A state space model based multistep adaptive predictive controller (MAPC) with disturbance modeling and Kalman filter prediction

N Rao Sripada* & D Grant Fisherb

*Department of Chemical Engineering, Indian Institute of Technology, Madras 600 036, India
bDepartment of Chemical Engineering, University of Alberta, Edmonton, Canada T6G 2G6

Received 21 November 1997; accepted 10 June 1999

A multistep adaptive predictive control strategy based on a state space model of the process has been developed. It can be compared with the Generalized Predictive Control algorithm. The emphasis in the development of the proposed control scheme is on modeling and elimination of disturbances. In the proposed scheme any prior information regarding the disturbances can be incorporated (by specifying certain polynomials and/or the noise covariances). If no prior information is available then the unknown unmodeled effects (such as noise, unmeasured load-disturbances and model process mismatch) can be represented by a residual model which can best be identified in a two-stage setting. This approach leads to satisfactory modeling of disturbances and good regulation via predictive control. Some important features of the proposed algorithm are: (i) it uses a state space model which allows separate modeling of u-to-y process dynamics, process and measurement noise; this is not possible in an ARMAX-type input/output model where process and measurement noise appear lumped in the noise polynomial; (ii) it uses a Kalman Filter (KF) to generate the predictions of the output; the KF can be easily tuned via noise covariances and is a simpler and better alternative to specifying or estimating a noise polynomial; (iii) there is no need to solve a Diophantine identity on-line; the result is reduced computation; and (iv) if residual modeling is used it leads to simpler and improved way of handling disturbances. The proposed control algorithm is presented for the single-input, single-output case. Applying the algorithm to multivariable processes is straightforward. Simulation examples are included to illustrate the advantages and performance of the proposed control scheme.

A self-tuning adaptive controller, broadly speaking, is a combination of a parameter estimation scheme and a suitable control strategy. The progress self-tuning control has made over the years has been summarized in a number of excellent review articles and text books. Self-tuning control has evolved from single-step predictive methods such as self-tuning regulator to multistep predictive control methods such as GPC. A large impetus for developing the latter class of algorithms has come from such industrially successful but non-adaptive, model-based multistep predictive control techniques such as DMC, IDCOM and MOCCA. However a single self-tuning control algorithm that can be applied to a major class of practical processes has not been developed yet, although algorithms such as GPC perform well in idealized situations.

The objective of the work reported here is to develop a comprehensive self-tuning adaptive controller that can be applied to practical industrial processes. The result is the multistep adaptive predictive controller (MAPC) reported in this paper. The major issues to be contended with in the application of adaptive control to industrial processes are handling of: (i) process time delays, (ii) noise, (iii) load-disturbances, (iv) parameter estimation, and (v) achieving user specified control performance. Some of these issues are briefly discussed before presenting an overview of the MAPC reported in this paper.

Process time delays

Time delays are common in process systems and are associated with flows and vary significantly. Therefore they are difficult to specify in advance and cause control performance to deteriorate. One approach to control processes with time delays is to use predictive control. However, single-step predictive control methods are very sensitive to errors in the a priori estimate of the time delay and hence control detuning is often required. An alternative is to
use predictive control based on the minimization of multi-stage cost function.

**Process disturbances**

Regulation (or disturbance rejection) is more important in industry than setpoint tracking. Offset-free rejection of load-disturbances with good dynamic behaviour requires the use of a proper load-disturbance internal model (cf. internal model principle). Often the load disturbances are assumed to change as random steps at random times and although this gives good results, it is not enough to handle more complicated disturbances such as random exponential rises or load-disturbances that affect the process via a load-disturbance transfer function. The minimum variance strategy handles noise optimally if correct noise structure is employed in prediction. However, the use of a fixed noise model is often unrealistic as the spectral characteristics of the noise vary significantly with time since the noise results from several disturbance sources. On-line identification of the noise model is usually done in self-tuning control but is associated with convergence difficulties and can give poor results. As an alternative a state space model and a Kalman Filter can be used to generate minimum variance estimates of the process output and has the advantage that it is easy to tune on-line by changing the noise covariances. How the disturbances are modeled can influence both model parameter estimation as well as subsequent controller synthesis.

**Model identification and parameter estimation**

Model identification is an important step in adaptive control and is crucial to its success. Recursive prediction error methods are often used in on-line parameter estimation and one such algorithm which is widely used is the recursive least squares (RLS) or its variant, the recursive extended least squares (RELS). A number of *ad hoc* as well as theoretical modifications are added to the standard parameter estimation methods to enhance their performance in the self-tuning context. Shah and Cluett\(^{20}\) review a number of modified least squares estimation schemes. The identification scheme requires: (i) selection of an appropriate model structure, and (ii) estimation of selected model parameters by a suitable (ie. robust) estimation method.

**Process constraints**

Constraints are important in process control and when present they may cause problems in the identification part of an adaptive controller. The proposed control algorithm can easily incorporate constraints, but the control calculations then are iterative. In the proposed scheme constraints are handled by the supervisory system, which adjusts the reference input to satisfy constraints. The supervisory system can also be used to generate other types of reference trajectories (eg. a filtered setpoint).

Using the supervisory system to implement the constraints has the advantage of dividing the control problem into two consecutive tasks: first the calculations on the reference trajectory; and second the calculation of the control action so that the output tracks the computed reference trajectory. One advantage of this separation is that the supervisory control calculations can be done at a different (longer) sampling interval and can be stopped during periods of steady state operation.

In addition to the problems of time delays and disturbances, a practical adaptive control scheme must be able to: (i) overcome the problems caused by Non Minimum Phase (NMP) or unstable zeros, which arise in discrete time control due to fast sampling or which may be present inherently in the process; and (ii) handle unstable and poorly damped processes; although most processes are characterized by stable, well-damped, dead-time dominated dynamics, unstable and/or complex poles are not uncommon (eg. reactor control) and can present significant control difficulties.

In the development of the proposed control algorithm all the above issues have been addressed. Consideration of these issues have led to the adoption of following ideas in the present work: (i) use of multistep or longrange predictive control; (ii) state space model-based representation of process dynamics and other unmodeled effects (such as noise, load-disturbances and model process mismatch; (iii) use of robust estimation; (iv) use of prior information regarding for example, noise and load disturbances affecting the process; and (v) in the absence of any prior information use of residual modeling and two-stage identification. Although some of these ideas have been in use for some time, the major contribution of the present work is better disturbance modeling and treatment and integration of the various ideas into one single algorithm, so that it can be
applied to a majority of industrial processes. A number of researchers have considered state space modeling in model-based predictive control area but in self-tuning control state space modeling has not been used much (except, for example, in LQG self-tuners), especially in the predictive control context.

Although the proposed control algorithm is inherently a multivariable control algorithm, it is presented for the single-input, single-output (SISO) case for ease of understanding and presentation. The design of MAPC involves the following steps: (i) process representation (or model selection for estimation and controller synthesis), (ii) model identification and parameter estimation, (iii) prediction of the process output, (iv) control calculations, (v) the supervisory system calculations, and (vi) the executive subsystem. A high-level schematic block diagram of MAPC is shown in Fig. 1 and the structure of the rest of paper closely follows this figure. The next six sections of the paper discuss the six main steps of the proposed scheme. This is followed by a discussion of the convergence and stability issues of MAPC and finally some SISO simulated examples to demonstrate some of its features.

**Process representation**

The process is assumed to be adequately represented by the locally linear model:

\[
A(z^{-1})y(t) = B(z^{-1})u(t - k) + r(t) \quad \ldots \quad (1)
\]

where \( A(z^{-1}) = 1 + a_1 z^{-1} + \ldots + a_n z^{-n} \) and \( B(z^{-1}) = b_0 z^{-d} + \ldots + b_m z^{-m} \). It may be noted that \( A \) and \( B \) can always be written this way by setting appropriate coefficients to zero if necessary. \( k \) is process pure time delay in number of sampling intervals, \( r(t) \) is a residual that includes all the unmodeled effects such as process and measurement noise, unmeasured load-disturbances and model process mismatch (MPM).

Depending on the assumptions made about the disturbances affecting the process, \( r(t) \) can assume several forms (e.g. MA, IMA, MA or IMA with a bias etc.). All these models to represent disturbances have advantages and disadvantages. In the approach followed in this paper it is decided to incorporate all available prior information regarding disturbances. The emphasis is on a formulation which requires minimum or no prior information regarding the disturbances. Following M'Saad et al.\textsuperscript{16} it is decided to model \( r(t) \) by:

\[
F(z^{-1})C(z^{-1})D(z^{-1}) \quad \ldots \quad (2)
\]

\( F, C \) and \( D \) are polynomials in \( z^{-1} \) (note that the argument \( z^{-1} \) is omitted for simplicity). \( F \) is a load-disturbance internal model, possibly with-zeros on the boundary of the unit circle, i.e. \( F\equiv0 \) when \( z^{-1}=1 \). It can model anything from a simple dc bias, random steps at random times or more general disturbances including periodic disturbances. \( C \) and \( D \) are stable.
polynomials used to model coloured process noise. It may also be noted that if \( F \) is set to \( \Delta \), then \( D \) is used to model load-disturbances that have dynamics. \( \{e(t)\} \)
 is a zero mean, random, white noise sequence (when \( C=D=1 \) and \( F=\Delta=1-\varepsilon^{-1} \), \( r(t) \) models Brownian motion) or a random sequence following Poisson distribution which means it is non-zero only at isolated instants of time (used to model, for example, random steps at random times). Note that \( r(t) \) is purely a deterministic process when \( Fr(t)=0 \) and purely a stochastic process when \( F=1 \).

This type of model is better suited for industrial process control applications than a simple ARIMA model used, for example, in GPC. Note that the above model is a generalization of the ARIMA model. Although one can model an arbitrary disturbance using the ARIMA model, the \( C \) polynomial may have to be of very high order (or equivalently \( A \) and \( B \) have to be over-parameterized to absorb \( D \)) because of the lack of denominator polynomial \( D \). In what follows, \( F \) is taken to be \( \Delta \), for simplicity.

One must know or estimate \( F \), \( C \) and \( D \) polynomials for use in the model and hence the control scheme. One of the contributions of the paper is to provide a method (via a state space model, noise covariances and KF) to avoid estimation and/or to provide a way to obtain the required information efficiently (e.g. residual modeling and two-stage identification). If one has prior information regarding the \( C \) and \( D \) polynomials this information can be directly used in the model. Alternatively most of the times at least the noise covariances can be specified. Then a stochastic state space model and a KF can be used for prediction. If no prior information is available at all, one can absorb \( C \) and \( D \) in \( F \) and estimate an AR model based on \( r(t) \). This is best done in a two-stage setting. \( r(t) \) can be viewed as a measured signal and the identification of the residual model is likely to be more successful than estimating the noise polynomial in a single stage using Pseudo Linear Regression methods (e.g. RELS) where estimates of the noise are used. The advantages and performance of residual modeling and two-stage identification have been reported by Sripada and Fisher.\(^{25} \) In this paper, the following three cases are considered.

Case 1

Prior information in the form of \( C, D \) (and \( F=\Delta \)) is available. One can then absorb \( F \) and \( D \) in \( A \) and \( B \) and write down the following state space model in observable canonical form (\( A_m=AFD \) and \( B_m=BFD \) where both \( A_m \) and \( B_m \) are adjusted to be of degree \( n \)):

\[
\begin{align*}
\hat{x}(t+1) &= \Phi \hat{x}(t)+\Delta \Delta u(t)+\Gamma e(t) \\
\hat{y}(t) &= H \hat{x}(t)+r(t)
\end{align*}
\]  

where

\[
\Phi = \begin{bmatrix} 0 & 0 & \cdots & 0 \\
1 & -a_{mn} & \cdots & -a_{m1} \\
0 & 0 & \cdots & -a_{mn} \\
\vdots & \vdots & \ddots & \vdots \\
0 & \cdots & \cdots & 0
\end{bmatrix}, \quad A = \begin{bmatrix} h_{mn} \\
h_{m-1} \\
\vdots \\
h_{n+1} \\
0
\end{bmatrix}
\]

\[
\Gamma = \begin{bmatrix} c_1 \\
c_2 \\
\vdots \\
c_m \\
0
\end{bmatrix}
\]

\[
H = [00 \cdots 01]
\]

The dimensions of \( \Phi, A, \Gamma, H \) are respectively \((n+k)\times(n+k), (n+k)\times1, (n+k)\times1 \) and \(1\times(n+k)\). This state space model is already in innovations form and it is straightforward to see that the KF is given by,

\[
\begin{align*}
\hat{x}(t|t-1) &= \Phi \hat{x}(t|t-1)+\Delta \Delta u(t)+\Gamma [y(t)-H x(t|t-1)] \\
\hat{y}(t) &= H x(t|t-1)+e(t)
\end{align*}
\]  

Case 2

No prior information regarding \( C \) is available. \( D \) may or may not be known. If \( D \) is known then it is absorbed in \( A \) and \( B \) as shown in case 1. If \( D \) is unknown then either it may simply be set to 1 or estimated along with \( A \) and \( B \) by over-parameterizing them. One can then use the following stochastic state space model (\( C \) is modeled by separate process and measurement noises),

\[
\begin{align*}
\hat{x}(t+1) &= \Phi \hat{x}(t)+\Delta \Delta u(t)+\Gamma w(t) \\
\hat{y}(t) &= H x(t|t-1)+e(t)
\end{align*}
\]
\( y(t) = Hx(t) + y(t) \)  

The state space model matrices (Φ and \( A \)) are defined as earlier based on the knowledge of \( A, B, D \) and \( F \) polynomials. Here, for simplicity, \( \Gamma \) is set to 1 and \( w(t) \) is taken as \([w_1(t) ... w_n(t)]\), i.e., \( w(t) \) is an \( n \)-dimensional noise vector. For this case the full KF equations have to be used as discussed in the following sections.

**Case 3**

Absolutely no prior information is available. Then one can absorb \( C \) and \( D \) in \( F \) and write \( \Delta F_r(t) = e(t) \). Then the input/output model becomes,

\[ AF_r \Delta y(t) = BF_r \Delta u(t-k) + e(t) \]

or \( \Delta w_0(t) = B_0 \Delta u(t-k) + e(t) \) and once again one gets a state space model in innovations form as in case 1. The state space model matrices must be written properly as before from the knowledge of the polynomials in the input/output model.

**Remarks:**

(i) If there is an auxiliary measured disturbance input, it can easily be included in the above models.

(ii) As discussed by Sripada and Fisher, this kind of modeling disturbances is very attractive when there is no prior information regarding the disturbances. \( A \) and \( B \) are identified in a first stage with proper precautions to obtain good estimates of the u-to-y model. The residual model, \( F_r \), is best identified in a second stage. Residual modeling and two-stage identification has been found to give improved results compared with estimating a noise polynomial in a single stage.

In what follows the proposed algorithm is presented for case 2. With slight modification (by noting that the state space model is already in innovations form) it can be adapted to the other two cases.

**One-step-ahead Kalman Filter prediction**

A Kalman Filter can be used to estimate the states of the stochastic state space model given by Eqs (7) and (8). The one-step-ahead prediction of the state has also been given.  

\[ x(t+1|t) = \Phi x(t|t-1) + A \Delta u(t) + K(t) r(t) \]

\[ y(t) = Hx(t|t-1) + e(t) \]

which is optimal in the sense of minimum variance estimation when the Kalman gain, \( K(t) \), is obtained from,

\[ K(t) = \Phi P(t|t-1) H (H P(t|t-1) H^T + R) \]

where the state covariance matrix, \( P(t|t-1) \), is updated according to,

\[ P(t+1|t) = (I - K(t) H) P(t|t-1) (I - K(t) H)^T + K(t) R K(t)^T \]

Here \( P(0|1) \) is the initial covariance matrix and is chosen such that \( P(0|1) = \Sigma \). The sequence \( \{ e(t) := y(t) - H(x(t|t-1)) \} \) is the innovations sequence.

**Remark:** If the matrices, \( \Phi, A, \Gamma \) and \( H \) are constant then the steady state Kalman Filter can be used, which gives the same asymptotic error covariance of the estimated state vector as the time varying, optimal KF. The steady state Kalman gain, \( K \), is calculated using

\[ K = \Phi P \Phi^T + H (H P H + R) \]

where \( P \) is the steady state value of the covariance matrix and is computed by solving the algebraic Riccati equation:

\[ P = \Phi P \Phi^T + H (H P H + R) \]

It may be noted that \( K \) and \( P \) can be computed a priori and thereby reduce computation.

**Model identification and parameter estimation**

The overall process model is composed of a model of the u-to-y process dynamics and a model of the disturbances. For adaptive control, the parameters of these models are estimated on-line from measured input/output data, \( \{ y(t) \} \) and \( \{ u(t) \} \). Such a model identification step forms an important part of an adaptive control scheme and in the proposed scheme is shown as the model identification block in Fig. 1. The improved least squares (ILS) algorithm is used as opposed to standard RLS in this study.

When the stochastic state space model given in case 2 is used (where the \( C \) polynomial is modeled via the process and measurement noise terms \( w(t) \) and \( v(t) \) respectively) the u-to-y dynamic model given by \( A \) and \( B \) polynomials has to be estimated. If \( D \) is known then the data are filtered by \( D \) before passing to the ILS algorithm. If \( D \) is not known, then either \( D \) can be assumed to be 1 or can be estimated simultaneously with \( A \) and \( B \) by over-parameterizing these polynomials.
Output prediction

The MAPC uses an output prediction block as shown in Fig. 1. It is possible to obtain multistep-ahead prediction of the state, $x(t+il)$ and hence $y(t+il)$ for $i \geq 1$ using the KF from measurements available up to time $t$. The KF uses the one-step-ahead estimate of the state, $x(t+1/t)$ (discussed earlier), to give minimum variance estimates of the future states and is given by:

$$x(t+il) = \Phi(t+1/t) x(t) + \Phi(t+1/t) \Lambda \Delta u(t+1)$$

... (15)

and

$$y(t+il) = Hx(t+il)$$

... (16)

for $i \geq 1$. With a little algebra one can write down the following prediction equation for time $t+N1$ to $t+N2$ in vector matrix notation

$$y = Gu + J$$

where $y=[y(t+N1/t), ..., y(t+N2/t)]', u=[\Delta u(t), ..., \Delta u(t+Nu-1)]'$ and $f=[f(t+N1), ..., f(t+N2)]'$ where $f(t+i)$ is the free response of the system at $t+i$, based on past inputs and is given by

$$f(t+i) = H \Phi(t+1/t) K(t) + L \Delta y(t)$$

... (18)

is due to the control increment to be implemented. $G$ is a lower triangular matrix with $g_{ij} = H \Phi(t+1/t)$ for $j \leq i$ and $g_{ij} = 0$ otherwise. $G$ is given by

$$G =
\begin{bmatrix}
  g_{N1} & 0 & 0 & \cdots & 0 \\
  g_{N1+1} & g_{N1} & 0 & \cdots & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  g_{N2} & g_{N2-1} & \cdots & g_{N2-NU} & 0 
\end{bmatrix}$$

... (19)

where $\{g_i\}$ are the step response coefficients of the system (based on $u$-to-$y$ dynamic model).

Remarks: (i) The proposed control scheme uses a control horizon, $Nu$, beyond which the control increments are set to zero, i.e. $\Delta u(t+i+1) = 0$ for $i > Nu$.

(ii) Note that the output prediction does not require solving a single or a bank of Diophantine identities, e.g. as in GPC; instead the prediction requires calculation of the Kalman Filter gain, $K(t)$, and $\Phi(t)$, $i \in [1,N2]$ which are the major computational steps. However, actual computation of $\Phi$ is simplified by noting that it is sparse with a special (canonical) structure. The worst case computation for obtaining the output prediction given by Eq. (17), for $n=3$, $N2=10$ and $Nu=2$, required for the proposed algorithm (without exploiting the special structure of $\Phi$ and excluding KF calculations) are approximately 810 flops where as GPC without using recursion of Diophantine Identity requires about 1345 flops.

The multistep predictive controller

The multistep predictive controller shown in Fig. 1 generates a trajectory of present and future control actions such that the predicted output trajectory produced by the Kalman Filter is as close to the reference trajectory as possible (in the sense that a user specified cost function is minimized). The supervisory system (also shown in Fig. 1) supplies the reference trajectory $y_{ref} = [y_{ref}(t+N1), ..., y_{ref}(t+N2)]'$ at time $t$. The control strategy is based on the minimization of the following multistep, weighted, quadratic cost function which is similar to the one used in GPC, DMC, MOCCA etc.:

$$J(N1,N2,NU,G_p,G_u) = (1/2)[y_{ref}(t) - y(t)]'T_y[y_{ref}(t) - y(t)] + \sum_{i=1}^{Nu} u(i)'T_u u(i)$$

... (18)
Here $N_2 \geq N_1 \geq 1$; $N_y := N_2-N_1+1$ is a finite output horizon; $N_u$ is a finite control horizon with $N_u \leq N_y$; $\Gamma_y$ and $\Gamma_u$ are possibly time varying weighting matrices on the tracking errors and control actions respectively. Substituting for $y$ by Eq. (17) and setting $dJ/du=0$, one obtains the following control-law:

$$u = G^* [y_{ref}]$$

where

$$G^* := (GT_y G + \Gamma_u)^{-1} GT_y$$

(19)

The control-law given by Eq. (19) involves very little computation when the process model parameters are fixed, since $G^*$ can then be computed a priori. In the self-tuning case, since the parameters change online, $G^*$ needs to be recalculated at every instant. It may be noted that the calculation of $G^*$ involves a matrix inversion which is a computationally significant step. The dimension of $G$ is $N_y x N_u$ but the dimension of the matrix to be inverted is only $N_u x N_u$. Because of the moving-window formulation it is generally possible to choose $N_y$ as low as 3 or 5 and $N_u$ as low as 1 or 2. Thus $G$ is usually a small matrix of the order of $5x2$ and the matrix to be inverted is only of the order $2x2$. The control-law is usually implemented in a receding-horizon sense, ie. only the first control increment, $\Delta u(t)$, is implemented and all the calculations are repeated at the next sampling interval. This means that only the product of the first row of $G^*$ and $[y_{ref}]$ needs to be computed.

Remarks: (i) The receding-horizon approach makes the control algorithm time invariant, i.e. the same control-law (cf. Eq. (19)) is used at each instant, unlike in a fixed-horizon LQG policy.

(ii) When the complete reference trajectory is known ahead of time and the process model is relatively time invariant with good parameters it is some times possible to implement the whole control vector, $u$, over the control horizon of $N_u$ and repeat the calculations every $N_u$ intervals. This saves some computation.

Choice of the controller parameters—The parameters $N_1$, $N_2$, $N_u$ and the weighting matrices $\Gamma_y$ and $\Gamma_u$ allow one to shape the dynamic response of the controlled process as well as to stabilize NMP processes or processes with open-loop unstable or poorly damped dynamics. If the output horizon and the control horizon are equal, the control problem is essentially solving a set of linear algebraic equations for the control vector. By choosing the control horizon less than the output horizon in the problem formulation one then has to solve an over-determined set of linear algebraic equations for the control vector. The proposed method solves this problem using the weighted least squares approach. Thus by assuming that the control increments beyond the control horizon to be zero instead of allowing them to be free, the method derives a lot of flexibility as well as reduces computation. In particular this method allows one to stabilize NMP processes. Choosing $\Delta u(t+i-1)=0$ for $i > N_u$ can also be interpreted as using $\infty$-weighting on $\Delta u(t+i-1)$, i.e. $[N_u+1, N_2]$ in the cost function.

Output horizon ($N_1$, $N_2$)—It has already been pointed out that using prediction over a range of points in the controller cost function rather than a single point makes the controller robust to incorrect specification of the process dead-time. If the time delay, $k$, of the process is exactly known then one can take $N_1=(k+1)$, since in that case the first future output that is directly affected by the current control action is at $t+k+1$. Thus using $y(t+i/t)$ to $y(t+k/t)$ in the cost function is redundant and hence by omitting these some computation can be saved. If the time delay is expected to vary or is not known with certainty then one can set $N_1=1$. In that case, even though $G$ is singular a solution can be forced to exist by a proper choice of the weighting matrices (eg. choose $\Gamma_u=1$). This is possible because of the over-determined set of equations for the solution of the control vector. Thus the proposed controller can perform well even if the knowledge of the process

Fig. 4—Performance of GPC in the presence of noise; system response: continuous, setpoint: dashed.
time delay is poor. Single-step cost functions do not offer this flexibility and controllers based on these may fail or perform poorly if the time delay is not exactly known. For NMP processes, N1 can be set beyond the initial wrong-way response in the process step response so that only the points that approach the setpoint in the right direction are included in the controller cost function.

N2 can be chosen so that N2=5 to 10. Alternatively the output horizon, N2, can be chosen equal to the rise time of the process step response and N2 is set appropriately. This is adequate in most cases. If the process is open-loop unstable or has complex poles, then obviously more points must be included in the output horizon. One approach for selection of N2 is through a simulation study using a rough model of the process.

As a default N1 can be set to 1 and N2 can be set to 10. Most processes give stable response and good performance with these settings. If more conservative performance is desired, N2 can be increased to cover the entire open-loop step response.

Control horizon (N0)—Usually there is no advantage in taking N0=0 where n is the order of the process. If there is no weighting then for N0=0 the controller gives output dead-beat control. In most cases N0 can be set to 1 or 2.

Weighting matrices (Γ1, Γ2)—The weighting matrices (Γ1 and Γ2) are especially useful in the multivariable case because they can be used to specify that any particular output (input) is more or less important than the other outputs (inputs) (ie. relative weighting among outputs and/or inputs). The weighting matrices can also be used to penalize the control action or to ensure that soft constraints on the process variables are not violated. In numerical terms, the weighting matrices are very helpful to force a stable solution to the control problem. For example, when G is poorly conditioned or almost singular then in the absence of any weighting Eq. (19) can give poor solutions. This may result in excessive oscillations or ringing in the control input or even closed-loop instability. By choosing the weighting matrices appropriately the matrix inversion in Eq. (20) can be done more robustly (ie. the condition number of [GT,G+Γ1] can be significantly reduced). This can be done from a knowledge of the process time scale (ie. sampling interval), a rough step response of the process and the desired closed-loop performance. In the SISO case it is simpler to choose Γ1=I and Γ2=λI, λ>0, so that

\[ G'=[GG+2λI]^{-1}G \]

in Eq. (20). There is only one tuning parameter, λ, to be adjusted on-line. If λ=0 then there is no weighting on the control action. If λ is large then the control action is heavily penalized.

The supervisory system

The MAPC assumes that the reference trajectory, \( y_{ref}=[y_{ref}(t)+N1], \ldots, y_{ref}(t+N2)] \) is available at time t. The supervisory system which is shown in Fig. 1 generates these values. The reference trajectory may be constant and equal to a setpoint over the output horizon or may vary. For example, the supervisory system can be used to generate a smoothed setpoint trajectory as in IDCOM where it is considered that a smoothed approach to the current setpoint, \( y_{sp} \) is required from the current output. In this case the supervisory system can calculate the reference trajectory from the following first order lag model:

\[ y_{ref}(t)+\alpha y_{ref}(t-1)+(1-\alpha)y_{op}(t), t=1, 2, \ldots, N2 \ldots (22) \]

where \( \alpha\in[0,1] \). MAPC is thus capable of both time varying and time invariant reference inputs. Another case is that the reference trajectory may be specified \( a \text{ priori} \) by the process operator. This ability to follow prespecified reference trajectories is useful in practical applications and there are a number of real examples where such an ability is needed. Some examples are robot motion and batch reactor control.
Fig. 6—Performance of non-adaptive MAPC in the presence of load-disturbances (No prior information).

where the temperature of the batch may have to follow a specified trajectory.

The reference inputs can also be adjusted on-line to satisfy process constraints. Constraints on process variables (e.g., $u$ and $y$) are very important in practical applications and are usually ignored in the adaptive control literature. Amplitude constraints on the control signal, $u$, are especially important. Saturation of the control signal may appear in an adaptive scheme, e.g., (i) during start-up or after parameters have converged; (ii) due to bad initialization; and (iii) due to unattainable desired performance (i.e., $y_{ref}$).

When saturation of the input signal occurs the process ceases to be linear and its effect is particularly deleterious on the parameter estimation, as the estimator then tends to identify a zero-gain process. The process can be forced to remain in the linear region by imposing smoother reference changes or calculating the reference signal such that the limits on $u$ are never violated. The supervisory system can be used to achieve this task. The discussion of the procedure to do this is beyond the scope of the present paper.

It may be noted that the constraints can be handled by the predictive controller directly. The advantage of using the supervisory system is that it admits more flexibility.

**The AI executive**

One of the requirements of a practical adaptive controller is that it should be easy to commission and operate. It is common in adaptive control practice to use heuristic fixes or safety jackets (e.g., in parameter estimation) so that the adaptive controller will work smoothly over a broader range of conditions. To do this it is usually suggested\(^\text{11,13}\) that another block be added to direct the overall operation of the adaptive controller as shown in Fig. 1. The functions of such a block can be manifold, e.g., (i) to start-up or commission the adaptive controller; (ii) to perform mode-switching; and (iii) to supervise model identification etc.

The knowledge based systems (AI) provide a suitable framework to implement these functions. The development of the AI executive is not the objective of the work reported here. The AI executive is only highlighted here for the sake of completeness.

**Convergence and stability properties of MAPC**

The stability and performance of MAPC depend in a complicated way on the convergence of the Kalman Filter and the stability of the predictive controller plus the parameter estimates. If the KF converges and gives asymptotically offset-free predictions of the output then the predictive controller ensures that there is no control offset. The dynamic performance of the controller depends on its parameters (e.g., $N_1, N_2, N_u$ and the weighting matrices). The dynamic behaviour of the KF depends on the initial conditions (e.g., $P(0|t-1)$) and the noise covariances. No concrete results regarding the stability of the composite system are available at this stage. However some remarks that are helpful in understanding the overall stability may be made as follows. The convergence properties of the KF are discussed by Goodwin and Sin\(^\text{10}\) and Kumar and Varaiya\(^\text{12}\). The needed result is summarized in the following theorem.

**Theorem**\(^\text{12}\)—Let $Q$ be a square-root of $R_u$ (i.e., $QQ^T=R_u$). Suppose $(\Phi, Q)$ is reachable, $(\Phi, H)$ is observable and $R_v>0$. Then $\lim_{t \to \infty} P(t+1|t) = P$ and $P$ is the unique, non-negative definite solution of the steady state or algebraic Riccati equation:

$$ P = \Phi P \Phi^T (H \Phi P \Phi^T + R_v)^{-1} H \Phi^T P + R_u. $$

In particular $P$ is independent of the initial covariance matrix $P(0|t-1)$.

Thus if the conditions of the theorem hold, which is not hard to verify, then the KF converges to a time invariant filter with steady state covariance given by $P$ and the steady state gain $K$ given by Eq. (13).

In the time varying or self-tuning case it is obvious that in order to give proper estimates of the states (or predictions of the output), the estimates of the process parameters should converge to those of the actual process in some sense (e.g. frequency domain terms). In the self-tuning case the KF gain matrix is computed on the basis of latest parameter estimates and the covariance $P(.)$ available from the previous step. If the parameter estimates converge, the KF gain converges to some steady state value as $t \to \infty$. 
Since the adaptive controller is explicit at least a local stability result can be roughly stated as follows: If the KF converges and the basic controller is stable with initial estimates of the process parameters, $\theta_0$, and the converged (or steady state) KF, then it is reasonable to expect that the time varying adaptive controller would be stable provided that $\theta(t)$ lies in a stable region around $\theta_0$. The stability of the controller depends only on the process u-to-y model, ie., $\theta(t)$. The parameter projection feature of ILS can be used to ensure that $\theta(t)$ stays in a stable region around $\theta_0$ or some other nominal model for which the scheme is found stable, for all time. For simple processes (eg., first or second order) it is relatively easy to define such regions.

Single-input, single-output simulation examples

Simulation examples are discussed in this section which illustrate selected features of the proposed control scheme. Where appropriate, comparisons with GPC are included. The simulations were carried out in MATLAB.

Setpoint tracking—The setpoint tracking ability of the proposed controller depends on the u-to-y model of the process. A second order underdamped (oscillatory) process, given by $Ay(t)=Bu(t)$ where $A=I-1.5z^{-1}+0.5z^{-2}$ and $B=0.75z^{-1}$, was controlled with $N_1=1$, $N_2=5$, $Nu=1$ and $\lambda=0$. 2A and 2B parameters were estimated with $\theta(0)=[0,0,1,0]$ and $P(0)=I$. Fig. 2 shows the performance (y vs. t) of the resulting adaptive controller. The response has settled down to normal one after a period of initial adaptation where the control was bumpy. Note how easy it is to obtain excellent response with almost default settings of the controller parameters.

Handling noise—Process measurements are typically noisy. If the measurement noise is not taken into account in any practical control scheme the control signal may exhibit excessive input/output variation which is not desirable. The KF used in MAPC optimally rejects noise and passes the minimum variance estimates of the output to the controller. The process used in this example is given by $Ay(t)=Bu(t)+Ce(t)/\Delta$ where $A$ and $B$ are as in above and $C=I-1.5z^{-1}+0.5z^{-2}$. $\{e(t)\}$ is a zero-mean white noise sequence with variance 0.01. Since only noise is present and there are no load-disturbances $F$ and $D$ are set to 0 and 1 respectively. In MAPC the $C$ polynomial is modeled by process and measurement noise in the state space model. The process was controlled with $N_1=1$, $N_2=5$, $Nu=1$ and $\lambda=0$ and with $R_e=0.01$ and $R_u=0.01$. Fig. 3 shows the resulting performance (y vs. t and u vs. t). This can be compared with the performance obtained using GPC with $C$ estimated. Same controller parameters were used. The results are shown in Fig. 4. A comparison (of Figs 3 and 4) shows that MAPC performs better than GPC (both output y and input u are better in case of MAPC). The sum of absolute errors (SAE) for output has been calculated for both cases from 50th sampling instant to 200th sampling instant. The SAE for MAPC is approximately 29.4 and that for GPC is 40.15 which confirms the visual comparison. The average absolute error over this period exhibited by MAPC is 0.146 and by GPC is 0.20. It may be noted that in both the cases the u-to-y model used corresponds to the actual process. With almost no effort and without having to use too much prior information (only noise covariances were used) improved performance was obtained. The performance of GPC is the best that can be obtained under the assumption of no prior information where as the performance of MAPC can be further improved by tuning the noise covariances.

The adaptive case (MAPC) is illustrated by Fig. 5. 2A and 2B parameters were estimated with $\theta(0)=[0,0,1,0]$ and $P(0)=I$. It may also be noted that the noise polynomial $C$ need not be estimated as well as no prior knowledge regarding it is required. The controller parameters were the same as before.

Handling load-disturbances—If an exact internal model of the load-disturbances is used, then exact cancellation of the disturbance occurs (depending on the controller parameters). GPC models disturbances as random steps at random times and hence if the disturbances have structure (ie. vary more slowly than can be modeled by a sequence of steps) then it can give relatively poor performance.

In this example, the process is given by $Ay(t)=Bu(t)+d(t)$ where $A=1-1.7z^{-1}$ and $B=0.75z^{-1}$. $d(t)$ is...
given by $d(t) = e(t)/(1-1.9e^{-1}+0.9e^{-2})$. $e(t)$ was non-zero at $t=50, 90, 160, 180$ and $225$. The performance ($y$ vs. $t$) of non-adaptive MAPC with $A$ and $B$ corresponding to actual process and with $F=\Delta$ and $D=1$ (almost no prior information is used) is shown in Fig. 6. The performance of non-adaptive MAPC with $F=\Delta$ and $D=1-0.9e^{-1}$ and $C=1$ (corresponding to an exact knowledge of the disturbances) is shown in Fig. 7. To illustrate residual modeling and two-stage identification it is assumed that no information regarding the disturbances is available. An AR residual model with initial parameters set to zero and $P(0)=1$ was identified and used. 10 AR parameters were estimated. $A$ and $B$ were taken to be the actual process polynomials (for fairness of comparison). The results are shown in Fig. 8. The results show that residual modeling and two-stage identification can give excellent performance (in this case the performance is the same as the one obtained with exact knowledge of the disturbances).

In all the runs the controller parameters were $N_1=N_2=N_u=1$ and $\lambda=0$. Note that if required the process model parameters can also be identified.

Handling MPM—The proposed method works well even when there is MPM due to, for example, reduced order modeling (under-parameterization). To illustrate this, the process given by $1/[(s+1)^2(3s+1)^2]$ was controlled using MAPC. The sampling period was taken as $T=1$ time unit. $2A$ and $2B$ parameters were estimated with no filtering of data. The parameters were first initialized to zero and identified in open-loop with random input excitation for first 50 sampling intervals before closing the loop. $P(0)=1$. The controller parameters were $N_1=1$, $N_2=20$, $N_u=2$ and $\lambda=1$, $R_a=0.1$ and $R_d=0.1$. The results are shown in Fig. 9. The performance is satisfactory (in view of the highly conservative (detuning) controller parameters used) and illustrates the robustness of the controller.

In summary the simulation examples presented here show that MAPC performs as expected and can give excellent results.

**Conclusions**

A multistep adaptive predictive control scheme has been developed. It can be applied to practical industrial processes. The proposed algorithm is based on the integration of the following key ideas: (i) use of multistep or long-range predictive control; (ii) state space model based representation of process dynamics and other unmodeled effects; (iii) use of Kalman filter for prediction of process output; (iv) use of robust estimation methods; (v) use of prior information; and (vi) use of multiple models (process dynamic model and residual model) and two-stage identification in the absence of any prior information.

The proposed algorithm compares favorably with GPC. Modeling and elimination of disturbances has been given special attention. The proposed scheme uses a more general model of the disturbances than the simple ARIMAX model used in GPC. The control algorithm can be adapted to any prior information.

In most self-tuning algorithms (including GPC) a disturbance model is estimated on-line along with process model parameters. Such an approach is not likely to be successful for various reasons. The proposed scheme provides a way to avoid this difficulty via residual modeling and two-stage identification.

There is no need to solve a Diophantine Identity on-line in the proposed scheme. The result is reduced computation.

A set of preliminary SISO simulated examples indicate excellent performance and justifies further evaluation including experimental studies.

**Acknowledgement**

The first author gratefully acknowledges the financial support from the Pool-Scheme of CSIR, India.
### Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A, B, C, D, F, F_x, A_m, B_m$</td>
<td>polynomials in $z^j$</td>
</tr>
<tr>
<td>$c_i, a_{nm}, b_{nm}$</td>
<td>coefficients of $C, A_m, B_m$ respectively</td>
</tr>
<tr>
<td>$e$</td>
<td>noise term</td>
</tr>
<tr>
<td>$f$</td>
<td>free response of the system</td>
</tr>
<tr>
<td>$G, G^*$</td>
<td>matrices</td>
</tr>
<tr>
<td>$g_i$</td>
<td>step response coefficients of the process</td>
</tr>
<tr>
<td>$H$</td>
<td>state space model matrix</td>
</tr>
<tr>
<td>$k$</td>
<td>process pure time delay</td>
</tr>
<tr>
<td>$K, K(t)$</td>
<td>Kalman filter gains</td>
</tr>
<tr>
<td>$N_1$</td>
<td>minimum output horizon</td>
</tr>
<tr>
<td>$N_2$</td>
<td>maximum output horizon</td>
</tr>
<tr>
<td>$N_y$</td>
<td>output horizon</td>
</tr>
<tr>
<td>$N_u$</td>
<td>control horizon</td>
</tr>
<tr>
<td>$P, P(t)$</td>
<td>covariance matrices of state estimation error</td>
</tr>
<tr>
<td>$Q$</td>
<td>square root of $R_w$</td>
</tr>
<tr>
<td>$R_w, R_s$</td>
<td>noise covariances</td>
</tr>
<tr>
<td>$r_s$</td>
<td>residual</td>
</tr>
<tr>
<td>$s$</td>
<td>laplace transform operator</td>
</tr>
<tr>
<td>$s$</td>
<td>process (control) input</td>
</tr>
<tr>
<td>$u$</td>
<td>vector of control increments</td>
</tr>
<tr>
<td>$v$</td>
<td>measurement noise</td>
</tr>
<tr>
<td>$w$</td>
<td>process noise</td>
</tr>
<tr>
<td>$x$</td>
<td>state vector</td>
</tr>
<tr>
<td>$y$</td>
<td>process output</td>
</tr>
<tr>
<td>$y$</td>
<td>vector of output predictions</td>
</tr>
<tr>
<td>$z^i$</td>
<td>backshift operator</td>
</tr>
</tbody>
</table>

### Greek letters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>reference trajectory parameter</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>differencing operator, $(1-z^\lambda)$</td>
</tr>
<tr>
<td>$\lambda, \Phi, \Gamma, \Lambda$</td>
<td>weighting on control action</td>
</tr>
<tr>
<td>$\Gamma_x, \Gamma_u$</td>
<td>state space model matrices</td>
</tr>
<tr>
<td>$\theta$</td>
<td>parameter vector</td>
</tr>
</tbody>
</table>

### Superscripts

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^T$</td>
<td>transpose</td>
</tr>
</tbody>
</table>

### Subscripts

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{ref}$</td>
<td>reference input</td>
</tr>
<tr>
<td>$sp$</td>
<td>setpoint</td>
</tr>
</tbody>
</table>

### Abbreviations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>artificial intelligence</td>
</tr>
<tr>
<td>AKPP</td>
<td>adaptive Kalman filter prediction</td>
</tr>
<tr>
<td>ARMAX</td>
<td>auto regressive moving average with exogenous input</td>
</tr>
<tr>
<td>ARIMAX</td>
<td>auto regressive integrated moving average with exogenous input</td>
</tr>
<tr>
<td>DMC</td>
<td>dynamic matrix control</td>
</tr>
<tr>
<td>GPC</td>
<td>generalized predictive control</td>
</tr>
<tr>
<td>IDCOM</td>
<td>identification and command</td>
</tr>
<tr>
<td>ILS</td>
<td>improved least squares</td>
</tr>
<tr>
<td>IMA</td>
<td>integrated moving average</td>
</tr>
<tr>
<td>KF</td>
<td>Kalman filter</td>
</tr>
<tr>
<td>LOG</td>
<td>linear quadratic gaussian</td>
</tr>
<tr>
<td>MA</td>
<td>moving average</td>
</tr>
<tr>
<td>MAPC</td>
<td>multistep adaptive predictive controller</td>
</tr>
<tr>
<td>MOCCA</td>
<td>multivariable, optimal, constrained control algorithm</td>
</tr>
<tr>
<td>MPM</td>
<td>model process mismatch</td>
</tr>
<tr>
<td>NMP</td>
<td>non minimum phase</td>
</tr>
<tr>
<td>RLS</td>
<td>recursive least squares</td>
</tr>
<tr>
<td>RELS</td>
<td>recursive extended least squares</td>
</tr>
<tr>
<td>SAE</td>
<td>sum of absolute errors</td>
</tr>
<tr>
<td>SISO</td>
<td>single-input, single-output</td>
</tr>
<tr>
<td>STR</td>
<td>self-tuning regulator</td>
</tr>
</tbody>
</table>

### References