Proceeding of the 2004 American Control Conference
Boston, Massachusetts June 30 - July 2, 2004

Using Interpolation to Simplify Explicit Model Predictive Control

J.A. Rossiter
Dept. of Automatic Control & Systems Engineering
University of Sheffield
S1 3JD. UK
email: J.A.Rossiter@sheffield.ac.uk
Tel. 44 114 2225685

P. Grieder
Automatic Control Laboratory
ETH Zürich
Switzerland
email: grieder@control.ee.ethz.ch
Tel. 41 01 6327313

Abstract— **Multi parametric quadratic programming gives a full off-line solution to a time varying quadratic programming (QP) problem arising during constrained predictive control. However, coding and implementation of this solution may be more burdensome than solving the original QP. This paper presents a two degree of freedom algorithm, which achieves a large decrease in both the online computation and data storage requirements with negligible deterioration of performance. Extensive simulation results are given to back this claim.**

Keywords: Predictive control, computational efficiency, constraint handling, quadratic programming

I. Introduction

This paper assumes the acceptance and widespread use of predictive control (e.g. [3], [4], [9]) and asks: what are the obstacles to wider applications of model predictive control (MPC)? Two of many answers: (i) the computational demand and (ii) the lack of transparency of the control action due to the use of an online optimization. These form the underlying motivation for this paper.

Typical MPC algorithms [3] are based on the minimization of a quadratic performance index subject to linear constraints, that is a quadratic programming (QP) problem. In many cases, due to structural limitations [13], it is not possible to solve a QP online and hence the control law must have a simpler implementation. Moreover, the QP itself can require a large number of iterations for convergence and although this is rarely needed, it must be allowed for if the algorithm is to perform properly. Hence there are two obstacles to applications arising from the need for a QP: (i) coding limitations may make a QP optimizer inpractical and (ii) the upper limit on computational time for convergence may make implementation on fast systems impossible.

This paper considers a recent contribution [1] to this problem, denoted multi parametric quadratic programming (MPQP). The basic idea is to solve, off-line, all possible QP problems that can arise on line. It is straightforward to show that within certain regions of the state space the optimum predicted input trajectory has a known affine dependence on the current state. MPQP finds all possible active sets and the associated regions and control trajectories. The solution of the QP can then be replaced by set membership tests; if the state is inside region 'r', then one should use the associated control trajectory. Though the MPQP boasts several advantages, the number of computed regions may grow exponentially in the prediction horizon [1], making it unsuitable for large problems.

The aim of this paper is to consider how one can reduce the data storage requirements and also the implementation time for the MPQP algorithm by allowing for a small degree of sub-optimality. The proposed procedure is based on two interpolations (e.g. [12], [10]) of input sequences which provide feasibility and stability for the closed-loop system. We will henceforth refer to this procedure as *I*nterpolation MPQP (IMPQP) control.

II. Background

A. Modelling and MPC

For convenience of exposition this paper makes use of state-space models, e.g.

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k \tag{1}$$

where $\mathbf{x}, \mathbf{u}, \mathbf{y}$ are the state, input and output respectively. Associated to the model are constraints:

$$\underline{\mathbf{u}} \leq \mathbf{u}_k \leq \overline{\mathbf{u}}; \quad \underline{\mathbf{x}} \leq \mathbf{x}_k \leq \overline{\mathbf{x}} \tag{2}$$

The performance index [15] to be minimised, in this paper, takes the form

$$\begin{array}{l} \min_{\mathbf{u}_k,\ k=0,\ldots,n_c-1} J = \sum_{k=1}^{\infty} \mathbf{x}_k^T Q \mathbf{x}_k + \mathbf{u}_{k-1}^T R \mathbf{u}_{k-1} \\ \text{s.t.} \quad \begin{cases} (1,2)\ \forall k \geq 0 \\ \mathbf{u}_k = -K_{LQR}\mathbf{x}_k, \quad k \geq n_c \end{cases} \end{array} \tag{3}$$

where K_{LQR} is the optimal unconstrained feedback gain minimising J in the absence of constraints (2). It is noted that, in general, practical limitations imply that only a finite number, that is n_c, of free control moves can be used. For these cases, (3) is implemented by imposing that the state $\mathbf{x}_{n_c}$ must be contained in a polytopic control invariant set $\mathcal{X}_I = \{\mathbf{x}_0 \in \mathbb{R}^n | \underline{\mathbf{x}} \leq \mathbf{x}_k \leq \overline{\mathbf{x}},\ \underline{\mathbf{u}} \leq K_{LQR}\mathbf{x}_k \leq \overline{\mathbf{u}},\ \mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k,\ \forall k \geq 0\}$ in order to guarantee constraint satisfaction. In order to also guarantee asymptotic stability, it is necessary to impose a terminal cost $V(x_{n_c}) =$

0-7803-8335-4/04/$17.00 ©2004 AACC

$x'_{n_c} P x_{n_c}$, such that $V(x_{n_c})$ is a local Lyapunov function for the set $\mathcal{X}_I$ [9].

For convenience (e.g. [14]), the degrees of freedom can be reformulated in terms of a new variable $\mathbf{c}_k$

$$\begin{array}{rcll} \mathbf{u}_k & = & -K_{LQR}\mathbf{x}_k + \mathbf{c}_k; & k = 0, ..., n_c - 1 \\ \mathbf{u}_k & = & -K_{LQR}\mathbf{x}_k; & k \geq n_c \end{array} \quad (4)$$

and hence the equivalent optimisation to (3) can be written as:

$$\min_{\mathbf{C}} \quad J = \mathbf{C}^T S \mathbf{C} + x_0 F \mathbf{C}, \text{ s.t. } N\mathbf{C} + M\mathbf{x}_0 - \mathbf{v} \leq 0 \quad \mathbf{C} = [\mathbf{c}_0^T, \cdots, \mathbf{c}_{n_c-1}^T]^T \quad (5)$$

Details of how to compute positive definite matrix S, matrices N, M and vector $\mathbf{v}$ are omitted as by now well known in the literature (e.g. [9], [14]).

Definition 1: We denote with $\mathcal{X}_f^N \subseteq \mathbb{R}^n$ the set of initial states $\mathbf{x}_0$ for which the optimal control problem (5) is feasible and with $\mathcal{X}_I$ the maximum admissible set corresponding to $\mathbf{C} = 0$ [5]:

$$\begin{aligned} \mathcal{X}_f^N &= \{\mathbf{x}_0 \in \mathbb{R}^n | \exists \mathbf{C} \in \mathbb{R}^{n_c m}, \\ &\qquad N\mathbf{C} + M\mathbf{x}_0 - \mathbf{v} \leq 0\}, \\ \mathcal{X}_I &= \{\mathbf{x}_0 \in \mathbb{R}^n, \ M\mathbf{x}_0 - \mathbf{v} \leq 0\}. \end{aligned}$$

m denotes the number of inputs and n_c the prediction horizon in (3). The feasible set $\mathcal{X}_f^N$ can also be expressed as $\mathcal{X}_f^N = \{\mathbf{x} \in \mathbb{R}^n | M_{max}\mathbf{x} \leq \mathbf{d}_{max}\}$.

Remark 2.1: The MPC algorithm is given by solving the QP optimisation (5) at every sampling instant and then implementing the first component of $\mathbf{C}$, that is $\mathbf{c}_0$ in the control law of (4). When the unconstrained control law is not predicted to violate constraints, the optimising $\mathbf{C}$ is zero so the control is given as $\mathbf{u} = -K_{LQR}\mathbf{x}$; this is the case whenever $\mathbf{x} \in \mathcal{X}_I$.

B. Multi parametric quadratic programming

Here only a summary of the key conclusions in [1] is given. Where implicit, the subscript $(.)_k$ is omitted hereafter.

Define the regions $\mathcal{S}_i = \{\mathbf{x} : M_i\mathbf{x} - \mathbf{d}_i \leq 0\}$, $i = 0, 1, ...$ such that within each region the active set is the same and hence the $\mathbf{C}$ optimising (5) has a known affine dependence on $\mathbf{x}$, that is, $\mathbf{x} \in \mathcal{S}_i \quad \Rightarrow \quad \mathbf{C} = -\hat{K}_i\mathbf{x} + \mathbf{t}_i$. Hence the optimal control action to be used in (4) can be determined from

$$\mathbf{x} \in \mathcal{S}_i \quad \Rightarrow \quad \mathbf{c} = \mathbf{e}_1^T(-\hat{K}_i\mathbf{x} + \mathbf{t}_i) \quad \Rightarrow \quad \mathbf{u} = -K_i x + \mathbf{p}_i \quad K_i = K_{LQR} + \mathbf{e}_1^T\hat{K}_i, \quad \mathbf{p}_i = \mathbf{e}_1^T\mathbf{t}_i \quad (6)$$

where $\mathbf{e}_1^T = [I, 0, 0, ...]$, I an identity matrix with the input dimension.

III. The IMPQP Algorithm

In this section we will present a novel interpolation control scheme based on MPQP which exhibits significantly lower complexity in its on-line application at the cost of suboptimal closed loop performance. However, extensive simulation results in the next section will illustrate that the performance decrease incurred by IMPQP control is not significant, in general.

IMPQP makes uses of two interpolations; we address each of these separately in the following subsections before concluding this section with an overview of the properties of IMPQP control.

A. IMPQP: Interpolation 1

Rather than using the individual control values $\mathbf{u}_k$, $k = 0, 1, ...$ in the optimisation of performance, the first interpolation of the IMPQP controller uses a mixture of input trajectories arising from two possible control laws. The major difference to earlier interpolation approaches (e.g. [12]) is the selection of the input trajectories; here we make use of some of the MPQP regions $\mathcal{S}_i$ to define one of the trajectories and hence this trajectory is guaranteed feasible for $\mathbf{x} \in \mathcal{X}_f^N$. More precisely the optimal unconstrained input sequence is interpolated with the input sequence which would be used if a given state was projected onto the nearest facet. The following lemma can be used to establish the nearest facet to a given state $\mathbf{x}$.

Lemma 3.1: (i) Define the polytopes P_j to be the minimal volume polytopes containing both the jth facet of $\mathcal{X}_f^N$ and the origin. (ii) Normalise the inequalities defining $\mathcal{X}_f^N$ according to

$$M_{max}\mathbf{x} - \mathbf{d}_{max} \leq 0; \ \mathbf{d}_{max}^T = \begin{bmatrix} 1 \\ 1 \\ \vdots \end{bmatrix}; \ M_{max} = \begin{bmatrix} \mathbf{m}_1^T \\ \mathbf{m}_2^T \\ \vdots \end{bmatrix} \quad (7)$$

(iii) Compute the values $\gamma_j = \mathbf{m}_j^T\mathbf{x}$ and compute j for which γ_j is a maximum. Then $\mathbf{x}$ lies in polytope P_j. The proof is obvious.

Theorem 3.1: Given $\mathbf{x} \in P_j$ and $\gamma_j = \mathbf{m}_j^T\mathbf{x}$ find the region $\mathcal{S}_i$ such that $\mathbf{x}/\gamma_j \in \mathcal{S}_i$. Then the control move

$$\mathbf{u} = -K_{LQR}\mathbf{x} + \mathbf{e}_1^T\mathbf{C}; \quad \mathbf{C} = -\hat{K}_i\mathbf{x} + \gamma_j\mathbf{t_i} \quad (8)$$

is the first move of a predicted control law with convergent predictions. Moreover, this control move is optimal if $\mathbf{x}$ lies on a facet of $\mathcal{X}_f^N$.

Proof: Given (7), the value γ_j is the normalised distance from the origin to state $\mathbf{x}$ (points on the facet giving $\gamma_j = 1$). Hence control law (8) is a scaled version of what would be used if $\mathbf{x}$ were on the facet which by linearity must therefore give rise to feasible predictions which remain inside $\mathcal{X}_f^N$ (or in fact a region of identical shape to $\mathcal{X}_f^N$ but scaled by γ_j). □

Having established some of the solution properties, we will now introduce Interpolation 1 of the proposed IMPQP

control procedure.

Algorithm 3.1: IMPQP Interpolation 1:

1) Off-Line: Solve (5) as an MPQP and remove all regions which are not located on the boundary (see Figure 1).
2) On-Line: For an initial state $\mathbf{x}$, compute $\gamma_j = \mathbf{m}_j^T \mathbf{x}, \quad j = 1, 2, ...$ and find j such that γ_j is a maximum.
3) On-Line: From the regions on the jth facet, find i such that $\mathbf{x}/\gamma_j \in \mathcal{S}_i$ and compute the corresponding sequence $\mathbf{C}$ from (8).
4) On-Line: The optimal $\mathbf{C}$ for the unconstrained LQR controller is $\mathbf{C} = 0$, so one will be interpolating between $\mathbf{C}$ and zero.
5) On-Line: Perform the minimisation

$$\min_{\alpha} \quad \alpha \quad \text{s.t.} \quad \begin{cases} N(\alpha \mathbf{C}) - (\mathbf{d} - M\mathbf{x}) \leq 0 \\ 0 \leq \alpha \leq 1 \end{cases} ; \quad (9)$$

to obtain $\mathbf{C}_1 = \alpha \mathbf{C}$.

The resulting closed-loop sequence is feasible and has the property of recursive feasibility[1] by design. However, convergence to $\mathcal{X}_I$ and hence to the origin cannot be guaranteed in general [11]. We therefore add a second interpolation [10] in the next section which will guarantee the closed loop trajectory to be convergent and stabilizing.

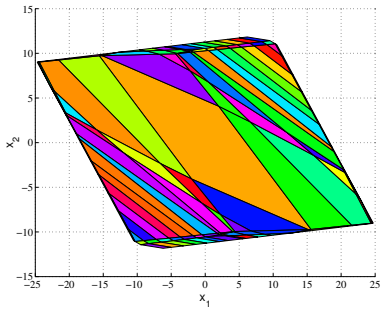
B. IMPQP: Interpolation 2

The key reason why stability is not automatic in Level 1 of IMPQP control is the lack of predictability in the input sequence. The classic approach of 'shifting' the input sequence of the previous time step in order to prove stability [9] does not apply here. That is, it is not possible in general to make the choice for $\mathbf{C}_{k+1} = [\mathbf{c}_{k+1|k}^T, \mathbf{c}_{k+2|k}^T, \cdots, \mathbf{c}_{k+n_c-1|k}, 0]^T$. Without the tail in the class of possible predictions, one cannot easily argue that the cost J is monotonically decreasing and in fact one can easily find Level 1 simulations with excellent performance (and convergence) but where, nevertheless, J does not change monotonically. The result of the following Lemma is well known in the literature (e.g. [9]).

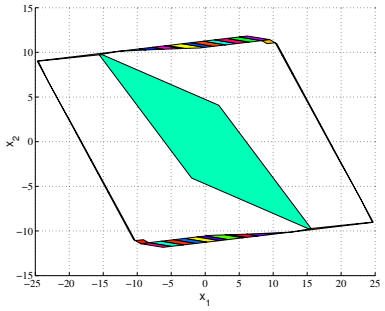
Lemma 3.2: If the input sequence $\mathbf{C}_{k+1} = [\mathbf{c}_{k+1|k}, \mathbf{c}_{k+2|k}, \cdots, 0]^T$ is in the class of possible predictions at time $k+1$ for problem (3), this is sufficient to guarantee stability in the nominal case.

However, interpolation 1 of the IMPQP scheme may not include the shifted input sequence. The proposal here is to add a second degree of freedom (e.g. [10]) which corresponds to an interpolation of the input sequence $\mathbf{C}_1$ of algorithm 3.1 and the 'tail' of the input sequence which was obtained at the previous time step. This will automatically provide for a proof of stability for the IMPQP scheme and may also serve to improve closed-loop performance. Next, the on-line interpolation 2 of the IMPQP controller is defined.

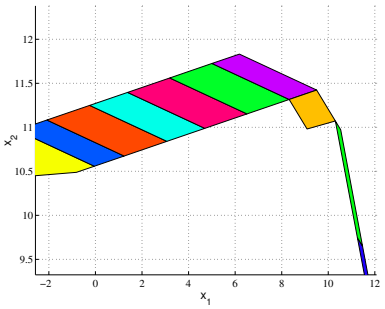
[1] Satisfaction of (9) implies the existence at the next sampling instant of a valid $\mathbf{C}$ such that $\mathbf{x} \in \mathcal{X}_f^N$.



(a) MPQP Controller.



(b) IMPQP Controller.



(c) Closeup of IMPQP Controller.

Fig. 1. Controller partition for a standard controller compared to the IMPQP controller.

Algorithm 3.2: IMPQP Interpolation 2:

1) At time $k-1$, store the input sequence obtained with Level 1 according to

$$\mathbf{C}_{tail} = [\mathbf{c}_{k|k-1}^T, \mathbf{c}_{k+1|k-1}^T, \cdots, \mathbf{c}_{k+n_c-2|k-1}^T, 0]^T \quad (10)$$

2) At time k, define a linear interpolation between the optimal $\mathbf{C}_1$ obtained by Level 1 and the tail $\mathbf{C}_{tail}$

$$\mathbf{C}_{mix} = (1-\beta)\mathbf{C}_1 + \beta \mathbf{C}_{tail}; \quad 0 \leq \beta \leq 1 \quad (11)$$

3) Minimise the predicted cost over the prediction class

in (11).

$$\begin{array}{ll} \min_\beta & J = \mathbf{C}_{mix}^T S \mathbf{C}_{mix} = \beta^2 f + 2\beta g + h \\ \text{s.t.} & \begin{cases} 0 \leq \beta \leq 1 \\ f = [\mathbf{C}_{tail} - \mathbf{C}_1]^T S [\mathbf{C}_{tail} - \mathbf{C}_1] \\ g = [\mathbf{C}_{tail} - \mathbf{C}_1]^T S \mathbf{C}_1 \end{cases} \end{array} \tag{12}$$

4) Implement the control law $\mathbf{u} = -K_{LQR}\mathbf{x}_k + \mathbf{e}_1^T \mathbf{C}_{mix}$.

β will be zero unless the solution obtained with Level 1 can be improved upon by moving towards $\mathbf{C}_{tail}$.

C. Complexity and Properties of IMPQP Control

Some properties of IMPQP control are stated next.

Theorem 3.2: Algorithm IMPQP has a guarantee of both recursive feasibility and stability in the nominal case.

Proof: By construction both $\mathbf{C}_1$ and $\mathbf{C}_{tail}$ are feasible and therefore, from convexity arguments, $\mathbf{C}_{mix}$ must also be feasible. Also, because $\mathbf{C}_{tail}$ is in the class of possible predictions, Lemma 3.2 must apply, and hence comes the guarantee of stability. □

The computational burden of IMPQP control is significantly smaller than comparable algorithms [1] and MPQP.

- It is only necessary to store regions which lie on a facet of $\mathcal{X}_f^N$, thus reducing the storage effort.
- A simple lookup table which associates facets to regions can be created which reduces the number of set-membership tests significantly.
- The additional on-line computations in (9) and (12) are also negligible since the implied minimizations are over scalars and thus are trivial.

It should be noted however, that the necessary off-line computation effort for IMPQP is not negligible. In the worst case it is necessary to solve $f_{\mathcal{X}_f^N} \cdot \sum_{r=1}^{R} f_r$ LPs where $f_{\mathcal{X}_f^N}$ and f_r denote the number of facets of the set $\mathcal{X}_f^N$ and region P_r, respectively. This requirement places a practical limit on the size of the problems IMPQP is applicable to. However, this also holds for other schemes which aim at simplifying the feedback solution through post-processing of the controller partitions.

IV. Numerical Examples

This section presents an extensive comparison of the IMPQP algorithm with traditional MPQP controllers. The infinite horizon controller presented in [6] (contained in the MPT toolbox [8]) is used as a basis for comparison. Both controllers cover the maximum control invariant set $\mathcal{X}_f^N$ and provide stability and feasibility properties. Hence it is necessary only to compare complexity and performance.

The comparison is based on 40 random systems with $2-3$ states and 2 inputs. The inputs for all systems were constrained to $-1 \leq u_{1,2} \leq 1$ and the states were limited to $-10 \leq x_i \leq 10$ $(i = 1, 2, 3)$.

Two different variations on the performance objective of (3) were considered: that is the cases of small and large weights on the input, i.e., $R_1 = 0.1I$ and $R_2 = 10I$. $Q = I$ was used throughout. For consistency with other work, the cases considered are identical to those presented in [7]; a comparison with the complexity reduction obtained in [7] is also discussed briefly at the end of the section.

A. Complexity comparisons

Figure 2 gives a comparison of the complexity of IMPQP versus the controller in [6]. The dotted lines/dashed lines display the number of regions which need to be stored for [6] and IMPQP respectively and the dash-dot lines display the maximum number of regions on any facet. The on-line effort for the set-membership test in [6] is proportional to the total number of regions whereas the IMPQP only needs to check the regions associated to the facet identified in step 3 of Algorithm 3.1.

- The average reduction in storage requirements for the IMPQP algorithm is 55%.
- The average reduction in on-line computation associated to set membership identification for the IMPQP algorithm is 89% (reduction by a factor of 10) .

B. Performance comparisons

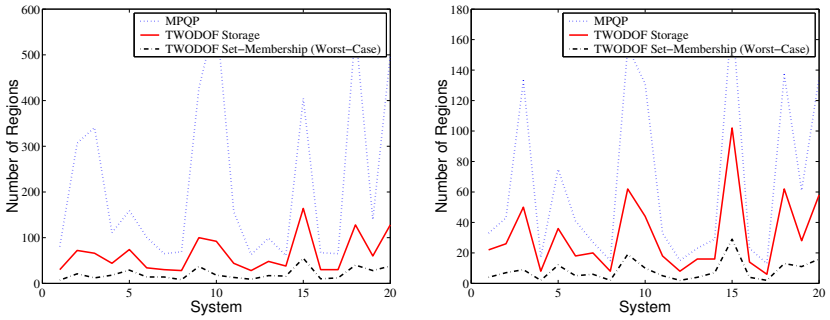
Figure 3 gives a comparison of the performance (calculating J of (3) for the closed-loop trajectories) of IMPQP versus the controller in [6]. Performance was evaluated by gridding the feasible state space and summing up all the closed loop trajectory cost. The average performance decrease over all runs is merely 2%.

C. Summary of comparison

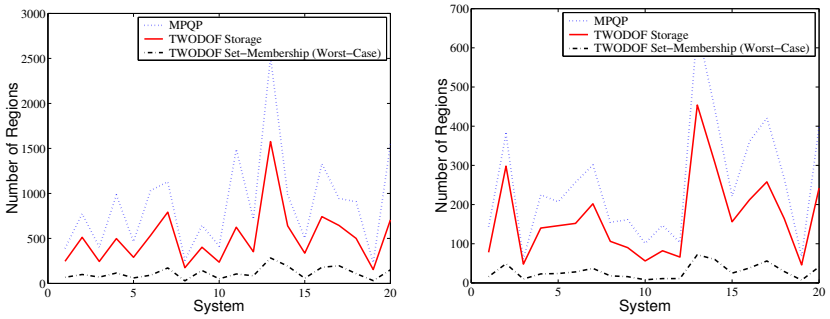
As can be gathered from Figures 2 and 3, IMPQP control exhibits a significant decrease in complexity in storage and set-membership test at virtually no cost in terms of performance. The authors in [7] obtained a decrease in complexity in storage and identification of 69% at the cost of a performance reduction of 0.01%, but their runtime results clearly indicated that the procedure is not applicable for medium size systems. Hence, IMPQP is the superior alternative if run-time and not storage space is the limiting factor in application. On the other hand, similar to other simplification techniques [16], [2], the IMPQP procedure relies on the a priori computation of the explicit control law, which may be prohibitive for large problems. Note that the procedures in [16], [2] may be used in combination with IMPQP to obtain further reductions in complexity.

V. Conclusion

A novel interpolation based control scheme was presented which allows for significant simplification of the on-line set-membership test necessary for MPQP control [1]. The procedure is based on two interpolations, the first of which aims at optimality while the second guarantees stability. In extensive simulations it was shown that the procedure

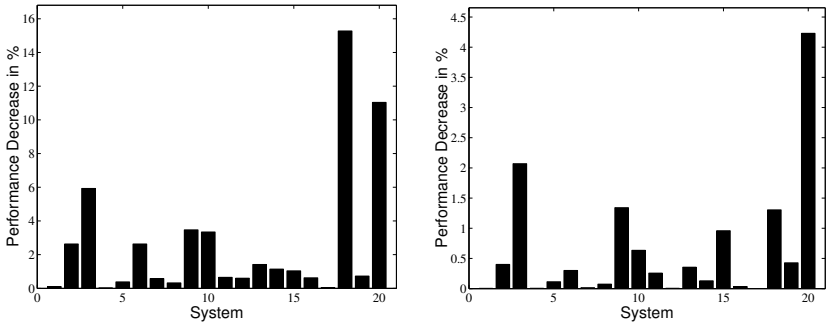


States $n = 2$, cost $R = 0.1I, Q = I$. States $n = 2$, cost $R = 10I, Q = I$.

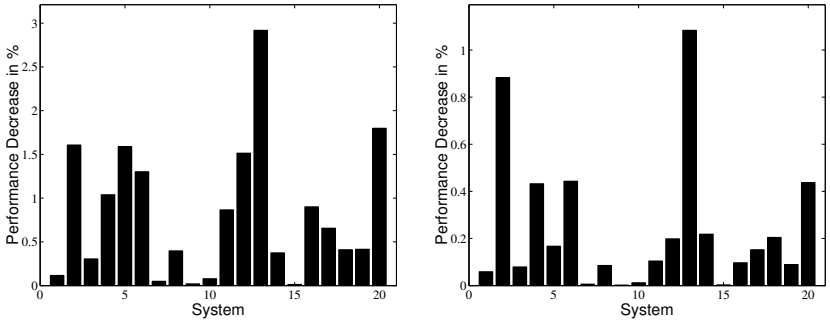


States $n = 3$, cost $R = 0.1I, Q = I$. States $n = 3$, cost $R = 10I, Q = I$.

Fig. 2. Comparison of IMPQP complexity versus standard MPQP [1] for 20 second-order systems.



States $n = 2$, cost $R = 0.1I, Q = I$. States $n = 2$, cost $R = 10I, Q = I$.



States $n = 3$, cost $R = 0.1I, Q = I$. States $n = 3$, cost $R = 10I, Q = I$.

Fig. 3. Comparison of IMPQP performance versus standard MPQP [1] for 20 second-order systems.

can reduce the necessary on-line effort for set-membership identification by a factor of 10 at the cost of minor performance degradation, making it an attractive option for fast processes. The necessary additional on-line optimization of a scalar is trivially implemented and does not require a significant amount of computation time. Future work will focus on the possibility of storing only one input per facet which will simplify storage and online computation even more.

REFERENCES

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

[2] F. Borrelli, M. Baotic, A. Bemporad, and M. Morari. Efficient on-line computation of constrained optimal control. In *Proc. 40th IEEE Conf. on Decision and Control*, December 2001.

[3] D.W. Clarke, C. Mohtadi, and P.S.Tuffs. Generalised predictive control, parts 1 and 2. *Automatica*, 23:137–160, 1987.

[4] C.R. Cutler and B.C. Ramaker. Dynamic matrix control—a computer control algorithm. *Proc. American Contr. Conf.*, vol. WP5-B, San Francisco, USA, 1980.

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

[6] P. Grieder, F. Borrelli, F.D. Torrisi, and M. Morari. Computation of the constrained infinite time linear quadratic regulator. In *Proc. 2003 American Contr. Conf.*, Denver, Colorado, USA, 2003.

[7] P. Grieder, P. Parillo, and M. Morari. Stability & feasibility of receding horizon control. In *European Control Conference*, Cambridge, UK, September 2003.

[8] M. Kvasnica, P. Grieder, M. Baotić, and M. Morari. Multi Parametric Toolbox (MPT). `http://control.ee.ethz.ch/~mpt`. In Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, Philadelphia, USA, March 2003.

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

[10] J.A. Mendez, B. Kouvaritakis, and J.A. Rossiter. State space approach to interpolation in mpc. *Int. journal of robust nonlinear control*, 10:27–38, 2000.

[11] J.A. Rossiter, B.Kouvaritakis, and M. Cannon. Stability proof for computationally efficient predictive control in the uncertain case. In *American Contr. Conf.*, Denver, Colorado, USA, June 2003.

[12] J.A. Rossiter, B. Kouvaritakis, and M. Cannon. Computationally efficient algorithms for constraint handling with guaranteed stability and near optimality. *Int. J. Control*, 74(17):1678–1689, 2001.

[13] J.A. Rossiter, P.W. Neal, and L. Yao. Applying predictive control to fossil fired power station. *Trans. Inst. Measurement and Control*, 24:177–194, 2002.

[14] J.A. Rossiter, M.J. Rice, and B. Kouvaritakis. A numerically robust state-space approach to stable predictive control strategies. *Automatica*, 38(1):65–73, 1998.

[15] P.O.M. Scokaert and J.B. Rawlings. Constrained linear quadratic regulation. *IEEE Trans. Automatic Control*, 43(8):1163–1169, August 1998.

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