15th Triennial World Congress, Barcelona, Spain

PREDICTIVE FUNCTIONAL CONTROL: MORE THAN ONE WAY TO PRESTABILISE

J.A. Rossiter *

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

Abstract: It is possible to prestabilise the predictions used within Predictive Functional Control in order to increase the likelihood of a stabilising control design. However, the minimal order approach to prestabilisation is not always a good basis for control design. This weakness is investigated and some non-minimal forms of prestabilisation are developed which are a much better basis for control.

Keywords: Predictive functional control, stability, prestabilisation, performance

1. INTRODUCTION

Predictive control is an intuitive control design method whereby one predicts the expected effects of differing control trajectories and then selects the trajectory which causes the most desirable expected behaviour. Such a procedure fits well with human based control and can lead to easier design. For instance such questions as over what horizon should one predict behaviour, what sort of future control trajectories should one use, can all be answered fairly easily. In academia there has been a tendency to link these decisions to optimal control (Scokaert *et al.*, 1998; Clarke *et al.*, 1987) as this enables one to use many well understood theoretical results. In particular apriori analysis of stability is straightforward e.g. (Kouvaritakis *et al.*, 1992; Rawlings *et al.*, 1993; Rossiter *et al.*, 1998). However,the link to optimal control takes the technique further away from engineering intuition which was key in early industrial variants of MPC (predictive control),e.g. (Cutler *et al.*, 1980; Richalet *et al.*, 1978). Here we focus on one commerical product (Richalet *et al.*, 1978) Predictive Functional Control (PFC) which has sought to relate controller design as much as possible to well understood engineering concepts. This simplification is at the expense of some potential optimality and power, nevertheless it has had extensive success in practice.

There is however some classes of problem for which PFC will often fail, that is unstable open-loop processes with factors $(s-a)/(s-ra)$, $r>1$ and multivariable processes. Here we concentrate on the former of these. Recent work (Rossiter, 2001; Rossiter, 2001b) has shown that it is possible to transcribe some of the work on guaranteed stability e.g. (Kouvaritakis *et al.*, 1992; Rossiter *et al.*, 1996) using prestabilisation before optimisation. Using prestabilised predictions one is able to stabilise processes that previously could not be handled with PFC and incorporate constraint handling and feasibility issues (Rossiter, 2002).

Suprisingly, although prestabilisation gives a good assurance of stability with simple tuning guidelines, it does not always give good performance. In fact (Rossiter, 2001c) at times a PFC algorithm based on unstable predictions gave better performance. The difficulty relates back to the prediction class adopted for doing prestabilisation. The method had followed common practice in predictive control of using changes in control as the degrees of freedom, however this may be ineffectual if the number of changes allowed is too small. In PFC the number of d.o.f. maybe just one which is insufficient for good performance in some

cases. The aim of this paper is develop alternative parameterisations of stabilising predictions which allow for good performance with just one d.o.f..

Here a brief summary of PFC algorithms based on prestabilised and non prestabilised predictions is given. A new parameterisation of prestabilised predictions is developed and a variety of examples will be used to contrast new and old approaches.

2. BACKGROUND

2.1 *The PFC algorithm*

We adopt the notation of y, u, r for process outputs, inputs and setpoint respectively. z^{-1} is the unit delay operator (i.e. $z^{-1}y_k = y_{k-1}$), y_k is the value of y at the kth sample and $y_{k+i|k}$ is the predicted value of y_{k+i} computed at sample k. PFC makes use of a system model to generate predictions of the process behaviour in terms of the current state and future inputs. Although more involved variants exist [1], to avoid over complicating this brief paper we concentrate on a PFC variant with just one d.o.f.. In PFC one chooses: (i) a lag (that is the time constant, say T_{PFC} and (ii) a single prediction horizon say T_h (denoted the coincidence horizon). The control move is selected to cause the predicted output to coincide with the response of a target 1st order lag T_h seconds ahead. Let T_h seconds correspond to n_y samples (i.e $n_yT = T_h$, T the sample period), then the online computation reduces to solving:

$$y_{k+n_y|k} = y_k + (r_{k+n_y} - y_k)(1 - e^{-\frac{T_h}{T_{PFC}}}) \quad (1)$$

2.2 *Independent models*

In PFC it is usual to use an IM (independent model) (Garcia *et al.*, 1982) for prediction. This can give significant improvements in sensitivity to measurement noise over the alternative of state realignment (Rossiter *et al.*, 2001). Also it is equivalent to a FIR model which is favoured in industry. The norm is to simulate the IM in parallel with the process, using the same inputs u. An IM is intended to represent the process as closely as possible so that it has matching inputs and outputs. If y_m is the output of the IM, in general, due to uncertainty, $y \neq y_m$.

With unstable processes a parallel simulation cannot work because the same input will not stabilise the IM and an uncertain plant. A typical solution e.g. (Richalet,) is to decompose the model into two parts (figures 1,2) where for a process G:

$$G = (I + M_2)^{-1}M_1 = \frac{n}{d} \quad (2)$$

[1] To cater for setpoints with high order dynamics

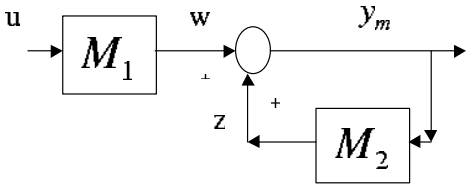

Fig. 1. Independent model used for prediction

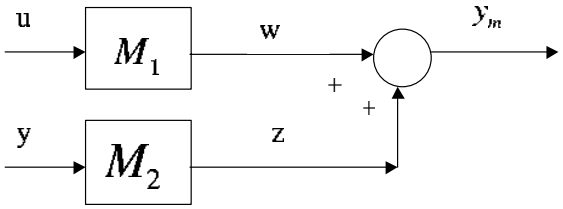

Fig. 2. Independent model for simulation

Both M_1 and M_2 are stable. Figure 1 is used for prediction and figure 2 for online simulation. A convenient decomposition in the SISO case is:

$$G = \frac{n_+n_-}{d_+d_-}; \quad M_1 = \frac{n_+}{d_-}; \quad M_2 = \frac{b_2}{n_-}; \quad b_2 = n_- - d_+ \quad (3)$$

n_+, d_+ are factors containing unstable roots.

Note: For ease of notation any process dead-time is assumed to be absorbed into n and n_+.

2.3 *Prestabilisation and prediction*

Using unstable predictions as a basis for a predictive control law design is unwise (Rossiter *et al.*, 1998). Even if the behaviour is predicted to be good within the horizon, it would be divergent thereafter and hence one can not make recursive feasibility claims (Kouvaritakis *et al.*, 1996) and instability is almost inevitable due to constraints. There is a need therefore to parameterise the degrees of freedom such that the predictions are stable. Here (Rossiter, 2001) we use the basic philosophy of (Rossiter *et al.*, 1996), that is place structure into the predicted future input trajectory to bring the unstable dynamics under control.

2.3.1. *Notation* Define vectors of future (arrow right) and past values (arrow left)

$$\Delta \underset{\rightarrow}{u} = \begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \vdots \\ \Delta u_{k+n_u-1} \end{bmatrix}; \quad \underset{\rightarrow}{y} = \begin{bmatrix} y_{k+1} \\ y_{k+2} \\ \vdots \\ y_{k+n_y} \end{bmatrix}$$

$$\Delta \underset{\leftarrow}{u} = \begin{bmatrix} \Delta u_{k-1} \\ \Delta u_{k-2} \\ \vdots \end{bmatrix}; \quad \underset{\leftarrow}{y} = \begin{bmatrix} y_k \\ y_{k-1} \\ \vdots \end{bmatrix}$$

The vector of future values can be any length, but n_y corresponds to the coincidence horizon and n_u the input horizon. Using the Toeplitz/Hankel

notation to compute predictions, for a given polynomial $n(z) = n_0 + n_1 z^{-1} + \ldots$, define

$$C_n = \begin{bmatrix} n_0 & 0 & 0 & \ldots \\ n_1 & n_0 & 0 & \ldots \\ n_2 & n_1 & n_0 & \ldots \\ \vdots & \vdots & \vdots & \vdots \\ n_m & n_{m-1} & n_{m-2} & \vdots \end{bmatrix} \quad (4)$$

$$H_n = \begin{bmatrix} n_1 & \ldots & n_{m-1} & n_m \\ n_2 & \ldots & n_m & 0 \\ \vdots & \vdots & \vdots & \vdots \\ n_m & 0 & \vdots & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Also define Γ_n as a tall and thin submatrix of C_n so that $[1, z^{-1}, z^{-2}...]\Gamma_n b = n(z)[1, z^{-1}, ...]b$ and note that dimensions are flexible to fit the context.

2.3.2. *Open-loop predictions* Set up consistency conditions around M_1 and M_2 at each future sample and solve as simultaneous equations. The predictions are given by (Rossiter, 2001):

$$\underset{\rightarrow}{y} = H\Delta\underset{\rightarrow}{u} + K_u\underset{\leftarrow}{u} + K_w\underset{\leftarrow}{w} + K_y\underset{\leftarrow}{y} + K_z\underset{\leftarrow}{z} \quad (5)$$

where L is a vector of ones, $\Delta = 1 - z^{-1}$ and

$$P = C_{d_-}^{-1}C_\Delta^{-1}C_{d_+}^{-1}; \quad H = P\Gamma_n$$

$$\begin{aligned} K_u &= PC_\Delta(C_nL + C_{n_-}H_{n_+}) \\ K_w &= -PC_\Delta(C_{n_-}H_{d_-} + C_{d_-}C_{b_2}L) - L \\ K_y &= PC_\Delta C_{d_-}(C_{b_2}L + H_{b_2}) + L \\ K_z &= -PC_\Delta C_{d_-}(C_{b_2}L + H_{n_-}) - L \end{aligned}$$

2.3.3. *Prestabilised predictions* Parameterisation (Rossiter, 2001) of future inputs to stabilise the predictions of (5) reduces to the constraint that $\Delta\underset{\rightarrow}{u}$ be selected such that

$$P^{-1}\underset{\rightarrow}{y} = \Gamma_{d_+}\gamma \quad (6)$$

with γ stable. With the minimal order $\Delta\underset{\rightarrow}{u}$ condition (6) can be solved analytically via a suitable diophantine equation, (Rossiter, 2001). The resulting predictions take the form

$$\begin{bmatrix} \Delta\underset{\rightarrow}{u} \\ \underset{\rightarrow}{y} \end{bmatrix} = \left[\begin{array}{c|c|c|c} C_u & C_w & C_y & C_z \\ \hline C_{yu} & C_{yw} & C_{yy} & C_{yz} \end{array}\right] \begin{bmatrix} \underset{\leftarrow}{u} \\ \underset{\leftarrow}{w} \\ \underset{\leftarrow}{y} \\ \underset{\leftarrow}{z} \end{bmatrix} + \begin{bmatrix} \Gamma_{d_+} \\ H_1 \end{bmatrix} \underset{\rightarrow}{c} \quad (7)$$

where $H_1 = [C_\Delta C_{d_-}]^{-1}\Gamma_n$.

Remark 2.1. Let r be the number of unstable poles, then with a minimal order solution for $\Delta\underset{\rightarrow}{u}$, the matrices C_u, C_w, C_y, C_z will have at most r non-zero rows, the first r rows. Also, for one d.o.f. $\underset{\rightarrow}{c} = c_k$, Γ_{d_+} will have $r+1$ rows.

2.4 *The PFC Algorithm*

Here we give the PFC algorithms (solution of equations (1)) based on open-loop and prestabilised predictions. Define the the variable

$$\Psi = e^{-\frac{T}{T_{PFC}}}; \quad \Psi^{n_y} = e^{-\frac{T_h}{T_{PFC}}} \quad (8)$$

and define $\mathbf{e}_{n_y}^T$ to be the n_y^{th} standard basis vector.

Algorithm 2.1. Open-loop predictions: Assume $\Delta\underset{\rightarrow}{u} = \Delta u_k$ and substitute predictions (5,8) into (1):

$$\begin{aligned} y_{k+n_y|k} &= \mathbf{e}_{n_y}^T[H\Delta\underset{\rightarrow}{u} + K_u\underset{\leftarrow}{u} + K_w\underset{\leftarrow}{w} + K_y\underset{\leftarrow}{y} + K_z\underset{\leftarrow}{z}] \\ &= (1-\Psi^{n_y})r_{k+n_y} + \Psi^{n_y}y_k \\ \Rightarrow \quad \Delta u_k &= P_r r_{k+n_y} - P_u\underset{\leftarrow}{u} - P_w\underset{\leftarrow}{w} - P_y\underset{\leftarrow}{y} - P_z\underset{\leftarrow}{z} \end{aligned} \quad (9)$$

$g = (\mathbf{e}_{n_y}^T H)^{-1}$, $P_u = -gK_u$, $P_w = -gK_w$, $P_y = [\Psi^{n_y}, 0, ...] - gK_y$, $P_z = -gK_z$, $P_r = g(1-\Psi^{n_y})$.

Algorithm 2.2. Prestabilised predictions: Assume $\underset{\rightarrow}{c} = c_k$ and substituting (7, 8) into (1) implies

$$\begin{aligned} y_{k+n_y|k} &= \mathbf{e}_{n_y}^T[H_1c_k + C_{yu}\underset{\leftarrow}{u} + C_{yw}\underset{\leftarrow}{w} + C_{yy}\underset{\leftarrow}{y} + C_{yz}\underset{\leftarrow}{z}] \\ &= (1-\Psi^{n_y})r_{k+n_y} + \Psi^{n_y}y_k \\ \Rightarrow \ \mathbf{e}_{n_y}^T H_1 c_k &= r_{k+n_y}(1-\Psi^{n_y}) + y_k\Psi^{n_y} \\ &\quad -\mathbf{e}_{n_y}^T[C_{yu}\underset{\leftarrow}{u} + C_{yw}\underset{\leftarrow}{w} + C_{yy}\underset{\leftarrow}{y} + C_{yz}\underset{\leftarrow}{z}] \\ \Rightarrow \quad \Delta u_k &= P_r r_{k+n_y} - P_u\underset{\leftarrow}{u} - P_w\underset{\leftarrow}{w} - P_y\underset{\leftarrow}{y} - P_z\underset{\leftarrow}{z} \end{aligned} \quad (10)$$

where $g = (\mathbf{e}_{n_y}^T H_1)^{-1}$, $h = \mathbf{e}_1^T\Gamma_{d_+}$, $P_u = hg\mathbf{e}_{n_y}^T C_{yu} - \mathbf{e}_1^T K_u$, $P_w = hg\mathbf{e}_{n_y}^T C_{yw} - \mathbf{e}_1^T K_w$, $P_y = hg\mathbf{e}_{n_y}^T C_{yy} - \mathbf{e}_1^T K_y$, $P_z = hg\mathbf{e}_{n_y}^T C_{yz} - \mathbf{e}_1^T K_z$.

3. THE WEAKNESS OF PRESTABILISATION

Although (Rossiter, 2001c) prestabilisation increases the likelihood of a stable closed-loop, the resulting control law may give poor performance. Next we discuss what causes this poor performance and suggest a means of overcoming it.

3.1 *Recursive feasibility and* the tail

One needs to understand how stability can be established for a predictive control algorithm. Apriori proofs e.g. (Kouvaritakis *et al.*, 1992), (Rawlings *et al.*, 1993) often use the concept of *the tail*. Let the control trajectory at time k be

$$\Delta\underset{\rightarrow}{u} = [\Delta u_k, \Delta u_{k+1|k}, \Delta_{k+2|k}, ...]^T \quad (11)$$

Now at the following sampling instant, the unused part (all except Δu_k) can be described as *the tail*:

$$\textit{the tail} = [\Delta u_{k+1|k}, \Delta_{k+2|k}, ...]^T \quad (12)$$

A sufficient condition for many Lyapunov based apriori stability proofs (assuming convergence of the predictions) is that *the tail* is in the class of predictions allowed at the new sampling instant, for the nominal case e.g. (Kouvaritakis *et al.*,

1992; Scokaert *et al.*, 1998) as then one can always choose a control trajectory (i.e. *the tail*) such that predicted performance is not worse than at the previous sampling instant. With new d.o.f. one should be able to improve predicted performance.

3.2 *Prestabilised PFC and* the tail

The prestabilised predictions (Rossiter, 2001) of (7) have the property that the predictions now, include *the tail* (by setting $c_k = 0$). This can be deduced (see remark 2.1) as setting $c_k = 0$ gives the minimal order solution for $\Delta \underset{\rightarrow}{u}$ that ensures stable prediction. When $c_k \neq 0$ the solution order is augmented by one; the extra order allowing the degrees of freedom in the predictions. *The tail* is one order less than (7) and therefore must coincide with the minimal order solution.

Herein however lies the weakness of predictions (7). The basis is a minimal order control trajectory that stabilises the process. Add onto this a desired set point and a large coincidence horizon[2] then the PFC algorithm finds a minimal order trajectory to stabilise the error dynamics. In essence this becomes equivalent to dead-beat control of the unstable dynamics. In some cases such a dead-beat action will be too agressive. Moroever, the control performance potential is very much linked to the prediction class used, so no amount of tuning with Ψ, n_y will remove this problem. The examples section will illustrate these comments. It should be noted that in conventional predictive control with prestabilised predictions e.g. (Kouvaritakis *et al.*, 1992; Rawlings *et al.*, 1993), overtuning (dead-beat behaviour) is avoided by allowing more degrees of freedom.

4. IMPROVING PRESTABILISATION

The presence of *the tail* in the prediction class is essential for straightforward stability analysis. However, using minimal order predictions with this property may give rather agressive control. In practice, one expects the closed-loop responses to be high order and smooth. In consequence when defining a prediction class for predictive control e.g. (Kouvaritakis *et al.*, 1998), (Scokaert *et al.*, 1998), (Rossiter *et al.*, 1996) one aims to ensure stability but also smoothness. In other words one uses higher order stabilising solutions such as the optimal predicted closed-loop performance. It is still straightforward to include *the tail* in a higher order prediction class (e.g. (Scokaert *et al.*, 1998; Kouvaritakis *et al.*, 1998)) and to restrict the number of degrees of freedom while maintaining the benefits.

[2] The work in (Rossiter, 2001c) indicated a need for large coincidence horizons with prestabilised predictions

The difficulty within PFC is that we do not want to fall back on optimal control results as in conventional MPC and hence it is not so straightforward to compute an ideal closed-loop response which can be used as a base. Here we propose an alternative approach.

4.1 *Stabilising prediction classes*

First, to maximise the likelihood of closed-loop stability, ensure the prediction class is parameterised so that the NDOF (no d.o.f.) solution contains *the tail*. Let all stabilising future input trajectories be parameterised in general terms as:

$$\Delta \underset{\rightarrow k}{u} = Mv_k + \Gamma_f \underset{\rightarrow}{c} = \begin{bmatrix} \Delta u_k \\ \Delta u_{k+1|k} \\ \vdots \end{bmatrix} \tag{13}$$

where notionally M, Γ_f have an infinite number of rows, f contains a factor d_+ and $\underset{\rightarrow}{c}$ contains degrees of freedom (here $\underset{\rightarrow}{c} = c_k$). Inclusion of *the tail* in $\Delta \underset{\rightarrow k+1}{u}$ with $c_{k+1} = 0$ implies

$$\Delta \underset{\rightarrow k+1}{u} = Mv_{k+1} = \begin{bmatrix} \Delta u_{k+1|k+1} \\ \Delta u_{k+2|k+1} \\ \vdots \end{bmatrix} = \begin{bmatrix} \Delta u_{k+1|k} \\ \Delta u_{k+2|k} \\ \vdots \end{bmatrix} \tag{14}$$

It is clear therefore that M, f must be defined in a mutually compatible way (for instance they must share common poles) to ensure (14) can hold. For instance the part of $\Delta \underset{\rightarrow k}{u}$ depending on f at sampling instant k must appear through the term Mv_{k+1} at the next sampling instant (even though in principle for the prediction class (13) f can be any function with a factor d_+ and M can be defined based on any stabilising trajectories).

4.2 *Definition of a prediction class*

Here we select a prediction class which is compatible with the underlying need in PFC for simplicity and without direct links to optimal control. Other prediction classes do exist and are the subject of ongoing research. The key desire is to choose a trajectory class with three properties:

- Stabilising
- Smooth (transfer function, not polynomial)
- The degrees of freedom can be introduced while allowing *the tail* in the NDOF solution.

The second of these requirements implies poles in the prediction class; an 'optimal' choice of these is ongoing research, however a good default solution is known to be the inverse of the unstable poles. Hence define $\hat{d}_+$ with roots p such that $d_+(1/p) = 0$. Consider prediction class (5), this can be represented in transfer function form as

$$\underset{\rightarrow}{y}(z) = \frac{n(z)\Delta \underset{\rightarrow}{u} + [1, z^{-1}, ...]Kv}{d_+ d_- \Delta} \tag{15}$$

$K = P^{-1}[K_u, K_w, K_y, K_z]$, $v = [\underset{\leftarrow}{u}^T, \underset{\leftarrow}{w}^T, \underset{\leftarrow}{y}^T, \underset{\leftarrow}{z}^T]^T$. Hence a requirement for stable prediction is that

$$n(z)\Delta \underset{\rightarrow}{u} + [1, z^{-1}, ...]Kv = d_+\gamma; \quad \Rightarrow \quad \underset{\rightarrow}{y} = \frac{\gamma}{d_-\Delta} \tag{16}$$

with γ stable. Next add in the requirement for poles $\hat{d}_+$ to be in $\Delta \underset{\rightarrow}{u}$ to give condition

$$n(z)\frac{\Delta \underset{\rightarrow}{\hat{u}}}{\hat{d}_+} + [1, z^{-1}, ...]Kv = d_+\frac{\hat{\gamma}}{\hat{d}_+};\ \Delta \underset{\rightarrow}{u} = \frac{\Delta \underset{\rightarrow}{\hat{u}}}{\hat{d}_+},\ \gamma = \frac{\hat{\gamma}}{\hat{d}_+} \tag{17}$$

Rearranging this gives the diophantine identity

$$n(z)\Delta \underset{\rightarrow}{\hat{u}} + \hat{d}_+[1, z^{-1}, ...]Kv = d_+\hat{\gamma} \tag{18}$$

Eqn.(18) is easy to solve; the minimal order solution for $\Delta \underset{\rightarrow}{\hat{u}}$ (this corresponds to the NDOF solution for $\Delta \underset{\rightarrow}{u}$ - the order of $\hat{\gamma}$ is unimportant) implies matrices $\hat{K}_1$, $\hat{K}_2$ such that

$$\Delta \underset{\rightarrow}{\hat{u}} = \hat{K}_1 v, \quad \hat{\gamma} = \hat{K}_2 v \tag{19}$$

As noted in Remark 2.1, K_1 will have at most r non-zero rows. The whole class of solutions, ensuring that f also has poles $\hat{d}_+$ takes the form

$$\Delta \underset{\rightarrow}{\hat{u}} = \hat{K}_1 v + \Gamma_{d_+} c_k; \quad \Delta \underset{\rightarrow}{u} = \Gamma_{\hat{d}_+^{-1}}[\hat{K}_1 v + \Gamma_{d_+} c_k] \tag{20}$$

Algorithm 4.1. New prestabilised predictions: Substitute input trajectory (20) into (15) to find the output predictions and select c_k to ensure coincidence as in (1). Use this c_k to compute Δu_k.

Remark 4.1. Because $\Delta \underset{\rightarrow}{\hat{u}}$ is taken to be the minimal order solution of (18), selecting $c_k = 0$ ensures that $\Delta \underset{\rightarrow k}{u}$ is *the tail* of $\Delta \underset{\rightarrow k-1}{u}$.

5. COMPARISON OF ALGORITHMS

In this section we will illustrate how the new prediction class improves on the performance obtainable with the old prediction class of (Rossiter, 2001c). We will use the same examples as in (Rossiter, 2001c), all of which have unstable poles.

5.1 *Examples used*

Several processes with different types of unstable poles/zeros are trialed.

- Example 1 (unstable pole at $z = 1.5$, $\Psi = 0.7$, $n_y = 12$).

$$G(z) = \frac{z^{-1} - 0.3z^{-2}}{1 - 1.9 + z^{-1}0.48z^{-2} + 0.18z^{-3}}$$

- Example 2 (unstable pole $z \approx 1.49$, unstable zero $z \approx 1.22$). $\Psi = 0.7$, $n_y = 12$.

$$G(z) = \frac{0.2126z^{-1} - 0.2594z^{-2}}{1 - 2.3967z^{-1} + 1.3499z^{-2}}$$

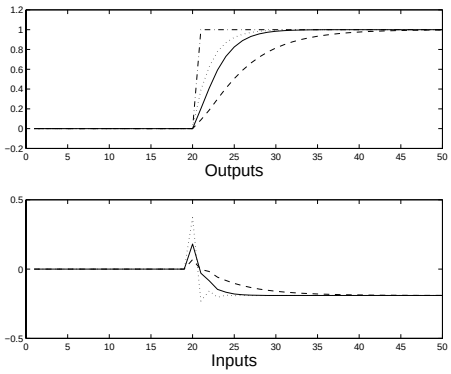

Fig. 3. Simulations for example 1

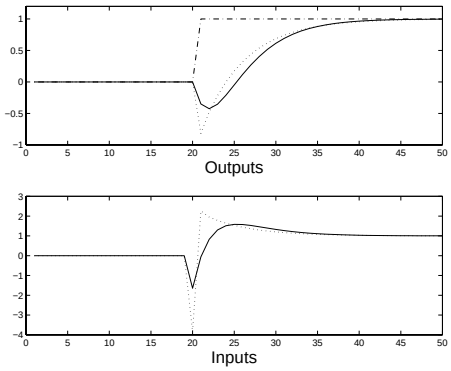

Fig. 4. Simulations for example 2

- Example 3 (unstable pole $z \approx 1.22$, unstable zero $z \approx 1.35$). $\Psi = 0.7$, $n_y = 12$

$$G(z) = \frac{0.18z^{-1} - 0.2432z^{-2}}{1 - 2.1262z^{-1} + 1.1052z^{-2}}$$

- Example 4 (2 unstable poles $z = 1.2068 \pm 0.1885i$). $\Psi = 0.7$, $n_y = 2$ ($n_y = 8$ for algorithm 2.2, unstable for lower n_y).

$$G(z) = \frac{0.2661z^{-1} - 0.2172z^{-2}}{1 - 2.4136z^{-1} + 1.4918z^{-2}}$$

5.2 *Simulation examples*

Typical closed-loop simulations are presented for examples 1-4 in figures 3-6 respectively. Dashed, dotted, solid and dash-dot lines are used for algorithms 2.1, 2.2, 4.1 and the reference respectively. Algorithm 2.1 cannot stabilise example 2 and algorithm 2.2 is displayed in figure 7 for example 4 as its best performance is so poor.

Overall algorithm 4.1 is the best - it has performed well on all the examples. Algorithm 2.1 has not stabilised example 2 and in fact can be hard to tune (Rossiter, 2001c) although when tunable its performance is OK (figures 3,5,6). Algorithm 2.2 is easy to tune to give stable responses (Rossiter, 2001c) which are usually satisfactory (figs 3-5) though it could be argued a little too jerky. However for some examples (e.g. figure 7) it cannot be tuned satisfactorily at all. The advantage of algorithm 4.1 is that it appears to perform

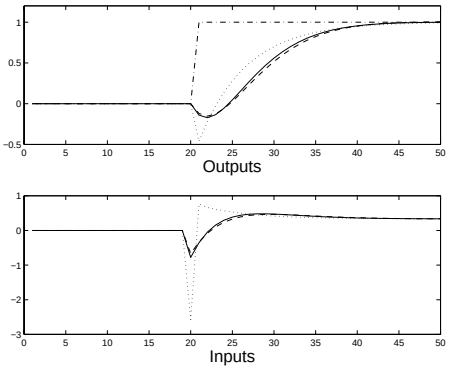

Fig. 5. Simulations for example 3

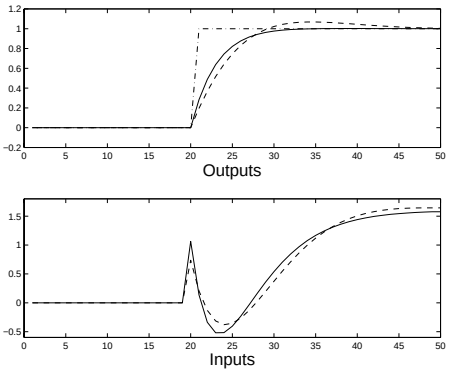

Fig. 6. Simulations for example 4

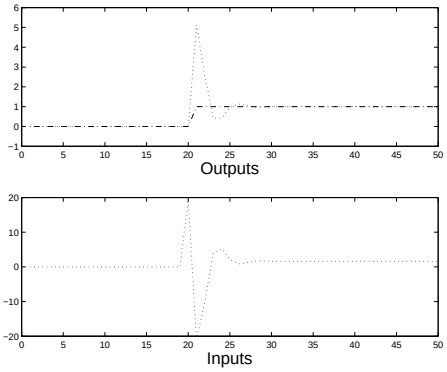

Fig. 7. Simulations for example 4

as well as and often better than both algorithms 2.1,2.2 and hence avoids the need for algorithm selection. Future work will look more carefully at whether this algorithm does indeed give easier tuning in general.

6. CONCLUSIONS

It is clear that the variants of PFC based on unstable predictions and prestabilised predictions have different strengths and weaknesses. It would be desirable to have a single algorithm that performed well in all scenarios, but was still simple. This paper has proposed a modification to the class of predictions which allows the definition of such an algorithm. Its superior performance is illustrated by examples.

7. REFERENCES

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

Cutler, C.R. and B.L. Ramaker (1980), Dynamic matrix control - a computer control algorithm, Proc. ACC, San Fransisco

Garcia, C.E. and M. Morari (1982), Internal Model control 1. A unifying review and some new results, I&EC Process Design and Development, 21, pp308-323

Kouvaritakis, B., J.A. Rossiter and A.O.T.Chang (1992), Stable Generalized predictive control: an algorithm with guaranteed stability, Proc IEE, 139, No.4, pp349-362

Kouvaritakis, B., J.R. Gossner and J.A. Rossiter (1996), Apriori stability condition for an arbitrary number of unstable poles, Automatica, Vol. 32, No.10, pp. 1441-1446

Kouvaritakis, B., J.A. Rossiter and M. Cannon (1998), Linear quadratic feasible predictive control, Automatica, 34, 12, pp1583-1592

Mosca, E. and J. Zhang (1992), Stable redesign of predictive control, Automatica, 28, pp1229-1233

Rawlings, J.B. and K.R. Muske (1993), The stability of constrained receding horizon control, Trans IEEE AC, 38, pp1512-1516

Richalet, J., A. Rault, J.L. Testud and J. Papon (1978), Model predictive heuristic control: applications to industrial processes, Automatica, 14, 5, pp413-428

Richalet, J., Commande predictive, R 7 423

Rossiter, J.A., J.R. Gossner and B. Kouvaritakis (1996) Infinite horizon stable predictive control, Trans. IEEE AC, 41, 10, pp1522-1527.

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

Rossiter, J.A. and J. Richalet (2001), Re-aligned models for prediction in MPC: a good thing or not ? APC6 (York)

Rossiter, J.A., Stable prediction for unstable independent models, submitted

Rossiter, J.A., (2001b), Predictive functional control of unstable processes, Report no. 807, University of Sheffield.

Rossiter, J.A., (2001c) Predictive Functional Control: to prestabilise or not ?, Internal Report 810, University of Sheffield.

Rossiter, J.A., (2002), Handling constraints with predictive functional control of unstable processes, Proc. ACC 2002

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