A MIXED-INTEGER MODEL PREDICTIVE CONTROL FORMULATION FOR LINEAR SYSTEMS

Lincoln F. L. Moro*[a], Ignacio E. Grossmann[b]
[a]Petrobras S.A. – Sao Paulo, Brazil
[b]Carnegie Mellon University - Pittsburgh, USA

Abstract

Most industrial model predictive controllers (MPC) use the traditional two-layer structure developed in the early 1980's, where the upper layer defines optimal steady-state targets for inputs and outputs, while the lower layer calculates the control moves that drive the system towards these steady-state targets. Typically both layers use continuous quadratic programming (QP) formulations to derive the optimal solutions. On the other hand, advances in mixed-integer programming (MIP) algorithms and their successful application to solve large scheduling problems in reasonable time show that MIP formulations have the potential of being applied advantageously to the multivariable model predictive control problem. In this paper, we present mixed-integer quadratic programming (MIQP) formulations for both layers and show that several difficulties faced in the MPC practical implementation can be overcome with this approach. In particular, it is possible to set explicit priorities for inputs and outputs, define minimum moves to overcome hysteresis, and deal with digital or integer inputs. The proposed formulation is applied to 2 benchmark problems and to a simulated industrial system and the results compared with those achieved by a traditional continuous MPC. The solutions of the MIQP problems are derived by a computer implementation of the Outer Approximation method (OA) also developed as part of this work.

Keywords

Mixed integer programming, predictive control, hybrid control.

1. Introduction

Most industrial model predictive controllers currently in use are based on the algorithms developed in the early 1980's (Qin and Badgwell, 2003). These algorithms have two main functions, i.e., to reduce the process variability through better dynamic control, and to move the operating point closer to the constraints, which in general results in significant economic benefits. In order to perform these functions, the usual practice is to adopt a hierarchical structure with two layers, where the upper layer deals with the steady-state problem of defining optimal targets for inputs and outputs, while the lower layer, responsible for the dynamic problem, calculates the control moves that drive the system towards these steady-state targets.

* To whom all correspondence should be addressed

The upper layer solves an optimization problem aiming at minimizing a linear combination of the projected steady–sate values of the inputs, and simultaneously minimizes the square of the moves to be imposed on these inputs. Linear relations among inputs and outputs, and constraints limiting the allowable range of both kinds of variables are also imposed. As a result of these constraints the problem may be infeasible, and this fact demands the implementation of a relaxation strategy in order to guarantee that some kind of solution is always found. The lower layer involves an optimization problem that is always feasible because it includes constraints only on the manipulated inputs.

We propose to replace both optimization problems by mixed-integer (MIP) formulations, thus building a hybrid MPC. Several advantages may result from such an approach; for instance, the possibility of assigning explicit priorities for the outputs, i.e., the definition of a preferential order of constraint relaxation in case the initial steady-state problem proves infeasible. The inputs can also receive explicit priorities to select the order in which they are to be moved to adjust each output. The formulation also makes it possible to set a minimum limit for the control moves, i.e., any movement must be greater than a limit that is defined large enough to overcome the hysteresis of valves significantly affected by this problem.

The MIP formulation also allows the controller to deal with discrete inputs, either manipulated variables or disturbances, i.e., variables that can assume only a set of discrete values like for instance, 0 or 1 (on or off).

Hybrid formulations for MPC have been developed and successfully used in industrial applications as described for instance by Bemporad and Morari (1999), Morari and Barić (2006), Zabiri and Samyudia (2006), and Oldenburg and Marquardt (2008). Nevertheless, most of these contributions address the control of hybrid systems, while the present work focus on the development of a mixed-integer algorithm based on the traditional MPC that can be advantageously applied to continuous systems, a capability that has been essentially ignored by the academic community.

One instance of such a possible advantage can be identified in systems where two or more inputs present similar influence on the outputs. Due to the intrinsic multivariable characteristic of the process and the controller, the inputs will be moved at the same time. But frequently a better approach would be to use one of them for smaller moves and the other for larger ones. This is the case when valves of different dimensions are set in parallel lines with precisely the intention of allowing better manipulation of the inputs. The larger valve should only be used for larger flowrate changes, since smaller ones may not be actually implemented due to valve hysteresis.

Another characteristic, also related to the multivariable nature of the controller, is the manipulation of independent variables that have only a small influence on an output, especially when this last variable hits a constraint. This is the case, for example, of the feed flowrate, an input that affects almost every output in the plant. The controller, as a rule, aims at maximizing the feed but this may be prevented by almost any output hitting a constraint. To cope with this situation, a frequent practice is the outright elimination of the response model relating the feed and several less-important outputs. The undesired side-effect of this practice is that the controller will be unable to move the feedrate when this is the only solution to avoid constraint violation on such outputs, thus compromising the overall performance.

The relaxation algorithm used in the steady state target calculation represents another opportunity for improvement. The usual algorithm basically dualizes some constraints, i.e., it transfers them to the objective function as terms minimizing the violation of the original constraints. This relaxation strategy frequently results in violations of the limits of variables that are currently within these limits, which is an undesirable change in the controller behavior. This happens because there is no straightforward way to determine which and how many

limits should be relaxed. Additionally, when violations are unavoidable, some inputs are no longer minimized (or maximized) without any obvious reason for the plant operators.

Hence the algorithm proposed in this paper includes binary variables to represent the decisions to move the manipulated inputs during the control horizon, and these decisions can be penalized in the objective function or subjected to a priority sequence. This ensures that the available spans of the less important inputs are exhausted before the algorithm moves the more important ones. Binary variables are also included to represent the decisions to allow violations of the upper or lower limits of the controlled outputs. In an analogous way, these decisions can be penalized or prioritized, meaning that a specific sequence of permissions to violate the limits can be defined.

The inclusion of these binary decisions provides the control engineer additional freedom to define the expected behavior of the algorithm, but on the other hand this behavior can be negatively affected by an inadequate definition of priorities or weights, and it is important to state that this paper does not attempt to address the stability and robustness properties in any other way than through examples, and we recognize that this is an important issue that deserves future theoretical work. Additionally, this paper does not analyze the application of the proposed algorithm to nonlinear or hybrid systems, and this may also constitute a worthy subject for future investigation.

It is interesting to add that the inclusion of the cited binary variables eases a future integration of the two layers that constitute the traditional linear controllers, since the main reason for keeping them as separate formulations is the fact that the original problem solved by the upper layer may be infeasible, and this is not the case in the proposed algorithm. The integration will probably result in a formulation whose theoretical properties can be more easily analyzed.

The outline of the paper is as follows. In Section 2, the usual continuous MPC formulation, comprising the static and dynamic layers, is presented. Section 3 describes the proposed mixed-integer formulation for the static layer, responsible for deriving the steady-state targets for inputs and outputs. Section 4 deals with the formulation of the mixed-integer dynamic layer, which calculates the control actions. The mixed-integer quadratic programming solver developed to derive the solutions of both layers is described in section 5. Section 6 covers the application of the proposed formulation on the 4-tank benchmark system, including a comparison of the results with the ones obtained with the traditional controller. Section 7 also deals with the application on a benchmark system, the Shell control problem. In Section 8 the proposed formulation is applied on a simulated industrial system and a comparison with the continuous controller is provided. Finally, Section 9 concludes this paper by briefly summarizing the results.

2. Continuous MPC formulation

According to Sotomayor *et al*. (2009), the MPC target calculation layer, also called steady-state linear optimizer, solves at each sampling instant a QP problem where the objective is to force one or more inputs to their bounds, while keeping the outputs inside the bounds. This problem may be defined as follows:

$$\underset{\Delta\tilde{u},\,\delta^{y}}{Min}\ \varphi^{ss} = \frac{1}{2}.\Delta\tilde{u}^{T}W_{0}\Delta\tilde{u} + W_{1}^{T}\Delta\tilde{u} + W_{2}^{T}\delta^{y} \tag{1}$$

subject to:

$$\begin{aligned}
&\Delta\widetilde{u} = \widetilde{u} - u \\
&\widetilde{y} = G_0 \Delta\widetilde{u} + \hat{y}_{k+n|k} \\
&u^{LB} \le \widetilde{u} \le u^{UB} \\
&y^{LB} \le \widetilde{y} + \delta^{y} \le y^{UB}
\end{aligned} \tag{2}$$

where:

u = vector of the current values of the inputs (implemented at time $k-1$),

$\widetilde{u}$ = vector of steady-state targets of the inputs,

$\widetilde{y}$ = vector of steady state targets of the outputs,

δ^{y} = vector of slack variables for the controlled outputs,

G_0 = steady-state gain matrix of the process,

k = the present time step,

n = settling time of the process in open loop,

W_0, W_1, W_2 = weight matrices,

u^{LB}, u^{UB} = bounds of the manipulated inputs,

y^{LB}, y^{UB} = bounds of the controlled outputs.

In the equations above, $\hat{y}_{k+n|k}$ represents the contributions of the past inputs to the predicted output at time step $k+n$, i.e., at the end of the time horizon.

The solution of the problem defined by equations (1) and (2) generates the input targets that are transferred to the MPC dynamic layer. The version of MPC we consider in this work is a modification of the quadratic dynamic matrix control (QDMC) as described in García and Morshedi (1986) and Soliman *et al.* (2008). This version solves the following optimization problem:

$$\min_{\Delta\hat{u}_k} \varphi^{qdmc} = \left(\overline{y}_k - y^{sp}\right)^T Q\left(\overline{y}_k - y^{sp}\right) + \Delta\overline{u}_k^T \Lambda \Delta\overline{u}_k + \left(\overline{u}_k - \widetilde{u}\right)^T R\left(\overline{u}_k - \widetilde{u}\right) \tag{3}$$

Subject to:

$$\begin{aligned}
&-\Delta u^{LB} \le \Delta\overline{u}_k \le \Delta u^{UB} \\
&u^{LB} \le \overline{u}_k \le u^{UB} \\
&\overline{y}_k = A\Delta\overline{u}_k + \hat{y}_k
\end{aligned} \tag{4}$$

where:

$\overline{y}_k = \left[\overline{y}_{k+1|k}^T, \overline{y}_{k+2|k}^T, \cdots, \overline{y}_{k+p|k}^T\right]^T$

y^{sp} = vector of set-points to the system outputs. These set-points are usually made equal to $\widetilde{y}$.

$\overline{u}_k = \left[\overline{u}_{k|k}^T, \overline{u}_{k+1|k}^T, \cdots, \overline{u}_{k+l-1|k}^T\right]^T$

$\Delta\overline{u}_k = \left[\Delta\overline{u}_{k|k}^T, \Delta\overline{u}_{k+1|k}^T, \cdots, \Delta\overline{u}_{k+l-1|k}^T\right]^T$

$\hat{y}_k = \left[\hat{y}_{k+1|k}^T, \hat{y}_{k+2|k}^T, ..., \hat{y}_{k+p|k}^T\right]^T$

Δu^{UB} = upper limit to the control moves,

l = control horizon,

p = prediction horizon,
Q, Λ and R are weighting matrices.

In equations (3) and (4) $\bar{y}_{k+\ell|k}$ represents the predicted output at time step $k+\ell$, based on information available at time step *k*, including the planned control moves. This prediction also includes the effect of measured disturbances that were not shown for the sake of simplicity. *A* is the dynamic matrix relating future input changes $\Delta\bar{u}_k$ to predicted outputs $\bar{y}_k$, and which for a finite step response model is given by:

$$A = \begin{bmatrix} S_1 & 0 & \cdots & 0 \\ S_2 & S_1 & \cdots & 0 \\ \vdots & \vdots & \cdots & S_1 \\ \vdots & \vdots & \cdots & \vdots \\ S_P & S_{P-1} & \cdots & S_{P-M+1} \end{bmatrix} \tag{5}$$

where the S_i are unit step response coefficients.

If we replace $\bar{y}_k = A\Delta\bar{u}_k + \hat{y}_k$ in eq. (3) and eliminate the constant terms, the objective function can be rewritten as:

$$\min_{\Delta\bar{u}_k} \varphi^{qdmc} = \frac{1}{2}\Delta\bar{u}_k{}^T\left(A^T Q A + \Lambda\right)\Delta\bar{u}_k + A^T Q\left(\hat{y}_k - y^{sp}\right)\Delta\bar{u}_k + \frac{1}{2}\left(\bar{u}_k - \tilde{u}\right)^T R\left(\bar{u}_k - \tilde{u}\right) \tag{6}$$

This is the well-known QDMC equation with the addition of the term for driving the inputs towards their steady state optimal values. The formulation is similar to the structure of several MPC packages widely applied in refining and petrochemical processes.

3. Steady-State Optimization using MIQP Approach

The development of optimization models involving discrete and continuous variables is not a trivial task, since for the same problem several alternative formulations may give rise to very different performance in terms of solution times (Vecchietti *et al.,* 2003). It is also known that the inclusion of discrete degrees of freedom can drastically increase the complexity of the problem in a way that may limit its applicability to relatively small systems (Till *et al*., 2004). In our case, we take the approach to depart as little as possible from the traditional continuous algorithm, and propose to replace the steady state target calculation described by equations (1) and (2) by a mixed-integer quadratic problem (MIQP), as described below.

The objective function is modified by including terms maximizing the decision to enforce the upper/lower limits on the outputs and minimizing the decision to move the inputs in the positive or negative directions. Both decisions are represented by binary variables:

$$\min_{\Delta\tilde{u}, y^v, z^{u+}, z^{u-}, z^y} \varphi^{miss} = \frac{1}{2} y^{vT} \varpi y^v - \pi^{yT} z^y + \frac{1}{2}\Delta\tilde{u}^T \mu \Delta\tilde{u} - \lambda^{uT}\Delta\tilde{u} + \pi^{uT}\left(z^{u+} + z^{u-}\right) \tag{7}$$

where:
y^v = amount of upper or lower limit violation for each output;
μ = matrix of minimum movement tuning parameters for the inputs;
λ^u = vector of profit tuning parameters for the inputs;

π^u = vector of priority parameters for the inputs;
π^y = vector of priority parameters for the outputs;
ϖ = matrix of weight parameters for output violations;
z^y = vector of binary variables to enforce the bounds of the outputs (if equal to 0, the limits are relaxed);
z^{u+} = vector of binary variables to increase inputs (if equal to 1, the input can be increased);
z^{u-} = vector of binary variables to decrease inputs (if equal to 1, the input can be decreased),

The vectors λ^u, π^u, z^{u+} and z^{u-} are m-dimensional, while the vectors π^y, and z^y are n-dimensional, where m is the number of inputs and n is the number of outputs. Additionally μ is an $m \times m$ diagonal matrix, and ϖ is an $n \times n$ diagonal matrix.

Equality constraints

In order to allow the addition of constraints for the minimum movement of the inputs, we introduce the variables $\Delta\tilde{u}^+, \Delta\tilde{u}^- \geq 0$, such that:

$$\Delta\tilde{u} = \Delta\tilde{u}^+ - \Delta\tilde{u}^- \tag{8}$$

Inequality constraints

Equations defining the amount of upper and lower limit violations for each CV:

$$y^v \geq y^{LB} - \left(\hat{y} + G_0 \Delta\tilde{u}\right) \tag{9}$$

$$y^v \geq \left(\hat{y} + G_0 \Delta\tilde{u}\right) - y^{UB} \tag{10}$$

where:
y^{LB} = vector of lower operation limits for the outputs,
y^{UB} = vector of upper operation limits for the outputs,
The violations of output limits y^v are nonnegative numbers,

$$y^v \geq 0 \tag{11}$$

The target values for the inputs must be placed within the allowable range, defined by the lower and upper limits, u^{LB} and u^{UB}:

$$u^{LB} \leq u + \left(\Delta\tilde{u}^+ - \Delta\tilde{u}^-\right) \leq u^{UB} \tag{12}$$

If the decision to enforce the limits of any output is taken, then the target for this input must remain within the allowable range defined by the Lower and Upper limits:

$$\tilde{y} + M\left(e - z^y\right) \geq y^{LB} \tag{13}$$

$$\tilde{y} - M\left(e - z^y\right) \leq y^{UB} \tag{14}$$

where M = big-M constant (scalar) and $e = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T$.

The values of the inputs u can only be changed if the corresponding binary decision variable (z^{u+} or z^{u-}) is selected:

$$\Delta\widetilde{u}^{+} \le Mz^{u+} \tag{15}$$

$$\Delta\widetilde{u}^{-} \le Mz^{u-} \tag{16}$$

The decisions to increase or decrease an input are mutually exclusive, i.e., an input cannot be simultaneously moved in the positive and negative directions:

$$z^{u+} + z^{u-} \le e \tag{17}$$

Once the decision to move an input is taken, the change to be applied to this input must be greater than the minimum threshold limit Δu^{LB} :

$$\Delta\widetilde{u}^{+} \ge \Delta u^{LB} \circ z^{u+} \tag{18}$$

$$\Delta\widetilde{u}^{-} \ge \Delta u^{LB} \circ z^{u-} \tag{19}$$

In equations (18) and (19) the symbol $\circ$ represents the element-wise multiplication of two matrices, also called Hadamard product (Johnson, 1990), as shown in eq. (20):

$$A \circ B = [a_{i,j}] \circ [b_{i,j}] = [a_{i,j}.b_{i,j}] \tag{20}$$

Equations (18) and (19) can be rewritten in terms of the elements of the vectors, i.e., one equation for each input. This procedure yields the equations (21) and (22):

$$\Delta\widetilde{u}_j^{+} \ge \Delta u_j^{LB}.z_j^{u+} \qquad \forall j = 1,...,m \tag{21}$$

$$\Delta\widetilde{u}_j^{-} \ge \Delta u_j^{LB}.z_j^{u-} \qquad \forall j = 1,...,m \tag{22}$$

In equations (21) and (22), $\Delta\widetilde{u}_j^{+}$ represents the j^{th} element of vector $\Delta\widetilde{u}^{+}$, and the same reasoning applies to the other vectors. The following equations of this section are written in this form, i.e., element by element instead of using vectors, since it helps for a better understanding of the actual algorithm implementation.

The movements to be applied to the inputs are subject to a sequence of priorities according to:

$$z_i^{u+} + z_i^{u-} \le z_j^{u+} + z_j^{u-} \qquad \forall i, j = 1,...,m; i \ne j, \pi_i^u > \pi_j^u \tag{23}$$

As previously explained π_j^u, z_j^{u+} and z_j^{u-} stand for the j^{th} element (i.e., the j^{th} independent variable) of the vectors π^u, z^{u+} and z^{u-} respectively. These vectors have the form shown bellow:

$$\pi^u = \begin{bmatrix} \pi_1^u & \pi_2^u & \cdots & \pi_m^u \end{bmatrix}^T,$$
$$z^{u+} = \begin{bmatrix} z_1^{u+} & z_2^{u+} & \cdots & z_m^{u+} \end{bmatrix}^T \text{ and}$$
$$z^{u-} = \begin{bmatrix} z_1^{u-} & z_2^{u-} & \cdots & z_m^{u-} \end{bmatrix}^T.$$

The constraints described by eq. (23) mean that the decision to move an input can only be selected if all the other inputs with lower priorities have also been moved. Note that the priorities are also used as weights in the objective function (eq. (7)), and so the order in which their values are selected as well as the absolute values themselves influence the results of the algorithm.

The decision to relax a limit of any output can only be selected when all inputs with lower priority than this output have already been moved. This condition is enforced by the following equation:

$$z_i^y \geq 1-(z_j^{u+}+z_j^{u-}) \tag{24}$$

This equation holds for any pair i, j so that $\pi_i^y > \pi_j^u$, $G_{ij} \neq 0$, and the input *j* is on. As a matter of fact, eq. (24) does not guarantee that the algorithm keeps output *i* within the bounds, because the allowable range for the inputs may already be exhausted.

4. Dynamic layer using MIQP Approach

The objective function in this case departs very little from the one defined by eq. (6), essentially only through the addition of a term minimizing the binary variables, as shown by eq. (25).

$$\min_{\Delta\overline{u}_k} \varphi^{qdmc} = \frac{1}{2}\Delta\overline{u}_k{}^T\left(A^TQA+\Lambda\right)\Delta\overline{u}_k + A^TQ\left(\hat{y}_k - y^{sp}\right)\Delta\overline{u}_k + \frac{1}{2}\left(\overline{u}_k-\widetilde{u}\right)^T R\left(\overline{u}_k-\widetilde{u}\right) + \pi^{u^T}\left(\overline{z}^{u+}+\overline{z}^{u-}\right) \tag{25}$$

The remaining equations of the present section describe the constraints involved in the algorithm.

The future values of the inputs must remain within the allowable range:

$$\Gamma_l\Delta\overline{u}_k + u \geq u^{LB} \tag{26}$$

where $\Delta\overline{u}_k$ represents the vector of changes in the inputs at the future time instants, and Γ_l is an $l \times l$ lower triangular matrix of ones, as shown in eq. (27):

$$\Gamma_l = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{27}$$

Eq. (26) represents $m \times l$ constraints that guarantee that the calculated future values of the inputs are not smaller than the lower bound, expressed by u^{LB}. Similar constraints are used to enforce the upper limit, as shown by eq. (28).

$$u^{UB} \geq \Gamma_l\Delta\overline{u}_k + u \tag{28}$$

Every future control action is constrained by the same binary variables $\overline{z}^{u+}$ and $\overline{z}^{u-}$ and if both are zero then the net control action must also be zero although each individual action in the control horizon l may be nonzero. On the other hand, if $\overline{z}_{u+}$ is equal to 1 then the net control action, i.e., the sum of all l future moves must be nonnegative, although any individual action within the control horizon may be negative.

The limit on the rate of change in the inputs is enforced by equation (29):

$$\Delta u^{UB} \circ \bar{z}^{u+} \geq \Delta \bar{u}_k{}^T \tag{29}$$

A similar equation enforces the rate-of-change in the negative direction:

$$\Delta \bar{u}_k \geq -\Delta u^{UB} \circ \bar{z}^{u-} \tag{30}$$

The control actions must be greater than the minimum rate of change and smaller than the maximum rate of change.

$$\Delta \bar{u}_k - \Delta u^{LB} \circ \bar{z}^{u+} + \Delta u^{UB} \circ \bar{z}^{u-} \geq 0 \tag{31}$$

$$-\Delta \bar{u}_k + \Delta u^{UB} \circ \bar{z}^{u+} - \Delta u^{LB} \circ \bar{z}^{u-} \geq 0 \tag{32}$$

Equations (31) and (32) guarantee that the lower and upper limits for the rate of change will be enforced on the first control move. This condition also demands the introduction of the following equation, defining that positive and negative moves are exclusive:

$$e \geq \bar{z}^{u+} + \bar{z}^{u-} \tag{33}$$

In this algorithm only a subset of the l future control moves can be nonzero. These moves are placed in future time instants selected by the algorithm, and such condition is enforced by eq. (34):

$$h \geq z^{hT}.e \tag{34}$$

where z^h is a vector of h binary variables that specifies whether nonzero control actions are allowed in time instant $k, (k = 1,...,h)$. The parameter h is the maximum number of future control actions that can be nonzero. Naturally $h \leq l$, and the controller is able to calculate h future control moves, which can be implemented up to the time instant l.

Equations (35) and (36) then define that the control action can only be nonzero when the binary variable z^h is selected, and additionally, constrain this action to remain between $-\Delta u^{UB}$ and Δu^{UB}:

$$\Delta u^{UB} \circ z^h \geq \Delta \bar{u}_k \qquad \forall k = 1,...,h \tag{35}$$

$$\Delta \bar{u}_k \geq -\Delta u^{UB} \circ z^h \qquad \forall k = 1,...,h \tag{36}$$

5. Mixed-integer quadratic programming solver

The MIQP problem described in the previous section is solved by an algorithm based on the Outer-Approximation method (Duran and Grossmann, 1986), consisting of a series of QP subproblems and MILP master problems. The algorithm follows closely the one described in Grossmann (2002) and considers the optimization problem shown by eq. (37):

$$\text{P}\begin{cases} \min\limits_{x} \varphi = \dfrac{1}{2} x^T C x + D^T x \\ s.t. \\ Ax + B \geq 0 \\ x^L \leq x \leq x^U \end{cases} \tag{37}$$

where x is the vector of variables x_i ($i = 1, \cdots, n$), which includes both continuous and discrete variables (in this paper we consider that the discrete variables are binary). The first n_b variables are binary and the subsequent ones are continuous. Vectors x^L and x^U contain the upper and lower bounds for x, which can also be described as:

$$x_i \in \{0,1\} \qquad \forall i = 1, \cdots, n_b \tag{38}$$

$$x_i \in [x_i^L, x_i^U] \qquad \forall i = n_b + 1, \cdots, n \tag{39}$$

where C is an $n \times n$ positive definite matrix, D is an $n-$dimensional vector, A is an $m \times n$ matrix, and B is an $m-$dimensional vector, where m is the number of constraints.

The algorithm consists of the following steps:

1 – Set the upper bound for the objective function $\varphi^{UB} = \infty$, and solve problem P as a relaxed QP, i.e., set $x_i \in [0,1], \forall i = 1, \cdots, n_b$ and let $x^{NLP,k}$, with $k = 0$ be the solution vector. If the solution is integral, which means that every $x_i^{NLP,0}, \forall i = 1, \cdots, n_b$ has a value of 0 or 1, then this solution is optimal for P. Otherwise, proceed with the algorithm.

2 –Linearize the objective function around $x^{NLP,k}$, set $k = k + 1$, and solve the following MILP:

$$\begin{aligned} &\underset{x}{Min}\, \varphi_{milp} = \alpha \\ &s.t. \\ &\alpha \geq \left(x^{NLP,j^T} C + D^T \right) x - \frac{1}{2} x^{NLP,j^T} C x^{NLP,j}, j = 1, \cdots k \\ &Ax + B \geq 0 \\ &x^L \leq x \leq x^U \\ &\alpha \in R^1 \end{aligned} \tag{40}$$

which yields a lower bound φ_{milp}^k, and the solution vector $x^{MILP,k}$.

3 - Fix the binary variables $x_i^k = x_i^{MILP,k}, \forall i = 1, \cdots, n_b$, and solve P with only the $x_i, \forall i = n_b + 1, \cdots, n$ as free variables, thus obtaining $x^{NLP,k}$. The NLP objective function value $\varphi^{NLP,k}$ represents an upper bound. We then set $\varphi^{UB} = \min\left\{\varphi^{NLP,k}\right\}$. If φ^{UB} is equal to $\varphi^{MILP,k}$ within a given tolerance, the algorithm converged and $x_i^{NLP,k}$ is the optimum solution. Otherwise, proceed to step 2.

The QP subproblem is solved using the QL algorithm, written in FORTRAN by Schittkowski (2005), while the MILP master problem is solved by lp_solve, which is a freely available LP/MILP solver written by M. Berkelaar at Eindhoven University of Technology.

6. Application to the four-tank benchmark control problem

The first benchmark control problem used in this work is a simulated version of the four-tank plant proposed originally by Johansson (2000). Actual laboratory plants as well as simulated versions of this benchmark have been used in several contributions, like for instance Gatzke *et al.* (2000), and Mercangöz and Doyle (2007).

Figure 1 depicts schematically the 4-tank process used in this study. Two pumps that suction from the storage tank located at the bottom of the plant fill the 4 tanks. The tanks 3 and 4 discharge into the corresponding tanks located below them, i.e, tanks 1 and 2, respectively. Two three-way valves distribute the incoming flows, q_a and q_b, according to a predefined proportion. The control scheme to be tested manipulates these two flows to drive some or all the four tank levels, h_1 through h_4 to their setpoints.

The four tank plant has some interesting properties that make it a suitable benchmark system for predictive controllers. Among others, one can cite the high degree of coupling among the subsystems, the nonlinear dynamics, and the fact that inputs are subject to hard constraints.

Figure 1. Schematic diagram of the 4-tank benchmark system

A mathematical model based on simple material balances and Bernoulli's Law represents reasonably well the plant's dynamic behavior. This model consists of equations (41) through (44).

$$\frac{dh_1}{dt} = -\frac{a_1}{S}\sqrt{2gh_1} + \frac{a_3}{S}\sqrt{2gh_3} + \frac{\gamma_a}{S}q_a \tag{41}$$

$$\frac{dh_2}{dt} = -\frac{a_2}{S}\sqrt{2gh_2} + \frac{a_4}{S}\sqrt{2gh_4} + \frac{\gamma_b}{S}q_b \tag{42}$$

$$\frac{dh_3}{dt} = -\frac{a_3}{S}\sqrt{2gh_3} + \frac{(1-\gamma_b)}{S}q_b \tag{43}$$

$$\frac{dh_4}{dt} = -\frac{a_4}{S}\sqrt{2gh_4} + \frac{(1-\gamma_a)}{S}q_a \tag{44}$$

where h_i, and a_i with $i \in \{1,2,3,4\}$ represent the liquid level and the discharge constant of tank i, respectively, while S is the cross section, which is the same for all tanks. The outlet flow from pump $j \in \{a,b\}$ is represented by q_j and the flow ratio of the 3-way valves is represented by γ_a and γ_b, and g is the gravitational acceleration. In this work we use the parameters of the plant built by Alvarado *et al.* (2011), which are described in Table 1.

Table 1. Parameters used in the simulation of the 4-tank system

It must be noticed that the feasible region in steady state can be easily derived from equations (41) through (44). Figure 2 depicts this feasible region considering the upper and lower limits defined in table 1.

Figure 2. Feasible region of the 4-tank system

The performance evaluation considers only the servo case in which the setpoints for the controlled variables are changed. No measured or unmeasured disturbances are considered, and additionally, the aim is to drive only h_1 and h_2 to their setpoints, while h_3 and h_4 may fluctuate within their reasonably wide allowable ranges (0.2m to 1.3m, as shown in Table 1).

In the following plots QP refers to the behavior of the variables when the traditional continuous Linear-MPC is controlling the system, while MIQP refers to the situation when the proposed mixed-integer algorithm is in use.

In these simulations the system is initially at steady state and the variables are set to the values shown in Table 2. After 2 minutes a change in the set point of one or more outputs is imposed, and the behavior of the simulated system is recorded. Both algorithms are executed at the same frequency, i.e., once every 6 seconds.

Table 2. Initial values for the inputs and outputs of the 4-tank system

The purpose of the first simulated test is to derive a set of tuning parameters that would allow both algorithms to respond to setpoint changes at an adequate speed without incurring in unstable behavior. The parameters themselves assume different values in both algorithms, since the formulations are also different. In addition to this, some parameters are only present in the mixed-integer formulation, which is the case for instance of the priorities for inputs and outputs. Specifically, we set the priorities for the inputs as $\pi_{h1}^{y} = 5$, $\pi_{h2}^{y} = 3$, $\pi_{h3}^{y} = 1$, and $\pi_{h4}^{y} = 1$ (as shown in Table 3), and this means that whenever it is not possible to drive both h_1 and h_2 to their setpoints, then h_1 should have the preference. This rule cannot be easily implemented in the continuous controller as it relies essentially on the minimization of the weighed errors of all variables.

Table 3 shows the values of the main tuning parameters used in both cases.

Table 3. Tuning parameters used in the 4-tank case

In this test a setpoint change for both h_1 and h_2 from their steady-state values to 0.3m is imposed at time instant 120 seconds. Figure 2 shows that this set of setpoints is feasible in steady state. As can be seen in Figure 3, both controllers are capable of driving h_1 and h_2 to their new setpoints, while keeping h_3 and h_4 within range. The MIQP algorithm proves to be somewhat faster but still stable, a fact that is confirmed by the behavior of the inputs q_a and q_b, depicted in Figure 4.

Figure 3. Behavior of the outputs when the setpoints of h_1 and h_2 are changed from 0.65m to 0.3m.

Figure 4. Behavior of the inputs q_a and q_b when the setpoints of h_1 and h_2 are changed from 0.65m to 0.3m.

It is necessary to stress that the inclusion of priorities could negatively impact the overall behavior of the controller, since the MIQP formulation has more constraints than the QP one and so could lead to suboptimal results. Nevertheless

this is not observed here, as can be seen by the comparison of the weighted squared cumulative offsets of the outputs in both cases presented in Figure 5. This is a usual measure of overall controller performance, and the results depicted in Figure 5, which consider the weights used in the QP algorithm, show that the MIQP formulation achieves smaller offsets that can be translated as better performance.

Figure 5. Weighted cumulative squared offsets of the outputs when the setpoints of h_1 and h_2 are changed from 0.65m to 0.3m.

One could argue that the smaller offsets are obtained at a significantly higher control cost, i.e., more frequent and larger moves imposed on the inputs, but that is not the case in this example, as can be seem in Figure 6. This figure depicts the cumulative squared moves weighted by the suppression factors used in the QP case. It is clear that the MIQP controller achieves smaller offsets with smaller control costs, and similar results are observed in the subsequent tests.

Figure 6. Weighted cumulative squared input moves of the inputs when the setpoints of h_1 and h_2 are changed from 0.65m to 0.3m.

The second test involves a more interesting situation because the set of setpoints is statically infeasible. In this case the new setpoints for h1 and h2 are 0.9m and 0.4 m, respectively, and Figure 2 shows that this constitutes an infeasible set. In the MIQP algorithm different priorities for the outputs are set and this means that in case of infeasible steady-state the algorithm should to follow a predefined order of relaxation of the outputs.

The simulation results are shown in Figures 7, and 8 and it can be seen that the MIQP controller selects h_1 to be kept at the setpoint while h_2 is driven towards its own setpoint but cannot reach it. This is the expected behavior, since we have defined a higher priority for h_1 when compared to the other outputs. On the other hand, the continuous algorithm also works as expected and keeps neither h_1 nor h_2 in their setpoints, but arrives at a compromise solution where the weighted error is minimized. This has even resulted in a certain amount of permanent violation of the lower bound for h_4.

The behavior of the MIQP algorithm is more consistent with situations where the outputs have different importance. This is the case, for instance, when several interrelated properties of a process stream are controlled and it is unfeasible to keep all of them within range. As a rule, the best solution is to enforce the constraints of the more important outputs while minimizing the violation of the others. This situation is very frequent in the petroleum industry, and the described behavior cannot be easily implemented using continuous controllers. In practice, a set of if-then-else rules must be arbitrarily implemented, a procedure that frequently results in undesirable behavior and is difficult to keep updated.

Naturally, when desired, the MIQP controller can be adjusted to exhibit the same behavior as the continuous just by setting all priorities to zero.

Figure 7. Behavior of the outputs when the setpoints of h1 and h2 are changed respectively to 1.0m and 0.4m.

Figure 8. Behavior of the intputs when the setpoints of h_1 and h_2 are changed respectively to 1.0m and 0.4m.

The third test consists of defining initially a setpoint for h_1 of 1.0m while leaving the other outputs to be controlled by range. Again, as can be seen in Figures 9 and 10, both controllers exhibit similar behavior, with the MIQP being faster. After the system stabilizes, a setpoint of 0.9m is defined for h_2.

Figure 9. Comparative behavior of the outputs when initially the setpoint of h_1 is changed to 1.0m and subsequently a setpoint of 0.9m is defined for h_2.

Although this set of setpoints lies within the feasible region in the static sense, to reach both of them the controllers would need to violate the constraints of h_3 or h_4 during a significant amount of time. As expected, the continuous controller tries to enforce both setpoints and this eventually results in an unstable behavior. On the other hand, the MIQP controller keeps h_1 in its setpoint, and simply starts driving h_2 slowly towards its desired value, without reaching it during the time horizon of the test.

Figure 10. Comparative behavior of the inputs when initially the setpoint of h_1 is changed to 1.0m and subsequently a setpoint of 0.9m is defined for h_2.

Again, the behavior of the MIQP controller is safer and, in general, easier to adjust because one can predefine the sequence of enforcement of the setpoints. It should be noted that the size of the MIQP problems is fairly small, since the static layer gives rise to a problem with 8 continuous and 8 binary variables, which can be solved in less than 0.1 seconds in an INTEL Core Duo 2.4GHz PC running Windows XP™. The dynamic layer is larger but still small, comprising 20 continuous and 14 binary variables, and can also be solved in less than 0.1 seconds in the same computer. These solution times are not significantly longer than the ones necessary to solve the QP problems of the traditional continuous algorithm.

7. Application to the Shell benchmark control problem

The second benchmark used in this work is the Shell control problem, which is the simulation of a heavy oil fractionator with one side-draw in addition to the distillate and bottoms products as depicted in Figure 11. This benchmark problem was first proposed by Prett and Morari (1987) and has been used in several contributions, like for instance, Zheng *et al.* (1994), Rodrigues and Odloak (2000), Kettunen *et al.* (2008), and Li *et al*. (2005). The last authors present the linear model for the fractionator, as shown in eq. (45).

$$y = G(s).u + G_d(s).d \tag{45}$$

where $u = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^T$ are the manipulated inputs of the process. u_1 represents the top product flowrate, u_2 represents the side product flowrate, and u_3 represents the heat duty removed from the column by the lower pumparound. $d = \begin{bmatrix} d_1 & d_2 \end{bmatrix}^T$ are the disturbances influencing the column, which are considered measured. d_1 is the heat duty of the intermediate pumparound, and d_2 represents the heat duty of the top pumparound, and their limits are defined by $|d_i| \le 0.5, i = 1,2$. $y = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}^T$ are the output variables, y_1 representing the composition of the top product, y_2 the composition of side product and y_3 the reflux temperature at the bottom of the column.

Figure 11. Diagram of the heavy oil fractionator and the associated inputs and outputs that constitute the Shell benchmark problem.

The transfer function matrices are given by Li *et al*. (*op.cit*.) and shown in equations (46) and (47).

$$G(s)=\begin{bmatrix} \dfrac{4.05e^{-27s}}{50s+1} & \dfrac{1.77e^{-28s}}{60s+1} & \dfrac{5.88e^{-27s}}{50s+1} \\ \dfrac{5.39e^{-18s}}{50s+1} & \dfrac{5.72e^{-14s}}{60s+1} & \dfrac{6.90e^{-15s}}{40s+1} \\ \dfrac{4.38e^{-20s}}{33s+1} & \dfrac{4.42e^{-22s}}{44s+1} & \dfrac{7.20}{19s+1} \end{bmatrix} \tag{46}$$

$$G_d(s)=\begin{bmatrix} \dfrac{1.20e^{-27s}}{45s+1} & \dfrac{1.44e^{-27s}}{40s+1} \\ \dfrac{1.52e^{-15s}}{25s+1} & \dfrac{1.83e^{-15s}}{20s+1} \\ \dfrac{1.14}{27s+1} & \dfrac{1.26}{32s+1} \end{bmatrix} \tag{47}$$

The control objective is to maintain the top and side compositions (y_1 and y_2) at their specifications (initially 0.0±0.005), while y_3 is to be controlled by range, i.e., must be kept within a reasonably wide operating region defined as $|y_3| \le 0.5$. The constraints on the inputs are set as $|u_i| \le 0.5$ and $|\Delta u_i| \le 0.2, i = 1,2,3$.

The Shell benchmark control problem is considered well adapted to the evaluation of predictive control algorithms due to the interaction among the variables, the relatively long dead-times and the possibility of conflicting process requirements that are not easily satisfied.

The performance evaluation, as is the case with the 4-tank system, consists of a comparison of the system behavior when controlled by MIQP algorithm and the traditional QP multivariable controller. The first step in this comparison is to tune both controllers in a way that results in similar responses when the system is subjected to the same disturbance. In particular we define the same suppression factors (0.1 for the three manipulated inputs, u_1 through u_3) and the same weights for the outputs (4.0, 2.0 and 2.0 for y_1, y_2, and y_3, respectively). The prediction horizon (120 time steps), optimization horizon (80 time steps) and control horizon (1 time steps) are also the same. Additionally we define different priorities for the outputs (4, 3 and 2 for y_1, y_2, and y_3, respectively) and for the inputs (2, 2 and 1 for u_1, u_2, and u_3 respectively), which means that it is more important to drive y_1 to its setpoint than y_2 and y_3, and that the algorithm should preferably move u_3 than u_1 and u_2, and so on.

Figures 12 and 13 show the closed-loop output responses and manipulated signals with both algorithms under the disturbance pattern $d = [0 \quad 0.5]^T$, imposed at the time step 20. The setpoints for y_1 and y_2 are kept at 0.0. Figure 12 shows that the performance in terms of time to bring back the controlled variables to their setpoints and the maximum deviation from these setpoints is similar.

Figure 12. Behavior of the outputs for the Shell benchmark system with a step change of 0.5 in d_2.

The difference is more clearly visualized in Figure 13, which depicts the behavior of the manipulated inputs. It can be seen that the MIQP applies less frequent but larger changes on the inputs, especially u_1, which is moved only once during the entire horizon due to its higher priority.

Figure 13. Comparative behavior of the manipulated inputs for the Shell benchmark system with a step change of 0.5 in d_2.

The different behavior of both algorithms can be better explained with the next simulated test. In this case the setpoints for for y_1 and y_2 were changed from 0.0 to 0.75 and -0.75, respectively, while the limits for y_3 were kept fixed, i.e., from -0.5 to 0.5. This set of setpoints is not feasible, a situation that frequently happens in practice and with which both algorithms deal in different ways. As can be seen in Figure 14, the continuous algorithm, as expected, drives both y_1 and y_2 towards their setpoints, but cannot reach them. Additionally y_3 is temporarily subjected to a large violation of its upper limit. On the other hand, the MIQP algorithm drives y_1 to its setpoint, since this variable has the highest priority, moves y_2 as close as possible to its setpoint, and achieves this result with essentially no violation of the limits of y_3 . This behavior is coherent with the priorities that were imposed and cannot be easily achieved with the continuous algorithm, which essentially minimizes the weighted average of the offsets in the outputs.

Figure 14. Comparative behavior of the outputs for the Shell benchmark system with a change in the setpoints of y_1 and y_2.

Figure 15 depicts the behavior of the manipulated inputs, which is similar in both cases.

Figure 15. Comparative behavior of the manipulated inputs for the Shell benchmark system with a change in the setpoints of y_1 and y_2.

This test shows that the MIQP formulation is capable of achieving better performance than the continuous one, while providing tuning parameters that can be used to better adjust the behavior to cope with different response specifications. It should be noted that similarly to the previous benchmark the size of the MIQP problems is small and the computational effort is not significantly larger than in the QP algorithm. In the present case the MIQP problem that corresponds to the static layer has 8 continuous and 8 binary variables, which can be solved in less than 0.1 seconds. The dynamic layer has 3 continuous and 7 binary variables, and can also be solved in less than 0.1 sec.

8. Application to an FCC simulation

The proposed formulation is also applied to a simulation of a Fluid Catalytic Cracking unit (FCC), as described by Moro and Odloak (1995).

The FCC is one of the most important refining units, and transforms intermediate oil fractions into light and more valuable hydrocarbon products. The FCC converter, which is the main equipment of such units, consists of three major sections: the separator vessel, the regenerator and the riser. The riser is a tubular reactor at whose bottom the preheated liquid feed is injected and mixed with hot fluidized catalyst flowing from the regenerator. This hot catalyst provides the energy for feed vaporization and for the endothermic cracking reactions. These reactions generate lighter hydrocarbons and also a high carbon-content solid, coke, which is deposited over the catalyst surface resulting in its deactivation. The catalyst is reactivated in the regenerator by burning the coke in a fluidized bed.

The MPC configuration used here was taken from the actual industrial implementation and includes 33 outputs and 11 manipulated inputs, and covers the plant subsection from the preheat train to the fractionator column. The steady state layer configured with these variables results in an MIQP with 55 continuous and 44 binary variables, as well as 165 constraints. The problem was solved in 0.25 seconds in an INTEL Core Duo 2.4GHz PC running Windows XP™. On the other hand, the dynamic layer consists of an MIQP with 11 continuous variables, 23 binary variables and 100 constraints, which can be solved in the same machine in less than 0.1 seconds.

Although each one of the variables is kept active in the simulated test described in the next section, the focus here is the control of just one variable, the regenerator temperature, which is mainly affected by the air injection. The air is injected through 3 different pipes and adjusted by 3 flow controllers, FC01, FC01A and FC02, as depicted in Figure 16.

Figure 16. Regenerator air subsystem

FC01 controls the flow in the main injection line and is responsible for about 60% of the total air. FC01A works as a complement to FC01 and is designed for frequent small adjustments. FC02 is responsible for about 15% of the total air flow and is connected to the regenerator second stage.

The best practice for this system consists in using the larger valve, i.e. FC01, only for aggressive control moves, while the smaller ones should be used to deal with the regular fluctuations. The application of frequent movements on the larger valve, besides being ineffective due to hysteresis, generates wear and tear that may lead to premature failure.

The usual approach adopted by control engineers to adjust the controller behavior in such cases is to increase the move suppression term (Λ in eq. (3)) of the input responsible for the larger valve. This usually does not result in the desired behavior, and impairs the MPC ability to deal with situations when aggressive control actions are necessary.

In this simulated test, we show that the mixed-integer formulation is able to generate this behavior, i.e., to move the larger valve only for larger flow modifications, and still provide adequate regulation of the regenerator.

In this simulation, the performances of the proposed MIQP algorithm and of the MPC currently used to control the plant are evaluated and compared. The system is allowed to reach steady state and then a change in the allowable range of the regenerator temperature – a controlled variable – is imposed. This change affects only the lower limit of the temperature, which is raised from 680^{o}C to 700^{o}C. The results are depicted in Figures 17 through 20, where the solid lines represent the behavior with the MIQP formulation, and the dotted lines the behavior with the traditional QP algorithm.

As it can be seen in Figure 17, the temperature profile is similar and adequate in both cases, with the MIQP algorithm being slightly faster in the beginning and bringing the temperature to the lower limit of the allowable range.

Figure 17. Regenerator temperature with the MIQP formulation (T-MI) and with the traditional QP (T-QP).

The behavior of the manipulated variables related to the air injection can be seen in Figures 18 through 20.

It can be noticed that with the MIQP formulation the manipulated variables remain approximately constant, while no setpoint changes are imposed on the controller. On the other hand, it is capable of vigorous action when such change happens. As previously described, this is exactly the kind of behavior that we were aiming for with this mixed-integer formulation.

Figure 18. Main air flow to the Regenerator with the MIQP formulation (Air1-MI) and with the traditional QP (Air1-QP).

Figure 19 shows that the MIQP algorithm uses the main air flow only for aggressive moves, while the traditional MPC makes frequent small adjustments, which can result in wear and tear in the valve.

Figure 19. Secondary air flow to the Regenerator with the MIQP formulation (Air1A-MI) and with the traditional QP (Air1A-QP).

The inputs are more stable with the MIQP formulation, but can be vigorously moved when this is necessary. The traditional formulation changes the inputs frequently, even when little improvement in the system behavior can be obtained.

Figure 20. Air flow to the Regenerator second stage with the MIQP formulation (Air2A-MI) and with the traditional QP (Air2A-QP).

It can be noticed that the two formulations yield different air flowrates in the final steady-state. This is due to the fact that the whole controller remained active during the testing procedure and other manipulated inputs, not analyzed here, also assumed different values.

It is to be expected that better characteristics can be obtained once the controller is retuned to utilize more freely the hybrid approach.

9. Conclusions

In this paper, we have presented MIQP formulations for the steady state and dynamic layers of industrial MPCs. These formulations allow the assignment of explicit priorities for the outputs and for the inputs, which means that one can define *a priori* a preferential order of constraint relaxation, and also the order in which the inputs are to be moved to adjust each output. This approach also allows the definition of a minimum limit for the control moves, which is important in cases where small actions cannot be implemented in practice, for instance due to valve hysteresis. The resulting combined formulation gives rise to MIQP problems that are only modestly harder to solve than the usual QP's, and were actually solved using an algorithm developed as part of this work, which was able to derive the solutions with very small computational effort.

The proposed mixed-integer MPC was applied to two simulated benchmarks and to an industrial case, and compared with the traditional continuous MPC. These tests were used to evaluate the effect of the priorities assigned to the inputs and outputs and also the resulting closed loop stability. The results show that the desired behavior was obtained, resulting in a more predictable, and as far as the simulated tests can show, more stable operation.

As a follow-up to this work, we intend to apply the algorithm to an industrial refining unit and evaluate its performance in comparison with the current MPC. Additionally, its stability and robustness properties will have to be theoretically investigated before the algorithm can be confidently applied in a large number of practical instances, but this step will be preceded by the development of an integrated formulation, where the steady-state and dynamic problems are simultaneously solved.

References

Alvarado, I., Limon, D., Muñoz de la Peña, D., Maestre, J.M., Ridao, M.A., Scheu, H., Marquardt, W., Negenborn, R.R., De Schutter, B., Valencia, F., Espinosa, J. (2011). A comparative analysis of distributed MPC techniques applied to the HD-MPC four-tank benchmark. *Journal of Process Control*, 21, 800–815

Bemporad, A., Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints, *Automatica* 35 (3) 407–427.

Duran, M.A., Grossmann, I.E. (1986). An outer-approximation algorithm for a class of mixed-integer nonlinear Programs. *Math Programming*, 36, 307-339.

García, C.E., Morshedi, A.M. (1986). Quadratic programming solution of dynamic matrix control (QDMC). *Chemical Engineering Communications*, 73-87.

Gatzke, E.P., Meadows, E.S., Wang, C., Doyle III, F.J. (2000). Model based control of a four-tank system, *Computers & Chemical Engineering*, 24, 1503–1509.

Grossmann, I.E. (2002). Review of Nonlinear Mixed-Integer and Disjunctive Programming Techniques. *Optimization and Engineering*, 3, 227–252, 2002.

Johansson, K.H. The quadruple-tank process (2000). *IEEE Transactions on Control Systems Technology,* 8 (3), 456–465

Johnson, C. (1990). *Matrix Theory and Applications*, American Mathematical Society, ISBN 0821801546.

Mercangöz, M., Doyle III, F.J. (2007). Distributed model predictive control of an experimental four-tank system. *Journal of Process Control*, 17, 297–308.

Kettunen, M., Zhang, P., Jämsä-Jounela, S. (2008). An embedded fault detection, isolation and accommodation system in a model predictive controller for an industrial benchmark process. *Computers and Chemical Engineering*, 32, 2966–2985.

Li, S., Zhang, Y., Zhu, Q. (2005). Nash-optimization enhanced distributed model predictive control applied to the Shell benchmark problem. *Information Sciences*, 170 329–349.

Morari, M., Barić, M. (2006). Recent developments in the control of constrained hybrid systems. *Computers and Chemical Engineering*, 30, 1619-1631.

Moro, L. F. L., Odloak, D. (1995). Constrained multivariable control of fluid catalytic cracking converters. *Journal of Process Control,* 5(1), 29–39

Oldenburg, J., Marquardt, W. (2008). Disjunctive modeling for optimal control of hybrid systems. *Computers and Chemical Engineering*, 32, 2346-2364.

Prett, D.M., Morari, M (1987). The Shell Process Control Workshop, *Butterworths*, Boston.

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

Rodrigues, M.A., Odloak, D. (2000). Output feedback MPC with guaranteed robust stability. *Journal of Process Control,* 10, 557-572.

Schittkowski, K. (2005). QL: A Fortran Code for Convex Quadratic Programming - User's Guide, Version 2.11. *University of Bayreuth*. Bayreuth, Germany. http://www.ai7.uni-bayreuth.de/QL.pdf (accessed in Oct 27,2011)

Soliman, M., Swartz, C.L.E., Baker, R. (2008). A mixed-integer formulation for back-off under constrained predictive control. *Computers and Chemical Engineering*, 32, 2409-2419.

Sotomayor, O.A.Z., Odloak, D., Moro, L.F.L. Moro (2009). Closed-loop model re-identification of processes under MPC with zone control. *Control Engineering Practice.* 17, 551-563.

Till, J., Engell, S., Panek, S., Stursberg, O. (2004). Applied hybrid system optimization: An empirical investigation of complexity. *Control Engineering Practice.* 12, 1291–1303.

Vecchietti, A., Lee, S., Grossmann, I.E. (2003). Modeling of discrete/continuous optimization problems: characterization and formulation of disjunctions and their relaxations. *Computers and Chemical Engineering*, 27, 433-448.

Zabiri, H., Samyudia, Y. (2006). A hybrid formulation and design of model predictive control for systems under actuator saturation and backlash. *Journal of Process Control*, *16*, 693-709.

Zheng, H., Li, W., Lee, J. H., Morari, M. (1994). State estimation based model predictive control applied to Shell control problem: a case study. *Chemical Engineering Science,* 49(3), 285-301.

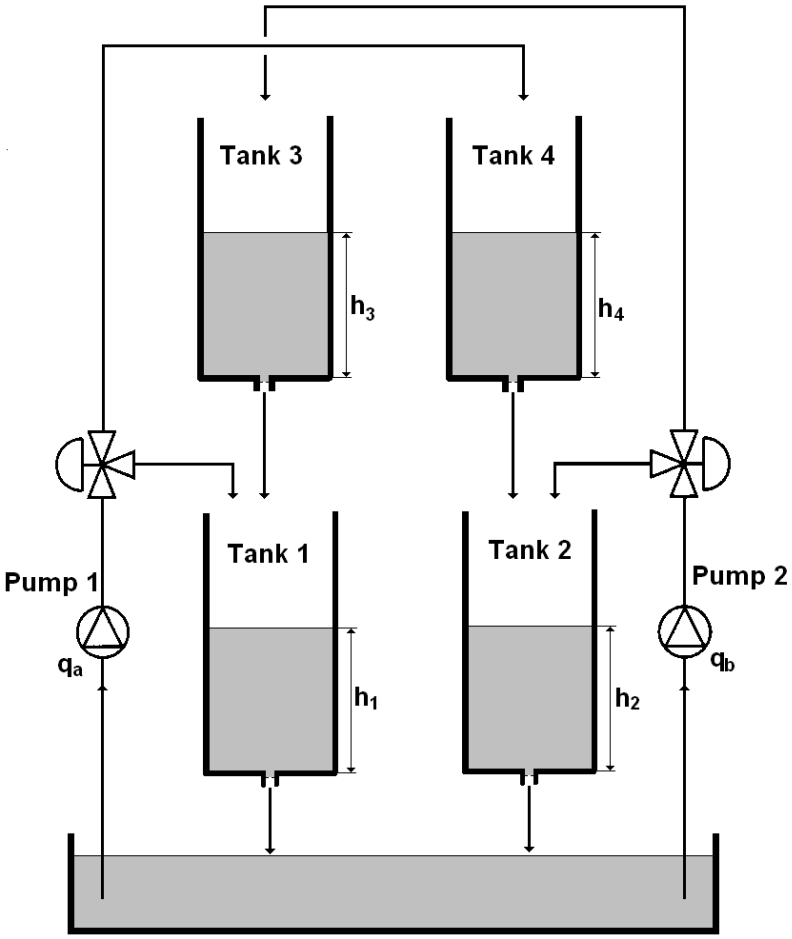

Figure 1. Schematic diagram of the 4-tank benchmark system

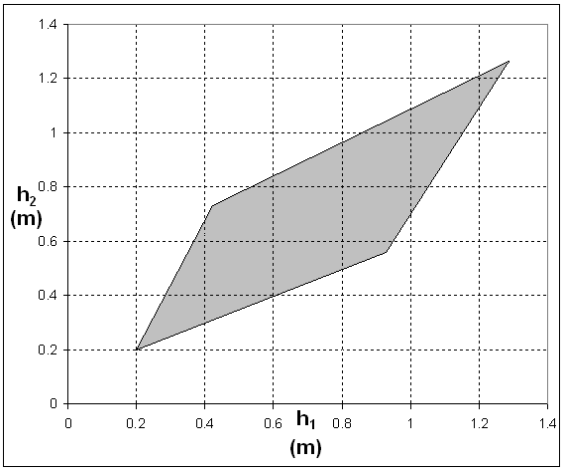

Figure 2. Feasible region of the 4-tank system

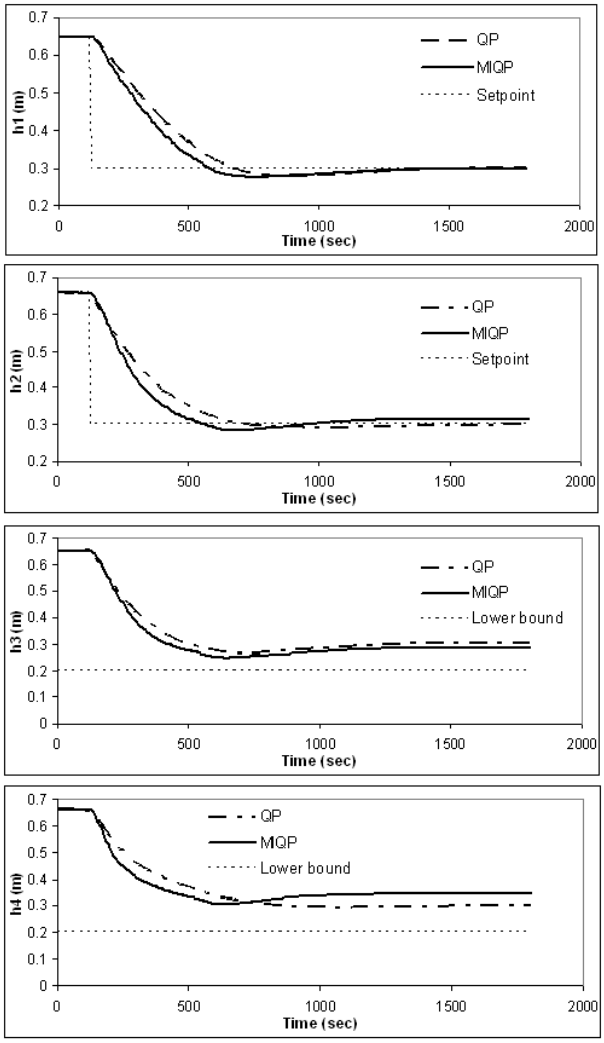

Figure 3. Behavior of the outputs when the setpoints of h_1 and h_2 are changed from 0.65m to 0.3m.

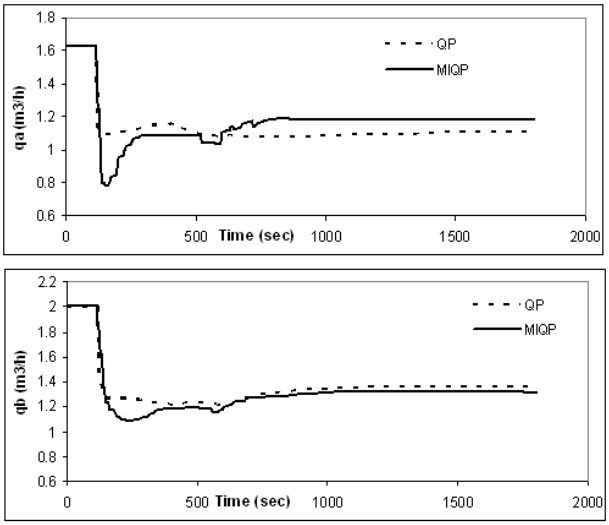

Figure 4. Behavior of the inputs q_a and q_b when the setpoints of h_1 and h_2 are changed from 0.65m to 0.3m.

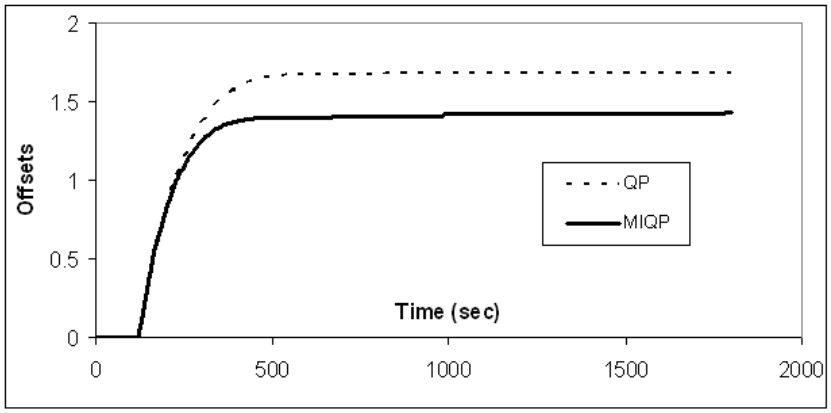

Figure 5. Weighted cumulative squared offsets of the outputs when the setpoints of h_1 and h_2 are changed from 0.65m to 0.3m.

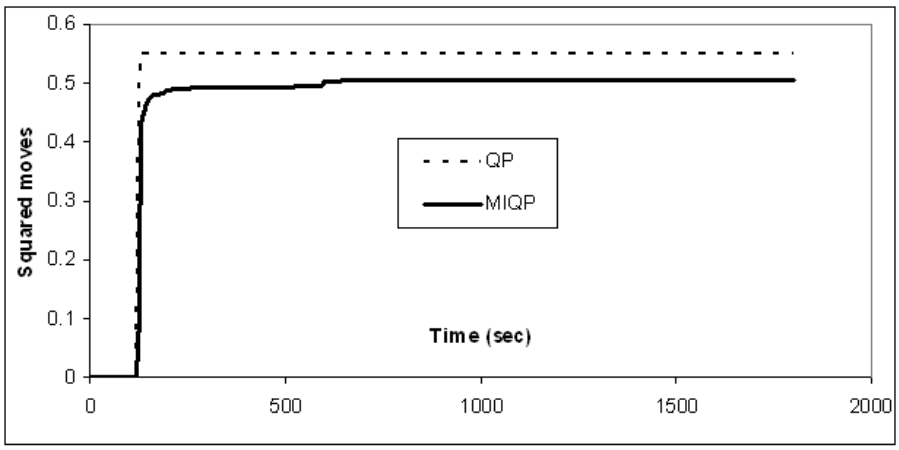

Figure 6. Weighted cumulative squared input moves of the inputs when the setpoints of h_1 and h_2 are changed from 0.65m to 0.3m.

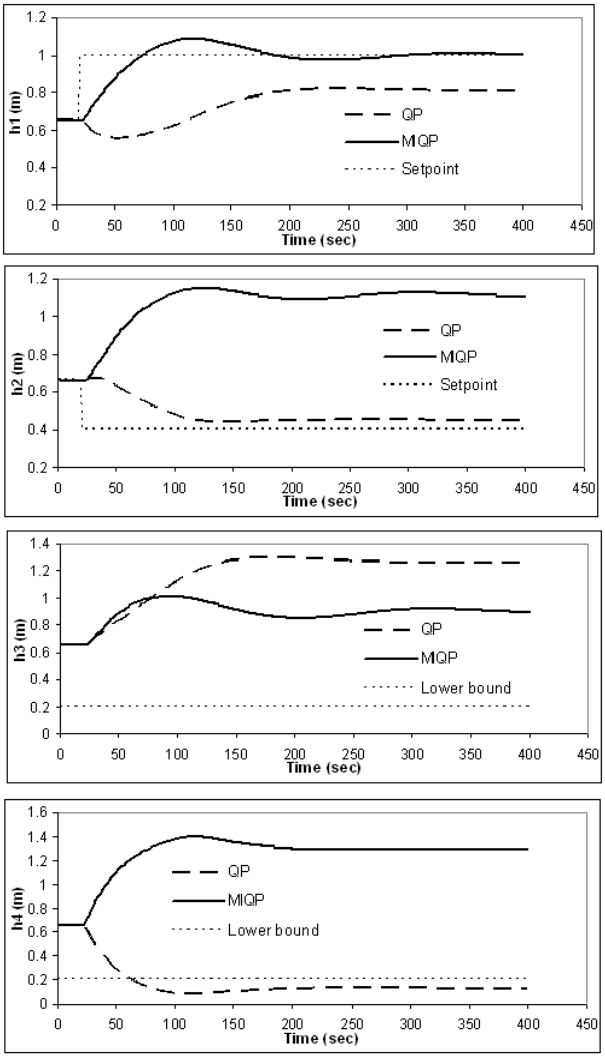

Figure 7. Behavior of the outputs when the setpoints of h1 and h2 are changed respectively to 1.0m and 0.4m.

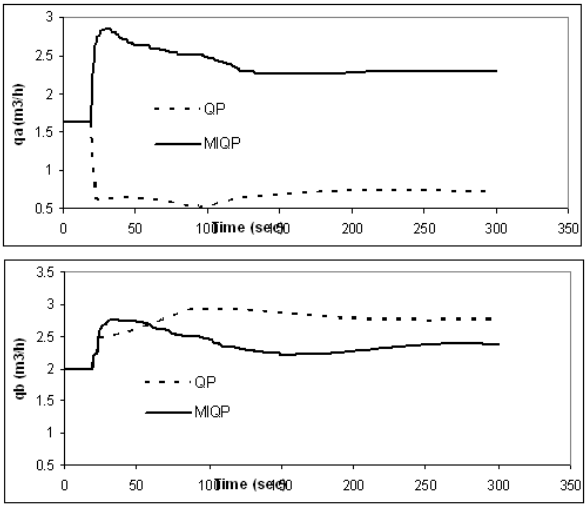

Figure 8. Behavior of the intputs when the setpoints of h_1 and h_2 are changed respectively to 1.0m and 0.4m.

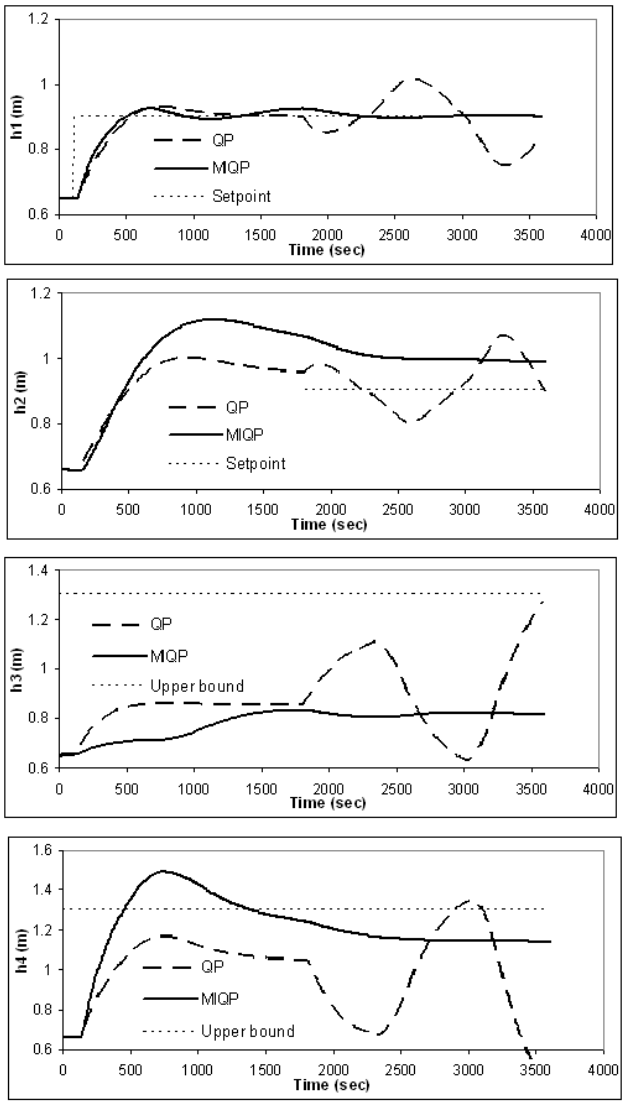

Figure 9. Comparative behavior of the outputs when initially the setpoint of h_1 is changed to 1.0m and subsequently a setpoint of 0.9m is defined for h_2.

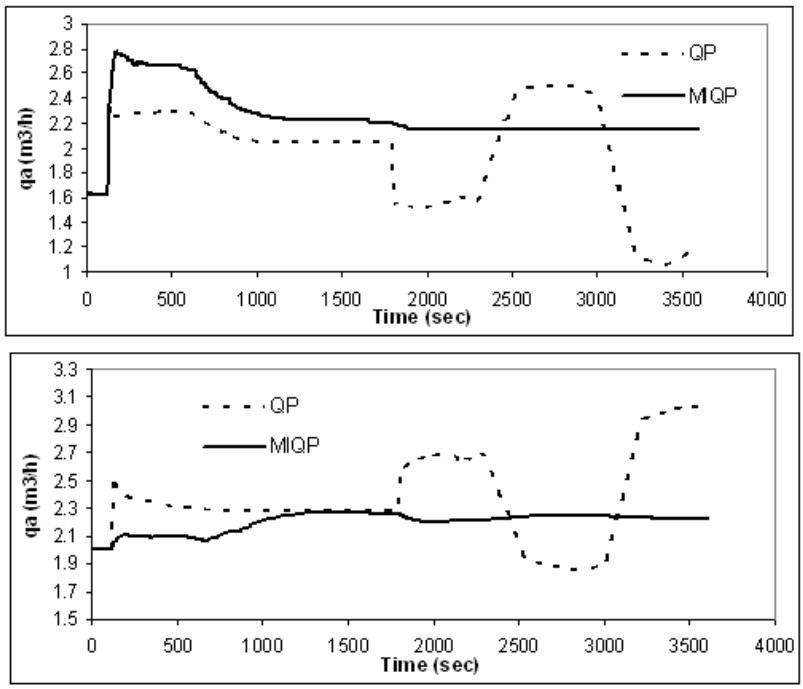

Figure 10. Comparative behavior of the inputs when initially the setpoint of h_1 is changed to 1.0m and subsequently a setpoint of 0.9m is defined for h_2.

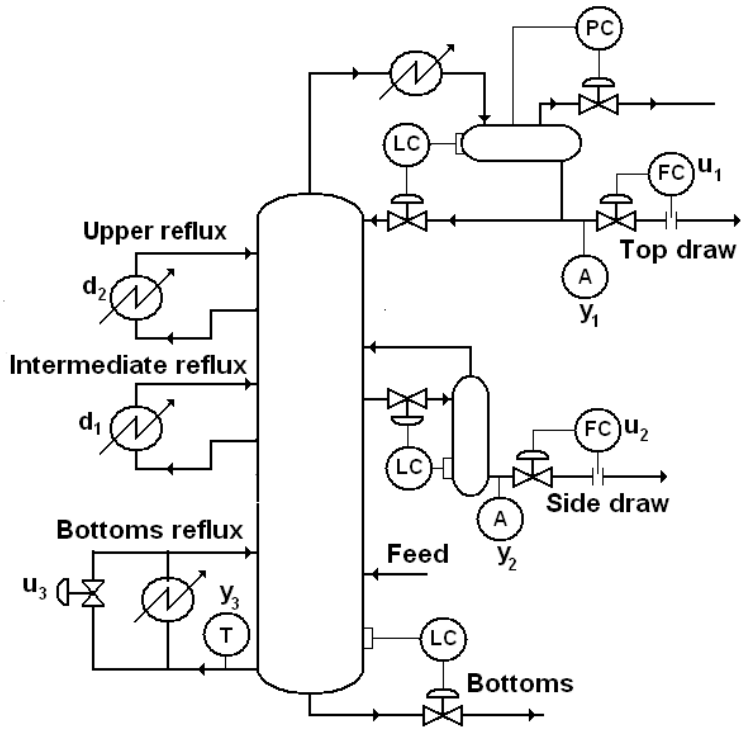

Figure 11. Diagram of the heavy oil fractionator and the associated inputs and outputs that constitute the Shell benchmark problem.

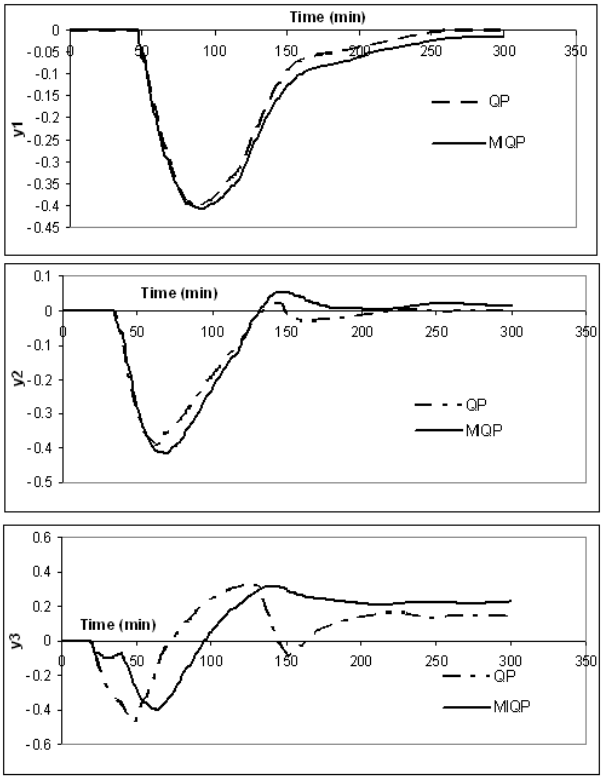

Figure 12. Behavior of the outputs for the Shell benchmark system with a step change of 0.5 in d_2.

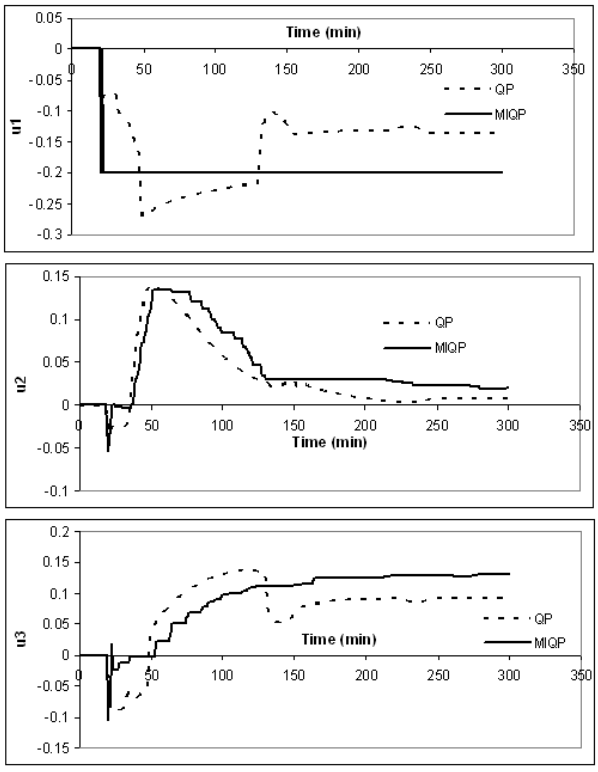

Figure 13. Comparative behavior of the manipulated inputs for the Shell benchmark system with a step change of 0.5 in d_2.

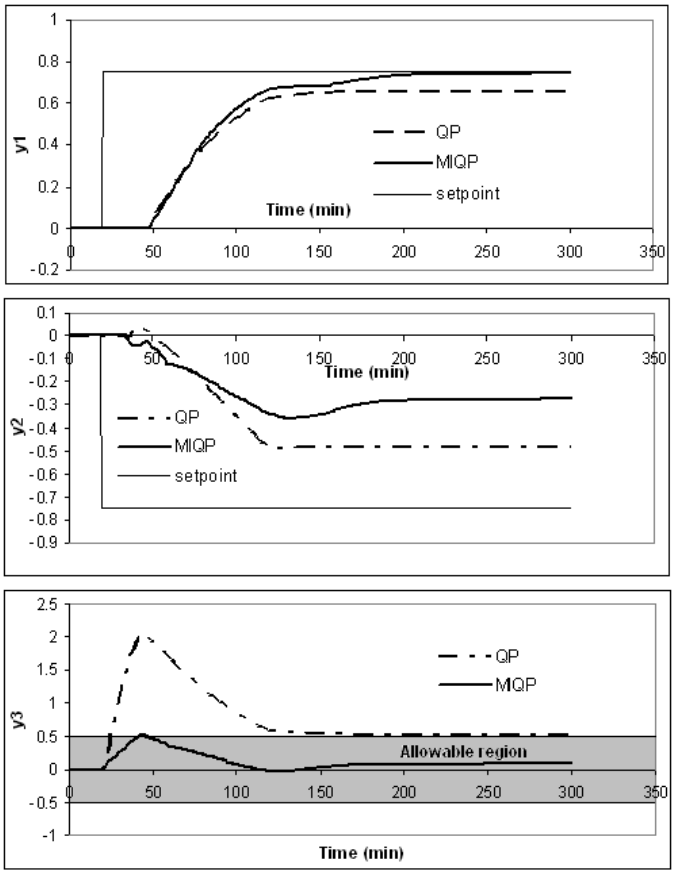

Figure 14. Comparative behavior of the outputs for the Shell benchmark system with a change in the setpoints of y_1 and y_2.

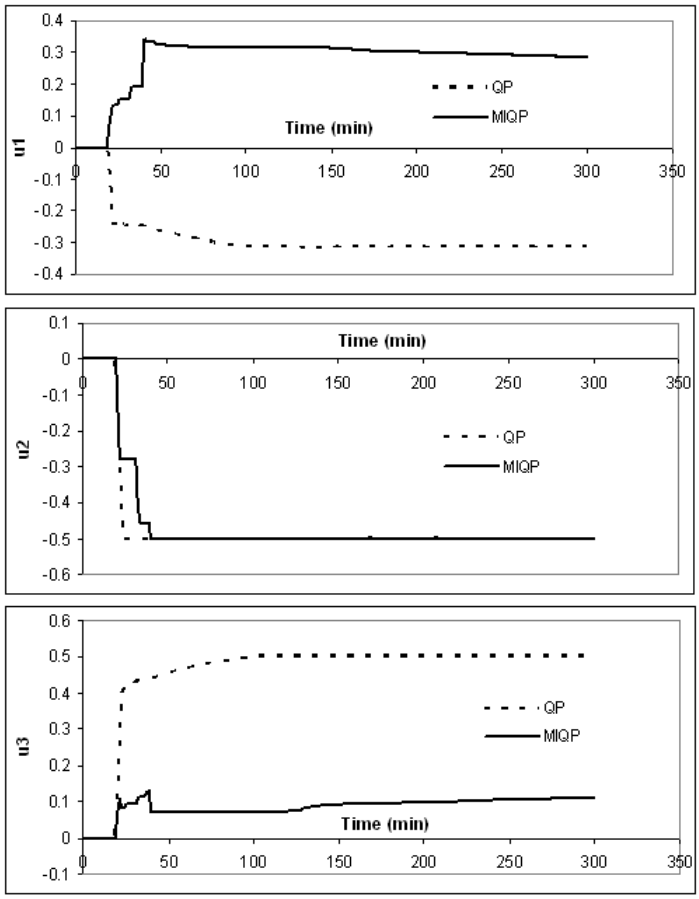

Figure 15. Comparative behavior of the manipulated inputs for the Shell benchmark system with a change in the setpoints of y_1 and y_2.

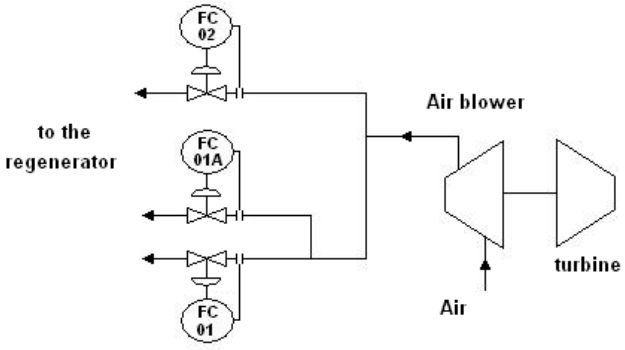

Figure 16. Regenerator air subsystem

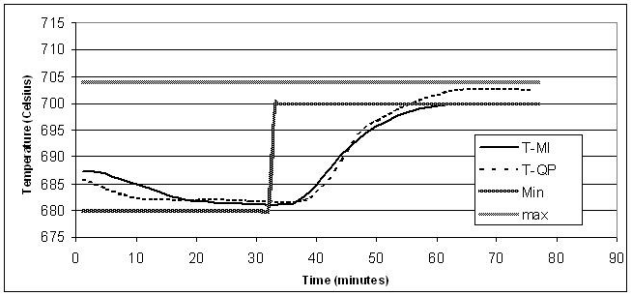

Figure 17. Regenerator temperature with the MIQP formulation (T-MI) and with the traditional QP (T-QP).

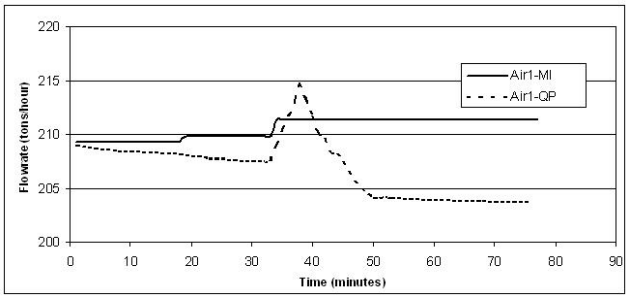

Figure 18. Main air flow to the Regenerator with the MIQP formulation (Air1-MI) and with the traditional QP (Air1-QP).

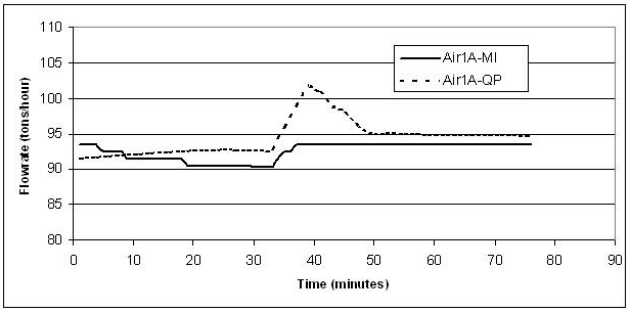

Figure 19. Secondary air flow to the Regenerator with the MIQP formulation (Air1A-MI) and with the traditional QP (Air1A-QP).

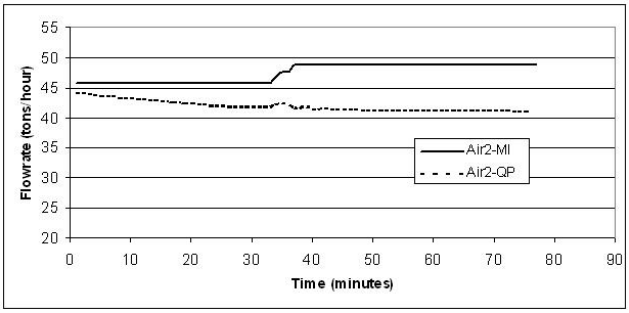

Figure 20. Air flow to the Regenerator second stage with the MIQP formulation (Air2A-MI) and with the traditional QP (Air2A-QP).

Table 1. Parameters used in the simulation of the 4-tank system

Symbol	Value	Unit	Description
h_1^u	1.36	m	Maximum level of tank 1
h_2^u	1.36	m	Maximum level of tank 2
h_3^u	1.30	m	Maximum level of tank 3
h_2^u	1.30	m	Maximum level of tank 4
h_i^l	0.20	m	Minimum level of tank $i \in \{1,2,3,4\}$
q_a^u	3.26	m^3/h	Maximum flow of q_a
q_b^u	4.00	m^3/h	Maximum flow of q_b
q_j^l	0.0	m^3/h	Minimum flow of $q_j, j \in \{a,b\}$
a_1	$1.31e^{-4}$	m^2	Discharge constant of tank 1
a_2	$1.51e^{-4}$	m^2	Discharge constant of tank 2
a_3	$9.27e^{-5}$	m^2	Discharge constant of tank 3
a_4	$8.82e^{-5}$	m^2	Discharge constant of tank 4

Table 2. Initial values for the inputs and outputs of the 4-tank system

Symbol	Value	Unit	Description
h_1^0	0.65	m	Initial value of the level of tank 1
h_2^0	0.66	m	Initial value of the level of tank 2
h_3^0	0.65	m	Initial value of the level of tank 3
h_4^0	0.66	m	Initial value of the level of tank 4
q_a^0	1.63	m^3/h	Initial value of q_a
q_b^0	2.00	m^3/h	Initial value of q_b

Table 3. Tuning parameters used in the 4-tank case

Symbol	Value in MIQP algorithm	Value in QP algorithm	Description
Λ_{qa}	2	1	Suppression factor of input q_a
Λ_{qb}	2	1	Suppression factor of input q_b
π_{qa}^{u}	1	-	Priority of input q_a
π_{qb}^{u}	0	-	Priority of input q_b
ϖ_{h1}	0.20	0.30	Weight of output h_1
ϖ_{h2}	0.20	0.30	Weight of output h_2
ϖ_{h3}	0.10	0.20	Weight of output h_3
ϖ_{h4}	0.10	0.20	Weight of output h_4
π_{h1}^{y}	5	-	Priority of output h_1
π_{h2}^{y}	3	-	Priority of output h_2
π_{h3}^{y}	1	-	Priority of output h_3
π_{h4}^{y}	1	-	Priority of output h_4
l	10	1	Control horizon
p	80	80	Prediction horizon
h	5	-	Nonzero future control actions

Highlights

- Development of mixed-integer quadratic formulation for model predictive controllers.
- The formulation allows the assignment of explicit priorities for the outputs and for the inputs.
- The formulation was successfully tested in simulated benchmarks and in an industrial case.
- The computational effort is not greatly increased when compared to the continuous MPC.

Possible Reviewers

1. Engell, S. (s.engell@bci.tu-dortmund.de)
2. De Prada, C. (prada@autom.uva.es)
3. Zeilinger, M (zeilinger@control.ee.ethz.ch)
4. Pistikopoulos, E. (e.pistikopoulos@imperial.ac.uk).
5. Stursberg, O. (stursberg@uni-kassel.de)
6. Christopher L.E. Swartz (swartzc@mcmaster.ca@mcmaster.ca)
7. Lee, Jay H. (jayhlee@kaist.ac.kr)

The equation is obtained by linearizing the quadratic objective function (eq. (37)) around the point $x^{NLP,j}$:

$$\varphi_{LIN} = \varphi(x^{NLP,j}) + \frac{\partial \varphi}{\partial x}(x^{NLP,j})(x - x^{NLP,j})$$

$$\varphi(x) = \frac{1}{2}x^T Cx + D^T x \text{ and } \frac{\partial \varphi(x)}{\partial x} = x^T C + D^T$$

$$\varphi_{LIN} = \frac{1}{2}x^{NLP,j^T} Cx^{NLP,j} + D^T x^{NLP,j} + (x^{NLP,j^T} C + D^T)(x - x^{NLP,j})$$

$$\varphi_{LIN} = \frac{1}{2}x^{NLP,j^T} Cx^{NLP,j} + D^T x^{NLP,j} + x^{NLP,j^T} Cx + D^T x - x^{NLP,j^T} Cx^{NLP,j} - D^T x^{NLP,j}$$

$$\varphi_{LIN} = x^{NLP,j^T} Cx + D^T x - \frac{1}{2}x^{NLP,j^T} Cx^{NLP,j}$$

$$\varphi_{LIN} = (x^{NLP,j^T} C + D^T)x - \frac{1}{2}x^{NLP,j^T} Cx^{NLP,j}$$