Knowledge-Based Intelligent Systems Advancements: Systemic and Cybernetic Approaches

Chapter title:

Self-tuning Control Systems: A Review of Developments

Authors:

Keith J Burnham, Ivan Zajic, Jens G Linden
Control Theory and Applications Centre
Coventry University, Coventry, UK
(ctac@coventry.ac.uk)

Keywords: adaptive control, bilinear systems, control law design procedures, fault diagnosis, nonlinear models, parameter estimation, self-tuning control

ABSTRACT

A concise technical overview of some of the key 'landmark' developments in self-tuning control (STC) is presented. The notion of two coupled sub-algorithms forming the basis of STC together with enhancements to produce adaptive on-line procedures is discussed as well as the potential limitations of such schemes. The techniques covered include optimal minimum variance, sub-optimal pole-placement and long range model-based predictive control. Based on the experiences of the authors in the industrial application of STC, extensions of the standard linear model-based approaches to encompass a class of bilinear model-based schemes, is proposed. Some on-going developments and future research directions in STC for bilinear systems are highlighted. These include the requirements for combined algorithms for control and fault diagnosis and the need for models of differing complexities.

INTRODUCTION

The general aim of the chapter is to provide the reader with an overview of some of the key developments in the field of linear model-based STC. It also includes an introduction to some of the definitions that allow the classification of the resulting STC forms. The definition of STC as being one form of adaptive control which requires two coupled subalgorithms, one for on-line estimation of a discrete-time mathematical model of a plant and the other for control law design and implementation, is presented. The notion of repeatedly updating the model parameters via recursive estimation is introduced. Whilst reference is made to authoritative texts on the subject, a brief review of recursive least squares and Kalman filtering is given, together with extensions to enhance the adaptivity of the schemes. Then, three main categorisations of control law design are considered in the order of their historical development, namely: optimal d-step ahead control strategies (where d is defined later), sub-optimal pole-placement control strategies and long range model-based predictive control. The above developments are based on assuming a linear model representation for the system to be controlled. Various extensions and refinements have been proposed, and the chapter will provide the details of some of these developments, particularly those of the authors and their colleagues.

In particular, research conducted by the first author has shown that it is often found that the on-line parameter estimation algorithms can produce wildly varying estimations in cases when STC is applied to nonlinear systems. In such cases, the self-tuning principle may become violated, and an extension of the above STC strategies to deal with a class of bilinear systems has been considered. Adopting such a bilinear model representation potentially allows STC to be applied to a wider range of systems for which the notion of linearisation at a point is replaced by that of bilinearisation over a range. A review of some the more recent developments in the area of STC assuming a bilinear model representation is therefore included. Finally, a section containing concluding remarks is given which resumes the overall coverage of the chapter.

A discussion on future open research directions in which the notion of a combined approach for realising control and fault diagnosis and the need for different model complexities is presented in a section on additional reading.

BACKGROUND – TECHNICAL REVIEW OF SELF-TUNING CONTROL

This chapter on '*Self-tuning Control Systems: A Review of Developments*' aims to inform the reader of the major developments and historical landmarks in the topic up to the present day. The earliest reference dates back to the first International Symposium on Self-Adaptive Flight Control in 1959 which was held at what is now the Wright-Patterson Air Force Base, Dayton, Ohio, USA (Gregory, 1959), where the concept of 'self learning' control was first proposed. However, due to the lack of available technology at that time, in terms of reliable computer hardware and software, it was a decade before this concept was to re-emerge. In fact it re-emerged under the name of self-tuning control (STC) in the 1970s and was notably driven in those earlier years by Kalman (1960), Peterka (1970), and Astrom and Wittenmark (1973), who are now recognized as the early pioneers in this field. The major breakthrough by Astrom and Wittenmark (1973) with the optimal d-step ahead minimum variance (MV) self-tuning regulator/controller (STR)/STC in which convergence was proved for the simplest case was perhaps the first landmark which led to a positive resurgence and increased interest in the subject. This was followed in 1975 by the development due to Clarke and Gawthrop (1975) with the generalised minimum variance (GMV) STC in which constraints on control effort could be implemented to achieve a realizable control system. This led naturally to the incremental forms of MV and GMV STC, in which inherent integral action is automatically achieved.

The reader will be reminded that a model is only an approximation, however sophisticated it may appear, and that all models are developed and used for purpose and convenience. In fact, the notion of 'models for purpose' will feature as an underlying thread throughout the chapter, with models for the purpose of control being necessarily simpler in structure than some of their counterparts, e.g. those for fault diagnosis. The above MV and GMV schemes belong to a family of control systems which can be described as Linear Quadratic Gaussian (LQG) since the assumed plant model is linear, the cost function to be minimized is quadratic and the noise affecting the output of system is assumed to be Gaussian. The resulting MV and GMV controllers were developed initially for the auto-regressive with exogenous inputs (ARX) model representations and subsequently extended to the auto-regressive moving average with exogenous inputs (ARMAX) case. The development of the incremental forms led to proposals which made use of ARIMAX model representations, in which the assumed noise model is modified. It should be noted that model structures are normally adopted for convenience and the models commonly used in STC are outlined in the Section on *STC Model Structures*. The MV and GMV STR/C strategies are also known, as stated earlier, as optimal d-step ahead predictive schemes, since it is possible to predict the output d-steps ahead with knowledge of the system input at the current time step. Indeed, this forms the basis of the schemes, since knowing the desired output allows a quadratic cost function to be minimised in order to determine the optimal input. Unfortunately, however, to achieve this goal the resulting optimal STC cancels the process zeros, consequently rendering these approaches inadequate when dealing with non-minimum phase (NMP) systems.

Recognition of the shortfalls of the d-step ahead optimal schemes led to another landmark, namely the proposal for sub-optimal pole-placement STC strategies. These schemes are able to achieve their goals without affecting or utilizing the process zeros. Such a scheme was proposed by Wellstead et al. (1979), and developed within the ARX and ARMAX framework. The resulting controllers were demonstrated to be able to overcome the implementational problems with NMP systems, as experienced by the optimal schemes. The development led to alternative forms, and the state-space pole-placement STC was subsequently proposed by Warwick (1981). This made use of the so-called implicit delay observable canonical form within an innovations state-space setting. Whilst both control strategies are identical in the absence of output measurement noise, they differ in their behaviour in the presence of noise: the latter being due to the increased degree of filtering through the state space model structure. An interesting observation in the state-space equivalent of the ARX model is that the steady-state Kalman filter (SKF) used within the state-variable feedback (SVF) control law, is that the SKF converges to the true states in n-steps, with n being the order of the system. In the case of the equivalent ARMAX model, convergence is dependent on the locations of the zeros of the noise colouring polynomial.

Perhaps the most significant landmark in the development of control law design procedures to date has been that of long range (i.e. greater than d-steps ahead) model-based predictive control. Such an approach was proposed by Clarke et al. (1987). This approach differs from the previous proposals in that the controller not only utilises the actual measured signals, but it also utilises future predicted signals, based on knowledge of the set point in advance. The approach developed in (Clarke et al., 1987) is known as generalised predictive control (GPC) and this is formulated in the incremental control framework, i.e. it utilises the ARIMAX model structure. The basis of the approach is to assume that no further action in terms of incremental controls will take place so that the future control remains constant up to a user defined prediction horizon h-steps ahead (where h is greater than d). By separating the contributions to the future outputs which can be accounted for at the current time, due to current at previous controls, allows a deficit to be predicted, which is essentially the predicted future error that would appear if no adjustment to the control action is made. Then, by representing these future predicted errors in vector form, it is possible to design a suitable quadratic cost function, the minimisation of which will yield a vector of optimal future incremental controls. At each time step the procedure is repeated, thus leading to the notion of a receding horizon approach. Details regarding these key developments of the control law design procedures are provided in the Section on *Control Law Design Procedures*.

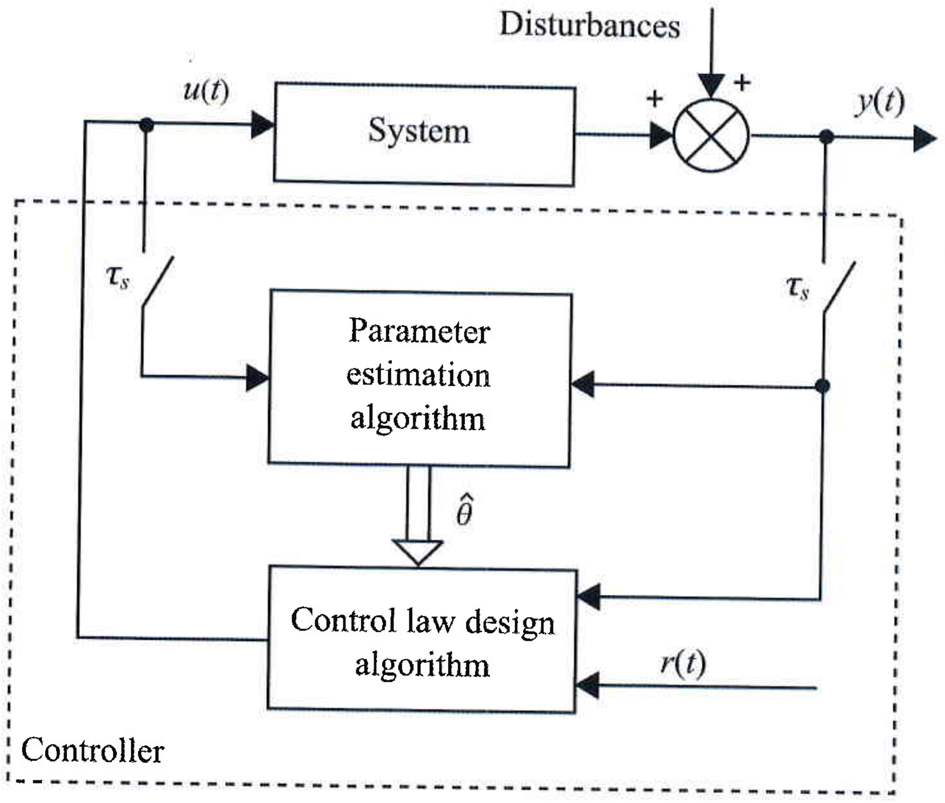
This historical-technical review will also consider the development of on-line parameter estimation algorithms for use in STC. Whilst only outlined briefly here, the developments are fully supported by reference material to the original works, where the reader can find detailed derivations. For example the reader will find the original development of the recursive least squares (RLS) algorithm of Plackett (1950), extensions to include extended least squares (ELS), use of forgetting factors and variable forms of forgetting (e.g. due to Fortescue et al. (1981)) to be of value. Utilisation of the Kalman filter (KF) for parameter estimation (following a brief review of its original development for linear state estimation, (Kalman, 1960)) is presented. Whilst the use of coupled KFs for joint state and parameter estimation will be briefly discussed, as well as the extended KF (EKF), e.g. (Young, 1974), for simultaneous state and parameter estimation, a detailed discussion is not given here. In parallel with developments in computer technology, the middle 1980s witnessed some important developments and enhancements in regard to the estimation algorithms used in STC. For example, for the first time it became possible to make repeated on-line use of forgetting factors (leading to variable forgetting factors), covariance matrix resetting techniques and the realisation of methods based on instrumental variables (Young, 1984). Aspects regarding the developments of the on-line parameter estimation algorithms are provided in the Section on *Parameter Estimation Procedures*.

SELF-TUNING CONTROL CONCEPT

Essentially a STC comprises two coupled subalgorithms, one for the online estimation of the parameters of an assumed model and the other for evaluating the control action from a suitable control law design procedure. In principle any estimation algorithm can be combined with any control law design algorithm, thus the scope is wide and the final choice of this combination will depend on the particular application. In the following, the estimation and control law design algorithms will be introduced separately. Later, in the simulation study in the Section on *Bilinear GPC* the algorithms are combined when a self-tuning linear GPC scheme is applied to a nonlinear system.

In order to fully exploit the STC concept the models upon which the model-based controllers are based are required to be repeatedly updated as the system is driven over the operational range of interest. If the operating range is small then a local linear model with fixed parameters may be sufficient. If, however, the operational range is increased the assumptions on local linearity for the system to be controlled may become violated. Under such conditions the overall closed-loop performance will become reduced due to the increase in the mismatch between the system and model. Alternative approaches using controller gain scheduling, look-up tables as well as multiple switched/blended model solutions have been considered. However, the notion of STC whereby the model parameters are continually updated, as the operating range is traversed, is in effect an infinite model approach, with the advantage that as the system and/or subsystem components change over time, then so do the resulting models. This repeated updating of the model parameters exploites the notion of certainty equivalence in that the estimated values are at each time step assumed to be correct. Taking the approach one step further, it may also be possible, using the same measured input/output data, to detect the onset of a fault condition. Such a concept enables to the establishment of thresholds within which non-violation of certain inequalities allows the implementation of adaptive control via STC, and conversely would allow either a fault detection, or an active fault tolerant control scheme to be triggered. Whilst it is possible, in principle, to combine any model-based control law design procedure with any suitable estimation algorithm, there are certain classifications of STC. The first is to consider the indirect (or explicit) and direct (or implicit) STC schemes. In an indirect direct approach, or explicit scheme, the control law is obtained from the estimated model parameters; the latter are explicitly available for interrogation/monitoring, thus allowing some degree of intervention between the two coupled algorithms. In the direct approach, on the other hand, the control law is direclty estimated from the input/output data along with the estimated model parameters; the latter being implicit within the scheme (i.e. not explicitly available). A further classification which, is possible in the case of both direct and indirect STC schemes is to make the distinction between non-dual and dual STC. In a non-dual STC the control action is required to perform the role of an ideal control signal only, whereas in the dual approach the control action is not only ideal for control, but is also an ideal signal from an estimation view point. In the remainder of the work is this chapter consideration is given to an explicit non-dual STC. In other words the control action is ideal for control only and the parameters are explicitly available from the estimation algorithm. It is also worth noting in the context of a linear STC applied to nonlinear systems that the self-tuning principle, which holds when estimated model parameters converge to steady values, may become invalidated. Thus further justifing a nonlinear, restricted here to bilinear, STC approach. A block diagram representation of a general explicit non-dual STC scheme is given in Figure 1.

Figure 1. Block diagram representation of an explicit non-dual STC, where $u(t)$, $y(t)$, $r(t)$, τ_s and θ are defined later.



STC MODEL STRUCTURES

A widely used and relatively simple model is the so-called ARX (auto regressive with exogenous inputs) model, where the additive disturbance on the output is assumed to be a white signal having zero mean value. An extension of this model structure is the so-called ARMAX (auto regressive moving average with exogenous inputs) model structure, where the noise is no longer assumed to be white, but is modelled as the output of a moving average process. A further extension is the ARIMAX (auto regressive integrated moving average with exogenous inputs) model. In order to proceed, the various model structures are briefly introduced. The ARMAX/ARIMAX model structure can be expressed in the form

$$A(q^{-1})y(t) = q^{-d}B(q^{-1})u(t) + \xi(t) \qquad (0.1)$$

where q^{-1} denotes the backward shift operator defined such that $q^{-i}y(t)=y(t-i)$ and t is the discrete-time index. When dealing with discrete time control it is normal to assume the existence of a zero-order-hold in the input channels, such that $d \geq 1$ represents the integer valued quantity D/τ_s rounded up; D being the system time delay and τ_s the adopted sampling interval. As such, d is regarded as the normalised system time delay. The sampled discrete-time system output and input signals at time t are denoted $y(t)$ and $u(t)$, respectively, and the polynomials $A(q^{-1})$ and $B(q^{-1})$ are defined as

$$A(q^{-1})=a_0+a_1q^{-1}+a_2q^{-2}+\dots+a_{n_a}q^{-n_a},\ a_0=1, \tag{0.2}$$

$$B(q^{-1})=b_0+b_1q^{-1}+b_2q^{-2}+\dots+b_{n_b}q^{-n_b},\ b_0\neq 0. \tag{0.3}$$

In STC the model parameter vector, denoted

$$\theta=\begin{bmatrix} a_1 & \dots & a_{n_a} & b_0 & \dots & b_{n_b}\end{bmatrix}^T \tag{0.4}$$

of the ARX model is required to be estimated (i.e. continuously updated) at each time step. The ARMAX and ARIMAX structures differ in the way the additive output disturbance signal, denoted $\xi(t)$, is modelled. The disturbance term in the case of the ARMAX model structure is described as a moving average process

$$\xi(t)=C(q^{-1})e(t) \tag{0.5}$$

where $e(t)$ is a discrete white noise signal having the variance σ_e^2 and which is coloured by the polynomial $C(q^{-1})$ defined as

$$C(q^{-1})=c_0+c_1q^{-1}+c_2q^{-2}+\dots+c_{n_c}q^{-n_c},\ c_0=1. \tag{0.6}$$

However, in many practical problems the disturbance process cannot sufficiently be described as a moving average process. Common examples for such situations are cases when the noise term contains an offset value, i.e. if $\xi(t)=C(q^{-1})e(t)+o(t)$, where $o(t)$ denotes a (potentially time-varying) offset. The disturbance term of the ARIMAX model structure can successfully deal with these cases and is defined as an integrated moving average process

$$\xi(t) = \frac{C(q^{-1})}{\Delta} e(t) \qquad (0.7)$$

where Δ is defined such that $\Delta = 1 - q^{-1}$. The ARIMAX model structure also offers inherent integration action which is exploited for the controller design in incremental form. Finally, the ARX model structure can be considered as a subset of the ARMAX model structure for the case where $n_c = 0$, i.e. the noise colouring polynomial $C(q^{-1}) = 1$. Note that in the case of $n_c > 0$ the parameter vector θ is extended to include the coefficients of the noise colouring polynomial, denoted c_i, $i = 1 \ldots n_c$, i.e.

$$\theta = \begin{bmatrix} a_1 & \ldots & a_{n_a} & b_0 & \ldots & b_{n_b} & c_1 & \ldots & c_{n_c} \end{bmatrix}^T, \qquad (0.8)$$

thus requiring ELS techniques to be employed.

PARAMETER ESTIMATION PROCEDURES

LINEAR LEAST SQUARES

The method of linear least squares (LLS) is perhaps the most basic and yet widely used approach for estimating the parameters of an assumed model structure of a system in control engineering. LLS is used as an off-line parameter estimator, i.e. for estimating the parameter vector, denoted θ, based on a batch of past input/output data pairs. This section provides a summary of the properties of the LLS method. Assume an ARX model structure, i.e. $C(q^{-1}) = 1$, expressed in the form

$$y(t) = -a_1 y(t-1) \ldots - a_{n_a} y(t-n_a) + b_0 u(t-d) \ldots + b_{n_b} u(t-d-n_b) + e(t) \qquad (0.9)$$

or alternatively as a linear regression, i.e.

$$y(t) = \varphi^T(t)\theta + e(t), \tag{0.10}$$

where the vector of observations, also known as the regression vector, is given by

$$\varphi(t) = \left[-y(t-1) \quad \ldots \quad -y(t-n_a) \quad u(t-d) \quad \ldots \quad u(t-d-n_b)\right]^T. \tag{0.11}$$

The regression vector comprises of $n_a + n_b + 1$ regressors, which are observed data in discrete time $t = 1,\ldots,N$, where N denotes the number of observations (measurements). The regression vector consists of the past values of the system output and the system input. It is interesting to note that the word 'regression' is derived from the Latin word 'regredi', which means 'to go back'.

The predicted system output, denoted $\hat{y}(t\,|\,\theta)$, based on the parameter vector θ can then be computed as

$$\hat{y}(t\,|\,\theta) = \varphi^T(t)\theta. \tag{0.12}$$

Thus the prediction error, or residual, between the measured and the predicted output can be expressed as

$$\varepsilon(t) = y(t) - \hat{y}(t\,|\,\theta). \tag{0.13}$$

The method of LLS estimates the parameter vector as a best fit between the measured output $y(t)$ and predicted output $\hat{y}(t\,|\,\theta)$ over $t = 1,\ldots,N$, such that the sum of squared residuals is minimised, i.e.

$$J_N(\theta) = \frac{1}{N}\sum_{t=1}^{N}[\varepsilon(t)]^2 = \frac{1}{N}\sum_{t=1}^{N}\left[y(t) - \varphi^T(t)\theta\right]^2. \tag{0.14}$$

The quadratic cost function eq. (0.14) can be solved analytically

$$\hat{\theta} = \arg\min_{\theta} J_N(\theta) \tag{0.15}$$

and the algorithm of LLS is then given by

$$\hat{\theta}(t) = \left[\sum_{t=1}^{N}\varphi(t)\varphi^T(t)\right]^{-1}\sum_{t=1}^{N}\varphi(t)y(t). \tag{0.16}$$

In order to evaluate the accuracy of the estimator consider the estimation error vector defined as

$$\tilde{\theta}=\theta-\hat{\theta}. \tag{0.17}$$

Since in practice the true parameter vector θ is not exactly known, it follows that the estimation error vector is also unknown. However, considering the covariance matrix corresponding to the estimation error vector, defined by

$$R=E\left[\tilde{\theta}\tilde{\theta}^T\right], \tag{0.18}$$

where $E[\cdot]$ denotes the mathematical expectation operator, it can be shown that

$$R=\left[\sum_{t=1}^{N}\varphi(t)\varphi^T(t)\right]^{-1}\sigma_e^2. \tag{0.19}$$

Commonly only the approximate scaled error covariance matrix is available, i.e.

$$P=\left[\sum_{t=1}^{N}\varphi(t)\varphi^T(t)\right]^{-1}, \tag{0.20}$$

which is readily observed to be related to the true covariance matrix via the unknown positive scalar σ_e^2. The scaled, matrix P can be computed together with $\hat{\theta}$ from eq. (0.16). The square roots of the diagonal elements of P correspond to the standard deviations of the individual estimated parameters. This is a useful observation which can be exploited, hence the LLS algorithm, via the error covariance matrix, automatically provides information about the accuracy of the estimates.

RECURSIVE LEAST SQUARES

In the STC framework there are practical issues, which require that it is necessary to perform on-line estimation at each time step in order to repeatedly update the estimated parameter vector $\hat{\theta}(t)$ as new observation data are obtained. For this type of problem the offline LLS method in is inefficient, because the observed data set grows larger and larger at each time step. Consequently the computation which ultimately results in the inversion of the matrix P becomes more costly and the demand on computer memory becomes higher as new observations are made. An efficient way to perform this type of on-line estimation is to make use of a RLS scheme. The general form of the RLS algorithm may be stated as

$$\begin{aligned}&[\text{New Parameter Vector}] = [\text{Previous Parameter Vector}] \\ &\qquad + [\text{Correction}][\text{Measured Output} - \text{Predicted Output}],\end{aligned} \tag{0.21}$$

where the new parameter vector, denoted $\hat{\theta}(t)$, is updated based on its previous value, denoted $\hat{\theta}(t-1)$, and the latest measured output $y(t)$. The RLS algorithm originally developed by Plackett (1950), is simply stated here, see e.g. (Ljung, 1999), as:

$$\begin{aligned}L(t) &= P(t-1)\varphi(t)\left[\lambda + \varphi^T(t)P(t-1)\varphi(t)\right]^{-1}, \\ \hat{\theta}(t) &= \hat{\theta}(t-1) + L(t)\left[y(t) - \varphi^T(t)\hat{\theta}(t-1)\right], \\ P(t) &= \left[P(t-1) - L(t)\varphi^T(t)P(t-1)\right]\lambda^{-1},\end{aligned} \tag{0.22}$$

where $0 < \lambda \leq 1$ is a forgetting factor used to repeatedly inflate elements of the covariance matrix, thus keeping the algorithm alert and assisting adaptation (Hsia, 1977). The choice of the forgetting factor is a compromise between algorithm alertness and noise sensitivity (Burnham et al., 1985). To alleviate this problem, use may be made of a variable forgetting factor $\lambda(t)$ which is adjusted as a function of the estimation prediction error to retain the information content within the algorithm (Fortescue et al., 1981; Wellstead and Sanoff, 1981). Whilst use of a forgetting factor facilitates the tracking of slow variation in parameters, a technique that facilitates the tracking of rapid parameter variation is that of covariance matrix reset. Such a scheme, which can be operated in conjunction with forgetting factors, may trigger reset on set point change, periodically or on detection of large errors in estimation.

It should be noted that unbiased parameter estimates can only be obtained from RLS if the observation vector and the noise sequence are uncorrelated (Young, 1974); true only in the case of a white output noise sequence. Alternatively the problem of biased estimates may be alleviated using algorithms such as ELS, recursive maximum likelihood (Hsia, 1977), recursive instrumental variables (Young, 1970) or a KF configured for parameter estimation (Randall et al., 1991), which is reviewed in following section. If poor parameter estimates are obtained due to insufficient input signal excitation cautious least squares (CLS) may be employed (Burnham and James, 1986; Randall and Burnham, 1994) in which the algorithm is kept alert without disturbing the plant. CLS is also useful when attempting to constrain the estimated parameters to remain within sensible regions based on experience and knowledge of the plant. CLS has been shown to be an adaptive form of online Tikhonov regularisation (Linden, 2005).

KALMAN FILTER CONFIGURED FOR PARAMETER ESTIMATION

The KF was originally developed for estimating the unmeasurable state vector of a linear dynamic system, however the KF finds application in parameter estimation as well. This is due in part to the fact that the KF allows individual forgetting for each parameter, i.e. selective adaptivity. Consider a time varying state-space representation of an unforced discrete-time system subject to white process noise

$$\begin{aligned} x(t+1) &= Ax(t)+v(t), \\ y(t) &= Cx(t)+e(t), \end{aligned} \tag{0.23}$$

where $x(t)$ is the state vector of dimension $n\times 1$, A is an $n\times n$ state transition matrix, $v(t)$ is an $n\times 1$ process noise vector, $y(t)$ is the measured system output, C is an $1\times n$ output vector and $e(t)$ is the measurement noise. The random processes $v(t)$ and $e(t)$ have zero mean values, i.e.

$$E[v_1(t)], E[v_2(t)]\ldots E[v_n(t)]=0,\ E[e(t)]=0. \tag{0.24}$$

The covariance matrices are

$$E[v(i)v^T(j)] = V\delta_{ij},$$
$$E[e(i)e^T(j)] = R\delta_{ij}, \tag{0.25}$$

where δ_{ij} is the Kronecker delta function, i.e. having value of unity if $j=i$ and null if $j \neq i$.

The processes are independent of each other, hence

$$E[v(t)e(t)] = 0. \tag{0.26}$$

The KF for state estimation comprises of two parts and is given by

Prediction (between samples based on the state equation):

The estimated state $\hat{x}(t\,|\,t-1)$ at time step t given information up to and including time step $t-1$ is computed as

$$\hat{x}(t\,|\,t-1) = A(t-1)\hat{x}(t-1\,|\,t-1) \tag{0.27}$$

and the update of the covariance matrix is

$$P(t\,|\,t-1) = A(t-1)P(t-1\,|\,t-1)A^T(t-1) + V(t-1). \tag{0.28}$$

Correction (at the sample instants based on the output equation):

The Kalman gain vector is given by

$$K(t) = \frac{P(t\,|\,t-1)C^T(t)}{R(t) + C(t)P(t\,|\,t-1)C^T(t)} \tag{0.29}$$

and the new corrected state estimate is then obtained from

$$\hat{x}(t\,|\,t) = \hat{x}(t\,|\,t-1) + K(t)\left[y(t) - C(t)\hat{x}(t\,|\,t-1)\right]. \tag{0.30}$$

The updated error covariance matrix is computed as

$$P(t\,|\,t) = P(t\,|\,t-1) - K(t)C(t)P(t\,|\,t-1). \tag{0.31}$$

The KF can be also configured for parameter estimation. Consider the ARX model structure expressed in the regression form

$$y(t) = \varphi^T(t)\theta(t) + e(t), \tag{0.32}$$

where the parameter vector is time-varying and may be defined as

$$\theta(t) = \theta(t-1) + v(t)\,. \tag{0.33}$$

The task is now to estimate the parameter vector $\theta(t)$. The similarity of the state equation in eq. (0.23) to eq. (0.33) and the output equation in eq. (0.23) to eq. (0.32) becomes obvious, hence the state-space model for the parameter estimation problem is stated

$$\begin{aligned} \theta(t) &= \theta(t-1) + v(t), \\ y(t) &= \varphi^T(t)\theta(t) + e(t), \end{aligned} \tag{0.34}$$

where the state transition matrix is simply the identity matrix and the output vector is the observation vector. The KF algorithm configured for parameter estimation is thus given by

Prediction (between samples based on the state equation and any other a prior knowledge):

$$\hat{\theta}(t \mid t-1) = \hat{\theta}(t-1 \mid t-1) \tag{0.35}$$

$$P(t \mid t-1) = P(t-1 \mid t-1) + V(t-1) \tag{0.36}$$

Correction (at the sampling instants based on the measurement from the output equation):

$$K(t) = \frac{P(t \mid t-1)\varphi(t)}{R(t) + \varphi^T(t)P(t \mid t-1)\varphi(t)} \tag{0.37}$$

$$\hat{\theta}(t \mid t) = \hat{\theta}(t \mid t-1) + K(t)\left[y(t) - \varphi^T(t)\hat{\theta}(t \mid t-1)\right] \tag{0.38}$$

$$P(t \mid t) = P(t \mid t-1) - K(t)\varphi^T(t)P(t \mid t-1) \tag{0.39}$$

The main difference between RLS and the KF for parameter estimation is the way in which the algorithms are tuned to track parameter variation. Whereas the RLS algorithm uses a scalar valued forgetting factor to give equal adaptivity for all parameters, the KF, via the diagonal elements in V in the covariance matrix prediction step, utilises selective adaptivity. In other words, rather than inflating the covariance matrix by dividing by a scalar less than unity as in RLS, the inflation step in the KF is carried out by addition of the matrix V. In this way varying degrees of adaptation may be realised, thus allowing a priori knowledge to be incorporated into the algorithm. Whilst it is usual to consider only the null or positive entries on the diagonal, the off-diagonal entries may also be exploited to build-in further knowledge on the cross-correlation between certain model parameters.

CONTROL LAW DESIGN PROCEDURES

MINIMUM VARIANCE REGULATOR/CONTROLLER

The minimum variance (MV) regulators and controllers are considered as a class of optimal schemes, where the optimality is defined by a prescribed cost function. The aim is to minimise the variance of the system output $y(t)$ via an optimal control input $u(t)$. The optimal value of $u(t)$, in the MV sense, is fulfilled when the following assumptions hold:

Assumption 1 *The system to be controlled is linear.*

Assumption 2 *The cost function J is quadratic.*

Assumption 3 *Noise affecting the system output is Gaussian.*

Thus the MV regulators/controllers are also regarded as belonging to the family of LQG (linear, quadratic, Gaussian) regulators/controllers.

MINIMUM VARIANCE REGULATOR

Consideration is initially restricted here to the regulator problem, i.e. the desired output or set point, denoted $r(t)$, is equal to zero. The MV regulator cost function is defined as follows

$$J_R = E\left[y^2(t+d)\right] \tag{0.40}$$

where d denotes the normalised system time delay. The objective is to determine the optimum value of the current system input $u(t)$, which minimises the cost function eq. (0.40) . Note that the current system input at discrete time t affects the future system output at time $(t+d)$. The MV algorithm can be derived assuming different model structures. However, for ease of derivation only the ARX models are considered here.

Prior to deriving the general form of the MV algorithm for any ARX model structure it is helpful and intuitive to consider the following particular example.

Example 1. Consider the system described by an ARX model structure, i.e. $C(q^{-1})=1$, having $n_a=2$, $n_b=1$ and $d=1$ expressed as a linear difference equation

$$(1+a_1q^{-1}+a_2q^{-2})y(t)=q^{-1}(b_0+b_1q^{-1})u(t)+e(t). \tag{0.41}$$

Expanding and rearranging to a more convenient form leads to

$$y(t)=-a_1y(t-1)-a_2y(t-2)+b_0u(t-1)+b_1u(t-2)+e(t), \tag{0.42}$$

where $y(t)$ is a linear combination of the past outputs and past inputs with the most recent input affecting the current output being delayed by one sample step. Since the objective is to determine the current input $u(t)$, shifting forward by one step leads to

$$y(t+1)=-a_1y(t)-a_2y(t-1)+b_0u(t)+b_1u(t-1)+e(t+1). \tag{0.43}$$

Note that in general (i.e. for any $d\geq 1$) it is possible to predict the output values up to time $(t+d)$ based on the current and past values of control actions. Consequently the MV schemes are also known as d-step ahead predictive schemes. In general, the optimal value of $u(t)$ is obtained by differentiating the cost function eq. (0.40) with respect to (w.r.t) the argument $u(t)$ and equating to zero for minimum, i.e.

$$u(t)=\arg\min_{u(t)} J_R(u(t)). \tag{0.44}$$

This procedure can be performed in four steps:

1) Expand quadratic cost function

Prior to expanding the cost function J_R a number of preliminary issues are highlighted. The output $y(t+1)$ in eq. (0.43) is unknown since the future random disturbance $e(t+1)$ is unpredictable. The quantity $y(t+1)$ can be separated in two parts as follows

$$y(t+1) = \hat{y}(t+1|t) + e(t+1), \tag{0.45}$$

where $\hat{y}(t+1|t)$ denotes the best prediction of $y(t+1)$ based on information available up to and including time t (in the sense of minimising the squared prediction error) and $e(t+1)$ is the unknown noise term. The term $\hat{y}(t+1|t)$ is then expressed as

$$\hat{y}(t+1|t) = -a_1 y(t) - a_2 y(t-1) + b_0 u(t) + b_1 u(t-1). \tag{0.46}$$

The cost function eq. (0.40) can then be expressed in the form

$$\begin{aligned} J_R &= E\left[y^2(t+1)\right] \\ &= E\left[\hat{y}(t+1|t) + e(t+1)\right]^2 \\ &= E\left[\hat{y}(t+1|t)\right]^2 + 2E\left[\hat{y}(t+1|t)e(t+1)\right] + E\left[e(t+1)\right]^2. \end{aligned} \tag{0.47}$$

Since the noise is independent of the predicted output the second term of eq. (0.47) vanishes. The third term, by definition, is the noise variance σ_e^2. The cost function J_R can thus be expressed as

$$J_R = E\left[\hat{y}(t+1|t)\right]^2 + \sigma_e^2. \tag{0.48}$$

Note that the minimal achievable cost of the above expression is the noise variance σ_e^2, since the term $[\hat{y}(t+1|t)]^2$ is forced to be null by the control action. The expansion of the cost function J_R can be carried out as follows

$$\begin{aligned} J_R &= E\left[\hat{y}(t+1|t)\right]^2 + \sigma_e^2 \\ &= (-a_1 y(t) - a_2 y(t-1) + b_0 u(t) + b_1 u(t-1))^2 + \sigma_e^2 \end{aligned} \tag{0.49}$$

by omitting terms that do not involve $u(t)$, define the modified cost function $\tilde{J}_R$, i.e.

$$\tilde{J}_R = 2b_0 u(t)(-a_1 y(t) - a_2 y(t-1) + b_1 u(t-1)) + b_0^2 u^2(t). \tag{0.50}$$

2) Differentiate with respect to the argument

The expanded cost function eq. (0.50) is differentiated w.r.t. $u(t)$ as follows

$$\frac{\partial \tilde{J}_R}{\partial u(t)} = 2b_0(-a_1 y(t) - a_2 y(t-1) + b_1 u(t-1)) + 2b_0^2 u(t). \tag{0.51}$$

3) Equate to zero for a minimum

The next step is to equate eq. (0.51) to zero for obtaining a minimum, hence

$$b_0(-a_1y(t)-a_2y(t-1)+b_1u(t-1))+b_0^2u(t)=0. \tag{0.52}$$

Note that since the system is linear a global minimum is obtained.

4) Determine control action

Rearranging eq. (0.52) to solve for $u(t)$ gives the MV regulator algorithm

$$u(t)=\frac{a_1y(t)+a_2y(t-1)-b_1u(t-1)}{b_0}. \tag{0.53}$$

Note that the above result reinforces the need for $b_0 \neq 0$. The MV regulator algorithm in the case of any value of n_a and n_b and for a fixed value of $d=1$ is then given by

$$u(t)=\frac{1}{b_0}\left[\sum_{i=1}^{n_a}a_iy(t+d-i)-\sum_{i=1}^{n_b}b_iu(t-i)\right]. \tag{0.54}$$

□

The general form of the MV regulator for an ARX model structure assuming $d \geq 1$ is now cosidered. The d-step ahead prediction of the system output is required. This is accomplished through the linear predictor. The predictor of $y(t+d)$ minimises the mathematical expectation of the squared prediction error $\varepsilon(t)$, i.e.

$$\begin{aligned}\hat{y}(t+j\,|\,t)&=\arg\min_{\hat{y}(t+j|t)} E\left[\varepsilon^2(t+j)\right],\\ &=\arg\min_{\hat{y}(t+j|t)} E\left[y(t+j)-\hat{y}(t+j\,|\,t)\right]^2,\end{aligned} \tag{0.55}$$

where $\hat{y}(t+j\,|\,t)$ denotes the prediction of $y(t+j)$ based on information available up to and including time t and over the range $j=1,\ldots,d$. Computing the prediction of the output by minimisation of eq. (0.55) for higher values of the delay $d>1$ is rather impractical and a recursive form of the d-step ahead predictor is developed instead, which can be relatively straightforwardly programmed. The d-step ahead predictor of the system output for the ARX model structure is given by

$$\hat{y}(t+j\,|\,t) = M_j(q^{-1})y(t) + N_j(q^{-1})u(t), \tag{0.56}$$

where the polynomials $M_j(q^{-1})$ and $N_j(q^{-1})$ are, respectively, defined as

$$M_j(q^{-1}) = m_{j,0} + m_{j,1}q^{-1} + m_{j,2}q^{-2} + \dots + m_{j,i}q^{-i},\ i = n_a - 1 = n_m, \tag{0.57}$$

$$N_j(q^{-1}) = n_{j,0} + n_{j,1}q^{-1} + n_{j,2}q^{-2} + \dots + n_{j,i}q^{-i},\ i = n_b + j - 1 = n_n. \tag{0.58}$$

The individual coefficients $m_{j,i}$ and $n_{j,i}$ are generated, respectively, as follows

$$m_{j,i} = \sum_{l=1}^{j}[-a_l m_{j-l,i}] - a_{j+i} \tag{0.59}$$

and

$$n_{j,i} = b_i - \sum_{l=1}^{j}[a_l n_{j-l,i-l}], \tag{0.60}$$

where $m_{j-l,i} = 0$ if subscript $j = l$, and the term $n_{j-l,i-l} = 0$ if $j = l$ or $l \geq i$. The procedure of generating the polynomials $M_j(q^{-1})$ and $N_j(q^{-1})$ is shown in the following illustrative example.

Example 2. Generate the coefficients of the polynomials $M_j(q^{-1})$ and $N_j(q^{-1})$ for the ARX model structure having $n_a = 3$, $n_b = 2$ and $d = 2$. The model is given by

$$y(t) = -a_1y(t-1) - a_2y(t-2) - a_3y(t-3) - b_0u(t-2) - b_1u(t-3) - b_2u(t-4). \tag{0.61}$$

Shifting forward by one step the prediction at time $(t+1)$ is computed as

$$\hat{y}(t+1\,|\,t) = -a_1y(t) - a_2y(t-1) - a_3y(t-2) - b_0u(t-1) - b_1u(t-2) - b_2u(t-3) \tag{0.62}$$

and shifting forward by one more step the prediction at time $(t+2)$ becomes

$$\hat{y}(t+2\,|\,t) = -a_1\hat{y}(t+1\,|\,t) - a_2y(t) - a_3y(t-1) - b_0u(t) - b_1u(t-1) - b_2u(t-2). \tag{0.63}$$

Substituting eq. (0.62) for $\hat{y}(t+1\,|\,t)$ in eq. (0.63) leads to

$$\begin{aligned}\hat{y}(t+2\,|\,t)=&\left(a_1^2-a_2\right)y(t)+\left(a_1a_2-a_2\right)y(t-1)+\left(a_1a_3\right)y(t-2)+b_0u(t)\\&+\left(b_1-a_1b_0\right)u(t-1)+\left(b_2-a_1b_1\right)u(t-2)+\left(-a_1b_2\right)u(t-3),\end{aligned}\tag{0.64}$$

which is the desired prediction of the system output at time $(t+d)$. The same results will now be obtained utilizing the predictor eq. (0.56). The $M_j(q^{-1})$ and $N_j(q^{-1})$ polynomials are computed recursively for $j=1,\ldots,d$. Starting with the prediction $j=1$, the $M_1(q^{-1})$ polynomial has order $n_m=n_a-1=2$ and, making use of eq. (0.59), its coefficients are computed as

$$\begin{aligned}m_{1,0}&=-a_1m_{0,0}-a_1=-a_1,\\m_{1,1}&=-a_1m_{0,1}-a_2=-a_2,\\m_{1,2}&=-a_1m_{0,2}-a_3=-a_3.\end{aligned}\tag{0.65}$$

The $N_1(q^{-1})$ polynomial has order $n_n=n_b+j-1=2$ and, utilizing eq. (0.60), the individual coefficients are computed as

$$\begin{aligned}n_{1,0}&=b_0-a_1n_{0,-1}=b_0,\\n_{1,1}&=b_1-a_1n_{0,0}=b_1,\\n_{1,2}&=b_2-a_1n_{0,1}=b_2.\end{aligned}\tag{0.66}$$

For the prediction $j=2$, the orders of the corresponding $M_2(q^{-1})$ and $N_2(q^{-1})$ polynomials are $n_m=n_a-1=2$ and $n_n=n_b+j-1=3$, respectively, so that the individual coefficients are obtained as

$$\begin{aligned}m_{2,0}&=\left(-a_1m_{1,0}-a_2m_{0,0}\right)-a_2=a_1a_1-a_2,\\m_{2,1}&=\left(-a_1m_{1,1}-a_2m_{0,1}\right)-a_3=a_1a_2-a_3,\\m_{2,2}&=(-a_1m_{1,2}-a_2m_{0,2})-a_4=a_1a_3,\end{aligned}\tag{0.67}$$

and

$$
\begin{aligned}
n_{2,0} &= b_0 - \left(a_1 n_{1,-1} + a_2 n_{0,-2}\right) = b_0, \\
n_{2,1} &= b_1 - \left(a_1 n_{1,0} + a_2 n_{0,-1}\right) = b_1 - a_1 b_0, \\
n_{2,2} &= b_2 - \left(a_1 n_{1,1} + a_2 n_{0,0}\right) = b_2 - a_1 b_1, \\
n_{2,3} &= b_3 - \left(a_1 n_{1,2} + a_2 n_{0,1}\right) = -a_1 b_2,
\end{aligned}
\tag{0.68}
$$

respectively.

□

Minimising the cost function eq. (0.40) and utilisng the d-step ahead predictor eq. (0.56) leads to the general MV regulator algorithm for an ARX model structure

$$
u(t) = \frac{1}{b_0}\left[-\sum_{i=0}^{n_m} m_{j,i} y(t-i) - \sum_{i=1}^{n_n} n_{j,i} u(t-i) \right], \tag{0.69}
$$

where $j = d$ and $b_0 = n_{d,0}$. Note that the recursive generation of $\hat{y}(t+j\,|\,t)$ from eq. (0.56) is not the only approach for developing the MV controller. A widely utilised alternative is the adoption of the so-called Diophantine equation (Clarke et al., 1987; Wellstead and Zarrop, 1991). This approach is directly applicable for any ARX, ARMAX and ARIMAX model structure.

MINIMUM VARIANCE CONTROLLER

In many industrial applications the aim is not just to drive the output to a zero value, as in the regulator case, but to track a reference signal $r(t)$, which is then referred to as a servo controller. The reference signal $r(t)$ is known up to and inlcuding time t. The servo controller MV cost function is defind as

$$
J_S = E\left[y(t+d) - r(t)\right]^2. \tag{0.70}
$$

In a similar manner to the regulator case, a derivation of the MV control algorithm is highlighted initially via a particular example which is then followed by a generalised algorithm for an ARX model structure.

Example 3. Consider a system described by an ARX model structure having $n_a = 2$, $n_b = 1$ and $d = 1$. As for the MV regulator, following the four step procedure, the first step is the expansion of the quadratic cost function, which is now defined by

$$J_S = E\left[y(t+1) - r(t)\right]^2, \tag{0.71}$$

where, substituting $\hat{y}(t+1\,|\,t) + e(t+1)$ for $y(t+1)$ defined in eq. (0.45), the cost function J_S becomes

$$\begin{aligned} J_S &= E\left[\hat{y}(t+1\,|\,t) + e(t+1) - r(t)\right]^2 \\ &= E\left[\hat{y}(t+1\,|\,t)\right]^2 - 2E\left[\hat{y}(t+1\,|\,t)r(t)\right] + E\left[r(t)\right]^2 \\ &\quad + E\left[e(t+1)\right]^2 - 2E\left[e(t+1)r(t)\right] + 2E\left[\hat{y}(t+1\,|\,t)e(t+1)\right]. \end{aligned} \tag{0.72}$$

Since the noise $e(t+1)$ is independent of $r(t)$ and $\hat{y}(t+1\,|\,t)$ the last two terms of eq. (0.72) vanish. Note that the variance of the reference signal $E\left[r(t)\right]^2 = \sigma_r^2$ enters the cost function and increases its reachable minimal value. Defining the modified cost function $\tilde{J}_S$, by omitting terms that do not involve $u(t)$, leads to

$$\tilde{J}_S = 2b_0u(t)(-a_1y(t) - a_2y(t-1) + b_1u(t-1) - r(t)) + b_0^2u^2(t). \tag{0.73}$$

The minimisation of the modified cost function $\tilde{J}_S$ can be computed analytically by differentiating $\tilde{J}_S$ w.r.t. the argument and subsequently setting the derivative $\frac{\partial \tilde{J}_S}{\partial u(t)}$ to zero. So that differentiating gives

$$\frac{\partial \tilde{J}_S}{\partial u(t)} = 2b_0(-a_1y(t) - a_2y(t-1) + b_1u(t-1) - r(t)) + 2b_0^2u(t) \tag{0.74}$$

and setting to zero for a minimum yields

$$-a_1y(t) - a_2y(t-1) + b_1u(t-1) - r(t) + b_0u(t) = 0. \tag{0.75}$$

Rearranging to solve for $u(t)$ gives

$$u(t)=\frac{a_1 y(t)+a_2 y(t-1)-b_1 u(t-1)+r(t)}{b_0}. \tag{0.76}$$

□

In a similar manner to the regulator case, it is straightforward to show that the general form of the MV controller for an ARX model can be derived as

$$u(t)=\frac{1}{b_0}\left[-\sum_{i=0}^{n_m} m_{j,i}\, y(t-i)-\sum_{i=1}^{n_n} n_{j,i}\, u(t-i)+r(t)\right], \tag{0.77}$$

which may be directly compared to the regulator case given by eq. (0.69).

Simulation Study: MV Controller

Consider the system described by the ARX model given by

$$y(t)=1.5y(t-1)-0.7y(t-2)+0.7u(t-1)+0.3u(t-2)+e(t) \tag{0.78}$$

having the noise variance $\sigma_e^2=1$. The system runs in an open-loop setting during the time interval $t=\langle 1,25\rangle$ and in a closed-loop setting with the MV controller eq. (0.77) during the time interval $t=(25,100\rangle$. The reference signal switches between ± 5 units with a period of 25 samples. In order to assess the ability of the controller to track the reference signal the mean square error (MSE) criterion is introduced. The MSE is defined as

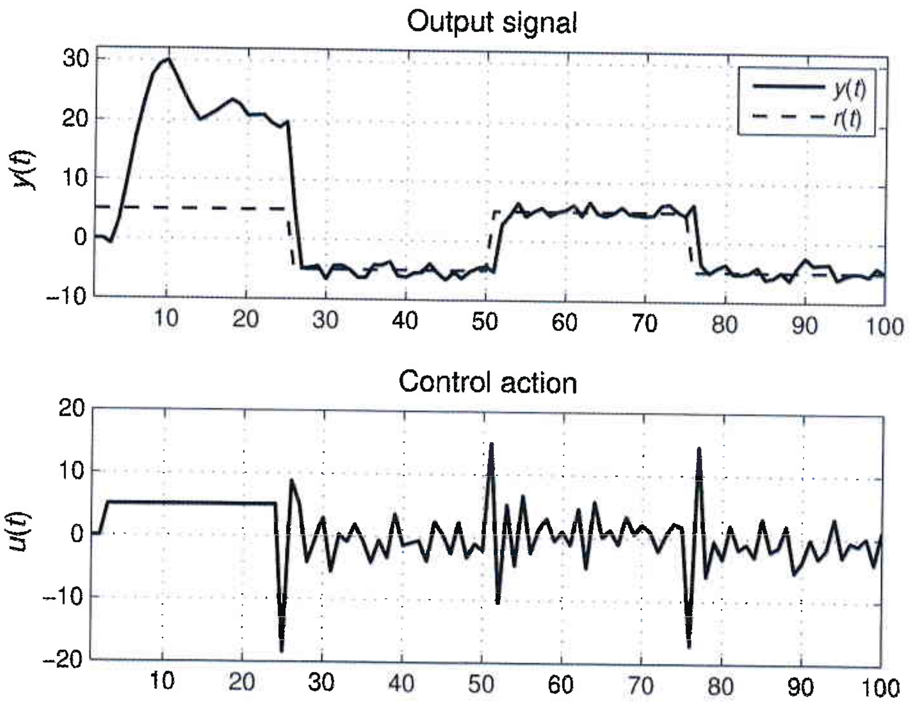
$$MSE=\frac{1}{N-t_0}\left[\sum_{t=t_0}^{N}(y(t)-r(t))^2\right], \tag{0.79}$$

where $N=100$ denotes the total number of discrete time steps and t_0 denotes the start of the evaluation. The mean square control (MSC) criterion is introduced in order to evaluate the usage of control effort, e.g. energy, and this is defined as

$$MSC=\frac{1}{N-t_0}\left[\sum_{t=t_0}^{N}u^2(t)\right]. \tag{0.80}$$

The results of simulation of the system together with the MV controller are shown in Figure 2. The performance in terms of MSE and MSC are $MSE = 4.22$ and $MSC = 19.46$, respectively, for $t_0 = 30$. It is evident that the MV algorithm achieves its control objectives during the closed-loop period.

Figure 2. Simulation of the MV controller for $t = \langle 1, 25 \rangle$ in the open-loop setting and for $t = (25, 100\rangle$ in the closed-loop setting.



GENERAL REMARKS ON THE MV CONTROLLER

The practical problems of choosing the sampling interval for the estimation of the model parameters are discussed here with connection to the MV controllers. Properties of the MV control are also discussed.

To illustrate some of the MV controller properties consider an ARX model structure having $n_a = 2$, $n_b = 1$ and $d = 1$. The discrete time transfer function for this system is

$$\frac{Y(q^{-1})}{U(q^{-1})} = \frac{q^{-1}(b_0 + b_1 q^{-1})}{1 + a_1 q^{-1} + a_2 q^{-2}}. \tag{0.81}$$

The MV control algorithm for this system is

$$u(t) = \frac{a_1 y(t) + a_2 y(t-1) - b_1 u(t-1)}{b_0} \tag{0.82}$$

or in transfer function form

$$\frac{U(q^{-1})}{Y(q^{-1})} = \frac{a_1 + a_2 q^{-1}}{b_0 + b_1 q^{-1}}. \tag{0.83}$$

Note that the denominator of the controller eq. (0.83) consists of the numerator of the system eq. (0.81), hence if the system zeros are outside the unit circle the controller poles become unstable. A system with the zeros outside the unit circle is known as a NMP system. This phenomenon occurs naturally or can be caused by inappropriate selection of the sampling interval τ_s; choice of τ_s is therefore crucial. It is recommended by the authors to choose τ_s such that $d \in <1,5>$, where 5 is considered to be high. In summary, since it is noted that MV controllers achieve their objectives by cancelling process zeros and that choice of τ_s too small can give rise to a NMP representation, it is important when dealing with MV control in practice to consider such factors.

The closed-loop transfer function for the given system and MV controller is

$$\frac{Y(q)}{R(q)} = \frac{a_1 q + a_2}{q^2} \tag{0.84}$$

hence the closed-loop poles lie at the origin and the response of such a system is as fast as possible; sometimes referred to as 'dead beat' response. This leads to excessively large demands on control action $u(t)$, which can be practically infeasible to realise.

GENERALISED MINIMUM VARIANCE CONTROLLER

The GMV controller has been proposed in order to overcome some of the issues connected with the concept of MV controllers. The issues are namely the excessive use of control effort in achieving the control objectives and the shortfall of controlling NMP systems. Introducing the cost function of the form

$$J_{GMV} = E\left[(Py(t+d) - Rr(t))^2 + (Qu(t))^2\right] \tag{0.85}$$

allows for a trade-off between tracking performance and cost of control. The scalars P, R and Q are user specified cost weighting parameters. Another formulation of the GMV cost function can be found in (Wellstead and Zarrop, 1991), where the cost weighting parameters are assumed to be polynomials. The cost weighting parameter Q is of particular importance, since for $Q > 0$ the control effort is constrained, having the effect of displacing the closed-loop poles away from the origin; i.e. no longer a deadbeat response and more practically realisable. The GMV controller is derived for the illustrative example.

Example 4. Consider a system described by an ARX model structure having $n_a = 2$, $n_b = 1$ and $d = 1$ expressed as a linear difference equation

$$y(t+1) = -a_1 y(t) - a_2 y(t-1) + b_0 u(t) + b_1 u(t-1) + e(t+1). \tag{0.86}$$

Separation of known (current and past) and unknown (future) information leads to

$$y(t+1) = \hat{y}(t+1\,|\,t) + e(t+1), \tag{0.87}$$

where

$$\hat{y}(t+1\,|\,t) = -a_1 y(t) - a_2 y(t-1) + b_0 u(t) + b_1 u(t-1). \tag{0.88}$$

The next step is to expand the quadratic cost function J_{GMV}, hence substituting eq. (0.87) into the cost function eq. (0.85) gives

$$\begin{aligned} J_{GMV} &= E\left[(P\hat{y}(t+1\,|\,t) + Pe(t+1) - Rr(t))^2 + (Qu(t))^2\right], \\ &= E\left[P\hat{y}(t+1\,|\,t)\right]^2 - E\left[2PR\hat{y}(t+1\,|\,t)r(t)\right] + E\left[Qu(t)\right]^2 \\ &\quad + E\left[Rr(t)\right]^2 + E\left[Re(t+1)\right]^2. \end{aligned} \tag{0.89}$$

The last two terms are weighted variances of $r(t)$ and $e(t+1)$, which forms the minimum achievable cost for J_{GMV}. Omitting terms which do not involve $u(t)$ leads to the modified cost function, which takes the form

$$\begin{aligned}\tilde{J}_{GMV} &= 2P^2 b_0 u(t)(-a_1 y(t) - a_2 y(t-2) + b_1 u(t-1)) \\ &+ (Pb_0 u(t))^2 - 2PRb_0 u(t) r(t) + (Qu(t))^2.\end{aligned} \tag{0.90}$$

Differentiation of the modified cost function $\tilde{J}_{GMV}$ w.r.t. $u(t)$ is computed as

$$\begin{aligned}\frac{\partial \tilde{J}_{GMV}}{\partial u(t)} &= 2P^2 b_0(-a_1 y(t) - a_2 y(t-2) + b_1 u(t-1)) \\ &+ 2(Pb_0)^2 u(t) - 2PRb_0 r(t) + 2Q^2 u(t)\end{aligned} \tag{0.91}$$

and setting to zero for a minimum

$$P^2 b_0(-a_1 y(t) - a_2 y(t-2) + b_1 u(t-1)) + P^2 b_0^2 u(t) - PRb_0 r(t) + Q^2 u(t) = 0 \tag{0.92}$$

leads to the GMV control algorithm

$$u(t) = \frac{Pb_0[P(a_1 y(t) + a_2 y(t-1) - b_1 u(t-1)) + Rr(t)]}{P^2 b_0^2 + Q^2}. \tag{0.93}$$

$\square$

The general form of GMV controller for an ARX model structure, which holds for any value of n_a, n_b and d can be derived by adopting the d-step ahead predictor

$$\hat{y}(t+j\,|\,t) = M_j(q^{-1}) y(t) + N_j(q^{-1}) u(t), \tag{0.94}$$

where $j = 1, \ldots, d$ and the polynomials $M_j(q^{-1})$ and $N_j(q^{-1})$ are defined in eq. (0.57) and eq. (0.58), respectively. The same procedure for obtaining the controller for the special case of $d = 1$ is followed, but with use made of the d-step ahead predictor eq. (0.94). The GMV control algorithm for an ARX model structure is then given by

$$u(t) = \left[P^2 n_{j,0}^2 + Q^2\right]^{-1} Pn_{j,0}\left[Rr(t) - P\sum_{i=0}^{n_m} m_{j,i} y(t-i) - P\sum_{i=1}^{n_n} n_{j,i} u(t-i)\right], \tag{0.95}$$

where $j=d$ and $n_{j,0}=b_0$. The GMV controller has advantageous over the MV scheme, but choice of the controller weighting P, R and Q is not immediately straightforward. For example too large a value for Q may result in the output not achieving the set point. Whilst there are ways to overcome this via careful choice of the other weightings, alternative incremental formulations offer immediate advantages.

INCREMENTAL GMV CONTROLLER

Recognising the difficulties in achieving a satisfactory trade-off via the cost weighting parameters and the additional potential problems due to the presence of non-zero mean output disturbances with the standard GMV scheme, prompted the need for an alternative approach and the incremental form of GMV (IGMV) was proposed. Such an approach guarantees a type-1 servo mechanism performance, hence a zero steady-state error is achieved for a constant reference signal. This is due to the inherent integral action within the IGMV scheme. To realise this scheme, the IGMV cost function is defined as

$$J_{IGMV}=E[y(t+d)-r(t)]^2+\lambda E[\Delta u(t)]^2, \qquad (0.96)$$

in which only a single weighting parameter λ is required. The derivation of the control algorithm is illustrated via an example.

Example 5. An ARIMAX model structure is used to derive the IGMV control algorithm. The model is given by

$$A(q^{-1})y(t)=q^{-d}B(q^{-1})u(t)+\frac{C(q^{-1})}{\Delta}e(t), \qquad (0.97)$$

where, for simplicity, the case of $C(q^{-1})=1$ is considered, hence yielding the ARIX model. Consideration is given to an example system in which $n_a=2$, $n_b=1$ and $d=1$. The model given by eq. (0.97) can be expressed as

$$(1+a_1q^{-1}+a_2q^{-2})\Delta y(t)=q^{-1}(b_0+b_1q^{-1})\Delta u(t)+e(t). \tag{0.98}$$

Defining the polynomial $\tilde{A}(q^{-1})=\Delta A(q^{-1})$ an expression for eq. (0.98) takes the form

$$(1+\tilde{a}_1q^{-1}+\tilde{a}_2q^{-2}+\tilde{a}_3q^{-3})y(t)=q^{-1}(b_0+b_1q^{-1})\Delta u(t)+e(t), \tag{0.99}$$

where

$$\tilde{a}_i=(a_i-a_{i-1}). \tag{0.100}$$

The aim is to determine $u(t)$, hence shifting the output d-steps forward leads to

$$y(t+1)=-\tilde{a}_1y(t)-\tilde{a}_2y(t-1)-\tilde{a}_3y(t-2)+b_0\Delta u(t)+b_1\Delta u(t-1)+e(t+1). \tag{0.101}$$

Assuming a zero mean Gaussian distributed white noise signal, the best prediction for $e(t+1)$ is zero. The predicted output at time $(t+d)$ is then expressed as

$$\hat{y}(t+1\,|\,t)=-\tilde{a}_1y(t)-\tilde{a}_2y(t-1)-\tilde{a}_3y(t-2)+b_0\Delta u(t)+b_1\Delta u(t-1) \tag{0.102}$$

and the system output at time $(t+1)$ can be re-expressed as

$$y(t+1)=\hat{y}(t+1\,|\,t)+e(t+1). \tag{0.103}$$

The next step in deriving of IGMV controller is the expansion of the quadratic cost function J_{IGMV}, hence substituting eq. (0.103) into the cost function J_{IGMV} gives

$$\begin{aligned} J_{IGMV}&=E[y(t+d)-r(t)]^2+\lambda E[\Delta u(t)]^2,\\ &=E[\hat{y}(t+1\,|\,t)]^2-2E[\hat{y}(t+1\,|\,t)r(t)]+\lambda E[\Delta u(t)]^2\\ &\quad+E[r(t)]^2+E[e(t+1)]^2. \end{aligned} \tag{0.104}$$

Defining the modified cost function $\tilde{J}_{IGMV}$, (omitting terms which do not involve $\Delta u(t)$) and exapanding leads to

$$\begin{aligned} \tilde{J}_{IGMV}&=2b_0\Delta u(t)(-\tilde{a}_1y(t)-\tilde{a}_2y(t-1)-\tilde{a}_3y(t-2)+b_1\Delta u(t-1)-r(t))\\ &\quad+b_0^2(\Delta u(t))^2+\lambda(\Delta u(t))^2. \end{aligned} \tag{0.105}$$

Differentiating w.r.t. the argument $\Delta u(t)$ and equating to zero for a minimum, leads to

$$\frac{\partial J_{IGMV}}{\partial \Delta u(t)} = 2b_0(-\tilde{a}_1 y(t) - \tilde{a}_2 y(t-1) - \tilde{a}_3 y(t-2) + b_1 \Delta u(t-1) - r(t)) \\ + 2b_0^2 \Delta u(t) + 2\lambda \Delta u(t) \tag{0.106}$$

and

$$b_0(-\tilde{a}_1 y(t) - \tilde{a}_2 y(t-1) - \tilde{a}_3 y(t-2) + b_1 \Delta u(t-1) - r(t)) + b_0^2 \Delta u(t) + \lambda \Delta u(t) = 0. \tag{0.107}$$

Rearranging the eq. (0.107) to solve for $\Delta u(t)$, the IGMV control algorithm is given by

$$\Delta u(t) = \frac{b_0(\tilde{a}_1 y(t) + \tilde{a}_2 y(t-1) + \tilde{a}_3 y(t-2) - b_1 \Delta u(t-1) + r(t))}{b_0^2 + \lambda}. \tag{0.108}$$

The applied control action to the plant is then computed as

$$u(t) = u(t-1) + \Delta u(t), \tag{0.109}$$

thus guaranteeing type-1 servo-mechanism performance.

□

The general form of the IGMV controller requires a d-step ahead predictor. In an ARIX case the predictor is derived in a similar manner to that for an ARX model structure eq. (0.55). The predictor for an ARIX model is given by

$$\hat{y}(t+j\,|\,t) = P_j(q^{-1}) y(t) + G_j(q^{-1}) \Delta u(t), \tag{0.110}$$

where $j = 1,\dots,d$ and the polynomials $P_j(q^{-1})$ and $G_j(q^{-1})$ are defined as

$$P_j(q^{-1}) = p_{j,0} + p_{j,1} q^{-1} + p_{j,2} q^{-2} + \dots + p_{j,i} q^{-i},\ i = n_{\tilde{a}} - 1 = n_p, \tag{0.111}$$

$$G_j(q^{-1}) = \sum_{l=0}^{j-1} (p_{l,0} q^{-l} \sum_{i=0}^{n_b} b_i q^{-i}),\ n_g = n_b + j - 1, \tag{0.112}$$

respectively, where the individual coefficients $p_{j,i}$ of the successive $P_j(q^{-1})$ polynomials are evaluated as follows

$$p_{j,i} = p_{j-1,i+1} + (a_i - a_{i+1}) p_{j-1,0},\ p_{0,0} = 1. \tag{0.113}$$

Note that the polynomial order n_g linearly increases as the number of predictions j increases. Minimising the cost function J_{IGMV} with respect to $\Delta u(t)$, utilising the d-step ahead predictor eq. (0.110) for an ARIX model structure leads to the IGMV controller

$$\Delta u(t) = [g_{j,0}^2 + \lambda]^{-1} g_{j,0} \left[r(t) - \sum_{i=1}^{n_g} g_{j,i} \Delta u(t-i) - \sum_{i=0}^{n_p} p_{j,i} y(t-i) \right], \tag{0.114}$$

where $j = d$ and $g_{j,0} = b_0$, with $u(t)$ finally being obtained as indicated in eq. (0.109).

General Remarks on the GMV Controller

The GMV controller is a natural extension of the MV controller. Whereby constraining the control effort of the MV controller the issues connected with NMP systems and excessive use of control action can be overcome. The choice of the cost weighting parameters is crucial and application specific. The P, R and Q parameters can be chosen either by an operator or adaptively (with some initial a priori values) within a STC framework. The former is discussed here. Setting $P = R = 1$ and $Q = 0$ results in MV control. Setting $P = R = 1$ and varying $Q > 0$ allows a trade-off between tracking ability and reduction of control effort. Hence, by over-constraining the control effort (e.g. energy) the GMV controller may not achieve the set point and steady-state errors occur. This can be overcome by retaining $P = 1$ and setting $R > 1$, which results in a new 'dummy' set point aim. Note that the importance of tracking ability versus reducing control cost is governed by the ratio $P : Q$ and not by their absolute values. The steady-state offset problems can also be overcome by using IGMV control, where inherent integral action guarantees type-1 performance. In addition only one tuning parameter λ is required. Note that choice of $\lambda = 0$ results in incremental MV control.

POLE PLACEMENT CONTROL

The next chronological development in the historic-technical review is that of self-tuning pole-placement (or pole-assignment) control (Wellstead et al., 1979). The aim of pole-placement control (PPC) is to match the closed-loop transient behaviour of a feedback system to a desired user prescribed form. Often referred to as eigenvalue assignment, the effect of PPC is that of relocation of the closed-loop poles of the system. The method is suitable for controller design where the performance criteria may be expressed in terms of the classical frequency or transient response. The approach has proven to be attractive to practising engineers, due probably to its close links with classical control. For the development of the PPC the system represented by noise free ARX model is considered

$$A(q^{-1})y(t) = q^{-d}B(q^{-1})u(t). \qquad (0.115)$$

The control law of the PPC is defined as

$$F(q^{-1})u(t) = G(q^{-1})y(t) + Mr(t), \qquad (0.116)$$

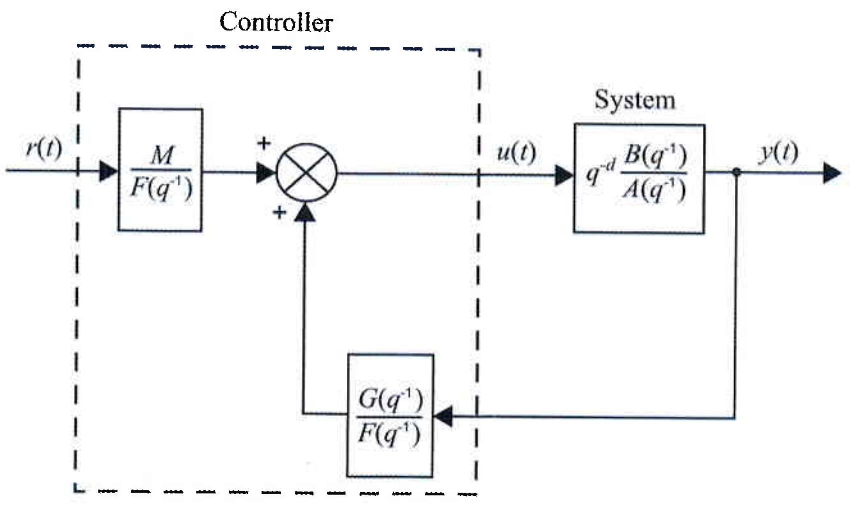
where the controller polynomials $F(q^{-1})$ and $G(q^{-1})$ are, respectively, defined as

$$F(q^{-1}) = f_0 + f_1 q^{-1} + f_2 q^{-2} + \cdots + f_{n_f} q^{-n_f}, f_0 = 1, \qquad (0.117)$$

$$G(q^{-1}) = g_0 + g_1 q^{-1} + g_2 q^{-2} + \cdots + g_{n_g} q^{-n_g}, g_0 \neq 0 \qquad (0.118)$$

having the corresponding recommended orders $n_f = n_b + d - 1$ and $n_g = n_a - 1$, respectively.

Figure 3. Pole-placement controller with compensator.


Controller
System
$r(t)$
$\frac{M}{F(q^{-1})}$
+
+
$u(t)$
$q^{-d}\frac{B(q^{-1})}{A(q^{-1})}$
$y(t)$
$\frac{G(q^{-1})}{F(q^{-1})}$

The system configured in closed-loop with the controller is depicted in Figure 3. The closed-loop transfer function is given by

$$\begin{aligned}\frac{Y(q^{-1})}{R(q^{-1})}&=\left[\frac{MB(q^{-1})}{F(q^{-1})A(q^{-1})}q^{-d}\right]\left[1-q^{-d}\frac{G(q^{-1})B(q^{-1})}{F(q^{-1})A(q^{-1})}\right]^{-1},\\&=\frac{q^{-d}B(q^{-1})M}{F(q^{-1})A(q^{-1})-q^{-d}G(q^{-1})B(q^{-1})}.\end{aligned}\qquad(0.119)$$

The aim is to assign the closed-loop poles to a specified location by equating the characteristic equation (denominator) of eq. (0.119) to a user specified design polynomial $\Gamma(q^{-1})$, i.e.

$$F(q^{-1})A(q^{-1})-q^{-d}G(q^{-1})B(q^{-1})=\Gamma(q^{-1}),\qquad(0.120)$$

where

$$\Gamma(q^{-1})=\gamma_0+\gamma_1 q^{-1}+\gamma_2 q^{-2}+\cdots+\gamma_{n_\gamma}q^{-n_\gamma},\gamma_0=1\qquad(0.121)$$

is the desired closed-loop characteristic polynomial having order $n_\gamma=n_a$. The controller polynomials $F(q^{-1})$ and $G(q^{-1})$ are related to model polynomials $A(q^{-1})$ and $B(q^{-1})$ via the Diophantine eq. (0.120). The desired transient response is designed through the polynomial $\Gamma(q^{-1})$, however by assigning poles for the closed-loop system the steady-state gain (SSG) will be affected. Making use of the final value theorem the closed-loop SSG is computed as

$$\text{SSG}=\left[q^{-d}\frac{B(q^{-1})M}{\Gamma(q^{-1})}\right]_{q^{-1}=1}=\frac{B(1)M}{\Gamma(1)}.\qquad(0.122)$$

The idea is to design the gain M such that $SSG=1$, hence the compensator for such a SSG is then

$$M=\frac{\Gamma(1)}{B(1)}.\qquad(0.123)$$

This approach, literally, cancels the offset due to $\Gamma(q^{-1})$ on the closed-loop SSG, so that, provided there is no model mismatch, the steady-state output match the reference signal $r(t)$.

Such a gain compensated PPC is then able to achieve both the transient response and desired steady-state gain simultaneously. The following illustrative example shows the design approach and the implementation of PPC.

Example 6. Consider the system having $n_a = 2$, $n_b = 1$ and $d = 1$ given by

$$(1-1.5q^{-1}+0.7q^{-2})y(t) = q^{-1}(0.7+0.3q^{-1})u(t)+e(t), \qquad (0.124)$$

where $e(t)$ is zero mean white Gaussian distributed measurement noise with variance $\sigma_e^2 = 0.5$. The open-loop poles of the system are $0.7500 \pm 0.3708\,i$, i.e. underdamped response. The aim is to achieve a critically damped response such that repeated closed-loop poles are defined to be 0.5 and 0.5, so that

$$\Gamma(q^{-1}) = 1.0000 - 1.0000q^{-1} + 0.2500q^{-2}. \qquad (0.125)$$

The Diophantine equation eq. (0.120) for $n_f = 1$ and $n_g = 1$ becomes

$$(1+f_1q^{-1})(1+a_1q^{-1}+a_2q^{-1}) - q^{-1}(g_0+g_1q^{-1})(b_0+b_1q^{-1}) = (1+\gamma_1q^{-1}+\gamma_2q^{-2}). \qquad (0.126)$$

By equating coefficients of like powers, the above expression may be reformulated in the convenient matrix form

$$\begin{bmatrix} 1 & -b_0 & 0 \\ a_1 & -b_1 & -b_1 \\ a_2 & 0 & -b_1 \end{bmatrix} \begin{bmatrix} f_1 \\ g_0 \\ g_1 \end{bmatrix} = \begin{bmatrix} \gamma_1 - a_1 \\ \gamma_2 - a_2 \\ 0 \end{bmatrix}. \qquad (0.127)$$

The unknown controller parameters may be computed directly from eq. (0.127) via matrix inversion or using Cramer's rule

$$\begin{aligned} f_1 &= b_1(b_1s_1 - b_0s_2)/\rho, \\ g_0 &= ((a_1b_1 - a_2b_0)s_1 - b_1s_2)/\rho, \\ g_1 &= a_2(b_1s_1 - b_0s_2)/\rho, \end{aligned} \qquad (0.128)$$

where

$$\begin{aligned}
\rho &= b_1^2 + a_2 b_0^2 - a_1 b_0 b_1, \\
s_1 &= \gamma_1 - a_1, \\
s_2 &= \gamma_2 - a_2.
\end{aligned} \tag{0.129}$$

Note that ρ is the determinant of the matrix in eq. (0.127). In order to compensate for the steady-state error, which occurs by relocating the original open-loop poles, the compensator M is introduced, i.e.

$$M = \frac{\Gamma(1)}{B(1)} = \frac{1+\gamma_1+\gamma_2}{b_0+b_1}. \tag{0.130}$$

The control action can then be determined from eq. (0.116), which may be expressed in the difference equation form as

$$u(t) = -f_1 u(t-1) + g_0 y(t) + g_1 y(t-1) + Mr(t). \tag{0.131}$$

Ω

The above pole placement controller has also been realised in state-space form utilising a minimal realisation representation, see (Warwick, 1981).

OUTLINE OF LONG RANGE PREDICTIVE CONTROL

The GMV and IGMV schemes are model-based d-step ahead predictive controllers. The accuracy of the prediction is closely related to the quality of the model. Not only the model parameters are required to be estimated, but also the integer valued normalised time delay of the system. If the estimated delay is less than the true system delay, then the controller attempts to generate large control action, which can destabilize the system. In the case of an overestimated delay the control is no longer optimal and the variance of the output signal may increase. The issues connected with the estimation of the delay or even a varying time delay of the system can be resolved by adopting a long range predictive control strategy. Instead of a d-step ahead prediction of the output, a prediction of the output up to a prediction horizon, denoted $H_p \geq d$, is performed, where H_p is a controller tuning parameter. Via long range prediction beyond the delay of the system and beyond the inverse response of NMP systems the control becomes stable and robust against model mismatch. One member of the class of long range predictive controllers, namely the GPC algorithm, will be covered here.

GENERALISED PREDICTIVE CONTROL

GPC has had a significant impact in terms of recent developments in control, as the currently widely adopted three term PID controller, when it become a popular choice as an industry standard. It is a popular model-based control method and is being used in industry. The approach was proposed and developed by Clarke *et al.* during the 1980's, see (Clarke et al., 1987). The idea of GPC is to minimise the variance of the future error between the output and set point by predicting the long range output of the system and separating the known contributions to future output from the unknown contributions. In this way a vector of future predicted errors can be used to generate a vector of future incremental controls. The aim is to minimise the GPC composite multi-stage quadratic cost function defined by

$$J_{GPC} = E\left[\sum_{j=d}^{H_p}[y(t+j)-r(t+j)]^2 + \sum_{j=1}^{H_c}\lambda[\Delta u(t+j-1)]^2\right] \tag{0.132}$$

with respect to current and future values of the incremental control action $\Delta u(t+j-1)$. The user specific tuning parameters are the prediction horizon, denoted $H_p \geq d$, the control horizon, denoted $H_c \geq 1$, and a cost weighting parameter λ. It is convenient here, during the derivation of the GPC algorithm, to consider the control horizon to be such that $H_c = H_p$; in practice however $H_c \leq H_p$. Note that, beyond H_c further incremental controls are assumed to be zero. The structure of the cost function for GPC can be seen as an extension of the cost function for IGMV, where the main difference is the idea of a long range receding horizon. Following this idea not only $y(t+d)$ is required to be predicted as in IGMV, but also the predictions $y(t+j)$, $j=d,\ldots,H_p$; with this concept providing a basic framework for long range predictive control. In the development of the GPC algorithm the ARIMAX model structure is considered

$$A(q^{-1})y(t)=q^{-d}B(q^{-1})u(t)+\frac{C(q^{-1})}{\Delta}e(t), \qquad (0.133)$$

where for simplicity $C(q^{-1})=1$, i.e. an ARIX model structure is assumed. The case of $C(q^{-1})>1$ is investigated in (Clarke and Mohtadi, 1989; Camacho and Bordons, 2004). The cost function J_{GPC} consists of future values of the reference signal $r(t+j)$, $j=d,\ldots,H_p$, which are assumed to be known in advance. Future values of the output are required to be predicted and future incremental values of the control action $\Delta u(t+j-1)$, $j=1,\ldots,H_c$, are yet to be determined. The following example illustrates the prediction of the output up to an horizon of $H_p=H_c=3$ steps.

Example 7. Consider a model having $n_a=2$, $n_b=1$ and $d=1$, hence

$$(1+a_1q^{-1}+a_2q^{-2})\Delta y(t)=q^{-1}(b_0+b_1q^{-1})\Delta u(t)+e(t) \qquad (0.134)$$

and defining the polynomial $\tilde{A}(q^{-1})=\Delta A(q^{-1})$ expression (0.134) becomes

$$(1+\tilde{a}_1q^{-1}+\tilde{a}_2q^{-2}+\tilde{a}_3q^{-3})y(t)=q^{-1}(b_0+b_1q^{-1})\Delta u(t)+e(t), \qquad (0.135)$$

where

$$\tilde{a}_i=(a_i-a_{i-1}). \qquad (0.136)$$

The output at time $(t+1)$ is then

$$y(t+1)=-\tilde{a}_1y(t)-\tilde{a}_2y(t-1)-\tilde{a}_3y(t-2)+b_0\Delta u(t)+b_1\Delta u(t-1)+e(t+1). \qquad (0.137)$$

Assuming zero mean white noise the prediction of $e(t+1)$ is null. The best prediction of the output in the sense of minimising the squared prediction error then becomes

$$\hat{y}(t+1\,|\,t) = -\tilde{a}_1 y(t) - \tilde{a}_2 y(t-1) - \tilde{a}_3 y(t-2) + b_0 \Delta u(t) + b_1 \Delta u(t-1)\,. \tag{0.138}$$

The prediction at time $(t+2)$ and $(t+3)$ is computed, respectively, as

$$\begin{aligned}\hat{y}(t+2\,|\,t) &= -\tilde{a}_1 \hat{y}(t+1\,|\,t) - \tilde{a}_2 y(t) - \tilde{a}_3 y(t-1) + b_0 \Delta u(t+1) + b_1 \Delta u(t) \\ &= -(\tilde{a}_2 - \tilde{a}_1\tilde{a}_1) y(t) - (\tilde{a}_3 - \tilde{a}_1\tilde{a}_2) y(t-1) - (0 - \tilde{a}_1\tilde{a}_3) y(t-2) \\ &\quad + b_0 \Delta u(t+1) + (b_1 - \tilde{a}_1 b_0)\Delta u(t) + (0 - \tilde{a}_1 b_1)\Delta u(t-1)\end{aligned} \tag{0.139}$$

and

$$\begin{aligned}\hat{y}(t+3\,|\,t) &= -\tilde{a}_1 \hat{y}(t+2\,|\,t) - \tilde{a}_2 \hat{y}(t+1\,|\,t) - \tilde{a}_3 y(t) + b_0 \Delta u(t+2) + b_1 \Delta u(t+1) \\ &= -(-\tilde{a}_1(\tilde{a}_2 - \tilde{a}_1\tilde{a}_1) - \tilde{a}_2\tilde{a}_1 + \tilde{a}_3) y(t) - (-\tilde{a}_1(\tilde{a}_3 - \tilde{a}_1\tilde{a}_2) - \tilde{a}_2\tilde{a}_2) y(t-1) \\ &\quad - (-\tilde{a}_1(0 - \tilde{a}_1\tilde{a}_3) - \tilde{a}_2\tilde{a}_3) y(t-2) + b_0 \Delta u(t+2) + (b_1 - \tilde{a}_1 b_0)\Delta u(t+1) \\ &\quad + (-\tilde{a}_1(b_1 - \tilde{a}_1 b_0) - \tilde{a}_2 b_0)\Delta u(t) + (-\tilde{a}_1(0 - \tilde{a}_1 b_1) - \tilde{a}_2 b_1)\Delta u(t-1).\end{aligned} \tag{0.140}$$

Note that $\Delta u(t),\ldots,\Delta u(t+j-1)$, $j=1,\ldots,H_c$, are unknown values of the future incremental control action, which are yet to be determined by minimisation of the multistage quadratic cost function J_{GPC}. Note that when $H_p = d$, IGMV is a special case of GPC where the only unknown is $\Delta u(t)$.

Ω

The predictor for an ARIMAX model structure, when considering the case of $C(q^{-1}) = 1$, can be computed as follows

$$\hat{y}(t+j\,|\,t) = P_j(q^{-1}) y(t) + G_j(q^{-1})\Delta u(t+j-1), \tag{0.141}$$

where $j=1,\ldots,H_p$ denotes the prediction and only last $j=d,\ldots,H_p$ values are used in the development of the GPC algorithm. The polynomials $P_j(q^{-1})$ and $G_j(q^{-1})$ are defined as

$$P_j(q^{-1}) = p_{j,0} + p_{j,1}q^{-1} + p_{j,2}q^{-2} + \ldots + p_{j,i}q^{-i},\ i = n_p = n_{\tilde{a}} - 1, \tag{0.142}$$

and

$$G_j(q^{-1}) = \sum_{l=0}^{j-1}(p_{l,0}q^{-l}\sum_{i=0}^{n_b} b_i q^{-i}),\ n_g = n_b + j - 1, \tag{0.143}$$

respectively, and where the individual coefficients $p_{j,i}$ of successive $P_j(q^{-1})$ polynomials can be computed as follows

$$p_{j,i} = p_{j-1,i+1} + (a_i - a_{i+1})p_{j-1,0}, \, p_{0,0} = 1. \tag{0.144}$$

Note, that the order of the $G_j(q^{-1})$ polynomial linearly increases as the number of the predictions j increases. The following illustrative example shows the prediction $H_p = H_c = 3$ utilising the predictor eq. (0.141).

Example 8. Consider a model having $n_a = 2$, $n_b = 1$ and $d = 1$. The prediction of the future outputs utilizing the predictor eq. (0.141) then becomes

$$\begin{aligned}
\hat{y}(t+1|t) &= p_{1,0}y(t) + p_{1,1}y(t-1) + p_{1,2}y(t-2) + g_{1,0}\Delta u(t) + g_{1,1}\Delta u(t-1), \\
\hat{y}(t+2|t) &= p_{2,0}y(t) + p_{2,1}y(t-1) + p_{2,2}y(t-2) + g_{2,0}\Delta u(t+1) + g_{2,1}\Delta u(t) \\
&\quad + g_{2,2}\Delta u(t-1), \\
\hat{y}(t+3|t) &= p_{3,0}y(t) + p_{3,1}y(t-1) + p_{3,2}y(t-2) + g_{3,0}\Delta u(t+2) + g_{3,1}\Delta u(t+1) \\
&\quad + g_{3,2}\Delta u(t) + g_{3,3}\Delta u(t-1).
\end{aligned} \tag{0.145}$$

The above predictions of the system output can be expressed in matrix form, where the known and unknown contributions to the predicted outputs are separated as follows

$$\begin{aligned}
\begin{bmatrix} \hat{y}(t+1|t) \\ \hat{y}(t+2|t) \\ \hat{y}(t+3|t) \end{bmatrix} &= \begin{bmatrix} p_{1,0} & p_{1,1} & p_{1,2} & g_{1,1} \\ p_{2,0} & p_{2,1} & p_{2,2} & g_{2,2} \\ p_{3,0} & p_{3,1} & p_{3,2} & g_{3,3} \end{bmatrix} \left.\begin{bmatrix} y(t) \\ y(t-1) \\ y(t-2) \\ \Delta u(t-1) \end{bmatrix}\right\} \text{known} \\
&\quad + \begin{bmatrix} g_{1,0} & 0 & 0 \\ g_{2,1} & g_{2,0} & 0 \\ g_{3,2} & g_{3,1} & g_{3,0} \end{bmatrix} \left.\begin{bmatrix} \Delta u(t) \\ \Delta u(t+1) \\ \Delta u(t+2) \end{bmatrix}\right\} \text{unknown}
\end{aligned} \tag{0.146}$$

hence transforming the derivation of the GPC algorithm into a straightforward problem involving matrix algebra.

Ω

In general, eq. (0.146) can be express as

$$\hat{\mathbf{y}} = \mathbf{f} + \mathbf{G}\mathbf{u} \tag{0.147}$$

where the vector of predicted outputs is given by

$$\hat{\mathbf{y}} = [\hat{y}(t+d\,|\,t), \hat{y}(t+d+1\,|\,t), \ldots, \hat{y}(t+H_p\,|\,t)]^T \tag{0.148}$$

and the vector of known contributions to $\hat{\mathbf{y}}$, which forms the free response of the system (Maciejowski, 2002), assuming zero incremental controls is given by

$$\mathbf{f} = \begin{bmatrix} P_d(q^{-1}) & \left(G_d(q^{-1}) - g_{d,0}\right)q \\ P_{d+1}(q^{-1}) & \left(G_{d+1}(q^{-1}) - g_{d+1,0} - g_{d+1,1}q^{-1}\right)q^2 \\ \vdots & \vdots \\ P_{d+H_p}(q^{-1}) & \left(G_{H_p}(q^{-1}) - g_{d+H_p,0} - \ldots - g_{d+H_p,H_p-1}q^{-(H_p-1)}\right)q^{H_p} \end{bmatrix} \begin{bmatrix} y(t) \\ \Delta u(t-1) \end{bmatrix}. \tag{0.149}$$

The Toeplitz lower triangular matrix $\mathbf{G}$ is defined as

$$\mathbf{G} = \begin{bmatrix} g_0 & 0 & \cdots & 0 \\ g_1 & g_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ g_{H_p-d} & g_{(H_p-d)-1} & \cdots & g_0 \end{bmatrix}, \tag{0.150}$$

where the leading j subscripts on the elements in $\mathbf{G}$ are omitted, since the diagonal (main and minor) elements are the same and not dependent on j. The vector of control actions, which is yet to be determined is given by

$$\mathbf{u} = [\Delta u(t), \Delta u(t+1), \ldots, \Delta u(t+H_p-d)]^T. \tag{0.151}$$

The cost function J_{GPC} can be expressed in the vector form as

$$J_{GPC} = (\hat{\mathbf{y}} - \mathbf{r})^T(\hat{\mathbf{y}} - \mathbf{r}) + \mathbf{u}^T \lambda \mathbf{u}, \tag{0.152}$$

where the vector of future set points (or reference signal) is defined as

$$\mathbf{r} = [r(t+d), r(t+d+1), \ldots, r(t+H_p)]^T. \tag{0.153}$$

The next step of the derivation of the GPC algorithm is to differentiate the cost function eq. (0.152) with respect to the vector of future incremental controls, i.e.

$$\begin{aligned}\frac{\partial J_{GPC}}{\partial \mathbf{u}} &= \left[(\hat{\mathbf{y}}-\mathbf{r})^T \frac{\partial}{\partial \mathbf{u}}(\hat{\mathbf{y}}-\mathbf{r})\right]^T + \left[\frac{\partial}{\partial \mathbf{u}}(\hat{\mathbf{y}}-\mathbf{r})\right]^T (\hat{\mathbf{y}}-\mathbf{r}) \\ &\quad + \left[\mathbf{u}^T \frac{\partial}{\partial \mathbf{u}} \lambda \mathbf{u}\right]^T + \left[\frac{\partial}{\partial \mathbf{u}} \mathbf{u}^T\right]^T \lambda \mathbf{u} \\ &= \left[(\hat{\mathbf{y}}-\mathbf{r})^T \mathbf{G}\right]^T + [\mathbf{G}]^T (\hat{\mathbf{y}}-\mathbf{r}) \\ &\quad + \left[\mathbf{u}^T \lambda\right]^T + [\mathbf{I}]^T \lambda \mathbf{u} \\ &= 2\mathbf{G}^T(\hat{\mathbf{y}}-\mathbf{r}) + 2\lambda \mathbf{u}\end{aligned} \qquad (0.154)$$

and substituting eq. (0.147) for the vector of predicted outputs $\hat{\mathbf{y}}$ leads to

$$\begin{aligned}\frac{\partial J_{GPC}}{\partial \mathbf{u}} &= 2\mathbf{G}^T(\mathbf{f}+\mathbf{G}\mathbf{u}-\mathbf{r}) + 2\lambda \mathbf{u} \\ &= 2\mathbf{G}^T(\mathbf{f}-\mathbf{r}) + 2(\mathbf{G}^T\mathbf{G}+\lambda \mathbf{I})\mathbf{u},\end{aligned} \qquad (0.155)$$

where $\mathbf{I}$ denotes an identity matrix of appropriate dimension. (In the case of $H_c = H_p$ it is of dimension $(H_p+1-d)\times(H_p+1-d)$.) The minimisation procedure is accomplished by setting $\frac{\partial J_{GPC}}{\partial \mathbf{u}} = 0$, hence

$$\mathbf{G}^T(\mathbf{f}-\mathbf{r}) + (\mathbf{G}^T\mathbf{G}+\lambda \mathbf{I})\mathbf{u} = 0. \qquad (0.156)$$

Rearranging the expression eq. (0.156) to solve for vector $\mathbf{u}$ leads to the GPC algorithm

$$\mathbf{u} = \left[\mathbf{G}^T\mathbf{G}+\lambda \mathbf{I}\right]^{-1} \mathbf{G}^T\left[\mathbf{r}-\mathbf{f}\right], \qquad (0.157)$$

where only the first term of the vector $\mathbf{u}$ is applied to the plant, hence

$$u(t) = u(t-1) + \Delta u(t). \qquad (0.158)$$

Throughout the derivation of the GPC algorithm the control horizon has been set such that $H_c = H_p$. However, the use of $H_c \le H_p$ is common in practice, which decreases the computational load. The control horizon is relatively simply implemented by reducing the dimension of the lower triangular matrix $\mathbf{G}$ by considering only the first H_c columns of $\mathbf{G}$ and the dimension of $\mathbf{u}$ is then $H_c \times 1$. The corresponding weighting matrix $\lambda \mathbf{I}$ is also required to be suitably truncated. The matrix inversion in eq. (0.157) for the special case of

$H_c = 1$, reduces to the division by a scalar, which is often used in practice due to ease of computation.

Choice of the Control and Prediction Horizons

The choice of the control and prediction horizons H_c and H_p is a crucial issue when implementing the GPC algorithm. The horizons act as tuning or design parameters and are application specific. The choice of these is rather difficult and only a basic introduction is stated here. A detailed discussion of choosing the horizons and the cost weighting parameter λ can be found in (Clarke and Mohtadi, 1989; Clarke, 1996). The prediction horizon should be large enough to incorporate the delay and transients of the system plus any possible NMP response. It is suggested that the prediction horizon should incorporate the rise time of the plant, in order to encompass the transient effects of the plant.

Numerical Study: GPC

Consider an ARX model structure having $n_a = 2$, $n_b = 1$ and $d = 1$ given by

$$y(t) = 1.5y(t-1) - 0.7y(t-2) + 0.7u(t-1) + 0.3u(t-2) + e(t). \qquad (0.159)$$

To illustrate the calculation of the $P_j(q^{-1})$ and $G_j(q^{-1})$ polynomials the first prediction for $j = 1$ is performed. Utilising the predictor eq. (0.141) the polynomial $P_1(q^{-1})$ for $n_p = n_{\tilde{a}} - 1 = 2$ is then

$$P_1(q^{-1}) = p_{1,0} + p_{1,1}q^{-1} + p_{1,2}q^{-1}, \qquad (0.160)$$

where the individual coefficients $p_{1,i}$, $i = 0\ldots2$, are computed utilising eq. (0.144) and eq. (0.136), hence

$$\begin{aligned} p_{1,0} &= p_{0,1} + (a_0 - a_1)p_{0,0} = 0 + (1 - a_1)1 = -\tilde{a}_1, \\ p_{1,1} &= p_{0,2} + (a_1 - a_2)p_{0,0} = 0 + (a_1 - a_2)1 = -\tilde{a}_2, \\ p_{1,2} &= p_{0,3} + (a_2 - a_3)p_{0,0} = 0 + (a_2 - 0)1 = -\tilde{a}_3. \end{aligned} \tag{0.161}$$

Utilisng eq. (0.143) the $G_1(q^{-1})$ polynomial is computed as

$$G_1 = b_0 + b_1 q^{-1}. \tag{0.162}$$

The predicted output at time $(t+1)$ is then

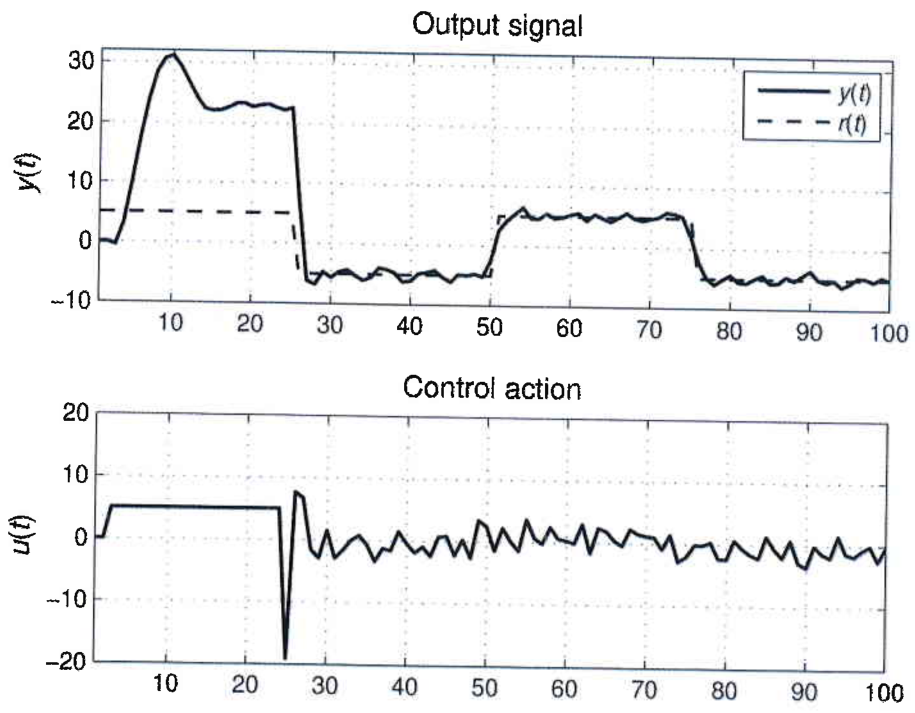
$$\hat{y}(t+1\,|\,t) = -\tilde{a}_1 y(t) - \tilde{a}_2 y(t-1) - \tilde{a}_3 y(t-2) + b_0 \Delta u(t) + b_1 \Delta u(t-1), \tag{0.163}$$

which is exactly the same solution as in eq. (0.138). Following the same procedure the prediction for $j=2$ and $j=3$ can be computed.

The simulation setup, as previously, involves the open-loop operation during the time interval $t = \langle 1, 25\rangle$ and closed-loop operation with the GPC controller eq. (0.157) during the time interval $t = (25, 100\rangle$. The reference signal switches between ± 5 units with a period of 25 samples. The performance criteria are the same as used previously, see eq. (0.79) and eq. (0.80). The noise variance is assumed to be $\sigma_e^2 = 0.5$ and the start of the performance evaluation is taken to be $t_0 = 30$. The horizons are chosen as $H_p = 3$ and $H_c = 2$ and the cost weighting parameter $\lambda = 0.1$.

The results of the simulation are shown in Figure 4. The performance in terms of the MSE and MSC criteria are $MSE = 0.77$ and $MSC = 3.15$, respectively, which in comparison to the MV performance, is a superior result. Note that the first change of the system output starts before the actual reference signal changes. Indeed this is one of the advantages of GPC over alternative conventional control strategies.

Figure 4. Simulation of the GPC controller for $t = \langle 1, 25 \rangle$ *in the open-loop setting and for* $t = (25, 100\rangle$ *in the closed-loop setting.*



A BILINEAR APPROACH TO STC FOR NONLINEAR INDUSTRIAL SYSTEMS

Recognition that real-world nonlinear systems exhibit different behaviour over the operating range, and locally linearised models are valid only for small regions about a single operating point, has prompted the desire to extend the STC concept to encompass a wider range of nonlinear systems. Since bilinear systems represent a small, but important subset of nonlinear systems within which linear systems coexist as a special subclass, attention is focused here on extensions of STC for bilinear systems. A diagrammatic representation of linear, bilinear and nonlinear systems is shown in Figure 5. Indeed many real-world processes can be more appropriately described using bilinear models, and a good summary can be found in (Mohler, 1970; Bruni et al., 1974; Espana and Landau, 1978; Figalli et al., 1984). Bilinear systems are characterised by linear behaviour in both state and control when considered separately, with the nonlinearity arising as a product of system state and control

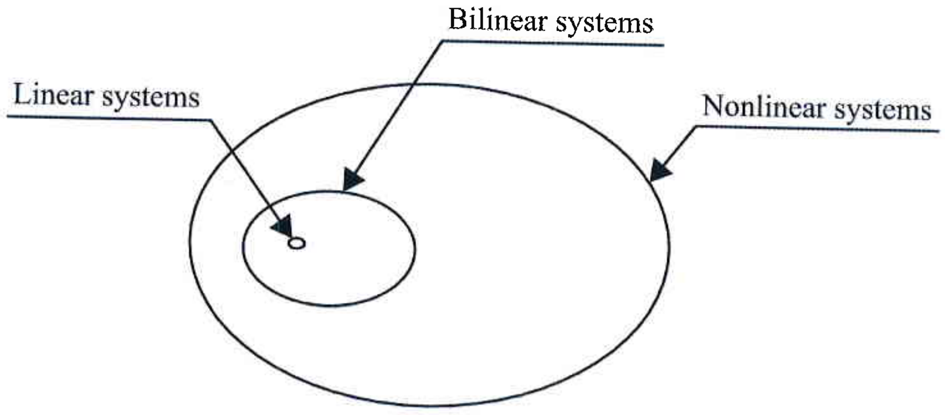
(Mohler, 1973). These processes may be found in areas such as engineering, ecology, medicine and socioeconomics. Thus the adoption of bilinear models, hence the development of bilinear model-based control, represents a significant step towards dealing with practical real-world systems.

Based on the above observations coupled with the potential advantages of improved control, e.g. improved efficiency, reduced wastage, increased profitability and improved product quality, the need to develop bilinear model-based control strategies is justified. Indeed, this has formed the topic of much research, with potential benefits, in practical applications, see (Burnham, 1991; Goodhart, 1991; Disdell, 1995; Dunoyer, 1996; Minihan, 2001; Ziemian, 2002; Martineau, 2004). This concept of adoption of the bilinear model-based approach is demonstrated by extending the linear GPC scheme to the bilinear case. The use of bilinear GPC (BGPC) increases the operational range of the controller over the use of the linear model-based GPC when applied to systems for which a bilinear model is more appropriate. A general single-input single-output bilinear system can be modelled using a nonlinear ARMAX (NARMAX) model representation, i.e.

$$\begin{aligned} y(t) &= \sum_{i=1}^{n_a} -a_i y(t-i) + \sum_{i=0}^{n_b} b_i u(t-d-i) \\ &\quad + \sum_{i=0}^{n_b}\sum_{j=1}^{n_a} \eta_{i,j} y(t-i-d) u(t-i-j-d+1) + \xi(t), \end{aligned} \tag{0.164}$$

where the a_i and b_i are assumed to correspond to the linear ARMAX model eq. (0.1) with the $\eta_{i,j}$ being the discrete bilinear coefficients which are required to be identified either on-line or off-line along with the a_i and b_i (Dunoyer, 1996).

Figure 5. Diagrammatic representation of bilinear systems as a subset of the wider class of nonlinear systems, and linear systems as a subclass of bilinear systems.



BILINEAR GPC

The predictive control law is based on a bilinear model eq. (0.164), which for the purpose of obtaining an explicit solution to the multi stage quadratic cost function eq. (0.132) is interpreted as a time-step quasi-linear model such that the bilinear coefficients are combined with either the a_i or b_i parameters. The combined parameters are either given by

$$\tilde{a}_i(t) = a_i - u(t-d-i)\eta(i-1) \qquad (0.165)$$

or by

$$\tilde{b}_i(t) = b_i + y(t-i)\eta(i). \qquad (0.166)$$

For example, by recombining the bilinear terms with the a_i coefficinets the bilinear model eq. (0.164) can be expressed as input dependent and potentially time varying linear model, i.e.

$$y(t) = \sum_{i=1}^{n_a} -\tilde{a}_i y(t-i) + \sum_{i=0}^{n_b} b_i u(t-d-i). \qquad (0.167)$$

The decision to accommodate the bilinearity with the a_i or b_i coefficients depends on a particular control situation and, to some extent, user choice. Since the vector of future incremental control actions eq. (0.151) is computed at each time instance this knowledge can be utilised during the cost function minimisation. For example, one can obtain the predictions of the future outputs by utilising the combination approach of eq. (0.165) with the most recent solution for the vector of incremental controls $\mathbf{u}$. Subsequently it may be advantageous to compute the next vector of incremental controls utilisng the combination approach of eq. (0.166). This latter approach of cyclic recombination of the bilinear terms has been shown to give rise to an improved overall performance (Dunoyer, 1996).

As a consequence of utilising the bilinear (bilinearised) model for the purpose of predicting the system output the prediction error decreases, hence the BGPC is more effective over the standard GPC. The BGPC algorithm retains the same structure as in the case of GPC eq. (0.157). However, since the $\tilde{a}_i(t)$ or $\tilde{b}_i(t)$ coefficients are potentially time varying and input or output dependent, respectively, the Toeplitz lower triangular matrix $\mathbf{G}$ and vector $\mathbf{f}$, which comprise of these coefficients, are required to be updated at each time step. Note that some of the complexity can be overcome by taking advantage of the common factors in the case when $H_c = 1$ (Vinsonneau, 2007). In general, however, the use of the BGPC leads to a higher computational load over the standard GPC.

Numerical Study: GPC, Self-tuning GPC, and BGPC

The system (plant) is represented by a second order single-input single-output ARX model having additional Hammerstein and bilinear nonlinearities. Similar structured nonlinear models have been assumed previously for replicating the characteristics of high temperature industrial furnaces, see (Goodhart et al., 1994; Dunoyer et al., 1997; Martineau et al., 2004), or for representing the thermodynamic processes within a heating ventilation and air conditioning system, see (Larkowski et al., 2009; Zajic et al., 2009). The nonlinear system has been chosen to show that bilinear controllers can be used to control nonlinear systems without using the adaptive control approach, leading to the use of less complex and robust controllers. The system takes the form