Pergamon

A. Rev. Control, Vol. 20, pp. 83–94, 1996

S0066-4138(97)00007-4
1367-5788/97 $32.00 + 0.00

ADAPTIVE PREDICTIVE CONTROL

D.W. Clarke
Department of Engineering Science, University of Oxford
Parks Road, Oxford OX1 3PJ, U.K.

Abstract: Progress in the design and use of Model-Based Predictive Control is reviewed. The two-degree-of-freedom solution of many predictive algorithms enables optimal set-point response and rejection of known disturbance patterns, improves robustness against model/plant mismatch, or provides a compromise between these objectives. The original lack of a comprehensive theory of stability has been overcome by adapting earlier work on terminal constraints in receding-horizon control. The finite-horizon methodology readily handles actuator, output, or state constraints.

Adaptive predictive control uses a parameter estimator. Recent elegant work by Niu and Fisher has extended the UDU updating method so that models of different order can be simultaneously estimated. Using an alternative Lagrange multiplier approach, the structure of the resulting equations is shown to be similar to certain MBPC solutions.

keywords: predictive control, adaptive control, parameter estimation

1 INTRODUCTION

The conceptual structure of a generic predictive controller, as displayed in Fig.1, is well-known. At

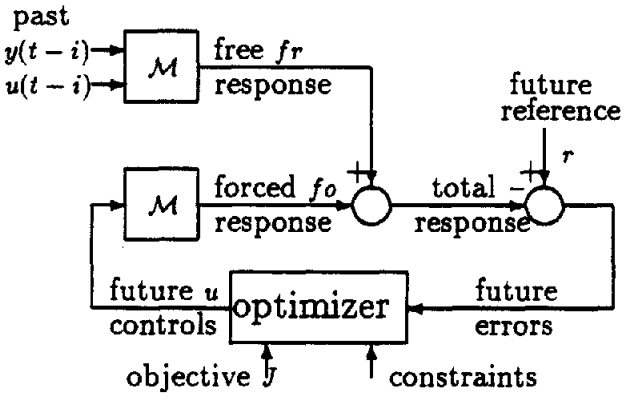

Figure 1: Basic structure of MBPC

its heart is a *model* $\mathcal{M}(\theta)$, parameterised by a set θ, which is used to predict the future behaviour of a plant. The prediction has two main components:

the FREE response : $fr(t+j|t), j > 0$, being the expected behaviour of the output $y(t+j)$ assuming zero future control actions (or zero deviations from a prescribed future policy such as imposing constant controls);

the FORCED response : $fo(t+j|t), j > 0$, being the *additional* component of output response due to the 'candidate' set of future controls $u(t+j), j \geq 0$.

For linear systems the total prediction, using the principle of superposition, is taken to be the sum

$$\hat{y}(t+j|t) = fr(t+j|t) + fo(t+j|t), \qquad (1)$$

and computed up to some chosen finite *prediction horizon* $j = N$. Associated with the prediction *range* $j = 1 \ldots N$ is a future

reference sequence : $r(t+j)$, being the target values we wish the output to attain,

from which the future

system errors : $e(t+j) = r(t+j) - \hat{y}(t+j|t)$

can be calculated. An optimizer, having a user-specified

objective function : a mathematical indicator $J(\mathbf{e}, \mathbf{u})$ of the desired performance of the controlled loop, and a

constraint set : the allowed regions for controls and output variables ($u \in \mathcal{U}$, etc.),

determines the 'best' set of future controls $\{u(t+j|t), j = 0 \cdots\}$. An open-loop strategy would simply assert this set of controls in sequence: what makes MBPC into a closed-loop feedback control law is the use of the *receding-horizon approach*, for which *only the first of the set - $u(t)$ - is transmitted to the plant* and the whole operation of prediction-optimization-control is repeated at

each sample. Feedback of the current plant output measurement $y(t)$ is via the prediction equations.

MBPC possesses many attributes which make it a successful approach to industrial control design:

Simplicity : the basic ideas of MBPC do not require complex mathematics and are 'intuitive'.

Richness : all of the basic MBPC components: the model, the horizons, the objective function, etc., can be tailored to the details of the problem in hand (this is attractive to academics who wish to explore the combinations for maximizing publications).

Practicality : it is often the resolution of problems such as satisfying control or output constraints which determine the utility of a controller - MBPC is one of very few procedures which can deal with these issues in a natural way.

Demonstrability : it works, as shown by many real applications in industry where MBPC is routinely and profitably employed. (This has caused a certain chagrin amongst those theoreticians who believe that useful controllers can only come via the assumptions, lemma, theorem route. In fact one stimulus for the current surge of interest in MBPC is that the reasons why the approach is profitable often differ from those normally considered in control theory).

Whilst academics had explored the stabilizing properties of receding-horizon control quite early on (Kleinman, 1970; Kwon and Pearson, 1977), it was the heuristic industrial approaches of IDCOM (Richalet *et al*, 1978) and DMC (Cutler and Ramaker, 1980) which independently showed the applicability of MBPC. The adaptive/self-tuning control community at the time could provide effective methods for acquiring models of plants, but were still invoking sensitive short-range (k-step-ahead) control designs (see, e.g. Clarke and Gawthrop, 1975). The use of long-range prediction emerged via EHAC (De Keyser and Van Cauwenberghe, 1985) EPSAC (Ydstie, 1984) and GPC (Clarke, Mohtadi and Tuffs, 1987), and these proved to be significantly more robust in practice compared with MV/GMV (see De Keyser, Van der Velde and Dumortier, 1988, and Soeterboek, 1992, for comparative studies). Though MBPC was seen to work, and work well, there was a lack of comprehensive stability results which could act as the theoretical underpinning of the subject and provide design guidelines (rather than the rules-of-thumb often used to set up MBPC tuning parameters). 'Thinking men' (Bitmead, Gevers and Wertz, 1990) suggested that infinite-horizon LQ was the answer, but this involved the loss of ability to solve the important problem of constraints (Prett and Garcia, 1988).

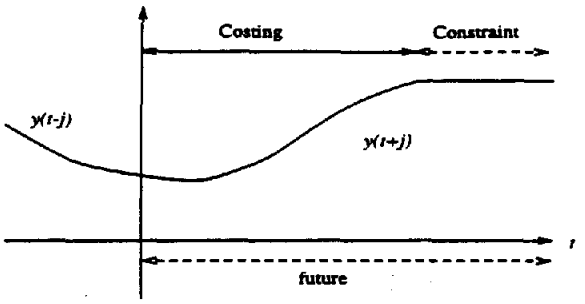

Figure 2: Typical costing and constraint horizons in CRHPC

The confluence of the three streams – heuristic industrial MBPC, adaptive control, and receding-horizon state-space theories – has recently stimulated several solutions to the problem of guaranteed stability (see Mosca, 1995, for an excellent overview). These all have a common theme: the use of some form of *terminal constraint* or *end-point weighting*. As an example, CRHPC (Clarke and Scattolini, 1991) and SIORHC (Mosca, Lemos and Zhang, 1990) consider optimizing a cost over the horizon $j = 1 \cdots N$, subject to the additional *equality constraints* that over a further range of m points the predicted output should precisely equal the future reference:

$$\hat{y}(t+j|t) = r(t+N), j = N+1 \cdots N+m, \quad (2)$$

as shown in Fig.2. Some of the proof techniques used can be further applied to the problem of *inequality* constraints, such as $Y_1 \leq \hat{y}(t+j|t) \leq Y_2$, leading to a greater understanding of how to choose 'constraint ranges' ($j \in NC_1 \cdots NC_2$) for stability. In fact it is increasingly clear that the interesting problems now lie in the 'best' resolution of *infeasibilities* and in the *deliberate* and programmed choice of constraints for enhanced performance (such as elimination of overshooting/undershooting).

The cornerstone of MBPC is the model. A good design includes mechanisms for obtaining the 'best' model and for making the controller robust against inevitable plant/model mis-match. As the recent IFAC benchmark (Yoon and Clarke, 1994) showed, such combinations are unbeatable.

2 MODELS

The choice of *structure* of a model has several consequences: it has to be rich enough to capture the essential dynamics of the plant, it has to provide the free and forced predictions, it has to be readily understood and verified by the plant engineers, and it has to be analysable to enable theoretical

predictions of likely behaviour. Consider the following examples:

Mass/energy transport: a control valve regulates the flow of a stream which splits into two parallel processes, one having a long deadtime. The outputs of the processes are then mixed. Here the step response will be characterised by the superposition of two exponentials, one being delayed.

Compliant robot arm: a flexible strip is actuated at one end by a torque motor. Flexibility leads to a (theoretically infinite) set of modes characterised by lightly-damped pole-zero pairs, together with two poles at the origin due to Newton's law.

Servo-hydraulic control: a materials-testing machine has a 13th-order lumped-parameter model whose coefficients depend on mostly fixed inertias and elasticities. The 3rd-order reduced model in continuous time has a set of parameters which are known functions of the specimen stiffness. Real-time evaluation of the stiffness enables the model parameters to be computed.

Appropriate models for these problems will range from 'black box' or general-purpose for the first case to 'grey box' for the third. In process control applications the general-purpose form of step/impulse-responses are used, as the processes are generally open-loop stable and slow, so that the computational overhead involved with the large number of parameters θ is not important. For embedded systems, such as servo-hydraulic or engine management, there is fast sampling (e.g. 1kHz), but the model structure can be closely *tailored* to a partly understood physical plant.

It has become recognised that it is important to include a *disturbance model* in a MBPC design. This has several consequences: the internal model principle leads to 'disturbance annihilating' controllers (e.g. assuming Brownian motion automatically gives integral control); a correct disturbance model can lead to minimum-variance control for maximum quality; an *assumed* model can be used to increase robustness. Theoretical analysis of MBPC uses state-space for stability, CARIMA for minimal parameter adaptive applications, and IIR for investigating predictions and disturbance performance. A typical IIR model is:

$$y(t) = M(q^{-1})u(t) + N(q^{-1})\xi(t) \qquad (3)$$

where q^{-1} is the backward-shift operator and M, N are infinite polynomials giving the control and disturbance dynamics. We can decompose the polynomial N in order to extract the first j terms:

$$N(q^{-1}) = N_j^*(q^{-1}) + q^{-j}N_j(q^{-1}), \qquad (4)$$

and similarly for M, so that a *minimum-variance* predictor is (after some manipulation):

$$\hat{y}(t+j|t) = M_{j+1}u(t-1) + \frac{N_j}{N}\{y(t) - Mu(t)\} + M_{j+1}^*u(t+j|t), \qquad (5)$$

for which the prediction error j-steps ahead is $N_j^*(q^{-1})\xi(t+j)$. As the polynomial N_j^* is of degree $j-1$ the error is orthogonal to the data, provided that ξ is a 'white noise' sequence. We note from eqn.5 that there are three basic terms in the prediction: the first two give the free fresponse and the third is forced response. The use of $j+1$ for the decomposition of M is because $y(t)$ is available for the prediction, but $u(t)$ (and indeed the whole future control sequence $u(t+j|t)$) has yet to be determined. Note further that $m_0 = 0$ as a consequence of the ZOH.

The second free response term $N_j/N\{y(t) - Mu(t)\}$ is not found in the early heuristic approaches. If the model is exact and there is no disturbance we have $y(t) = Mu(t)$, so the term makes no contribution (and indeed there is then no feedback from the current measurement $y(t)$). However, we use feedback *precisely* because there is mismatch and disturbance, so the *filtering* effect of N_j/N can be used to eliminate disturbance and/or improve robustness. See that the forced response is independent of the assumed disturbance model: the 'dynamic matrix' corresponding to this term will be the same no matter the assumed N and by implication the 2DOF property of separate closed-loop responses to set-points and disturbances is assured. Of course to work properly the heuristic IDCOM/DMC must *implicitly* assume a noise model, and it is easy to show that the Brownian-motion form of $N = 1/\Delta$, where $\Delta = 1 - q^{-1}$, gives the predictions used in practice.

We cannot use an infinite number of parameters in M, so we truncate when the 'settling time' of the plant is reached and hope that this leads to at most 30-50 elements of θ. This is fine if the plant is well-damped and Type 0 (the first example), but not for the other cases. A model which has a more general application is the Controlled Autoregressive and Integrated Moving Average (CARIMA) form as used in GPC:

$$A(q^{-1})y(t) = B(q^{-1})u(t) + \frac{T(q^{-1})}{\Delta}\xi(t), \qquad (6)$$

where the polynomial A should capture at least the unstable or lightly damped plant poles. In practice the rule should be 'to use a low-order A and a high-order B', for then strange phase behaviour due to material transport effects can be modelled adequately. Even a first-order A will give a good fit to 'long-term' exponential behaviour in M, leaving B to look after the initial transients. This implies that a significantly reduced number

of parameters can be used compared with simple step/impulse models.

A denominator of Δ in the noise term is widely assumed, leading as it does to integral/incremental control laws, though other polynomials are possible if clear prior knowledge is available about the noise characteristics as in UPC. The polynomial T could be used for optimal disturbance rejection, though its role in robustness enhancement is more convincing. Comparing the GPC model eqn.6 with the basic model eqn.3 we find that:

$$M(q^{-1}) = \frac{B(q^{-1})}{A(q^{-1})} \quad \text{and} \quad N(q^{-1}) = \frac{T(q^{-1})}{A(q^{-1})\Delta}.$$

'New' variants of GPC are proposed now and then to replace the Diophantine recursion of the original papers. In practice the predictions of GPC can be computed as in Clarke and Mohtadi (1989):

Forced response : $fo(t+j|t) = \sum_{i=1}^{j} g_i \Delta u(t+j-i|t)$, where $\{g_i\}$ is the set of step-response coefficients and the u's are the projected future controls;

Free response : $fr(t+j|t)$ is evaluated based on zero future values of $\xi(t+i)$ and of a constant control, so that $u^*(t+i|t) = u(t-1), i \geq 0; = u(t+i), i < 0$. Filtered signals u^f and y^f are maintained using $y^f(t) = y(t)/T(q^{-1})$, A_1 is defined by $A_1 = A\Delta$, and the recursion:

$$A_1(q^{-1})y^f(t+j|t) = B(q^{-1})\Delta u^{*f}(t+j|t)$$

is computed. The actual prediction $fr(t+j|t)$ then is deduced from:

$$T(q^{-1})y^f(t+j|t) = fr(t+j|t).$$

This leads to an efficient algorithm for computing the free response.

Given the model structure the parameters θ need to be estimated. For DMC simple step tests can be applied, though more recently the usefulness of PRBS probing and cross-correlation has been rediscovered. With general input signals one of the many variants of recursive least squares can be employed, as in self-tuning control. Here the current model is used to 'predict the present', giving a prediction error:

$$\begin{aligned} \epsilon(t) &= y(t) - \hat{y}(t|t-1) \qquad (7) \\ &= y(t) - f(\mathbf{u}(t-1), \mathbf{y}(t-1), \hat{\theta}(t-1)), \end{aligned}$$

where $f(.)$ depends on the model structure. RLS then chooses $\hat{\theta}$ to minimise $\sum \epsilon^2$ over the past data. Successful estimation depends crucially on data filtering $T_e(q^{-1})$ and typically band-pass filters are used which accentuate the signal energy in the critical frequency range (e.g. around the design bandwidth of the closed loop).

MBPC and parameter estimation are dual problems: for the control action we choose $\tilde{u}$ to minimise a cost defined over a given future horizon, whilst to estimate we choose $\hat{\theta}$ to minimise a cost defined over a given past horizon. As using $\epsilon(t)$ in RLS essentially minimises one-step-ahead variance, the estimated parameters may be less effective for the multi-step objectives of MBPC. LRPI (Shook, Mohtadi and Shah, 1991, 1992) makes use of the dual nature of control/estimation by minimising a cost depending on the envisaged multi-step predictions. This gives a guide to what filters T_e are most effective.

3 COST FUNCTIONS, PERFORMANCE AND ROBUSTNESS

There are many cost functions used in MBPC, mostly some version of the quadratic form:

$$\begin{aligned} J_{GPC} &= \sum_{j=N1}^{N2} \mu_j (r(t+j) - \hat{y}(t+j|t))^2 \\ &+ \sum_{j=1}^{NU} \lambda_j \Delta u(t+j-1)^2, \qquad (8) \end{aligned}$$

where N_1, N_2 are the *costing horizons* and NU is the *control horizon*. The coefficients μ_j, λ_j give weightings (perhaps 0, 1, or exponentially increasing) on future behaviour. The reference sequence $r(t+j)$ may be equal to the current set-point $w(t)$ for all j, or equal to a known 'future set-point' sequence, or be generated by a simple recursion which starts at the current output $y(t)$ and tends exponentially to the setpoint $w(t)$ for large j (the *reference trajectory*). The predictions $\hat{y}$ could be as described above, or of the output $\phi(t+j)$ of an *auxiliary model* $\phi(t) = P(q^{-1})y(t)$, where P is chosen to improve some aspects of performance, such as minimising overshoot.

Of course the cost-function J is only a convenient and algorithmically tractable token for what we really wish to achieve. For large NU, N_2 its minimisation could lead to minimum-variance control – often a real objective. But in general what we are after are the usual rather loose indicators of 'good' control: stability, margins, 'tight' control, damping, bandwidth, etc. Fortunately experience is growing about the affect of the choice of the 'tuning knobs' $N_1, \cdots$ on these indicators, reflected in statements such as 'the more the short-horizon cost is ignored compared with the long-range cost, the more smooth and robust the control will become'. The quadratic nature of the cost is also a convenience, leading to analytical solutions, to stability proofs, and to frequency-domain interpretations. If, however, inequality constraints are frequently active, there might be algorithmic advantages in using LP optimization with a modulus rather than a quadratic objective.

With J_{GPC} above there are $D_e = N_2 - N_1 + 1$ 'degrees of freedom' in the future system errors, so we cannot expect to be able to calculate more than D_e independent values of control (e.g. if $\lambda = 0$ then we should be able to chose a set of controls $\{u(t+j)\}$ to make J precisely 0). To get 'nice' solutions we impose some *structure* onto our control policy in order to reduce the number of degrees of freedom D_u to be less than D_e. For example in 'mean-level' control we impose $D_u = 1$ by considering a constant future level of control $u(t+j) = u(t), \forall j > 0$ and hence compute the 'control increment sequence': $\tilde{\mathbf{u}} = [\Delta u(t), 0, 0, \cdots]$. We could, for example, with PFC consider a set of weighted basis functions:

$$\mathbf{u} = \sum_{i=1}^{D_u} \alpha_i \mathbf{u}_i,$$

where $\mathbf{u}_i$ is predetermined. In GPC, as in DMC, we assume a control horizon NU beyond which increments are zero, giving the vector of postulated control increments:

$$\tilde{\mathbf{u}} = [\Delta u(t), \Delta u(t+1), \cdots, \Delta u(t+NU-1), 0, 0, \cdots]'.$$

The cost-function can be augmented by end-point state weighting (GPCW - Demircioglu and Clarke, 1993) or by terminal constraints (CRHPC), and still lead to an analytic solution using Lagrange multipliers. We will consider the CRHPC case to see how a solution is obtained. The prediction equations become:

Costing range : $\hat{\mathbf{y}} = \mathbf{fo} + \mathbf{fr} = \mathbf{G}\tilde{\mathbf{u}} + \mathbf{fr}$ where $\hat{\mathbf{y}} = [\hat{y}(t+N_1|t), \ldots, \hat{y}(t+N_2|t)]'$;

Constraint range : $\hat{\mathbf{y}}_c = \mathbf{G}_c\tilde{\mathbf{u}} + \mathbf{fr}_c$, where $\hat{\mathbf{y}}_c = [\hat{y}(t+N+1|t), \ldots, \hat{y}(t+N+m|t)]'$.

The dimension of the vector $\hat{\mathbf{y}}$ is D_e and that of $\tilde{\mathbf{u}}$ is $D_u = NU$. The 'dynamic matrices' are $\mathbf{G}$ and correspond to the superposition of individual step responses $\{g_i\}$ due to the control increments. For simplicity, put $\mu_j = 1, \lambda_j = \lambda$ and let $\mathbf{w}$ be the vector of future reference values. Then if μ is a vector of Lagrange multipliers, the augmented cost of CRHPC becomes:

$$J(\tilde{\mathbf{u}}, \mu) = \mathbf{e}'\mathbf{e} + \lambda\tilde{\mathbf{u}}'\tilde{\mathbf{u}} - 2\mu'\mathbf{e}_c,$$

$$\text{where } \mathbf{e}_c = \mathbf{w}_c - \hat{\mathbf{y}}_c = \mathbf{w}_c - \mathbf{G}_c\tilde{\mathbf{u}} - \mathbf{fr}_c,$$

and the system error $\mathbf{e}$ is $\mathbf{w} - \hat{\mathbf{y}} = \mathbf{w} - \mathbf{G}\tilde{\mathbf{u}} - \mathbf{fr}$. Hence:

$$\begin{aligned} J(\tilde{\mathbf{u}}, \mu) &= (\mathbf{w} - \mathbf{G}\tilde{\mathbf{u}} - \mathbf{fr})'(\mathbf{w} - \mathbf{G}\tilde{\mathbf{u}} - \mathbf{fr}) \\ &\quad + \lambda\tilde{\mathbf{u}}'\tilde{\mathbf{u}} - 2\mu'\mathbf{e}_c \\ &= \tilde{\mathbf{u}}'\mathbf{G}'\mathbf{G}\tilde{\mathbf{u}} - 2\tilde{\mathbf{u}}'\mathbf{G}'(\mathbf{w} - \mathbf{fr}) \\ &\quad + (\mathbf{w} - \mathbf{fr})'(\mathbf{w} - \mathbf{fr}) + \lambda\tilde{\mathbf{u}}'\tilde{\mathbf{u}} - 2\mu'\mathbf{e}_c. \end{aligned}$$

To minimise this we set gradients to 0:

$$\begin{aligned} \nabla_{\tilde{\mathbf{u}}} = 0 \quad &: \quad 2\mathbf{G}'\mathbf{G}\tilde{\mathbf{u}} - 2\mathbf{G}'(\mathbf{w} - \mathbf{fr}) \\ &\quad + 2\lambda\tilde{\mathbf{u}} + 2\mathbf{G}_c'\mu = 0 \\ \nabla_{\mu} = 0 \quad &: \quad \mathbf{e}_c = 0 = \mathbf{w}_c - \mathbf{G}_c\tilde{\mathbf{u}} - \mathbf{fr}_c. \end{aligned}$$

This can be rearranged into the following set of equations:

$$\begin{bmatrix} \mathbf{G}'\mathbf{G} + \lambda\mathbf{I} & \mathbf{G}_c' \\ \mathbf{G}_c & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}} \\ \mu \end{bmatrix} = \begin{bmatrix} \mathbf{G}'(\mathbf{w} - \mathbf{fr}) \\ \mathbf{w}_c - \mathbf{fr}_c \end{bmatrix}. \tag{9}$$

An explicit, and fairly complex, formula for $\tilde{\mathbf{u}}$ is available using results in Fletcher (1990) and in the original CRHPC paper. However, as only the first element of the control vector is required it is surely better to use direct elimination in eqn.9 (the explicit formula involves matrix inversions which are badly conditioned for open-loop unstable plant). We note that the equation is for $NU + m$ variables, corresponding to the degrees-of-freedom NU in the control and the m terminal constraints $\mathbf{e}_c = \mathbf{0}$.

Similar derivations are used for cases with no terminal constraint (e.g. GPC) and with end-point state weighting where J is augmented by a term $[\hat{\mathbf{x}}(t+N) - \tilde{\mathbf{w}}]'\mathbf{Q}[\hat{\mathbf{x}}(t+N) - \tilde{\mathbf{w}}]$. Here $\mathbf{x}$ is a vector of n successive values of the 'partial state', defined by the recursion $A(q^{-1})x(t) = u(t-1)$. We note that *constraining* the end-point state to $\tilde{\mathbf{w}}$ makes the algorithm similar to CRHPC, but further benefits can be obtained by simply having a non-infinite cost on the state, leading to GPCW. A further possibility is to consider the behaviour of the closed loop at $t = \infty$ and add extra terms to the cost J which weight steady-state errors.

(Sufficient) proofs of stability depend on showing that for noise-free case with $r = 0$ the value of $J(t)$ decreases with t, so that J is a Liapunov function (see, for example, Kouvaritakis, Rossiter and Chang, 1992). This approach stems from the early finite horizon results of Kleinman (1970), Kwon and Pearson (1975) and rely on careful choice of terminal constraints or on 'long enough' horizons. It is interesting that the proofs can be extended to nonlinear models and to problems with state *inequality* constraints (Mayne and Michalska, 1990).

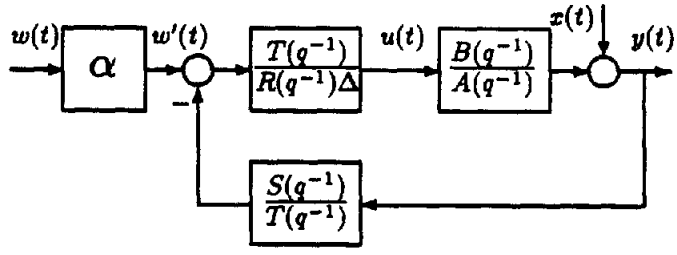

Figure 3: 2 DOF structure of GPC

Suppose that the reference sequence $r(t+j) = w(t)$, where $w(t)$ is the current set-point. Then for

most types of costing used in MBPC when minimised for incremental controls $\tilde{u}$, it can be shown that the closed loop satisfies:

$$\alpha T(q^{-1})w(t) = R(q^{-1})\Delta u(t) + S(q^{-1})y(t), \quad (10)$$

where the constant α multiplying the set-point is due to the guaranteed unity closed-loop gain. The equivalent MBPC closed-loop in terms of these *fixed* polynomials is shown in Fig.3. The polynomials R, S are solutions of the Diophantine equation:

$$R(q^{-1})A(q^{-1})\Delta + B(q^{-1})S(q^{-1}) = P_c(q^{-1})T(q^{-1}). \quad (11)$$

The polynomial P_c gives the closed-loop poles and is a complex function of all the design horizons and weights adopted: many papers give expressions for P_c though often the easiest computation is by iterating the associated Riccati equation. Indeed, some approaches to MBPC *prescribe* P_c and deduce appropriate weights to put into the cost function, but this seems to be rather excessive. What we are after is (*inter alia*) the time-domain response which also depends on the zeros. Hence most MBPC afficionados know what design settings give 'special' values of P_c and the corresponding effect of modifying these settings on the transient behaviour. In particular the following are used as base-lines (Clarke, Mohtadi and Tuffs, 1987):

Mean-level : $NU = 1, N2 \rightarrow \infty$ gives $P_c = A$. This simple control law does not attempt to shift the poles from the open-loop positions. Nevertheless if $T = 1$ disturbance rejection is state-dead-beat. Response is 'slow' to set-point changes, but 'fast' to disturbances.

State-dead-beat : Here $P_c = 1$: as fast as can be. This arises from many combinations of design knobs. With GPC $NU = n = N_1, N_2 \geq 2n - 1, \lambda = 0$, where n is the order of the plant plus 1 due to the use of integral control. With CRHPC $N = n - 1, m = n$. With GPCW $NU = N_2 \leq n, \mathbf{Q} = \alpha \mathbf{I}$ with $\alpha \rightarrow \infty$, etc.

State-dead-beat control drives the state (system errors plus derivatives) to the origin in the minimal number of steps, which is ensured by the imposition of the terminal equality constraints. GPC is able to achieve the same by a solution which gives $J = 0$. Moving the future time for achievement of rest at the origin (e.g. by increasing N) is a simple way of 'de-tuning' the loop.

Models are invariably wrong, so realistic design procedures must include some mechanism for providing robustness against mis-match (Robinson and Clarke, 1991, Yoon and Clarke, 1995). In many cases this is done by de-tuning: increasing target horizons, reducing D_u or increasing the control weight λ. All these affect the nominal response to set-point changes. However the 2DOF structure of MBPC can be exploited in that only the disturbance rejection depends on the polynomial T. Moreover T shapes robustness, for if A_o, B_o are the *real* plant polynomials, then the *real* characteristic equation remains stable if A, A_o have the same number of unstable roots and:

$$\left|\frac{B(e^{-j\omega})}{A(e^{-j\omega})} - \frac{B_o(e^{-j\omega})}{A_o(e^{-j\omega})}\right| < \left|\frac{P_c(e^{-j\omega})}{A(e^{-j\omega})}\right| \cdot \left|\frac{T(e^{-j\omega})}{S(e^{-j\omega})}\right|, \quad (12)$$

for all $\omega \in (0, \pi)$. This can be proved, for example, by the small-gain theorem. The bound of eqn.12 guarantees stability if the mismatch at any frequency ω is less than some value determined by the design polynomial T. The role of T can be seen intuitively by inspecting the equivalent loop of Fig.4, (where M^* is not free but depends on the other loop polynomials). There are several guide-

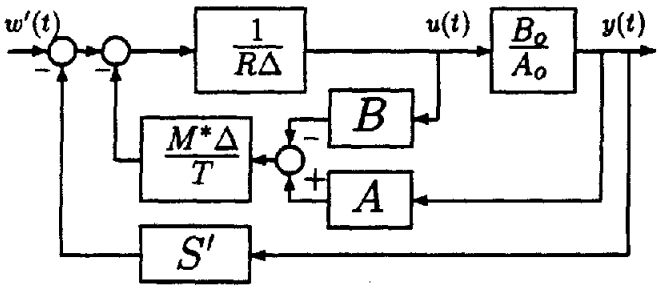

Figure 4: Equivalent structure for robustness calculations

lines which give a 'good' choice of T (see Yoon and Clarke 1995). For example $T = A(1 - \beta q^{-1})^{N1}$ where β is in the neighbourhood of the dominant pole of A gives results remarkably close to an H^∞ design using the Youla parametrisation (where M^*/T in Fig.4 is replaced by an arbitrary transfer function Q). In practice $\beta = 0.8$ always seems to work...

4 CONSTRAINTS

Practical constraints are imposed say by saturation and rate limits in actuators, or by prespecified requirements on output or state variables, such as the maximum acceptable temperature in a catalytic cracker. Hence some are inevitable, some mandatory, some desirable, some optional and so on. The general formulation considers one or more auxiliary variables $v^i(t)$ which are modelled by:

$$A^i(q^{-1})v^i(t) = B^i(q^{-1})u(t - d^i) + x^i(t).$$

For example if $A^i = 1$ and $B^i = \Delta$, $v(t)$ corresponds to the control moves $\Delta u(t)$. Constraints on $v(t)$ are typically of the form $V_1 \leq v(t) \leq V_2$, but in predictive control we can look ahead to see if the *current* control action set will induce violations of *future* constraints, i.e. $V_1 \leq v(t + j|t) \leq V_2$. Each

future constraint adds to the dimensionality of the problem, so a *constraint range* is adopted such that only $j \in (NC_1, NC_2)$ is considered. Suitable manipulation then leads to a QP problem for which a cost such as J in eqn.8 is to be minimised, subject to $\mathbf{C}\bar{\mathbf{u}} - \mathbf{d}(t) \geq \mathbf{0}$.

Because QP is computationally expensive, the most efficient means of solution is a matter of urgent research, particularly for fast systems. One approach is Fletcher's active set method which first discovers those constraints above satisfying *equality* and then uses an algorithm like eqn.9. For $NC_2 \rightarrow \infty$, the CHRPC and GPC$^\infty$ stability results are extensible to the constrained case, *provided* that there is a feasible solution. Indeed, infeasibilities are likely to be increasingly present as profitability pressures drive plants harder toward constraints, for then disturbances could move the outputs into 'forbidden regions'. The use of equality terminal constraints as in CRHPC to guarantee stability further complicates the feasibility problem, as they reduce the number of degrees of freedom in the control set available for satisfying the desired inequality constraints. Hence the 'best' way of tackling infeasibilities remains an important practical issue in MBPC (see Rossiter and Kouvaritakis, 1993).

5 RECURSIVE LEAST SQUARES AND UDU

RLS, applied in nearly all practical self-tuning control designs (see, e.g., Harris and Billings, 1981; Mosca, 1995), is used for estimating parameters in the model:

$$y(t) = \theta' \mathbf{x}(t) + e(t), \tag{13}$$

where $\theta, \mathbf{x}$ are n-vectors. The update equation is:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \mathbf{S}^{-1}(t)\mathbf{x}(t)\epsilon(t), \tag{14}$$

where $\epsilon(t)$ is the prediction error:

$$\epsilon(t) = y(t) - \hat{\theta}(t-1)'\mathbf{x}(t), \text{ and} \tag{15}$$

$$\mathbf{S}(t) = \mathbf{S}(t-1) + \mathbf{x}(t)\mathbf{x}'(t). \tag{16}$$

As is well-known, the normal update is in terms of propagating an *inverse* $\mathbf{P}(t) = \mathbf{S}^{-1}(t)$ via the RLS equations:

$$\mathbf{k}(t) = \frac{\mathbf{P}(t-1)\mathbf{x}(t)}{1 + \mathbf{x}'(t)\mathbf{P}(t-1)\mathbf{x}(t)} \tag{17}$$

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \mathbf{k}(t)\epsilon(t) \tag{18}$$

$$\mathbf{P}(t) = [\mathbf{I} - \mathbf{k}\mathbf{x}'(t)]\mathbf{P}(t-1), \tag{19}$$

where the 'covariance update' is so named as $\mathbf{P}$ is proportional to the covariance matrix of the estimated parameters.

One major problem in applications is that eqn.19 is ill-conditioned, and implementation using these equations directly often leads to negative-definite $\mathbf{P}$ and then to inaccurate results. As $\mathbf{P}$ *must* be positive-definite, the common solution is to use a factored form of $\mathbf{P}$ and update the factors rather than the full matrix. One approach is the UDU form in which we write:

$$\mathbf{P}(t) = \mathbf{U}(t)\mathbf{D}(t)\mathbf{U}'(t), \tag{20}$$

where $\mathbf{D}$ is a diagonal matrix and $\mathbf{U}$ is an upper-triangular matrix with unit entries along the diagonal. Define $\bar{\mathbf{U}}, \bar{\mathbf{D}}$ as $\mathbf{U}(t-1), \mathbf{D}(t-1)$ respectively and insert the definitions into eqn.19 to get:

$$\begin{aligned}\mathbf{UDU}'(t) &= \bar{\mathbf{U}}\bar{\mathbf{D}}\bar{\mathbf{U}}' - \frac{\bar{\mathbf{U}}\bar{\mathbf{D}}\bar{\mathbf{U}}'\mathbf{x}(t)\mathbf{x}'(t)}{1 + \mathbf{x}'(t)\bar{\mathbf{U}}\bar{\mathbf{D}}\bar{\mathbf{U}}'\mathbf{x}(t)}\bar{\mathbf{U}}\bar{\mathbf{D}}\bar{\mathbf{U}}' \\ &= \bar{\mathbf{U}}\left[\bar{\mathbf{D}} - \frac{\bar{\mathbf{D}}\bar{\mathbf{U}}'\mathbf{x}\mathbf{x}'\bar{\mathbf{U}}\bar{\mathbf{D}}}{1 + \mathbf{x}'\bar{\mathbf{U}}\bar{\mathbf{D}}\bar{\mathbf{U}}'\mathbf{x}}\right]\bar{\mathbf{U}}'.\end{aligned}$$

Define the vectors $\mathbf{f} = \bar{\mathbf{U}}'\mathbf{x}, \mathbf{g} = \bar{\mathbf{D}}\mathbf{f}$, this gives:

$$\mathbf{UDU}'(t) = \bar{\mathbf{U}}\left[\bar{\mathbf{D}} - \frac{\mathbf{g}\mathbf{g}'}{1 + \mathbf{f}'\mathbf{g}}\right]\bar{\mathbf{U}}'.$$

If the term within the brackets can be decomposed as $\mathbf{U}_n\mathbf{D}_n\mathbf{U}_n'$, then the update is:

$$\mathbf{UDU}'(t) = \bar{\mathbf{U}}\mathbf{U}_n\mathbf{D}_n\mathbf{U}_n'\bar{\mathbf{U}}'$$

$$\rightarrow \mathbf{D}(t) = \mathbf{D}_n, \mathbf{U}(t) = \bar{\mathbf{U}}\mathbf{U}_n.$$

Hence the RLS equations can be implemented using updating of $\mathbf{U}, \mathbf{D}$, as suggested originally by Thornton and Bierman (1978). We note that the basic problem reduces to factorizing:

$$\bar{\mathbf{D}} - \frac{\mathbf{g}\mathbf{g}'}{1 + \mathbf{f}'\mathbf{g}} \text{ into: } \mathbf{UDU}',$$

where $\mathbf{f} = \bar{\mathbf{U}}'\mathbf{x}, \mathbf{g} = \bar{\mathbf{D}}\mathbf{f}$ are n-vectors and $\mathbf{D}, \bar{\mathbf{D}}$ are $n \times n$ diagonal matrices. To assist the development we first define some notation:

$$\begin{aligned} d_i &= i^{th} \text{ element of } \mathbf{D} \\ g_i &= i^{th} \text{ element of } \mathbf{g} \\ \mathbf{g}^i &= [g_1\, g_2 \ldots g_i\, 0 \ldots 0]' \text{ i.e. } \mathbf{g} = \mathbf{g}^n \\ \mathbf{e}_i &= i^{th} \text{ unit-vector: } [0 \ldots 1 \ldots 0]' \\ \mathbf{U} &= [\mathbf{u}^1\, \mathbf{u}^2 \ldots \mathbf{u}^n], \text{ where } u_i^i = 1 \end{aligned}$$

and note that $\mathbf{UDU}' = \sum_1^n d_i \mathbf{u}^i \mathbf{u}^{iT}$.

6 SIMULTANEOUS ESTIMATION OF MODELS

We start with some basic LS theory for the 'batch' (off-line) case and consider the *linear-in-the-parameters* model:

$y(t) = \theta_n' \mathbf{x}(t) + e(t)$ with θ_n, $\mathbf{x}$ being $n \times 1$ vectors,

for which m stacked equations corresponding to successive time samples are:

$\mathbf{y} = \mathbf{X}\theta_n + \mathbf{e}$ with $\mathbf{y}, \mathbf{e}$ being $m \times 1$ and $\mathbf{X}$ being $m \times n$.

The 'normal' LS solution minimises $J = \mathbf{e}'\mathbf{e}$ using $\nabla J_\theta = 0$, so:

$$\hat{\theta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}. \tag{21}$$

We have that $\hat{\mathbf{y}} = \mathbf{X}\hat{\theta}$ is the model's prediction and if $\tilde{\theta} = \theta - \hat{\theta}, \tilde{\mathbf{y}} = \mathbf{y} - \hat{\mathbf{y}}$ we get the residual sequence:

$$\begin{aligned} \tilde{\mathbf{y}} &= \mathbf{X}\theta + \mathbf{e} - \mathbf{X}\hat{\theta} = \mathbf{X}\tilde{\theta} + \mathbf{e} \\ &= \mathbf{X}\theta + \mathbf{e} - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'[\mathbf{X}\theta + \mathbf{e}] \\ &= [\mathbf{I} - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}']\mathbf{e}. \end{aligned}$$

By multiplication by $\mathbf{X}'$ we see that $\mathbf{X} \perp \tilde{\mathbf{y}}$: 'the data is orthogonal to the error'.

We now consider model reformulation to allow for simultaneous estimation. The conventional use of RLS/UDU for parameter estimation with a difference-equation model:

$$y(t) + a_1 y(t-1) + \cdots + a_{na} y(t-na) =$$

$$b_1 u(t-1) + \cdots + b_{nb} nu(t-nb)$$

simply moves the terms $y(t-i)$ to the RHS of the standard form:

$$y(t) = \theta' \mathbf{x}(t),$$

to give:

$$\begin{aligned} \mathbf{x} &= [-y(t-1), \ldots, -y(t-na), \\ &\quad u(t-1), \ldots, u(t-nb)]' \\ \theta &= [a_1, \ldots, a_{na}, b_1 \ldots, b_{nb}]'. \end{aligned}$$

The ingenious idea of Niu and Fisher (1992, 1993) is to move *all* the y's to the RHS and to *interleave* the data and parameters so that, taking the case $na = k = nb$ and $n = 2k$:

$$\begin{aligned} \phi &= [-y(t-k), u(t-k), -y(t-k-1), \\ &\quad u(t-k-1), \ldots - y(t-1), u(t-1), -y(t)]' \\ \theta &= [a_k, b_k, a_{k-1}, b_{k-1}, \ldots a_1, b_1, 1]' \end{aligned}$$

In particular we note that the $(n+1)^{th} = (2k+1)^{th}$ parameter is simply 1 - the coefficient associated with the current output $y(t)$.

Hence we have rewritten the model in the form:

$$0 = \theta'_{n+1}\phi(t),$$

where $\theta'_{n+1} = [\theta_n \; 1]$ and $\phi' = [\mathbf{x}' \; -y(t)]$. Then the stacked equations for LS become:

$$\mathbf{0}_m = [\mathbf{X} \; -\mathbf{y}] \begin{pmatrix} \theta_n \\ 1 \end{pmatrix} + \mathbf{e} = \mathbf{Z}\theta_{n+1} + \mathbf{e}.$$

We can use Lagrange multipliers to minimise $J(\hat{\theta}_{n+1}, \mu) = \frac{1}{2}\mathbf{e}'\mathbf{e} + \mu(\hat{\theta}_{n+1} - 1)$, as we know that $\hat{\theta}_{n+1} = 1$. Minimising via $\nabla_\theta = 0$ gives:

$$\mathbf{Z}'\mathbf{Z}\hat{\theta}_{n+1} + \mu\mathbf{e}_{n+1} = \mathbf{0}_{n+1},$$

where $\mathbf{e}_{n+1} = [0\,0 \cdots 1]'$. This solution can be written as:

$$\mathbf{S}_{n+1}\hat{\theta}_{n+1} = \begin{bmatrix} \mathbf{0}_n \\ -\mu \end{bmatrix},$$

or in partitioned form:

$$\begin{bmatrix} \mathbf{X}'\mathbf{X} & -\mathbf{X}'\mathbf{y} \\ -\mathbf{y}'\mathbf{X} & \mathbf{y}'\mathbf{y} \end{bmatrix} \begin{vmatrix} \hat{\theta}_n \\ 1 \end{vmatrix} = \begin{bmatrix} \mathbf{0}_n \\ -\mu \end{bmatrix} \tag{22}$$

Note the similarity between the structure of eqn.22 and that of eqn.9 in predictive control. The first n equations of eqn.22 give the solution eqn.21 and the last equation the value of μ:

$$\begin{aligned} -\mu &= -\mathbf{y}'\mathbf{X}\hat{\theta}_n + \mathbf{y}'\mathbf{y} \\ &= \mathbf{y}'\tilde{\mathbf{y}} = (\theta'\mathbf{X}' + \mathbf{e}')\tilde{\mathbf{y}} = \mathbf{e}'\tilde{\mathbf{y}} \\ &= \mathbf{e}'[\mathbf{I} - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}]\mathbf{e} \\ &= \tilde{\mathbf{y}}'\tilde{\mathbf{y}}. \end{aligned}$$

That is $-\mu$ is the loss function of the estimated model $J(\hat{\theta}_n) = J_n$, say. Now from the theory of matrices we know the inverse of a partitioned matrix:

$$\begin{bmatrix} \mathbf{A} & \mathbf{v} \\ \mathbf{v}' & \alpha \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1}(\mathbf{I} - \mathbf{v}\mathbf{v}'\mathbf{A}^{-1}/\beta) & -\mathbf{A}^{-1}\mathbf{v}/\beta \\ -\mathbf{v}'\mathbf{A}^{-1}/\beta & 1/\beta \end{bmatrix}, \tag{23}$$

where $\beta = \alpha - \mathbf{v}'\mathbf{A}^{-1}\mathbf{v}$. Applying this result to eqn.22 we get:

$$\begin{bmatrix} \hat{\theta}_n \\ 1 \end{bmatrix} = \mathbf{S}_{n+1}^{-1} \begin{bmatrix} \mathbf{0}_n \\ -\mu \end{bmatrix}$$

$$= \begin{bmatrix} . & (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}/\beta \\ \mathbf{y}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}/\beta & 1/\beta \end{bmatrix} \begin{bmatrix} \mathbf{0}_n \\ -\mu \end{bmatrix}$$

$$= \begin{bmatrix} . & \hat{\theta}_n/\beta \\ \hat{\theta}'_n/\beta & 1/\beta \end{bmatrix} \begin{bmatrix} \mathbf{0}_n \\ -\mu \end{bmatrix}, \tag{24}$$

where $\beta = \mathbf{y}'\mathbf{y} - \mathbf{y}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} = J_n$. The solution from eqn.24 is clearly the same as before, but we note the following crucial points:

1. the last column of $\mathbf{S}_{n+1}^{-1}$ is $[\hat{\theta}'_n/\beta \;\; 1/\beta]'$;
2. β is simply J_n, the loss function of the nth order model.

Hence the inverse of $\mathbf{S}_{n+1}$ gives, as its RH column *both* the parameter estimates *and* the associated loss function. Hence we turn to the UDU method to evaluate the inverse.

7 USING THE UDU METHOD

Define $\mathbf{C}_{n+1} = \mathbf{S}_{n+1}^{-1}$ and decompose as $\mathbf{UDU}'$.

$$\text{i.e. } \mathbf{C}_{n+1} = \sum_{i=1}^{n+1} d_i \mathbf{u}^i \mathbf{u}^{iT},$$

$$\text{with } \mathbf{u}^i = [u_1^i\, u_2^i \ldots u_i^i\, 0\, 0\, 0]', u_i^i = 1,$$

and look along the bottom row and RH column. Clearly then, by comparing with eqn.24:

$$d_{n+1} = 1/\beta = 1/J_n \qquad (25)$$

$$\mathbf{u}^{n+1} = \hat{\theta}_{n+1} = \begin{pmatrix} \hat{\theta}_n \\ 1 \end{pmatrix}. \qquad (26)$$

Note that the matrix in eqn.22 can be written:

$$\begin{bmatrix} \mathbf{A} & \mathbf{v} \\ \mathbf{v}' & \alpha \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{0}_n \\ \mathbf{0}_n' & 0 \end{bmatrix} + \frac{1}{\beta} \begin{pmatrix} \mathbf{w} \\ 1 \end{pmatrix} (\mathbf{w}'\ 1)$$

$$\text{Hence } \mathbf{C}_{n+1} = \begin{bmatrix} \mathbf{C}_n & \mathbf{0}_n \\ \mathbf{0}_n' & 0 \end{bmatrix} + d_{n+1} \mathbf{u}^{n+1} \mathbf{u}^{n+1T}.$$

$\mathbf{C}_n = \mathbf{S}_n^{-1}$ corresponds to the least-squares solution when the right-hand column of $\mathbf{Z}$ has been eliminated, i.e. with one fewer parameter. The sequence of operations is therefore as before but with the model:

$$0 = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n + e,$$

where now $\theta_n = 1$. Hence, by induction, for all i in $(0, n)$:

1. $d_{i+1} = 1/J_i$, the i-parameter model loss;
2. $\mathbf{u}^{i+1}$ is a vector containing the estimates of the i-parameter model, then 1, then zeros.

The i-parameter model is:

$$0 = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_i x_i + x_{i+1} + e,$$

or, in input-output terms:

$$-x_{i+1} = \theta_1 x_1 + \theta_2 x_2 + \theta_i x_i + e.$$

Suppose for example $i = n - 2 = 2k - 2$. Then

$$\mathbf{x}' = [-y(t-k), u(t-k), \ldots$$

$$, -y(t-2), u(t-2), -y(t-1)].$$

So the model is:

$$0 = -\theta_1 y(t-k) + \theta_2 u(t-k) + \cdots$$

$$+ \theta_{2k-2} u(t-2) - y(t-1),$$

and by simply shifting time along by one this is:

$$y(t) + \theta_{2k-3} y(t-1) + \cdots + \theta_1 y(t-k-1) =$$

$$\theta_{2k-2} u(t-1) + \cdots + \theta_2 u(t-k-1).$$

This is a difference-equation model of order $k-1$. Similarly with $i = n - 4$ we obtain a difference-equation model of order $k-2$, and so on.

Hence the procedure is to choose a maximum possible order k_{max} for our model and interleave $n = 2k_{max} + 1$ past I/O data into the ϕ-vector. Then the UDU updating procedure gives:

Table 1: UDU method applied to simulated data

The U matrix

1	0.144	-0.743	-.005	0.7	.11	.325	.066	.057
0	1	1.74	-.978	0.5	.097	.232	.062	.041
0	0	1	.0050	-1.5	-.25	.003	.004	.229
0	0	0	1	0.8	-.85	.871	.19	.317
0	0	0	0	1	.17	-1.04	-.22	.028
0	0	0	0	0	1	0.8	-.81	.90
0	0	0	0	0	0	1	.21	-.99
0	0	0	0	0	0	0	1	.8
0	0	0	0	0	0	0	0	1

The D diagonal (inverse)

4.3e+003 9.8 27 4 0.000046 4 0.000036 4 0.000036

1. all the reduced order models $k = 1 \ldots k_{max}$ in the u-vectors, spaced by 2
2. (inverses of) the corresponding loss-functions in the **D** diagonal entries.

As an example of the use of the method, the plant:

$$y(t) - 1.5y(t-1) + 0.7y(t-2) = 0.8u(t-1) + 0.5u(t-2)$$

was simulated with a square-wave input and no noise. A (maximum of) 4^{th} order model was estimated using the UDU method, giving the results shown in the Table. Looking at the loss functions given by $\mathbf{D}_i^{-1}$ we see that the first 'good' model is of 2^{nd} order. Higher-order models involve pole/zero cancellation. For example, the 4^{th} order model:

$$y(t) - 0.997y(t-1) + 0.0284y(t-2) + 0.229y(t-3) + \cdots$$

$$= 0.8u(t-1) + 0.902u(t-2) + \cdots$$

has poles at $0.75 \pm 0.371j, -0.2514 \pm 0.1375j$ and zero at $-0.625, -0.2512 \pm 0.0138j$. Hence the 'true' poles are correctly estimated and the other singularities approximately cancel. The parameter trajectories for the 4^{th} order model are shown in Fig.5. These stay at zero until input excitation arises and rapidly move to the converged values.

8 FIDDLE FACTORS

The nice and especially useful property of the **D** entries is they are the inverse of the loss-functions. In particular:

1. $d_1 = 1/\sum_1^N y(t)^2$
2. $d_{n+1} = 1/\sum_1^N \epsilon^2(t)$, where ϵ is the one-step prediction error of the n-parameter model.

In particular it is worth noting that d_1 corresponds to the mean-square plant output, and that

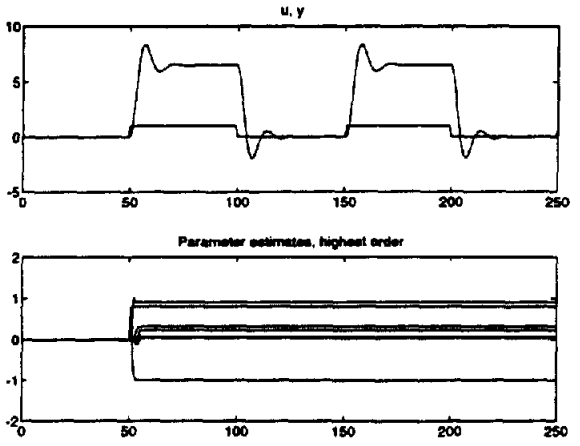

Figure 5: I/O data and parameter trajectories

d_1/d_n is the normalised mean-square 'explanation error' (or prediction accuracy) of the model, being the fraction of ms output not generated by the predicted output. For the noise-free/full-parameter case the ratio will tend to 0 (i.e. the 'signal/residual' ratio tends to ∞).

Consider now the *conditioning* of the update, which depends on the condition number of $\mathbf{S}$, for it can be shown that for the noise-free case:

$$\tilde{\theta}(t) = \mathbf{S}(t)^{-1}\mathbf{S}(t-1)\tilde{\theta}(t-1),$$

and that:

$$||\tilde{\theta}(t)|| \leq \kappa_{t-1}||\tilde{\theta}(t-1)||.$$

It is known with 'ordinary' RLS that $\mathbf{P}$ can become ill-conditioned (and indeed lose rank if the model is over-parameterised). With the UDU approach the conditioning depends strongly on that of the diagonal matrix $\mathbf{D}$ and in particular on the 'signal/residual' ratio described above. Imposing an upper bound on this ratio guarantees the conditioning and has a simple interpretation as the 'required accuracy' of the model. This is equivalent to the method of *matrix regularization* often used in RLS, which bounds the eigenvalues of $\mathbf{P}$. Note further that $\det \mathbf{P}$ and $\text{trace}\,\mathbf{P}$ are simply $\prod d_i$ and $\sum d_i$ respectively, as $\mathbf{U}$ is upper-triangular with 1's down the diagonal. Hence 'fiddles' such as constant-trace algorithms are easily achieved by manipulating elements of $\mathbf{D}$ only.

As is usual in RLS, as N increases all elements of $\mathbf{D}$ shrink to 0 and the updating loses its 'alertness'. Commonly forgetting (fiddle) factors are used, but in this approach it is better to revert to 'asymptotic data windows'. Suppose M is the memory length, then the equivalent forgetting factor is $\lambda = (M-1)/M$. Note that the use of a common forgetting factor, which affects all elements of $\mathbf{D}$ in the same way, does *not* affect the conditioning. However, for large t under stationary excitation, it can be shown that:

1. $d_1 \rightarrow 1/[M\mathcal{E}y^2]$

2. $d_{n+1} \rightarrow 1/[M\mathcal{E}\epsilon^2]$.

Hence for a given output variance, an *imposed* lower bound of D_1 gives an upper bound for M, thus retaining 'alertness'; if the variance of y reduces (possibly due to a lower excitation), the effective value of M increases - a useful property. It is common with self-tuning to use *variable* forgetting factors (e.g. Fortescue/Ydstie) to achieve:

1. ϵ small $\rightarrow$ slow or no update;

2. ϵ large $\rightarrow$ fast update.

The development of this is facilitated by the fact that the vector $\mathbf{f} = \mathbf{U}'\mathbf{x}$ of the UDU update has elements which are precisely the innovation ϵ for the corresponding (reduced-order) model. Hence the product $\gamma = f_{n+1}^2 d_{n+1}$ equals (approximately) $\epsilon(t)^2/M\epsilon_0^2$ - just what is required by the Fortescue method. In our case a small value of γ implies slow forgetting and *vice versa*.

Finally we might wish to impose *restricted* exponential forgetting, in which we forget old information in $\mathbf{P}$ only in the direction of the new data $\mathbf{x}$ rather than in other directions for which there is no new information. Hence we replace the usual update

$$\mathbf{S}(t) = \lambda\mathbf{S}(t-1) + \mathbf{x}\mathbf{x}'$$

$$\text{by} \quad \mathbf{S}(t) = \mathbf{S}(t-1) + w(t)\mathbf{x}\mathbf{x}',$$

where w is chosen so that both give exactly the same result in the $\mathbf{x}$ direction. Hence postmultiplying both equations by $\mathbf{x}$ and equating, we get:

$$\begin{aligned} \mathbf{S}(t)\mathbf{x} &= \lambda\mathbf{S}(t-1)\mathbf{x} + \mathbf{x}\mathbf{x}'\mathbf{x} \\ &= \mathbf{S}(t-1)\mathbf{x} + w(t)\mathbf{x}\mathbf{x}'\mathbf{x}. \end{aligned}$$

This gives after some manipulation:

$$w(t) = 1 - \frac{1-\lambda}{\mathbf{x}'\mathbf{P}(t-1)\mathbf{x}} = 1 - \frac{1-\lambda}{\mathbf{f}'\mathbf{g}}.$$

Let us now incorporate the Fortescue VFF, as from above:

$$1 - \lambda = \frac{1}{M} = \gamma\frac{\epsilon(t)^2}{\epsilon_0^2}.$$

$$\text{Hence:} \quad w(t) = 1 - \frac{\gamma}{\mathbf{f}'\mathbf{g}},$$

a readily computed quantity. Code which includes all these factors is then:

```
function [un,dn,f,fact] = modudu (x,uold,dold)
%
%     udu estimation: includes--
%        restricted directional forgetting
%        Ydstie-like acceleration
%
%     x:     data vector (column) time t
%     dold:  D at t-1, stored as a vector
```

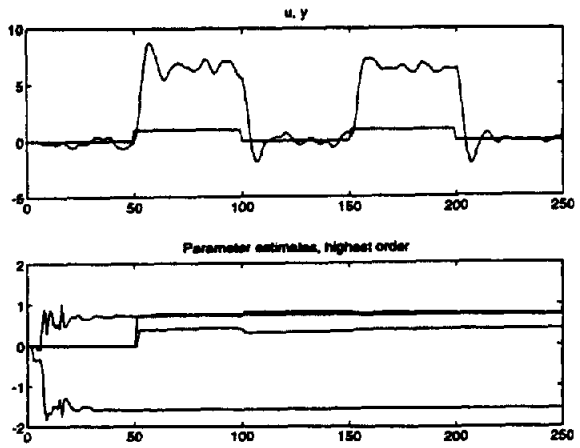

Figure 6: UDU estimation: noisy data

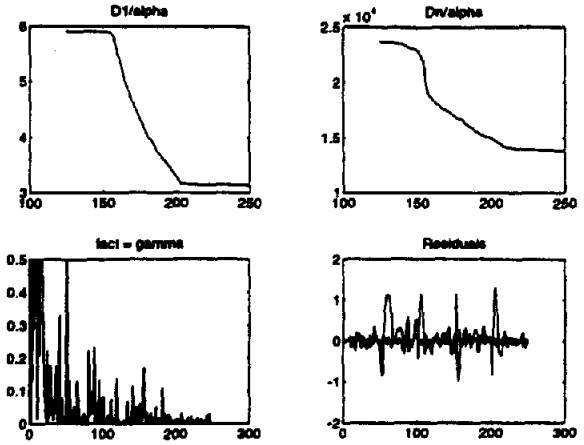

Figure 7: Trajectories of the key variables

```
f = uold'*x; g = dold.*f; n = length(x);
bo = 1; dn = dold; mu = f; un = eye(n);
fact = (f(n)^2)*dold(n);
if fact > 0.5, fact = 0.5, end;
ff = f'*g; if ff > eps,
           g = (1-fact/ff)*g; end;  v = g;

for j = 1:n,
       bn = bo + f(j)*g(j);
       dn(j) = dold(j)*bo/bn;
       mu = -f(j) / bo; bo = bn;
       if j > 1,
         for i = 1:j-1,
             a = uold(i,j);
             un(i,j) = a + v(i)*mu;
             v(i) = v(i) + a*v(j);
         end;
       end;
end;
return;
end;
```

The same system as the previous example was simulated, except now there is noise added. An imposed lower bound for $d_1 = \alpha$ was chosen to be 10^{-4} (so the memory length M is bounded above by roughly $10^4/\mathcal{E}y^2$). The 'accuracy' parameter d_1/d_{n+1} was also set to 10^{-4}; this also bounds the conditioning of $\mathbf{D}$. Fig.6 shows the data and parameter trajectories for the 2^{nd} order model estimate and Fig.7 the variations of the variable γ, the first and last $\mathbf{D}$ entries, and the residuals of the 1^{st} and 2^{nd} order models. Note how the 'acceleration factor' γ and the d entries settle as the model estimates converge.

9 CONCLUSIONS

Predictive control has been shown to be effective in applications, despite the original lack of a sound theoretical foundation. New approaches to the guarantee of stability based on monotonically decreasing cost functions show that some form of terminal constraints or weighting is useful. For a self-tuned predictive controller, the novel UDU estimator of Niu *et al* appears to be highly effective as it provides models of all orders up to a user-chosen maximum, and its structure can be easily exploited to give a wide variety of forgetting factors.

10 ACKNOWLEDGEMENTS

Research into MBPC at Oxford is supported by the UK Engineering and Physical Sciences Research Council and by the CEC.

11 REFERENCES

Bitmead,R.R., M. Gevers and V. Wertz (1990). *Adaptive optimal control: the thinking man's GPC.* Prentice-Hall. Englewood Cliffs, NJ

Clarke,D.W. and P.J. Gawthrop (1975). Self-tuning control. *Proc.IEE*, **122**, pp.929-934

Clarke,D.W., C. Mohtadi and P.S. Tuffs (1987). Generalized predictive control. Part 1: The basic algorithm. *Automatica*, **23**, No.2, pp.137-148, and Generalized predictive control. Part 2: Extensions and interpretations. *Automatica*, **23**, No.2, pp.149-160

Clarke,D.W. and C. Mohtadi (1989). Properties of Generalized Predictive Control. *Automatica*, **25**, No.6, pp.859-875

Clarke,D.W. and R. Scattolini (1991). Constrained receding-horizon predictive control. Proc.IEE, **138**, Pt.D, No.4, pp.347-354

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

De Keyser,R.M.C. and A.R. Van Cauwenberghe (1985). Towards robust adaptive control with extended predictive control. CDC, Athens

De Keyser,R.M.C., P.G.A. Van de Velde and F.A.G. Dumortier (1988). A comparative study of self-adapting long-range predictive control methods. *Automatica*, **24**, No.2, pp.149-163

Demircioglu,H. and D.W. Clarke (1992). CGPC with guaranteed stability properties. *Proc.IEE*, **139**, Pt.D, No.4, pp.371-380

Demircioglu,H. and D.W. Clarke (1993). Generalised predictive control with end-point state weighting. *Proc.IEE*, **140**, Pt.D, No.4, pp.275-282

Fletcher, R. (1990). *Practical methods of optimization.* John Wiley and Sons, NY

Harris,C.J. and S.A. Billings (eds.) (1981). *Self-tuning and Adaptive Control.* Peter Perigrinus, Stevenage, UK

Kleinman,D.L. (1970). An easy way to stabilise a linear constant system. *IEEE Trans.AC*, **15**, No.6, p.692

Kouvaritakis,B., J.A. Rossiter and A.O.T. Chang (1992). Stable generalised predictive control: an algorithm with guaranteed stability. *Proc.IEE*, **139**, Pt.D, No.4, pp.349-362

Kwon,W.H. and A.E. Pearson (1975). On the stabilization of a discrete constant linear system. *IEEE Trans.AC*, **20**, No.6. pp.800-801

Mayne,D.Q. and H. Michalska (1990). Receding horizon control of nonlinear systems. *IEEE Trans.AC*, **35**, No.7, pp.814-824

Meadows,E.S. and J.B. Rawlings (1993). Receding horizon control with an infinite horizon. ACC, San Francisco

Mosca,E., J.M. Lemos and J. Zhang (1990). Stabilising I/O receding-horizon control. Proc. 29th CDC, Honolulu.

Mosca,E. (1995). *Optimal Predictive and Adaptive Control.* Prentice-Hall, NJ.

Niu,S., D.G. Fisher and D. Xiao (1992). An Augmented UD identification algorithm. *Int.J.Control*, **56**, 193-211.

Niu,S. and D.G. Fisher (1993). Information forgetting using the augmented UD identification algorithm. *ACC*, San Francisco.

Prett,D.M. and C.E. Garcia (1988). *Fundamental process control.* Butterworths, U.K.

Richalet,J., A. Rault, J.L. Testud and J. Papon (1978). Model predictive heuristic control: applications to industrial processes. *Automatica*, **14**, No.5, pp.413-428

Robinson,B.D. and D.W. Clarke (1991). Robustness effects of a prefilter in generalised predictive control. *Proc.IEE*, **138**, Pt.D, No.1 pp.2-8

Rossiter,J.A. and B. Kouvaritakis (1993). Constrained stable generalised predictive control. *Proc.IEE*, **140**, Pt.D, No.4, pp.243-254

Soeterboek,R. (1992). *Predictive Control. A Unified Approach.* Prentice-Hall, NJ.

Shook,D.S., C. Mohtadi and S.L. Shah (1991). Identification for long-range predictive control. *Proc.IEE*, **138**, Pt.D, No.1, pp.75-84

Shook,D.S., C. Mohtadi and S.L. Shah (1992). A control-relevent identification strategy for GPC. *IEEE Trans.AC*, **37**, No.7, pp.975-980

Thornton,C.L. and G.J. Bierman (1978). Filtering and error analysis via the UDU covariance factorization. *IEEE Trans AC*, **AC-23**, 5, 901-907.

Ydstie,B.E. (1984). Extended horizon adaptive control. IFAC World Congress, Budapest

Yoon,T-W. and D.W. Clarke (1994) Adaptive predictive control of the benchmark plant. *Automatica*, **30**, No.4, 621-628

Yoon,T-W. and D.W. Clarke (1995) Observer design in receding-horizon predictive control. *Int.J.Control*, **61**, pp.171-191