The Potential of Interpolation for Simplifying Predictive Control and Application to LPV Systems

John Anthony Rossiter[1], Bert Pluymers[2], and Bart De Moor[2]

[1] Department Automatic Control and Systems Engineering, Mappin Street, University of Sheffield, S1 3JD, UK
j.a.rossiter@sheffield.ac.uk

[2] Department of Electrical Engineering, ESAT-SCD-SISTA, Kasteelpark Arenberg 10, Katholieke Universiteit Leuven, B-3001 Heverlee (Leuven), Belgium
{bert.pluymers,bart.demoor}@esat.kuleuven.be

Summary. This paper first introduces several interpolation schemes, which have been derived for the linear time invariant case, but with an underlying objective of trading off performance for online computational simplicity. It is then shown how these can be extended to linear parameter varying systems, with a relatively small increase in the online computational requirements. Some illustrations are followed with a brief discussion on areas of potential development.

1 Introduction

One of the key challenges in predictive control is formulating an optimisation which can be solved fast enough while giving properties such as guaranteed closed-loop stability and recursive feasibility. Furthermore one would really like good expectations on performance. A typical compromise is between algorithm or computational complexity and performance/feasibility. This paper looks at how reparameterising the input sequence using interpolation gives one possible balance, that is, it focuses on maximising feasible regions for a given algorithm/computational complexity without sacrificing asymptotic performance. The paper also considers some of the barriers to progress and hence suggests possible avenues for further research and in particular the potential for application to nonlinear systems. Several types of interpolation will be discussed, including interpolation between control laws [17, 1], where complexity is linked to the state dimension and interpolations based on parametric programming solutions [4].

Section 2 gives background information and Section 3 introduces the conceptual thinking in how interpolation techniques can widen feasibility while restricting complexity; to aid clarity, this is introduced using linear time invariant (LTI) models. Section 4 then extends these concepts to allow application to LPV and some classes of nonlinear systems. Section 5 gives numerical illustrations and the paper finishes with a discussion.

2 Background

This section introduces notation, the LPV model used in this paper, basic concepts of invariance, feasibility and performance, and some prediction equations.

2.1 Model and Objective

Define the LPV model (*uncertain* or *nonlinear case*) to take the form:

$$x(k+1) = A(k)x(k) + B(k)u(k), \qquad k = 0, \ldots, \infty, \tag{1a}$$

$$[A(k)\ B(k)] \in \Omega \triangleq \mathrm{Co}\{[A_1\ B_1], \ldots, [A_m\ B_m]\}, \tag{1b}$$

The specific values of $[A(k)\ B(k)]$ are assumed to be unknown at time k. Other methods [5, 6] can take knowledge of the current values of the system matrices or bounded rates of change of these matrices into account but these cases are not considered in this paper. However, it is conceivable to extend the algorithms presented in this paper to these settings as well.

When dealing with LTI models ($m = 1$), we will talk about the *nominal case.* The following feedback law is implicitly assumed :

$$u(k) = -Kx(k); \quad \forall k. \tag{2}$$

For a given feedback, the constraints at each sample are summarised as:

$$\begin{array}{l} x(k) \in \mathcal{X} = \{x : A_x x \leq \mathbf{1}\},\ \forall k \\ u(k) \in \mathcal{U} = \{u : A_u u \leq \mathbf{1}\},\ \forall k \end{array} \quad \Rightarrow \quad x(k) \in \mathcal{S}_0 = \{x : A_y x \leq \mathbf{1}\}, \forall k. \tag{3}$$

where $\mathbf{1}$ is a column vector of appropriate dimensions containing only 1's and $A_y = [A_x; -A_u K]$. We note that the results of this paper have been proven only for feedback gains giving quadratic stabilisability, that is, for feedback K, there must exist a matrix $P = P^{\mathrm{T}} > 0 \in \mathbb{R}^{n_x \times n_x}$ such that

$$\Phi_j^{\mathrm{T}} P \Phi_j \leq P, \quad \forall j, \quad \Phi_j = A_j - B_j K. \tag{4}$$

Problem 1 (Cost Objective). *For each of the algorithms discussed, the underlying aims are: to achieve* robust *stability, to optimise performance and to guarantee* robust *satisfaction of constraints. This paper uses a single objective throughout. Hence the algorithms will seek to minimise, subject to robust satisfaction of (3), an upper bound on:*

$$J = \sum_{k=0}^{\infty} (x(k)^{\mathrm{T}} Q x(k) + u(k)^{\mathrm{T}} R u(k)). \tag{5}$$

2.2 Invariant Sets

Invariant sets [2] are key to this paper and hence are introduced next.

Definition 1 (Feasibility and robust positive invariance). *Given a system, stabilizing feedback and constraints (1,2,3), a set $\mathcal{S} \subset \mathbb{R}^{n_x}$ is feasible iff $\mathcal{S} \subseteq \mathcal{S}_0$. Moreover, the set is robust positive invariant iff*

$$x \in \mathcal{S} \quad \Rightarrow \quad (A - BK)x \in \mathcal{S}, \quad \forall [A\ B] \in \Omega. \tag{6}$$

Definition 2 (MAS). *The largest feasible invariant set (no other feasible invariant set can contain states outside this set) is uniquely defined and is called the Maximal Admissible Set (MAS, [7]).*

Define the closed-loop predictions for a given feedback K as $x(k) = \Phi^k x(0)$; $u(k) = -K\Phi^{k-1}x(0)$; $\Phi = A - BK$, then, under mild conditions [7] the MAS for a controlled LTI system is given by

$$\mathcal{S} = \bigcap_{k=0}^{n} \{x : \Phi^k x \in \mathcal{S}_0\} = \{x : Mx \leq \mathbf{1}\}, \tag{7}$$

with n a finite number. In future sections, we will for the sake of brevity use the shorthand notation $\lambda S \equiv \{x : Mx \leq \lambda \mathbf{1}\}$. The MCAS (maximum control admissible set) is defined as the set of states stabilisable with robust constraint satisfaction by the specific control sequence:

$$\begin{aligned} u_i &= -Kx_i + c_i, \qquad && i = 0, ..., n_c - 1, \\ u_i &= -Kx_i, && i \geq n_c. \end{aligned} \tag{8}$$

By computing the predictions given a model/constraints (1,3) and control law (8), it is easy to show that, for suitable M, N, the MCAS is given as ([18, 19]):

$$\mathcal{S}_{\text{MCAS}} = \{x : \exists C \ \text{ s.t. } \ Mx + NC \leq \mathbf{1}\}; \quad C = [c_0^{\mathrm{T}} \ \ldots \ c_{n_c-1}^{\mathrm{T}}]^{\mathrm{T}}. \tag{9}$$

In general the MAS/MCAS are polyhedral and hence ellipsoidal invariant sets [9], $\mathcal{S}_E = \{x | x^{\mathrm{T}} P x \leq 1\}$, are suboptimal in volume [12]. Nevertheless, unlike the polyhedral case, a maximum volume $\mathcal{S}_E$ is relatively straightforward to compute for the LPV case. However, recent work [11, 3] has demonstrated the tractability of algorithms to compute MAS for LPV systems. This algorithm requires an outer estimate, e.g. $\mathcal{S}_0$, constraints at each sample (also $\mathcal{S}_0$) and the model Φ.

2.3 Background for Interpolation

Define several stabilizing feedbacks $K_i, i = 1, \ldots, n$, with K_1 the preferred choice.

Definition 3 (Invariant sets). *For each K_i, define closed-loop transfer matrices Φ_{ij} and corresponding robust invariant sets $\mathcal{S}_i$ and also define the convex hull $\overline{\mathcal{S}}$:*

$$\Phi_{ij} = A_j - B_j K_i, \quad j = 1, ..., m; \quad \mathcal{S}_i = \{x : x \in \mathcal{S}_i \Rightarrow \Phi_{ij} x \in \mathcal{S}_i, \forall j\}, \tag{10}$$

$$\overline{\mathcal{S}} \triangleq \mathrm{Co}\{\mathcal{S}_1, \ldots, \mathcal{S}_n\}. \tag{11}$$

Definition 4 (Feasibility). *Let $\Phi_i(k) = A(k) - B(k)K_i$, then [1] the following input sequence and the corresponding state predictions are recursively feasible within $\overline{\mathcal{S}}$:*

$$\begin{aligned} u(k) &= -\textstyle\sum_{i=1}^{n} K_i \prod_{j=0}^{k-1} \Phi_i(k-1-j)\hat{x}_i, \\ x(k) &= \textstyle\sum_{i=1}^{n} \prod_{j=0}^{k-1} \Phi_i(k-1-j)\hat{x}_i, \end{aligned} \tag{12}$$

if one ensures that

$$x(0) = \sum_{i=1}^{n} \hat{x}_i, \quad \text{with} \quad \begin{cases} \hat{x}_i = \lambda_i x_i, \\ \sum_{i=1}^{n} \lambda_i = 1, \ \lambda_i \geq 0, \\ x_i \in \mathcal{S}_i. \end{cases} \tag{13}$$

Definition 5 (Cost). *With $\tilde{x} = [\hat{x}_1^{\mathrm{T}} \ \dots \ \hat{x}_n^{\mathrm{T}}]^{\mathrm{T}}$, Lyapunov theory gives an upper bound $\tilde{x}^{\mathrm{T}}P\tilde{x}$ on the infinite-horizon cost J for predictions (12) using:*

$$P \geq \Gamma_u^{\mathrm{T}} R \Gamma_u + \Psi_i^{\mathrm{T}} \Gamma_x^{\mathrm{T}} Q \Gamma_x \Psi_i + \Psi_i^{\mathrm{T}} P \Psi_i, \quad i = 1, \dots, m, \tag{14}$$

with $\Psi_i = \mathrm{diag}(A_i - B_iK_1, \dots, A_i - B_iK_n)$, $\Gamma_x = [I, \dots, I]$, $\Gamma_u = [K_1, \dots, K_n]$.

These considerations show that by on-line optimizing over $\tilde{x}$, one implicitly optimizes over a class of input and state sequences given by (12). Due to recursive feasibility of these input sequences, this can be implemented in a receding horizon fashion.

3 Interpolation Schemes for LTI Systems

Interpolation is a different form of methodology to the more usual MPC paradigms in that one assumes knowledge of different feedback strategies with significantly different properties. For instance one may be tuned for optimal performance and another to maximise feasibility. One then interpolates between the predictions (12) associated with these strategies to get the best performance subject to feasibility. The underlying aim is to achieve large feasible regions with fewer optimisation variables, at some small loss to performance, and hence facilitate fast sampling. This section gives a brief overview and critique of some LTI interpolation schemes; the next section considers possible extensions to the LPV case.

3.1 One Degree of Freedom Interpolations [17]

ONEDOF uses trivial colinear interpolation, hence in (12) use:

$$x = \hat{x}_1 + \hat{x}_2; \quad \hat{x}_1 = (1-\alpha)x; \quad \hat{x}_2 = \alpha x; \quad 0 \leq \alpha \leq 1. \tag{15}$$

Such a restriction implies that α is the only d.o.f., hence optimisation is trivial. Moreover, if K_1 is the optimal feedback, minimising J of (5) over predictions (15,12) is equivalent to minimising α, $\alpha \geq 0$. Feasibility is guaranteed only in $\bigcup_i \mathcal{S}_i$.

Algorithm 1. *[ONEDOFa] The first move is $u = -[(1-\alpha)K_1 + \alpha K_2]x$ where:*

$$\alpha = \min_{\alpha} \quad \alpha \quad \text{s.t.} \quad [M_1(1-\alpha) + M_2\alpha]x \leq \mathbf{1}; \quad 0 \leq \alpha \leq 1. \tag{16}$$

M_1 and M_2 define mutually consistent [17] invariant sets corresponding to K_1 and K_2 respectively as $\mathcal{S}_i = \{x | M_i x \leq \mathbf{1}\}$.

Algorithm 2. *[ONEDOFb] The first move is $u = -[(1-\alpha)K_1 + \alpha K_2]x$ where:*

$$\alpha = \min_{\alpha,\beta} \quad \alpha \quad \text{s.t.} \quad \begin{cases} M_1(1-\alpha)x \leq (1-\beta)\mathbf{1}, \\ M_2\alpha x \leq \beta\mathbf{1}, \\ 0 \leq \beta \leq 1; \quad 0 \leq \alpha \leq 1. \end{cases} \tag{17}$$

This is solved by $\alpha = (\mu - 1)/(\mu - \lambda)$ where $\mu = \max(M_1 x), \quad \lambda = \max(M_2 x)$.

Summary: It can be shown that ONEDOFa will, in general, outperform ONEDOFb and have a larger feasible region. However, a proof of recursive feasibility has not been found for ONEDOFa whereas it has for ONEDOFb. Convergence proofs only exist for some cases [17], although minor modifications to ensure this are easy to include, e.g. [16]. However, the efficacy of the method relies on the existence of a known controller K_2 with a sufficiently large feasible region.

3.2 GIMPC: MPC Using General Interpolation

GIMPC [1] improves on ONEDOF by allowing full flexibility in the decomposition (12) of x and hence ensures (a priori): (i) a guarantee of both recursive feasibility and convergence is straightforward and (ii) the feasible region is enlarged to $\overline{\mathcal{S}}$. But the number of optimisation variables increases to $n_x + 1$.

Algorithm 3 (GIMPC). *Take a system* (1)*, constraints* (3)*, cost weighting matrices Q, R, controllers K_i and invariant sets $\mathcal{S}_i$ and compute a suitable P from (14). Then, at each time instant, solve the following optimization:*

$$\min_{\hat{x}_i,\lambda_i} \; \tilde{x}^{\mathrm{T}} P \tilde{x}, \quad \textit{subject to } (13), \tag{18}$$

and implement the input $u = -\sum_{i=1}^{n} K_i \hat{x}_i$.

Summary: The increased flexibility in the decomposition of x gives two benefits: (i) a guarantee of both recursive feasibility and convergence is straightforward and (ii) the feasible region is enlarged to $\overline{\mathcal{S}}$. The downside is an increase in the number of optimisation variables.

3.3 GIMPC2 Interpolations

GIMPC includes the restriction (13) that $\sum_{i=1}^{n} \lambda_i = 1, \ \lambda_i \geq 0$. However, [15] showed that such a restriction is unnecessary when the sets S_i are polyhedral. Removing the constraints on λ_i: (i) the feasible region may become substantially larger than $\overline{\mathcal{S}}$; (ii) reduces the number of optimisation variables (computation) and (iii) facilitates better performance.

Algorithm 4 (GIMPC2). *Using the same notation as algorithm 3, at each time instant, given the current state x, solve the following optimization problem on-line*

$$\min_{\hat{x}_i} \ \tilde{x}^{\mathrm{T}} P \tilde{x}, \quad \textit{subject to} \quad \begin{cases} \sum_{i=1}^{n} M_i \hat{x}_i \leq \mathbf{1}, \\ x = \sum_{i=1}^{n} \hat{x}_i, \end{cases} \tag{19}$$

and implement the input $u = -\sum_{i=1}^{n} K_i \hat{x}_i$, where the M_i defines a generalized MAS $\mathcal{S}_i'$ with mutually consistent constraints. See Algorithm 6 for details.

Summary: If the constraints on λ_i implicit in algorithm 3 (or eqn.(13)) are removed one gets two benefits: (i) the feasible region may become substantially larger (illustrated later) than $\overline{\mathcal{S}}$ and moreover (ii) the number of optimisation variables reduces. One still has guarantees of recursive feasibility and convergence. So GIMPC2 outperforms GIMPC on feasibility, performance and computational load. The main downside is that the associated set descriptions $\mathcal{S}_i'$ maybe more complex. This is discussed later, for instance in Algorithm 6.

3.4 Interpolations to Simplify Parametric Programming (IMPQP)

One area of research within parametric programming [4] solutions to MPC is how to reduce the number of regions. Interpolation is an under explored and simple avenue. Interpolation MPQP (IMPQP) [16] takes only the outer boundary of the MCAS. In any given region, the associated optimal $\mathbf{C}$ (9) can be summarised as: $x \in \mathcal{R}_i \quad \Rightarrow \quad \mathbf{C} = -K_i x + p_i$. For other x, for which a scaled version (by $1/\rho$) would lie in $\mathcal{R}_i$ on the boundary, then the following control law can be shown to give recursive feasibility and convergence:

$$\frac{x}{\rho} \in \mathcal{R}_i \quad \Rightarrow \quad \mathbf{C} = \rho(-K_i x + p_i). \tag{20}$$

Algorithm 5 (IMPQP). Offline: *Compute the MPQP solution and find the regions contributing to the boundary. Summarise the boundary of the MCAS in the form $M_b x \leq \mathbf{1}$ and store the associated regions/laws.*

Online: *Identify the active facet from $\rho = \max_j M_b(j,:)x$. With this ρ, find a feasible and convergent $\mathbf{C}$ from (20) and then perform the ONEDOFa interpolation*

$$\min_{\alpha} \alpha \quad \text{s.t.} \quad Mx + N\alpha C \leq \mathbf{1}, \tag{21}$$

and implement $u = -Kx + \alpha e_1^T C$.

Summary: For many MPQP solutions, the IMPQP algorithm [16] can be used to reduce complexity by requiring storage only of boundary regions and their associated control laws. Monte-Carlo studies demonstrated that, despite a huge reduction in set storage requirements, the closed-loop behaviour was nevertheless often close to optimal.

3.5 Other Algorithms

For reasons of space we give only a brief statement here. Other avenues currently being explored include so called Triple mode strategies [8], where the prediction structure has an extra non-linear mode to enlarge the terminal region. The design of this extra mode must take account of the LPV case. Another possibility, easily extended to the LPV case, is based on interpolation between the laws associated to the vertices of some invariant set. This technique, as with parametric methods, may suffer from issues of complexity.

4 Extensions to the LPV Case

The previous section dealt with the nominal case. This section shows how the interpolation methods can be extended to nonlinear systems which can be represented by an LPV model. In particular, it is noted that recursive feasibility was established via feasible invariant sets (MAS or MCAS). Hence, the main conjecture is that all of the interpolation algorithms carry across to the LPV case, with only small changes, as long as one can compute the corresponding invariant sets.

4.1 Invariant Sets and Interpolation for GIMPC and ONEDOFb

The GIMPC and ONEDOFb algorithms work on terms of the form $\max_j M(j,:)x_i$. For any given MAS, this value is unique and hence one can use, the set descriptions $\mathcal{S}_i$ of minimal complexity. Thus extension to the LPV case is straightforward, as long as polyhedral sets $\mathcal{S}_i$ exist and one replaces J with a suitable upper bound [1]. The implied online computational load increases marginally because the sets $\mathcal{S}_i$ for the LPV case are likely to be more complex.

An alternative method to perform interpolation in the robust setting is given in [20]. This method requires the use of nested ellipsoidal invariant sets, which can significantly restrict the size of the feasible region, but which allow interpolation to be performed without constructing a state decomposition as in (13).

4.2 Invariant Sets and Interpolation for GIMPC2 and ONEDOFa

The algorithm of [11] was defined to find the minimum complexity MAS of an LPV system for a single control law. Thus redundant constraints are removed at each iterate. However, for the GIMPC2 algorithm, constraints may need to be retained [15] even where they are redundant in the individual $\mathcal{S}_i$, because the implied constraints may not be redundant in the combined form of (16,19). Thus, the MAS must be constructed in parallel to identify and remove redundant constraints efficiently. One possibility, forming an augmented system, is introduced next. (There are alternative ways of forming an augmented system/states [15]; investigations into preferred choices are ongoing.)

Algorithm 6 (Method to find mutually consistent MAS for the LPV case).

1. *Define an augmented system*

$$X(k+1) = \Psi(k)X(k); \tag{22}$$

$$\Psi(k) = \begin{bmatrix} A(k) - B(k)K_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & A(k) - B(k)K_n \end{bmatrix}; \quad X = \begin{bmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_n \end{bmatrix}.$$

Define a set $\hat{\Omega}$ with $\Psi \in \hat{\Omega}$, describing the allowable variation in Ψ due to the variations implied by $[A(k)\ B(k)] \in \Omega$.

2. *Constraints (3) need to be written in terms of augmented state X as follows:*

$$A_u \underbrace{[-K_1, -K_2, \cdots]}_{\hat{K}} X(k) \le \mathbf{1}, \qquad k = 0, \dots, \infty, \tag{23a}$$

$$A_x[I, I, \cdots]X(k) \le \mathbf{1}, \qquad k = 0, \dots, \infty. \tag{23b}$$

3. *Assume that an outer approximation to the MAS is given by (23). Then letting $u = -\hat{K}X$, this reduces to $\mathcal{S}_o = \{X : M_o X \le \mathbf{1}\}$ where the definition of M_o is obvious.*
4. *Follow steps 2-5 of Algorithm in [11] to find the robust MAS as $\mathcal{S}_a = \{X : M_a X \le \mathbf{1}\}$.*

Remark 1 (Feasible region for robust GIMPC2). *Given the constraint $x = \sum_{i=1}^{n} x_i$, then one can find a projection of $\mathcal{S}_a$ to x-space from X-space as follows:*

$$\mathcal{S}_{G2} = \{x : \ \exists X \text{ s.t. } M_a X \le \mathbf{1}, \ x = [I, I, \dots, I]X\}. \tag{24}$$

Algorithm 7 (GIMPC2 for the LPV case). *Given a system* (1)*, constraints* (3)*, cost weighting matrices $Q = Q^{\mathrm{T}} > 0, R = R^{\mathrm{T}} > 0$, asymptotically stabilizing controllers K_i, corresponding polyhedral robust invariant sets $\mathcal{S}_a = \{X : M_a X \le \mathbf{1}\}$ and P satisfying* (14)*, solve on-line at each time instant, the following problem:*

$$\min_{\hat{x}_i} \tilde{x}^{\mathrm{T}} P \tilde{x}, \textit{subject to} \quad \begin{cases} x = [I, I, \dots, I]X, \\ M_a X \le \mathbf{1}, \end{cases} \tag{25}$$

and implement input $u = -[K_1,\ K_2,\ \dots\ ,\ K_n]X$.

Theorem 1. *Algorithm 7 guarantees robust satisfaction of* (3) *and is recursively feasible and asymptotically stable for all initial states $x(0) \in \mathcal{S}_{G2}$.*

Proof: from the invariance and feasibility of $\mathcal{S}_a$, irrespective of the values $A(k), B(k)$ (or $\Psi(k)$):

$$x(k) \in \mathcal{S}_{G2} \quad \Rightarrow \quad x(k+1) \in \mathcal{S}_{G2}. \tag{26}$$

As one can always choose new state components to match the previous predictions (one step ahead), repeated choice of the same decomposition gives convergence from quadratic stability (4) associated to each K_i, and hence system Ψ. Deviation away from this will only occur where the cost $J = \tilde{x}^{\mathrm{T}} P \tilde{x}$ can be made smaller still, so the cost function (25) acts as a Lyapunov function. □

Summary: Extension to the LPV case is not straightforward for GIMPC2 and ONEDOFa because the form of constraint inequalities implicit in the algorithms is $M_1 x_1 + M_2 x_2 + ... \leq \mathbf{1}$ and this implies a fixed and mutual consistent structure in M_i; they can no longer be computed independently! This requirement can make the matrices M_i far larger than would be required by say GIMPC. Once consistent sets $\mathcal{S}_i$ have been defined, the interpolation algorithms GIMPC2 and ONEDOFa are identical to the LTI case, so long as the cost J is replaced by a suitable upper bound.

4.3 Extension of IMPQP to the LPV Case

Extension of IMPQP to the LPV case is immediate given the robust MCAS (RMCAS) with the addition of a few technical details such as the use of an upper bound on the cost-to-go. A neat algorithm to find the RMCAS makes use of an autonomous model [10] (that is model (1) in combination with control law (8)) to represent d.o.f. during transients, for instance:

$$z_{k+1} = \Psi z_k; \quad z = \begin{bmatrix} x \\ C \end{bmatrix}; \quad \Psi = \left[\begin{array}{c|c} \Phi & B \; 0 \\ \hline 0 & U \end{array}\right]; \quad U = \begin{bmatrix} 0 & I_{(n_c-1)n_u \times (n_c-1)n_u} \\ 0 & 0 \end{bmatrix}. \tag{27}$$

Given (1), Ψ has an LPV representation. Define the equivalent constraint set as $\mathcal{S}_0 = \{x : \tilde{A}_y z \leq \mathbf{1}\}$. One can now form the MAS for system (27) with these constraints using the conventional algorithm. This set, being linear in both x and C, will clearly take the form of (9) and therefore can be deployed in an MPQP algorithm. One can either form a tight upper bound on the cost [1] or a simpler, but suboptimal choice, would be $J = C^T C$. Guaranteed convergence and recursive feasibility is easy to establish and the main downside is the increase in the complexity of the RMCAS compared to the MCAS.

Summary: Application of IMPQP to the LPV case can be done through the use of an autonomous model to determine the RMCAS. Apart from the increase in offline complexity and obvious changes to the shape of the parametric solution, there is little conceptual difference between the LTI and LPV solutions.

4.4 Summary

We summarize the changes required to extend nominal interpolation algorithms to the LPV case.

1. The simplest ONEDOF interpolations can make use of a robust MAS, in minimal form, and apart from this no changes from the nominal algorithm are needed. The simplest GIMPC algorithm is similar except that the cost needs to be represented as a minimum upper bound.
2. More involved ONEDOF interpolations require non-minimal representations of the robust MAS to ensure consistency between respective $\mathcal{S}_i$, and hence require many more inequalities. The need to compute these simultaneously also adds significantly to the offline computational load.
3. The GIMPC2 algorithm requires both mutual consistency of the MAS and the cost to be replaced by a minimum upper bound.
4. Interpolation MPQP requires the robust MCAS which can be determined using an autonomous model representation, although this gives a large increase in the dimension of the invariant set algorithm. It also needs an upper bound on the predicted cost.

It should be noted that recent results [14] indicate that in the LPV case the number of additional constraints can often be reduced significantly with a modest decrease in feasibility.

5 Numerical Example

This section uses a double integrator example with non-linear dynamics, to demonstrate the various interpolation algorithms, for the LPV case only. The algorithm of [19] (denoted OMPC) but modified to make use of robust MCAS [13] is used as a benchmark.

5.1 Model and Constraints

We consider the nonlinear model and constraints:

$$\begin{aligned} x_{1,k+1} &= x_{1,k} + 0.1(1 + (0.1x_{2,k})^2)x_{2,k}, \\ x_{2,k+1} &= x_{2,k} + (1 + 0.005x_{2,k}^2)u_k, \end{aligned} \tag{28a}$$

$$-0.5 \le u_k \le 1, \qquad [-10\ \ -10]^{\mathrm{T}} \le x_k \le [8\ 8]^{\mathrm{T}}, \qquad \forall k. \tag{28b}$$

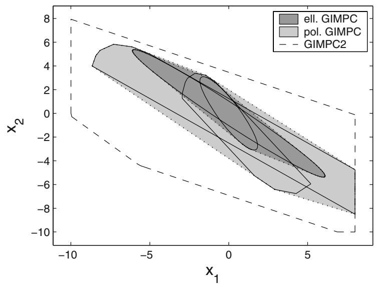
An LPV system bounding the non-linear behaviour is given as:

$$A_1 = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}, B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 1 & 0.2 \\ 0 & 1 \end{bmatrix}, B_2 = \begin{bmatrix} 0 \\ 1.5 \end{bmatrix}. \tag{29}$$

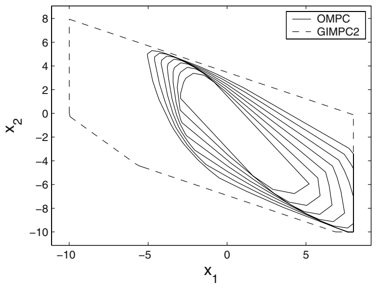
The nominal model ($[A_1 \;\; B_1]$) is used to design two robustly asymptotically stabilizing feedback controllers: the first is the LQR-optimal controller $K_1 = [0.4858\; 0.3407]^{\mathrm{T}}$ for $Q = \mathrm{diag}(1, 0.01), R = 3$ and the second $K_2 = [0.3\; 0.4]^{\mathrm{T}}$ has a large feasible region. Both controllers are robustly asymptotically stabilizing for system (29) and are hence also stabilizing for system (28).

5.2 Feasible Regions and Computational Load

Figure 1(a) presents the feasible regions for the various interpolations and for completeness also demonstrates the improvement compared to using the largest volume invariant ellipsoids. It is clear that GIMPC2 gives substantial feasibility increases compared to GIMPC/ONEDOF and indeed also compared to IMPQP (Figure 1(b)) for $n_c = 6$. The only increase in online computation arising due to the move from LTI to LPV systems is from the number of inequalities describing the invariant sets (work in progress may reduce this significantly). For completeness table 1 shows the numbers of d.o.f. and the numbers of inequalities for each algorithm. IMPQP is excluded from this table as the online computation is linked to the number of regions and hence is fundamentally different.



(a) Feasible regions of GIMPC using ellipsoidal and polyhedral invariant sets and GIMPC2.



(b) Feasible regions of IMPQP for $n_c = 0, \ldots, 6$ and GIMPC2.

Fig. 1. Feasible regions for different algorithms for model (29) using feedback laws K_1 and K_2

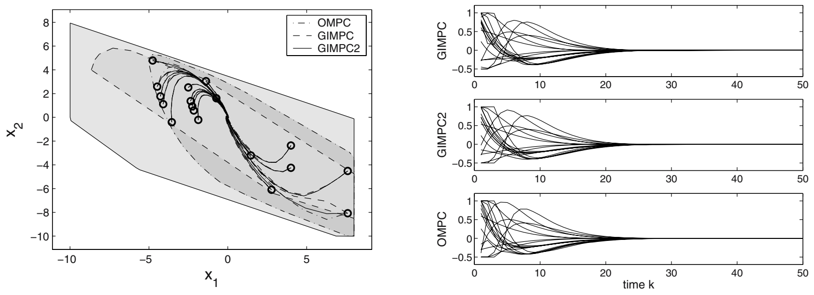
Table 1. Numbers of inequalities and d.o.f. required by GIMPC, GIMPC2 and OMPC for model (29)

	GIMPC	GIMPC2	OMPC
No. inequalities	22	63	506
No. d.o.f.	$n_x + 1 = 3$	$n_x = 2$	$n_c = 6$

5.3 Control Performance and Robust Closed-Loop Behaviour

It is useful to consider how the closed-loop performance, within the respective feasible regions, compares to 'optimal' (here taken as OMPC). Figure 2 depicts simulation results for GIMPC, GIMPC2 and OMPC, starting from initial states on the boundary of the intersection of the respective feasible regions. All three algorithms are stabilizing and result in nearly identical trajectories. The average control cost (according to (5)) of algorithms GIMPC and GIMPC2 is respectively 1.7% and 0.3% higher than OMPC with $n_c = 6$.

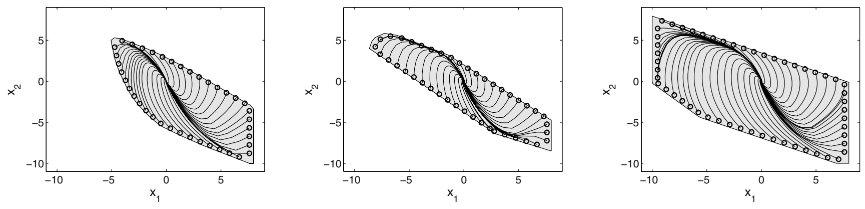
Evidence is also provided by way of closed-loop state trajectories in figure 3 that each of these algorithms is robustly feasible and convergent for the entire feasible region.



(a) State trajectories for the 3 different algorithms.

(b) Input sequences for the 3 different algorithms.

Fig. 2. Trajectories for GIMPC, GIMPC2 and OMPC for plant model (28) using feedback laws K_1 and K_2 and design model (29), starting from initial states at the boundary and the inside of the intersection of the feasible regions



(a) State trajectories for OMPC.

(b) State trajectories for GIMPC.

(c) State trajectories for GIMPC2.

Fig. 3. Trajectories for OMPC, GIMPC and GIMPC2 for plant model (28) using feedback laws K_1 and K_2 and design model (29), starting from initial states at the boundaries of the respective feasible regions

6 Conclusions and Future Directions

This paper has applied interpolation techniques to nonlinear systems which can be represented, locally, by an LPV model. The interpolation algorithms allow a degree of performance optimisation, have guarantees of recursive feasibility and convergence, while only requiring relatively trivial online computation. In fact the main requirement is the offline computation of the MAS or MCAS, with some structural restrictions. Notably, interpolations such as GIMPC2 may give far larger feasible regions than might be intuitively expected.

Nevertheless some questions are outstanding: (i) There is interest in whether interpolation concepts can be used effectively for more complicated non-linearities. (ii) This paper tackles only parameter uncertainty whereas disturbance rejection/noise should also be incorporated - some current submissions tackle that issue. (iii) It is still unclear what may be a good mechanism for identifying the underlying feedbacks K_i or strategies which give large feasible regions although Triple mode ideas [8] seem potentially fruitful. (iv) Interpolation has yet to be tested extensively on high order processes. (v) Finally, there is a need to devise efficient algorithms for computing low complexity, but large, invariant sets for high order systems.

Acknowledgments. To the Royal Society and the Royal Academy of Engineering of the United Kingdom. Research partially supported by research council KUL: GOA AMBioRICS, CoE EF/05/006 Optimization in Engineering, Flemish Government:POD Science: IUAP P5/22.

References

[1] M. Bacic, M. Cannon, Y. I. Lee, and B. Kouvaritakis. General interpolation in MPC and its advantages. *IEEE Transactions on Automatic Control*, 48(6):1092–1096, 2003.

[2] F. Blanchini. Set invariance in control. *Automatica*, 35:1747–1767, 1999.

[3] F. Blanchini, S. Miani, and C. Savorgnan. Polyhedral lyapunov functions computation for robust and gain scheduled design. In *Proceedings of the Symposium on nonlinear Control Systems (NOLCOS), Stuttgart, Germany*, 2004.

[4] F. Borrelli. *Constrained Optimal Control for Linear and Hybrid Systems.* Springer-Verlag, Berlin, 2003.

[5] A. Casavola, F. Domenico, and F. Giuseppe. Predictive control of constrained nonlinear systems via lpv linear embeddings. *International Journal of Robust and Nonlinear Control*, 13:281–294, 2003.

[6] L. Chisci, P. Falugi, and G. Zappa. Gain-scheduling MPC of nonlinear systems. *International Journal of Robust and Nonlinear Control*, 13:295–308, 2003.

[7] E.G. Gilbert and K. T. Tan. 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, 1991.

[8] L. Imsland and J.A. Rossiter. Time varying terminal control. In *Proceedings of the IFAC World Congress 2005, Prague, Czech Republic*, 2005.

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

[10] B. Kouvaritakis, J.A. Rossiter, and J. Schuurmans. Efficient robust predictive control. *IEEE Transactions on Automatic Control*, 45(8):1545–1549, 2000.

[11] B. Pluymers, J. A. Rossiter, J. A. K. Suykens, and B. De Moor. The efficient computation of polyhedral invariant sets for linear systems with polytopic uncertainty description. In *Proceedings of the American Control Conference (ACC), Portland, USA*, pages 804–809, 2005.

[12] B. Pluymers, J. A. Rossiter, J. A. K. Suykens, and B. De Moor. Interpolation based MPC for LPV systems using polyhedral invariant sets. In *Proceedings of the American Control Conference (ACC), Portland, USA*, pages 810–815, 2005.

[13] B. Pluymers, J. A. Rossiter, J. A. K. Suykens, and B. De Moor. A simple algorithm for robust MPC. In *Proceedings of the IFAC World Congress 2005, Prague, Czech Republic*, 2005.

[14] B. Pluymers, J. A. K. Suykens, and B. De Moor. Construction of reduced complexity polyhedral invariant sets for LPV systems using linear programming. *Submitted for publication*, 2005, (http://www.esat.kuleuven.be/˜sistawww/cgi-bin/pub.pl).

[15] J. A. Rossiter, Y. Ding, B. Pluymers, J. A. K. Suykens, and B. De Moor. Interpolation based MPC with exact constraint handling : the uncertain case. In *Proceedings of the joint European Control Conference & IEEE Conference on Decision and Control, Seville, Spain*, 2005.

[16] J. A. Rossiter and P. Grieder. Using interpolation to improve efficiency of multiparametric predictive control. *Automatica*, 41(4), 2005.

[17] J. A. Rossiter, B. Kouvaritakis, and M. Bacic. Interpolation based computationally efficient predictive control. *International Journal of Control*, 77(3):290–301, 2004.

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

[19] P. O. M. Scokaert and J. B. Rawlings. Constrained linear quadratic regulation. *IEEE Transactions on Automatic Control*, 43(8):1163–1168, 1998.

[20] Z. Wan and M. V. Kothare. An efficient off-line formulation of robust model predictive control using linear matrix inequalities. *Automatica*, 39(5):837–846, 2003.