prediction and control horizons were $N = 15$ and $N_u = 10$ respectively. Note that being this a two-input, two-output system the number of vertices to be considered is $2^{30}$ instead of $2^{15}$. Furthermore, the number of decision variables in the optimization problem is doubled. The weighting matrices were:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} 12 & 0 \\ 0 & 12 \end{bmatrix}$$

An uncertainty of $\pm 0.02$ meters is considered to affect both liquid levels. Note that a random noise of $\pm 0.01$ meters has been added to both levels. On the other hand, tank 1 suffers an unexpected loss of liquid at sampling time $t = 60$ that reduces the level 0.1 meters. Finally, in the simulation the following constraints were taken into account when computing the control signal:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \leq x(k) \leq \begin{bmatrix} 0.6 \\ 0.7 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \end{bmatrix} \leq u(k) \leq \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \quad \begin{bmatrix} -0.05 \\ -0.05 \end{bmatrix} \leq \Delta u(k) \leq \begin{bmatrix} 0.05 \\ 0.05 \end{bmatrix} \quad (24)$$

This example will be used to discuss how the increasing horizons affect the computational burden of the proposed strategy compared to that of the original min-max problem[4]. Figure 6 shows the average, maximum and minimum speed-up (computed as $\frac{\text{flops}_{\text{original}}}{\text{flops}_{\text{proposed}}}$) for different values of the prediction horizon. It can be seen that even when the number of vertices is small (i.e., $N$ small) the speed-up is rather high. Moreover, as the prediction horizon grows the speed-up increases exponentially. Thus, for a given hardware the user can pick the desired prediction horizon from a wider range of admissible values. Note how the speed-up is clearly variable for any given value of $N$. This is due to the different number of iterations of the numerical solver

[4]For both cases, the min-max problem was solved using the same numerical solver provided with *fmincon* function of *Matlab*.

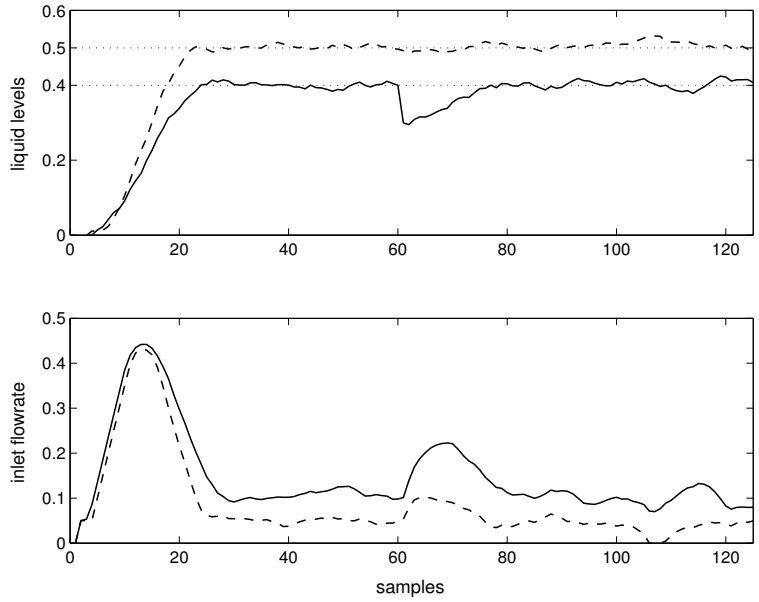


Figure 5: Liquid levels and inlet flows for the proposed MMMPC (tank 1 solid plot, tank 2 dashed plot).

depending on the value of $x$ and the constraints.

Furthermore, the computational burden of the minimization of the proposed upper bound is lower than that of the minimization of the LMI bound. This is illustrated in figure 7 in which the average, maximum and minimum speed-up over the LMI bound is plotted. Note that the average speed-up grows in a linear trend that leads to a much lower computational burden for typical values of the prediction horizon.

The accuracy of the proposed bound in the predictive control scheme proved to be very high. Table 1 shows the deviation from the optimal cost obtained using the exact maximum and the upper bound computed using the LMI solver. Different simulations using different values of $N$ have been made and the minimum, average and maximum deviations have been computed. Whereas the maximum deviation from the real optimal cost can be noticeable for lower values of $N$, it is quite remarkable that it tends to be smaller as the prediction horizon grows. Moreover, the average deviation is much lower and also decreases with higher horizons. On the other hand, the deviation from the optimal cost obtained with the LMI bound is always very small even at

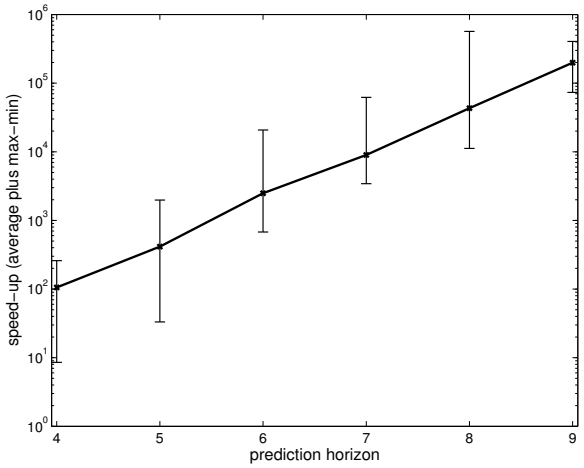


Figure 6: Maximum, average and minimum speed-up over the original min-max problem for different values of the prediction horizon. Note that a *logarithmic scale* is used in the vertical axis. Speed-up is computed at each sampling time as $\frac{\text{flops}_{\text{original}}}{\text{flops}_{\text{proposed}}}$.

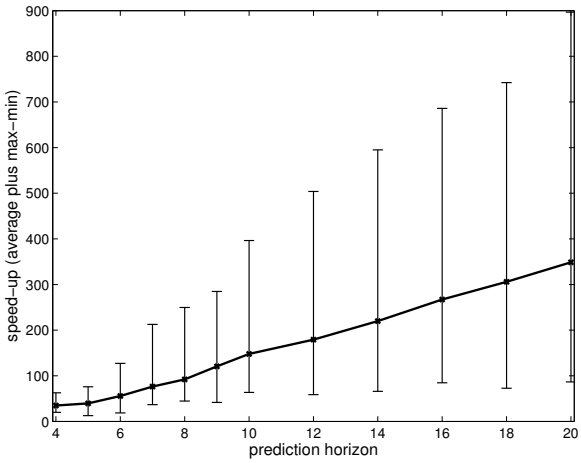


Figure 7: Maximum, average and minimum speed-up over the minimization of the LMI upper bound problem for different values of the prediction horizon. Speed-up is computed at each sampling time as $\frac{\text{flops}_{\text{LMI}}}{\text{flops}_{\text{proposed}}}$.

| N | Real (min) % | Real (avg) % | Real (max) % | LMI (min) % | LMI (avg) % | LMI (max) % |
|---|---|---|---|---|---|---|
| 4 | 2.88 | 19.3 | 44.2 | 0.00 | 0.44 | 4.74 |
| 5 | 1.69 | 17.8 | 43.9 | 0.00 | 1.22 | 4.76 |
| 6 | 1.07 | 14.7 | 42.7 | 0.00 | 2.18 | 4.77 |
| 7 | 1.19 | 11.1 | 36.97 | 0.00 | 2.5 | 5.47 |
| 8 | 1.03 | 12.2 | 27.1 | 0.00 | 2.76 | 7.21 |
| 9 | 0.27 | 5.59 | 25.5 | 0.00 | 2.21 | 7.44 |
| 10 | n/a | n/a | n/a | 0.00 | 2.17 | 7.7 |
| 15 | n/a | n/a | n/a | 0.00 | 2.43 | 9.71 |
| 20 | n/a | n/a | n/a | 0.00 | 1.88 | 10.3 |

Table 1: Deviation from the original MMMPC optimal cost and the LMI bound optimal cost (minimum, average and maximum) for different values of the prediction horizon (N) in the simulation example of section 5.

its maximum. Notice here that the deviation tends to grow with the prediction horizon. It is, however, very small even for values of $N = 20$ (an average deviation under 5%, and a maximum about 10%)[5]. Figures 8 and 9 illustrate the evolution of the deviation between the optimal costs using the proposed bound and both the exact cost and the LMI bound cost. It can be seen that the maximum deviation from the exact cost is when the state is very far from its desired value. However, the optimal costs are very close when the state is nearer its desired state. On the other hand, as expected from data in table 1, the optimal costs using the proposed bound and the LMI bound are nearly the same through the entire simulation. Thus, from this example it can be concluded that the proposed scheme is much more efficient, from a computational point of view, than the original MMMPC and the LMI bound. Furthermore its accuracy is comparable to that of the LMI bound. When the computational burden of the original min-max problem is taken into account it seems that the little average deviation from the exact maximum is a price very low for a much greater range of processes to which this type of control can be applied.

[5]Note that in this two input, two output system a prediction horizon $N = 20$ yields an augmented matrix of dimension 41 in the maximization problem.

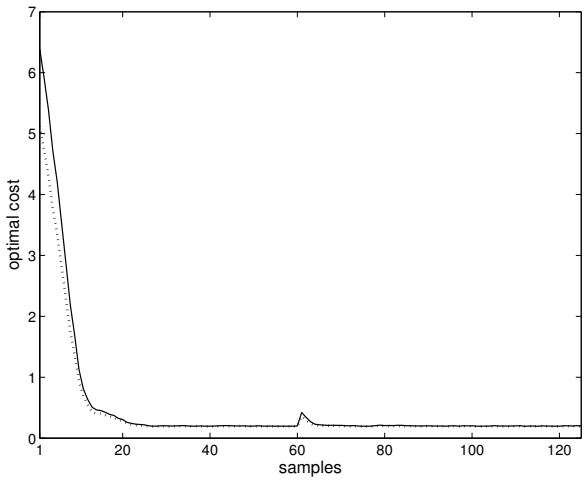


Figure 8: Optimal cost for the original MMMPC (dotted) and the proposed controller (solid) for a simulation with $N_u = 5, N = 9$.

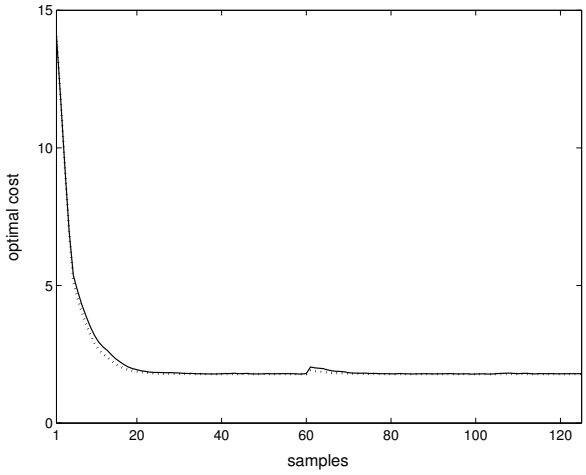


Figure 9: Optimal cost when the upper bound is computed using the LMI bound (dotted) and the proposed bound (solid) for a simulation with $N_u = 5, N = 20$.

# 6 Conclusions

An MMMPC based on an efficient upper bound of the worst case cost has been presented in this paper. It has a much lower computational burden than other approaches based on LMI techniques and can be implemented in dedicated industrial control hardware. The price to be paid is a moderate increment in the conservativeness of the bounds obtained. However, its little computational burden opens new fields of applications of MMMPC controllers.

# References

[1] P.J. Campo and M. Morari. Robust Model Predictive Control. In *Proc. American Control Conference*, pages 1021–1026, June 10-12 1987.

[2] E.F. Camacho and C. Bordóns. *Model Predictive Control*. Springer-Verlag, 1999.

[3] S.M. Veres and J.P. Norton. Predictive Self-Tuning Control by Parameter Bounding and Worst Case Design. *Automatica*, 29(4):911–928, 1993.

[4] J.H. Lee and Zhenghong Yu. Worst-case formulations of model predictive control for systems with bounded parameters. *Automatica*, 33(5):763–781, 1997.

[5] P.O.M. Scokaert and D.Q. Mayne. Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic Control*, 43(8):1136–1142, 1998.

[6] E. F. Camacho and Manuel Berenguel. Robust Adaptive Model Predictive Control of a Solar Plant with Bounded Uncertainties. *International Journal of Adaptive and Signal Processing*, 11:311–325, 1997.

[7] Y.H. Kim and W.H. Kwon. An application of min-max generalized predictive control to sintering processes. *Control Engineering Practice*, 6:999–1007, 1998.

[8] T. Alamo, D.R. Ramirez, and E.F. Camacho. Efficient implementation of constrained min-max model predictive control with bounded uncertainties: A vertex rejection approach. *Journal of Process Control*, 15:149–158, 2005.

[9] D.R. Ramírez, M.R. Arahal, and E.F. Camacho. Min-Max Predictive Control of a Heat Exchanger using a Neural Network Solver. *IEEE Trans. on Control Systems Technology*, 12(5):776–786, 2004.

[10] D.R. Ramírez, E.F. Camacho, and M.R. Arahal. Implementation of Min-Max MPC using hinging hyperplanes. application to a heat exchanger. *Control Engineering Practice*, 12(9):1197–1205, 2004.

[11] D.R. Ramírez and E.F. Camacho. On the piecewise linear nature of Min-Max Model Predictive Control with bounded uncertainties. In *Proc. 40th Conference on Decision and Control, CDC'2001*, December, 4-7 2001.

[12] A. Bemporad, F. Borrelli, and M. Morari. Min-max Control of Constrained Uncertain Discrete-Time Linear Systems. *IEEE Transactions on Automatic Control*, 48(9):1600–1606, 2003.

[13] E.C. Kerrigan and J.M. Maciejowski. Feedback min-max model predictive control using a single linear program: Robust stability and the explicit solution. *International Journal of Robust Nonlinear Control*, 14:295–413, 2004.

[14] T. Johansen and A. Grancharova. Approximate explicit constrained linear model predictive control via orthogonal search tree. *IEEE Transactions on Automatic Control*, 48(5):810–815, 2003.

[15] P. Tøndel, T. A. Johansen, and A. Bemporad. Evaluation of piecewise affine control via binary search tree. *Automatica*, 39(5):945–950, 2003.

[16] V. Sakizlis, N. M. P. Kakalis, J. D. Perkins V. Dua, and E. N. Pistikopoulos. Design of robust model-based controllers via parametric programming. *Automatica*, 40:189–201, February 2004.

[17] V. Sakizlis, J. D. Perkins V. Dua, and E. N. Pistikopoulos. Robust model-based tracking control using parametric programming. *Computers & Chemical Engineering*, 28:195–207, January 2004.

[18] T. Alamo, D. Muñoz de la Peña, D. Limón Marruedo, and E.F. Camacho. Constrained min max predictive control: Modifications of the objective function leading to polynomial complexity. *IEEE Transactions on Automatic Control*, 50:710–714, 2005.

[19] M.V. Kothare, V. Balakrishnan, and M. Morari. Robust constrained model predictive control using linear model inequalities. *Automatica*, 32(10):1361–1379, 1996.

[20] A. Casavola, M. Giannelli, and Edoardo Mosca. Min-Max Predictive Control strategies for input-saturated polytopic uncertain systems. *Automatica*, 36:125–133, 2000.

[21] Y. Lu and Y. Arkun. Quasi-Min-Max MPC algorithms for LPV systems. *Automatica*, 36(4):527–540, 2000.

[22] F. A. Cuzzola, J. C. Geromel, and M. Morari. An improved approach for constrained robust Model Predictive Control. *Automatica*, 38(7):1183–1190, 2002.

[23] Z. Wan and M.V. Kothare. An efficient off-line formulation of robust model predictive control using linear matrix inequalities. *Automatica*, 39:837–846, 2003.

[24] D.Q. Mayne. Control of Constrained Dynamic Systems. *European Journal of Control*, 7:87–99, 2001.

[25] J. Rossiter, B. Kouvaritakis, and M. Rice. A numerically robust state-space approach to stable-predictive control strategies. *Automatica*, 34:65–73, 1998.

[26] M.S. Bazaraa and C.M. Shetty. *Nonlinear Programming. Theory and Algorithms*. John Wiley & Sons, 1979.

[27] Y.E. Nesterov, H. Wolkowicz, and Y. Ye. Semidefinite programming relaxations of non-convex quadratic optimization. In H. Wolkowicz, R. Saigal, and L. Vanderberghe, editors, *Handbook of Semidefinite Programming: Theory, Algorithms and Applications*, page 34, Boston,MA, 2000. Kluwer Academics Publishers.

[28] B.O. Ogunnaike and W.H. Ray. *Process dynamics, modeling, and control*. Oxford University Press, 1994.