

promoting access to White Rose research papers



Universities of Leeds, Sheffield and York
<http://eprints.whiterose.ac.uk/>

This is an author produced version of a paper published in **Control Engineering Practice**.

White Rose Research Online URL for this paper:
<http://eprints.whiterose.ac.uk/43194>

Published paper

Valencia-Palomo, G., Rossiter, J.A. (2011) *Efficient suboptimal parametric solutions to predictive control for PLC applications*, Control Engineering Practice, 19 (7), pp. 732-743
<http://dx.doi.org/10.1016/j.conengprac.2011.04.001>

Efficient suboptimal parametric solutions to predictive control for PLC applications

G. Valencia-Palomo^{a,*}, J.A. Rossiter^{a,1}

^a*Department of Automatic Control and Systems Engineering,
University of Sheffield, South Yorkshire, U.K. S1 3JD.*

Abstract

The prime aim of this paper is to embed a predictive control (MPC) algorithm with constraint handling capabilities into a Programmable Logic Controller (PLC). In order to achieve it, this paper develops parametric approaches to MPC but differs from more conventional approaches in that it pre-defines the complexity of the solution rather than the allowable suboptimality. The paper proposes a novel parameterisation of the parametric regions which allows efficiency of definition, effective spanning of feasible region and also highly efficient search algorithms. Despite the suboptimality, the algorithm retains guaranteed stability, in the nominal case. A laboratory test was carried out to demonstrate the code on real hardware and the effectiveness of the solution.

Keywords: Predictive control, multi-parametric quadratic programming, PLC.

1. Introduction

Parametric solutions to predictive control (MPC) (Bemporad et al., 2002a,b; Pistikopoulos et al., 2002) have the key advantages of (i) giving transparency to the control law which may have advantages in safety critical or highly regulated environments and (ii) having the potential to significantly reduce the on line computational load/complexity. This paper concerns itself primarily with the second of these points because often the potential to reduce complexity is not realised; for instance, especially with high order systems, the optimal parametric solution may be very complex so that implementation is more difficult or slower than including an online quadratic programme (QP). The aim is to achieve a reduction of complexity and data storage to the point the controller can be used in a Programmable Logic Controller (PLC) as this hardware represents the standard computer in industry.

Several authors have tried to tackle this issue with various research directions. Some authors have looked at optimising the efficiency of storage of the parametric solution combined with fast algorithms for implementation of the solution (e.g. Borrelli et al., 2001; Tondel et al., 2003; Christophersen et al., 2007). Other authors have considered beginning from a suboptimal parametric solution in the hope that such a solution may be far simpler, but with a small loss in performance only (e.g. Johansen, 2003; Johansen et al., 2002; Johansen and Grancharova, 2003;

*Corresponding author.

Email address: g.valencia-palomo@sheffield.ac.uk (G. Valencia-Palomo)

¹Tel.: +44 (0)114 222 5685; Fax.: +44 (0)114 222 5661.

18 Grieder et al., 2004; Bemporad and Filippi, 2003, 2006). One approach in the literature has looked
19 at using orthogonal spaces to speed up search times and sub-divides the parametric space into
20 small enough regions to quantify the sub-optimality as small enough, either from a performance
21 perspective or in terms of feasibility (Johansen, 2003). Another alternative is to look at making the
22 regions larger (Bemporad and Filippi, 2003) by allowing some relaxation of the optimality, while
23 ensuring feasibility. However, critically for the motivation here, none of the works above are able
24 to give any strict bounds on the resulting complexity of the solution which thus may still be worse
25 than desired.

26 A less explored avenue is to base the parametric solution on points rather than regions (Canale
27 et al., 2009, 2010a,b). In simple terms one predefines the optimum strategy for a number of
28 possible initial conditions and then online select from these the one which is closest to the actual
29 initial condition. However, once again the main failing of this approach is that it is hard to get
30 bounds on the complexity of the solution because the focus of the work is on ensuring that the
31 suboptimality meets some guaranteed requirement which thus can be conservative. The examples
32 in those papers use numbers such as $10^4 - 10^6$ vertices. An even less explored avenue is the potential
33 to use interpolation (Rossiter and Grieder, 2005) to give a convex blend from nearby points, thus
34 reducing the number of points/regions required while ensuring feasibility.

35 Here the intention is to take a different viewpoint from those which either start from the explicit
36 optimal and define efficient searches or look for ways of trading complexity with performance.
37 Instead this paper proposes to predefine the complexity of the solution and then ask whether one
38 is able to get sufficient performance and guarantees of feasibility. The argument taken is that
39 any result which is based on sub-division until the difference from the optimal is small will, in
40 general, lead to a large number of regions. In general, a convergence and feasibility guarantee
41 with a pre-defined complexity are wanted; and then to ask what level of performance it can get
42 from that, accepting that it will be suboptimal in comparison to a more complex solution. The
43 advantage of predefining the complexity is the possibility, a priori, of giving strict limits on data
44 storage and sample time requirements and thus being much cheaper and simpler to implement on
45 systems which have available only low computational power like a PLC but may still require fast
46 sample times (Valencia-Palomo and Rossiter, 2010).

47 Hence the key contribution of this paper is a proposed approach which reduces complexity of
48 the parametric solution and does not give large growth in complexity with the state dimension.
49 Moreover, the online coding requirements are trivial as compared to the online implementation of
50 a QP solver. In line with some concepts adopted by earlier authors (Johansen and Grancharova,
51 2003; Grieder et al., 2004; Bemporad and Filippi, 2006) this paper intends to make use of regular
52 shapes as this enables very efficient search methods and simple polytope definitions with predefined
53 complexity.

54 The paper is organised as follows: Section 2 will give a brief background on a standard MPC
55 algorithm and explicit solutions; Section 3 discusses about complexity of polytope representations;
56 Section 4 then introduces the proposed suboptimal parametric solution with proofs of feasibility
57 and convergence; Section 5 presents some Monte-Carlo numerical illustrations; Section 6 presents
58 the hardware overview and implementation of the algorithm; Section 7 presents an experimental
59 example; the paper finishes with the conclusions in Section 8.

2. Background on predictive control 60

2.1. Model and constraints 61

This paper assumes a standard state space model of the form 62

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k; \quad \mathbf{y}_k = \mathbf{C}\mathbf{x}_k; \quad (1)$$

where $\mathbf{x}_k \in \mathbb{R}^{n_x}$, $\mathbf{u}_k \in \mathbb{R}^{n_u}$, $\mathbf{y}_k \in \mathbb{R}^{n_y}$ are the states, inputs and outputs at sample k respectively. 63
It is assumed that these are subject to polytopic constraints at every sample instant, for example: 64

$$\mathbf{A}_u \mathbf{u}_k \leq \mathbf{b}_u; \quad \mathbf{A}_{\Delta u} \Delta \mathbf{u}_k \leq \mathbf{b}_{\Delta u}; \quad \mathbf{A}_y \mathbf{y}_k \leq \mathbf{b}_y; \quad (2)$$

where $\Delta \mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_{k-1}$. 65

In the context of predictive control, it is common to take the following quadratic performance index as the objective to be minimised at each sample 66
67

$$J = \sum_{i=1}^{\infty} \{ \mathbf{x}_{k+i}^T \mathbf{Q} \mathbf{x}_{k+i} + \mathbf{u}_{k+i-1}^T \mathbf{R} \mathbf{u}_{k+i-1} \}, \quad (3)$$

with $\mathbf{Q} \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{R} \in \mathbb{R}^{n_u \times n_u}$ positive definite state and input cost weighting matrices; as, under some mild conditions, this allows a straightforward stability guarantee in the nominal case. In the unconstrained case, the control law is given as $\mathbf{u}_k = -\mathbf{K}\mathbf{x}_k$, where the *optimal* feedback gain \mathbf{K} is obtained via the solution of the corresponding Linear Quadratic Regulator (LQR) problem. However, it is noted here that this paper omits the fine details associated to integral action and offset free tracking to simplify the presentation. 68
69
70
71
72
73

2.2. Optimal MPC (OMPC) 74

The key idea in Sokaert and Rawlings (1998); Rossiter et al. (1998), is to embed into the predictions the unconstrained optimal behaviour and handle constraints using perturbations about this. Assuming that \mathbf{K} is the *optimal* feedback, the input predictions are defined as follows: 75
76
77

$$\mathbf{u}_{k+i} = \begin{cases} -\mathbf{K}\mathbf{x}_{k+i} + \mathbf{c}_{k+i} & i = 0, \dots, n_c - 1 \\ -\mathbf{K}\mathbf{x}_{k+i} & i \geq n_c, \end{cases} \quad (4)$$

where n_c is the prediction horizon for the perturbations \mathbf{c}_k . It is known that, the input predictions (4) and associated state predictions for model (1) satisfy constraints (2) if 78
79

$$\mathbf{M}\mathbf{x}_k + \mathbf{N} \underline{\mathbf{c}}_k \leq \mathbf{1}; \quad \underline{\mathbf{c}}_k = [\mathbf{c}_k^T, \mathbf{c}_{k+1}^T, \dots, \mathbf{c}_{k+n_c-1}^T]^T; \quad (5)$$

for suitable \mathbf{M} , \mathbf{N} with a finite number of rows (Gilbert and Tan, 1991). Details of how to compute these matrices are omitted as by now well known in the literature (Gilbert and Tan, 1991; Sokaert and Rawlings, 1998; Rossiter, 2003). 80
81
82

Definition 2.1 (MAS). Define the Maximum Admissible Set (MAS) as the region in the state space for which the unconstrained feedback $\mathbf{u}_k = -\mathbf{K}\mathbf{x}_k$ satisfies constraints, i.e. $\{\mathbf{x}_k \in \mathbb{R}^{n_x} \mid \mathbf{M}\mathbf{x}_k \leq \mathbf{1}\}$ 83
84
85

86 **Definition 2.2 (MCAS).** Define the Maximum Controllable Admissible Set (MCAS) as the re-
 87 gion in the state space whereby it is possible to find a $\underline{\mathbf{c}}_k$ such that the future trajectories satisfy
 88 constraints; i.e. $\{\mathbf{x}_k \in \mathbb{R}^{n_x} | \exists \underline{\mathbf{c}}_k \in \mathbb{R}^{n_c n_u} \text{ s.t. } \mathbf{M}\mathbf{x}_k + \mathbf{N}\underline{\mathbf{c}}_k \leq \mathbf{1}\}$

89 The MCAS gets larger as n_c increases, but usually up to a finite limit if the state constraints
 90 give a closed region (Gilbert and Tan, 1991). It is not the purpose of this paper to consider nuances
 91 in that discussion topic. n_c is taken to be large here and in fact it is known from work in parametric
 92 solutions that often a relatively small finite value is enough to capture the maximum MCAS.

93 In practice, optimal predictions may violate constraints (2), so prediction class (4) is used
 94 instead. It is easy to show (Scokaert and Rawlings, 1998; Rossiter et al., 1998) that the optimisation
 95 of J over input predictions (4) is equivalent to minimising $\underline{\mathbf{c}}_k^T \mathbf{S} \underline{\mathbf{c}}_k$ ($\mathbf{S} = \mathbf{B}^T \mathbf{W} \mathbf{B} + \mathbf{R}$, $\mathbf{W} - (\mathbf{A} -$
 96 $\mathbf{B}\mathbf{K})^T \mathbf{W} (\mathbf{A} - \mathbf{B}\mathbf{K}) = \mathbf{Q} + \mathbf{K}\mathbf{R}\mathbf{K}$) and thus, in the absence of constraints, the optimum is $\underline{\mathbf{c}}_k = 0$.
 97 Where the unconstrained predictions would violate constraints and non-zero $\underline{\mathbf{c}}_k$ would be required
 98 to ensure constraints are satisfied.

99 **Algorithm 2.1 (OMPC).** The OMPC algorithm is

$$\underline{\mathbf{c}}_k^* = \arg \min_{\underline{\mathbf{c}}_k} \underline{\mathbf{c}}_k^T \mathbf{S} \underline{\mathbf{c}}_k \text{ s.t.} \quad (6)$$

100 Use the first element of $\underline{\mathbf{c}}_k^*$ in the control law of (4), with \mathbf{K} . This algorithm will find the global
 101 optimal, with respect to (3), whenever that is feasible and has guaranteed convergence/recursive
 102 feasibility in the nominal case.

103 **Definition 2.3.** For initial states $\mathbf{x}_k = \mathbf{v}_j$, define the corresponding optimal control sequences as
 104 $\underline{\mathbf{c}}_k = \underline{\mathbf{c}}_{j,k}$. By definition therefore the recursive use of the this sequence of \mathbf{c}_{k+i} values in (4) will
 105 give input/state trajectories that satisfy constraints and converge to the origin.

106 Definition 2.3 explicitly associates the optimal trajectory $\underline{\mathbf{c}}_{j,k}$ with the initial condition \mathbf{v}_j this
 107 will be useful in the next sections where the vertices (\mathbf{v}_j 's) of the feasible regions are important
 108 elements of the proposed algorithm.

109 2.3. Parametric solutions

110 The solution of problem (6) has a parametric solution (mp-QP) (Bemporad et al., 2002b) of
 111 the form

$$\mathbf{x}_k \in \mathcal{R}_r \Rightarrow \underline{\mathbf{c}}_k^* = -\mathbf{K}_r \mathbf{x}_k + \mathbf{t}_r; \quad \mathcal{R}_r = \{\mathbf{x}_k : \mathbf{M}_r \mathbf{x}_k \leq \mathbf{d}_r\}; \quad (7)$$

112 for suitable $\mathbf{K}_r, \mathbf{t}_r, \mathbf{M}_r, \mathbf{d}_r$ where the interiors of the polytopes \mathcal{R}_r do not overlap and the union
 113 gives the MCAS. The main weakness of parameteric solutions is that the number of regions r
 114 can grow very quickly both with state dimension and indeed n_c . This paper seeks alternative but
 115 suboptimal parametric solutions which require far fewer regions.

116 3. Facet or vertex representations of polytopes and parametric solutions

117 The main assumption of this paper is that an efficient parametric realisation requires an as-
 118 sumption of regular polytopes, such as n_x -dimensional cubes, as opposed to the more general
 119 shapes possible for \mathcal{R}_r . This is because such an assumption carries several simple benefits: (i) the

number of facets may be small; (ii) the number of vertices may be small and equispaced to some extent; (iii) the shape, facets and vertices are regular and hence easy to handle and define. This allows for very efficient search algorithms. In fact, the assumption in this paper is slightly different from a cube, although the proposed algorithm is seeded by a cube.

This section will first present a discussion about the complexity of facet and vertex based representations of polytopes (Section 3.1 and 3.2). It then goes to present convexity arguments for MPC (Section 3.3), the definition of the particular shape used (Section 3.4) and the algorithm to solve the point location problem (Section 3.5). The section finalises with a brief summary of the results (Section 3.6).

3.1. Facet based parametric solutions

A major concern is related to the efficiency of search algorithms. It is recognised (Borrelli et al., 2001; Rossiter and Grieder, 2005) that one can define very efficient algorithms for finding an *active facet* [Definition 3.1] and thus an MPC algorithm which utilises this mechanism (there is an explicit link between the active facet and the control law) has the potential to be very efficient; it has also been shown that efficient mp-QP parameterisations, can use active facet computations to infer the set membership computations in (7).

Definition 3.1. Consider a closed polytope, containing the origin, given as $\mathcal{R} = \{\mathbf{x}_k : \mathbf{R}\mathbf{x}_k \leq 1\}$, $\mathbf{R} \in \mathbb{R}^{q \times n_x}$ with q (non-redundant) inequalities and facets $\{\mathcal{R}_1^f, \dots, \mathcal{R}_q^f\}$. Then for any state \mathbf{x}_k , the active facet is \mathcal{R}_a^f with

$$a = \arg \min_{j \in \{1, \dots, q\}} \mathbf{e}_j^T \mathbf{R}\mathbf{x}_k, \quad (8)$$

where \mathbf{e}_j is the j th standard basis vector.

Thus, the search algorithm efficiency depends only on q , the number of inequalities defining \mathcal{R} . However, this is where the goal of a highly efficient search algorithm may break down because there may be many different optimal solutions contributing to a single facet; moreover there may be too many inequalities. Thus this paper considers to what extent a suboptimal solution which enforces a single solution on each facet and/or keeps the number of facets small is amenable to guarantees of feasibility and convergence.

3.2. Facet vs vertex representations of polytopes

A key issue for the user is to ask which representation is more efficient, one based on the vertices or one based on facets. For an arbitrary region such as the MAS one cannot give a simple answer to this. It is difficult to form a systematic link between the number of vertices and facets except for a few cases such as the n_x -dimensional cube. For a general n_x -dimensional polytope, it is possible that some facets have far more than n_x vertices and equally some vertices contribute to many more than n_x facets.

In summary, if one wants efficient or consistent relationships between facets and vertices then one is steered towards using regular polytopes.

155 *3.3. Convexity*

156 Assume that some global optimum and feasible control/state sequences $\underline{\mathbf{c}}_{i,k} = \{\mathbf{c}_k, \mathbf{c}_{k+1}, \dots\}$,
 157 $\underline{\mathbf{x}}_{i,k} = \{\mathbf{x}_k, \mathbf{x}_{k+1}, \dots\}$, $\mathbf{M}\mathbf{v}_i + \mathbf{N}\underline{\mathbf{c}}_{i,k} \leq \mathbf{d}$, are known for a set of initial points $\mathbf{x}_k = \mathbf{v}_i$, $i = 1, 2, \dots, m$.
 158 Then one can use convexity arguments to show that the following sequence $\underline{\mathbf{c}}_k$ is also feasible.

$$\left. \begin{aligned} \mathbf{x}_k &= \sum_{i=1}^m \lambda_i \mathbf{v}_i \\ \underline{\mathbf{c}}_k &= \sum_{i=1}^m \lambda_i \underline{\mathbf{c}}_{i,k} \\ \underline{\mathbf{x}}_k &= \sum_{i=1}^m \lambda_i \underline{\mathbf{x}}_{i,k} \\ \lambda_i &\geq 0, \quad \sum_{i=1}^m \lambda_i = 1 \end{aligned} \right\} \Rightarrow \mathbf{M}\mathbf{x}_k + \mathbf{N}\underline{\mathbf{c}}_k \leq \mathbf{d}. \quad (9)$$

159 Moreover, by definition of (4), the use of $\underline{\mathbf{c}}_k$ will give rise to convergent state trajectories.

160 **Lemma 3.1.** *Assuming n_x states, then given n_x+1 affinely independent vertices \mathbf{v}_i , $i = 1, \dots, n_x+1$
 161 and the corresponding optimal control sequences $\underline{\mathbf{c}}_{i,k}$, a feasible convergent sequence for the initial
 162 state \mathbf{x}_k is given from:*

$$\underline{\mathbf{c}}_k = \left[\underline{\mathbf{c}}_{1,k}, \underline{\mathbf{c}}_{2,k}, \dots, \underline{\mathbf{c}}_{n_x+1,k} \right] \underbrace{\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{n_x+1} \end{bmatrix}}_{\Lambda}; \quad (10)$$

$$\Lambda = \begin{bmatrix} \mathbf{v}_1 & \dots & \mathbf{v}_{n_x+1} \\ 1 & \dots & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix};$$

163 if $\lambda_i \geq 0$, $\forall i$; the condition $\sum_{i=1}^{n_x+1} \lambda_i = 1$ is given by definition of Λ in eqn. (10).

164 **Proof:** This is obvious from (9). □

165 **Definition 3.2 (Simplex).** *Define a simplex $\mathcal{S} \subset \mathbb{R}^{n_x}$ as the convex hull of n_x+1 affinely indepen-
 166 dent vertices: one vertex is the origin and the remaining n_x vertices define a facet not intersecting
 167 with the origin.*

168 **Corollary 3.1.** *The computation of λ_i in (10) will be such that $\sum_{i=1}^{n_x+1} \lambda_i = 1$ and $\lambda_i \geq 0$ if \mathbf{x}_k is
 169 inside the simplex \mathcal{S} made from these vertices. This is obvious from trivial vector algebra.*

170 **Theorem 3.1.** *If \mathbf{x}_k lies inside the simplex defined by n_x vertices and the origin, then a feasible
 171 convergent trajectory can be determined from (10) using those vertices.*

172 **Proof:** The proof follows directly from the Lemma and Corollary above. □

173
 174 The key insight offered in this section is that one can use convexity arguments very easily to
 175 form convergent and feasible trajectories for an arbitrary state \mathbf{x}_k if one has knowledge of feasible
 176 and convergent trajectories $\underline{\mathbf{c}}_{i,k}$, $\underline{\mathbf{x}}_{i,k}$ for a large enough number of possible initial points \mathbf{v}_i (here
 177 denoted vertices) spanning the space and, if \mathbf{x}_k lies inside the polytope defined by those vertices
 178 and the origin. Nevertheless, it is necessary to select the right n_x vertices, not only to ensure

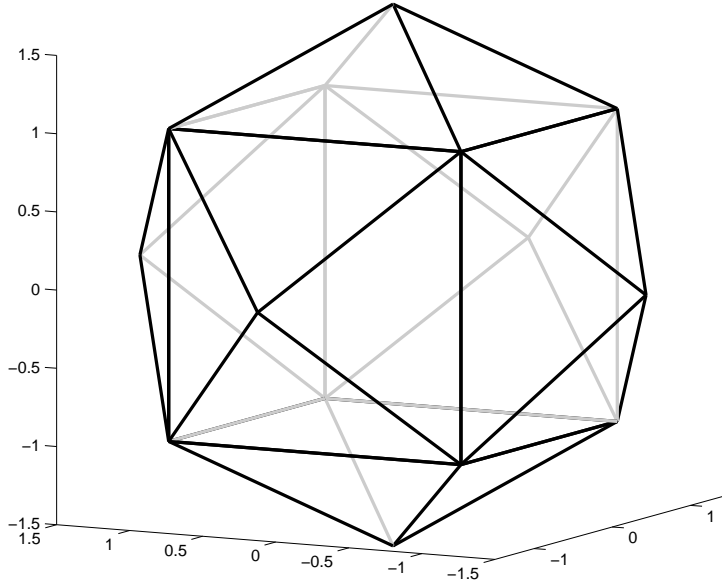


Figure 1: Three dimensional cube with extended centres to the facets.

$\lambda_i \geq 0$, $\sum_{i=1}^{n_x+1} \lambda_i = 1$ but also in general it will be non-trivial to identify the *best*² n_x vertices 179
from m possible, $m \gg n_x$. For instance, Definition 3.1 may suggest a facet which has many more 180
than n_x vertices in which case several different combinations of n_x vertices from these may give a 181
simplex in which \mathbf{x}_k lies and satisfying the requirement on λ_i . 182

3.4. Building a regular polytope for which each facet has only n_x vertices 183

If a facet has only n_x vertices, then one can use convexity arguments to argue that, for any 184
 \mathbf{x}_k for which that facet is active, there is a unique affine linear combination of the vertices as in 185
(9) which will give a feasible convergent trajectory. This section outlines one such set of polytopes 186
which are easy to define and has only n_x vertices per facet. In essence, the proposed shape will be 187
based from a cube and a cross-polytope. 188

Definition 3.3. Define a polytope $\mathcal{P} \subset \mathbb{R}^{n_x}$ as the following convex hull: 189

$$\mathcal{P} = \text{hull}\{\mathcal{C}, \mathcal{X}\}; \quad (11)$$

where \mathcal{C} is a $n_x D$ cube of volume 2^{n_x} centered in the origin, i.e. the convex hull of all sign 190
permutations of the vertices $[\pm 1, \pm 1, \dots, \pm 1]^T$; and \mathcal{X} is a $n_x D$ cross-polytope of volume $(2\mu)^{n_x} / n_x!$ 191
(with $\mu = \sqrt{n_x}$) centered in the origin, i.e. the convex hull of the vertices $\pm \mathbf{e}_i \mu$, $i \in \{1, \dots, n_x\}$. 192

Remark 3.1. The reader will note that \mathcal{P} is a full-dimensional polytope by construction and de- 193
fines the corners of an $n_x D$ cube and the centre of the facets of this cube stretched to the same 194
modulus (\mathbf{e}_i being the i th standard basis vector). The total number of vertices of this polytope is 195
given as $2^{n_x} + 2n_x$. The total number of facets, however, grow quicker with n_x , i.e. $n_x 2^{n_x}$. 196

Figure 1 demonstrates what \mathcal{P} might look like in the 3D case and the reader can clearly see 197
that each facet has 3 vertices, but vertices contribute to 4 or 6 facets each. 198

²The quality of the vertices is measured by the cost (3) of the resulting trajectory obtained from those vertices.
In general, the smaller simplex the better.

199 *3.5. Locating the active facet/vertices on proposed simple shape*

200 Having defined the shape in Definition 3.3, one wants to know how easily one can identify the
 201 active facet for an arbitrary initial point \mathbf{x}_k , that is to find which are the nearest n_x vertices such
 202 that \mathbf{x}_k lies inside the simplex formed by those vertices and the origin, thus satisfying (9). This
 203 section shows that for vertices defined as in Definition 3.3, one can find these vertices with a trivial
 204 search, simpler even than that used in Definition 3.1. The idea is simply to find the nearest n_x
 205 vertices to a point; these will always form a simplex, with the origin, enclosing the point. Further
 206 note that $n_x - 2$ vertices can be given by inspection due to the regular definitions of the vertices.

207 **Algorithm 3.1 (Define active vertices).** *Assume an $n_x D$ space, an initial point $\mathbf{x} = [x_1, \dots, x_{n_x}]^T$
 208 and define the j vertices of \mathcal{P} as \mathbf{t}_j . The active vertices \mathbf{t}_j^* which form a simplex with the origin
 209 containing \mathbf{x} in its interior can be found with the following algorithm.*

- 210 1. Create a vector $\mathbf{p} = [p_1, \dots, p_{n_x}]$, so that its elements p_i 's are the indices of the elements of
 211 \mathbf{x} sorted from the largest to the smallest absolute values.
- 212 2. The active vertices include standard basis vectors corresponding to p_1, \dots, p_{n_x-2} , with the
 213 choice being $\mathbf{t}_i^* = \mu \mathbf{e}_{p_i} \text{sgn}(x_{p_i})$, $i = 1, \dots, n_x - 2$.
- 214 3. The vertex $\mathbf{t}_{n_x-1}^*$ is calculated with $\mathbf{t}_{n_x-1}^* = \sum_{i=1}^{n_x} \mathbf{e}_{p_i} \text{sgn}(x_{p_i})$. The remaining vertex $\mathbf{t}_{n_x}^*$ is
 215 taken from the selection of one of the vertices $\mathbf{v}_1, \mathbf{v}_2$; where

$$\begin{aligned} \mathbf{v}_1 &= \mu \mathbf{e}_{p_{n_x-1}} \text{sgn}(x_{p_{n_x-1}}); \\ \mathbf{v}_2 &= \sum_{i=1}^{n_x-1} \mathbf{e}_{p_i} \text{sgn}(x_{p_i}) - \mathbf{e}_{p_{n_x}} \text{sgn}(x_{p_{n_x}}). \end{aligned}$$

216 *The choice is taken by selecting the vertex that is nearer to \mathbf{x} according to the Euclidean
 217 distance.*

- 218 4. Given $[\mathbf{t}_1^*, \dots, \mathbf{t}_{n_x}^*]$ and the origin, define the corresponding input trajectory using eqn. (10).

219 **Remark 3.2.** *Note that only a fixed number of control laws, given by $2^{n_x} + 2n_x$, needs to be stored.*

220 **Theorem 3.2.** *If $\mathbf{x} \in \mathcal{P}$, Algorithm 3.1 always defines vertices which in turn define a simplex with
 221 \mathbf{x}_k inside the simplex. Consequently, the vertices can be used to find a feasible convergent trajectory
 222 using (10).*

223 **Proof:** Note that \mathcal{P} can be tessellated by j simplices \mathcal{S}_j , each of them formed by the convex
 224 hull of the origin and the n_x vertices of the j th facet, with $j \in \{1, \dots, n_x 2^{n_x}\}$. Then it is obvious
 225 that \mathbf{x} will lie in one of the \mathcal{S}_j 's, specifically in the simplex dubbed \mathcal{S}_x . Algorithm 3.1 constructs
 226 the simplex \mathcal{S}_x using the vertices $[\mathbf{t}_1, \dots, \mathbf{t}_{n_x-1}]$ calculated in Step 2 and 3, completes it with one
 227 more vertex: \mathbf{v}_1 or \mathbf{v}_2 calculated in Step 3, the choice depends in the dimension (n_x) of the simplex.
 228 Therefore, Algorithm 3.1 can always select \mathcal{S}_x (which is probably a suboptimal choice) if $\mathbf{x} \in \mathcal{P}$.

229 Finally, it is noted that some vertices of \mathcal{P} not included in \mathcal{S}_x could also allow a smaller (valid)
 230 simplex for \mathbf{x} . The most likely vertex to swap is the vertex (\mathbf{v}_1 or \mathbf{v}_2) discarded to form \mathcal{S}_x reducing
 231 the size of the simplex and therefore validating if \mathcal{S}_x is or not the best option; Step 3 determines
 232 which vertex gives the smallest simplex on the basis that this gives best feasibility although in fact
 233 both alternatives would be feasible in general! □

3.5.1. Illustration of algorithm to find active vertices

Consider the initial points $\mathbf{x}_1 = [0.2, 0.3, 1]^T$, $\mathbf{x}_2 = [0.5, 0.8, 1]^T$. It is shown that the algorithm gives different choices for the last active vertices; the first $n_x - 1$ are the same.

1. The largest component of \mathbf{x}_1 is the 3rd, so include $\mu\mathbf{e}_3 = [0, 0, \mu]^T$ as \mathbf{t}_1^* . As the next highest component is the 2nd, and the 3rd component is positive, therefore, include $\mathbf{t}_2^* = \mathbf{e}_3 + \mathbf{e}_2 + \mathbf{e}_1$. Finally, the last vertex \mathbf{t}_3^* is chosen from $\mathbf{e}_3 + \mathbf{e}_2 - \mathbf{e}_1 = [-1, 1, 1]^T$ and $\mu\mathbf{e}_2$, whichever is closer; in this case $\mathbf{t}_3^* = [-1, 1, 1]^T$. Including the origin to complete the simplex, eqn. (10) then gives $\Lambda = [0.4, .25, .05, 0.3]^T$ which are clearly all positive and $\sum \lambda_i = 1$.
2. In the case of \mathbf{x}_2 , clearly one should include $\mathbf{t}_1^* = \mu\mathbf{e}_3$ and $\mathbf{t}_2^* = [1, 1, 1]^T$ and then one from $[-1, 1, 1]^T$, $[0, \mu, 0]^T$. In this case the $\mathbf{t}_3^* = [0, \mu, 0]^T$ is closest to \mathbf{x}_2 . Including the origin to complete the simplex, eqn. (10) then gives $\Lambda = [0.29, 0.5, 0.17, 0.04]^T$ which are clearly all positive and $\sum \lambda_i = 1$.

3.5.2. Comments on algorithm for finding active vertices

The reader will note that Algorithm 3.1 identifies the active vertices (and implicitly a facet with only n_x vertices) without at any point needing to define the facet representation for the implied shape. This is key later when the shape will be distorted and hence the vertex combinations making up the facets cannot be determined simply; as will be seen these do not need to be known and the paper uses only the vertices throughout. Also, the number of facets grows faster than the number of vertices.

It is noted that eqn. (10) requires a matrix inversion. However, the matrix to be inverted comprises columns which are mostly scaled standard basis vectors and one or two columns of the form $[\pm 1, \pm 1, \dots]^T$. Consequently the inversion is trivial in general and one could easily generate highly efficient code for this. Having identified Λ , the predicted control is computed from (10) and thus the overall algorithm is highly efficient.

3.6. Summary

This section has outlined the key background information required to develop an efficient parametric solution.

1. Definition of a regular polytope for which all facets have n_x vertices.
2. A link between optimal trajectories at n_x predefined initial values and possible feasible trajectories at other points inside the corresponding simplex.
3. A fixed number of vertices $2^{n_x} + 2n_x$, with the associated control laws, needs to be stored.
4. A highly efficient algorithm for identifying the active vertices for an arbitrary initial point.

It remains to modify this polytope to more general shapes or feasible regions, but as will be shown none of the efficiency benefits or key attributes are lost in doing so.

4. A suboptimal parametric MPC algorithm

This section first develops the vertex based polytopes to encompass a region close to the MCAS and therefore applicable to OMPC. Then, it proposes an efficient suboptimal parametric predictive control law.

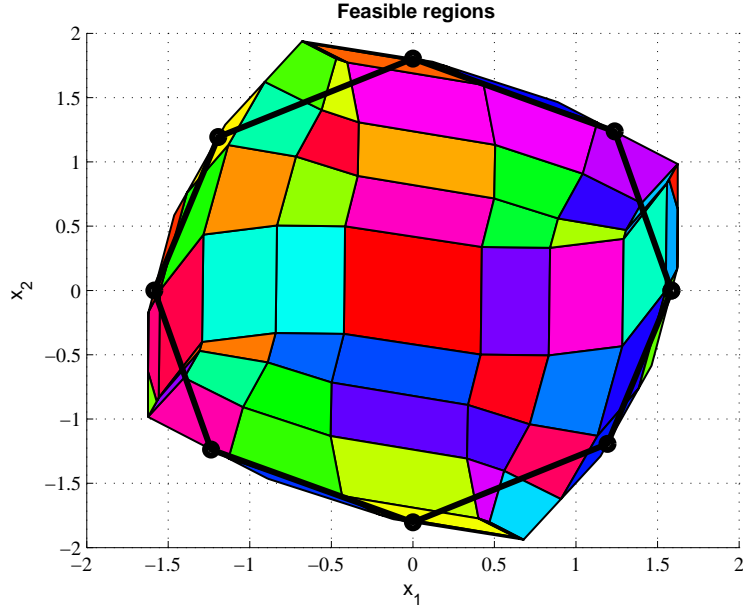


Figure 2: Controller partition obtained with mp-QP and \mathcal{P} (in thick line) for a two state example.

272 *4.1. Definition of the vertices of the feasible region*

273 The first objective is to find vertices corresponding to the polytope of Definition 3.3 which lie
 274 on the boundary of the MCAS as these are the points furthest from the origin, in those directions,
 275 for which a feasible control trajectory is known.

276 **Algorithm 4.1.** *Given the vertices \mathbf{t}_j of \mathcal{P} , find points in the same direction which lie on the*
 277 *MCAS by performing the following optimisations*

$$\beta_j^* = \arg \max_{\beta_j, \underline{\mathbf{c}}_{j,k}} \beta_j \quad \text{s.t.} \quad \mathbf{M}\mathbf{t}_j\beta_j + \mathbf{N}\underline{\mathbf{c}}_{j,k} \leq \mathbf{1}, \quad \beta_j > 0; \quad (12)$$

278 and hence define the vertices $\mathbf{V}_j = \beta_j^* \mathbf{t}_j$ and corresponding optimal sequences $\underline{\mathbf{c}}_{j,k}$ [Capital \mathbf{V} is
 279 used to distinguish between MCAS vertices and lower case \mathbf{t} for the regular shape]. Define the
 280 convex hull $\mathcal{P} = \text{hull}\{\mathbf{V}_1, \mathbf{V}_2, \dots\}$.

281 The vertices \mathbf{V}_j will describe a polytope \mathcal{P} which may be close to the MCAS, but clearly smaller
 282 in general due to the predefined assumptions on the directions of the vertices³ and restrictions to
 283 the complexity. The argument that will be made is that the loss in the volume of \mathcal{P} as compared to
 284 the MCAS is countered by the huge gains in the simplicity of definition as the vertices have regular
 285 directions as given in Definition 3.3 and are small in number. An example is given in Figure 2. It
 286 is clear that there is a huge gain as compared to the MAS and the feasible region for \mathcal{P} is quite
 287 close to the MCAS given the simplicity of the assumed shape.

³It can also be considered an *axis rotation* of \mathcal{P} in order to match the semi-axis of the largest ellipsoid contained in the MCAS. This will make \mathcal{P} even closer to the MCAS. This has not been done here.

Remark 4.1. As noted earlier, the controller partition obtained with mp-QP \mathcal{R} may have a very large number of facets and it is not obvious, a priori, which vertices will make up those facets (it is not necessarily the same as for the shape in Figure 1). In this paper, no attempt is made to compute the facet representation as this is not needed.

4.2. Locating the active vertices from polytope \mathcal{P}

The advantage of building \mathcal{P} from a regular polytope is that the directions of the vertices are the same and therefore locating the simplex in which a current point lies can be based on the same logic as Algorithm 3.1.

Algorithm 4.2 (Define active vertices). Assume an $n_x D$ space, an initial point $\mathbf{x} = [x_1, \dots, x_{n_x}]^T$ and use the vertices \mathbf{V}_j produced in Algorithm 4.1. The active vertices \mathbf{V}_j^* which form a simplex with the origin containing \mathbf{x} in its interior can be found with the following algorithm.

1. Define a correspondence $\mathbf{t}_i \equiv \mathbf{V}_i$, in that the values differ only by a positive scaling factor β_i^* .
2. Follow the steps of Algorithm 3.1 (note that $\mathbf{v}_1, \mathbf{v}_2$ of step 3 needs to be scaled to the corresponding \mathbf{V}_j term in order to select the nearest to \mathbf{x}).
3. The final choices \mathbf{V}_i^* 's will be the corresponding \mathbf{V}_i 's from \mathbf{t}_j^* obtained from Algorithm 3.1.

4.3. Proposed suboptimal control law

This section defines the proposed MPC algorithm assuming that the only information available to the user is:

- The vertices \mathbf{V}_j of \mathcal{P} on the boundary of the MCAS.
- The optimal sequences $\underline{\mathbf{c}}_{j,k}$ corresponding to each vertex.
- The constraint inequalities (2).
- The tail of the optimum sequence taken at the previous sample; typically the tail is taken as $\underline{\mathbf{c}}_{tail,k} = [\mathbf{c}_{k|k-1}, \mathbf{c}_{k+1|k-1}, \dots, \mathbf{c}_{k+n_c-2|k-1}, 0]^4$.

The proposed MPC algorithm is as follows.

Algorithm 4.3. At each sample perform the steps:

1. If the current state satisfies $\mathbf{M}\mathbf{x}_k \leq 1$, the unconstrained control law $\mathbf{u}_k = -\mathbf{K}\mathbf{x}_k$ is feasible and should be used, otherwise do steps 2-6.
2. Using the current state \mathbf{x}_k , find the active vertices \mathbf{V}_j^* (and corresponding $\underline{\mathbf{c}}_{j,k}$'s) using Algorithm 4.2.
3. Find the feasible control sequence $\underline{\mathbf{c}}_k$ (using \mathbf{V}_j^* 's, $\underline{\mathbf{c}}_{j,k}$'s) with eqn. (10) [Note, strictly this is guaranteed feasible iff $\lambda_i > 0$, $i = 1, \dots, n_x + 1$]. If $\lambda_i < 0$ implement the first element of $\underline{\mathbf{c}}_{tail,k}$ in (4), update the sample instant and return to Step 1.
4. Define the optimum sequence as $(\alpha \in \mathbb{R})$

$$\underline{\mathbf{c}}_k^* = (1 - \alpha) \underline{\mathbf{c}}_k + \alpha \underline{\mathbf{c}}_{tail,k} . \quad (13)$$

⁴ $k + i|k$ means the prediction for sample $k + i$ made at sample k .

321 5. Perform the optimisation

$$\begin{aligned}
\min_{\alpha} \quad & J_c = (1 - \alpha) \underline{\mathbf{c}}_k + \alpha \underline{\mathbf{c}}_{tail,k} \\
\text{s.t.} \quad & \mathbf{M}\mathbf{x} + \mathbf{N}\underline{\mathbf{c}}_k^* \leq 1; \\
& 0 \leq \alpha \leq 1.
\end{aligned} \tag{14}$$

322 Use the optimum α to compute $\underline{\mathbf{c}}_k^*$ of eqn. (13).

323 6. Implement the first element of $\underline{\mathbf{c}}_k^*$ in (4), update the sample instant and return to Step 1.

324 **Theorem 4.1.** *In the nominal case, Algorithm 4.3 has a guarantee of convergence and recursive*
325 *feasibility.*

326 **Proof:** Convergence relies on the standard argument that the inclusion of $\underline{\mathbf{c}}_{tail,k}$ allows an upper
327 bound on the cost J_c and moreover $J_c(k) \leq J_c(k-1) - \mathbf{c}_{k-1}^T \mathbf{S}\mathbf{c}_{k-1}$ and therefore J_c is Lyapunov;
328 once $J_c = 0$ the system is inside the MAS and unconstrained control applies. Recursive feasibility
329 also follows automatically from $\underline{\mathbf{c}}_{tail,k}$ being feasible by definition. \square

330
331 Ironically, there is no guarantee that the set \mathcal{P} is invariant and thus trajectories which begin
332 in \mathcal{P} may go outside \mathcal{P} and thus the optimal $\underline{\mathbf{c}}_k$ produced by Step 3 of Algorithm 4.3 may be
333 infeasible (because $\lambda_i < 0$). In this case one may rely more extensively on the tail until the state
334 trajectory re-enters \mathcal{P} . However, insisting on invariance of $\underline{\mathbf{c}}_k$ as opposed to $\underline{\mathbf{c}}_k^*$ each sample
335 instant would require either more complex set definitions and/or more demanding optimisations
336 and thus defeats the object of this paper.

337 Nevertheless, should this be desired, one could easily determine, offline, a separate region
338 $\mathcal{P}_2 \subset \mathcal{P}$ such that $\mathbf{x}_k \in \mathcal{P}_2 \Rightarrow \mathbf{x}_{k+i} \in \mathcal{P}, \forall i > 0$, thus preserving the feasibility of both $\underline{\mathbf{c}}_{tail,k}$
339 and $\underline{\mathbf{c}}_k$.

340 **Remark 4.2.** *This paper has not looked at robustness issues but it is also worth repeating that the*
341 *focus here is simplicity and in general algorithms with strong robustness results come at a price of*
342 *increased complexity.*

343 5. Numerical examples

344 This section will demonstrate several aspects of the proposed algorithm, but primarily the
345 two main attributes: (i) despite the restriction to so few points, the performance is only slightly
346 suboptimal and moreover recursive feasibility and convergence are achieved for all initial points in
347 \mathcal{P} and (ii) the complexity is always low relative to the norms in the literature for a given state
348 dimension. To this end, this section will give four examples, one with 2 states (slightly under
349 damped mass-spring-damper), two with 3 states (one a simplified two-input-two-output model of
350 electrical power generation) and one with 4 states. In each case $n_c = 5$ as much beyond this offered
351 little benefit by way of feasibility. Details of the systems used for the examples can be found in
352 the Appendix.

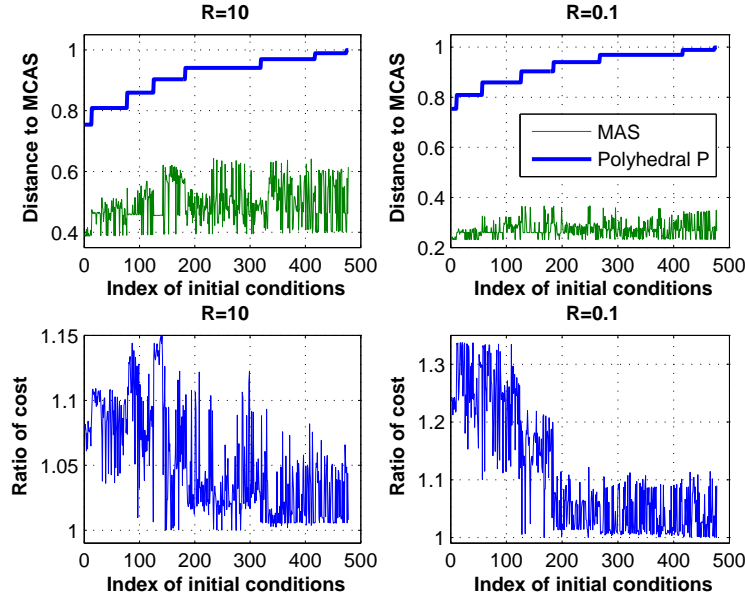


Figure 3: Cost and feasibility comparisons for example 1.

5.1. Performance and feasibility comparisons

Figures 3-6 show the efficacy of the proposed approach as compared to a standard OMPC algorithm in terms of performance and feasibility for two different choices of control weighting \mathbf{R} . Several initial conditions are chosen randomly with Monte-Carlo method.

- The top plots show the normalised size (distance to boundary of \mathcal{P} from the origin) of the feasible region for a large number of different directions, as compared to the MCAS. For completeness the figure includes the corresponding distance for the MAS, that is when $n_c=0$. The directions are ordered by normalised magnitude of the distance from the origin to \mathcal{P} .
- The lower plots show the normalised performance of Algorithm 4.3 as compared to OMPC, again for a large number of random initial points on the boundary of \mathcal{P} ; the same points as for the upper figures.

These figures demonstrate two clear conclusions. First, despite the very restricted shape of \mathcal{P} , there can be significant feasibility improvements as compared to the MAS, although unsurprisingly it is smaller than the MCAS which may have a very complex shape. Second, the performance degradation from using a suboptimal strategy is often quite small (here only the 3rd example has serious performance degradation) and, unsurprisingly again, the loss in performance was dependent on the initial condition.

In order to further emphasise the feasibility improvements of \mathcal{P} over the MAS and the proximity of the boundary of \mathcal{P} to the MCAS, the unsigned volumes for these polytopical sets are shown in Table 1. In this table, it is evident that \mathcal{P} allows larger feasible sets than the MAS, specially in the cases when the *optimal* feedback gain \mathbf{K} is highly tuned. On the other hand, the volume of \mathcal{P} is obviously smaller than the MCAS due its restricted shape, and in fact, there may be cases when the approximation is unacceptable (e.g. a non-symmetric MCAS) where other strategies not covered in this paper may be needed.

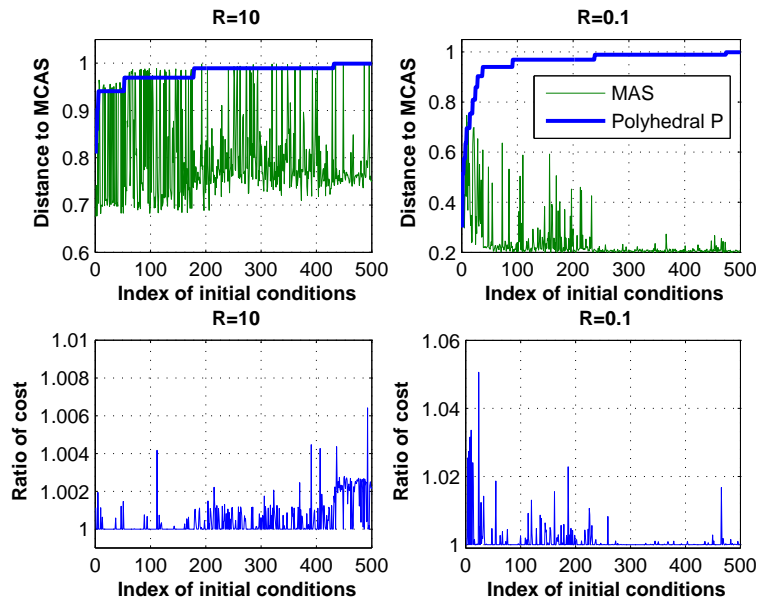


Figure 4: Cost and feasibility comparisons for example 2.

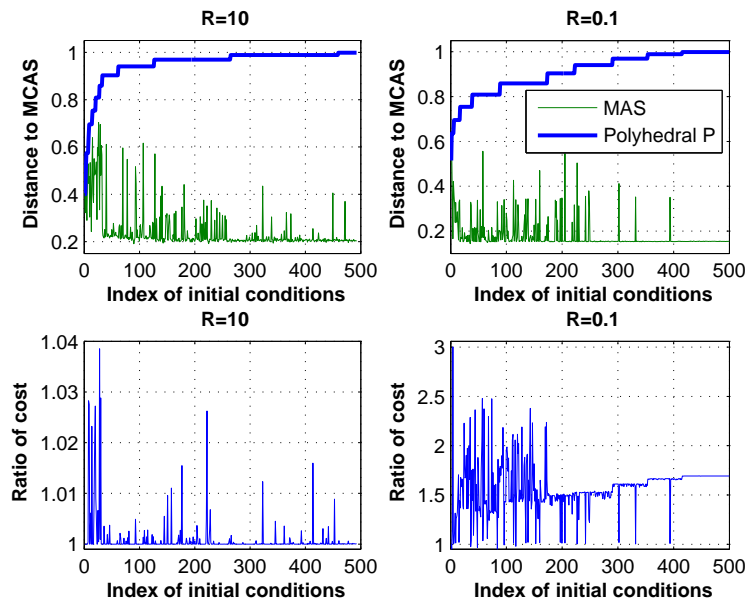


Figure 5: Cost and feasibility comparisons for example 3.

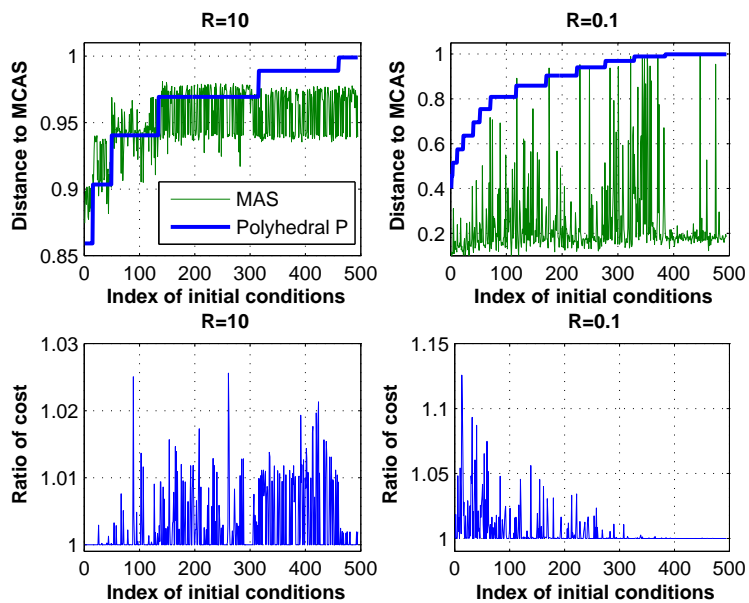


Figure 6: Cost and feasibility comparisons for example 4.

Table 1: Unsigned volumes of the polytopes of the examples.

	Example 1		Example 2		Example 3		Example 4	
	$\mathbf{R}=10$	$\mathbf{R}=0.1$	$\mathbf{R}=10$	$\mathbf{R}=0.1$	$\mathbf{R}=10$	$\mathbf{R}=0.1$	$\mathbf{R}=10$	$\mathbf{R}=0.1$
MAS	4.99	0.703	50.48	5.59	2313	209	18736	414
\mathcal{P}	17.91	8.23	69.58	52.69	9607	5643	19232	7512
MCAS	20.74	9.43	86.65	73.42	12983	10611	27447	13056

377 5.2. Complexity comparisons

378 In this section complexity of the proposed strategy is compared with the complexity of the ex-
379 act parametric solution to the optimisation problem (mp-QP) (Bemporad et al., 2002a) and to an
380 algorithm to approximate the exact parametric solution (Bemporad and Filippi, 2006). It is worth
381 to repeat here that the aim of methods proposed in this paper is not to approximate the parametric
382 solution of the optimisation problem, is rather to provide a MPC algorithm with constraint han-
383 dling capabilities for the implementation in hardware with very low processing power and memory
384 resources (e.g. a PLC) achieving enough guarantees of stability and recursive feasibility.

385 The idea of Bemporad and Filippi (2006) is to partition the MCAS into simplices where the
386 approximate solution inside each simplex is given by linear interpolation of the exact solution at
387 the vertices; for each simplex, if the resulting absolute error in the objective exceeds a prescribed
388 tolerance then it is divided into smaller simplices where it applies recursively. The initial set of
389 simplices is obtained with a Delaunay tessellation (DT) (Yeppremyan and Falk, 2005) of the MCAS.
390 The resulting approximate solution is organised over a tree structure for efficiency of evaluation, a
391 similar tree structure based on cubes was used in Johansen and Grancharova (2003).

392 The approach of Bemporad and Filippi (2006) gives substantial reduction in the number of
393 regions compared to mp-QP (Bemporad et al., 2002a) and the resulting size of the feasible set for
394 the approximate solution is the same as the MCAS. However, as argued in the introduction, any
395 result which is based on sub-division until the difference from the optimal is small will, in general,
396 lead to a large number of regions. Even if the error bound is generously relaxed the minimum
397 number of simplicies to describe the MCAS will depend on the shape and number of facets of the
398 MCAS (for the algorithm that determines the initial tessellation) and in this case tree structure
399 will not be very useful, all simplicies will be at the first level of the tree (just after the root of the
400 tree which is the set \mathbb{R}^{n_x}) and a conventional search will follow, i.e. the simplices have to be tested
401 one by one until the simplex containing \mathbf{x} is found. In contrast, Algorithm 3.1 will find the simplex
402 containing \mathbf{x} in a very small fixed number of arithmetic operations.

403 It is very important to mention that the original paper of Bemporad and Filippi (2006) does
404 not discuss issues about implementing this parametric approximation in the receding horizon
405 context, so stability and recursive feasibility cannot be guaranteed a priori and further conditions
406 need to be considered.

407 Table 2 focuses on the complexity of the solution as compared to a standard parametric ap-
408 proach of Bemporad et al. (2002a) and the approximation of Bemporad and Filippi (2006). Specif-
409 ically, the table compares (i) the number for regions for a standard mp-QP; (ii) the minimum
410 number of simplices needed to describe the MCAS obtained from a DT which is the initialization
411 algorithm from Bemporad and Filippi (2006) (with no error bounds to get the minimum number
412 of simplices); and (iii) vertices for Algorithm 4.3. This is a fair statement of data storage require-
413 ments. However, it should be noted further, that the search efficiency for the proposed algorithm
414 is far better, even if the number of regions/simplices/vertices were the same.

415 It is clear that the proposed algorithm allows significant reductions in complexity.

416 6. Implementation of the algorithm on a Programmable Logic Controller

417 The principal objective of Algorithm 4.3 is to be as efficient as possible to lower memory and
418 processing requirements, thus improving the potential for the application on standard industrial
419 hardware. This section introduces the PLC with its limitations and describes the implementation
420 of Algorithm 4.3.

Table 2: Number of regions obtained using mp-QP, simplices from a Delaunay tessellation (DT) and vertices using Algorithm 4.3.

	Regions (mp-QP)		Simplices (DT)		Vertices (Alg. 4.3)	
	R=10	R=0.1	R=10	R=0.1	R=10	R=0.1
Example 1	67	63	12	12	8	8
Example 2	77	77	31	30	14	14
Example 3	297	285	99	92	14	14
Example 4	727	251	137	94	24	24



Figure 7: Allen Bradley PLC – SCL500 processor family.

6.1. Allen Bradley – Rockwell Automation PLC

421

PLCs are by far the most accepted computers in industry which offer a reliable, safe and robust system. The arrangement and packaging of the PLC system is tailored for ease of integration into on-site control racks or cabinets with minimal effort. PLCs are also suited for the ease of implementation of standard wiring terminations. Each of these means that any on-site technician will have no difficulty or require any additional skills or tools when it comes to installing a controller. PLC systems also offer an added advantage that the program can be monitored online. The visual nature of the language means that the user can view in real-time the changing nature of bits, the value of counters or timers etc and how these relate to the overall program structure. Another advantage which PLC systems afford is that in the most cases, adjustments to programs can be made online without having to take the process or system under control offline. This is obviously an attractive property to industry as shutting down parts of a process can be a very costly affair indeed. However, it must be noted that this does present some safety issues which must be carefully addressed beforehand.

422

423

424

425

426

427

428

429

430

431

432

433

434

Nevertheless, normally their use is only to implement control sequences in open loop and/or different structures of PID controllers. For the purposes of this paper, the implementation is based on the family of SLC500 processors belonging to the Allen Bradley PLC systems, e.g. see Fig. 7.

435

436

437

The Allen Bradley set of PLC includes 64 Kbs of memory size with $0.37 \mu\text{-sec}$ of bit execution and the facilities to be programmed in 3 of 5 languages in agreement with the IEC 61131-3 stan-

438

439

440 dard using Control Logix 5000TM software programming package. Each of these allows for any
 441 combination of programming languages to be used for a single project. These three languages are:
 442 **1. Ladder Diagram** is a graphical language that uses a standard set of symbols to represent
 443 relay logic. The basic elements are coils and contacts which are connected by links. *Ladder logic*
 444 is thus a highly visual, easy to understand, program and diagnose as previously stated.
 445 **2. Function Block Diagram** is a graphical language that corresponds to circuit diagrams. The
 446 elements used in this language appear as blocks wired together to form circuits. The wires can
 447 communicate binary and other types of data between *Function Block Diagram* elements (e.g. Real,
 448 Integers, etc.).
 449 **3. Structured Text** is a general purpose, high-level programming language, similar to PASCAL
 450 or C. *Structured Text* is particularly useful for complex arithmetic calculations, allows to create
 451 boolean and arithmetic expressions as well a structured programming constructs such as conditional
 452 statements (IF, THEN, ELSE). Functions and function blocks can be invoked in this language.

453 6.2. PLC programming issues

454 There are some barriers and criteria required before an MPC algorithm can be coded effectively
 455 into the PLC; these are discussed next.

456 The SCL500 Control-Logix Controllers together with RS-Logix 5000 allows for the memory
 457 allocation of matrices (which it refers to as data arrays), for up to 3 dimensions. However, with the
 458 exception of one-dimensional, simple element by element arithmetic, it cannot be easily performed
 459 other matrix operations, notably: transposition, inversion and multiplication. To achieve such
 460 functions, it is thus necessary to code functions from scratch within the software to perform such
 461 operations. One could code an entire control algorithm in Structured Text, but for the ease
 462 of understanding by technicians it is strongly advisable to program most of the algorithm in a
 463 graphical language. In this way, the technical staff could view all the realtime data of the controller
 464 and debug the program if need it, in a more intuitive way. Finally, all the computations to calculate
 465 the next control movement should be done in a limited time dictated by the sampling period, so
 466 the computational load should be kept as low as possible to avoid accumulated errors during the
 467 tests. There is a need therefore to check and evaluate the algorithm timing after coding while
 468 noting that a bigger program inevitably requires more memory and therefore a more powerful PLC
 469 with the associated cost.

470 6.3. Implementation of Algorithm 4.3 on the target PLC

471 Algorithm 4.3 is programmed in the High Priority Periodic Execution Group (This periodicity
 472 is set up with the chosen sample time). The file structure of the program is shown in Figure 8,
 473 and the description of the routines is presented next:

474 ▷ **MPC.MAIN** (Ladder Logic Diagram). This is the main routine whose purpose is to control the pro-
 475 gram execution, calling routines in the correct order. Specifically, this routine first calls **Observer**;
 476 if \mathbf{x}_k satisfies $\mathbf{M}\mathbf{x}_k \leq 1$, sets $\underline{\mathbf{c}}_{\rightarrow k}^* = 0$, $\underline{\mathbf{c}}_{\rightarrow tail, k+1} = 0$ and calls **Controller_Output**; otherwise, calls
 477 the subroutines sequentially: **Vertex_Id**, **Optimisation** and **Controller_Output**.

478 ▷ **Observer** (Structured text). This subroutine is used to reconstruct the state vector using a
 479 Kalman filter. Invokes the subroutine **Matrix_Multiply** to complete the operations.

480 ▷ **Vertex_Id** (Structured text). This subroutine gets from **Data_Vertex** the optimal sequence $\underline{\mathbf{c}}_{\rightarrow k}$
 481 associated with the active vertices using Algorithm 4.2 and eqn. (10).

482 ▷ **Data_Vertex** (Structured text). This is not properly a program routine, is a data bank. Contains
 483 the information about the vertices \mathbf{V}_j of the polytope \mathcal{P} and associated optimal solutions $\underline{\mathbf{c}}_{\rightarrow j}$.

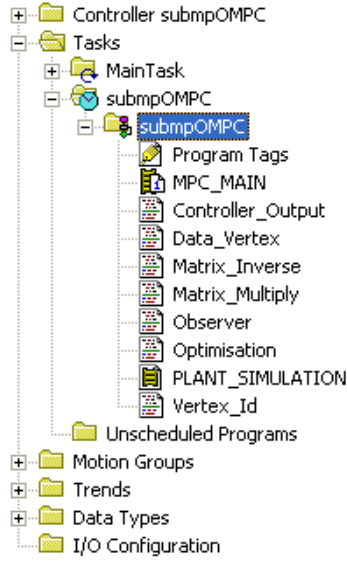


Figure 8: File structure of Algorithm 4.3 in the target PLC.

- ▷ **Optimisation** (Structured text). This subroutine is used to calculate the optimum $\underline{\mathbf{c}}_k^*$ from equation (13) finding the optimal α in problem (14). Before ending, this routine stores $\underline{\mathbf{c}}_{tail,k+1}$. 484
- ▷ **Controller_Output** (Structured text). This subroutine sends to the plant the calculated output using $\mathbf{u}_k = -\mathbf{K}\mathbf{x}_k + \mathbf{c}_k$. 485
- ▷ **Matrix_Multiply** (Structured Text). The matrix dimensions are passed to this along with two matrices. The routine returns the resulting answer matrix of the multiplication. 486
- ▷ **Matrix_Inverse** (Structured Text). The matrix dimensions are passed to this along with one matrix. The routine returns the resulting inverted matrix using an augmented matrix with Gaussian elimination. 487
- ▷ **PLANT_SIMULATION** (Ladder Logic Diagram). This subroutine is for development purposes only. It is used to simulate the plant dynamics/response and thus off-line testing of the controller coding. 488

7. Experimental laboratory test 489

This section shows the experimental results from applying the MPC law via the PLC, the aim is to show the effectiveness of Algorithm 4.3 on real hardware. The process consists of a motor fitted with a speed sensor, the control objective is to regulate the speed of the motor by manipulation of the input voltage. The mathematical model of the system with a sampling time of 0.5 sec is: 490

$$\mathbf{x}_{k+1} = \begin{bmatrix} 0.93 & -0.007 \\ 1 & 0 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \mathbf{u}_k;$$

$$\mathbf{y}_k = \begin{bmatrix} -0.1078 & 29.68 \end{bmatrix} \mathbf{x}_k;$$

with constraints $-3.5V \leq \mathbf{u} \leq 3.5V$, $-0.05 \leq \Delta \mathbf{u} \leq 0.05V$. In order to get a closed polytope, the output is bounded to $-1100 \text{ RPM} \leq \mathbf{y} \leq 1100 \text{ RPM}$. The experimental validation of the model is shown in Fig. 9. 491

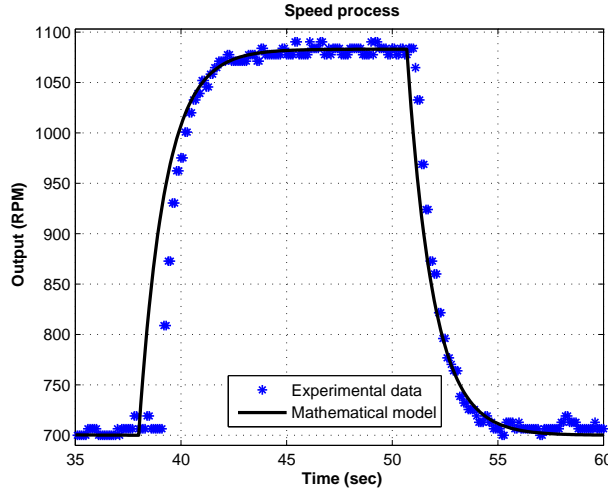


Figure 9: Model validation of the speed process.

503 In order to allow tracking of a time-varying reference, it is necessary to use an augmented state
 504 vector formulation:

$$\xi_k = [\mathbf{x}_k \quad \hat{\mathbf{d}}_k \quad \mathbf{u}_{k-1} \quad \mathbf{r}_k]^T, \quad (15)$$

505 where \mathbf{x}_k are the model states of the controlled plant, $\hat{\mathbf{d}}_k$ is the disturbance estimate, \mathbf{u}_{k-1} is the
 506 previous control input and \mathbf{r}_k is the reference signal. Therefore, the dynamics are formulated in
 507 $\Delta\mathbf{u}$ -form; in this framework, the system input at time k is $\Delta\mathbf{u}_k$ whereby \mathbf{u}_{k-1} is an additional
 508 state in the dynamical model, i.e. the system input can be obtained as $\mathbf{u}_k = \mathbf{u}_{k-1} + \Delta\mathbf{u}_k$.

509 **Remark 7.1.** Note that the augmented vector ξ_k replaces \mathbf{x}_k in all the algebra presented in Sec-
 510 tions 2-6.

511 The tuning parameters for the controller are $n_c = 3$, $\mathbf{R} = \mathbf{I}$, $\mathbf{Q} = \mathbf{I}$. Figure 10 shows the conven-
 512 tional mp-QP partitions and the polytope \mathcal{P} for this problem in the case when $\hat{\mathbf{d}}_k = 0$, $\mathbf{u}_{k-1} = 0$.
 513 In this case submpOMPC only needs to store information about 42 vertices and the associated
 514 control laws opposed to 97 regions from the mp-QP.

515 The program uses 15% of the available storage of the PLC including required memory for
 516 I/O, running cache and other necessary subroutines as it can be seen from the properties of the
 517 controller with the RSLogix[®] programming tool in Figure 11. This number indicates that one can
 518 even think in embed more than one controller in a single PLC as long as the scheduling is possible.

519 Two setpoint step changes are demanded; the results show that the proposed controller is
 520 tracking the setpoint accurately as can be seen in Figure 12. To complete the assessment of the
 521 implemented program, the diagnostics tool from the hardware (shown in Fig. 13) displays that the
 522 time for scanning the program each sample time which in the worst case is 9.85 ms while the elapsed
 523 time between triggers (sampling instants) for the speed process oscillates around 100.00 ± 0.425
 524 ms. The significance of this is the potential to apply the algorithm on much faster processes.

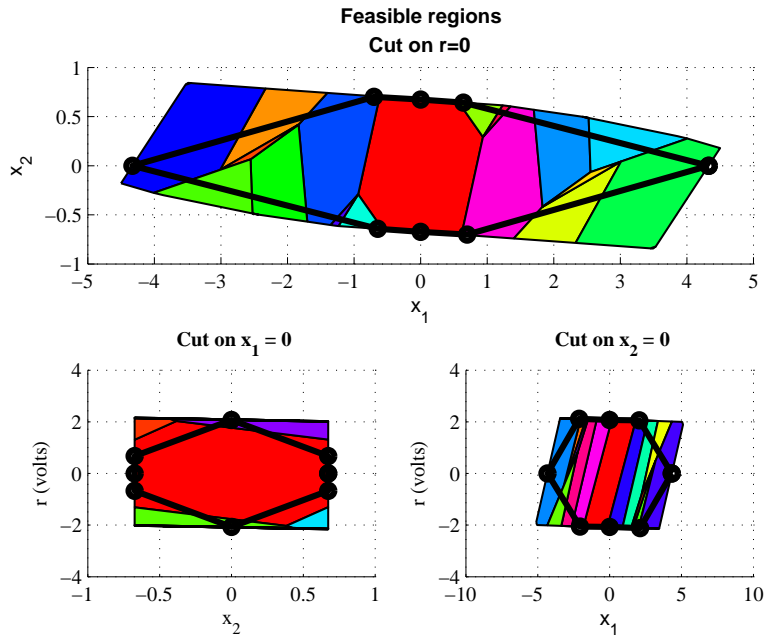


Figure 10: Cuts on the mp-QP controller partition and polytope \mathcal{P} for the speed process ($\hat{\mathbf{d}}_k = 0$, $\Delta \mathbf{u}_{k-1} = 0$).

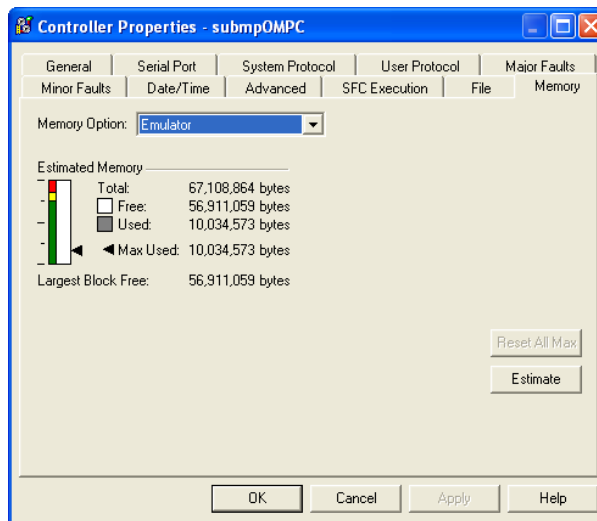


Figure 11: Algorithm 4.3 memory usage on the target PLC.

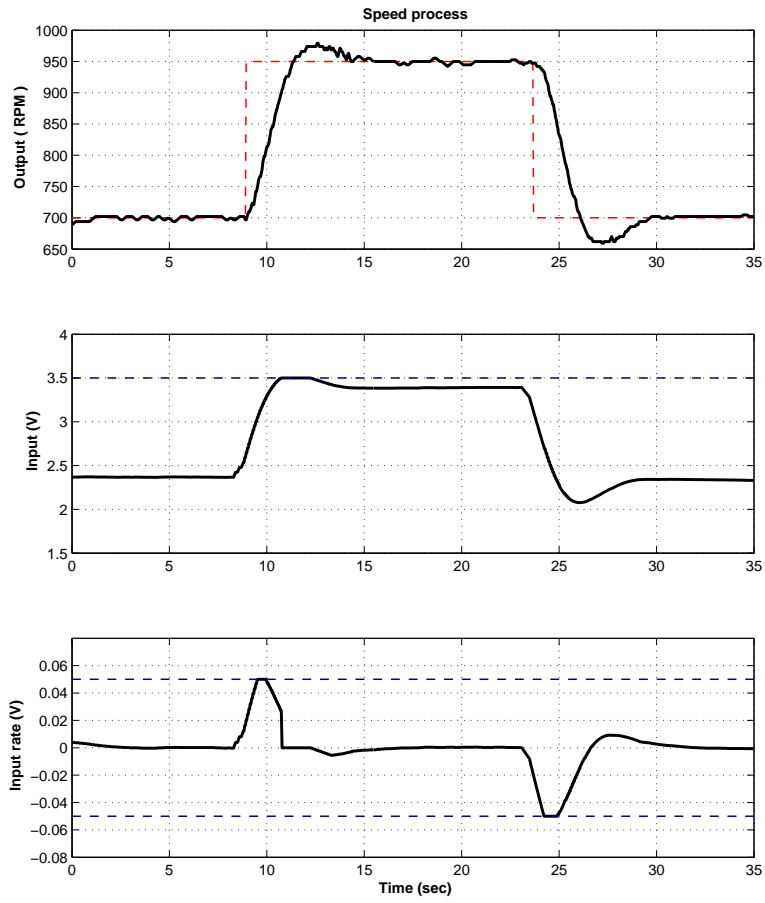


Figure 12: Experimental test for the speed process.

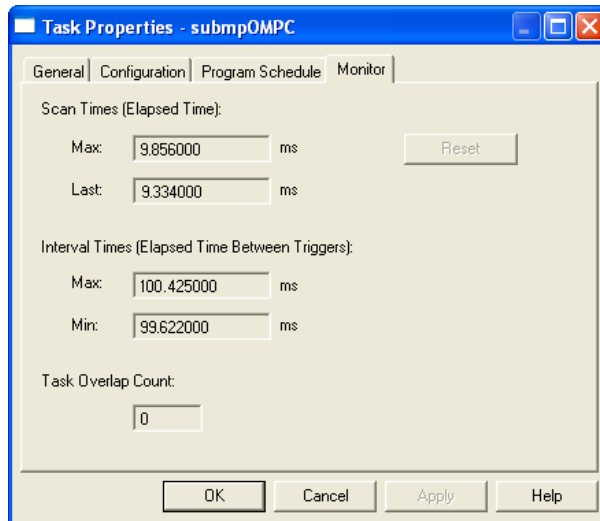


Figure 13: Execution time and sampling jittering of Algorithm 4.3.

8. Conclusions

525

This paper has demonstrated that a simple but very efficient constrained MPC algorithm can be effectively coded in a standard PLC unit. Thus, it is an industrial affordable alternative for replacing standard controllers with poor performance particularly in loops primarily controlled with PLC units. Along with this, the paper has made two important contributions.

First, it has evolved work on efficient parametric solutions to predictive control by removing the requirement for a specified bound on suboptimality and instead replaced that assumption with a specified bound on solution complexity. This is done without loss of recursive feasibility and stability and is a key advance when considering application to systems with low processor capability or highly complex optimal parametric solutions.

Secondly, it has proposed a novel choice of regions which allows for highly efficient search algorithms in conjunction with a simple definition which allows more shape flexibility than n_x D cubes. Specifically, the choice of regions allows one to define the feasible region in terms of facets with at most n_x vertices thus allowing simple convexity statements and thus simple computations; critically the facets do not need to be enumerated explicitly offline and are computed implicitly, with a trivial iteration, as required; hence the data storage requirements are lower.

While it would be impossible to give a generic statement that the proposed approach always gives at most a given degree of suboptimality or percentage loss in feasibility, the numerical examples show that these comparisons are easy to determine and in many cases the gain in simplicity and efficiency far outweighs any performance loss. Future work will look at the potential of using this approach recursively, that is to derive smaller additional polytopes near the boundary as is done typically by authors using a n_x D cube assumption.

References

547

- Bemporad, A., Borrelli, F., Morari, M., 2002a. Model predictive control based on linear programming - The explicit solution. *IEEE Transactions on Automatic Control* 47 (12), 1974–1985.
- Bemporad, A., Filippi, C., 2003. Suboptimal explicit receding horizon control via approximate multiparametric quadratic programming. *Journal of Optimization Theory and Applications* 117 (1), 9–38.
- Bemporad, A., Filippi, C., 2006. An algorithm for approximate multiparametric convex programming. *Computational optimization and applications* 35 (1), 87–108.
- Bemporad, A., Morari, M., Dua, V., Pistikopoulos, E., 2002b. The explicit linear quadratic regulator for constrained systems. *Automatica* 38 (1), 3–20.
- Borrelli, F., Baotic, M., Bemporad, A., Morari, M., 2001. Efficient on-line computation of constrained optimal control. In: *Proceedings of IEEE Conference on Decision and Control*.
- Canale, M., Fagiano, L., Milanese, M., 2009. Set membership approximation theory for fast implementation of model predictive control laws. *Automatica* 45 (1), 45–54.
- Canale, M., Fagiano, L., Milanese, M., 2010a. Efficient Model Predictive Control for Nonlinear Systems via Function Approximation Techniques. *IEEE Transactions on Automatic Control* 55 (8), 1911–1916.
- Canale, M., Fagiano, L., Razza, V., 2010b. Approximate NMPC for vehicle stability: Design, implementation and SIL testing. *Control Engineering Practice* 18 (6), 630–639.
- Christoffersen, F., Kvasnica, M., Jones, C., Morari, M., 2007. Efficient evaluation of piecewise control laws over a large number of polyhedra. In: *Proceedings of European Control Conference*.
- Gilbert, E., Tan, K., 1991. Linear systems with state and control constraints: The theory and application of maximal output admissible sets. *IEEE Transactions on Automatic Control* 36 (9), 1008–1020.
- Grieder, P., Wan, Z., Kothare, M., Morari, M., 2004. Two level model predictive control for the maximum control invariant set. In: *Proceedings of American Control Conference*.
- Johansen, T., 2003. Reduced explicit constrained linear quadratic regulators. *IEEE Transactions on Automatic Control* 48 (5), 823–828.

- 572 Johansen, T., Grancharova, A., 2003. Approximate explicit constrained linear model predictive control via orthogonal
573 search tree. *IEEE Transactions on Automatic Control* 48 (5), 810–815.
- 574 Johansen, T., Petersen, I., Slupphaug, O., 2002. Explicit sub-optimal linear quadratic regulation with state and input
575 constraints. *Automatica* 38 (7), 1099–1111.
- 576 Pistikopoulos, E., Dua, V., Bozinis, N., Bemporad, A., Morari, M., 2002. On-line optimisation via off-line parametric
577 optimisation tools. *Computers and Chemical Engineering* 24 (2-7), 175–185.
- 578 Rossiter, J., 2003. *Model-based predictive control, a practical approach*. Prentice Hall Int.
- 579 Rossiter, J., Grieder, P., 2005. Using interpolation to improve efficiency of multiparametric predictive control. *Auto-*
580 *matica* 41 (4), 637–643.
- 581 Rossiter, J., Kouvaritakis, B., Rice, M., 1998. A numerically robust state-space approach to stable predictive control
582 strategies. *Automatica* 34 (1), 65–73.
- 583 Scokaert, P., Rawlings, J., 1998. Constrained linear quadratic regulation. *IEEE Transactions on Automatic Control*
584 43 (8), 1163–1168.
- 585 Tondel, P., Johansen, T., Bemporad, A., 2003. Evaluation of piecewise affine control via binary search tree. *Auto-*
586 *matica* 39 (5), 945–950.
- 587 Valencia-Palomo, G., Rossiter, J., 2010. PLC implementation of an auto-tuned predictive control based on minimal
588 plant information. *ISA Transactions*. *In press*, doi: 10.1016/j.isatra.2010.10.002.
- 589 Yepremyan, L., Falk, J., 2005. Delaunay partitions in R^n applied to non-convex programs and vertex/facet enumer-
590 ation problems. *Computers & Operations Research* 32 (4), 793–812.

591 **Appendix A. Systems for the numerical examples**

592 **Example 1**

$$\mathbf{A} = \begin{bmatrix} -0.2 & -1 \\ 1 & 0 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad \mathbf{C} = [1 \quad 0];$$

$$\bar{\mathbf{u}} = 0.4 = -\underline{\mathbf{u}}; \quad \overline{\Delta \mathbf{u}} = 10 = -\underline{\Delta \mathbf{u}}; \quad \bar{\mathbf{y}} = 10 = -\underline{\mathbf{y}}.$$

593 **Example 2**

$$\mathbf{A} = \begin{bmatrix} 1.4 & -0.105 & -0.108 \\ 2 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 0.2 \\ 0 \\ 0 \end{bmatrix};$$

$$\mathbf{C} = [5 \quad 7.5 \quad 5];$$

$$\bar{\mathbf{u}} = 4 = -\underline{\mathbf{u}}; \quad \overline{\Delta \mathbf{u}} = 1 = -\underline{\Delta \mathbf{u}}; \quad \bar{\mathbf{y}} = 12 = -\underline{\mathbf{y}}.$$

594 **Example 3**

$$\mathbf{A} = \begin{bmatrix} 0.914 & 0 & 0.04 \\ 0.166 & 0.135 & 0.005 \\ 0 & 0 & 0.135 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 0.054 & -0.075 \\ 0.005 & 0.147 \\ 0.864 & 0 \end{bmatrix};$$

$$\mathbf{C} = \begin{bmatrix} 1.799 & 13.216 & 0 \\ 0.823 & 0 & 0 \end{bmatrix};$$

$$\bar{\mathbf{u}} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} = -\underline{\mathbf{u}}; \quad \overline{\Delta \mathbf{u}} = \begin{bmatrix} 0.25 \\ 0.5 \end{bmatrix} = -\underline{\Delta \mathbf{u}}; \quad \bar{\mathbf{y}} = \begin{bmatrix} 3 \\ 1 \end{bmatrix} = -\underline{\mathbf{y}}.$$

595 **Example 4**

$$\mathbf{A} = \begin{bmatrix} 0.9 & -0.105 & -0.108 & 0.2 \\ 0.6 & 0 & 0 & -0.1 \\ 0 & 0.8 & 0 & 0.3 \\ 0 & 0 & 0.8 & 0 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0.5 \end{bmatrix};$$

$$\mathbf{C} = [5 \quad 7.5 \quad 5 \quad 1];$$

$$\bar{\mathbf{u}} = 2 = -\underline{\mathbf{u}}; \quad \overline{\Delta \mathbf{u}} = 0.08 = -\underline{\Delta \mathbf{u}}; \quad \bar{\mathbf{y}} = 2.4 = -\underline{\mathbf{y}}.$$