ADAPTIVE CONTROL OF FLEXIBLE SYSTEMS

by

Martin Lambert
St. John's College,
Oxford.

"Gratum anus rodentum"

Anon.

Adaptive Control of Flexible Systems

Martin Lambert

St. John's College, Oxford.

Submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy at the University of Oxford. Trinity 1987.

ABSTRACT

This thesis reports the successful application of the recently introduced Generalised Predictive Control self-tuner to the high-performance positioning of a real flexible single-link robot arm.

The large amount of experimental time available on this high bandwidth system allowed exhaustive testing of the 'tuning-knobs' and 'design-filters' available to the user for tailoring the closed-loop. Based upon these experiments a coherent philosophy for configuring GPC in practice is generated.

Configuration details found to be necessary for satisfactory GPC control of this high-order neutrally stable and non-minimum-phase plant, with its lightly damped resonant modes, are isolated. In particular it is found that band-pass filtering of data is essential for stable offset-free control using finite-order models of the plant. These aspects are considered in detail both theoretically and experimentally.

In this application, as is often the case in practice, some information about the plant dynamics is available beforehand. Novel methods for the inclusion of this prior knowledge are introduced and their beneficial effects on the convergence of the recursive least squares estimation scheme, upon which most self-tuners are based, are demonstrated in simulation and experiment.

A novel high-speed 68010/20 multi-processor computer system is described which allows the implementation of GPC at the required high sample rate (60Hz). The software division of the self-tuning algorithm into concurrently and sequentially executing tasks is discussed in detail.

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my supervisor D.W. Clarke for providing the funds for the hardware developed during this course of study and for correcting this thesis.

Particular thanks go to Ron Daniel for educating me in the courageous approach to debugging computer hardware, to Mike Irving for teaching me not to interrupt and to Coorous Mohtadi for his endless enthusiasm and encyclopediac knowledge of references. Thanks are also due to Eugene Lambert for joining with me to form half of the most successful Varsity Pistol team on record.

Additional thanks go to Antony Fraser, Yousef al Assaf, Terence Tsang, Dan MacMichael, Dave Peel, Dario Boriani and Smith Associates who gave me a job (and Oliver Jacobs who convinced me that I should take it).

Final thanks to Victoria for her patient proof-reading and to my parents for making it all possible.

This thesis was funded by the U.K. Science and Engineering Research Council.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

Self-tuning control can either be considered as the automation of the identification, specification and implementation phases of conventional off-line control design (Kalman, 1958) or as the principal alternative to the robust strategies (Zames, 1981) for dealing with uncertainty in the dynamics of the plant to be controlled. Whatever the case it is apparent that extensive experimental studies are required both to vindicate existing theoretical developments and to argue effectively the case for a more widespread acceptance of self-tuning control in industry. The latter has often been held back by a lack of effective 'jacketting' (Clarke, 1981) to ensure integrity in the face of practical problems not encountered in simulation. An example of this is the common absence from many self-tuning controllers of even the simplest signal conditioning.

This thesis chiefly concerns experimental trials of an advanced self-tuning algorithm, Generalised Predictive Control (GPC), on a compliant single-link robot arm. The availability of some prior knowledge from analytic modelling is used to investigate the effect its incorporation has on the convergence properties of the GPC self-tuner. While self-tuning has traditionally been tested on stable and sluggish chemical process plant the neutrally stable and NMP* compliant arm with its lightly damped resonant modes provides an excellent illustration that this need not be a restriction.

Before proceeding further this chapter reviews the fields of self-tuning control, the main historical developments which lead to GPC and compliancy as a control problem involving lightly damped resonances.

* Non-minimum phase.

1.1 Self-Tuning Control

The basic elements of self-tuning control are discussed below. Sections 1.1.1-1.1.3 consider the models used to describe plant behaviour and discuss aspects of their identification; the choice of the order of a model to be fitted to plant input-output data and the least-squares parameter estimation algorithm commonly used. Subsidiary sections consider the modelling errors which arise during identification and means of retaining estimator adaptivity to cope with time-varying dynamics. Section 1.1.4 charts the evolutionary route which lead to GPC while section 1.1.5 outlines the aims, advantages and disadvantages of self-tuning control.

1.1.1 The Process Model

It is conventional to assume that the underlying continuous-time system to be controlled has dynamics governed by linear time-invariant differential equations eg:

$$T(t) = Jd^2\Theta(t)/dt^2 \qquad (1\text{-}1)$$

where T is a torque input to some inertia J undergoing an angular deflection Θ. The immediate objection that real plant are invariably non-linear is overcome by restricting the model to hold in some limited region about an 'operating point' with the model (1-1) being a local linearisation.

Conventional continuous-time design uses the Laplace transform to enable analysis to be performed in the polynomial domain eg:

$$T(s) = Js^2\Theta(s) \qquad (1\text{-}2)$$

Controllers designed in this way (eg. Gawthrop, 1987), expressed as transfer functions in s, can be implemented in discrete-time by sampling rapidly and making some Euler-type approximation to the differential operator ie. $s=(1-z^{-1})/h^{*}$. Alternatively design can be performed directly in discrete-time by using the z-transform (Ragazzini & Franklin, 1958) which is the discrete-time equivalent of the Laplace transform, operating on sequences rather than continuous functions of time.

The discrete-time approach, perhaps due to the explosion in digital computation, is overwhelmingly popular in the field of self-tuning control and is used throughout this thesis. In discrete-time the linear differential equations become linear difference equations:

$$y(t)+a_1y(t-1)+\ldots+a_ny(t-n) = b_0u(t-k-1)+\ldots+b_{n-1}u(t-k-n-2)+\eta(t)$$

where y(t) is now the sampled output variable and u(t) is the sampled input to the system. The associated polynomial equation in the z-transform is

$$A(z^{-1})y(t) = B(z^{-1})u(t-k-1) + \eta(t) \qquad (1\text{-}3)$$

where

$$A(z^{-1}) = 1 + a_1z^{-1} + \ldots + a_nz^{-n}$$
$$B(z^{-1}) = b_0 + b_1z^{-1} + \ldots + b_{n-1}z^{-n+1}$$

and k+1 is the plant delay, always at least unity due to the zero-order-hold element of a discrete-time controller. The term $\eta(t)$ represents any unmeasurable disturbances affecting the process output and non-linear effects arising from any significant move away from the operating-point.

The CARMA (Astrom, 1970) and CARIMA (Tuffs and Clarke, 1985) discrete-time models prevalent in the literature are both based upon attempts

* h is the sample interval.

to find a reasonable statistical description for the disturbance term $\eta(t)$. The Controlled-Auto-Regressive-Moving-Average model represents $\eta(t)$ as

$$A(z^{-1})y(t) = B(z^{-1})u(t-k-1) + C(z^{-1})\xi(t) \qquad (1\text{-}4)$$

where $C(z^{-1})$ is a stable polynomial (Astrom, 1970) operating upon a zero-mean uncorrelated random sequence, $\xi(t)$. The Controlled-Auto-Regressive-Integrated-Moving-Average model is a recent improvement (Belanger, 1983) where

$$A(z^{-1})y(t) = B(z^{-1})u(t-k-1) + \frac{C(z^{-1})\xi(t)}{\Delta(z^{-1})} \qquad (1\text{-}5)$$

Since $\xi(t)$ can represent white noise or random amplitude pulses occurring at random intervals the integrated disturbance $C(z^{-1})\xi(t)/\Delta(z^{-1})$ may be Brownian motion (Peterka, 1984) or step-like in nature. As well as being a more realistic model for industrial disturbances the non-zero mean of this integrated disturbance forces an integrator (Tuffs, 1984) into the structure of controllers designed on the basis of (1-5). This property is an advantage of CARIMA over CARMA models and is a straightforward consequence of the internal model principle (Francis & Wonham, 1976).

Finally note that the schism between continuous and discrete-time design has recently been narrowed by the δ-transform (described in Chapter 3) which, for rapid sampling, places both methodologies on a common footing (Goodwin et al, 1986).

1.1.2 Identification: Choice of model order

Assuming that the $A(z^{-1})$ and $B(z^{-1})$ polynomials of the CARMA model are available, the GPC control law (see chapter 2) is based upon using the deterministic part of the model ie.

$$A(z^{-1})y(t) = B(z^{-1})u(t-k-1) \qquad (1\text{-}6)$$

to make predictions of future plant behaviour. It is also possible to incorporate $C(z^{-1})$, if known, into the control calculation (Clarke et al, 1987). However the difficulties involved in obtaining this polynomial (Tuffs, 1984) allied with the efficacy of alternative ways of dealing with disturbances (ie. $T(z^{-1})$ in section 2.6.3) mean that $C(z^{-1})$ is rarely used in practice. Its main use is for a realistic discrete-time simulation of the type of disturbance encountered in practice.

Unfortunately the underlying continuous-time plant, and therefore the discrete-time model (1-3), is seldom exactly known beforehand. The **identification** phase of control involves fitting a model of the form (1-6) to the observed behaviour of the continuous-time system. Identification procedures, although they can take many forms, may loosely be grouped into three main approaches:

° Prior physical/mathematical modelling of the continuous-time system. This model will have a unique discrete-time equivalent of the form (1-3) which may be calculated by a standard procedure (Jury, 1964).

° Off-line identification using some fitting criterion (eg. least squares) with a model of the form (1-6) regressed on a batch of data generated by a variety of test signals such as steps or PRBS*.

° On-line identification where a recursive algorithm (eg. recursive least squares) fits a model of the form (1-6) to running i/o data. If these

* Pseudo Random Binary Sequence.

estimates are employed in the on-line redesign of the controller which is itself generating the running i/o data then this is known as self-tuning and is the context in which GPC was first introduced (Clarke et al, 1985).

Since the first option can be virtually impossible and is often economically unjustifiable it is not considered further. This leaves the latter two approaches which are united by the fact that the order of the model to be fitted to the i/o data must be chosen. This is because the exact orders of $A(z^{-1})$ and $B(z^{-1})$ and the delay k are also usually unknown. If the estimated model is written as

$$\hat{A}(z^{-1})y(t) = \hat{B}(z^{-1})u(t-k-1) + x(t) \qquad (1-7)$$

the choice of the orders of $\hat{A}$ and $\hat{B}$ and the time delay k involves balancing the following factors:

° The overriding criterion of suitability for the model is simply that it must accurately describe the important behaviour of the actual plant (Wellstead & Zanker, 1982) ie. x(t) is 'small' in some sense.

° It is an unfortunate fact that most real plant exhibit behaviour which can only be exactly modelled by very high order models (Seborg et al, 1986).

° Computational effort and the order of the resulting controller and, in most cases, the difficulty of identification rise with the assumed order of the model (Ljung & Soderstrom, 1983).

Consequently the choice of model order is a classical engineering trade-off, difficult to quantify in meaningful terms and owing much to experience. While analytic procedures such as Product Moment testing are available (Wellstead, 1978; Wahab, 1984) the order of the dominant dynamics

of the plant are usually apparent from previous running data or from simple prior tests such as pulse or step responses. For example, the pulse response of Fig.1.1 is predominantly that of a system with an overall time delay of three samples, an integrator and a first-order lag. Consequently second-order $\hat{A}$ and $\hat{B}$ polynomials would be estimated with the assumed delay, neglecting the unit delay from the ZOH ie. k in (1-7), set to two. Note that to allow for fractional time delay (Clarke, 1981), say due to the controller calculation taking up a significant slice of the sample interval, the order of $B(z^{-1})$ is usually increased by one from (1-3) to $\delta B=\delta A=n$.

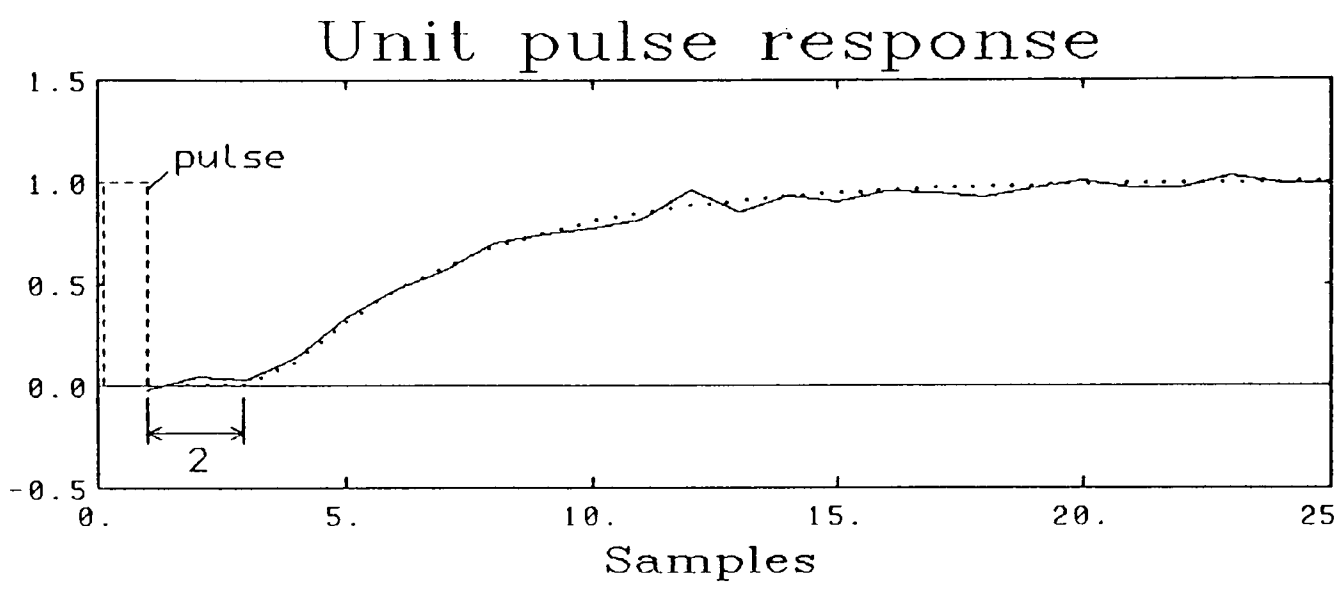


Fig.1.1: A 'typical' pulse response

In some cases the order of the dominant dynamics will not be immediately apparent, and the time delay especially can often be difficult to gauge. In these cases the order of the denominator polynomial $\hat{A}(z^{-1})$ may be over-estimated to ensure that possibly high-order dynamics can be modelled. More commonly the order of $\hat{B}(z^{-1})$ is over-estimated to cope with uncertainty in the time delay. Unlike many previous algorithms GPC has been shown to operate successfully with over-parameterised models and the possibility of common factors (Mohtadi, 1986).

The disturbance term x(t)

Since the GPC control strategy is model-based, using the estimated model (1-7) to predict future plant behaviour, the term x(t) which represents the lumped differences between model and actual plant behaviour (see Fig.1.2) is expected to have a marked influence on the resulting control. This is indeed the case, as will be seen in section 2.6.3 and the experimental results of chapter 5.

The term x(t) in the estimated model (1-7) is an error term principally incorporating

- the unmeasurable disturbances and non-linear effects acting upon the actual plant and represented by the noise term $\eta(t)$ in (1-3).
- the modelling errors involved in attempting to describe high-order actuality (1-3) by a probably lower-order approximation ie. plant-model-mismatch through neglected dynamics.
- the estimation errors caused by finite amounts of data corrupted by unmeasurable disturbances, sensor noise etc.

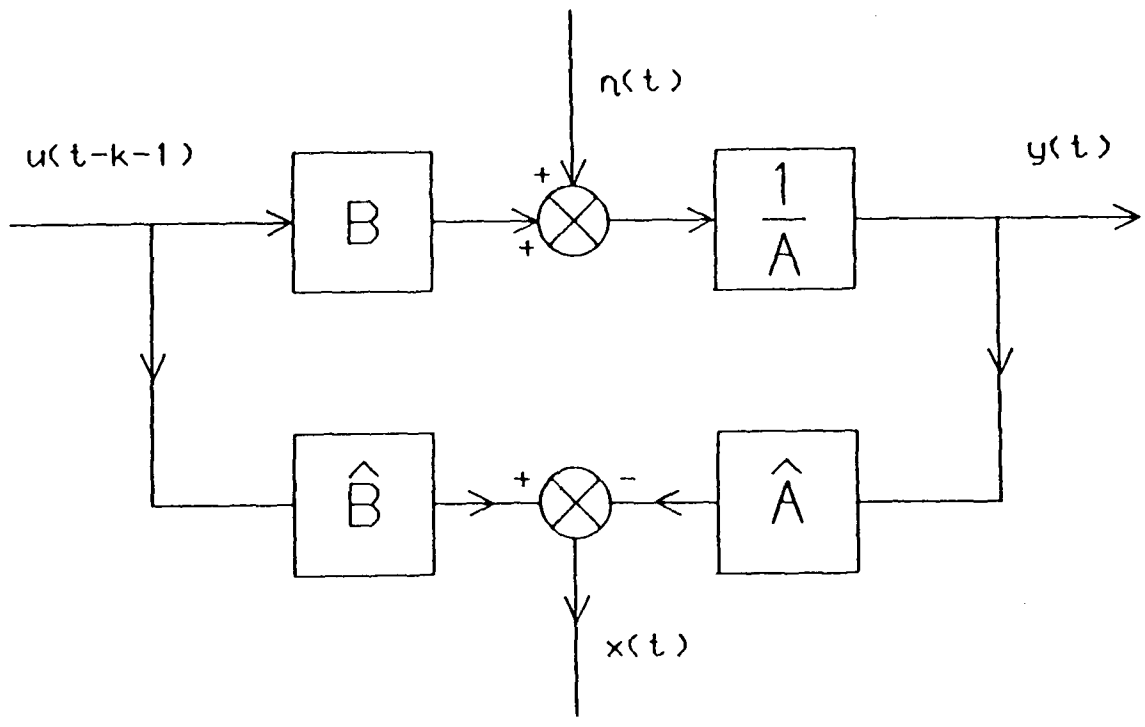


Fig.1.2: The disturbance term x(t)

This list provides a good indication of the complex nature of the term x(t). For example, large model errors at high frequencies only make a

significant contribution to x(t) if the control input u(t) has significant components at these high frequencies; friction only makes a significant effect when control amplitudes are 'small'.

However this description of x(t) perhaps paints too black a picture for when this sub-optimal model is used in the control calculation a variety of tuning-knobs and design polynomials (sections 2.4.1 and 2.6) are available to ameliorate its effect. Rather than striving for higher-order and more accurate models current research is aimed at the problem of fitting low-order models without degrading the eventual control performance (see sections 2.6.3 and 5.2.1).

1.1.3 Identification: Parameter Estimation

The model to be estimated (1-7) may be rewritten in a linear-in-the-parameters form

$$y(t) = \hat{\theta}^T(t)\phi(t) + x(t) \qquad (1\text{-}8)$$

where $\hat{\theta} = [\ -\hat{a}_1\ -\hat{a}_2\ \ldots\ -\hat{a}_n\ \ \hat{b}_0\ \ \hat{b}_1\ \ldots\ \ \hat{b}_{n-1}\ ]^T$

and $\phi(t) = [\ y(t-1)\ \ldots\ y(t-n)\ u(t-k-1)\ \ldots\ u(t-k-n-2)\ ]^T$

Estimation of the parameter vector $\hat{\theta}(t)$, off-line or on-line, given running i/o data up to time t is commonly achieved by the Recursive Least Squares algorithm (Clarke, 1981) where the estimates at time t, $\hat{\theta}(t)$, are chosen to minimise the quadratic cost functional

$$J = \sum_{i=0}^{t} \Lambda(i)[\ y(i)-\hat{\theta}(t)^T\phi(i)\ ]^2 + [\hat{\theta}(t)-\hat{\theta}(0)]^T P(0)^{-1}[\hat{\theta}(t)-\hat{\theta}(0)] \qquad (1\text{-}9)$$

ie. to make the error term x(t) in (1-8) small in the sum-of-squares sense. The new estimates at time t, $\hat{\theta}(t)$, are obtained from the recursion

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)[y(t)-\hat{\theta}(t-1)^T\phi(t)] \qquad (1\text{-}10)$$

$$K(t) = \frac{P(t-1)\phi(t)}{\beta(t) + \phi(t)^T P(t-1)\phi(t)} \qquad (1\text{-}11)$$

$$P(t) = [I - K(t)\phi^T(t)]\frac{P(t-1)}{\beta(t)} \qquad (1\text{-}12)$$

where, if the vectors θ and ϕ have N elements, the covariance matrix P(t) is NxN and the Kalman gain vector is Nx1. The positive weightings $\Lambda(i)$ in the cost are governed by the choice of the exponential forgetting factor, $\beta(t)$:

$$\Lambda(i) = \prod_{j=1}^{t-i} \beta(t-j) \qquad i \neq t \qquad (1\text{-}13)$$

$$= 1 \qquad i=t$$

By appropriate choice of $\beta(t)$ the weighting $\Lambda(i)$ can be used to attach greater or lesser importance to particular data sets $\{y(i),\phi(i)\}$. The second term in the cost, outside the summation, represents the initial conditions for the recursive algorithm.

The RLS algorithm is used for its speed of convergence, ease of implementation (Seborg et al, 1986) and numerical stability, the latter being ensured by mechanising (1-10) to (1-12) using the UD factorisation of Bierman (1977). A large body of research (eg. Fortescue et al, 1981; Hagglund, 1983; Ydstie et al, 1985) has been devoted to devising choices for the forgetting factor $\beta(t)$ to allow continued adaptivity without overdue sensitivity to transient disturbances and without catastrophic numerical effects such as 'covariance blowup' (Hodgson, 1982).

° Conventional RLS, where $\beta(t)=1$, loses adaptivity with time as $P(t)\rightarrow 0$ and $K(t)\rightarrow 0$ (Clarke, 1981).

° Fixed-forgetting RLS, where $\beta(t)=\kappa$ $0<\kappa\leq 1$, effectively only fits the model (1-8) over the last $1/(1-\kappa)$ data sets (the 'asymptotic sample length'). This retains adaptivity but suffers from 'blowup' and 'bursting' where, during periods of low data, the covariance matrix P(t) increases exponentially. When some new data arrives this can cause excessive parameter motion which in turn can cause a destabilising control transient known as 'bursting'.

° Tuffs' variable forgetting factor attempts to overcome these problems by choosing

$$\beta(t) = 1 - \frac{\phi(t)^T P(t-1)\phi(t)}{1 + \phi(t)^T P(t-1)\phi(t)} \qquad (1\text{-}14)$$

which aims to keep constant a scalar measure of the information reaching the estimator (Tuffs, 1984). The forgetting factor can only lie in the range $0\rightarrow 1$, being 1 when $\phi(t)=0$ (ie. no information) and being close to 0 when $\phi(t)^T P(t-1)\phi(t) >> 1$. Therefore under conditions of low excitation forgetting stops, avoiding blowup, while for high excitation the rate of forgetting increases, allowing rapid adaption. These may be stated as the basic requirements for forgetting.

° Fortescue's variable forgetting factor is chosen as

$$\beta(t) = 1 - [1-\phi(t)^T K(t)]\frac{[y(t)-\theta(t-1)^T\phi(t)]^2}{\Sigma_0} \qquad (1\text{-}15)$$

This choice uses the value of the weighted least-squares cost as its scalar measure of information and aims to hold it constant. The user-chosen value of Σ_0 governs the sensitivity of this approach.

Many more attempts at retaining adaptivity without blowup, involving not only variable exponential forgetting but variable linear forgetting (Goodwin & Sin, 1984)

$$P'(t) = P(t) + Q \qquad (1\text{-}16)$$

where Q is a variable positive definite matrix and the so-called directional forgetting of Kulhavy (1986)

$$P'(t) = P(t) + P(t)\phi(t)\left[\frac{1}{\alpha} - \phi(t)^T P(t)\phi(t) \right]^{-1} \phi(t)^T P(t) \qquad (1\text{-}17)$$

make this area one of great current interest.

Continued adaptivity in a self-tuner is required in order to relax the initial assumption of section 1.1.1 that the plant dynamics are time-invariant. If the dynamics change, due say to a large move in the operating point for a non-linear plant or due to component wear, the self-tuner should ideally track these changes and keep the closed-loop 'tight'. This is, as discussed in section 1.1.5, one of the main alternative philosophies to the robust design strategies such as H^∞ (Zames, 1981) where, given bounds on the uncertainties in the plant model from time-varying dynamics, non-linearities etc., a fixed controller is designed. This is designed to provide possibly lower-performance control which is, however, less sensitive to variations in the plant dynamics (Astrom, 1986).

1.1.4 Control: Predictive strategies in Self-tuning

The first recognisable self-tuner, generally attributed to Kalman (1958), was based upon a simple cancellation controller. The controller parameters were obtained from a difference equation model of the form (1-7)

which was estimated on-line to make the combined algorithm self-tuning. Due to the limited computational technology of the day Kalman's implementation and paper had little initial impact.

Subsequent work in the self-tuning control field has mainly stayed in the polynomial domain (as opposed to state-space) and can be divided arbitrarily into two classes (Tuffs, 1984): first, methods which rely on the minimisation of a quadratic cost functional in the process variables, and second, the placement of closed-loop poles at pre-specified locations. Recently these relatively independent streams of research have come together with the Generalised Predictive Controller of Clarke et al (1984,5,7) and the Generalised Pole-Placer of Lelic & Zarrop (1986) sharing many common features. Since GPC is the control strategy applied throughout this thesis this review of the control side of self-tuning restricts itself to those predictive algorithms minimising a quadratic cost.

The seminal work of Kalman was eventually revived with the key papers of Peterka (1970) and Astrom & Wittenmark (1973) introducing the minimum-variance regulator. These algorithms used a predictor across the plant delay for the plant output $y(t+k)$ and minimised a single-stage cost in the prediction $\hat{y}(t+k|t)$. Problems with these algorithms arose from a lack of identifiability in the closed-loop (parameters could lie along a linear manifold), unstable control for NMP* plant (due to cancellation) and no provision for set-point tracking. Ad hoc modifications to overcome these problems soon followed such as fixing a single parameter to an arbitrary value to aid identification (Astrom & Wittenmark, 1973; Clarke, 1981), and extensions to include set-point tracking (Wittenmark, 1973).

Generalised Minimum Variance (Clarke & Gawthrop, 1975) extended the single-stage cost-function to include a penalty on the system error and control signal, which gave correct set-point following and a partial solution for NMP systems via control detuning. This paper introduced 'tuning-knobs' ie. design polynomials (1975) and transfer functions (Gawthrop, 1977; Clarke

* Non-minimum phase.

& Gawthrop, 1979) allowing flexible controller objectives. This was further extended by Clarke & Gawthrop (1979) to allow control weighting to be changed on-line without damaging the estimates. This was at the expense of losing the fully implicit implementation. The advantage of implicit algorithms, where the controller parameters are estimated directly, is their computational simplicity. The advantage of explicit algorithms, where the model parameters are estimated directly and used in the synthesis of the control law, is their ability to accept overparameterisation of $\hat{B}(z^{-1})$ to encompass uncertainty in the plant delay. Another advantage is that the 'tuning-knobs' (controller specification) may be changed on-line.

Interpretations for the GMV method such as model-reference and detuned model-reference control were provided by Gawthrop (1977). Gawthrop (1980) further reformulated GMV in continuous-time (hybrid GMV), basically using a pseudo-continuous-time inverse model to reduce the relative order of the overall plant model. This can eliminate the NMP discrete-time zeros which arise from discretisation of continuous-time plant with high relative order (Astrom et al, 1984). Hybrid GMV can then control previously NMP plant. Some problems remained with the GMV approach, including the requirement for accurate prior knowledge of the plant time-delay and the inability to stabilise open-loop unstable and NMP systems.

The above algorithms were derived on the basis of the CARMA model (1-4) with zero-mean noise and relied upon estimation of any unmeasurable d.c. disturbance to reject steady-state offsets (Clarke et al, 1983a). Recently it has been found that incremental self-tuning algorithms derived using a process model with an integrated noise structure (Belanger, 1983; Cameron & Seborg, 1983) the CARIMA model (1-5), automatically include integral action into the controller. The integral action arises from the requirement to reject steady-state offsets which are modelled as a non-zero mean integrated moving-average disturbance.

Early in the development of self-tuners, Peterka & Astrom (1973) presented a method based on the minimisation of a multi-stage cost functional. This overcame the problem of NMP zeros while maintaining the minimum variance objective (for vanishingly small control weighting). Based in the state-space this LQG (linear-quadratic-gaussian) method was, however, found to be computationally very taxing. Lam (1980) extended the approach and Clarke et al (1985a) further generalised the LQG approach of Lam by adding engineering design features similar to those of GMV. The algorithm included an N-stage horizon with weighted control increments and the CARIMA model, which includes the desired integral action in the resulting control law. This algorithm was shown to cope with variable plant delay, non-minimum-phasedness and open-loop unstable plant. However, this method, where the estimated model (1-7) is transformed into its equivalent state-space representation, is sensitive to over-parameterisation and is computationally taxing. There are ways of alleviating the load; for example Lam (1980) iterates the associated Riccati equation only once per cycle.

The concept of long-range prediction in the polynomial domain was introduced by Richalet et al (1978) with IDCOM, which used a weighting-sequence model and an **ad hoc** method for avoiding offsets with no weighting on control. The DMC method of Cutler & Ramaker (1980) introduced the important concept of a limited control horizon beyond which the control increments are assumed to be zero. However it also suffered from bad parameterisation (step-response model) and a heuristic solution to the offset problem. NMP plant could be controlled but, due to the parameterisation, open-loop unstable plant could not. These early long-range predictive algorithms are compared in Clarke & Zhang (1987a).

Generalised Predictive Control (Clarke et al, 1985) is a synthesis of the above long-range predictive methods. It is a receding-horizon method, calculating a series of future controls every sample but only exerting the first control of the sequence . The DMC-like assumption of a control-horizon

(see section 2.4.1) brings benefits in practical 'robustness' and simplified computation. The follow-up paper (Mohtadi & Clarke, 1986) introduces interpretations for many current control algorithms as being subsets of GPC obtained by choosing particular values for the output and control horizons. GPC overcomes all the problems that LQG overcame and seems less sensitive to overparameterisation. It may be considered as a version of LQG for an I/O* model, or as an extension to the GMV predictive controller over a larger time horizon.

1.1.5 Aims of Self-Tuning Control

Control strategies designed to overcome plant uncertainty are divided by Astrom (1986) into two complementary approaches: robust and adaptive control. Seborg et al. (1986) further divides adaptive control into self-tuners, stability-based methods (eg. Model-Reference-Adaptive-Control), gain-scheduling and other miscellaneous approaches. Robust controllers require a prior model of the process with known bounds on the model uncertainties. This model is used to design a fixed-gain controller which is insensitive to variations in the plant within the specified bounds. The majority of self-tuning controllers described in the literature are based on the certainty-equivalent principle (Jacobs, 1981) where a recursively estimated model (1-7) is directly substituted into an on-line controller design calculation (eg. section 2.5) to give a time-varying control law which ideally converges to the controller which would have been designed given exact knowledge of the plant (Clarke, 1981).

The main aims of self-tuning, loosely ordered in terms of their relative importance, are

° To simplify the commissioning phase of a control design. Initial tuning of even the simple PID algorithm can be a complex and time-consuming

* Input/output ie. CARMA or CARIMA.

task (Clarke & Gawthrop, 1979). Higher-order controllers would often yield improved performance but cannot be tuned on a trial-and-error basis (Wittenmark, 1975) and prior modelling, followed by a conventional synthesis, is often difficult and/or economically unjustifiable (Clarke, 1982). These existing problems can be overcome by self-tuners which, given minimal prior information regarding the process, can automatically tune to give high-performance control using input-output data from test-signals, a previous controller or even, in some cases, the tuning-transient of a self-tuner commissioned in closed-loop.

- To retain high-performance control for time-varying systems. These variations can be caused by shifting operating points in non-linear systems, component wear, changing feed quality etc. For this the self-tuner must be in full closed-loop with the running i/o data being used, via the estimator, to redesign the controller which is itself generating the data.

- To provide the user with performance-related tuning-knobs whose choice directly affects closed-loop bandwidth, the level of control activity, degree of overshoot etc. and are therefore more meaningful to an operator than the P, I and D terms of a classical 'three-term' controller. The adaption mechanism is used to achieve this desired closed-loop performance independently of the plant.

- Wittenmark (1975) also highlights the often ignored fact that typical DDC* systems involve several loops which would require a vast amount of operator tuning and retuning. Continual, or even periodic, retuning of these loops via a self-tuner should improve the total plant performance. It is worth noting that with estimation disabled self-tuners return to fixed-gain controllers which may have a PID structure (eg. Cameron & Seborg, 1983).

Of course there are problems associated with self-tuning control. Astrom (1986) notes that the certainty-equivalent principle may lead to poor performance when the estimated model is inaccurate. This may be observed in

* Direct Digital Control.

the experimental results of section 5.4. The usual and somewhat arbitrary choice of model order (Seborg et al, 1986) can cause difficulties especially when underestimated ie. neglected dynamics (Rohrs et al, 1984). The retention of adaptivity with time is a complex problem with contradictory requirements to maintain estimator gain during periods of high excitation in the data while avoiding catastrophic phenomena such as blowup in periods of low excitation (Astrom, 1983). This can be achieved by careful 'jacketting' (Clarke, 1982) of the estimator such as deadbands, variable forgetting factors, spike filters etc. Finally the complex non-linear and time-varying nature of a self-tuner makes stability, convergence and robustness analyses extremely difficult. Existing analyses (Astrom, 1983) are based upon assumptions about the underlying process which are hard to verify in practice. This problem is usually sidestepped by proving algorithms in extensive simulation and experimental studies such as that presented in chapter 5.

To conclude, Astrom (1986) notes that while robust control systems respond faster to variations in the process parameters (inside the specified bounds) they generally utilise higher loop gains making them more sensitive to noise. Adaptive algorithms, while responding slower, do not require bounds on the model uncertainties and achieve a better final response especially for NMP systems. Astrom suggests that using a robust control design in an adaptive system, ie. moving away from the certainty-equivalent principle, is a logical but difficult continuation for future research.

1.2 Control of Flexible Systems

Flexible systems may be roughly defined as those whose transfer-function descriptions are composed of lightly damped resonant modes ie. poles

near the stability boundary. Being distributed parameter systems (van den Bossche et al, 1986) these transfer-functions are theoretically of infinite order and in practice are usually of a much higher order than is commonly met in process control. Interest in the control of these systems has principally come from two areas: firstly, from research into space structures composed of light flexible links and, secondly, from research into the effect of compliancy on robot manipulators. A famous example uniting both these areas is the Remote Space Shuttle Manipulator (Nguyen et al, 1982).

The majority of the work undertaken for this thesis concerns the application of the GPC self-tuner (mentioned in section 1.1.4 and described in chapter 2) to the control of the latter type of flexible system: a direct-drive single-link robot arm designed to exhibit exaggerated flexibility. This compliant arm, described in section 4.1, is considered to offer excellent research opportunities for the following reasons:

- Several other attempts at the control of similar arms have been made (see section 1.2.1) and limited comparisons can therefore be made between GPC and these existing control strategies.

- It is interesting to investigate how GPC will need to be configured to cope with high-order flexible systems which are borderline stable while satisfying higher-performance requirements than encountered in typical detuned process control.

- The recent move (see section 1.1.4) to incremental algorithms, required for offset-free control, has magnified control problems due to plant-model-mismatch (Ortega et al, 1985). Given that this compliant arm has negligible sensor noise, has been accurately analytically modelled (Fraser, 1987) and is theoretically of infinite order it should provide a useful test-bed for investigating the effects of the inevitable PMM encountered by controllers designed on the basis of finite-order models.

- In the context of robotics the set-point trajectory may be assumed to be known in advance, say from a path planner higher up in some control

hierarchy. Since GPC can conveniently incorporate knowledge of future set-points into its calculation it is expected, and shown, that this feature is a principal advantage of GPC over alternative strategies.

- The non-linearities present in this direct-drive compliant arm, stiction and saturation, are well defined and their effects, together with methods to overcome them, may easily be investigated.
- The requirement for fast sampling gave rise to a subsidiary research area of substantial interest: the design and development of a coarse-grained parallel-processing architecture for the GPC algorithm (see chapter 4).

1.2.1 Literature survey

Kanoh et al (1986) echo most authors in the literature when they state that conventional robots are mechanically stiffened to avoid structural deformation which would reduce their accuracy - accuracy of course being a prime requirement for robots. This stiffening results in quite massive robots with increased material costs, increased energy consumption and larger inertial forces which cause difficulty in tracking and deteriorate positioning accuracy. Stiffened robots are then bounded in their maximum speed by the requirement to avoid exciting the high-frequency structural resonances (Meckl & Seering, 1986) and more seriously, according to Sweet & Good (1985), by the actuation realistically available.

The seminal paper of Cannon & Schmitz (1984) suggests that the next generation of industrial robots will need to be much lighter, for quicker responses and more efficient energy usage ie. a greater proportion of the actuation energy being used to move the load rather than the arm itself. To overcome the problem of compliance they advocate direct end-point sensing, where the sensor is at one end of the flexible link and the actuator is at the other (non-collocation), combined with sophisticated control techniques.

Conventional robot controllers use sensors collocated with the actuators and, given joint position, evaluate the end-point position by assuming the link to be rigid. This is clearly inappropriate for compliant robots where the links undergo significant deflection. However direct end-point sensing is considered to be a very difficult control problem (Cannon & Schmitz, 1984; Meldrum & Balas, 1986). This is because of the non-minimum-phase transfer-function relating end-point position to actuator input combined with the somewhat primitive control algorithms used.

An alternative to direct end-point sensing is described by Nemir et al (1986) who use joint position plus strain gauge measurements down the link to reconstruct the deflected shape of the arm and hence infer end-point position. This may be justified because of the expense or complication of (remote) end-point sensing and shows good results.

Conventional robots are mainly powered by a.c. or d.c. motors via harmonic drives for speed reduction. Sweet & Good (1985) studied the effects of drive flexibility, drive stiction/friction and actuator saturation. They concluded that these non-linear characteristics of the drive system exhibit resonant/anti-resonant or compliant-like behaviour. This can be overcome either by local feedback linearisation (Khorasani & Kokotovic, 1985) or by using direct-drive motors, where the motors drive the link directly with no intermediate speed reduction, or a combination of the two approaches (Cannon & Schmitz, 1984).

Initial attempts at the control of predominantly single-link compliant arms involved deriving an accurate analytical model of the arm from which a reduced order model, suitable for the design of a controller, may be obtained by truncating (Truckenbrodt, 1981) the modes above some critical frequency. This is considered a reasonable approach since higher frequency modes make smaller contributions to the total response (Kanoh et al, 1986) and may be regarded as disturbances. However, while controllers and actuators, generally being of a low-pass nature, are not considered likely to

significantly excite these neglected dynamics (control spillover) the inevitable mismatch between the finite-order models and high-order reality (observation spillover) can easily cause instability (Balas, 1978).

The majority of controllers developed for these compliant links (eg. Cannon & Schmitz, 1984) use a combination of joint sensing, end-point sensing and strain-gauge sensing to drive a state observer designed on the basis of the reduced order model. Given the estimated dynamic state of the plant LQG state feedback is used to drive the end-point position to the set-point. However Cannon & Schmitz (1984) found that this relatively high order compensator was not robust to modelling errors and they moved to a lower order sub-optimal compensator with excellent results.

Since, for a compliant robot, the payload is expected to be a large proportion of the total robot mass the dynamics of the arm will change significantly with payload, say throughout the course of a pick-and-place operation. This theoretically can be overcome by introducing an adaptive outer loop into the control scheme. Cannon & Schmitz (1984) suggest gain-scheduling which assumes that payload masses and their effect upon the model are known in advance. Model Reference Adaptive Control has also been a popular candidate for the adaptive control of compliant arms. Siciliano et al (1986) obtained stable MRAC control on a non-linear simulation with collocated sensors whereas Meldrum & Balas (1986) showed that their MRAC algorithm was unstable on a non-linear simulation with end-point sensing (non-collocation). The authors attribute this to the system failing to satisfy a positive realness condition which is closely related to the non-minimum-phasedness of the non-collocated system. Nemir et al (1986) obtained reasonable control using a self-tuning pole-placer based on feedback of estimated end-point position on a real compliant arm. However the authors note that unless future controller set-points are somehow included in the control calculation (a realistic possibility in the field of robotics where the trajectory is known **a priori**) a lag of at least the system time delay

must be tolerated. This delay is shown to be removed by the Generalised Pole-Placement algorithm of Lelic & Wellstead (1986) which, loosely interpretable as a pole-placing version of GPC, successfully incorporated future set-points in experiments on a conventional rigid robot. For this reason combined with the fact that GPC was developed with the problem of non-minimum-phasedness in mind Chapter 5 describes trials of the GPC self-tuner on an end-point sensed direct-drive compliant link.

1.3 Structure of the Thesis

The general structure of the thesis is itemised below. A more detailed introduction precedes the individual chapters.

Chapter 1 presents background theory and literature surveys pertaining to the fields of self-tuning control and compliancy in robotics. The section on self-tuning control discusses basic modelling considerations, the description of disturbances, identification, parameter estimation, retention of adaptivity via data forgetting and the evolutionary route which lead to the GPC control strategy - the knowledge of all of which is assumed in the following chapters. The section on compliant robots concentrates on compliancy as a control problem involving lightly-damped resonances and NMP behaviour and surveys previous attempts at control.

Chapter 2 gives a detailed description of the GPC control algorithm, providing a more accessible interpretation of the controller as a judicious combination of a predictor with a deterministic cost function. The need for an incremental formulation is stressed and the common sources of offsets explained. Unlike previous descriptions of the GPC multi-step-ahead predictor which advocate recursion of a diophantine equation a more efficient method,

based upon iterating the estimated plant model, is described and shown (in appendices A.1-A.3) to be equivalent.

The design filters $P(z^{-1})$, $\lambda Q(z^{-1})$ and $T(z^{-1})$, used to tailor the closed-loop performance, are considered in some detail together with their incorporation into the model iterative scheme. Heuristic interpretations united with simulated tuning exercises are used to suggest and reinforce guidelines for their choice. In particular the filter $T(z^{-1})$, of critical importance for incremental control, is examined with useful modelling and signal conditioning aspects being clarified. Finally the extension of GPC to self-tuning and some implementation details are given.

Chapter 3 investigates the possibilities arising from incorporation of prior knowledge into the least squares estimator upon which self-tuning GPC is based. A brief modelling section reports the derivation of a transfer-function for the dynamics of the real compliant arm experimented on in chapter 5. This model then provides an excellent example of the availability of prior knowledge for experimental purposes. Methods for the inclusion of typical prior information are proposed and their effect shown in simulation and experiments on the arm. The usefulness of δ-transforms as a means of simplifying this inclusion is also investigated.

Chapter 4 describes in detail the mechanics of the compliant arm, the high-speed parallel-processing computer hardware developed to control it and the software engineering of the GPC self-tuner. As far as hardware is concerned attention is paid to the aspects of multi-processor computing required to achieve the necessary high sample rates. The software description emphasises the task-oriented structure of a self-tuning controller, a natural target for useful division into concurrently executing tasks. Details are given of the means by which synchronisation and communication are handled between concurrent tasks.

Chapter 5 reports the successful trials of the GPC self-tuner on the NMP, neutrally stable and lightly damped compliant arm modelled in chapter 3. All the tuning-knobs and design-filters of GPC are applied, varied and their effect on closed-loop performance observed. This adds to the body of knowledge whose practical aim is to ensure that a GPC user can, through the choice of these parameters, tailor the closed-loop as required. In particular the need for incremental or integrating control, to eliminate offsets, is demonstrated along with the associated need for $T(z^{-1})$, as discussed in chapter 2. This application also represents the first practical trials of GPC's pre-specified set-points in a context where they can justifiably be said to be available.

Chapter 6 draws conclusions from the results presented throughout this thesis and unites items whose significance is felt in more than one chapter. Suggestions for further work, consistent with the research undertaken through this course of study, are presented.

Appendix A presents some proofs too unwieldy to be incorporated into the text. Most important amongst these is the proof that the predictions obtained from the diophantine and model-iterative methods are identical.

Appendix B provides a list of the principal symbols used in equations throughout the text.

Appendix C is a bibliography for those sources referenced in the text.

Chapter 2

GENERALISED PREDICTIVE CONTROL

GPC is based upon the cumulative theoretical and practical experience gained by the Oxford Self-Tuning research group since the early '70s. It differs from its predecessors mainly by having a more generalised cost function (sections 2.4 and 2.5.4) based upon considering a bank of predictions of the plant output, rather than a single prediction. As was noted in Chapter 1 the majority of predictive control strategies are basically a combination of a predictor, a cost function, an algorithm to choose a control sequence which minimises this cost and, in the adaptive case, a parametric identification scheme. The first section, 2.1, provides an overview of how these components are combined to form a self-tuning controller. The following sections, 2.2-2.4, then describe the individual components of the algorithm's least complex formulation, positional GPC. Avoidance of offsets in the steady-state leads to the incremental formulation for GPC as described in section 2.5. Heuristic interpretations for the effects of the design polynomials $T(z^{-1})$, $P(z^{-1})$ and $\lambda Q(z^{-1})$ are then given in section 2.6 which, it is hoped, should help in their choice. Section 2.7 details the extension of GPC to the adaptive case and finally section 2.8 presents some aspects of this algorithm's implementation.

2.1 Overview of the GPC algorithm

Given a dynamic model of the process to be controlled (section 2.2) together with the current measured dynamic state of the process, in terms of past outputs and control inputs, predictions can be made for the future plant

outputs. Clearly future outputs will depend on future, as yet unexerted, control inputs. Bearing this in mind, and assuming linearity, the predictions $\hat{y} = \{y(t+1), y(t+2), \ldots, y(t+j)\}^T$ can be split into the sum of two prediction sequences

$$\hat{y} = \hat{y}_d + \hat{y}_f \qquad (2\text{-}1)$$

One of these sequences, the so-called 'free' response $\hat{y}_f$, represents the component of the future predictions unaffected by the future controls ie. the predicted response from the current state with no future controls exerted. The remaining sequence, the 'forced' response $\hat{y}_d$, represents the predicted response from zero state due solely to the future controls.

While $\hat{y}_f$ cannot be changed it is the control problem to choose the future control actions, and hence $\hat{y}_d$, such that the net predictions $\hat{y}$ approach the set-point. Given an accurate model and predictor, at least in the steady-state, the actual output should then also tend to the set-point. Section 2.3 describes the basic predictor and its extension to multi-step-ahead prediction while section 2.5.3 describes how incremental prediction is required for steady-state accuracy.

The future controls are chosen to minimise a least-squares cost of the form

$$J = (\hat{y}-w)^T(\hat{y}-w) \qquad (2\text{-}2)$$

which attempts to drive <u>all</u> the predictions close, in the mean-square sense, to the future set-point sequence w. The overall approach, as described in the preceding paragraphs, is illustrated in Fig.2.1.

The diagram shows a control sequence $\{\tilde{u}\}$ chosen in order to 'boost' an exponentially decaying free response to the set-point. Since the control sequence is chosen to minimise (2-2) the slightly oscillatory predicted

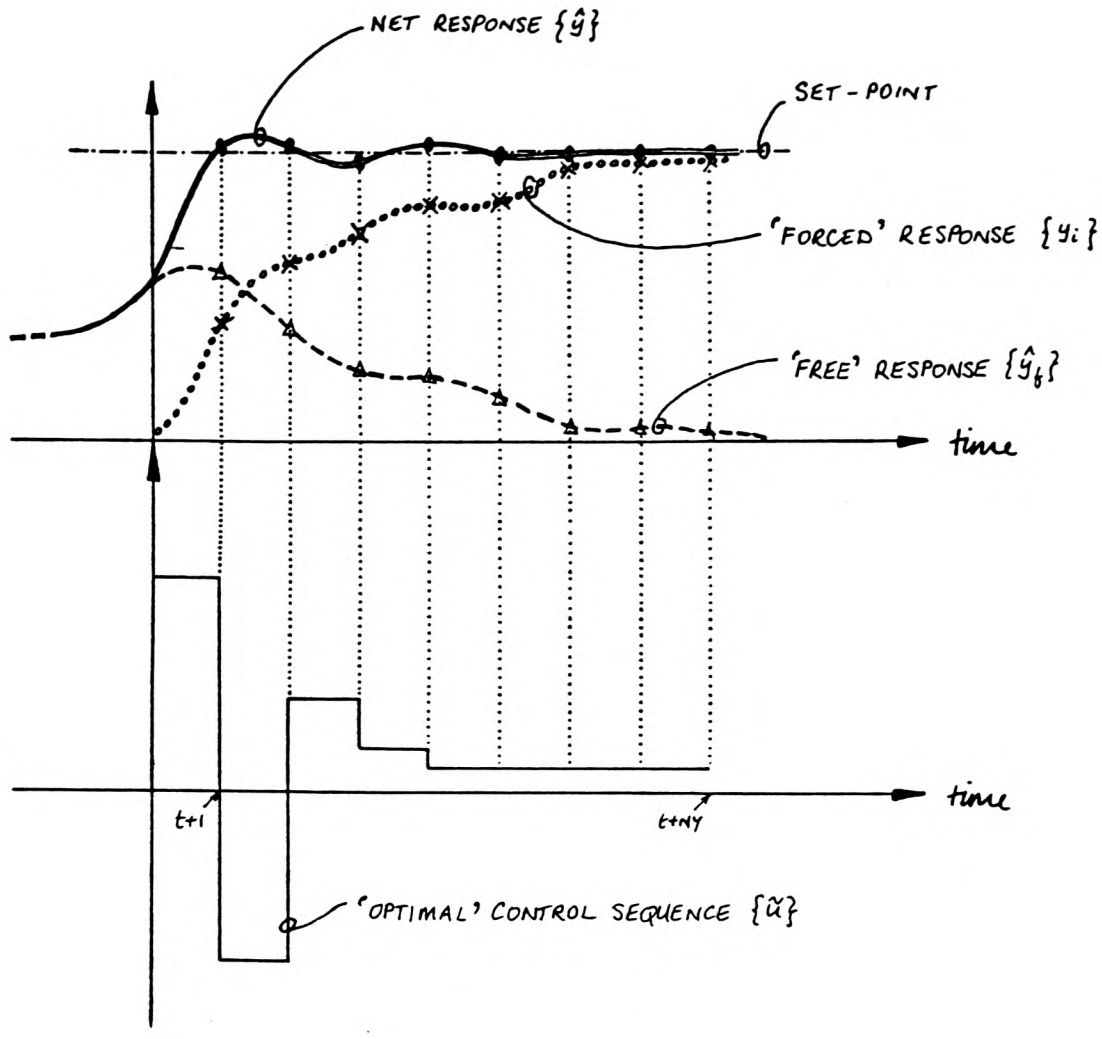


Fig.2.1: Predicted Free and Forced responses

response is, in that respect, the 'best' that is possible. The cost is described in far greater detail in sections 2.4 and 2.5.4, introducing the prediction horizons and cost weights which give GPC its considerable flexibility of configuration. These sections go on to show how substituting for the predictor in the cost gives (2-2) directly in terms of the future controls. An analytic expression for the future controls required to minimise the cost may then be simply obtained by differentiation with respect to the future controls. In all cases only the first control of the calculated sequence is exerted; the predictions, minimisation and control calculation being repeated on every subsequent controller sample. This makes GPC a receding-horizon control strategy (Kwon & Pearson, 1978). The restricted range over which GPC predicts the future allows it to be viewed either as an

extended version of GMV (Clarke & Gawthrop, 1975) or a limited version of LQ control (Peterka, 1984).

2.2 The Model

The control algorithms mentioned in this thesis, those described in the literature survey of section 1.1.4, and GPC as detailed in this chapter, require a mathematical model of the process to be controlled. GPC, as the latest member of the predictive family of controllers, uses this model to make predictions, based upon current measurements, of the future plant behaviour.

The rest of this chapter assumes that a least-squares identification scheme, such as that described in section 1.1.3, has provided a plant model of the form

$$\hat{A}(z^{-1})y(t) = \hat{B}(z^{-1})u(t-k-1) + x(t) \qquad (2\text{-}3)$$

where x(t) represents the net effect of plant-model-mismatch (due to estimation errors, unmodelled higher-order dynamics and non-linearities) and any load disturbances affecting the actual process. This model can exactly describe the actual process behaviour although x(t) is unmeasurable and is also generally assumed to be unpredictable (see section 2.3.1).

In the following sections a slight abuse of notation is introduced. While it is assumed that a model of the form (2-3) has been identified the circumflex representing estimated polynomials is dropped for clarity. The plant-delay k, again for simplicity, can either be assumed to be zero or incorporated as leading zero coefficients in the $B(z^{-1})$ polynomial.

2.3 Prediction: Single-stage and Multi-stage

A predictive controller requires a prediction of y(t+j), where j is some positive non-zero integer, given information up to time t and assumptions about future control inputs. The Clarke-Gawthrop (1975) predictive scheme is perhaps the most widespread method used to obtain these predictions through the definition of the so-called diophantine equation. This diophantine may either be solved to give a predictor equation (explicit methods) or substituted into the model (2-3) and the coefficients of the resultant predictor directly estimated (implicit methods). The resultant predictions, however, can also be obtained by iterating the known part of the plant model (2-3) from current state (de Keyser & van Cawenberghe, 1981). Both approaches are described below and proven to be equivalent in appendix A.1.

2.3.1 The Diophantine approach

The plant model (2-3) may be rewritten as

$$y(t+j) = \frac{Bu(t+j-1)}{A} + \frac{x(t+j)}{A} \qquad (2\text{-}4)$$

Assume that, after receiving the measurement y(t), the future controls $\{\tilde{u}\} = \{u(t), \ldots, u(t+j-1)\}$ are known and consider the disturbance term $x(t+j)/A(z^{-1})$. Expanding $1/A(z^{-1})$ as an infinite series, the disturbance becomes an infinite sequence containing future disturbances $\{x(t+1), \ldots, x(t+j)\}$ about which nothing is assumed known at time t, together with past disturbances $\{x(0),\ldots,x(t)\}$ which affected the past outputs in some way.

Consider the diophantine equation

$$\frac{1}{A} = E_j + \frac{z^{-j}F_j}{A} \qquad \text{or} \qquad 1 = E_jA + z^{-j}F_j \tag{2-5}$$

where E_j and F_j are polynomials in z^{-1} of degree j-1 and na-1 respectively. In the absence of detailed knowledge of x(t+j) this amounts to <u>assuming</u> that {x(t)} is an uncorrelated random sequence with (2-5) splitting the disturbance sequence into unknown future, $E_jx(t+j)$, and known past, $z^{-j}F_jx(t+j)/A$, components. Substitution of (2-5) into (2-4) yields

$$y(t+j) = F_jy(t) + E_jBu(t+j-1) + E_jx(t+j) \tag{2-6}$$

The predictor is then defined as

$$\hat{y}(t+j|t) = F_jy(t) + E_jBu(t+j-1) \tag{2-7}$$

with the associated prediction error

$$e(t+j|t) = E_jx(t+j) \tag{2-8}$$

These are in future referred to as the j-step-ahead prediction and the j-step-ahead prediction error. Since the degree of E_j = j-1 this predictor can be seen to be a reasonable strategy since the prediction error depends only on future disturbances which are, for practicality, assumed to be unpredictable. Indeed for the case when x(t) <u>is</u> a zero-mean uncorrelated RV and the model is exact this is the minimum-variance predictor (Clarke, 1981).

2.3.2 Model Iterative Approach

The current dynamic state of the plant is completely defined at time t by the measurement y(t) and the state vectors {y(t-1), y(t-2), ...,y(t-na)} and {u(t-1),u(t-2), ...,u(t-nb-1)}. The predictions (2-7) may also be obtained by iterating the known part of the plant model (2-4) from the current state and shifting previous predictions into the state vector, given that we know or assume future controls ie.

$$\hat{y}(t+1|t) = -a_1 y(t) - a_2 y(t-1) - \ldots - a_{na} y(t-na+1) + B(z^{-1})u(t) \qquad (2\text{-}9)$$

$$\hat{y}(t+2|t) = -a_1 \hat{y}(t+1|t) - a_2 y(t) - \ldots - a_{na} y(t-na+2) + B(z^{-1})u(t+1)$$

$$\hat{y}(t+3|t) = -a_1 \hat{y}(t+2|t) - a_2 \hat{y}(t+1|t) - a_3 y(t) - \ldots - a_{na} y(t-na+3) + B(z^{-1})u(t+2)$$

$$\vdots$$

It is shown in appendix A.1 that, for the same model (2-4), same inputs, outputs and disturbances, these predictions are identical to those obtained from the diophantine approach. This method, however, has the advantage that it is simpler and faster. While it was suggested in de Keyser & van Cawenburghe (1981) equivalence is not proven and they do not consider the case when $j < na$.

2.3.3 Multi-stage Prediction

The bulk of previous predictive controllers, eg. EHAC of Ydstie (1984) or GMV of Clarke & Gawthrop (1979), predict one future output either beyond or at the plant delay. However there has been a recent move towards prediction over a range of future outputs, say for {y(t+1),...,y(t+10)}, GPC itself being based upon this approach. There are basically three possible methods of extending the single-stage predictor to multi-stage prediction:

(i) The diophantine equation (2-5) could be solved for each j over the range of interest and the associated predictions obtained from substitution in (2-7). This method is, by comparison with the following methods, computationally taxing and suffers from the further drawback that the computational burden increases with j.

(ii) If the diophantine equation for the prediction nearest the present is solved Clarke et al. (1987) show that solutions for the diophantine equations associated with further predictions may be obtained in a simple recursive fashion starting from that solution. The details of this approach are not discussed since the third option is again simpler and computationally cheaper. It should be noted that the computational burden also increases with j.

(iii) The plant model may be iterated as shown in (2-9) from current state to the initial prediction horizon and on to the final prediction horizon. This has the advantage that the generated predictions feed back directly into the states of the iteration resulting in a computationally more efficient implementation. There is also the further advantage that the computational burden remains constant with j.

Although all three of these methods, as described, implicitly assume that the bank of predictions are contiguous in time ie. $\{...,y(t+5),y(t+6),y(t+7),...\}$ there are no difficulties applying these methods to non-contiguous prediction ie. $\{...,y(t+5),y(t+7),y(t+10),...\}$ In methods (ii) and (iii) any predictions not required are simply cycled back into the iteration.

2.3.4 Use of Multi-Stage Prediction in Control

Of course, in a practical controller at time t the future controls $\{\tilde{u}\} = \{u(t),u(t+1),...,u(t+j-1)\}$ are unknown. Indeed it is the control

problem to use the predictions, which themselves depend upon these controls, in order to determine the future control sequence. This seemingly paradoxical situation was resolved in section 2.1 by noting that a particular choice of the future controls $\{\tilde{u}\}$ will place the predictions $\{\tilde{y}\} = \{y(t+1), y(t+2), \ldots, y(t+j)\}$ at corresponding values. It is the job of the controller to choose the control sequence which 'best' steers the predictions, and thus hopefully the future outputs, to the set-point.

Since the model (2-4) is linear the **principle of superposition** applies ie. we can consider the prediction sequence to be the sum of two subsidiary prediction sequences:

(i) $\{\hat{y}_f\}$: The predictions of the plant's 'free' response from current state with no future controls applied ie. if the system went open-loop.

$$\hat{y}_f(t+1|t) = -a_1y(t)-a_2y(t-1)-\ldots-a_{na}y(t-na+1) + b_1u(t-1)+\ldots+b_{nb}u(t-nb)$$
$$\hat{y}_f(t+2|t) = -a_1\hat{y}_f(t+1|t)-a_2y(t)-\ldots-a_{na}y(t-na+2)+b_2u(t-1)+\ldots+b_{nb}u(t-nb+1)$$
$$\hat{y}_f(t+3|t) = -a_1\hat{y}_f(t+2|t)-\ldots-a_{na}y(t-na+3) + b_3u(t-1)+\ldots+b_{nb}u(t-nb+2)$$
$$\vdots$$

(ii) $\{\hat{y}_d\}$: The prediction of the plant's 'driven' response from zero state with the future, as yet unknown, control sequence $\{\tilde{u}\}$ applied.

$$\hat{y}_d(t+1|t) = b_0u(t)$$
$$\hat{y}_d(t+2|t) = -a_1\hat{y}_d(t+1|t) + b_0u(t+1) + b_1u(t)$$
$$\hat{y}_d(t+3|t) = -a_1\hat{y}_d(t+2|t) - a_2\hat{y}_d(t+1|t) + b_0u(t+2) + b_1u(t+1) + b_2u(t)$$
$$\vdots$$

It should be clear from comparing (2-9) with the above equations that

$$\hat{y}(t+j|t) = \hat{y}_f(t+j|t) + \hat{y}_d(t+j|t) \qquad (2\text{-}10)$$

It is obvious that the free response of a real plant will rarely of itself tend to the set-point so the driven response must be chosen such that, when added to the free response, the full predictions $\hat{y}(t+j|t)$ approach the set-point.

Before continuing on to see how the control sequence is arrived at it is convenient to put the predictions into a certain structure: the **key vector form** (Clarke et al. 1987). This is achieved by noting that the sequence $\{\hat{y}_d\}$, as it evolves from zero state, may also be obtained from the convolution sum (proven in appendix A.4)

$$\hat{y}_d(t+j|t) = \sum_{i=0}^{j-1} h_i u(t+j-i-1) \qquad (2\text{-}11)$$

where h_i are the ordinates of the unit pulse response of $B(z^{-1})/A(z^{-1})$ <u>not</u> $z^{-1}B(z^{-1})/A(z^{-1})$ ie.

$$\begin{aligned}
\hat{y}_d(t+1|t) &= h_0 u(t) \\
\hat{y}_d(t+2|t) &= h_0 u(t+1) + h_1 u(t) \\
\hat{y}_d(t+3|t) &= h_0 u(t+2) + h_1 u(t+1) + h_2 u(t) \\
&\vdots
\end{aligned}$$

Then from (2-10) and (2-11) the key vector form for the predictions is

$$\hat{\mathbf{y}} = \mathbf{H}\tilde{\mathbf{u}} + \hat{\mathbf{y}}_f \qquad (2\text{-}12)$$

where the vectors are all NY x 1.

$$\hat{\mathbf{y}} = [\ \hat{y}(t+1|t), \hat{y}(t+2|t), \ \ldots\ , \hat{y}(t+NY|t)\ ]^T$$

$$\tilde{\mathbf{u}} = [\ u(t), u(t+1), \ \ldots\ , u(t+NY-1)\ ]^T$$

$$\hat{\mathbf{y}}_f = [\ \hat{y}_f(t+1|t), \hat{y}_f(t+2|t), \ \ldots\ , \hat{y}_f(t+N|t)\ ]^T$$

and **H** is the lower-triangular matrix NY x NY:

$$\mathbf{H} = \begin{bmatrix} h_0 & & \cdots & 0 \\ h_1 & h_0 & \cdots & 0 \\ . & .. & \cdots & . \\ : & & & . \\ h_{NY-1} & h_{NY-2} & \cdots & h_0 \end{bmatrix}$$

with an associated vector of prediction errors, $y(t+j) - \hat{y}(t+j|t)$, given from (2-8) by

$$\hat{\mathbf{e}} = [\ E_1 x(t+1), E_2 x(t+2), \ \ldots\ , E_{NY} x(t+NY)\ ]^T$$

If the plant delay, k, is more than zero then the first k rows of **H** will be zero. If the predictions start at $t+N_1$ rather than at t+1 and $N_1 > k$ then the leading row is non-zero. This key vector form splits the predictions into two parts; that which it is possible to change, $\mathbf{H}\tilde{\mathbf{u}}$, and that which is fixed, $\hat{\mathbf{y}}_f$. With this in mind section 2.4 describes how this form is utilised in the control calculation.

2.4 The Cost Function: Specification and Minimisation

'Optimal' control in the polynomial domain usually involves minimisation of some cost which is a function of predicted deviations from the future set-point and future control activity. The cost function is chosen to embody sensible engineering objectives while allowing typical trade-offs

to be made. Furthermore the cost should be formulated such that simple modifications to it should have useful and easily understood effects eg. control detuning. Minimisation of this cost usually corresponds to driving the differences between the future predicted outputs and the future set-points to zero while not allowing the control activity to exceed available levels.

The majority of cost functions in the literature are introduced in a stochastic framework where some conditional expectation of a cost is minimised. This approach, however, can be misleading and in this thesis all costs are deterministic. The control laws derived from the minimisation of these deterministic costs can subsequently be shown to minimise the stochastic cost functions, which involve actual future outputs rather than predictions, given the usual quite restrictive assumptions about the structure of the disturbance ie. gaussian white noise (Clarke et al, 1987).

2.4.1 The Generalised Cost Function

$$J = \sum_{i=N_1}^{NY} (\hat{y}(t+j|t)-w(t+j))^2 + \sum_{j=1}^{NU} \lambda u(t+j-1)^2 \qquad (2\text{-}13)$$

In words, the sum of the squares of the predicted errors (different to the prediction errors of (2-8)) over the time range $t+N_1$ to t+NY plus the weighted sum of the squares of the future control sequence (λ being a positive constant). The chief innovation in this cost is the control horizon $NU \leq NY$ originating from the DMC algorithm of Cutler and Ramaker (1980). This is equivalent (Mohtadi, 1986) to putting infinite weighting on controls beyond u(t+NU-1) which effectively fixes them to zero. The initial prediction horizon N_1 also proves of use in configuring the GPC algorithm for pole-placement (Mohtadi & Clarke, 1986). The cost is therefore minimised subject

to the restriction that only NU future controls are available to bring the predictions to the set-point. This has the effect of reducing the required order of the key vector form of the predictions, which becomes

$$\hat{\mathbf{y}} = \mathbf{H}_1\tilde{\mathbf{u}} + \hat{\mathbf{y}}_f \qquad (2\text{-}14)$$

where the truncated matrices $\mathbf{H}_1$ and $\tilde{\mathbf{u}}$ are NYxNU and NUx1 respectively

$$\mathbf{H}_1 = \begin{bmatrix} h_0 & \cdots & 0 \\ h_1 & h_0 & \cdots & 0 \\ \cdot & \cdot\cdot & \cdots & \cdot \\ \cdot & \cdot\cdot & \cdots & h_0 \\ \cdot & \cdot\cdot & \cdots & \cdot \\ : & & & h_{NY-NU-1} \\ h_{NY-1} & h_{NY-2} & \cdots & h_{NY-NU} \end{bmatrix} \qquad \tilde{\mathbf{u}} = \begin{bmatrix} u(t) \\ u(t+1) \\ : \\ u(t+NU-1) \end{bmatrix}$$

and $\hat{\mathbf{y}}$ and $\hat{\mathbf{y}}_f$ and the prediction error vector $\hat{\mathbf{e}}$ are as before.

The Cost Horizons: N_1, NY and NU

The choice of these horizons in the cost (2-13) may be used to satisfy overlapping theoretical and practical objectives. From a theoretical standpoint the horizons (sometimes used together with the design filter $P(z^{-1})$ of section 2.6.1) can be chosen to configure GPC to behave identically to many existing control strategies. Mohtadi and Clarke (1986) use this to unify such differing approaches as the extended-horizon predictive controller of Ydstie (1984), the LQ-type controller of Peterka (1984) and the pole-placement strategy of Wellstead et al (1979). From a practical standpoint

these horizons have been found to provide useful tuning-knobs, the variation of which allows the closed-loop to be tailored to the user's requirements.

The remainder of this section outlines the guidelines used in practice for the choice of N_1, NY and NU. For further details and experimental results obtained through the choice and variation of these horizons see chapter 5.

° **N_1** is termed the INITIAL PREDICTION HORIZON. If the plant-delay is known to be at least k samples then N_1 is set equal to or more than k to avoid superfluous calculations.

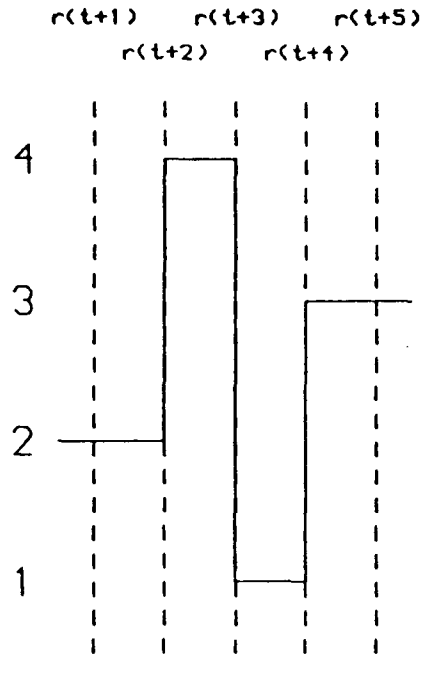
° **NY** is termed the FINAL PREDICTION HORIZON. It is usually chosen to encompass the majority of the plausible settling time, given the available actuation, of the closed-loop. More importantly it should exceed the duration of any negative-going NMP transient and be at or beyond the maximum possible plant-delay.

° **NU** is termed the CONTROL HORIZON and is usually chosen to reflect the complexity of the process under control. For simple plant ie. stable and low-order NU=1 while for more complex plant NU$\geq$2 provided actuation is cheap. As a rule of thumb NU should be set equal to the number of unstable or badly damped poles of the process. However as NU$\rightarrow$NY the GPC controller tends to minimum-variance control so NU should be chosen with caution.

Normal vs. Pre-specified Set-points

The cost (2-13) implies that future set-points w(t+j), j = N_1 to NY, are known in advance. In the context of process control, where GPC was first applied, this is not usually the case and for normal set-point following it is conventional to assume that future set-points will equal the current set-point ie. w(t+j) = w(t) for all j = N_1 to NY. When future set-points are available, say in the form of a sequence { ... r(t-1), r(t), r(t+1), ..}, and

trajectory following is required (eg. batch reactors and robot position control) the pre-specified set-points may simply be substituted into the control calculation (2-18) of section 2.4.2 as $w(t+j) = r(t+j)$. Fig.2.2 illustrates the difference between normal and pre-specified set-points with an example set-point trajectory and $N_1 = 1$, NY = 4.



	Set-point vectors w	
t:	[2 4 1 3]	[2 2 2 2]
t+1:	[4 1 3 3]	[4 4 4 4]
t+2:	[1 3 3 ?]	[1 1 1 1]
t+3:	[3 3 ? ?]	[3 3 3 3]
	pre-spec.	normal

Fig.2.2: Normal vs. Pre-specified Set-points

An experimental illustration of the differing effects of the two approaches is given in section 5.3.1.

2.4.2 Minimisation of the Cost Function

The cost function (2-13) may be rewritten in vector form to correspond with the key vector form for the predictions

$$J_1 = (\hat{\mathbf{y}}-\mathbf{w})^T(\hat{\mathbf{y}}-\mathbf{w}) + \lambda\tilde{\mathbf{u}}^T\tilde{\mathbf{u}} \qquad (2\text{-}15)$$

where the vectors $\hat{\mathbf{y}}$ and $\tilde{\mathbf{u}}$ are as defined in (2-10) and (2-14) and

$$\mathbf{w} = \{w(t+1), w(t+2), \ldots, w(t+NY)\}^T$$

Substituting the key vector form for the predictions,

$$J = (\mathbf{H}_1\tilde{\mathbf{u}} + \hat{\mathbf{y}}_f - \mathbf{w})^T(\mathbf{H}_1\tilde{\mathbf{u}} + \hat{\mathbf{y}}_f - \mathbf{w}) + \lambda\tilde{\mathbf{u}}^T\tilde{\mathbf{u}} \qquad (2\text{-}16)$$

differentiating w.r.t. the vector of future controls $\tilde{\mathbf{u}}$ and setting the result to zero

$$\frac{\partial J}{\partial \tilde{\mathbf{u}}} = 2\mathbf{H}_1^T\mathbf{H}_1\tilde{\mathbf{u}} + 2\mathbf{H}_1^T(\hat{\mathbf{y}}_f-\mathbf{w}) + 2\lambda\tilde{\mathbf{u}} = 0 \qquad (2\text{-}17)$$

gives the vector of controls

$$\tilde{\mathbf{u}} = (\mathbf{H}_1^T\mathbf{H}_1 + \lambda\mathbf{I})^{-1}\mathbf{H}_1^T(\mathbf{w}-\hat{\mathbf{y}}_f) \qquad (2\text{-}18)$$

These controls force the bank of future predictions to the minimum of the cost function (2-13) with a restriction on their magnitudes given by the control weighting λ. The first control of the sequence u(t) is then exerted and the whole process, as shown in section 2.8, repeated at subsequent sample instants. This approach, where only the first control of the sequence $\{\tilde{u}\}$ is

actually exerted, is known as the receding-horizon approach (Kwon & Pearson, 1978).

2.5 Incremental GPC

The positional derivation of GPC given in the preceding sections suffers from three important drawbacks; NU-offset, prediction offset and λ-offset. All of these effects, as their names imply, result in a potential steady-state offset between the output variable and the set-point. The incremental form of GPC described below in sections 2.5.2-2.5.4 is shown to overcome these problems. The derivation follows that of sections 2.2 to 2.4 with incremental forms for the model, the predictor and the cost function.

2.5.1 The Offset Problem

° **NU-offset**, as its name implies, arises from incorporating the control horizon NU $\leq$ NY into the cost function (2-13). This horizon specifies that the next NY predicted outputs should be steered to the set-point by appropriate choice of the next NU controls subject to the terminal condition that the final NY-NU controls are zero. This last condition results in the last NY-NU predictions being the model's free response from the predicted state at time t+NU. For stable plant without integrators this free response will take the form of an eventually exponential, possibly oscillatory, decay to zero. As NU decreases from NY the errors arising from this predicted decay contribute more towards the cost. This is reflected in GPC's choice of the NU controls in $\tilde{\mathbf{u}}$ which may allow greater initial errors, $\hat{y}(t+j|t)-w(t+j)$, in order to reduce those in the later decaying stage. This results in large steady-state offsets, especially for NU<<NY.

Since this effect has not been described before in the context of GPC a simulation example is provided for illustration. Consider the plant

$$y(t) = \frac{1}{(1+10s)(1+5s)}\, u(t) \qquad (2\text{-}19)$$

which is sampled every 2 seconds. Assume that the discrete-time version of (2-19) is exactly known and that a set of GPC controllers have been designed for it with λ=0, N_1=1, NY=5 and NU ranging from 1 to 5. Each controller was run, in simulation, with a constant set-point of 20 units until steady-state was achieved. Then the steady-state vector of future controls $\tilde{\mathbf{u}}_{ss}$ was calculated from

$$\tilde{\mathbf{u}}_{ss} = (\mathbf{H}_1^T\mathbf{H}_1)^{-1}\mathbf{H}_1^T(\mathbf{w}-\hat{\mathbf{y}}_f^{ss}) \qquad (2\text{-}20)$$

Fig.2.3 then shows the predictions $\hat{y}_{ss}(t+j|t)$ for j = 1 to 5 arising from $\tilde{\mathbf{u}}_{ss}$ ie.

$$\hat{\mathbf{y}}_{ss} = \mathbf{H}_1\tilde{\mathbf{u}}_{ss} + \hat{\mathbf{y}}_f^{ss} \qquad (2\text{-}21)$$

which is chosen to minimise the cost (2-13) in the steady-state.

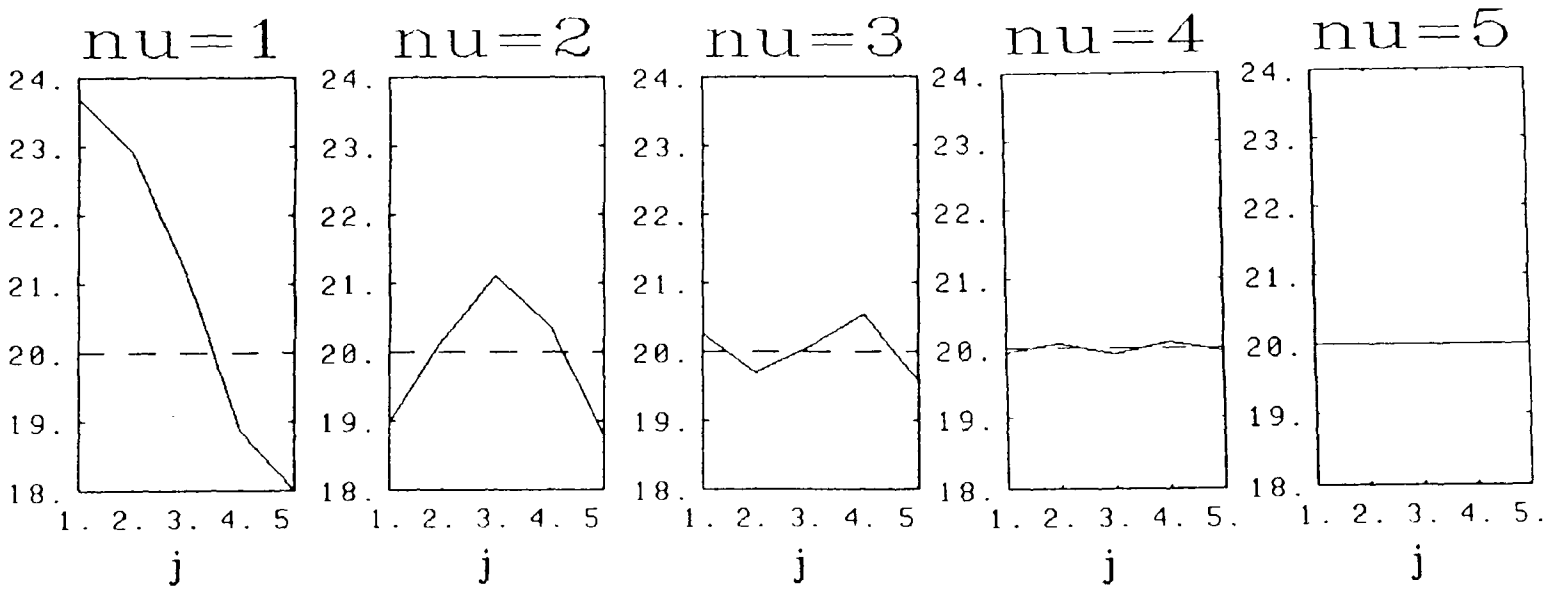


Fig.2.3: Steady-state predictions vs. NU

The offsets arising in $\hat{y}(t+1|t)$ show how the problem increases as NU

decreases from NY. When NU=NY the optimal steady-state control sequence

$$u(t+1) = u(t+2) = \ldots = u(t+5) = \bar{u} = \frac{A(1)}{B(1)}\bar{y} = \frac{A(1)}{B(1)}\bar{w}$$

may be exerted resulting in zero steady-state offset. If an integrator were present in the plant this offset would not occur since the optimal steady-state control would be zero satisfying the terminal constraint in the cost.

° **Prediction offset** arises from the fact that positional GPC attempts to drive a set of future predictions, and indirectly therefore future outputs, to the set-point. Consider the key vector form for the predictions

$$\hat{y} = H_1\tilde{u} + \hat{y}_f \qquad (2\text{-}22)$$

with the associated error vector $\hat{e} = [E_1x(t+1),\ E_2x(t+2),..\ E_{NY}x(t+NY)]^T$. If the disturbance sequence $\{x(t)\}$ in (2-3) has a finite mean, say due to an error in the estimated plant dc gain, then the error vector will have a dc component $\bar{e} = [E_1(1)\bar{x},\ E_2(1)\bar{x},..\ E_{NY}\bar{x})]^T$ ie. a steady offset on the predictions. So when, in the steady-state, the predictions are near the set-point there is no guarantee that the actual plant output is also near the set-point. The steady-state accuracy (ignoring NU-offset) of the positional algorithm therefore depends heavily upon the accuracy of the estimates, which is clearly not a robust feature (Hodgson, 1982 or Belanger, 1983).

° **λ-offset** arises from the positional cost function specification

$$J = \sum_{i=N_1}^{NY} [\hat{y}(t+j|t)-w(t+j)]^2 + \sum_{j=1}^{NU} \lambda u(t+j-1)^2 \qquad (2\text{-}23)$$

If in the steady-state the control settles to a finite value it may be advantageous, in terms of minimising the cost, to accept a small steady-state error between the predictions and the set-point which requires a reduced steady-state control action. Examples of this are given in Hodgson (1982). Note that if λ is zero or the plant has natural integrators ie. zero steady-state control, this problem does not appear.

2.5.2 The Incremental Model

The existing model (2-3) if multiplied on both sides by the differencing operator $\Delta(z^{-1}) = 1-z^{-1}$ gives

$$A(z^{-1})\Delta y(t) = B(z^{-1})\Delta u(t-1) + \Delta(z^{-1})x(t) \qquad (2\text{-}24)$$

Incremental GPC iterates the known part of this model, $A\Delta y(t)=B\Delta u(t-1)$, to give the prediction sequences. Note that any d.c. component of $x(t)$ is eliminated by the differencing operator $\Delta(z^{-1})$.

2.5.3 The Incremental Predictor

As in section 2.2 a diophantine identity may first be defined and solved to yield an incremental predictor equation. Identical predictions are then shown to be obtainable, as before, by simply iterating the incremental model (2-24). The incremental diophantine identity is

$$1 = E_j A\Delta + z^{-j}F_j \qquad (2\text{-}25)$$

which, when substituted into (2-24), gives the predictor equations

$$\hat{y}(t+j|t) = F_j y(t) + E_j B\Delta u(t+j-1) \qquad (2\text{-}26)$$

and

$$e(t+j|t) = E_j \Delta x(t+j) \tag{2-27}$$

The prediction error $e(t+j|t)$ is now insensitive to d.c. components of $\{x(t)\}$ which are eliminated by the difference operator $\Delta(z^{-1})$. Thus, in the absence of ramp-like disturbances, in the mean the predictions should converge to the actual future outputs, overcoming the problem of prediction offset.

A bank of predictions, identical to those obtained from the diophantine approach, may be obtained by iterating the known part of the model (2-24) from the current states $\{\Delta y(t), \Delta y(t-1), \ldots, \Delta y(t-na+1)\}$ and $\{\Delta u(t-1), \Delta u(t-2), \ldots, \Delta u(t-nb-1)\}$ assuming that future control <u>increments</u> are known ie.

$$\hat{\Delta y}(t+1|t) = -a_1 \Delta y(t) - a_2 \Delta y(t-1) - \ldots - a_{na} \Delta y(t-na+1) + B(z^{-1}) \Delta u(t) \tag{2-28}$$

$$\hat{\Delta y}(t+2|t) = -a_1 \hat{\Delta y}(t+1|t) - a_2 \Delta y(t) - \ldots - a_{na} \Delta y(t-na+2) + B(z^{-1}) \Delta u(t+1)$$

$$\hat{\Delta y}(t+3|t) = -a_1 \hat{\Delta y}(t+2|t) - a_2 \hat{\Delta y}(t+1|t) - \ldots - a_{na} \Delta y(t-na+3) + B(z^{-1}) \Delta u(t+2)$$

$$\vdots$$

As predictions are required for future outputs rather than future output increments one further step is required:

$$\hat{y}(t+1|t) = y(t) + \hat{\Delta y}(t+1|t)$$

$$\hat{y}(t+2|t) = \hat{y}(t+1|t) + \hat{\Delta y}(t+2|t)$$

$$\vdots$$

$$\hat{y}(t+NY|t) = \hat{y}(t+NY-1|t) + \hat{\Delta y}(t+NY|t)$$

It is shown in Appendix A.2 that these predictions are identical to those which would be obtained from solving the diophantine identity for each prediction horizon j=1 to NY and substituting in (2-26).

Proceeding, as before, from **superposition** the predictions may be split into two sequences: $\{\tilde{y}_f\}$, the predicted response from current state with no future control increments and $\{\tilde{y}_d\}$, the predicted response from zero state with the assumed sequence of future control increments applied. The sequence $\{\tilde{y}_d\}$, evolving from zero state, may be obtained from the convolution sum (Appendix A.5)

$$\hat{y}_d(t+j|t) = \sum_{i=0}^{j-1} g_i \Delta u(t+j-i-1) \qquad (2\text{-}29)$$

where g_i are the ordinates of the unit step response of $B(z^{-1})/A(z^{-1})$. The incremental **key vector form** for the predictions is

$$\hat{\mathbf{y}} = \mathbf{G}_1 \tilde{\delta u} + \hat{\mathbf{y}}_f \qquad (2\text{-}30)$$

where the matrix $\mathbf{G}_1$ is lower-triangular NY x NU:

$$\mathbf{G}_1 = \begin{bmatrix} g_0 & & \cdots & 0 \\ g_1 & g_0 & \cdots & 0 \\ . & .. & \cdots & . \\ . & .. & \cdots & g_0 \\ . & .. & \cdots & . \\ : & & & g_{NY-NU-1} \\ g_{NY-1} & g_{NY-2} & \cdots & g_{NY-NU} \end{bmatrix}$$

and the vectors concerned are given by

$$\hat{\mathbf{y}} = [\ \hat{y}(t+1|t), \hat{y}(t+2|t), \ \ldots\ , \hat{y}(t+NY|t)\ ]^T$$

$$\tilde{\delta u} = [\ \Delta u(t),\ \ldots\ ,\Delta u(t+NU-1)]^T$$

$$\hat{y}_f = [\ \hat{y}_f(t+1|t),\hat{y}_f(t+2|t),\ \ldots\ ,\hat{y}_f(t+N|t)]^T$$

2.5.4 The Incremental Cost Function: Specification and Minimisation

Incremental GPC modifies the cost function to

$$J = \sum_{i=N_1}^{NY} [\hat{y}(t+j|t)-w(t+j)]^2 + \sum_{j=1}^{NU} \lambda[\Delta u(t+j-1)]^2 \qquad (2\text{-}31)$$

Since the control weighting now penalises control increments rather than control amplitudes the steady-state contribution to the cost of a steady control signal will be zero. This, of course, removes the problem of λ-offset. This modification, however, fundamentally changes the cost function which must now be minimised by NU future control increments rather than by NU finite controls. Since the terminal condition on the last NY-NU controls in (2-31) is now that they must be constant (as opposed to zero in positional GPC) the problem of NU-offset is also removed by this new cost (see section 2.5.1).

The cost is minimised exactly as in section 2.4.2 giving the vector of future control increments

$$\tilde{\delta u} = (G_1^T G_1 + \lambda I)^{-1} G_1^T (w-\hat{y}_f) \qquad (2\text{-}32)$$

The first increment $u(t) = u(t-1) + \Delta u(t)$ is then exerted and the whole process repeated.

2.6 The Design Filters: $P(z^{-1})$, $\lambda Q(z^{-1})$ and $T(z^{-1})$

GPC provides several 'tuning-knobs' with which the behaviour of the closed-loop may be modified. Three of these 'tuning-knobs' have already been mentioned in section 2.4.1 - the horizons N_1, NY and NU. This section deals with three more - the design filters $P(z^{-1})$, $\lambda Q(z^{-1})$ and $T(z^{-1})$.

The finite-horizon nature of the GPC cost-function combined with its receding-horizon implementation, while resulting in a fixed linear controller (for no adaptation) of finite and known order, does not admit a particularly meaningful expression for the closed-loop in terms of P, λQ or T. Therefore it is difficult to obtain general analytical interpretations for the effects of these design filters. However this section outlines the current philosophy governing their choice based on associated heuristic explanations of their effect. While there is little rigour present the 'rules-of-thumb' arrived at have generally proven simple to apply and effective in their application. For examples see chapter 5, Lambert (1987) and Al-Assaf (1987).

For each of the design filters their incorporation into the incremental GPC control calculation is described, a heuristic interpretation is given and, based upon this, the choice of their values is discussed with reference to a relevant simulation example.

2.6.1 The design filter $P(z^{-1})$

When $P(z^{-1})$ is utilised GPC attempts to steer predictions of an auxiliary output $\psi(t+j) = P(z^{-1})y(t+j)$ to the set-point in place of predictions of the actual output $y(t+j)$. The d.c. gain of the filter, $P(1)$, must therefore be set to unity to avoid unnecessary offsets.

Incorporation of $P(z^{-1})$

To obtain predictions of the filtered output $\psi(t) = P(z^{-1})y(t)$ the incremental diophantine equation (2-25) is modified to

$$\frac{P_n}{P_d} = E_j \Delta A + z^{-j} \frac{F_j}{P_d} \qquad (2\text{-}33)$$

yielding the prediction equation

$$\psi(t+j) = Py(t+j) = E_j B \Delta u(t+j-1) + \frac{F_j}{P_d} y(t) + E_j \Delta x(t+j) \qquad (2\text{-}34)$$

The predictions, $\hat{\psi}(t+j|t) = P(z^{-1})\hat{y}(t+j|t) = E_j B \Delta u(t+j-1) + \frac{F_j}{P_d} y(t)$, are now used to minimise the cost

$$J = \sum_{i=N_1}^{NY} (P(z^{-1})\hat{y}(t+j|t) - w(t+j))^2 + \sum_{j=1}^{NU} \lambda[\Delta u(t+j-1)]^2 \qquad (2\text{-}35)$$

In practice, then, the difference that $P(z^{-1})$ makes to the incremental GPC control calculation (2-32) is that the augmented plant $P(z^{-1})B(z^{-1})/A(z^{-1})$ is iterated from zero state to obtain the step response parameters g_i in the matrix G_1 and from current state to obtain the predicted 'free' response vector $\hat{y}_f$. Given these filtered values the incremental GPC control is generated as before

$$\tilde{\delta u} = (G_1^T G_1 + \lambda I)^{-1} G_1^T (w - \hat{y}_f) \qquad (2\text{-}36)$$

where G_1 and $\hat{y}_f$ are composed of the filtered values.

Interpretations of $P(z^{-1})$

° When the control horizon equals the prediction horizon ie. NU=NY$\geq$k and the control weighting λ is set to zero Mohtadi (1986) states that GPC becomes the minimum-variance controller for the augmented plant PB/A with the closed-loop (given the model (2-24) and incremental control) becoming

$$y(t) = \frac{1}{P(z^{-1})} w(t) + E_1(z^{-1}) \frac{\Delta x(t)}{P(z^{-1})} \qquad (2\text{-}37)$$

In this case $P(z^{-1})$ allows a minimum-variance model-following interpretation with $P(z^{-1})$ being the inverse of the desired model.

° Since all GPC's tuning-knobs essentially vary only in the manner in which they detune the closed-loop from minimum-variance control this inverse-model interpretation suggests that the choice of $P(z^{-1})$ may be viewed as setting an upper limit on the possible bandwidth of the closed-loop. Choosing a high-pass $P(z^{-1})$ ensures that the closed-loop will always be detuned from the minimum-variance low-pass model $1/P(z^{-1})$.

° In practice GPC will always be detuned from minimum-variance by setting NU < NY and, often, by incorporating finite control weighting, λ. Then the model-following interpretation becomes, at best, approximate and $P(z^{-1})$ should be viewed as a generalisation of the derivative term of a PID controller. Since the filtered predictions $\hat{\psi}(t+j|t)$ lead the predictions of the actual output, for high-pass $P(z^{-1})$, it can be used to reduce overshoots and generally slow the closed-loop down by a simple injection of phase advance into the loop.

Choice of $P(z^{-1})$

° Current practice chooses $P(z^{-1})$ to be a stable first-order polynomial having unity gain ie.

$$P(z^{-1}) = \frac{1}{1-\alpha}[1 - \alpha z^{-1}] \qquad (2\text{-}38)$$

where $0<\alpha\leq 1$. Unless NU, NY and λ are such that inverse-model following should result the choice of $P(z^{-1})$ as a higher order polynomial or transfer function is not commensurate with the imprecision in the interpretations currently available for $P(z^{-1})$.

° To illustrate the choice of α in (2-38) consider the requirement to reduce overshoot in the control of the system

$$G(s) = \frac{}{s^2} \qquad (2\text{-}39)$$

where the actuation is limited to ± 10 units and the set-point varies between 0 and 50 units. Fig.2.4 shows a series of set-point responses for incremental GPC with $NU=N_1=1$, $NY=10$, $\lambda=1e\text{-}8$, $Q(z^{-1})=1$, $T(z^{-1})=1$ and an exact discrete-time model of (2-39). Initially, when $P(z^{-1})$ is set to unity, overshoot occurs due to actuator saturation (typical of the application described in chapters 3 and 5).

A reasonable initial estimate of the polynomial $P(z^{-1})$ required to remove the overshoot comes from the maximum bandwidth interpretation given above. From the first set-point change the maximum possible settling time that may be expected from the closed-loop, given the available actuation, is approximately 50 samples (ie. roughly twice the time taken for full actuation to achieve half the set-point). Choosing $1/P(z^{-1})$ to itself have a settling

time of 50 samples ie. a time constant of approximately 50/3 samples* ($\alpha = e^{-3/50} = 0.94$) means that if GPC was configured for model-following minimum-variance it should have adequate actuation to obtain the desired closed-loop $1/P(z^{-1})$ without overshoot. Since the GPC currently specified with $P(z^{-1})$ is detuned from model-following minimum-variance the overshoot, given this choice of $P(z^{-1})$, would be expected to be reduced. Intuitively this choice of $P(z^{-1})$ should be conservative.

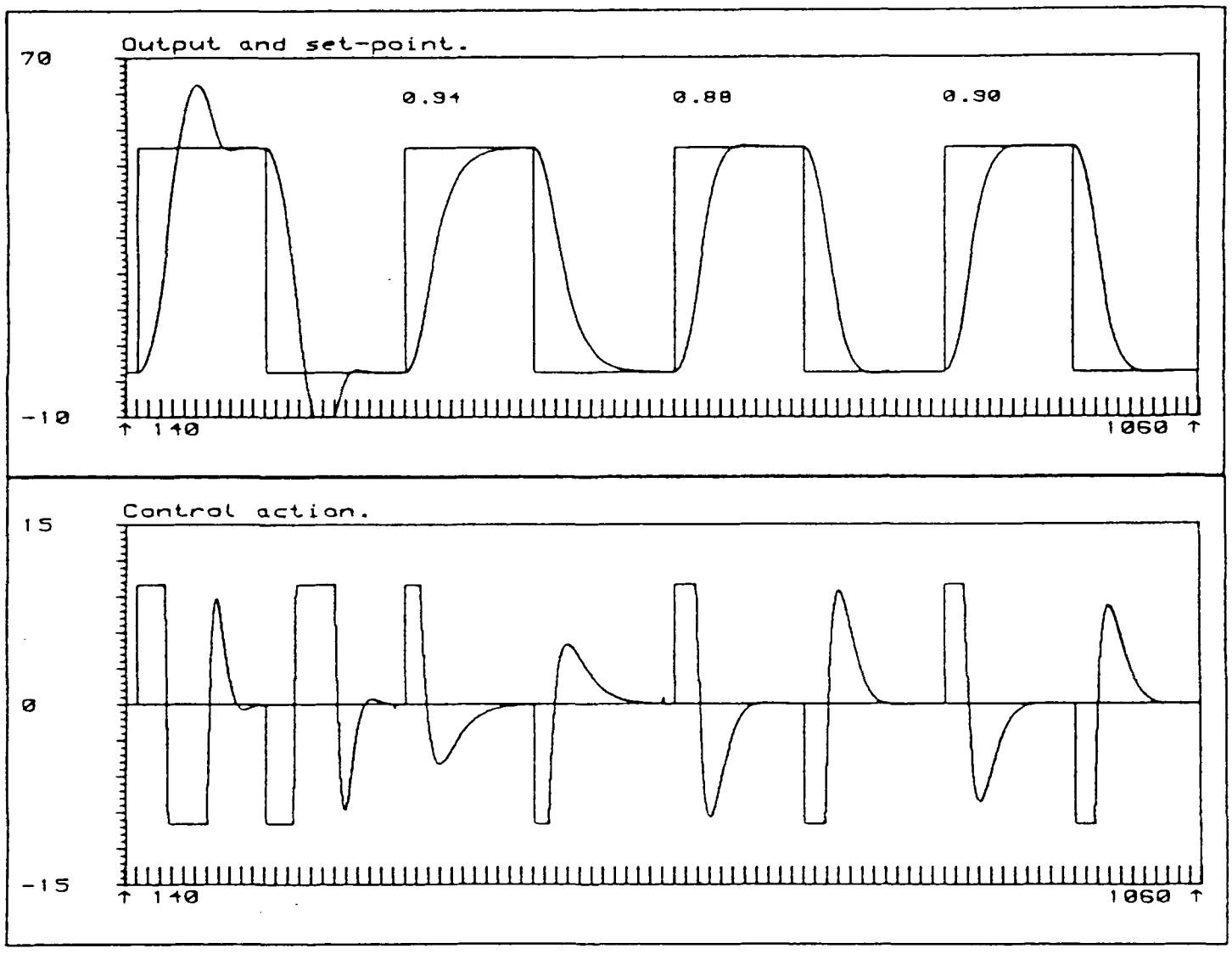


Fig.2.4: Choice of $P(z^{-1})$

The second set-point response indeed has no overshoot but inspection of the control signal indicates that, if actuation is cheap, a faster closed-loop is possible. A reasonable choice for a new time-constant of $P(z^{-1})$ would be, say, 8 samples ($\alpha = 0.88$) roughly half-way between 0 and 50/3. This results in the third set-point response which has a minute overshoot. Further

* T in the continuous-time equivalent $1/P(s)=1/(1+sT)$.

fine-tuning is of course possible; for example the fourth set-point response chooses $\alpha = 0.9$.

2.6.2 The design filter $\lambda Q(z^{-1})$

When $\lambda Q(z^{-1})$ is utilised GPC imposes a dynamic costing on the future control actions in the cost-function

$$J = \sum_{i=N_1}^{NY} (\hat{y}(t+j|t)-w(t+j))^2 + \sum_{j=1}^{NU} \lambda(i)[Q(z^{-1})u(t+j-1)]^2 \qquad (2\text{-}40)$$

The d.c. gain of the filter $\lambda Q(1)$ is set to zero to avoid the problem of λ-offset mentioned in section 2.5.1. A simplified structure for $\lambda Q(z^{-1})$ is described below and guidelines for its choice, illustrated by a simulation example, are discussed.

Simplified $\lambda Q(z^{-1})$

While theoretically $Q(z^{-1})$ could be chosen to give a frequency-dependent penalisation on future controls in the GPC cost (2-31), say to avoid exciting a system resonance, in practice it is chosen simply as

$$J = \sum_{i=N_1}^{NY} (\hat{y}(t+j|t)-w(t+j))^2 + \sum_{j=1}^{NU} \lambda[\Delta(z^{-1})u(t+j-1)]^2 \qquad (2\text{-}41)$$

ie.
$$\lambda(i)Q(z^{-1}) = \lambda(1-z^{-1}) \qquad (2\text{-}42)$$

The reasons for this simplification are:

- The lack of any quantitative interpretation for the effect of $\lambda Q(z^{-1})$ argues against a complex structure.

° The effects, in simulation, of variations in higher-order choices of $\lambda Q(z^{-1})$ are much reduced in GPC compared with the role $\lambda Q(z^{-1})$ played in the precursing GMV control algorithm (Mohtadi, 1986).

° The implementation of a higher-order $\lambda Q(z^{-1})$ is both complex and computationally time-consuming (Mohtadi, 1986).

° Perhaps the main justification for this simplified structure is that it manages to provide an effective tuning-knob whose effect can be easily understood.

The actual value of λ used in the GPC control calculation is not chosen directly but via a scaling and an offset

$$\lambda = \lambda_{min} + \hat{B}(1)^2 \lambda_{rel} \qquad (2\text{-}43)$$

where λ_{min} is a 'small' number and λ_{rel} is the 'tuning-knob' chosen such that $\lambda \simeq \hat{B}(1)^2 \lambda_{rel}$. The implementation of this simplified choice has already been detailed in the section on incremental GPC control (2.5).

Interpretation of $\lambda Q(z^{-1}) = \lambda(1-z^{-1})$

° With reference to the cost-function (2-41) λ is principally chosen to trade-off future predicted errors against the control activity required to reduce them. Since λ weights control increments the high-frequency components of possible future controls are penalised to a greater extent than the low-frequencies.

° Alternatively, looking at the GPC control calculation

$$\tilde{\delta u} = (G_1^T G_1 + \lambda I)^{-1} G_1^T (w - \hat{y}_f) \qquad (2\text{-}44)$$

the weighting λ may be viewed as simply reducing the 'magnitude' of the matrix $(G_1^T G_1 + \lambda I)^{-1}$. Also if the initial prediction horizon N_1 is chosen to be less than k, the plant delay, the matrix $G_1^T G_1$ becomes singular (section 2.5.3) and a small value of λ is required to allow the inversion $(G_1^T G_1 + \lambda I)^{-1}$. This small value is always present due to the term λ_{min} in (2-43).

° To explain the scaling (2-43) consider the case where λ_{rel}, and hence λ, has been adjusted to give a desired level of control activity. Assume that during the period of control the plant gain changes and that, ideally, the parameter estimator tracks this change. Since the plant gain has changed different levels of future control activity arise which, however, give approximately the same levels of error between set-point and predictions in the cost (2-41). Thus, if λ remains at its previous value, it will tend to either over-penalise or under-penalise control activity after the change. The simple scaling $\lambda = \lambda_{rel}\hat{B}(1)^2$ suggested by Lam (1980) attempts to make the effect of λ independent of changes in the plant gain. This scaling is an ad-hoc modification carried over from infinite-stage controller design and really assumes that changes in gain will be manifested as changes in $\hat{B}(1)$ rather than $\hat{B}(1)/\hat{A}(1)$.

Choice of λ_{rel}

To illustrate the choice of λ_{rel} consider the requirement to reduce actuator activity in the control of the system

$$G(s) = \frac{1}{1 + 5s} \qquad (2\text{-}45)$$

where the set-point varies between 0 and 50 units. Fig.2.5 shows a series of set-point responses for incremental GPC, $\lambda Q(z^{-1}) = \lambda(1-z^{-1})$, with $NU=N_1=1$, $NY=5$, $P(z^{-1})=T(z^{-1})=1$ and the exact discrete-time model of (2-45) assumed to

be known. Assume that for reasons of economy or actuator wear it is desired to reduce the level of control activity from that shown in the first set-point response where $\lambda_{rel}=0$, $\lambda=\lambda_{min}=1e-8$ (ie. effectively zero).

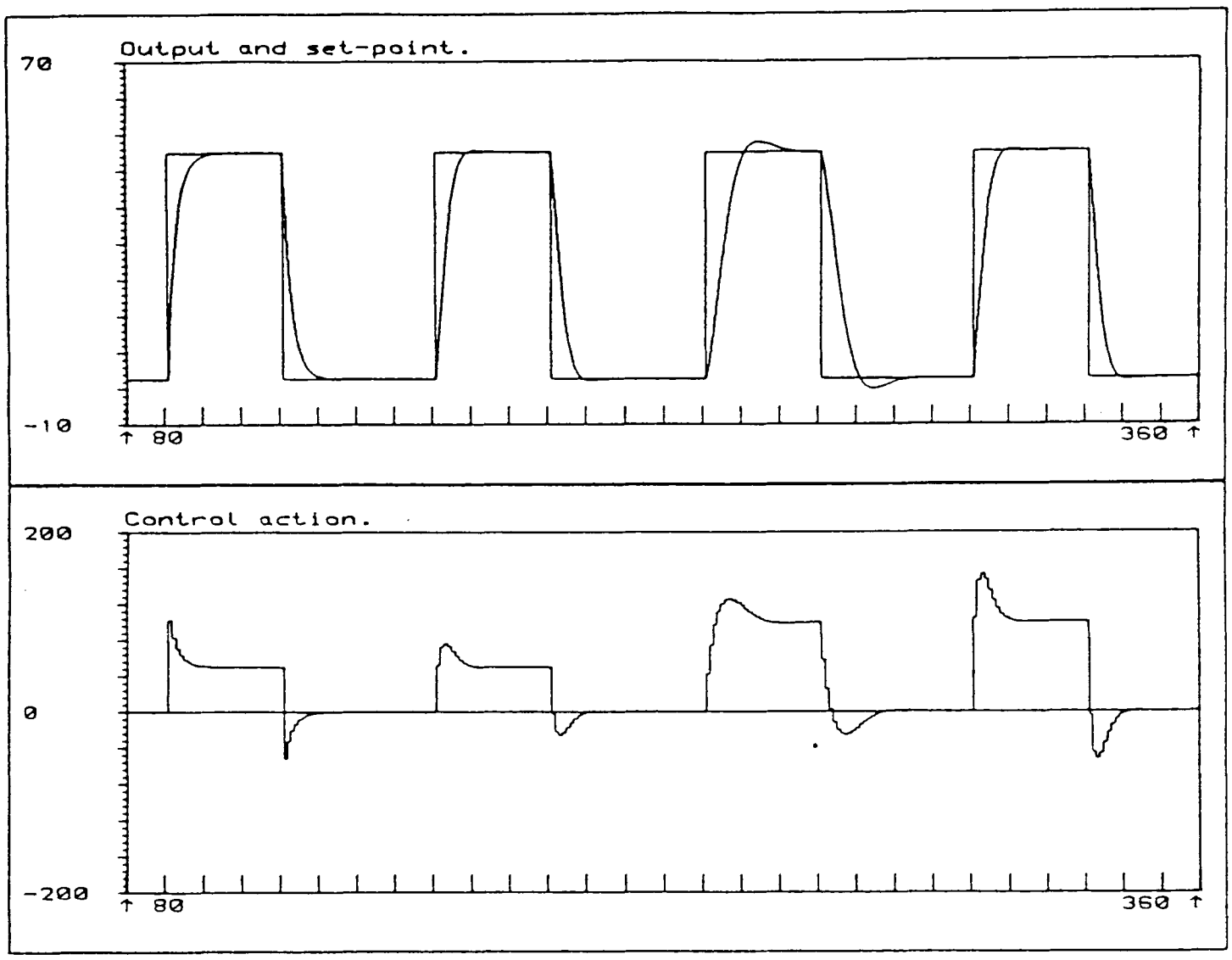


Fig.2.5: Choice of λ_{rel}

A reasonable initial estimate for λ_{rel} comes from the expression for the control (2-44). If λ is set equal to $G_1^T G_1$ then the initial control increment should be exactly halved and subsequent controls should also be detuned (although not by half) from the first case. Since $G_1^T G_1$ is available this requirement becomes $\lambda_{min} + \hat{B}(1)^2\lambda_{rel} \simeq \hat{B}(1)^2\lambda_{rel} = G_1^T G_1 \simeq 1$ ie. $\lambda_{rel} = 1/\hat{B}(1)^2 \simeq 30$. The second set-point response is indeed detuned as predicted and it is interesting to note that the trade-off between reducing control activity and the deterioration in closed-loop performance is quite favourable. As has often been noted in the literature (Moden & Soderstrom,

1978) a quite large reduction in control variance often results in minimal deterioration in the closed-loop.

The effect of failing to scale λ in the presence of gain-changes is investigated in the third set-point response where the gain of the system decreases by a factor of 2 (the estimates set to follow exactly) and λ is held constant at $\lambda = 1$. Since GPC must exert larger control actions for much the same errors between predictions and the set-point the old value of $\lambda = 1$ over-penalises the control in the cost resulting in an overshoot in the third set-point response. In the fourth set-point response the scaled value of λ is used ie. $\lambda=\lambda_{rel}\hat{B}(1)^2=30x0.008=0.25$ which reduces control variance with little 'loosening' of the closed-loop, the resultant control variance being similar to the second set-point response despite the different amplitudes arising from the plant gain-change.

2.6.3 The design filter $T(z^{-1})$

The filter $T(z^{-1})$ is a tuning-knob which may be incorporated into both the estimation and control calculations. While its effects are immediate and intuitively acceptable it is the most difficult design filter to interpret analytically. Loosely speaking $T(z^{-1})$ is used to restrict the bandwidth over which the GPC controller attempts to predict the future and over which the estimator attempts to model the underlying plant. This restriction reduces the sensitivity of the overall self-tuning algorithm to load disturbances acting upon the process and to the net modelling errors arising from neglected dynamics, estimation errors and non-linearities. This makes it a vital tuning-knob which, especially for incremental estimation and control, can mean the difference between stability and instability in self-tuning GPC.

The required modifications to the parameter estimator and to the GPC control calculation are detailed below with heuristic interpretations for the effect of $T(z^{-1})$ on parameter estimation, on the predictor and its joint effect when applied to both estimator and predictor. Drawing on these interpretations a simplified structure for $T(z^{-1})$ is suggested and guidelines given for the choice of its parameters with reference to a simulation example. Further examples of the effect of $T(z^{-1})$ may be seen in the results of chapter 5.

$T_e(z^{-1})$ - Estimator design polynomial incorporation

The design filter $T_e(z^{-1})$ simply filters the states of the linear regression (1-8) used in the least-squares estimation algorithm of section 1.1.3. For example, in the incremental case, the regression

$$\Delta y(t) = z(1-\hat{A}(z^{-1}))\Delta y(t-1) + \hat{B}(z^{-1})\Delta u(t-1) \qquad (2\text{-}46)$$

becomes

$$\frac{\Delta y(t)}{T_e} = z(1-\hat{A}(z^{-1}))\frac{\Delta y(t-1)}{T_e} + \hat{B}(z^{-1})\frac{\Delta u(t-1)}{T_e} \qquad (2\text{-}47)$$

the filtered measurements being regressed against filtered 'Δy's and 'Δu's.

$T_c(z^{-1})$ - Controller design polynomial incorporation

The design filter $T_c(z^{-1})$ may again be introduced by a modification to the diophantine equation (2-25), in the incremental case

$$T_c = \bar{E}_j A\Delta + z^{-j}\bar{F}_j \qquad (2\text{-}48)$$

which gives the prediction equation

$$y(t+j) = \bar{E}_j B\frac{\Delta u(t+j-1)}{T_c} + \bar{F}_j\frac{y(t)}{T_c} + \bar{E}_j\frac{\Delta x(t+j)}{T_c} \qquad (2\text{-}49)$$

The predictions, $\hat{y}(t+j|t) = \bar{E}_j B\frac{\Delta u(t+j-1)}{T_c} + \bar{F}_j\frac{y(t)}{T_c}$, are then used to minimise the original cost (2-31). However, as before, the diophantine equation is not actually solved in practice. Instead the filtered model

$$A(z^{-1})\frac{\Delta y(t)}{T_c} = B(z^{-1})\frac{\Delta u(t-1)}{T_c} \qquad (2\text{-}50)$$

is iterated to obtain the predictions. The equivalence of the predictions obtained from this approach and those obtained from the solution of the diophantine equation is proven in appendix A.3. It is also shown that the g_i elements of the matrix G_1 in the GPC control calculation (2-32) remain the same (as without T_c) whereas the vector $\hat{y}_f$, whose elements are the predicted 'free' response, must be obtained by iterating (2-50) from the current filtered state $\{\Delta y(t)/T_c, \Delta y(t-1)/T_c, \ldots, \Delta u(t-1)/T_c, \ldots\}$ with zero future control increments (passing through the filter $1/T_c$) applied. The resulting vector of full-valued filtered predictions $\{\hat{y}(t+1|t)/T_c, \hat{y}(t+2|t)/T_c, \ldots, \hat{y}(t+NY|t)/T_c\}$ must then be 'deconvolved' to obtain predictions of the actual output ie.

$$\hat{y}(t+1|t) = t_0\hat{y}'(t+1|t) + t_1 y'(t) + \ldots + t_{nt} y'(t-nt+1)$$
$$\hat{y}(t+2|t) = t_0\hat{y}'(t+2|t) + t_1\hat{y}'(t+1|t) + \ldots + t_{nt} y'(t-nt+2)$$
$$\vdots$$

where t_i are the coefficients of the filter $T_c(z^{-1})$ and the prime denotes filtered quantities eg. $\hat{y}'(t+1|t) = \hat{y}(t+1|t)/T_c$.

Interpretation of $T_e(z^{-1})$

Consider the case where the least-squares algorithm is being used to fit the low-order incremental model

$$\hat{A}(z^{-1})\frac{\Delta y(t)}{T_e} = \hat{B}(z^{-1})\frac{\Delta u(t-1)}{T_e} + e_T(t) \qquad (2\text{-}51)$$

to a record of input/output data generated by an actual higher-order system

$$A_0(z^{-1})y(t) = B_0(z^{-1})u(t-1) + \eta(t) \qquad (2\text{-}52)$$

where $\eta(t)$ is due to unmeasurable disturbances, non-linearities etc. This situation is illustrated schematically in Fig.2.6. The least-squares algorithm (in the absence of forgetting*) attempts, given a record of past 'y's and 'u's, to minimise $\sum_{t_0}^{t} e_T^2(i)$ by choosing $\hat{A}$ and $\hat{B}$ in (2-51). The effect of $T_e(z^{-1})$ can only be compared with its absence; for example positional least-squares would attempt to fit the model

$$\hat{A}(z^{-1})y(t) = \hat{B}(z^{-1})u(t-1) + e(t) \qquad (2\text{-}53)$$

by choosing $\hat{A}$ and $\hat{B}$ to minimise $\sum_{t_0}^{t} e^2(i)$. Comparing the two least-squares costs

without T_e (positional): $$J = \sum_{t_0}^{t} e^2(i) \qquad (2\text{-}54)$$

with T_e (incremental): $$J = \sum_{t_0}^{t} e_T^2(i) = \sum_{t_0}^{t} \left[\frac{\Delta e(i)}{T_e}\right]^2 \qquad (2\text{-}55)$$

* see section 1.1.3.

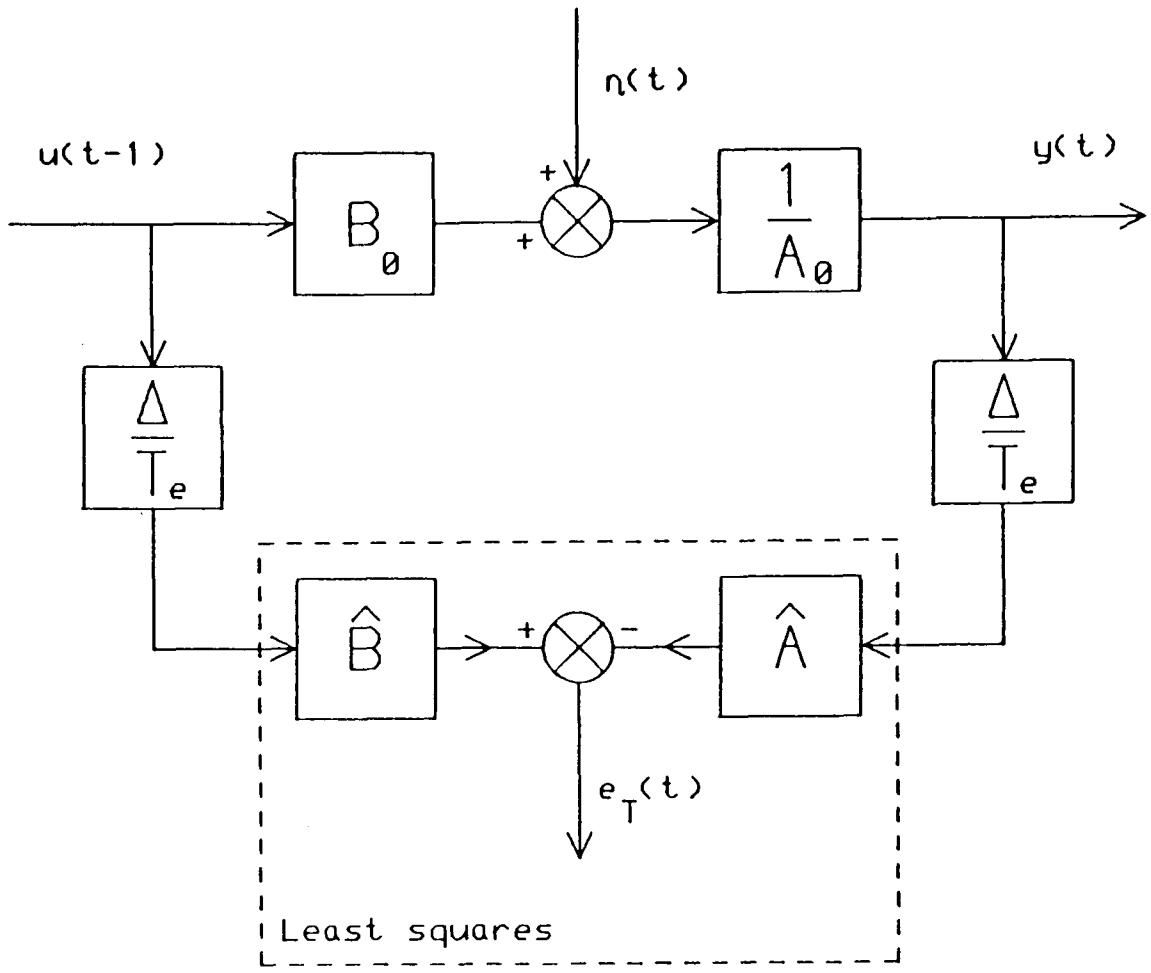


Fig.2.6: Incremental estimation with $T_e(z^{-1})$

The effect of Δ/T_e is therefore equivalent to fitting the unfiltered model (2-53) by minimising the filtered cost (2-55) ie. Δ/T_e provides a frequency-dependent weight (see Fig.2.8) on the least-squares fit. This can be used to allow a low-order model to more accurately represent a complex plant over a restricted frequency range, within the Nyquist limit, at the possible expense of increased inaccuracy outside that range. It can also be used to reduce the sensitivity of the estimator to transient load disturbances occurring during the period of estimation which have frequency components outside the frequency range where the plant has significant gain.

Interpretations of $T_c(z^{-1})$

° The effect of the filter $T_c(z^{-1})$ is seen to be independent of scaling (as is T_e) by considering the scaled diophantine equation

$$kT_c = k\bar{E}_j A\Delta + z^{-j}k\bar{F}_j \qquad (2\text{-}56)$$

where k is an arbitrary scalar. The effect of T_c (see later) always derives from the polynomial ratios $k\bar{E}_j/kT_c$ and $k\bar{F}_j/kT_c$ which are independent of the scaling k. Nevertheless in the following it is assumed, for simplicity, that $T(1)=1$ even though the above shows this assumption to be unnecessary.

° Incremental GPC is based upon using a plant model

$$A(z^{-1})\Delta y(t) = B(z^{-1})\Delta u(t-1) + \Delta x(t) \qquad (2\text{-}57)$$

to make predictions of the future behaviour of the plant under control. The coefficients of $A(z^{-1})$ and $B(z^{-1})$ in this model come from the <u>incremental</u> least squares estimator (see section 2.7) and the term $\Delta x(t)$ represents the net effect of plant-model-mismatch (ie. unmodelled dynamics, estimation errors and non-linearities) and unmeasurable load disturbances acting upon the actual system. If $\Delta x(t)$ is always zero then $T_c(z^{-1})$ does not have an effect since from (2-59), regardless of its value, the predictions will always be the same.

° When $T_c(z^{-1})$ is <u>not used</u> the incremental Clarke-Gawthrop predictor (sections 2.3.1 and 2.5.3) splits the disturbance term into

$$\frac{1}{A(z^{-1})}x(t+j) = E_j(z^{-1})\Delta x(t+j) + \frac{F_j(z^{-1})}{A(z^{-1})}x(t) \qquad (2\text{-}58)$$

where $E_j\Delta x(t+j)$ represents the effect of future error increments (since E_j is of order j-1) and $F_j x(t)/A$ represents the effect of current and past errors. This expression (2-58) is, of course, the source of the diophantine equation. The predictor, from (2-57) and (2-58), is then

$$\hat{y}(t+j|t) = E_j B\Delta u(t+j-1) + F_j y(t) = \frac{B}{A}u(t+j-1) + \frac{F_j}{A}x(t) \qquad (2\text{-}59)$$

° When $T_c(z^{-1})$ is used, as seen from the modified diophantine (2-48), the disturbance term is split as

$$\frac{1}{A(z^{-1})}x(t+j) = \bar{E}_j\frac{\Delta x(t+j)}{T_c} + \frac{\bar{F}_j x(t)}{AT_c} \qquad (2\text{-}60)$$

where $\bar{E}_j\Delta x(t+j)/T_c$ now represents the effect of future filtered error increments and $\bar{F}_j x(t)/AT_c$ represents the effect of current and past filtered errors. The resultant predictor, from (2-57) and (2-60), is

$$\hat{y}(t+j|t) = \bar{E}_j B\frac{\Delta u(t+j-1)}{T_c} + \frac{\bar{F}_j y(t)}{T_c} = \frac{B}{A}u(t+j-1) + \frac{\bar{F}_j x(t)}{AT_c} \qquad (2\text{-}61)$$

Therefore by comparing the latter expressions in (2-59) and (2-61) $T_c(z^{-1})$ determines, admittedly in a non-straightforward fashion, the manner in which the Clarke-Gawthrop predictive scheme incorporates the effect of past errors/disturbances {x(t), x(t-1), ...} into the predictions.

Assuming that $1/T_c$ is chosen low-pass then over the low-frequency range where $T(e^{-j\omega h}) \simeq 1$, from the diophantine equation (2-48), the frequency response of the polynomials E_j, F_j and $\bar{E}_j$,$\bar{F}_j$ ie. with and without T_c will be approximately the same. For higher frequencies the effects of F_j and $\bar{F}_j/T_c$ in (2-59) and (2-61) are difficult to compare since F_j and $\bar{F}_j$ depend in a complex way on A and T_c. Moreover if the coefficients of $A(z^{-1})$ are obtained

from an on-line estimator neither F_j nor $\bar{F}_j$ can be calculated beforehand thus making a meaningful prior choice of $T_c(z^{-1})$ more difficult. However when $1/T_c$ is chosen to be low-pass it is generally found that beyond the breakpoint frequency of $1/T_c$ the magnitude of the frequency response of $\bar{F}_j/T_c$ is much decreased from that of F_j.

As an example consider an estimated model obtained from the compliant arm application of section 3.1. Fig. 2.7 shows the frequency response of the model, the filter Δ/T_c and the comparative frequency responses of F_j and $\bar{F}_j/T_c$ which satisfy the incremental diophantine equations (2-25) and (2-48) for the range of prediction horizons j=1,4,7 and 10. Solid lines represent the gain of F_j while dotted lines refer to $\bar{F}_j/T_c$. The axes are log-log over the Nyquist frequency range $\omega h = 0.01$ to π rads. Clearly the inclusion of T_c reduces the sensitivity of the predictions to x(t) over a high-frequency band which broadens with increasing j.

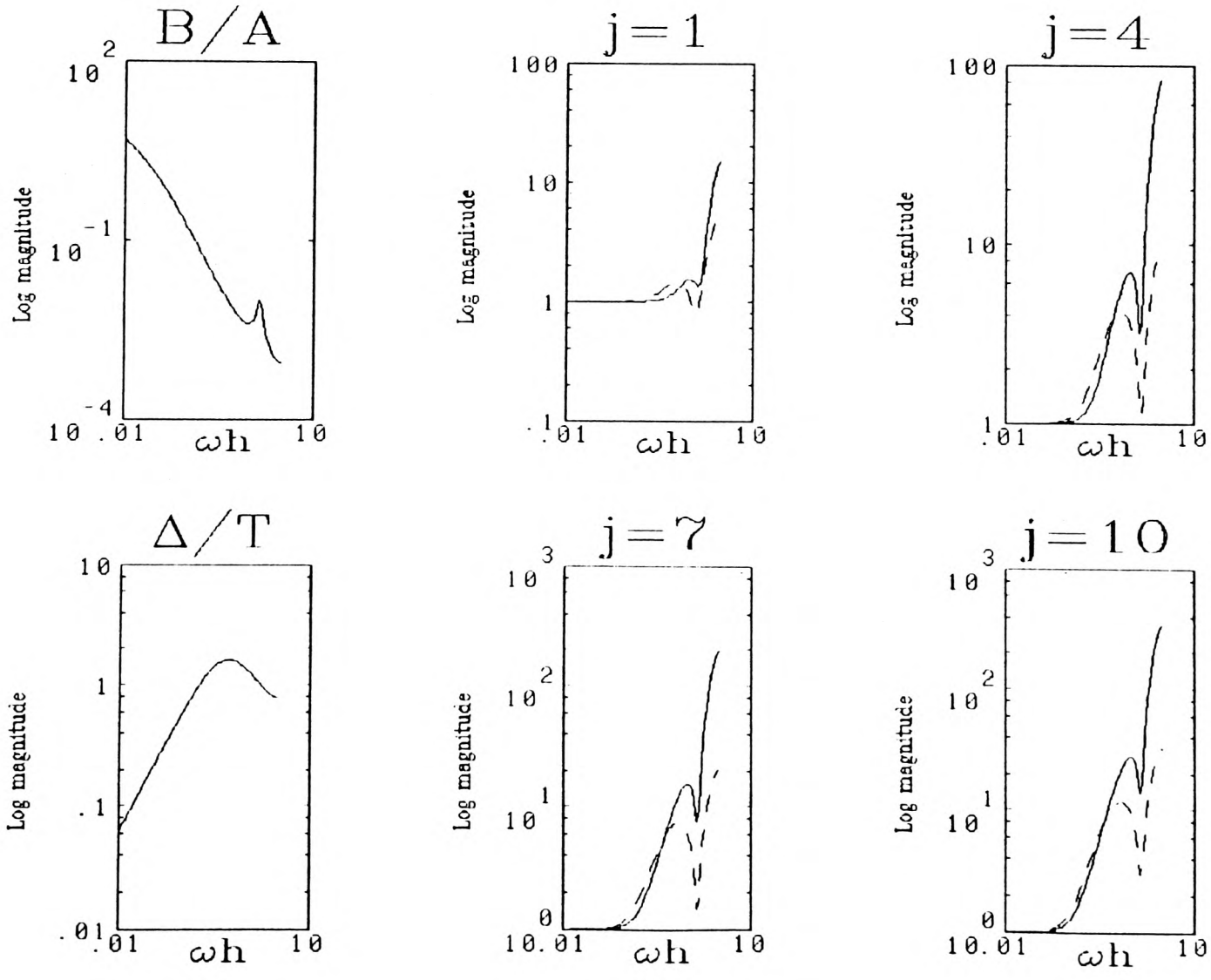


Fig.2.7: Relative sensitivity of predictors to x(t)

Interpretations for the joint effect of $T_e(z^{-1})$ and $T_c(z^{-1})$

From (2-51) incremental estimation with the design filter $T_e(z^{-1})$ yields a model

$$y(t) = \frac{B}{A}u(t-1) + \frac{T_e}{A\Delta}e_T(t) \qquad (2\text{-}62)$$

where the least-squares algorithm has minimised $\sum_{t_0}^{t} e_T^2(i)$. According to Ljung & Soderstrom (1983), this model (2-53 or 2-62) may be interpreted as the 'best' single-step-ahead predictor for $\Delta y(t)/T_c$ given filtered regressors. The choice, for the predictor, of $T_c = T_e$ splits the infinite error sequence $T_e e_T(t)/A\Delta$ into future and past components by defining the diophantine equation

$$\frac{T}{A\Delta} = \bar{E}_j + \frac{z^{-j}\bar{F}_j}{A\Delta} \qquad (2\text{-}63)$$

which yields the predictor

$$\hat{y}(t+j|t) = \frac{B}{A}u(t+j-1) + \frac{\bar{F}_j}{A\Delta}e_T(t) \qquad (2\text{-}64)$$

This prediction is optimal in the sense that the prediction error $\bar{E}_j e_T(t+j)$ depends only on future errors $e_T(t+j)$ which are assumed to be unpredictable. Since the least-squares algorithm with $T_e(z^{-1})$ endeavours to make $e_T(t)$ small (in the mean-square sense) the choice $\Delta/T_c = \Delta/T_e$ seems to match the estimated single-step-ahead predictor to the de Keyser & van Cawenberghe multi-step-ahead predictor.

Choice of $T(z^{-1})$

The choice of $T(z^{-1})$ will only be described in the context of incremental estimation and control because, firstly, incremental GPC is now used exclusively in practice to avoid the offset problems of section 2.5.1 and, secondly, because the stabilising effect of $T(z^{-1})$ is found to be more important for incremental controllers (see section 5.2.1).

Simplified structure for $T(z^{-1})$

Bearing in mind that, in the incremental case, the effect of the disturbance x(t) is felt through the transfer-function $\Delta(z^{-1})/T(z^{-1})$ it is proposed to specify Δ/T as a stable second-order band-pass filter

$$\frac{\Delta}{T} = \frac{1-z^{-1}}{(1-\beta z^{-1})^2} \qquad (2\text{-}65)$$

with repeated roots at $z=\beta$. This choice has the type of frequency response shown in Fig.2.8. The estimator polynomial $T_e(z^{-1})$ is chosen equal to the controller polynomial $T_c(z^{-1})$ for the reasons outlined above. The principal reasons for this simplified structure for $T(z^{-1})$ are

- The imprecision in the interpretations of the effect of T_e and T_c argues, as with the design filters $P(z^{-1})$ and $\lambda Q(z^{-1})$, against a more sophisticated choice of structure for $T(z^{-1})$. In addition, this structure and the choice of the parameter β has been found, both in simulation and in practice, to be effective and relatively simple.

- A second-order $T(z^{-1})$ allows a band-pass interpretation (where a first-order $T(z^{-1})$ would be high-pass) for Δ/T which is commensurate with the

restricted modelling interpretation of $T_e(z^{-1})$. The attenuation of the effect of d.c. offsets, low and high-frequency disturbances has proven to be necessary in this and parallel applications (Lambert, 1987; Al-Assaf, 1987). Higher order choices of $T(z^{-1})$ have yet to be found necessary in any reported application.

Choice of β

° Unlike the $P(z^{-1})$ and $\lambda Q(z^{-1})$ design filters, used to fine-tune performance once closed-loop stability has been achieved, the $T(z^{-1})$ filters must often be specified beforehand in order to stabilise the closed-loop. Ideally it should be chosen such that if destabilising disturbances x(t) are present (ie. due to periodic load disturbances or unmodelled dynamics) it will reduce their effect whereas if they are not present (ie. high signal-noise ratio, correct model order) it should make little difference to the resultant closed-loop. Of course it is preferable to err on the side of caution as once stability has been achieved the choice of $T(z^{-1})$ may subsequently be relaxed (ie. $T \rightarrow 1$).

Choosing a default value for β is complicated by the fact that the effect of $T(z^{-1})$ depends upon the frequency response of the plant, assumed unknown beforehand. However a simple choice for β may be obtained from the following heuristic argument.

° The sample period, h, of a discrete-time controller is conventionally chosen to be approximately a tenth (Goodwin & Sin, 1984) of the plausible settling-time of the closed-loop (hence the default choice NY=10). The desired closed-loop is then very approximately

$$G_c(s) \simeq \frac{1}{1+s(10h/3\ )} \qquad (2\text{-}66)$$

with a breakpoint at $0.33/h$ rads^{-1}. To obtain this closed-loop GPC must be able to predict accurately over this bandwidth at least. However at very low frequencies the incremental estimator does not even attempt to model the plant, leaving it up to the inherent integral action of the incremental GPC controller to ensure offset-free control. Disturbances at frequencies beyond those of the desired closed-loop should be attenuated in their effect on the estimator (causing a poor model) and on the predictions (causing excessive and destabilising high frequency control actions). The default choice of $\beta = 0.8$, placing the double break-point at $0.22/h$ rads^{-1}, has been adopted as a sensible default. The effect of this choice, with reference to the above, is shown in Fig.2.8 where the solid line gives the gain with frequency of $\Delta/T(z^{-1})$ while the dotted line shows $|G_c(j\omega)|$.

It must be emphasised that while the default choice $\beta = 0.8$ has been found satisfactory as an initial stabilising choice fine tuning is possible with $\beta \to 0$ speeding the load-disturbance rejection (and in the presence of unmodelled dynamics speeding the set-point response).

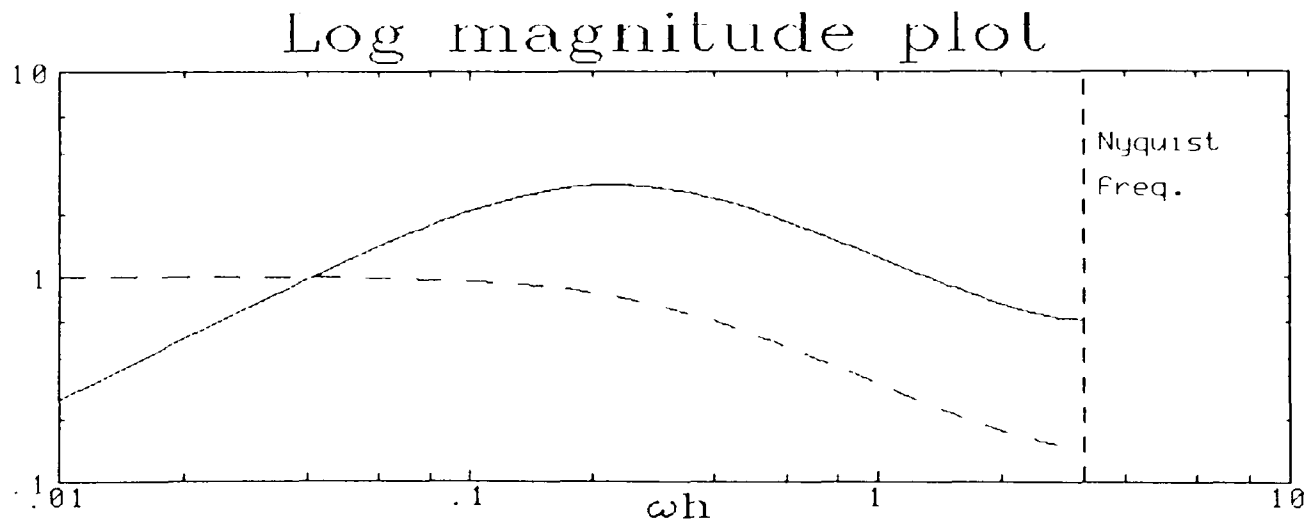


Fig.2.8: Frequency response of $\Delta/T(z^{-1})$

o To illustrate the foregoing arguments consider the requirement to

stabilise the system

$$G(s) = \frac{1}{s(1+s)} \qquad (2\text{-}67)$$

which is to be sampled at 1Hz with a first-order estimator model. This example, while not a difficult control problem in itself, is intended to illustrate the effect of $T(z^{-1})$ on the control of a plant with neglected dynamics which is being sampled somewhat slowly (ie. a much simplified version of the real compliant-arm system described in section 3.1). Fig.2.9 shows a series of set-point responses with incremental GPC $NU=N_1=1$, $NY=5$, $\lambda=1e\text{-}8$, $P(z^{-1})=1$. On each rising set-point change the estimated parameter set is zeroed and the covariance matrix of the estimator is set to 10I to allow quick parameter adaption. After the tuning transient a steady 'dcy' disturbance of 10 units

$$y(t) = G(s)u(t) + dcy(t) \qquad (2\text{-}68)$$

is first applied and then removed. Then the 'tuned' set-point response may be observed on each following downward-going set-point change. Initially $T = (1-0.8z^{-1})^2$ and GPC tunes, on the basis of its first-order model, to give stable if sluggish control. The 'shelving' nature of the response is typical for model-based controllers in the presence of neglected dynamics. Once stable control has been achieved the choice of T may be relaxed in order to speed up the set-point response (in the presence of neglected dynamics) and the load-disturbance rejection (principally in the absence of neglected dynamics). This is illustrated in the second set-point response where GPC retunes with $T = (1-0.7z^{-1})^2$. This sudden retuning is meant to show the eventual result of a slow retuning in practice due to some forgetting scheme (Wittenmark & Astrom, 1984) maintaining estimator adaptivity. The control response to load

disturbances is slightly more active but the 'tuned' response to the downward-going set-point change is far quicker.

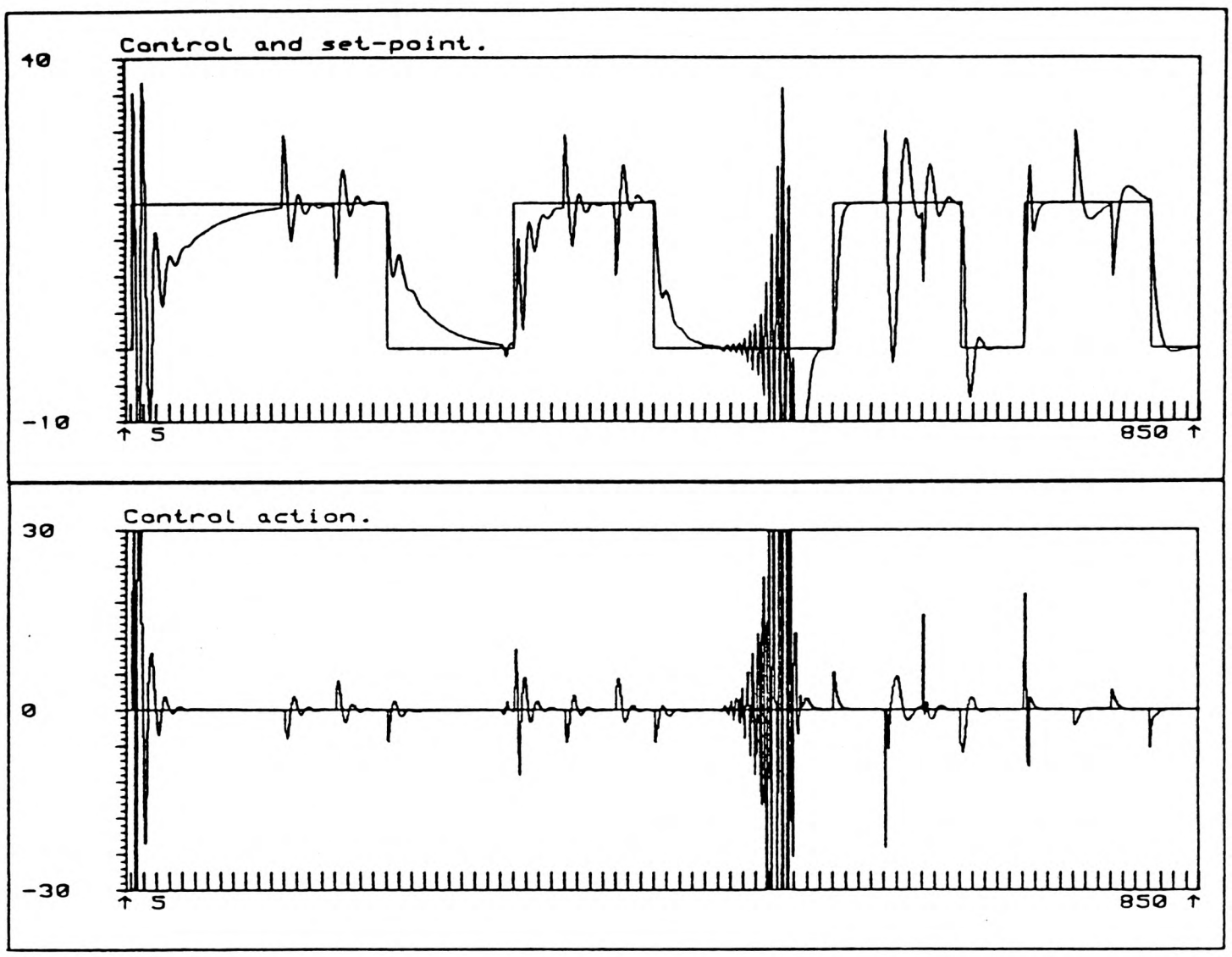


Fig.2.9: Choice of $T(z^{-1})$

The period of instability resulted from removing T (ie. T=1) and reinitialising the estimator. Thus GPC, without T, is incapable of stabilising this loop without resorting to a 'large' λ and a massively detuned closed-loop. Stability is restored by reinitialising the estimator, leaving T = 1 but re-estimating a model of the correct order. On the third upward-going set-point change the estimates are exact due to the large amount of information gained while returning the plant to stability and the response is excellent. However the effect of the 'dcy's is great; the control is more active than previously and the parameter estimates are damaged causing very poor disturbance rejection and a deteriorated downward-going set-point

response. The final set-point changes are with exact parameterisation and $T = (1-0.8z^{-1})^2$ to show that, in the absence of neglected dynamics, T only affects load disturbance rejection. The resulting disturbance rejection is similar to the under-parameterised case and the 'dcy's' effect on the estimator is reduced, the downward-going set-point response having only a slight overshoot.

2.7 Adaptive GPC

The derivation of GPC given in the previous sections assumes that the a_i, b_i coefficients of the plant model (2-3 or 2-24) are known. Of course these parameters will rarely be known beforehand and are likely to change during the working life of the controller eg. component wear. Consequently some form of identification scheme must be used to obtain estimated plant coefficients $\hat{a}_i$ and $\hat{b}_i$.

The identification algorithm most commonly used with GPC is the least squares method described in section 1.1.3. The least squares estimates $\hat{a}_i$ and $\hat{b}_i$ are simply substituted directly into the foregoing control calculation in place of a_i and b_i. This direct substitution, ignoring the fact that these parameters are not exact, is known as the **certainty equivalent** approach (Jacobs, 1981). However, the conclusions of sections 2.5 and 2.6.3 advocate the use of incremental RLS, with the differenced data low-pass filtered by a second-order $T_e(z^{-1})$, matched with the incremental GPC control law incorporating the design filter $T_c(z^{-1}) = T_e(z^{-1})$. This should ensure stable and accurate control.

2.8 Implementation of GPC

The sample-by-sample implementation of the incremental GPC controller is summarised in the flow-chart of Fig.2.10.

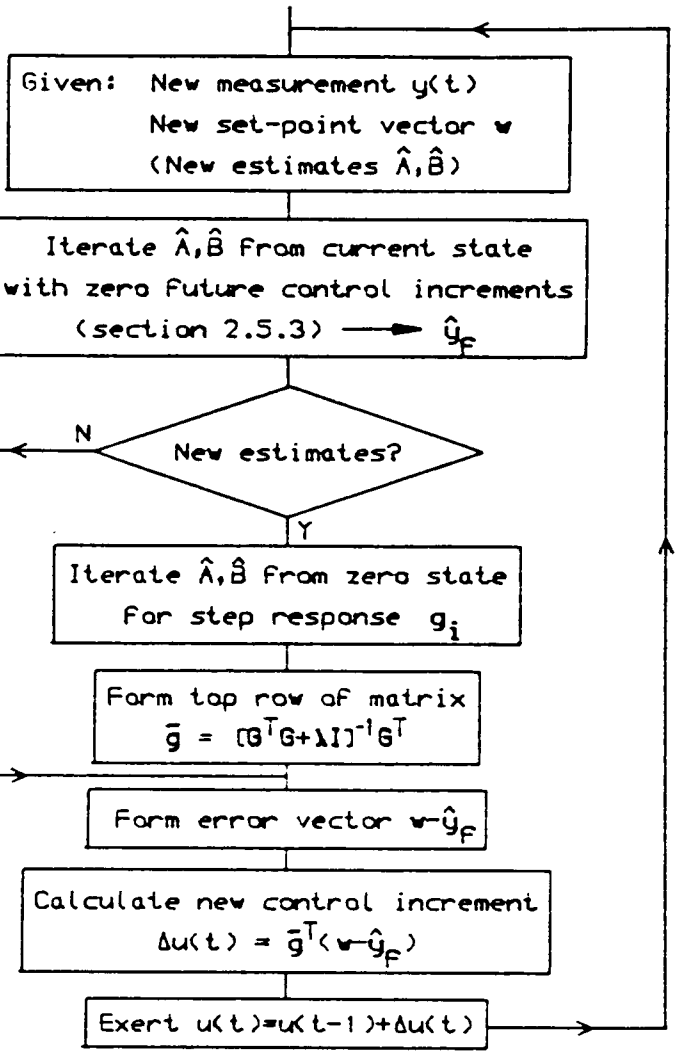


Fig.2.10: The GPC algorithm

The flow-chart assumes that, if configured as a self-tuner, new estimates are available from a separate estimation task. Details of the inclusion of this estimation task are given in the software section of chapter 4.

Chapter 3

INCLUSION OF PRIOR KNOWLEDGE

The preceding chapters describe the recursive least squares estimator and the GPC control law which together give the GPC self-tuner. This chapter addresses the problem of commissioning the estimator in such a way that it incorporates the type of prior knowledge of plant dynamics commonly available in practice. This might seem to contradict a fundamental objective of self-tuning control (Kalman, 1958) which is to minimise the amount of prior commissioning information required. However, when this data is available it is shown here, in simulation and experiment, that if properly incorporated it can result in improved convergence characteristics and a lessened sensitivity to noise and load disturbances. It can also lead to a reduced computational workload for the estimator.

Since an accurate mathematical model (Fraser, 1987) has been derived for the compliant arm of section 4.1, the details of which are summarised in section 3.1, this provides an excellent opportunity for a practical study of the benefits and drawbacks of the inclusion of prior knowledge. Sections 3.2 and 3.3 present two methods for its incorporation into the RLS estimator. Simulation examples are appended for plausible types of prior information eg. known gain, known time constant etc. Throughout these examples, bearing in mind the nature of the application, particular attention is paid to aspects pertaining to lightly damped flexible systems.

Much has been made recently (Goodwin et al, 1986; Edmunds, 1984; Peterka, 1986) of the potential benefits of moving from the z-transform description of discrete-time to alternatives such as Goodwin's δ-transform. Section 3.4 shows that estimation of models in δ may also have advantages in the context of the inclusion of prior knowledge. Section 3.5 presents a set

of experimental results derived from real compliant arm data which illustrate the effectiveness of the δ-transform and the methods of sections 3.2 and 3.4 in practice. The final section assesses the usefulness of these approaches, draws conclusions from the results presented and outlines useful routes for further research.

3.1 Mathematical Model of the Arm

Detailed analytical models for compliant arms similar to the type described in section 4.1 are widely available in the literature (eg. Cannon & Schmitz, 1984). The model obtained by Fraser (1987) is given below following a brief description of its derivation. This particular derivation is quoted since it leads to a linear Laplace transform transfer-function relating torque applied at the hub to tip displacement. Consider the schematic diagram of Fig.3.1

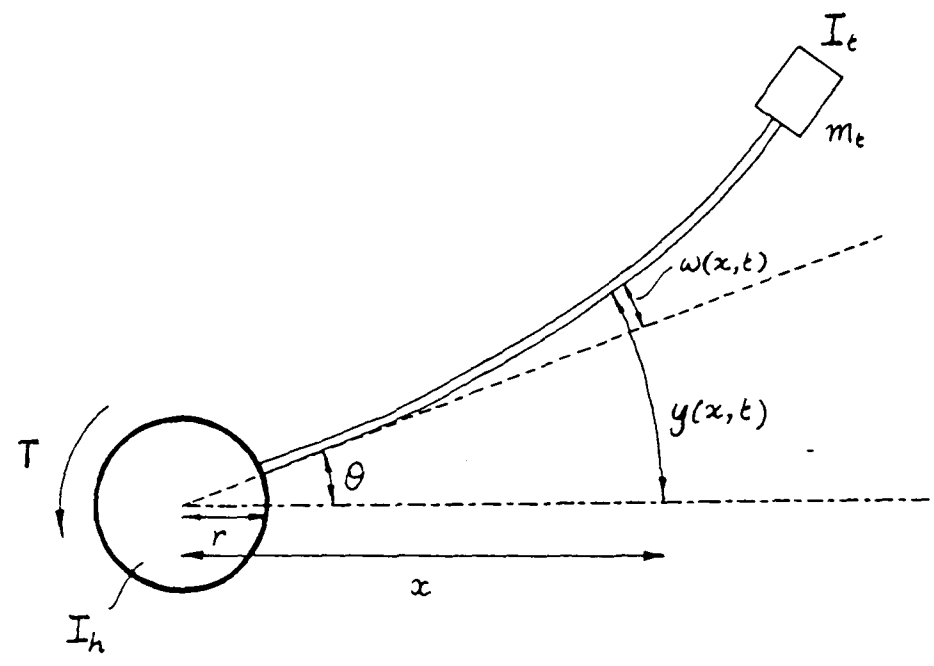


Fig.3.1: Schematic of the O.U. Compliant Arm

where the circumferential displacement $y(x,t) = w(x,t) + (x+r)\theta$ for small flexural deflections $w(x,t)$. The beam, being long and slender, is assumed to satisfy the requirements for Euler-Bernoulli beam theory. Fraser's derivation

uses Hamilton's Principle to obtain a partial differential equation governing the deflections along the beam

$$EI\frac{\partial^4 y}{\partial x^4} + m\frac{\partial^2 y}{\partial t^2} = 0 \qquad (3\text{-}1)$$

where EI is the flexural rigidity and m is the mass per unit length of the link. The effect of the hub driving torque T, the control input in this case, is felt through a bank of four boundary conditions for (3-1) ie.

$$y(0,t) - r\theta = 0 \qquad (3\text{-}2)$$

$$EI\frac{\partial^2 y(0,t)}{\partial x^2} + T - rEI\frac{\partial^3 y(0,t)}{\partial x^3} - I_h\frac{\partial^2 \theta}{\partial t^2} = 0$$

$$EI\frac{\partial^2 y(L,t)}{\partial x^2} + I_t\frac{\partial^3 y(L,t)}{\partial x \partial t^2} = 0$$

$$EI\frac{\partial^3 y(L,t)}{\partial x^3} - M_t\frac{\partial^2 y(L,t)}{\partial t^2} = 0$$

where L is the length of the link. Taking Laplace transforms at this stage converts (3-1) to an ordinary differential equation in s, the general solution of which is given by

$$\bar{y}(x,s) = A\sin\beta x + B\cos\beta x + C\sinh\beta x + D\cosh\beta x \qquad (3\text{-}3)$$

where $\beta^4 = -ms^2/EI$. The complex 'constants' A, B, C and D (themselves transcendental functions of β and hence s) may be evaluated by substituting (3-3) into the transformed boundary conditions obtained from (3-2). Since it is only required, for control purposes, to relate the deflection of the tip to the torque applied at the hub, this operation is simplified by substituting x=L. This gives a transfer-function where both numerator and denominator are complex transcendental functions of s. To obtain a more

useful transfer-function these functions can be replaced by their infinite Taylor expansions about s=0. Factorising the resulting infinite power-series in s gives the required linear transfer-function in terms of products of quadratic factors ie.

$$\frac{y_{tip}(s)}{T(s)} = \frac{L}{I_t s^2} \cdot \prod_{i=1}^{i=\infty} \frac{(1+s^2/\alpha_i^2)}{(1+s^2/\omega_i^2)} \qquad (3\text{-}4)$$

This transfer-function, being of infinite order, is not immediately useful for the purposes of control design (Wang et al, 1986) and must be truncated to give a finite order model accurate over the bandwidth of the controller. Fraser advocates truncating (3-4) by neglecting higher frequency product terms ie.

$$\frac{y_{tip}(s)}{T(s)} \simeq \frac{L}{I_t s^2} \cdot \prod_{i=1}^{i=k} \frac{(1+s^2/\alpha_i^2)}{(1+s^2/\omega_i^2)} \qquad (3\text{-}5)$$

where k is finite. This is justified by noting that the neglected higher frequency terms $(1+s^2/\alpha_i^2)$ and $(1+s^2/\omega_i^2)$, i>k, will be approximately unity over the lower frequency band in which the model is expected to be accurate. The alternative truncation method of expanding (3-4) as a sum of partial fractions, removing the higher frequency fractions and recombining what is left is shown to give a poor description of the system zeros (ie. the truncated numerator).

The above analysis does not consider the effects of stiction and friction at the hub. However Fraser shows experimentally that these non-linear effects may be considered to introduce an effective Coulomb damping term into each of the denominator pole-pairs of (3-5) without significantly altering the resonant frequencies ω_i. The corresponding effect on the numerator is not considered to be as important here since the numerator is

always estimated with no prior information assumed known about it. For the purposes of this chapter and the experiments of chapter 5 only the first flexible modes will be considered ie.

$$\frac{y_{tip}(s)}{T(s)} \simeq \frac{s^2 + 2\zeta_o\omega_o s + \omega_o^2}{I_t s^2 (s^2 + 2\zeta_p\omega_p s + \omega_p^2)} \qquad (3\text{-}6)$$

The practical reasons for this choice are discussed in more detail in section 5.1.2.

3.2 Functional Relations between the Parameters

Many types of prior knowledge lead to the estimated parameters being functionally related to each other (Dasgupta et al, 1984). These relations may be incorporated as a constraint on the standard least squares algorithm (Clarke, 1981) which would otherwise attempt to minimise the cost

$$J = \sum_{i=0}^{t} \left[y(i) - \phi(i)^T\hat{\theta}(t) \right]^2 = \left[Y - X^T\hat{\theta}(t) \right]^T \left[Y - X^T\hat{\theta}(t) \right] \qquad (3\text{-}7)$$

w.r.t $\hat{\theta}(t)$ by solving the simultaneous equations

$$\frac{\partial J}{\partial \hat{\theta}_1} = \frac{\partial J}{\partial \hat{\theta}_2} = \ldots = 0 \qquad (3\text{-}8)$$

Assume, however, that a functional relation is known between elements of the parameter set ie.

$$f(\hat{\theta}_1, \hat{\theta}_2, \ldots) = 0 \qquad (3\text{-}9)$$

If it is required to minimise the least squares cost (3-7) subject to this constraint (3-9) then, by the method of Lagrange multipliers (Stephenson, 1973), the simultaneous equations in $\hat{\theta}_i$ become

$$\begin{aligned}
&\frac{\partial J}{\partial \hat{\theta}_1} + \lambda \frac{\partial f}{\partial \hat{\theta}_1} = 0 \\
&\frac{\partial J}{\partial \hat{\theta}_2} + \lambda \frac{\partial f}{\partial \hat{\theta}_2} = 0 \\
&\qquad \vdots \\
&\frac{\partial J}{\partial \hat{\theta}_n} + \lambda \frac{\partial f}{\partial \hat{\theta}_n} = 0
\end{aligned} \qquad (3\text{-}10)$$

where λ is the scalar Lagrange multiplier. Equations (3-9) and (3-10), if solved, give the constrained minimisation of $J(t)$ w.r.t. $\hat{\theta}(t)$. Unfortunately the derivatives $\partial f/\partial \hat{\theta}_i$ are often complicated non-linear functions of $\hat{\theta}_i$ and a general analytic solution to the constrained problem is not possible. Note that standard least squares, setting $\partial J/\partial \theta_i = 0$, will converge to the constrained solution if the constraints are correct ie. $\lambda \to 0$ with time.

However an exact or approximate solution to the constrained problem is often possible. For several types of prior knowledge the derivatives $\partial f/\partial \hat{\theta}_i$ are completely independent of $\hat{\theta}$ and easily evaluated (eg. sections 3.2.2-3.2.4). A secondary possibility is that if the standard least squares update is performed to give $\hat{\theta}_{ls}$ then estimates of the derivatives may be generated (eg. section 3.2.5). Consider then the solution of the approximate constrained equations with $\partial f/\partial \hat{\theta}_i$ either evaluated or estimated from $\hat{\theta}_{ls}$

$$\begin{aligned}
&\frac{\partial J}{\partial \hat{\theta}_1} + \lambda \left[\frac{\partial f}{\partial \hat{\theta}_1} \right]_{\hat{\theta}_{ls}} = \frac{\partial J}{\partial \hat{\theta}_1} + \lambda \hat{n}_1 = 0 \\
&\frac{\partial J}{\partial \hat{\theta}_2} + \lambda \left[\frac{\partial f}{\partial \hat{\theta}_2} \right]_{\hat{\theta}_{ls}} = \frac{\partial J}{\partial \hat{\theta}_2} + \lambda \hat{n}_2 = 0 \\
&\qquad \vdots
\end{aligned} \qquad (3\text{-}11)$$

From the second form of the cost function (3-7) this set of equations may be put in matrix form

$$-\mathbf{X}^T\mathbf{Y} + \mathbf{X}^T\mathbf{X}\hat{\theta} + \lambda\hat{\mathbf{n}} = 0 \qquad (3\text{-}12)$$

and the vector $\hat{\theta}$ may be split into the standard least squares estimates $\hat{\theta}_{ls}$ plus a perturbation $\delta\hat{\theta}$ required to satisfy the approximate equations (3-11) ie.

$$-\mathbf{X}^T\mathbf{Y} + \mathbf{X}^T\mathbf{X}\hat{\theta}_{ls} + \mathbf{X}^T\mathbf{X}\delta\hat{\theta} + \lambda\hat{\mathbf{n}} = \mathbf{X}^T\mathbf{X}\delta\hat{\theta} + \lambda\hat{\mathbf{n}} = 0 \qquad (3\text{-}13)$$

since $\hat{\theta}_{ls}$ sets $-\mathbf{X}^T\mathbf{Y} + \mathbf{X}^T\mathbf{X}\hat{\theta}_{ls} = 0$. This leaves

$$\delta\hat{\theta} = -\lambda[\mathbf{X}^T\mathbf{X}]^{-1}\hat{\mathbf{n}} \qquad (3\text{-}14)$$

with λ still unknown. From (3-9), using the first term of Taylor's expansion

$$f(\hat{\theta}_{ls} + \delta\hat{\theta}) \simeq f(\hat{\theta}_{ls}) + \delta\hat{\theta}^T\left[\frac{\partial f}{\partial \hat{\theta}}\right]_{\hat{\theta}_{ls}} = f(\hat{\theta}_{ls}) + \hat{\mathbf{n}}^T\delta\hat{\theta} = 0 \qquad (3\text{-}15)$$

Combining (3-14) and (3-15)

$$\lambda = \frac{f(\hat{\theta}_{ls})}{\hat{\mathbf{n}}^T[\mathbf{X}^T\mathbf{X}]^{-1}\hat{\mathbf{n}}} \quad \Rightarrow \quad \delta\hat{\theta} = -\frac{[\mathbf{X}^T\mathbf{X}]^{-1}\hat{\mathbf{n}}}{\hat{\mathbf{n}}^T[\mathbf{X}^T\mathbf{X}]^{-1}\hat{\mathbf{n}}} \cdot f(\hat{\theta}_{ls}) \qquad (3\text{-}16)$$

gives the constrained estimates

$$\hat{\theta} = \hat{\theta}_{ls} + \delta\hat{\theta} \qquad (3\text{-}17)$$

Note that the above analysis describes the incorporation of a single known functional relation between the parameters $\hat{\theta}_i$. The examples of sections

3.2.2-3.2.5 show that one functional relation corresponds to the knowledge of one continuous-time quantity. Prior information about additional quantities leads to additional functional relations. The inclusion of multiple constraints, however, is not so straightforward and is detailed in appendix A.7. For simplicity all the following examples concern single functional relations.

3.2.1 Implementation within RLS

The implementation of (3-16) and (3-17) is extremely simple given the existing RLS mechanisation. Consider the expression for the Kalman gain from section 1.1.3

$$K(t) = \frac{P(t-1)\phi(t)}{1 + \phi(t)^T P(t-1)\phi(t)} \qquad (3\text{-}18)$$

Since the covariance matrix after the standard least-squares recursion (1-10) to (1-12) is given by $P(t) = [X^T X]^{-1}$ the Kalman gain calculation (eg. Bierman, 1977) can merely be repeated with $\hat{n}$ replacing the data vector $\phi(t)$ and the unity element removed from the denominator of (3-18). The final parameter update becomes

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{P(t-1)\phi(t)}{1 + \phi(t)^T P(t-1)\phi(t)}.\varepsilon(t) - \frac{P(t)\hat{n}(t)}{\hat{n}(t)^T P(t)\hat{n}(t)}.f(\hat{\theta}_{ls}) \qquad (3\text{-}19)$$

where $\varepsilon(t) = y(t) - \hat{\theta}_{ls}(t-1)^T \phi(t)$.

To illustrate the usefulness of this approach the following sections give some functional relations likely to arise from prior knowledge typically available for plant characteristics. The effect of including these constraints is demonstrated in simulation examples and, in section 3.5, in

experiments on the compliant arm. For the following the coefficient vector $\hat{\theta}$ is defined as

$$\begin{aligned}\hat{\theta} &= \{-\hat{a}_1,-\hat{a}_2,\ldots,-\hat{a}_{na},\ \hat{b}_0,\ \hat{b}_1,\ \ldots\ ,\ \hat{b}_{nb}\}^T \\ &= \{\hat{\theta}_1,\ \hat{\theta}_2\ ,\ldots,\hat{\theta}_{na},\hat{\theta}_{na+1},\hat{\theta}_{na+2},\ldots,\hat{\theta}_{na+nb+1}\}^T\end{aligned} \qquad (3\text{-}20)$$

3.2.2 Exact constraint: Knowledge of Dynamic Gain

Consider the case where the dynamic gain of the plant, g, is known. If a z-transform model of the form (1-7) is to be estimated subject to this knowledge the coefficients $\hat{a}_j,\hat{b}_i$ must obey

$$g = \frac{\hat{B}(1)}{\hat{A}(1)} = \frac{\Sigma\hat{b}_i}{1 + \Sigma\hat{a}_j} \qquad (3\text{-}21)$$

which, from (3-20), corresponds to the functional relation

$$f(\hat{\theta}) = g\hat{\theta}_1 + g\hat{\theta}_2 + \ldots + g\hat{\theta}_{na} + \hat{\theta}_{na+1} + \ldots + \hat{\theta}_{na+nb+1} - g = 0 \qquad (3\text{-}22)$$

with the associated differential vector

$$\frac{\partial f}{\partial \hat{\theta}} = \{\ g\ ,\ \ g\ ,\ \ \ldots\ \ ,\ g\ ,\ \ 1\ ,\ \ \ldots\ \ ,\ \ 1\ \}^T \qquad (3\text{-}23)$$

Note that there is no approximation involved in forming $\partial f/\partial\hat{\theta}$ and the higher derivatives are zero. This means that the truncated Taylor series (3-15), and therefore the constrained minimisation of the least-squares cost, are exact.

As an example Fig.3.2 compares the simulated initial tuning transients of incremental self-tuning GPC, configured with and without prior

knowledge of the gain, on the plant

$$\frac{y(t)}{u(t)} = G(s) = \frac{50}{(1+10s)(1+5s)} \qquad (3\text{-}24)$$

sampled at 1Hz. The estimator was, in both cases, initialised with $P(0)=1000I$, $\hat{\theta}(0)=\{0\ 0\ 1\ 0\}^T$, $\beta(t)=1$ and a correctly parameterised model ie. 2 $\hat{b}_i$ and 2 $\hat{a}_j$ estimated parameters. The initialisation of $\hat{\theta}(0)$ is the standard default used in practice since an all-zero $\hat{\theta}(0)$ means that GPC never exerts an initial control signal and estimation cannot proceed. The GPC controller was configured with $N_1 = NU = 1$, $NY=5$, $\lambda=0.0$ and pre-specified set-point following.

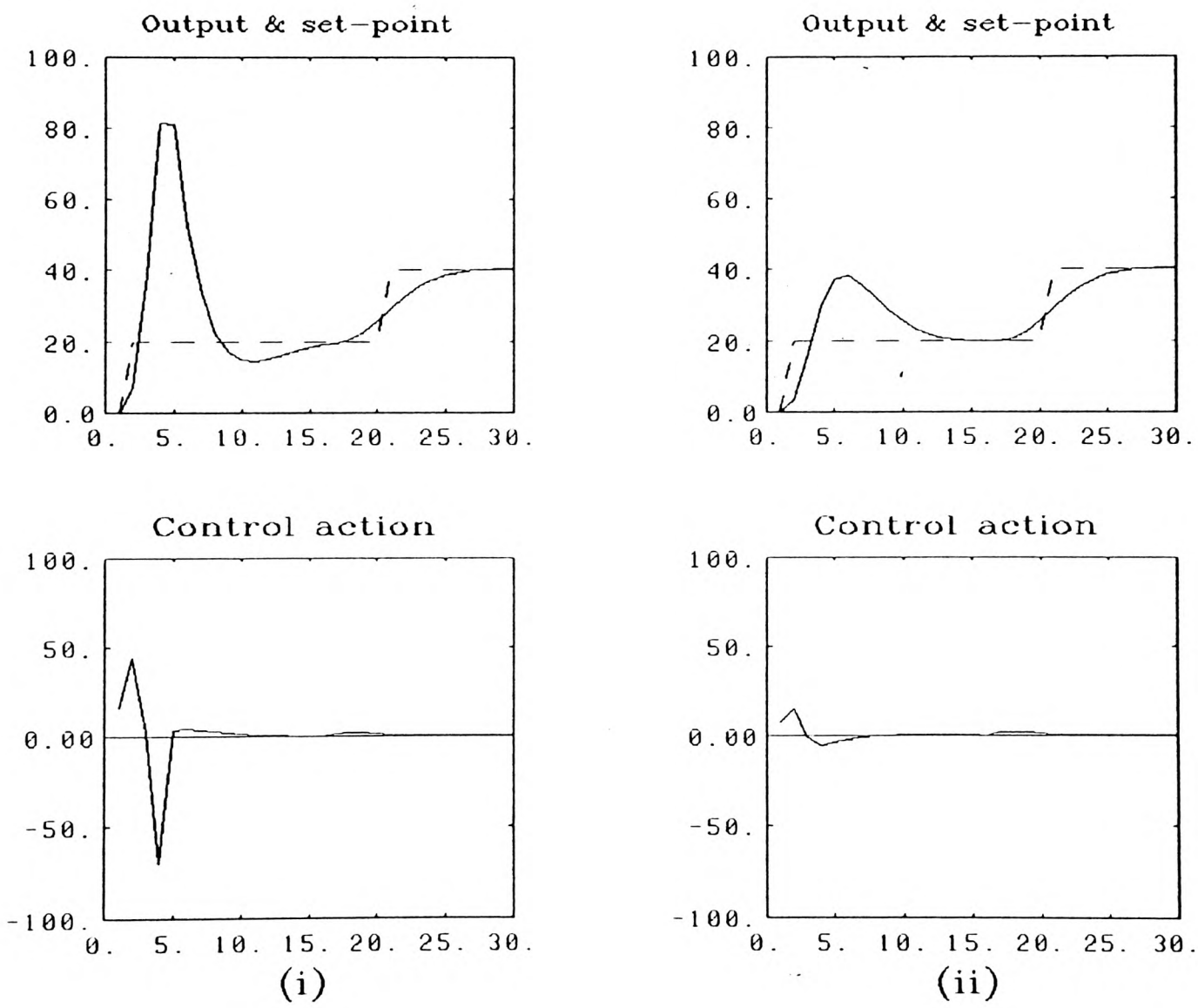


Fig.3.2: Knowledge of dynamic gain

The left-hand pair of graphs (i) show the output and control tuning transients arising from standard least-squares while the right-hand pair of

graphs (ii) show the comparative effect of incorporating the knowledge that g = 50 ie.

$$f(\hat{\theta}) = 50\hat{\theta}_1 + 50\hat{\theta}_2 + \hat{\theta}_3 + \hat{\theta}_4 - 50 = 0$$

$$\frac{\partial f}{\partial \hat{\theta}} = \{ 50 \; 50 \; 1 \; 1 \}^T$$

Clearly incorporation of the prior knowledge of gain reduces both the duration and severity of the tuning transient. Note also that both methods converge to the exact model giving identical second set-point responses.

3.2.3 Exact constraint: Knowledge of a Real Factor

Consider the case where a real root of the $A(z^{-1})$ polynomial is known ie.

$$\hat{A}(z^{-1}) = (1-\beta z^{-1})\hat{A}'(z^{-1}) \qquad (3\text{-}25)$$

where β is known. This gives rise to a functional relation between the parameters from substituting $z=\beta$ into (3-25) ie.

$$A(z^{-1})\Big|_{z=\beta} = 1 + a_1\beta^{-1} + a_2\beta^{-2} + \ldots + a_{na}\beta^{-na} = 0 \qquad (3\text{-}26)$$

ie.

$$f(\hat{\theta}) = \beta^{na-1}\hat{\theta}_1 + \beta^{na-2}\hat{\theta}_2 + \ldots + \beta\hat{\theta}_{na-1} + \hat{\theta}_{na} - \beta^{na} = 0 \qquad (3\text{-}27)$$

with the associated differential vector

$$\frac{\partial f}{\partial \hat{\theta}} = \{ \beta^{na-1}, \; \beta^{na-2}, \; \ldots, \; \beta, \; 1, \; 0, \; 0, \; \ldots \}^T \qquad (3\text{-}28)$$

Note that the vector $\partial f/\partial \hat{\theta}$ does not depend on $\hat{\theta}_i$ and therefore need not be approximated in (3-11). This, as with knowledge of gain, allows an exact constrained minimisation of the least-squares cost.

As an example of the effect of incorporating knowledge of a real factor consider the plant

$$\frac{y(t)}{u(t)} = G(s) = \frac{1}{s(1+10s)} \qquad (3\text{-}29)$$

sampled at 0.5Hz. Fig.3.3 compares the simulated initial tuning transient of incremental self-tuning GPC, with and without the information that an integrator is present ie.

$$\hat{A}(z^{-1}) = (1-z^{-1})\hat{A}'(z^{-1})$$

The recursive least-squares estimator was, in both cases, initialised with $P(0)=1000I$, $\hat{\theta}(0)=\{0\ 0\ 1\ 0\}^T$, $\beta(t)=1$ and a correctly parameterised model ie. 2 $\hat{b}_i$ and 2 $\hat{a}_j$ estimated parameters. The GPC controller was configured with $N_1 = NU = 1$, NY=5, λ=0.0 and pre-specified set-point following.

The left-hand pair of graphs (i) show the tuning transients arising from standard least-squares while the right-hand pair of graphs show the effect of incorporating the prior knowledge that the pole polynomial $A(z^{-1})$ has a factor $(1-z^{-1})$ ie.

$$f(\hat{\theta}) = \hat{\theta}_1 + \hat{\theta}_2 - 1 = 0$$

$$\frac{\partial f}{\partial \hat{\theta}} = \{\ 1\ 1\ 0\ 0\ \}^T$$

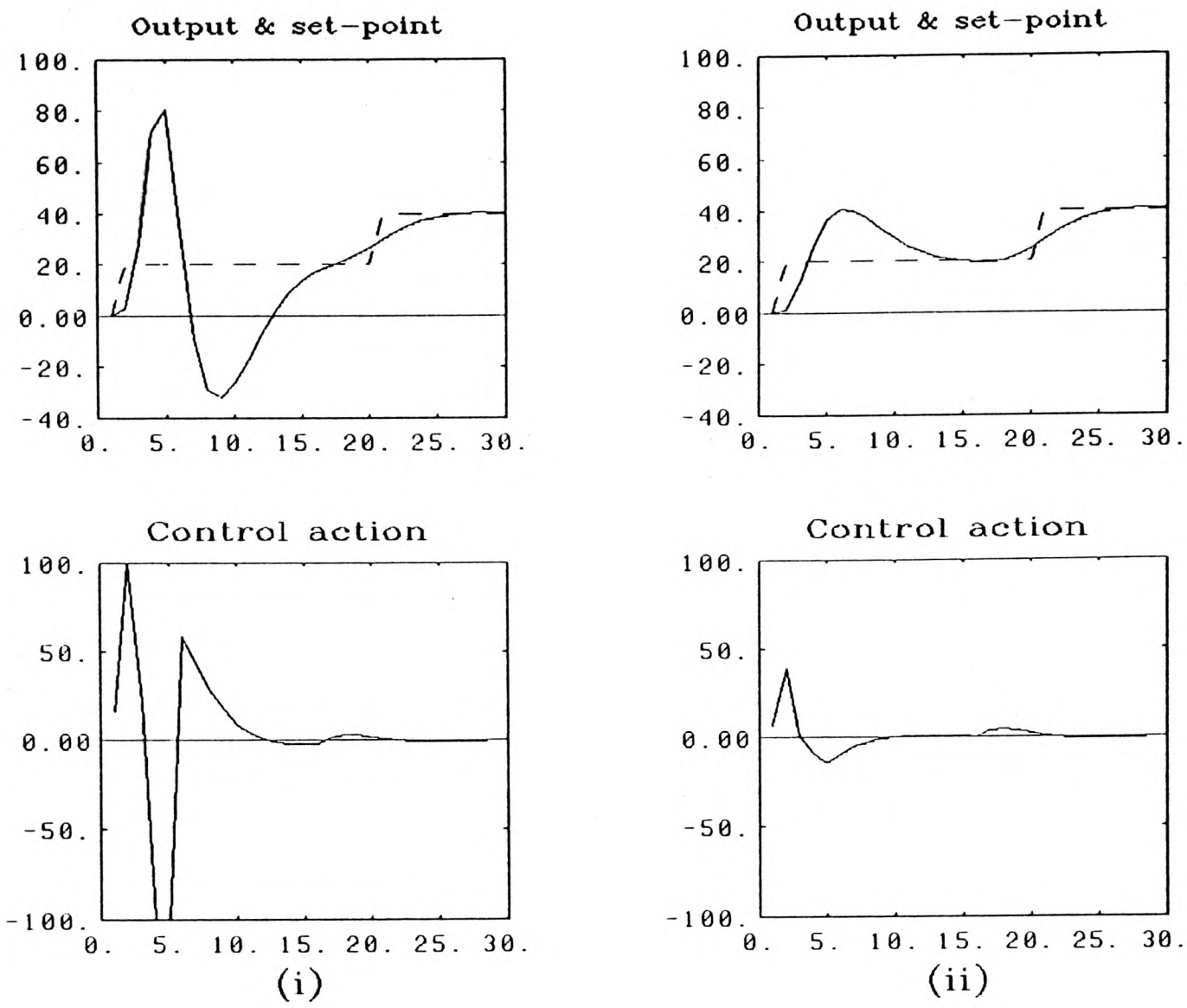


Fig.3.3: Knowledge of a real factor

The inclusion of the known factor clearly again both improves the convergence rate of the self-tuner and reduces the severity of the initial tuning transient.

3.2.4 Inexact constraint: Partial knowledge of a Complex Factor

The previous examples involved exact constraints. To illustrate the case of an inexact constraint, in the context of flexible systems, consider the plant

$$\frac{y(t)}{u(t)} = G(s) = \frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2} \qquad (3\text{-}30)$$

sampled at 5Hz with $\omega=2\pi$ rads^{-1}(1Hz) and $\zeta=0.1$. Assume that the resonant frequency ω is known and that the damping ζ is known to be small. The z-transform polynomial corresponding to the denominator of (3-30) is

$$A(z^{-1}) = 1 - 2e^{-\zeta\omega h}\cos(\omega h\sqrt{(1-\zeta^2)})z^{-1} + e^{-2\zeta\omega h}z^{-2} \qquad (3\text{-}31)$$

where h is the sample interval. When ζ is small this is given approximately by

$$A(z^{-1}) \simeq 1 - 2(1-\zeta\omega h)\cos(\omega h)z^{-1} + (1-2\zeta\omega h)z^{-2} \qquad (3\text{-}32)$$

giving the functional relation

$$f(\hat{\theta}) = \hat{\theta}_1 + \cos(\omega h)\hat{\theta}_2 - \cos(\omega h) = 0 \qquad (3\text{-}33)$$

with the associated differential vector

$$\frac{\partial f}{\partial \hat{\theta}} = \{\ 1\ \cos(\omega h)\ 0\ 0\ \}^T \qquad (3\text{-}34)$$

Fig.3.4 compares the simulated initial tuning transients of incremental self-tuning GPC, with and without this prior knowledge. The recursive least-squares estimator was, in both cases, initialised with $P(0)=1000I$, $\hat{\theta}(0)=\{0\ 0\ 1\ 0\}^T$, $\beta(t)=1$ and a correctly parameterised model ie. 2 $\hat{b}_i$ and 2 $\hat{a}_j$ estimated parameters. The GPC controller was configured with $N_1=1$, NU = 2, NY=5, $\lambda=0.0$ and pre-specified set-point following.

The left-hand pair of graphs (i) show the transients arising from standard least-squares while the right-hand pair of graphs show the effect of incorporating the prior knowledge of resonant frequency and light damping. A further set-point response shows that the converged behaviour is identical in both cases. This indicates that the approximations made were quite accurate.

Clearly the convergence rate is again improved and the severity of the initial tuning-transient reduced.

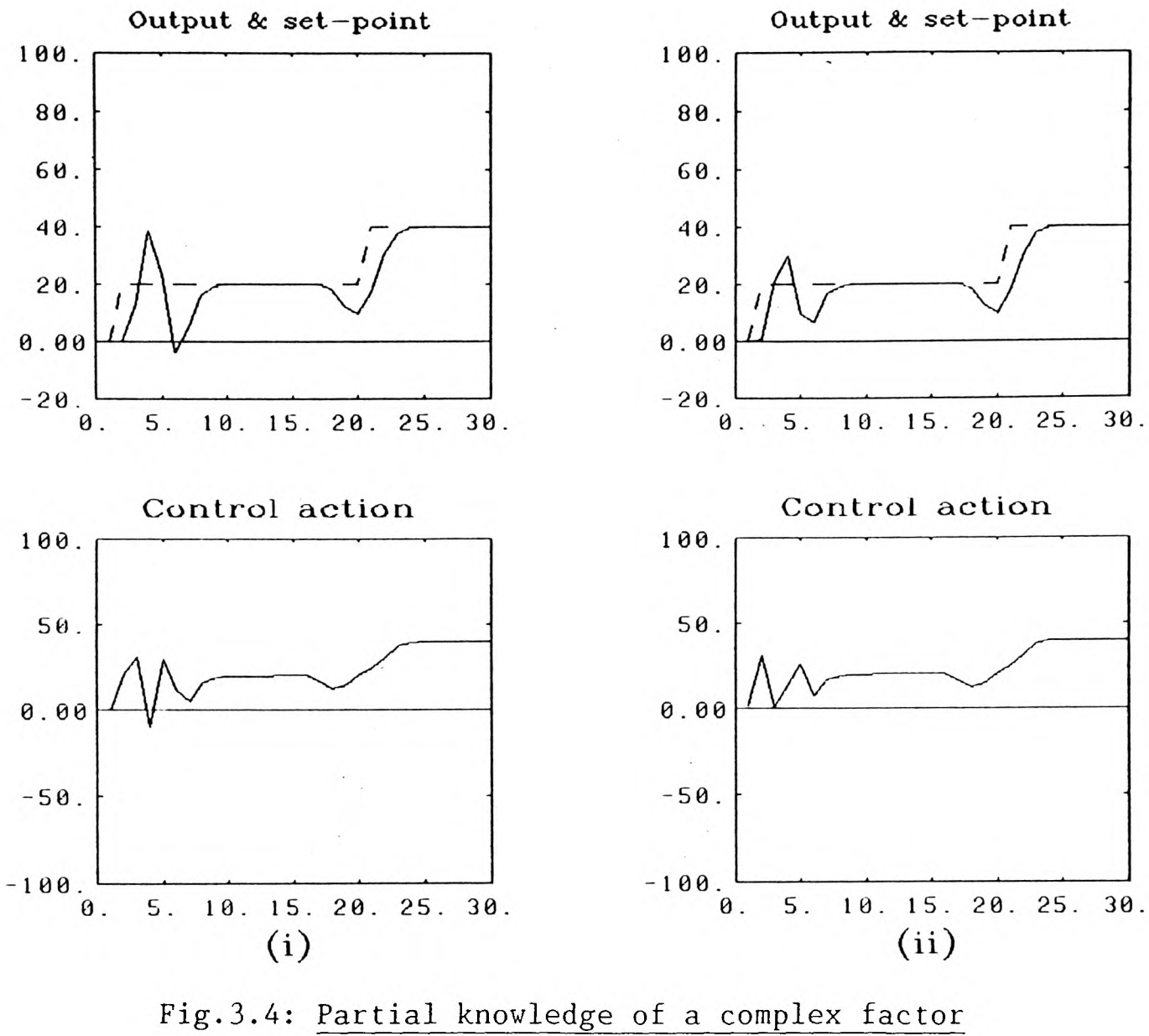


Fig.3.4: Partial knowledge of a complex factor

3.2.5 Approximate constraint: Partial knowledge of a Complex Factor

The previous examples either involved constraints where the Taylor expansion (3-15) does not go beyond the first derivative term and is therefore exact or where this is approximately the case (section 3.2.4). A further possibility deserving of investigation occurs when $\partial f/\partial \hat{\theta}_i$ depends on $\hat{\theta}_i$. An approximate solution is to proxy $\hat{\theta}_i$ by the most recent least-squares estimates $\hat{\theta}_{ls}$. To investigate the effectiveness of this approach consider the plant

$$\frac{y(t)}{u(t)} = G(s) = \frac{\omega^2}{(1+sT)(s^2+\omega^2)} \qquad (3\text{-}35)$$

sampled at 0.5Hz where ω=0.1 rads^{-1} and T=5 secs. The z-transform polynomial corresponding to the denominator of (3-35) is

$$A(z^{-1}) = (1-\alpha z^{-1})(1-2e^{-\zeta\omega h}\cos(\omega h\sqrt{(1-\zeta^2)})z^{-1}+e^{-2\zeta\omega h}z^{-2}) \qquad (3\text{-}36)$$

where h is the sample interval and $\alpha=e^{-h/T}$. Assuming that the resonance is known to have negligible damping ie. $\zeta\simeq 0$, this simplifies to

$$\begin{aligned} A(z^{-1}) &= (1-\alpha z^{-1})(1-2\cos\omega h z^{-1}+z^{-2}) \qquad (3\text{-}37)\\ &= 1-(\alpha+2\cos\omega h)z^{-1}+(2\alpha\cos\omega h+1)z^{-2}-\alpha z^{-3} \end{aligned}$$

giving the functional relation

$$f(\hat{\theta}) = \hat{\theta}_3^2 - \hat{\theta}_1\hat{\theta}_3 - \hat{\theta}_2 - 1 = 0 \qquad (3\text{-}38)$$

with the associated differential vector

$$\frac{\partial f}{\partial \hat{\theta}} = \{\ -\hat{\theta}_3\ -1\ (2\hat{\theta}_3-\hat{\theta}_1)\ 0\ 0\ 0\ \}^T \qquad (3\text{-}39)$$

Fig.3.5 compares the simulated initial tuning transients of incremental self-tuning GPC based on standard RLS and the approximately constrained RLS. The estimator was in both cases initialised with P(0)=1000I, $\hat{\theta}(0)=\{0\ \ 0\ 1\ 0\ 0\ 0\}^T$, $\beta(t)$=1 and a correctly parameterised model ie. 3 $\hat{b}_i$ and 3 $\hat{a}_j$ coefficients estimated. The GPC controller was configured with N_1=1, NU=2, NY=5, λ=1e-4 and pre-specified set-point following.

Unlike the previous results this constrained minimisation performs worse than standard least squares with a poorer tuning transient, although it does converge to give identical behaviour on the second set-point response. This may reasonably be attributed to both the early approximations to $\partial f/\partial\hat{\theta}_i$

being inaccurate and the ommission of the $\partial^2 f/\partial\hat{\theta}_i \partial\hat{\theta}_j$ terms in the Taylor series making the method even more inexact.

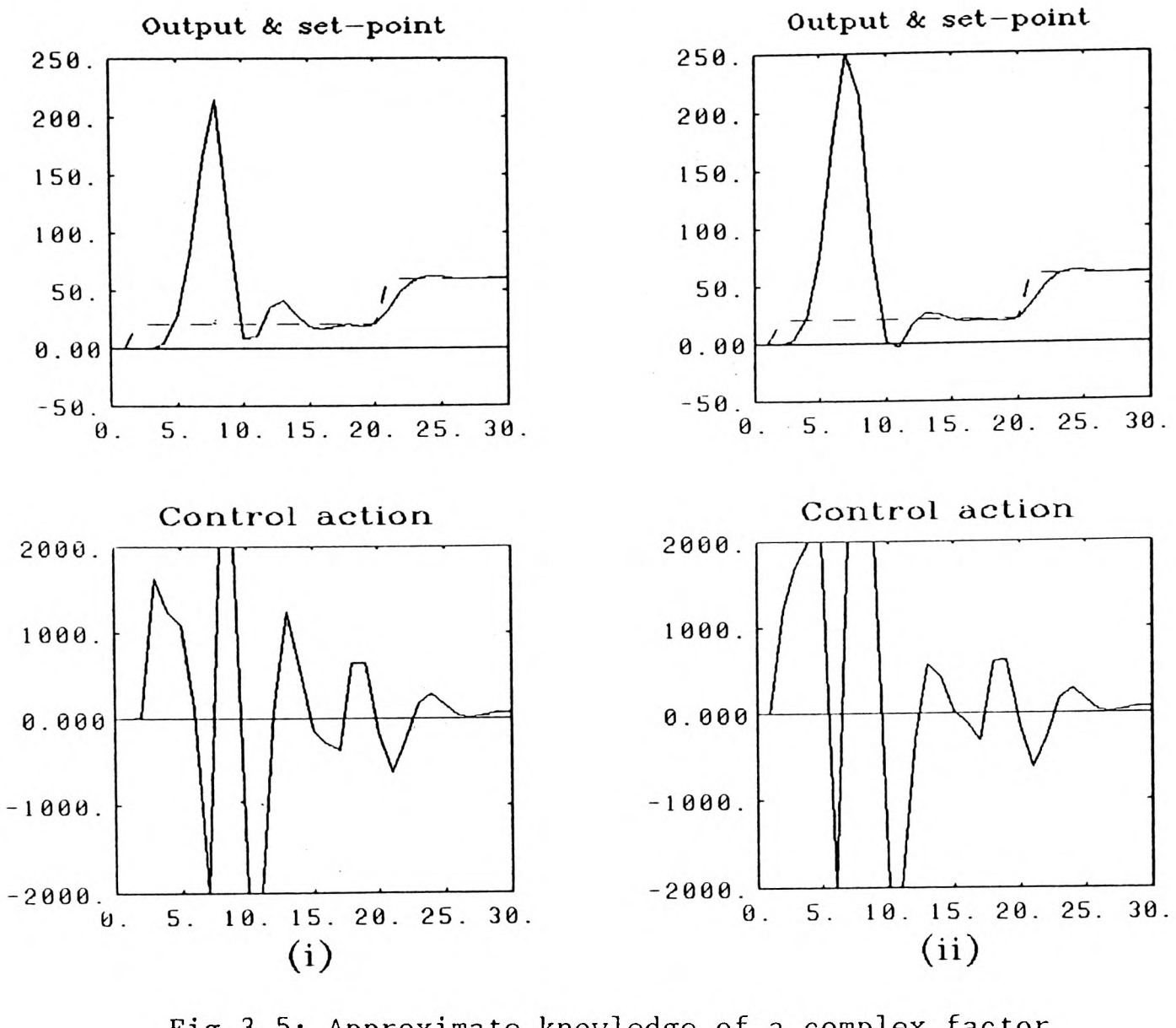


Fig.3.5: Approximate knowledge of a complex factor

3.3 Regressor Filtering

An alternative method to that of the preceding section, and one which can easily incorporate more than one known factor, is given by the regressor filtering method described below. While this method is not as flexible as the functional relation method (eg. it cannot use knowledge of gain) it has the advantage that it reduces the computational workload. Consider the standard estimation problem of fitting a CARMA-type dynamic model

$$\hat{A}(z^{-1})y(t) = \hat{B}(z^{-1})u(t-1) + e(t) \qquad (3\text{-}40)$$

to a record of input/output data up to time t, $\{y(i),u(i);\ 0<i\leq t\}$, in such a way that the a posteriori modelling error e(i) is small. Suppose further that a factor of $\hat{A}(z^{-1})$ is known exactly ie. $\hat{A} = \hat{A}_1 A_2$ where it is A_2 that is known. Both $\hat{A}$ and A_2 may be assumed to have a unit leading coefficient. The standard least-squares algorithm, without this prior knowledge, chooses the coefficients of the $\hat{A}$ and $\hat{B}$ polynomials in order to minimise the cost function

$$J = \sum_{i=0}^{t} \left[y(i) - z(1-\hat{A})y(i-1) - \hat{B}u(i-1) \right]^2 \qquad (3\text{-}41)$$

Given the known factor A_2 this may be rewritten

$$J = \sum_{i=0}^{t} \left[A_2 y(i) - z(1-\hat{A}_1)A_2 y(i-1) - \hat{B}u(i-1) \right]^2 \qquad (3\text{-}42)$$

or

$$J = \sum_{i=0}^{t} \left[y'(i) - z(1-\hat{A}_1)y'(i-1) - \hat{B}u(i-1) \right]^2 \qquad (3\text{-}43)$$

where the prime denotes a quantity filtered by the all-zero filter $A_2(z^{-1})$.

The important point is that the set of coefficients of $\hat{A}_1$ and $\hat{B}$ which minimises the modified cost (3-43) must also minimise the initial cost (3-41), subject to the constraint that A_2 is a factor of $\hat{A}$, since the costs are identical. Minimisation of the modified cost, barring the effect of initial conditions, is simply achieved by submitting measurements and y-regressors filtered by $A_2(z^{-1})$ to the standard recursive least squares recursion (1-10) to (1-12).

This has the advantage that, as well as incorporating the prior knowledge of the factor A_2, the number of parameters to be estimated is reduced. Despite the new requirement for filtering by A_2 this will in general substantially decrease the amount of computation required for the update. For

the purposes of control the full-order plant could be reformed by simple polynomial multiplication ie. $\hat{A}=\hat{A}_1\hat{A}_2$. It is easily shown that, when both methods minimise the same quadratic cost subject to the same constraints, regressor filtering yields the same estimates as those obtained from the functional relation method of section 3.2 with small differences from initial conditions. This equivalence, together with the improved performance obtained from inclusion of prior knowledge of a factor, is demonstrated in the following simulation example.

3.3.1 Exact constraint: Knowledge of a Real Factor

Consider the plant

$$y(t) = \frac{1}{s(1+10s)}\, u(t) + \xi(t) \qquad (3\text{-}44)$$

sampled at 0.5 Hz, where $\xi(t)$ is uncorrelated gaussian noise of r.m.s value 0.1. The plant is driven by an uncorrelated uniformly distributed input u(t) of r.m.s value 2.0 to provide a data sequence $\{y(t),u(t)\}$. It is assumed that the integrator is known to be present. From the data record thus generated three variations of recursive least-squares are used to estimate the discrete-time transfer-function $\hat{B}(z^{-1})/\hat{A}(z^{-1})$ corresponding to (3-44). The three algorithms used were

(i) The standard least-squares recursion (1-10) to (1-12) where 2 $\hat{a}_j$ and 2 $\hat{b}_i$ were estimated.

(ii) The modified least-squares estimator of section 3.2.1, constrained by a functional relation between the estimated parameters arising from the known factor $A_2= 1-z^{-1}$, configured as shown in the similar simulation example of section 3.2.3.

(iii) The constrained estimator of this section, where the known integrator factor $\Delta=1-z^{-1}$ is incorporated by modifying the regression to

$$\Delta y(t) = -\hat{a}_1 \Delta y(t-1) + \hat{b}_0 u(t-1) + \hat{b}_1 u(t-2) \qquad (3\text{-}45)$$

In all three cases positional data was used, there was no forgetting (ie. $\beta(t)=1$) and the recursion was initialised with $P(0)=1000I$ and $\hat{\theta}(0)=0$.

Fig.3.6 compares the roots of the estimated $A(z^{-1})$ polynomials obtained as the various algorithms recursively estimate from time t=0. The lower traces show the roots generated by the unmodified least-squares recursion. The upper traces overlay the roots generated by the regressor filtering estimator (shown dotted) with the roots obtained from the functional relation estimator (shown solid). Dashed lines represent the exact roots from (3-44).

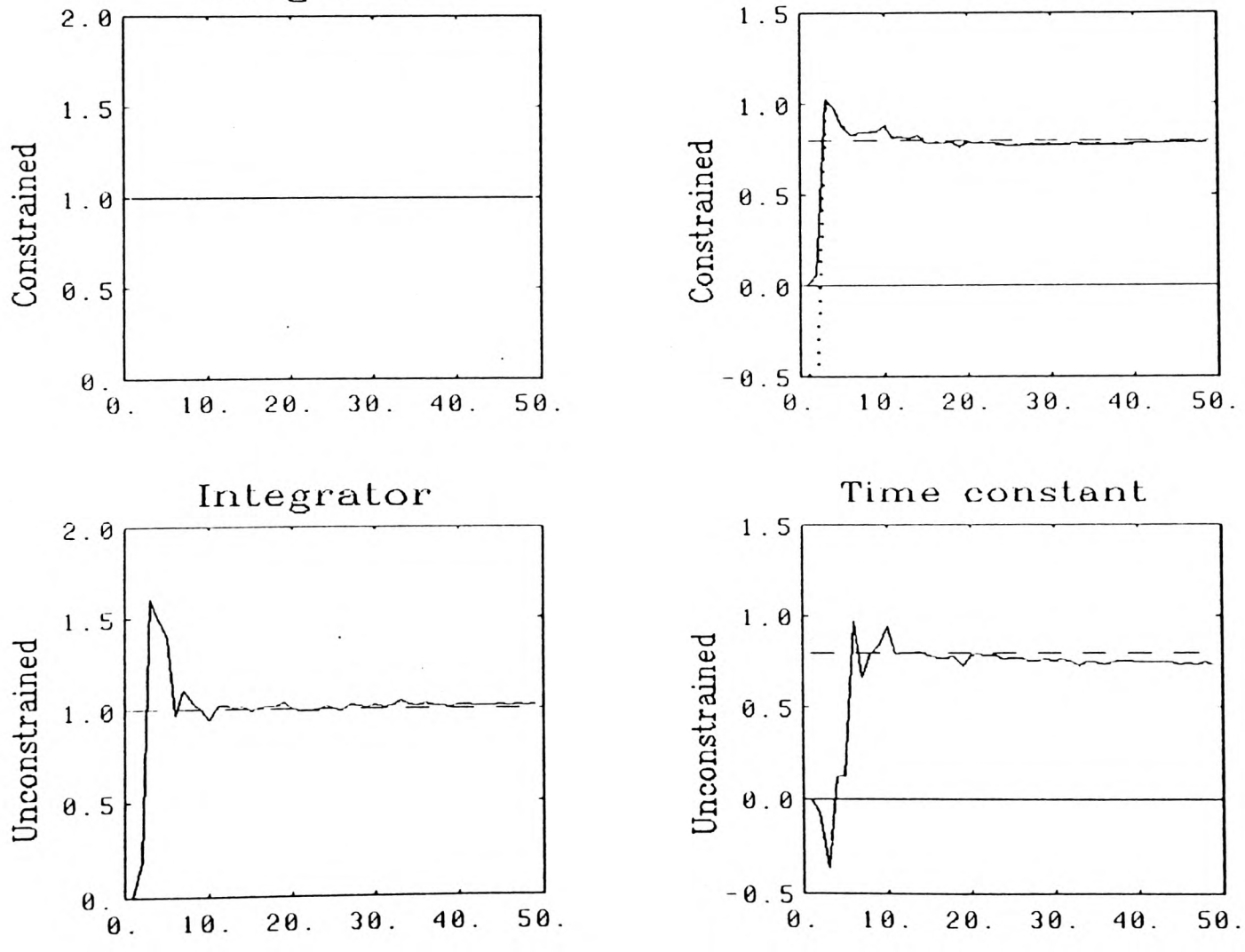


Fig.3.6: Knowledge of a real factor

It is clear that, after the slightly differing initial conditions die away the two upper traces coincide, showing the asymptotic equivalence of the two constrained algorithms. Note that both these algorithms provide an exact constraint, with the integrator being forced into the estimated $A(z^{-1})$ polynomial from the first sample. Note also the improved convergence and accuracy, in the face of measurement noise, resulting from the inclusion of this prior knowledge. Section 3.2.4 showed the improvement this made for a self-tuner running in closed-loop on the same example plant.

3.3.2 Effect of Prior Knowledge on Load-disturbances

The majority of the preceding examples illustrate the improved convergence rate resulting from the inclusion of prior knowledge. This section shows by simulation that, as would intuitively expected, the constrained estimators are also less sensitive to unmeasurable load disturbances acting upon the process. Consider the plant and data of the previous example

$$y(t) = \frac{1}{s(1+10s)}\, u(t) + \xi(t) + d(t) \qquad (3\text{-}46)$$

with the only difference being an additive disturbance d(t) which is unity over the 25th to 35th samples and zero elsewhere. A fixed-forgetting factor $\beta(t)=0.9$ is also used to prevent the estimator from converging before the onset of the disturbance. Fig.3.7 compares the roots of the estimated $A(z^{-1})$ polynomials obtained by the regressor filtering method of the previous section (known integrator) and standard least squares. Dashed lines represent the exact roots, calculated from (3-46).

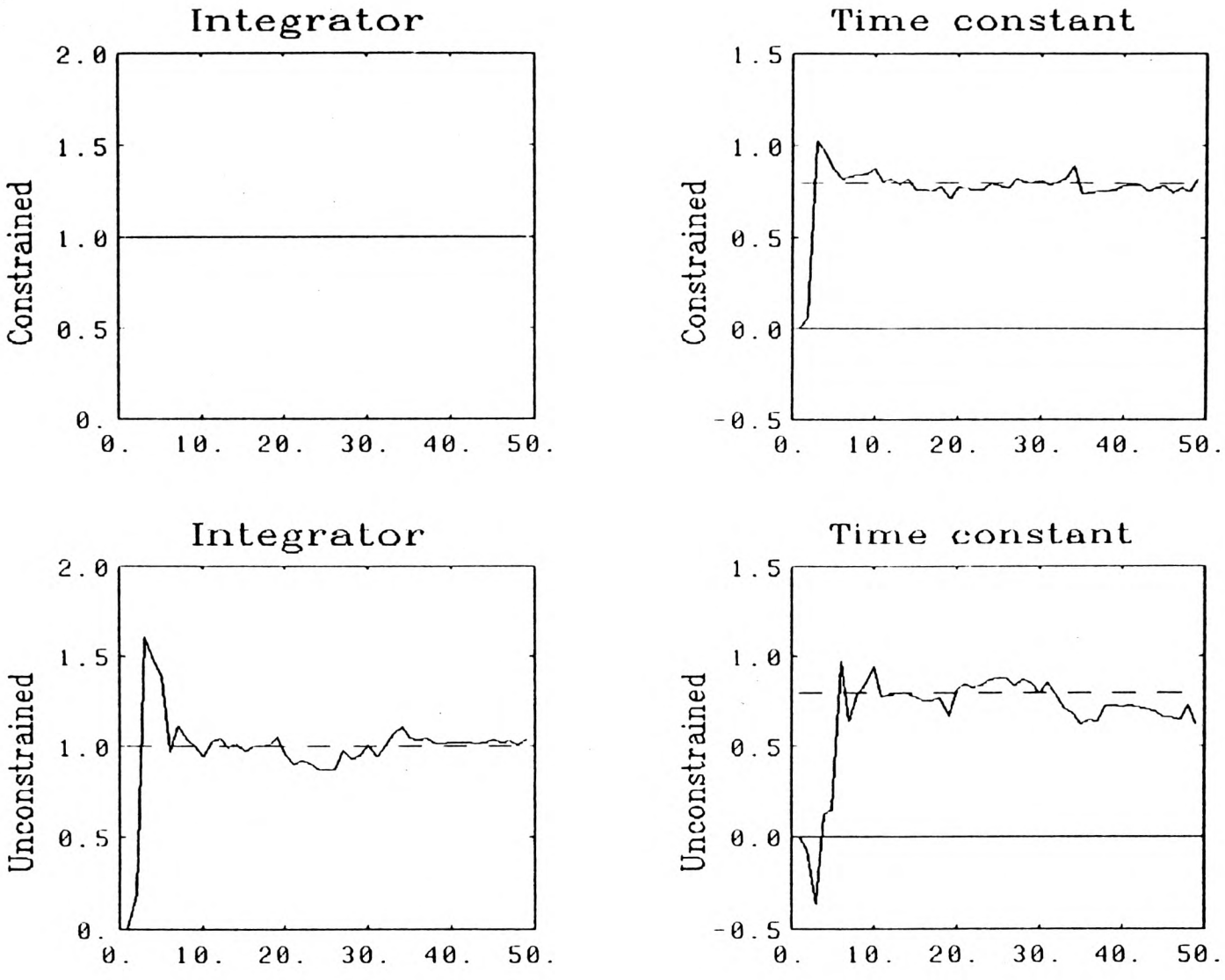


Fig.3.7: Effect of prior knowledge on load-disturbances

Clearly the constrained estimator not only converges more rapidly than standard RLS but also suffers to a far less extent from the load disturbance's addition at t=25 and its removal at t=35.

3.4 The δ-transform

In the context of incorporating prior knowledge into a self-tuner the δ-transform of Gawthrop (1980)* and Goodwin et al (1986) may prove useful. Since the use of the δ-transform in discrete-time control is somewhat novel sections 3.4.1-3.4.3 discuss its advantages and disadvantages, its implementation and the estimation of models in δ. Transfer-functions in the z-transform become transfer-functions of the same order in δ via the direct

* Gawthrop calls it the s′ operator.

map

$$\delta = \frac{z-1}{h} \qquad (3\text{-}47)$$

where h is the sample interval. Note that δ is also the forward Euler approximation to the derivative operator s. The theoretical advantage for which this operator was introduced is that, as $h \to 0$, the transfer-functions in δ tend to their continuous-time counterparts in s (Gawthrop, 1980). This is intuitively due to the Euler approximation becoming more exact. Provided the sample rate is chosen high enough, relative to the bandwidth of the plant, this can be used to unify continuous- and discrete-time control design. It also results in a reduced model parameterisation as described in section 3.4.4. The actual benefits of incorporating prior knowledge via the δ-transform are illustrated later in section 3.5.2 in experiments on the compliant arm.

3.4.1 The δ-transform: Advantages and disadvantages

The principal advantages of the δ-transform over the z-transform which are quoted in the literature are summarised below

(i) Middleton & Goodwin, (1986) show improvements in numerical properties for algorithms implemented using the δ-transform. In particular, for finite word-length arithmetic, the δ-transform improves coefficient representation, reduces the sensitivity of the controller to round-off noise and improves the conditioning of the on-line model manipulations used in controller design eg. solving diophantine equations.

(ii) Gawthrop (1980) shows that moving to the δ-transform may result in a reduced parameterisation for systems having a relative order ≥ 2 in

continuous-time. This effectively results from eliminating the extraneous discrete-time zeros which come from the sampling process (Astrom et al, 1984).

(iii) Goodwin et al (1986) use the unifying aspects of the δ-transform to successfully apply continuous-time MRAC techniques, via δ, to a minimum-phase continuous-time system system with NMP discrete-time sampling zeros. The conventional discrete-time MRAC design is unstable due to cancellation of these NMP zeros.

(iv) The incorporation of a time scaling into the operator δ means that, for high sample rates, the estimation and control need not necessarily proceed at the same rate.

Further advantages, pertaining to the inclusion of prior knowledge into the estimation scheme, are

(v) Prior information is normally available in terms of continuous-time quantities such as damping factors, frequencies etc. Since the δ-transform model tends towards its continuous-time equivalent as $h \to 0$ the methods of sections 3.2 and 3.3 should in general be easier to apply to δ-models (see section 3.5.2).

(vi) The reduced parameterisation mentioned in (ii) is actually a form of apriori knowledge and fulfils one of the objectives of its inclusion by reducing the number of coefficients to be estimated.

There are, of course, potential disadvantages to this approach

(i) All the above advantages arise only at high sample rates and, apart from the reduced parameterisation, the mechanisation for δ-transforms is far more time consuming than for z-transforms. To illustrate this consider the FLOP* count for one iteration of a transfer-function model in z^{-1} or δ where

* Floating-point operation: approximately one multiplication plus one add.

both numerator and denominator are of order n (eg. 3-66):

$$z^{-1}: \quad 2n$$

$$\delta \quad : \quad 3n+2$$

ie. an increase of at least 50%.

(ii) Higher-order filtering is required within a δ-transform estimator or controller which increases the amount of computation again and introduces phase effects. This is to counter the increased amplification of high-frequency disturbances by δ.

(iii) As with continuous-time the treatment of time-delay, especially unknown time-delay, is extremely difficult. Use can be made of Pade approximations (Gawthrop, 1987) or other alternatives developed for continuous-time design.

3.4.2 Implementation of δ-transform controllers and estimators

Whereas z-transforms are implemented using the unit delay operator z^{-1}, δ-transforms are implemented directly as δ. Since this operator is anti-causal and amplifies high-frequencies both the mechanisation and the filtering considerations are more complex than with the z-transform. The basic operations required for a self-tuner: prediction by iterating an estimated model, filtering of estimator regressors etc. require a mechanisation for transfer-functions in δ acting on sampled variables ie.

$$y(t) = \frac{G(\delta)}{F(\delta)}u(t) = \frac{g_{ng}\delta^{ng} + g_{ng-1}\delta^{ng-1} + \ldots + g_0}{f_{nf}\delta^{nf} + f_{nf-1}\delta^{nf-1} + \ldots + f_0}u(t) \qquad (3\text{-}48)$$

where $nf \leq ng$. Before proceeding to the mechanisation of this operation note that since $\delta^j = (\overline{z-1}/h)^j$ the numerator, expressed as a z-transform, will contain a term u(t+ng) and the denominator a term y(t+nf). The transfer-function $G(\delta)/F(\delta)$ therefore has an implicit delay of nf-ng samples.

To calculate y(t), given u(t), an intermediate variable

$$v(t) = \frac{u(t)}{F(\delta)} \tag{3-49}$$

is defined and the intermediate states $\{v(t), \delta v(t), \ldots, \delta^{nf} v(t)\}$ updated by the recursion

$$\begin{aligned} v(t) &= v(t-1) + h(\delta v(t-1)) \\ \delta v(t) &= \delta v(t-1) + h(\delta^2 v(t-1)) \\ &\vdots \qquad\qquad \vdots \\ \delta^{nf-2} v(t) &= \delta^{nf-2} v(t-1) + h(\delta^{nf-1} v(t-1)) \\ \delta^{nf-1} v(t) &= \delta^{nf-1} v(t-1) + h(\delta^{nf} v(t-1)) \end{aligned} \tag{3-50}$$

and

$$\delta^{nf} v(t) = \frac{1}{f_{nf}} \{ u(t) - f_0 v(t) - \ldots - f_{nf-1} \delta^{nf-1} v(t) \} \tag{3-51}$$

where the ordering of these equations is important. This assumes that the old state vector at time t-1, $\{v(t-1), \delta v(t-1), \ldots, \delta^{nf} v(t-1)\}$, is stored from sample to sample. The output may then be calculated from the simple scalar product

$$y(t) = G(\delta) v(t) = g_{ng} \delta^{ng} v(t) + g_{ng-1} \delta^{ng-1} v(t) + \ldots + g_0 v(t) \tag{3-52}$$

3.4.3 Estimation of δ-transform models

Expanding the standard z-transform CARMA model

$$A(z^{-1})y(t) = B(z^{-1})u(t-1) + e(t) \qquad (3\text{-}53)$$

gives

$$(1+a_1z^{-1}+ \ldots +a_nz^{-n})y(t) = (b_0+b_1z^{-1}+ \ldots +b_nz^{-n})u(t-1) + e(t)$$

where the order of the $B(z^{-1})$ polynomial is n to allow for the possibility of fractional time delay (Peterka, 1986). Multiplying both sides by z^{n+1} and substituting for z in terms of δ this becomes

$$(\delta^{n+1}+\bar{a}_n\delta^n+\ldots+\bar{a}_1\delta+\bar{a}_0)y(t) = (\bar{b}_n\delta^n+\bar{b}_{n-1}\delta^{n-1}+\ldots+\bar{b}_1\delta+\bar{b}_0)u(t) + \frac{e(t+n+1)}{h^{n+1}}$$

or

$$\bar{A}(\delta)y(t) = \bar{B}(\delta)u(t) + \bar{e}(t+1) \qquad (3\text{-}54)$$

where the unit difference in orders of the $\bar{B}$ and $\bar{A}$ polynomials is due to the inevitable delay of the ZOH. This is equivalent to knowing that $\bar{A}$ has a factor $(1+h\delta)$, ie. a unit time advance, so (3-54) can be rewritten

$$A(\delta)y(t) = B(\delta)u(t-1) + \bar{e}(t) \qquad (3\text{-}55)$$

where $A(\delta)$ and $B(\delta)$ are the same order. Note that (3-55) introduces an abuse of notation with, for the remainder of this chapter, a_i and b_j referring both the coefficients of the z-transform and δ-transform models where the context makes it clear. Since $\delta^j y(t)$ involves near 'differentiation' of the data Goodwin et al (1986) incorporate a polynomial prefilter (similar to $T(z^{-1})$

of section 2.6.3)

$$T(\delta) = \delta^n + t_{n-1}\delta^{n-1} + \ldots + t_0 \qquad (3\text{-}56)$$

by writing

$$y(t) = \frac{T(\delta)-A(\delta)}{T(\delta)}\, y(t) + \frac{B(\delta)}{T(\delta)}u(t-1) + \frac{1}{T(\delta)}\,\bar{e}(t) \qquad (3\text{-}57)$$

Note that since both $A(\delta)$ and $T(\delta)$ have unity leading coefficients only data up to time t-1 is required on the RHS of (3-57). Denoting filtering by $T(\delta)$ with a prime this becomes

$$y(t) = E(\delta)y'(t) + B(\delta)u'(t-1) + \bar{e}'(t) \qquad (3\text{-}58)$$

where

$$E(\delta) = T(\delta) - A(\delta) \qquad (3\text{-}59)$$

which can be written in the standard regression form

$$y(t) = \theta^T\phi(t-1) + \bar{e}'(t) \qquad (3\text{-}60)$$

with

$$\phi(t-1) = \{y'(t),\ldots,\delta^{n-1}y'(t),u'(t-1),\ldots,\delta^n u'(t-1)\}^T$$

$$\theta = \{e_0, \ldots, e_{n-1}, b_0, \ldots, b_n\}^T$$

The coefficient vector θ may now be estimated using the standard least-squares recursion of section 1.1.3. The estimated $A(\delta)$ polynomial can be reconstructed from (3-59) since $T(\delta)$ is known.

Finally, since the pre-filtering by $1/T(\delta)$ has finite d.c. gain, Goodwin et al (1986) suggests additional filtering of both sides of (3-60) by

the high-pass transfer-function

$$\frac{\delta}{\delta+\varepsilon} \qquad (3\text{-}61)$$

(where ε is a 'small' number) to eliminate offsets. This of course corresponds to moving to an incremental z-transform algorithm.

3.4.4 Reduced Parameterisation of δ-transform models

Consider the continuous-time plant in the absence of disturbances

$$y(t) = \frac{\tilde{B}(s)}{\tilde{A}(s)}u(t) = \frac{\tilde{b}_m s^m + \tilde{b}_{m-1} s^{m-1} + \ldots + \tilde{b}_0}{s^n + \tilde{a}_{n-1} s^{n-1} + \ldots + \tilde{a}_0}u(t) \qquad (3\text{-}62)$$

where $m<n$ and there is no delay, fractional or otherwise. This may be exactly discretised to a transfer-function in δ

$$y(t) = \frac{B(\delta)}{A(\delta)}u(t) = \frac{b_{n-1}\delta^{n-1} + .. + b_m\delta^m + .. + b_0}{\delta^n + a_{n-1}\delta^{n-1} + \ldots + a_0}u(t) \qquad (3\text{-}63)$$

It is well known that the zeros of the polynomial $\tilde{A}(s)$ are directly related to the zeros of $A(\delta)$. If $s=p_i$ denote the zeros of $\tilde{A}(s)$ then the zeros of $A(\delta)$ lie at $\delta=(e^{p_i h}-1)/h^*$. Unfortunately the relation between the zeros of $\tilde{B}(s)$ and $B(\delta)$ is not so straightforward (Astrom et al, 1984). Nevertheless Gawthrop (1980) shows that, as $h\to 0$ relative to the plant dynamics, the coefficients of $B(\delta)$ tend towards the continuous-time coefficients $\tilde{b}_i$. Consequently the high-order coefficients b_i of $B(\delta)$, where $i>m$, must tend to zero.

* Where h is the sample interval.

For high sample rates Goodwin et al (1986) therefore suggests neglecting these coefficients ie.

$$y(t) \simeq \frac{B_1(\delta)}{A(\delta)} u(t) = \frac{\bar{b}_m \delta^m + \bar{b}_{m-1}\delta^{m-1} + \ldots + \bar{b}_0}{\delta^n + a_{n-1}\delta^{n-1} + \ldots + a_0} u(t) \qquad (3\text{-}64)$$

where $a_j \simeq \tilde{a}_j$ and $\bar{b}_i \simeq \tilde{b}_i$ of (3-62). Since this is only asymptotically exact (as $h \to 0$) it may be interpreted as attempting to approximate the continuous-time model (3-62) using the Euler approximation to s. The disadvantage of moving to an inexact model must be balanced against the advantage of fewer $\bar{b}_i$ coefficients to be estimated (possibly improved convergence) and reduced controller calculations (if implemented in δ).

Effect of fractional time delay

The theoretical appeal of this approach is somewhat marred by the inevitable presence of fractional time delay due to the finite computation time of the controller. Since this computation time limits the maximum sample rate it must be the case that fast sampling controllers in particular will have large fractional delays. Consider the plant with a fractional delay τ

$$y(t) = e^{-s\tau} \frac{\tilde{B}(s)}{\tilde{A}(s)} u(t) \qquad (3\text{-}65)$$

where $0 \leq \tau < h$. This delay causes the exact discretisation (3-63) to become

$$\frac{y(t)}{u(t-1)} = \frac{B_2(\delta)}{A(\delta)} = \frac{b_n \delta^n + b_{n-1}\delta^{n-1} + \ldots + b_0}{\delta^n + a_{n-1}\delta^{n-1} + \ldots + a_0} \qquad (3\text{-}66)$$

as described in section 3.4.3 (the a_j coefficients are unchanged). Now the continuous-time approximation $u(t-\tau)\simeq u(t-h)$ corresponds, from (3-65), to

$$\frac{y(t)}{u(t-h)} \simeq \frac{\tilde{B}(s)}{\tilde{A}(s)} \qquad (3\text{-}67)$$

which allows Goodwin's approximation (3-64) to be applied to (3-67) ie.

$$\frac{y(t)}{u(t-1)} \simeq \frac{B_3(\delta)}{A(\delta)} = \frac{b_m\delta^m + b_{m-1}\delta^{m-1} + \ldots + b_0}{\delta^n + a_{n-1}\delta^{n-1} + \ldots + a_0} \qquad (3\text{-}68)$$

where $a_j \simeq \tilde{a}_j$ and $b_i \simeq \tilde{b}_i$ of (3-65). Note however that h can no longer tend to zero since the continuous-time delay τ would then tend to an infinite discrete-time integer delay. The reduced parameterisation (3-68) should be viewed as an approximation to G(s) for small h dependent on the further approximation $u(t-\tau)\simeq u(t-h)$. It is fortunate that, when the fractional delay is purely due to computation, the fastest possible sample rate corresponds to $h=\tau$. For rapid sampling the latter approximation will therefore often be accurate and the reduced parameterisation of (3-68) should be used rather than (3-64) of Goodwin et al (1986).

Equivalent Reduced Parameterisation for z-transform models

Note that the approximation (3-68) also introduces an implicit delay of n-m samples (above the unit ZOH delay) which is not present in the continuous-time model (3-65). This has the important consequence that the reduced parameterisation is not unique to the δ-transform. Substituting

$\delta=(z-1)/h$ into (3-68) gives the equivalent approximation

$$\frac{y(t)}{u(t-m-n-1)} \simeq \frac{b'_0 + b'_1 z^{-1} + \ldots + b'_m z^{-m}}{1 + a_1 z^{-1} + .. + a_n z^{-n}} \qquad (3\text{-}69)$$

This is interesting since it suggests that the disparity between continuous- and discrete-time control is due to effective time delays introduced by sampling processes with finite continuous-time relative order. The validity of this approximation is investigated experimentally in section 3.5.1. Note that $A(z^{-1})$ is again not directly affected by this approximation.

3.4.5 Inclusion of Prior Knowledge with δ-models

Before proceeding it is noted that the vector of parameters to be estimated, θ in (3-60), does not contain terms directly in the coefficients a_j of (3-66) and consequently (for the purposes of including prior information) it is necessary to modify (3-58) to

$$\frac{\delta^n y(t)}{T(\delta)} = (\delta^n - A(\delta))\frac{y(t)}{T(\delta)} + B(\delta)\frac{u(t-1)}{T(\delta)} + \frac{\bar{e}(t)}{T(\delta)} \qquad (3\text{-}70)$$

ie. the linear regression

$$\delta^n y'(t) = \theta_1^T \phi(t-1) + \bar{e}'(t) \qquad (3\text{-}71)$$

where

$$\phi(t-1) = \{y'(t), \ldots, \delta^{n-1} y'(t), u'(t-1), \ldots, \delta^n u'(t-1)\}^T$$

$$\theta_1 = \{-a_0, \ldots, -a_{n-1}, b_0, \ldots, b_n\}^T$$

Examples of the use of this regression for the incorporation of prior knowledge are given in section 3.5.2.

3.5 Experimental Results

To investigate the usefulness of the δ-transform and the methods of the two preceding sections in an experimental context consider the compliant arm application of section 3.1. The truncated continuous-time model relating applied hub torque u(t) to the tip's angular displacement y(t) is

$$\frac{y(t)}{u(t)} = G(s) = \frac{(s^2 + 2\zeta_o\omega_o s + \omega_o^2)}{Js^2(s^2 + 2\zeta_p\omega_p s + \omega_p^2)} \qquad (3\text{-}72)$$

From a combination of simple experimentation with a signal generator, analytical modelling and estimation performed in the experiments of chapter 5 the resonant frequency ω_p is known to be $2\pi x(13.5\pm0.5Hz)$ $rads^{-1}$ with very light damping $\zeta_p<0.01$. It is far more difficult to obtain prior information about the numerator anti-resonance which is therefore assumed unknown.

Given real data from a typical experiment on the arm (see chapter 5) section 3.5.1 shows how, for z-transform models, regressor filtering can be used to include the double integrators while the functional relation of section 3.2.4 can incorporate the partial knowledge of the remaining complex denominator factor. Section 3.5.2 then shows how the use of the approximate δ-transform model (3-68) simplifies the incorporation of this prior knowledge and leads to the estimation of only four coefficients. To allow comparisons to be made all the experiments are performed on the same data. The convergence properties of the respective constrained estimates are compared and their steady values used to design GPC controllers whose performances on the arm are used as an index of the accuracy of the final models.

3.5.1 Constrained estimation of a z-transform model

The z-transform estimator model corresponding to (3-72) is

$$\frac{y(t)}{u(t-1)} = \frac{\hat{B}(z^{-1})}{\hat{A}(z^{-1})} = \frac{\hat{b}_0 + \hat{b}_1 z^{-1} + \ldots + \hat{b}_4 z^{-4}}{1 + \hat{a}_1 z^{-1} + \ldots + \hat{a}_4 z^{-4}} \qquad (3\text{-}73)$$

with a fourth-order $\hat{B}$ to allow for fractional time delay due to computation. Since the denominator is known to have double integrators this can be reduced to

$$\frac{\Delta^2 y(t)}{u(t-1)} = \frac{\hat{B}(z^{-1})}{\hat{A}'(z^{-1})} = \frac{\hat{b}_0 + \ldots + \hat{b}_4 z^{-4}}{1 + \hat{a}'_1 z^{-1} + \hat{a}'_2 z^{-2}} \qquad (3\text{-}74)$$

and implemented using the regressor filtering method of section 3.3. The partial knowledge of the remaining complex factor $\hat{A}'$, and its incorporation via the functional relation method of section 3.2, is as described in section 3.2.4. With the reduced parameter set of $\hat{A}'$, $\hat{B}$ the relation becomes

$$f(\hat{\theta}) = \hat{\theta}_1 + \cos(\omega h)\hat{\theta}_2 - \cos(\omega h) = 0 \qquad (3\text{-}75)$$

with

$$\frac{\partial f}{\partial \hat{\theta}} = \{\, 1 \;\; \cos(\omega h) \;\; 0 \;\; 0 \;\; 0 \;\; 0 \,\}^T \qquad (3\text{-}76)$$

Fig.3.8 shows the estimated coefficients of $A(z^{-1})$ as they converge over a period of 50 samples. The solid lines represent standard RLS estimating (3-73) with no prior knowledge while the dashed lines show the effect of including the double integrators by the regressor filtering (3-74). The dotted lines illustrate the combined effect of the latter plus the inclusion of the partial knowledge of the denominator resonance via the

functional relation (3-75). The improvement in convergence, particularly through knowledge of the integrators, is quite dramatic.

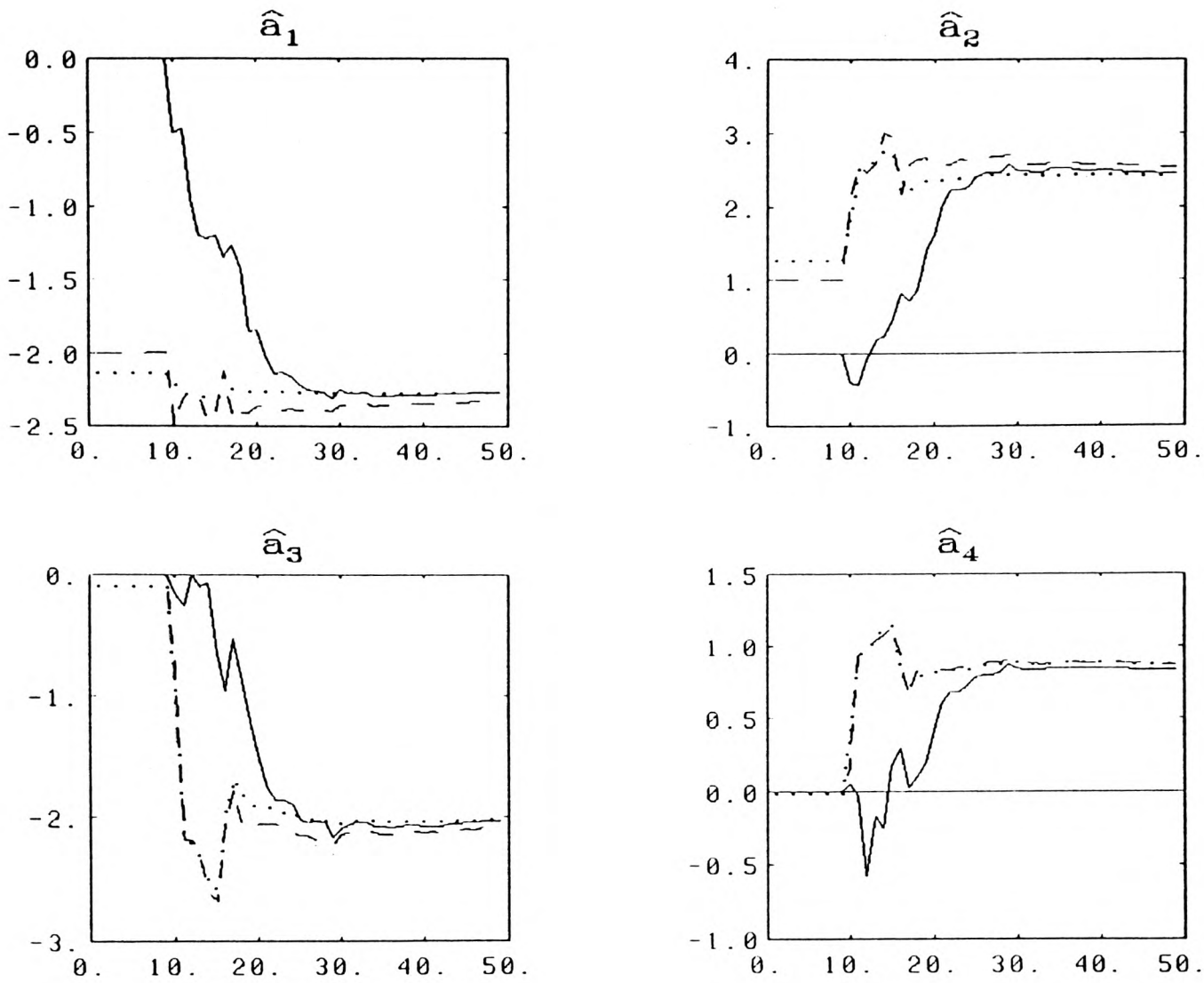


Fig.3.8: Constrained estimation of a z-transform model

An additional reduction in the number of parameters to be estimated may be achieved by using the approximation (3-69) of section 3.4.4 ie.

$$\frac{\Delta^2 y(t)}{u(t-3)} = \frac{\hat{b}'_0 + \hat{b}'_1 z^{-1} + \hat{b}'_2 z^{-2}}{1 + \hat{a}'_1 z^{-1} + \hat{a}'_2 z^{-2}} \qquad (3\text{-}77)$$

together with the constraints discussed above. Fig.3.9 compares the estimated coefficients of $A'(z^{-1})$ in (3-77) (shown solid) with the unconstrained estimates (3-73) (shown dashed) and the estimates (3-74) constrained by knowledge of the integrators and resonant frequency (shown dotted), the latter also being shown by the dotted trace of Fig.3.8. While the approximation (3-77) results in a slightly 'wilder' convergence period than

the 'exact' constrained estimates this is still minor compared to the improvement over unconstrained RLS.

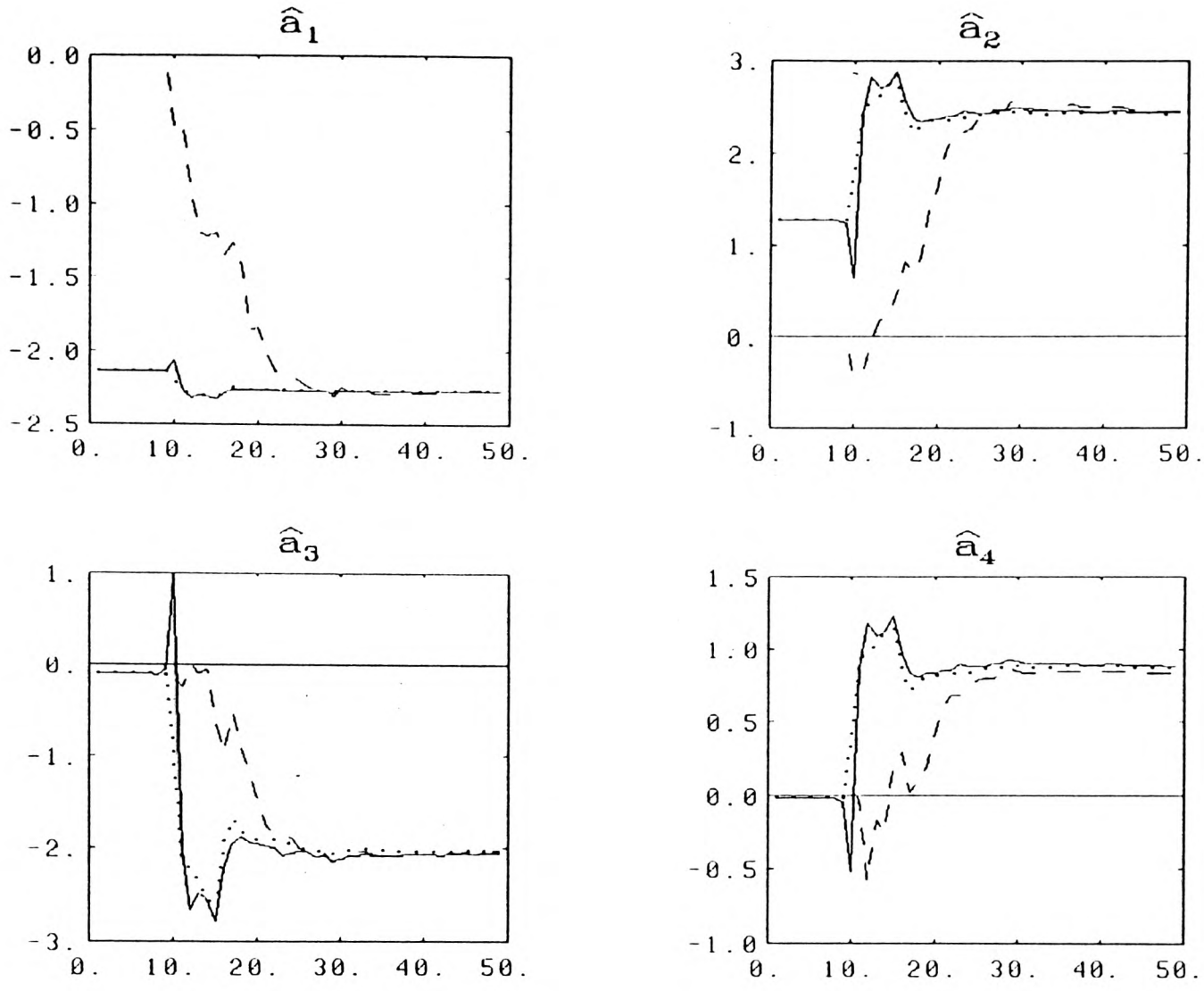


Fig.3.9: Reduced parameterisation for z-transform models

In all four cases the estimator used incremental data with $T_e(z^{-1})=(1-0.6z^{-1})^2$, $P(0)=1000I$ and $\hat{\theta}(0) = \{0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\}^T$ or $\{0\ 0\ 1\ 0\ 0\ 0\ 0\}^T$. In the latter three cases the polynomial multiplication $\hat{A} = (1-z^{-1})^2\hat{A}'$ was used to obtain the $\hat{a}_i$ coefficients at each sample. Fig.3.10 compares the set-point responses of GPC controllers designed using the converged models from the four experiments in the order that they were described. Although the responses are not identical, even allowing for the unrepeatability of a real process, they are sufficiently close to conclude that the modelling has not been degraded by the constraints which must therefore be accurate. Note that the final approximate model (3-77) only behaves slightly differently vindicating the usefulness of its reduced parameterisation.

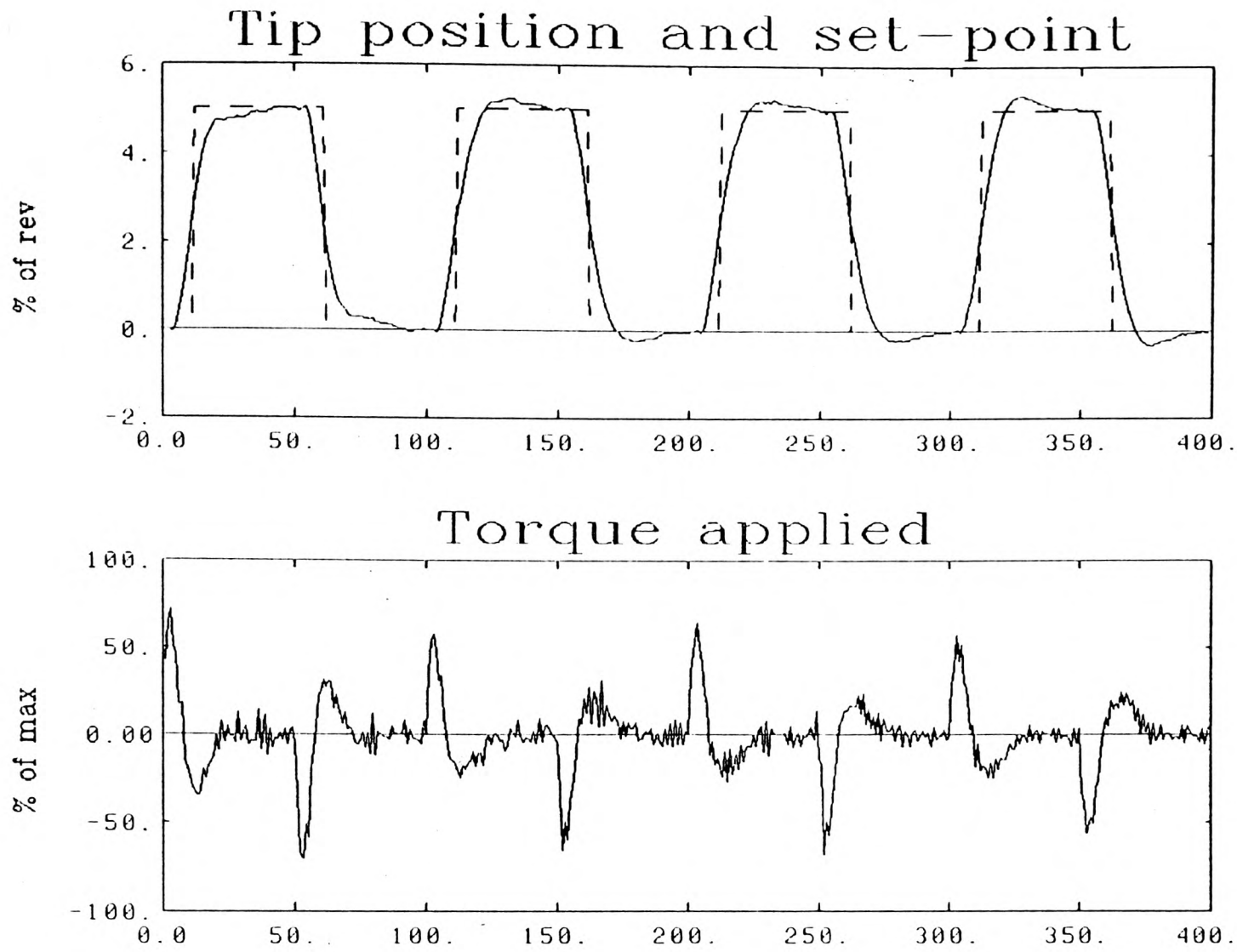


Fig.3.10: GPC validation of constrained estimates

3.5.2 Constrained estimation of a δ-transform model

- The δ-transform estimator model exactly corresponding to (3-72) and (3-73) is

$$\frac{y(t)}{u(t-1)} = \frac{\hat{B}(\delta)}{\hat{A}(\delta)} = \frac{\hat{b}_4\delta^4 + \hat{b}_3\delta^3 + \ldots + \hat{b}_0}{\delta^4 + \hat{a}_3\delta^3 + \ldots + \hat{a}_0} \qquad (3\text{-}78)$$

which may be estimated from the linear regression (3-71) with the data filtered by $1/T(\delta)$ and $\delta/(\delta+\varepsilon)$ as described in section 3.4.3. Given the 50 samples of real data Fig.3.11 shows the estimates $\hat{a}_i$ and $\hat{b}_j$ of (3-78) as they evolve with time. The standard RLS recursion (1-10) to (1-12) was used with

P(0)=1000I, no forgetting, $T(\delta)=(1+0.05\delta)^4$ and $\varepsilon=6$. Given the sample rate of 60Hz the latter two choices correspond (approximately) to low-pass filtering by $T(z^{-1})=(1-0.67z^{-1})^4$ and high-pass filtering by $(1-z^{-1})/(1-0.9z^{-1})$. This is intended to give a similar overall band-pass characteristic to that suggested in section 2.6.3. Note the large differences in the magnitudes of the coefficients, characteristic of δ and s models.

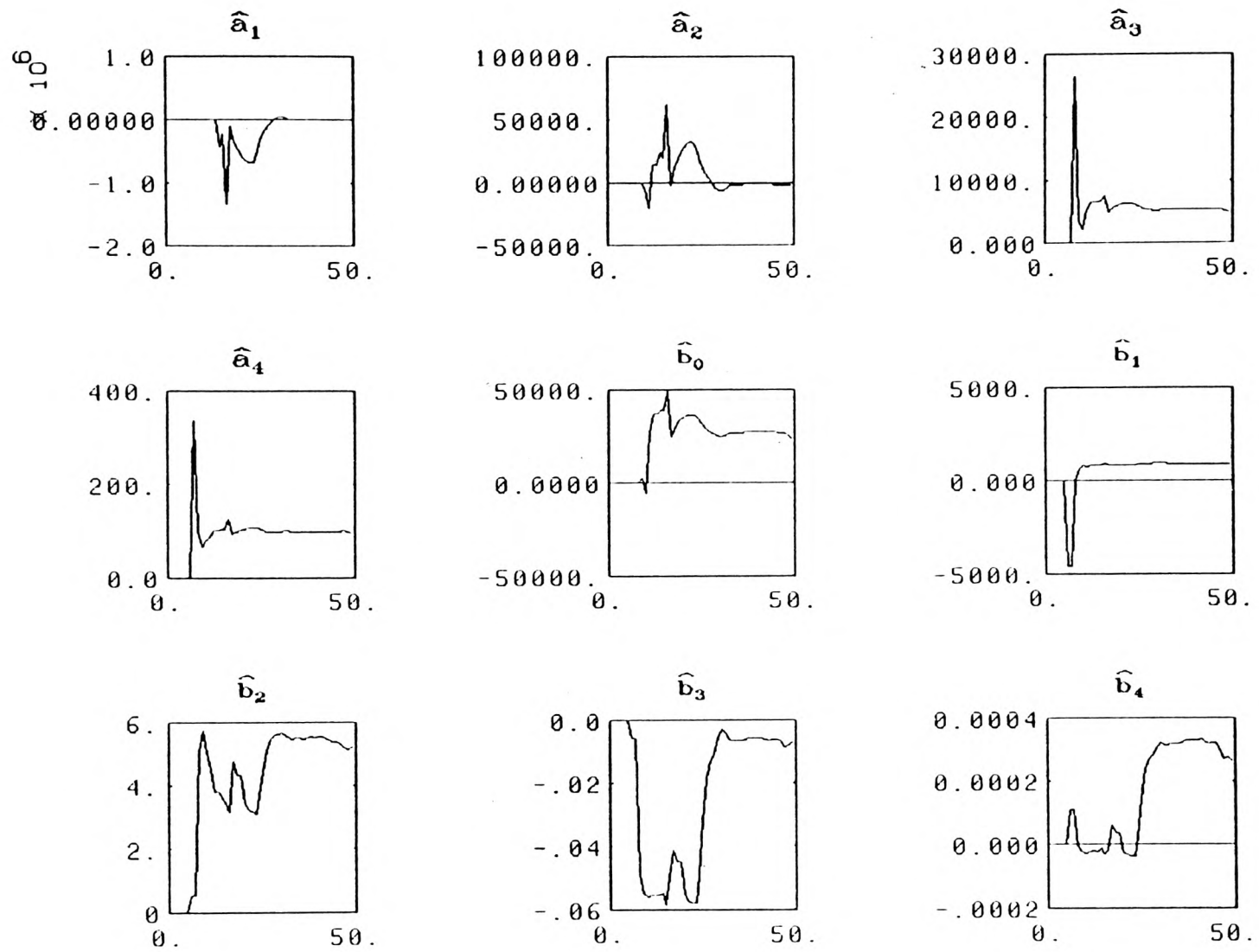


Fig.3.11: Unconstrained estimation of a δ-transform model

° Next consider the inclusion of the double integrators and the known resonant frequency ie.

$$\frac{y(t)}{u(t-1)} = \frac{\hat{B}(\delta)}{\hat{A}(\delta)} \simeq \frac{\hat{b}_4\delta^4 + \hat{b}_3\delta^3 + \ldots + \hat{b}_0}{\delta^2(\delta^2 + \hat{a}_3\delta + \omega_p^2)} \qquad (3\text{-}79)$$

Given the direct root map from A(s) to A(δ) the incorporation of the integrators in (3-79) is exact. The inclusion of ω_p^2 comes however from the

approximation that $A(\delta) \simeq A(s)$ for rapid sampling, this being unaffected by fractional time delay (see section 3.4.4), giving the simplified regression

$$\delta^4 y'(t) + \omega_p^2 \delta^2 y'(t) = -\hat{a}_3 \delta^3 y'(t) + b_4 \delta^4 u'(t-1) + \ldots + b_0 u'(t-1) \qquad (3\text{-}80)$$

with the known quantity $\omega_p^2 \delta^2 y'(t)$ being brought over to the LHS and only six coefficients now to be estimated. Fig.3.12 shows the convergence of this reduced set of estimates with the same filters and initial conditions as before.

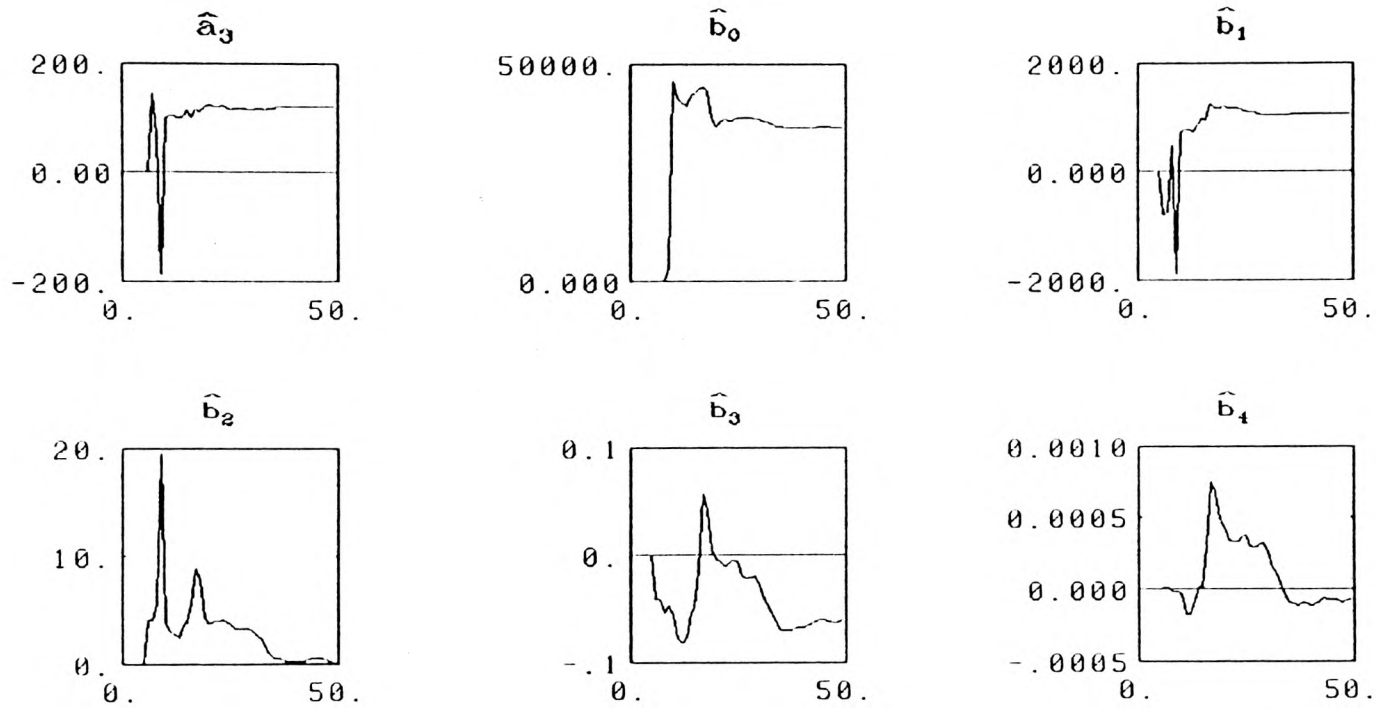


Fig.3.12: Constrained estimation of a δ-transform model

° A final reduction in the number of estimated coefficients may be made by assuming that the 60Hz sample rate is sufficiently rapid to allow the reduced parameterisation of section 3.4.4 to adequately describe the plant ie.

$$\frac{y(t)}{u(t-1)} = \frac{B_3(\delta)}{A(\delta)} \simeq \frac{\hat{b}_2 \delta^2 + \hat{b}_1 \delta + \hat{b}_0}{\delta^2(\delta^2 + \hat{a}_3 \delta + \omega_p^2)} \qquad (3\text{-}81)$$

and only four coefficients to be estimated. Fig.3.13 shows the even more rapid convergence of this limited parameter set with the same filters and initial conditions as in the previous two examples.

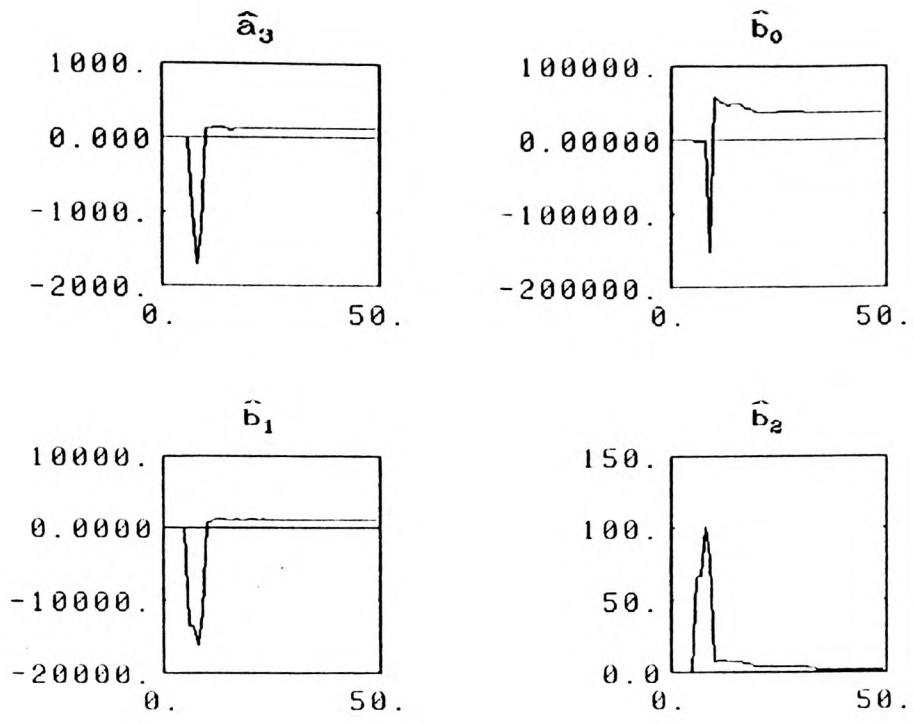


Fig.3.13: Reduced parameterisation for δ-transform models

While the convergence rate has steadily been improved and the computational workload decreased this is only of use if the constrained models (3-80) and (3-81) accurately describe plant behaviour. Fig.3.14 shows a series of set-point responses on the compliant arm with the GPC controller designed on the basis of the converged models (3-79), (3-80) and (3-81) respectively. The responses are virtually identical showing the constraints to be both accurate and effective in practice. In fact they are significantly more uniform than the corresponding responses of the constrained z-transform models of Fig.3.10.

Note that there is no point overlaying Figs. 3.11-3.13. The size of the initial rapid parameter movements is roughly proportional to the size of P(0) relative to the regression data. Since the magnitude of the data changes considerably between experiments, with regressor states being removed, only the convergence rate can sensibly be compared.

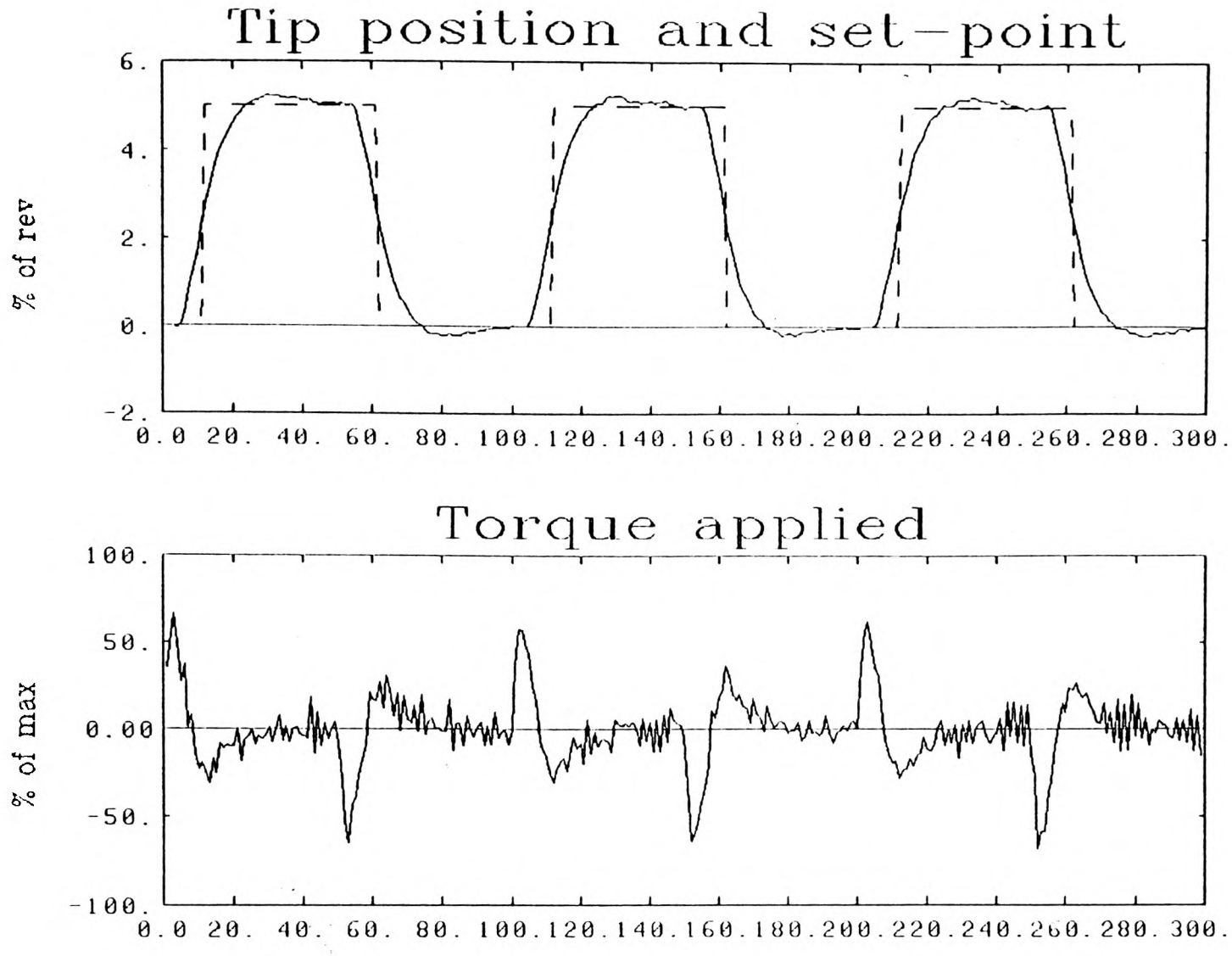


Fig.3.14: GPC validation of constrained estimates

3.6 Conclusions and Further Work

The incorporation of correct prior knowledge of plant dynamics has been seen, in the many simulation and experimental examples of this chapter, to improve the convergence properties of the RLS estimation scheme. The constrained estimates converge quicker, considerably reducing the severity of simulated tuning transients from self-tuning GPC initialised in closed-loop. Sections 3.3.1 and 3.3.2 show that the constrained estimates are also less sensitive to noise and load-disturbances.

It is worth comparing the two methods developed to allow the inclusion of this prior information into the estimator. The principal

advantages and disadvantages of the functional relation method (FRM) and the regressor filtering method (RF) are

° With FRM both the constrained and unconstrained RLS estimates are generated at every sample. It is therefore a simple matter to remove or replace the constraint if the prior knowledge upon which it is based either becomes inaccurate or changes. This is not the case with RF where the regression states, eg. $\phi(t)$ in (3-71), and therefore the covariance matrix $P(t)$ depend upon the known factor.

° FRM is more flexible than RF in that it can incorporate many types of prior information such as known gain, partial knowledge of complex factors etc. while RF can only include known factors.

° Unfortunately FRM can only realistically incorporate a single constraint, see section 3.2 and A.7, whereas RF can include as many factors as are known. These last two points suggest that FRM and RF can usefully be combined as demonstrated in section 3.5.1.

° FRM requires additional calculations in excess of standard RLS while RF actually reduces the computations.

The δ-transform of Goodwin et al (1986) was introduced as a means of aiding the inclusion of prior knowledge at high sample rates. The advantages of using δ may be split into two categories:

(i) Prior information is normally available from continuous-time and pertains to the pole polynomial $\tilde{A}(s)$ in (3-65). Since for rapid sampling $A(\delta) \simeq \tilde{A}(s)$ the experimental example of section 5.3.2 verifies that this considerably simplifies the incorporation of prior information.

(ii) While little is ever known about the zero polynomial $\tilde{B}(s)$ the approximate reduced parameterisation of (3-68) may be interpreted as prior information of the order of $\tilde{B}(s)$. Section 3.4.4 shows how a unit time delay

must be incorporated to deal with the certain presence of fractional time delay from finite computation times at high sample rates. This is shown to be an improvement on the approximation advocated by Goodwin et al (1986).

Chapter 4

HARDWARE AND SOFTWARE

This chapter describes the experimental rig and the controller hardware and software used to support the GPC application study of chapter 5. The control requirement is for accurate high-speed positioning of an extremely flexible single-link arm designed and built by the O.U. Robotics group to study difficulties in controlling compliant systems. Existing implementations of GPC at Oxford, running on a DEC LSI-11/73 micro-computer with hardware floating-point, could only achieve a maximum sampling rate below 1Hz. The compliant arm, however, requires a sample rate well beyond 30Hz for reasons discussed in section 5.1.3. This need for rapid sampling, common to the control of fast electro-mechanical systems such as robots, lead to the hardware/software solution described in the following sections.

The devotion of the majority of a chapter to this solution is justified for the following reasons. Modern control laws, in the pursuit of improved performance, are indubitably becoming more complex with increased computational loads. A typical example of this is the increase in algorithmic complexity from GMV (Clarke & Gawthrop, 1975) to GPC (Clarke et al, 1987). For these algorithms to self-tune at high sample rates modern computing technology must be used. Recognising that the trend in modern high-speed computing is towards parallel processing (eg. transputer), and bearing in mind that a real-time controller is fundamentally task-oriented, the detailed division of an adaptive controller into tasks capable of concurrent execution on separate processors is considered a valuable research objective. This is seldom addressed in the control literature apart from De Keyser et al (1985) who describe a multi-processor system used for a complex real-time simulation of a cutter suction dredging ship.

Section 4.1 describes the mechanics of the arm, its drive system and instrumentation. Derivation of a mathematical model for the arm dynamics has already been briefly dealt with in section 3.1. Section 4.2 provides a brief description of the hardware developed to control this arm while section 4.3 presents a simple task-oriented parallel architecture for the GPC algorithm which, by separating estimation from control, has achieved sample rates beyond 100Hz.

4.1 O.U. Robotics' Single-link Compliant Arm

The arm shown in Figs. 4.1 and 4.2 was designed and built for two principal reasons. Firstly, to act as a test-bed for advanced control schemes applied to a compliant system and secondly to study the perturbations in the modal shapes and modal frequencies caused by changing the end mass. The latter is important in the context of a working compliant robot arm performing, say, a pick and place operation.

4.1.1 Mechanical components of the Arm

- The motor, a G9M4 printed armature DC servo motor, directly drives the shaft to which the compliant link is attached. It was chosen as a high quality, high torque-to-weight motor with a virtually constant current-to-torque characteristic over a wide speed range. Having no gearbox has the advantage of avoiding gearbox backlash and friction. However the arm loading is supported on planar bearings within the motor which means that a substantial amount of rotational friction is present.

° The link itself is made from CS80 hardened and tempered spring steel which can undergo reasonable deformations while remaining elastic. The dimensions were chosen to avoid torsional effects, ie. twisting about the

Fig.4.1: O.U. Robotics' Single-link Compliant Arm

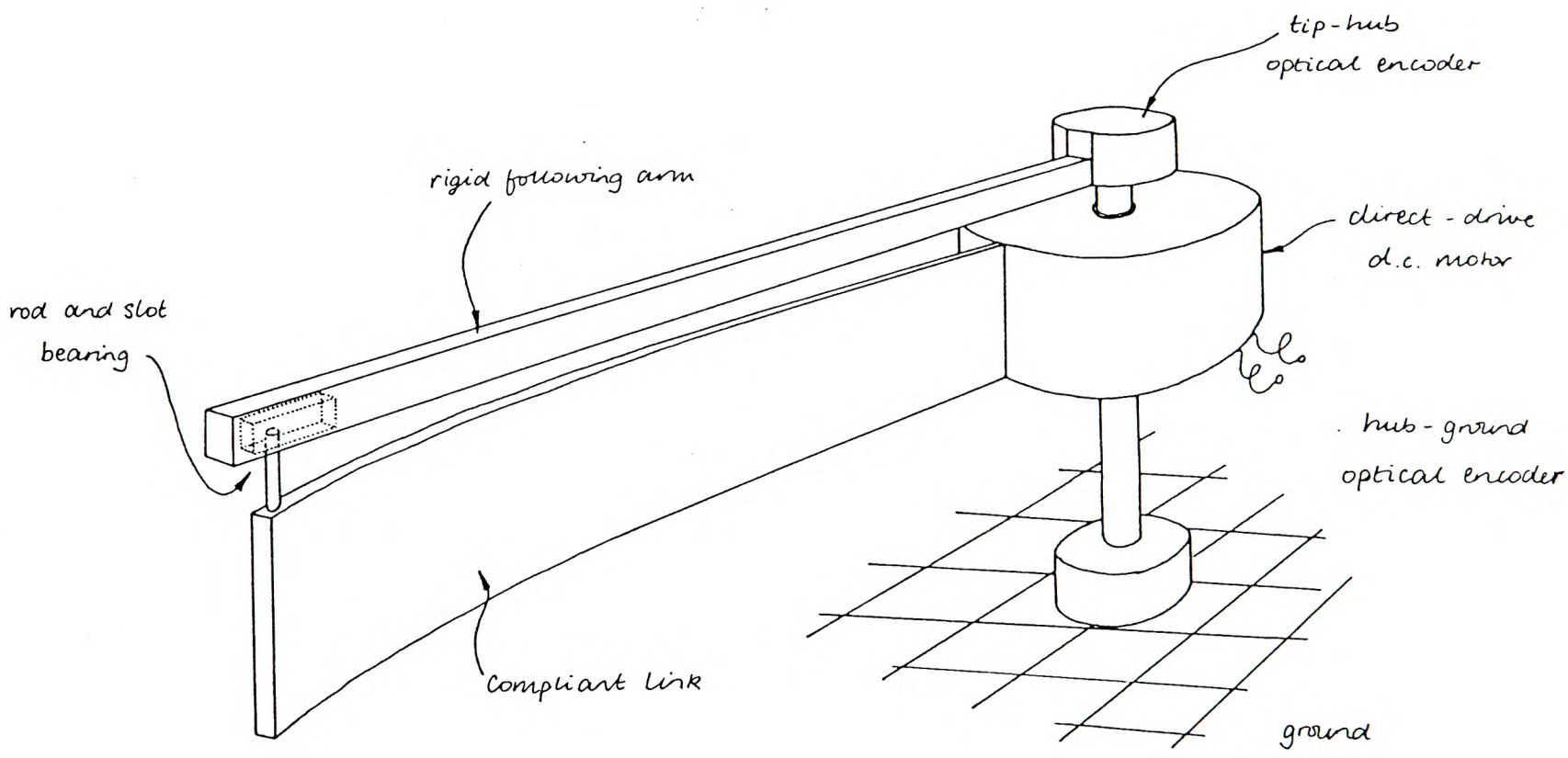


Fig.4.2: Components of the Compliant Arm

longitudinal axis of the link, while allowing 'slender' approximations in any analysis.

° The instrumentation used on the arm is:

(i) A small optical encoder mounted below the motor whose housing is fixed to the motor housing and whose shaft is fixed co-axially to the motor shaft. This provides 4000 counts per revolution ie. an angular resolution of 0.09° for hub-to-ground angular displacement. This encoder is inherently digital with the consequence that the only sensor noise present is due to quantisation.

(ii) An identical optical encoder is mounted above the motor with its shaft again fixed co-axially to the motor shaft. The body of this encoder is fixed to the following arm as shown in the diagram. This aluminium **following arm** is fixed to the end-point of the compliant arm while at the hub it is free to rotate relative to the motor shaft. The rod and slot arrangement shown in the inset of Fig.4.2 allows the compliant arm to flex (shortening the tip-to-hub distance) as the rod moves in the follower slot. This gives 0.09° resolution of tip-to-hub angular displacement. The sum of the two encoder positions clearly gives the absolute tip-to-ground angular displacement to a net resolution of 0.18°, a linear deflection of ≃ 1mm.

(iii) Strain gauges are fitted to the compliant arm near to the base. Their reading is directly related to the bending moment at the root which is itself a measure of the difference between the motor torque input and the hub acceleration multiplied by the hub inertia. These were not actually used in the experiments conducted in this thesis.

4.2 Computer Hardware

The high-speed computer system shown in Figs. 4.3 and 4.4 is based around a VME backplane and incorporates multiple 68010/20 processor boards,

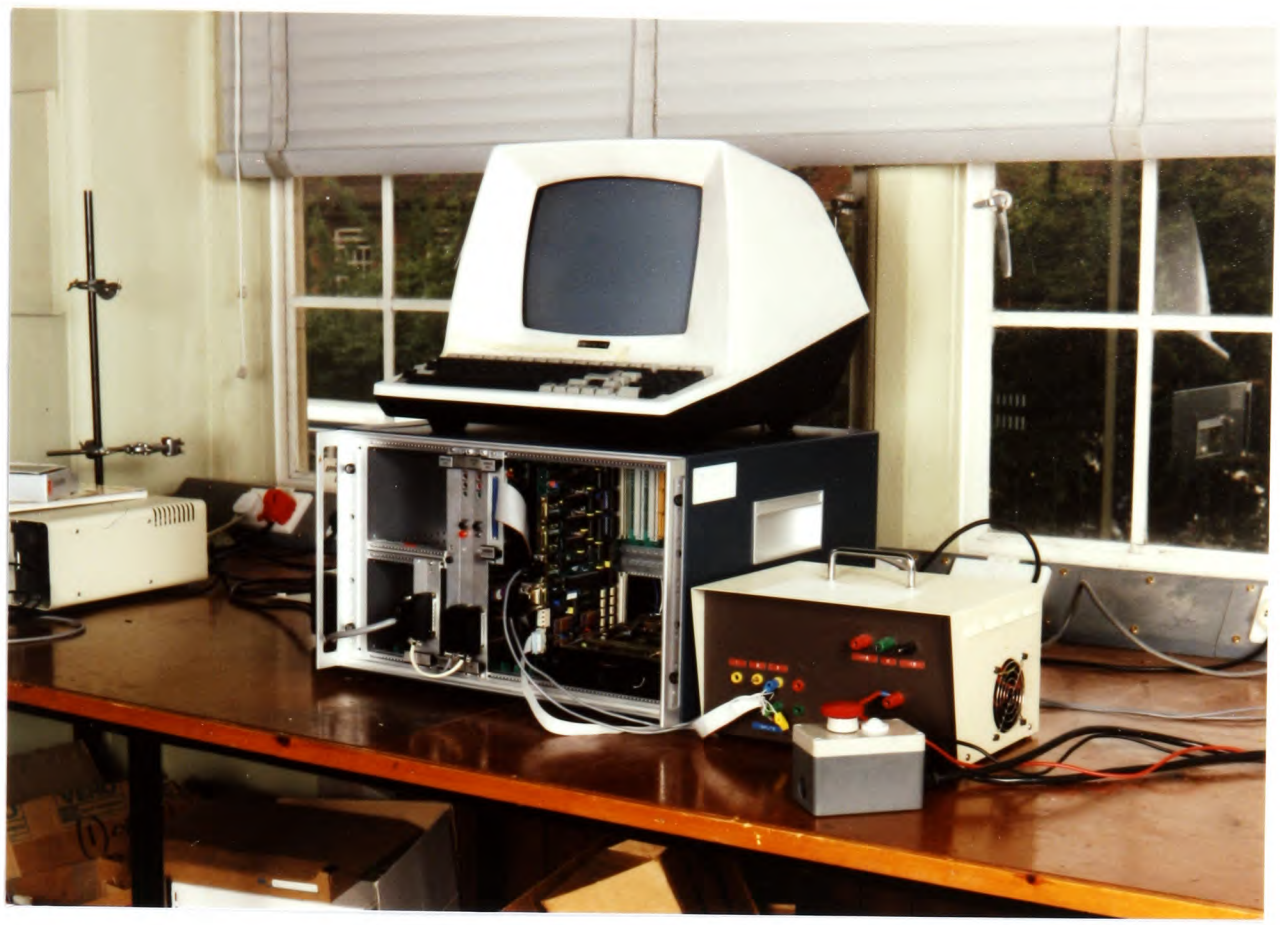


Fig.4.3: The VME multi-processor system

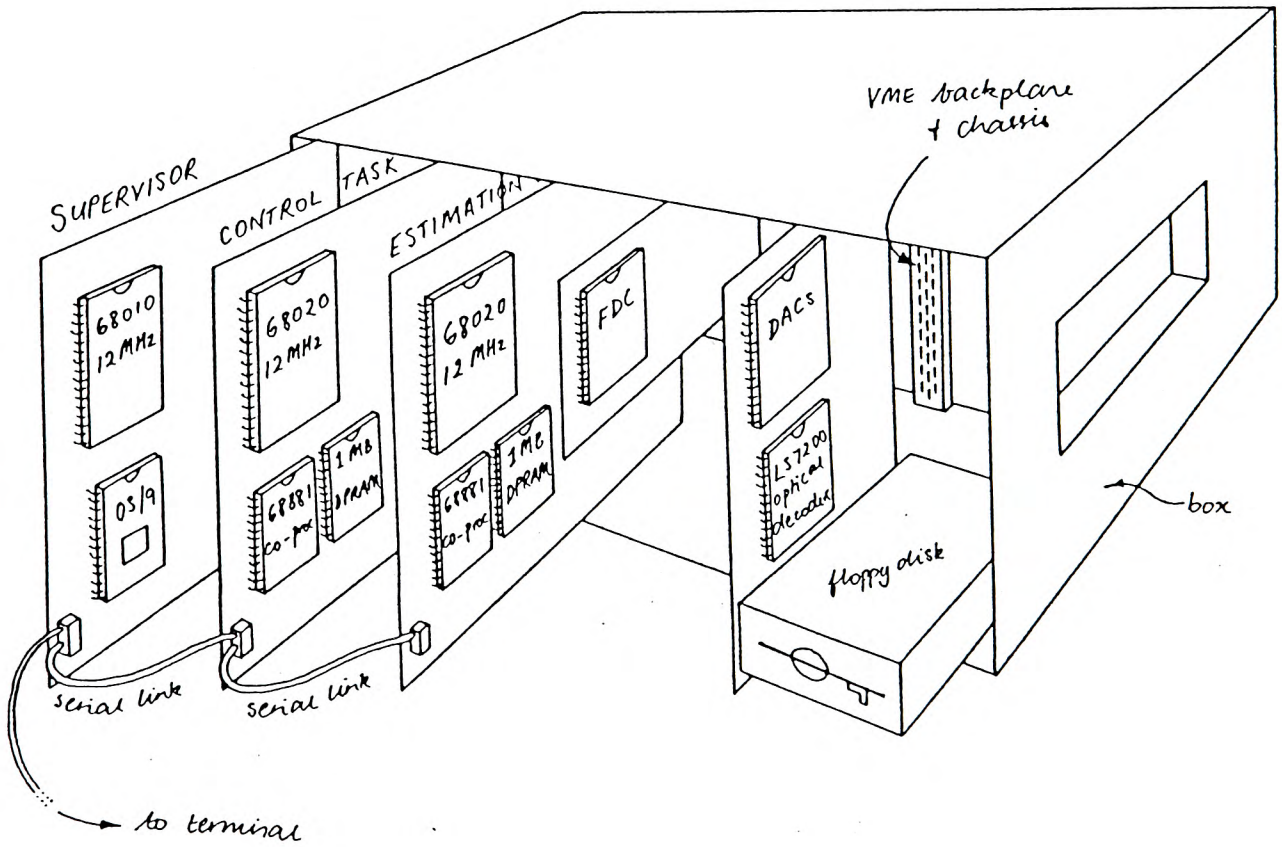


Fig.4.4: Components of the VME system

an analogue i/o board and a floppy disk controller board and disk drive. These processors are subdivided into one SUPERVISORY processor and many TASK processors. The main difference is that the SUPERVISOR supports a 'friendly' disk-based high-level operating system while the TASK processors need only support a simple debug monitor. The SUPERVISOR, using the high-level utilities of an operating system looks after non-time-critical tasks such as downloading programs to the TASK processors' dual-ported RAM, run-time supervision and post-run-time analysis. The TASK processors, fast boards with hardware floating-point, execute this high-speed control/estimation code during run-time. The absence of an operating system on these boards reduces their cost and increases execution speed, high-level routines such as formatted terminal i/o not being required.

Each processor has its own on-board memory for program and data storage. The SUPERVISOR and TASK processors can communicate with each other and with any peripheral devices, such as the analogue i/o board, over the VME bus. The TASK processors have dual-ported RAM, which may be accessed both by the on-board processor and from the VME bus, to facilitate high-speed inter-processor communication via mail-boxes (section 4.3.1.3). The boards are prioritised in their access to the VME bus with the TASK processors, requiring unimpeded run-time access to the analogue i/o board and each other's DPRAM, being higher priority than the SUPERVISOR processor. Multi-function peripheral chips on the TASK processor boards allow the implementation of real-time interrupt driven control software. Further details of the organisation of this system are given in the task-division of section 4.3. In the following sections the hardware elements are described individually together with the rationale governing their choice.

Note that the code for these SUPERVISOR and TASK processors was developed on a separate commercial VME 68010 development system and ported across to the target system via the floppy disk. This convenience was

possible since the SUPERVISOR processor, operating system, disk drive and controller are functionally identical to those from the development system.

4.2.1 The VME bus

The VME protocol arose from the need for an interfacing system to interconnect data processing, data storage and peripheral control devices in a closely coupled hardware configuration. It sets an electrical and mechanical standard to which a plethora of processor boards and their peripherals adhere. For those boards which obey the protocol it provides:

- High-speed fully asynchronous parallel data transfer over the 32 bit Data Transfer Bus (DTB). Four gigabytes of memory are directly addressable from the 32 bit address bus.
- Orderly transfer of the DTB from one MASTER to another via a Bus Arbitration Module (section 4.2.2) which guarantees that only one device controls the DTB at a given time. Furthermore the arbitration module allows prioritisation of DTB masters with four priority levels each driving a daisy-chain. This allows multiple processors to share global resources and, since the VME allocation is hardware-based, speed of allocation is high.
- Prioritised interrupt handling where devices can request service from an INTERRUPT HANDLER. These interrupt requests may be prioritised into a maximum of seven levels. There may also be more than one interrupt handler, each servicing only a subset of the bus interrupts.
- Utilities such as clock lines, bus initialisation (RESET) and failure detection eg. if a MASTER accidentally addresses a location where there is no SLAVE the BUS TIMER intervenes and terminates the cycle.

The functional modules which comprise a typical VME system are illustrated in Fig.4.5. The VME bus was chosen in preference to alternative bus structures primarily because of the availability of a VME analogue i/o

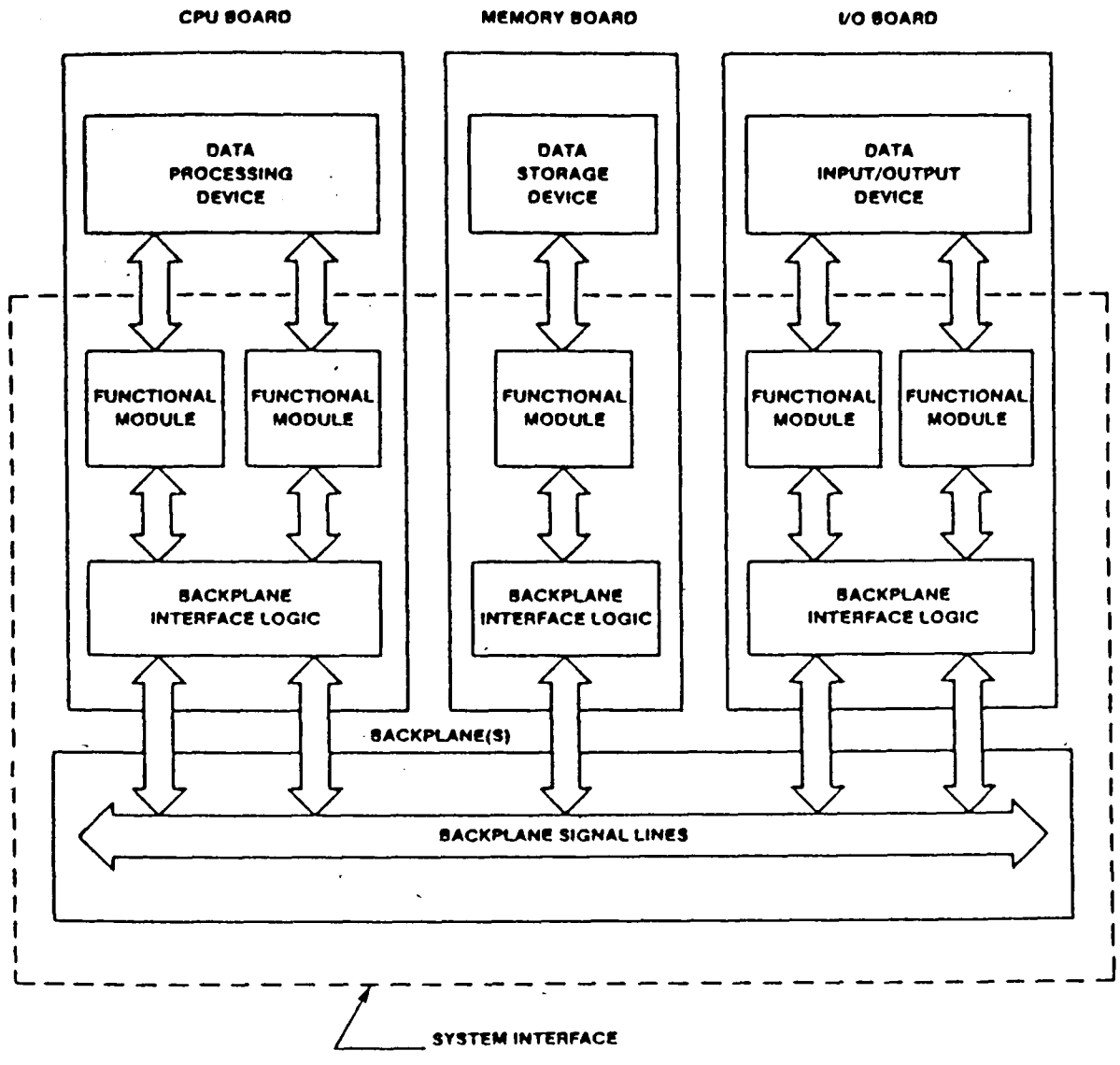


Fig.4.5: Functional modules on a VME bus

peripheral board (section 4.2.4) combining counter/decoders, for the outputs from the optical encoders on the compliant arm, with digital-to-analogue converters (DACs). A secondary merit was the close association of the VME bus with the Motorola 680X0 family of processor boards, especially the new high-speed 68020/68881 processor/coprocessor board. In practice the VME bus appears as a printed-circuit backplane onto which VME-specification boards mate. The VME specification also extends to the card-frame, the enclosure and even to the power supply.

Since the analogue board was a 16-bit peripheral (designed for the older 68000 processor) it was only necessary to construct a single-height 'J1' backplane VME system. This is a sub-set of the full VME implementation which differs only in that the DTB is reduced to 16 data lines and 24 address lines.

4.2.2 The Supervisory Processor

The supervisory processor occupies slot 1 of the backplane and therefore, to obey the VME protocol, must provide the bus arbitration logic (DTB transfer and interrupt acknowledge) for the multi-processor system. This logic acts independently of the on-board CPU and may be considered to occupy an imaginary slot 0 as shown in Fig.4.5. The remainder of the board minus the arbitration logic then behaves as a normal processor board at the head of the bus grant and interrupt acknowledge daisy chains.

The processor board chosen to be the SUPERVISOR was a Motorola 68010 board with 512K of RAM running at 12.5 MHz without floating-point hardware. It was chosen because, as noted in the introductory section, it supported the same operating system, OS9/68000, as the development system. This allows transferral of executable code from development system to target system via identically formatted floppy disks and also quick development of high-level programs capable of using the SUPERVISOR operating system's built-in utilities.

4.2.3 Task Processors

The TASK processors occupy slots directly below the SUPERVISOR and, consequently, are not required to provide bus arbitration logic. The boards are based on Motorola 68020 CPUs with 68881 floating-point coprocessors and have 1MB of on-board dual-ported RAM. As the 68881 is mapped directly into the CPU register space and not accessed as an addressable peripheral it offers a large speed increase over floating-point implemented in software. On-board jumpers enable the dual-port RAM to be mapped into the VME address space. To the on-board CPU the RAM appears at addresses $0\text{-}FF,FFF_H$ whereas to other boards on the VME bus the same RAM could appear at addresses $300,000_H$-

$3FF,FFF_H$ as illustrated in the VME address map of Fig.4.6. An on-board dual-port RAM controller ensures that the same RAM location is never accessed simultaneously by the CPU and from the VME bus. The TASK processors also have a Motorola 68901 multi-function-peripheral chip which allows real-time processing via a bank of programmable counter/timers capable of generating interrupts. Code for these TASK processors is compiled and linked on the development system but stored in a raw binary format, not the modular format normally generated for programs running under the OS9 operating system, and transferred across to the target system on floppy disk. Before run-time the program is loaded byte-by-byte from disk directly to the relevant TASK processor's dual-port RAM by a SUPERVISOR utility (down_load() of section 4.3.2.2).

4.2.4 Analogue i/o

The TJP-AN02 analogue board comprises 4 D/A channels with 12-bit resolution over a -10 to +10V range together with 4 counter/decoder channels for the optical encoder outputs and some unused parallel i/o. Although the board has a periodic interrupt capability this was not used since the decoder channels are basically continuously running counters which can be read on demand. The DACS have a maximum settling time of 4μs which is far in excess of requirement. The encoders provide 4000 pulses per revolution and the counter/decoders have a 16 bit range ie. they can count for 16 revolutions without 'wrapping round' and giving a spurious result. The counters are always initialised to 32,767 to allow 8 revolutions in either direction. The board is activated by a DTB MASTER writing to, or reading from, an address whose high-order bits address the analogue i/o board (see Fig.4.6) and whose low-order bits select the required DAC or decoder.

4.2.5 The VME Address space

All the preceding boards are mapped into VME address space by appropriate selection of on-board jumpers or DIP switches. The VME addresses of the various boards' registers, dual-port RAM, program areas, mail-boxes etc. are shown in Fig.4.6. To access VME address space a processor must basically attempt to read from, or write to, an off-board address. The SUPERVISOR's bus arbitration logic interprets this as a bus request and if the bus is not being used by a higher-priority MASTER data transfer can then take place.

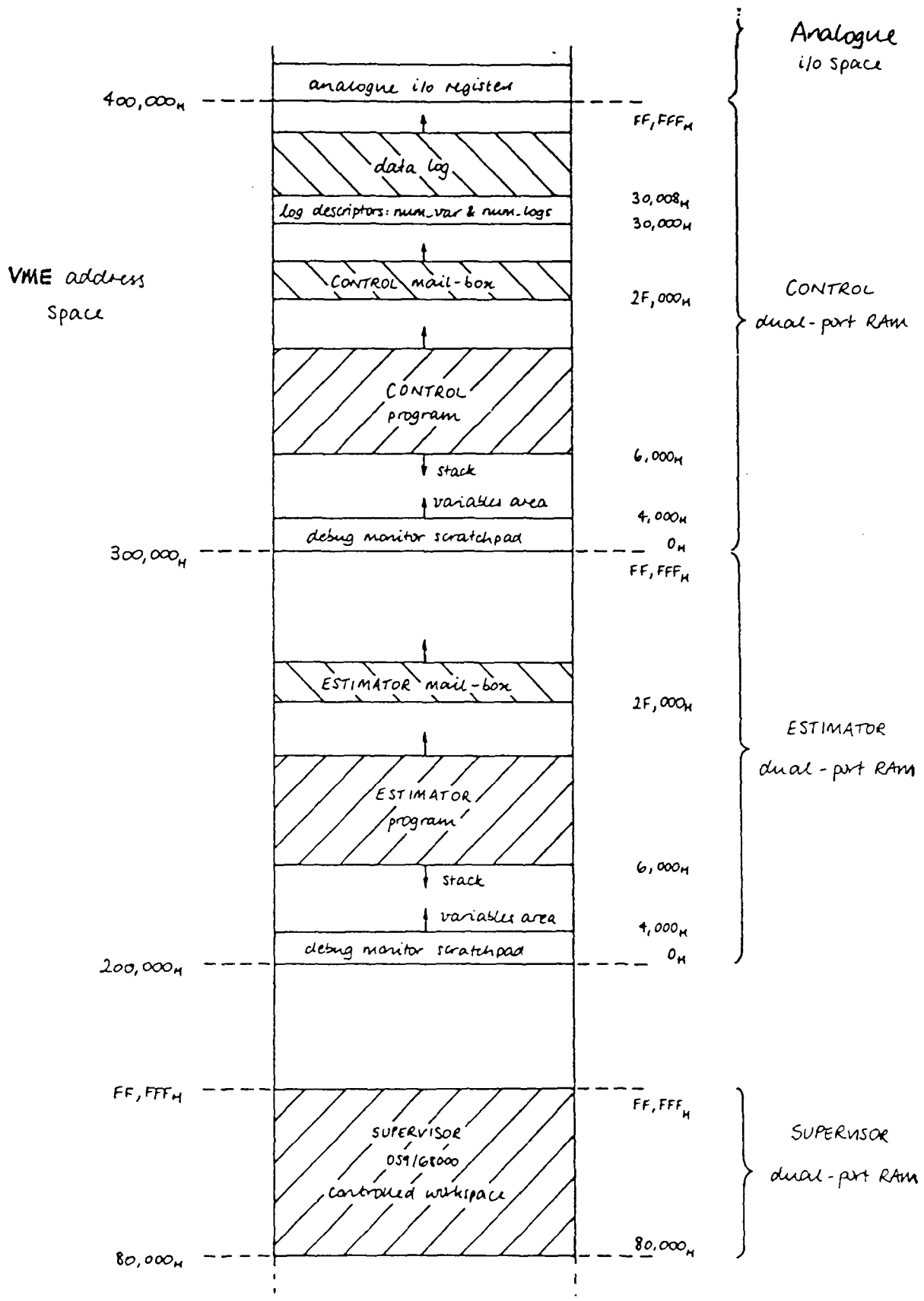


Fig.4.6: VME Address Map

4.3 Task-Division for the GPC Self-tuner

Due to the constraints on execution speed mentioned in the introduction to this chapter the GPC algorithm was subdivided into a multitude of tasks; firstly to those executing concurrently on separate processors, and then further to those executing sequentially on the same processor. The main division between processors is shown below in Fig.4.7.

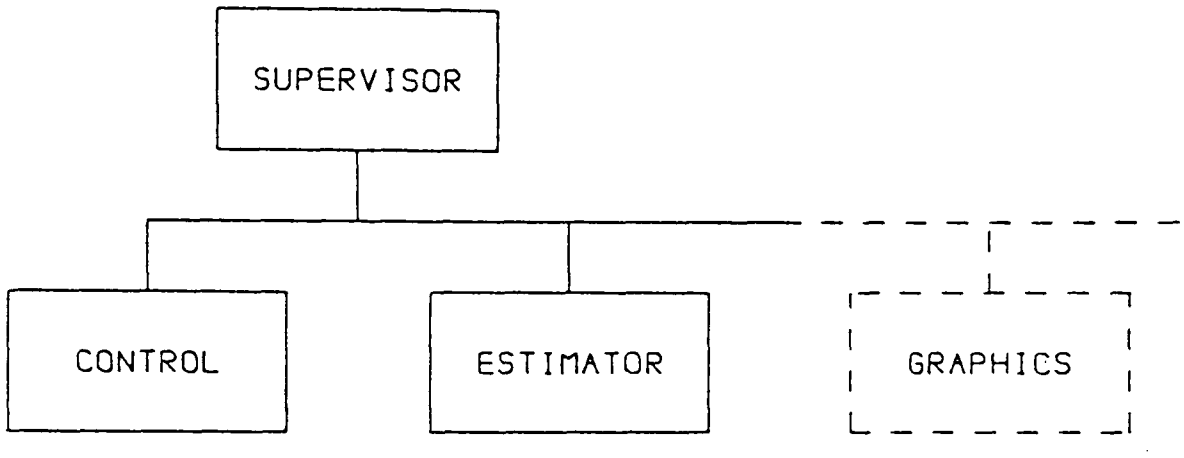


Fig.4.7: Task division between Processors

The majority of the computational burden of a self-tuning controller arises from the control calculation, estimation update and controller redesign. The ancillary requirements of analogue i/o, synchronisation, data-logging etc. account for only a small fraction of the total CPU time. Optimal task division, in terms of speed, would require all the concurrent run-time tasks to complete in the same time (Allworth, 1981). These considerations are reflected in the task structure adopted for this multi-processor application. The CONTROL processor is responsible for the control calculation and all ancillary tasks while the ESTIMATOR processor performs the on-line estimation and controller redesign. This structure allows the SUPERVISOR and CONTROL processors to run together as a fixed-parameter controller, independent of the ESTIMATOR processor, which greatly simplified development.

The SUPERVISOR processor is mainly responsible for pre- and post-run-time requirements such as program download to the TASK processors, graphical

data analysis and permanent storage of data logs to disk. However, during run-time the SUPERVISOR processor also provides the set-point trajectory in real-time and monitors for a keyboard abort. These divisions are summarised in Fig.4.8.

	SUPERVISOR	CONTROL	ESTIMATOR
PRE-RUN-TIME:	• Program download • Task initialisation	• Wait For 'GO'	• Wait For 'GO'
RUN-TIME:	• Pass set-point trajectory to CONTROL task • Monitor keyboard Abort	• Analogue input/output • Echo i/o data to ESTIMATOR • GPC control calculation • Log to high memory	• Wait For new i/o data • Recursive estimation • Controller re-design • Transfer new controller parameters to CONTROL
POST-RUN-TIME:	• Error analysis • Graphics Dump • Transfer CONTROL logs to disk • VAX communication	• Wait For 'GO'	• Wait For 'GO'

Fig.4.8: Task division for the GPC self-tuner

This structure for GPC should be equally applicable to most alternative self-tuning control strategies. It is aimed at the next generation of parallel processors and languages (eg. transputers and OCCAM), with the following sections detailing the successful software engineering of a real-time parallel-processing self-tuner. Section 4.3.1 provides an overview of the task structure after which, in sections 4.3.2-4.3.4 respectively the SUPERVISOR, CONTROL and ESTIMATOR tasks and component sub-tasks are described.

4.3.1 Overview of the Task-Division

The SUPERVISOR, CONTROL and ESTIMATOR tasks of Fig.4.7 were actually implemented while the GRAPHICS task is shown as a possible expansion. Rather than proceeding directly to the complexity of individual tasks and sub-tasks

this section presents an overview of how the run-time SUPERVISOR, CONTROL and ESTIMATOR tasks (see Fig.4.8) work together to achieve the rapid sampling which made the experiments of chapter 5 possible. Section 4.3.1.1 shows how the control and estimation and controller re-design calculations may be divided into concurrently executing tasks while section 4.3.1.2, with reference to a state transition diagram, summarises the interactions between the processor tasks during run-time. The method by which secure interprocessor communication is implemented is then given in section 4.3.1.3. Details of the pre- and post-run-time activities are given in the later sections on the SUPERVISOR sub-tasks gpc_super() and gpc_init().

4.3.1.1 Separation of Estimation from Control

The division of the self-tuning GPC calculation between the CONTROL and ESTIMATOR tasks merits further attention. Chapter 2 left the incremental GPC control calculation as

$$\tilde{\delta u} = (G_1^T G_1 + \lambda I)^{-1} G_1^T (w - \hat{y}_f) \qquad (4\text{-}1)$$

where G_1 is a matrix composed of the ordinates of the step response of the estimated model $\hat{A}$, $\hat{B}$ from zero state, w is the vector of future set-points and $\hat{y}_f$ is the vector of predicted future outputs from current state with no future control increments applied. Since only the first control $\Delta u(t)$ of $\tilde{\delta u}$ will be exerted it is only the top row of the matrix $(G_1^T G_1 + \lambda I)^{-1} G_1^T$ that is actually required. The matrix G_1 only depends on the estimated model $\hat{A}$ and $\hat{B}$ whereas $\hat{y}_f$ depends both upon $\hat{A}$ and $\hat{B}$ and upon the current dynamic state of the plant. Thus the control calculation (4-1) can be divided into the controller redesign, where the top row of $(G_1^T G_1 + \lambda I)^{-1} G_1^T$ is calculated from new estimates $\hat{A}$ and $\hat{B}$ and the control calculation where $\hat{y}_f$ is calculated

given the latest estimates, $(w-\hat{y}_f)$ is formed and the scalar product of $(w-\hat{y}_f)$ with the top row of $(G_1^T G_1+\lambda I)^{-1} G_1^T$ gives the new control increment $\Delta u(t)$.

4.3.1.2 Run-time Task-Division

Fig.4.9 presents a state transition diagram which describes how the concurrently executing tasks of the three processors interact during run-time, this being the most complex phase for the software. The diagram follows Allworth (1981) with numbered boxes representing processing states and non-numbered boxes representing actions. Flag shaped boxes depict flag variables or 'semaphores' resident in either the CONTROL or ESTIMATOR mail-boxes. Whenever an action sets a flag the relevant flag is placed adjacent to the action. Program flow from states or actions depends upon the values these flags take, as shown by the arrowed lines. An asterix means "anything else".

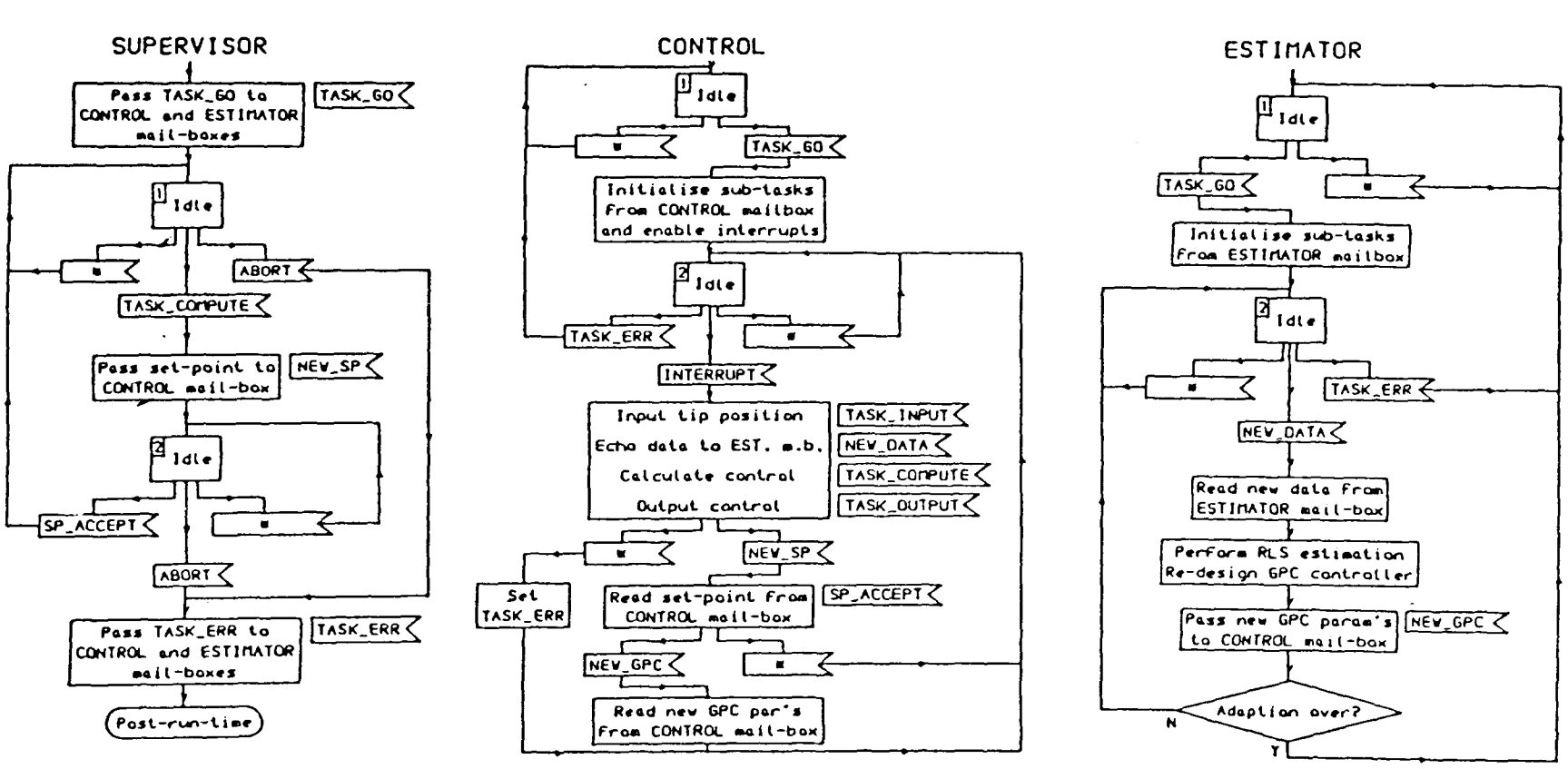


Fig.4.9: Run-time state-transition diagram

The SUPERVISOR initiates run-time by setting the flag [TASK_GO] in both the CONTROL and ESTIMATOR mail-boxes. Both the CONTROL and ESTIMATOR

tasks, if activated in pre-run-time*, then proceed from their starting idle state to initialise their internal variables from data passed to their respective mail-boxes by a pre-run-time SUPERVISOR sub-task. The CONTROL task also enables its on-board interrupts to 'tick' at the appropriate sample rate. Both CONTROL and ESTIMATOR tasks then enter a second idle state. The CONTROL task's idle is broken either by an error condition being flagged, say due to a keyboard abort trapped by the SUPERVISOR, or by an on-board interrupt. On receiving an interrupt the CONTROL task reads the encoders, echoes this and additional regression data to the ESTIMATOR mail-box, calculates the new GPC control u(t) and passes it to the DACs. The sequence of these actions is flagged in the CONTROL mail-box for synchronisation purposes. If at the end of this i/o and calculation the arrival of a new set-point from the SUPERVISOR has not been flagged the flag [TASK_ERR] is set causing the CONTROL task to return to its first idle state. If the set-point does arrive the CONTROL task acknowledges its receipt by setting [SP_ACCEPT] and, if the parameters representing a redesigned GPC controller have arrived, reads them from the mail-box. This latter arrival is not as critical as the new set-point, and processing continues in its absence. This completes the main CONTROL cycle which returns to the second idle state till the next interrupt or error.

During run-time the SUPERVISOR task oscillates between two idle states which both react to a keyboard abort by setting [TASK_ERR] in both the CONTROL and ESTIMATOR mail-boxes and proceeding to a post-run-time phase. In the first idle state it polls the CONTROL mail-box waiting for the task to start calculating u(t), as flagged via [TASK_COMPUTE], upon which it sends the next set-point sequence w(t) to the CONTROL mail-box. Then SUPERVISOR enters its second idle state, waiting for CONTROL to acknowledge receipt of the set-point. When this is flagged via [SP_ACCEPT], the SUPERVISOR returns to the first idle state. This procedure ensures that the SUPERVISOR task,

* see gpc_init() of section 4.3.2.2.

with a minimal run-time workload, only sends one set-point sequence to the CONTROL task per sample.

Meanwhile, if active, the ESTIMATOR task sits in an idle state waiting either for an error to be flagged via [TASK_ERR] or for the arrival of new data from the CONTROL task to be flagged by [NEW_DATA]. Any error condition causes the ESTIMATOR task to return to its initial idle state. When the arrival of new data is flagged ESTIMATOR reads it in from its mail-box, performs RLS estimation and redesigns the GPC controller. The new GPC parameters are immediately written to the CONTROL mail-box and their arrival flagged via [NEW_GPC]. Then, unless the adaption duration has been exceeded, the ESTIMATOR task returns to its main idle state. Given the communication protocol explained in the next section the ESTIMATOR task can proceed completely asynchronously to the CONTROL task.

4.3.1.3 Communication between Tasks

Data transfer between the parallel-executing SUPERVISOR, CONTROL and ESTIMATOR tasks (Fig.4.9) is achieved through small areas of dual-ported RAM, known as mail-boxes (Holt et al, 1978), resident on the respective processor boards. Data from a source processor is first transferred to the destination mail-box and then, later, from there to the destination's internal variables. Direct transferral of data from one task's internal variables to another's internal variables is avoided since, say during the transfer of a matrix, the destination processor may itself be using the matrix while it is being altered. The intermediate storage in the mail-box ensures that the entire matrix arrives without the destination processor attempting access and, subsequently, the destination processor can copy the matrix to its internal variables at a convenient time when it does not require to access the matrix.

Flag bytes within the mail-box are used to signal and acknowledge the arrival of new data in a simple protocol (Holt et al, 1978) which ensures

secure communication. The two actions of reading from and writing to a mail-box are schematically explained in Fig.4.10.

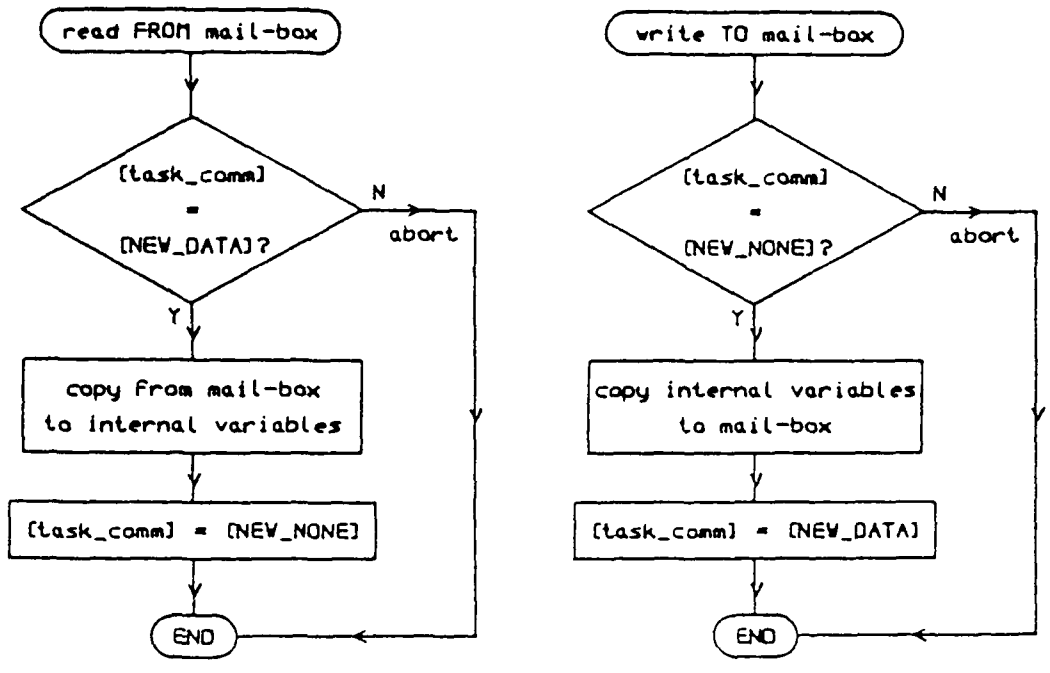


Fig.4.10: Mail-box data transfer protocol

The flag [task_comm] is used to signal the complete arrival of new data from the source and later its acceptance by the destination. There may be several of these flags per mail-box to separate different data parcels (eg. set-points and new estimates). It should be apparent that this protocol excludes the possibility of a simultaneous read and write from a mail-box. However it should be noted that it does not exclude the possibility of two processors attempting to write to the same mail-box at the same time. If the system were to be expanded this could become a problem and a more complex protocol would be required.

4.3.2 The SUPERVISOR Task

The SUPERVISOR task is the most complex of the main processor TASKS and comprises three distinct phases as shown in Fig.4.8. It is quite different from the CONTROL and ESTIMATOR tasks, being the only task to communicate with the user via the terminal. It is itself composed of sub-tasks shown schematically in Fig.4.11 and described in the following

sections. For each sub-task a schematic is given outlining the flow of execution together with a brief description of its purpose.

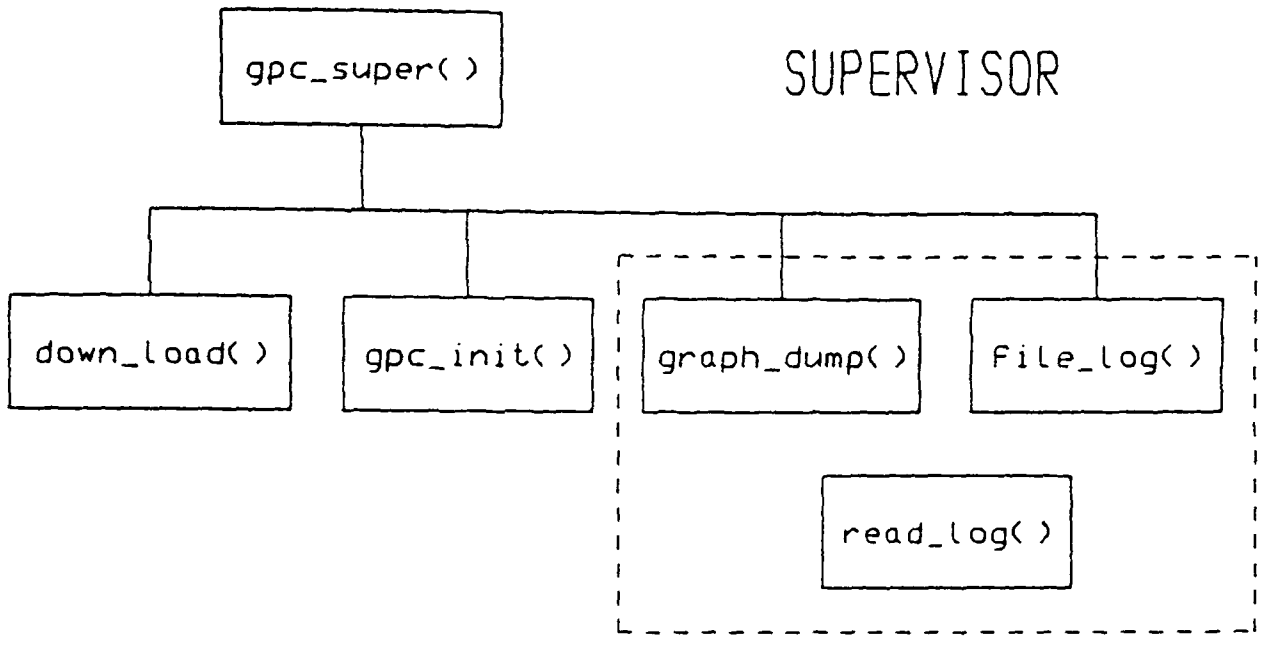


Fig.4.11: SUPERVISOR sub-tasks

The SUPERVISOR task effectively simulates the upper levels of a hierarchical robot control system. These upper levels would be responsible for interaction with the user, high-level sensing of the work environment (eg. via a vision system), path planning and inverse-kinematic calculations for multi-link robots. The set-point trajectory coming from the path planner and inverse-kinematics task would then be passed in real-time to the lowest or servo level.

4.3.2.1 SUPERVISOR sub-task: gpc_super()

As shown in Fig.4.11 the main sub-task gpc_super() coordinates the other SUPERVISOR sub-tasks down_load(), gpc_init(), graph_dump(), and file_log(). It is menu driven with the options listed below:

- Menu_0 activates the sub-task down_load() which, prior to run-time, loads the CONTROL and ESTIMATOR executable code to the dual-port RAM of the respective TASK processors.

° Menu_1 activates the code in the ESTIMATOR processor's DPRAM (see section 4.3.4.1).

° Menu_2 activates the code in the CONTROL processor's DPRAM (see section 4.3.3.1).

° Menu_3 uses the KERMIT file transfer facility to talk, via a second SUPERVISOR serial port, to the debug monitors on the TASK processors. This can be useful for on-line debugging of TASK software.

° Menu_4 activates the sub-task gpc_init() which queries the user, via a set of menu options, for initialisation data for the CONTROL and ESTIMATOR tasks. This data is loaded into the relevant processors' mail-boxes before a further gpc_init() menu selection initiates run-time. During run-time gpc_init() feeds the user-selected set-point trajectory to the CONTROL task on every sample and continually monitors the terminal keyboard for a keystrike emergency abort. Post-run-time is entered via this keyboard abort.

° Menu_5 activates the sub-task graph_dump() which gives an immediate graphical display of the previous experiment.

° Menu_6 activates the sub-task file_log() which files data-logs from the CONTROL processor's high DPRAM to floppy disk.

° Menu_7 spawns to the operating system command shell for file-management or to run ancillary programs. Spawning has the advantage that all SUPERVISOR variables are preserved (Holt et al, 1978).

° Menu_8 terminates the SUPERVISOR task and returns to the operating system.

4.3.2.2 SUPERVISOR sub-task: down_load()

The executable code for the TASK processors is generated in raw binary format on the development system mentioned in the introduction to section 4.2 and copied onto floppy disk. The floppy is then taken to the

target system where the sub-task down_load(), activated as a menu option from gpc_super(), copies it byte-by-byte to the TASK processor's dual-port RAM until an EOF is reached.

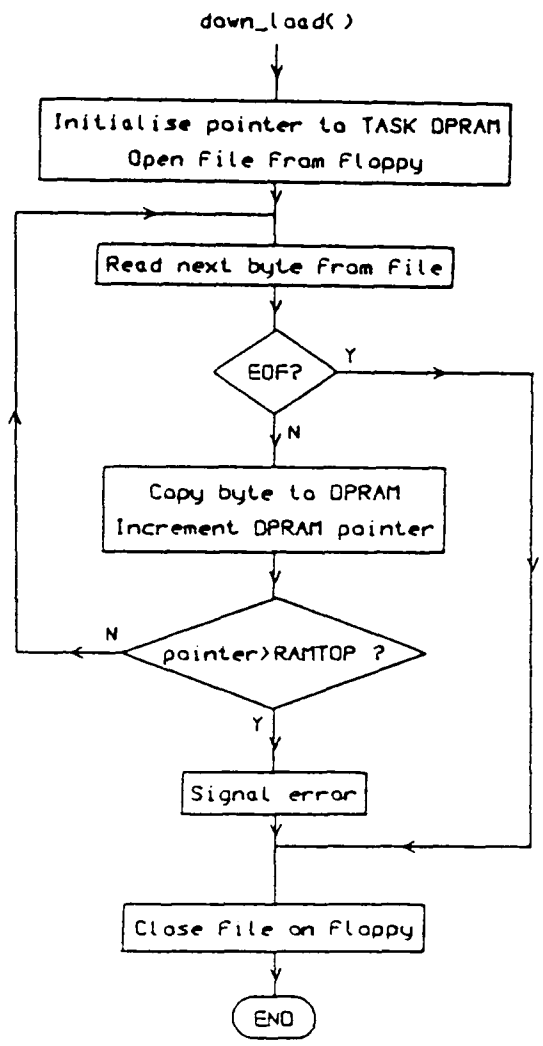


Fig.4.12: SUPERVISOR sub-task: down_load()

The principal parameters for this sub-task are the destination processor, CONTROL or ESTIMATOR, and the base address respective to the TASK processor to which the code should be loaded (normally 6000_H). The schematic of Fig.4.12 explains the basic operation of this sub-task.

4.3.2.3 SUPERVISOR sub-task: gpc_init()

The large sub-task gpc_init() is responsible for initialising the CONTROL and ESTIMATOR tasks and for run-time supervision. As the former involves querying the user for all the durations, tuning-knobs etc. it is

menu driven. All the values selected by menu responses are immediately copied to the relevant TASK mail-box where they will be read to internal TASK variables during the TASK initialisation phase which follows gpc_init() sending a [TASK_GO] signal.

° Menu_0 selects the CONTROL sample rate and the maximum duration of the experiment.

° Menu_1 selects values of the GPC tuning-knobs N_1, NY, NU, λ, $T(z^{-1})$ and $P(z^{-1})$ for both the CONTROL and ESTIMATOR tasks.

° Menu_2 selects initial values for the ESTIMATOR task; the adaption duration, initial covariances and an initial estimated model from file.

° Menu_3, given the initial estimates from menu_2, performs an initial controller design (top row of $(G_1^T G_1+\lambda I)^{-1} G_1^T$). This is principally for the non-adaptive case where the ESTIMATOR task is inactive. The resultant controller parameters are copied to the CONTROL task.

° Menu_4 reads in set-point trajectory from file, selects whether or not it is to be repeated (until a keyboard abort or a CONTROL task time-out) and chooses whether to use pre-specified or normal set-point sequences (see section 2.4.1).

° Menu_5, if the CONTROL and ESTIMATOR tasks have already been activated from the gpc_super() menu but are idling in their initialisation loop, sets the [task_state] flags in the CONTROL and ESTIMATOR mail-boxes to [TASK_GO]. Thereupon the TASKs immediately copy the contents of the mail-boxes to their internal variables and enter their run-time phases. If either the CONTROL or the ESTIMATOR task is not idling this is reported. Execution can however proceed if the ESTIMATOR task is inactive; the CONTROL task simply runs as a fixed-parameter controller designed on the basis of the information entered from menu_2. The run-time gpc_init() schematic Fig.4.13 illustrates how menu_5 proceeds to send the new set-point vector **w** every sample while monitoring for a keyboard abort. The keyboard abort sets the mail-box flags [task_state] to [TASK_END] which, for example, causes the

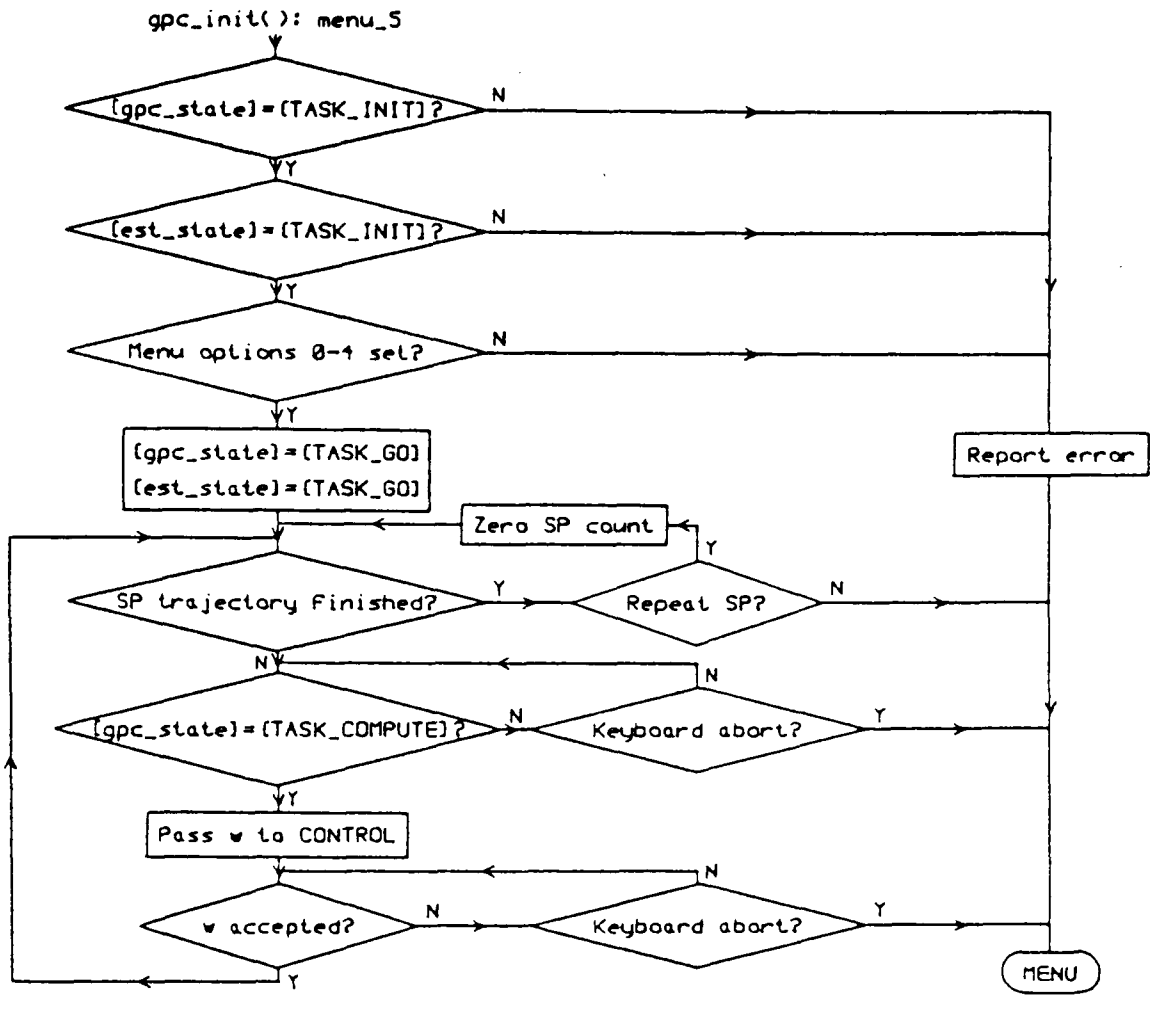


Fig.4.13: SUPERVISOR sub-task: gpc_init(), menu 5

CONTROL task to zero the DACs and return to its initialisation phase. If the CONTROL task terminates from a time-out or error condition the DACs will also be zeroed, the arm will stop moving and the keyboard abort must be used to call a further sub-task to report the cause of termination. Note that Menu_5 will not execute unless all the previous menu options have been visited.

- Menu_6 reads back the contents of the TASK mail-boxes to SUPERVISOR internal variables; both for a check and to examine the latest estimates in a self-tuning experiment.

- Menu_7 returns control to the main sub-task gpc_super().

4.3.2.4 SUPERVISOR sub-task: graph_dump()

The sub-task graph_dump() is activated by a gpc_super() menu option after the SUPERVISOR run-time phase has been terminated by the gpc_init() keystroke abort. Graph_dump() first uses the further sub-task read_log() to read back, from CONTROL's high DPRAM, the number of different variables and instants logged in the previous experiment (see Fig.4.15). The user is then queried from the terminal for the scales, offsets, number of instants to be plotted etc. before the particular logged variable (see data_logger() of section 4.3.3.3) is plotted.

This sub-task only provides a rough graphical output for on-the-spot inspection. More detailed graphics are obtained by transferring the data log to disk using the sub-task file_log() and then using the KERMIT file transfer facility to copy this file to a microVAX where the data is entered into the CTRLC control toolbox program.

4.3.2.5 SUPERVISOR sub-task: file_log()

The post-run-time sub-task file_log() is also activated by a gpc_super() menu option. Its purpose is to transfer all or part of the data log from the CONTROL DPRAM to permanent storage as a file on the floppy disk. Once the file is stored in a file structure accessible to the SUPERVISOR's OS9 operating system the KERMIT file transfer utility can be used to copy it to sophisticated analysis packages on a microVAX mini-computer. The main details of this sub-task are given in the schematic of Fig.4.14, the operation of the contributory sub-task read_log() being explained in the next section. File_log() also inserts a text banner into the start of the file detailing the controller configuration that produced the log.

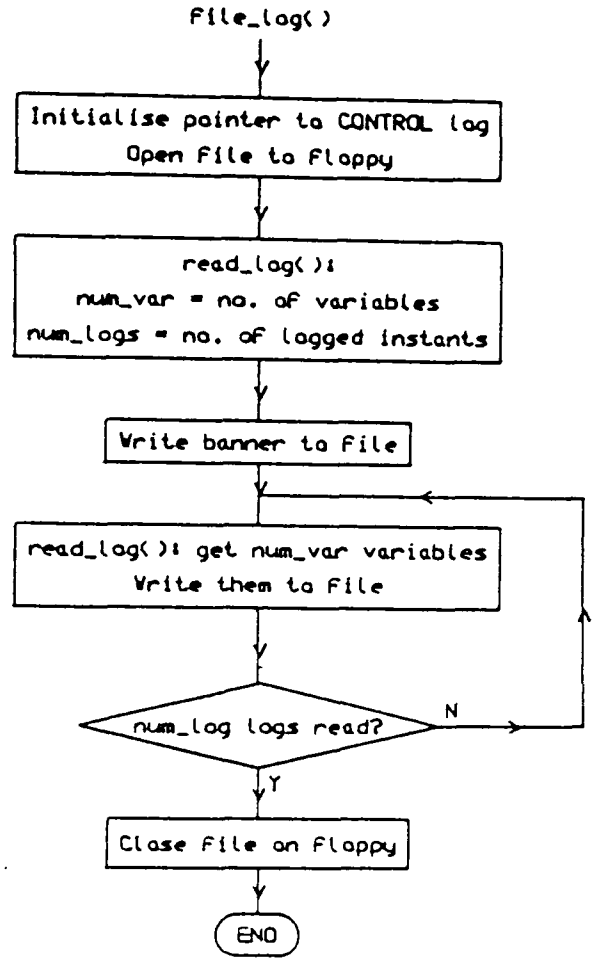


Fig.4.14: SUPERVISOR sub-task: file_log()

4.3.2.6 SUPERVISOR sub-task: read_log()

The sub-task read_log() is the complement of the CONTROL sub-task data_logger(), reading back the logged data from the CONTROL DPRAM. Like data_logger() it has two modes, initialisation and read-back. The SUPERVISOR sub-tasks graph_dump() and file_log() will, in preparing to recover a data log, first activate read_log() in its initialisation mode where it sets pointers to the data log in CONTROL DPRAM and returns the number of different variables logged together with the total number of logged instants. Subsequently read_log() is repeatedly activated in read-back mode, each time returning one sample's set of logged variables.

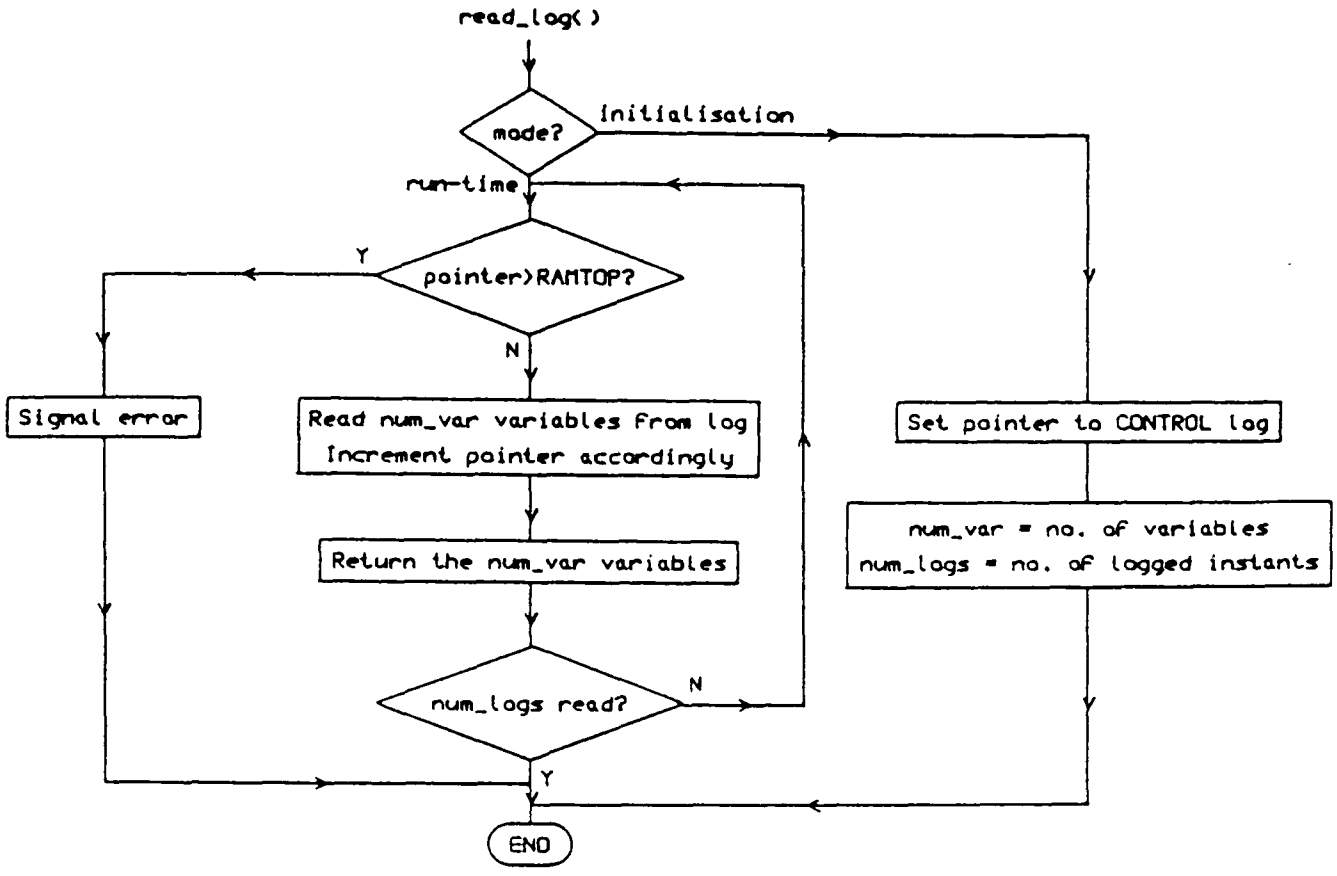


Fig.4.15: SUPERVISOR sub-task: read_log()

4.3.3 The CONTROL Task

The CONTROL task is responsible for servicing the analogue i/o board, performing the GPC control calculation (4-1), logging data and providing the sample-by-sample synchronisation for the run-time SUPERVISOR and ESTIMATOR tasks. It is itself composed of the sub-tasks shown schematically below

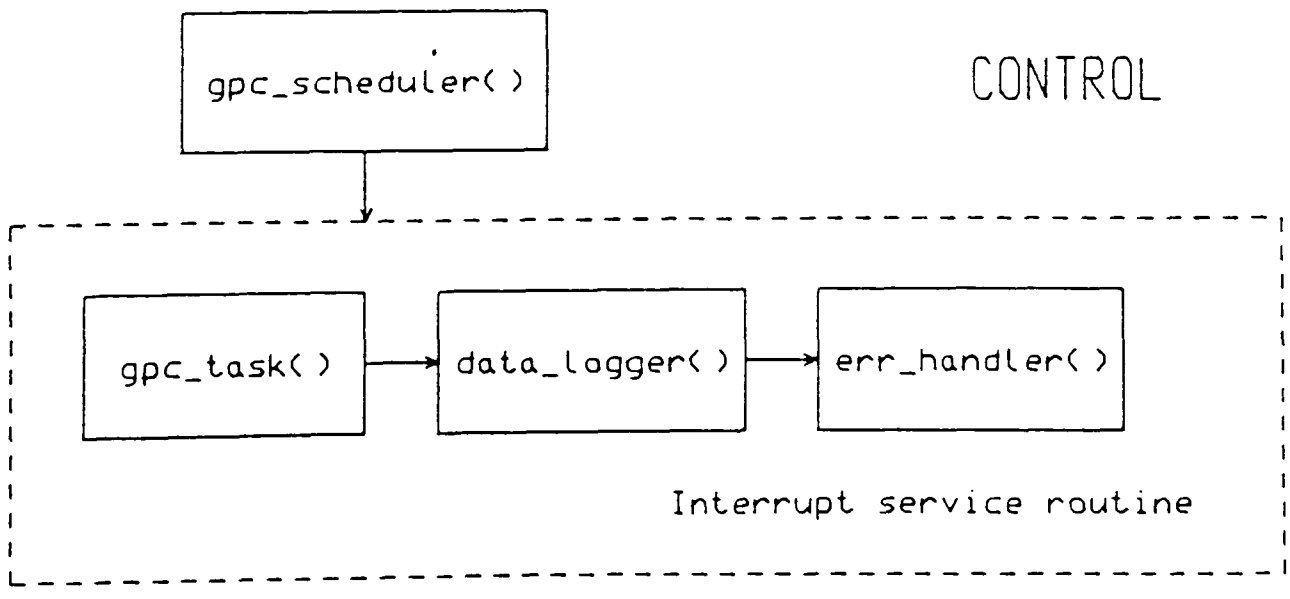


Fig.4.16: CONTROL sub-tasks

These sub-tasks are described in the following sections. For each sub-task a schematic is given outlining the flow of execution together with a brief description of the sub-task's purpose. References will be made throughout these descriptions to the flag [task_state]. This flag is resident in CONTROL's mail-box and may be accessed by any of the tasks or sub-tasks. It is the principal mechanism for synchronisation between the concurrently executing TASKS.

4.3.3.1 CONTROL sub-task: gpc_scheduler()

As shown in Fig.4.16 this sub-task coordinates the other sub-tasks gpc_task(), data_logger() and err_handler(). At the time of initial execution (from a SUPERVISOR sub-task gpc_super() menu option) it zeros the DACs, sets the flag [task_state] to [TASK_INIT] and then sits in a tight loop waiting for [task_state] to become [TASK_GO]. The SUPERVISOR sub-task gpc_init() meanwhile loads all the variables required by gpc_scheduler() and its sub-tasks (eg. sample rates, gains etc.) into the mail-box and, when ready, sets [task_state] to [TASK_GO].

Upon receipt of [TASK_GO] gpc_scheduler() initialises the sub-task err_handler(), loads the on-board MC68901 counter/timer registers with the appropriate values to generate interrupts at the sample frequency, initialises the sub-task gpc_task() by copying the required initial values from the mail-box to its internal variables and initialises the sub-task data_logger() to point at unused RAM. When this initialisation phase is complete run-time proper begins with the enabling of the DACs, the encoders and finally the interrupts. Then gpc_scheduler() simply loops monitoring a global error flag [err_code]. The interrupt service routine then sequentially executes gpc_task(), data_logger() and err_handler() in the background at the required sample rate. If the CONTROL task has been active for a period

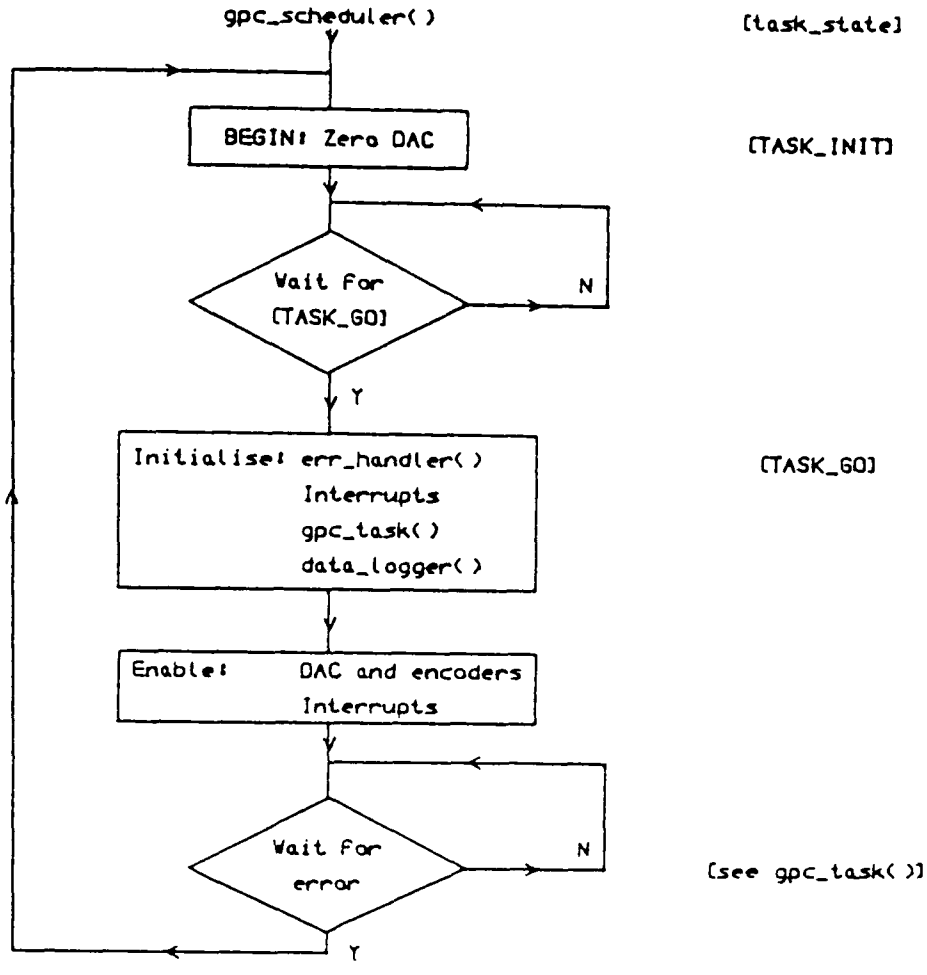


Fig.4.17: CONTROL sub-task: gpc_scheduler()

exceeding that specified in the initialisation phase (see gpc_init() of section 4.3.2.3) or the SUPERVISOR sub-task gpc_init() sets [task_state] to [TASK_END] upon receipt of a keystrike abort or any sub-task detects an error this is flagged via [err_code] causing the interrupts to be disabled and gpc_scheduler() to return to its beginning.

4.3.3.2 CONTROL sub-task: gpc_task()

The sub-task gpc_task() has two modes, initialisation and run-time. Once the SUPERVISOR sub-task gpc_init() has copied all the initialisation data to the CONTROL mail-box and flagged [TASK_GO] it is activated by gpc_scheduler() in its initialisation mode. Thereupon it copies the initial values from the mail-box to its internal variables.

During run-time the first action of the interrupt service routine, periodically activated at the chosen sample rate, is to execute gpc_task() in its run-time mode. Gpc_task() then sets the flag [task_state] to [TASK_INPUT] and reads the encoders to infer absolute tip position. This new measurement, together with earlier filtered measurements and control inputs, are then echoed to the ESTIMATOR mail-box (see rls_input() of section 4.3.4.2). When this input phase is complete gpc_task() sets [task_state] to [TASK_COMPUTE] and, given the latest measurement, performs the GPC control calculation (section 4.3.1.1) to obtain the new control u(t). Note that this calculation is the major contributor to the duration of the interrupt service routine and varies in execution time with the parameters of GPC ie. the horizons N_1 and NY, the order of the design polynomials $P(z^{-1})$, $Q(z^{-1})$ and $T(z^{-1})$ and the order of the assumed model of the process $A(z^{-1})$ and $B(z^{-1})$. When u(t) is available gpc_task() sets [task_state] to [TASK_OUTPUT] and performs any output conditioning such as desaturation or switched mode actuation (Tuffs, 1984) before passing the control to the DAC.

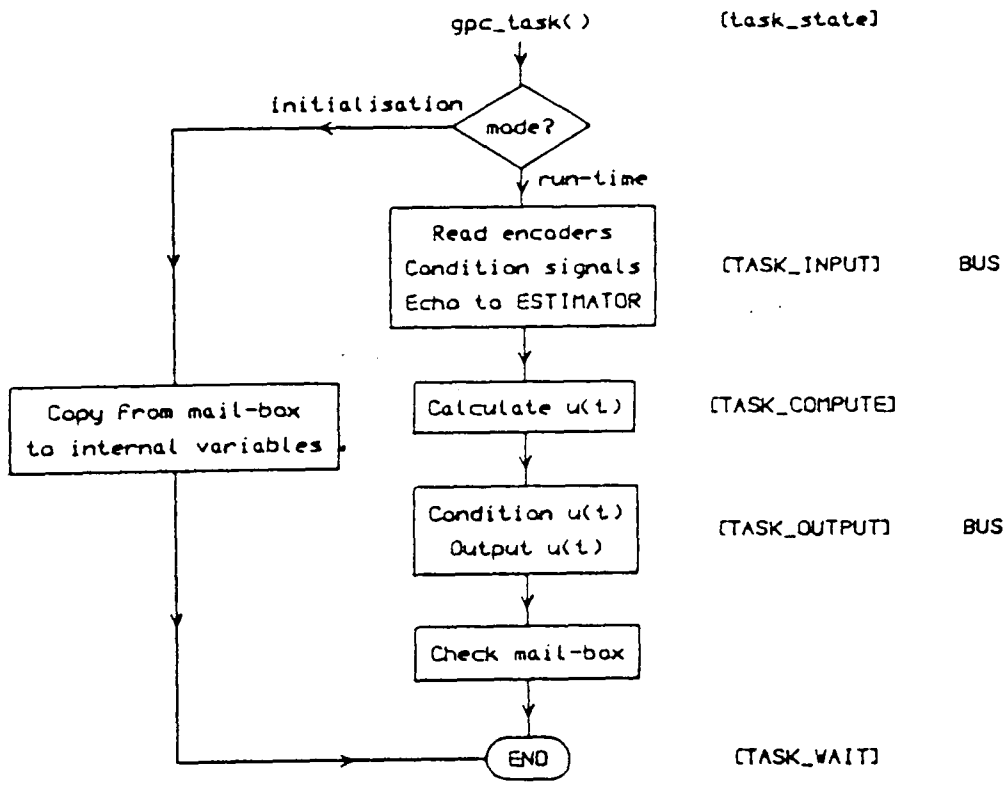


Fig.4.18: CONTROL sub-task: gpc_task()

Finally gpc_task() checks the mail-box for new data, the arrival of which is signalled by the state of flags within the mail-box (see section 4.3.1.3). Incoming data may be divided into two categories: that which must arrive every sample and that which may arrive at any sample. In the present implementation the set-point vector **w**(t) is the only example of the former. In the event of the relevant flag signalling the absence of a new set-point an error condition is flagged and run-time terminates via the sub-task err_handler(). Otherwise the new set-point vector is copied from the mail-box to an internal array. Examples of the second category are when any of the controller parameters are changed, for instance the estimated model. Then the new estimated model polynomials $\hat{B}(z^{-1})$ and $\hat{A}(z^{-1})$ and the new top row of the matrix $(\mathbf{G}_1^T\mathbf{G}_1+\lambda I)^{-1}\mathbf{G}_1^T$ must be copied from the mail-box to internal variables. After checking the mail-box gpc_task() sets [task_state] to [TASK_WAIT] and terminates.

4.3.3.3 CONTROL sub-task: data_logger()

The sub-task data_logger() also has two modes, initialisation and run-time. In its initialisation mode it sets up a descriptor immediately below the RAM logging area, writing to it the number of different variables logged and a zeroed counter for the number of logged instants. The run-time data_logger() mode is activated as part of the interrupt service routine immediately after gpc_task() terminates. Every sample it logs a set of global variables such as the measured variable y(t) and the control effort u(t) to the area in the CONTROL task's on-board RAM which lies above the CONTROL program (Fig.4.6), incrementing the log counter. Then even if the CONTROL task terminates prematurely the log itself contains all the information required for the SUPERVISOR sub-task read_log() to recover all the logged data up to that time. If the log becomes so large with time that it threatens

to overflow the available RAM an error is flagged for the sub-task err_handler() described in the following section.

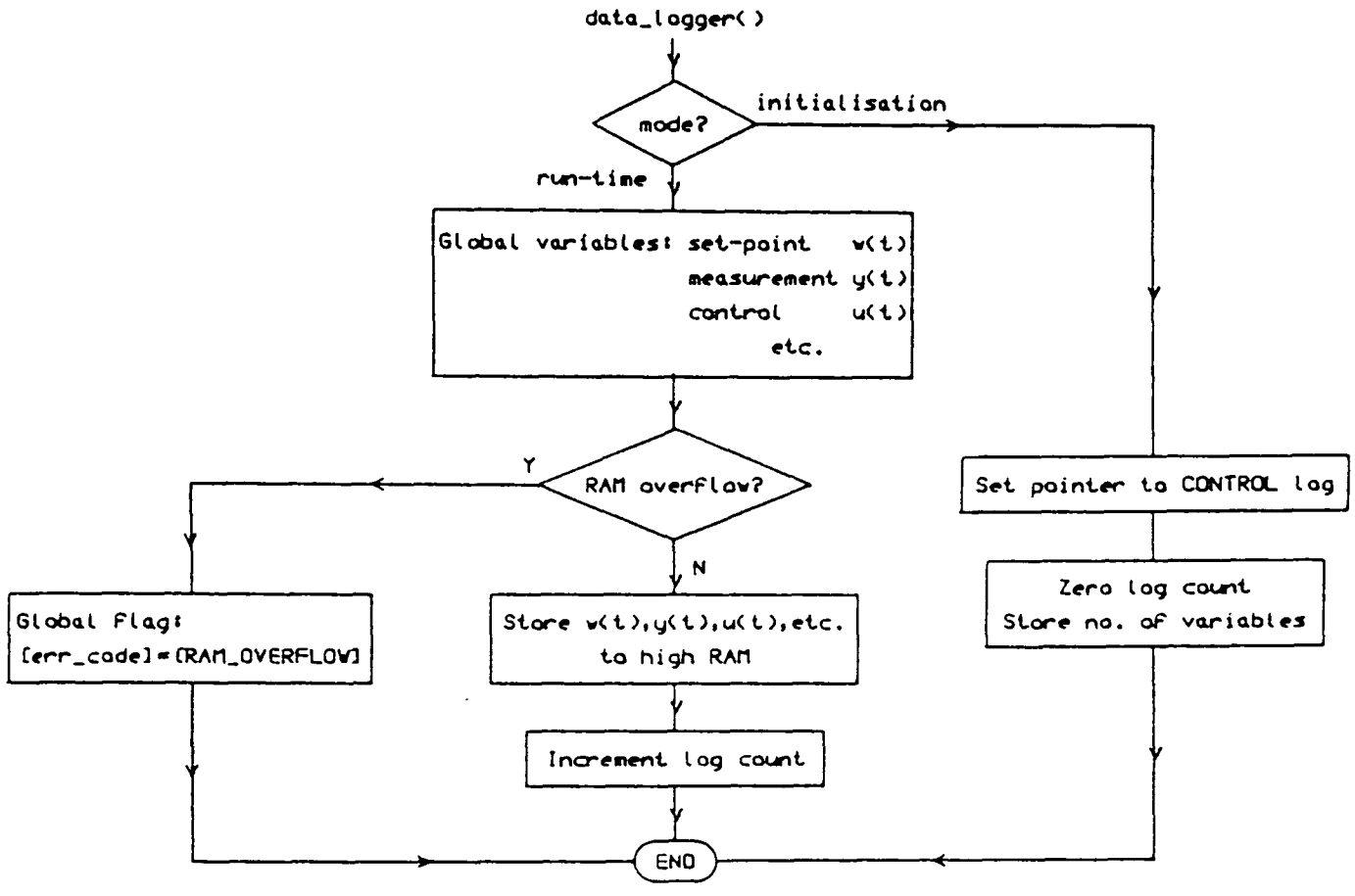


Fig.4.19: CONTROL sub-task: data_logger()

4.3.3.4 CONTROL sub-task: err_handler()

The sub-task err_handler() is the final sub-task to be activated as part of the interrupt service routine. If an error occurred in any of the previous sub-tasks gpc_scheduler(), gpc_task() or data_logger() a global variable [err_code] is set to a value indicating the source and nature of the error. In the absence of error the variable [err_code] stays at its initial value [NO_ERROR]. Err_handler() checks this variable and, in the event of an error ie. [err_code] not equal to [NO_ERROR], it disables the interrupts, sets the mail-box flag [task_state] to [TASK_ERR] and copies the value of [err_code] to another mail-box flag [task_err_code]. In this way the CONTROL task returns to the initialisation phase of gpc_scheduler() and the

SUPERVISOR task, upon receiving a keyboard abort, can note that an error has occurred and, via [task_err_code], report the nature of the error (see gpc_init() of section 4.3.2.3).

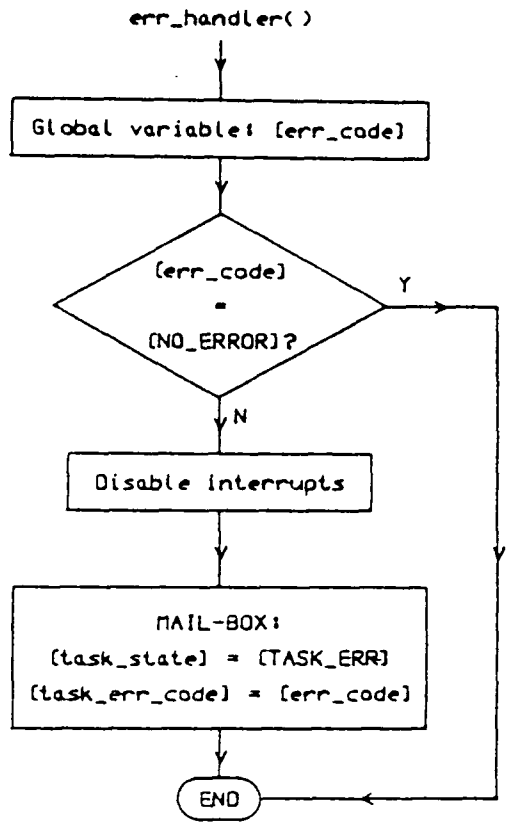


Fig.4.20: CONTROL sub-task: err_handler()

The last action of the interrupt service routine is to check an on-board counter/timer register for the possibility of an already pending interrupt. If this is present an error condition is flagged for err_handler() on the next sample and the sample rate or controller complexity must then be decreased to ensure that the interrupt service routine can complete before the next interrupt occurs.

4.3.4 The ESTIMATOR Task

The ESTIMATOR task, by monitoring the torques applied to the arm and the resulting deflections, recursively updates the estimated model of the plant and re-designs the controller implemented by the CONTROL task. When the

re-designed controller parameters are regularly passed to the CONTROL processor the overall system becomes a self-tuning controller (see section 4.3.1). The ESTIMATOR task is itself composed of sub-tasks shown schematically below

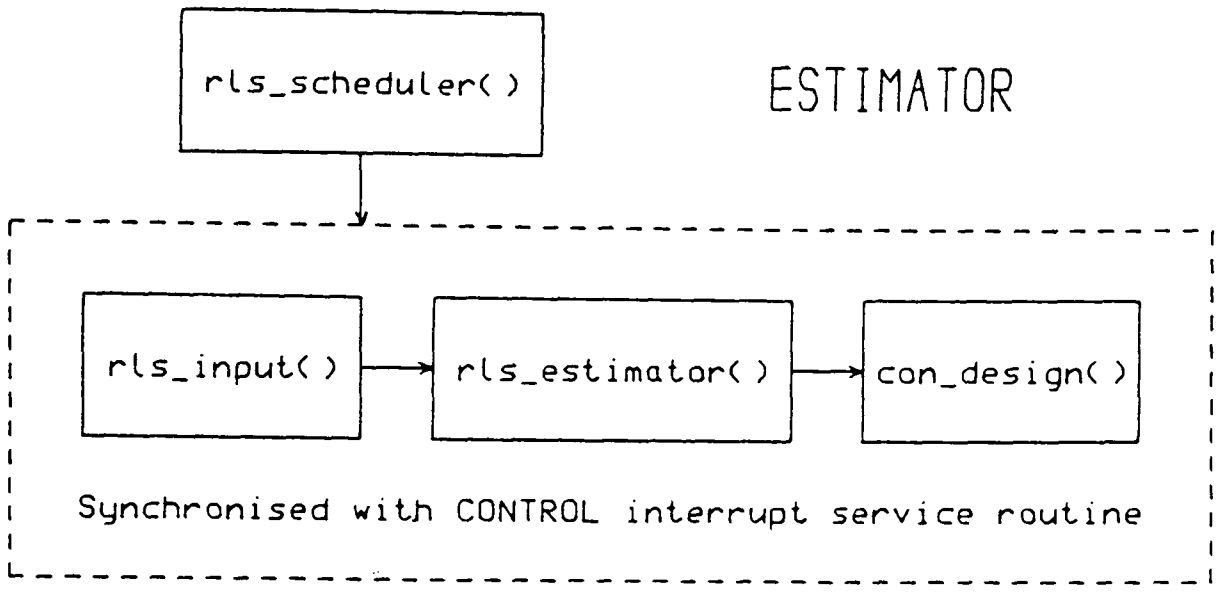


Fig.4.21: ESTIMATOR sub-tasks

These sub-tasks are described in the following sections. For each sub-task a schematic is given outlining the flow of execution together with a brief description of the sub-task's purpose.

4.3.4.1 ESTIMATOR sub-task: rls_scheduler()

As shown in Fig.4.21 this sub-task co-ordinates the other sub-tasks rls_input(), rls_estimator() and con_design(). At the time of initial execution it initialises the ESTIMATOR mail-box, setting the flag [task_state] to [TASK_INIT], and then sits in a tight loop waiting for [task_state] to become [TASK_GO]. The SUPERVISOR sub-task gpc_init() meanwhile loads all the variables required by rls_scheduler() and its sub-tasks (eg. adaption duration, initial covariances, estimated model order etc.) into the mail-box and, when ready, sets [task_state] to [TASK_GO].

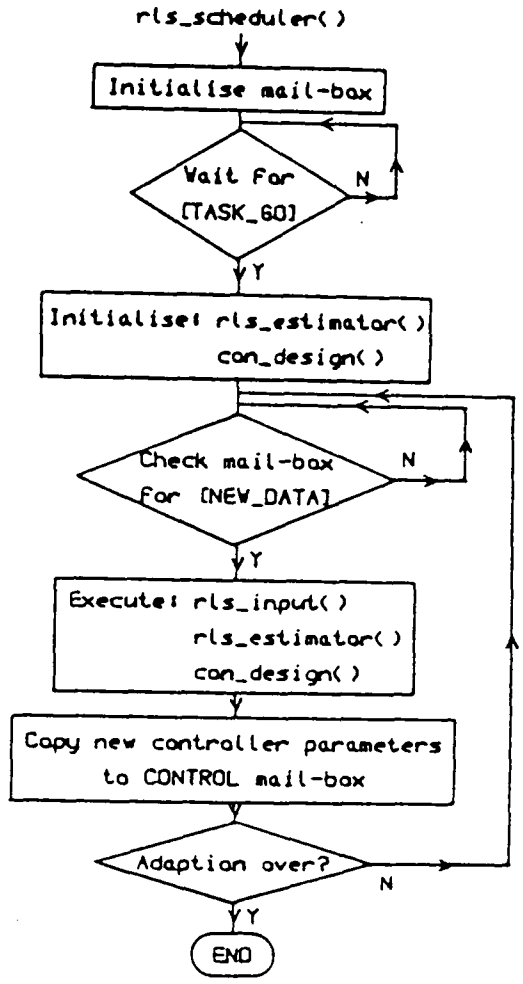


Fig.4.22: ESTIMATOR sub-task: rls_scheduler()

Upon receipt of [TASK_GO] rls_scheduler() initialises the sub-tasks rls_estimator() and con_design() which copy these initial values from the mail-box to their internal variables. When this initialisation phase is complete run-time proper begins with rls_scheduler() entering a loop synchronised, via mail-box flags, to the CONTROL task sample-rate. In the body of this loop rls_scheduler() first waits for the arrival of new data to be signalled by the mail-box flag [task_comm]. This data consists of the latest measurement and the filtered (by $T_c(z^{-1})$) components of the regressor vector $\phi(t)$ required for the least-squares update (1-10) to (1-12). When the new data has arrived rls_scheduler() activates the sub-task rls_input() which copies this data to internal variables and conditions it prior to estimation.

Then the sub-task rls_estimator(), operating upon the new data set, updates the least-squares estimates of the $A(z^{-1})$ and $B(z^{-1})$ polynomials of (1-7). These estimates are then directly fed to the sub-task con_design() where the top row of the matrix $(G_1^T G_1 + \lambda I)^{-1} G_1^T$ is calculated. The new

estimates $\hat{A}$ and $\hat{B}$, together with the new top row of the matrix $(G_1^T G_1 + \lambda I)^{-1} G_1^T$, are then written to the CONTROL processor's mail-box and, unless the adaption period chosen by the SUPERVISOR sub-task gpc_init() has been exceeded, the loop is reexecuted.

4.3.4.2 ESTIMATOR sub-task: rls_input()

The recursive least squares estimation algorithm of section 1.1.3 requires, for each update, a data parcel comprising a time-contiguous set of measurements and applied control actions. Given that the estimation and controller redesign calculation is computationally slightly more taxing than the control calculation there will exist a sample rate where the ESTIMATOR task cannot complete its update in a single CONTROL sample. To ensure that the ESTIMATOR task operates on time-contiguous data the entire data parcel is passed to the ESTIMATOR mail-box by the control sub-task gpc_task() at every sample and the ESTIMATOR sub-task rls_input() reads the mail-box whenever it is ready.

The regressor vector for the RLS update is available as a combination of the state vectors $\{y'(t-1), y'(t-2), \ldots, y'(t-na)\}$ and $\{u'(t-1), \ldots, u'(t-nb-1)\}$ used in the GPC control calculation (see chapter 2). The prime denotes filtering by the $T(z^{-1})$ polynomial. The sub-task rls_input() copies the regressor vector $\phi(t)=\{y'(t-1), .., y'(t-na), u'(t-1), .., u'(t-nb-1)\}$ to an internal array and forms the latest filtered measurement from

$$y'(t) = \frac{1}{t_0}[y(t) - t_1 y'(t-1) - \ldots - t_{nt} y'(t-nt)] \qquad (4\text{-}2)$$

Note that this arrangement means that the estimator polynomial $T_e(z^{-1})$ must equal the controller polynomial $T_c(z^{-1})$, this restriction being considered

acceptable for the reasons outlined in section 2.6.3. Note also that the read from the ESTIMATOR mail-box obeys the protocol described in section 4.3.1.3, allowing the ESTIMATOR and CONTROL tasks to proceed asynchronously.

Rls_input() also reads potentially new values for the controller horizons N_1, NY and NU and the control weighting λ from the ESTIMATOR mail-box. This allows the controller configuration to be changed on-line from the SUPERVISOR task if required.

4.3.4.3 ESTIMATOR sub-task: rls_estimator()

The sub-task rls_estimator() has two modes, initialisation and run-time. Before run-time the supervisory sub-task rls_scheduler() activates rls_estimator() in its initialisation mode where initial values for the recursive least squares algorithm (ie. model order, initial covariances etc.)

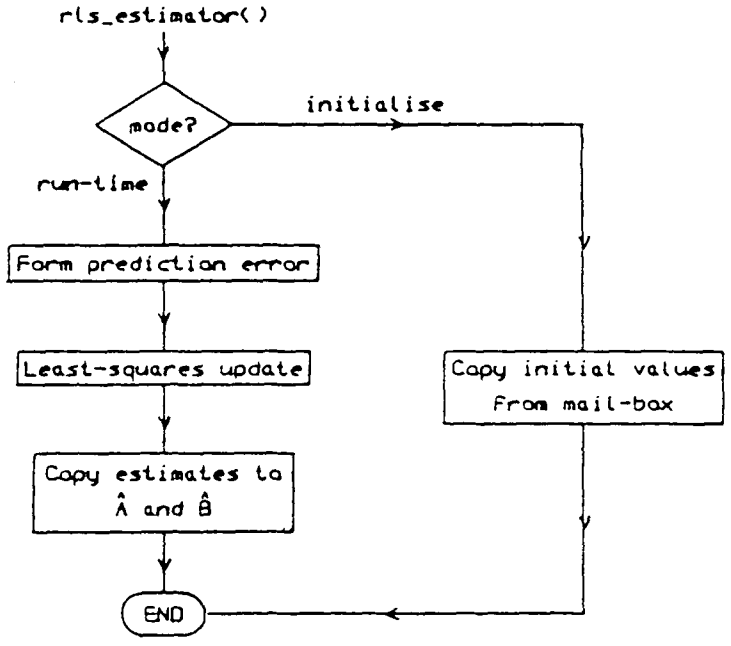


Fig.4.23: ESTIMATOR sub-task: rls_estimator()

are copied from the ESTIMATOR mail-box to internal variables. Subsequently rls_scheduler() activates rls_estimator() in its run-time mode where, given the filtered regressor vector and measurements from rls_input(),

rls_estimator() calculates the prediction error and performs the least-squares measurement update using the U-D factorisation of Bierman (1977). The new parameter vector $\hat{\theta}(t)$ is then copied to the $\hat{A}$ and $\hat{B}$ polynomials and rls_estimator() terminates.

4.3.4.4 ESTIMATOR sub-task: con_design()

The final sub-task to be activated from the main loop of rls_scheduler() is the controller redesign sub-task con_design(). This sub-task again has two modes, initialisation and run-time. Before run-time rls_scheduler() activates con_design() in its initialisation mode where initial values for the GPC controller design (ie. prediction horizons, tuning-knobs etc.) are copied from the ESTIMATOR mail-box to internal variables. Subsequently rls_scheduler() activates con_design() in the run-time mode which iterates the newly estimated model $\hat{A}$ and $\hat{B}$ from rls_estimator() for the pulse or step response (section 2.5.3) and then in the latter case uses the Bierman U-D measurement update to form the top row of the matrix $(G_1^T G_1+\lambda I)^{-1} G_1^T$ as described in Mohtadi (1986). This calculation comprises the majority of the ESTIMATOR task duration.

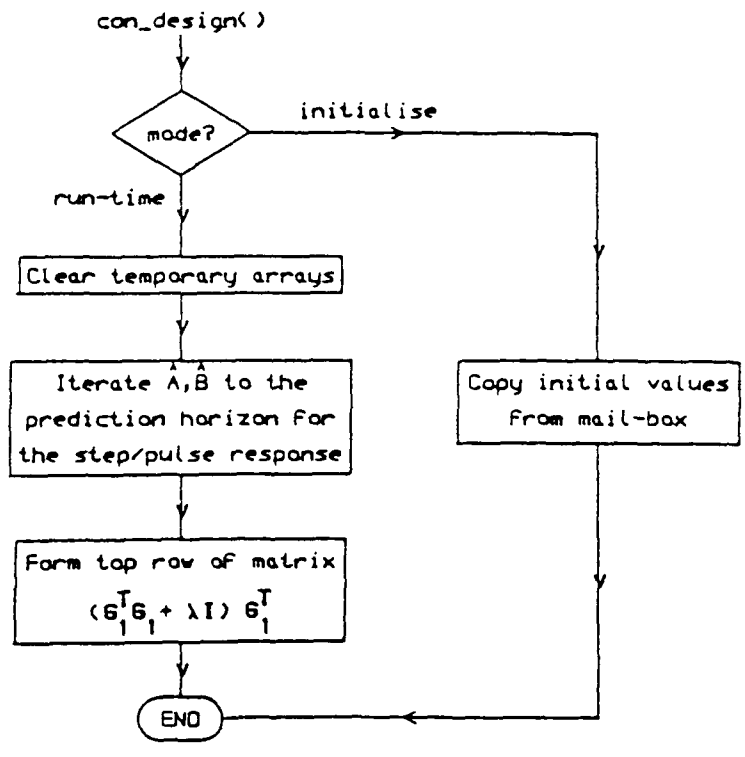


Fig.4.24: ESTIMATOR sub-task: con_design()

Chapter 5

EXPERIMENTAL WORK

This chapter presents experimental results obtained from the application of GPC to the compliant arm described in section 4.1 and modelled in section 3.1. The principal experimental objectives are itemised below:

- To illustrate the possibilities for closed-loop tailoring arising from the choice of the GPC tuning-knobs N_1, NU, NY and design-filters $P(z^{-1})$, $\lambda Q(z^{-1})$, $T(z^{-1})$ in the context of the control of a real system.
- To investigate the problems of plant-model-mismatch arising from low-order linear modelling of a high-order non-linear plant and, in particular, to demonstrate how the design filter $T(z^{-1})$ can be used to overcome them. While $T(z^{-1})$ also has a load-disturbance tailoring interpretation this may be treated separately from the PMM since, unless deliberately introduced, there are no external disturbances and the only sensor noise present is due to small quantisation errors from the optical encoders (section 4.1).
- To illustrate how adaptive control can cope with the inevitable time-varying dynamics of a working compliant robot. While the dynamics of the arm during normal running are practically time-invariant, provision has been made to vary the end-mass allowing significant perturbations to be made to the link dynamics.
- To evaluate the usefulness of pre-specified set-points over normal set-points on a real plant. Knowledge of future set-points is assumed in the context of robotics. The ease with which this information may be incorporated into the GPC control calculation is expected to be one of GPC's main advantages over alternative strategies.

° To investigate the specific applicability of the GPC control strategy to a high-order multi-modal and NMP* flexible system.

° To illustrate configuration requirements for a system with strict amplitude limits on the available actuation.

Collectively these results address the problem of configuring the GPC self-tuner, previously used for the control of slow and noisy process plant, for a high-performance application where accurate noise-free measurements are available and actuation is assumed 'cheap', although limited in amplitude. Section 5.1 describes the initial commissioning phase and the structure of the subsequent experiments. For accurate positioning of the link tip section 5.2 shows that incremental GPC must be used and illustrates the resultant increased dependency on $T(z^{-1})$ for stability. Section 5.3 systematically evaluates the effect of the tuning-knobs, design filters and the use of pre-specified set-points. Whereas these sections deal with the non-adaptive GPC control strategy section 5.4 presents results obtained from self-tuning GPC. Conclusions and observations are given in section 5.5 together with a strategy for configuring GPC in practice.

5.1 Experimental Details

Before proceeding to the experimental results it is useful to consider the commissioning phase; initial experimentation with a PID controller and the subsequent choice of sample rate and estimator model order for GPC. The typical structure and layout of the experiments are described together with the units used in the figures and text.

* Non-minimum phase.

5.1.1 Initial commissioning

A simple PID controller was used initially to facilitate development and testing of the multi-processor multi-tasking hardware and software described in chapter 4 and also to gain an idea of the attainable sample rate of the far more complex GPC algorithm. The PID also provided ratification that a higher-order controller (than PID) would be needed for high-performance control of this complex system.

When the GPC controller was successfully imbedded in the VME target system the initial estimates of the GPC plant model (5-1) were obtained off-line from i/o data generated by the PID running at the appropriate sample rate. The PID was implemented with integral desaturation and with the derivative action on the output to avoid 'derivative kick' as detailed in Clarke, (1983).

Fig.5.1 shows a typical series of set-point responses obtained using this PID controller*. Performance can perhaps be improved by further tuning of the controller gains. However the phase-lag introduced by increasing the gain of the integral term, to eliminate the steady-state offsets, is profoundly destabilising and increasing the proportional gain causes large overshoots and an oscillatory closed-loop. Increasing the derivative gain is also destabilising which Fraser (1987) shows, by a root-locus method, to be due to the complex nature of the plant. While acknowledging that this PID is not perfectly tuned it is still worth comparing it with the far superior set-point following of incremental GPC (eg. Fig.5.19).

5.1.2 Choice of Model-order

Following standard practice (see section 1.1.2) the initial GPC controllers running on the compliant arm were commissioned upon the basis of

* The gains are not given due to meaningless units.

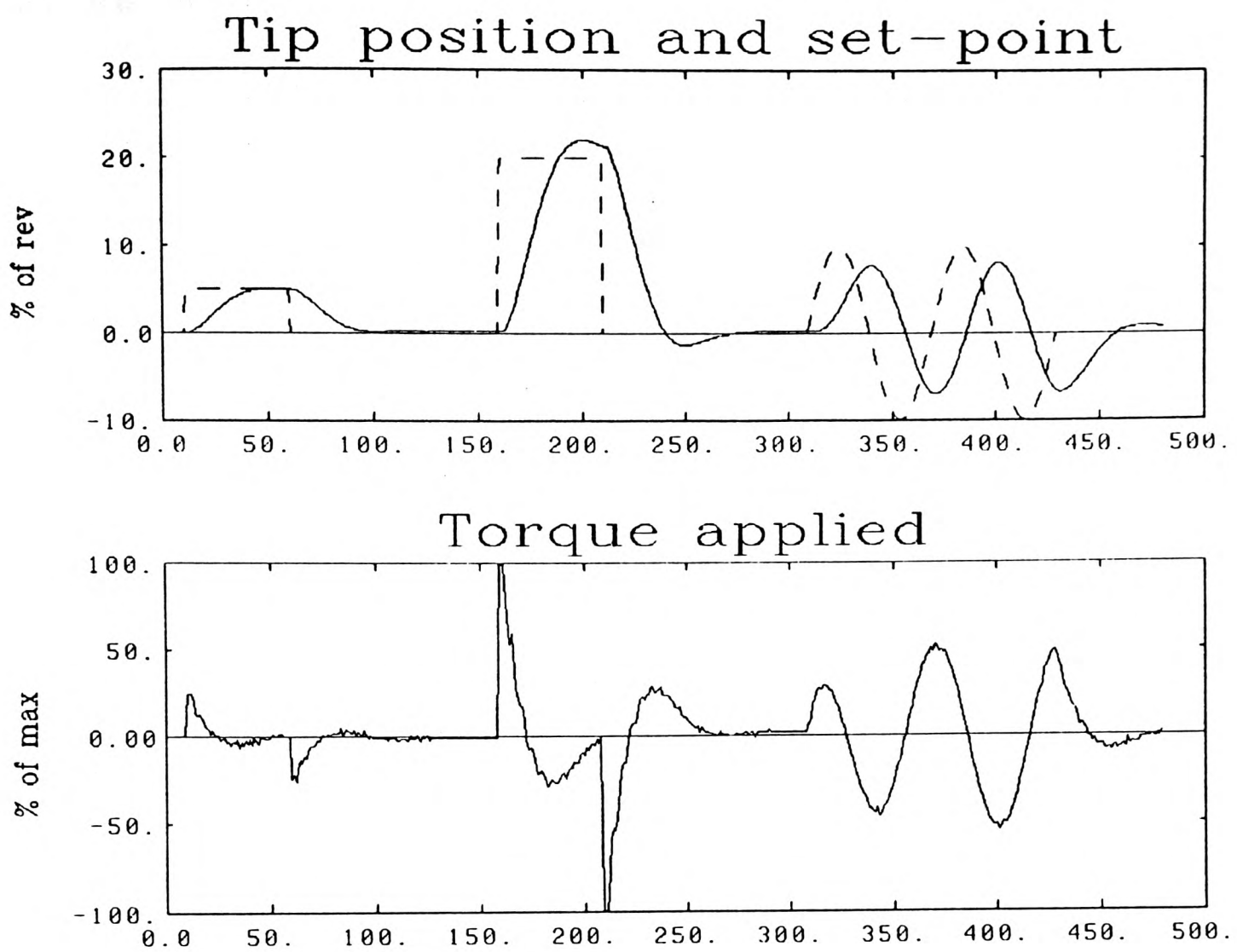


Fig.5.1: Typical PID set-point responses

an estimated model whose order was chosen to match the apparent dominant dynamics of the plant. Given sampling considerations described in the following section the dominant dynamics were taken to be a pair of integrators (rigid link behaviour) plus one lightly damped resonant mode (compliant behaviour) ie. a fourth-order transfer-function. From inspection of the PID runs the integer plant delay was assumed to be effectively zero (not including the unit ZOH* delay) and therefore, unless stated otherwise, the models estimated for the GPC runs of this chapter were of the form

$$y(t) = \frac{\hat{b}_0 + \hat{b}_1 z^{-1} + \hat{b}_2 z^{-2} + \hat{b}_3 z^{-3} + \hat{b}_4 z^{-4}}{1 + \hat{a}_1 z^{-1} + \hat{a}_2 z^{-2} + \hat{a}_3 z^{-3} + \hat{a}_4 z^{-4}} u(t-1) \qquad (5\text{-}1)$$

* Zero Order Hold.

There are 5 $\hat{b}_i$ coefficients to cater for the certain presence of a fractional time-delay from the control calculation. The parameter estimates ($\hat{a}_j$, $\hat{b}_i$) were, unless stated to the contrary, usually initialised to zero and estimated off-line from i/o data generated from a previous PID or GPC run using the basic least-squares recursion (1-10) to (1-12) with P(0)=1000I and no assumed prior knowledge of structure (chapter 3).

This of course neglects the contribution to the linear plant dynamics from higher frequency resonant modes, at 50Hz and beyond, and to the non-linear dynamics from friction, saturation etc. It should also be noted that no anti-aliasing filtering was performed on the measured signal due to the complication of digital encoder signals. While the encoders could be sampled at a rate well beyond the main controller sample rate and filtered in discrete-time this was considered too computationally taxing. Consequently aliased high-frequency measurements must be treated as a possible additional source of load disturbance.

5.1.3 Choice of Sample-rate

Both theory (Fraser, 1987) and experiment, using a signal generator, place the first two resonant modes at approximately 15Hz and 50Hz respectively with the default end-mass in place. There was considered to be upper and lower limits on the choice of the sample rate for the GPC controller. The lower limit arose from modelling considerations, it being intended to model the link as a linear fourth-order transfer-function ie. a double integrator plus lightly-damped resonant mode. In order for the discrete-time system to 'see' the first mode it must, from Nyquist, sample at at least 30Hz. The upper limit arose from practical hardware and software considerations, since it was not considered possible to have a full implementation of GPC running at more than 100Hz. Any simplifications in the

structure of GPC (eg. Farsi & Karam, 1987) were avoided since they would restrict the generality of the results.

From initial PID results the minimum plausible settling time, for an average set-point change, was estimated to be approximately 0.25 seconds. Employing the standard rule-of-thumb mentioned in section 2.6.3 raises the minimum sampling rate to beyond 40Hz. The upper limit was also reduced by the consideration that later modifications to the algorithm, such as the incorporation of $P(z^{-1})$ and $T(z^{-1})$, would substantially reduce the maximum sample rate. With these bounds in mind a compromise sample rate of 60Hz was chosen allowing a maximum prediction horizon of 15 samples (0.25 seconds).

5.1.4 Experiment structure

The majority of experiments described in the following sections involve following a repetitive set-point trajectory while comparing the effect on the closed-loop stability, speed and overshoot etc. of varying the GPC tuning-knobs and design-filters. Each experiment attempts to bear in mind the eventual aim of stable, rapid and accurate control of tip position. Changes in the GPC control law are usually made by bringing the arm to rest, transferring the modified parameters from the SUPERVISOR task to the CONTROL task (see chapter 4), and then proceeding. As all the changes occur in the steady-state the runs may be displayed as being continuous.

Since the arm is somewhat under-torqued and therefore prone to actuator saturation, where the object of the experiment is to compare the effects of various values of a tuning-knob or filter, the linear region resulting from small set-point changes is generally used. However several runs illustrate the effects upon the closed-loop of saturation.

All runs in section 5.2 and 5.3 are for 'fixed-parameter' GPC ie. a GPC controller designed on the basis of a fixed model estimated previously

from running i/o data generated either by a PID or another GPC controller. The experiments of section 5.4 are for the self-tuning version of GPC where the model upon which GPC is based is re-estimated at every sample by the ESTIMATOR task and the controller itself re-designed. The reason for this separation of sections is that it is first intended to investigate the suitability of the GPC control strategy for the compliant arm and then to investigate whether it may successfully be made self-tuning.

5.1.5 Units

° Set-points and measured tip positions are given in percent of a revolution ie. +100% corresponds to one revolution in the positive direction.

° Control actions are given in percent of full-range torque ie. +100% control corresponds to maximum torque in the positive direction while -100% control is maximum torque in the opposing direction.

5.2 Positional vs. Incremental GPC

As explained in section 2.5.1 incremental algorithms (Tuffs & Clarke, 1985) must be used to overcome the offset problems encountered with precursing positional algorithms. Consequently the majority of the results presented in this chapter are for the incremental version of GPC. For completeness this section verifies that under experimental conditions positional GPC suffers as predicted from steady-state offsets and that the incremental formulation successfully eliminates them.

Fig.5.2 shows a comparison of the positional and incremental algorithms where attention is to be paid to steady-state offsets in both the set-point response and also in the response to external load-disturbances.

Initially the CONTROL task (section 4.3.2) is configured as fixed-parameter positional GPC with N_1=NU=1, NY=10, P=1, λ=1e-8 and $T_c=(1-0.8z^{-1})$ based on a positional model estimated with the filter $T_e=T_c$ (section 2.6.3). The resulting set-point response is fast but inaccurate with a large (4%) offset in the steady-state. The control can settle to a steady finite value, despite the two integrators of the plant, due to the large amount of motor friction present (approximately 4% of the actuator range). In the steady-state following the downward-going set-point change a load-disturbance is applied by manually deflecting the arm, first one way and then the other. The positional controller again fails to eliminate the offset.

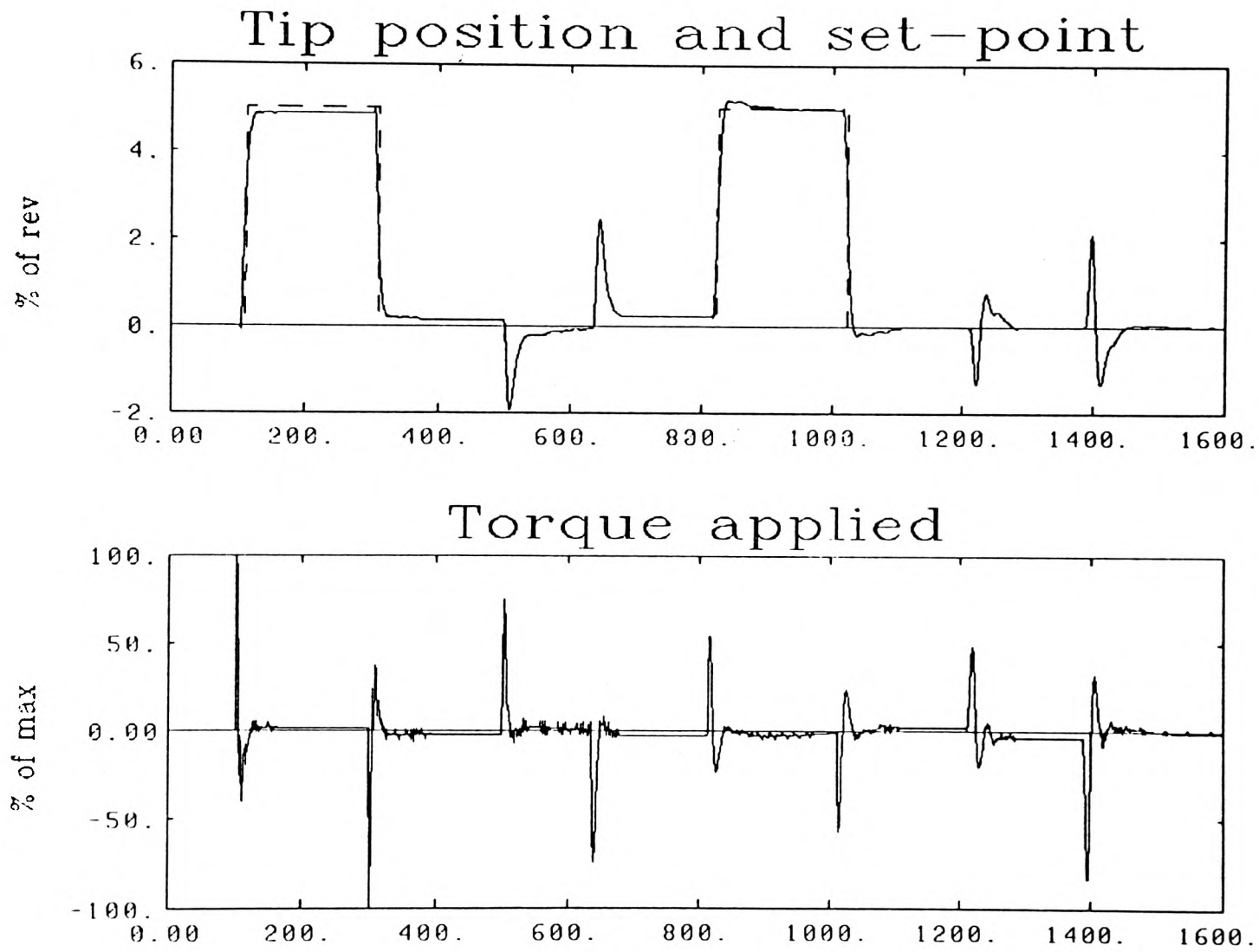


Fig.5.2: Positional vs. Incremental GPC

Before the next upward-going set-point change the CONTROL task is reconfigured to fixed-parameter incremental GPC with N_1=NU=1, NY=10, P=1, λ=1e-8 and $T_c=(1-0.8z^{-1})^2$ based on an incremental model estimated with the

filter $T_e(z^{-1})=T_c(z^{-1})$. The order of $T(z^{-1})$ was increased to two for the reasons given in section 2.6.3. The steady-state offset is seen to be eliminated both in the set-point response, although there is a quite large overshoot, and in the load disturbance rejection by the incremental version of GPC. Again small finite steady-state control actions do not affect the output due to motor friction.

Note that the effect of friction is noticeably greater in the negative direction (it being more difficult to remove a positive offset) which introduces the common problem of direction dependent behaviour. It should also be stressed that the overshoot on the set-point response and the perhaps over-active load-disturbance rejection can be reduced or 'tailored' by the use of the design knobs and filters shown in the following section, (5.3). This particular experiment is only intended to illustrate the fundamental difference between positional and incremental algorithms in their treatment of offsets.

5.2.1 The requirement for $T(z^{-1})$

While Fig.5.2 shows that the move to incremental GPC eliminates steady-state offsets it can also seriously degrade the algorithm's overall performance, in the absence of $T(z^{-1})$, due to an increased sensitivity to high-frequency disturbances, modelling errors etc. Fig.5.3 shows the responses of both positional and incremental GPC, without the design filter $T(z^{-1})$, to illustrate this degradation. In both cases fixed-parameter GPC was used with N_1=NY=1, NY=8, λ=1e-8, $P(z^{-1})=T_c(z^{-1})=1$ based on models, positional and incremental respectively, estimated from the same data with $T_e(z^{-1})=1$.

The oscillatory unbroken lines are the actual experimental tip positions and the control actions which caused them. The corresponding dotted lines are results obtained when identically configured GPC controllers were

applied, in simulation, to the estimated models upon which they were based. The latter responses therefore show the closed-loop behaviour which would have been obtained if the estimates were exact ie. x(t)=0 in (2-3). The disturbance term x(t) lumps together all the actual deviations from this ideal response, which in this experiment are primarily due to plant-model-mismatch rather than load-disturbances since the results are repeatable. Differences between the two traces represent the effect of x(t).

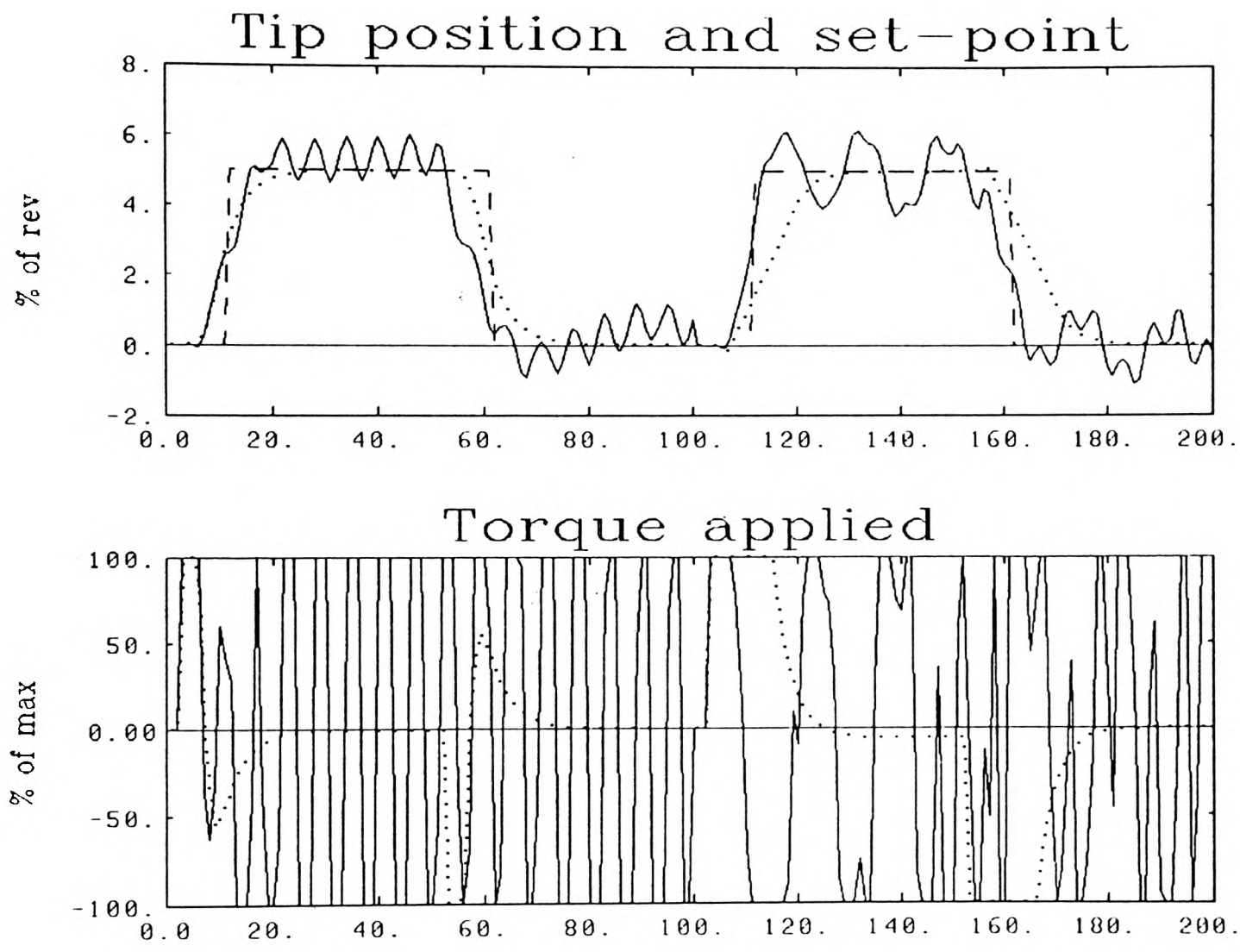


Fig.5.3: Positional and Incremental GPC without $T(z^{-1})$

Given the greater difference between actual and simulated results for the incremental algorithm it may be presumed that the incremental model is less accurate than the positional model. Clearly the model accuracy over the controller bandwidth has suffered from the high-frequency fit imposed by incremental least-squares (section 2.6.3). Furthermore Fig.5.4 shows that the incremental predictors on which the GPC controller is based are also more

sensitive to x(t) than the positional predictors. Given the respective estimated models the F_j polynomials of (2-5) and (2-59), for the positional and incremental controllers, which incorporate x(t) into the predictions, may be calculated and their magnitudes compared over the Nyquist frequency range. Fig.5.4 shows that over a large frequency range, which widens with the displacement into the future of the predictions, the magnitudes of the incremental F_j polynomials, $|F_j(e^{-j\omega h})|$, are far greater.

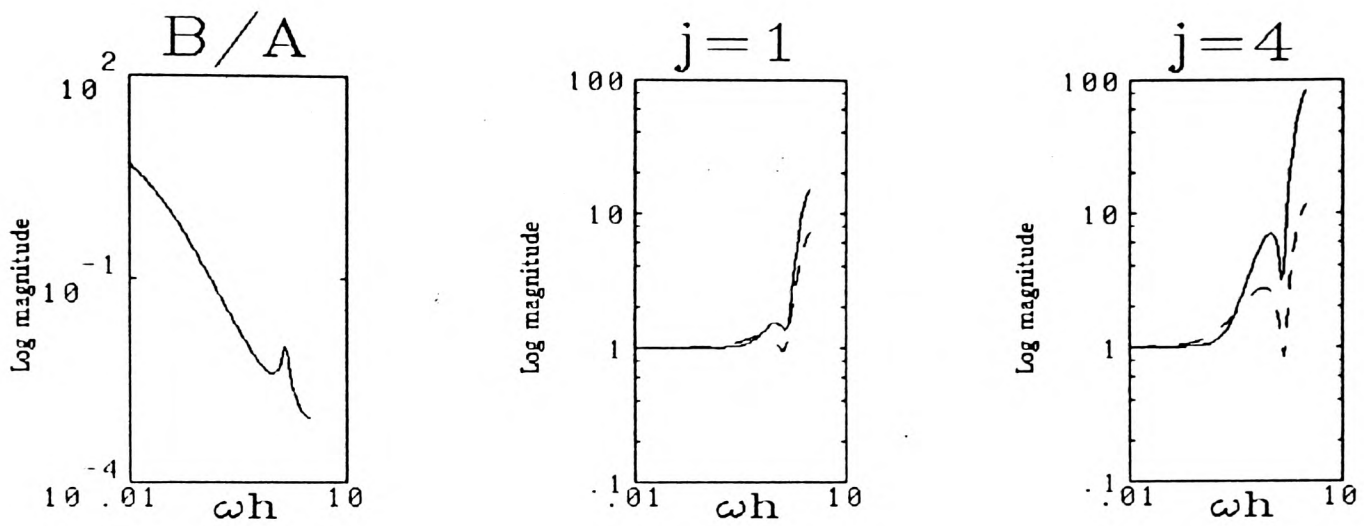


Fig.5.4: Sensitivity of the GPC control law to x(t)

Despite the fact that the incremental and positional controllers are fundamentally different, with the incremental controller having an inherent integrator, the increased sensitivity of both the incremental estimator and predictor at high frequencies mean that, almost regardless of the particular control strategy chosen, incremental algorithms must be protected by the

design filter $T(z^{-1})$ or some equivalent (Tham et al, 1987). Fig.5.2 of the previous section showed how $T(z^{-1})$ dramatically improved the control performance by reducing the sensitivity of the estimator and controller at high frequencies.

5.2.2 The Effect of Friction

To investigate the role that hub frictional effects play in this application a discrete-time simulation was developed using CTRLC*. A typical model estimated, using $T_e(z^{-1})$, from one of the experiments of the following sections was simulated in cascade with an idealised friction element as shown in Fig.5.5. To avoid confusion let the simulated model with friction be denoted by P2 while the model without friction is P1. Note that it is only possible to approximate the friction, which really depends upon the angular hub velocity, by using an estimate of the tip velocity $\Delta y(t)=y(t)-y(t-1)$, and assuming that it gives rise to a constant 6% torque opposing the direction of motion.

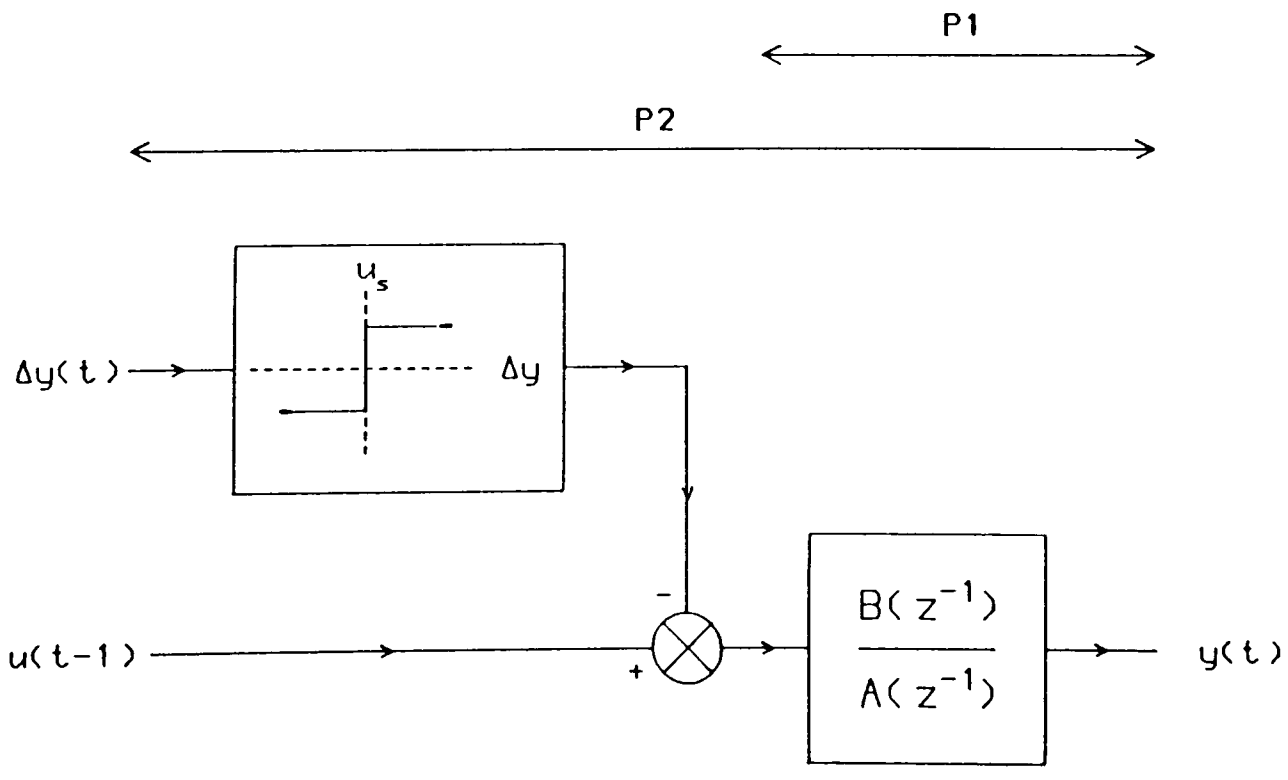


Fig.5.5: Structure of CTRLC simulation

* System Control Technology's Computer-Aided Workbench.

To see whether this simulation is accurate Fig.5.6 compares an actual run on the compliant arm using incremental GPC with N_1=NU=1, NY=8, λ=1e-8, $P(z^{-1})=T(z^{-1})=1$ with identically configured simulated GPCs controlling P1 (shown dotted) and P2 (shown solid). All these GPC controllers were designed upon the basis of the incremental estimated model from section 5.2.1, obtained without $T_e(z^{-1})$, which will be denoted by MO.

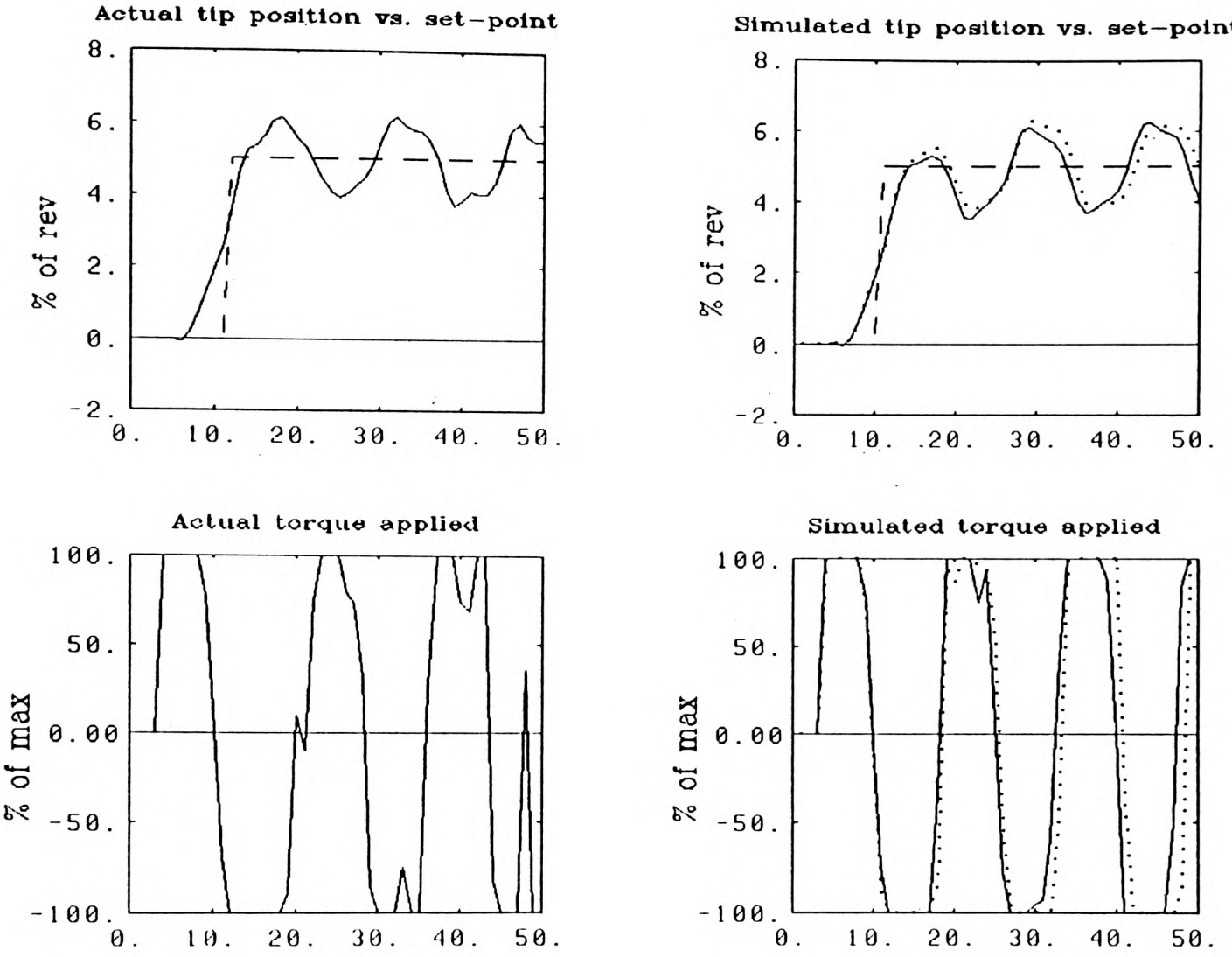


Fig.5.6: Effect of friction on control

The conclusions to be drawn from Fig.5.6 are:

(i) The simulation quite accurately reproduces the actual behaviour of the plant.

(ii) The amount of friction simulated (6%), which is in excess of that measured experimentally (≃4%), does not significantly affect the simulated closed-loop. Therefore the poor performance, in the absence of $T(z^{-1})$, is

primarily due to large differences between M0 and P1 ie. the estimated model upon which GPC is based and the simulated model 'behind' the friction.

Despite conclusion (ii) it is still possible that, during the estimation phase, friction caused the modelling errors which so seriously degrade the control perfomance. To investigate whether this is the case GPC, in simulation, was allowed to self-tune on P1 and P2 to obtain two models, M1 and M2 respectively, to illustrate the effect of friction on the estimates. Fig.5.7 compares the simulated set-point responses of fixed-parameter GPCs, configured as before and designed on the basis of M1 and M2, controlling the same simulated plant P2. Since the simulated responses are so similar, as are the frequency responses of M1 and M2, it may be concluded that the friction does not greatly affect the estimator.

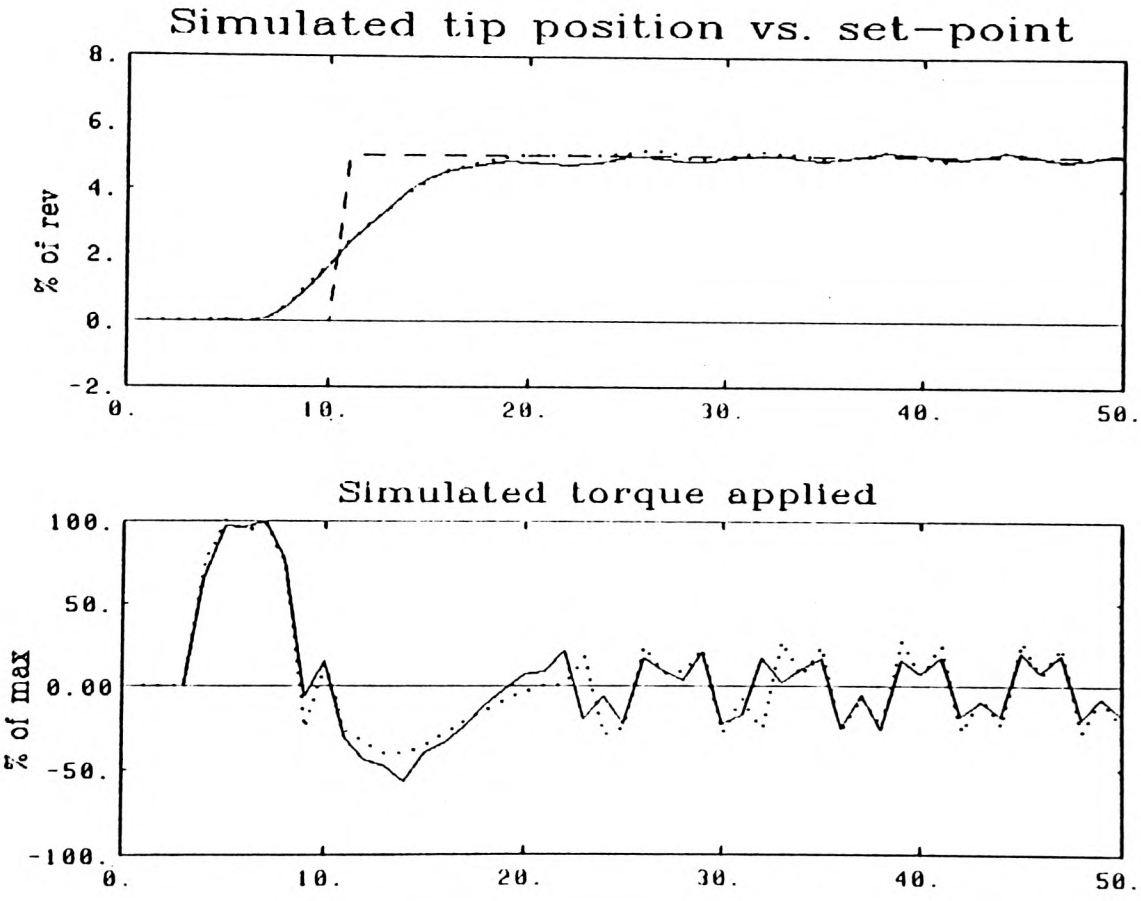


Fig.5.7: Effect of friction on the estimates

Furthermore the set-point responses of Fig.5.7 are far superior to those of Fig.5.6 with a large residual control limit cycle having only a small effect on the output. Fig.5.2 of section 5.2 shows how the use of

$T(z^{-1})$ in the estimator and controller gives similarly improved performance with a much reduced control limit cycle causing a minute steady-state output vibration of ≃ 1 quantisation (0.025%). Since it is impossible to control the output to within a quantisation level there is little that can be done to improve the steady-state without higher resolution sensing. For an adaptive GPC (see section 5.4) the estimator could be protected during prolonged low signal levels by the use of dead-bands (Jota, 1987).

To summarise, friction is not found to significantly deteriorate the performance of the GPC controller or the RLS estimator upon which it is based. What effect it has is seen to be greatly reduced by the incorporation of $T(z^{-1})$.

5.3 Tailoring the Closed-loop

The various tuning-knobs and design-filters of GPC give the user great flexibility in tailoring the closed-loop to requirement. For this not to be problematical a practical strategy is required governing their choice. This section systematically investigates their effects, providing interpretations and categorising them in terms of their relative usefulness. All the results presented below are for non-adaptive GPC.

5.3.1 Normal vs. Pre-specified set-points

There are two principal methods, normal and pre-specified, of incorporating the set-point signal into GPC. It is justifiable for this application to assume that a set-point trajectory would be available, say from a path-planner higher up in a hierarchical control scheme. Fig.5.8 shows an experimental run where a set-point trajectory of a small rectangular step (5 units), a large rectangular step (20 units) and a 1Hz sinusoid (amplitude

10 units) is used for 'pre-specified' GPC and repeated for 'normal' GPC. Throughout the run fixed-parameter incremental GPC is used with N_1=NU =1, NY=10, λ=1e-8, P=1, $T_c=(1-0.8z^{-1})^2$ based upon an incremental model estimated with $T_e(z^{-1})=T_c(z^{-1})$.

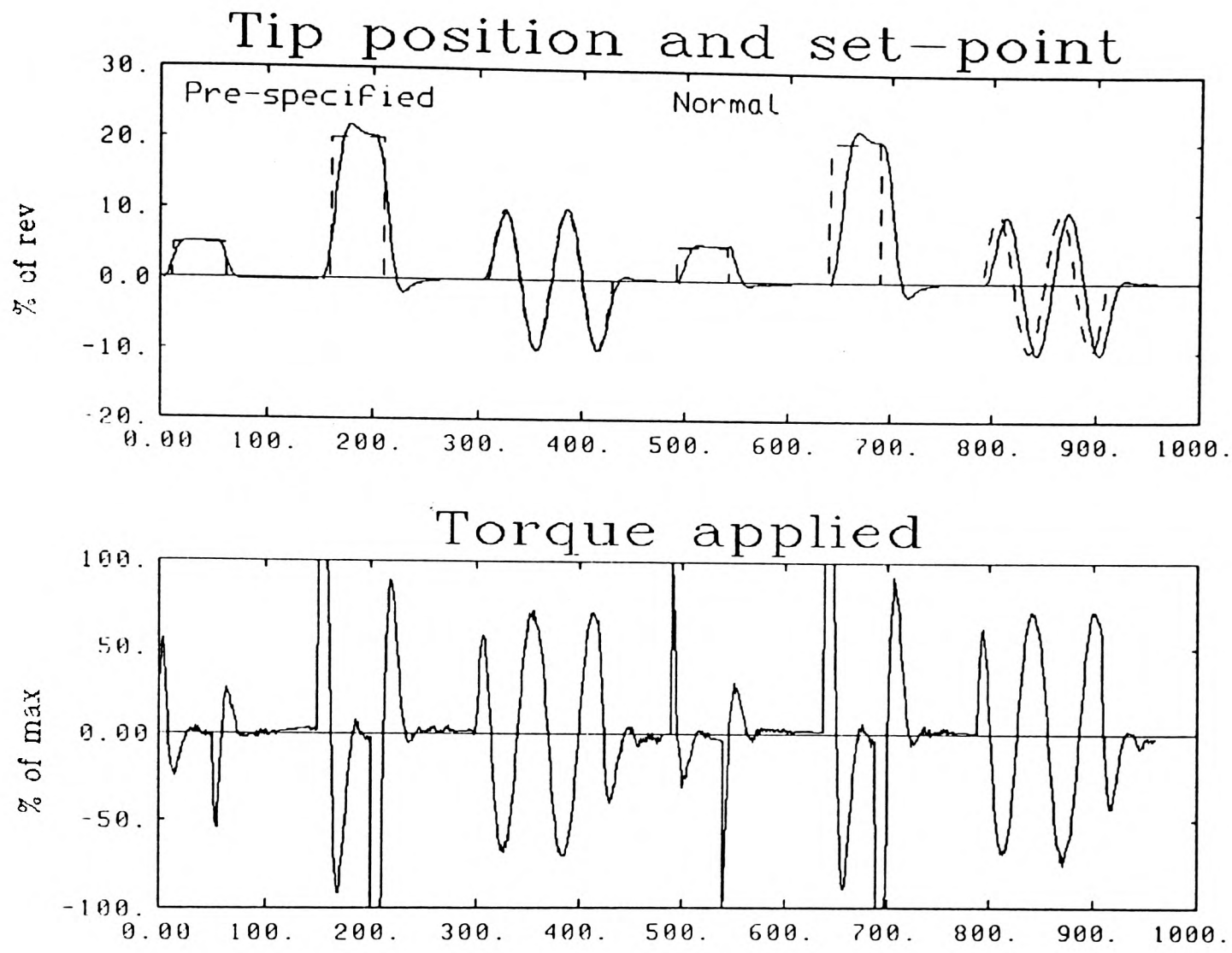


Fig.5.8: Normal vs. Pre-specified Set-points

Comparing the two small rectangular set-point responses it is seen that (apart from the obvious delay in the 'normal' response since 'normal' GPC recognises a set-point change NY-1 samples later than 'pre-specified' GPC) the initial control actions of 'normal' GPC are greater than those of 'pre-specified' GPC, causing greater overshoot. Intuitively the 'pre-specified' GPC sees the upward-going demand draw closer sample-by-sample and reacts more smoothly before the upward-going set-point change actually occurs.

Comparing the two large rectangular set-point responses it is seen that, again ignoring the inevitable delay, the responses are now virtually identical. This is due to the effect of actuator saturation. During the NY-1 samples immediately preceding the step change in the set-point, when the internal set-point vector **w** differs between 'normal' and 'pre-specified' GPC, both controllers remain saturated at u_{gpc}= +100 units. The saturation is due to a combination of the larger set-point change and the comparably high gains of both forms of GPC. By the time the controllers leave saturation on the downward-going control transient there is no difference between the internal set-point vectors and the algorithms are equivalent.

Finally, comparing the two responses to the sinusoidal set-point, it is seen that at low frequencies around 1Hz the main difference is that 'normal' GPC lags the set-point whereas 'pre-specified' GPC follows the set-point almost perfectly. As proven in appendix A.6 pre-specified set-points may be viewed as the effective use of an anti-causal set-point pre-filter, introducing phase-advance into the closed-loop.

5.3.2 Effect of $P(z^{-1})$

The compliant arm has quite limited torque actuation available for high-speed response. Large set-point changes, without the design filter $P(z^{-1})$, exhibit overshoot directly arising from actuator saturation. As described in section 2.6.1 $P(z^{-1})$ can be used to reduce or remove this overshoot. This is illustrated in the experimental run shown in Fig.5.9 where the set-point varies between 0 and 20 units (an angular deflection of 72^0). Fixed-parameter incremental GPC is used throughout with NY=8, N_1=NU=1, λ=1e-8, $T_c=(1-0.6z^{-1})^2$ based on an incremental model estimated with $T_e(z^{-1}) = T_c(z^{-1})$. Before every upward-going set-point change $P(z^{-1})$ is

changed and the controller parameters affected (eg. the top row of $[G_1^T G_1 + \lambda I]^{-1} G_1^T$) are passed to the CONTROL task.

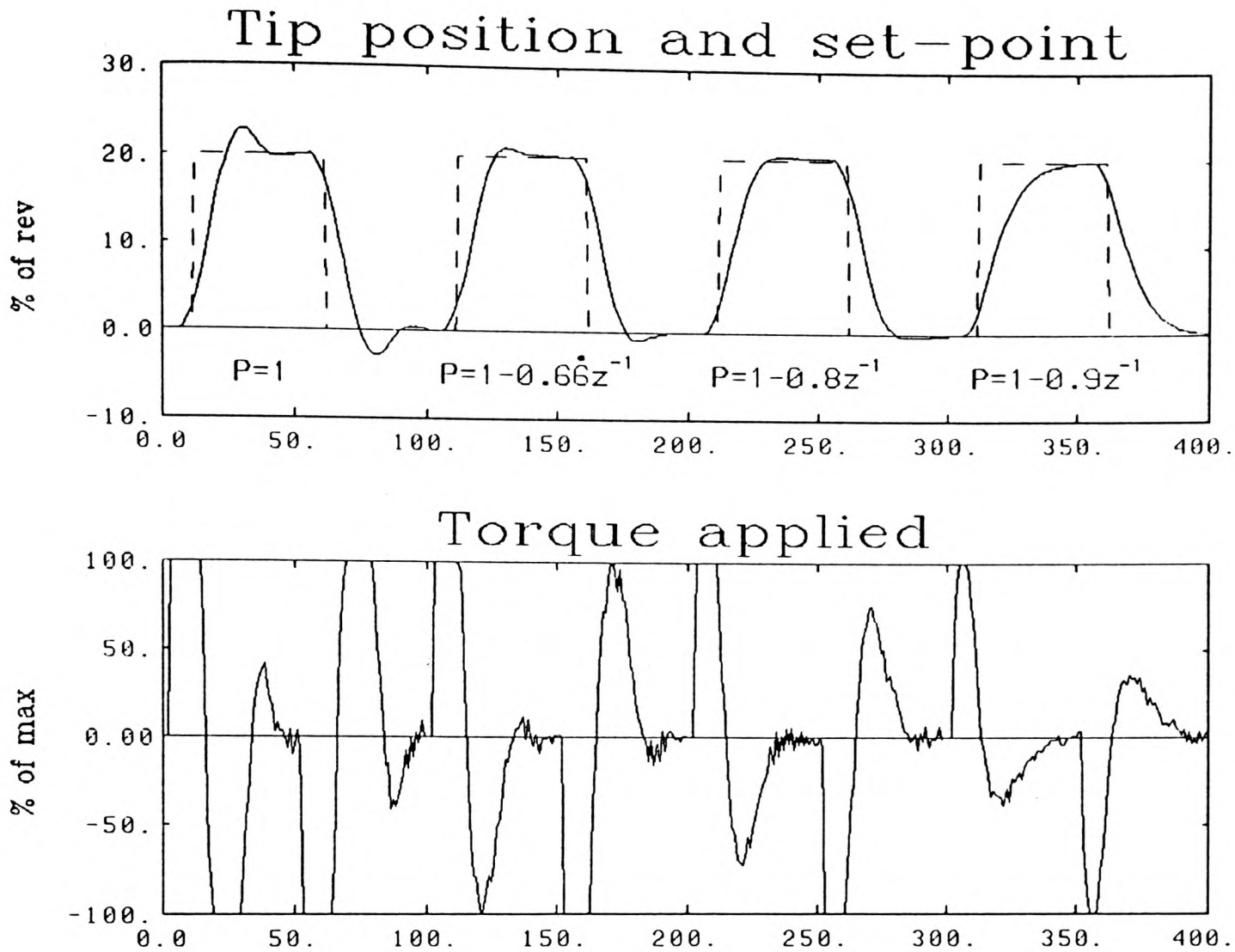


Fig.5.9: The effect of the design filter $P(z^{-1})$

Initially, with $P(z^{-1})=1$, there is a large overshoot which is clearly caused by actuator saturation on the 'braking' control transient. Remembering that the d.c. gain P(1) must be unity $P(z^{-1})$ was, on the subsequent set-point responses, changed to $P = 3-2z^{-1}$ (pole at z=0.66), to $P = 5-4z^{-1}$ (slower pole at z=0.8) and finally to $P = 10-9z^{-1}$ (even slower pole at z=0.9). The results agree well with the interpretations presented for the effect of $P(z^{-1})$, reducing and finally eliminating the overshoot at the expense of a progressively more sluggish closed-loop. However it should be noted that the actual settling time remains much the same from the initial to the final set-point changes. Note also that the sensitivity of the control to unmodelled

dynamics (such as friction) near the steady-state is reduced with the slower choices of $P(z^{-1})$.

5.3.3 Effect of $\lambda Q(z^{-1})$

The experimental run shown in Fig.5.10 shows the effect upon the closed-loop of varying λ in the simplified form of the design filter

$$\lambda Q(z^{-1}) = \lambda(1-z^{-1}) \qquad (5\text{-}2)$$

described in section 2.6.2. Fixed-parameter incremental GPC is used throughout with NY=8, N_1=NU=1, $P(z^{-1})$=1, $T_c=(1-0.6z^{-1})^2$ based on an incremental model estimated with the estimator polynomial $T_e(z^{-1})= T_c(z^{-1})$. For simplicity the scaling $\lambda = \lambda_{min} + \lambda_{rel}\hat{B}(1)^2$ is not used directly. For this fixed-parameter experiment λ is specified and λ_{rel} given parenthetically.

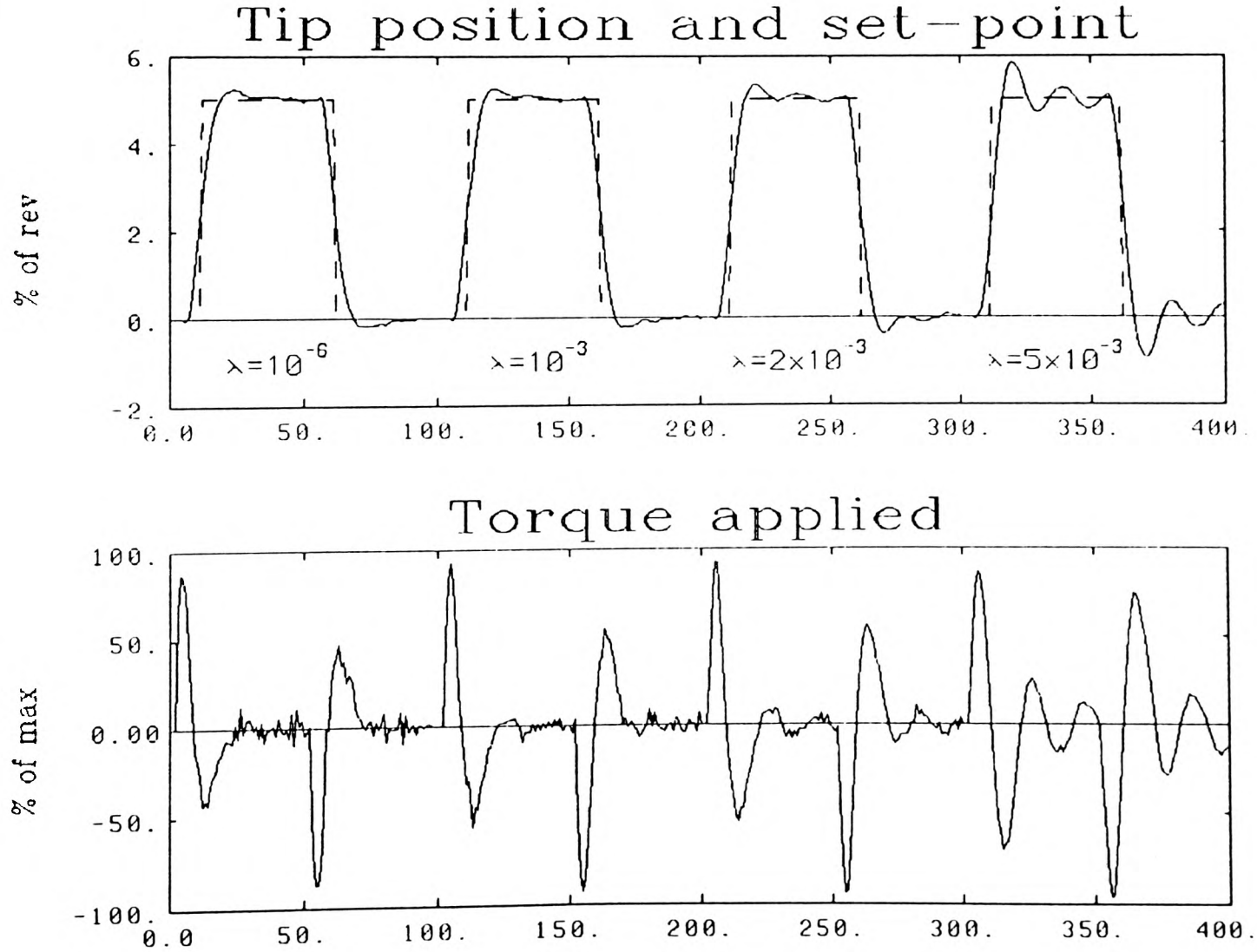


Fig.5.10: The effect of the design filter $\lambda Q(z^{-1})$

Before every upward-going set-point change λ is changed and the controller parameters affected (the top row of the matrix $[G_1^T G_1 + \lambda I]^{-1} G_1^T$) are passed to the CONTROL task. The initial value λ=1e-6 is effectively zero since its effect on the matrix $[G_1^T G_1 + \lambda I]^{-1}$ for this system and sample rate is negligible. A value of this magnitude would be used for λ_{min} to ensure that the inversion never becomes singular. The control performance is adequate apart from excessive control activity in the steady-state due to friction. This activity is substantially reduced by increasing λ to λ=1e-3 (λ_{rel}=150) with no deterioration in the overall response. However further control detuning with λ=2e-3 (λ_{rel}=300) and λ=5e-3 (λ_{rel}=750) results in progressively more 'wallowing' control which would be unacceptable.

Clearly the use of λ in this simplified structure provides a coarse detuning action on the control signal and therefore a useful and easy to interpret tuning-knob.

5.3.4 Effect of $T(z^{-1})$

Given that the closed-loop has been stabilised by the choice of $T = (1-0.8z^{-1})^2$ it is interesting to investigate the scope for subsequent fine-tuning of the $T(z^{-1})$ polynomial. This default choice for $T(z^{-1})$ may over-restrict the bandwidth over which the estimator attempts to model the plant and over which GPC predicts future plant behaviour (section 2.6.3). Tuning $\beta \rightarrow 0$ in (5-3) increases this bandwidth at the expense of increasing sensitivity to modelling errors and load disturbances. However a limited trade-off may be favourable as illustrated in Fig.5.11. Data generated from an active run (ie. plenty of information) was used to estimate off-line a group of models (with 5 $\hat{b}_i$ and 4 $\hat{a}_j$ parameters, k=1) over a range of values

of β where

$$T_e(z^{-1}) = (1-\beta z^{-1})^2 \qquad (5\text{-}3)$$

Fig.5.11 shows the effect on the closed-loop of using these models with incremental GPC; choosing NY=8, N_1=NU=1, λ=1e-8, $P(z^{-1})$=1. Before each upward set-point change the relevant model and the associated controller polynomial $T_c(\beta,z^{-1}) = T_e(\beta,z^{-1})$ are passed to the CONTROL task. This run, therefore, comprises a series of fixed-parameter GPC set-point responses to illustrate the effect of varying $T(z^{-1})$.

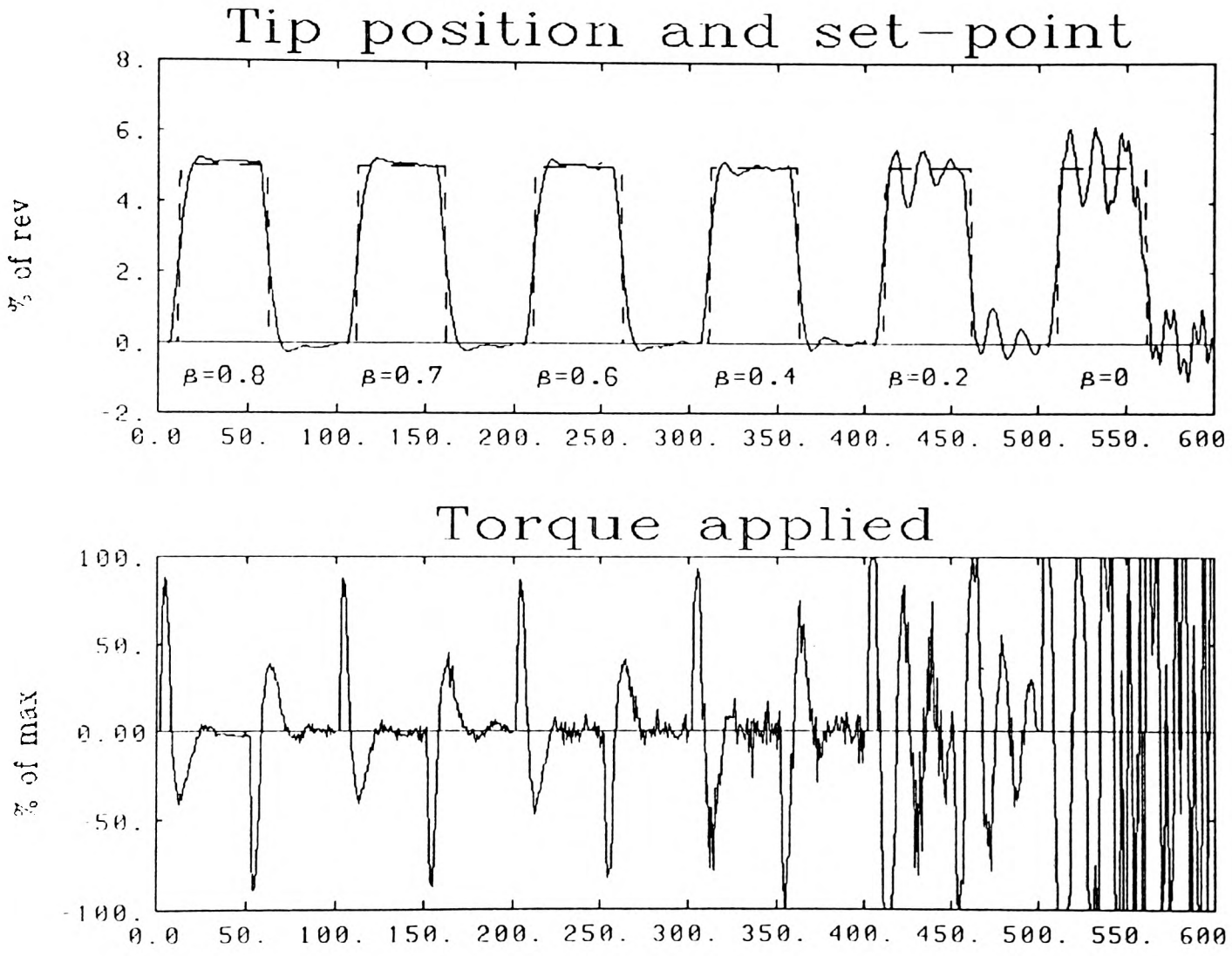


Fig.5.11: The stabilising effect of the design filter $T(z^{-1})$

As predicted $T(z^{-1})$ does not have a major effect on the gross servo response but rather it introduces a compromise between sluggish rejection of offsets in the steady-state (β=0.8, β=0.7) and virtual unstable rejection as

$\beta \to 0$. The choice $\beta=0.6$ seems to give the 'best' balance between smooth offset rejection and over-sensitivity to modelling errors. Since $T(z^{-1})$ has this major an effect it may be concluded that, in the known absence of load-disturbances (the experiment is repeatable), a substantial amount of plant-model-mismatch is present.

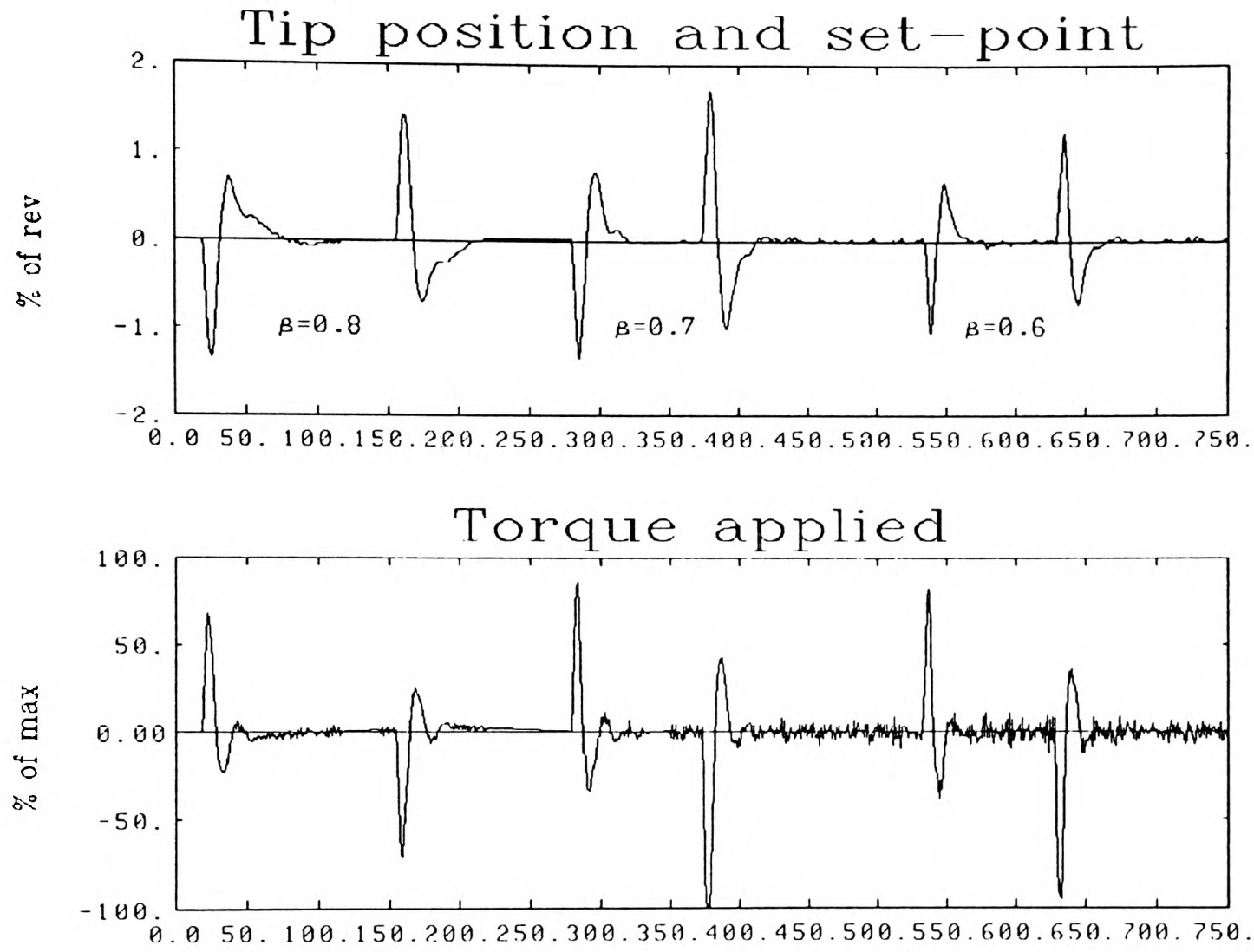


Fig.5.12: The load-disturbance tailoring effect of $T(z^{-1})$

$T(z^{-1})$ also has a more prosaic interpretation as a means of tailoring the response to load disturbances without affecting the servo response. Fig.5.12 shows GPC regulation, configured as above, interrupted by manually applied load disturbances caused by flicking the end-point. While this gave uneven disturbances the general trend is clear. As $T(z^{-1})$ is speeded up by moving from $\beta=0.8$ to $\beta=0.7$ and then to $\beta=0.6$ the speed and activity with which the disturbances are rejected increases. The corresponding set-point

responses of Fig.5.11 verify that over these values of β the gross set-point response is virtually unchanged.

5.3.5 Effect of N_1

The experimental run shown in Fig.5.13 shows the effect upon the closed-loop of varying the initial prediction horizon N_1 (see the GPC cost section 2.4). Fixed-parameter incremental GPC is used throughout with NU=1, NY=10, λ=1e-8, $P(z^{-1})=1$, $T_c=(1-0.6z^{-1})^2$ based on an incremental model estimated with the polynomial filter $T_e(z^{-1})= T_c(z^{-1})$. Before every upward-going set-point change N_1 is changed and the controller parameters affected (ie. the top row of the matrix $[G_1^T G_1+ \lambda I]^{-1} G_1^T$) are passed to the CONTROL task.

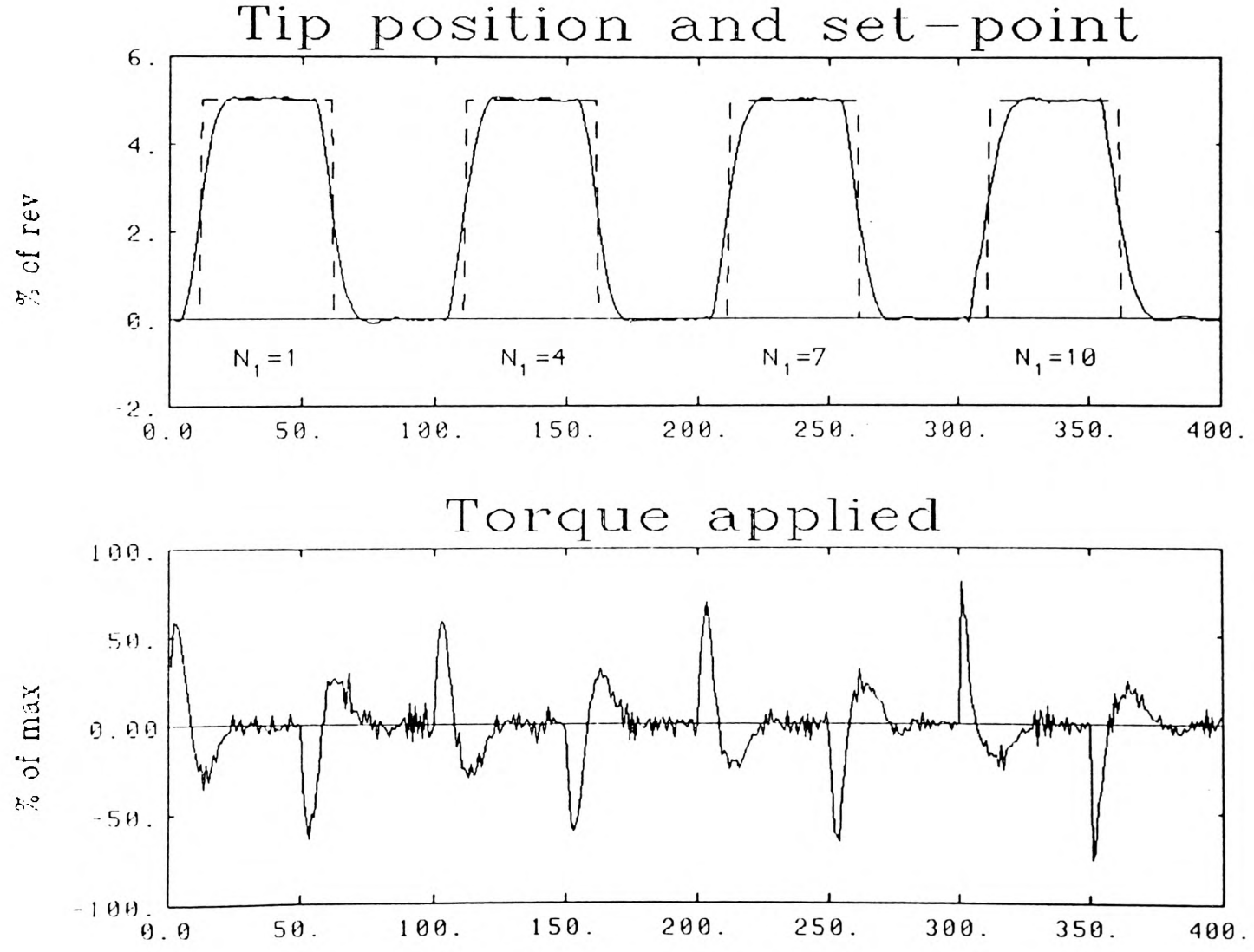


Fig.5.13: The effect of the tuning-knob N_1

For the four set-point responses shown N_1 was varied from N_1=1 to N_1=4 and then to N_1=7 and finally to N_1=10. It is interesting to note that changing N_1 has little effect upon the closed-loop. The size of the initial control transient increases slightly with N_1 although the settling time decreases slightly. This uniformity implies that the terminal error $\hat{y}(t+10|t)-w(t+10)$ is dominating the GPC cost function (2-31). Given the nature of the plant, basically a double integrator, this conclusion becomes less surprising. When NU=1 the cost is minimised subject to the constraint on future controls that

$$u(t+1) = u(t+2) = \ldots = u(t+NY-1) = u(t)$$

which means that the predicted 'forced' response $\hat{y}_d(t+j|t)$ (section 2.5.3) must be roughly parabolic in nature. Given the quadratic nature of the cost the last few predicted errors will therefore dominate, the situation getting 'worse' as NY increases. Consequently, for a plant of this type where NU cannot realistically be increased to 2 or more (see section 5.3.7) N_1 does not play a significant role. Increasing N_1 can therefore usefully be used to greatly decrease the amount of computation required to obtain u(t). For shorter NY and longer NU it is theoretically expected (Mohtadi, 1986) that N_1 should be of greater significance in its effect on the closed-loop.

5.3.6 Effect of NY

Fig.5.15 shows the effect upon the closed-loop of varying the final prediction horizon NY. Fixed-parameter incremental GPC is used throughout with N_1=NU=1, λ=1e-8, $P(z^{-1})$=1, $T_c=(1-0.6z^{-1})^2$ based on an incremental model

estimated with the polynomial filter $T_e(z^{-1}) = T_c(z^{-1})$. Before every upward-going set-point change NY is changed and the controller parameters affected (the top row of the matrix $[G_1^T G_1 + \lambda I]^{-1} G_1^T$) are passed to the CONTROL task.

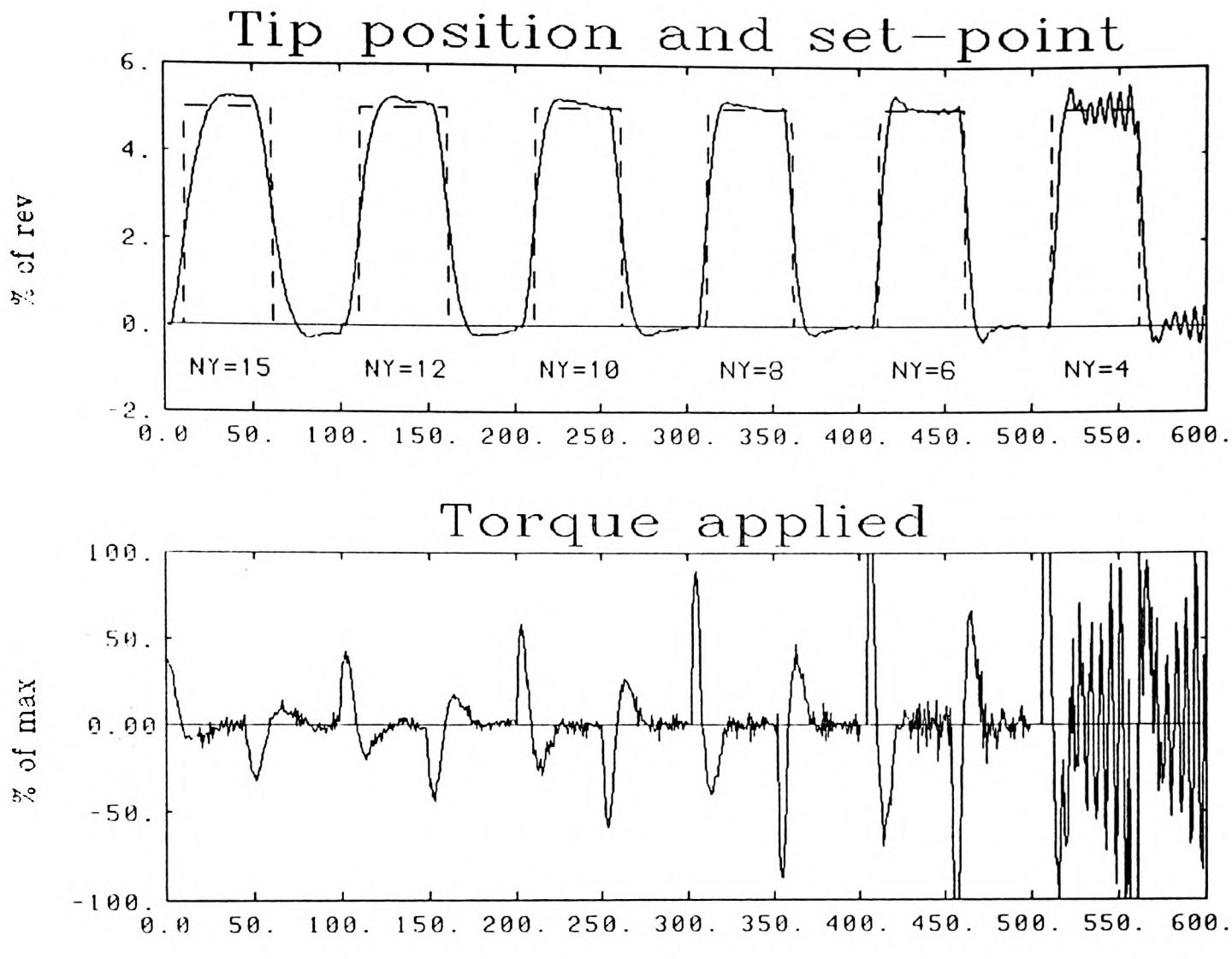


Fig.5.14: The effect of the tuning-knob NY

Through the run NY is steadily decreased from a horizon of 15 samples down to a horizon of 4 samples. As predicted in (Mohtadi, 1986) and observed in (Clarke et al, 1985) decreasing NY increases the speed of the closed-loop. Since this clearly involves increasing loop gain it comes as no surprise that at NY=4 the loop becomes virtually unstable with the increased gains acting upon the modelling errors (a combination of neglected dynamics and estimation errors). The default choice NY=10, chosen to match the typical sample rate of approximately a 1/10th of the plausible settling-time of the closed-loop, provides smooth control with a slightly persistent overshoot. The choice NY=8 might be chosen here as the best compromise between fast set-point response

and over-sensitivity to unmodelled dynamics such as friction, in the steady-state, as observed when NY=6.

NY is perhaps the most useful tuning-knob available to a GPC user; its effect on the speed of the closed-loop is predictable, relatively plant independent and restricted in the number of values it can take. Its detuning effect is different to λ which simply penalises the control in order to detune the loop, often resulting in 'wallowing' low-gain control (section 5.3.3). NY seems to detune the loop more directly, resulting in smoother detuned set-point responses than can be had by significant values of λ weighting.

5.3.7 Effect of NU

The experimental run of Fig.5.15 shows the effect upon the closed-loop of varying the control horizon NU. Fixed-parameter incremental GPC is used throughout with N_1=1, NY=15, λ=1e-5, $P(z^{-1})$=1, $T_c=(1-0.6z^{-1})^2$ based on an incremental model estimated with the polynomial filter $T_e(z^{-1})= T_c(z^{-1})$.

Initially NU=1 and, as theory predicts, the large prediction horizon of 15 samples (0.25 seconds) and presence of λ combine to give extremely detuned control. Before the second upward-going set-point change (t=100) NU is changed to 2 and the new top row of the matrix $[G_1^T G_1 + \lambda I]^{-1} G_1^T$ is passed to the CONTROL task. The resulting increase in gain of the controller is dramatic with the friction limit cycle oscillating between the actuator saturation limits. The continuation NU=3 is higher gain still and and an experiment could not be performed for fear of damaging the arm.

Clearly NU represents an extremely coarse tuning knob for controller gain. Despite the increase in gain the main reason for the deterioration in performance is that the GPC cost with NU>1 is minimised with no constraint on the amplitudes of the future unexerted controls u(t+1), .. ,u(t+NU-1). The

cost (2-31) can often be minimised by a large initial control u(t) which 'assumes' that even larger corrective controls u(t+1), .. , u(t+NU-1) may subsequently be applied. Of course the cost for NU>1 should be minimised subject to the knowledge that future controls cannot exceed actuator saturation limits (eg. Bohm, 1985; Tsang, 1987). When NU=1 this is, of course, not a problem since if $|u(t)| > u_{max}$ then all that can be done is to limit u(t) and apply it.

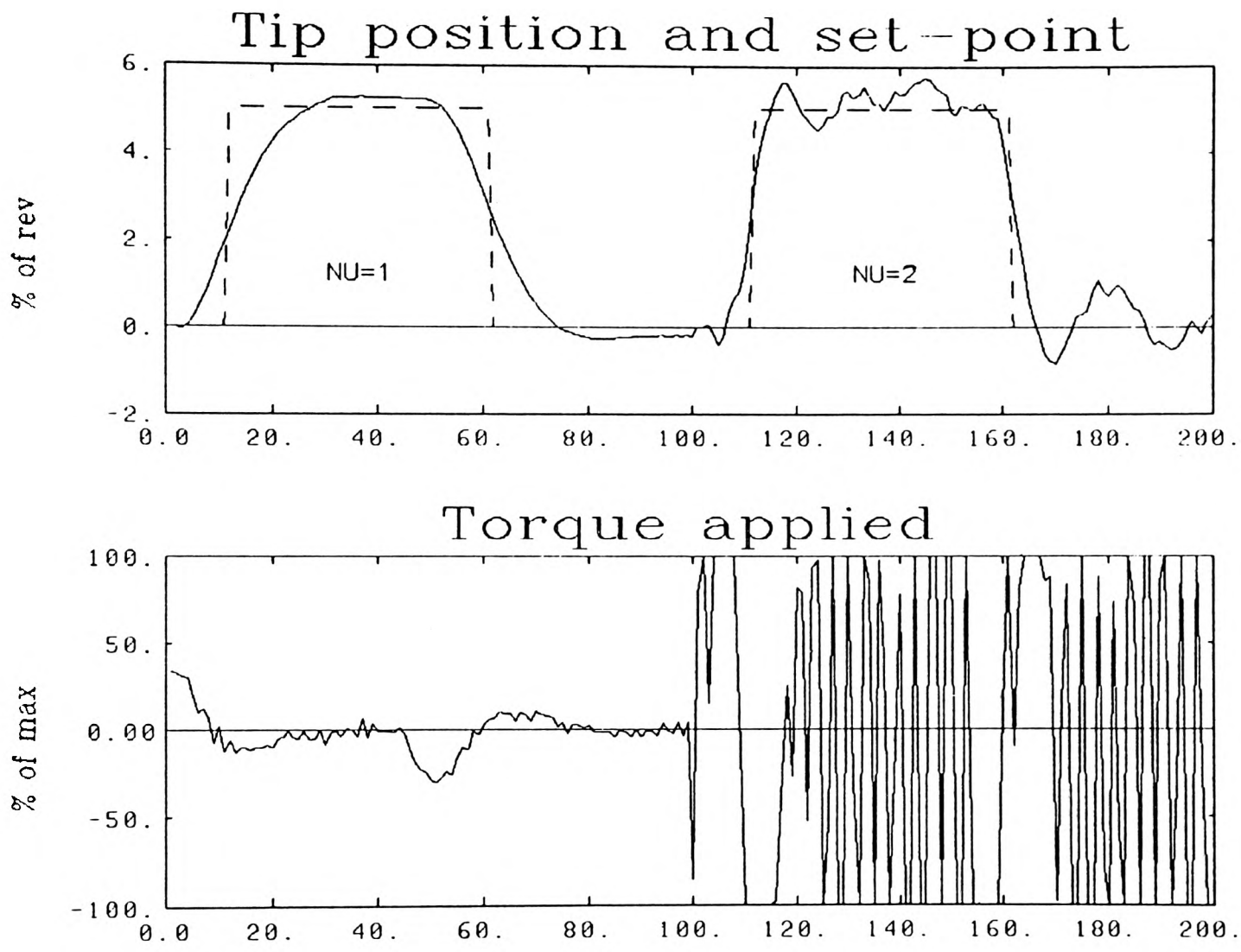


Fig.5.15: The effect of the tuning-knob NU

5.4 Adaptive GPC

The results of the preceding section show that a fixed-parameter incremental GPC controller can provide rapid and accurate control of tip position for the compliant arm. The controller design did not require

sophisticated prior modelling of the system since it was based upon an auto-regressive model directly estimated from test input/output data (section 1.1.3). In fact the only prior information required was the order of the dominant dynamics and the desired closed-loop bandwidth. Section 5.4.1 shows how, starting from this minimal information, full self-tuning GPC can rapidly converge to give this high-performance control. Section 5.4.2 then shows how continued adaptivity can be used to maintain high-performance control in the face of significant changes in the arm's dynamics with changing payload which would otherwise seriously degrade control.

5.4.1 Self-tuning GPC

Conventional attempts at the control of compliant arms (see section 1.2.1) usually start with the analytic derivation of an accurate model relating torque applied at the hub to the angular deflection of the tip. This model, since it must describe the behaviour of a system with distributed compliance, turns out to be of theoretically infinite order. In order to design a controller, usually in continuous-time, the model is truncated to consist only of the first few significant modes. The resultant fixed controller based upon this truncated model, which has presumably been verified against actual link behaviour, is then applied to the arm.

Fig.5.16 shows the control obtainable on the O.U. compliant arm from a considerably simpler and less time-consuming approach; that of self-tuning. During the 600 samples (10 seconds) of Fig.5.16 the GPC controller is self-tuning ie. at every sample the ESTIMATOR task re-designs the GPC controller based on a model it has recursively estimated from running i/o data acquired up to the previous sample and transfers the updated controller parameters to the CONTROL task. What is significant is that, apart from the underlying

decision on sample rate, the only initial information given to the GPC self-tuner was the estimated order of the dominant dynamics: chosen as fourth order (ie. 5 $\hat{b}_i$ and 4 $\hat{a}_j$ parameters) to describe the double integrators of the rigid mode plus the first lightly damped resonant mode. Starting from a zeroed model estimate it 'survives' an active tuning transient in the first 50 or so samples and quickly converges to the type of high-performance control achieved in section 5.3. The controller was configured with the standard settings NU=1, N_1=1, NY=10, λ=1e-4, P=1 and $T_c=(1-0.6z^{-1})^2$. The estimator was configured with $T_e(z^{-1})=T_e(z^{-1})$, an initial covariance of 1000I and no forgetting (ie. $\beta(t)=1$).

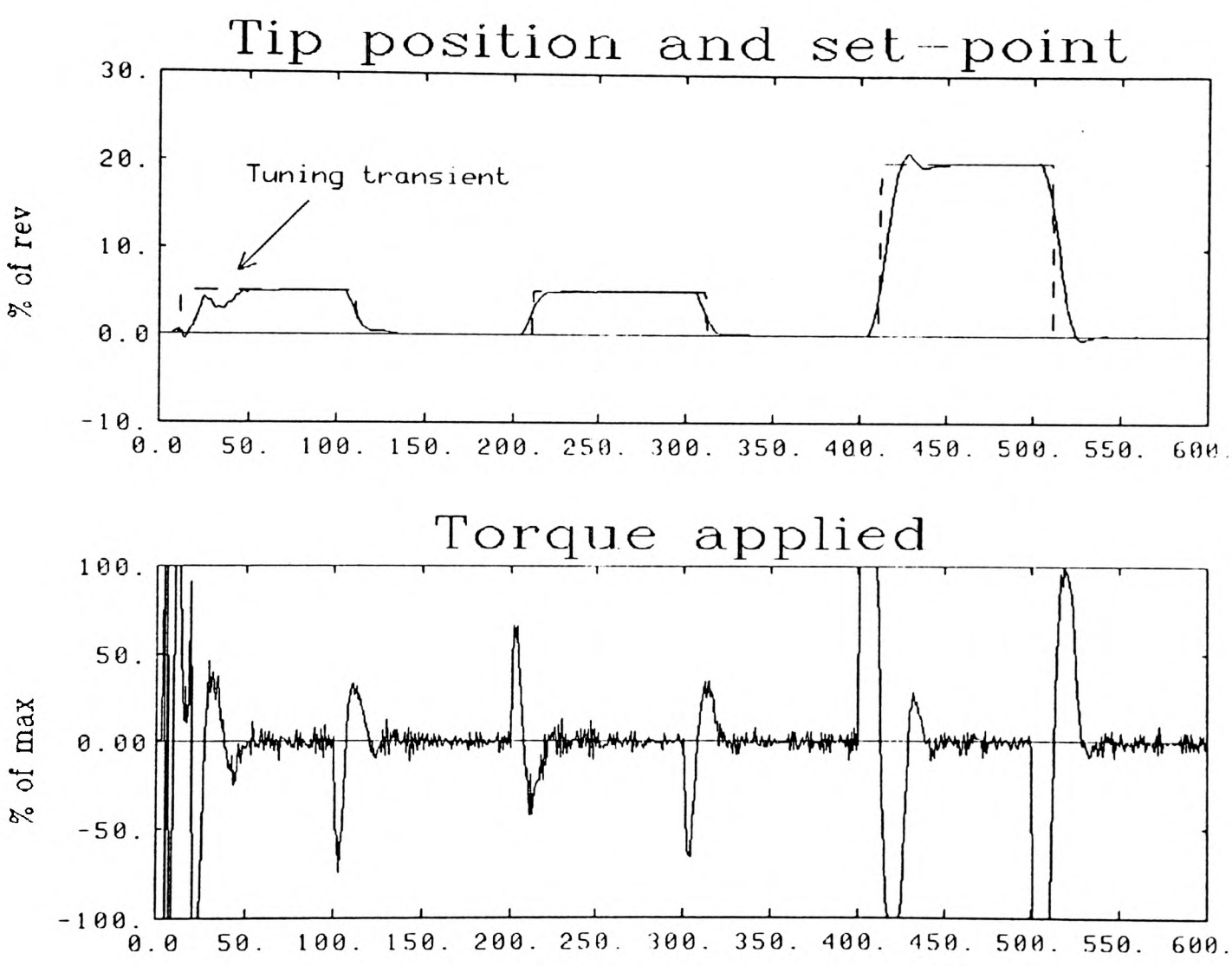


Fig.5.16: Self-tuning GPC

It is not intended to suggest that a self-tuner should be initialised in this fashion, in fact reducing λ to λ=1e-5 results in an even more active tuning transient which reaches saturation before the estimated model is

sufficiently accurate to return the control to the linear region; the self-tuner does not 'survive' the initial tuning. It is intended to show that the self-tuner, properly protected, can tune quickly (≈50 samples) to give excellent control with a minimal amount of information about the plant. As mentioned in section 5.1.1 self-tuners are conventionally initialised with a model estimated from PID or test signal data and use the techniques of section 1.1.3 to retain adaptivity and subsequently tune to a more accurate model.

5.4.2 Time-varying dynamics

The dynamics of a working compliant robot would change rapidly with time due to a varying payload at the end-effector and, for a multi-link robot, due to the strong dependence of inertias and inertial forces on the orientation of the robot. Continued adaptivity in a controller is one possible way to maintain high-performance control in the face of these time-varying dynamics.

The initial set-point response of Fig.5.17 shows fixed-parameter GPC control designed on the basis of a model estimated when the arm had no payload excepting its self-weight. At t=150, during steady-state, a payload is attached to the tip and the subsequent responses show the performance of the fixed controller with this additional end-mass in place (which reduces the first modal frequency from ≈15Hz to ≈14Hz and substantially increases the arm inertia as seen at the hub). Performance clearly deteriorates, with persisting overshoots and an oscillatory response. This is attributed to the GPC controller having been designed on the basis of a now obsolete model. The controller was configured with the standard settings NU=1, N_1=1, NY=10, λ=1e-4, P=1 and $T_c=(1-0.6z^{-1})^2$.

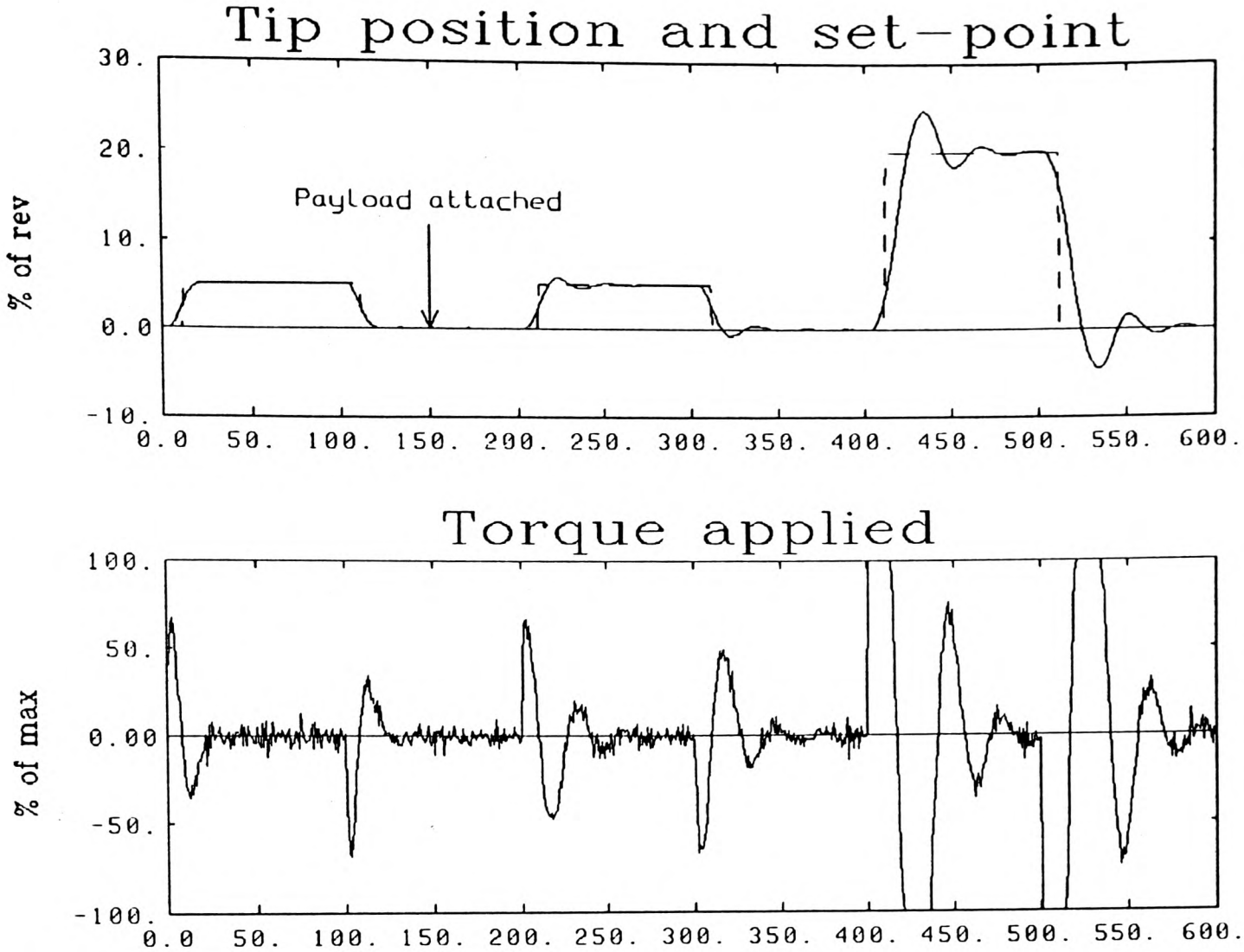


Fig.5.17: Fixed-parameter GPC with varying end-mass

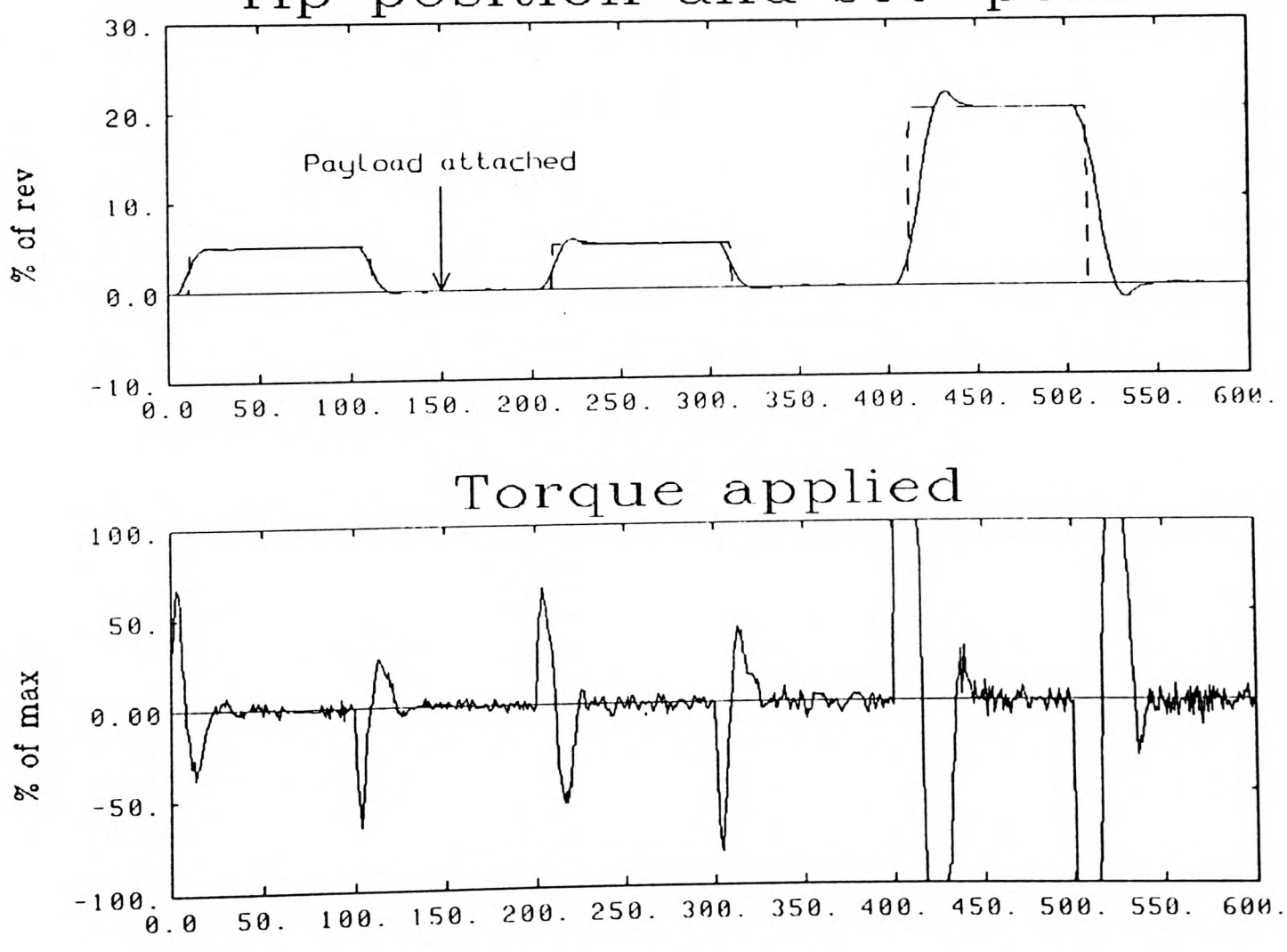


Fig.5.18: Self-tuning GPC with varying end-mass

Fig.5.18 repeats the above experiment with an identically configured GPC which, however, self-tunes throughout. Adaptivity is retained with time by using a fixed forgetting factor of 0.98 (ie. an asymptotic sample length of 50 samples*) with an initial covariance of 1000I, the initial tuning transient not being shown. As before the additional payload is fitted at t=150. The following upward-going set-point response at t=210 exhibits some overshoot but the adaption reacts quickly and it is much reduced from that observed in the previous Fig.5.17. The downward-going set-point response at t=310 is virtually returned to how it was before the payload was added (t=110) with an increased actuator signal to compensate for the increased inertia. The response to the large set-point changes is excellent, despite an inevitable overshoot from saturation which could be removed by $P(z^{-1})$. Fig.5.18 therefore shows how even the most primitive retention of estimator adaptivity can be used to maintain high-performance control in the face of time-varying dynamics.

5.5 Conclusions

The results presented in the previous sections of this chapter allow some conclusions to be drawn on the applicability of GPC for the control of compliant arms (and lightly damped systems in general) and on the ease with which a GPC user can set up and modify the controller characteristics. A final tuning exercise is described, illustrating the order in which the tuning-knobs and design filters are varied and the typical compromises involved.

* see section 1.1.3.

5.5.1 General Experimental Conclusions

The GPC controller, designed on the basis of an auto-regressive model estimated by the simple least-squares algorithm, provides excellent control in the face of very lightly damped resonances, multiple integrators, NMP behaviour and substantial motor non-linearity ie. friction and saturation. The ease with which stable control was achieved and then tailored is attributed to a set of useful (ie. understandable and effective) design parameters which allow sophisticated detuning from minimum-variance control (cancellation). It is also due to the fact that the closed-loop bandwidth does not approach the frequency of the first resonant mode (ie. the closed-loop could not follow a 10Hz sinusoidal set-point). Even so the link flexes considerably during set-point responses as illustrated in the lowest plot of Fig.5.19 which shows the typical angular deflections between tip and hub.

The incremental formulation of GPC is demonstrated to be essential for eliminating steady-state offsets in the closed-loop. The associated requirement for the filter $T(z^{-1})$, especially for incremental control and estimation, is also graphically illustrated.

The certainty equivalent matching of a least-squares estimator to the GPC controller gives a self-tuner which quickly converges to give the quality of control obtained from off-line estimation and controller design. The use of retained adaptivity in the estimator, via data forgetting, is shown to maintain high-performance control in the face of time-varying dynamics.

5.5.2 Categorisation of the GPC tuning-knobs

- Normal vs. Pre-specified set-points: Where the future set-point trajectory is available the use of pre-specified set-points in the GPC control calculation results in a beneficial inclusion of phase advance into

the closed-loop (see appendix A.6). Since pre-specified set-points equate to the use of an anti-causal set-point prefilter, external to the loop, it should not be expected to improve the loop stability or robustness.

° $\underline{T(z^{-1})}$: Since $T(z^{-1})$ can spell the difference between stability and instability it ranks as the most important design filter. This conclusion is in direct contrast to the fact that it is largely ignored in alternative control algorithms. There are however a few equivalents such as the 'surrogate noise filter' of Wellstead & Sanoff (1981). The experimental results of the preceding sections and parallel industrial trials indicate that $T(z^{-1})$ should always be present in both the estimator and control calculations, especially following the move to <u>incremental</u> algorithms. Its primary use is to ensure stability by reducing the sensitivity of estimator and predictor to the high-frequency modelling errors which will inevitably arise when attempting to describe a real high-order plant by a limited order model.

Once the plant has been stabilised $T(z^{-1})$ can be used to tailor the closed-loop response to load-disturbances without grossly affecting the servo response. This latter use for $T(z^{-1})$ is akin to its earlier role in the GMV algorithm (Clarke, 1982). A simple second-order $T(z^{-1})$ is proposed for use with incremental GPC and, given a sensible sample-rate, default values for its coefficients are suggested and shown experimentally to stabilise the closed-loop. It also seems that the $T_e(z^{-1})$ filter used in the estimator should be set equal to the $T_c(z^{-1})$ filter used in control.

° $\underline{\lambda Q(z^{-1})}$: The simplified structure of $\lambda Q(z^{-1})=\lambda(1-z^{-1})$ provides a straightforward reduction in control <u>rate</u> activity without the problem of λ-offset (section 2.5.1). However $\lambda(1-z^{-1})$ is limited in its usefulness by the fact that it weights future control increments with no consideration of the predicted errors in the cost (2-31). The consequence of this is that, while

its effect is perhaps the easiest to predict, excessive use reduces the rate at which the control can change to such an extent that the closed-loop tends to 'wallow'. In the adaptive case a simple scaling may be used to reduce the influence of plant gain on λ's effect. On the whole the role of $\lambda Q(z^{-1})$ for control detuning is much reduced from GMV by the superior alternatives mentioned below.

- $P(z^{-1})$: The simplified structure of $P(z^{-1})=(1-\alpha z^{-1})/(1-\alpha)$ provides a predictable injection of phase advance into the loop, analogous to the derivative term of a PID controller. This interpretation of $P(z^{-1})$, as opposed to the inverse model-following possibility, seems of more immediate use for this application. The 'anticipation' can be used to reduce overshoot and the amplitude of the high-frequency friction limit cycle. If chosen to reduce overshoot on a large (saturating) set-point change the response to smaller (non-saturating) set-point changes will, of course, be slowed but this is an unavoidable trade-off arising from applying linear control to a non-linear process. Since $P(z^{-1})$ slows the closed-loop response, rather than simply reducing the level of rate actuation, this is considered a more refined means of detuning than $\lambda Q(z^{-1})$. The resulting responses and actuation signals are also noticeably smoother with $P(z^{-1})$ present.

- NY: The final prediction horizon NY is obviously the primary tuning-knob for the GPC algorithm. Decreasing NY increases the speed of the closed-loop at the expense of higher loop gains which can lead to instability. At least for the common case where the control horizon NU=1 the horizon NY provides a very predictable means of manipulating closed-loop speed and controller gain. The default setting for NY, chosen to encompass the majority of the plausible closed-loop settling time, is verified to be a reasonable initial choice both in the experiments of the preceding sections and parallel industrial trials (De Keyser & Van Cawenberghe, 1985; Lambert, 1987;

Al-Assaf, 1987). It is also worth noting that reducing NY also reduces the phase advance attainable from using pre-specified set-points.

° NU: The control horizon NU gives an extremely coarse means of altering the controller gain and, therefore, the closed-loop speed. Any conclusions drawn here are, however, limited to the use of NU in systems with tight actuation constraints. Then NU is virtually useless since the control oscillates between the saturation limits due to the unconstrained nature of the GPC cost minimisation. The choice NU=1 has however, in the majority of reported applications, proved to be the most useful setting. Of course, the choice NU=1 does not suffer from any desaturation problems. It will be interesting to see the effect of NU given a constrained minimisation of the GPC cost (2-31) over the set of possible future controls (eg. Bohm, 1985).

° N_1: Little can be said about the effect of varying the initial prediction horizon N_1 since it, in the relevant experiment, actually had little effect. This is thought to be a consequence of NU being 1 and the plant being neutrally stable. It is also expected, for the same reasons, to be the case for unstable plant. It does seem that as N_1 approaches NY the gain increases but the effect is marginal.

5.5.3 Configuring the GPC controller

These conclusions may usefully be summarised by outlining a standard procedure for configuring incremental GPC through the choice of the above design parameters. The compliant arm is used as an example of how to apply this procedure.

° Choose the sample interval to be approximately 1/10th of the plausible settling time of the closed-loop and set NY appropriately ie. NY=10. This choice of NY should encompass the useful response time of the plant and allow adequate degrees-of-freedom for later tuning.

° Place the initial prediction horizon at or just beyond the plant delay ie. $N_1 \geq k$ with $N_1 = 1$ chosen in this case.

° Set the remaining tuning knobs to NU=1, $\lambda=\lambda_{min}$ (chosen by comparison with $\mathbf{G}^T\mathbf{G}$) and, as argued in section 2.6.3, $T_c(z^{-1}) = T_e(z^{-1}) = (1-0.8z^{-1})^2$.

The first 500 samples of Fig.5.19 show the performance given by this default GPC configuration with pre-specified set-points. Given stable control the tuning-knobs and design-filters can subsequently be varied to tailor the closed-loop to the user's requirement. The order in which this fine-tuning is described is the order in which, in practice, the design parameters should be varied.

° $T(z^{-1})$ may be altered to increase or decrease the rate at which offsets and load disturbances are rejected.

° NY should be varied to obtain the primary balance between speed of closed-loop response and loop gain.

° Beyond this $P(z^{-1})$ can be used to reduce overshoot and 'smooth' the closed-loop.

° By comparison with the preceding options the control weighting λ does not play a substantial role except for guaranteeing the invertibility of the matrix $(\mathbf{G}^T\mathbf{G}+\lambda\mathbf{I})$. It is, however, interesting to note that λ acts as the Lagrange multiplier in the constrained GPC minimisation for $NU \geq 2$ (Tsang, 1987). It can also be useful for temporary detuning through periods of adaption.

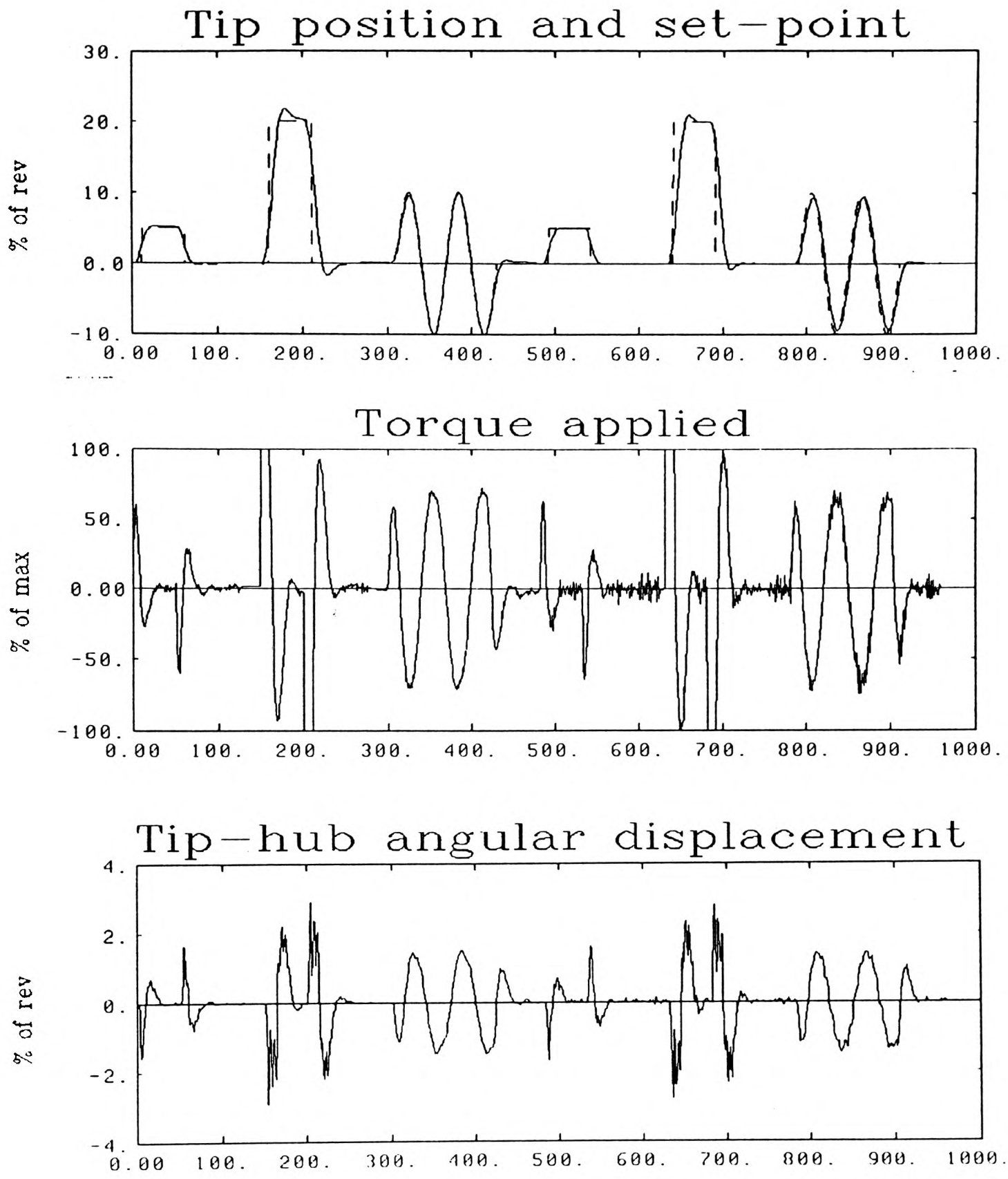


Fig.5.19: Tuning the GPC controller

The second 500 samples of Fig.5.19 shows the 'tuned' set-point responses with $T(z^{-1})$ varied to $(1-0.6z^{-1})^2$ and NY decreased to NY=8 as suggested by the results of sections 5.3.2 and 5.3.6. This gives a more active closed-loop with offsets being eliminated more rapidly. The choice of $P(z^{-1})=3-2z^{-1}$ is also used to reduce overshoot on the large set-point changes. It is interesting to note that this unavoidably introduces a slight

phase lag and magnitude roll-off on the sinusoidal set-point response since $\psi(t)=P(z^{-1})y(t)$ is being steered to the set-point rather than $y(t)$.

The approach itemised above has generally led, in several reported applications (see section 5.5.2), to stable and accurate control with minimal difficulty encountered in the tuning phase. The default or initial setting has especially proven to be a 'robust' selection in practice.

Chapter6

CONCLUSIONS AND FURTHER WORK

6.1 Conclusions

One of GPC's greatest strengths is that it provides the user with a great degree of control over the resulting closed-loop behaviour through the choice of a set of tuning-knobs N_1, NY and NU and design-filters $P(z^{-1})$, $\lambda Q(z^{-1})$ and $T(z^{-1})$. These parameters are designed to be simply related to meaningful continuous-time quantities such as speed of response, damping and overshoot. In particular heuristic interpretations have been given which corroborate the observed effect of the design filters. These interpretations, together with the experimental examples and simulated tuning exercises, have been used to outline rules for the user to follow in tailoring the closed-loop. While many simplifying assumptions have of necessity been made the design filters (and tuning knobs) are shown to be straightforward to choose and effective in their choice.

The need for incremental algorithms to overcome the offset problems prevalent in many preceding control strategies is highlighted together with the associated and equally important need for filtering all incremental data by $T(z^{-1})$. This little understood ancillary requirement has lead to poor incremental performance in some application reports (eg. Tham et al, 1987) and has an important bearing on basic modelling aspects such as the bandwidth over which the plant model is required to be accurate and how this should effect controller design.

The major part of the work in this project concerns the testing of the GPC algorithm on a somewhat novel experimental rig. The O.U. single-link

compliant arm, built to exhibit exaggerated flexibility, represents a move away from control of the generally slow and stable low-order process plant to which self-tuning is usually applied. This high-order system, neutrally stable with lightly damped structural resonances, is also unusual in being relatively noise-free with a consequential high-performance requirement for the closed-loop.

It is verified experimentally that incremental GPC control yields accurate offset-free control of tip position and that data conditioning by $T(z^{-1})$ is required to ensure stability. With the protection afforded by $T(z^{-1})$ the experimental results show excellent control performance, even when the self-tuner was initialised with minimal prior information. The major restriction on closed-loop performance turned out to be the limited actuation available. While this was a drawback it nevertheless provided useful information as to how GPC performs in a tightly saturated and hence non-linear system. It is interesting to note that the quite large amount of motor friction present ($\simeq$4% of the motor's torque range) does not present a particular problem, especially after the incorporation of $T(z^{-1})$. It does, however, result in a steady-state limit cycle whose measured amplitude is approximately one angular quantisation. This can only be reduced by either increasing the resolution of the angular displacement sensing or, possibly, by the use of additional measurements (see section 6.2).

Although adaption aspects were not concentrated on it is shown experimentally that even the most naive estimator configuration (least squares with fixed-forgetting) can maintain high-performance control in the face of strongly time-varying dynamics.

A secondary part of the work described in this thesis concerns means of incorporating prior knowledge into the least squares estimator and its effect upon the self-tuner's performance. This coincides well with the compliant arm application where easily available knowledge pertaining to the

arm dynamics is successfully used to dramatically reduce the number of parameters to be estimated with no degradation in the converged models. Simulation and experimental results show that the incorporation of many kinds of plausibly available prior knowledge is both straightforward and effective in improving the convergence characteristics of the estimator and can reduce the computational workload. In particular the recent δ-transform operator is shown to have useful implications for simplifying the inclusion of prior knowledge into self-tuners running at high sample rates (relative to the open-loop bandwidth).

The implementation of a self-tuning controller proved a complicated project in itself. The hardware and software engineered in the course of this research enabled full GPC to self-tune at sample rates in excess of 100Hz. This is considered pertinent both to the control of other electromechanical systems requiring rapid sampling and, perhaps more commonly, to control computers overseeing multiple loops. To a first approximation a controller capable of sampling a single loop at 100Hz should also be capable of sampling 100 different loops at 1Hz, an adequate rate for the majority of process plant.

The detailed division of the GPC self-tuning control algorithm into a task structure, with some tasks executing sequentially while others execute concurrently, lays some foundations for the inevitable move to parallel computing architectures and enhanced software specifications (eg. ADA, OCCAM). The virtually complete separation of the estimation task from the control task also mirrors the suggestion of Wittenmark, (1975), that a self-tuner may usefully be configured as a periodic retuner rather than for continual adaptivity, perhaps retuning several control loops.

As far as compliancy in robotics is concerned GPC can clearly cope well with the lightly damped and NMP nature of the link dynamics arising from

end-point feedback. This is principally because GPC does not attempt to directly cancel the plant dynamics (minimum-variance) and provides sophisticated detuning facilities to retain good performance while detuning from minimum-variance. With the same actuation there is no doubt that the lower inertia of a compliant arm is capable of swifter controlled motion than a comparable rigid arm. It should be noted that the following arm mechanism used to sense end-point position has a beneficial effect as far as control is concerned, reducing the effect of the arm's structural resonances on the tip position. While this was not the initial intention it is worth considering since the next generation of compliant robot (Daniel et al, 1987) supports the weight of the motors or drive train on a rigid casing which is also used for end-point sensing.

Finally the literature often reports that self-tuning control has gained little industrial acceptance. This, it is conjectured, is perhaps due to it being too different from conventional control schemes allied with the relatively small number of reported full-scale industrial trials. Both of these objections may be diminished by the appearance of commercial but much simplified self-tuners such as the SATT ECA40 (previously NAF Autotuner) and TCS 6355 which are becoming increasingly popular. However the ease with which satisfactory closed-loop performance was attained in this application and the quality of that performance on this and parallel industrial trials, without any prior modelling or plant-specific design, suggests that sophisticated self-tuners such as GPC also justifiably deserve to be a much more prevalent control tool.

6.2 Further Work

With regard to the compliant arm application it would be useful to investigate the possible use of high-bandwidth strain feedback at the root in a cascade strategy designed to overcome the friction non-linearity of the motor. The effect that this inner loop would have on the arm dynamics as observed by the outer loop should also be of interest. The extension of the methods used throughout this thesis to a multi-link compliant (or rigid) robot is also expected to provide much scope for valuable further research.

With regard to the work done on the inclusion of prior knowledge into self-tuners the logical continuations are considered to be:

(i) Investigation of the likely deleterious effects on closed-loop behaviour arising from incorporating inaccurate prior data.

(ii) The unification of simple initialisation approaches such as the step-up-step-down method of Al-Assaf (1987) or the relay method of Astrom (1985) with the mechanisms presented in this thesis for the inclusion of the knowledge garnered from these prior tests into advanced self-tuners.

(iii) A full implementation of GPC in the δ-transform to further investigate the advantages and disadvantages to be had over the conventional z-transform.

APPENDIX A

DERIVATIONS AND PROOFS

All expressions and equations taken from sections referring to these appendices are presented with minimal explanation. Where possible, however, the equation number from the relevant section is given.

A.1 Proof that iterating the known part of the CARMA model (1-4) gives the same predictions as the diophantine approach (sections 2.3.1 & 2.3.2).

Proving that each method yields identical predictions is equivalent to proving that the prediction errors are identical ie.

$$\varepsilon(t+j|t) = e(t+j|t) = y(t+j) - \hat{y}(t+j|t) \qquad (A-1)$$

where $\varepsilon(t+j|t)$ and $e(t+j|t)$ are the errors associated with model iteration and diophantine solution respectively. The proof proceeds in stages.

(i) Consider the predictions and prediction errors of the diophantine method

$$\hat{y}(t+j|t) = E_j Bu(t+j-1) + F_j y(t) \qquad (2-7)$$

$$e(t+j|t) = E_j x(t+j) \qquad (2-8)$$

and the model iterative method

$$\hat{y}(t+1|t) = -a_1 y(t) - a_2 y(t-1) - \ldots - a_{na} y(t-na+1) + Bu(t) \qquad (2\text{-}9)$$

$$\hat{y}(t+2|t) = -a_1 \hat{y}(t+1|t) - a_2 y(t) - \ldots - a_{na} y(t-na+2) + Bu(t+1)$$

$$\hat{y}(t+3|t) = -a_1 \hat{y}(t+2|t) - a_2 \hat{y}(t+1|t) - \ldots - a_{na} y(t-na+3) + Bu(t+2)$$

$$\vdots$$

$$\varepsilon(t+1|t) = x(t+1) \qquad (A\text{-}2)$$

$$\varepsilon(t+2|t) = x(t+2) - a_1 \varepsilon(t+1|t)$$

$$\varepsilon(t+3|t) = x(t+3) - a_1 \varepsilon(t+2|t) - a_2 \varepsilon(t+1|t)$$

$$\vdots$$

$$\varepsilon(t+j|t) = x(t+j) - a_1 \varepsilon(t+j-1|t) - a_2 \varepsilon(t+j-2|t) - \ldots - a_{j-1} \varepsilon(t+1|t)$$

or $$\varepsilon(t+j|t) = x(t+j) - a_1 \varepsilon(t+j-1|t) - \ldots - a_{na} \varepsilon(t+j-na|t)$$

where the latter two cases represent $j \leq na$ and $j > na$ respectively and require slightly different treatments. When $j \leq na$ the output state vector in the jth model iteration of (A-2) contains a mixture of past 'y's and previous predictions '$\hat{y}$'s. When $j > na$ the state vector simply contains previous predictions.

(ii) The proof is by induction. Assume for the moment that the prediction errors up to j-1 are identical ie.

$$\varepsilon(t+1|t) = e(t+1|t) = E_1 x(t+1) \qquad (A\text{-}3)$$

$$\vdots$$

$$\varepsilon(t+j-2|t) = e(t+j-2|t) = E_{j-2} x(t+j-2)$$

$$\varepsilon(t+j-1|t) = e(t+j-1|t) = E_{j-1} x(t+j-1)$$

APPENDIX A

Substituting (A-3) into (A-2) gives

$$j<na: \quad \varepsilon(t+j|t) = x(t+j)-a_1z^{-1}E_{j-1}x(t+j)-\ldots-a_{j-1}z^{-j+1}E_1x(t+j) \qquad (A-4)$$

$$j>=na: \quad \varepsilon(t+j|t) = x(t+j)-a_1z^{-1}E_{j-1}x(t+j)-\ldots-a_{na}z^{-na}E_{j-na}x(t+j)$$

If it can be proven that, in both cases, $\varepsilon(t+j|t)=E_jx(t+j)$ and that $\varepsilon(t+1|t)=e(t+1|t)$ then by induction it is successively proven that

$$\varepsilon(t+2|t) = (1-a_1z^{-1}E_1)x(t+2) = E_2x(t+2) = e(t+2|t) \qquad (A-5)$$

$$\varepsilon(t+3|t) = (1-a_1z^{-1}E_1-a_2z^{-2}E_2)x(t+3) = E_3x(t+3) = e(t+3|t)$$

$$\vdots$$

which is as required.

(iii) Considering first the case when $j \leq na$ it is required, from (A-4), to prove that

$$E_j + a_1z^{-1}E_{j-1} + \ldots + a_{j-1}z^{-j+1}E_1 = 1 \qquad (A-6)$$

The corresponding bank of diophantine equations

$$1/A = E_1 + z^{-1}F_1/A \qquad (A-7)$$

$$1/A = E_2 + z^{-2}F_2/A$$

$$\vdots$$

$$1/A = E_j + z^{-j}F_j/A$$

may be scaled and summed to give

$$\frac{(1 + a_1z^{-1} + \ldots + a_{j-1}z^{-j+1})}{A} = E_j+a_1z^{-1}E_{j-1}+ \ldots +a_{j-1}z^{-j+1}E_1+z^{-j}(h.o.t^*)$$

* Higher order terms.

where the final term in parenthesis contains all terms of order greater than or equal to j. This grouping is possible since the polynomial E_k is of order k-1. Comparing low powers of z^{-1}

$$E_j + \ldots + a_{j-1}z^{-j+1}E_1 = \text{first j terms of } \left\{ \frac{1 + a_1z^{-1} + \ldots + a_{j-1}z^{-j+1}}{A} \right\}$$

which simplifies as

$$\begin{aligned} \text{first j terms of } \{..\} &= \text{first j terms of } \left\{ \frac{A - z^{-j}(\text{h.o.t})}{A} \right\} \\ &= \text{first j terms of } \left\{ 1 - \frac{z^{-j}(\text{h.o.t})}{A} \right\} \\ &= 1 \end{aligned}$$

and therefore proves (A-6).

(iv) The proof for j>na is virtually the same as in (iii), except that it is now required to prove

$$E_j + a_1z^{-1}E_{j-1} + \ldots + a_{na}z^{-na}E_{j-na} = 1 \qquad \text{(A-8)}$$

Repeating the scaling and summing exercise with the last na diophantine equations of (A-7) gives

$$\frac{(1 + a_1z^{-1} + \ldots + a_{na}z^{-na})}{A} = E_j + a_1z^{-1}E_{j-1} + \ldots + a_{na}z^{-na}E_{j-na} + z^{-j}(\text{h.o.t})$$

which immediately reduces to

$$E_j + a_1z^{-1}E_{j-1} + \ldots + a_{na}z^{-na}E_{j-na} + z^{-j}(\text{h.o.t}) = 1$$

Comparing low powers of z^{-1} proves (A-8).

(v) To complete the proof by induction it is only necessary to show that $\varepsilon(t+1|t) = e(t+1|t)$. This is trivial since solving the first diophantine equation gives $E_1 = 1$, so

$$\varepsilon(t+1|t) = x(t+1) = E_1 x(t+1) = e(t+1|t)$$

QED

A.2 Proof that iterating the known part of the CARIMA model (1-5) gives the same predictions as the incremental diophantine approach (see section 2.5.3).

This proof follows from that given in the preceding section. Consider the incremental diophantine equation and resulting predictor

$$1 = E_j A\Delta + z^{-j} F_j \tag{2-25}$$

$$\hat{y}(t+j|t) = E_j B\Delta u(t+j-1) + F_j y(t) \tag{2-26}$$

These expressions can also be obtained by replacing A with $A\Delta$ and B with $B\Delta$ in the corresponding equations of section A.1. Therefore the diophantine predictions can be calculated by iterating the model

$$y(t) = \frac{B\Delta}{A\Delta} u(t-1) \tag{A-9}$$

from current state. With reference to Fig.A.1

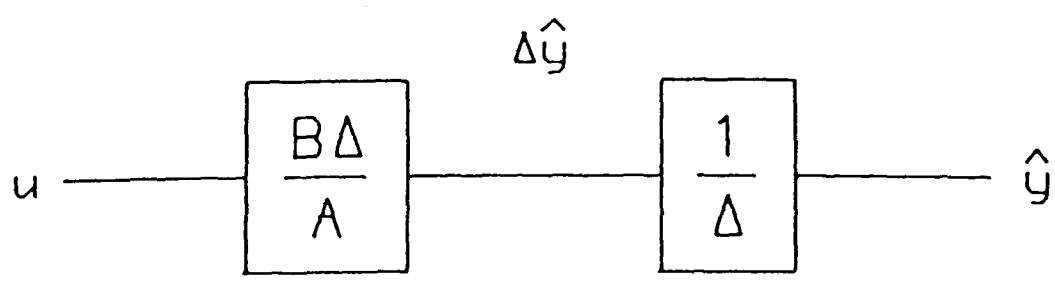


Fig.A.1: Incremental model iteration

APPENDIX A

the incremental model-iterative method first iterates the model $B\Delta/A$ ie.

$$\hat{\Delta y}(t+1|t) = -a_1\Delta y(t)-a_2\Delta y(t-1)- \ldots -a_{na}\Delta y(t-na+1) + B\Delta u(t) \qquad (2\text{-}28)$$

$$\hat{\Delta y}(t+2|t) = -a_1\hat{\Delta y}(t+1|t)-a_2\Delta y(t)- \ldots -a_{na}\Delta y(t-na+2) + B\Delta u(t+1)$$

$$\hat{\Delta y}(t+3|t) = -a_1\hat{\Delta y}(t+2|t)-a_2\hat{\Delta y}(t+1|t)- \ldots -a_{na}\Delta y(t-na+3) + B\Delta u(t+1)$$

$$\vdots$$

while the further step

$$\hat{y}(t+1|t) = y(t) + \hat{\Delta y}(t+1|t)$$

$$\hat{y}(t+2|t) = \hat{y}(t+1|t) + \hat{\Delta y}(t+2|t)$$

$$\vdots$$

$$\hat{y}(t+j|t) = \hat{y}(t+j-1|t) + \hat{\Delta y}(t+j|t)$$

iterates the $1/\Delta$ stage. The incremental model iteration is therefore equivalent to the incremental diophantine approach.

A.3 Proof that iterating the known part of the CARIMA model filtered by $\underline{T}_c(z^{-1})$ gives the same predictions as the modified incremental diophantine approach (see section 2.6.3).

This proof is similar to that given in the preceding section. Consider the modified incremental diophantine equation and resulting predictor

$$1 = E_j\frac{A\Delta}{T_c} + z^{-j}\frac{F_j}{T_c} \qquad (2\text{-}48)$$

$$\hat{y}(t+j|t) = E_j\frac{B\Delta}{T_c}u(t+j-1) + \frac{F_j}{T_c}y(t) \qquad (2\text{-}49)$$

APPENDIX A

These can be obtained from the corresponding equations of section A.1 by replacing A with $A\Delta/T_c$, B with $B\Delta/T_c$ and F_j with F_j/T_c. This is possible since $1/T_c$ may be expanded as an infinite series, giving polynomial equations of infinite order. This means that the diophantine predictions (2-49) may be calculated by iterating the model

$$y(t) = \frac{B\Delta}{T_c} \cdot \frac{T_c}{A\Delta} u(t-1) \qquad (A-10)$$

from current state. Fig.A.2 illustrates how the incremental model-iterative approach incorporates $T_c(z^{-1})$:

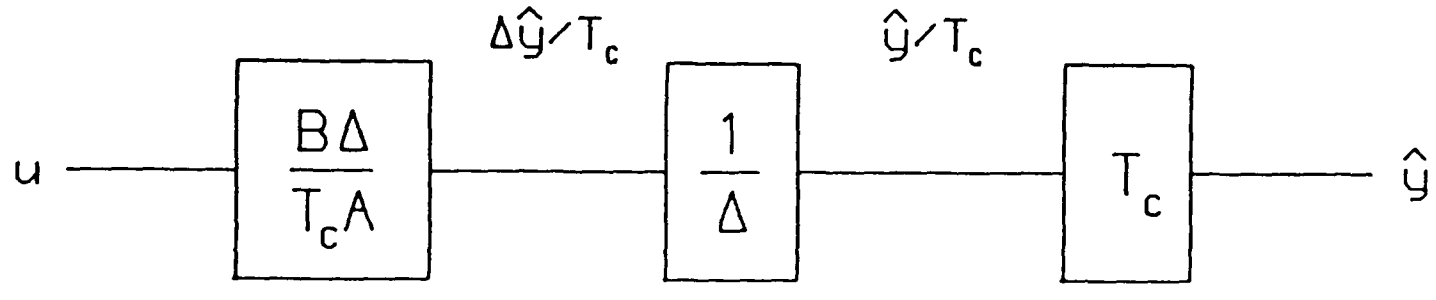


Fig.A.2: Incremental model iteration with $T_c(z^{-1})$

First $B\Delta/AT_c$ is iterated, then $1/\Delta$ and finally the predictions are 'deconvolved' by iterating T_c. Therefore, from (A-10), the incremental model-iterative and diophantine approaches are equivalent with $T_c(z^{-1})$ incorporated.

A.4 Positional 'forced' response as a Convolution Sum.

Consider the weighting-sequence model

$$y(t) = \frac{B}{A} u(t-1) = \left\{ \sum_{i=0}^{\infty} h_i z^{-i} \right\} u(t-1) \qquad (A-11)$$

obtained by expanding 1/A as an infinite series. The latter term, advanced in time by j samples, gives

$$y(t+j) = \sum_{i=0}^{\infty} h_i u(t+j-i-1) \qquad (A\text{-}12)$$

Since the predicted forced response $\{\hat{y}_d\}$ is given by iterating this model from zero state at time t (see section 2.3.4) controls before u(t) may be zeroed and the summation limits modified to

$$\hat{y}_d(t+j|t) = \sum_{i=0}^{j-1} h_i u(t+j-i-1) \qquad (2\text{-}11)$$

A.5 Incremental 'forced' response as a Convolution Sum.

The forced response in terms of control increments is given by iterating the model (see section A.2)

$$y(t+j) = \frac{B}{A\Delta} \Delta u(t+j-1) \qquad (A\text{-}13)$$

from zero state at time t. By comparison with the preceding section the convolution sum must be

$$\hat{y}_d(t+j|t) = \sum_{i=0}^{j-1} g_i \Delta u(t+j-i-1) \qquad (2\text{-}29)$$

where g_i are the ordinates of the unit pulse response of $B/A\Delta$ which is, of course, the unit step response of B/A.

APPENDIX A

A.6 Pre-specified set-points as an anti-causal set-point pre-filter.

Consider the incremental GPC control calculation

$$\Delta u(t) = \bar{g}^T(w-\hat{y}_f) \qquad (2\text{-}32)$$

where $\bar{g}^T$ is the top row of the matrix $(G_1{}^TG_1 + \lambda I)^{-1}G_1{}^T$. Multiplying this out, for normal and pre-specified set-points, gives

NORM: $\Delta u(t) = \bar{g}_0 w(t+1) + \bar{g}_1 w(t+1) + \ldots + \bar{g}_{NY-1} w(t+1) - \bar{g}^T\hat{y}_f$

PRE: $\Delta u(t) = \bar{g}_0 w(t+1) + \bar{g}_1 w(t+2) + \ldots + \bar{g}_{NY-1} w(t+NY) - \bar{g}^T\hat{y}_f$

Fig.A.3 shows how the latter expression can be interpreted as a set-point prefilter external to the loop. The term $\bar{g}^T\hat{y}_f$ represents the effect of feedback in the loop. At low frequencies the frequency response of the anti-causal prefilter $\bar{g}(e^{j\omega h}) \simeq \Sigma\bar{g}_i$ but, as the frequency content of the set-point signal increases, the two approaches diverge with increasing phase advance from the pre-specified set-points.

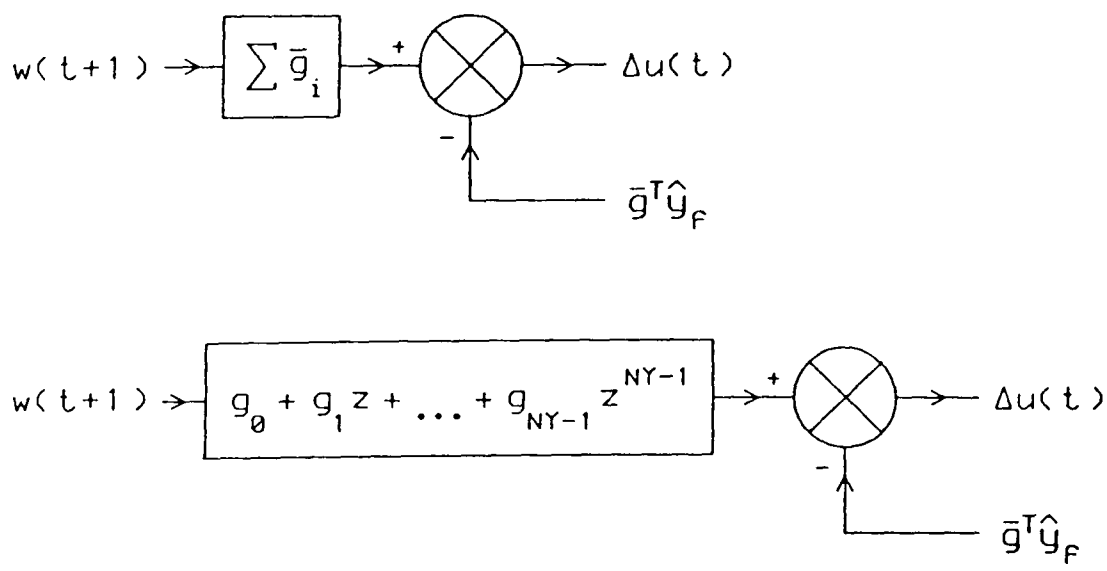


Fig.A.3: Pre-specified vs. Normal set-points

A.7 Multiple Functional Relations

Consider the case where two functional relations between the parameters to be estimated are known ie.

$$f_1(\hat{\theta}_1, \hat{\theta}_2, \ \ldots\ ) = f_2(\hat{\theta}_1, \hat{\theta}_2, \ \ldots\ ) = 0 \qquad \text{(A-14)}$$

The minimisation of the least squares cost (3-7) subject to these constraints may, following section 3.2, be obtained by solving the (approximate) Lagrange multiplier equations

$$\begin{aligned} \frac{\partial J}{\partial \hat{\theta}_1} + \lambda_1 \left[\frac{\partial f_1}{\partial \hat{\theta}_1}\right]_{\hat{\theta}_{ls}} + \lambda_2 \left[\frac{\partial f_2}{\partial \hat{\theta}_1}\right]_{\hat{\theta}_{ls}} &= \frac{\partial J}{\partial \hat{\theta}_1} + \lambda_1 \hat{n}_{11} + \lambda_2 \hat{n}_{12} = 0 \qquad \text{(A-15)} \\ \frac{\partial J}{\partial \hat{\theta}_2} + \lambda_1 \left[\frac{\partial f_1}{\partial \hat{\theta}_2}\right]_{\hat{\theta}_{ls}} + \lambda_2 \left[\frac{\partial f_2}{\partial \hat{\theta}_2}\right]_{\hat{\theta}_{ls}} &= \frac{\partial J}{\partial \hat{\theta}_2} + \lambda_1 \hat{n}_{21} + \lambda_2 \hat{n}_{22} = 0 \end{aligned}$$

which in matrix form becomes

$$-X^T Y + X^T X \hat{\theta} + \lambda_1 \hat{n}_1 + \lambda_2 \hat{n}_2 = 0 \qquad \text{(A-16)}$$

Splitting $\hat{\theta} = \hat{\theta}_{ls} + \delta\hat{\theta}$ gives

$$\delta\hat{\theta} = -\lambda_1 [X^T X]^{-1} \hat{n}_1 - \lambda_2 [X^T X]^{-1} \hat{n}_2 \qquad \text{(A-17)}$$

To solve for the multipliers λ_1 and λ_2 consider the <u>two</u> Taylor expansions

$$\begin{aligned} f_1(\hat{\theta}_{ls} + \delta\hat{\theta}) &\simeq f_1(\hat{\theta}_{ls}) + \hat{n}_1^T \delta\hat{\theta} = 0 \qquad \text{(A-18)} \\ f_2(\hat{\theta}_{ls} + \delta\hat{\theta}) &\simeq f_2(\hat{\theta}_{ls}) + \hat{n}_2^T \delta\hat{\theta} = 0 \end{aligned}$$

which, together with (A-17), give the simultaneous equations

$$f_1(\hat{\theta}_{ls}) = \lambda_1 \hat{n}_1^T [X^T X]^{-1} \hat{n}_1 + \lambda_2 \hat{n}_1^T [X^T X]^{-1} \hat{n}_2 \qquad \text{(A-19)}$$
$$f_2(\hat{\theta}_{ls}) = \lambda_1 \hat{n}_2^T [X^T X]^{-1} \hat{n}_1 + \lambda_2 \hat{n}_2^T [X^T X]^{-1} \hat{n}_2$$

The solutions for λ_1 and λ_2 may be substituted into (A-17) to obtain the perturbation $\hat{\delta\theta}$ from the least squares estimates $\hat{\theta}_{ls}$ required to satisfy the functional relations (A-14).

APPENDIX B

NOMENCLATURE AND SYMBOLS

B.1 General Nomenclature

$A(.)$	Polynomial in terms of an argument '.'
$\hat{A}(.)$	Estimated polynomial
$A(.)\|_{.=\beta}$	Value of the polynomial when its argument is set to β
δA	Order of a polynomial, the highest power of its argument
na	Equivalent to δA
a_j	j^{th} coefficient of a polynomial $A(.)$
$\hat{a}_j$	j^{th} estimated coefficient of a polynomial $A(.)$
$B(.)/A(.)$	Transfer-function as a ratio of polynomials
$A(1)$	Steady-state gain of $A(.)$
$a(t)$	Value of a continuous or sampled signal at time t
$\{a(t)\}$	Sequence of sampled values
$a(t-j)$	Value of $a(t)$ at the jth previous sample
$\bar{a}$	Mean value of a signal $a(t)$
$a'(t)$	Filtered version of $a(t)$
$\hat{a}(t+j\|t)$	Prediction of $a(t)$, in j samples time, given data available at time t
$\mathbf{a}$	General vector
$\mathbf{a}^{SS}$, $\mathbf{a}_{SS}$	Steady-state vector

B.2 Principal Symbols

$A(z^{-1})$, $B(z^{-1})$ $C(z^{-1})$	Polynomials of the CARMA and CARIMA process model associated with $y(t)$, $u(t)$ and $\xi(t)$
$E_j(z^{-1})$	Polynomial in the j-step-ahead Diophantine equation, order j-1
$F_j(z^{-1})$	Polynomial in the j-step-ahead Diophantine equation

APPENDIX B

$P(z^{-1})$, P_n/P_d	Phase advance design-filter
$Q(z^{-1})$	Control weighting design-filter
$T_e(z^{-1})$	Estimator signal conditioning design-filter
$T_c(z^{-1})$	Controller signal conditioning design-filter
D	Diagonal factor of P(t)
dcy(t)	Additive disturbance on the process output
e(t), $e_T(t)$	Modelling errors
e(t+j|t), ε(t+j|t)	j-step-ahead prediction errors
$\hat{\mathbf{e}}$	Vector of prediction errors
f, f()	Functional relation between the elements of θ
g	Dynamic gain of the process g≡B(1)/A(1)
g_i	Ordinates of the unit step response of B/A or PB/A
$\mathbf{G}$, $\mathbf{G}_1$	Matrices whose elements are g_i
$\bar{\mathbf{g}}$	Top row of the matrix $(\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I})^{-1}\mathbf{A}^T$ where A= $\mathbf{H}$, $\mathbf{H}_1$, $\mathbf{G}$ or $\mathbf{G}_1$
$\bar{g}_i$	Element of $\bar{\mathbf{g}}$
G(s)	Laplace transform transfer-function
h	Sample rate of a digital controller
H	Subscript to denote hexadecimal numbers
h_i	Ordinates of the unit pulse response of B/A or PB/A
$\mathbf{H}$, $\mathbf{H}_1$	Matrices whose elements are h_i
$\mathbf{I}$	Identity matrix
J	Value of a quadratic cost function
k	Process time delay in samples, not including ZOH
K(t)	'Kalman' gain in the recursive least squares estimator
m, n	Order of a polynomial
$\hat{n}_i$	Approximate partial derivative of f()
$\hat{\mathbf{n}}$	Vector whose elements are $\hat{n}_i$
N	Dimension of a vector or matrix
NU	Control horizon
NY	Final prediction horizon

N_1	Initial prediction horizon
p_i	Pole location
P(t)	'Covariance' matrix in the recursive least squares estimator
Q	Linear forgetting matrix
r(t)	Set-point trajectory
s	Laplace transform operator, $s \equiv d/dt$
U	Upper triangular factor of P(t)
u(t)	Control signal input to the process
$\tilde{\mathbf{u}}$, $\tilde{\{u\}}$	Vector or sequence of future controls
$\tilde{\delta\mathbf{u}}$	Vector of future control increments
w(t)	Set-point for the process measured variable
$\mathbf{w}$	Vector of future set-points
x(t)	General disturbance signal
X	Matrix whose rows are past regressors $\phi(t)$
y(t)	Measured variable of the process
y_f, $\{y_f\}$	Predicted 'free' reponse of the process
$\hat{y}_i$, $\hat{\{y_i\}}$	Predicted 'forced' reponse of the process
$\hat{\mathbf{y}}$, $\hat{\{y\}}$	Predicted combined reponse of the process
Y	Vector whose elements are past measurements y(t)
z	Forward-shift operator, $za(t) = a(t+1)$
α	Tuning-knob for the simplified structure of $P(z^{-1})$
β	Tuning-knob for the simplified structure of $T(z^{-1})$
$\beta(t)$	Forgetting factor in the RLS estimator
δ	Discrete-time operator $\delta \equiv (z-1)/h$
Δ, $\Delta(z^{-1})$	Differencing operator $\Delta \equiv (1-z^{-1})$
ε	A 'small' number
$\varepsilon(t)$	A-priori estimation error
ζ	Damping factor of a second-order pole pair
$\eta(t)$	Unmeasurable disturbance
θ, $\hat{\theta}(t)$	Process parameter vector, and its estimate

APPENDIX B

$\hat{\theta}_{ls}$	Recursive least squares estimates of θ
$\hat{\delta\theta}$	Perturbation from $\hat{\theta}_{ls}$ required to satisfy a constraint
κ	Fixed forgetting factor
λ, λ_{rel}	Control-weighting and relative weighting for $Q(z^{-1})$, Lagrange multipliers
$\Lambda(i)$	Error-weight in the RLS cost function
$\xi(t)$	Zero mean uncorrelated random variable
Σ_0	Tuning-knob for Ydstie's variable forgetting factor
τ	Fractional time delay
$\phi(t)$	Regressor vector in the RLS estimator
$\psi(t)$	Pseudo-output of the process, $\psi(t)=P(z^{-1})y(t)$
ω	Resonant frequency of a second-order pole-pair or simply frequency (rad/s)

APPENDIX C

REFERENCES

Al-Assaf,Y., (1987), "Self-tuning Control: Theory and Applications": D. Phil. Thesis, Oxford University, in preparation.

Allworth,S.T., (1981), "Introduction to Real-time Software Design": The Macmillan Press.

Astrom,K.J., (1970), "Introduction to Stochastic Control Theory": Academic Press.

Astrom,K.J., (1983), "Theory and Applications of Adaptive Control - A Survey": Automatica, Vol.19, No.5, pp.471-486.

Astrom,K.J., (1985), "Autotuning, Adaption and Expert Control": ACC, Boston.

Astrom,K.J., (1986), "A Comparison between Robust and Adaptive control of uncertain systems": Procs. 2nd IFAC Workshop on Adaptive Systems in Control and Signal Processing, pp.37-42.

Astrom,K.J. and Wittenmark,B., (1973), "On Self-tuning Regulators": Automatica, Vol.9, No.2, pp.185-199.

Astrom,K.J., Hagander,P. and Sternby,J., (1984), "Zeros of sampled data systems": Automatica, Vol.20, No.1, pp.31-39.

Balas,M.J., (1978), "Feedback Control of Flexible Systems": IEEE Trans., Vol.AC-23, No.4, pp.673-679.

Belanger,P.R., (1983), "On Type 1 Systems and the Clarke-Gawthrop regulator": Automatica, Vol.19, No.1, pp.91-94.

Bierman,G.J., (1977), "Factorisation methods for discrete system estimation": Academic Press.

Bohm,J., (1985), "LQ Self-tuners with Signal Level Constraints": Procs. 7th IFAC Symposium on Identification and System Parameter Estimation, York, UK, pp.131-136.

Cameron,F. and Seborg,D.E., (1983), "A Self-tuning Controller with a PID structure": Int. J. Control, Vol.38, No.2, pp.401-417.

Cannon,R.H. and Schmitz,E., (1984), "Initial Experiments on the End-Point Control of a Flexible One-Link Robot": IJRR, Vol.3, No.3.

Clarke,D.W., (1981), "Self-tuning and Adaptive Control: Introduction to Self-tuning Controllers": (ed. Harris and Billings) Peter Peregrinus.

Clarke,D.W., (1982), "The Application of Self-tuning Control": Trans.Inst.M.C., Vol.5, No.2, pp.59-69.

Clarke,D.W., (1983), "PID Algorithms and their Computer Implementation": OUEL Internal Report No.1482/83.

APPENDIX C

Clarke,D.W. and Gawthrop,P.J., (1975), "Self-tuning Controller": Proc. IEE, Vol.122, No.9, pp.929-934.

Clarke,D.W. and Gawthrop,P.J., (1979), "Self-tuning Control": Proc. IEE, Vol.126, No.6, pp.633-640.

Clarke,D.W., Hodgson,A.J. and Tuffs,P.S., (1983), "The Offset Problem and k-incremental predictors in Self-tuning Control": Proc. IEE, Vol.130, Pt.D., No.5, pp.217-225.

Clarke,D.W., Mohtadi,C. and Tuffs,P.S., (1984), "Generalised Predictive Control. Part 1: The basic algorithm. Part 2: Extensions and interpretations": OUEL Reports 1555/84 & 1557/84.

Clarke,D.W., Tuffs,P.S. and Mohtadi,C., (1985), "Self-tuning Control of a Difficult Process": 7th IFAC Symposium on Identification and System Parameter Estimation, York.

Clarke.D.W., Kanjilal,P.P. and Mohtadi,C., (1985a), "A Generalised LQG approach to Self-tuning Control: Part 1 - Aspects of design": Int.J.Control, Vol.41, No.6, pp.1509-1523.

Clarke,D.W., Mohtadi,C. and Tuffs,P.S., (1987), "Generalised Predictive Control. Part 1: The basic algorithm. Part 2: Extensions and interpretations": Automatica, Vol.23, No.2, pp.137-160.

Clarke,D.W. and Zhang,L., (1987a), "Long-range Predictive Control using Weighting-sequence Models": Proc. IEE, Vol.134, Pt.D, No.3, pp.187-195.

Cutler,C.R. and Ramaker,B.L., (1980), "Dynamic Matrix Control - a computer control algorithm": JACC, San Fransisco.

Daniel,R.W., Irving,M.A., Fraser,A.R. and Lambert,M.R., (1987), "The Control of Compliant Manipulator Arms": Proceedings of the International Symposium on Robotics Research, August, Santa Cruz.

Dasgupta,S., Anderson,B.D. and Kaye,R.J., (1984), "Identification of Economically Parameterised Systems": IFAC 9th World Congress, Budapest, Hungary.

De Keyser,R.M. and Van Cauwenberghe,A.R., (1981), "A self-tuning multistep predictor application": Automatica, Vol.17, No.1, pp.167-174.

De Keyser,R.M. and Van Cawenberghe,A.R., (1985), "Extended Prediction Self-Adaptive Control": Procs. 7th IFAC Symposium on Identification and System Parameter Estimation, York, UK, pp.1255-1260.

De Keyser,R.M., De Coen,L. and Verdiere,P., (1985), "Multi-microprocessor Simulation of a Cutter Suction Dredging Ship": Procs. 7th IFAC Conf. on Digital Computer Applications to Process Control, Vienna, Austria.

Edmunds,J.M., (1984), "Identifying Sampled Data Systems using Difference Operator Transfer Functions": Control Systems Centre, UMIST, Report 627.

Farsi,M. and Karam,K.Z., (1987), "Self-tuning Control of a Robot Manipulator": 4th IEE Workshop on Self-tuning and Adaptive Control, pp.7/1-7/10.

Fortescue,T.R., Kershenbaum,L.S. and Ydstie,B.E., (1981), "Implementation of Self-Tuning Regulators with Variable Forgetting Factors": Automatica, Vol.17, No.6, pp.831-835.

Francis,B.A. and Wonham,W.M., (1976), "The Internal Model Principle of Control Theory": Automatica, Vol.12, No.5, pp.457-467.

Fraser,A.R., (1987), "Perturbation techniques in the Dynamics and Control of Flexible Manipulators": D. Phil. Thesis, Oxford University, in preparation.

Gawthrop,P.J., (1977), "Some Interpretations of the Self-tuning Controller": Proc. IEE, Vol.124, No.10, pp.889-894.

Gawthrop,P.J., (1980), "Hybrid Self-tuning Control": Proc. IEE, Vol.127, Pt.D, No.5, pp.229-236.

Gawthrop,P.J., (1987), "Continuous-Time Self-tuning Control: Vol.1 - Design": Research Studies Press, John Wiley & Sons.

Goodwin,G.C. and Sin,K.S., (1984), "Adaptive Filtering, Prediction and Control": Prentice-Hall.

Goodwin,G.C., Leal,R.L., Mayne,D.Q. and Middleton.R.H., (1986), "Rapprochement between Continuous and Discrete Model Reference Adaptive Control": Automatica, Vol.22, No.2, pp.199-209.

Hagglund,T., (1983), "New estimation techniques for adaptive control": Report LUTFD2/(TFRT-7254)/1-038/(1983) Lund Institute of Technology.

Hodgson,A.J., (1982), "Problems of Integrity in Applications of Adaptive Controllers": D. Phil. Thesis, Oxford University and OUEL report 1436/82.

Holt,R.C., Graham,G.S., Lazowska,E.D. and Scott,M.A., (1978), "Structured Concurrent Programming with Operating Systems Applications": Addison-Wesley.

Jacobs,O.L., (1981), "Self-tuning and Adaptive Control: Introduction to Adaptive Control": (ed. Harris and Billings) Peter Peregrinus.

Jota,F.G., (1987), "The application of self-tuning control technique to a multi-variable process": D. Phil. Thesis, Oxford University.

Jury,E.I., (1964), "Theory and Application of the z-transform Method": Wiley, New York.

Kalman,R.E., (1958), "Design of a Self-optimising Control System": Trans. ASME, Vol.80, pp.468-478.

Kanoh,H., Tzafestas,S., Lee,H.G. and Kalat,J., (1986), "Modelling and Control of Flexible Robot Arms": IEEE Proc. 25th CDC, Athens, Greece.

Khorasani,K. and Kokotovic,P.V., (1985), "Feedback Linearisation of a flexible manipulator near its rigid body manifold", Systems & Control Letters, Vol.6, No.3, pp.187-192.

Kulhavy,R., (1986), "Directional tracking of Regression-type Model Parameters": 2nd IFAC Workshop on Adaptive Systems in Control and Signal Processing, Lund, Sweden.

Kwon,W.H. and Pearson,A.E., (1978), "On Feedback Stabilisation of time-varying discrete systems": IEEE Trans. Autom. Control, Vol.AC-23, No.3, pp.479-481.

Lam,K.P., (1980), "Implicit and Explicit Self-tuning Controllers": D. Phil. Thesis, Oxford University and OUEL report 1134/80.

Lambert,E.P., (1987), "The Industrial Application of Long Range Predictive Control": D. Phil. Thesis, Oxford University, in preparation.

Lelic,M.A. and Zarrop,M.B., (1986), "A Generalised Pole-placement Self-tuning Controller - Part 1: The Basic Algorithm": UMIST Control Systems Centre Report No. 657.

Lelic,M.A. and Wellstead,P.E., (1986), "A Generalised Pole-placement Self-Tuning Controller - Part 2: An Application to Robot Manipulator Control": UMIST Control Systems Centre Report No. 658.

Ljung,L. and Soderstrom,T., (1983), "Theory and Practice of Recursive Identification": MIT Press.

Meckl,P.H. and Seering,W.P., (1986), "Feedforward Control Techniques to achieve Fast Settling Time in Robots": ACC, Seattle, Washington.

Meldrum,D.R. and Balas,M.J, (1986), "Application of Model Reference Adaptive Control to a Flexible Remote Manipulator Arm": ACC, Seattle, Washington.

Middleton,R.H. and Goodwin, G.C., (1986), "Improved Finite Word Length Characteristics in Digital Control using Delta Operators": IEEE Trans.AC, Vol.AC-31, No.11.

Moden,P.E. and Soderstrom,T., (1978), "On the Achievable Accuracy in Stochastic Control": IEEE Proc. 25th CDC, Athens, Greece.

Mohtadi,C., (1986), "Studies in Advanced Self-tuning Algorithms": D.Phil. Thesis, Oxford University.

Mohtadi, C. and Clarke,D.W., (1986), "Generalised Predictive Control, LQ, or Pole-placement: A unified approach": IEEE Proc. 25th CDC, Athens, Greece.

Nemir,D., Koivo,A.J. and Kashyap,R.L., (1986), "Control of Gripper Position of a Compliant Link using Strain Gauge Measurements": IEEE Proc. 25th CDC, Athens, Greece.

Nguyen,P.K., Ravindran,R., Carr,R. and Gossain,D.M., (1982), "Structural Flexibility of the Shuttle Remote Manipulator System Mechanical Arm": Proc. Guidance and Control Conf., AIAA paper no.82-1536, August.

Ortega,R., Praly,L. and Landau,I.D, (1985), "Robustness of Discrete-Time Direct Adaptive Controllers": IEEE Transactions on Automatic Control, Vol.AC-30, No.12.

Peterka,V., (1970), "Adaptive Digital Regulation of Noisy Systems": IFAC Symposium on Identification and Process Parameter Estimation, Prague.

Peterka,V., (1984), "Predictor-based Self-tuning Control": Automatica, Vol.20, No.1, pp.39-50.

Peterka,V., (1986), "Control of Uncertain Processes: Applied Theory and Algorithms": Kybernetika, Vol.22, pp.1-102.

Peterka,V. and Astrom,K.J., (1973), "Control of Multivariable Systems with unknown but constant parameters": IFAC Symposium on Identification and System Parameter Estimation, The Hague, Netherlands, pp.535-544.

Ragazzini,J.R. and Franklin,G.F., (1958), "Sampled-data Control Systems": McGraw-Hill.

Richalet,J., Rault,A., Tesud,J.L. and Papon,J., (1978), "Model Predictive Heuristic Control: Applications to Industrial Processes": Automatica, Vol.14, No.5, pp.413-428.

Rohrs,C.E., Athans,M., Valavani,L. and Stein,G., (1984), "Some design guidelines of discrete-time adaptive controllers": Automatica, Vol.20, No.5, pp.653-660.

Seborg,D.E., Edgar,T.F. and Shah,S.L., (1986), "Adaptive Control Strategies for Process Control: A Survey": AIChE Journal, June.

Siciliano,B., Yuan,B.S. and Book,W.J., (1986), "Model Reference Control of a One Link Flexible Arm": IEEE Proc. 25th CDC, Athens, Greece.

Stephenson,G., (1973), "Mathematical Methods for Science Students": Longman, London and New York.

Sweet,L.M. and Good,M.C., (1985), "Redefinition of the Robot Motion-Control Problem": IEEE Control Systems Magazine, August, pp.18-25.

Tham,M.T., Morris,A.J. and Montague,G.A., (1987), "Self-tuning Process Control: a Review of some Algorithms and their Application": Proceedings of ACC, Minneapolis, pp.996-1001.

Truckenbrodt,A., (1981), "Truncation problems in the Dynamics and Control of Flexible Mechanical Systems": IFAC World Congress, Kyoto.

Tsang,T.C., (1987), "Self-tuning Control": 1st Year Transfer Report, Oxford University.

Tuffs,P.S., (1984), "Self-tuning Control: Algorithms and Applications": D.Phil. Thesis, Oxford University and OUEL report 1567/85.

Tuffs,P.S. and Clarke,D.W., (1985), "Self-tuning Control of Offset: a Unified Approach": Proc. IEE, Vol.132, Pt.D, No.3, pp.100-110.

Van den Bossche,E., Dugard,L. and Landau,I.D., (1986), "Modelling and Identification of a Flexible Arm": ACC, Seattle, Washington.

Wahab,W., (1984), "Model Order Determination and Parameter Identification of a Linear Model of a Robot Manipulator Arm": UMIST Control Systems Centre Report No.602.

Wang,W.J., Lu,S.S. and Hsu,C.F., (1986), "Output Feedback Control of a Flexible Robot Arm": IEEE Procs. 25th CDC, Athens, Greece.

Wellstead,P.E., (1978), "An Instrumental Product Moment Test for Model Order Estimation": Automatica, Vol.14, No.1, pp.89-92.

Wellstead,P.E., Edmunds,J.M., Prager,D. and Zanker,P., (1979), "Self-tuning pole/zero assignment regulators": Int. J. Control, Vol.30, No.1, pp.1-26.

Wellstead,P.E. and Sanoff,S.P., (1981), "Extended Self-Tuning Algorithm": Int. J. Control, Vol.34, No.3, pp.433-455.

Wellstead,P.E. and Zanker,P., (1982), "Techniques of Self-tuning": Optimal Control Applications & Methods, Vol.3, pp.305-322.

Wittenmark,B., (1973), "A Self-tuning Regulator": Report 7311, Division of Automatic Control, Lund Institute of Technology.

Wittenmark,B., (1975), "Stochastic adaptive control methods: a Survey": Int.J.Control, Vol.21, No.5, pp.705-730.

Wittenmark,B. and Astrom,K.J., (1984), "Practical Issues in the Implementation of Adaptive Control": Automatica, Vol.20, No.5, pp.595-605.

Ydstie,B.E., (1984), "Extended Horizon Adaptive Control": IFAC 9th World Congress, Budapest, Hungary.

Ydstie,B.E., Kershenbaum,L.S. and Sargent,R.W.H, (1985), "Theory and Application of an Extended Horizon Self-Tuning Controller": AIChE Journal, Vol. 31, pp.1771.

Zames,G., (1981), "Feedback and Optimal Sensitivity: Model Reference transformations, Multiplicative seminorms, and Approximate Inverses": IEEE Trans. Auto. Cont., Vol.AC-26, pp.301.