1996

Adaptive Non-linear Predictive Control

Edward Katende

ADAPTIVE NON-LINEAR PREDICTIVE CONTROL

by

Edward Katende

Department of Chemical and Biochemical Engineering

Faculty of Engineering Science

Submitted in partial fulfilment

of the requirements for the degree of

Doctor of Philosophy

Faculty of Graduate Studies

The University of Western Ontario

London, Ontario

April 1996

ISBN 0-612-15061-5

ABSTRACT

Most predictive control algorithms, including the Generalized Predictive Control (GPC) (Clarke *et al.* ,1987) are based on linear dynamics. Many processes are severely non-linear and would require high order linear approximations. Another approach, which is presented here, is to extend the basic adaptive GPC algorithm to a non-linear form. This provides a non-linear predictive controller which is shown to be very effective in the control of processes with non-linearities that can be suitably modelled using general Volterra, Hammerstein and bilinear models. In developing this algorithm, the process dynamics are not restricted to a particular order as is the case with the current non-linear adaptive algorithms. Simulations are presented using a number of examples and the steady state properties are discussed.

The Non-Linear Generalized Predictive Control (NLGPC) algorithm is tested on a non-linear batch reactor system by carrying out a number of experiments and comparing its performance with other control strategies. The NLGPC is shown to outperform the constrained Self-Tuning PID (STPID) controller by Katende and Jutan (1993) and the Generalized Minimum Variance (GMV) controller by Clarke and Gawthrop (1975). It is also shown to have better performance than the well known GPC algorithm by Clarke *et al.* (1987). The advantage of the NLGPC over the other controllers is attributed to its adaptive nature and use of non-linear process models in its design.

Keywords: adaptive control, non-linear control, predictive control, non-linear processes

ACKNOWLEDGEMENTS

The author would like to express his grateful thanks to a number of individuals who have collectively ensured the completion of this work. Foremost among them is Dr. A. Jutan who supervised the research and has been a source ot invaluable guidance and encouragement.

The author is grateful to all his fellow graduate students, in particular the process control group, for their support, friendship and above all, for their special contribution to a pleasant working environment.

Thanks are extended to all members of the Faculty of Engineering Science at the University of Western Ontario who gave freely of their time and expertise.

Finally, the author would like to thank the Commonwealth Scholarship and Fellowship Plan for the grant of a Scholarship to pursue his PhD program.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

NOMENCLATURE

All notations in this text are also defined in appropriate places.

A	Polynomial in a Process Model
B	Polynomial in a Process Model
C	Polynomial in a Process Model
d	Dead Time
D	Polynomial in a Process Model
E	Expected Value
E_i	Polynomial at i Step Ahead
F_i	Polynomial at i Step Ahead
g	Self-tuning PID Parameters
G_i	Polynomial at i Step Ahead
H	Transfer Function
H_i	Polynomial at i Step Ahead
i	Time Instant
I	Cost Function or Objective Function or Criterion Function
J	Cost Function or Objective Function or Criterion Function
k	Time Variable
K	Kalman Gain

Kc	Proportional Gain
n	Process Noise
N	Process Noise
N1 or N_1	Minimum Cost Horizon
N2 or N_2	Maximum Cost Horizon
NU or Nu	Control Horizon
P	Covariance Matrix or Model Following Polynomial
q	Back Shift Operator
Q	Polynomial
R	Polynomial
S	Polynomial
t	Time or Sampling Instant
T	Observer or Filtering Polynomial
Td	Derivative Time
Ti	Integral Time
x	'Pseudo' Input
X	Information Matrix for RLS
y	Process Output
v'	Estimated or Predicted Process Output
u	Process Input
w	Setpoint or Reference Trajectory
z	Back Shift Operator

α,γ	Self-Tuning Controller Parameters (GMV)
ε	White Noise
Δ	Integrator
ϵ	Prediction Error
θ	Parameter Vector for RLS
λ	Control Weighting for NLGPC, GPC or Forgetting Factor for STPID
ξ	Constraint for STPID or White Noise in CARIMA Model
σ	Forgetting Factor

ACRONYMS

CRHPC	Constrained Receding Horizon Predictive Control
CARIMA	Controlled Autoregressive Integrated Moving Average
CARMA	Controlled Autoregressive Moving Average
DMC	Dynamic Matrix control
EHAC	Extended Horizon Adaptive Control
EPSAC	Extended Predictive Self-Adaptive Control
FIR	Finite Impulse Response
GMV	Generalized Minimum Variance
GPC	Generalized Predictive Control
LP	Linear Programming
LQ	Linear Quadratic
LRPC	Long Range Predictive Control

MAC	Model Algorithmic Control
MIMO	Multi-Input Multi-Output
MPC	Model Predictive Control
MPHC	Model Predictive Heuristic Control
MUSMAR	Multi-Step Multivariable Adaptive Control
NLGPC	Non-Linear Generalized Predictive Control
NLP	Non-Linear Programming
PFC	Predictive Functional Control
PID	Proportional Integral Derivative
SGPC	Stable Generalized Predictive Control
SISO	Single-Input Single-Outpt
STPID	Self-Tuning PID

CHAPTER 1

INTRODUCTION

1.1 GENERAL

During the past three decades, major advances in control theory have occurred as affordable computer control systems have become readily available. As a group, adaptive controllers have focused primarily on linear process models. From a parameter update point of view, it is advisable to have the parameters in a linear space. However, from the process dynamics view point, linear dynamics can present a severe limitation for highly non-linear processes. Adaptive control applications based on linear dynamics are very numerous and the basic engineering theory and applications have been presented in literature.

For a large number of systems a linearizing approach is acceptable. However, for some non-linear systems linearizing can result in poor performance. Some of these highly non-linear systems include pH control in some chemical or biochemical processes, or thickness control for a rolling mill. In order to obtain improved control over these systems, while maintaining the benefits of self-tuning control techniques, it is necessary to take account of the non-linear dynamics in an appropriate manner.

Atherton (1975) and Cook (1986) indicate that because of the large number of different types of non-linearity which can occur in practice, extending a basic linear control scheme to account for all possibilities is unrealistic. A sensible way of tackling the general problem is to employ a framework within which a large number of non-linear processes can be adequately approximated. For self-tuning such a framework is provided by the Volterra and Hammerstein models (Agarwal and Seborg, 1987).

Agarwal and Seborg proposed two self-tuning control strategies for non-linear control problems. Their strategies are applicable to a broad class of non-linear single-input, single-output systems which can include arbitrary nonlinear functions of the most recent input.

The aim of this research is to explore adaptive non-linear predictive control strategies which have the attractive features of a generalized predictive controller, but without the restriction to linear dynamics. This is an advantage over the current discrete non-linear algorithms which usually restrict the process dynamics to second order. Performance is also improved by using a non-standard cost-function.

In this thesis a new Non-Linear Generalized Predictive Control (NLGPC) algorithm with a general non-linear and bilinear model structure (Lachmann, 1982 and Svoronos *et al.*, 1981) is developed. A non-standard and probably more appropriate cost function is used. In predictive controllers, minimization of a cost function yields the predictive control law, hence the choice of the criterion function is of paramount importance. The NLGPC algorithm developed here can deal with both linear and non-linear processes and is developed in the framework of the GPC algorithm.

Various linear and non-linear models are simulated and the performance of the NLGPC algorithm is compared to the standard adaptive GPC. The NLGPC performance is also evaluated using an adiabatic continuous stirred-tank reactor (CSTR). Experimental studies were carried out using predictive control methods and the results are compared to those obtained using self-tuning PID (STPID, by Katende and Jutan, 1993) and generalised minimum variance (GMV, by Clarke and Gawthrop, 1975).

1.2 OBJECTIVES

The following is a brief outline of the objectives of this research:

- Develop a new Non-linear Generalized Predictive Control (NLGPC) algorithm.
- Extend the Non-linear Generalized Predictive Control (NLGPC) to account for bilinearity in input/output signals.
- Simulate various models of different complexity under (i) Non-linear Generalized Predictive Control (NLGPC) , (ii) Generalized Predictive Control (GPC).
- Implement the these Predictive Control algorithms on a batch reactor system.
- Compare the performance of Predictive Control to Generalized Minimum Variance (GMV), and the self-tuning PID (Katende and Jutan, 1993) algorithms.

1.3 OVERVIEW

The thesis is broken into three major sections. In chapter two, the literature is reviewed, chapters three, four and five give the theoretical developments, and chapters six and seven discuss both the simulation and experimental results. The following is a brief outline of the thesis:

Chapter 2

In this chapter, progress in the theory of adaptive/model based predictive control is reviewed. Various approaches are discussed with particular emphasis on model predictive control and non-linear predictive control. The need for a new NLGPC algorithm is justified. The capabilities of the existing adaptive/predictive controllers are also discussed.

Chapter 3

Theory of GPC is discussed in this chapter. The main algorithm is derived as given in literature. Its features are also discussed and illustrated in detail.

Chapter 4

In this chapter, the new NLGPC algorithm is proposed. A second order NLGPC is also presented. Detailed discussion of the algorithm is given. Steady state performance of the NLGPC as opposed to that of the GPC are also discussed in detail.

Chapter 5

The performance of a NLGPC with penalty on the pseudo-input as opposed to that of the NLGPC with penalty on the actual input are compared. The need for a modified cost function is justified.

Chapter 6

Simulations of various models under NLGPC and GPC are shown in this chapter. The performance of the two algorithms are compared and discussed in detail.

Chapter 7

In this chapter, experimental studies of the Predictive Control, self-tuning PID (by Katende and Jutan, 1993) and GMV (Clarke and Gawthrop, 1975) are presented.

Chapter 8

General conclusions are made in this chapter. Recommendations for future studies are also given.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

Research on adaptive control started in the early 1950s. It was initiated by the poor performance of ordinary, constant gain, linear feedback control systems when used to control processes with nonlinear dynamic behaviour. Some of the early work done is discussed by Gregory (1959) and Mishkin and Braun (1961).

In the 1960s there were many contributions to control theory, which were important for the development of adaptive control. State space and stability theory were introduced. There were also important results in stochastic control theory. Dynamic programming, introduced by Bellman (1957, 1961) and dual control theory introduced by Feldbaum (1960a,b, 1961a,b, 1965), increased the understanding of adaptive processes. Fundamental contributions were also made by Tsypkin (1971), and major developments in system identification and parameter estimation by Åström and Eykhoff (1971).

Active research on adaptive control theory started again in the 1970s. The progress in control theory during the previous two decades has contributed to an improved understanding of adaptive control. Much of the theory on adaptive control of chemical process systems was developed by Åström and Wittenmark (1973), Wittenmark (1973), Åström (1974), and Clarke and Gawthrop (1975). Few industrial applications were reported initially. However, considerable progress has been made since then and a number of industrial applications have been reported. For example, on paper machines given in Cegrell and Hedquist (1975), and Wittenmark (1974), on adaptive autopilot for ships given in Åström and Källström (1980) and many others.

The most recent work and some of the most powerful and practical adaptive control systems belong to the class of Long Range Predictive Control (LRPC) systems developed by researchers such as Peterka (1984), Ydstie (1984), DeKeyser and Van Cauwenberge (1982), Clarke *et al* (1987a,b), Lee and Sullivan (1988), Favier *et al* (1988), Soeterboek (1992), Lee *et al* (1992) and Scattolini and Schiavoni (1995). There are a large number of excellent review articles covering the field of adaptive control, e.g. Seborg *et al.* (1986), Åström (1987), De Keyser *et al.*(1988) and Ljung and Gunnarson (1990), plus excellent introductory textbooks such as Åström and Wittenmark (1989) and Ljung (1987).

As a trend, most of the adaptive control literature has focused primarily on linear process models. Adaptive control applications based on linear dynamics have been presented in, for example, Harris and Billings (1985) and Warwick (1988).

As indicated by Atherton (1975) and Cook (1986) that for processes with certain types of non-linearity, non-linear control schemes should be considered. Agarwal and Seborg (1987) use Volterra and Hammerstein models to develop a self-tuning controller. These models provide a framework within which a large number of non-linear processes can be adequately approximated.

This chapter provides an overview of model based predictive control and the relevant non-linear (predictive/adaptive) control strategies.

2.2 MODEL PREDICTIVE CONTROL (MPC)

The earliest MPC design was proposed by Dawkins and Briggs (1965), though IDCOM (IDentification/COMmand) by Richalet *et al.* (1978) was the first to be used in practice. The method, sometimes called Model Algorithmic Control (MAC), has been analyzed by Reid *et al.* (1979, 1981); extensions of IDCOM to the multivariable case is given by Martin and Van Horn (1982). Other analytical comments are given by Bruijin *et al* (1980), and Bruijin and Verbruggen (1984) who add control weighting to the IDCOM cost function.

During late 1970's various articles started to appear showing an increased interest in MPC by the industry, principally publications by Richalet *et al.* (1976, 1978) presenting Model Predictive Heuristic Control (MPHC) (later known as Model Algorithmic Control (MAC)) and those by Cutler and Ramaker (1980) presenting Dynamic Matric Control

(DMC). A dynamic process model is explicitly used in both algorithms in order to predict the effect of the future control actions at the output; these are determined by minimizing the predicted error subject to operational restrictions. An impulse response model is used in MAC whereas a step response model is used in DMC.

Other areas of interest included, developing Long Range Predictive Control (LRPC) strategies for processes formulated with input/output models. Research on long range predictive control (LRPC) was motivated as a result of self-tuning methods based on GMV approach (Clarke and Gawthrop, 1975,1979) and the pole-placement algorithm (Wellstead *et al.*,1979; Åström and Wittenmark, 1980), could not cope with a number of problems. For example, the implicit GMV self-tuner is robust against model order assumptions but can perform badly if the plant dead time varies. Also the explicit pole placement method can cope with variable dead time but not with model over-parametization.

Other LRPC designs within the main-stream of self-tuning literature which is generally based on CARMA/ARMAX models include the Extended Horizon Adaptive Control (EHAC) of Ydstie (1984) and the Extended Prediction Self Adaptive Control (EPSAC), which was refined over a series of practical applications by De Keyser and Van Cauwenberghe (1981, 1982a,b, 1985). Other related designs include LQG design (e.g. Clarke *et al.* (1985) and the General Predictive Control (GPC) design by Clarke *et al.* (1987) which is perhaps the most popular method at the moment. There are numerous other LRPC formulations amongst which are: Multistep Multivariable Adaptive Control

(MUSMAR) by Greco *et al.* (1984), Multipredictor Receding Horizon Adaptive Control (MURHAC) by Lemos and Mosca (1985), Predictive Functional Control (PFC) by Richalet *et al.* (1987) and Unified Predictive Control by Soeterboek (1992).

MPC has also been formulated in the state space context (Lee et. al, 1994). This not only allows for the use of well known theorems of the state space theory, but also facilitates their generalization to more complex cases such as systems with stochastic disturbances and noise in the measured variables. By extending the step response model and using known state space techniques, processes with integrators can also be treated. The state estimation techniques arising from stochastic optimal control can be used for predictions without additional complications (Lee *et al.*, 1994). This perspective leads to simple tuning rules for stability and robustness: the MPC controller can be interpreted to be a compensator based on a state observer and its stability, performance and robustness are determined by the poles of the observer (which can be directly fixed by adjustable parameters) and the poles of the regulator (determined by the horizons, weightings, etc.). An analysis of the inherent characteristics of all the MPC algorithms (especially of the GPC) from the point of view of the Gaussian quadratic optimal linear theory can be found in the book by Bitmead *et al.* (1990).

Garcia *et al.* (1989) provide a review of MPC history, formulation, key features, applications and references. Ricker (1989) provides a technical overview and more detailed presentations can be found in the books by Prett and Garcia (1988) and Morari and Zafiriou (1989) and some conference proceedings edited by McAvoy *et al.* (1989). In books by Camacho and Bordon (1995) and Soeterboek (1991) detailed description of model based predictive control schemes can be found. MPC software packages and application assistance are available commercially. Applications include refinery processes (Cutler and Hawkins, 1987; Martin *et al.* 1986), pulp and paper processes (Matsko, 1985; Ruiz *et al.*, 1986) and a number of miscellaneous applications including a wind tunnel, glass furnace, steam generators, etc. (Mehra *et al.*, 1982) and a large municipal sewage system by Ricker (1989).

2.2.1 General Structure

2.2.1.1 Process Model

The process can be multi-input multi-output, non-square, linear-system with measured and / or unmeasured disturbances and hard constraints on the manipulated and / or output variables.

The concept of MPC is not limited to any particular model form but the model form chosen strongly affects the implementation and computational requirements. The most common model form is the truncated step (or convolution) model which can, in principle, be obtained simply by putting a unit step into the process input(s) and recording the output impulse response(s). Note that with this approach the process model is simply a vector of constants obtained simply by discretizing a recorded step response. It is used directly in the controller implementation and does not have to be transformed into transfer function or state space form. However, the model can be of almost any form, including transfer function (Prett and Garcia, 1988), state space (Ricker, 1989; Navratil *et al.*, 1989), unstable systems (Morari and Zafirou, 1989) and non-linear systems (Economou *et al.*, 1986; Bequette, 1991).

2.2.1.2 Objective Function

The various MPC algorithms propose different cost functions for obtaining the control. The general aim is that the future output (y) on the considered horizon should follow a determined reference signal (w) and, at the same time, the control effort (Δu) necessary for doing so should be penalized. The general expression for such an objective function will be:

$$J(N_1,N_2,N_u)=E\{\sum_{j=N_1}^{N_2}\delta(j)[y(t+j|t)-w(t+j)]^2+\sum_{j=1}^{N_u}\lambda(j)[\Delta u(t+j-1)]^2\} \qquad (2.1)$$

N_1 and N_2 are the minimum and maximum cost horizons and Nu is the control horizon, which does not necessarily have to coincide with the maximum horizon. The meaning of N_1 and N_2 is rather intuitive. They mark the limits of instants in which it is desirable for the output to follow the reference. Thus, if a high value of N_2 is taken, will provoke a smooth response of the process. Note that in processes with dead time d there is no reason for N_1 to be less than d because the output will not begin to evolve until instant t+d. Also if the process is non-minimum phase, this parameter will allow the first instants of inverse response to be eliminated from the objective function. The coefficients δ(j) and λ(j) are sequences that consider the future behaviour, usually constant values or exponential sequences are considered. For example it is possible to obtain an exponential weight of δ(j) along the horizon by using:

$$\delta(j) = \alpha^{N_2 - j} \qquad (2.2)$$

If α lies between 0 and 1, inclusively, this indicates that the errors farthest from instant t are penalized more than those nearest to it, giving rise to smoother control with less effort. If, on the other hand, α > 1 the first errors are more penalized, provoking a tighter control. In Predictive Functional Control (PFC) the error is only counted at certain points (coincidence points); this is easily achieved in the objective function giving a unity value to the elements of sequence δ(j) at said points and zero at the others.

One of the advantages of predictive control is that if the future evolution of the reference is known *a priori*, the system can react before the change has effectively been made, thus avoiding the effects of delay in the process response. The future evolution of reference r(t+k) is known beforehand in many applications, such as robotics, servos or batch processes; in other applications a noticeable improvement in performance can be obtained even though the reference is constant by simply knowing the instant when the value changes and getting ahead of this circumstance. In the objective function, the majority of methods use a reference trajectory w(t+k) which does not necessarily have to coincide with the real reference. It is normally a smooth progression from the actual value of the output y(t) towards the known reference by means of the first order system:

$$w(t)=y(t) \qquad w(t+k)=\alpha w(t+k-1)+(1-\alpha)r(t+k) \quad k=1 \ldots N \qquad (2.3)$$

where α is a parameter contained between 0 and 1 (the closer the value is to unity the smoother the approximation) that constitutes an adjustable value that will influence the dynamic response of the system.

In practice all processes are subject to restrictions. The actuators have a limited field of action as well as a determined slew rate, as is the case of valves, limited by the positions of totally open or closed and by the response rate. Construction reasons, safety or environmental ones or even the sensor scopes themselves can cause limits in the process variables such as levels in tanks, flows in piping or maximum temperatures and pressures; moreover, the operational conditions are normally defined by the intersection of certain constraints for basically economic motives, so that the control system will operate close to the boundaries. All of this makes the introduction of constraints in the function to be minimized necessary. Many predictive algorithms intrinsically take into account constraints (MAC, DMC) and have therefore been very successful in industry, whilst others can incorporate them *a posteriori*, for example the GPC (Camacho, 1993).

2.2.1.3 Control Law

In order to obtain values u(t+k|t) it is necessary to minimize the objective function. To do this the values of the predicted outputs $\hat{y}$ (t+k|t) are calculated as a function of past values of inputs and outputs and of future control signals, making use of the model chosen and substituted in the cost function, obtaining an expression whose minimization lead to the looked for values. An analytical solution can be obtained for the quadratic criterion if the model is linear and there are no constraints, otherwise an iterative method of optimization should be used. Whatever the method, obtaining the solution is not easy because there will be N_2-N_1+1 independent variables, a value which can be high (in order of 10 to 30). In order to reduce this degree of freedom a certain structure may be proposed for the control law. Furthermore, it can be found (De Keyser, 1991) that this structuring of control law produces an improvement in robustness and in the general behaviour of the system, basically due to the fact that allowing the free evolution of the manipulated variables (without being structured) may lead to undesirable high frequency control signals and at the worst to instability.

2.2.2 Dynamic Matrix Control (DMC)

First proposed by Cutler and Remaker (1980), DMC uses the step response to model the process, only taking into account the N first terms, therefore assuming the process to be stable and without integrators. This value of N is also the prediction horizon, using a control horizon Nu ≤ N. As regards the disturbances, their value will be considered to be the same as at instant t along all the horizon, that is, to be equal to the measured value of the output (y_m) less the one postulated by the model.

$$\hat{n}(t+k|t)=\hat{n}(t|t)=y_m(t)-\hat{y}(t|t) \tag{2.4}$$

Where $\hat{n}$ is the predicted noise disturbance and k=0,1,2,... sequence. Therefore the predicted value of the output will be

$$\hat{y}(t+k|t)=\sum_{i=1}^{k} g_i\Delta u(t+k-i)+\sum_{i=k+1}^{N} g_i\Delta u(t+k-i)+\hat{n}(t+k|t) \tag{2.5}$$

where g_i are the sample output values for step input Δu. The first term contains the future control actions to be calculated, the second one contains past values of the control actions and is therefore known, and the last one represents the disturbances. The cost function may consider future output errors only, or it may include the control efforts, in which case, presents the well known generic form. One great advantage of this method is that it allows complex dynamics such as non-minimum phase or delays to be described with ease. One of the characteristics of this method making it very popular in industry is the addition of constraints, in such a way that the constraint equations are added to the minimization. Optimization (numerical because of the presence of constraints) is carried out at each sampling instant and the value of u(t) is sent to the process as is normally done in all MPC methods. The inconveniences of this method are: (1) the size of the problem (involving a great burden of calculation) and (2) the inability to work with unstable processes.

2.2.3 Model Algorithmic Control (MAC)

Richalet *et al.* (1976, 1978) were the first researchers to present Model Predictive Heuristic Control (MPHC) which later became Model Algorithmic Control (MAC), whose software is called IDCOM (IDentification-COMmand). It is very similar to the DMC with a few differences. Unlike the DMC, it uses an impulse response model (also known as weighting sequence or convolution model) valid only for stable processes with stationary disturbances. The predicted output is related to the past and future input values by the equation

$$\hat{y}(t+k|t)=\sum_{i=1}^{N} h_i u(t+k-i|t)=H(z^{-1})u(t+k|t) \tag{2.6}$$

where h_i is the sampled output when the process is excited by a unitary impulse of equal amplitude as the sampling period. The impulse response is truncated and only N values are considered (thus only stable processes without integrators can be represented). $H(z^{-1})$ is a polynomial of order N, z^{-1} being the backward shift operator.

Furthermore, this method makes no use of the control horizon concept, so in the cost function the number of control signals is equal to that of the future outputs. It introduces a reference trajectory as a first order system which evolves from the actual output to the setpoint according to a determined time constant. The variance of the error between this trajectory and the output is what one aims at minimizing in the objective function. The disturbance can be treated as in DMC or their estimation can be carried out by the following recursive expression:

$$\hat{n}(t+k|t)=\alpha\hat{n}(t+k-1|t)+(1-\alpha)(y_m(t)-\hat{y}(t|t)) \tag{2.7}$$

with $\hat{n}(t|t)=0$ and $0\leq\alpha<1$. α is an adjustable parameter closely related to the response time, the bandwidth and the robustness of the closed loop system (Garcia *et al.*, 1989). It also takes into account constraints in the actuators as well as in the internal variables or secondary outputs.

2.2.4 Extended Prediction Self Adaptive Control (EPSAC)

Extended Prediction Self Adaptive Control (EPSAC) by De Keyser *et al.* (1985) proposes a constant control signal starting from the present moment while using a sub-optimal predictor instead of solving a Diophantine equation. For predicting the process is modelled by the transfer function

$$A(z^{-1})y(t) = B(z^{-1})u(t-d)+v(t) \tag{2.8}$$

where d is the delay and v(t) the disturbance. The model can be extended by a term $D(z^{-1})d_m(t)$, with $d_m(t)$ being a measurable disturbance in order to include feedforward effect. Using this method the prediction is obtained as shown in De Keyser *et al.* (1988). One characteristic of this method is that the control law structure is very simple as it is reduced to considering that the control signal is going to stay constant from instant t, that is $\Delta u(t+k)=0$ for $k>0$. In other words the control horizon is reduced to unity and therefore the calculation is reduced to one single value of u(t). To obtain this value a cost function is minimized:

$$\sum_{k=d}^{N} \gamma(k)[w(t+k)-P(z^{-1})\hat{y}(t+k|t)]^2 \tag{2.9}$$

$P(z^{-1})$ being a design polynomial with unit static gain and factor $\gamma(k)$ being a weighting sequence. The control signal can be calculated analytically (which is an advantage over the DMC and MAC) in the form:

$$u(t) = \frac{\sum_{k=d}^{N} \alpha_k [w(t+k) - P(z^{-1})\hat{y}(t+k|t)}{\sum_{k=d}^{N} \gamma(k)\alpha_k^2} \tag{2.10}$$

α being the discrete impulse response of the system.

2.2.5 Extended Horizon Adaptive Control (EHAC)

First proposed by Ydstie (1984), EHAC tries to keep the future output (calculated by Diophantine equation) close to the reference at a period of time after the process delay and also permits different manoeuvres. This formulation considers the process modelled by transfer function without taking the model disturbances into account:

$$A(z^{-1})y(t) = B(z^{-1})u(t-d) \tag{2.11}$$

It aims at minimizing the discrepancy between the model and the reference at instant t+N:

$$J=\hat{y}(t+N|t)-w(t+N), \qquad with \quad N \geq d \tag{2.12}$$

The solution to this problem is not unique (unless N=d); a possible alternative is to consider that the control horizon is unity, that is

$$\Delta u(t+k-1)=0 \qquad 1<k \leq N-d \tag{2.13}$$

or to minimize the control effort:

$$J=\sum_{k=0}^{N-d} u^2(t+k) \tag{2.14}$$

There is an incremental version of EHAC that allows the disturbances to in the load to be dealt with easily, it is expressed as

$$J=\sum_{k=0}^{N-d} \Delta u^2(t+k) \tag{2.15}$$

In this formulation a predictor of N steps is used as follows

$$\hat{y}(t+n|t)=y(t)+F(z^{-1})\Delta y(t)+E(z^{-1})B(z^{-1})\Delta u(t+N-d) \tag{2.16}$$

$E(z^{-1})$ and $F(z^{-1})$ being polynomials satisfying the equation

$$(1-z^{-1})=A(z^{-1})E(z^{-1})(1-z^{-1})+z^{-N}F(z^{-1})(1-z^{-1}) \tag{2.17}$$

with the degree of E being equal to N^{-1}. One advantage of this method is that a simple explicit solution can easily be obtained, resulting in

$$u(t) = u(t-1)+\frac{\alpha_0(w(t+N)-\hat{y}(t+N|t))}{\sum_{k=0}^{N-d} \alpha_i^2} \tag{2.18}$$

α_k being the coefficient corresponding to $\Delta u(t+k)$ in the prediction equation. Thus the control law only depends on the process parameters and can therefore easily be made self-tuning if it has an on-line identifier. As can be seen the only parameter of adjustment is the horizon of prediction N, which simplifies its use but provides little freedom for the design. Another disadvantage of this method is that the reference trajectory cannot be used because the error is only considered at each point, so that certain offsets in the performance cannot be eliminated.

2.2.6 Generalized Predictive Control (GPC)

The GPC method was first proposed by Clarke *et al.* (1987) and has received a lot of attention both from industry and academia. It has been successfully implemented in many industrial applications (Clarke, 1988), showing good performance and a certain degree of robustness with respect to overparameterization or poorly known delays. It can handle many different control problems for a wide range of plants with a reasonable number of design variables, which have to be specified by the user depending upon a prior knowledge of the plant and control objectives.

The basic idea of GPC is to calculate a sequence of future control signals in such a way that it minimizes a multistep cost function defined over a prediction horizon. The index to be optimized is the expectation of a quadratic function measuring the difference between the predicted system output and some predicted reference sequence over the horizon plus a quadratic function measuring the control efforts.

The GPC has many advantages, among which it provides an analytical solution (in absence of constraints), it can deal with unstable and non-minimum phase processes and incorporates the concept of control horizon as well as the consideration of weighting of control increments in the cost function. The general set of choices available for GPC leads to a greater variety of control objectives compared to other MPC approaches, some of which can be considered as subsets or limiting cases of GPC.

In spite of great success of GPC in industry, there was original lack of theoretical work on the properties of predictive control. Initially, there was lack of accountability on stability and robustness of GPC. In fact, the majority of stability results are limited to the infinite horizon case and there is a lack of a clear theory relating the closed loop behaviour to design parameters such as horizons and weighting sequences.

Bearing in mind the need to solve some of the these problems, a variation of the standard formulation of GPC was developed by Clarke and Scattolini (1991) called Constrained Receding-Horizon Predictive Control (CRHPC) and similar work by De Nicolao and Scattolini (1994) and Mosca and Zhang (1992), which allows stability and robustness results to be obtained for small horizons. The idea basically consists of deriving a future control sequence so that the predicted output over some future time range is constrained to be at reference value exactly. Some degrees of freedom of some future control signals are employed to force the output, whilst the rest is available to minimize the cost function over a certain interval.

To overcome the lack of stability results for GPC, a new formulation has been proposed by Kouvaritakis *et al.* (1992) and Rossiter and Kouvaritakis (1994) which ensures that the associated cost function is monotonically decreasing, guaranteeing closed-loop stability. The algorithm is called Stable Generalized Predictive Control (SGPC) and is based on stabilizing the loop before the application of the control strategy. Now the future values of the reference that are sent to the closed-loop must be calculated first and the system inputs are obtained in turn as a function of these values.

2.3 CONSTRAINED MPC

It is not very realistic to formulate a control problem considering all signals to posses an unlimited range since all processes in practice are subject to constraints (Camacho and Bordon, 1995). Actuators have a limited range of action and a limited slew rate, as is the case of control valves which are limited by a fully closed and fully open position and a maximum slew rate. Construction and / or safety reasons, as well as sensor range, cause bounds in process variables, as in the case of levels in tanks and flows in pipes. Furthermore, in practice, the operating points of plants are determined to satisfy economic goals and lie at the intersection of certain constraints. The control system normally operates close to limits and constraint violations are likely to occur. The control system, especially for long range predictive control, has to anticipate constraint violations and correct them in an appropriate way.

2.3.1 Constraints and GPC

From Clarke *et al* (1987) the GPC control actions can be calculated by computing vector **u** of future control increments that minimizes a quadratic objective function given by

$$J=(Gu+f-w)^T(Gu+f-w) + \lambda u^T u \tag{2.19}$$

where **w** is a vector of future setpoints, **G** is a matrix and **f** a vector formed from the Diophantine equations, and λ is the control weighting. The optimal solution of this problem is found by solving the linear equation

$$u=(G^T G+\lambda I)^{-1}G^T(w-f) \tag{2.20}$$

In practice, the normal way of using a GPC is to compute the current control signal u(t) and apply it to the process. If u(t) violates the constraint it is fixed at its limits, either by the control program or by the actuator. The case of u(t+1),...,u(t+N) violating the constraints is not even considered as in most cases these signals are not even computed.

This way of operating does not guarantee that the optimum will be obtained when constraints are violated. The main purpose of GPC, which is to apply the best possible control signal by minimizing the objective function J, will not be achieved.

Not considering constraints on manipulated variables to their full extent, may results in higher values of the objectives function and thus in a poorer performance of the control system. However, manipulated variables can always be kept to their limits either by the control program or by the actuator and this is not the main reason for treating constraints in an appropriate way. Violating the limits on the controlled variables may be more costly and dangerous as it could cause damage to equipment and losses in production. For example, in most batch reactors the quality of the production requires some of the variables to be kept within specified limits violating these limits may create a bad quality product and in some cases the loss of the whole batch. When the limits have been set because of safety reasons, the violation of these limits could cause damage to equipment, spillage, or in most cases the activation of the emergency system which will normally produce an emergency stop of the process, losing and /or delaying production, and a normally costly start up procedure.

Constraint violations on the output variables are not contemplated when the only way of handling constraints is by clipping the manipulated variables. One of the main advantages of MPC, its prediction capabilities, is not used to its full potential by operating in this way. Control systems, especially long-range predictive control should anticipate constraint violation and correct them in an appropriate way.

The constraints acting on the process can originate from amplitude limits in the control signals, slew rate limits of the actuator and the limits on the output signals, and can be described respectively by

$$\begin{gathered} \underline{U} \le u(t) \le \overline{U} \quad \forall t \\ \underline{u} \le u(t)-u(t-1) \le \overline{u} \quad \forall t \\ \underline{y} \le y(t) \le \overline{y} \quad \forall t \end{gathered} \tag{2.21}$$

These constraints can be expressed as

$$\begin{aligned} 1\underline{U} \le Tu+u(t-1)1 &\le 1\overline{U} \quad \forall t \\ 1\underline{u} \le u &\le 1\overline{u} \quad \forall t \\ 1\underline{y} \le Gu+f &\le 1\overline{y} \quad \forall t \end{aligned} \tag{2.22}$$

where 1 is an N vector whose entries are ones and T in an N x N lower triangle matrix whose entries are ones. The constraints can be expressed in condensed form as

$$Ru \le c \tag{2.23}$$

with

$$R = \begin{bmatrix} I_{NxN} \\ -I_{NxN} \\ T \\ -T \\ G \\ -G \end{bmatrix} \quad c = \begin{bmatrix} 1\overline{u} \\ -1\underline{u}1\overline{U}-1u(t-1) \\ -1\underline{U}+1u(t-1) \\ 1\overline{y}-f \\ -1\underline{y}+f \end{bmatrix} \tag{2.24}$$

The constraints on the output variables of the type $\underline{y} \le$ y(t) $\le y$ are normally imposed because of the safety reasons. Other types of constraints can be set on the process controlled variables to force the response of the process to have certain characteristics, as shown by Kutnetsov and Clarke (1994), and can also be expressed in a similar manner.

2.3.1.1 Band Constraints

Sometimes one wishes the controlled variables to follow a trajectory within a band. In the food industry, for example, it is very usual for some operations to require a temperature profile that has to be followed with a specified tolerance.

This type of requirement can be introduced in the control system by forcing the output of the system to be included in the band formed by the specified trajectory plus-minus the tolerance. That is

$$\underline{y}(t) \leq y(t) \leq \bar{y}(t) \tag{2.25}$$

These constraints can be expressed in terms of the increments of the manipulated variables as follows

$$\begin{aligned} Gu &\leq \bar{y}-f \\ Gu &\geq \underline{y}-f \end{aligned} \tag{2.26}$$

2.3.1.2 Overshoot Constraints

In some processes overshoots are not desirable. In the case of robotics, for example, an overshoot may produce a collision with the workplace or with the piece it is trying to grasp.

Overshoot constraints have been treated by Kutnetsov and Clarke (1994) and are very easy to implement. Every time a change is produced in the setpoint, which is considered to be kept constant for a sufficiently long time, the following constraints are added to the control system

$$y(t+j) \leq w(t) \quad for\ j = N_{o1}...N_{o2} \tag{2.27}$$

where N_{O1} and N_{O2} define the horizon where the overshoot may occur (N_{O1} and N_{O2} can always be made equal to 1 and N if this is not known). These constraints can be expressed in terms of the increments of the manipulated variables as follows

$$Gu \leq 1w(t)-f \tag{2.28}$$

2.3.1.3 Monotonic Behaviour

Some control systems tend to exhibit oscillations, known as kick back, on the controlled variables before they have gone over the setpoints. These oscillation are not desirable in general because, amongst other reasons, they may cause perturbations in other processes. Constraints can be added to the control system to avoid this type of behaviour by imposing a monotonic behaviour on the output variables. Each time a setpoint changes, and is again considered to be kept constant for a sufficiently long period, new constraints with the following form are added to the control system:

$$\begin{matrix} y(t+j) \leq y(t+j+1) & if\ \ y(t) < w(t) \\ y(t+j) \geq y(t+j+1) & if\ \ y(t) > w(t) \end{matrix} \tag{2.29}$$

These type of constraints can be expressed in terms of the manipulated variables as follows

$$Gu+f \leq \left[\frac{0^T}{G'}\right]u + \left[\frac{y(t)}{f'}\right] \tag{2.30}$$

where G' and f' result from clipping the last n rows (n is the number of the output variables) of G and f. These constraints can be expressed as

$$\begin{bmatrix} G_0 & 0 & \cdots & 0 \\ G_1-G_0 & G_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ G_{N-1}-G_{N-2} & G_{N-2}-G_{N-3} & \cdots & G_0 \end{bmatrix} u \leq \begin{bmatrix} y(t)-f_1 \\ f_1-f_2 \\ \vdots \\ f_{N-1}-f_N \end{bmatrix} \tag{2.31}$$

2.3.1.4 Non-minimum Phase Behaviour

Some processes exhibits a type of non-minimum phase behaviour. That is, when the process is excited by a step in its input the output variables tends to first move in the opposite direction prior to moving to the final position. This kind of behaviour may not be desirable in many cases.

Constraints can be added to the control system to avoid this type of behaviour. The constraints take the form

$$\begin{array}{lll} y(t+j) \leq y(t) & \quad if & y(t)<w(t) \\ y(t+j) \geq y(t) & \quad if & y(t)>w(t) \end{array} \tag{2.32}$$

These constraints can be expressed in terms of the increments of the manipulated variables as follows

$$Gu \geq 1y(t) - f \tag{2.33}$$

2.3.1.5 Actuator Nonlinearities

Most actuators in industry exhibit dead zones and other type of nonlinearities. Controllers are normally designed without taking into account actuator nonlinearities. Because of the predictive nature of MPC, actuator nonlinearities can be dealt with as suggested by Chow and Clarke, (1994).

Dead zones can be treated by imposing constraints on the controller in order to generate control signals outside the dead zone, say $(\underline{u}_d, \bar{u}_d)$ for dead zone on the slew rate of actuators and $(\underline{U}_d, \bar{U}_d)$ for the dead zone on the amplitude of actuators. That is

$$\begin{gathered} 1\underline{U}_d \leq Tu + u(t-1)1 \leq 1\overline{U}_d \\ 1\underline{u}_d \leq u \leq 1\bar{u}_d \end{gathered} \tag{2.34}$$

The feasible region generated by this type of constraints is non-convex and the optimization problem is difficult to solve as pointed out.

2.3.1.6 Terminal State Constraints

These types of constraints appear when applying CRHPC of Clarke and Scattolini

(1991) where the predicted output of the process is forced to follow the predicted reference during a number of sampling periods m after the costing horizon N_y. The terminal state constraints can be expressed as a set of equality constraints on the future control increments by using the prediction equation for $y_m=[y(t+N_y+1)^T....Y(t+N_y+m)^T]^T$:

$$y_m = G_m u + f_m \tag{2.35}$$

If the predicted response is forced to follow the future reference setpoint w_m, the following equality constraint can be established

$$G_m u = w_m - f_m \tag{2.36}$$

All constraints treated so far can be expressed as $Ru < c$ and $Au=a$. The GPC problem when constraints are taken into account consists of minimizing a quadratic objective function subject to a set of linear constraints. That is, the optimization of a quadratic function with linear constraints, what is usually known as q quadratic programming problem (QP).

2.4 NON-LINEAR (PREDICTIVE) CONTROL

In this section it is not attempted to review the whole area of non-linear control theory but to discuss only the non-linear (predictive/adaptive) control strategies which we consider relevant to this research. In section 2.3.1 a self-tuning controller for non-linear processes developed by Agarwal and Seborg (1987) is described in detail and section 2.3.2 gives a non-linear multistep predictive algorithm developed by Brengel and Seider (1989).

2.4.1 A Self-tuning Controller for Non-linear Systems

This control strategy was proposed by Agarwal and Seborg (1987). They consider the non-linear discrete-time process model of the form

$$A(z^{-1})y(t)=\sum_{i=1}^{N} b_i u^{r_i}(t-k)Y_i(t-1)+d+C(z-1)\xi(t) \tag{2.37}$$

$$Y_i(t-1)\equiv g_i[y(t-1-m),\ m=0,1,\ldots;\quad u(t-k-j),\ j=1,2,\ldots],\ i=1,2,\ldots,N \tag{2.38}$$

where t is the sampling instant, t=1,2,3,...; k is the known time delay expressed as one plus the largest integer multiple of the sampling period smaller than or equal to the time delay ($k \geq 1$); y(t) is the measured output at time t; u(t) is the manipulated input at time t; $\xi(t)$ is zero-mean random noise independent of previous inputs and outputs; d is the unknown disturbance; g_i are known single-valued time-invariant functions involving known parameters; z^{-1} is the backward shift operator; b_i are system parameters and $A(z^{1})$ and

$C(z^{-1})$ are polynomials with the assumption that all roots of $C(z^{-1})$ lie inside the unit circle in the z-plane. Parameters r_i, N, n_A, n_C are known integers with $r_i \geq 0$.

They considered the following performance index

$$I = E\{[P(z^{-1})y(t+k) - R(z^{-1})w(t)]^2 + [Q(z^{-1})u(t)]^2\} \qquad (2.39)$$

where E is the expectation operator conditioned with respect to data up to time t, w(t) is the setpoint, $R(z^{-1})$ and $Q(z^{1})$ are rational transfer functions, and $P(z^{1})$ is an observer polynomial.

The control is rather complex and is obtained by minimizing the above cost function and utilizing the Diophantine equation. The main drawback of this method is its complex algebraic manipulations.

2.4.2 Multistep Nonlinear Predictive Controller

Brengel and Seider (1989) present a model predictive formulation to control MIMO nonlinear processes. It directly extends the model predictive concepts that have been exploited for the control of linear systems (e.g., IMC and Q/DMC). This algorithm uses multistep predictor, unlike the algorithm by Economou *et al.* (1986) and Li and Biegler (1988), which were limited to a single predictive step. Brengel and Seider (19889) derive the multistep predictor through a linearization of the ordinary equations at several instants within a sampling interval, leading to recursive, algebraic equations that relate the predicted outputs to future and past values of the manipulated inputs. This multistep algorithm was shown to result in superior performance over single-step methods. Furthermore, this multistep algorithm does not require iterative calculations, as required by Economou at al. (1986) and is more efficient for each predictive step. The algorithm also easily handles constraints involving the state variables and manipulated inputs.

2.4.2.1 The Algorithm

Brengel and Seider (1989) aim to develop a general algorithm that can efficiently control nonlinear processes near and within operating regimes characterized by hysteresis and periodic or even chaotic operation. To meet this goal, they developed an algorithm which constructs an easily calculable, explicit expression that relates the process outputs to the manipulated inputs. Their Non-Linear Prediction (NLP) is simplified by replacing the ODEs with an expression for $x_{k+1},\ldots,x_{K+P}$ as a function of the manipulated inputs at past and future sampling instants.

With the dynamic constraints replaced, the NLP can easily be solved, using programs such as OPT (Lang and Biegler, 1987) and MINOS(Murtagh and Saunders, 1983), for the optimal values of the manipulated inputs and the associated process trajectory, given a set of weighting coefficients and tuning parameters. Brengel and Seider's result assumes that there is no process/ model mismatch and that all of the disturbance are measurable. (For a nonlinear process, because the principle of superposition does not hold, an unmeasured disturbance is analogous to process/model mismatch.) In practice, however, the nonlinear model only approximates the process performance, sensor noise exists, and unmeasured disturbances normally enter the process.

To solve the nonlinear control problem, the control algorithm needs to compensate for the process/model mismatch and sensor noise. Since the model is expected to deviate from the process, it is not important to obtain very accurate values of $x_{K+1},\ldots,x_{K+P}$. Instead, it should be more efficient to obtain approximate solutions that display the proper dynamics trends while requiring few computations. To achieve this, Brengel and Seider developed the following approximation method.

Brengel and Seider (1989) consider a non-linear model which is used to predict the vector of n outputs, x, at a each of P future sampling instants as a function of the vector of n manipulated inputs, u, at M future sampling instants, where k is the index of the current sampling instant. They, thus, formulate the following NLP

$$\begin{aligned} \min_{u_k,\ldots,u_{k+M-1}} \sum_{j=1}^{P} [\gamma_j^2(x_{k+j}^{SP}-x_{k+j})^T C(x_{k+j}^{SP}-x_{k+j}) + \\ \beta_j^2(u_{k+j-1}-u_{k+j-2})^T D(u_{k+j-1}-u_{k+2})] \end{aligned} \tag{2.40}$$

subject to

$$\Lambda\dot{x}=f\{\,;u\} \qquad x\{t=0\}=x_0 \qquad u\{t=0\}=u_0$$

$$g\{x;u\}=0$$

$$u^L \leq u_{k+j-1} \leq u^H \qquad j=1,\ldots,M$$

$$u_{k+M}=u_{k+M+1}=\ldots=u_{k+P-1}=u_{k+M-1} \qquad \beta_j=0 \quad \forall j>M$$

$$h\{x;u\} \leq 0$$

Let x_k and u_{k-1} be the deviation variables from the steady state at the current sampling instant, just prior to a step change in the inputs, and expand the right-hand side of the differential equation about x_k and u_{k-1} in a first-order Taylor series:

$$\Lambda\dot{x}=f\{0,0\}+\left(\frac{\partial f}{\partial x}\right)_{x_k,u_{k-1}}(x-x_k)+\left(\frac{\partial f}{\partial u}\right)_{x_k,u_{k-1}}(u-u_{k-1}) \tag{2.41}$$

Defining another set of deviation variables gives

$$x=x-x_k, \quad u=u-u_{k-1} \tag{2.42}$$

The differential equation can be expressed as

$$\Lambda\dot{x}=f\{0,0\}+\left(\frac{\partial f}{\partial x}\right)_{0,0}x+\left(\frac{\partial f}{\partial u}\right)_{0,0}u$$
$$\Lambda\dot{x}=f\{0,0\}+J_x x+J_u u \tag{2.43}$$

Taking the Laplace Transform and simplifying gives

$$x\{s\}=(s\Lambda-J_x)^{-1}\left[J_u u\{s\}+\frac{1}{s}f\{0,0\}\right] \tag{2.44}$$

Then, for a u step change in the input vector at the current sampling instant,

$$u\{s\}=\frac{1}{s}u \tag{2.45}$$

and

$$x\{s\}=\frac{1}{s}(s\Lambda-J_x)^{-1}[J_u u+f\{0,0\}] \tag{2.46}$$

By use of a general expansion, the inverse can be expressed analytically, and

$$\frac{1}{s}(s\Lambda-J_x)^{-1}=\frac{B}{s}+\sum_{i=1}^{n}\frac{C_i}{s-\lambda_i} \tag{2.47}$$

where λ is a vector of n eigenvalues of $\Lambda^{-1}J_x$. Substituting the above equation into (2.46) gives

$$x\{s\} = \left[\frac{B}{s} + \sum_{i=1}^{n} \frac{C_i}{s-\lambda_i}\right](J_u u + f\{0,0\}) \tag{2.48}$$

The above equation expresses the linearized dynamic constraints in the continuous Laplace domain. To obtain the output vectors at future sampling instants as a function of the past and future manipulated inputs, Brengel and Seider (1989) invert the above equation. They thus obtain the resulting expression in the time domain at t=T

$$x_{k+1} = [B + \sum_{i=1}^{n} C_i e^{\lambda_i T}](J_u u_k + f_k) \tag{2.49}$$

Therefore, in the above equation deviation variables can be replaced and then equation generalized to apply for projections of the output vector j sampling instants into the future. The above equation can be derived using the state transition matrix. However, this approach requires calculation of the eigenvectors of $\Lambda^{-1}J_x$, a step not required in this algorithm.

Normally, sampling time is in the order of the smallest time constant of the process model. However, variations in the smallest time constant may require the linearization to be performed more often. An alternative to reducing the sampling time, to obtain more accuracy and better control, is to implement the local linearization at equally spaced times within the sampling intervals.

Brengel and Seider (1989) indicate that the use of repeated linearization of the nonlinear ODEs differs significantly from the aforementioned methods of controlling nonlinear processes in which the model is linearized once and the linear model is used over the entire predictive horizon (The latter is analogous to the linearized constraints for a single iteration of the SQP algorithm). For highly nonlinear processes, this assumption can be expected to result in poor, or even unstable, control.

The problem of modifying the model to more accurately represent the process dynamics is difficult to address theoretically. When no measurement noise exists and the dynamic equations have the correct terms, the parameters can be updated to eliminate the process/model mismatch. With measurement noise or an incorrect structure for the model, it may be prudent to restrict the number of parameters that are updated, the rate at which they can change, or some combination of the two, to obtain a more effective updating procedure. Such strategies are analogous to those employed for parameter estimation and adaptive control. The fact that almost the whole process model should be known entirely, *a priori*, is the main drawback of this method.

2.5 RECURSIVE PARAMETER ESTIMATION

In many applications, such as self-tuning control algorithms, it is necessary to estimate the process variables and noise model parameters, or the controller parameters (for the implicit case) recursively as data become available. In this section some of the most common algorithms, namely, recursive least squares, recursive generalized least squares, recursive extended least squares and recursive maximum likelihood are presented.

Consider a system described by the following CARMA (controlled autoregressive moving average) model structure

$$A(z^{-1})y_t = B(z^{-1})u_{t-f} + H(z^{-1})\varepsilon_t \tag{2.50}$$

where ε_t is a white noise sequence, f is the number of whole periods of time delay and

$$A(z^{-1}) = 1 + a_1 z^{-1} + \ldots + a_{n_a} z^{-n_a} \tag{2.51}$$

$$B(z^{-1}) = b_1 z^{-1} + \ldots + b_{n_b} z^{-n_b} \tag{2.52}$$

and $H(z^{-1})$ is a noise filter polynomial. Each of the recursive estimation algorithms mentioned above can be described by the following common algorithm (Warwick and Rees, 1986)

$$\hat{\theta}_t=\hat{\theta}_{t-1}+K_t\varepsilon_t \tag{2.53}$$

$$K_t=\frac{P_{t-1}x_t}{\lambda+x_t^TP_{t-1}x_t} \tag{2.54}$$

$$P_t=\frac{1}{\lambda}\left\{P_{t-1}-\frac{P_{t-1}x_tx_t^TP_{t-1}}{\lambda+x_t^TP_{t-1}x_t}\right\} \tag{2.55}$$

where $\hat{\theta}$ is a vector of parameter estimates based on the input/output data available up to and including time t, ε_t is an estimate of the one-step ahead prediction error, K is the Kalman gain, and P_t is a matrix proportional to the variance-covariance matrix of the parameter estimates. The scalar λ is a discounting factor which in the standard algorithms for estimating fixed but unknown parameters is set to one but when it is desired to discount past data in an exponential fashion this term is set between .90 and 1.0.

2.5.1 Recursive Least Squares (RLS)

Consider the model (2.50) with the noise filter $H(z^{-1})=1$. y_t can be expressed in terms of past values of y and u as

$$y_t = -a_1 y_{t-1} - \dots - a_{n_a} y_{t-n_a} + b_1 u_{t-f-1} + \dots + b_{n_b} u_{t-f-n_b} + \varepsilon_t \tag{2.56}$$

$$y_t = x_t^T \theta + \varepsilon_t \tag{2.57}$$

where

$$x_t = (-y_{t-1}, \dots, -y_{t-n_a}, u_{t-f-1}, \dots, u_{t-f-n_b})^T$$

$$\theta = (a_1, \dots, a_{n_a}, b_1, \dots, b_{n_b})^T$$

So, at any sampling instant t the least squares estimates of θ based on observations $Y_t(y_1, y_2, \dots, y_t)$ can be obtained by solving the following equations

$$\hat{\theta}_t = (X_t^T X_t)^{-1} X_t^T Y_t \tag{2.58}$$

where X_t is $(x_1, x_2, \dots, x_t)^T$

Defining P_{t-1} as (also see 2.69)

$$P_{t-1}=(X_{t-1}^{T}X_{t-1})^{-1} \tag{2.59}$$

where

$$P_{t}=(X_{t}^{T}X_{t})^{-1}=(X_{t-1}^{T}X_{t-1}+x_{t}x_{t}^{T})^{-1} \tag{2.60}$$

we can rewrite (2.60) as

$$P_{t}=(P_{t-1}^{-1}+x_{t}x_{t}^{T})^{-1} \tag{2.61}$$

Using a well known matrix inversion lemma (Noble, 1969) the P_t matrix can be expressed as

$$P_{t}=P_{t-1}-\frac{P_{t-1}x_{t}x_{t}^{T}P_{t-1}}{(1+x_{t}^{T}P_{t-1}x_{t})} \tag{2.62}$$

Substituting this expression for P_t into the least squares equation (2.58) gives

$$\hat{\theta}_t = \left[P_{t-1} - \frac{P_{t-1}x_t x_t^T P_{t-1}}{(1+x_t^T P_{t-1} x_t)}\right]\left[X_{t-1}Y_{t-1} + x_t Y_t\right] \tag{2.63}$$

$$= P_{t_1} X_{t-1}^T Y_{t-1} - \frac{P_{t-1}x_t}{(1+x_t^T P_{t-1} x_t)} x_t^T P_{t-1} X_{t-1}^T Y_{t-1}$$

$$+ \frac{P_{t-1}x_t(1+x_t^T P_{t-1}x_t - x_t^T P_{t-1}x_t}{(1+x_t^T P_{t-1} x_t)} Y_t \tag{2.64}$$

Upon substituting

$$\hat{\theta}_{t-1} = P_{t-1} X_{t-1}^T Y_{t-1} \tag{2.65}$$

and collecting terms we get

$$\hat{\theta}_t = \hat{\theta}_{t-1} + K_t(Y_t - x_t^T \hat{\theta}_{t-1}) \tag{2.66}$$

where

$$K_t = \frac{P_{t-1}x_t}{(1+x_t^T P_{t-1} x_t)} \tag{2.67}$$

The recursive least square algorithm (2.65), (2.66), (2.67) can thus be seen to be the general form (2.53), (2.54), (2.55) with λ = 1.0, and with the one-step prediction error given by

$$\hat{\varepsilon}_t = (Y_t - x_t^T \hat{\theta}_{t-1}) \tag{2.68}$$

Thus RLS algorithm necessitates the recursive updating of two equations, one for the parameter estimates $\hat{\theta}_t$ and one for the matrix P_t which is proportional to covariance matrix of the estimates

$$P_t = (X_t^T X_t)^{-1} = \frac{1}{\sigma_t^2} cov(\hat{\theta}_t) \tag{2.69}$$

Initial values $\hat{\theta}(0)$ and P(0) are needed to start the algorithm. Taking P(0)=αI where α is large, is a common choice when little prior knowledge of $\hat{\theta}(0)$ exists. In this case the recursive and non-recursive version of least squares are equivalent. The simplicity of the recursive algorithm stems from replacing the matrix inversion at each stage by a simple scalar inversion.

To allow for time varying parameters, one may discount past data in an exponential fashion by minimizing

$$\sum_{j=1}^{t} \lambda^{t-j} \varepsilon_j^2 \tag{2.70}$$

with $\lambda < 1.0$.

2.5.2 Recursive Generalized Least Squares (RGLS)

Biased parameter estimates are obtained when using linear least squares estimation if the disturbance term is correlated. Let us consider the model (2.50) with a noise filter

$$H(z^{-1}) = \frac{1}{C(z^{-1})} \tag{2.71}$$

This allows flexibility in accounting for autocorrelation errors.

The model can therefore be written as

$$A(z^{-1})y_t = B(z^{-1})u_{t-1} + n_t \tag{2.72}$$

where

$$C(z^{-1})n_t=\varepsilon_t \tag{2.73}$$

n_t is correlated, but the autocorrelation function is known. The sequence ε_t is uncorrelated and the $C(z^{-1})$ is invertible. The system equation can therefore be rewritten as

$$A(z^{-1})y_t^*=B(z^{-1})u_{t-f}^*+\varepsilon_t \tag{2.74}$$

where

$$y_t^*=C(z^{-1})y_t \tag{2.75}$$

and

$$u_t^*=C(z^{-1})u_t \tag{2.76}$$

Then the least squares can be applied, once the values $y^*(t)$, $u^*(t)$ are calculated, in order to obtain the estimates of the parameters within the polynomials $A(z^{-1})$ and $B(z^{-1})$. The filter polynomial $C(z^{-1})$ is also estimated.

The idea of RGLS, therefore, is to alternately apply the ordinary RLS algorithm to each of these in turn. The following is a summary of the algorithm:

1. Fit model equation (2.50) with modifications shown in equation (2.71) to data by using linear least squares parameter estimation.

2. Test the residuals obtained for whiteness. If they are white then stop.

3. Using linear least squares estimate $C(z^{-1})$.

4. Filter the input and output through $C(z^{-1})$

5. Go to 1

2.5.3 Recursive Extended Least Squares (RELS)

Lets take the noise filter $H(z^{-1})=C(z^{-1})$ in the model equation (2.50), where $C(z^{-1})$ and $A(z^{-1})$ are assumed to have no common factors.

The model can therefore be expressed as

$$y_t = x_t^T \theta + \varepsilon_t \tag{2.77}$$

where

$$x_t^T = [-y_{t-1}, \ldots, -y_{t-n_a}, u_{t-f-1}, \ldots, u_{t-f-n_b}, \varepsilon_{t-1}, \ldots, \varepsilon_{t-n_c}]$$

$$\theta^T = [a_1, \ldots, an_a, b_1, \ldots, b_{n_b}, c_1, \ldots, c_{n_c}]$$

This set of equations suggests that RLS be used on the extended x_t and θ vectors. However, the ε_t is not known. We thus proceed to replace x_t^T vector by

$$\begin{aligned}\hat{x}_t^T &= [-y_t,\dots,-y_{t-n_a},u_{t-f-1},\dots,u_{t-f-n_b},\dots,\\ &\hat{\varepsilon}_{t-1},\dots,\hat{\varepsilon}_{t-n_c}]\end{aligned} \tag{2.78}$$

where ε_t is the estimated residual sequence given by

$$\hat{\varepsilon}_t=y_t-\hat{x}_t^T\hat{\theta}_{t-1} \tag{2.79}$$

The standard recursive algorithm can therefore be used with the extended θ_t^1 vectors defined above.

2.5.4 Recursive Maximum Likelihood (RML)

The recursive least squares parameter estimation method deals with models that are linear in parameters. Maximum likelihood estimation however, covers a more general class of problems.

Consider model equation (2.50) with the noise filter $H(z^{-1})= C(z^{-1})$. Polynomial $C(z^{-1})$ has roots inside the unit circle. If the ε_t sequence is assumed to be normally distributed then the maximum likelihood estimates of the parameters, conditional upon starting values for equation (2.50), are given by minimizing the sum of squares function

(Box and Jenkins, 1990)

$$\sum_{t=1}^{N} \varepsilon_t^2 \tag{2.80}$$

The off-line minimization of this function is obtained by iterating the following equation

$$\hat{\varepsilon}_t = y_t + \hat{a}_1 y_{t-1} + \ldots + \hat{a} y_{t-n_a} - \hat{b}_1 u_{t-f-1} - \ldots$$

$$- \hat{b}_{n_b} u_{t-f-n_b} - \hat{c}_1 \hat{\varepsilon}_{t-1} - \ldots - \hat{c}_{n_c} \varepsilon_{t-n_c} \tag{2.81}$$

Iteration is continued until the residual sequence calculated has a minimum sum of squa.. .. The off-line algorithm is usually based upon linearizing the model about the current parameter estimates, solving the approximate linear least squares problem, and iterating in this manner until the parameter estimates no longer change. At this point a minimum of the sum of squares has been reached.

A recursive approximation to this algorithm is given by Soderstrom (1973) and Soderstrom *et al.* (1978).

2.6 CONCLUSIONS

Various predictive control algorithms have been introduced and proven with great success during the past two decades. It has also been indicated that the current MPC strategies have their own drawbacks. For example, DMC and MAC involve a great burden of calculation and also cannot work with unstable processes, the EPSAC has no flexibility on control horizon and the EHAC neither does it include disturbance in its design nor use the reference trajectory except at instant t+N. All the above strategies including the elegant GPC are based on linear process models. There are some non-linear control strategies but they also have their own drawbacks and are usually very difficult to implement.

Many industrial processes display non-linear behaviour. Not all these processes can be approximated in a satisfying way by a linear model. For this reason there is a great need for some good effective control strategies for processes which have a non-linear structure. In practice, measured process signals are also often overlapped by some coloured or white noise. It is clear then that there is a strong need for an easily applied non-linear control algorithm along the lines of the popular GPC. In this thesis it is proposed that the newly developed NLGPC helps to serve this need. Before the introduction of the NLGPC algorithm, an analytical discussion of the GPC algorithm will be presented in the next chapter.

CHAPTER 3

ANALYSIS OF THE GENERALIZED PREDICTIVE CONTROL ALGORITHM

3.1 INTRODUCTION

The long range predictive control (LRPC) design by Clarke et al. (1987) namely, General Predictive Control (GPC), has received a lot of attention and success since its presentation. Clarke's GPC has a number of attractive technical features:

- it can be applied to open-loop unstable and/or non-minimum phase processes.
- the stability and tracking properties are unaffected by stable or unstable pole-zero cancellations (Crisalle et al., 1989).
- it can handle unknown, variable time delays even in multi-variable applications (Shah et al., 1987).
- it provides offset free control for step inputs and performs well in presence of moderate

noise and disturbances with relatively simple filter predictors (Mohtadi, 1989; Wittenmark, 1989).

In this chapter the capabilities of the GPC are demonstrated as indicated in literature. All the properties discussed are verified by simulation. The relation between GPC with state space LQ control law is illustrated, that is, as the input and output control horizons tend to infinity the GPC automatically leads to infinity LQ control law. All the work given in this chapter is not new but is used later as a theoretical basis for the new NLGPC algorithm. Some of the examples are extracted from the book by Soeterboek (1992). Also similar work done by McIntosh *et al.* (1991).

3.2 GPC Theoretical Development

Clarke and Mohtadi (1989) used the following Controlled Auto-Regressive Integrated Moving Average (CARIMA) model to derive a GPC

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + \frac{T(q^{-1})}{\Delta}\xi(t) \tag{3.1}$$

where $\xi(t)$ is the disturbance, $T(q^{-1})$ is an observer/filtering polynomial and $\Delta=(1-q^{-1})$.

For purposes of model following polynomial $P(q^{-1})$ is introduced and thus the diophantine identity is written as

$$P(q^{-1})=E_j(q^{-1})A\Delta+q^{-1}F_j(q^{-1}) \tag{3.2}$$

Multiplying the model equation by E_jq^j and substituting for $E_jA\Delta$ gives

$$Py(t+j)=G_j\Delta u(t+j-1)+Fy(t)+E_jT\xi(t+j) \tag{3.3}$$

where $G_j=E_jB$. Since E_jT and ξ are uncorrelated then $E_jT\xi=0$.

For j = 1,2 ..N the above equation can be written in matrix form as

$$\hat{y}=Gu+f \tag{3.4}$$

where the vectors are N x 1:

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

$$u=[\Delta u(t),\Delta(t+1),\ldots,\Delta(t+N-1)]^T$$

$$f=[F_1y(t+1),F_2y(t+2),\ldots,F_Ny(t+N)]^T$$

and the matrix G is a lower triangular of dimension N x N.

In order to compute the vector of controls they propose the cost function J_{GPC} to be minimized

$$J_{GPC} = \sum_{j=N_1}^{N_2} [\hat{y}(t+j) - w(t+j)]^2 + \sum_{j=1}^{NU} \lambda_j (\Delta u(t+j-1))^2 \tag{3.5}$$

where N_1 is the minimum costing horizon, N_2 is the maximum costing horizon, NU is the control horizon, λ is the control weight and w is the reference trajectory.

The minimization of J_{GPC} (i.e by writing $dJ_{GPC}/du=0$) results in the increment vector:

$$\Delta u = (G^T G + \lambda I)^{-1} G^T (w - f) \tag{3.6}$$

so that the current control law is given by:

$$u(t) = u(t-1) + g^T (w - f) \tag{3.7}$$

where g^T is the first row of $(G^T G + \lambda I)^{-1} G^T$.

So, at each sampling interval:

(1) given y(t) and previous values of y, u, the predictions of freely responding process are computed and compared with future set-points w, which may be known or assumed equal to w(t).

(2) for the given {N1, N2, NU, λ, $\hat{y}$, w} the optimal control vector Δu is computed.

(3) the first element $u(t)=u(t-1)+\Delta u(t)$ is implemented and sequences are shifted ready for

the next sampling interval.

3.3 Relation of GPC with State Space LQ Design

Consider a plant of the form

$$A\Delta y(t)=B\Delta u(t-1) \tag{3.8}$$

Note that the disturbance is not included in order to consider stability and numerical properties. The discrete state space model of the above system can thus be written as

$$x(t+1)=ax(t)+b\Delta u(t) \tag{3.9}$$

$$y(t)=cx(t) \tag{3.10}$$

where a is the state transition matrix which is taken to be in observable canonical form (thus obtained from $A\Delta$), b is the vector of B parameters and polynomials P (the auxiliary model) and T (the observer or filtering polynomial) (see Clarke *et al.* (1985), and $c=[1,0,0,\dots,0]$.

Note that the number of states is max(deg($A\Delta$),deg(B)) and :

$a_i=0$ for $i >$ deg($A\Delta$)

$b_i=0$ for $i >$ deg(B)

The cost function in the state-space formulation can be written as:

$$J=\sum_{i=1}^{N_2}[x(t+i-1)^T Q x(t+i-1)+\lambda(t+i-1)\Delta u(t+i-1)^2] \tag{3.11}$$

where Q=c^{T}c.

The solution is obtained by iterating the algebraic Riccati equation (ARE) below.

Measurement update:

$$P^*(t)=P(i+1)-P(i+1)b[\lambda(i)+b^T P(i+1)b]^{-1}b^T P(i+1) \tag{3.12}$$

Time update:

$$P(i)=Q+a^T P^*(i)a \tag{3.13}$$

$$k=[\lambda(t)+b^T P(t)b]^{-1}b^T P(t)a \tag{3.14}$$

$$\Delta u(t)=-kx(t) \tag{3.15}$$

P is called the "covariance matrix". N$_2$ iterations are performed backwards starting from Q, the terminal covariance matrix. Note that since both the "one shot" (GPC) method (discussed earlier) of cost minimization and the dynamic programming approach of state-space (LQ) succeed in minimizing the same cost under certain conditions (i.e. linear system), the resulting control law must be the same because there is only one minimum and so their stability characteristics must be identical.

Note that fixing the projected control signal in the future is equivalent to employing a large penalty on the appropriate increment (i.e. $\lambda(i) \rightarrow \infty$). That means that the measurement update need not be updated for the particular values of i. Two special cases of GPC are considered below.

Lemma:

The closed loop system is stable if the system (a,b,c) is stabilizable (controllable) and detectable (observable) and if:

(i) $N_2 \to \infty$, $NU = N_2$ and $\lambda(i) > 0$ or

(ii) $N_2 \to \infty$, $NU \to \infty$ where $NU \leq N_2 - n + 1$ and $\lambda(i) = 0$

Proof:

Part (i): Is easily proven from the stability conditions of the state-space LQ controller. The cost function tends to infinite stage cost and for convergence to the algebraic Riccati equation (ARE) solution. Q can be positive semi-definite if λ is positive definite for all terminal covariances (Kwakernaak and Sivan, 1972). Part(ii): For the first n-1 iterations of the Riccati equation only the time updates are necessary. Note that the terminal covariance Q is of rank one. Each of the time-updates increases the rank of the matrix P by one and after n-1 iterations the matrix $P(N_2-n+1)$ is of full rank. It is seen that $P(N_2-n+1)$ is $\Pi a^{i\prime} c c^{\prime} a^{i}$ - the observability Grammian which is guaranteed positive definite for the structure assumed.

It therefore follows that as $NU \to \infty$ the iterations above converge to the ARE solution for all values $\lambda > 0$.

3.4 Control Law Implementation

In order to implement the control law above the following equation is proposed by Astrom and Wittenmark (1989) , Clarke *et al.* (1987):

$$R(q^{-1})\Delta u(t)=T(q^{-1})w(t)-S(q^{-1})y(t) \tag{3.16}$$

where R, S are polynomial associated with equation (3.7), u is the control effort, w is the set-point or reference signal and y is the system output. Equation (3.7) can be written as:

$$\Delta u(t)=h(w-F\hat{y}(t)-G\Delta u(t-1))$$

where h=[h_{N1} ... h_{N2}]. The above equation is equivalent to equation (7) with

$$T=T_{obs}+q^{-1}\left(\sum_{j=N_1}^{N_2} h_j G_j\right), \quad R=T_{obs}\left(\sum_{j=N_1}^{N_2} h_j\right), \quad S=\sum_{j=N_1}^{N_2} h_j F_j$$

The polynomials T, R and S are of degree max (degree(T_{obs}), degree(B)), degree(T_{obs}) and degree max(degree(A),degree(T_{obs})+ degree(P)-N1), respectively.

3.5 Simulation

The GPC can effectively be used to control most of the processes which are difficult to control with simpler controllers if it is properly tuned. It is not an easy task to tune a GPC controller since it has many tuning nobs, namely, the time horizons and the control weightii.g. It is the purpose of this simulation to get an idea how the elegant GPC can be tuned in the most optimal way and thus extend this knowledge to the new NLGPC algorithm. The influence of control horizon, NU, cost horizon, N2, observer polynomial, P, and control weighting, λ, on a closed loop system, for processes of different complexity is studied in this section.

3.5.1 The influence of NU on the closed-loop system.

Settings: $N1=1$, $N2=25$, $P=1$, $T=1$, $\lambda=0$

Process:

$$H(z^{-1})=\frac{0.029z^{-1}(1+0.928z^{-1})}{(1-0.882z^{-1})(1-0.905z^{-1})}$$

Model: Identical to Process

Parameters: NU=1,3

NU	R	S	T
1	$1+0.025q^{-1}$	$.925-0.7695q^{-1}$	.3595
3	$1+0.7235q^{-1}$	$41.5245-21.4832q^{-1}$	20.3839

Increasing NU makes the controller more active and the closed-loop system faster (see figures 3.1 and 3.2). For NU = 1 a mean level controller is approximated, while for NU =degree(A) + 1 a dead beat controller is obtained. The characteristic equations of a closed loop system ($AR+q^{-1}BS$) are given as:

$$1-1.7343q^{-1}+0.7543q^{-2}$$

and

$$1-0.1407q^{-1}-0.002q^{-2}-0.0007q^{-3}$$

for NU = 1 and NU = 3 respectively. For NU = 1 the roots of the characteristic equation are 0.8671 $\pm$ 0.0504i and zero. For NU = 3 roots are 0.0992 $\pm$ 0.0377i and 0.0578. Thus increasing NU leads to a more stable closed-loop system.

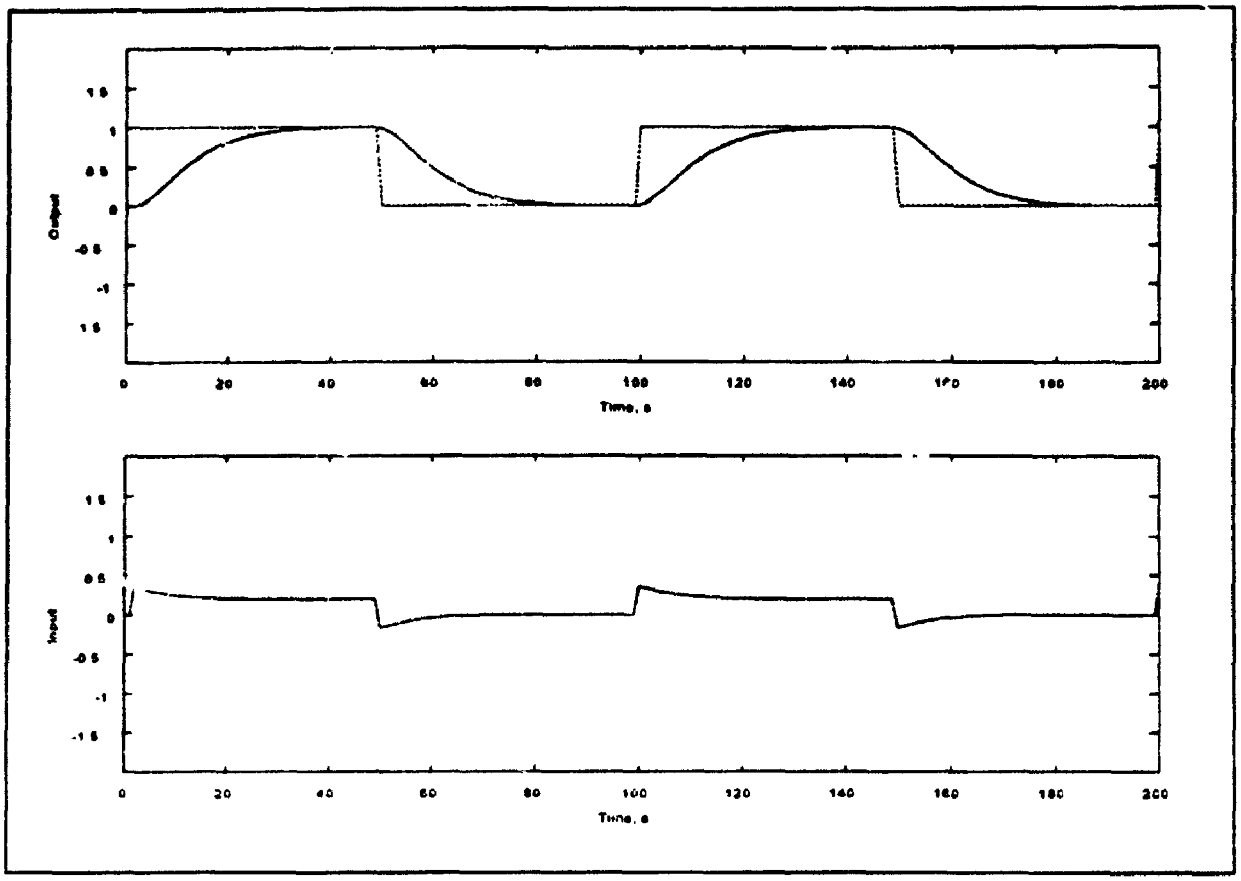

Figure 3.1: The effect of NU on the closed-loop system for NU=1

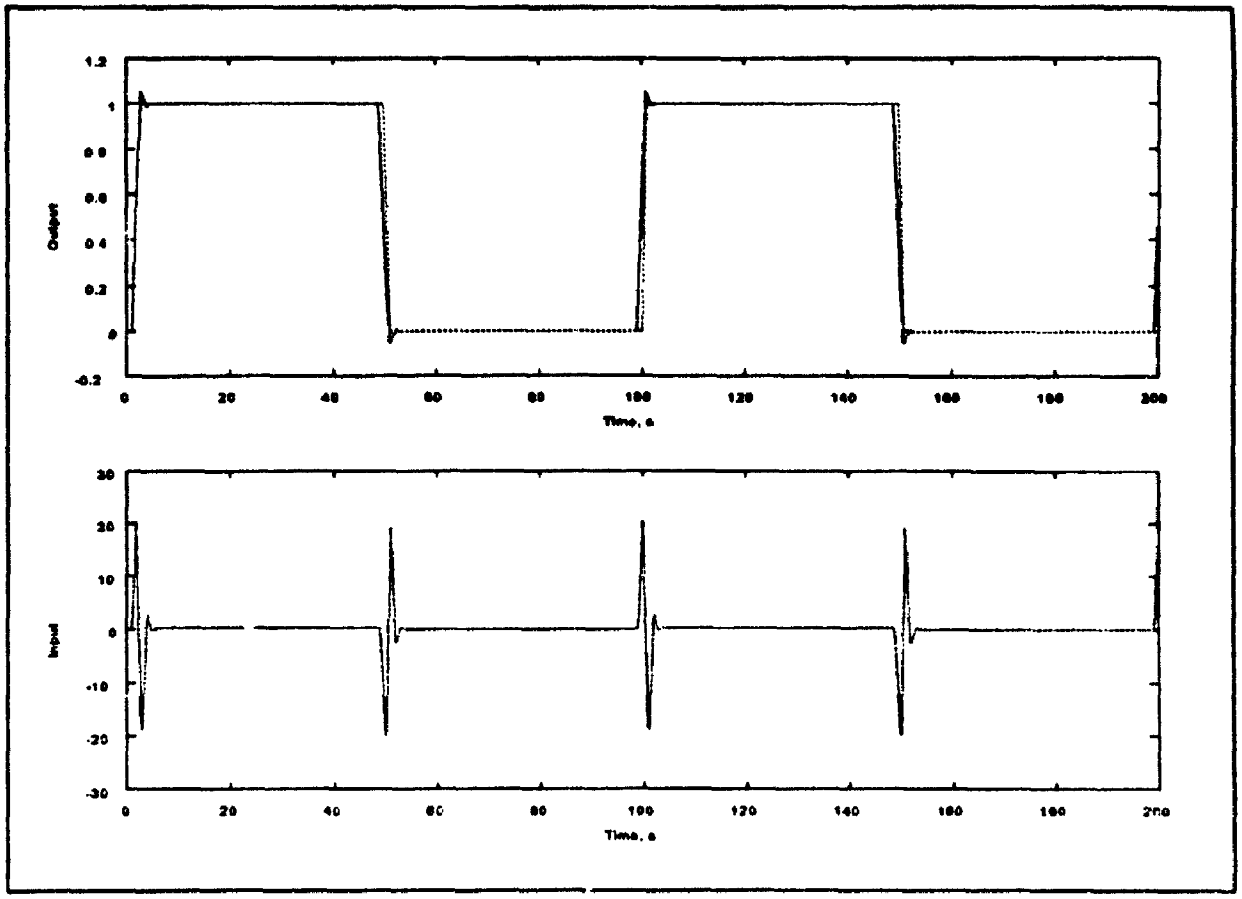

Figure 3.2: The effect of NU on the closed-loop system for NU=3

3.5.2 The influence of NU on the closed loop when N2=NU and $\lambda \neq 0$

Settings: N1=1,P=1, T=1, λ=0.01

Process: As in (1)

Model: Identical to the process

Parameters: N2=NU=2, 10

N2	NU	R	S	T
2	2	$1+0.2582q^{-1}$	$13.6483-7.6665q^{-1}$	6.1372
10	10	$1+0.3441q^{-1}$	$16.3551-10.2167q^{-1}$	6.4020

Suppose that, t_s, is the settling time of the closed-loop system for N2 $\rightarrow \infty$, at every set-point change. Then the closed-loop system is the same for all N2 $\geq t_s$, because in this case the contribution of the predicted process output $\hat{y}(k+i)$ and the controller output $u(k+i-1)$ for $i=t_s+1,\ldots,\infty$ to the criterion function is constant (see figures 3.3 and 3.4). With N2=NU $\rightarrow \infty$ an infinite LQ controller is approximated.

The characteristic equations of the closed-loop system are given as:

$$1-1.340q^{-1}+0.4825q^{-2}$$

and

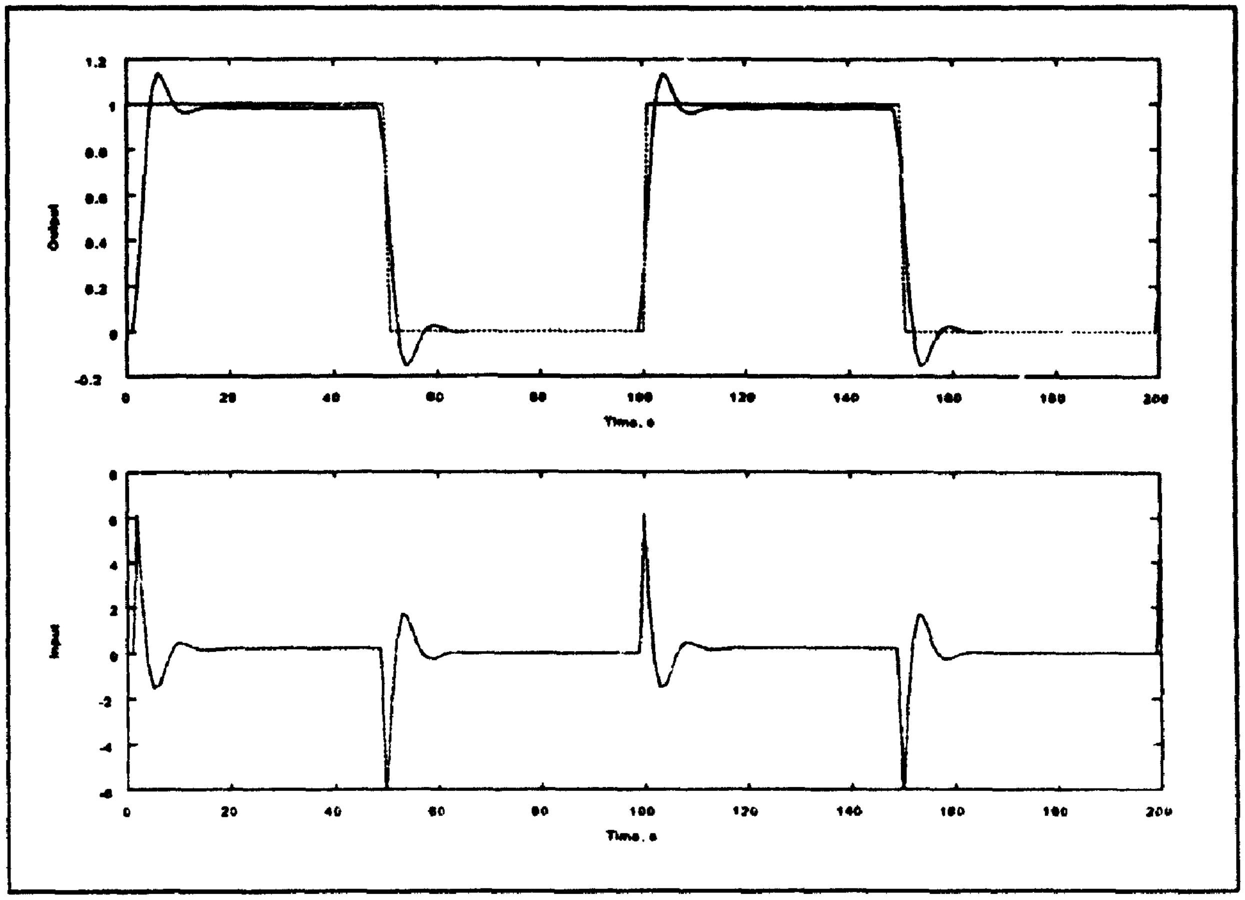

Figure 3.3: The effect of NU on the closed-loop system for N2=NU=2

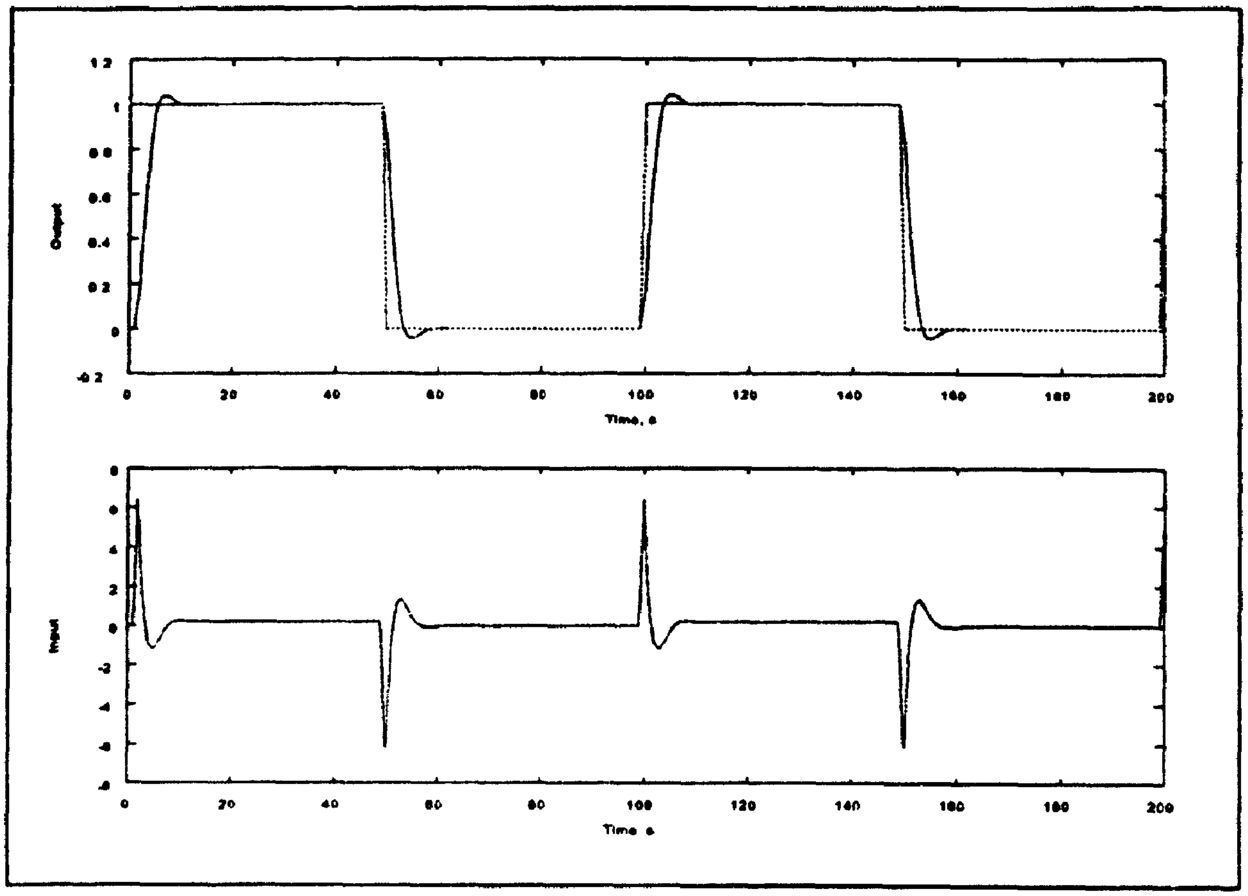

Figure 3.4: The effect of NU on the closed-loop system for N2=NU=10

$$1-0.9696q^{-1}+0.3278q^{-2}$$

for N2=NU=2 and N2=NU=10 respectively. For N2=NU=2 the roots of the characteristic equation are 0.5670 ± 0.4012i and zero. For N2=NU=10 roots are 0.4848 ± 0.3045i and zero.

3.5.3 The influence of N2 on the closed loop when NU=1 and λ=0

Settings:NU= 1, N1=1, P=1, T=1, λ=0
Process: As in (1)
Model: Identical to Process
Parameters: N2=2, 25

N2	R	S	T
2	$1+0.4790q^{-1}$	$24.9116-14.2242q^{-1}$	10.9815
25	$1+0.0259q^{-1}$	$0.925-0.7695q^{-1}$	0.3595

In this case, increasing the horizon makes the controller move from a constrained minimum variance (Katende and Jutan, 1993) to a mean-level controller. From figures 3.5 and 3.6, it is obvious that increasing N2 makes the closed-loop system respond slower to step-wise set-point changes.

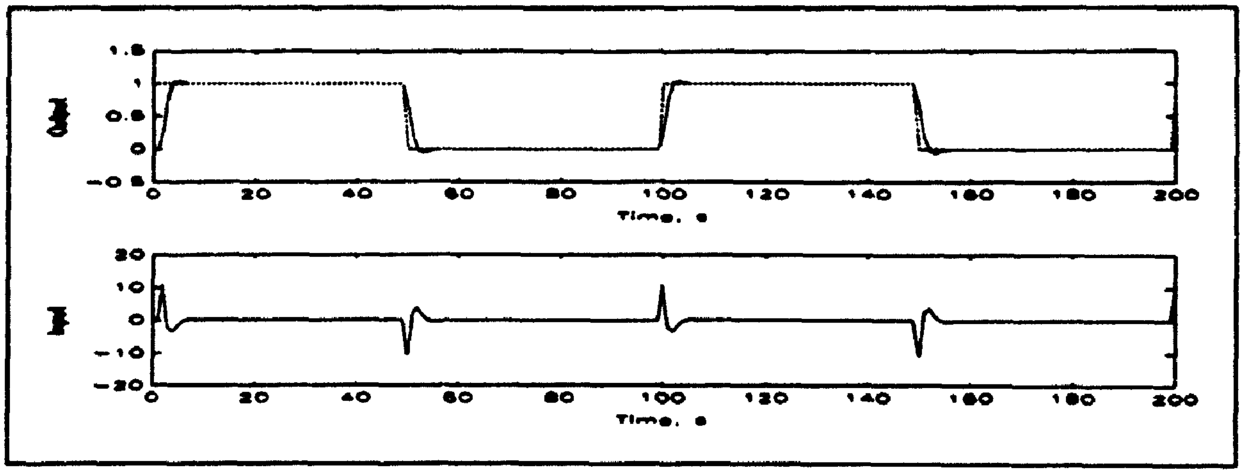

Figure 3.5: The effect of N2 on the closed loop system for N2=2

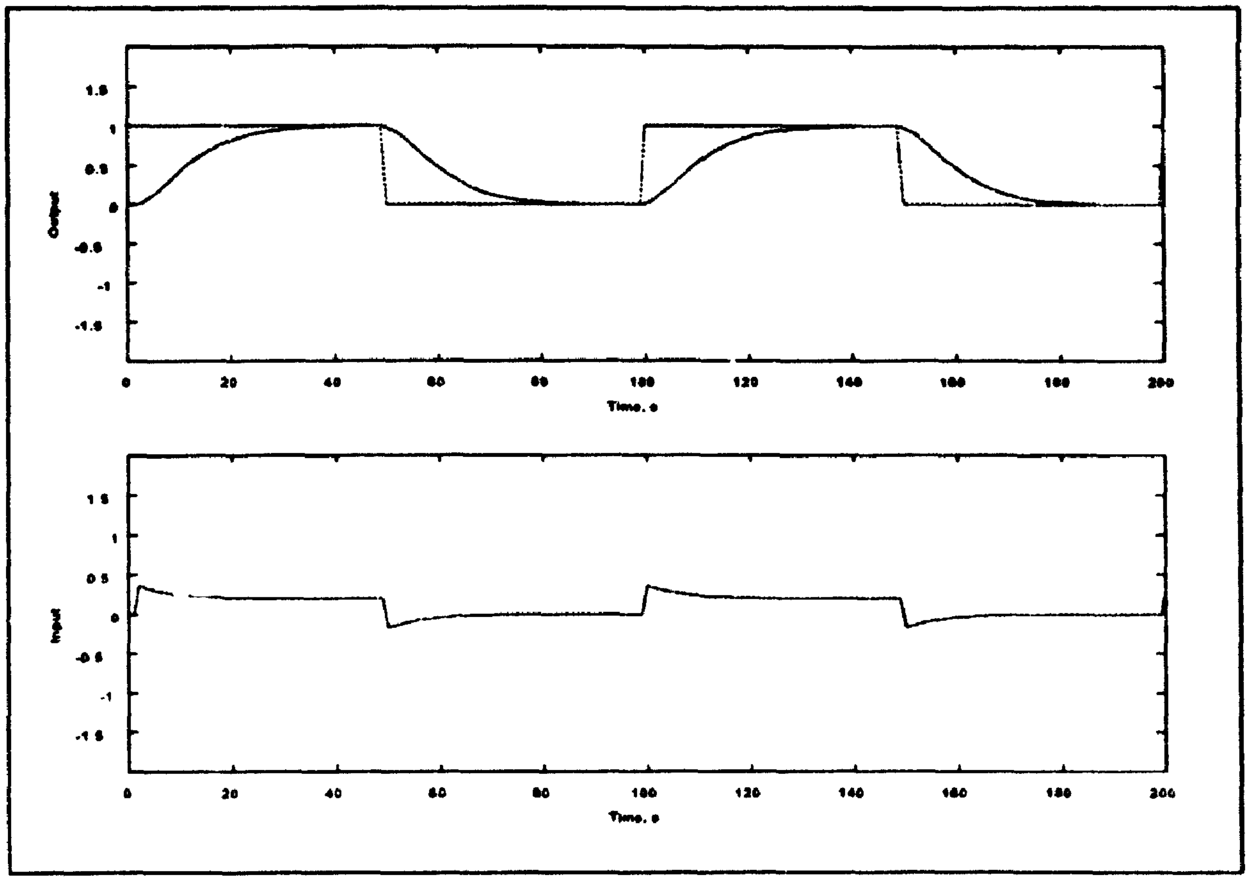

Figure 3.6: The effect of N2 on the closed-loop system for N2=25

This effect can easily be explained by the fact that the controller moves from a constrained minimum variance to a mean-level controller as N2 increases. When N2 → ∞ the step response of the closed loop system will be as slow as that of the open-loop system.

3.5.4 The influence of N2 on the closed-loop system in controlling a non-minimum phase process

Settings: NU=3, N1=1, P=1, T=1

Process:

$$H(z^{-1})=\frac{-0.5954z^{-1}(1-1.2269z^{-1})}{1-1.591z^{-1}+0.7261z^{-2})}$$

Model: Identical to the process

Parameters: N2=3, 4, 20

N2	R	S	T
3	$1-1.2269q^{-1}$	$-2.6722+1.2195q^{-1}$	-1.6795
4	$1-1.2188q^{-1}$	$-2.6611+1.2115q^{-1}$	-1.2284
20	$1+1.1160q^{-1}$	$0.5325-1.1093q^{-1}$	1.5393

From figures 3.7 and 3.8, it can be seen that the closed loop system is unstable for N2=3, 4. Increasing the N2 further makes the closed loop system stable. Figure 3.9 shows a situation when the closed loop of a non-minimum phase process is stabilized for N2=20.

The characteristic equations of a closed-loop system are given as:

$$1-1.2269q^{-1}$$

$$1-1.2254q^{-1}$$

$$1-0.7920q^{-1}$$

for N2=3, 4 and 20 respectively. For N2=3 the root of the characteristic equation is 1.2269, for N2=4 the root is 1.2254 and for N2=20 the root is 0.7920.

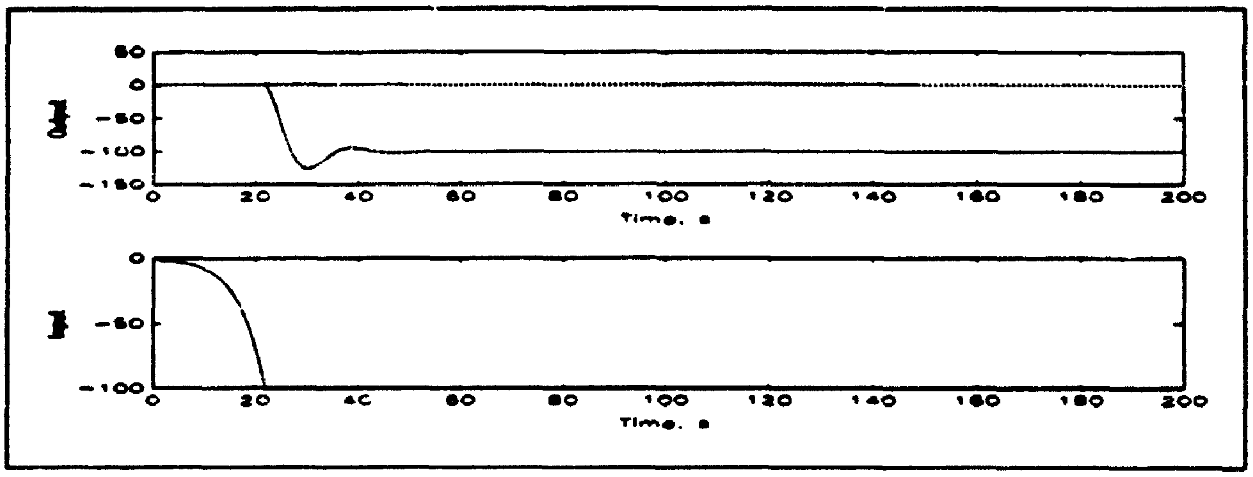

Figure 3.7: The effect of N2 on the closed-loop (NMP) system for N2=3

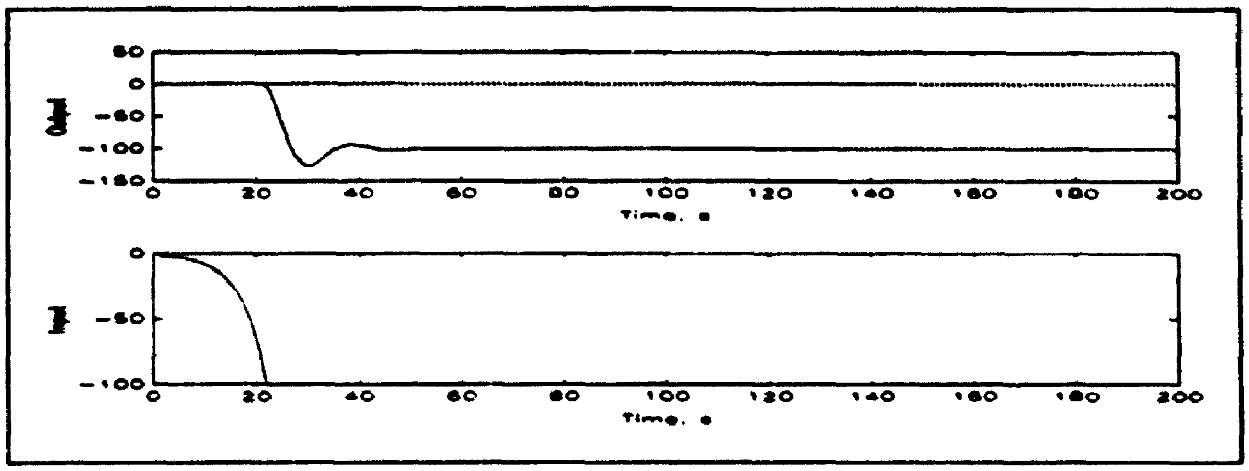

Figure 3.8: The effect of N2 on the closed loop (NMP) system for N2=4

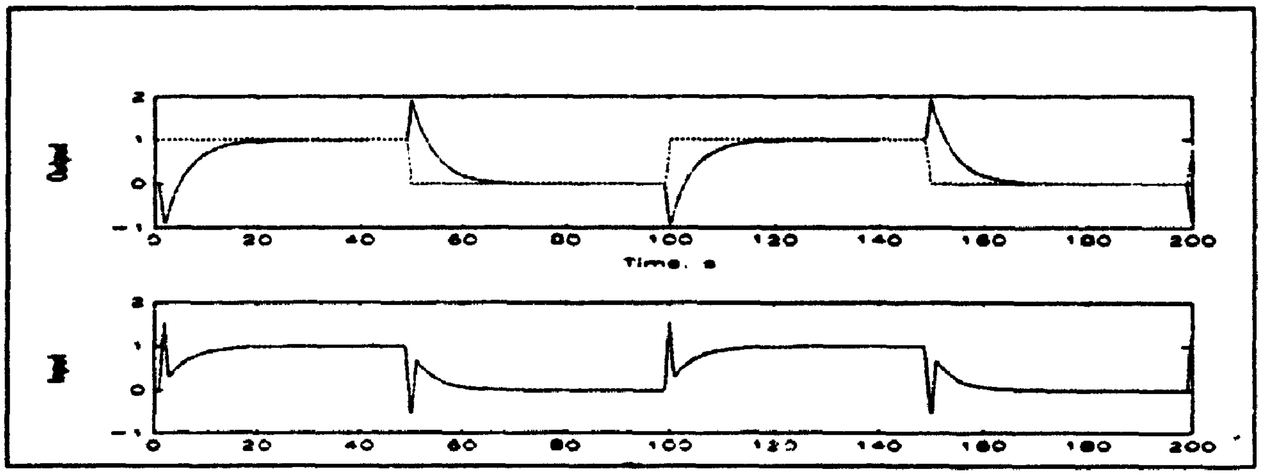

Figure 3.9: The effect of N2 on the closed loop (NMP) system for N2=20

3.5.5 Influence of P on the closed loop system

Settings: N1=2, N2=4, NU=3, T=1, $\lambda=0$

Process:

$$H(z^{-1})=\frac{0.029z^{-1}(1-0.928z^{-1})}{(1-0.0882z^{-1})(1-0.905z^{-1})}$$

Model:Identical to Process

Parameters: P=1, or P=1-0.85q^{-1}

P	R	S	T
1	1+0.6873q^{-1}	37.9568- 20.4070q^{-1}	17.8852
1-0.85q^{-1}	1+0.9078q^{-1}	59.6633- 26.9548q^{-1}	33.0877

Figures 3.10 and 3.11 show the effect of P on the closed loop system. By taking the roots of the characteristic equation (AR+q^{-1}BS), it is noticed that all the closed loop poles are at the origin in case of P=1. Thus, a dead-beat controller is selected. For the case of P=1-0.85q^{-1}, the closed-loop poles are at -0.85 and zero.

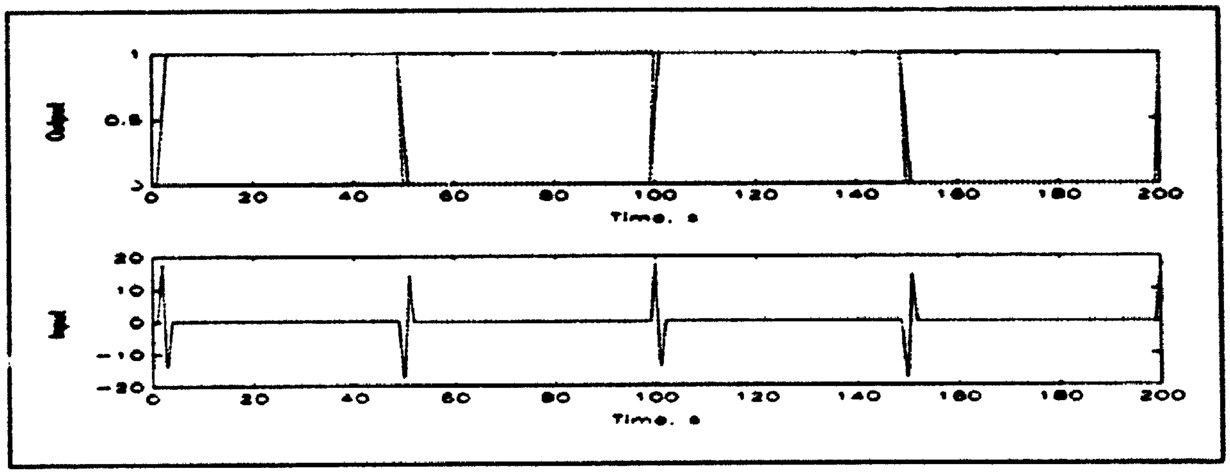

Figure 3.10: The effect of P on the closed-loop system for P=1

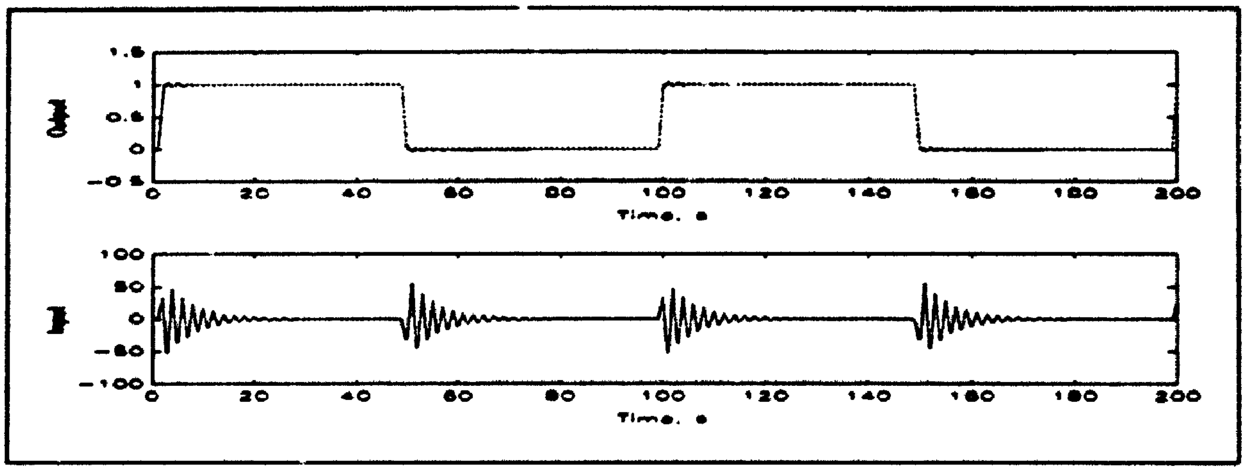

Figure 3.11: The effect of P on the closed-loop system for $P=1-0.85q^{-1}$

3.6 CONCLUSIONS

The GPC tends to a mean-level controller as $N2 \rightarrow \infty$ if $NU=1$, $N1=1$, $\lambda=0$ and $P=1$. A mean level controller provides a simple step in control following a step change in set-point which will drive the process output exactly to the new set-point at steady state. The use of polynomial $P(q^{-1})$ weighting allows GPC to achieve model-following. For exact model following the controller attempts to cancel the process zeros and it is therefore recommended to set $P=1$ in case of non-minimum phase processes. GPC is equivalent to stable state dead-beat controller if the system is observable and controllable and $NU=\text{degree}(A)+1$, $N2=\text{degree}(A)+\text{degree}(B)$, $P=1$, $\lambda=0$. State dead-beat control places all of the closed-loop poles at the origin. It has also been shown that GPC tends to infinite LQ control when $N2=NU \rightarrow \infty$.

CHAPTER 4

NON-LINEAR PREDICTIVE CONTROL

4.1 INTRODUCTION

As a group, adaptive controllers have focused primarily on linear process models. From a parameter update point of view, it is advisable to have the parameters in a linear space. However, from the process dynamics view point, linear dynamics can present a severe limitation for highly non-linear processes. Adaptive control applications based on linear dynamics are very numerous and the basic engineering theory and applications have been presented in, for example, Harris and Billings (1985) and Warwick (1988).

Multistep long range predictive controllers have been well received by industry. Early long-range multi-step predictive algorithms such as DMC (Cutler and Ramaker, 1980) were based on deterministic impulse or step response models. Long-range multi-step predictive controllers based on auto-regressive moving average (ARMA) models (De Keyer and Van Cauwenberghe, 1982; Peterka, 1984; Mosca *et al.*, 1984) were also developed and these led to fewer parameters in the model representation. These controllers could be made adaptive if combined with an appropriate recursive parameter estimation routine. The Generalized Predictive Control (GPC) algorithm (Clarke *et al.*, 1987 I & II) is a generalization of the previous multi-step predictive algorithms, as well as being the natural long-range extension of the well-known Generalized Minimum Variance (GMV) controller of Clarke and Gawthrop (1979). Garcia and Morari (1982) and Soeterboek (1992) showed that all model-based predictive schemes (DMC, PCA, MAC, IDCOM, GPC, EPSAC, EHAC) are structurally similar.

GPC has attractive features in that it can be applicable to :(1) non-minimum phase processes (2) open-loop unstable processes or processes with badly damped poles (3) processes with variable or unknown dead-time (4) processes with unknown order (5) multi-input, multi-output (MIMO) processes. The GPC is, however, based on linear process models.

For a large number of systems a linearizing approach is acceptable. However, for some non-linear systems linearizing can result in poor performance. Some of these highly non-linear systems include pH control in some chemical or biochemical processes, or thickness control for a rolling mill. In order to obtain improved control over these systems, while maintaining the benefits of self-tuning control techniques, it is necessary to take account of the non-linear dynamics in an appropriate manner.

Atherton (1975) and Cook (1986) indicate that because of the large number of different types of non-linearity which can occur in practice, extending a basic linear control scheme to account for all possibilities is unrealistic. A sensible way of tackling the general problem is to employ a framework within which a large number of non-linear processes can be adequately approximated. For self-tuning such a framework is provided by the Hammerstein model, Agarwal and Seborg (1987).

Agarwal and Seborg proposed two self-tuning control strategies for nonlinear control problems. Their strategies are applicable to a broad class of non-linear single-input, single-output systems which can include arbitrary nonlinear functions of the most recent input.

In this chapter a newly developed non-linear predictive control algorithm is introduced based on a process suitably modelled by a Hammerstein model. The controller can thus deal with both linear and non-linear systems and is developed in the framework of the GPC algorithm.

4.2 THEORETICAL DEVELOPMENT

4.2.1 Process Model

We consider non-linear and bilinear discrete-time process models described by Agarwal and Seborg (1987)

$$A(q^{-1})y(t)=\sum_{i=1}^{n} B_i(q^{-1})u^i(t-d-1)+B_{n+1}(q^{-1})u(t-d-1)y(t-1)+C(q^{-1})\xi(t) \qquad (4.1)$$

where t is the sampling instant, t=1,2,3...; d is the time delay and the extra delay is due to sample and hold; y(t) is the measured output at time t; u(t) is the manipulated input at time t; ξ(t) is zero-mean random noise independent of previous inputs and outputs; **A**, **B**$_i$

and C are polynomials given by

$$A(q^{-1})=1+a_1q^{-1}+\ldots+a_{n_A}q^{-n_A}$$

$$B_i(q^{-1})=b_{0i}+b_{1i}q^{-1}+\ldots+b_{n_{B_i}}q^{-n_{B_i}}$$

$$C(q^{-1})=1+c_1q^{-1}+\ldots+c_{n_C}q^{-n_C}$$

with the assumption that all roots of $C(q^{-1})$ lie inside the unit circle in the q-plane. Parameters n, n_A, n_B and n_C are known integers.

The following is a general form of discrete time Hammerstein model

$$A(q^{-1})y(t)=B(q^{-1})f(u(t)) \tag{4.5}$$

where f is a non-linear function, y is the process output. A, B are polynomials in the back shift operator q^{-1}. Equation (4.6) type of models is a special case of the non-linear model, equation (4.1), shown above with no bilinear term.

$$\begin{aligned} A(q^{-1})y(t)=B_1(q^{-1})u(t)+B_2(q^{-1})u^2(t)+ \\ \ldots B_n(q^{-1})u^n(t) + C(q^{-1})e(t) \end{aligned} \tag{4.6}$$

The advantage of the above models is that they are linear in the parameters, and thus can be easily estimated using, for example, recursive least squares.

4.2.2 A ger.eral form of NLGPC

4.2.2.1 Modelling and Prediction

Consider a process described by the following nth order Hammerstein model

$$A(q^{-1})y(t) = q^{-(d+1)}\sum_{j=1}^{n} B_j u^j(t) + \frac{C(q^{-1})}{\Delta}\xi(t) \qquad (4.7)$$

where y is the system output, u is the system input, ξ is the disturbance, A, B_j, C are polynomials, d is the dead-time (one is due to sample and hold) and Δ is $1\text{-}q^{-1}$.

We denote

$$\sum_{j=1}^{n} B_j(q^{-1})u^j(t) = x(t) \qquad (4.8)$$

where x(t) is a 'pseudo' input. A bilinear term can be included in the definition of x(t) when using a process model with a bilinear term.

We can write (4.7) as

$$A(q^{-1})y(t) = q^{-(d+1)}x(t) + \frac{C(q^{-1})}{\Delta}\xi(t) \qquad (4.9)$$

or

$$y(t) = \frac{q^{-(d+1)}}{A(q^{-1})}x(t) + \frac{C(q^{-1})}{\Delta A(q^{-1})}\xi(t) \qquad (4.10)$$

We denote the last term as, a noise or disturbance N(t), whence

$$N(t) = \frac{C(q^{-1})}{\Delta A(q^{-1})}\xi(t) \tag{4.11}$$

From (4.9) with the pseudo input x, y is linear in x and polynomials A and B_j can be estimated on-line from equation (4.7) using a recursive least squares method. C is usually set equal to T (a tuning polynomial) which is known as the observer polynomial.

Further if we let D=ΔA and C=T, the prediction of the stochastic disturbance, N, at t+i is given by

$$N(t+i) = \frac{T}{D}\xi(t+i) \tag{4.12}$$

In order to separate (4.12) into future and past terms, the following Diophantine equation is used

$$\frac{T}{D} = E_i + q^{-i}\frac{F_i}{D} \tag{4.13}$$

where E_i and F_i are polynomials with degrees

$n_{Ei} = i-1$

$n_{Fi} = \max(n_T - i, n_D - 1)$

Thus (4.12) can be written as

$$N(t+i) = E_i\xi(t+i)+\frac{F_i}{D}\xi(t) \tag{4.14}$$

In equation (4.14) the first term on the RHS consists of future noise only and is thus unknown. The second term can be calculated using the available data at time t.

Multiplying (4.10) by F_i/T, rearranging and using $D=\Delta A$ yields

$$\frac{F_i}{D}\xi(t) = \frac{F_i}{T}\left[y(t)-\frac{q^{-d}}{A}x(t-1)\right] \tag{4.15}$$

The i-step-ahead predictor for the process output (4.10) is thus given as

$$y(t+i) = \frac{q^{-d}}{A}x(t+i-1)+\frac{F_i}{T}\left[y(t)-\frac{q^{-d}}{A}x(t-1)\right]+E_i\xi(t+i) \tag{4.16}$$

which gives the minimum variance output prediction as

$$\hat{y}(t+i) = \frac{q^{-d}}{A}x(t+i-1)+\frac{F_i}{T}\left[y(t)-\frac{q^{-d}}{A}x(t-1)\right] \tag{4.17}$$

with the prediction error $\epsilon(t+i)$ given by

$$\epsilon(t+i) = E_i\xi(t+i)=y(t+i)-\hat{y}(t+i) \tag{4.18}$$

Since this prediction error consists of future noise only and because ξ(t) is assumed to be white noise with zero mean, the variance of the prediction error is constant.

We can rewrite (4.17) after applying the certainty equivalent principle as

$$\hat{y}(t+i) = \frac{q^{-d}}{A}x(t+i-1)+\frac{F_i}{T}[y(t)-\hat{y}(t)] \tag{4.19}$$

with

$$\hat{y}(t) = \frac{q^{d}}{A}x(t-1) \tag{4.20}$$

For the i-step-ahead predictor (4.19) the first term on the RHS has both future (unknown) and past (known) terms. The following Diophantine equation is used to separate (4.19) into future and past terms

$$\frac{1}{A} = G_i + q^{-i+d}\frac{H_i}{A} \qquad i \geq d+1 \tag{4.21}$$

Degree $G_i \leq$ i-d-1

Degree $H_i = n_A$-1,

Therefore G_i is given as

$$G_i = g_0 + g_1 q^{-1} + \ldots + g_{i-1} q^{-i+d+1} \tag{4.22}$$

Using (4.21) the i-step-ahead predictor becomes

$$\hat{y}(t+i) = G_i\, x(t+i-d-1) + \frac{H_i}{A} x(t-1) + \frac{F_i}{T}[y(t) - \hat{y}(t)] \tag{4.23}$$

The term $G_i x(t+i-d-1) = G_i \Sigma^{n}_{j=1} B_j u^j(t+i-d-1)$ contains only current and future controller outputs.

4.2.2.2 Matrix Notation

The i-step-ah~~ea~~d predictor (4.23) for $i=d+1,..,N_2$ can be written in matrix form as (Soeterboek, 1992)

$$\hat{y} = Gx + H\bar{x} + Fc \tag{4.24}$$

where

$$\hat{y} = [\hat{y}(t+d+1), \ldots, \hat{y}(t+N_2)]^T$$

$$x = [x(t), \ldots, x(t+N_2-d-1)]^T$$

$$c = [c(t), c(t-1), \ldots]^T$$

with

$$\bar{x}(t) = \frac{x(t)}{A}$$

$$c = \frac{y(t) - \hat{y}(t)}{T}$$

The G, H, F matrices are constructed as follows:

$$G = \begin{bmatrix} g_0 & 0 & \cdots & 0 \\ g_1 & g_0 & \cdots & \vdots \\ \vdots & & \ddots & 0 \\ g_{N_2-d-1} & \cdots & & g_0 \end{bmatrix}$$

$$H = \begin{bmatrix} H_{d+1} \\ \vdots \\ H_i \\ \vdots \\ H_{N_2} \end{bmatrix}$$

$$F = \begin{bmatrix} F_{d-1} \\ \vdots \\ F_i \\ \vdots \\ F_{N_2} \end{bmatrix}$$

This matrix form will be used later to develop the control law.

4.2.2.3 Predictive Control Law

In order to obtain the control law the following GPC style cost function is minimized

$$J = \sum_{i=N_1}^{N_2} [P\hat{y}(t+i) - P(1)w(t+i)]^2 + \lambda \sum_{i=1}^{Nu} (x(t+i-1))^2 \qquad (4.33)$$

where P is a monic polynomial and

$$P(1) = \sum_{j=0}^{n_p} P_j$$

λ is the control weight, w is the set-point. Note that the penalty is on the 'pseudo input' x. This is appropriate if we consider that for non-linear processes, the usual input u, often enters indirectly as a function of u, i.e x, and it is this 'net effect' of u that we wish to penalize.

If the cost function J is minimized with respect to vector x, then any local optimum for x satisfies

$$\frac{\partial J}{\partial x} = 0$$

Since x is a function of u, the optimal u can be calculated from optimal x.

In matrix form we can write the cost function as

$$J = (\hat{y}^* - w^*)^T(\hat{y}^* - w^*) + \lambda x^T x \tag{4.36}$$

where

$$w^* = [P(1)w(t+N_1),\ldots,P(1)w(t+N_2)]^T$$

$$\hat{y}^* = [P\hat{y}(t+N_1),\ldots,P\hat{y}(t+N_2)]^T$$

$$x = [x(t),\ldots,x(t+Nu-1)]^T$$

Now, for (4.36) the gradient with respect to **x** is

$$\frac{\partial J}{\partial x} = 2\frac{\partial \hat{y}^*}{\partial x}(\hat{y}^* - w^*) + 2\lambda x \qquad (4.40)$$

Repeating equation (4.24) gives

$$\hat{y}^* = Gx + H\bar{x} + Fc \qquad (4.41)$$

Differentiating we have

$$\frac{\partial \hat{y}^*}{\partial x} = G^T \qquad (4.42)$$

Therefore

$$\frac{\partial J}{\partial x} = 2G^T(Gx + H\bar{x} + Fc - w^*) + 2\lambda x \qquad (4.43)$$

or

$$\frac{\partial J}{\partial x} = 2(G^TG+\lambda I)x+2G^T(H\bar{x}+Fc-w^*) \tag{4.44}$$

Setting (4.44) to zero, we get

$$x = (G^TG+\lambda I)^{-1}G^T(w^*-H\bar{x}-Fc) \tag{4.45}$$

Equation (4.45) gives the optimal x vector, from which as in GPC, we require only the first element to implement the control law.

We obtain the current x value at time t (first element of x) by the following equation

$$x(t) = v^Tw^*-h^T\bar{x}-f^Tc \tag{4.46}$$

where

$$v^T=\chi^TG^T \qquad 1 \times (N_2-N_1+1) \tag{4.47}$$

$$\chi^T = e_1^T R_v^{-1} \qquad 1 \ x \ (N_2 - \hat{d}) \tag{4.48}$$

$$e_1^T = [1,0,\ldots,0]^T \qquad 1 \ x \ Nu \tag{4.49}$$

$$R_v = (G^T G + \lambda I) \qquad Nu \ x \ Nu \tag{4.50}$$

$$h^T = \nu^T H \qquad 1 \ x \ n_H + 1 \tag{4.51}$$

$$f^T = \nu^T F \qquad 1 \ x \ n_F + 1 \tag{4.52}$$

Once v, **h** and **f** are obtained, the current optimal value of x can be calculated.

4.2.2.4 Computation of the optimal u

From (4.8) we know that once the optimal pseudo input x is known we then go ahead to compute the process input u

$$\sum_{i=1}^{n} B_i u^i = x \tag{4.53}$$

We can write (4.53) as

$$\phi = \sum_{j=1}^{n} b_{0j} u_{opt}^{j} + (B_j - b_{0j})\bar{u}^{j} - x = 0 \tag{4.54}$$

where ū involves only the past u values.

In order to obtain u_{opt}, Newton's method can be used. The gradient of ϕ with respect to u_{opt} is given as follows

$$\phi' = \frac{\partial \phi}{\partial u_{opt}} = \sum_{j=1}^{n} b_{0j} \frac{\partial u_{opt}^{j}}{\partial u_{opt}} \tag{4.55}$$

The optimal control u_{opt} is obtained by iteration of the following equation

$$u_{opt}(m+1) = u_{opt}(m) - \frac{\phi}{\phi'} \tag{4.56}$$

It is possible to derive a direct algorithm for second order non-linearities. This is detailed in the following section.

4.2.3 Quadratic NLGPC Algorithm

For this algorithm we minimize the cost function with respect to the input directly. An analytical solution could be obtained by setting the Jacobian to zero or a singe step numerical solution can be obtained solving the Jacobian and the Hessian iteratively. This algorithm is found to be very time efficient through simulation.

4.2.3.1 Modelling and Prediction

A second order non-linear Hammerstein model with non-stationary noise can be written in the well known CARIMA form (with non-linearity in u)

$$A(q^{-1})y(t) = B_1(q^{-1})u(t-1) + B_2(q^{-1})u^2(t-1) + \frac{T(q^{-1})}{\Delta}\xi(t) \tag{4.57}$$

where $\xi(t)$ is the disturbance, $T(q^{-1})$ is an observer/filter polynomial and $\Delta=(1-q^{-1})$.

The polynomial $P(q^{-1})$ is introduced, and this gives rise to the diophantine identity

$$T(q^{-1})P(q^{-1})=E_j(q^{-1})A\Delta+q^{-1}F_j(q^{-1}) \tag{4.58}$$

where $j=1,N$ and N=prediction horizon.

Multiplying the model equation by E_jq^j and substituting for $E_jA\Delta$ gives

$$T(q^{-1})Py(t+j) = \sum_{i=1}^{2} G_{ij}\Delta u^i(t+j-1) + Fy(t) + E_jT\xi(t+j) \tag{4.59}$$

where $G_{ij}=E_jB_i$. We set all future, $E_jT\xi(t+j)$ values to zero for minimum variance prediction

For j=1,2 ..N equation (4.59) can be written in matrix form as

$$\hat{y}=\sum_{i=1}^{2} G_i u^{i}+f \tag{4.60}$$

where the vectors ŷ, **u**, **u**², **f** are N x 1 and setting T=1:

$$\hat{y}=[P\hat{y}(t+1),P\hat{y}(t+2),\ldots,P\hat{y}(t+N)]^T \tag{4.61}$$

$$u=[\Delta u(t),\Delta u(t+1),\ldots,\Delta u(t+N-1)]^T \tag{4.62}$$

$$u^2=[\Delta u^2(t),\Delta u^2(t+1),\ldots,\Delta u^2(t+N-1)]^T \tag{4.63}$$

$$\begin{gathered} f = [(G_1-g_0)\Delta u(t) + F_1 y(t),(G_2-g_2-g_0)\Delta u(t+1) \\ F_2 y(t+1),\ldots, G_N-g_N-\ldots-g_0)\Delta u(t+N-1) + F_N y(t+N-1) \end{gathered} \tag{4.64}$$

and the matrix G is a lower triangular of dimension N x N.

4.2.3.2 Predictive control law - Quadratic Model

In order to compute the vector of controls the following cost function J is minimized

$$J = \sum_{j=N_1}^{N_2} [\hat{y}(t+j)-w(t+j)]^2 + \sum_{j=1}^{NU} \lambda_j[\Delta u(t+j-1)]^2 \tag{4.65}$$

where N_1 is the minimum cost horizon, N_2 is the maximum cost horizon, NU is the control horizon, and λ is the control weight. Note that the cost function is identical to that defined by Clarke *et al* (1987) in the derivation of the GPC algorithm.

In matrix form we write

$$J = E\{(\hat{y} - w)^T(\hat{y}-w) + \lambda u^T u\} \tag{4.66}$$

or

$$J = E(G_1u+G_2u^2+f-w)^T(G_1u+G_2u^2+f-w) + \lambda u^T u \tag{4.67}$$

Let us denote

$$v=G_1u+G_2u^2+f-w \tag{4.68}$$

where v is a column vector. Equation (4.67) becomes

$$J=E\{v^Tv+\lambda u^Tu\} \tag{4.69}$$

Let the $\mathbf{G}_1$ and $\mathbf{G}_2$ matrices be represented by column vectors $\mathbf{g}_0$, $\mathbf{g}_1$, $\mathbf{g}_2$,... , that is:

$$G_1 = [g_0^{(1)}\ g_1^{(1)}\ g_2^{(1)}\ ...] \tag{4.70}$$

$$G_2 = [g_0^{(2)}\ g_1^{(2)}\ g_2^{(2)}\ ...] \tag{4.71}$$

4.2.3.3 Objective Minimization

For optimal control we minimize J in equation (4.69) with respect to **u**. This is achieved in several steps:

$$\frac{\partial J}{\partial u_i}=v^T\frac{\partial v}{\partial u_i}+\frac{\partial v^T}{\partial u_i}v+2\lambda u^T\frac{\partial u}{\partial u_i} \tag{4.72}$$

or

$$\frac{\partial J}{\partial u_i} = 2v^T\frac{\partial v}{\partial u_i} + 2\lambda u_i \tag{4.73}$$

where v can be written as a sum of outer products

$$\begin{aligned} v = & \; u_0 g_0^{(1)} + u_1 g_1^{(1)} + \ldots + u_i g_i^{(1)} + \ldots + u_{N-1} g_{N-1}^{(1)} \\ & + u_0^2 g_0^{(2)} + \ldots + u_i^2 g_i^{(2)} + \ldots + u_{N-1}^2 g_{N-1}^{(2)} \; + f + w \end{aligned} \tag{4.74}$$

N-1 is the number of columns of G_1 and G_2 matrices.

We now express the partial derivative of v with respect to u_i as linear combinations of the columns of G_1 and G_2

$$\frac{\partial v}{\partial u_i} = g_i^{(1)} + 2u_i g_i^{(2)} \tag{4.75}$$

which is a column vector. Multiplying both sides of equation (4.75) by $2v^T$ we obtain

$$2v^T \frac{\partial v}{\partial u_i} = 2v^T g_i^{(1)} + 42u_i v^T g_i^{(2)} \tag{4.76}$$

which is a scalar.

We can now obtain the partial derivative of J with respect to u as the row vector

$$\frac{\partial J}{\partial u} = \left[\frac{\partial J}{\partial u_i} \; \frac{\partial J}{\partial u_i} \; \cdots \; \frac{\partial J}{\partial u_{N-1}} \right] \tag{4.77}$$

That is

$$\frac{\partial J}{\partial \boldsymbol{u}}=[2\boldsymbol{v}^T g_0^{(1)}\ 2\boldsymbol{v}^T g_1^{(1)}\ \dots\ 2\boldsymbol{v}^T g_{N-1}^{(1)}] + [4u_0\boldsymbol{v}^T g_0^{(2)}\ 4u_1\boldsymbol{v}^T g_1^{(2)}\ \dots\ 4u_{N-1}\boldsymbol{v}^T g_{N-1}^{(2)}] + [2\lambda u_0\ 2\lambda u_1 \dots 2\lambda u_{N-1}] \tag{4.78}$$

or

$$\frac{\partial J}{\partial \boldsymbol{u}} = 2\boldsymbol{v}^T G_1 + 2\lambda \boldsymbol{u}^T + 4\boldsymbol{v}^T[u_0 g_0^{(2)}\ u_1 g_1^{(2)}\ \dots\ u_{N-1} g_{N-1}^{(2)}] \tag{4.79}$$

Finally, we can rewrite the above equation in a more compact form as

$$\frac{\partial J}{\partial \boldsymbol{u}} = 2\boldsymbol{v}^T G_1 + 4\boldsymbol{v}^T G_2 diag(\boldsymbol{u}) + 2\lambda \boldsymbol{u}^T \tag{4.80}$$

The gradient in (4.80) can also be used to calculate the Hessian matrix directly and used for function minimization in say, Newton's method.

The Hessian of J with respect to **u** is written as

$$\frac{\partial^2 J}{\partial u^2} = \begin{bmatrix} \frac{\partial^2 J}{\partial u_0^2} & \frac{\partial^2 J}{\partial u_0 \partial u_1} & \cdots & \frac{\partial^2 J}{\partial u_0 \partial u_{N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial u_{N-1} \partial u_0} & \cdots & \cdots & \frac{\partial^2 J}{\partial u_{N-1}^2} \end{bmatrix} \tag{4.81}$$

After much algebraic manipulation (see appendix),the Hessian can be written as

$$\begin{aligned}\frac{\partial^2 J}{\partial u^2} &= 2G_1^T G_1 + G_1(G_2 diag(u)) + 2(G_2 diag(u))^T G_1 \\ &+ 4(G - 2diag(u))^T (G_2 diag(u)) + 2\lambda I + 4diag(v^T G_2)\end{aligned} \tag{4.82}$$

As is usually the case with receding horizon schemes, the current control law is given as the first element of vector u, obtained formally as a solution to (4.80) by equating to zero or numerically, using (4.80) and (4.82) iteratively.

If we denote

$$\phi = \frac{\partial J}{\partial u_{opt}}; \qquad \phi' = \frac{\partial J^2}{\partial u_{opt}^2}$$

the optimal value of current controller output, u(t), can be obtained by iteration of the following equation

$$u_{opt}(n+1) = u_{opt}(n) - \frac{\phi}{\phi'}$$

This algorithm can now be used to calculate the optimal control vector u_{opt} for which we implement the first element only. This algorithm is based on a process described by a second order Hammerstein model and using a criterion function given by equation (4.67) in the controller design.

4.2.4 Performance of NLGPC at Steady State

Prediction error:

The i-step ahead predictor for the process output is

$$y(t+i) = \frac{q^{-(d+1)}}{A(q^{-1})}x(t+i)+\frac{C(q^{-1})}{\Delta A(q^{-1})}\xi(t+i) \tag{4.85}$$

Multiplying (4.85) by F_i/T, rearranging yields

$$\frac{F_i}{\Delta A}\xi(t+i) = \frac{F_i}{T}\left[y(t+i)-\frac{q^{-d}}{A}x(t+i-1)\right]$$

Multiplying the above equation by E_i and rearranging we obtain the prediction error

$$E_i\xi(t+i) = \frac{\Delta AE_i}{T}\left[y(t+i)-\frac{q^{-d}}{A}x(t+i-1)\right]$$

We allow parameter estimates for A and B_i and substitute for y(t+i) with equation (4.85) we get

$$E_i\xi(t+i) = \frac{\Delta AE_i}{T}\left[\frac{q^{-d}}{A}x(t+i-1)-\frac{q^{-\hat{d}}}{\hat{A}}\hat{x}(t+i-1)+N(t+i)\right] \tag{4.88}$$

as the prediction error.

Prediction error elimination:

Soeterboek (1992) indicates that for a stable system driven by a signal v(t) that satisfies $f_\xi\,(q^{-1})v(t)=0$ as t tends to infinity, then all signals $v_i(t)$ in the system $f_\xi\,(q^{-1})v_i(t) =0$ (where f_ξ is a stable polynomial). Soeterboek considered only linear cases. We can extend the same argument to our system.

Let the disturbance, as t tends to infinity, satisfy:

$$f_\xi\, N(t) = 0 \tag{4.89}$$

where f_ξ is a polynomial.

For a stable closed-loop system, as t tends to infinity, we have

$$f_\xi\, y(t) = 0 \tag{4.90}$$

If f_ξ is a factor of ΔA then the prediction error at steady state is equal to zero, since prediction error as given by equation (4.88) is

$$E_i\xi(t+i) = \frac{\Delta A E_i}{T}\left[\frac{q^{-d}}{A}x(t+i-1)-\frac{q^{-d}}{\hat{A}}\hat{x}(t+i-1)+N(t+i)\right]$$

Furthermore, Soeterboek (1992) indicates that if the system is stable and driven by disturbance N(t) and reference trajectory w(t) which in steady state satisfy $f_\xi N(t)=0$ and $f_w w(t)=0$ where f_w is a polynomial. Then y(t) and u(t) (linear system) at steady state satisfy:

$$f_m\, y(t)=0 \qquad ; \qquad f_m\, u(t)=0 \tag{4.92}$$

where f_m = min{ f_ξ , f_w }(i.e. f_m is the polynomial with smallest degree for which $f_m w(t)=0$ and $f_m N(t)=0$.

For our system assuming the system is stable and driven by disturbance N(t) and reference trajectory w(t) at steady state we have $f_\xi N(t)=0$ and $f_w w(t)=0$. At steady state we can therefore write:

$$f_m\, y(t)=0 \qquad ; \qquad f_m\, x(t)=0 \tag{4.93}$$

where f_m = min{ f_ξ, f_w }

If f_m is a factor of ΔA then for a stable closed-loop system, the steady state errors do not occur irrespective of modelling errors.

We also know that

$$N(t) = \frac{C(q^{-1})}{\Delta A(q^{-1})}\xi(t) \tag{4.94}$$

Let C=1, then

$$\Delta A(q^{-1})N(t) = \xi(t) \tag{4.95}$$

where $\xi(t)$ is white noise with zero mean. From equation (4.89) and (4.95) we can say that f_ξ is a factor of ΔA and from (4.90) and (4.94) we can say that f_m is a factor of ΔA hence confirming that the NLGPC has no steady state errors irrespective of any presence of modelling errors.

4.2.5 Performance of GPC at steady state

For the GPC prediction error is given as

$$E_t\xi(t+i) = \frac{\Delta A E_t}{T}\left[y(t+i) - \frac{q^{-d}B}{A}u(t+i-1)\right]$$

If the parameters A and B are estimated and since we simulated non linear Hammerstein process models, we substitute for y(t+i) and we thus get

$$E_t\xi(t+i) = \frac{\Delta A E_t}{T}\ [\ \frac{q^{-d}B_1}{A}u(t+i-1) + \frac{q^{-d}B_2}{A}u^2(t+i-1) + \ldots$$
$$- \frac{q^{-d}\hat{B}}{\hat{A}}u(t+i-1)\ +\ N(t+i)\ ]$$

as the prediction error. Using the same argument as we used above for the NLGPC case, if f_ξ is a factor of ΔA, GPC will eliminate the steady state prediction error if and only if the process output y(t+i) is a linear process. However, for our case we have a non-linear Hammerstein process models which leads to the presence of the non-linear u-terms in the prediction error equation. For this case f_ξ being a factor of ΔA does not guarantee elimination of prediction error.

4.3 CONCLUSIONS

A general form of Non-linear Generalized Predictive Control (NLGPC) has been derived based on processes modelled by a non-linear and bilinear model structure which includes Hammerstein model structure with a general non-linear input signal.

In order to define how well the predicted process output tracks the set-point, a cost function is used. Typically, such a cost function is a function of predicted process output, set-point (reference trajectory), and process output. The simplest cost function that can be used for predictive controller design is :

$$J = \sum_{j=N_1}^{N_2} [\hat{y}(t+j)-w(t+j)]^2+\sum_{j=1}^{NU} \lambda_j[\Delta u(t+j-1)]^2 \qquad (4.98)$$

Minimization of J yields a controller which minimizes the tracking error between the predicted process output and the reference trajectory. If there is no model mismatch and / or no constraints, the process will track the reference trajectory exactly on the sampling instants. If there is a time delay, d, the minimum cost horizon, N1, is set to d+1.

The optimal value of 'x' over the prediction horizon is obtained analytically by minimization of J with respect to 'x', see equation (4.33). The optimal u is obtained numerically from the optimal 'x'. As a result the future tracking error is minimized with all elements of u penalized. The subsequent numerical search for u can lead to ı.. .tiple solutions, some of which are clearly inadmissible, such as complex or wrong sign solutions.

A second order Non-linear Generalized Predictive Control algorithm has also been derived based on processes modelled by the second order Hammerstein model structure. The usual cost function is mnimized to obtain the control law. Evaluation of these algorithms is treated in the following two chapters.

CHAPTER 5

EFFECT OF COST FUNCTION ON THE PERFOMANCE OF NLGPC

5.1 INTRODUCTION

Although adaptive control of non-linear processes has been studied in the past, few researchers have considered modifying the criterion function used in controller design to improve the performance. In predictive control, minimization of a criterion function yields the predictive control law, therefore, the choice of the criterion function is of great importance. In the past modification of the criterion function has been avoided by focussing primarily on linear processes, and quadratic objective functions.

In this chapter the effect of different criterion functions on the performance of non-linear predictive control strategies is explored. Here two non-linear predictive control algorithms are explored using a second order Hammerstein model structure. For the first algorithm a conventional quadratic criterion function is used with no modifications. A different and probably more appropriate cost function is used, for a second algorithm in which analytical expressions for the gradients are obtained. Lastly, the Non-linear Generalized Predictive Control (NLGPC) algorithm developed in chapter 4 is briefly discussed.

5.2 THEORETICAL DEVELOPMENT

5.2.1 Penalizing the Process Input

A quadratic cost function is mostly chosen as a mathematical convenience. Nevertheless, it is also somewhat intuitive for linear process models with constrained control. However, when we consider non-linear models of the Hammerstein form, a number of different options (within the basic quadratic functional form) are possible. For a general non-linear input, do we weight just the square of the linear term u - what if there is no linear terms in the process model? This leads us to the concept of the 'pseudo' input x. It now makes sense to penalize the square of x which in turn contains the manipulated variable u as it appears in the process dynamics.

5.2.1.1 Modelling and Prediction

A second order non-linear Hammerstein model with non-stationary noise can be written in the well known CARIMA form (with non-linearity in u)

$$A(q^{-1})y(t) = B_1(q^{-1})u(t-1) + B_2(q^{-1})u^2(t-1) + \frac{T(q^{-1})}{\Delta}\xi(t) \qquad (5.1)$$

where $\xi(t)$ is the disturbance, $T(q^{-1})$ is an observer/filter polynomial and $\Delta=(1-q^{-1})$.

The polynomial $P(q^{-1})$ is introduced, and this gives rise to the diophantine identity

$$T(q^{-1})P(q^{-1})=E_j(q^{-1})A\Delta+q^{-1}F_j(q^{-1}) \tag{5.2}$$

where $j=1,N$ and $N=$prediction horizon.

Multiplying the model equation by E_jq^j and substituting for $E_jA\Delta$ gives

$$T(q^{-1})Py(t+j) = \sum_{i=1}^{2} G_{ij}\Delta u^i(t+j-1) + Fy(t) + E_jT\xi(t+j) \tag{5.3}$$

where $G_{ij}=E_jB_i$. We set all future$_j$ $ET\xi(t+j)$ values to zero for minimum variance prediction

For $j=1,2$ ..N equation (5.3) can be written in matrix form as

$$\hat{y}=\sum_{i=1}^{2} G_i u^i+f \tag{5.4}$$

where the vectors $\hat{y}$, u, u^2, f are N x 1 and setting $T=1$:

$$\hat{y}=[P\hat{y}(t+1),P\hat{y}(t+2),\ldots,P\hat{y}(t+N)]^T \tag{5.5}$$

$$u=[\Delta u(t),\Delta u(t+1),\ldots,\Delta u(t+N-1)]^T \tag{5.6}$$

$$u^2=[\Delta u^2(t),\Delta u^2(t+1),\ldots,\Delta u^2(t+N-1)]^T \tag{5.7}$$

$$\begin{aligned} f = [&(G_1-g_0)\Delta u(t) + F_1\, y(t), (G_2-g_2-g_0)\Delta u(t+1) \\ &+ F_2\, y(t+1),\ldots, G_N-g_N-\ldots-g_0)\Delta u(t+N-1) \\ &\qquad + F_N\, y(t+N-1)]^T \end{aligned} \tag{5.8}$$

and the matrix G is a lower triangular of dimension N x N.

5.2.1.2 Predictive control law - Quadratic Model

As indicated in chapter 4, in order to compute the vector of controls the following cost function J is minimized

$$J = \sum_{t=N_1}^{N_2} [\hat{y}(t+j)-w(t+j)]^2 + \sum_{j=1}^{NU} \lambda_j[\Delta u(t+j-1)]^2 \tag{5.9}$$

where N_1 and N_2 are minimum and maximum cost horizons respectively, NU is the control horizon, and λ is the control weight. Note that the cost function is identical to that defined by Clarke et al (1987) in the derivation of the GPC algorithm.

In matrix form we write

$$J = E\{(\hat{y} - w)^T(\hat{y}-w) + \lambda u^T u\} \tag{5.10}$$

or

$$J = E(G_1 u + G_2 u^2 + f - w)^T(G_1 u + G_2 u^2 + f - w) + \lambda u^T u \tag{5.11}$$

For consistency with earlier results, let us denote

$$v = G_1 u + G_2 u^2 + f - w \tag{5.12}$$

where **v** is a column vector. Equation (5.11) becomes

$$J = E\{v^T v + \lambda u^T u\} \tag{5.13}$$

Let the $\mathbf{G}_1$ and $\mathbf{G}_2$ matrices be represented by column vectors $\mathbf{g}_0$, $\mathbf{g}_1$, $\mathbf{g}_2$,... , that is:

$$G_1 = [g_0^{(1)}\ g_1^{(1)}\ g_2^{(1)}\ \ldots] \tag{5.14}$$

$$G_2 = [g_0^{(2)}\ g_1^{(2)}\ g_2^{(2)}\ \ldots] \tag{5.15}$$

5.2.1.3 Objective Minimization

Again as shown in chapter 4, for optimal control we minimize J in equation (5.13) with respect to u.

$$\frac{\partial J}{\partial u_i}=v^T\frac{\partial v}{\partial u_i}+\frac{\partial v^T}{\partial u_i}v+2\lambda u^T\frac{\partial u}{\partial u_i} \tag{5.16}$$

or

$$\frac{\partial J}{\partial u_i} = 2v^T\frac{\partial v}{\partial u_i} + 2\lambda u_i \tag{5.17}$$

where v can be written as a sum of outer products

$$\begin{aligned} v &= u_0g_0^{(1)}+u_1g_1^{(1)}+\ldots+u_ig_i^{(1)}+\ldots+u_{N-1}g_{N-1}^{(1)} \\ &+ u_0^2g_0^{(2)}+\ldots+u_i^2g_i^{(2)}+\ldots+u_{N-1}^2g_{N-1}^{(2)} + f+w \end{aligned} \tag{5.18}$$

N-1 is the number of columns of G_1 and G_2 matrices.

We now express the partial derivative of v with respect to u_i as linear combinations of the columns of G_1 and G_2

$$\frac{\partial v}{\partial u_i}=g_i^{(1)}+2u_ig_i^{(2)} \tag{5.19}$$

which is a column vector. Multiplying both sides of equation (5.19) by $2v^T$ we obtain

$$2v^T\frac{\partial v}{\partial u_i}=2v^Tg_i^{(1)}+4u_iv^Tg_i^{(2)} \tag{5.20}$$

which is a scalar.

We can now obtain the partial derivative of J with respect to u as the row vector

$$\frac{\partial J}{\partial u} = \left[\frac{\partial J}{\partial u_0}\ \frac{\partial J}{\partial u_i}\ \cdots\ \frac{\partial J}{\partial u_{N-1}}\right] \tag{5.21}$$

That is

$$\begin{aligned}\frac{\partial J}{\partial u}&=[2v^Tg_0^{(1)}\ 2v^Tg_1^{(1)}\ \dots\ 2v^Tg_{N-1}^{(1)}]\\ &+ [4u_0v^Tg_0^{(2)}\ 4u_1v^Tg_1^{(2)}\ \dots\ 4u_{N-1}v^Tg_{N-1}^{(2)}]\\ &+ [2\lambda u_0\ 2\lambda u_1 \dots 2\lambda u_{N-1}]\end{aligned} \tag{5.22}$$

or

$$\frac{\partial J}{\partial u} = 2v^TG_1+2\lambda u^T+4v^T[u_0g_0^{(2)}\ u_1g_1^{(2)}\ \dots\ u_{N-1}g_{N-1}^{(2)}] \tag{5.23}$$

Finally, we can rewrite the above equation in a more compact form as

$$\frac{\partial J}{\partial u} = 2v^TG_1+4v^TG_2diag(u)+2\lambda u^T \tag{5.24}$$

The gradient in (5.24) can also be used to calculate the Hessian matrix directly and used for function minimization in say, Newton's method.

The Hessian of J with respect to u is written as

$$\frac{\partial^2 J}{\partial u^2}=\begin{bmatrix} \frac{\partial^2 J}{\partial u_0^2} & \frac{\partial^2 J}{\partial u_0 \partial u_1} & \cdots & \frac{\partial^2 J}{\partial u_0 \partial u_{N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial u_{N-1} \partial u_0} & \cdots & \cdots & \frac{\partial^2 J}{\partial u_{N-1}^2} \end{bmatrix} \tag{5.25}$$

After much algebraic manipulation (see appendix), the Hessian can be written as

$$\begin{aligned}\frac{\partial^2 J}{\partial u^2}&=2G_1^TG_1+G_1(G_2diag(u))+2(G_2diag(u))^TG_1\\&+4(G-2diag(u))^T(G_2diag(u))+2\lambda I+4diag(v^TG_2)\end{aligned} \tag{5.26}$$

As is usually the case with receding horizon schemes, the current control law is given as the first element of vector u, obtained formally as a solution to (5.24) by equating to zero or numerically, using (5.24) and (5.26) iteratively.

In the next section we modify the development by considering what may be a more natural way of penalising the effect of the non-linear control on the cost function J.

5.2.[illegible] Penalizing the Process Pseudo Input

In choosing a cost function J of the form given by (5.9) in conjunction with the quadratic model form in (5.1), we could argue that we should penalize the control input in J in a form which more appropriately reflects its structure in the model. This leads to a modified objective.

In order to compute the vector of controls the following cost function J is minimized

$$J = \sum_{j=N_1}^{N_2} [\hat{y}(t+j)-w(t+j)]^2+\sum_{j=1}^{NU} \lambda_j \Delta[B_1(1)u(t+j-1)+B_2(1)u^2(t+j-1)]^2 \qquad (5.27)$$

where

$$B = b_0 + b_1 q^{-1} + \ldots \quad ; \quad B(1)=\sum_{i=1}^{n_b} b_i$$

Notice that the control variable u affects the objective J in the same way as it affects the model. Also averaged steady state weights $B_i(1)$ are used to reflect similar scaling that appears in the model.

In matrix form we write

$$J = E\{(\hat{y}-w)^T(\hat{y}-w) + \lambda[B(1)u + B_2(1)u^2]^T[B(1)u + B_2(1)u^2]\} \tag{5.29}$$

or

$$J = E\{(G_1u+G_2u^2+f-w)^T(G_1u+G_2u^2+f-w) + \lambda[B_1(1)u+B_2(1)u^2]^T[B_1(1)u+B_2(1)u^2]\} \tag{5.30}$$

Let us denote

$$z=B_1(1)u+B_2(1)u^2 \tag{5.31}$$

$$v=G_1u+G_2u^2+f-w \tag{5.32}$$

where v and z are column vectors. Equation (5.30) becomes

$$J = E\{v^Tv+\lambda z^Tz\} \tag{5.33}$$

Let the $\mathbf{G}_1$ and $\mathbf{G}_2$ matrices be represented by column vectors $\mathbf{g}_0$, $\mathbf{g}_1$, $\mathbf{g}_2$,... as before.

5.2.2.1 Objective Minimization

For optimal control we minimize J in equation (5.33) with respect to u. This is again achieved in several steps:

$$\frac{\partial J}{\partial u_i} = \nu^T \frac{\partial \nu}{\partial u_i} + \frac{\partial \nu^T}{\partial u_i}\nu + \lambda\left[z^T \frac{\partial z}{\partial u_i} + \frac{\partial z^T}{\partial u_i} z\right] \tag{5.34}$$

or

$$\frac{\partial J}{\partial u_i} = 2\nu^T \frac{\partial \nu}{\partial u_i} + 2\lambda z^T \frac{\partial z\nu}{\partial u_i} \tag{5.35}$$

where v can be written as

$$\begin{aligned}\nu = u_0 g_0^{(1)} + u_1 g_1^{(1)} + \dots + u_i g_i^{(1)} + \dots + u_{N-1} g_{N-1}^{(1)} + u_0^2 g_0^{(2)} + \\ + u_i^2 g_i^{(2)} + \dots + u_{N-1}^2 g_{N-1}^{(2)} + f + w\end{aligned} \tag{5.36}$$

N-1 is the number of columns of G_1 and G_2 matrices.

Using the same arguments as before we can obtain the final expression for the gradient as

$$\begin{aligned}\frac{\partial J}{\partial \boldsymbol{u}} &= [2\nu^T g_0^{(1)} \;\; 2\nu^T g_1^{(1)} \;\dots\; 2\nu^T g_{N-1}^{(1)}] \\ &+ [4u_0\nu^T g_0^{(2)} \;\; 4u_1\nu^T g_1^{(2)} \;\dots\; 4u_{N-1}\nu^T g_{N-1}^{(2)}] \\ &+ 2\lambda[B_1(1)\boldsymbol{u} + B_2(1)\boldsymbol{u}^2]^T[B_1(1) + B_2(1)\boldsymbol{u}^2]\end{aligned} \tag{5.37}$$

Finally, we can rewrite the above equation in a more compact form as

$$\frac{\partial J}{\partial u} = 2v^{T}G_1 + 4v^{T}G_2 diag(u) + 2\lambda[B_1(1)u + B_2(1)u^2]^T[B_1(1) + B_2(1)u] \tag{5.38}$$

By setting equation (5.38) to zero, a solution for u can be obtained numerically.

Note that similar arguments were used to obtain NLGPC for more complex processes in Katende and Jutan (1996). In that paper a general form of Hammerstein and Volterra process models were considered. They defined the process model as:

$$A(q^{-1})y(t) = q^{-(d+1)}x(t) + \frac{C(q^{-1})}{\Delta}\xi(t) \tag{5.39}$$

where x(t) is a 'pseudo' input and is any function of u(t). They define a cost function as:

$$J = \sum_{i=N_1}^{N_2} [P\hat{y}(t+i) - P(1)w(t+i)]^2 + \lambda\sum_{i=1}^{Nu} (x(t+i-1))^2 \tag{5.40}$$

After some algebraic manipulation they arrive at a solution for the optimal 'pseudo' control x as

$$x = (G^TG + \lambda I)^{-1}G^T(w^* - H\bar{x} - Fc) \tag{5.41}$$

where w* is a modified vector of w ; c is a vector of prediction error through a filter T and G, H and F are polynomials defined by the diophantine equations. They obtain u from x by using Newton's method. It could be noted that this algorithm is a general form of what we have obtained above.

5.3 DISCUSSION

The value of the control weight λ is , in general, difficult to choose, a priori, and is often considered an on-line tuning parameter - although there are some guidelines available in the literature (Lam, 1980), for the standard objective weighting. These guides may be of use with weighting on the 'pseudo' input x, since this converts the non-linear form to an equivalent linear one. Choice of λ, when we weight one of a number of non-linear functions of u, would doubtless be very 'ad hoc' in its approach. Selection of weighting style may not be a question of taste either, since, as some later simulations show quite different control results. In effect, with different components of x dominating in different ranges, we end up with an effective weighting given as function of u.

An additional practical problem is the question of multiple solutions for the optimal control value u_{opt}. In the example we encountered, it was obvious which the correct solution was since the other candidates were rejected on heuristic grounds. If, say, Newton's method was used for the iterations, a good initial guess (the previous control action was used) was often sufficient to converge to the correct option.

Based on the above discussion the following cost function is defined for the NLGPC with penalty on 'pseudo' input

$$J=\sum_{i=N_1}^{N_2}[P\hat{y}(t+i)-P(1)w(t+i)]^2+\lambda\sum_{i=1}^{N_u}(x(t+i-1))^2 \tag{5.42}$$

Note that the 'x' is a non-linear function of the controller output, u. This way the control weighting affects both the linear and non-linear parts of the input signal. In many ways this latter weighting is more effective since the effect on the output y occurs not only through u but also through the various non-linear inputs.

Thus by penalizing x we are actually penalizing the 'net effect' of the input on the process output. The optimal value of x over the prediction horizon is obtained by minimization of J with respect to x. An analytical solution is obtained since x is linear. The optimal u is then obtained numerically from the optimal x.

5.4 SIMULATION STUDIES

In order to evaluate the effect of different control weighting choices on the design of non-linear predictive controllers, simulation studies were carried out for two second order non-linear processes based on a Hammerstein model structure.

Process Models Simulated

Model 1:

$$(1+022q^{-1}-.7991q^{-2})y(t)=q^{-1}(.029+.0269q^{-1})u(t)+q^{-1}(.021+.003q^{-1})u^{2}(t)+\xi(t)$$

This model is open-loop stable with a second order non-linearity in the input .It also has first order B-polynomials. The tuning parameters for both NLGPC were set as follows: N1=2; N2=3; NU=1; P=1; T=1 and λ=0.0001

Model 2:

$$(1+0.022q^{1} - 0.799q^{2})y(t) = q^{-1}(0.029 + .0269q^{-1} + 0.0156q^{-2})u(t)$$
$$+ q^{-1}(0.021+0.003q^{-1} + 0.0023q^{-2})u^{2}(t)+\xi(t)$$

The above model is open-loop stable since all the roots of A-polynomial are inside the unit circle of the q-plane. It has a second order non-linearity in the input signal with second order B-polynomials. The tuning parameters for both NLGPC were set as follow: N1=2; N2= 3; NU=1; P=1; T=1; and λ=0.0001.

The process parameters were estimated using a standard UDU version of RLS (Bierman, 1977). Persistent excitation was guaranteed by turning off the adaptation whenever the increment in the input signal became minimal. An exponential forgetting factor was adopted with values ranging from 0.98 to unity (no forgetting). The parameter estimates were initialized in simulation with b_{01} and b_{02} equal to unity and the rest equal to very small numbers. The diagonal of the initial covariance matrix was set to 10^4. For all simulation runs noise variance was set to 0.001.

The figures shown below indicate the behaviour of the closed-loop system with 200 samples. The set-point was switched between two level (one and zero) for all simulation runs. The input signal, u, was limited between 100 and -100.

5.4.1 Results

Figure 5.1 and 5.2 show a large difference in performance between two NLGPC algorithms, one with a penalty on the input and the other with a penalty on the pseudo input, when applied to a non-linear process (model 1). The NLGPC algorithm with penalty on the input controls the non-linear process model 1 with a slower response and the input seem to be less noisy as shown in figure 5.1. On the other hand, the NLGPC with penalty on the pseudo input controls the non-linear process model well enough with faster response as shown in figure 5.2. Figure 5.3 demonstrates the performance of NLGPC with penalty on input on the non-linear process model 2 and figure 5.4 shows the corresponding performance of the NLGPC with penalty on pseudo input on the same model. Here control action is much smoother and much of the noise response is eliminated.

The different control behaviour cannot be attributed solely to different weighting in the objective function. Here, since the control objectives, and hence the optimal solutions are different, the control behaviour is expected to be different as well. Here the two optimal solutions are quite different as well. Here, the two optimal solutions are quite

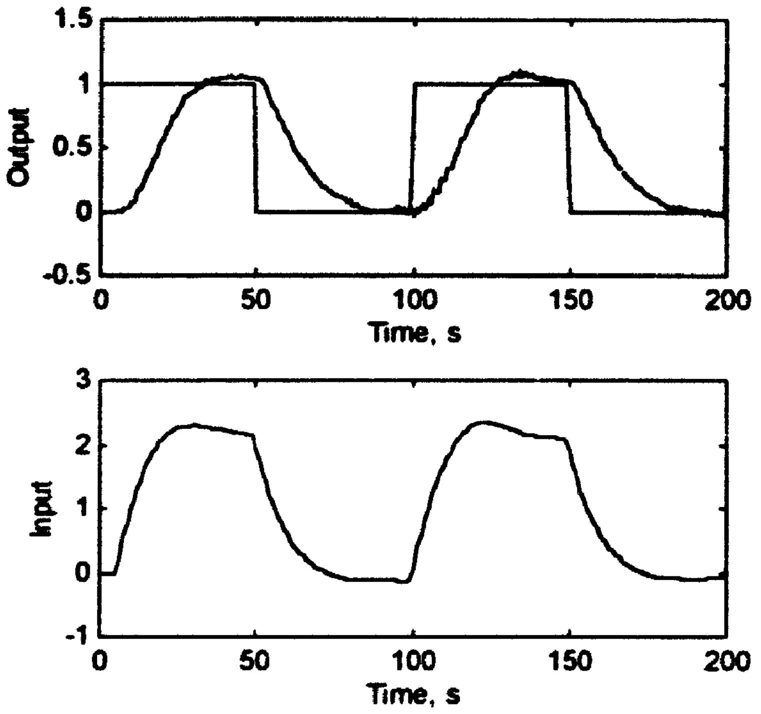

Figure 5.1: Adaptive NLGPC with penalty on the input (controlling process model 1)

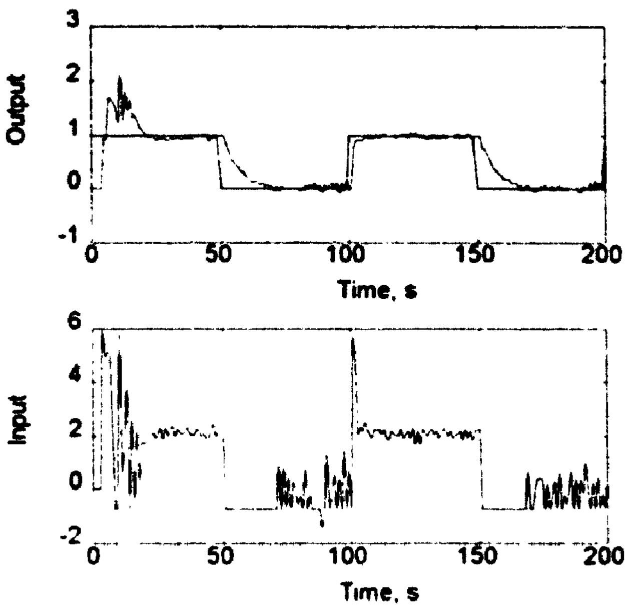

Figure 5.2: Adaptive NLGPC with penalty on the 'pseudo' input (controlling process model 1)

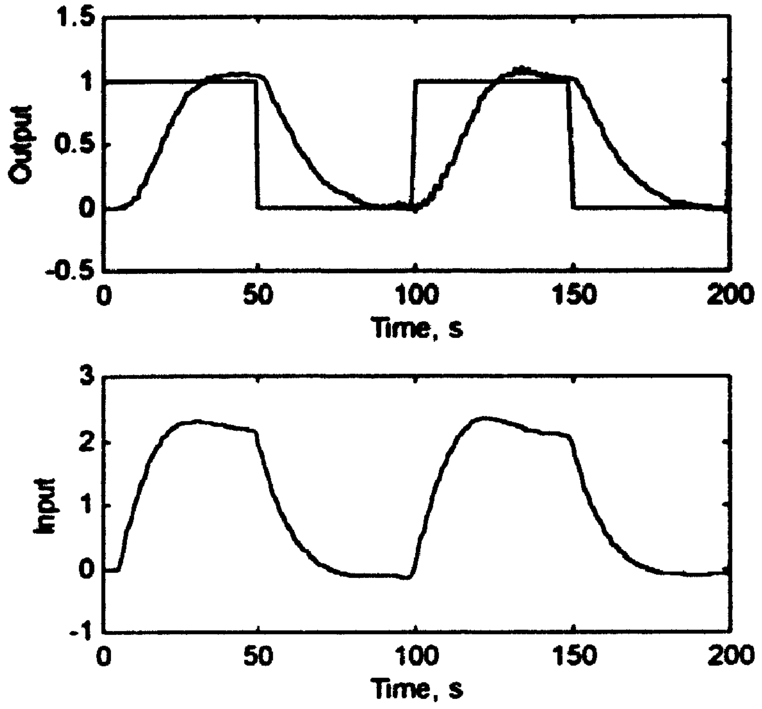

Figure 5.3: Adaptive NLGPC with penalty on the input (controlling process model 2)

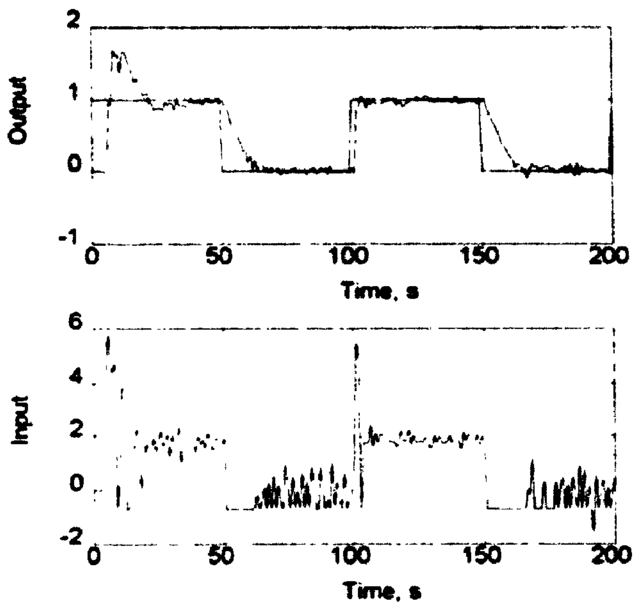

Figure 5.4: Adaptive NLGPC with penalty on the 'pseudo' input (controlling process model 2)

different in their sensitivities to the same noise level. The 'quality' of the final control is a function of trial and error tuning and we were not able to match this as well as we wanted to. Nevertheless, the question of what is the more appropriate objective function remains open and these results show that both control quality and sensitivity to noise will be affected. No doubt stability is also an issue but this was not treated in this initial study.

5.5 CONCLUSIONS

Analytical expressions for the gradient matrix and the Hessian matrix have been obtained for the optimal control solution using a second order NLGPC algorithm. The results allow for reduction in on-line calculational load. Typical computation time is reduced by 50% compared to the case where a purely numerical approach is taken. An additional issue of the choice of penalisation in the objective function is considered. The performance of two NLGPC algorithms based on different criterion functions has been evaluated using a number of simulation runs. The first algorithm was based on a quadratic cost function with penalty on the input signal u. For the second algorithm the 'pseudo' input x was penalised which in turn contains the manipulated variable. It has been shown through simulation that selection of control weighting style is very important but the best choice remains an open question.

CHAPTER 6

SIMULATION STUDIES TO EVALUATE NLGPC PERFORMANCE

6.1 INTRODUCTION

In practice, before a controller is implemented on a real time process, it is important to analyze the dynamics of the closed loop system through simulation in order to study the performance of the controller designed. This will not only allow the controller designer to anticipate the behaviour of the closed loop system but also give him an idea of any difficulties that might arise during the actual implementation of the controller. Certain conclusions drawn from the simulation results may deviate from what was expected, if that happens some alterations may be required for the controller design before implementation.

In this chapter, simulation studies carried out on two different types of control algorithms, namely, the generalized predictive control (GPC) and the non-linear generalized predictive control algorithm are presented. A number of different process models of different complexity are simulated. The performance of the NLGPC is then compared to that of the GPC by using simulation results of both cases.

6.2 PROCESS MODELS (HAMMERSTEIN STRUCTURE)

In order to evaluate the usefulness of the new non-linear predictive control algorithm simulation studies were carried out for a linear process (model 1), two non-linear processes (models 2 and 3) based on a Hammerstein model structure and one non-linear process with a bilinear term (model 4). These processes were also modelled as approximate linear processes in order to develop control algorithms for the standard GPC for comparison purposes. Starting with first order polynomials, $A(q^{-1})$ and $B(q^{-1})$ polynomial orders were gradually increased until the best approximate CARMA model was selected which suitably matched the dynamics of the Hammerstein model, in the given range. These best fitting CARMA models were used in the GPC design routine.

Process Models Simulated

Model 1:

$$(1+0.022q^{-1}-0.7991q^{-2})y(t)=q^{-1}(0.0290+0.0269q^{-1})u(t)+\xi(t)$$

Model 2:

$$(1+.022q^{-1}-.7991q^{-2})y(t)=q^{-1}(.029+.0269q^{-1})u(t)+q^{-1}(.021+.003q^{-1})u^{2}(t)+\xi(t)$$

Model 3:

$$\begin{aligned}(1+0.022q^{-1}-.7991q^{-2})y(t)&=q^{-1}(0.029+.0269q^{-1}+.0156q^{-2})u(t)\\&+q^{-1}(0.021+0.003q^{-1}+0.0023q^{-2})u^{2}(t)+\xi(t)\end{aligned}$$

Model 4:

$$\begin{aligned}(1+.022q^{-1}-.7991q^{-2})y(t)&=q^{-1}(.029+.0269q^{-1})u(t)+q^{-1}(.023+.0012q^{-1})u^{2}(t)\\&q^{-1}(.021+.003q^{-1})u(t)y(t-1)+\xi(t)\end{aligned}$$

The process model parameters are summarized below in Table 6.1. For all simulation runs the noise variance was set to .001.

Table 6.1: Summary of model parameters.

Model 1 (Linear)

$A=1+0.0220q^{-1}-0.7991q^{-1}$

$B1=0.0290+0.0269q^{-1}$

$B2=0$

$C=1$

Parameter Settings: $N1=2$ $N2=3$ $NU=1$ $P=1$ $T=1$ $\lambda=0.0001$

Model 2 (2nd order non-linearity in input)

$A=1+0.0220q^{-1}-0.7991q^{-2}$

$B1=0.0290+0.0269q^{-1}$

$B2=0.0210+0.0030q^{-1}$

$C=1$

Parameter Settings: $N1=2$ $N2=3$ $NU=1$ $P=1$ $T=1$ $\lambda=0.0001$

Model 3 (2nd order non-linearity)

$A=1+0.0220q^{-1}-0.7991q^{-2}$

$B1=0.0290+0.0269q^{-1}+0.0156q^{-2}$

$B2=0.0210+0.0030q^{-1}+0.0023q^{-2}$

$C=1$

Parameter Settings: $N1=2$ $N2=3$ $NU=1$ $P=1$ $T=1$ $\lambda=0.0001$

<u>**Model 4 (Bilinear)**</u>

$A = 1 + 0.0220q^{-1} - 0.799q^{-2}$

$B1 = 0.0290 + 0.0269q^{-1}$

$B2 = 0.230 + 0.0012q^{-1}$

$B3 = 0.021 + 0.003q^{-1}$

The parameters of the A, B1 and B2 polynomial were estimated using a standard UDU version of RLS (Bierman, 1977). Persistent excitation was guaranteed by turning off the adaptation whenever the increment in the input signal became minimal. An exponential forgetting factor was adopted with values ranging from 0.98 to unity (no forgetting). The parameter estimates were initialized in simulation with b_{01} and b_{02} equal to unity and the rest equal to very small numbers. The diagonal of the initial covariance matrix was set to 10^{3}.

The figures shown below indicate the behaviour of the closed-loop system with 200 samples. The set-point was switched between two levels (one and zero) for all simulation runs. The noise variance was set at 10^{-4}. The input signal was limited between 100 and -100.

6.2.1 Results

Figure 1 shows the performance of the NLGPC on process model 1. Model 1 has no non-linear term as B2=0. As expected the performance of the standard GPC on process model 1 shown in figure 6.2 is similar to that of NLGPC shown in figure 6.1. One can easily observe that for a linear process model, the general form of NLGPC produces a similar response to that of the GPC. However, the NLGPC seems to be more active at the instant of set-point change. This may be due to the fact that the control weighting on x includes the effect of the b-parameters whereas if we weight u directly its 'effective parameter' is unity.

By contrast, figure 6.3 and 6.4 show a large difference in performance between NLGPC and GPC when applied to the non-linear process. The NLGPC algorithm controls the non-linear process model 2 very well as shown in figure 6.3. On the other hand, the GPC could not control the non-linear process model well enough as shown in figure 6.4. Unacceptably large process output occurs or sometimes the closed loop system went unstable when the GPC was loosely tuned. Better, though still unsatisfactory, results were obtained when the GPC was heavily penalized. Even after much tuning of the GPC a persistent offset (due to the second order input non-linearity) was experienced which could not be properly eliminated by the integral action. Figure 6.5 demonstrates the performance of NLGPC on the non-linear process model 3 and figure 6.6 shows the corresponding performance of the GPC on the same model. It is clear that GPC was unable to control the non-linear processes well (model 2 and 3) whereas the NLGPC showed good performance with both linear and non-linear models.

The bilinear model (model 4) was simulated under the NLGPC control (figure 6.7). It can be observed that good control was obtained thus showing the extended ability of the NLGPC algorithm to control bilinear systems as well.

In these examples the NLGPC outperformed the GPC for the non-linear processes. Note that, for a linear CARMA model the NLGPC defaults to the structure of the GPC, but the control weighting in the objective function is different. An analysis of the performance of the NLGPC is provided below.

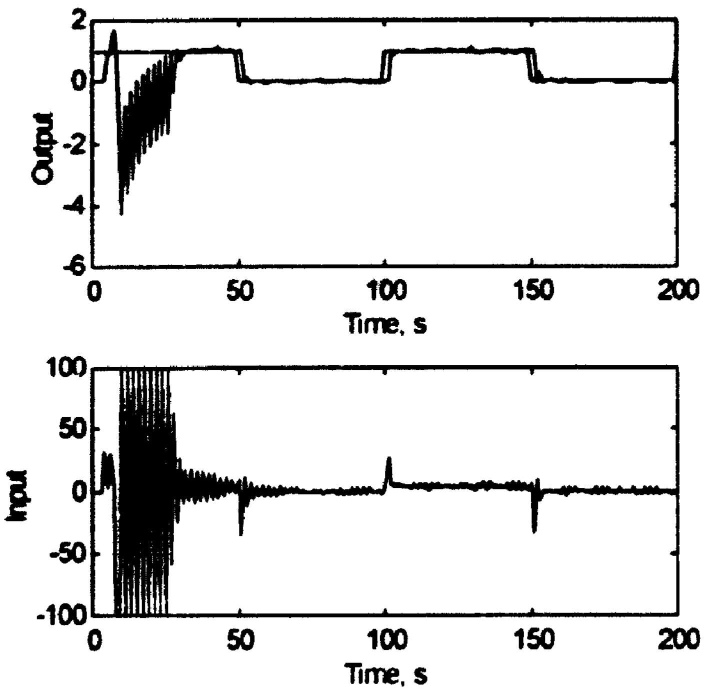

Figure 6.1: Adaptive NLGPC on Linear Process model 1

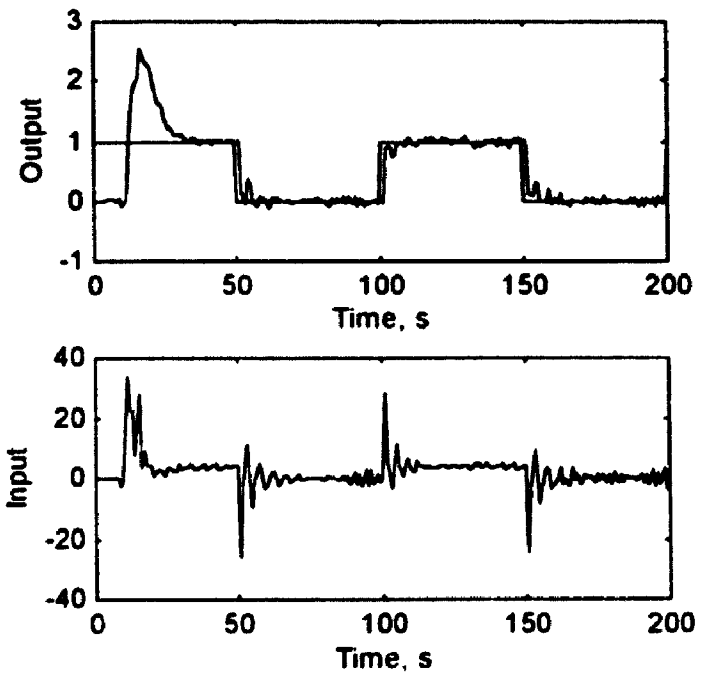

Figure 6.2: Adaptive GPC on a Linear Process model 1

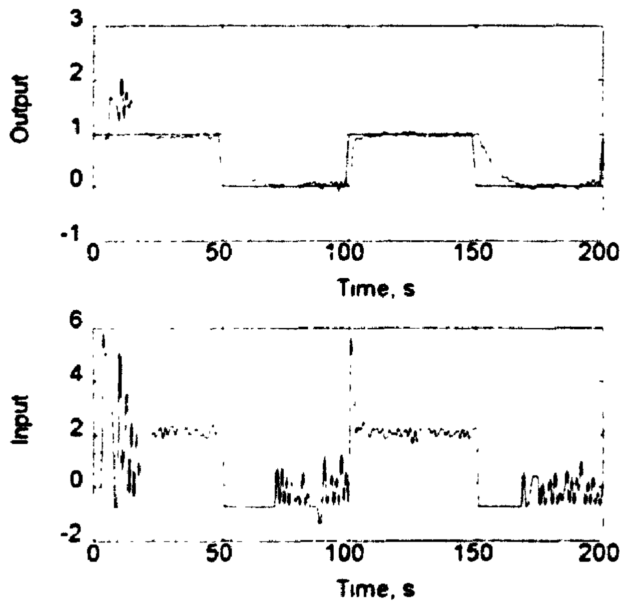

Figure 6.3: Adaptive NLGPC on Non-linear Process model 2

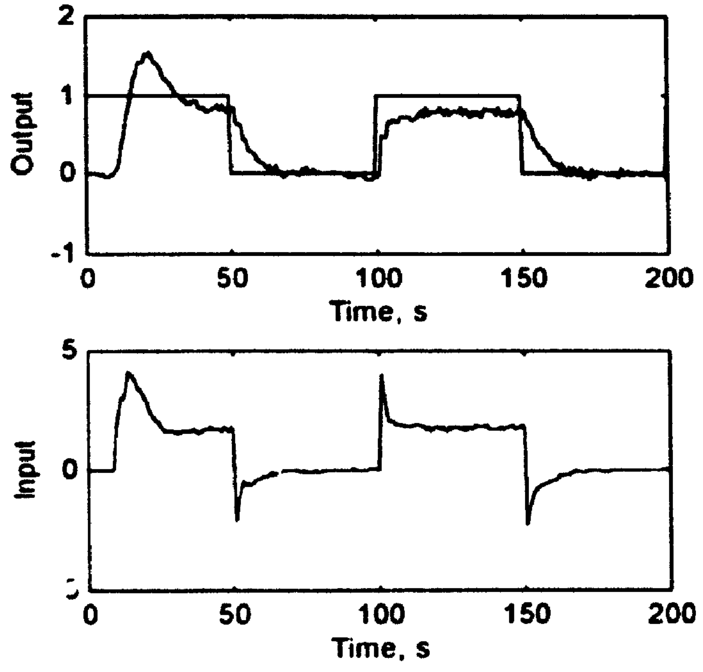

Figure 6.4: Adaptive GPC on Non-linear Process model 2

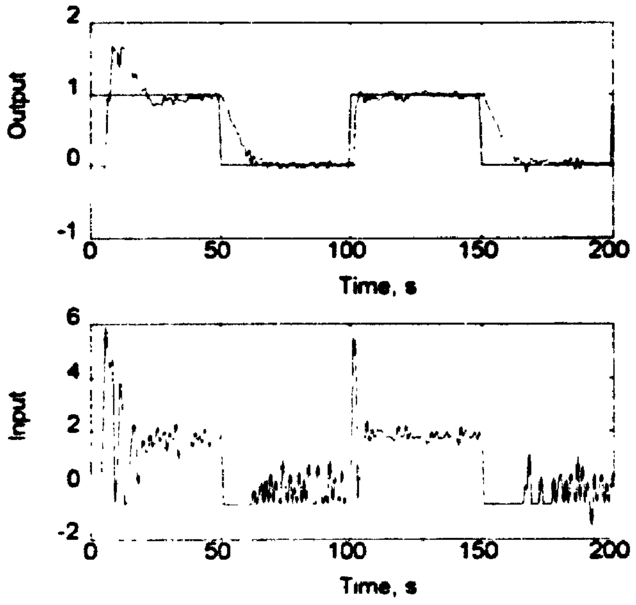

Figure 6.5: Adaptive NLGPC on Non-linear Process model 3

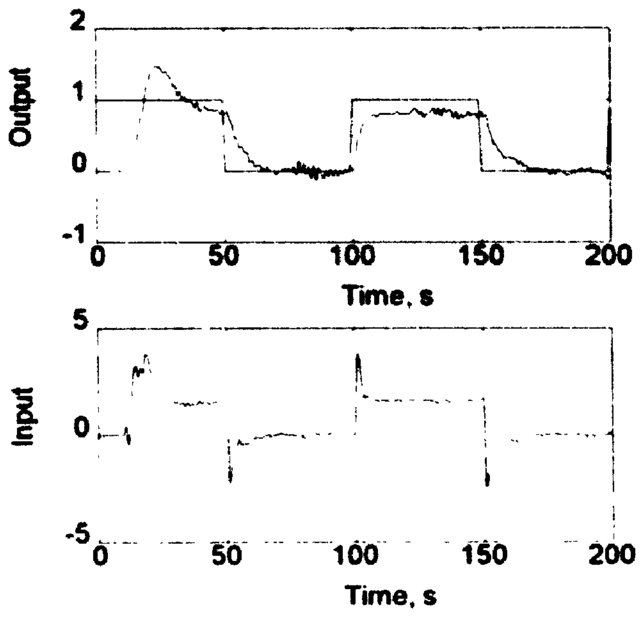

Figure 6.6: Adaptive GPC on a Non-linear Process model 3

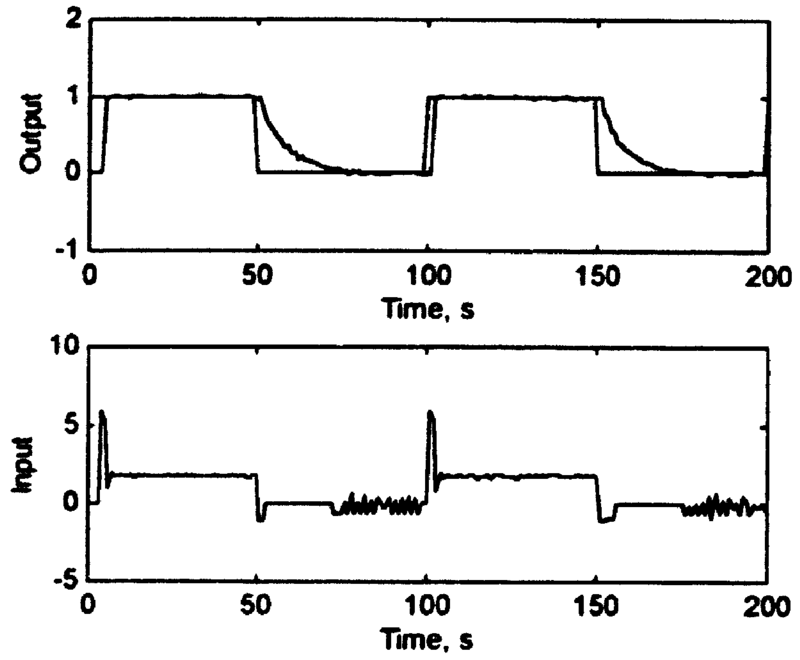

Figure 6.7: Adaptive NLGPC on Bilinear Process model 4

6.3 CSTR TEMPERATURE CONTROL

An adiabatic continuous stirred-tank reactor (CSTR) with a zero order, exothermic reaction and no external cooling (Agarwal and Seborg (1987), Steadman (1978)) was simulated in order to evaluate the performance of the NLGPC algorithm on a set of dynamics based on a real process. Agarwal and Seborg (1987) write the unsteady-state energy balance for the reactor as

$$MC_p\frac{dT_0}{dt'} = F_iC_p(T_i-T_0)+k_0V(-\Delta H)\exp(\frac{-\Delta E}{RT_0})$$

where t' is continuous time in hours; R is the universal gas constant; V is the holdup volume; M is the mass holdup of liquid; F_i is the feed flow rate; C_p is the specific heat of liquid; $-\Delta H$ is tne heat of reaction; k_0 is the reaction rate constant; ΔE is the activation energy; T_i is the feed temperature; and T_0 is the outlet temperature.

As in Agarwal and Seborg(1987), the outlet temperature T_0, and the flowrate, F_i were considered to be the output and manipulated variables respectively. The feed temperature, T_i, is the unmeasured load variable. In their paper Agarwal and Seborg illustrate the steady state process conditions and gains when the load is set at , T_i=456K. Here F_i passes through a minimum at T_0=478.2K. At this value of T_0, the process gain is discontinuous. Reducing F_i to below the minimum of F_i leads to a runaway condition.

We simulated the simple Hammerstein model

$$y(t)+a_1y(t-1)=b_1u(t-1)+b_2u^2(t-1)$$

to represent the reactor dynamics and based the corresponding NLGPC algorithm on this Hammerstein model, that is, $A = 1 + a_1 q^{-1}$; $B_1 = b_1$; $B_2 = b_2$. The tuning parameters were set as $N_1 = 2$; $N_2 = 3$; $P = 1$; $T = 1$ $N_u = 1$ and $\lambda = 0.0001$. The parameters a_1, b_1 and b_2 were initialized to small numbers and estimated using a recursive least squares method as described earlier. Persistent excitation was guaranteed by shutting off the estimation whenever the increment in the input signal became minimal. F_i was constrained between 10000 and 50000 Kg/h and the setpoint was changed between 473 and 463K every 50 sampling instants. Process variables were set according to table 6.2.

Table 6.2: CSTR Data

Variable	Value
V	231.5 l
M	166.7 kg
F_i	Flow (Manipulated Variable)
Cp	0.85 kcal kg^{-1} K^{-1}
-H	785 kcal kg^{-1}
k_0	2.265 x 10^9 kg l^{-1} h^{-1}
E	20456 kcal kg^{-1} $mole^{-1}$
T_i	Inlet Temperature (unmeasured load)
T_o	Outlet Temperature (controlled variable)
R	1.9872 cal g-mol^{-1} K^{-1}

The process was simulated using a fourth order Runge-Kutta numerical integration method. Figure 6.8 shows the results of a closed loop run under the NLGPC when the load, T_i is set to 456K. At each optimal control calculation two values of u are obtained. A selection of the correct one to implement was based on a simple logic check. A logic 'check routine' was included in the simulation program to reject any values which do not give outlet temperature values close enough to the setpoint as well as negative or imaginary values. From figure 6.8 it can be seen that the NLGPC controls the CSTR temperature very well while using a simple Hammerstein model for parameter estimation and the setpoint practically coincident with the output temperature.

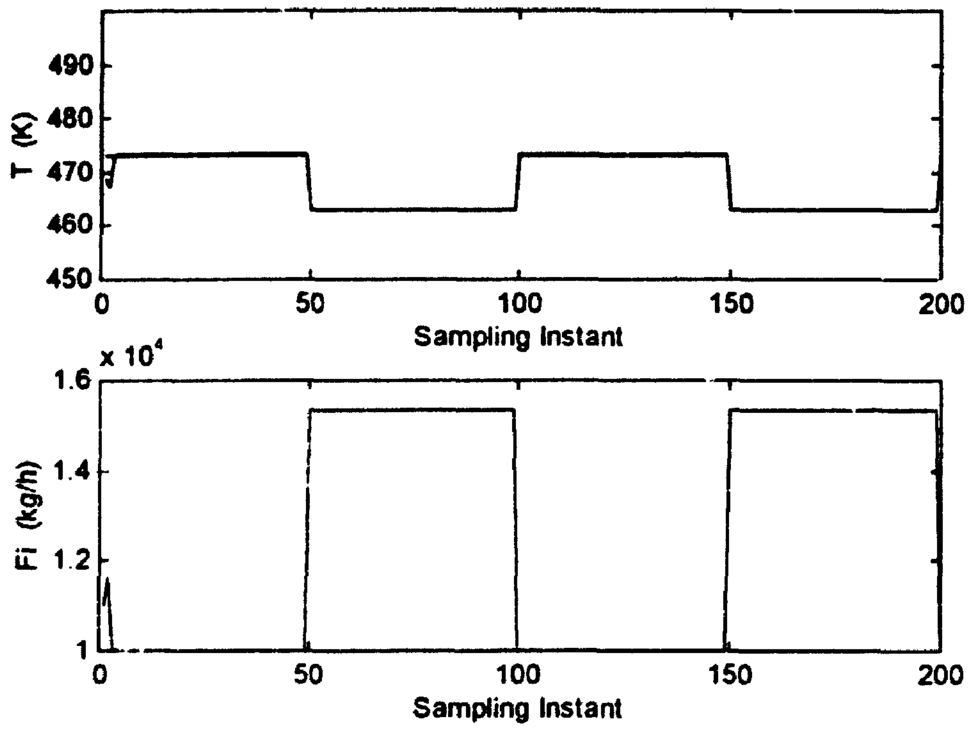

Figure 6.8: Temperature Control of CSTR using NLGPC

6.4 CONCLUSIONS

A non-linear generalized predictive control algorithm, based on a class of non-linear dynamics that can be suitably modelled using Volterra and Hammerstein structures has been simulated to control a number of processes of different complexity. Non-linear Generalized Predictive Control (NLGPC) showed an advantage over generalized predictive control (GPC) in that NLGPC controlled processes with high non-linearity in input signal very well. A lot of simulation runs were carried out but only interesting cases were used to demonstrate the performance of the NLGPC and compare it with the GPC. The NLGPC performed very well when used to control an adiabatic continuous stirred-tank reactor (CSTR).

A proof describing the existence of an offset when a non-linear Hammerstein process model is simulated and controlled by a linear model based GPC is given in section 4.2.4 and 4.2.5. Furthermore, the offset existence in case of GPC controlling a non-linear process based on a Hammerstein model can qualitatively be attributed to the change in sign of the input deviation for the even terms in the simulated non-linear process model which is not catered for in the controller design.

CHAPTER 7

Experimental evaluation of Predictive Temperature Control for a Batch Reactor System

7.1 INTRODUCTION

It is a well known fact that in practice processes are non-linear in nature. In fact the major sources of non-linearities in practical applications can be known. For example, combined mass and energy balances can result in bilinear or non-linear equations for the process control problems. Process thermodynamics, kinetics and characteristics of control valve can all result in non-linear equations.

Most of the applications in industry are based on controllers designed around linear models. Unfortunately, in some cases these controllers fail to achieve the required goals. Many attempts have been made to modify the PID controller, the most popular controller in industry, into a self-tuner to cater for possible process non-linearities. For example, Vega and co-workers (1991) introduced an explicit self-tuning PID controller with two nested loops. One loop is the optimization loop and the other is the parameter estimation loop. Such an algorithm is difficult to implement and can lead to a complex closed loop system. Other self-tuning PID controllers have a narrow application range for example the self-tuning PID controller by Cameron and Seborg (1983) (Vega *et al.*, 1991). In order to introduce the integral action in their self-tuning PID algorithm, Cameron and Seborg introduce a restrictive relationship between the constraint polynomial in their objective function and a polynomial that occurs in the controller algorithm. They also need to introduce an additional design parameter for tuning purposes. They use a first order filter polynomial which, together with the other restrictions, may limit the maximum order of the process model upon which the controller is based.

- Katende and Jutan (1993) also introduced an algorithm for an implicit self-tuning controller which has a PID structure. Some of their ideas were adopted from the work of Åström and Wittenmark (1973), Clarke and Gawthrop (1975). Their method is based on the minimisation of a general quadratic cost function of prediction errors and control efforts over a single step. They used a general Box-Jenkins model (Box and Jenkins, 1990) to describe the process, so that a stochastic framework is taken into account and models of variable complexity can be considered. They also used a recursive least squares identification scheme to update the parameter estimates. The solution to the control design problem is given as a recursive on-line algorithm which allows the PID parameters to be updated at each sampling interval or leads to a converged parameter set for a fixed process model.

However, all the above algorithms, are based on linear process models which may create problems in handling very non-linear processes over a wide range of operating conditions. Katende and Jutan (1996) developed an adaptive non-linear predictive control strategy which has the attractive features of a generalized predictive controller, but without the restriction to linear dynamics. This is an advantage over the current discrete non-linear algorithms which usually restrict the process dynamics to second order. Performance is also improved by using a non-standard cost-function. The Non-linear Generalized Predictive Control (NLGPC) algorithm can deal with both linear and non-linear processes and is developed in the framework of the GPC algorithm.

The aim of the experimental studies is to illustrate the performance of the NLGPC on a batch reactor system. The NLGPC is used to control the temperature of the reactor contents and the results are compared to those obtained using a GPC, GMV and STPID algorithms on the same reactor system. In order to achieve this, we set up a number of experimental runs to test the NLGPC strategy. This was done by interfacing a personal computer with a batch reactor system. A closed loop system was set up and data was collected. In order to compare the NLGPC with other existing strategies, a number of experimental runs were carried out on a batch reactor system under the self-tuning PID control (Katende and Jutan, 1993),GMV control (Clarke and Gawthrope, 1979), GPC control (Clarke *et al.*, 1987).

In this chapter, the system used to run the experiments is described in section 7.2.

The performance of a new self-tuning PID controller on a batch reactor system is illustrated in section 7.3, and the results for GMV control are discussed in section 7.4. Lastly, the performance of the GPC when connected to the batch reactor is discussed in section 7.5 and that of the NLGPC is detailed in section 7.6.

7.2 PROCESS DESCRIPTION

The reactor system is a 2 litre batch reactor filled with water and then sealed. The reactor contents were heated or cooled by the fluid in the jacket. The process input (or the controller output) signal was split into two signals which were directed to separate steam and water valves. At full heating the steam valve was 100% open (allowing maximum flow rate of 14.46lb/h) and the water valve fully closed. Conversely, at full cooling the water valve was 100% open (allowing a maximum flow rate of 42.42lb/h) and the steam valve fully closed. This strategy is known as split range or parametric control (Jutan and Uppal, 1984).

A jet pump or mixer which instantly mixes steam and water and feeds it to the reactor jacket was employed to implement the parametric control. The water pressure was set at 50psi and steam pressure at 35psi to ensure water flow at the mixer even at full heating. The temperature of the reactor contents was measured as the process output. A schematic diagram of the process is shown in figure 7.1.

The fluid enters the reactor jacket from the bottom, and exits through a float type steam trap at the top of the jacket. The steam trap acts as a back pressure device, which allows the water to develop superheated temperatures in the jacket. It also has its normal steam trapping function at higher operating temperatures. Steam is coupled with cooling water to provide sufficient energy removal. It is more energy efficient to have a variable cooling level. In the heat-up cycle, we would not wish to waste energy on a predetermined and fixed minimum cooling level. This minimum cooling level is usually determined by the cool-down profile. In fact, many batch reactors are run with a fixed level of cooling, and temperature change is achieved by varying steam flow. A better approach is to use split range parametric control used here.

The reactor has, typically, three cooling/heating sections which are: the full cooling section, the cooking section and the full heating section. During the full cooling stage there is more cold water than steam injected into the reactor jacket, and at times it is simply cold water which is injected into the system. This would mean a rapid drop in temperature of the reactor content (for our study the reactor content was water). The rate at which the temperature of reactor content drops will depend on the specific heat capacity of water in the cooling jacket, the specific heat capacity of the reactor content, the heat transfer coefficient between the reactor content and the cooling water and the fouling factors. During the cooking stage large changes in temperature of reactor content are minimized. During the full heating stage more steam than cold water is injected into the reactor jacket, and at times 100% steam is injected into the system. This would mean rapid rise in temperature of the reactor content. The rate at which the temperature of reactor content rises will depend on the latent heat of steam (in addition to the factors mentioned above). Because the latent heat of steam is much higher than the specific heat of water, there is a big difference in the rate of heating and cooling. The gain of such a heating/cooling system is often non-linear, since it is typically much easier to add energy to the reactor, than to remove it. The temperature controller would often need to have several sets of controller tuning constants to cope with this induced non-linearity. By altering the curves/lines describing the movement of steam and cold water valves (parametric curves) one can reduce or increase the non-linearity of the system but never reach complete elimination of the non-linear dynamics.

Jutan and Rodriguez (1987) describe a design to determine a set of parametric curves $u_1(z)$ and $u_2(z)$, using trial and error as well as engineering judgement, that would tend to decrease or increase the non-linearity between the parametric variable (control signal), z, and the reactor steady state temperature, T. u_1 is the manipulated steam valve opening and u_2 the cooling water valve opening. They give two equations to describe the parametric relationship

$$u_1 = f_1(z)+c_1$$
$$u_2 = f_2(z)+c_2$$

where f_1 describes a mapping function between u_1 and z; f_2 describes a mapping function between u_2 and z; c_1 and c_2 are constants. Figure 7.2 is extracted from Jutan and

Rodriguez's (1987) paper illustrating typical parametric curves for steam and water.

Although no exothermic reaction is carried out here, internal heat generation can be simulated by injecting steam into the reactor directly. We also concentrated our studies on setpoint disturbance since this strategy would take us into different operating ranges and bring out the nonlinearities in the process.

Digital data collection was achieved using a 486DX personal computer interfaced with a Data Translation Board, DT2801 series. Programming was done in real-time, utilizing QuickBASIC, C and C++ programming language and software packages like ©Matlab and ©Simulink as well as Simulink Predictive Adaptive Control Environment (SPACE) by Scokaert *et al.* (1995).

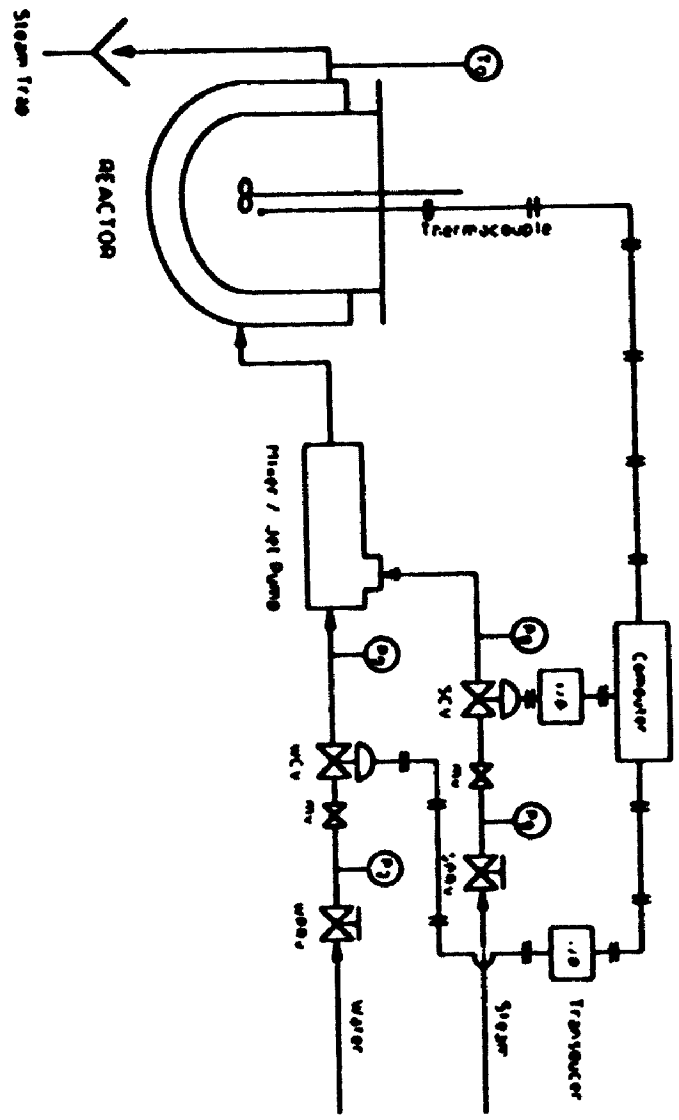

Figure 7.1: Schematic Diagram of a Batch Reactor System

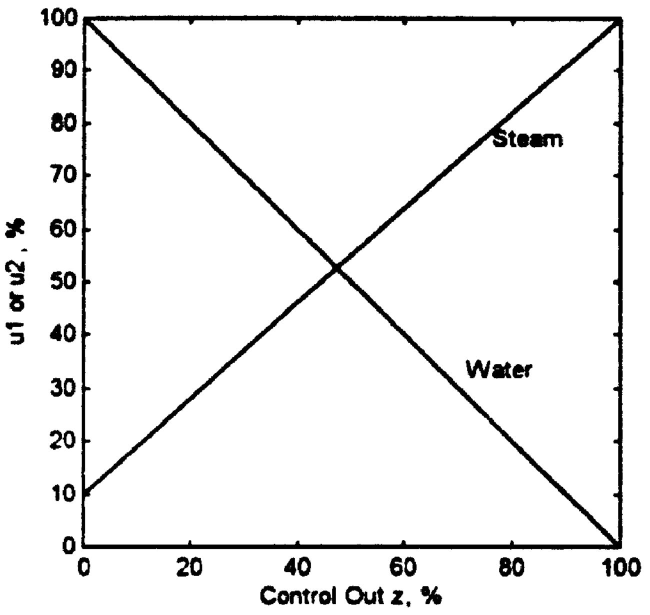

Figure 7.2: Split-range Parametric Control

7.3 REACTOR TEMPERATURE CONTROL UNDER STPID

A constrained self-tuning PID (STPID) controller (Katende and Jutan, 1993) does not require one to have the *priori* knowledge of the process model. The controller output is

$$\nabla^d u_t = -g(z^{-1}) y_t \tag{7.1}$$

where

$$g(z^{-1}) = g_0 + g_1 z^{-1} + g_2 z^{-2} \tag{7.2}$$

The three parameters of the polynomial $g(z^{-1})$ are estimated on-line, via a recursive parameter estimator. For this experiment, the recursive parameter estimator used was the standard recursive least squares method with modifications suggested by Bierman (1977) to ensure stability and convergence in discrete form.

Note that this kind of self-tuning PID controller is in an implicit form, that is, the process model parameters have been re-parameterized to give the controller parameters. Therefore, we directly estimate the controller parameters. This is in contrast with the explicit self-tuning controllers whereby the process model parameters are estimated and then the diophantine equation is numerically computed to give the controller parameters.

7.3.1 Discussion of Results

Table 7.1 gives the summary of the initial conditions for the experimental run under the constrained self-tuning PID controller. Note that the forgetting factor λ is initially set at 0.9 and then increased exponentially until it approaches the value of 1. Introduction of the exponential forgetting factor implies the minimization of a time-weighted least squares. In this manner, observations made in the distant past are given very low weights and will have little influence on the current parameter estimates. Conversely, the most recent observations are weighted most heavily and therefore contribute most to the current estimates. This method therefore allows the scheme to track slowly changing parameter values.

Table 7.1: Initial conditions for the experimental run with STPID

Run Type	STPID control algorithm parameters
Temperature setpoint change: From 30°C to 60°C From 60°C to 40°C	$b=1$ time delay $d=1$ order of ∇ -for non stationary disturbance. $\xi=1$ constraint $\lambda=0.9-1$ exponentially increasing forgetting factor $\hat{G}^T=[1\ .1\ .1]$

A value of the constraint ξ which was chosen by trial and error was set at a value of unity. The setpoint was then changed from 30°C to 60°C and then finally to 40°C. This was intended to test the adaptability of the controller at different operating ranges. The non-linearity of this batch reactor system was discussed in Katende (1992). Here a standard PID controller was also tried but had poor performance.

The convergence of the g parameters is shown figure 7.3. It can be seen that converges of the g parameters was reached at each setpoint. From the optimal g parameter estimates we obtain the conventional PID controller parameters using the following equations

$$g_0 = Kc\left(1 + \frac{Td}{T} + \frac{T}{Ti}\right) \tag{7.3}$$

$$g_1 = -Kc\left(1 + 2\frac{Td}{T}\right) \tag{7.4}$$

$$g_2 = Kc\frac{Td}{T} \tag{7.5}$$

where T is the sampling interval. Figure 7.4 shows the variation of these parameters with time.

Figure 7.5 shows the variation of reactor temperature (process output) with time and that of steam flow and water flow (process inputs) with time. It can clearly be seen that the process output was kept at its setpoint with the self-tuning PID control despite the wide changes in the operating range. At the beginning of the experiment the setpoint was set to 30°C. From figure 5 it can be seen that the controller responds in a very smooth fashion which results in a smooth process response with an overshoot of approximately 5°C. The only problem is that the rise time and response or settling time are quite high (approximately 2250 seconds and 9000 seconds respectively). Once steady state is reached the controller maintains the reactor temperature at the desired value. When the setpoint was stepped-up to 60°C from 30 C, a number of problems were encountered. The controller responds slowly but steadily to increase the flow of steam and decrease the flow of cold water This smooth transition resulted in a high overshoot (approximately 26°C) and high rise time (approximately 900 seconds). This operating range is considered to be the most non-linear and poor performance of a self-tuning PID, which is based on a linear model, is not surprising. When the setpoint was stepped-down to value of 40°C, the controller performance improved considerably. The undershoot was quite small, although the controller response was still not satisfactory.

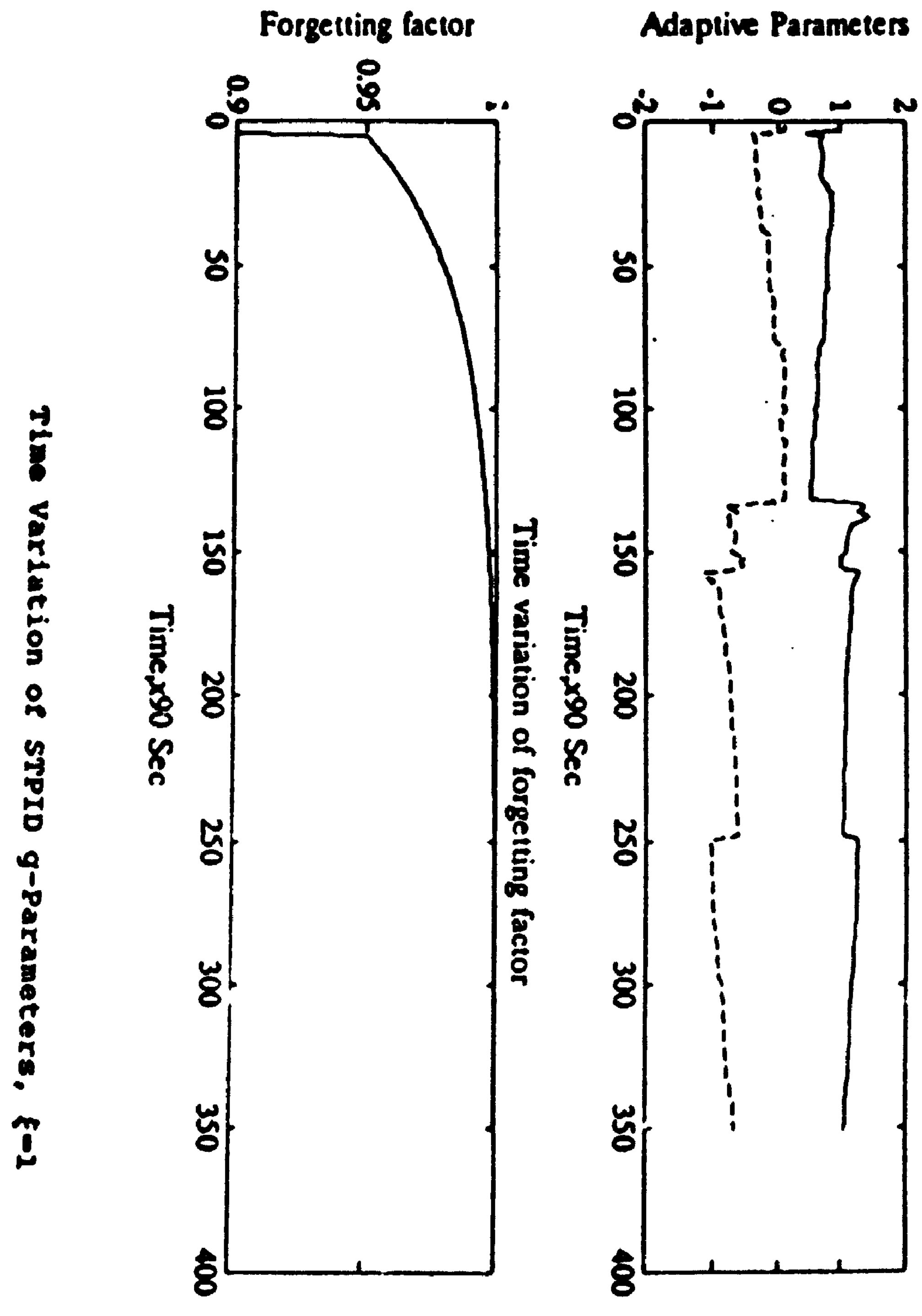

Figure 7.3: Time Variation of STPID Parameters

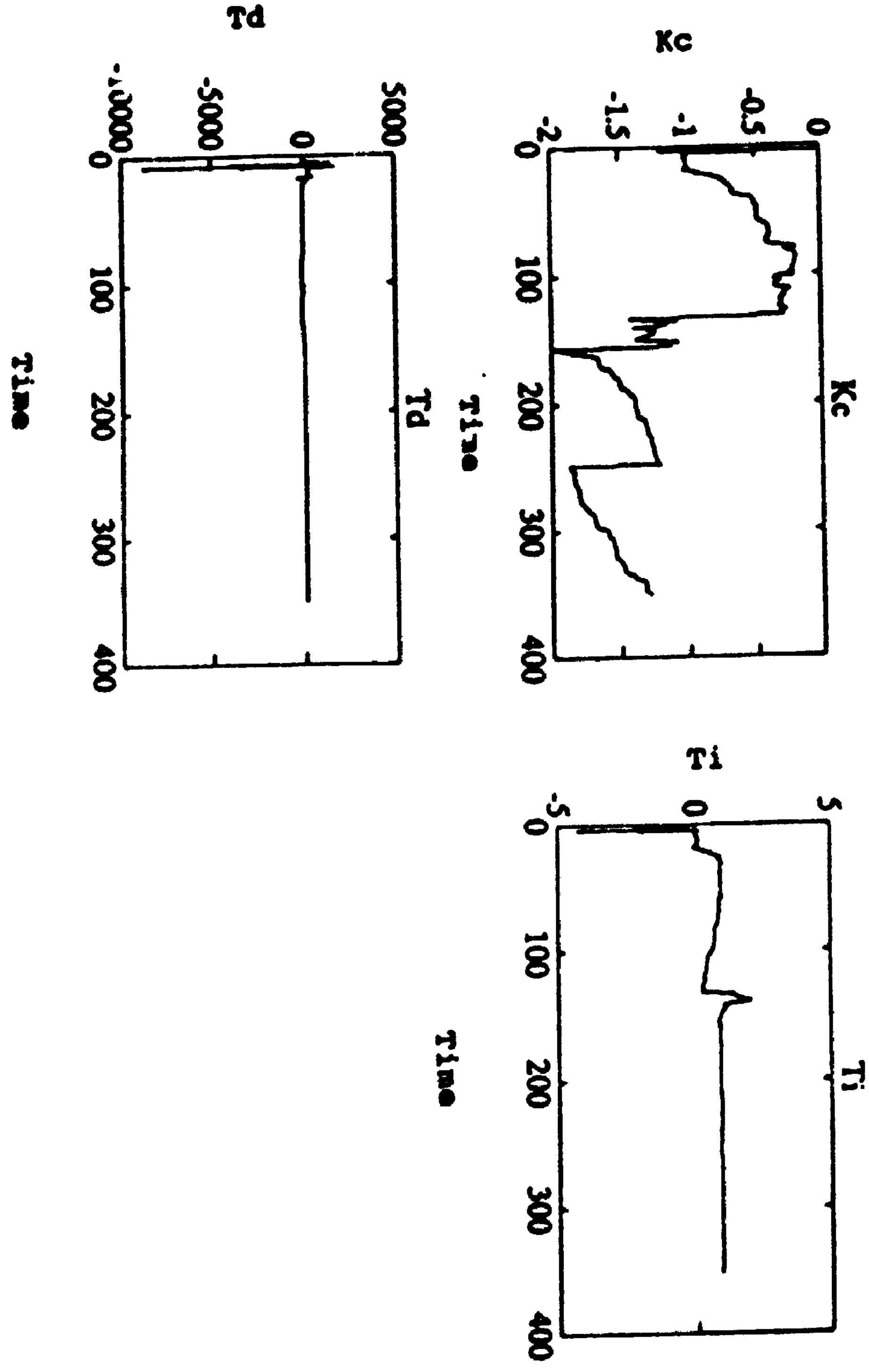

Figure 7.4: Variation of PID Parameters with Time

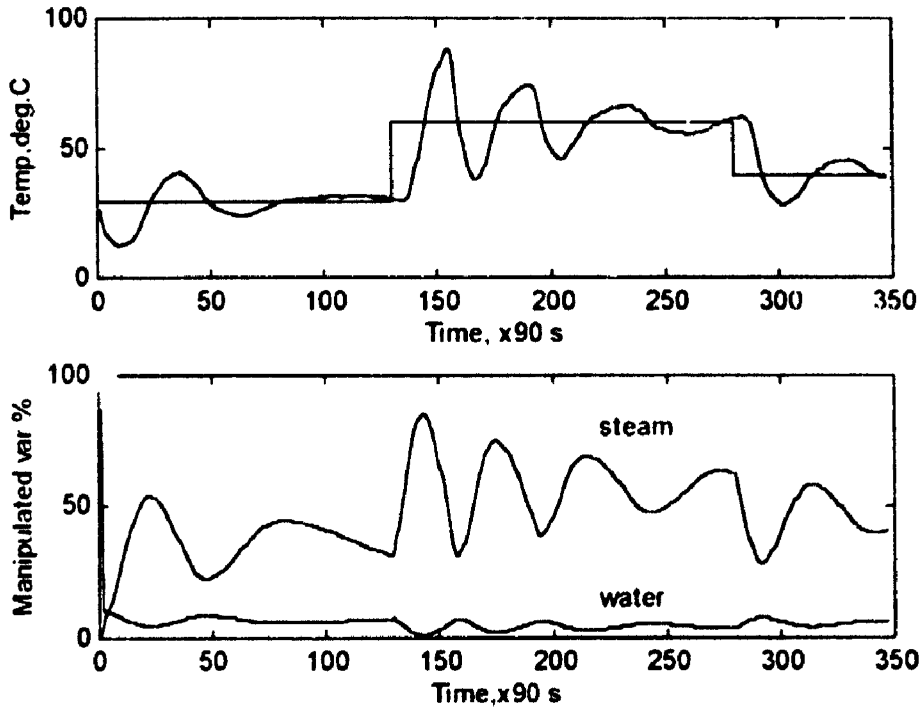

Figure 7.5: STPID Temperature Control of a Batch Reactor

7.4 Reactor Temperature Control Under GMV

Clarke and Gawthrop's (1979) Generalized Minimum Variance (GMV) controller, requires one to have some *a priori* knowledge of the process model. In particular its performance is quite sensitive to the process dead time. It also does not perform well for non-minimum phase processes. The controller output is given as

$$\nabla^d u_t = -\frac{\alpha(z^{-1})}{\gamma(z^{-1})} y_t \tag{7.6}$$

where

$$\alpha(z^{-1}) = \alpha_0 + \alpha_1 z^{-1} + \ldots + \alpha_m z^{-m}$$

$$\gamma(z^{-1}) = \gamma_0 + \gamma_1 z^{-1} + \ldots + \gamma_l z^{-l}$$

and

$l \geq s+p+k$

$m = r + \max\{q-k-1; p+d-1\}$

where s,r,p and q are the values of order of polynomials $\tilde{\omega},\delta,\phi$ and θ respectively (of Box-Jenkins model, L ɔx and Jenkins (1990).

The parameters $\theta^T=[\alpha_0,\alpha_1,\ldots,\alpha_m,\gamma_0,\gamma_1,\ldots,\gamma_l]$ can be updated every sampling interval via a recursive parameter estimator and used in the control law. For this experiment, the recursive parameter estimator used was a simple recursive least squares method with modifications suggested by Bierman (1977) for the same reasons as described above. Because the GMV algorithm is given in an implicit form, we can directly estimate the controller parameters.

7.4.1 Discussion of Results

Table 7.2 gives the summary of the initial conditions for the experimental run under GMV control. The forgetting factor λ is initially set at 0.9 and then increased exponentially until it approaches the value of 1. Introduction of the exponential forgetting factor implies the minimization of a time-weighted least squares function as discussed in section 7.3.1. A value of the constraint ζ which was chosen by trial and error was set at a value of unity. The setpoint was then changed from 30°C to 60°C and then finally to 40°C. The convergence of the estimated θ parameters is shown figure 7.6. It can be seen that converges of the θ parameter estimates to their true values was reached at each setpoint.

Figure 7.7 shows the variation of reactor temperature (process output) with time and that of steam flow and water flow with time (process inputs). From the figure one can see that the process output was kept at its setpoint under the GMV control despite the wide changes setpoint settings.

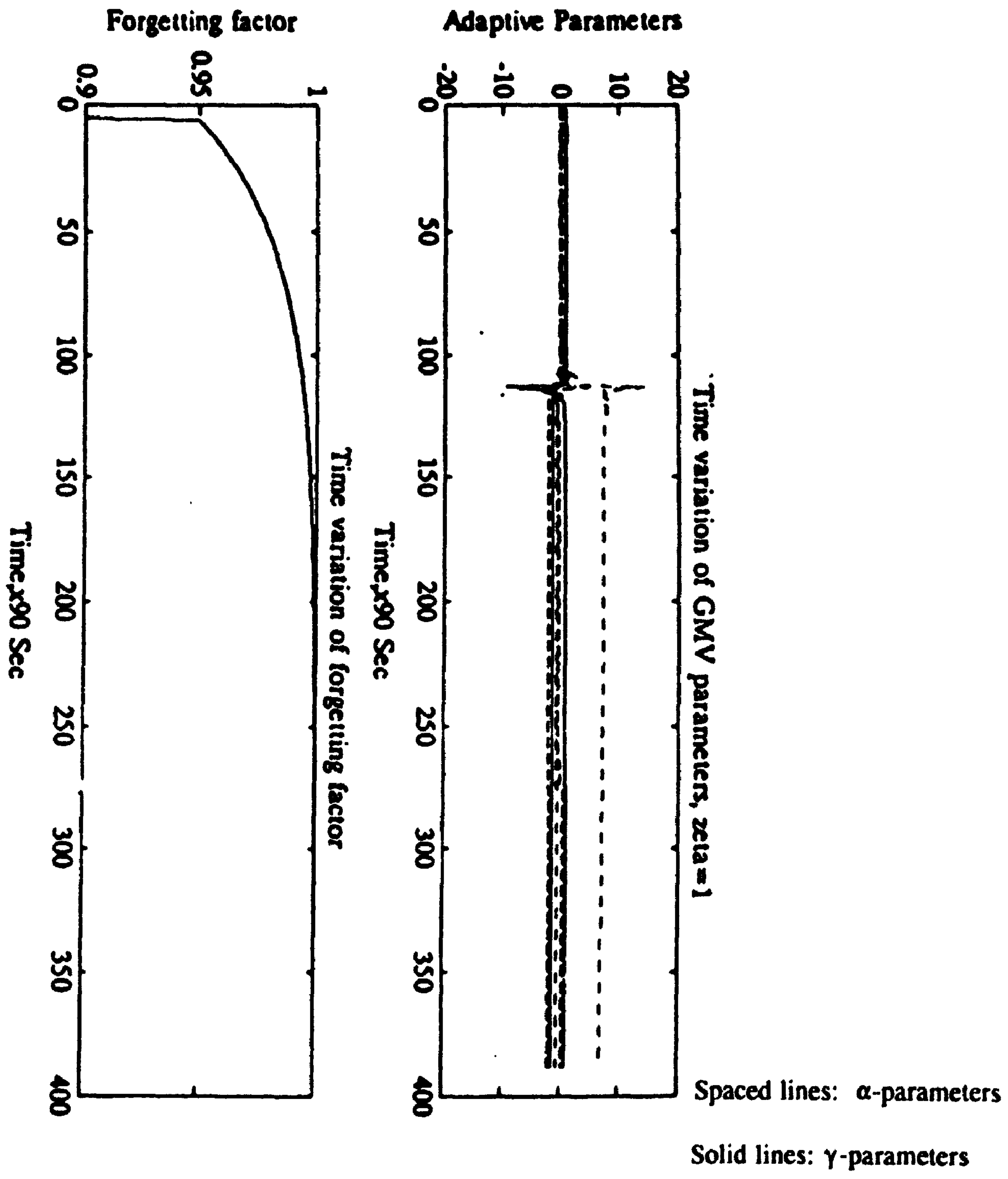

Figure 7.6: Time Variation of CMV Parameters and Forgetting Factor

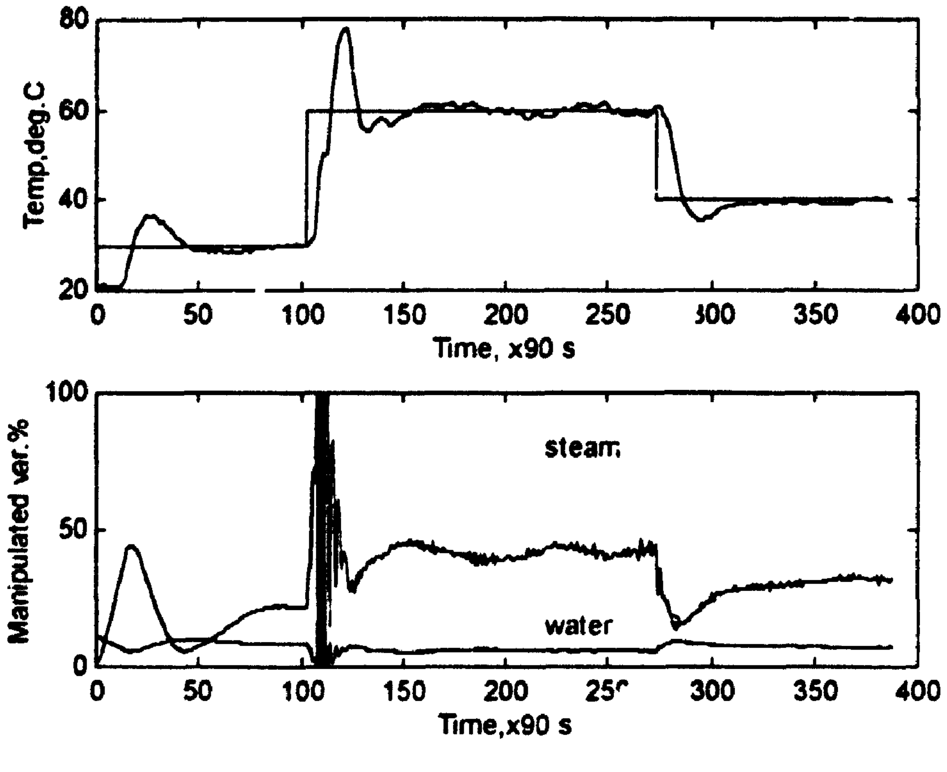

Figure 7.7: GMV Temperature Control of a Batch Reactor

Table 7.2: Initial conditions for experimental run with GMV

Run Type	GMV control algorithm parameters
Temperature setpoint change:	m=4 number of α parameters
From 30°C to 60°C	l=4 number of γ parameters
From 60°C to 40°C	b=1 time delay
	d=1 order of ∇ -for non stationary disturbance.
	ξ=1 constraint
	λ=0.9 - 1 exponentially increasing forgetting factor
	θ^T=[.9423 .2441 -.0291 -.1680 .9164 .2582 .0044 -.0381] (Initial values for θ)

First, we started by setting the setpoint to 30°C just like the case with the self-tuning PID. Figure 7.7 shows that at the beginning of the experiment the controller responds in a comparably smooth fashion which results in a smooth process response with an overshoot of approximately 9°C. The rise time and response or settling time were barely acceptable at approximately 1800 seconds and 7200 seconds respectively, because of a relatively slow controller response. Once steady state is reached the controller tracks the setpoint in a very tight manner. When the setpoint was stepped-up to 60°C from 30°C, again a few problems were encountered. The controller responds relatively fast and aggressively to increase the flow of steam and decrease the flow of cold water. The aggressive response resulted in an average overshoot (approximately 16°C), high response time (approximately 4500 seconds) and high rise time (approximately 1350 seconds). Since this is considered to be the most non-linear operating region, the performance of the GMV is commendable. When the setpoint was stepped-down to value of 40°C, the controller performance improved considerably. Just like the case with self-tuning PID the controller response was still not satisfactory but the undershoot was admissible.

7.5 REACTOR TEMPERATURE CONTROL UNDER GPC

The GPC by Clarke *et al.* (1987) discussed in chapter 3 is used to control the temperature of a batch reactor system. The incremental control vector obtained by minimizing the cost function is also given in chapter 3 as:

$$\Delta u=(G^{T}G+\lambda I)^{-1}G^{T}(w-f) \tag{7.9}$$

so that the current control law is given by:

$$u(t)=u(t-1)+g^{T}(w-f) \tag{7.10}$$

where g^T is the first row of $(G^TG+\lambda I)^{-1}G^T$.

So, at each sampling interval:

(1) given y(t) and previous values of y, u, the predictions of freely responding process are computed and compared with future set-points w, which may be known or assumed equal to w(t).

(2) for the given {N_1, N_2, NU, λ, y hat, w} the optimal control vector Δu is computed.

(3) the first element $u(t)=u(t-1)+\Delta u(t)$ is implemented and sequences are shifted ready for the next sampling interval.

Note that the GPC algorithm has an explicit form. We can therefore estimate the process parameters first and then use the estimated parameters in the controller design. Figure 7.8 illustrates the schematic diagram of the GPC controller used in the experiment under ©Matlab/Simulink environment and a modified version of SPACE (Scokaert *et al.*, 1995).

7.5.1 Discussion of Results

Table 7.3 gives the summary of the initial conditions for the experimental run under GPC control. Note that the forgetting factor is set at a constant value of 0.98. The control weighting λ which was also chosen by trial and error was set at a value of 0.181. The setpoint was then changed from 30°C to 60°C and then finally to 40°C.

Figure 7.9 shows the variation of reactor temperat..e (process output) with time and that of steam flow and water flow with time (process inputs). Figure 7.9 illustrates that the process output was kept at its setpoints under the GPC control despite the non-linearity of the process.

Table 7.3: Initial conditions for experimental run with GPC

Run Type	GPC control algorithm parameters
Temperature setpoint change: From 30°C to 60°C From 60°C to 40°C	N1=2, N2=4, Nu=1, d=1, λ=0.181, Δt=90 sec

Rules of thumb formulated in literature were used to arrive at the parameter settings of the GPC and of course fine tuning was necessary to achieve the desired performance. The minimum cost horizon N1 was set to 2, that is, N1 was set to be equal to d+1. The maximum cost horizon N2 was set to 4 after trying many different values. Our aim was to achieve quick controller response without causing too much controller activity or high overshoots. The control horizon NU was set to 1 and the control weighting λ was set to 0.181 after trying out a number of values. Higher values of λ did not seem to improve controller activity.

As before, at the beginning of the experiment the setpoint was set to 30°C. From figure 7.9 it can be seen that the controller responds in a very smooth fashion which results in a smooth process response with an overshoot of approximately 10°C. The rise time and response or settling time are small (approximately 270 seconds and 850 seconds respectively), due to a very fast controller response. This is an improvement over the previous two cases. Once steady state is reached the controller tightly maintained the reactor temperature at the desired value. When the setpoint was stepped-up to 60°C from 30°C, a few problems were encountered. The controller responded in an aggressive manner to increase the flow of steam and decrease the flow of cold water. This aggressive controller response resulted in an average overshoot (approximately 18°C) and small rise time (approximately 400 seconds). This operating range is considered to be the most non-linear and the performance of a GPC which is based on a linear model, though satisfactory, is not surprising. When the setpoint was stepped-down to a value of 40°C, the controller performance improved considerably. The undershoot was quite small and the controller response was very good. The above results show the advantage of GPC over the previous two strategies.

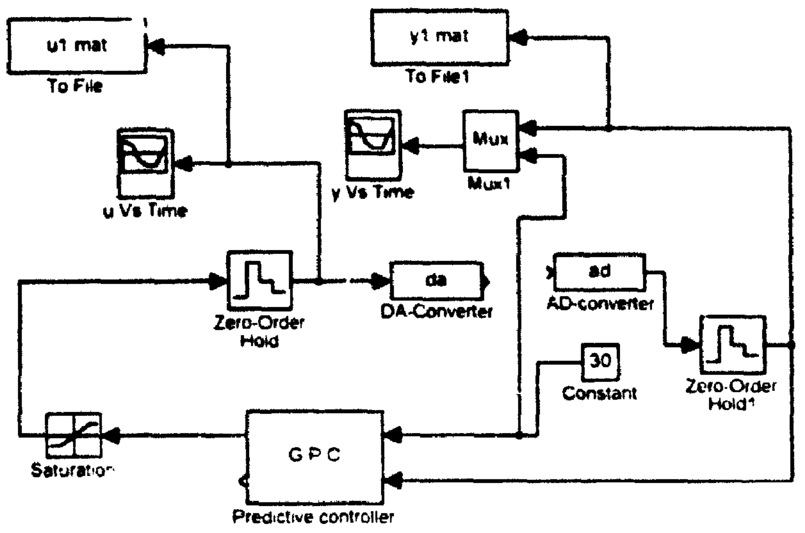

Figure 7.8: Simulink Setup for GPC Real-Time Run

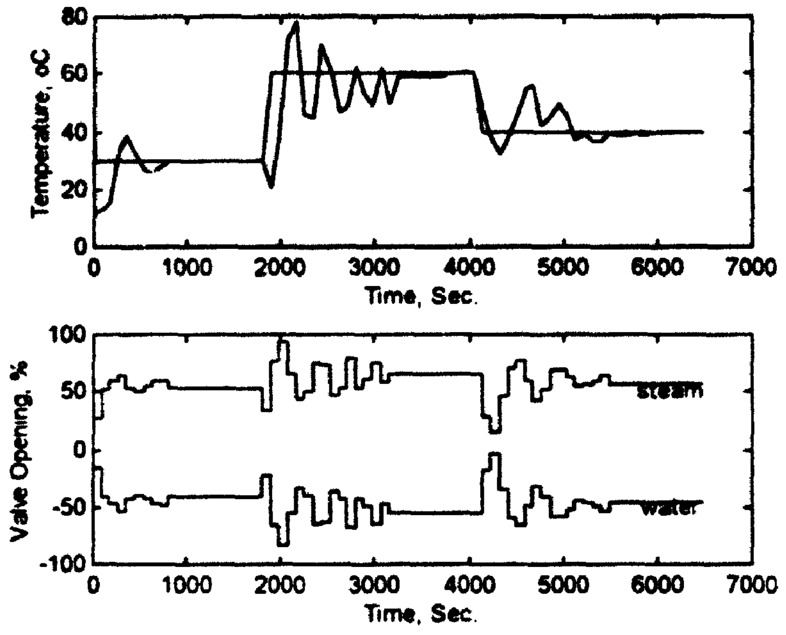

Figure 7.9: GPC Temperature Control of a Batch Reactor

7.6 REACTOR TEMPERATURE CONTROL UNDER NLGPC

7.6.1. A general form of NLGPC

The NLGPC discribed in chapters 4,5 and 6 is also used to control the temperature of a batch reactor system. It was shown in chapter 4, that by minimizing the cost function J with respect to vector x, we obtain x as:

$$x = (G^T G+\lambda I)^{-1} G^T (w^* - H\bar{x} - Fc) \qquad (7.11)$$

Equation (7.11) gives the optimal x vector, from which as in GPC, we require only the first element to implement the control law. In order to obtain u_{opt} from the computed x value, Newton's method can be used.

Note that the NLGPC algorithm also has an explicit form. We can therefore estimate the process parameters first and then use the estimated parameters in the controller design. Figure 7.10 illustrates the schematic diagram of the NLGPC controller used in the experiment taking advantage of ©Matlab/Simulink environment and a modified version of SPACE.

7.6.2 Discussion of Results

Table 7.4 gives the summary of the initial conditions for the experimental run under NLGPC control. Note that the forgetting factor is set at 0.98. The control weighting λ which was also chosen by trial and error was set at a value of 9.6. The setpoint was then changed from 30°C to 60°C and then finally to 40°C.

Figure 7.11 shows the variation of reactor temperature (process output) with time and that of steam flow and water flow with time (process inputs). It can clearly be seen that the process output was kept at its setpoints under the NLGPC control despite the non-linearity of the system and the wide changes in the setpoints.

Table 7.4: Initial conditions for experimental run with NLGPC

Run Type	NLGPC control algorithm parameters
Temperature setpoint change:	$N1=2$, $N2=5$, $Nu=1$, $d=1$, $\lambda=9.6$,
From 30°C to 60°C	$\Delta t=90$ sec., A = 1st order B1 = 1st order
From 60°C to 40°C	B2 = 1 order

In order to arrive at the tuning parameters shown in table 7.4, we initially set these parameters to values determined by the knowledge we acquired through experience. Then fine tuning was done, essentially by trial and error, to arrive at the values shown.

The minimum cost horizon N1 was set to $d+1$ so that the tracking error over the entire prediction horizon is included in the objective function. For example, if N1 was set to d+2 the tracking error at t+d+1 would not be included in the criterion. The maximum cost horizon N2 was set to 5. This was necessary since we needed to design a controller with a quick response, at the same time avoiding high overshoots and undesirable oscillation in the process output. From experience it was noted that increasing the value of N2 to a large value slows down the response of the process under NLGPC. NU was set to 1, lowest possible value since, from experience, large values of NU tend to slow down the computation and may also result in poor performance. The control weighting λ was set to 10.6, after trying out many values. This value gave the best controller activity and best overall closed-loop robustness.

At the beginning of the experiment the setpoint was set to 30°C just like the previous runs. From figure 7.11 it can be seen that the controller responds in a very smooth fashion which results in a smooth process response with an overshoot of approximately 8°C. As before with GPC, the rise time and response or settling time were very small (approximately 270 seconds and 85 seconds respectively), due to a fast controller response. Once steady state is reached the controller tightly maintains the reactor temperature at the desired value. When the setpoint was stepped-up to 60°C from 30°C, the NLGPC utilized its non-linear nature to keep the reactor temperature at its desired value with minimal strain. The controller quickly responds to increase the flow of steam and decrease the flow of cold water in a relatively smooth fashion. This transition resulted in a low overshoot (approximately 12°C) and small rise time (approximately 350 seconds). This operating range is considered to be the most non-linear and the excellent performance of the NLGPC proves its advantage over the linear model based controllers. When the setpoint was stepped-down to a value of 40°C, the controller performance remained good. The undershoot was quite small and the controller response was very satisfactory.

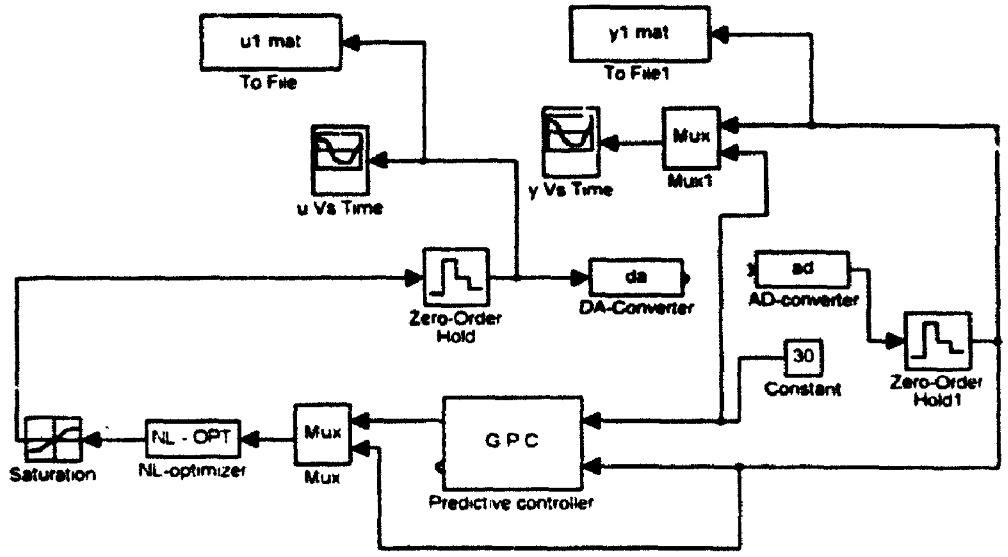

Figure 7.10: Simulink Setup for NLGPC Real-Time Run

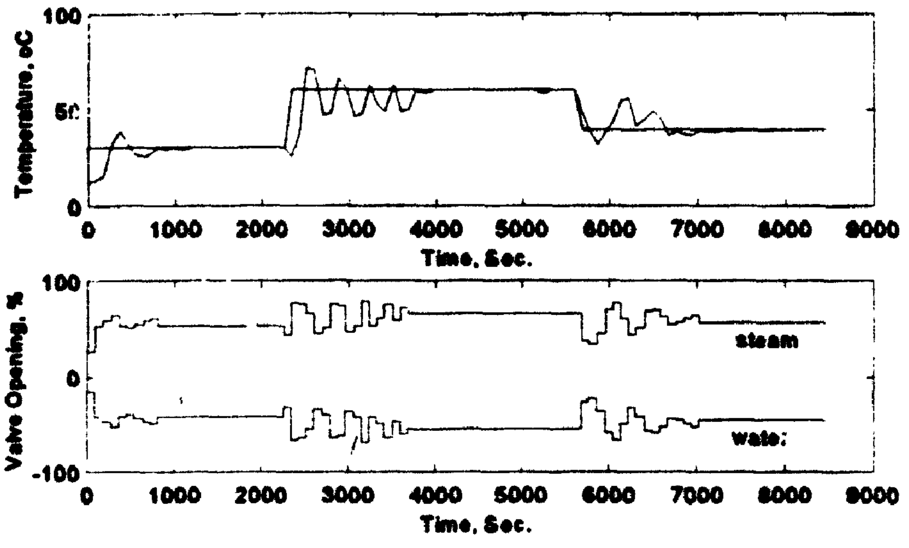

Figure 7.11: NLGPC Temprature Control of a Batch Reactor

7.7 COMPARISON OF STRATEGIES

From the analysis summarized in the tables below the NLGPC, GPC, GMV and the constrained STPID controllers can handle the non-linear, time varying processes with different levels of ability. The predictive algorithms (NLGPC and GPC) outperformed both the GMV and the self-tuning PID controllers.

When the setpoint was set at 30°C all the controllers had an overshoot of approximately 10°C (see table 7.5) except for the STPID which had a 5°C overshoot. However, the response or settling time and rise time were far much better with the predictive controllers, 850 sec. and 270 sec. respectively, and worst with the self-tuning PID. The controller activity was more or less moderate for all controllers.

When the setpoint was stepped-up to 60°C more non-linearity was experienced and hence the linear model based controllers had the most difficulty (see table 7.6). The overshoot was more or less the same for all controllers except for the self-tuning PID which had the largest overshoot of 26°C. The NLGPC had the best response or settling time (1500 seconds) and rise time (350 seconds). The controller activity was much more moderate for both the NLGPC and the self-tuning PID.

When the setpoint was stepped-down from 60°C to 40°C the process exhibited less non-linearity in its dynamics and thus controller activity was quite good for all controllers. As shown in table 7.7 the response time and rise time were much better with the NLGPC and GPC controllers (1300 seconds and 90 seconds respectively for both cases).

It is clear that the predictive controllers outperformed the self-tuning PID and the GMV controllers in controlling the non-linear reactor system. Again because of the wide range and severe non-linearities here, the NLGPC has an advantage over the GPC which, nevertheless, does an admirable job.

Table 7.5: 30 °C Setpoint Tracking

Controller	Max. Overshoot (°C)	Response Time (Sec.)	Rise Time (Sec.)	Controller Activity
NLGPC	8	850	270	Moderate
GPC	10	850	270	Moderate
GMV	9	7200	1800	Moderate
STPID	5	9000	2250	Moderate

Table 7.6: 60 °C Setpoint Tracking

Controller	Max. Overshoot (°C)	Response Time (Sec)	Rise Time (Sec.)	Controller Activity
NLGPC	18	1500	350	Moderate
GPC	18	1550	400	High
GMV	18	4500	1350	High
STPID	26	9900	900	Moderate

Table 7.7: 40 °C Setpoint Tracking

Controller	Max. Undershoot (°C)	Response Time (Sec.)	Rise Time (°C)	Controller Activity
NLGPC	17	1300	90	Moderate
GPC	18	1300	90	Moderate
GMV	9	3600	1350	Moderate
STPID	8	3600	1350	Moderate

Table 7.8: Controllers tested and their inial conditions

Controller	Parameter Settings
NLGPC	N1=2, N2=4, Nu=1, d=1, λ=9.6, Δt=90 sec.
GPC	N1=2, N2=4, Nu=1, d=1, λ=0.181, Δt=90 sec.
GMV	delay b=1, number of α-par=4, number of β-par=4, cont. weig ξ=1 ff=0.9-1 exp., Δt=90 sec.
STPID	delay b=1, cont. Weig. ξ=1, ff=0.9-1(exp.), Δt=90 sec.

7.8 CONCLUSIONS

Despite the nonlinear and time-varying characteristic of the reactor, good control with differing degrees was obtained for a wide operating range using the NLGPC, GPC, GMV and STPID control strategies. All these controllers were able to identify the process on-line and excellent convergence of parameters was obtained for each case. The controllers proved to be robust and perform reasonably well under noisy conditions and changing system parameters. In spite of this good average control performance, the two model predictive schemes, GPC and NLGPC. showed distinctive advantages over the simpler schemes. Closed-loop system response and settling was significantly improved. It was also easier to choose reasonable guesses for the controller parameters such as prediction and control horizons. The limitations of the linear based GPC were also apparent for this high non-linear system. This allowed the NLGPC to further improve the control performance and provide the most suitable control run. Note that in the experimental setup one cannot increase the severity of the non-linear nature of the controlled process beyond a certain point, that way, the GPC and all the other linear controllers did not fail as was the case with simulation where process non-linearity could easily be varied.

CHAPTER 8

CONCLUSIONS

8.1 GENERAL

A new non-linear generalized predictive control algorithm was developed based on a class of non-linear dynamics that can be suitably modelled using Volterra and Hammerstein structures. Non-linear Generalized Predictive Control (NLGPC) showed an advantage over generalized predictive control (GPC) in that NLGPC controlled processes with high non-linearity in input signal very well. A number of simulation runs were used to demonstrate the performance of the NLGPC and compare it with a GPC. The NLGPC performed very well when used to control an adiabatic continuous stirred-tank reactor (CSTR) through simulation.

Analytical expressions for the gradient matrix and the Hessian matrix were also obtained for the optimal control solution using a second order NLGPC algorithm. The results allow for reduction in on-line calculational load. Typical computation time is reduced by 50% compared to the case where a purely numerical approach is taken. An additional issue of the choice of penalization in the objective function is considered. The performance of two NLGPC algorithms based on different criterion functions were evaluated using a number of simulation runs. The first algorithm was based on a quadratic cost function with penalty on the input signal. For the second algorithm the 'pseudo' input was penalised which in turn contains the manipulated variable. It was shown through simulation that selection of control weighting style is very important in determining the controller performance.

Despite the nonlinear and time-varying characteristic of the batch reactor system, good control with differing degrees was obtained for a wide operating range using the NLGPC, GPC, GMV and STPID control strategies. All these controllers were able to identify the process on-line and excellent convergence of parameters was obtained for each case. The controllers proved to be robust and perform reasonably well under noisy conditions and changing system parameters. In spite of this average control performance, the two model predictive schemes, GPC and NLGPC, showed distinctive advantages over the simpler schemes. Closed-loop system response and settling was significantly improved. It was also easier to choose reasonable guesses for the controller parameters such as prediction and control horizons. The limitations of the linear based GPC were also apparent for this highly non-linear system. This allowed the NLGPC to further improve the control performance and provide the most suitable control run.

8.2 SUGGESTIONS FOR FUTURE STUDIES

The NLGPC algorithm was extensively tested through simulation. However, only one type of experiment was used to evaluate this controller on a laboratory scale. Although the process chosen for the experiments was non-linear and exhibited non-minimum phase properties, a number of other processes or types of experiments should be tried to further explore the capabilities of this controller. Such experiments may include control of concentration or pH in a reactor, control of biochemical processes and control of bottoms or top product of a distillation column. The NLGPC should eventually be applied to industrial processes to see whether or not the results can be extrapolated from those obtained in the laboratory and simulation.

Although extensive simulation studies have been carried out to test the performance of the NLGPC on non-linear processes, no stability or robustness proofs have been done. It is therefore suggested that these proofs should be carried out and a design with guaranteed stability be obtained.

Another approach which is very desirable for real processes is to consider constraints while designing the NLGPC algorithm. This means that an optimal solution should be obtained which does not violate constraints on both input and output variables. This approach would be to add constraints to the objective function and solving the non-linear problem numerically.Soft constraints, using lagrange multipliers would be the simplest. Hard constraints would be more of a challenge.

APPENDIX A

REAL-TIME CONTROL

A.1 INTRODUCTION

The very first uses of real-time computing were in military applications related to missile firing and radar networks. The computers developed for this application were later adapted for industrial use. However, the growth of real-time applications was held back due to the high cost and need for alternative backup systems in case of computer failure.

The invention of the minicomputer was a major breakthrough in terms of the cost of building real-time systems. Computers such as Digital Equipment corporation's PDP series of minicomputers found widespread application as a data acquisition and control device in the laboratory. The 1960s and 1970s can be regarded as the era of minicomputers in this field.

Another major breakthrough came with the arrival of the microprocessor. The most important feature these microprocessor-based systems brought to real-time computing was the ability to distribute the computing tasks among many processors, thus increasing the reliability of such systems. The use of large-scale integration (the ability to put a large number of electronic circuits on small silicon wafers) led to decreasing costs.

One of the microcomputers (a computer based on microprocessor) that found widespread application in the laboratory was the Apple II computer (Joseph, 1989). This can be attributed to its low cost (from mass marketting), ease of use, and open architecture. The latter allows easy interfacing of external (peripheral) devices to the computer. Soon after the introduction of the Apple II computer, a number of companies began to manufacture and market interface boards that allow easy connection between the data acquisition and control signals and the computer. User-friendly software was also available. The major drawback of the Apple II was the limited capability of its microprocessor.

With the introduction of the IBM PC, microcomputers became as powerful as the minicomputers of the past. Again, the open architecture of this system made it a very popular candidate for use in data acquisition and control applications. Today, there are a large number of companies that manufacture and market data acquisition and control hardware and software for the IBM family of computers and compatibles.

In this chapter we describe the major components of a microcomputer-based data acquisition and control batch reactor system available in our laboratory, Advanced Process Control Research Laboratory, which are: the Personal Computer, the thermocouple (sensor), valves (actuators), related instrumentation that provide a connection with the reactor (A/D D/A converter) and the software (programs) that drives the microcomputer hardware.

A.2 HARDWARE

A.2.1 Data Acquisition and Control System

The computer system used in our simulation and experimental runs is a 486DX IBM compatible PC. It offers sufficient power to monitor 10-100 sensors (we had only one sensor, the thermocouple), high speed data acquistion, a fairly good amount of data processing, and a large storage capacity. The computer monitor was used to display graphical trends of the process variables in real time. The computer also sends control signals to the reactor system via hardware interfacing.

A.2.2 Sensors and Actuators

The thermocouple (measurement sensor for the reactor system) generates an analog signal. It uses a voltage generated by the junction of two metals at different temperatures to generate an output signal. The signal generated is very weak and noisy, therefore, it requires some amplification, level shifting and noise filtering which is referred to as signal conditioning.

Air to open control valves were used to manipulate steam flow and water flow. The valves are connected to the I/P (Current to Pressure) transducers which receive a 4-20mA analog signal from the computer.

A.2.3 Analog Interface

We used a Data Translation DT2801 series board as the analog-to-digital (A/D) digital-to-analog (D/A) converter to accept a thermocouple reading (analog input) and also send a signal to the I/P transducers (analog output) which manipulate the steam and water control valves. The signal generated by the thermocouple is conditioned by the DT board signal conditioner with the voltage levels of -10 to +10 range (bipolar). This signal is then converted to digital form by the A/D converter. A/D converter is interfaced to the microprocessor bus which transfers the data to the computer CPU for computation.

We setup a control algorithm through software (QBASIC and C/C++) which computes the value of the control signal (digital value). This digital value is sent to the D/A converter via a microprocessor bus which generates an analog voltage. The output voltage is then converted to a 4-20mA analog signal which is received by the I/P transducers which are responsible for the opening/closing of the steam and water valves.

A.3 SOFTWARE

For simulation, the software is needed to do the necessary numerical computation to generate a response for a given process model driven by a controller output. The aim of the controller is to keep the process response at a desired setpoint despite changes due to disturbance or setpoint.

For real-time control the software serves similar purposes as in simulation except that here we deal with the "real-world". The following is the outline of the software driver for the reactor teperature control system:

(i)- Start the run and enter the necessary data (e.g. sampling interval, horizons, initial guesses, etc.).

(ii)- Subroutine to read temperature (analog input).

(iii)- Subroutine to compute control moves (the control algorithm).

(iv)- Subroutine to send signal to I/P control valve transducers (analog output).

(v)- Subroutine to display temperature trend, control moves, etc.

(vi)- Subroutine for control menu (keyboard event handler, e.g. terminate program).

(vii)- Subroutine to trap errors.

A.3.1 Running a Matlab Simulation

A number of script files were written in Matlab to perform simulation and control of different process models some of which were derived from energy and mass balances and others obtained as statistical ARIMA, CARIMA, Box and Jenkin's (BJ) or Hammerstein/Volterra models.

Four different custom Matlab libraries are provided with the necessary files to run simulations with four different controllers. It may be necessary for the user to edit the data files or even the script files to make changes to the process models and controller parameters to suit their needs. There is no need to edit the function files called by the script file.

The custom Matlab library called STPID under the directory STPID provide the user with the necessary files to run simulations with a self-tuning PID controller. Once under this directory one can perform simulations by typing "rlspid" at the matlab prompt. The program will then prompt the user to enter the necessary data or parameters. A custom Matlab library to run simulations with the GMV controller is called GMV under the direcory GMV. Once under this directory the user can perform simulations by typing "rlssim" at the matlab command line. The program will also querry the user to enter the necessary data or parameters. Another custom Matlab library to run simulations with the GPC algorithm is called GPC and is provided under the directory GPC. By typing "gpc_ex" at the Matlab command line one can perform simulations under this directory. It may be necessary for the user to edit the script file "GPC_EX.M" to make changes to the process model and controller parameters. The script file contains a data table which can easily be edited to suit one's needs. Lastly, a custom Matlab library called NLGPC under the directory NLGPC is provided with the necessary M-files to run simulations with the NLGPC algorithm. Once under this directory the user can type "nlgpc_ex" at the Matlab command line to perform the simulations. If there is a need to make changes to the process model and controller parameters, the user can edit the script file "NLGPC_EX.M". The script file contains the data in form of a table which can easily be edited. If there is any need to make any further changes one can edit the major function files "GPCSIM.M" and "NLGSIM.M" for the GPC and NLGPC algorithms respectively.

A.3.2 Setting up and Running a Simulink Simulation

Briefly described below is how the user may setup the simulink worksheet for use with three different types of control strategies, and how Simulink may be run.

A.3.2.1 Setting up a Simulink Simulation

The custom Matlab libraries mentioned in section A.3.1 above are needed in order to run the Simulink simulations involving blocks build for the self-tuning PID, GPC and NLGPC algorithms. This means that the directories leading to STPID, GPC, and NLGPC libraries should be included in the Matlab path in the "STARTUP.M" file. Another Simulink GPC simulator used is the one provided by Scokaert et al. (1995), Simulink Predictive Control Environment (SPACE). The SPACE directory which leads to space files is also included in the Matlab path. By typing "adaptall" a window containing all the three controller blocks (STPID, GPC, NLGPC) is displayed. The user can now select any block from here to connect to the process for simulation. Also typing "space" gives rise to a window containing a constrained GPC controller block. Figures A.1, A.2, A.3 and A.4 show sample simulation setup for simulation under self-tuning PID, GPC, NLGPC and SPACE's constrained GPC.

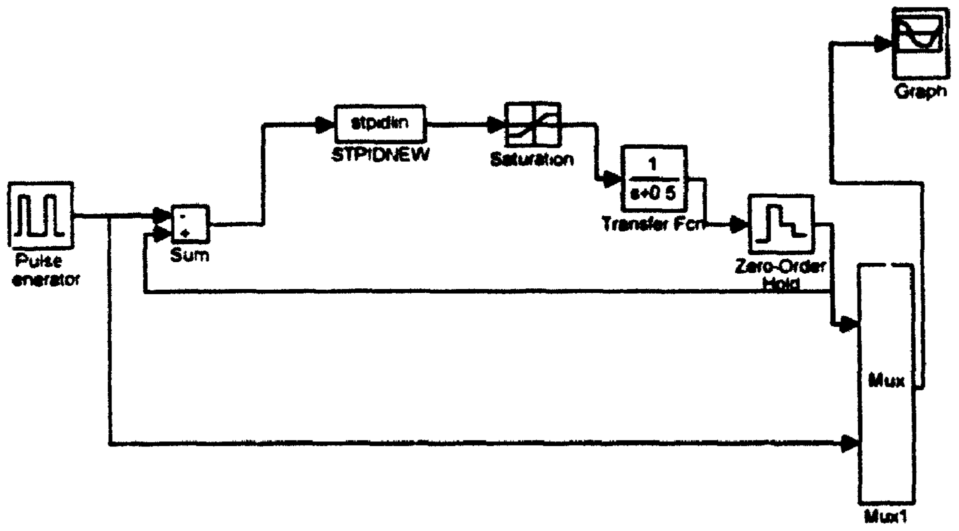

Figure A.1: Simulink Setup for Self-tuning PID (STPID) Simulation Run

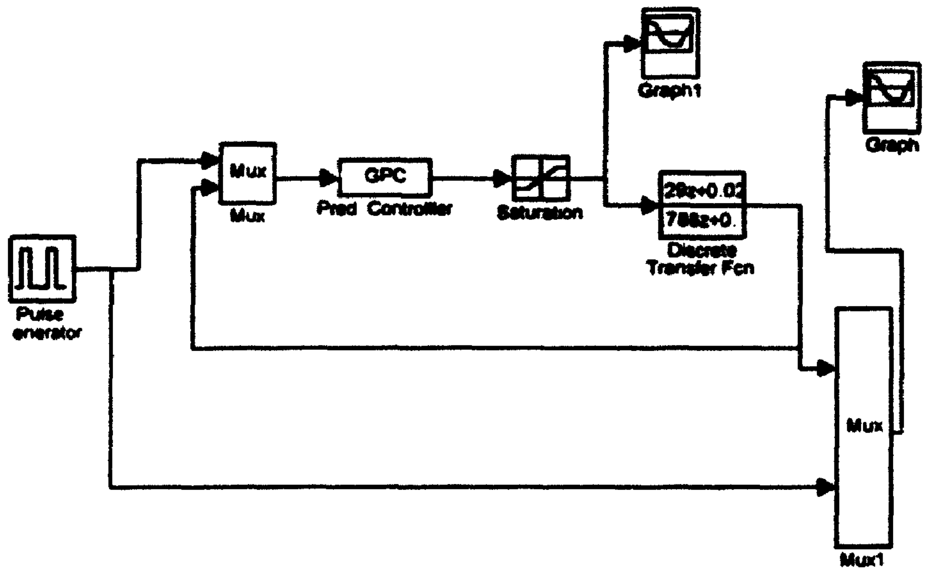

Figure A.2: Simulation Setup for GPC Simulation Run

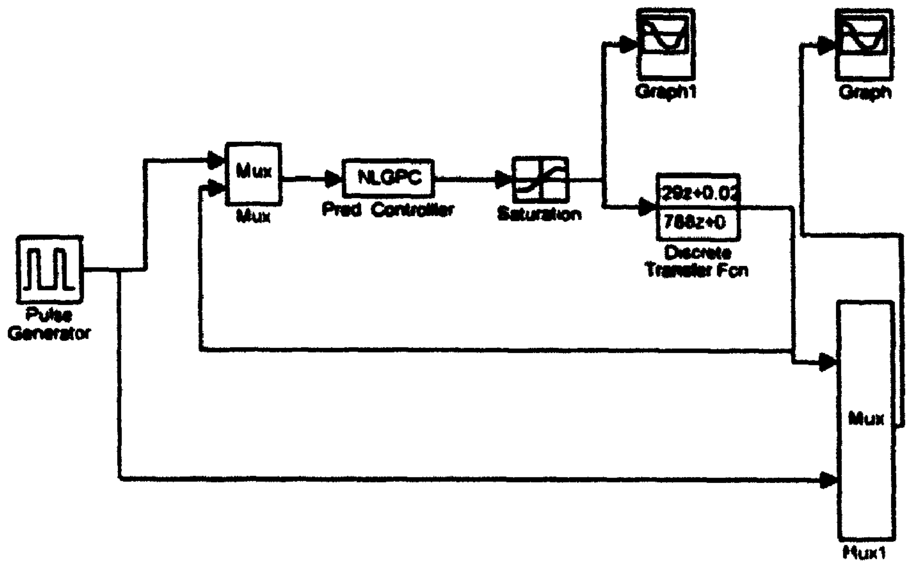

Figure A.3: Simulink Setup for NLGPC Simulation Run

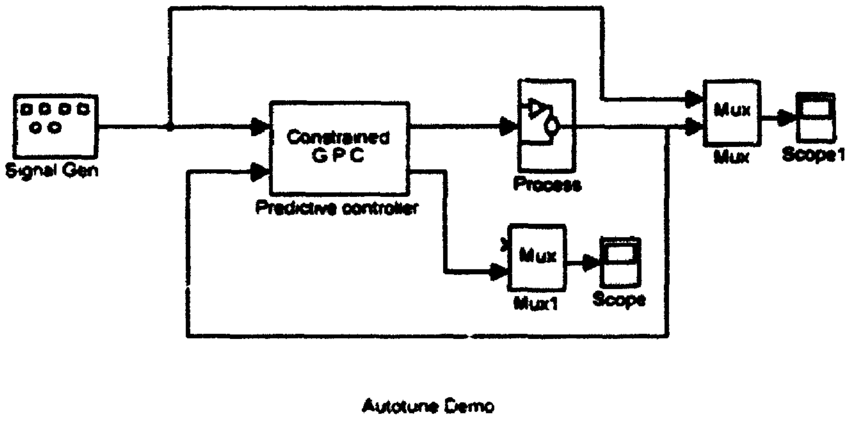

Figure A.4: Simulink Setup for GPC (SPACE) Simulation Run

A.3.2.2 Running a Simulink Simulation

The Simulink user's guide details simulation parameters that must be adequately set before running a simulation. Of particular interest are "Min Step Size" and "Max Step Size" items in the simulation control dialog window. Setting both parameters to the controller sampling interval results in a faster simulation than when the maximum step size is smaller than the controller sampling interval. In the latter case, however, the inter-sample behaviour of the process output is captured, which may be useful for analysis of simulation results. Setting these parameters in a proper way can also be very useful if simulink is used for real-time during which proper memory management is very vital.

A.3.3 Running a QBASIC or C/C++ Programs for Real-Time Control

Real-time temperature control of a batch reactor can be achieved by running anyone of the custom written software packages under the directory PRO. These packages include PRO, PRO2, PRO3 (all written in QUICKBASIC) and REALGPC (written in C).

The PRO software package for real-time control with either Generalized Minimum Variance (GMV) or conventional PID is envoked by typing "pro" at the DOS command line. PRO2 package which is envoked by typing "pro2" at the DOS command line is basically used to collect data for open-loop reactor identification. PRO3 package which is also envoked at the DOS command line by typing "pro3" is a more complete package which enables the user to select from three controllers: GMV, self-tuning PID and the standard PID.

All these programs are stand-alone packages which can be run on any personal computers provided the necessary hardware interfacing is taken care of. The programs could be used to control any processes with minor adjustments, if at all needed. A typical screen display of PRO3 shows the trend of the process output (controlled variable), the trend of the process input (manipulated variable), control menu and other necessary parameters.

REALGPC is also a stand-alone software package which enables the user to implement a GPC algorithm on the process. By typing "realgpc" at the DOS command line one can run this program. This program is not complete as some of the menu control handlers are still being developed.

A.3.4 Setting up and Running Integrated Simulink and C/C++ Programs for Real-Time Control

One can have access to Matlab's handy tooboxes / programs and simulink's worksheet by integrating C/C++ programs with Matlab / Simulink environment. Here real-time control is performed in Simuilink environment by making either DOS calls (if using DOS-based driver) or windows calls (if using windows-based driver for external I/Os). For our case we used DOS-based drivers (a windows-based driver is still being developed) to control the temperature of a batch reactor with a Matlab-coded GPC algorithm and a Matlab-coded NLGPC algorithm.

The drivers were developed as Simulink blocks (making DOS calls to C/C++

written programs) which can be connected to the external process. Figure A.5 shows the connections without any controller feedback. Under the directory EXPLINK, the files ADDA.M, ADDAOX.M and ADDAOX2.M can be found. Figure A.6 was generated by typing "adda" at the Matlab command line. One can generate figure A.6 by typing "addaox" at the Matlab command line. This diagram illustrates the actual connection of GPC (from SPACE) to thereactor system making use of Matlab's toolboxes in real-time. Figure A.7 was generated by typing "addaox2" at the Matlab command line. Figure A.7 shows the connection of NLGPC to the reactor also making use of Matlab's m-files and toolboxes.

By clicking on the "Simulation" and then "Start" in Simulink worksheet for all the above examples illustrated a real-time control of a batch reactor can be started. Note that parameter setting may be necessary, for example, sampling interval, cost and control horizons, time delay and process model parameters.

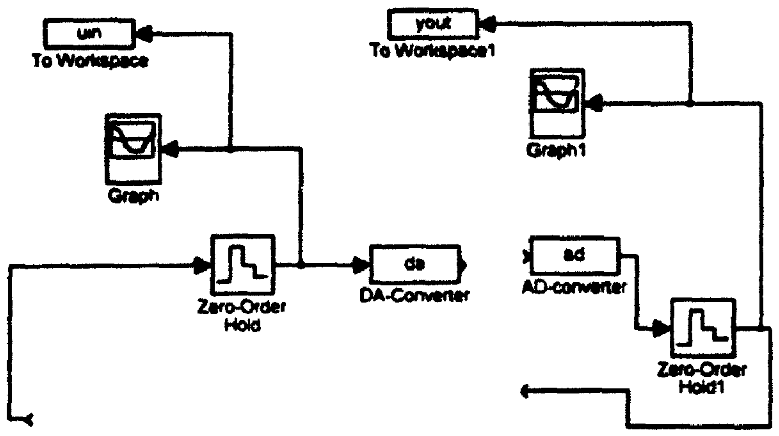

Figure A.5: Simulink Setup illustrating A/D D/A Connections to a Real Process without a Controller

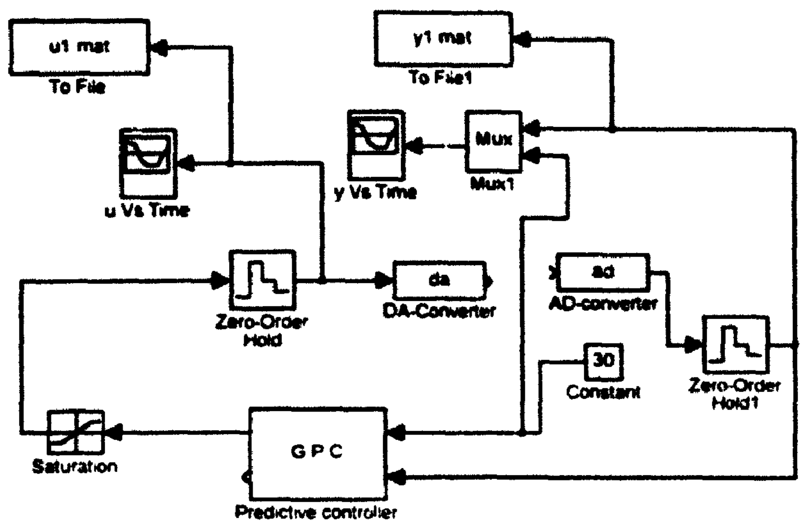

Figure A.6: Simulink Setup for GPC Real-Time Run

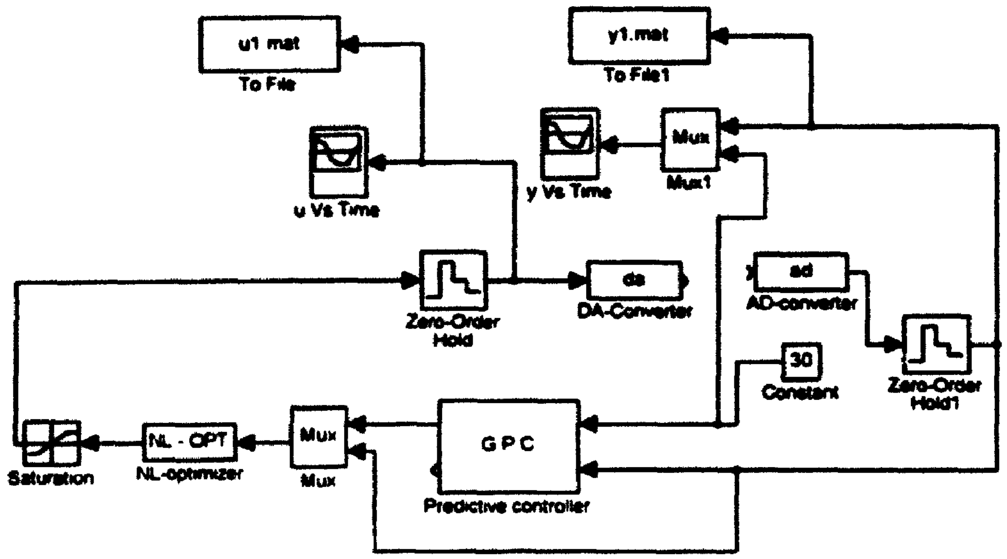

Figure A.7: Simulink Setup for NLGPC Real-Time Run

APPENDIX B

JACOBIAN AND HESSIAN EQUATIONS OF QUADRATIC NLGPC

The following is a matrix representation of the second partial derivative of J with respect to **u**:

$$\frac{\partial^2 J}{\partial u^2} = \begin{bmatrix} \frac{\partial^2 J}{\partial u_0^2} & \frac{\partial^2 J}{\partial u_0 \partial u_1} & \cdots & \frac{\partial^2 J}{\partial u_0 \partial u_{N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial u_{N-1} \partial u_0} & \cdots & \cdots & \frac{\partial^2 J}{\partial u_{N-1}^2} \end{bmatrix} \tag{B1}$$

First, obtain the second partial derivative of each of the elements in the above matrix

$$\frac{\partial}{\partial u_i}\left(\frac{\partial J}{\partial u_j}\right) = \frac{\partial}{\partial u_i}\left[2v^T \frac{\partial v}{\partial u_j} + 2\lambda u_j\right] \tag{B2}$$

Rearranging the above equation we get

$$\frac{\partial}{\partial u_i}\left(\frac{\partial J}{\partial u_j}\right)=\frac{\partial}{\partial u_i}\left[2v^T\frac{\partial v}{\partial u_j}\right]+2\lambda\delta_{ij} \qquad \text{(B3)}$$

Substituting for the term in square brackets on the RHS using equation (4.75) we obtain

$$\frac{\partial}{\partial u_i}\left(\frac{\partial J}{\partial u_j}\right)=\frac{\partial}{\partial u_i}\left[2v^Tg_j^{(1)}+4u_jv^Tg_j^{(2)}v^T\frac{\partial v}{\partial u_j}\right]+2\lambda\delta_{ij} \qquad \text{(B4)}$$

Rearranging the above equation and differentiating through the RHS we have

$$\frac{\partial}{\partial u_i}\left(\frac{\partial J}{\partial u_j}\right)=2\frac{\partial v^T}{\partial u_i}\left[2v^Tg_j^{(1)}+24u_jv^Tg_j^{(2)}\right]+4\delta_{ij}v^Tg_j^{(2)}+2\lambda\delta_{ij} \qquad \text{(B5)}$$

Again we use equation (4.75) for substitution in the above equation

$$\frac{\partial}{\partial u_i}\left(\frac{\partial J}{\partial u_j}\right)= 2\left[g_j^{(1)}+2ug_j^{(2)}\right]^T\left[g_j^{(1)}+2ug_j^{(2)}\right]+4\delta_{ij}v^Tg_j^{(2)}+2\lambda\delta_{ij} \qquad \text{(B6)}$$

Multiplying through and collecting terms we get

$$\frac{\partial}{\partial u_i}\left(\frac{\partial J}{\partial u_j}\right) = 2\left[g_i^{(1)^T} g_j^{(1)} + 2g_i^{(1)} u_j g_j^{(2)} + 2u_j g_i^{(2)^T} g_j^{(1)} + 4u_{u_j}(g_i^{(2)^T} g_j^{(2)})\right] + 4\delta_{ij} v^T g_j^{(2)} + 2\lambda\delta_{ij} \quad \text{(B7)}$$

Using equation (B7) and putting all elements in a matrix form we arrive at a second partial derivative of J with respect to u:

$$\begin{aligned}\frac{\partial^2 J}{\partial u^2} &= 2G_1^T G_1 + G_1(G_2 diag(u)) + 2(G_2 diag(u))^T G_1 \\ &+ 4(G - 2diag(u))^T (G_2 diag(u)) + 2\lambda I + 4diag(v^T G_2)\end{aligned} \quad \text{(B8)}$$

REFERENCES

Agarwal, M. and D.E. Seborg (1987). Self-tuning controllers for non-linear systems. *Automatica*, 23, (2), 204.

Åström, K.J. (1974). A self-tuning regulator for non-minimum phase systems. Report TFRT-3113, Department of Automatic Control, Lund Institute of Technology.

Åström, K.J. (1987). Adaptive feedback control. *Proc. IEEE* 75, 185.

Åström, K.J. and P. Eykhoff (1971). System identification - a survey. *Automatica*, 7, 123.

Åström, K.J. and B. Wittenmark (1973). On self-tuning regulators. *Automatica*, 9, 185.

Åström, K.J. and B. Wittenmark (1980). Self-tuning controllers based on pole-zero placement. *IEE Proc.*, 127, D, 120.

Åström, K.J. and B. Wittenmark (1989). *Adaptive Control.* Addison-Wesley.

Atherton, D.P. (1975). *Non-linear Control engineering.* Van Nostrand Reinhold.

Bellman, R. (1957). *Dynamic Programming.* Princeton University P:ess.

Bellman, R. (1961). *Adaptive Processes - A Guided Tour.* Princeton University Press.

Bequette, B.W. (1991). Non-linear Predictive Control of a CSTR Using Multi-rate Sampling Disturbance Region. *Can.J.Ch.E.*

Bierman, G.J. (1977). *Factorization Methods for Discrete Sequential Estimation.* Academic Press, New York.

Bitmead, R.R., M. Gevers, and V. Wertz (1990). *Adaptive Optimal Control. The Thinking man's GPC*. Prentice-Hall.

Box, G.E.P. and G.M. Jenkins (1990). *Time Series Analysis Forecasting and Control*, San Francisco: Holden-Day.

Brengel, D.D. and W.D. Seider (1989). Multistep Nonlinear Predictive Controller. *Ind. Eng. Chem. Res.*, **28**, 1812.

Bruijin, P.M., L.J. Bootsma and H.B. Verbruggen (1980). Predictive control using impulse response models. *IFAC Symp. on Digital Computer Applications to Process Control*, Dusseldorf, F.R.G.

Bruijin, P.M. and H.B. Verbruggen (1984). Model algorithmic control using impulse response models. *Journal A*, **25**, 69.

Camacho, E.F. and C. Bordons (1995). *Model Predictive Control in the Process Industry*. Springer, London.

Cameron, F. and D.E. Seborg (1983). A self-tuning controller with a PID structure. *Int. J. Control*, **38**, 401.

Cegrell, T and T. Hedqvist (1975). Successful adaptive control of paper machines. *Automatica*, **11**, 53.

Chow, C.M. and D.W. Clarke (1994). *Advances in Model-Based Predictive Control*. Chapter: Actuator Nonlinearities in Predictive Control. Oxford University Press.

Clarke, D.W. (1988). Application of Generalized Predictive Control to Industrial Processes, *IEEE Control Systems Magazine*, **8**, 49.

Clarke, D.W. (1994). Advances in Model-Based Predictive Control. In *Advances in*

Model-Based Predictive Control, D.W. Clarke, Ed., Oxford: Oxford Univ. Press.

Clarke, D.W. and P.J. Gawthrop (1975). A self-tuning controller. *IEE Proc.*, **122**, 929.

Clarke, D.W. and P.J. Gawthrop (1979). Self-tuning control. *IEE Proc.*, **126**, 633.

Clarke, D.W., P.P. Kanjilal and C. Mohtadi (1985). A generalized LQG approach to self-tuning control. *Int. J. Control*, **41**, 1509.

Clarke, D.W. and C. Mohtadi (1989). Properties of generalized predictive control. *Automatica*, **25**, 149.

Clarke, D.W., C. Mohtadi and P.S. Tuffs (1987). Generalized predictive control - Part I. The Basic Algorithm. *Automatica*, **23**, 137.

Clarke, D.W., C. Mohtadi and P.S. Tuffs (1987). Generalized predictive control - Part II. Extensions and Interpretations. *Automatica*, **23**, 149.

Clarke, D.W. and R. Scattolini (1991). Constrained Receding-horizon Predicitve Control. *Proceedings IEE*, **138** (4): 347.

Cook, P (1986). *Non-linear dynamic systems*, Prentice-Hall.

Crisalle, O.D., D.E. Seborg and D.A. Mellichamp (1989). Theoretical analysis of Long Range Predictive Controllers. *Proc. 1989 Am. Control Conference*, Pittsburg, PA, 570.

Cutler, C.R. and R.B. Hawkins (1987). Constrained Multivariable Control of a Hydrocracker Reactor. *Proceedings ACC*, Minneapolis, U.S.A.

Cutler, C.R. and B.L. Ramaker (1980). Dynamic Matrix Control. A computer Control Algorithm, *JACC*, San Francisco.

Dawkins, J. and P.A.N. Briggs (1965). A method for using weighting functions as system description in optimal control. *Proc. IFAC Symp.*, Teddington, U.K.

De Keyser, R.M.C. and A.R. Van Cauwenberghe (1981). Self-tuning predictive control. *Journal AIChE*, 23, 1.

De Keyser, R.M.C. and A.R. Van Cauwenberghe (1982a). Applications of self-tuning predictive control. *Journal AIChE*, 23, 1.

De Keyser, R.M.C. and A.R. Van Cauwenberghe (1982b). Typical application possibilities for self-tuning predictive control. *Proc. IFAC. Symp. on Identification and System Parameter Estimation*, Washington, DC.

De Keyser, R.M.C. and A.R. Van Cauwenberghe (1985). Extended prediction adaptive control. *Proc. IFAC. Symp. on Identification and System Parameter Estimation*, York, U.K.

De Keyser, R.M.C., P.G.A. Van de Velde, and F.G.A. Dumortier (1988). A Comparative Study of Self-adaptive Long-range Predictive Control Methods. *Automatica*, 24 (2): 149.

De Nicolao, G. And R. Scattolini (1994). *Adavances in Model Predictive Control.* Chapter: Stability and Output Terminal Constraints in Predictive Control. Oxford University Press.

Economou, C.G., M. Morari and B.O. Palsson (1986). Internal Model Control. Extension to Non-linear Systems. *Ind. Eng. Chem. Process Des. Dev.*, 25 403.

Favier, G., D. Dubois and C. Rougerie (1988). On Predictive Control: A comparison and some extensions. *Conference Proceedings Springer-Verlag Lecture Notes in Control and Information Sciences* #111.

Feldbaum, A.A. (1960-1961). Dual control theory I-IV. *Aut. & Remote Control*, 21, 874; 21, 1033; 22, 1; 22, 109.

Feldbaum, A.A. (1965). *Optimal Control Systems*. Academic Press, New York.

Fjeld, M. and R.G. Wilhelm, Jr (1981). Self-tuning regulators - the software way. *Control Engng*, Oct., 99.

Garcia, C.E. and M. Morari (1982). Internal Model Control. 1. A unifying review and some new results. *I & EC Process Des. Dev.*, 21, 308.

Garcia, C.E., D.M. Prett and M. Morari (1989). Model Predictive Control: Theory and Practice - A Survey. *Automatica*, 25 (3): 335.

Greco, C., Menga, G., Mosca, E. And G. Zappa (1984). Performance Improvement of Self-tuning Controllers by Multistep Horizons: the MUSMAR Approch. *Automatica*, 20: 681.

Gregory, P.C., Ed. (1959). Proc. *Self Adaptive Flight Control Symposium*. Wright-Patterson Air Force Base, Ohio: Wright Air Development Center.

Harris, C. and Billing, S. (Eds.) (1985). *Self-tuning Control: theory and applications*, Peter Peregrinus Ltd.

Joseph, B. (1989). *Real-Time Personal Computing. For Data Acquisition and Control.* Prentice Hall, Eaglewood Cliffs, New Jersey.

Jutan, A. and E.S. Rodriguez II (1987). Application of parametric control concepts to decoupler design and heating control design for a batch reactor. *Can. J. Ch.E.*, 65, 858.

Jutan, A. and A. Uppal (1984). Combined feedforward feedback servo control scheme for an exothermic batch reactor. *I.E.C. Proc. Des. Dev.*, 23, 597.

Katende, E., (1992) *A New Constrained Self-tuning PID controller*, M.E.Sc. Thesis,UWO: London,On., Can.

Katende, E. and A. Jutan, (1993). A New Constrained Self-tuning PID controller, *Can.J.Ch.E.*, 71, 625.

Katende, E and A. Jutan (1995). Non-linear Predictive Control, *ACC Proc.*, FP10, Seattle, USA.

Katende, E and A. Jutan (1996). Non-linear Predictive Control of Complex Processes, accepted for public.ition: In *Ind. Eng. Chem.*

Kouvaritakis, B., Rossiter, J.A. and A.O.T. Chang (1992). Stable Generalized Predictive Control: An Algorithm with Guaranteed Stability. *Proceedings IEE*, Part D, 139 (4): 349.

Kutnetsov, A.G. and D.W. Clarke (1994). *Advances in Model Predictive Control.* Chapter: Application of Constrained GPC for Improving Performance of Controlled Plants. Oxford University Press.

Kwakernaak, H and R. Sivan (1972). *Linear Optimal Control Systems.* Wiley.

Lam, R.P. (1980). *Implicit and Explicit Self-tuning Controllers*, D.Phil Thesis Univ. of Oxford.

Lachmann, K. H. (1982). Parameter-adaptive control of a class of non-linear processes, *6th IFAC symp. Ident. Syst. Param.* Est. Arlington, VA pp.372.

Lang, Y.D. and L.T. Biegler (1987). A Unified Algorithm for Flowsheet Optimization. *Comp. Chem. Eng.*,11 (2), 143.

Lee J.H., M.S. Gelormino, and M. Morari, (1992). Model Predictive Control of Multirate sampled-data systems: A State Space Approch, *Int. J. Contr.*, **55**, pp.153-191.

Lee, P.L. and G.R. Sullivan (1988). Generic model control (GMC). *Comput. Chem. Eng.*, **12**, 6, 573.

Lemos, J.M. and E. Mosca (1985). A Multipredictor-based LQ Self-tuning Controller. In *IFAC Symp. On Identification and System Parameter Estimation*, York, UK, 137.

Li, W.C. and L.T. Biegler (1988). Process Control Strategies for Constrained Nonlinear Systems. *Ind. Eng. Chem. Res.*, 27, 1421.

Ljung, L. (1987). *System Identifications: Theory for the User. Prentice-Hall*, Englewood Cliffs, NJ.

Ljung, L. and S. Gunnarson (1990). Adaptation and tracking in system identification - A survey. *Automatica*, **26**, 7.

MacGregor, J.F. (1976). Optimal choice of the sampling interval for discrete process control. *Technometrics* **18**, 151.

Martin, G.D., J.M. Caldwell and T.E. Ayral (1986). Predictive Control Applications for the Petroleum Refining Industry, *Japan Petroleum Institute - Petroleum Refining Conference*, Tokyo, Japan.

Matsko, T.N. (1985). Internal Model Control fur Chemical Recovery. *Chem. Eng. Prog.*, **81**, 12, 46.

McAvoy, T.J., Y. Arkun and E. Zafiriou, Ed., (1989). Model-Based Process Control -

Proceedings of the 1988 IFAC Workshop. Pergamon Press, Oxford.

McIntosh, A.R., S.L. Shah and D.G. Fisher (1991). Analysis and Tuning of Adaptive Generalized Predictive Control. *Can.J.Ch.E.*, **69**, 97

Mehra, R.K., R. Rouhani, J. Eterno, J. Richalet and A. Rault, (1982). Model Algorithmic Control: Review and Recent Development. *Chem. Process Control II Conf.*, Sea Island, GA. 287.

Mishkin, E. and L. Braun (1961). *Adaptive Control Systems*. New York: McGraw-Hill.

Mohtadi, C. (1989). On the role of prefiltering in parameter estimation and control. In Shah and Dumont, Eds., 121.

Morari. M. And E. Zafiriou (1989). *Robust Process Control*. Prentice-Hall.

Mosca, E., G. Zappa and C. Manfredi (1984). Multistep horizon self-tuning controllers: the MUSMAR approach, *IFAC 9th World Congress*. Budapest, Hungary.

Mosca, E. And J. Zhang (1992). Stable Recdesign of Predictive Control. *Automatica*, **28**, 1229-1233.

Murtagh, B.A. and M.A. Sauders (1983). MINOS 5.0 *User's Guide*; Systems Opt. Lab, Stanford University, Palo Alto, CA.

Noble, B. (1969). *Applied Linear Algebra*. Eaglewood Cliffs. Prentice Hall.

Novratil, J.P., K.Y. Lim and D.G. Fisher (1989). Disturbance Feedback in Model Predictive Control Systems. In McAvoy et al., Ed. *Proc. of the 1988 IFAC Workshop*, Pergamon Press, Oxford.

Peterka, V. (1984). Predictor based self-tuning control. *Automatica*, **20**, 39.

Prett, D.M. and C.E. Garcia (1988). *Fundamental Process Control*. Butterworth

Publishers.

Reid, J.G., D.E. Chaffin and J.T. Silverthorn (1981). Output predictive algorithmic control: precision tracking with application to terrain following. *AIAA J. Guidance Control*, 4, 502.

Reid, J.G., R.K. Mehra and E. Kirkwood, Jr. (1979). Robustness properties of output predictive dead-beat control: SISO case. *Proc. CDC*, Fort Lauderdale, Florida.

Richalet, J. S. Abu el Ata-Doss, C. Arber, H.B. Kuntze, A. Jacubash amd W. Schill (1987). Predictive Functional Control. Application to Fast and Accurate Robots. *In Proc. 10th IFAC Congress*, Munich.

Richalet, J., A. Rault, J.L. Testud and J.Papan (1976). Algorithmic Control of Industrial Processes. In *4th IFAC Symposium on Identification and System Parameter Estimation*. Tbilisi URSS.

Richalet, J., A. Rault, J.L. Testud and J.Papan (1978). Model predictive heuristic control: applications to industrial processes. *Automatica*, **14**, 413.

Ricker, N.L. (1989). Model-Predictive Control of Process with Many Inputs and Outputs. *Control and Dynamics Systems*, 36, T. Leondes (Ed.), Academic Press, San Diego.

Rossiter, J.A. and B. Kouvaritakis (1994). *Advances in Model-Based Control*. Chapter: Advances in Generalized and Constrained Predictive Control. Oxford University Press.

Ruiz, J., E. Muratore, A. Aryal and D. Durand (1986). Optimal Management of Pulp Mill Production Departments and Storage Tanks. *6th IFAC/IFIP/IMEKO Conference on Instrumentation and Automation in Paper, Rubber, Plastics and*

Polymerization Industries, Arkon, OH.

Scattolini R. And N. Schiavoni (1995). A Multi-rate Model-Based Predictive Controller, *IEEE Trans. Aut. Contr.*, **40**, pp. 1093

Scokaert, P.O.M., C.M. Chow and D.W. Clarke, (1995). *Simulink Predictive Contol Environment (SPACE)*, Report No. OUEL 2057/95, Univ. Oxford.

Seborg, D.E., S.L. Shah and T.F. Edgar (1986). Adaptive Control Strategies for Process Control: A Survey, *AIChE Journal*, **32**, 881-913.

Shah, S.L., C. Mohtadi and D.W. Clarke (1987). Multivariable adaptive control without prior knowledge of the delay matrix. *Systems Control Lett.*, **9**, 295.

Soeterboek, R. (1992). *Predictive control - A unified approach*, Prentice-Hall.

Svoronos, S., G. Stephanopoulos and R. Aris. (1981). On bilinear estimation and control, *Int. J. Control*, **34**, pp. 651.

Tsypkin, Y.Z. (1971). *Adaptation and Learning in Automatic Systems*. Academic Press. New York.

Vega,P., P. Prada and V. Aleixandre (1991). Self-tuning predictive PID controller (1991). *IEE Proc.*, D, **138**, 303.

Warwick, K. (Ed.) (1988). *Implementation of self-tuning controllers*. Peter Peregrinus Ltd.

Warwick, K. and D. Rees (Ed.) (1986). *Indusrial Digital Control Sysems*. Peter Peregrinus Ltd.

Wellstead, P.E., J.M. Edmunds, D. Prager and P. Zanker (1979). Self-tuning pole/zero assignment regulators. *Int. J. Control*, **30**, 1.

Wellstead, P.E., D. Prager and P. Zanker (1979). Pole assignment self-tuning regulator. *IEE Proc.*, **126**, 781.

Wittenmark, B. (1973). *A self-tuning regulator*. Thesis, TFRT-T1003, Department of Automatic Control, Lund Institute of Technology.

Wittenmark, B. (1989). Adaptive control: Implementation and Application Issues. In Shah and Dumont, Ed., 103.

Ydstie, B.E. (1984). Extended horizon adaptive control, *Proc. 9th IFAC World Conference*, Budapest, Hungary, 133.