This is a repository copy of Predictive Functional Control: More Than One Way To Pre-stabilise.

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/84451/

Monograph:

Reuse
Unless indicated otherwise, fulltext items are protected by copyright with all rights reserved. The copyright exception in section 29 of the Copyright, Designs and Patents Act 1988 allows the making of a single copy solely for the purpose of non-commercial research or private study within the limits of fair dealing. The publisher or other rights-holder may allow further reproduction and re-use of this version - refer to the White Rose Research Online record for this item. Where records identify the publisher as the copyright holder, users can verify any specific terms of use on the publisher's website.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.
Predictive functional control: more than one way to prestabilise.

by J A Rossiter

RESEARCH REPORT NO. 809

October 2001
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 commercial 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, 2001c).

Surprisingly, although prestabilisation gives a good assurance of stability with simple tuning guidelines, it does not always give good performance. In fact (Rossiter, 2001) 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 to 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 pre-stabilised and non pre-stabilised predictions is given. A new parameterisation of pre-stabilised 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. \( \tau^{-1} \) is the unit delay operator (i.e. \( \tau^{-1}y_k = y_{k-1} \)), \( y_k \) is the value of \( y \) at the \( k \)th sample and \( y_{k+1|k} \) is the predicted value of \( y_{k+1} \) 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 \( T_{PPC} \)) 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 \( T_h \) samples correspond to \( n_y \) samples (i.e. \( n_yT = T_h \), \( T \) the sample period), then the following computation reduces to solving:

\[
y_{k+n_y} = y_k + (1 - e^{-\frac{T_h}{T_{PPC}}})
\]  

(1)

2.2 Independent model

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 \( G \):

\[
G = (I + M_2)^{-1} M_1 = \frac{n}{d}
\]  

(2)

Fig. 1. Independent model used for prediction

![Fig. 1. Independent model used for prediction](image1)

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:

\[
G = \frac{n_+ n_-}{d_+ d_-}; \quad M_1 = \frac{n_+}{d_+}; \quad M_2 = \frac{b_2}{n_-}; \quad b_1 = n_- d_+
\]  

(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 Pre-stabilisation and prediction

Using unstable predictions as a basis for a pre-stabilising control law design is unwise (Rossiter et al., 1997). 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., 1997), 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)

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

\[
\Delta y_s = \begin{bmatrix} \Delta u_{k-1} \\ \Delta u_{k-2} \\ \vdots \\ \Delta u_{k-n_u-1} \end{bmatrix}; \quad y_s = \begin{bmatrix} y_k \\ y_{k-1} \\ \vdots \\ y_{k-n_u} \end{bmatrix}
\]

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

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

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

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

2.3.2. Open-loop predictions Set up consistency conditions around $M_0$ and $M_2$ at each future sample and solve as simultaneous equations. The predictions are given by (Rossiter, 2001):

$$y = H \Delta u + K_u y + K_w w + K_y y + K_z z$$  \hspace{1cm} (5)$$

where $L$ is a vector of ones, $\Delta = 1 - z^{-1}$ and $P = C_{d,1} C_{\Delta}^{-1} C_{d,1}^T$, $H = PT \Gamma_n$

$$K_u = PC_{\Delta}(C_n L + C_n H_{n+1})$$

$$K_w = -PC_{\Delta}(C_n H_{d,1} + C_{d,1} C_{b,2} L) - L$$

$$K_y = PC_{\Delta} C_{d,1}(C_{b,2} L + H_{b,2}) + L$$

$$K_z = -PC_{\Delta} C_{d,1}(C_{b,2} L + H_{n+1}) - L$$

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

$$P^{-1} y = \Gamma_{d+} \gamma$$  \hspace{1cm} (6)$$

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

$$\begin{bmatrix} \Delta u \\ y \\ \end{bmatrix} = \begin{bmatrix} C_u & C_w & C_y & C_z \\ C_{u,2} & C_{u,2} & C_{y,2} & C_{y,2} \\ C_{y,2} & C_{y,2} & C_{y,2} & C_{y,2} \\ \end{bmatrix} \begin{bmatrix} u \\ w \\ y \\ z \\ \end{bmatrix} + \begin{bmatrix} \Gamma_u y \\ H_1 \end{bmatrix} \gamma$$  \hspace{1cm} (7)$$

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

Remark 2.1. Let $\tau$ be the number of unstable poles, then with a minimal order solution for $\Delta u$, the matrices $C_u, C_w, C_y, C_z$ will have at most $7$ non-zero rows, the first $\tau$ rows. Also, for one d.o.f.

$$\gamma = e_{c,1}, \Gamma_{d,1}$$ will have $\tau + 1$ rows.

2.4 The PPC Algorithm

Here we give the PPC algorithm (solution of equations (1)) based on open-loop and prestabilised predictions. Define the the variable $\Psi = e^{-T \theta}$ and define $\theta_n$ to be the $n^th$ standard basis vector.

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

$$y_{k+n+1} = e_n y_{k}$$

$$\Rightarrow \Delta u_k = P_t r_k \Delta u_k - P_{u,u} - P_{u,w} - P_{y,y} - P_{z,z}$$  \hspace{1cm} (9)$$

$$\psi_n = e^{-T \theta_n}$$

Algorithm 2.2. Prestabilised predictions: Assume $\gamma = \gamma_t$ and substituting (7,8) into (1) implies

$$y_{k+n+1} = e_n y_{k}$$

$$\Rightarrow e_n y_{k+1} = e_n y_{k} + e_n \theta + e_n \gamma$$

$$\Rightarrow e_n y_{k+1} = e_n y_{k} + e_n \psi_n$$

$$\Rightarrow e_n y_{k+1} = e_n y_{k} + e_n \psi_n$$

$$\Rightarrow e_n y_{k+1} = e_n y_{k} + e_n \psi_n$$

$$\Rightarrow e_n y_{k+1} = e_n y_{k} + e_n \psi_n$$

3. THE WEAKNESS OF PRESTABILISATION

Although (Rossiter, 2001) 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

$$\Delta u = [\Delta u_k, \Delta u_{k+1}|k, \Delta u_{k+2}|k, \ldots]^T$$  \hspace{1cm} (11)$$

Now at the following sampling instant, the unused part (all except $\Delta u_k$) can be described as the tail:

$$\text{the tail} = [\Delta u_{k+1}|k, \Delta u_{k+2}|k, \ldots]^T$$  \hspace{1cm} (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.,
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 \( \Delta y \) 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 aggressive. Moreover, the control performance potential is very much linked to the prediction class used, so no amount of tuning with \( \Psi \), \( n_p \) 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 aggressive 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.

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:

\[
\Delta u_{k} = M u_k + \Gamma f \leq \begin{bmatrix}
\Delta u_k \\
\Delta u_{k+1|k} \\
\vdots
\end{bmatrix}
\]

where notionally \( M \), \( \Gamma f \) have an infinite number of rows, \( f \) contains a factor \( d_k \) and \( \Delta \) contains degrees of freedom (here \( \Delta = c_k \)). Inclusion of the tail in \( \Delta u_{k+1|k} \) with \( c_{k+1|k} = 0 \) implies

\[
\Delta u_{k+1} = M u_{k+1} = \begin{bmatrix}
\Delta u_{k+1|k+1} \\
\Delta u_{k+2|k+1} \\
\vdots
\end{bmatrix} = \begin{bmatrix}
\Delta u_{k+1|k} \\
\Delta u_{k+2|k} \\
\vdots
\end{bmatrix}
\]

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 \( \Delta u_{k} \) depending on \( f \) at sampling instant \( k \) must appear through the term \( M u_{k+1} \) at the next sampling instant (even though in principle for the prediction class (13) \( f \) can be any function with a factor \( d_k \) 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 \( d_k \) with roots \( p \) such that \( d_k(1/p) = 0 \). Consider prediction class (5), this can be represented in transfer function form as

\[
y(z) = \frac{n(z)\Delta y + [1, z^{-1}, ...]Kv}{d_+d_-\Delta}
\]

\(^2\) The work in (Rossiter, 2001) indicated a need for large coincidence horizons with prestabilised predictions.
\[ K = P^{-1}[K_u, K_w, K_y, K_z], \nu = [y^T, u^T, z^T]^T. \]

Hence a requirement for stable prediction is that
\[ n(z)\Delta u + [1, z^{-1}, ...]Kv = d_4 \gamma; \quad \Rightarrow \quad \gamma = \frac{\Delta u}{d_4} \Delta \]
(16)

with \( \gamma \) stable. Next add in the requirement for poles \( d_4 \) to be in \( \Delta u \) to give condition
\[ n(z)\Delta \hat{u} + [1, z^{-1}, ...]Kv = d_4 \gamma; \quad \Delta \hat{u} = \frac{\Delta u}{d_4}, \quad \gamma = \frac{\Delta \hat{u}}{d_4} \]
(17)

Rearranging this gives the diophantine identity
\[ n(z)\Delta \hat{u} + d_4 [1, z^{-1}, ...]Kv = d_4 \gamma \]
(18)

Eqn.(18) is easy to solve; the minimal order solution for \( \Delta \hat{u} \) (this corresponds to the NDOF solution for \( \Delta u \) - the order of \( \gamma \) is unimportant) implies matrices \( \hat{K}_1, \hat{K}_2 \) such that
\[ \Delta \hat{u} = \hat{K}_1 v, \quad \gamma = \hat{K}_2 v \]
(19)

As noted in Remark 2.1, \( \hat{K}_1 \) will have at most \( r \) non-zero rows. The whole class of solutions, ensuring that \( f \) also has poles \( d_4 \) takes the form
\[ \Delta \hat{u} = \hat{K}_1 v + \Gamma_{d_4} c_k; \quad \Delta \hat{u} = \Gamma_{d_4} [\hat{K}_1 v + \Gamma_{d_4} c_k] \]
(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 \( \Delta u_k \).

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

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, 2001). We will use the same examples as in (Rossiter, 2001), all of which have unstable poles.

5.1 Examples used

Several processes with different types of unstable poles/zeroes 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 \( z \approx 1.22 \)). \( \Psi = 0.7, n_y = 12 \).
  \[ G(z) = \frac{0.2126z^{-1} - 0.2594z^{-2}}{1 - 2.3967z^{-1} + 1.3499z^{-2}} \]

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, 2001) although when tunable its performance is OK (figures 3,5,6). Algorithm 2.2 is easy to tune to give stable responses (Rossiter, 2001) 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
6. CONCLUSIONS

It is clear that the variants of PFC based on unstable predictions and prestabilized 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

Richalet, J., Commande predictive, R 7 423
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