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

Interpolation based predictive control

J.A. Rossiter
Dept. of Automatic Cont. & Systems Eng.
University of Sheffield, Mappin Street
Sheffield, S1 3JD, UK
email: J.A.Rossiter@sheffield.ac.uk
Tel. 44 114 2225685

B. Kouvaritakis M. Bacic
Dept. of Engineering Science
Parks Road
Oxford. OX1 3PJ
email: basil.Kouvaritakis@eng.ox.ac.uk
Tel. 44 1865 273105

***Abstract*— This paper investigates interpolation based predictive control and presents a study of the properties and therefore limitations of the approach. This understanding is used to develop an efficient algorithm with guarantees of recursive feasibility and stability.**

Keywords: Predictive control, computational efficiency, interpolation, constraint handling

I. Introduction

Constrained MPC [6] calls for the online optimisation of a cost subject to a number of constraints. The usual choice for the cost is quadratic in the degrees of freedom (d.o.f.), whereas the constraints are: (i) input/state constraints which are usually linear and (ii) stability constraints which can be linear or quadratic depending on whether the deployed terminal region is polytopic or ellipsoidal. With linear only constraints, the online optimisation reduces to a Quadratic Program (QP) [14], but this computation could be considered excessive in some scenarios. Restricting the number of d.o.f. seems a good option, but this could lead to significant suboptimality and/or feasibility problems.

A potentially effective solution to the above is a re-characterisation of the d.o.f. Instead of using individual predicted moves (typical trajectories are, $[1,0,0,\cdots]$, $[0,1,0,\cdots]$ etc.) one can instead interpolate between a set of predetermined predicted trajectories with desirable attributes [9], [12]. The inclusion into this set of the 'tail', the extension (shift) to current time of the trajectory computed at the previous time instant, enables the assertion of feasibility and closed loop stability [4], [7]. Enriching the set of pre-selected predicted trajectories with the inclusion of the LQ optimal trajectory [7] affords the extra advantage of convergence to an optimal control law.

Interpolation [5] can give significant computational advantages, in fact enormous if interpolation is co-linear [9]. However, the user is still left with some dilemmas:

- Co-linear interpolation implies non-inclusion of the tail and hence creates difficulties in guaranteeing recursive feasibility and stability [11]. Moreover the size of the feasible region can be limited.
- General non co-linear interpolation MPC [5] can guarantee feasibility/stability and has larger feasible regions, but requires the solution of an online SDP optimisation.

This paper presents an analysis of some interpolation schemes. Section 2 proposes an extension to general interpolation [5] which can be implemented through QP or Linear Programming (LP). Section 3 introduces existing co-linear algorithms, provides analysis and proposes a modification to give recursive feasibility. Section 4 gives numerical examples.

II. Linear programming formulation of general interpolation

Invariant [1] feasible sets provide a convenient means of deriving a recursive guarantee of feasibility. With the view to enlarging the size of such regions of attraction, interest has been focused on maximal admissible sets (MAS) [8] which are convex and polytopic. In this section we propose a simple extension to general interpolation[1] [5] is based on polytopic rather than ellipsoidal invariant sets.

A. Notation and feasible regions

This paper assumes a state space model and constraints

$$\begin{array}{c}\mathbf{x}_{k+1}=A\mathbf{x}_k+B\mathbf{u}_k;\quad \mathbf{y}_k=C\mathbf{x}_k\\ \underline{\mathbf{u}}\le\mathbf{u}_k\le\overline{\mathbf{u}};\quad \underline{\mathbf{x}}\le\mathbf{x}_k\le\overline{\mathbf{x}}\end{array} \tag{1}$$

where $\mathbf{x},\mathbf{u},\mathbf{y}$ are the state, input and output respectively and where the above inequalities apply on an element-by-element basis. Next we show how a predicted trajectory, and corresponding invariant set, can be made up by a convex linear combination of several control laws.

Lemma 2.1: The MAS [8], for the system of (1) under the control law $\mathbf{u}=-K_i\mathbf{x}$ is defined as:

$$\mathcal{S}_i=\{\mathbf{x}: M_i\mathbf{x}-\mathbf{d}_i\le 0\} \tag{2}$$

Proof: For a linear system with fixed state feedback, it is well known that the predicted input/output are linear in the current state. Hence linear constraints can all be expressed as $\mathbf{m}_i^T\mathbf{x}(k)\le d_i$. Stacking all the relevant constraints together, for a suitably large horizon, gives (2). Details of $M_i,\ \mathbf{d}_i$ can be obtained through straightforward algebra. □

[1]This paper allowed for uncertainty and hence was based on ellipsoidal invariant sets. As a consequence an SDP optimisation was required which is computationally demanding.

Theorem 2.1: The convex hull of $\mathcal{S}_i,\ i = 1, ..., \rho$ is invariant and feasible under the control law

$$\mathbf{u}(k+j|k) = -\sum_{i=1}^{\rho} \lambda_i K_i \Phi_i^j \mathbf{x}_i(k) \quad \text{where} \quad \left\{ \begin{array}{l|l} \mathbf{x}(k) = \sum_{i=1}^{\rho} \lambda_i \mathbf{x}_i(k); & \mathbf{x}_i(k) \in \mathcal{S}_i \\ \sum_{i=1}^{\rho} \lambda_i = 1; & \lambda_i \geq 0 \\ \Phi_i = A - BK_i & \end{array} \right. \tag{3}$$

Proof: Every initial condition $\mathbf{x}(k)$ in the convex hull of $\mathcal{S}_i$ can be decomposed as per (3). Then under (3) the state vector predictions are given by

$$\mathbf{x}(k+j|k) = -\sum_{i=1}^{\rho} \lambda_i \mathbf{x}_i(k+j|k); \quad \mathbf{x}_i(k+j|k) = \Phi_i^j \mathbf{x}_i(k) \tag{4}$$

and by the invariance of $\mathcal{S}_i,\ \mathbf{x}_i(k) \in \mathcal{S}_i$, implies that $\mathbf{x}_i(k+j|k) \in \mathcal{S}_i$ and hence $\mathbf{x}(k+j|k)$ will lie in the convex hull of $\mathcal{S}_i$. Corresponding to predictions (4), the control law of (3) can be rewritten as $\mathbf{u}(k+j|k) = -\sum_{i=1}^{\rho} \lambda_i K_i \mathbf{x}_i(k+j|k)$ which, by the triangle inequality implies

$$\sum_{i=1}^{\rho} \min_{\mathbf{x}_i \in \mathcal{S}_i} (\lambda_i K_i \mathbf{x}_i(k+j|k)) \leq \mathbf{u}(k+j|k) \tag{5}$$

However the feasibility of $\mathcal{S}_i$ implies

$$\sum_{i=1}^{\rho} \max_{\mathbf{x}_i \in \mathcal{S}_i} (\lambda_i K_i \mathbf{x}_i(k+j|k)) \leq \sum_{i=1}^{\rho} \lambda_i \overline{\mathbf{u}} = \overline{\mathbf{u}} \tag{6}$$

Similar arguments apply to the upper bound and hence (3) is feasible in $\bigcup_i \mathcal{S}_i$. □

Corollary 2.1: The predicted control law

$$\mathbf{u}(k+j|k) = -\sum_{i=1}^{\rho} \lambda_i K_i \Phi^j \hat{\mathbf{x}}_i, \quad \mathbf{x}(k) = \sum \hat{\mathbf{x}}_i \tag{7}$$

results in feasible predicted trajectories iff there exist $\hat{\mathbf{x}}_i,\ \lambda_i$ such that

$$M_i \hat{\mathbf{x}}_i \leq \lambda_i \mathbf{d}; \quad \sum \lambda_i = 1 \tag{8}$$

Proof: Let $\mathbf{x}_i = \hat{\mathbf{x}}_i / \lambda_i$, then (7) is equivalent to $\mathbf{x}_i \in \mathcal{S}_i$. The rest follows from Theorem 2.1. □

B. Interpolation based predictive control

Interpolation is most effective when the introduction of a new $K_i,\ \mathcal{S}_i$ results in a new convex hull which is bigger [12] than would arise from adding a extra d.o.f. to a more conventional MPC algorithm such as in [13]. Retuning [3] can be an effective way of enlarging feasible regions whereas adding a single d.o.f. to the input trajectory is not. Here, a predictive control algorithm is derived which uses the predictions of (7) and thus whose region of attraction is the convex hull of all the associated MAS.

Define an index of performance by the quadratic cost:

$$J = \sum_{k=1}^{\infty} \mathbf{x}(k)^T Q \mathbf{x}(k) + \mathbf{u}(k-1)^T R \mathbf{u}(k-1) \tag{9}$$

The aim is to compute and minimise this J subject to predictions/constraints of (3,7).

Theorem 2.2: The predicted cost under the control law of (7) is quadratic in $\hat{\mathbf{x}}_i$ and is given as:

$$J = \tilde{\mathbf{x}}^T P \tilde{\mathbf{x}}; \quad \tilde{\mathbf{x}} = [\hat{\mathbf{x}}_1^T, \hat{\mathbf{x}}_2^T, \cdots, \hat{\mathbf{x}}_\rho^T]^T \tag{10}$$

where P is defined as the positive definite solution of the Lyapunov equation:

$$P = \Gamma_u^T R \Gamma_u + \Psi^T \Gamma_x^T Q \Gamma_x \Psi + \Psi^T P \Psi$$
$$\Psi = \text{diag}[\Phi_1, \cdots,], \ \Gamma_x = [I, I, \cdots], \ \Gamma_u = [K_1, K_2, \cdots] \tag{11}$$

Proof: Obvious on substituting (4, 7) into (9). □

Algorithm 2.1: An interpolation MPC algorithm is given by minimizing the cost of (10) w.r.t. $\tilde{\mathbf{x}}$ given in (10) and subject to the corresponding constraints of (3):

$$\min_{\hat{\mathbf{x}}_i,\ i=1,\ldots,\rho} \tilde{\mathbf{x}}^T P \tilde{\mathbf{x}} \quad \text{s.t.} \quad \left\{ \begin{array}{l} \mathbf{x}(k) = \sum \hat{\mathbf{x}}_i \\ M_i \hat{\mathbf{x}}_i \leq \lambda_i \mathbf{d} \\ \sum \lambda_i = 1 \\ \lambda_i \geq 0 \end{array} \right. \tag{12}$$

Theorem 2.3: Algorithm 2.1 has guaranteed recursive feasibility and closed loop convergence.

Proof: Assume feasibility at time k, then at sampling instant $k+1$ one can choose a decomposition $\hat{\mathbf{x}}_i$ so that

$$\tilde{\mathbf{x}}(k+1|k+1) = \tilde{\mathbf{x}}(k+1|k) \tag{13}$$

which is known from Theorem 2.1 to be feasible. Standard arguments show that (13) implies $J(k+1) \leq J(k)$ and hence the control strategy is stabilising. □

When interpolating between two trajectories $(\mathbf{x}_1,\ \mathbf{x}_2)$ of which one (say $\mathbf{x}_1$) is generated by the LQ optimal, it may be convenient to replace the online optimisation of Algorithm 2.1 by an LP. This is not strictly equivalent to the QP solution in general, but nevertheless it could be argued makes good practical sense and also has a guarantee of convergence/feasibility through the monotonicity of λ_2.

Algorithm 2.2: For the case that $\rho = 2,\ \lambda_2 = 1 - \lambda_1,\ \hat{\mathbf{x}}_2 = \mathbf{x} - \hat{\mathbf{x}}_1$, an efficient LP interpolation MPC algorithm is given by:

$$\min_{\hat{\mathbf{x}}_1,\ \lambda_2} \lambda_2 \quad \text{s.t.} \quad \left\{ \begin{array}{l} M_1 \hat{\mathbf{x}}_1 \leq (1-\lambda_2)\mathbf{d} \\ M_2[\mathbf{x} - \hat{\mathbf{x}}_1] \leq \lambda_2 \mathbf{d} \\ 0 \leq \lambda_2 \leq 1 \end{array} \right. \tag{14}$$

Numerical illustrations in section 4 will demonstrate how effective the proposed interpolation of this section is in increasing the volume of the feasible region.

III. Co-linear interpolation

A. Existing co-linear algorithm

Algorithm 2.2 gives a significant reduction in online computation when compared with conventional algorithms with SDP or QP solvers. Furthermore, restricting the number of interpolation trajectories to only two requires just $n+1$ d.o.f. For scenarios where there is a need for further computational reductions, it may be convenient to use co-linear interpolation [9], [12] according to which $\mathbf{x}_1,\ \mathbf{x}_2$

and $\mathbf{x}$ are all taken to be in the same direction. With this restriction eqns.(3) become

$$\hat{\mathbf{x}}_1 = (1-\alpha)\mathbf{x};\ \hat{\mathbf{x}}_2 = \alpha\mathbf{x};\ \mathbf{u}(k+j|k) = -K_1\Phi_1^j\hat{\mathbf{x}}_1 - K_2\Phi_2^j\hat{\mathbf{x}}_1 \tag{15}$$

where for convenience $\lambda_1,\ \lambda_2$ have been substituted by $(1-\alpha),\ \alpha;\quad 0 \le \alpha \le 1$. From (2) the feasibility of the predictions can be ensured iff

$$\mathbf{q}(k)\alpha - \mathbf{p}(k) \le 0; \quad \left\{ \begin{array}{c} \mathbf{q}(k) = (M_2 - M_1)\mathbf{x}(k) \\ \mathbf{p}(k) = \mathbf{d} - M_1\mathbf{x}(k) \end{array} \right. \tag{16}$$

Algorithm 3.1: [12], [11] At each sampling instant, perform the minimisation

$$\min_{\alpha} \quad \alpha \quad \text{s.t.} \quad \left\{ \begin{array}{c} \mathbf{q}(k)\alpha - \mathbf{p}(k) \le 0 \\ 0 \le \alpha \le 1 \end{array} \right. \tag{17}$$

and compute the current control action from

$$\mathbf{u}_k = -[(1-\alpha)K_1 + \alpha K_2]\mathbf{x}(k) \tag{18}$$

Remark 3.1: Minimising α is equivalent [12] to minimising J over predictions (15). Algorithm 3.1 is very efficient with the main computational burden being the update of $\mathbf{q}(k),\ \mathbf{p}(k)$.

However, the co-linearity requirement poses some serious control theoretic problems [7], [11] as one can no longer satisfy (13); therefore one cannot assert a monotonicity property for J or α. Furthermore, a proof (should it exist) that feasibility at initial time ensures feasibility at all future instants is still unknown. The next section introduces modifications to guarantee recursive feasibility while retaining the very significant computational advantages.

B. Extensions of co-linear interpolation

For algorithm 3.1 $\mathbf{x}(k) \in \bigcup_i \mathcal{S}_i \not\Rightarrow \mathbf{x}(k+1) \in \bigcup_i \mathcal{S}_i$. Next, recursive feasibility is assured by requiring that each time instant the implied poles of interpolation $\hat{\mathbf{x}}_1,\ \hat{\mathbf{x}}_2$ are so adjusted to satisfy the constraints of (14).

First, introduce some assumptions and notation.

- Scale the rows of $M_i,\ \mathbf{d}$ such that each element of $\mathbf{d}_i = 1$ and hence define the notation

$$\mu_i = |M_i\mathbf{x}| \equiv \max_i \mathbf{e}_i^T M\mathbf{x} \tag{19}$$

  $\mathbf{e}_i^T$ is the ith standard basis vector. This gives an alternative definition of the MAS in (2):

$$\mathcal{S}_i = \{\mathbf{x} : |M_i\mathbf{x}| \le 1\} \quad \text{or} \quad \mathcal{S}_i = \{\mathbf{x} : \mu_i \le 1\} \tag{20}$$

- Define

$$\left. \begin{array}{rcl} \hat{\mathbf{x}}_1 = (1-\alpha)\mathbf{x} & \& & (1-\alpha)\mu_1 \le 1 \\ \hat{\mathbf{x}}_2 = \alpha\mathbf{x} & \& & \alpha\mu_2 \le 1 \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} \hat{\mathbf{x}}_1 \in \mathcal{S}_1 \\ \hat{\mathbf{x}}_2 \in \mathcal{S}_2 \end{array} \right. \tag{21}$$

Due to space limitations, we now give a brief summary only of the result. Note that $\mu_1 \le 1 \Rightarrow \mathbf{u} = -K_1\mathbf{x}$ so this case is not repeated.

Theorem 3.1: The following choice of α in (22) guarantees that $\hat{\mathbf{x}}_1 \in \mathcal{S}_1,\quad \hat{\mathbf{x}}_2 \in \mathcal{S}_2$. Given that $\mu_1 > 1,\ \mu_2 < 1$, this is also the smallest α for which one can guarantee that $|M_1(1-\alpha)\mathbf{x}| + |M_2\alpha\mathbf{x}| \le 1$.

$$\alpha = \frac{\mu_1 - 1}{\mu_1 - \mu_2} \tag{22}$$

Proof: Considering conditions (19,21) along with the inequalities of (14), it is clear that valid solutions of α are given from

$$(1-\alpha)\mu_1 + \alpha\mu_2 \le 1; \quad 0 \le \alpha \le 1 \tag{23}$$

From which (22) is obvious. If $\mu_1 > \mu_2$, and $(1-\alpha)\mu_1 + \alpha\mu_2 = 1$, then using $\alpha_2 < \alpha$ implies that $(1-\alpha_2)\mu_1 + \alpha_2\mu_2 > 1$. □

Remark 3.2: An alternative insight comes from drawing a straight line between $\mathbf{x}_1 = \mathbf{x}/\mu_1$ and $\mathbf{x}_2 = \mathbf{x}/\mu_2$ noting that then $\mathbf{x}_i$ is on the boundary of $\mathcal{S}_i$ so any convex linear combination of these predictions must be feasible. It then remains to find a θ (and hence the implied α) such that $\mathbf{x} = (1-\theta)\mathbf{x}_1 + \theta\mathbf{x}_2$.

Algorithm 3.2: Let the control law be defined[2] as

$$\begin{array}{lll} \mu_1 > 1 & \Rightarrow & \mathbf{u}_k = \frac{1-\mu_2}{\mu_2-\mu_1}K_1\mathbf{x} + \frac{\mu_1-1}{\mu_2-\mu_1}K_2\mathbf{x} \\ \mu_1 \le 1 & \Rightarrow & \mathbf{u}_k = -K_1\mathbf{x} \end{array} \tag{24}$$

The control law is undefined if $\mu_2 > 1$ and $\mu_1 > 1$.

The choice of α in (22) guarantees feasibility of the predictions (15). It remains to show that moreover one can guarantee recursive feasibility, that is:

$$\mathbf{x}_k \in \bigcup(\mathcal{S}_1, \mathcal{S}_2) \quad \Rightarrow \quad \mathbf{x}_{k+1} \in \bigcup(\mathcal{S}_1, \mathcal{S}_2) \tag{25}$$

As $\mathbf{x} \in \mathcal{S}_1 \Rightarrow \mathbf{u} = -K_1\mathbf{x}$, it is only required to prove that $\mathbf{x}_k \in \mathcal{S}_2 \quad \Rightarrow \quad \mathbf{x}_{k+1} \in \bigcup(\mathcal{S}_1, \mathcal{S}_2)$.

Lemma 3.1: Define, with $0 \le \theta \le 1$, the sets:

$$\begin{array}{rcl} T_1 & = & \{\hat{\mathbf{x}}_1 : M_1\hat{\mathbf{x}}_1 - (1-\theta)\mathbf{d} \le 0\} \\ T_2 & = & \{\hat{\mathbf{x}}_2 : M_2\hat{\mathbf{x}}_2 - \theta\mathbf{d} \le 0\} \end{array} \tag{26}$$

For the choice of α given in (22) and $\hat{\mathbf{x}}_1 = (1-\alpha)\mathbf{x},\ \hat{\mathbf{x}}_2 = \alpha\mathbf{x}$, it follows that

$$\theta = \alpha\mu_2 \quad \Rightarrow \quad \hat{\mathbf{x}}_1 \in T_1,\ \hat{\mathbf{x}}_2 \in T_2 \tag{27}$$

Proof: This follows directly from substition of (22). □

Theorem 3.2: $\mathbf{x} = \hat{\mathbf{x}}_1 + \hat{\mathbf{x}}_2,\ \hat{\mathbf{x}}_1(k) \in T_1,\ \hat{\mathbf{x}}_2(k) \in T_2$ and $\mathcal{S}_1 \subset \mathcal{S}_2$ are sufficient to ensure that

$$\mathbf{x}_k \in \mathcal{S}_2 \quad \Rightarrow \quad \mathbf{x}_{k+1} \in \mathcal{S}_2 \tag{28}$$

and hence that control law of (24) has a recursive feasibility guarantee.

Proof: $\hat{\mathbf{x}}_1(k) \in T_1 \quad \Rightarrow \quad \hat{\mathbf{x}}_1(k+1) \in T_1$. Moreover, if $\mathcal{S}_1 \subset \mathcal{S}_2$ one can state that:

$$T_1 \subset T_{2,\theta}; \quad T_{2,\theta} = \{\mathbf{x} : M_2\mathbf{x} - (1-\theta)\mathbf{d} \le 0\} \tag{29}$$

and hence $\hat{\mathbf{x}}_1(k+1) \in T_1 \Rightarrow \hat{\mathbf{x}}_1(k+1) \in T_{2,\theta}$ Finally, it is also obvious from definitions (26,29) that

$$\left. \begin{array}{c} \hat{\mathbf{x}}_2 \in T_2 \\ \hat{\mathbf{x}}_1 \in T_{2,\theta} \end{array} \right\} \quad \Rightarrow \quad \hat{\mathbf{x}}_1 + \hat{\mathbf{x}}_2 \in \mathcal{S}_2 \tag{30}$$

□

[2]This takes the same form as (18) but with α given as in (22), that is $\alpha = -\frac{\mu_1-1}{\mu_2-\mu_1}$.

C. Comparison of Algorithms 3.1, 3.2

The reader is given next a summary of how the proposed algorithm 3.2 differs from that of algorithm 3.1.

1) Algorithm 3.2 guarantees that $\mathbf{x}_k \in \mathcal{S}_2 \Rightarrow \mathbf{x}_{k+1} \in \mathcal{S}_2$. This is not the case for algorithm 3.1, even though it has a larger region of attraction.
2) Algorithm 3.2 is more cautious than 3.1 in that it will often choose a larger value of α. This is because:

$$|[(1-\alpha)M_1+\alpha M_2]\mathbf{x}(k)| \leq 1 \not\Rightarrow (1-\alpha)\mu_1+\alpha\mu_2 \leq 1 \quad (31)$$

3) The formulation of (24) facilitates staightforward interpretation as the optimisation is removed.

Due to co-linearity one may not establish for either algorithm, a monotonicity of cost proof [7], [11]. A convenient alternative was presented in [11]. If the tests of [11] should fail then alternative even more cautious approaches are needed, e.g. [10], which restrict the allowable values of α further still. This is unsurprising as rigorous guarantees of stability often come at the price of a sacrifice in performance.

IV. Examples

This section will illustrate the use of algorithms developed in this paper. Several aspects will be demonstrated: (i) the potential increase in the feasible region using interpolation; (ii) the good performance achieved by algorithm 3.2 and (iii) a comparison of performance/feasibility with the global optimal [13].

The following double integrator model will be used for the numerical study:

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.0787 \end{bmatrix}; \quad y_k = [1 \;\; 0]\mathbf{x}_k \quad (32)$$

with input and state limits

$$-1 \leq u_k \leq 1; \quad -2 \leq [1 \;\; 1]\mathbf{x}_k \leq 2 \quad (33)$$

The optimal control law for various weights $Q,\ R$ are in table 1. The prefered choice is K_1.

	$Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$	$Q = \begin{bmatrix} 0.04 & 0.2 \\ 0.2 & 1 \end{bmatrix}$
$R = 0.1$	$K_1 = [2.83 \;\; 2.83]$	$K_2 = [0.55 \;\; 3.02]$

Table 1. Optimal feedback K for different weights

Figures 1,2 show the feasible regions that arise for algorithms 2.1, 2.2, 3.1, 3.2 and OMPC with $n_c = 1, 2, 3, 4, 5, 20$ (n_c denotes the number of d.o.f.).

- Figure 1 shows that OMPC has a very small feasible region unless n_c is large and also that interpolation (i.e. algorithms 2.1, 2.2) is more effective at increasing the feasible region than increasing n_c in [13].
- Figure 2 shows 'guaranteed' feasible regions (dark shading), that is $\bigcup(\mathcal{S}_1, \mathcal{S}_2)$ for algorithms 3.1, 3.2. Algorithm 3.1 has a larger feasible region (light shading) but it is non-convex, and difficult to utilise effectively.

A number of initial conditions are used to illustrate the variability of closed-loop behaviour (marked in figure 3) that is possible around the state space. Simulations are performed only where the initial state is feasible. OMPC is infeasible with small n_c for all these points, so the cost and the state trajectories are for $n_c = 20$; this is to benchmark the other algorithms. The corresponding evolutions of α are given in figure 4 and the closed-loop costs, corresponding to J are given in Table 2.

Algorithm				
OMPC($n_c = 20$)	2.1	3.1	3.2	2.2
25.8	28.2	26.0	0	29.4
2.07	2.07	0	0	2.80
6.12	6.12	6.12	7.14	6.22
60.2	93.6	62.0	123.8	225.3
16.6	16.6	0	0	40.3
10.82	11.02	10.86	0	11.02
10.23	10.23	10.23	11.19	14.90
46.62	48.89	46.63	70.64	160.66
18.93	19.21	18.93	32.21	21.12

Table 2: Closed-loop run-time costs corresponding to J (0 implies infeasibility).

1. Algorithm 2.1 (solid lines in Fig. 4a) gives the largest 'guaranteed' feasible region and excellent performance (close to OMPC) despite using only 3 d.o.f..
2. Algorithm 3.1 gives excellent performance (close to OMPC). However, although for this example it retains recursive feasibility, no proof/counter proof yet exists.
3. Algorithm 3.2 maintains the state within $\bigcup(\mathcal{S}_1, \mathcal{S}_2)$, and hence ensures feasibility. However, it is clearly more conservative than algorithm 3.1.
4. Algorithm 2.2 (dotted lines in Fig.4a) has the worst performance. The price of extending the feasibility region without using a large computational load is that the performance criteria must be modified away from the most desirable one.
5. Algorithms (2.1, 3.1, 3.2) which do not require α to be monotonic have faster convergence of α; of these 3.2 has the slowest convergence. Algorithm 2.2 guarantees monotonicity of α and yet has the slowest convergence.

V. Conclusions

This paper has shown the potential benefits of using interpolation to generate predictive control algorithms as opposed to the more usual technique [13] of allocating indivdual control values as the d.o.f. With interpolation one can achieve: (i) larger feasibility regions for the same number of d.o.f./computational loading and (ii) performance that is surprisingly close to the global optimum with a far smaller on line computation.

The traditional weakness of interpolation algorithms is that it is less straightforward to give guarantees of stability and recursive stability. This paper derives algorithms such that: (i) by allowing the number of d.o.f. to be one greater than the state dimension, this problem is removed and (ii) even with just one d.o.f. one can give guarantees (within a more restricted region).

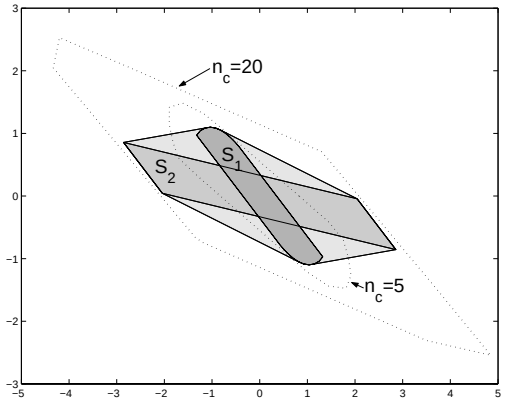

Fig. 1. Feasible regions for algorithms 2.1, 2.2, 3.2 and OMPC with $n_c = 5, 20$.

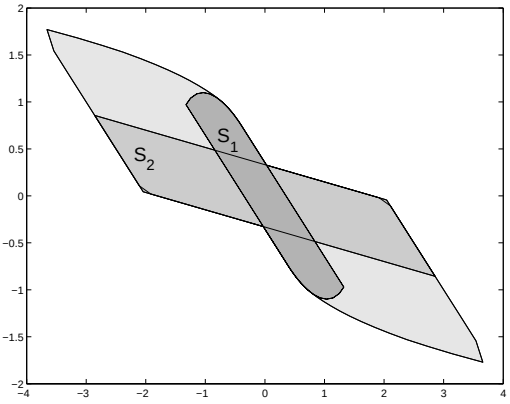

Fig. 2. Feasible regions for for algorithm 3.1.

Future work will look at the selection of the underlying controllers and dealing with disturbances/uncertainty.

REFERENCES

[1] F. Blanchini, Set invariance in control, Automatica, 35, 1747-1767, 1999.

[2] Clarke, D.W., C. Mohtadi and P.S. Tuffs (1987). Generalised predictive control, Parts 1 and 2, *Automatica*, **23**, pp. 137-160

[3] M.V. Kothare, V. Balakrishnan and M. Morari, Robust constrained model predictive control using linear matrix inequalities, Automatica, 32, 1361-79, 1996.

[4] Kouvaritakis, B., Rossiter, J.A., and Cannon, M., 1998, Linear quadratic feasible predictive control, *Automatica*, 34, , 1583-1592, 1998

[5] M. Bacic, M. Cannon, Y.I. Lee and B. Kouvaritakis, General interpolation in MPC and its advantages, Trans IEEE AC, 2003, 48, 6, 1092-1096

[6] D.Q. Mayne, J.B. Rawlings, C.V. Rao and P.O.M. Scokaert, Constrained model predictive control: stability and optimality, *Automatica*, 36, pp789-814

[7] J.A. Mendez, B. Kouvaritakis and J.A. Rossiter, State Space approach to interpolation in MPC, International journal of robust nonlinear control, 2000, 10, pp27-38

[8] E.G. Gilbert and K. T. Tan, 1991, Linear systems with state and control constraints: the theory and application of maximal output admissable sets, *IEEE Trans AC*, **36**, 9, pp1008-1020

[9] J.A. Rossiter, M.J.Rice, J. Schuurmanns and B. Kouvaritakis, A computationally efficient contrained predictive control law, American Control Conf., 1998.

[10] J.A. Rossiter and B. Kouvaritakis, Reducing computational load for LQ optimal predictive controllers, Proceedings UKACC, 606-611, 1998

[11] J.A. Rossiter, B. Kouvaritakis and M. Cannon, Stability proof for computationally efficient predictive control in the uncertain case, Proc. ACC, 2003.

[12] J.A. Rossiter, B. Kouvaritakis and M. Cannon, 2001, Computationally efficient algorithms for constraint handling with guaranteed stability and near optimality, *IJC*, 74, 17, 1678-1689

[13] Scokaert, P.O.M. and J. B. Rawlings (1998), Constrained linear quadratic regulation, *IEEE Trans AC*, 43, 8, pp1163-1168

[14] T.T.C. Tsang and D.W. Clarke, Generalised predictive control with input constraints, IEE Proceedings Pt. D, 6, 451-460, 1988.

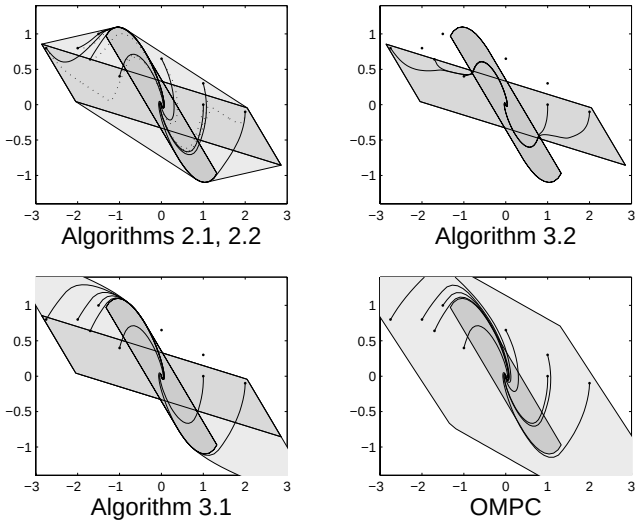

Fig. 3. State trajectories for different initial conditions.

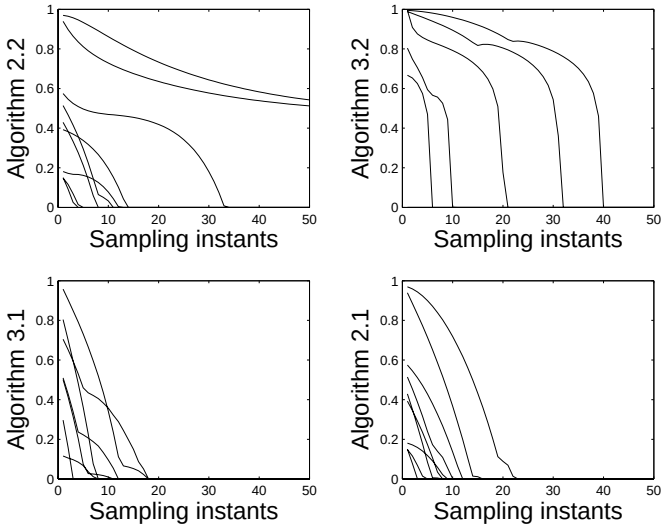

Fig. 4. Variation in α for different initial conditions.