Stable prediction for unstable independent models

by J A Rossiter

RESEARCH REPORT NO. 812

October 2001

Stable prediction for unstable independent models

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

Abstract

The predictive control developments in the literature have often been based on realigned models whereas industrial packages have made much use of independent models. The transferal of many of the stability results to the independent model case is not straightforward; this paper develops one possible means of achieving this. The necessity of the developments is demonstrated for unstable open-loop processes.

1 Introduction

Predictive control is by now a popular strategy [4] and has acheived significant takeup in industry [8, 2]. However, there are still significant differences between popular industrial variants and those appearing in the academic literature. One noticeable difference is that the literature tends to make use of state-space or transfer function models whereas industrial packages are often based on FIR (Finite impulse response) models. The implication of this is that some strong stability results that have appeared in the literature (e.g.[5, 6, 7, 12, 14, 15]) need significant reworking to be applicable, even though the general principles of the proofs still apply. However, such reworking is rarely supplied in the literature. This paper considers one issue where transferal of the stability results is not straightforward and develops a systematic base for such a transferal.

There are a few principal components in predictive control algorithms [1]. These are a prediction model, a performance index, some tuning parameters and an online computation to find the current control. The computation[1] involves optimising the performance index, with respect to prespecified degrees of freedom in the prediction model. The success of such a strategy usually depends upon using a sensible performance index, sensible values for the tuning parameters and having predictions that are accurate enough. A poor choice of performance index can give either overtuned (not robust) control or undertuned control, a poor choice of tuning parameters likewise and inaccurate predictions due to model uncertainty can cause inappropriate control actions or even instability.

In addition to these points, there is another important but perhaps less well understood issue, that is the need to parameterise the degrees of freedom appropriately for the process to be controlled.

[1]Some variants such as PFC (Predictive functional control) do not involve optimisation

For instance a common policy (e.g [1, 2, 7]) is to parameterise the degrees of freedom as the changes in control, but usually in industrial applications, for computational reasons, only one or two control changes are allowed in the predictions even though more would be desirable [15]. More recent work in the literature on apriori stability results has shown that one can restrict the degrees of freedom in the future output trajectories or equivalently place more structure into the input trajectories in order to ensure nominal stability at least. The key concept adopted is to force the predictions to behave in a convergent fashion beyond a certain point; such a convergence is assured by either an implied stabilising feedback or by forcing the predicted state into a stable manifold. The benefit of this is that the costing horizon can be taken to infinity and then stability is established from a simple Lyapunov analysis.

As stated at the outset however, the above works have concentrated on the use of state-space and transfer function models. This implies the use of state realignment in prediction [13] whereby one assumes that one can initialise a model with the process measurements and then use this in prediction. With the exception of [3], the use of realigned models for prediction is widespread in the literature, however it has been noted [13] that this does not always give the best predictions. In fact this is implicitly recognised in many industrial packages such as DMC and PFC where much use has been made of FIR representations which unsuprisingly are often less sensitive to measurement noise. More recently, the PFC algorithm has adopted independent models (IM) [3] in place of FIR models. In fact for prediction the use of an IM is equivalent to the use of an FIR model [13]. However IM have the advantage of requiring less parameters and also there is no implied truncation error.

The apparent weakness of IM (or FIR models) is that they do not easily take account of open-loop unstable processes which are common in some industries, however simple mechanisms [9] for overcoming this do exist. What is more significant was pointed out in [14], it is not generally wise or safe to use unstable predictions as a basis for a predictive control law design. There is a need to ensure that the prediction class is stable, even for unstable open-loop plant. It does not take much experiment with the GPC algorithm [1] on plants of the form $(s-a)/(s-ra)(s+b)$ to see the truth of this. Hence this is a clear gap in the literature. The work of [12, 14] gives a good solution, or reparameterisation of the degrees of freedom to ensure stable predictions, for realigned models. However, as yet the issue of how to form stable predictions for a unstable process represented by an independent model is unsolved. This is a necessary first step in the transferal of the many apriori stability results. Moreover, such a solution is required before algorithms such as PFC can take systematic account of unstable plant. Due to its 'intuitive' and simple philosophy, PFC cannot really use terminal constraints (e.g. [5, 15]) but it requires stable predictions for the intuitive arguments to be effective. Although it is also important to reformulate the MPC algorithms deploying terminal constraints for the IM case, that is not done here.

In this paper we first give some background to independent models and then in section 3 develop a neat algebraic form for the predictions. This form is used in section 4 to divise a simple structure for the future input trajectory that is guaranteed stabilising. Section 5 then gives examples of how effective this structure is and demonstrates enormous improvements in performance on conventional approaches.

2 Background

2.1 Independent models of unstable processes

Let y, u be the process outputs/inputs respectively and y_m the output of the independent model (IM). In general due to uncertainty $y \neq y_m$. The norm is to simulate the IM in parallel with the process, using the same inputs u. However, with an unstable process this cannot work because it is unlikely that the same input would stabilise the IM and an uncertain plant. The solution in [9] is to decompose the model into two parts as in Figure 1. Hence if the process is modelled by G, then

$$G = (I + M_2)^{-1} M_1 \tag{1}$$

where both M_1 and M_2 are stable. Next note that, in the nominal case of $y = y_m$ (recall y is the process output), the output of figure 2 is the same as that of figure 1, so equivalent to using the structure of Figure 1, one could use the structure of figure 2. In this case if u is stable, so is w and if y is stable so is z. Hence when the process is stabilised, so is the output of the independent model given in Figure 2.

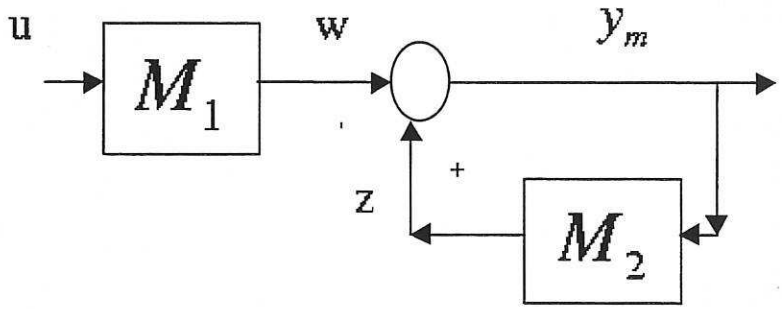

Figure 1. Independent model used for prediction

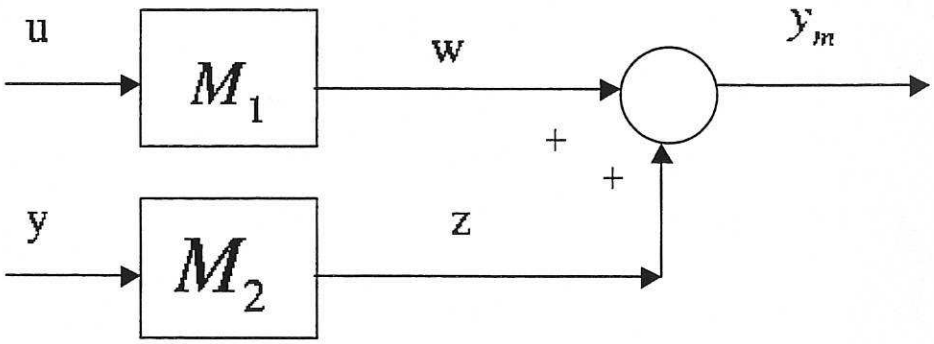

Figure 2. Implementation of independent model for open-loop simulation

A convenient decomposition in the SISO case is as follows:

$$G = \frac{n}{d} = \frac{n_+ n_-}{d_+ d_-}; \quad M_1 = \frac{n_+}{d_-}; \quad M_2 = \frac{b_2}{n_-}; \quad b_2 = n_- - d_+ \tag{2}$$

where n_+, d_+ are the numerator/denominator factors respectively containing roots outside the unit circle (unstable). It is clear that both M_1 and M_2 have stable poles.

2.2 Independent models for prediction

When using an independent model for prediction, a different structure is required because future y are unknown; in fact precisely because these are what we want to predict. In this case the structure of figure 2 cannot help us. Hence, we can use partial state realignment. That is return to figure 1 and realign the loop variable y_m on the process output measurement, then use this as a basis for prediction, where now the only unknown is the future input variable u; these values of course are the usual degrees of freedom used in MPC.

2.3 Forming predictions using Toeplitz matrices

Assuming a model transfer function model (we ignore noise/disturbances so as not to introduce unnecessary notation for this paper) of the form

$$d(z)y_k = n(z)u_k \tag{3}$$

where y_k, u_k are the output/input respectively at the kth sampling instant. Define vectors of future (arrow pointing right) and past values (arrow pointing left)

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

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

In general the vector of future values can be any length, but the lengths n_y, n_u correspond to output and input prediction horizons common in GPC and will be used later. For computing predictions, we will use the Toeplitz/Hankel notation as this is algebraically very neat and also easy to code[2]. Write equation (3) as a difference eqn. $D(z)y_k = n(z)\Delta u_k$, where $D = d\Delta$, at each future time instant and then place the set of simultaneous equations in matrix/vector format e.g.

$$C_D \underset{\rightarrow k}{y} + H_D \underset{\leftarrow k}{y} = C_n \Delta \underset{\rightarrow k-1}{u} + H_n \Delta \underset{\leftarrow k-1}{u} \tag{4}$$

where for a generic polynomial $n(z)$,

$$C_n = \begin{bmatrix} n_0 & 0 & 0 & \dots \\ n_1 & n_0 & 0 & \dots \\ n_2 & n_1 & n_0 & \dots \\ \vdots & \vdots & \vdots & \vdots \\ n_m & n_{m-1} & n_{m-2} & \vdots \end{bmatrix}; \quad H_n = \begin{bmatrix} n_1 & \dots & n_{m-1} & n_m \\ n_2 & \dots & n_m & 0 \\ \vdots & \vdots & \vdots & \vdots \\ n_m & 0 & \vdots & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{5}$$

Clearly (4) implies a prediction equation

$$Y = H\Delta \underset{\rightarrow}{u} + P\Delta \underset{\leftarrow}{u} + Q \underset{\leftarrow}{y} \tag{6}$$

with $H = C_D^{-1}C_n$, $P = C_D^{-1}H_n$, $Q = -C_D^{-1}H_D$.

[2]Although an efficient coding is equivalent to less transparent mechanisms via diophantine identities [11]

Remark 2.1 *It is noted that throughout this paper, because Toeplitz/Hankel matrices can be viewed as operators, matrix and vector dimensions are defined flexibly. That is dimensions are obvious from the context and will not be defined explicitly in this paper. So for instance* $\underset{\leftarrow}{x}$ *can be interpreted as both* $[x_k]$ *and also* $[x_k, x_{k-1}, ..., x_{k-n}]^T$. *Moreover in computations of the form* $v = K_1\underset{\leftarrow}{x} + K_2\underset{\leftarrow}{x}$ *where initially* K_1, K_2 *have different dimensions, then* K_1 *and/or* K_2 *are understood to be padded with zeros as required so that* $v = [K_1 + K_2]\underset{\leftarrow}{x}$.

Also note the further definition that Γ_n *is a tall and thin variant on the square matrix* C_n *so that* $[1, z^{-1}, z^{-2}...]\Gamma_n b = n(z)[1, z^{-1}, ...]b$. *Finally note the commutative and inverse properties*

$$C_n C_d = C_d C_n; \quad C_d^{-1} = C_{1/d} \tag{7}$$

2.4 A typical MPC algorithm

The basis of predictive control is the computation of the predicted system behaviour and how this depends on the selected degrees of freedom. For instance a typical performance index takes the form:

$$J = \|W_y(\underset{\rightarrow}{r} - \underset{\rightarrow}{y})\|_2^2 + \|W_u\Delta\underset{\rightarrow}{u}\|_2^2 \tag{8}$$

where W_y, W_u are weighting matrices. Substitution of (6) into (8) and minimisation w.r.t $\Delta\underset{\rightarrow}{u}$ gives

$$\Delta\underset{\rightarrow}{u} = [H^T W_y H + W_u]^{-1} H^T W_y[\underset{\rightarrow}{r} - P\Delta\underset{\leftarrow}{u} - Q\underset{\leftarrow}{y}] \tag{9}$$

Only the first element $\Delta u_k = [I, 0, 0, ...]\Delta\underset{\rightarrow}{u}$ is implemented.

3 Prediction based on an independent model without prestabilisation

When an IM is used, prediction is slightly more involved. Here we assume that past outputs are known and use Figure 1 as a basis for prediction. It is assumed that $\underset{\leftarrow}{y}$ is always defined on plant measurements so there is partial state realignment around M_2 but no realignment around the dynamics in M_1.

3.1 The basis of prediction with IM implementation of figure 1

Set up a prediction equations around M_1 and M_2 separately:

$$\begin{aligned} C_{d-}\underset{\rightarrow}{w} &= C_{n+}\underset{\rightarrow}{u} + H_{n+}\underset{\leftarrow}{u} - H_{d-}\underset{\leftarrow}{w} \\ C_{n-}\underset{\rightarrow}{z} &= C_{b_2}[\underset{\rightarrow}{y}_m + \hat{d}] + H_{b_2}\underset{\leftarrow}{y} - H_{n-}\underset{\leftarrow}{z} \\ \underset{\rightarrow}{y}_m &= \underset{\rightarrow}{z} + \underset{\rightarrow}{w} \\ \underset{\rightarrow}{y} &= \underset{\rightarrow}{y}_m + \hat{d} \end{aligned} \tag{10}$$

where $\hat{d}$ represents a correction for offset which is used in IM based MPC (and DMC etc.) to ensure integral action. Define

$$\hat{d} = L(\underset{\leftarrow}{y} - \underset{\leftarrow}{z} - \underset{\leftarrow}{w}) \tag{11}$$

where L is a vector of ones. For convenience (to ensure that $J = 0$ is consistent with zero offset) it is usual to express the degrees of freedom in terms of future input increments (rather than absolute inputs), hence

$$\underset{\rightarrow}{u} = E\Delta\underset{\rightarrow}{u} + L\underset{\leftarrow}{u} \tag{12}$$

where E is a lower triangular matrix of ones. One can then rewrite (10) as

$$\begin{array}{rcl} C_{d-}\underset{\rightarrow}{w} & = & C_{n+}(E\Delta\underset{\rightarrow}{u} + L\underset{\leftarrow}{u}) + H_{n+}\underset{\leftarrow}{u} - H_{d-}\underset{\leftarrow}{w} \\ C_{n-}\underset{\rightarrow}{z} & = & C_{b_2}[\underset{\rightarrow}{z} + \underset{\rightarrow}{w} + L(\underset{\leftarrow}{y} - \underset{\leftarrow}{z} - \underset{\leftarrow}{w})] + H_{b_2}\underset{\leftarrow}{y} - H_{n-}\underset{\leftarrow}{z} \\ \underset{\rightarrow}{y}_m & = & \underset{\rightarrow}{z} + \underset{\rightarrow}{w} \\ \underset{\rightarrow}{y} & = & \underset{\rightarrow}{y}_m + L(\underset{\leftarrow}{y} - \underset{\leftarrow}{z} - \underset{\leftarrow}{w}) \end{array} \tag{13}$$

With an IM, prediction reduces to the solution of the simultaneous equations in eqns.(11,13).

3.2 An algebraic solution for the prediction equations

In the first instance, solve the simultaneous equations (13) for $\underset{\rightarrow}{y}_m$:

$$\begin{array}{rl} [I - C_{n-}^{-1}C_{b_2}]\underset{\rightarrow}{y}_m = \{ & C_{d-}^{-1}[C_{n+}(E\Delta\underset{\rightarrow}{u} + L\underset{\leftarrow}{u}) + H_{n+}\underset{\leftarrow}{u} - H_{d-}\underset{\leftarrow}{w}] \\ & +C_{n-}^{-1}[C_{b_2}L(\underset{\leftarrow}{y} - \underset{\leftarrow}{z} - \underset{\leftarrow}{w}) + H_{b_2}\underset{\leftarrow}{y} - H_{n-}\underset{\leftarrow}{z}]\} \end{array} \tag{14}$$

However, we further note that $[I - C_{n-}^{-1}C_{b_2}] = C_{d+}C_{n-}^{-1}$. Hence we can now write that

$$\begin{array}{rcl} \underset{\rightarrow}{y}_m & = & C_{d+}^{-1}C_{n-}\{ \; C_{d-}^{-1}[C_{n+}(E\Delta\underset{\rightarrow}{u} + L\underset{\leftarrow}{u}) + H_{n+}\underset{\leftarrow}{u} - H_{d-}\underset{\leftarrow}{w}] \\ & & \quad +C_{n-}^{-1}[C_{b_2}L(\underset{\leftarrow}{y} - \underset{\leftarrow}{z} - \underset{\leftarrow}{w}) + H_{b_2}\underset{\leftarrow}{y} - H_{n-}\underset{\leftarrow}{z}]\} \\ & = & C_{d+}^{-1}C_{d-}^{-1}\{ \; C_{n-}[C_{n+}(E\Delta\underset{\rightarrow}{u} + L\underset{\leftarrow}{u}) + H_{n+}\underset{\leftarrow}{u} - H_{d-}\underset{\leftarrow}{w}] \\ & & \quad +C_{d-}[C_{b_2}L(\underset{\leftarrow}{y} - \underset{\leftarrow}{z} - \underset{\leftarrow}{w}) + H_{b_2}\underset{\leftarrow}{y} - H_{n-}\underset{\leftarrow}{z}]\} \end{array} \tag{15}$$

Putting common terms together gives

$$\begin{array}{rcl} \underset{\rightarrow}{y}_m & = & C_d^{-1}\{ \; C_{n-}[C_{n+}E\Delta\underset{\rightarrow}{u} + (C_{n+}L + H_{n+})\underset{\leftarrow}{u}] \\ & & \quad -(C_{n-}H_{d-} + C_{d-}C_{b_2}L)\underset{\leftarrow}{w} + C_{d-}(C_{b_2}L + H_{b_2})\underset{\leftarrow}{y} - C_{d-}(C_{b_2}L + H_{n-})\underset{\leftarrow}{z}\} \end{array} \tag{16}$$

Finally adding in the equation $\underset{\rightarrow}{y} = \underset{\rightarrow}{y}_m + L(\underset{\leftarrow}{y} - \underset{\leftarrow}{z} - \underset{\leftarrow}{w})$ and tidying up gives

$$\begin{array}{rcl} \underset{\rightarrow}{y} & = & H\Delta\underset{\rightarrow}{u} + P_u\underset{\leftarrow}{u} + P_w\underset{\leftarrow}{w} + P_y\underset{\leftarrow}{y} + P_z\underset{\leftarrow}{z} \\ H & = & C_d^{-1}C_nE \\ P_u & = & C_d^{-1}(C_nL + C_{n-}H_{n+}) \\ P_w & = & -C_d^{-1}(C_{n-}H_{d-} + C_{d-}C_{b_2}L) - L \\ P_y & = & C_d^{-1}C_{d-}(C_{b_2}L + H_{b_2}) + L \\ P_z & = & -C_d^{-1}C_{d-}(C_{b_2}L + H_{n-}) - L; \end{array} \tag{17}$$

For simplicity one may which to represent this as

$$\underset{\rightarrow}{y} = H\Delta\underset{\rightarrow}{u} + Mv; \qquad M = [P_u, P_w, P_y, P_z]; \qquad v = \begin{bmatrix} \underset{\leftarrow}{u} \\ \underset{\leftarrow}{w} \\ \underset{\leftarrow}{y} \\ \underset{\leftarrow}{z} \end{bmatrix} \tag{18}$$

where clearly there is a nice separation between the part $H\Delta\underset{\rightarrow}{u}$ depending on the d.o.f. (often called 'forced response') and the notional 'free response' part Mv, i.e. the response should the input remain unchanged.

Remark 3.1 *The predictive control law is obtained by substituting (18) into (8) and minimising wrt to* $\Delta \underset{\rightarrow}{u}$. *This gives*

$$\Delta \underset{\rightarrow}{u} = (H^T W_y H + W_u I)^{-1} H^T W_y [\underset{\rightarrow}{r} - Mv] \tag{19}$$

of which only the first element Δu_k *is implemented. Clearly* v *comprises known data obtained from the IM and the process.*

Remark 3.2 *The predictions of (18) are correct algebraically and easy to code given the summary of (17). However, as noted in [14] and any simple examples based on the GPC algorithm, the resulting control law based on these will often be unstable. This can be due to:*

1. *Numerical ill-conditionning (divergent predictions cause numerical difficulties in computing* H, M *and the control law accurately) for large* n_y.
2. *The objective* J *is ill-posed in that with small number of degrees of freedom it is difficult to make the output predictions near stable. Hence even though* J *itself maybe small, the predicted errors beyond the output horizon* n_y *will diverge rapidly.*
3. *As one increases* n_u *to overcome this, one must increase* n_y *(guidelines give* $n_y \gg n_u$*) and then numerical ill-conditionning may occur.*

In conclusion one cannot choose n_y *small or* n_y *large. Terminal conditions can overcome the restriction on small* n_y *to some extent, but that issue is not the topic of this paper.*

4 Stabilising the predictions from a realigned model

The solution implicit in the use of endpoint constraints (e.g. [7, 12]) is to to find a parameterisation of future inputs that forces the output predictions to be stable (and hence errors do not diverge beyond the output horizon n_y). Once the output predictions are stable one can reliably compute J for $n_y = \infty$, should one want to, by using Lyapunov equations[3]. The problem that is solved here is to find a mechanism for ensuring stable predictions from (18) without invoking constraints, say via Lagrange multipliers, on the latter part of the prediction; these constraints can exacerbate numerical ill-conditionning [14].

Start with eqn.(16). This is arranged as follows (noting that $E = C_\Delta^{-1}$):

$$\begin{aligned} \underset{\rightarrow m}{y} \;=\; & C_d^{-1} C_\Delta^{-1} \{ \; [C_n \Delta \underset{\rightarrow}{u} + C_\Delta (C_n L + C_{n\_} H_{n+}) \underset{\leftarrow}{u} \\ & - C_\Delta (C_{n\_} H_{d\_} + C_{d\_} C_{b_2} L) \underset{\leftarrow}{w} + C_\Delta C_{d\_} (C_{b_2} L + H_{b_2}) \underset{\leftarrow}{y} - C_\Delta C_{d\_} (C_{b_2} L + H_{n\_}) \underset{\leftarrow}{z} \} \end{aligned} \tag{20}$$

Now separate into parts to be chosen (that is those depending on $\Delta \underset{\rightarrow}{u}$) and known parts (that is those depending upon v):

$$\begin{aligned} \underset{\rightarrow m}{y} &= C_{d\_}^{-1} C_\Delta^{-1} C_{d+}^{-1} \{ C_n \Delta \underset{\rightarrow}{u} + M_s v \} \\ M_s &= [K_u, K_w, K_y, K_z]; \\ K_u &= C_\Delta (C_n L + C_{n\_} H_{n+}) \\ K_w &= -C_\Delta (C_{n\_} H_{d\_} + C_{d\_} C_{b_2} L) \\ K_y &= C_\Delta C_{d\_} (C_{b_2} L + H_{b_2}) \\ K_z &= -C_\Delta C_{d\_} (C_{b_2} L + H_{n\_}) \end{aligned} \tag{21}$$

[3] Infinite horizons are an easy way to give apriori stability guarantees

where v is defined in (18). Finally rewrite in terms of z-transforms

$$\underset{\rightarrow}{y}_m(z) = \frac{n(z)\Delta\underset{\rightarrow}{u}(z) + x(z)}{d_-(z)d_+(z)\Delta(z)}; \quad x(z) = [1, z^{-1}, ...]M_s v \tag{22}$$

Clearly the output predictions are stable iff the numerator term contains a factor $d_+(z)$. Hence one gets the following constraint on $\Delta\underset{\rightarrow}{u}(z)$:

$$n(z)\Delta\underset{\rightarrow}{u}(z) + x(z) = d_+(z)\gamma(z) \quad \Rightarrow \underset{\rightarrow}{y}_m(z) = \frac{\gamma(z)}{d_-(z)\Delta(z)} \tag{23}$$

where $\gamma(z)$ is a degree of freedom (taken here to be a polynomial). Constraint (23) (a diophantine equation) is easy to solve and gives a parameterisation of inputs (and output predictions) in the following form)

$$\begin{array}{rcl} \Delta\underset{\rightarrow}{u} & = & K_1 M_s v + \Gamma_{d+}\underset{\rightarrow}{c} \\ \underset{\rightarrow}{y}_M & = & [C_\Delta C_{d-}]^{-1}[\gamma + \Gamma_n \underset{\rightarrow}{c}] \\ \gamma & = & K_2 M_s v \end{array} \tag{24}$$

where matrices K_1, K_2 depend upon (23) and are trivial to compute and $\underset{\rightarrow}{c}$ comprises the d.o.f. in the solution. The notation $\Gamma_{(.)}$ is defined in remark 2.1.

In order to ensure offset free prediction use eqn. (11) to give:

$$\begin{array}{rcl} \underset{\rightarrow}{y} & = & [C_\Delta C_{d-}]^{-1}[K_2 M_s v + \Gamma_n \underset{\rightarrow}{c}] + L(\underset{\leftarrow}{y} - \underset{\leftarrow}{z} - \underset{\leftarrow}{w}) \\ \underset{\rightarrow}{y} & = & H_1 \underset{\rightarrow}{c} + M_y v \end{array} \tag{25}$$

where $H_1 = [C_\Delta C_{d-}]^{-1}\Gamma_n$, $M_y \equiv [C_\Delta C_{d-}]^{-1}K_2 M_s + [0, -L, L, -L]$.

Remark 4.1 *Minimising J w.r.t. c does not give rise to the same control law as minimising w.r.t $\Delta\underset{\rightarrow}{u}$, even assuming of course the same number of degrees of freedom. This is because predictions of (24) are already stabilised, so all the d.o.f. in c can be used for performance optimisation. However in (18) some of the d.o.f. contained within $\Delta\underset{\rightarrow}{u}$ will be needed to counteract the effect of the unstable poles and moreover it is unlikely that the predictions will remain bounded beyond the prediction horizon. Hence, the prestabilised approach is to be preferred because in the part of the predictions not within the cost, it is known that the predictions are stable. This has significant advantages for feasibility analysis (e.g. [6, 10]). It also allows straightforward extension of horizons to $n_y = \infty$ if desired.*

Remark 4.2 *The predictive control law is obtained by substituting from (24,25) into (8) and minimimising wrt $\underset{\rightarrow}{c}$.*

$$\begin{array}{rl} \underset{\rightarrow}{c} = & [\Gamma_{d+}^T W_u \Gamma_{d+} + H_1^T W_y H_1]^{-1} \\ & [\Gamma_{d+} W_u K_1 M_s v + H_1^T W_y(\underset{\rightarrow}{r} - M_y v)] \end{array} \tag{26}$$

The optimum $\Delta\underset{\rightarrow}{u}$ comes from substitution of (26) into (24).

5 Examples

5.1 Example 1

The following unstable process was used

$$G = \frac{z^{-1} - 1.5z^{-2} + 0.36z^{-3}}{1 - 3.3z^{-2} + 3z^{-2} - 0.8z^{-3}} \tag{27}$$

which has poles at $2, 0.8, 0.5$ and zeros at $1.2, 0.3$. Simulations were performed for various output and input horizons of which typical results are illustrated here. The main emphasis is to compare the use of predictions equations (6,18,25).

5.1.1 Simulation results

In the first instance we took $n_u = 1$. In this case the prestabilised predictions gave stable and satisfactory closed-loop performance for $n_y \geq 10$. With predictions (6,18) we did not get good performance for any n_y. In figure 3, we illustrate the closed-loop simulations to a unit step demand for predictions (25) with $n_u = 1, n_y = 15$. It is clear that with just one degree of freedom (which implies a low computational load), the performance is good.

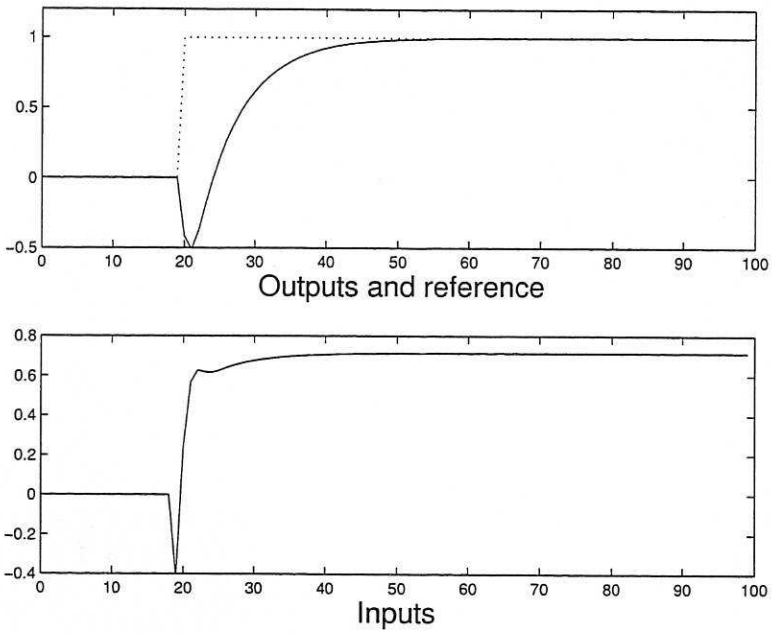

Figure 3. Simulations for $n_y = 15,\ n_u = 1$

Next we illustrate a case where stable performance is achieved with all 3 prediction equations, $n_y = 15,\ n_u = 4$. The interesting part of this simulation (figure 4) is the latter part (after the 60th sample) where measurement noise is introduced. The dashed line is for prediction eqn. (6) and the solid lines for prediction eqn.(25)[4]. It is clear using total state realignment resulted in poor noise rejection where as using only partial state realignment gives far better results. This topic is the subject of further investigation.

[4]Prediction eqn.(18) has equally good noise rejection

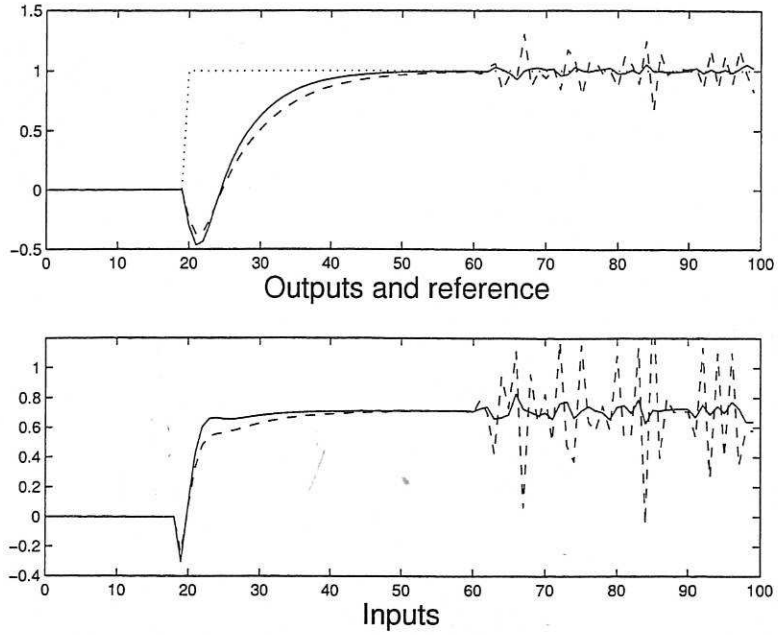

Figure 4. Simulations for $n_y = 15$, $n_u = 4$ and with measurement noise

5.1.2 Numerical ill-conditionning

For completeness a check was made on the maximum horizon for which conventional algorithms would cope. For prediction horizons greater than this closed-loop oscillation and instability began to occur due to numerical ill-conditionning. It is noted that these examples are on MATLAB so 16 significant figures are allowed. We tested (25) with $n_u = 4$ upto $n_y = 100$ without any indication of problems.

Predictions (6)	Predictions (18)	Predictions (25)
22	27	> 100

Table 1. Prediction horizon at which failure occurs

It is clear from this table that without prestabilisation numerical ill-conditionning can occur at low prediction horizons whereas with prestabilisation there is no illconditionning. This benefit could be far more important in industrial processors with less significant figures.

5.1.3 Example 2

This example has a Laplace transform

$$G(s) = \frac{s-1}{s^2 - 1.5s - 1} \tag{28}$$

and is sampled at 0.2sec sampling rate. Here the results are summarised in tabular form (Tables 2,3 below) only for different choices of n_u. For $n_u > 1$ the results are similar so a single table is used. The weightings are $W_u = W_y = 1$.

Once again three conclusions are immediately clear. Using prestabilised equations:

- allows the use of less degrees of freedom (e.g. $n_u = 1$ is possible).

- gives a far greater range of output horizons for which closed-loop stability is achieved.
- avoids numerical ill-conditionning ('failed' is due to this).

	$n_u = 1$		
n_y	Predictions (6)	Predictions (18)	Predictions (25)
$n_y < 9$	unstable	unstable	unstable
9	unstable	unstable	very slow
10-11	unstable	unstable	slow
12-18	unstable	unstable	good
$n_y > 18$	Very slow	Very slow	good

Table 2. Description of nominal closed-loop performance

	$n_u > 1$		
n_y	Predictions (6)	Predictions (18)	Predictions (25)
$n_y < 10$	unstable	unstable	unstable
10	unstable	unstable	very slow
11-12	unstable	unstable	slow
13-14	unstable	unstable	good
15-17	slow	slow	good
18-40	good	good	good
41-48	failed	good	good
$n_y > 48$	failed	failed	good

Table 3. Description of nominal closed-loop performance

6 Conclusion

It is known that using independent models can improve noise rejection and also has parallels with the FIR models commonly adopted in industry. However, extension of this concept to the unstable case is not straightforward, especially given the difficulties with tuning guidelines and numerical ill-conditionning. In this paper the method of prediction prestabilisation has been developed for the IM case and demonstrated to provide a very effective solution to both these problems. Tuning becomes easier as beyond a logical minimum output horizon, stability is achieved with any number of degrees of freedom. Also numerical ill-conditionning is avoided which could be vital in processors with less than 16 significant figures or where large output horizons are desired. The technique is straightforward, in fact a simple reparameterisation of the future input trajectory, and allows the user to have a good assurance (though not an apriori guarantee) of stability without the need for terminal constraints.

There are other benefits to be investigated which constitute future work. The underlying motivation was applications in PFC where terminal constraints would not be acceptable and there is a minimalist, but intuitive, approach to control design. Hence the next step is to investigate whether this approach will allow systematic extension of PFC to the unstable case. Secondly it is known that the use of independent models in prediction can allow better noise rejection than realigned models. This expectation was borne out in example 1. Future work will investigate whether this benefit does carry over more generally to the unstable case and whether prestabilisation helps or hinders. A more

obvious gap in the literature is to extend the guaranteed stability results of [5, 7, 15] to make use of the independent form of model.

References

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

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

[3] 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

[4] C.E. Garcia, D.M. Prett and M. Morari, 1989, Model predictive control: theory and practice, a survey, Automatica, 25, p335-348

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

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

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

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

[9] J.Richalet, Commande predictive, R 7 423 (Internal report, ADERSA)

[10] J.A. Rossiter, B.Kouvaritakis and J.R. Gossner, Feasibility and stability results for constrained stable generalised predictive control, Automatica, 31,6, pp863-877 , 1995

[11] J.A. Rossiter, Notes on multi-step ahead prediction based on the principle of concatenation, Proc. IMechE, 1993, Vol.207, pp261-263

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

[13] J.A. Rossiter and J. Richalet, Re-aligned models for prediction in MPC: a good thing or not ? submitted to APC6

[14] J.A. Rossiter, M.J.Rice and B. Kouvaritakis, A numerically robust state-space approach to stable predictive control strategies, 1998, Vol. 34, No. 1, pp65-73 Automatica

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

[16] T-W. Yoon and D.W. Clarke, 1995, Observer design in receding horizon predictive control, IJC, 61, pp171-191