Generalised Predictive Control
A study and application

Karl J. Prince

Thesis presented in fulfilment of the requirements for the degree of Master of Science in Electrical Engineering at the University of Cape Town.

Thesis supervisor
Associate Professor Martin Braae

April 1996

ABSTRACT

This dissertation investigates the Generalised Predictive Control (GPC) method derived by Clarke, Mohtadi and Tuffs in the mid eighties. GPC is an adaptive predictive control algorithm, of which there are number, and has proven to be one of the more popular variants having been applied to various control applications. The theory of the GPC method is studied in detail and a computer simulation program has been written for analyses. While the theory is perhaps not straightforward the actual use of GPC is quite easy. There are 'tuning knobs' available which provide a simple means of tuning the response to match specifications.

The GPC method is extended to an overall Long-Range Predictive Control (LRPC) method using the Long-Range Predictive Identification (LRPI) developed by Shook, Mohtadi and Shah in the early nineties. The new identification algorithm replaces the recursive least squares (RLS) used in the original GPC and is essentially a duplicate of the control law. The effect of using LRPI as opposed to RLS is investigated theoretically by use of the simulator. In the simulations carried out comparison of the GPC and LRPC responses depended on the GPC parameters employed. On the whole the LRPI scheme improved the responses though, especially when disturbances and model changes were investigated.

Both the GPC and LRPC methods are applied to a practical problem based on an industrial flotation plant. The flotation plant simulator is a multivariable process which consists of a four-input-four-output system. Because of the adaptive nature of the control methods it was possible to implement the control solution as a diagonal system. In this way a single-input-single-output (SISO) system was used on each of the input-output pairs and the interaction of the plant was not directly addressed. Both methods produced stable control of the flotation plant simulator. The use of GPC tuning parameters to tune responses and the flexibility thus derived is demonstrated on this practical application. When compared with a default set of parameters the LRPC method does respond differently to GPC and seems to provide an overall better response to this application.

The design of a robust control system is necessary for industrial applications. The dissertation investigates the use of the VMEbus industrial computer and the Windows NT operating system as the basis for a rugged industrial control system. While it is obvious that the hardware must be robust the thesis stresses the necessity for an operating system that is suited to control applications. With modern day trends this includes such features as robustness, multi-tasking, a real-time capability and a graphical user interface (GUI). The conclusion drawn is that the VMEbus computer is suited for robust control applications. The Windows NT operating system would be suited for only certain applications, since there are limits to its performance, but this is not considered a disadvantage since no single operating system would probably meet all the specifications entirely.

ACKNOWLEDGMENTS

Thanks to Professor Martin Braae for his supervision and guidance and whose assistance was invaluable. Thanks also to the guys in the control group,

Warren Carew
Cameron Haines
Jacek Narozny
Ignatious Thithi

Their suggestions and sometimes priceless information has duly assisted in the completion of this thesis.

Thanks to all my family and friends who supported me with their understanding and patience.

I gratefully acknowledge the financial support given to myself and this project by Mintek.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

GPC	Generalised Predictive Control
LRPC	Long-Range Predictive Control
LRPI	Long-Range Predictive Identification
CARMA	Controlled Auto-Regressive Moving Average
CARIMA	Controlled Auto-Regressive Integrated Moving Average
PI	Performance Index
ISE	Integral Square Errors
ITAE	Integral Time-Multiplied Absolute Error
GUI	Graphical User Interface
FAQ	Frequently Asked Questions

CHAPTER 1

INTRODUCTION

1.1. Introduction

Predictive control first appeared in the late seventies and has since then become a well known and applied control strategy. A number of predictive controllers have been proposed during this time span. These include Dynamic Matrix Control (DMC) [8], Generalised Predictive Control (GPC), Extended Horizon Adaptive Control (EHAC) [20], Model Algorithmic Control (MAC) [21] and Extended Prediction Self-Adaptive Control (EPSAC) [22].

Predictive controllers are model-based i.e. a model of the process is used to design the controller. Obtaining the model of the process is thus an important part of the overall control. This has been the focus of an innovative extension to Generalised Predictive Control (GPC) called Long-Range Predictive Identification (LRPI).

The aim of this thesis is then to study the Long-Range Predictive Identification algorithm, as an extension to GPC. Furthermore, the thesis investigates the application of a GPC/LRPI controller in simulation and to a practical problem.

1.2. Generalised Predictive Control

GPC was originally formulated in the late eighties by Clarke *et al.* (1987). Since then the method has become a popular one and has been applied successfully to a number of practical problems [1,2,7].

GPC falls into the category of adaptive predictive controllers. The adaptive control methodology is essentially the design (at times on-line) of a controller based on an on-line estimated plant model. Predictive control is an algorithm that defines a control criterion at the present time in terms of predictions of future plant outputs and also future plant inputs.

Chapter 2 takes a closer look at the GPC strategy of Clarke *et al.*

1.3. Long Range Predictive Control

The development of the GPC control method relied on the fact that the model of the process was appropriate. A recursive least squares (RLS) identification strategy was used for model parameter estimation and incorporated into the GPC controller.

Shook *et al.* have, however, suggested that the RLS model is not optimum for GPC and that an identification strategy more closely linked to the control strategy should be adopted. This identification procedure is known as LRPI and the overall control strategy is labeled Long-Range Predictive Control (LRPC).

Essentially LRPI extends the single-step-ahead prediction of the RLS estimator to a multi-step-ahead identification procedure. The prediction horizons of LRPI are linked to the horizons of GPC algorithm, thus more closely matching the modeling and control schemes.

Chapter 3 studies this identification strategy in conjunction with GPC.

1.4. An Application

To observe the effect of GPC and LRPI in practice, they have been applied to a laboratory flotation rig [3] which models an industrial flotation system.

The process is a multivariable system. As opposed to single variable (SISO) systems, which have a single input and a single output, a multivariable system has multi-inputs and multi-outputs. Besides having more inputs and outputs, the phenomenon of interaction occurs between inputs i.e. one input affects more than one output (see Fig. 1.1).

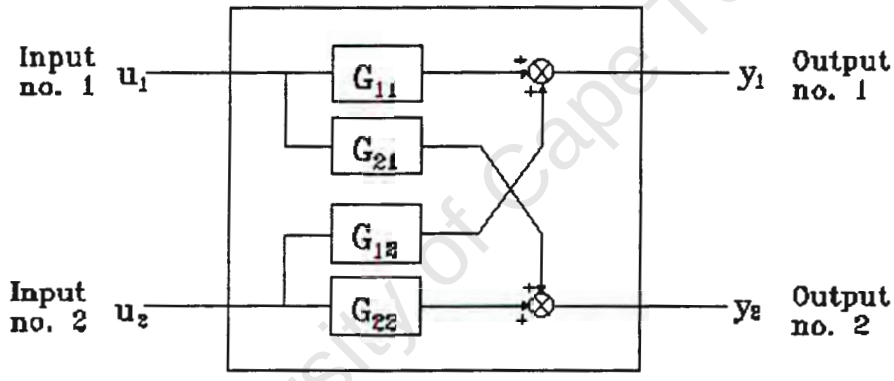

Figure 1.1 A multivariable process showing interaction

The flotation rig is a four input, four output system. It consists of a four tank arrangement each with an associated output valve used for level control.

The controller is implemented in a computer. An investigation into the use of an industrial computer has been done, highlighting the important fact that ordinary personal computers are not suited for industrial applications [4]. Further evaluations were made with regards to an appropriate operating system and programming software for robust control use.

Chapter 4 discusses the equipment used for the thesis, detailing the interface between the control software and the hardware. Chapter 5 shows the results of applying GPC to a problem of a practical nature.

CHAPTER 2

GENERALISED PREDICTIVE CONTROL

2.1. Introduction

In this chapter the theory of Generalised Predictive Control (GPC) is discussed and demonstrated.

Before studying the mathematical strategy of GPC, it is worthwhile analysing a brief outline of the GPC algorithm. A graphical summary is shown in Fig. 2.1.

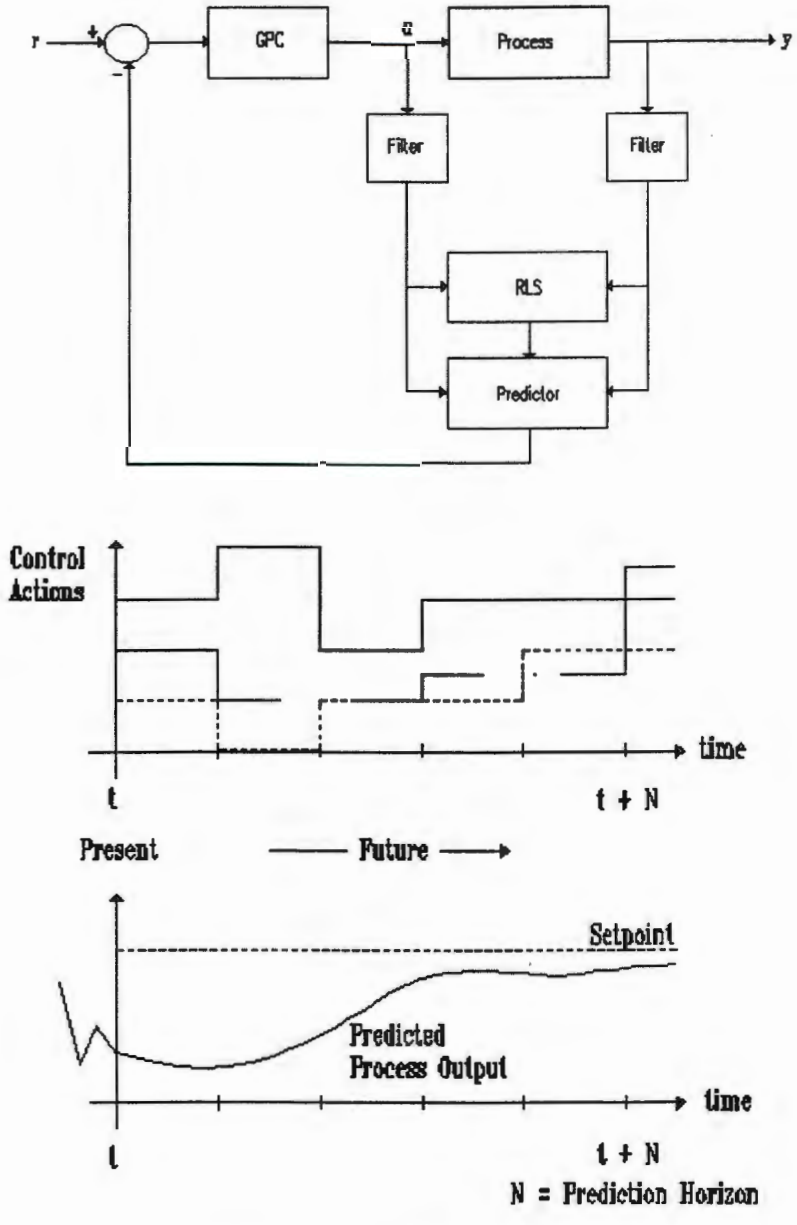

Figure 2.1 Outline of GPC control strategy

Studying the plots of the setpoint, output and control signals, the GPC algorithm is summarised as follows:

- Consider a particular time *t* as the present.
- For a given future setpoint, the process output, calculated from a prediction model (obtained from a recursive least squares algorithm), is calculated over a prediction horizon, *N*.
- A number of different sets of control actions are suggested for the prediction but only the best strategy (i.e. one which minimises some appropriate cost function of output errors and control actions) will be selected.
- The selected set of control actions is applied at time *t*.
- The procedure then moves on to the next time instant and is repeated.

So the GPC control strategy predicts the process outputs based on future sets of control signals and minimises some cost function in order to select the appropriate control action. From the outline above, we can see that GPC is not conceptually difficult and should make good practical sense.

2.2. The Plant Model for GPC

One of the attractions of the GPC control strategy is that its plant model closely matches industrial process where step disturbances and Brownian motion are significant sources of noise. In this section the plant model used in GPC is derived.

For control about a particular operating point, consider a locally-linearised model

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + x(t) \tag{2.1}$$

where

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \ldots + a_{na} q^{-na}$$
$$B(q^{-1}) = b_0 + b_1 q^{-1} + b_2 q^{-2} + \ldots + b_{nb} q^{-nb}$$

q^{-1} represents the backward-shift operator. The *A*-polynomial is monic. Variable *y(t)* is the output, *u(t)* is the control input and *x(t)* is the disturbance term.

The description of *x(t)* deserves a closer look as ideally it should describe the disturbances typically encountered in practice. Consider the single input/output relationship given by the linear difference equation of (2.1):

$$y(t) = \frac{B(q^{-1})}{A(q^{-1})}u(t-1) + \frac{1}{A(q^{-1})}\xi(t) \tag{2.2}$$

where $\xi(t)$ is a zero-mean uncorrelated random sequence (white-noise).

The white-noise term in the above description has been entered as a direct error in the difference equation. This is commonly referred to as the equation error model structure or the auto-regressive model.

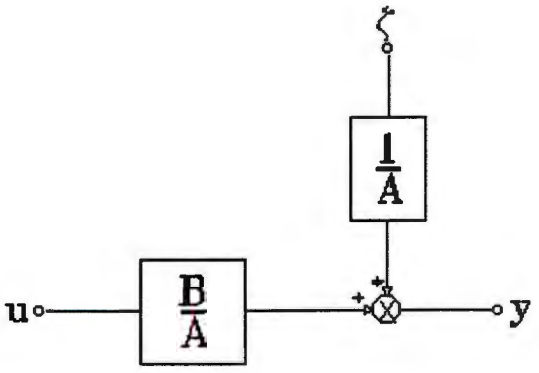

Figure 2.2 Equation error model structure

This model structure, as shown in Fig. 2.2, is not a natural description for a physical system as $\xi(t)$ goes through the denominator dynamics before being added to the system output.

Although the above model is a simple one it has no degree of freedom to sufficiently describe properties of the disturbance term. In order to provide the flexibility required the equation error can be described as a moving average of the white-noise

$$x(t) = C(q^{-1})\xi(t) \tag{2.3}$$

where

$$C(q^{-1}) = 1 + c_1 q^{-1} + c_2 q^{-2} + \ldots + c_{nc} q^{-nc}$$

and $C(q^{-1})$ is a monic polynomial.

With this disturbance term substituted in (2.1), we obtain the CARMA (Controlled Auto-Regressive and Moving- Average) model. While we have added flexibility to our model, the description for the disturbance term does not include a term for offset disturbances. Because of this, control laws deduced with this model are particularly sensitive to noise with a non-zero mean value. In order to realise an additive random walk description for the disturbance term an appropriate model would be

$$x(t) = \frac{C(q^{-1})}{\Delta}\xi(t) \tag{2.4}$$

where Δ is the differencing operator i.e.

$$\Delta = 1 - q^{-1}$$

Substituting (2.4) into (2.1) gives us the CARIMA (Controlled Auto-Regressive and Integrated Moving-Average) model.

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + \frac{C(q^{-1})}{\Delta}\xi(t) \tag{2.5}$$

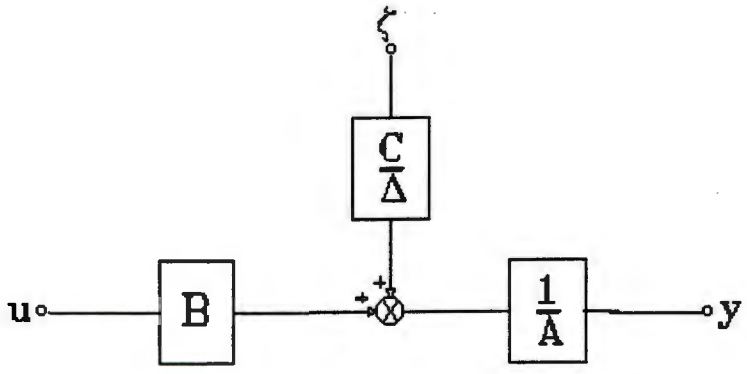

Figure 2.3 CARIMA model structure

This is the model which Clarke *et al.* [5] used to develop the GPC control strategy. GPC has this particular noise model embedded in its formulation so that it has a tailored response to particular circumstances. The effect of this model description will be that controllers derived from it will have the ability to reject step output disturbances and brownian motion type disturbances. While these certainly do not represent all disturbances they are typical and encountered often in practice.

2.3. The Prediction Equations

The Generalised Predictive Control algorithm is based on the prediction of the process output at j steps ahead, $y(t+j)$. The prediction equations generate the future process outputs by extending the known information using the CARIMA plant model, as depicted in Fig 2.3.

Since we are trying to find a j-step-ahead predictor based on information that we have available i.e. information up until time t, we shall split the disturbance term into two terms. One term will characterise disturbances up to and including time t, the other future disturbances.

$$\begin{aligned} q^j \frac{C}{A\Delta}\xi(t) &= q^j E_j(q^{-1})\xi(t) + \frac{F_j(q^{-1})}{A\Delta}\xi(t) \\ &= \underbrace{e(t+k)}_{\text{future disturbances}} + \underbrace{e'(t)}_{\text{past and present disturbances}} \end{aligned} \tag{2.6}$$

where E_j and F_j are polynomials in q^{-1} (see Appendix A)

$$E_j(q^{-1}) = e_0 + e_1 q^{-1} + \ldots + e_{j-1} q^{-j+1}$$

$$F_j(q^{-1}) = f_{j0} + f_{j1} q^{-1} + \ldots + f_{jna} q^{-na}$$

Rearranging (2.6) as the following Diophantine equation

$$C(q^{-1}) = E_j(q^{-1})A(q^{-1})\Delta + q^{-j}F_j(q^{-1}) \tag{2.7}$$

where E_j and F_j are polynomials in q^{-1} as before. The prediction interval is given by j.

The following derivation for the predicted process outputs has the C-polynomial set to unity for simplification purposes. Multiplying (2.5) by q^j in order to obtain an equation representing future process outputs (omitting the arguments (q^{-1}) for brevity), we obtain

$$y(t+j) = \frac{B}{A}u(t+j-1) + \frac{1}{A\Delta}\xi(t+j)$$
$$A\Delta y(t+j) = B\Delta u(t+j-1) + \xi(t+j) \tag{2.8}$$

Solving for $A\Delta$ from the Diophantine equation (2.7) and substituting into (2.8) yields

$$y(t+j) = E_j B\Delta u(t+j-1) + F_j y(t) + E_j \xi(t+j) \tag{2.9}$$

From (2.9) we can deduce that all the noise components are in the future and will therefore be excluded in the optimal predictor. The minimum variance prediction of $y(t+j)$ given data known at time t is then,

$$\hat{y}(t+j|t) = G_j\Delta u(t+j-1) + F_j y(t) \tag{2.10}$$

where

$$G_j = E_j B$$

Hence the future output signals are made up of known signal values at time t and also of future control inputs which have yet to be determined.

2.4. The GPC Control Law

Consider a future set-point (reference signal) represented by

$$r(t+j) \qquad j=1,2,...$$

The aim of the ensuing control law should then be to eliminate any deviation of $y(t+j)$ from $r(t+j)$ but at the same time to limit the control action required.

In order to decide on an objective function for GPC it is useful to investigate the work upon which GPC has been based. The self-tuning regulator of Astrom and Wittenmark [6] was formulated using Minimum Variance (MV) control. The controller is obtained by minimising the following function,

$$J(t) = \mathbf{E}\left\{(y(t+1) - r(t+1))^2\right\} \tag{2.11}$$

As can be seen, the function is that of trying to limit the output variance. The function is minimised at time t for a particular $u(t)$. At time $t+1$ a new minimisation is required for $u(t+1)$. It is well known that this method performs well only for plants with stable zeros i.e. minimum phase systems. Non-minimum phase systems have control laws which require excessive control input in order to bring about optimal output variance.

It does this by the cancellation of plant zeros (stable and unstable) which leads to internal instability. In order to eliminate this problem the next step is to modify the cost function (2.11) so that it includes a penalty not only on the output but also on the control signal. The new cost function is then,

$$J(t) = \mathbf{E}\left\{(y(t+1) - r(t+1))^2 + \lambda u(t)^2\right\} \tag{2.12}$$

This has been termed as General Minimum Variance control (GMV) [7] which implements a one-step-ahead optimal control law. Although GMV is capable of an internally stable control law, this is not true for all λ. A variation of the GMV control law requires replacing $u(t)$ with $\Delta u(t)$.

$$J(t) = \mathbf{E}\left\{(y(t+1) - r(t+1))^2 + \lambda(\Delta u(t))^2\right\} \tag{2.13}$$

where $\Delta u(t)$ is the incremental control input of the system.

Δu(t) is used to eliminate the problem inherent in the GMV cost function of (2.12). The original GMV cost function does not admit[1] zero static error in the case of a non-zero constant reference. This would have allowed the output, y(t), to remain at a non-zero constant value with the control input being zero.

A controller developed with the modified GMV cost function still, however, fails for some unstable and non-minimum phase plants. A further variation has been made by Clarke, giving the generalised predictive control law which minimises:

$$J(N_1, N_2) = \mathbf{E}\left\{\sum_{j=N_1}^{N_2}\left(\hat{y}(t+j|t) - r(t+j)\right)^2 + \sum_{j=1}^{N_2}\lambda(j)(\Delta u(t+j-1))^2\right\} \tag{2.14}$$

A window for the cost function is defined by the minimum and maximum horizons $N_1 \rightarrow N_2$. $\lambda(j)$ is a control weighting sequence. $\hat{y}(t+j|t)$ represents the predicted process output at time $t+j$ given information up to time t.

The minimisation of this function produces $\Delta u(t), \Delta u(t+1), \ldots, \Delta u(t+N_2-1)$, but only $\Delta u(t)$ is implemented. At the next time interval, $t+1$, a new minimisation is performed, thus shifting the control horizons and finding a new solution to the optimal control problem. This known as Receding Horizon Control and the resulting control law belongs to the class known as Open-Loop-Feedback-Optimal control [5]. The approach can be summarised as follows :

- the future set-point sequence $r(t+j)$ is calculated
- predicted process outputs $y(t+j|t)$ are determined
- the quadratic function (2.14) is minimised finding an appropriate control sequence $u(t+j)$
- the first control signal $u(t)$ is implemented
- repeat minimisation procedure at next sample interval

[1] except when the open loop plant contains an integrator

2.5. The GPC Solution

As can be seen from (2.9), *y(t+j)* consists of three components:

(1) a term depending on future control actions still to be determined
(2) a term depending on past control actions and measured variables known at time t
(3) a term depending on future noise signals.

For the following discussion we will set some of the parameters to default values to simplify the derivation: $N_1 = 1$ and $\lambda(j)$ is assumed a constant, λ.

As mentioned in (2.10) we can ignore the future noise signals giving,

$$\begin{aligned}
\hat{y}(t+1) &= G_1\Delta u(t) + F_1 y(t) \\
\hat{y}(t+2) &= G_2\Delta u(t+1) + F_2 y(t) \\
&\vdots \\
\hat{y}(t+N_2) &= G_{N_2}\Delta u(t+N_2-1) + F_{N_2} y(t)
\end{aligned}$$

It would be convenient if *y(t+j)* could then be divided into two parts; one, *f(t+j)*, known at time *t* and the other made up of future signals. This can be achieved by,

$$\begin{aligned}
f(t+1) &= \left[G_1(q^{-1}) - g_{10}\right]\Delta u(t) + F_1 y(t) \\
f(t+2) &= q\left[G_1(q^{-1}) - g_{21}q^{-1} - g_{20}\right]\Delta u(t) + F_2 y(t) \\
&\vdots
\end{aligned}$$

where $G_i(q^{-1}) = g_{i0} + g_{i1}q^{-1} + \ldots$

Define a vector **f** composed of the signals known at time *t*,

$$\mathbf{f} = \left[f(t+1), f(t+2), \ldots, f(t+N_2)\right]^T$$

Define a vector of future control increments and a vector of predicted control outputs,

$$\begin{aligned}
\tilde{\mathbf{u}} &= \left[\Delta u(t), \Delta u(t+1), \ldots, \Delta u(t+N_2-1)\right]^T \\
\hat{\mathbf{y}} &= \left[\hat{y}(t+1), \hat{y}(t+2), \ldots, \hat{y}(t+N_2)\right]^T
\end{aligned}$$

Writing the equations at the top of the page in key vector form:

$$\mathbf{y} = \mathbf{G}\tilde{\mathbf{u}} + \mathbf{f} \qquad (2.15)$$

where the matrix **G** is composed of the parameters of the step-response,

$$\mathbf{G}=\begin{bmatrix} g_{10} & 0 & 0 & \cdots & 0 \\ g_{21} & g_{20} & 0 & \cdots & 0 \\ g_{32} & g_{31} & g_{30} & \cdots & 0 \\ \vdots & & & & \vdots \\ g_{N_2(N_2-1)} & g_{N_2(N_2-2)} & & \cdots & g_{N_20} \end{bmatrix}=\begin{bmatrix} g_0 & 0 & 0 & \cdots & 0 \\ g_1 & g_0 & 0 & \cdots & 0 \\ g_2 & g_1 & g_0 & \cdots & 0 \\ \vdots & & & & \vdots \\ g_{N_2-1} & g_{N_2-2} & & \cdots & g_0 \end{bmatrix} \tag{2.16}$$

From (2.16) it can be seen that $\mathbf{G}$ is an $N_2 \times N_2$ lower-triangular matrix.

The expectation of the cost-function in (2.14) is now written as:

$$J = E\left\{(\mathbf{y}-\mathbf{r})^T(\mathbf{y}-\mathbf{r})+\lambda\tilde{\mathbf{u}}^T\tilde{\mathbf{u}}\right\} \tag{2.17}$$

where $\mathbf{r}=\left[r(t+1),r(t+2),\ldots,r(t+N_2)\right]^T$

The minimisation of J [5] results in the future incremental control vector:

$$\tilde{\mathbf{u}}=\left(\mathbf{G}^T\mathbf{G}+\lambda\mathbf{I}\right)^{-1}\mathbf{G}^T(\mathbf{r}-\mathbf{f}) \tag{2.18}$$

The projected control increments in (2.18) span from t to $t+N_2-1$ yielding an open loop strategy based upon information up until time t. As aforementioned, only the first control increment i.e. $\Delta u(t)$ is implemented giving the current control signal,

$$u(t)=u(t-1)+\mathbf{g}^{-T}(\mathbf{r}-\mathbf{f}) \tag{2.19}$$

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

2.6. The GPC Tuning Parameters

We now take a look at the GPC tuning parameters and their suggested values which have been thoroughly studied and are well understood [5,27]. Recall the GPC cost function:

$$J(N_1,N_2)=E\left\{\sum_{j=N_1}^{N_2}\left(\hat{y}(t+j|t)-r(t+j)\right)^2+\sum_{j=1}^{N_2}\lambda(j)\left(\Delta u(t+j-1)\right)^2\right\}$$

This is the introduction to three of the GPC tuning parameters (refer also to Fig. 2.4).

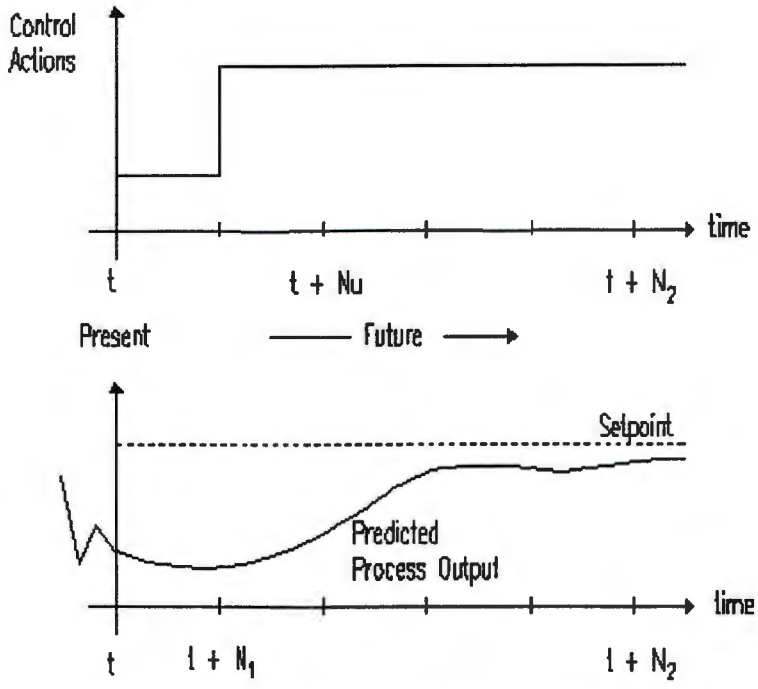

Figure 2.4 The GPC parameters

N_1: *the minimum costing horizon.* This parameter relates to the dead-time of the process controlled. If the dead-time, k, is known then N_1 is usually set equal to k. Setting N_1 less than k leads to extra calculations which have no effect in computing the control signal i.e. this means that the initial k-1 rows of the **G**-matrix are set to zero and have no effect in the computations. If, however, the dead-time is unknown or variable, then N_1 can be set to 1 and still provide a stable solution.

N_2: *the maximum costing horizon.* In practice it has been found that N_2 should be made large.

$\lambda(j)$: *the control weighting sequence.* This signal provides damping for the control signal and can generally be kept constant i.e. $\lambda(j) = \lambda$. From (2.13) and (2.14) the connection between the GPC and minimum variance control can be seen. In MV λ is kept small to ensure that the cost function maintains minimal output variance, only being used to prevent harsh control action. Small λ is thus suggested to provide further damping of control action.

An important tuning parameter not mentioned thus far is that of the *control horizon.* Clarke *et al.* have taken this strategy from the method of Dynamic Matrix Control [8]. The principle is that after a particular time $Nu < N_2$ the future control increments are set to zero. This means that

$$\Delta u(t+j-1) = 0 \qquad j > Nu$$

where Nu is the control horizon. The reason for this is that if an incorrect dead-time is assumed $\mathbf{G}^T\mathbf{G}$ is singular and therefore a particular value for λ would be required in order to find a stable control law. The value for λ would of course not be known *a priori* and would limit the effectiveness of the adaptive approach.

This means that future control actions after the control horizon have been completely damped. A further implication of the control horizon can be seen by taking a look at the now modified G-matrix:

$$\mathbf{G}_{Nu} = \begin{bmatrix} g_0 & 0 & 0 & \cdots & 0 \\ g_1 & g_0 & 0 & \cdots & 0 \\ g_2 & g_1 & g_0 & \cdots & 0 \\ \vdots & & & & \vdots \\ g_{N_2-1} & g_{N_2-2} & & \cdots & g_{N-Nu} \end{bmatrix} \tag{2.20}$$

The dimension of $\mathbf{G}_{Nu}$ is now $N \times Nu$. An added advantage is that the inversion required in (2.18) is now less computationally intensive.

Clarke *et al.* suggest that for typical industrial plant models $Nu = 1$ is adequate. Higher values for Nu are required for more complex systems.

While the choices for the parameters seem varied, there are two controller setups that have been supported i.e. mean-level and state-dead-beat control. The settings for these are shown in Table 2.1.

Table 2.1 GPC parameter settings

	Mean-Level	State-Dead-Beat
N_1	small	k
N_2	large	$\geq 2k - 1$
Nu	1	k

where k is the order of the plant model plus 1.

2.7. A Simple GPC Example

In order to make clearer the procedure for a GPC, a simple example is carried out, following the method. Consider a first order plant of the following general formulation,

$$y(t) = \frac{b_0 + b_1 q^{-1}}{1 + a_1 q^{-1}} u(t-1)$$

For this example the GPC parameters are selected to be,

$N_1 = 1$ min. *prediction horizon*
$N_2 = 3$ max. *prediction horizon*
$Nu = 3$ min. *control horizon*
$\lambda = 0.1$ *damping factor*

From recurrence equations (2.7):

$$e_j = f_{j0}$$
$$E_{j+1} = E_j + q^{-j} e_j \qquad F_{j+1} = F_j - A\Delta e_j$$

Initialising:

$$E_1 = 1 \qquad F_1 = q^{-1}(1 - A\Delta) = (1 - a_1) + a_1 q^{-1}$$
$$e_0 = 1 \qquad f_{10} = (1 - a_1)$$
$$f_{11} = a_1$$

Continuing,

$$E_2 = E_1 + q^{-1} e_1 = 1 + q^{-1} e_1 \qquad F_2 = F_1 - A\Delta e_1 = \left[1 - a_1(1 - a_1)\right] + a_1(1 - a_1)q^{-1}$$
$$e_1 = f_{10} \qquad f_{20} = 1 - a_1(1 - a_1)$$
$$f_{21} = a_1(1 - a_1)$$

$$E_3 = E_2 + q^{-2} e_2 = 1 + q^{-1} e_1 + q^{-2} e_2 \qquad F_3 = F_2 - A\Delta e_2 = \left[1 - a_1(1 - a_1(1 - a_1))\right] + a_1(1 - a_1(1 - a_1))q^{-1}$$
$$e_2 = f_{20} \qquad f_{30} = 1 - a_1(1 - a_1(1 - a_1))$$
$$f_{31} = a_1(1 - a_1(1 - a_1))$$

For the rest of the example we shall choose a specific plant to demonstrate the procedure (for an on-line GPC strategy the plant is identified via a recursive least squares algorithm - RLS).

Let $A(q^{-1}) = 1 - 0.5q^{-1}$
$B(q^{-1}) = 1 + 0.9q^{-1}$

From the GPC identity: $1 = E_j A\Delta + q^{-j} F_j$
Note that the C polynomial is set to 1 for this example.

$$j=1 \qquad 1=(1-1.5q^{-1}-0.5q^{-2})+q^{-1}(1.5-0.5q^{-1})$$
$$j=2 \qquad 1=(1+1.5q^{-1})(1-1.5q^{-1}-0.5q^{-2})+q^{-2}(1.75-0.75q^{-1})$$
$$j=3 \qquad 1=(1+1.5q^{-1}+1.75q^{-2})(1-1.5q^{-1}-0.5q^{-2})+q^{-2}(1.875-0.875q^{-1})$$

$$G_1 = E_1 B = 1+0.9q^{-1}$$
$$G_2 = E_2 B = (1+1.5q^{-1})(1+0.9q^{-1}) = 1+2.4q^{-1}+1.35q^{-2}$$
$$G_3 = E_3 B = (1+1.5q^{-1}+1.75q^{-2})(1+0.9q^{-1}) = 1+2.4q^{-1}+3.1q^{-2}+1.575q^{-3}$$

Now we find the predicted process outputs, splitting them into information known and unknown at time t,

$$\mathbf{y} = \mathbf{G}\tilde{\mathbf{u}} + \mathbf{f}$$

$$\begin{bmatrix} \hat{y}(t+1|t) \\ \hat{y}(t+2|t) \\ \hat{y}(t+3|t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2.4 & 1 & 0 \\ 3.1 & 2.4 & 1 \end{bmatrix} \begin{bmatrix} \Delta u(t) \\ \Delta u(t+1) \\ \Delta u(t+2) \end{bmatrix} + \begin{bmatrix} 1.5y(t)-0.5y(t-1)+0.9\Delta u(t-1) \\ 1.75y(t)-0.75y(t-1)+1.35\Delta u(t-1) \\ 1.875y(t)-0.875y(t-1)+1.575\Delta u(t-1) \end{bmatrix}$$

In order to find the control law we now have to find,

$$(\mathbf{G}^T\mathbf{G}+\lambda\mathbf{I})^{-1}\mathbf{G}^T = \begin{bmatrix} 0.5250 & 0.2653 & -0.0691 \\ -0.9947 & 0.1133 & 0.2653 \\ 0.6908 & -0.9947 & 0.5250 \end{bmatrix}$$

Using the first row of the above matrix, a setpoint and the previously defined vector $\mathbf{f}$ we can obtain the GPC control law,

$$\Delta u(t) = \left[\textit{first row of } (\mathbf{G}^T\mathbf{G}+\lambda\mathbf{I})^{-1}\mathbf{G}^T \right] [\mathbf{r}-\mathbf{f}]$$

$$\Delta u(t) = \begin{bmatrix} 0.5250 & 0.2653 & -0.0691 \end{bmatrix} \begin{bmatrix} r(t+1)-1.5y(t)+0.5y(t-1)-0.9\Delta u(t-1) \\ r(t+2)-1.75y(t)+0.75y(t-1)-1.35\Delta u(t-1) \\ r(t+3)-1.875y(t)+0.875y(t-1)-1.575\Delta u(t-1) \end{bmatrix}$$

$$\Delta u(t) = 0.525r(t+1)+0.2653r(t+2)-0.0691r(t+3)$$
$$-1.1222y(t)+0.401y(t-1)-0.7218\Delta u(t-1)$$

The control signal then implemented at time t is:

$$u(t) = 0.525r(t+1) + 0.2653r(t+2) - 0.0691r(t+3)$$
$$-1.1222y(t) + 0.401y(t-1) - 0.2782u(t-1) - 0.7218u(t-2)$$

This example demonstrates the method used for formulating the simulator used in section 2.9. All that is required is essentially the iteration of the example covered to obtain the relevant control increments at each sample time.

2.8. Extending the GPC Method

In a real process, noise is present and, for an accurate model of the system, cannot be ignored. Consider the noise term in the CARIMA model,

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

Throughout the literature [5,9,11,13] it has been stated that estimation of $C(q^{-1})$ in practical applications, is not a very successful procedure. With this difficulty it is instructive to use a particular polynomial, $T(q^{-1})$, in order to find a plant model closely related to the ideal. In this sense the T-polynomial is regarded as a fixed observer [5].

This allows one to represent the disturbance term and rewrite the Diophantine equation of (2.7) as,

$$x(t) = \frac{T(q^{-1})}{\Delta}\xi(t)$$
$$T(q^{-1}) = E_j(q^{-1})A(q^{-1})\Delta + q^{-j}F_j(q^{-1}) \qquad (2.21)$$

The implication for the GPC strategy can be noted by once again deriving a result for the predicted process outputs. The process output is represented by

$$Ay(t) = Bu(t-1) + \frac{T}{\Delta}\xi(t)$$

Multiplication by $q^j E_j \Delta$

$$E_j A \Delta y(t+j) = E_j B \Delta u(t+j-1) + E_j T \xi(t+j)$$

From (2.21)

$$(T - q^j F_j)y(t+j) = G_j \Delta u(t+j-1) + E_j T \xi(t+j)$$

The future process outputs are now given by

$$\hat{y}(t+j) = G_j \Delta u^f(t+j-1) + F_j y^f(t)$$

where $u^f(t) = \frac{1}{T}u(t)$ and $y^f(t) = \frac{1}{T}y(t)$ represent filtered versions of the original signals.

As before, this representation of the predicted outputs depends on signals known at time t and future control actions yet to be determined. It is, therefore, advantageous to split the control signals into past and future components. In trying to find a way to do this, it can be noted that the cost function and constraints used to determine the control law are all based on $\Delta u(t+j)$. So the term representing future control signals should include Δu rather than Δu^f.

A solution is provided by the following Diophantine equation

$$G_j(q^{-1}) = G_j'(q^{-1})T(q^{-1}) + q^{-j}H_j(q^{-1})$$

The future process outputs now are,

$$\hat{y}(t+j) = \underbrace{G_j'\Delta u(t+j-1)}_{\text{future control signals}} + \underbrace{H_j u^f(t-1) + F_j y^f(t)}_{\text{signals known at time } t}$$

To obtain the control signal the minimization procedure used before is once again applied.

While no detailed study of observer polynomial was done, it is important to state it's impact on the GPC design; it has been shown by Robinson and Clarke [10] to be the provision of robustness. The T-polynomial is usually implemented as a low-pass filter to lessen the problem of high frequency noise created by Δ which has high-pass characteristics. It is also used to attenuate the effects of unmodeled dynamics present in high frequencies. Ideally $T(q^{-1}) = C(q^{-1})$ but on-line identification or estimation of the noise polynomial is virtually impossible. Suggestions have been made [1,9,10] as to the implementation and structure of the T-polynomial.

2.9. Simulations

Effect of parameter variations

Simulations were executed in order to show the effectiveness of GPC and typical results are presented here. Firstly to demonstrate the use of the GPC parameters, simulations with parameter variations were done.

The plant considered for the next few simulations is,

$$\frac{2}{s(s+2)} \Rightarrow \frac{0.864q^{-1}}{1-1.135q^{-1}+0.135q^{-2}}$$

This is a second order plant with a pole at $s=-2$ and $s=0$. Note that the sample time is set to the industrial standard of 1 [s]. The parameter variations for each simulation are shown in Table 2.2. The resulting simulations are shown in Fig.'s 2.5 - 2.8.

Table 2.2 GPC parameter settings for simulations

	Simulation 1	Simulation 2	Simulation 3	Simulation 4
N_1	1	1	1	1
N_2	10	5	10	10
Nu	1	1	3	3
λ	0.1	0.1	0.1	0.9

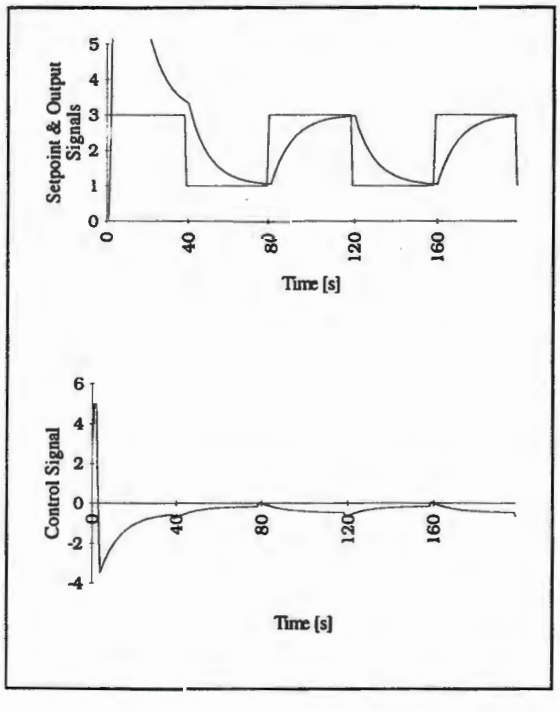

Figure 2.5 Simulation 1 :
$N_1=1, N_2=10, Nu=1, \lambda=0.1$

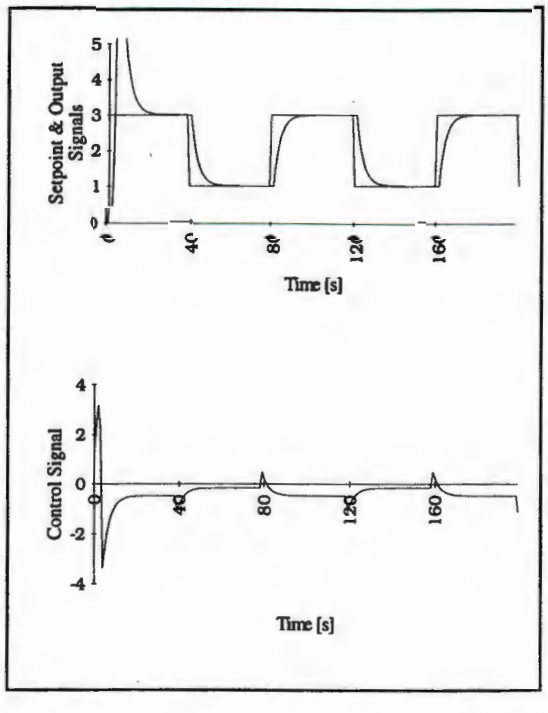

Figure 2.6 Simulation 2 :
$N_1=1, N_2=5, Nu=1, \lambda=0.1$

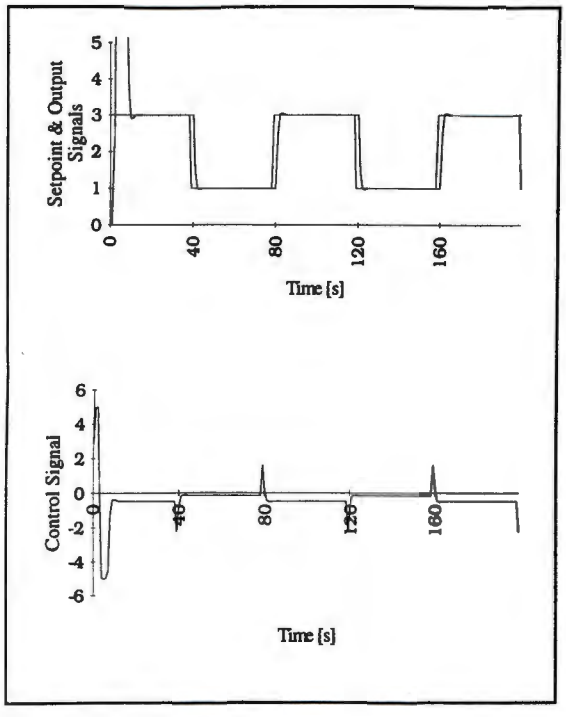

Figure 2.7 Simulation 3 :
$N_1 = 1, N_2 = 10, Nu = 3, \lambda = 0.1$

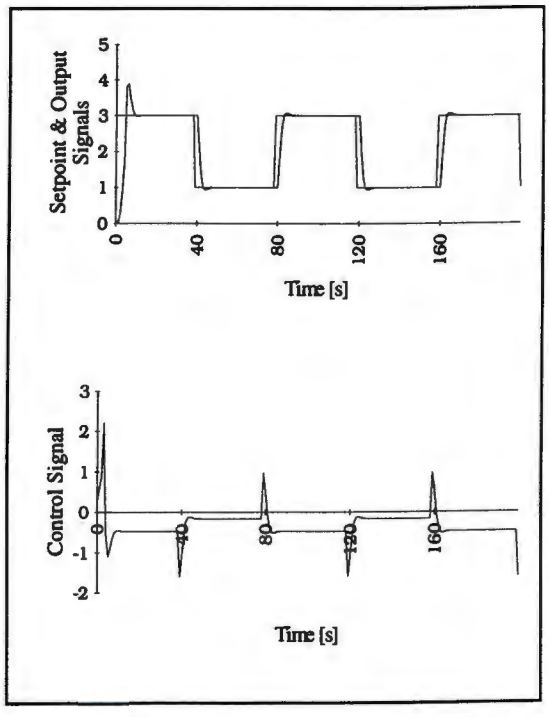

Figure 2.8 Simulation 4 :
$N_1 = 1, N_2 = 10, Nu = 3, \lambda = 0.9$

Comparing simulations 1 and 2 shows that an increase in the prediction horizon, N_2, results in a slower response. The faster response of simulation does need a larger control signal.

In simulation 3 the control horizon, *Nu*, has been increased and has produced a harsher control signal. There is thus a faster response time (considerably faster than simulation 1), but an overshoot in the step response has been induced.

Simulation 4 shows the use of λ in trying to damp the control signal. The initial control effort required is considerably less than that of simulation 3. The damping of the control signal has seen a decrease in response time and an increase in overshoot.

While the initial control signal is excessively large and the corresponding output has a large output error this is not a major factor. This response can be attributed to the initial estimate of the plant model. The initial estimate used in the simulations had the model set to zero. Thus by starting with no plant model GPC was still capable of adapting. In practice, one usually has an idea of what the plant model should look like, but as has been shown, this need not be a major factor.

Thus simulations 1 - 4 demonstrate the effectiveness and ease of use of the GPC tuning parameters. Clearly a larger control horizon speeds up the system response with an increase in prediction horizon slows the system down. Fine tuning of the control signal is provided by the damping factor.

Disturbance rejection

To demonstrate the effect of load and input disturbances, the simulations 5 and 6 were carried out. Table 2.3 shows the GPC parameters used in each of the simulations.

Table 2.3 GPC parameter settings for simulations

	Simulation 5	Simulation 6
N_1	1	1
N_2	5	5
Nu	1	2
λ	0.1	0.1

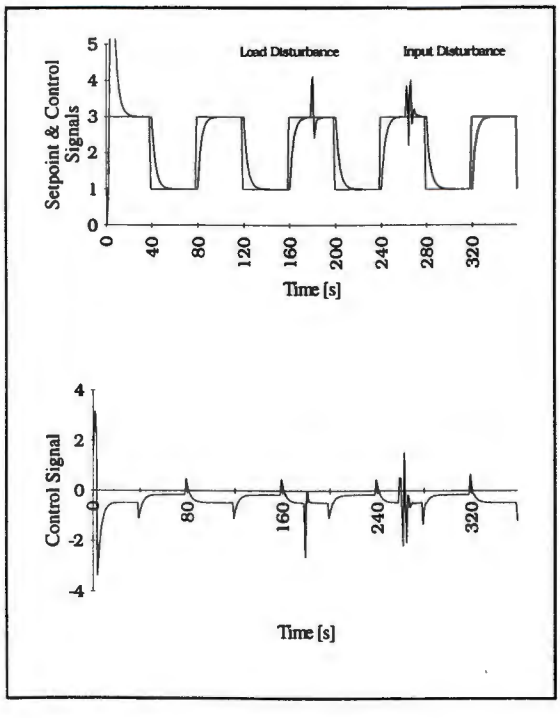

Figure 2.9 Simulation 5: showing load and input disturbance rejection
$N_1 = 1, N_2 = 5, Nu = 1, \lambda = 0.1$

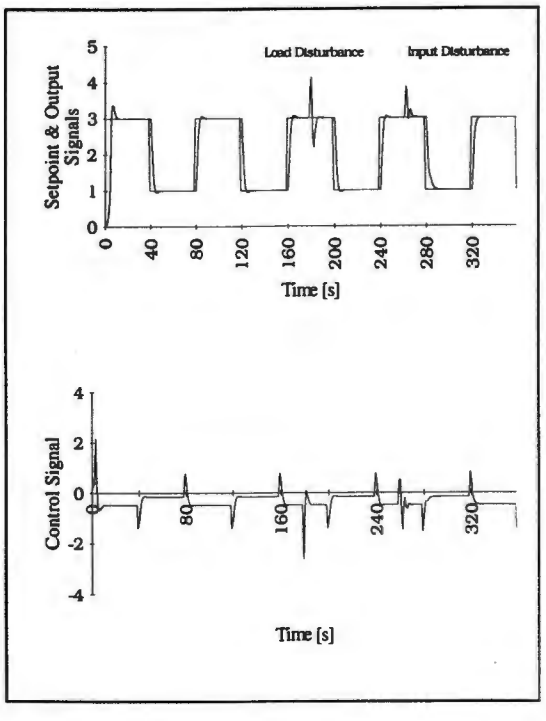

Figure 2.10 Simulation 6: showing load and input disturbance rejection
$N_1 = 1, N_2 = 5, Nu = 2, \lambda = 0.1$

Both GPC setups in simulations 5 and 6 have shown the ability to reject input and load disturbances. The system in simulation 6 shows faster disturbance rejection times than that of simulation 5 owing to the more harsh[2] control signal. Once again it can be seen that an increase in control horizon has produced a more active control signal.

Effect of model change

In order to demonstrate the adaptive ability of the GPC scheme, a simulation was carried out in which the model of the actual process was varied at intervals. Fig. 2.11 shows the result of this simulation. Each model change is indicated on the graph while the particular models are given in Table 2.4.

[2]Harsh refers to the high frequency content of the signal

This last simulation uses large system changes, although unlikely to be encountered practically, to demonstrate the versatility of the GPC scheme. While the basic algorithm implemented has not taken into account the robustness considerations of Robinson [5], it performs fairly robustly, and has not shown instability in the simulation. Note too that no more than two setpoint steps were required before GPC adapted to a new model.

Table 2.4 Order of models for simulation shown in Fig. 2.11

Model 1	$\frac{1}{s+0.1}$
Model 2	$\frac{5}{s(s+5)}$
Model 3	$\frac{1}{5s+1}$
Model 4	$\frac{s+2}{s^2+4s+8}$
Model 5	$\frac{e^{-2s}}{5s+1}$

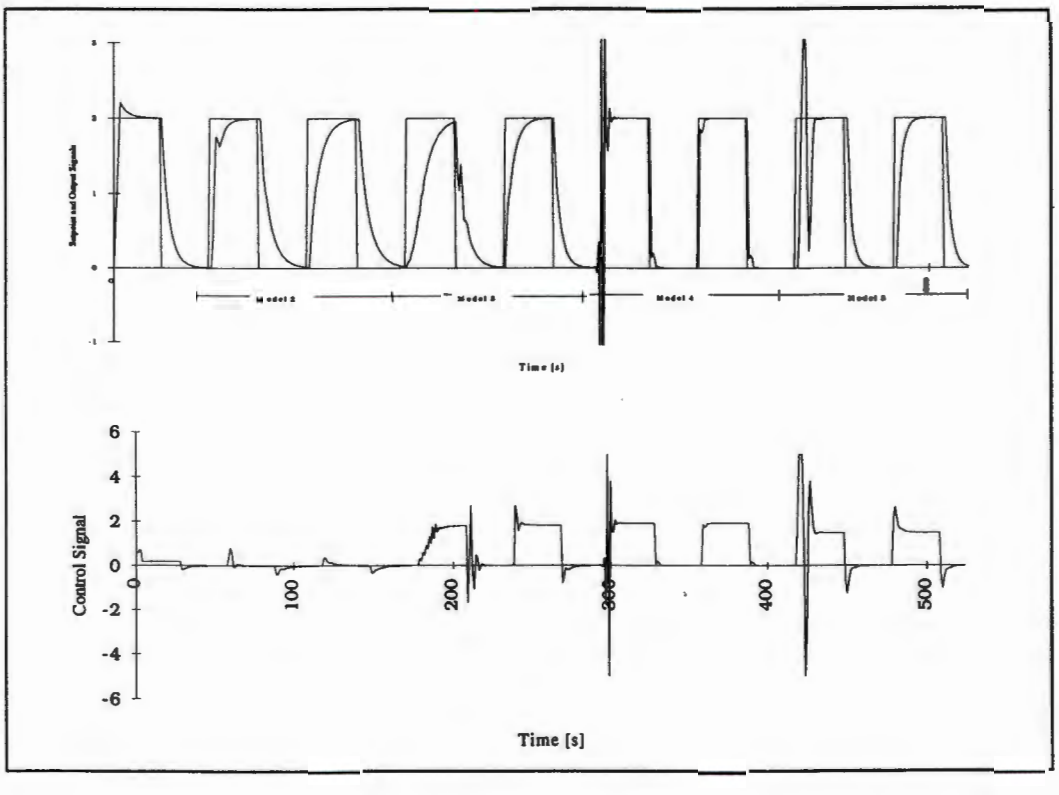

Figure 2.11 Simulation showing GPC adapting to various models

2.10. Summary

The GPC theory developed by Clarke *et al.* has been outlined and detailed in this chapter. The features of this predictive control method is the plant model adopted, i.e. the CARIMA model, and the tuning parameters available to the user. The tuning parameters have suggested guidelines well documented in the literature[1,5].

Simulations were executed in order to demonstrate the GPC algorithm. Specific tests were carried out to test the effect of parameter variation on the control scheme and these highlighted the ease with which the control method may be tuned.

CHAPTER 3

LONG-RANGE PREDICTIVE CONTROL

3.1. Introduction

In chapter 2 the GPC control algorithm was discussed at length . Emphasis was placed on the design of the control law but hardly any mention was made of the identification strategy. The Recursive Least Squares implemented was simply accepted as the identification strategy for the adaptive control and no further attention was given to it. Quoting Shook *et al* [11].

> "Generalised predictive control is universally accepted to be a more powerful approach than the GMV[1] approach. The 'Achilles heel' of its self-tuning version, however, is the *identification algorithm*, which was introduced more as an afterthought than as an integral part of the design."

This chapter will examine the identification of the plant, extending the identification to a dual of the GPC control law.

3.2. Identification Strategy for GPC

As mentioned in chapter 2, the plant on which GPC is based is modeled according to the following input-output formulation,

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

where $\Delta = 1 - q^{-1}$

Recall that the GPC control law minimises a cost function that is based on future predictions of the output assuming relevant assumptions have been made of future control actions,

$$J(N_1, N_2) = E\left\{\sum_{j=N_1}^{N_2}\left(y(t+j|t) - r(t+j)\right)^2 + \sum_{j=1}^{N_2}\lambda(j)\left(\Delta u(t+j-1)\right)^2\right\} \tag{3.2}$$

For the purposes of this discussion only the first term of the cost function is studied,

$$J_{GPC} = E\left\{\sum_{j=N_1}^{N_2}\left(\hat{y}(t+j|t) - r(t+j)\right)^2\right\} \tag{3.3}$$

[1] Generalised Minimum Variance

Finding the control signal is, of course, only part of the control strategy. The process output predictions, $\hat{y}(t+j|t)$, in (3.3) must be obtained from a model of the plant which has to be found by some process identification scheme. In Clarke *et al.* [3] a recursive least squares (RLS) estimator is the typical identification method used. It has been shown [11] that using RLS as is, does not provide the best identification strategy for GPC. The method of LS attempts to minimise the cost function,

$$J_{LS} = E\left\{\sum_{t=1}^{N}\left(y(t+1)-\hat{y}(t+1|t)\right)^2\right\} \tag{3.4}$$

From (3.4) it is easy to see that LS minimises a cost function of only one-step-ahead predictions. This varies significantly from the j-step-ahead predictions of the GPC control law.

The overall criterion for obtaining a GPC control law is given by a cost function which eliminates setpoint error:

$$J_{Overall} = E\left\{\sum_{j=N_1}^{N_2}\left(r(t+j)-y(t+j)\right)^2\right\} \tag{3.5}$$

It is obvious that this control law is impractical since one would not know the future process outputs i.e. *y(t+j)*.

In order to derive a more practical solution for the future process outputs, consider the j-step ahead prediction error $\varepsilon(t+j|t) = y(t+j)-\hat{y}(t+j|t)$. Expanding (3.5) gives:

$$J_{Overall} = E\left\{\sum_{j=N_1}^{N_2}\left[r(t+j)-\hat{y}(t+j|t)-\varepsilon(t+j|t)\right]^2\right\}$$

$$= E\left\{\begin{array}{l}\displaystyle\sum_{j=N_1}^{N_2}\left[r(t+j)-\hat{y}(t+j|t)\right]^2 \\ \displaystyle+\sum_{j=N_1}^{N_2}\left[\varepsilon(t+j|t)\right]^2 \\ \displaystyle-\sum_{j=N_1}^{N_2}2\left[r(t+j)-\hat{y}(t+j|t)\right]\left[\varepsilon(t+j|t)\right]\end{array}\right\} \tag{3.6}$$

Note that the first term of (3.6) is the cost function that GPC minimises to find a controller.

The second term of (3.6), $\sum_{j=N_1}^{N_2}\left[\varepsilon(t+j|t)\right]^2$, represents the identification criterion i.e.

$$J_{Identification} = E\left\{\sum_{j=N_1}^{N_2}\left[y(t+j) - \hat{y}(t+j|t)\right]^2\right\} \tag{3.7}$$

Comparing this term with (3.4) it is clear that the entire control objective must use a model that is identified not only by using a single step ahead prediction but from N_1 up to N_2 steps ahead. The model parameter estimation strategy now attempts to model the predictions in the same range used for finding the control law, therefore having a cost function similar to J_{GPC}. Using (3.7) for identifying the process model parameters is called Long-Range Predictive Identification (LRPI).

The third term of (3.6) is a cross coupling of the control and identification terms. For a reasonably accurate model, the prediction errors for all j are relatively uncorrelated with the predicted control errors. This means that the expected value of the cross coupling is zero [12]. LRPI is developed by Shook *et al.* [11,12] with the assumption that this term tends to zero asymptotically as the prediction horizon tends to infinity .

3.3. Long-Range Predictive Identification - LRPI

Having found a suitable cost function to represent a long-range predictive identification strategy it is now necessary to find a means of implementing LRPI in an adaptive approach.

3.3.1. A Recursive Solution for LRPI

The nominal process model is described as,

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

The j-step ahead predictor

$$\begin{aligned} \hat{y}(t+j|t) &= E_j\hat{B}\Delta u(t+j-1) + F_j\, y(t) \\ 1 &= E_j\hat{A}\Delta + q^{-j}F_j \end{aligned} \tag{3.9}$$

Note that we again assume that the observer T-polynomial is equal to one.

Using the prediction error criterion, the LRPI cost function can now be written as,

$$
\begin{aligned}
J_{LRPI} &= \sum_{j=N_1}^{N_2}\left[y(t+j)-\hat{y}(t+j|t)\right]^2 \\
&= \sum_{j=N_1}^{N_2}\left[y(t+j)-E_j\hat{B}\Delta u(t+j-1)-F_j y(t)\right]^2 \\
&= \sum_{j=N_1}^{N_2}\left[E_j\hat{A}\Delta y(t+j)-E_j\hat{B}\Delta u(t+j-1)\right]^2 \\
&= \sum_{j=N_1}^{N_2}\left[E_j\hat{A}\Delta\frac{B^o}{A^o}u(t+j-1)-E_j\hat{B}\Delta u(t+j-1)\right]^2 \\
&= \sum_{j=N_1}^{N_2}\left[E_j\hat{A}\left(\frac{B^o}{A^o}-\frac{\hat{B}}{\hat{A}}\right)\Delta u(t+j-1)\right]^2
\end{aligned}
\tag{3.10}
$$

Using the same strategy for the above and realising that the least squares criterion is simply a one-step ahead predictor i.e. that N_1 and N_2 are equal to one, we can derive a cost function based on the single prediction error.

$$
\begin{aligned}
J_{LS} &= \left[y(t+1)-\hat{y}(t+1|t)\right]^2 \\
&= \left[\hat{A}\left(\frac{B^o}{A^o}-\frac{\hat{B}}{\hat{A}}\right)\Delta u(t-1)\right]^2
\end{aligned}
$$

Furthermore, using a data prefilter, $L(q^{-1})$, with the least squares identification, the least squares cost function is represented by,

$$
J_{LS} = \left[L\hat{A}\left(\frac{B^o}{A^o}-\frac{\hat{B}}{\hat{A}}\right)\Delta u(t-1)\right]^2 \tag{3.11}
$$

Although similarities can be seen between the final results of (3.10) and (3.11), a meaningful comparison can only really be made if we take a look at the frequency domain characteristics of each of the cost functions.

Using Parseval's theorem applied to the LRPI cost function,

$$
J_{LRPI} = \frac{T_s}{2\pi}\int_{-\frac{\pi}{T_s}}^{\frac{\pi}{T_s}}\left[\sum_{j=N_1}^{N_2}\left|E_j\hat{A}\Delta\right|^2\left|\frac{B^o}{A^o}-\frac{\hat{B}}{\hat{A}}\right|^2\Phi_u(\omega)\right]d\omega \tag{3.12}
$$

where T_s is the sample time and $\Phi_u(\omega)$ is the power spectral density of the input signal. We may also observe that the only term that is dependent on the prediction parameters N_1 and N_2 is the magnitude of E_j.

Similarly, the least squares criterion transforms to,

$$J_{LS} = \frac{T}{2\pi}\int_{-\frac{\pi}{T}}^{\frac{\pi}{T}} \left|L\hat{A}\Delta\right|^2 \left|\frac{B^o}{A^o} - \frac{\hat{B}}{\hat{A}}\right|^2 \Phi_u(\omega)d\omega \tag{3.13}$$

Comparing the frequency domain descriptions LRPI and LS cost functions we find that the only difference is the summation of E_j and the data prefilter, $L(q^{-1})$. Equating the LRPI cost function to a LS cost function with a data prefilter we obtain,

$$\begin{aligned} |L(\omega)|^2 &= \sum_{j=N_1}^{N_2} |E_j(\omega)|^2 \\ L(q^{-1})L(q) &= \sum_{j=N_1}^{N_2} E_j(q^{-1})E_j(q) \end{aligned} \tag{3.14}$$

where $L(q)$ is the complex conjugate of $L(q^{-1})$, similarly for $E_j(q^{-1})$.

Having modified the identification strategy we now have to apply this together with GPC to get the overall adaptive predictive control algorithm. Doing this also allows us to compare the filtering actions required for control and identification. Consider the process output given by

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

Notice that the disturbance term is modeled as filtered white noise (as discussed in chapter 2). Note too that we have included the observer polynomial $T(q^{-1}) \neq 1$ for generality. A graphical representation of the long-range predictive control scheme is shown in Fig. 3.1.

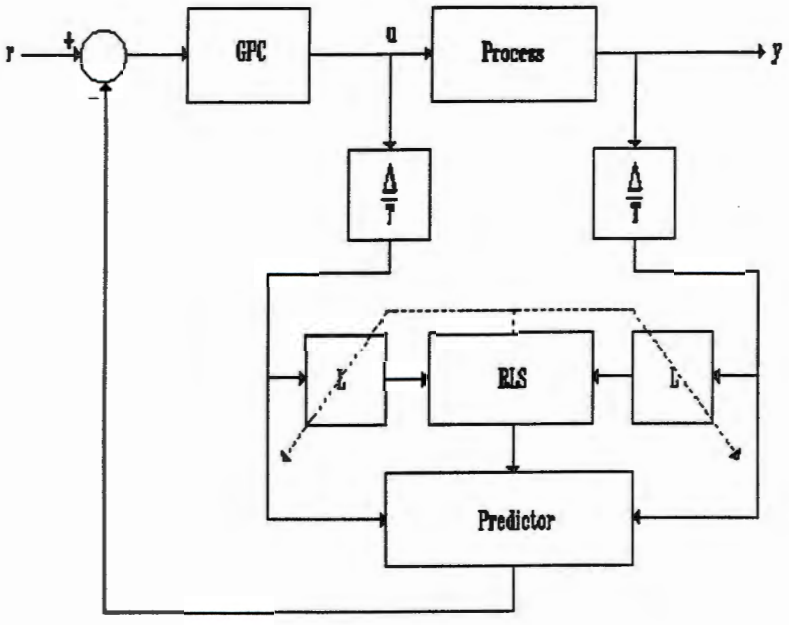

Figure 3.1 Block diagram showing GPC with LRPI-based estimation in closed loop

The total filtering action for identification of the plant is then

$$L_T(q^{-1}) = \frac{L(q^{-1})\Delta}{T(q^{-1})} \tag{3.15}$$

It is obvious then that there is more filtering required for estimation than there is for control. The overall control scheme combining GPC and LRPI is known as long-range predictive control (LRPC) [11].

3.3.2. The Data Prefilter $L(q^{-1})$

If the filter L has degree equal to the maximum prediction horizon, N_2, then there exists a unique L of order $(N_2 - N_1 + 1)$ that satisfies (3.14). The filter L can be found using a spectral factorisation algorithm. Shook *et al.* [11,13] have suggested the use of Peterka's spectral factorisation routine.

An important observation made by Shook *et al.* is that the filter L is a strong function of the maximum prediction horizon, N_2, while being a comparatively weak function of the estimated plant polynomial, $\hat{A}$, thus it is not sensitive to model mismatch or changes in the model of the plant.

The aim of spectral factorisation is to find is to find a polynomial $L(q^{-1})$ having all it's roots inside the stability region such that,

$$L'(q)L'(q^{-1}) = L(q)L(q^{-1}) \tag{3.16}$$

where $L'(q^{-1})$ is another polynomial that has roots inside and outside the stability region.

Consider the polynomial $L'(q^{-1})$. Find,

$$M(q) = L'(q)L'(q^{-1}) \tag{3.17}$$

i.e. by multiplying by it's complex conjugate, where

$$L'(q^{-1}) = l'_0 + l'_1 q^{-1} + \cdots + l'_{(N_2-1)} q^{-N_2+1}$$
$$M(q) = m_{(N_2-1)} q^{-1} + m_{(N_2-1)-1} q^{-(N_2-1)+1} + \cdots + m_0 + \cdots m_{(N_2-1)} q^{N_2-1}$$

The filter is then computed using Peterka's algorithm [14] shown in the following steps:

Step 1. *Initialise* $L = 1$

Step 2. $$Q = \frac{M(q^{-1})}{L(q^{-1})} q^{-N_2+1} + \frac{R(q^{-1})}{L(q^{-1})}$$
where Q and R represent the quotient (ratio of polynomials) and remainder polynomial.

Step 3. $L = q^{-N_2+1} Q$
Reordering the position of the coefficients

Step 4. $$L = \frac{L}{l_0}$$
Divide all coefficients by first coefficient

Step 5. Return to Step 1 and repeat until convergence

Step 6. When convergence is obtained, do

$$L(q^{-1}) = \frac{L(q^{-1})}{\sum_{k=0}^{N_2-1} l_k}$$
in order to obtain unity filter gain.

As discussed by Bohm *et al.* [14], this algorithm is very efficient, requiring only a polynomial division algorithm. In order to test for convergence the norm of the remainder polynomial is to be within a specified tolerance band. It has, however, been suggested that for practical adaptive control a single iteration per control sample is sufficient. In the adaptive approach a new M-polynomial is found at every time sample but the L-polynomial is retained from the previous sample. An important point that enhances the practical applicability of this method is the fact that at each sample the L-polynomial is stable (even though it might not be a spectral factor of L').

3.4. Simulations

Effect of parameter variations

Simulations similar to those used to demonstrate the GPC strategy were performed to test the LRPI. In order to make a direct comparison, the same parameter settings for the simulations were used. The GPC parameter variations for each simulation are shown in Table 3.1. The resulting simulations are shown in figures 3.2 - 3.5.

In order to make a direct numerical comparison of the GPC and LRPC schemes, two performance indexes were compared. These are the integral square error (ISE) and the integral time-multiplied absolute-error (ITAE). For a detailed discussion of these criteria see chapter 5 (essentially the ISE criterion quantifies speed of response and the ITAE quantifies the settling time). The ISE and ITAE values for the GPC and LRPC simulations are shown in Table 3.2.

Table 3.1 LRPC parameter settings for simulations

	Simulation 1	Simulation 2	Simulation 3	Simulation 4
N_1	1	1	1	1
N_2	10	5	10	10
Nu	1	1	3	3
λ	0.1	0.1	0.1	0.9

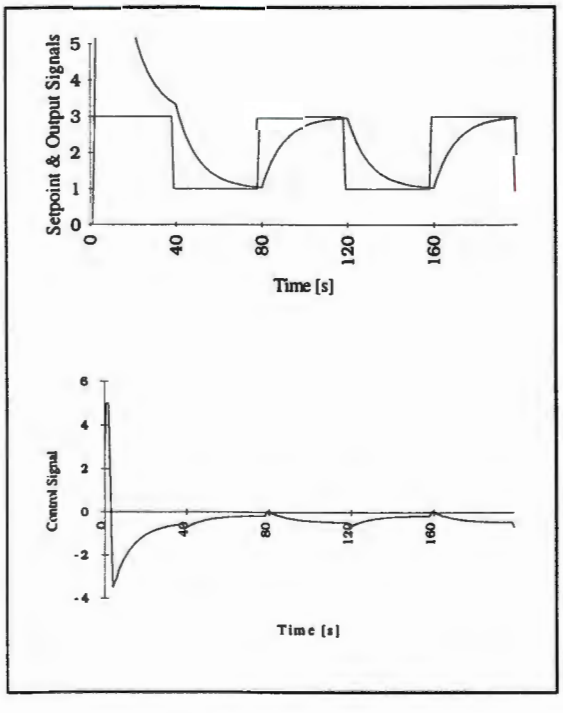

Figure 3.2 Simulation 1 :
$N_1 = 1, N_2 = 10, Nu = 1, \lambda = 0.1$

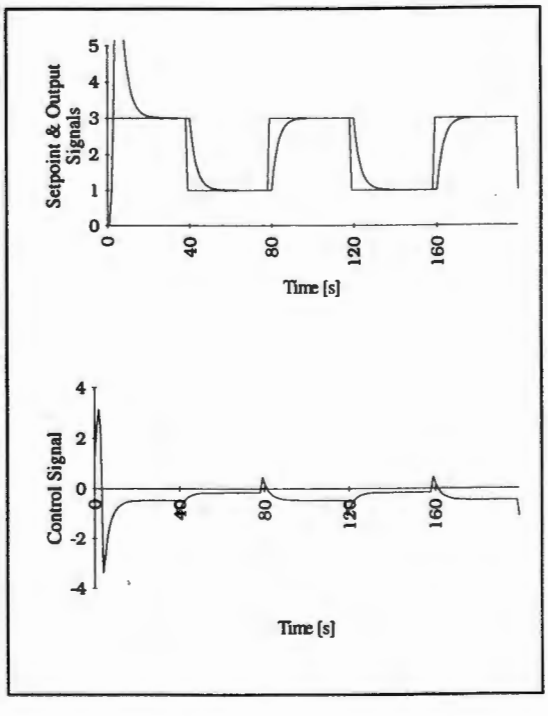

Figure 3.3 Simulation 2 :
$N_1 = 1, N_2 = 5, Nu = 1, \lambda = 0.1$

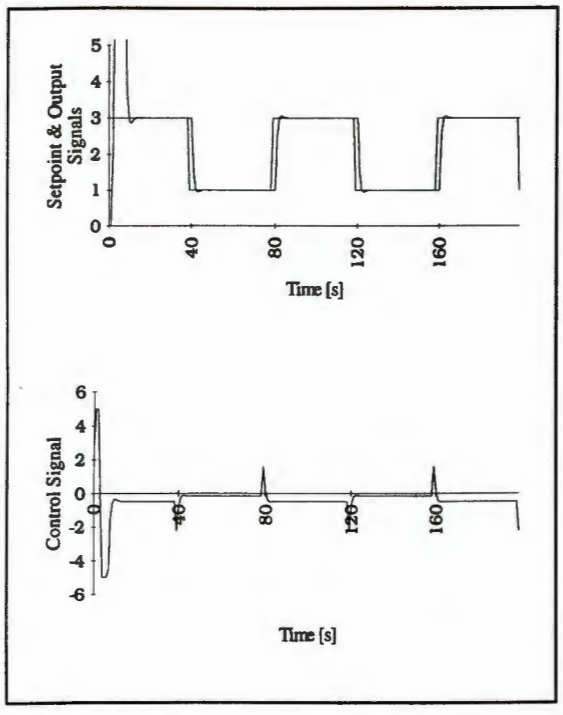

Figure 3.4 Simulation 3 :
$N_1 = 1, N_2 = 10, Nu = 3, \lambda = 0.1$

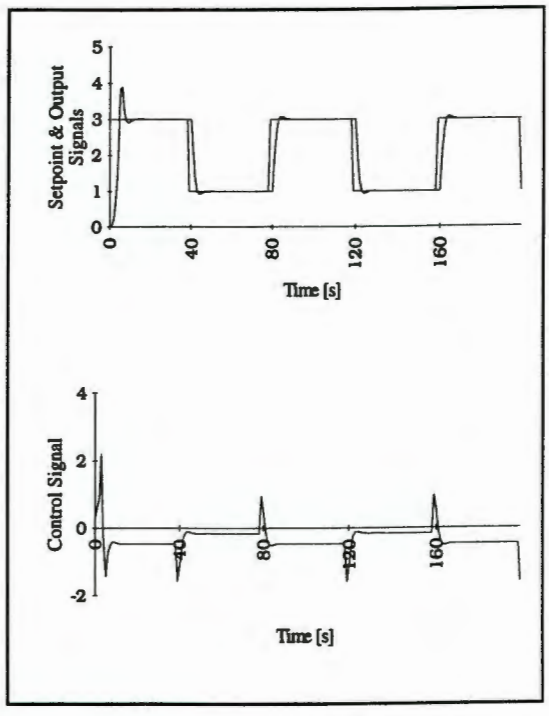

Figure 3.5 Simulation 4 :
$N_1 = 1, N_2 = 10, Nu = 3, \lambda = 0.9$

Closer comparison of GPC and LRPC responses are given via the ISE and ITAE values in Table 3.2.

Table 3.2 ISE and ITAE for GPC and LRPC comparisons of simulations 1 - 4.

		ISE	ITAE
Simulation 1	GPC	923.8	11445.2
	LRPC	912.8	11494.5
Simulation 2	GPC	159.3	4131.7
	LRPC	157.7	4133.2
Simulation 3	GPC	428.6	2261.3
	LRPC	428.7	2276.0
Simulation 4	GPC	67.8	2427.5
	LRPC	67.7	2437.6

In simulations 1 and 2 LRPC provided the slightly faster response as can be seen from the ISE values. Simulations 3 and 4 show virtually the same speed of response. In all the simulations the LRPC responses had larger ITAE values, indicating a more active response than the GPC simulations.

Disturbance rejection

To demonstrate the effect of load and input disturbances, the simulations in Fig. 3.6 and Fig. 3.7 were carried out. Table 3.3 shows the GPC parameters used in each of the simulations.

Table 3.3 LRPC parameter settings for simulations

	Simulation 5	Simulation 6
N_1	1	1
N_2	5	5
Nu	1	2
λ	0.1	0.1

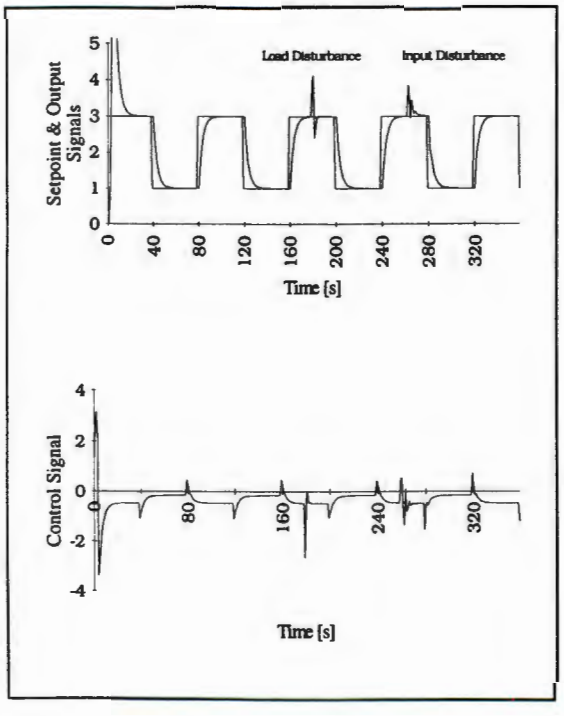

Figure 3.6 Simulation 5: showing load and input disturbance rejection
$N_1 = 1, N_2 = 5, Nu = 1, \lambda = 0.1$

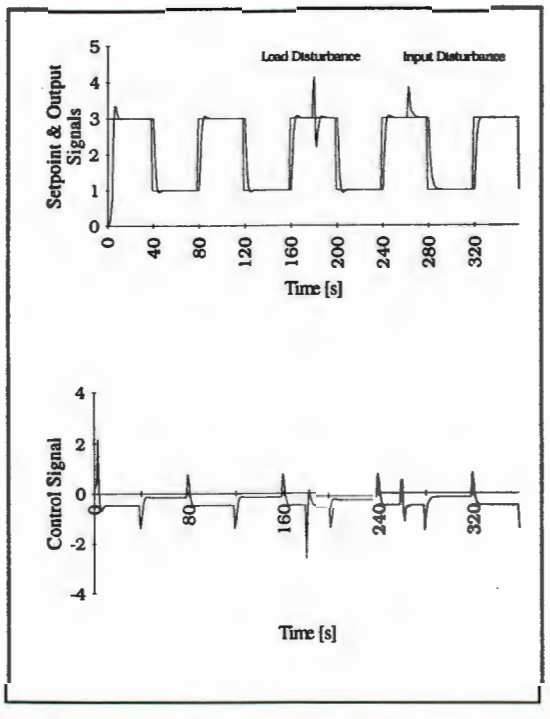

Figure 3.7 Simulation 6: showing load and input disturbance rejection
$N_1 = 1, N_2 = 5, Nu = 2, \lambda = 0.1$

Closer comparison of GPC and LRPC responses are given via the ISE and ITAE values in Tables 3.4 and 3.5.

Table 3.4 ISE and ITAE for GPC and LRPC comparisons of simulations 5.

	GPC		LRPC	
	Load Disturbance 160 - 180 [s]	Input Disturbance 240 - 280 [s]	Load Disturbance 160 - 180 [s]	Input Disturbance 240 - 280 [s]
ISE	15.2	16.2	15.2	14.2
ITAE	2121.5	3554.2	2073.1	3004.3

Table 3.5 ISE and ITAE for GPC and LRPC comparisons of simulations 6.

	GPC		LRPC	
	Load Disturbance 160 - 180 [s]	Input Disturbance 240 - 280 [s]	Load Disturbance 160 - 180 [s]	Input Disturbance 240 - 280 [s]
ISE	13.2	11.0	13.1	11.1
ITAE	1712.3	1981.4	1709.5	2053.3

In simulation 5 the LRPC response provided better disturbance rejection than the GPC solution. This can be seen from the lower ITAE values, especially the input disturbance rejection. In simulation 6 LRPC was able to improve the load disturbance rejection slightly but no improvement was made on the input disturbance. From the graphs it can be seen, however, that LRPC provided the smoother responses to the disturbances, with a much less active control signal.

Effect of model change

Simulations to demonstrate the adaptive ability of the LRPC scheme are now presented. In Fig. 3.8, the results for a system varying simulation run are shown. Each model change is indicated on the graph while the particular models are given in Table 3.6.

Table 3.6 Order of models for simulation shown in Fig. 3.8

Model 1	$\frac{1}{s+0.1}$
Model 2	$\frac{5}{s(s+5)}$
Model 3	$\frac{1}{5s+1}$
Model 4	$\frac{s+2}{s^2+4s+8}$
Model 5	$\frac{e^{-2s}}{5s+1}$

While close comparison of Fig. 3.6 and Fig. 2.11 will reveal that LRPI has brought an improvement to GPC, it is useful to again look at the ISE and ITAE. This is shown in Table 3.7.

3.5. Summary

Chapter 3 introduced the concept of long-range predictive control (LRPC). The original GPC scheme of Clarke *et al.* used a simple RLS algorithm as an identification strategy for parameter estimation. Shook et al. demonstrated that this was not an ideal solution and suggested the use of a long-range predictive identification (LRPI) strategy.

Essentially LRPI uses multi-step ahead prediction to develop a parameter estimation strategy which is then a dual of the GPC control scheme, extending the overall method to an LRPC design.

Simulations similar to those of chapter 2 were executed in order to demonstrate the effect of the LRPI strategy.

CHAPTER 4

EQUIPMENT

4.1. Introduction

In this chapter a closer look at the equipment used for the implementation of a long-range predictive controller is taken. A GPC control strategy was investigated on a laboratory flotation plant. This plant simulates the fluid dynamics of a typical industrial flotation process circuit. A description of the computer system (both hardware and software) used for the implementation is also given.

4.2. The Flotation Plant Simulator

Flotation is a process for separating valuable solid raw materials from waste based on the difference in surface properties of the raw materials [23]. In the process a mixture of particles and water, each with different surface properties, is prepared and is known as a slurry. Air bubbles are introduced into the slurry causing one of the minerals to adhere to the bubbles forming bubble-particle aggregates. These aggregates will rise to the surface of the slurry and can be removed as a frothy concentrate. The remaining slurry from which mineral particles are not removed is known as the tailings. The product circuit of a typical flotation process found in the mineral extraction industry is shown in Fig. 4.1.

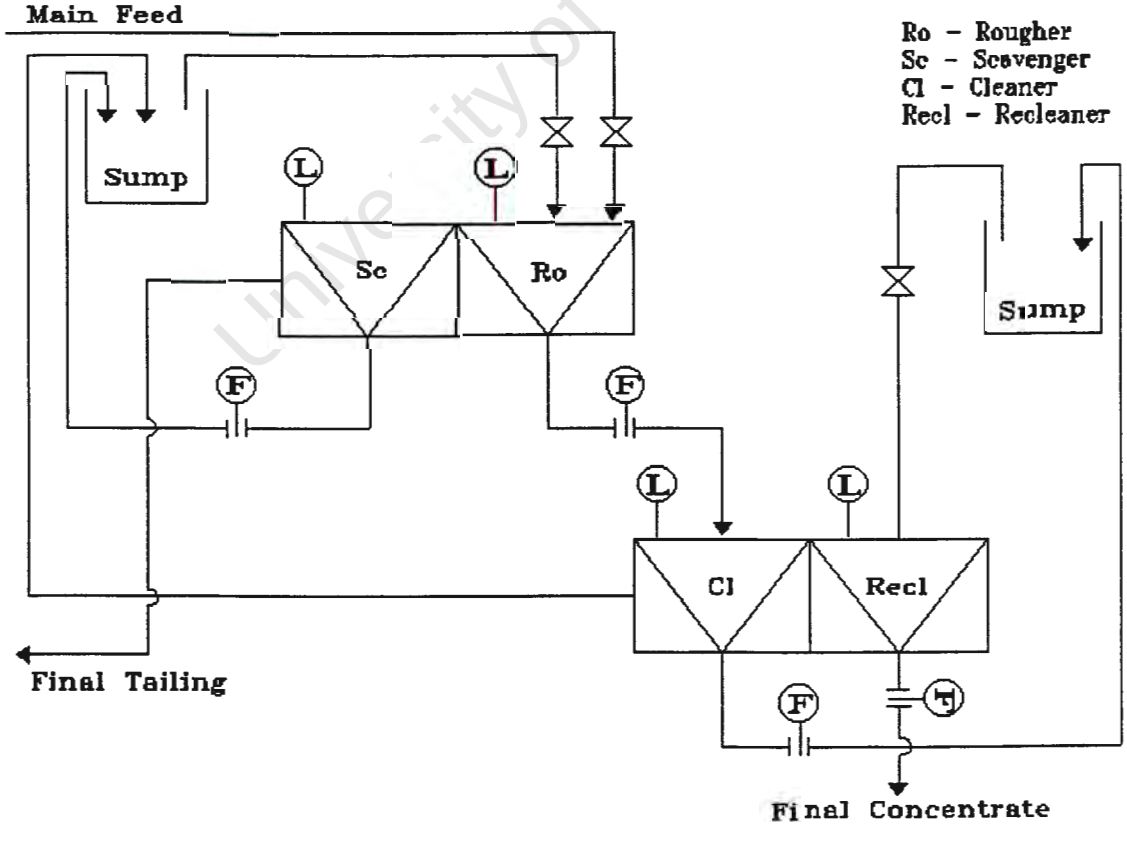

Figure 4.1. Product circuit for a typical mineral extraction process

In Fig. 4.1., the scavenger, rougher, cleaner and recleaner are known as flotation cells. In a typical flotation cell the input flow is divided into two outputs:

- *The concentrate* - the rate of concentrate output is dependent on the product level in the flotation cell.
- *The tailing* - controlling the rate of the tailing output controls the level of product in the flotation cell.

A flotation cell simulator is shown in Fig. 4.2. The product used in the simulator is water. The flotation cell characteristics are simulated as follows:

- *The concentrate* - this is the output of the inverted U-piece. Concentrate output is only simulated once the product level is higher than the cross-piece. The breather pipe prevents the siphoning of any product once the level falls below the cross-piece height. An inverted U-piece is used mainly because the flow meter has to be placed in an upwardly flowing stream for accurate measurement.
- *The tailing* - this output is controlled via the tailing control valve which in turn controls the product level.

Two other outputs are shown in Fig. 4.2. These are:

- The cell overflow - this output bypasses the tailing valve as a precaution in case of flooding. This may result if the combined concentrate and tailing outputs are less than the input flow. In this way flooding of the cell is avoided without affecting the total product mass in the flotation circuit.
- The system overflow - this output prevents flooding when the total product in the flotation circuit exceeds the product capacity.

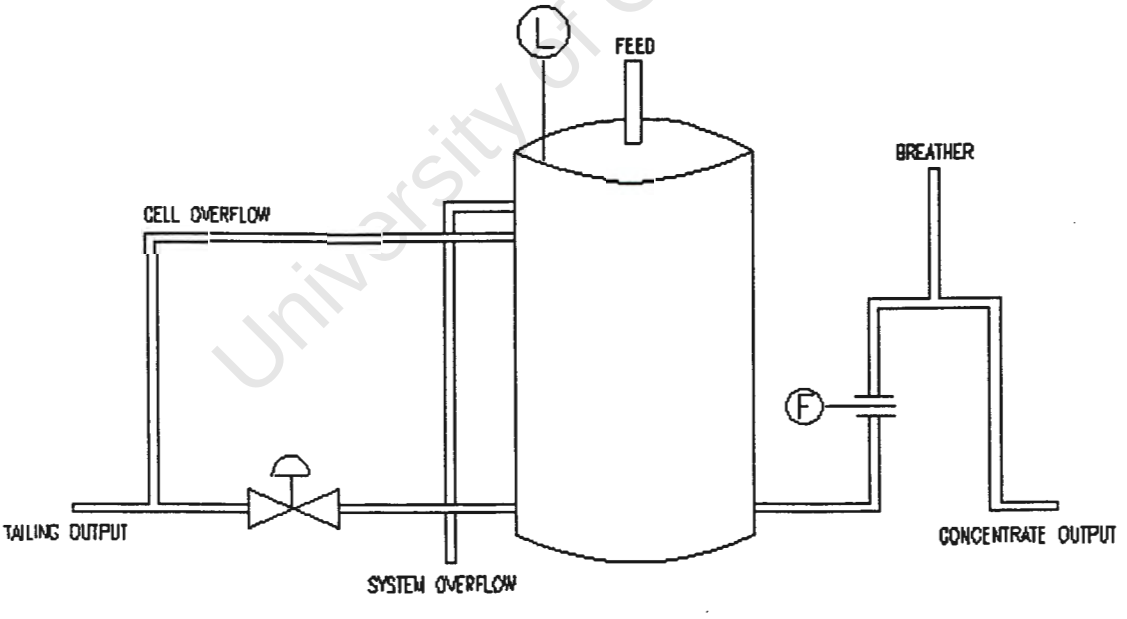

Figure 4.2. Flotation cell setup for flotation plant simulator

Using the method aforementioned for describing a flotation cell simulator, a flotation plant simulator of the product circuit in Fig. 4.1. can be developed. The flotation plant simulator at the U.C.T. Control Laboratory [24] is shown in Fig. 4.3 and Fig. 4.4.

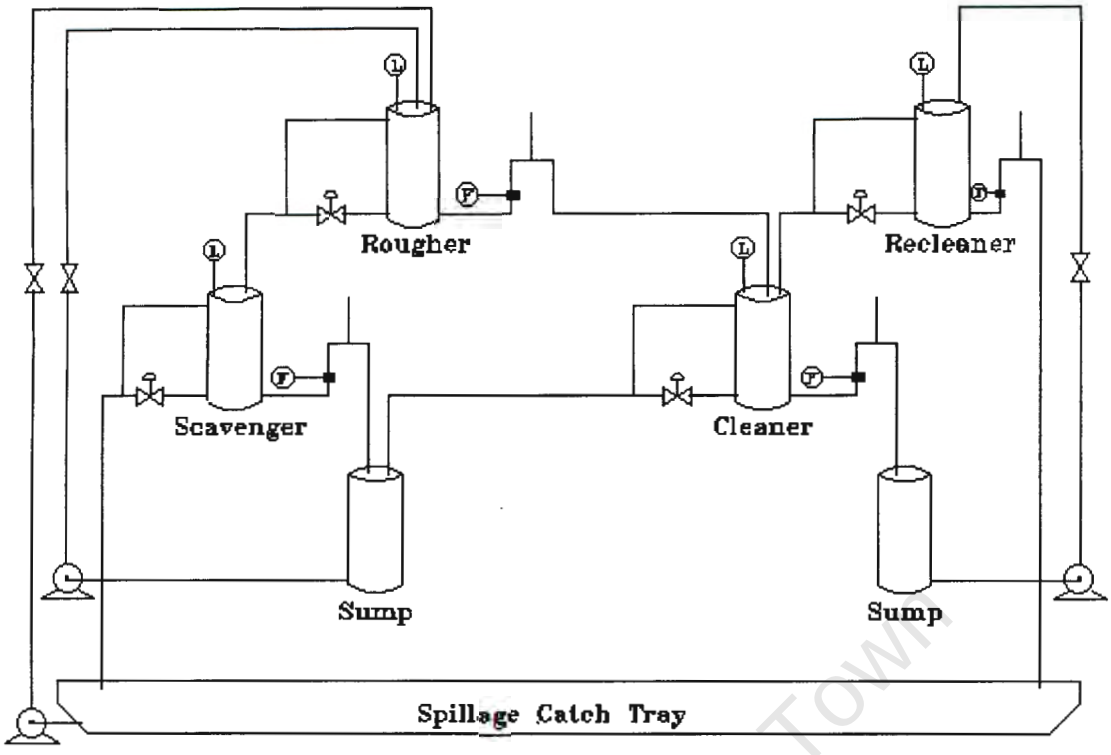

Figure 4.3. Flotation plant simulator based on product circuit in Figure 4.1.

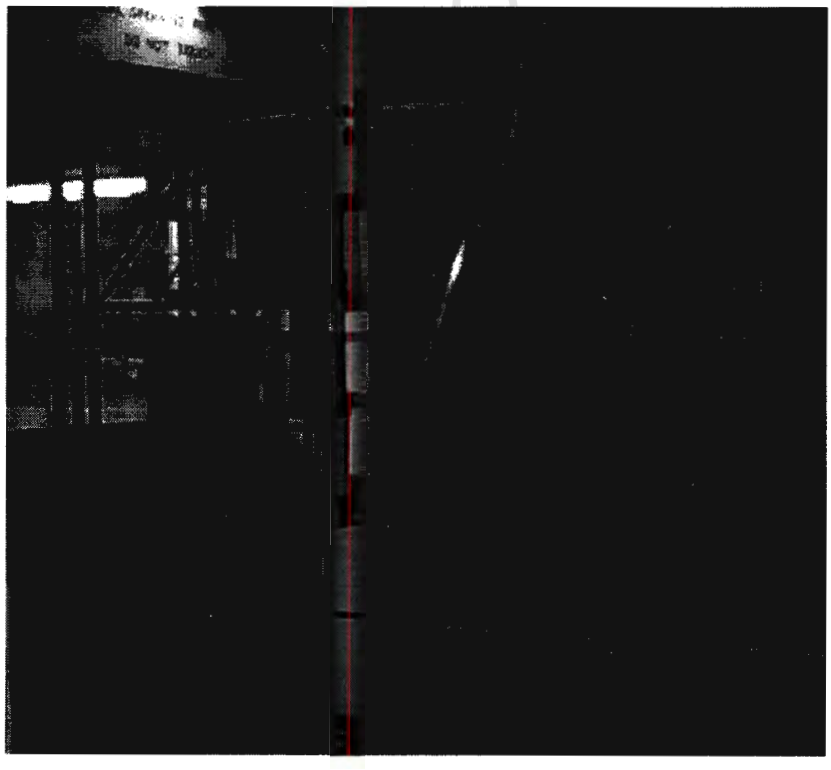

Figure 4.4. Photograph of flotation plant simulator

4.3. Instrumentation of the Flotation Plant Simulator

For the purposes of this thesis the signals of interest from the flotation plant simulator were the level settings. The instrumentation used was thus the level probes in the flotation cell simulators. Capacitive level probes are used in each of the tanks. This is setup in the following way:

- A strip of aluminium is placed on the inside wall of the tank (since the tanks are made of plastic). This strip of aluminium acts as a reference electrode for the capacitance measurement.
- The probe is then fixed in the centre of the tank.

The product dynamics of the flotation plant simulator are controlled via the pneumatic valves installed on the tailing outputs of each of the flotation cells. Each valve has a current to pressure converter to convert the electrical control signal into valve action.

4.4. Control System

For the thesis the design of a stand-alone control system was investigated. This system would comprise of an industrial computer and a real-time multi-tasking operating system.

Over the last decade advanced control algorithms have become easier to implement with the increase in available computer power. Personal computers (PC's) have become increasingly popular and the design and simulation of control systems are less dependent on the hardware i.e. almost any modern PC is capable of the task.

As popular and economically viable as PC's have become, one important fact has to be remembered: PC's are not suited to the industrial environment, they are simply not robust enough [4,25,26].

Computers used in industry have to survive the harsh environments encountered, taking into account the cause of hardware failure such as abnormal temperatures, vibration, dust and moisture etc. Industrial computers have to have these factors included in their product and design specification so that the chance of failure is minimised.

This thesis investigated the industrial VMEbus manufactured by OR Industrial Computers.

4.4.1. The VMEbus System

The VMEbus has been utilised in industry for about a decade and is reputedly one of the more popular industrial BUSes. Compared to a PC the VMEbus is more suited to the industrial environment. VMEbus boards have a standard temperature range of 0° to 70°C. The shock and vibration strength of the VMEbus boards are superior to that of the cheaper PC systems. If necessary, the VMEbus system can be protected in

watertight and dustproof housings. Noise immunity is a design aspect that the manufacturers have also taken into account. Furthermore the VMEbus conforms to the 19" standard which makes fitting components and hardware upgrading easy.

The flexibility of the VMEbus allows it to be used as a development platform as well as for the actual implementation of the control system. This not only saves cost, but also decreases the risk of failure due to porting from the development platform to the implementation platform. The VMEbus setup utilised is shown in Table 4.1.

Table 4.1. The VMEbus system components

V486DX-C5I Processor Module (16Mb DRAM)
IBMIO2-CDS I/O Module
PC Box
1 Gb Hard Drive & Stiffy Drive
Keyboard
Mouse
Super VGA Monitor
Analog I/O Board
Digital I/O Board
Card Cage (inc. VMEbus slots,PSU)
Housing for Card Cage

The processor module is an IBM-AT compatible motherboard. The motherboard has a VMEbus and a ISAbus interface allowing it to be used in ISAbus systems (ordinary PC's) as well. The processor is an Intel 80486-DX2 running at 50 MHz. There is also 16 Mb of DRAM on the motherboard.

The I/O (input/output) module is an IBM compatible multifunction I/O board. It contains the VGA graphics controller, floppy disk controller, an IDE interface for hard disks, a SCSI controller and serial and parallel ports. The VMEbus system is shown in Fig. 4.5.

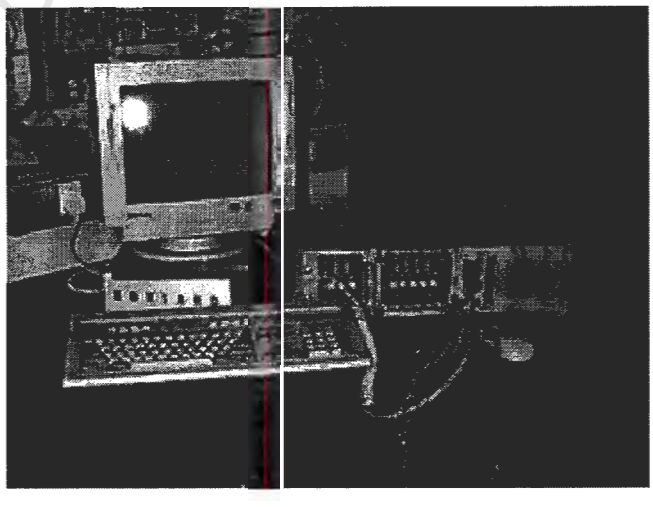

Figure 4.5. The VMEbus system

4.4.2. The Operating System

An operating system for control applications requires multi-tasking and real-time capability for accurate implementation of the control algorithms. While there are a few operating systems that fit this description (e.g. iRMX, VMS), cost, however, played an even more important role. It was finally decided to use the Microsoft Windows NT Workstation operating system.

Windows NT is a 32-bit operating system which has a graphical user interface (GUI). It is designed to be robust, responding predictably to both hardware and software error conditions. One unique feature of the operating system is its portability. Windows NT has been designed to be run on different hardware platforms, such as *x*86, MIPS or ALPHA. User written applications need only be recompiled for the particular platform in order to be ported. This feature is provided by the Windows NT Kernel and Hardware Abstraction Layer (HAL). Essentially the HAL isolates the kernel code from the hardware, so that the same 'virtual machine' is presented to the NT executive, device drivers etc.

It has been our experience that the operating system requires considerable resources (memory, processor speed etc.) in order to function satisfactorily. This is, however, an increasingly common trend in modern software and no other option exists other than upgrading the hardware.

Windows NT as a Real-Time System :
Firstly it is necessary to attempt to define a real-time system:

> A real-time system is one in which the correctness of the computation not only depends on the logical correctness of the computation, but also upon the time at which the result is produced. If the timing constraints of the system are not met, system failure is said to have occurred. [uninet comp.realtime real-time FAQ]

While a common mistake is to interpret real-time as meaning fast, we can see from this definition that speed has nothing to do with it. Real-time would be more aptly described as predictable.

A true real-time operating system must respond to events within a predictable time frame and must be independent of any other activities currently being executed. Windows NT does not conform wholly to this standard, but has reduced constraints, being able to respond fairly quickly to events but not as deterministic. Windows NT is said to 'be good enough to service events so that the response time should be satisfied on average' [15].

Although Windows NT has sufficient real-time capabilities, it falls short of and was probably not designed to compete with specialised real-time operating systems. For true real-time capabilities special hardware needs to be used in conjunction with these operating systems. These real-time systems are dedicated systems and are

usually not very flexible, while Windows NT provides additional features (e.g. a GUI, virtual memory management) but suffers a reduction in response time.

Windows NT as a Multi-Tasking System :
Versions of the 16-bit Windows, such as the popular Microsoft Windows 3.1, have a limited multi-tasking capability known as non-preemptive multi-tasking. It works on the principle that one application has to inform the operating system that it has completed its processing before another application is assigned processor time. This has the disadvantage that the user can occasionally lose valuable time waiting for an application to finish before being allowed to continue working on some other application.

The Windows NT operating system overcomes this problem by using preemptive multi-tasking. Preemptive multi-tasking allows multiple applications to run by ensuring that no one application has sole access to the system resources. Each running application is assigned time slices by the Windows NT kernel based on their priority. Higher priority processes obviously have 'first choice' for available time. Since no one application controls all the system resources, the operating system is much more robust.

To understand how applications are able to 'multi-task', it is necessary to understand how programs run under Windows NT [16]. Once an application is invoked, the instance of the running program is known as a process. No code is executed at this time since processes really only make available address space (and at times other resources) for code and data. In order for the code to be executed, a process must contain one or more threads. If a process has many threads, they are seen to be able to execute code concurrently in the process - this is accomplished by the Windows NT operating system which provides each thread with time quantums. It is possible to set the priority of threads so that a time critical thread, with highest priority, is executed first.

This lends itself to a semi-real-time system, provided the sampling is not executed at high frequency. Time critical high frequency sampling could result in sluggish performance of the rest of the operating system. While this may not sound like an attractive prospect, it must be mentioned again, that Windows NT is hardware resource dependent. Using a more powerful processor (or indeed a multi-processor) and more memory could see vast improvements in performance.

For the control system investigated, real-time sampling of the order of 1 second was required. Windows NT was adjudged to perform adequately for these purposes. The sampling for the control system was executed by using the available Windows NT timing commands. Each controller was implemented as a separate thread.

4.4.3. The Programming Language

The programming language used for coding the control algorithm was C/C++. While this code is portable to most C-compilers it was necessary to choose a compiler for writing windows applications. Since the program was written to run under the Windows NT operating system it was decided that the application would be a 32-bit

one. The C-compiler decided on was the Microsoft Visual C++ compiler version 2.0 which was specifically designed to write 32-bit applications for the Windows NT and Windows 95 environments.

4.4.4. Problems Encountered with the Control System

Numerous problems were experienced during the initial installation of the Windows NT operating system on to the VMEbus computer. After investigation it was discovered that the SCSI-interface of the VMEbus was not compatible with Windows NT. This necessitated the replacement of the SCSI hard drive with an IDE hard drive. The ethernet card too was unsupported by NT but no solution could be found to overcome this problem, which was not essential for this thesis.

A major problem was encountered during programming the input/output (I/O) routines for Windows NT. It was not possible to access the VMEbus from the operating system and therefore use the I/O cards (even though access was possible from the DOS operating system).

4.4.5. A Solution for the Windows NT - VMEbus Interface Problem

The problem entails directly accessing physical memory from the Windows NT environment. The Windows NT memory management creates a virtual address space for each application and the user loses the ability to access the physical address space.

In order to use the VMEbus interface the VME logic must be enabled. This is done by setting a bit in a particular I/O port (setting bit 1 in I/O port 163H, known as the VMEbus configuration register). The VMEbus modules are then usually accessed by what is known as short I/O access. The short I/O access is done through a memory sector in the ATbus memory.

ATbus		**VMEbus**
0D0000H	⇐Short I/O access⇒	xx0000H
0DF7FFH		xxF7FFH

Thus the I/O is memory-mapped and it is necessary to be able to access a specific physical memory address in order to communicate with a particular VME module e.g. the analog I/O card.

Trying to access the VMEbus from the Windows NT operating system yields the same problem. In Windows NT, applications cannot write directly to physical memory addresses, and in fact, the memory required by the VME is initially marked as invalid by the Windows NT memory manager.

Accessing the I/O port 163H in the Windows environment was not a problem. The command to do this, though, was not portable to Windows NT, which meant that aside from being unable to access the VME modules, the VMEbus could not be enabled. The problem is that in Windows NT, application programs have insufficient processor privilege to access I/O ports.

The solution to this problem was to write a device driver. A device driver is essentially a 32-bit program which has enough processor privilege to perform whatever task is required. The device driver would therefore give a process access to the VMEbus port and the physical memory associated with the system.

4.5. Summary

Chapter 4 detailed the physical layout of the flotation plant simulator on which a GPC/LRPC application was to be carried out. A description of the control system was also given. This included a discussion of the hardware (the VMEbus) and relevant software used (operating system and programming languages). The problems encountered in setting up the control system are specified along with the steps taken to overcome them.

CHAPTER 5

AN APPLICATION OF LONG-RANGE PREDICTIVE CONTROL

5.1. Introduction

The Generalised Predictive Control (GPC) and Long-Range Predictive Control (LRPC) methods were implemented on a flotation plant simulator. The physical setup of the plant was discussed in chapter 4. In this chapter the results obtained for the control of the plant are presented.

The flotation plant simulator is a multivariable plant having four inputs and four outputs i.e. it is a "four by four" system. The four inputs correspond to the four control valve positions. The four outputs are given by the level of water in each tank. The aim is then to control the level of the water in the tanks by means of the output control valves.

5.2. The Plant Model

The four tanks in the circuit emulate the rougher, scavenger, cleaner and recleaner of the flotation process. The transfer function matrix model of the system is:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \underbrace{\begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \\ g_{41} & g_{42} & g_{43} & g_{44} \end{bmatrix}}_{\textit{Plant Transfer Function Matrix}} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{5.1}$$

where

y_1 = *Rougher Level* u_1 = *Rougher Valve*

y_2 = *Scavenger Level* u_2 = *Scavenger Valve*

y_3 = *Cleaner Level* u_3 = *Cleaner Valve*

y_4 = *Recleaner Level* u_4 = *Recleaner Valve*

5.3. GPC Structure

Since the GPC method that was studied was based on a single-input single-output (SISO) system, it was decided to implement a diagonal controller system for control of the simulator. This meant that there would be only four GPC controllers, one for each of the input/output pairs. In order to understand why this should be feasible,

consider a 2-input-2-output multivariable system (since the mathematics becomes slightly less confusing),

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$y_1 = g_{11}u_1 + g_{12}u_2 \quad \ldots \quad Loop\ 1$$
$$y_2 = g_{21}u_1 + g_{22}u_2 \quad \ldots \quad Loop\ 2$$

The controller that is needed for control of output, y_1, in the first loop is defined by,

$$u_1 = k_{11}r_1 - k_{11}y_1$$

Where k_{11} is the controller and r_1 is the setpoint for the first loop. Using this value for the control signal, u_1, and substituting into the second loop it is possible to obtain an equation showing that the second loop is now modified,

$$y_2 = \left[g^*_{22}\right]u_2 + \left[g_{21}(k^{-1}{}_{11} + g_{11})^{-1}\right]r_1 \tag{5.2}$$

Where g^*_{22} represents the model of the second loop that has been changed and is a function of k_{11}. It is also evident that interaction from the setpoint of the first loop exists [17,18]. From the equation we can deduce that an adaptive approach should be able to accommodate the change in plant model that is introduced.

While not an optimal solution, this would clearly demonstrate the adaptive capability of GPC as the model of each of the tanks would change with the amount of interaction from the other tanks.

5.4. Performance Indexes

The results obtained are compared with each other using two performance indexes. A performance index (PI) is used to indicate the 'goodness' of a system [19]. The first performance index used is that of the integral-square error (ISE) criterion in which the quality of system performance is evaluated by

$$PI_{ISE} = \int_0^\infty e(t)^2\,dt \tag{5.3}$$

Systems responding with a quick response time i.e. having a rapid decrease in initial error, will have smaller ISE values. Comparison of ISE values thus incorporates a comparison of the rise time of the plants.

The second performance index used is that of the integral of time-multiplied absolute-error (ITAE) criterion. This is shown by

$$PI_{ITAE} = \int_0^{\infty} t|e(t)|dt \tag{5.4}$$

The ITAE neglects large initial errors but penalises later transient response errors heavily. Systems with a small overshoot and well damped oscillations will have lower ITAE values.

5.5. Generalised Predictive Control

Six experiments were carried out to test the performance of the GPC algorithm. The sample time for the application is set to 1 [s].

The GPC tuning parameters are set to the following values. Note that the values given for N_1, N_2 and Nu are given in terms of sampling intervals. A standard GPC parameter setup was obtained by experimentation. These parameter defaults are used for experiment 1.

Table 5.1 GPC parameter settings for flotation plant experiments

	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5
N_1	1	1	1	1	1
N_2	50	50	50	10	90
Nu	1	2	1	1	1
λ	0.5	0.5	0.01	0.5	0.5

Each experiment consists of four groups of graphs. Each group shows one of the tank levels stepped and the corresponding responses of all four tanks. These graphs and performance index values are presented in Appendix B.

A summary of the ISE and ITAE values obtained for experiments 1 - 5 is given below in Table 5.2 and Table 5.3.

Table 5.2 Tables summarising ISE values for experiments 1 - 5

Experiment 1				
	Tank Level Stepped			
ISE for Tank	Recleaner	Cleaner	Scavenger	Rougher
Recleaner	4.73545	0.00625	0.00475	0.01335
Cleaner	0.99635	4.24855	0.56380	0.71175
Scavenger	0.03060	0.06220	4.31300	0.03335
Rougher	0.10495	0.11645	0.09590	4.38185

Experiment 2				
	Tank Level Stepped			
ISE for Tank	Recleaner	Cleaner	Scavenger	Rougher
Recleaner	4.72740	0.01405	0.01360	0.00640
Cleaner	1.96685	2.45005	0.85475	0.97880
Scavenger	0.02485	0.06815	3.93700	1.41435
Rougher	0.09730	0.19705	0.10715	2.93320

Experiment 3				
	Tank Level Stepped			
ISE for Tank	Recleaner	Cleaner	Scavenger	Rougher
Recleaner	4.3435	0.0134	0.0039	0.0097
Cleaner	1.1233	5.3241	0.4277	0.7679
Scavenger	0.0482	0.0377	3.9410	0.0479
Rougher	0.1150	0.1522	0.1032	4.9375

Experiment 4				
	Tank Level Stepped			
ISE for Tank	Recleaner	Cleaner	Scavenger	Rougher
Recleaner	4.4769	0.0212	0.0315	0.0359
Cleaner	0.9727	2.5798	0.4705	0.7771
Scavenger	4.7155	0.2559	3.2116	0.0025
Rougher	0.4320	0.1760	0.1006	1.3582

Experiment 5				
	Tank Level Stepped			
ISE for Tank	Recleaner	Cleaner	Scavenger	Rougher
Recleaner	6.7865	0.0237	0.0108	0.0083
Cleaner	1.0484	5.8250	0.7093	0.8319
Scavenger	0.1957	0.8943	9.1545	3.3798
Rougher	0.3926	0.3631	0.1604	8.8276

Table 5.3 Tables summarising ITAE values for experiments 1 - 5

Experiment 1				
	Tank Level Stepped			
ITAE for Tank	Recleaner	Cleaner	Scavenger	Rougher
Recleaner	1426.70	79.53	65.59	140.13
Cleaner	1484.83	2240.66	846.15	1100.31
Scavenger	201.98	407.35	1408.31	248.78
Rougher	478.81	559.18	479.01	1491.27

Experiment 2				
	Tank Level Stepped			
ITAE for Tank	Recleaner	Cleaner	Scavenger	Rougher
Recleaner	1399.66	148.55	143.99	73.51
Cleaner	2128.76	1996.83	978.94	1346.41
Scavenger	234.93	440.64	1184.53	988.76
Rougher	482.13	740.38	483.43	1143.70

Experiment 3				
	Tank Level Stepped			
ITAE for Tank	Recleaner	Cleaner	Scavenger	Rougher
Recleaner	1332.15	125.90	42.53	145.71
Cleaner	1614.48	2520.87	829.58	1226.71
Scavenger	277.24	299.41	1314.93	295.96
Rougher	515.06	639.62	471.51	1670.78

Experiment 4				
	Tank Level Stepped			
ITAE for Tank	Recleaner	Cleaner	Scavenger	Rougher
Recleaner	1740.35	230.22	295.64	272.47
Cleaner	1575.48	1978.22	771.26	1178.19
Scavenger	2152.43	919.74	1675.43	1627.87
Rougher	582.32	732.08	449.89	621.14

Experiment 5				
	Tank Level Stepped			
ITAE for Tank	Recleaner	Cleaner	Scavenger	Rougher
Recleaner	2440.01	165.47	83.54	85.63
Cleaner	1592.82	2382.83	904.74	1426.32
Scavenger	702.46	1909.48	4342.56	3896.93
Rougher	900.08	873.38	614.20	2730.71

Experiment 1 has the GPC parameters set to 'normal' values to serve as a comparison for parameter variations in the other simulations.

Experiment 2 shows the effects of increasing the control horizon, *Nu*. All the responses to the step tests were faster than those of experiment 1. Only the recleaner step showed hardly any improvement but this can be ascribed to the fact that the tank was being filled up as fast as is physically possible. A feature of this set of responses is the higher ITAE values which represents a more active output signal. This is explained by the control signal shown in Fig. 5.2 which should be compared with the control signal for experiment 1 in Fig 5.1. Fig. 5.3 and Fig. 5.4 show the histograms of the control signals obtained. The mean and standard deviation values acquired from these histograms are shown in Table 5.4.

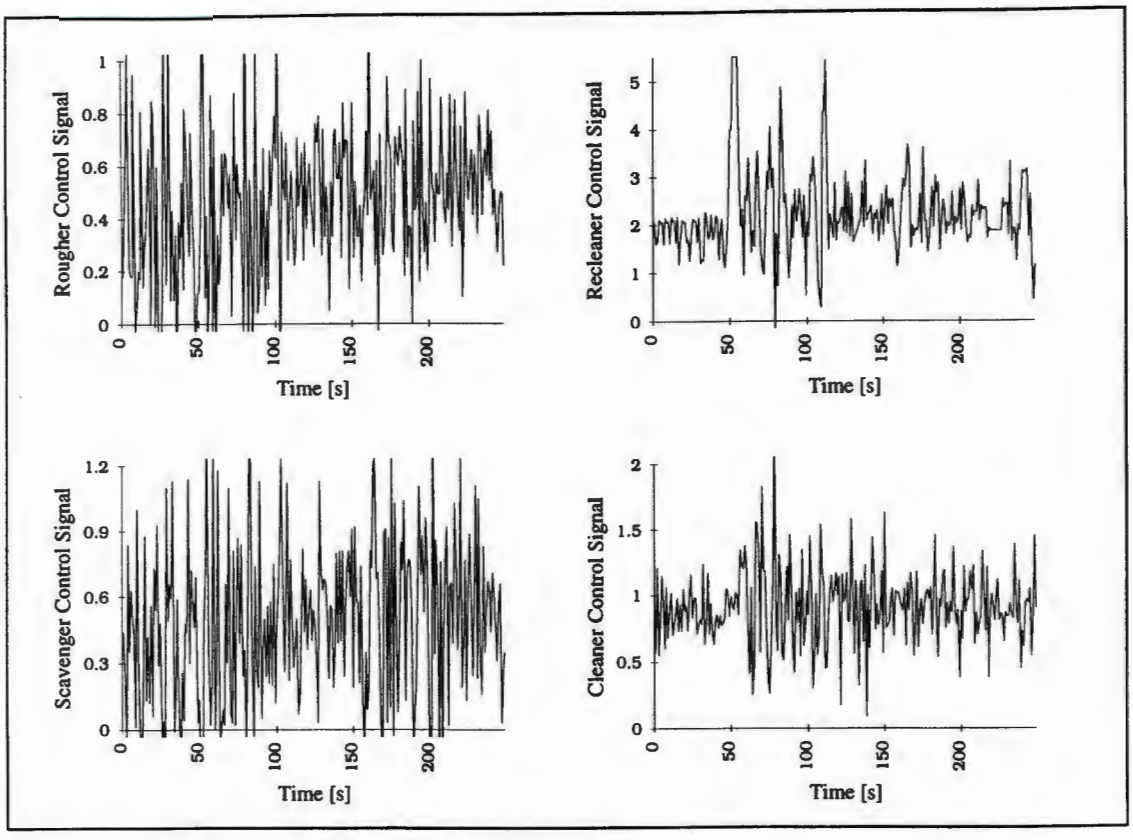

Figure 5.1 Control signal for experiment 1: Recleaner stepped

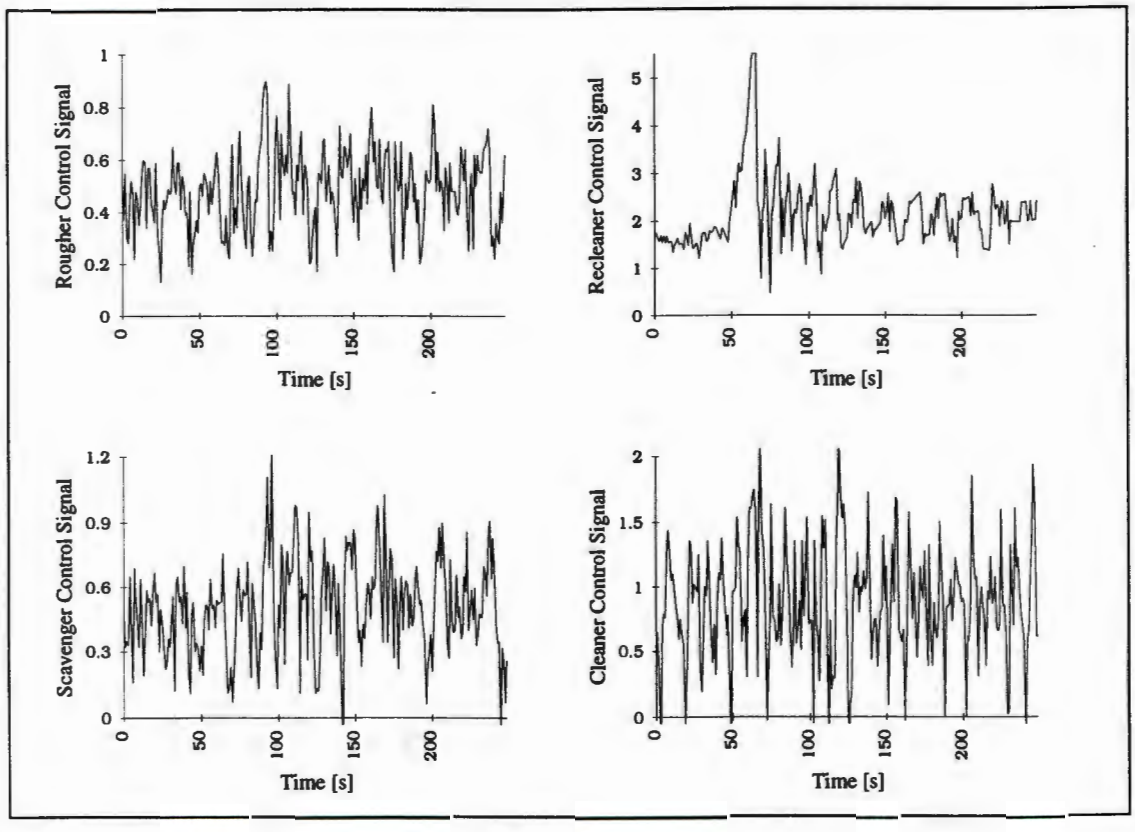

Figure 5.2 Control signal for experiment 2: Recleaner stepped

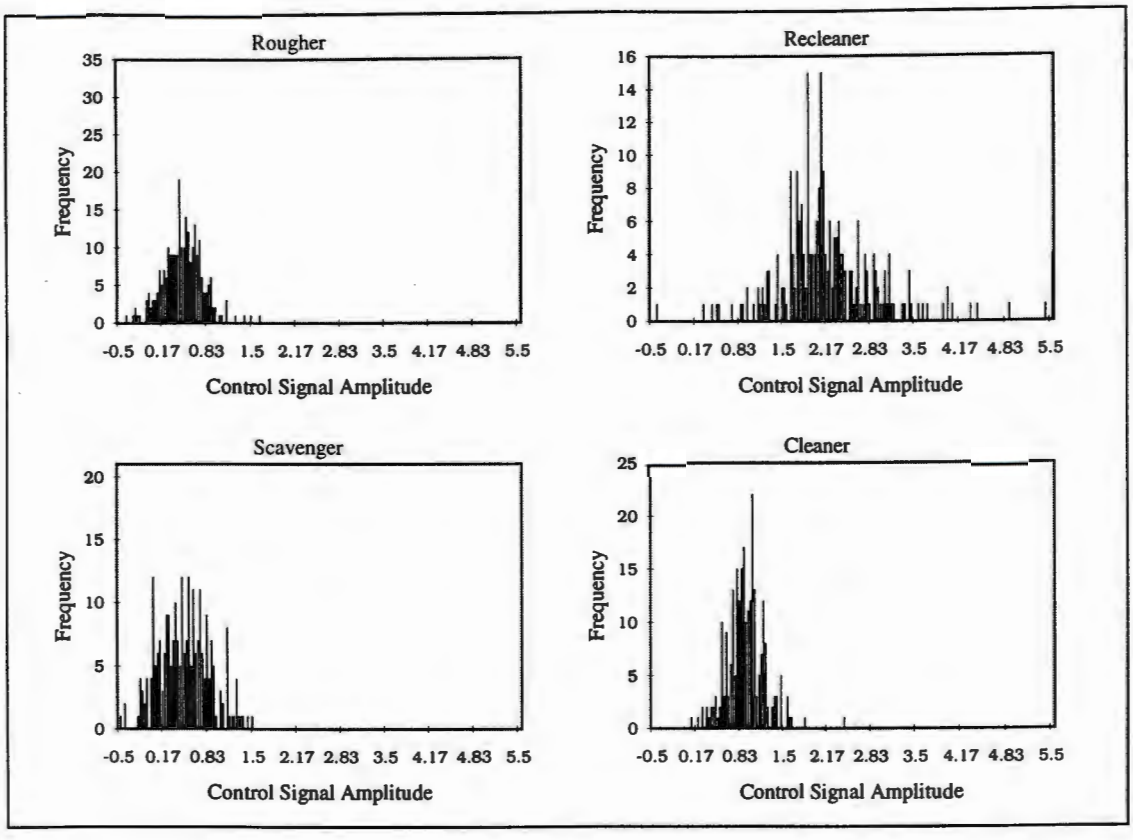

Figure 5.3 Control signal histograms for experiment 1: Recleaner stepped

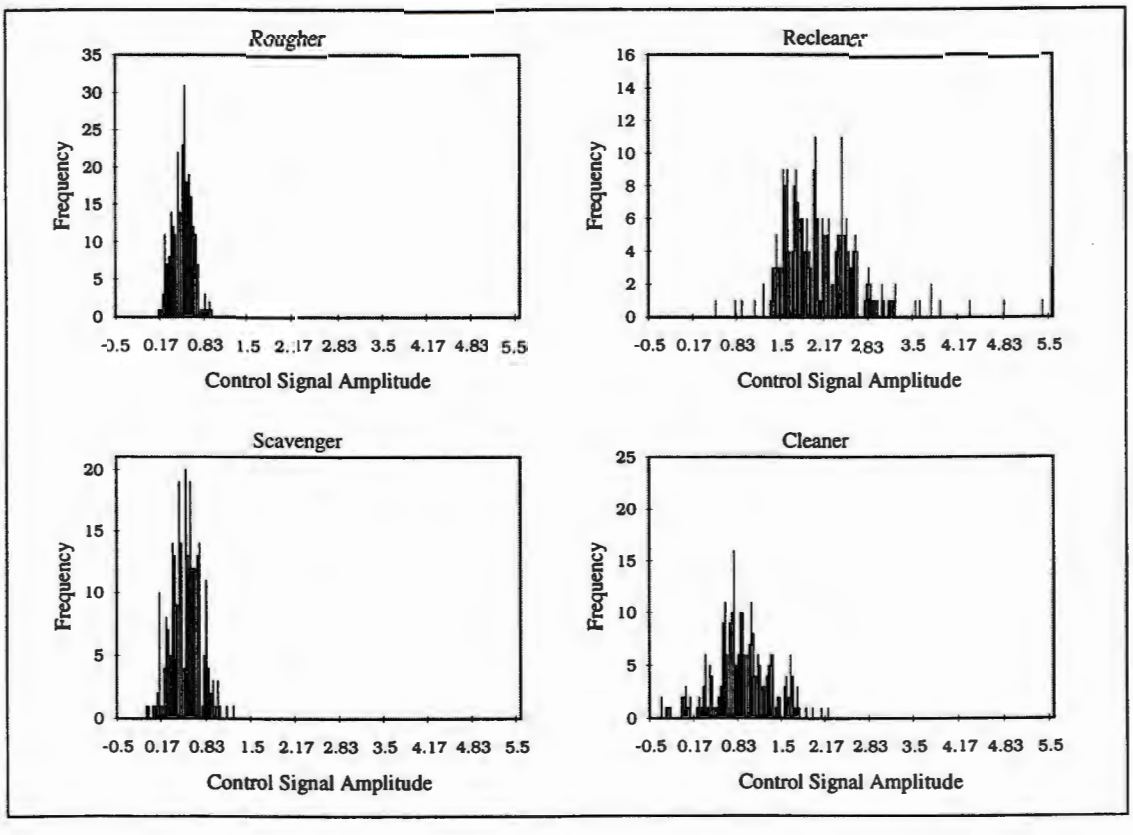

Figure 5.4 Control signal histograms for experiment 2: Recleaner stepped

Table 5.4 Mean and standard deviations for control signals

	Experiment 1		Experiment 2	
	Mean	Standard Deviation	Mean	Standard Deviation
Recleaner	2.217	0.845	2.136	0.710
Cleaner	0.911	0.289	0.880	0.453
Scavenger	0.481	0.378	0.507	0.222
Rougher	0.465	0.301	0.483	0.150

The above plots illustrate clearly the difference between the control signals. The increased control horizon has decreased the very active control signal of experiment 1. There has also been a decrease in the harshness[1] of the control signal, except for the cleaner response. This is evident when observing the histogram plots; the responses of experiment 2 have smaller standard deviation values indicating the smaller control increment amplitudes. The cleaner response shows an increased standard deviation value confirming the increase in control increment amplitudes. The harshness of the signal can be explained by the noisy level of the cleaner tank (caused by the splashing water). The more active control signal is able to control this level, whereas the less active signal resorts to large control increments for control. A problem rejecting the interaction disturbance between rougher and scavenger was evident for the rougher stepped in Fig 5.5.

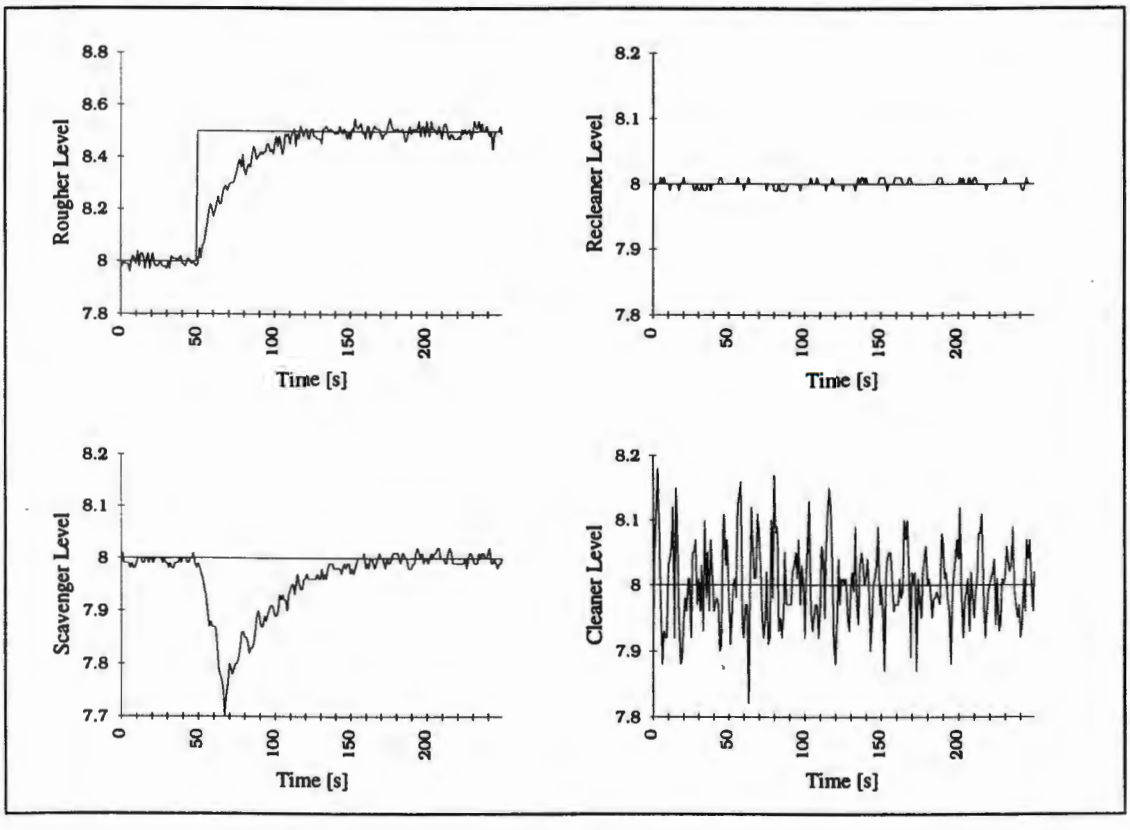

Figure 5.5 Experiment 2: Rougher stepped

[1]The harshness of a signal is defined as the high frequency amplitude of the signal

Experiment 3 demonstrates the effect of a very small damping factor, λ. From the theory it is evident that the smaller λ is, the less dampened the control signal is. Fig. 5.6 shows the control signal for the recleaner stepped. This result should be compared with that of Fig. 5.1 and Fig. 5.2 (note that the scales are the same even though the signals have larger values). The histograms for the relevant control signals is given in Fig 5.7. The mean and standard deviation values are displayed in Table 5.5.

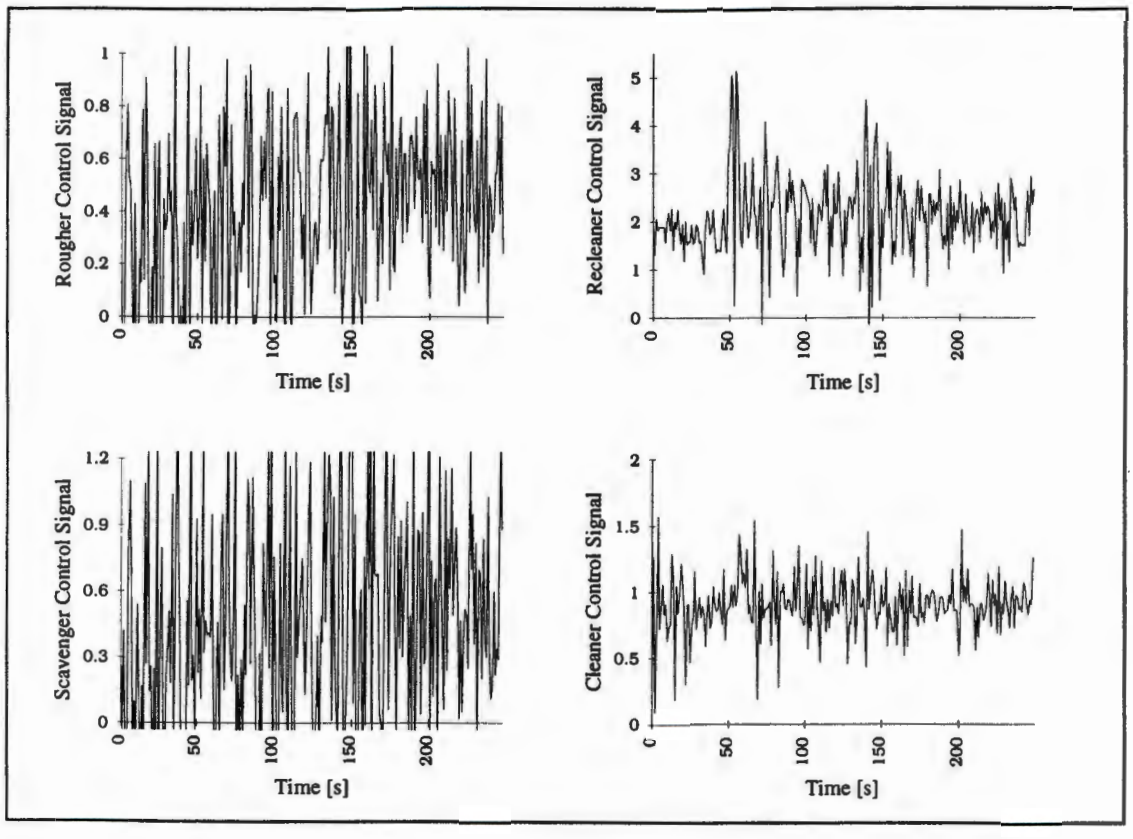

Figure 5.6 Control signal for experiment 3: Recleaner stepped

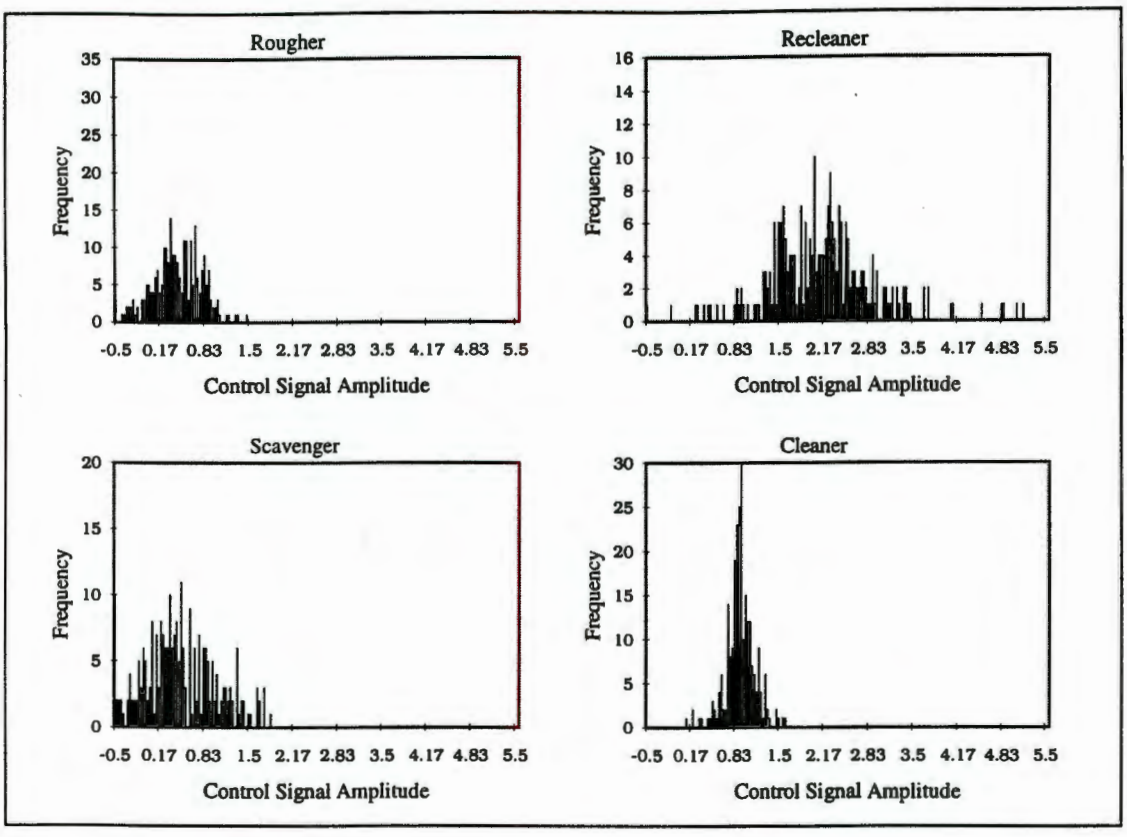

Figure 5.7 Control signal histograms for experiment 3: Recleaner stepped

Table 5.5 Mean and standard deviations for control signals

	Experiment 1		Experiment 3	
	Mean	Standard Deviation	Mean	Standard Deviation
Recleaner	2.217	0.845	2.124	0.815
Cleaner	0.911	0.289	0.903	0.216
Scavenger	0.481	0.378	0.460	0.539
Rougher	0.465	0.301	0.411	0.360

By comparing the histograms with that of experiment 1 (and from table 5.5) it is noted that the control increments have increased in amplitude for the rougher and scavenger responses i.e. the control signals are harsher. An interesting result is that for the more active tank level (output signal) of the cleaner a decrease in λ has led to smaller control increments, which is shown in the control signal histogram by the narrower amplitude range. Since decreasing λ causes large control increments (reflected by a more dynamic response) the ITAE values for experiment 3 are generally larger than the experiment 1 values indicating the more active outputs.

Experiment 4 shows the responses to a small prediction horizon, N_2. This set of tests was characterised by fast step responses but also by large overshoots and oscillations. The oscillations introduced can be seen clearly on the scavenger and recleaner level plots. The problem of rejecting disturbances and limiting interaction can be seen clearly in Fig. 5.8.

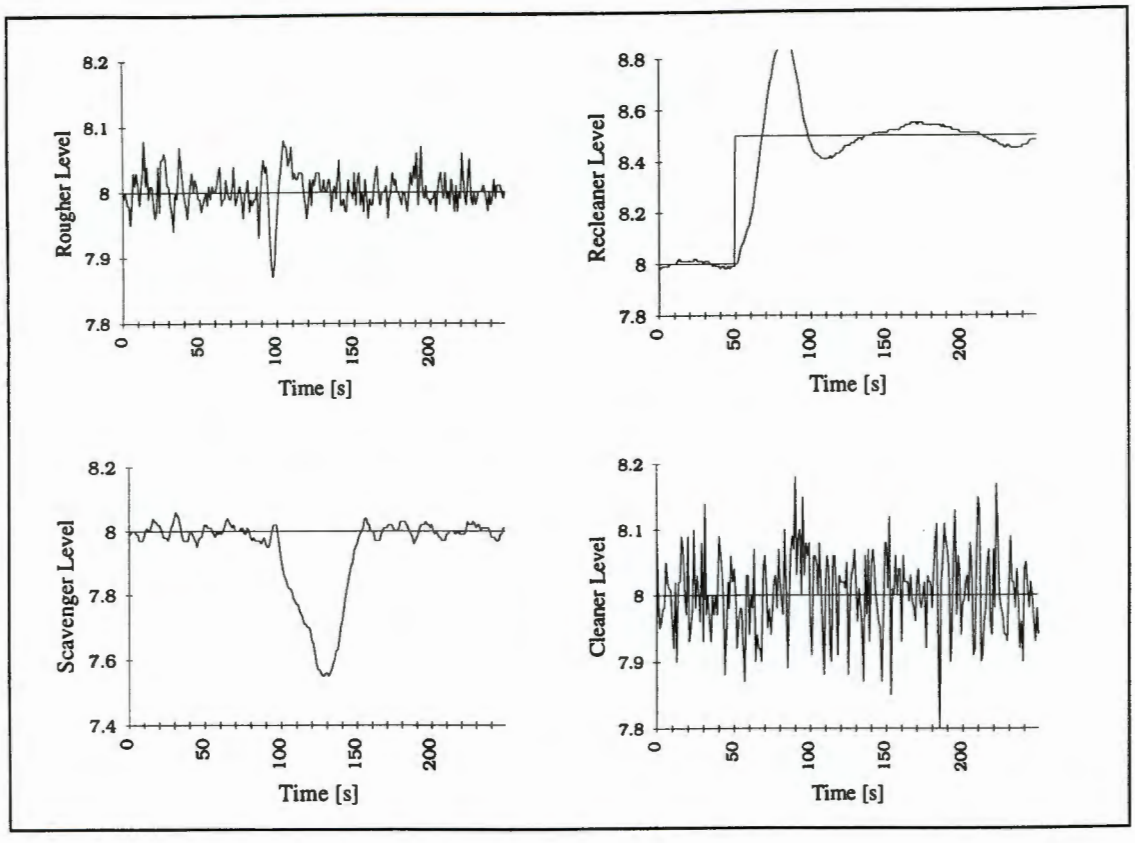

Figure 5.8 Experiment 4: Recleaner stepped

At ~90 [s] the rougher level experienced a large disturbance. Note the interaction effect on the scavenger level, in which it took more than 50 [s] to reject the interaction disturbance. Clearly the shortened prediction horizon is not able to provide a complete control solution. While it is able to provide a fast response, the interaction and disturbance rejection is limited and the steady state response is oscillatory.

Experiment 5 presents the effects of a large prediction horizon, N_2. The main feature of this set of data are sluggish responses. This is evident from the large ISE values presented. The interaction disturbance rejection is also hampered by slow responses as can be seen in Fig. 5.9 and Fig. 5.10 (page 53) on the rougher and scavenger levels. A major problem was revealed in Fig. 5.11 (page 54) when the scavenger failed to reach the setpoint on its step test. The scavenger level could also not be maintained when the rougher level was stepped (Fig. 5.12 - page 55). It can be concluded thus, that a very large prediction horizon rendered the system response sluggish and was not capable of rejecting large interaction disturbances satisfactorily.

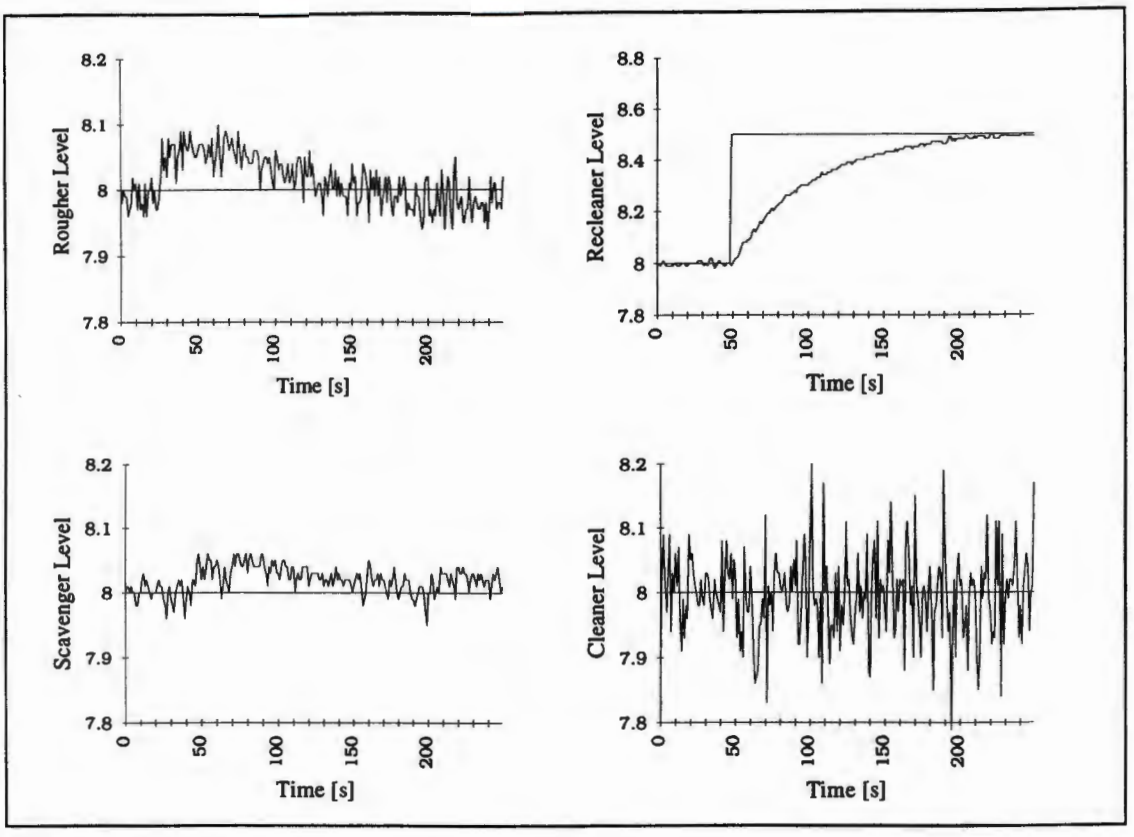

Figure 5.9 Experiment 5: Recleaner stepped

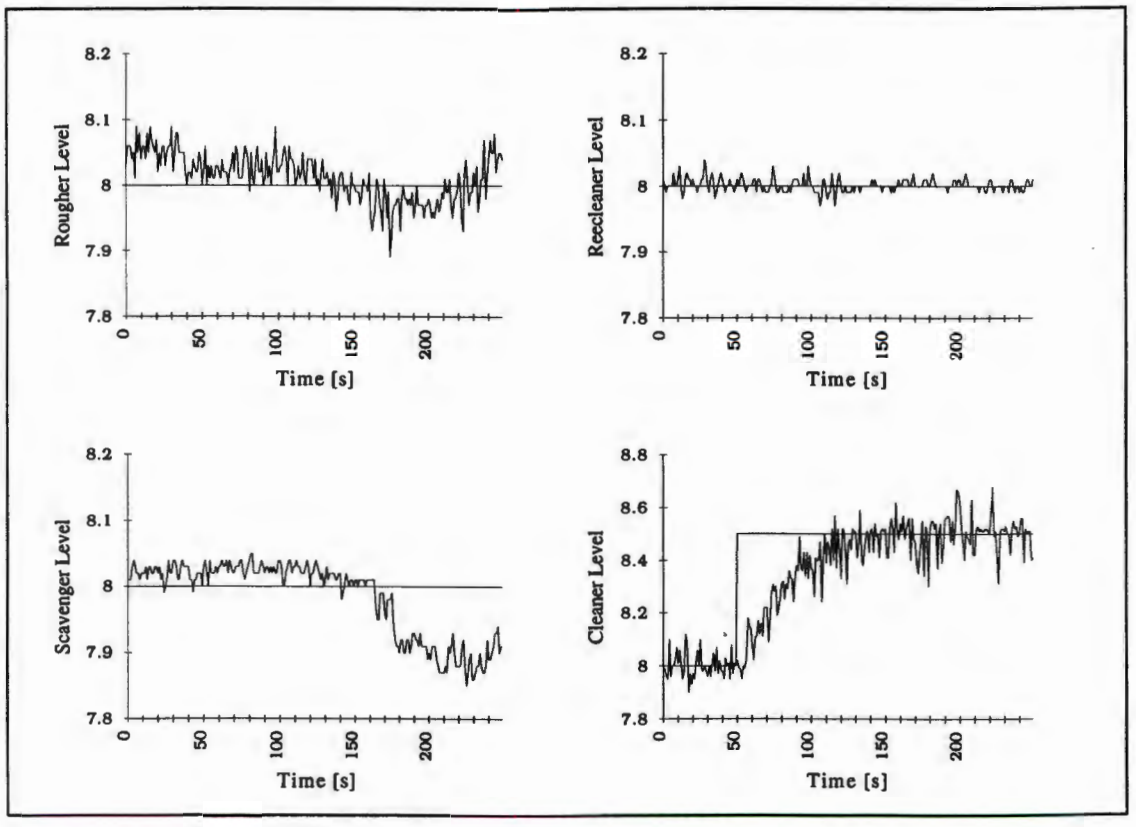

Figure 5.10 Experiment 5: Cleaner stepped

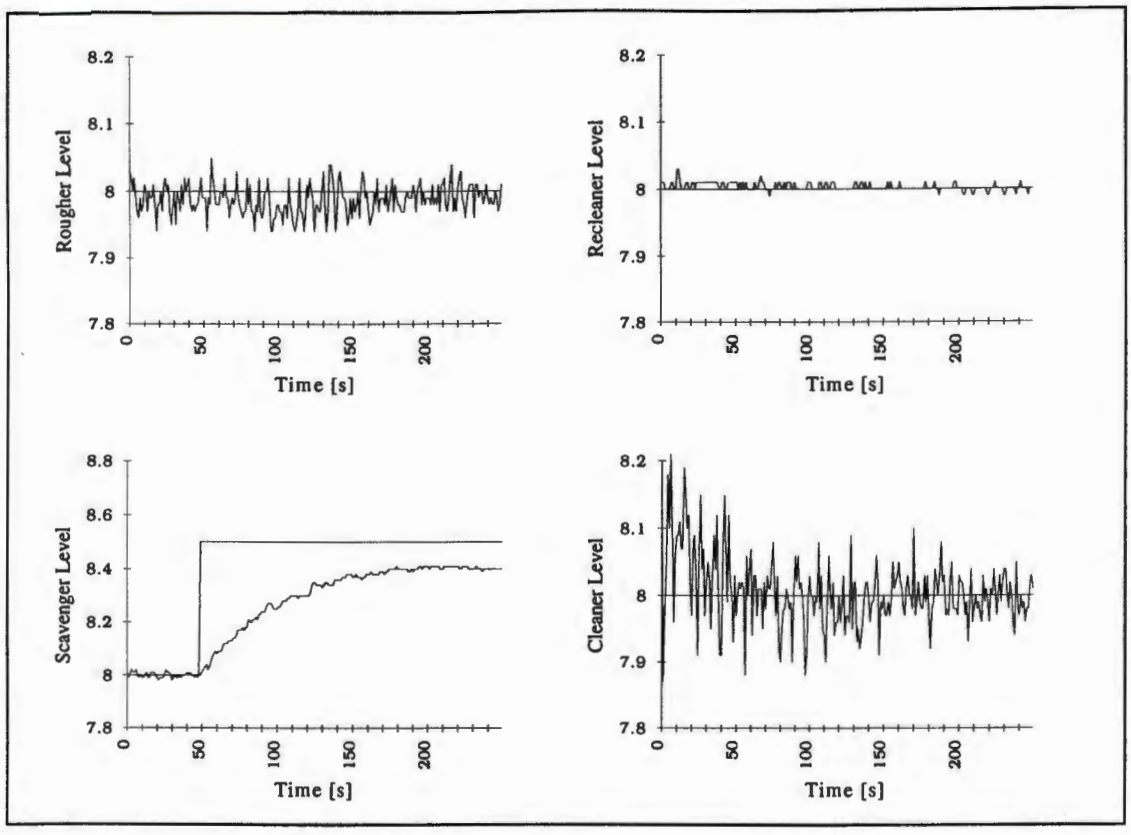

Figure 5.11 Experiment 5: Scavenger stepped

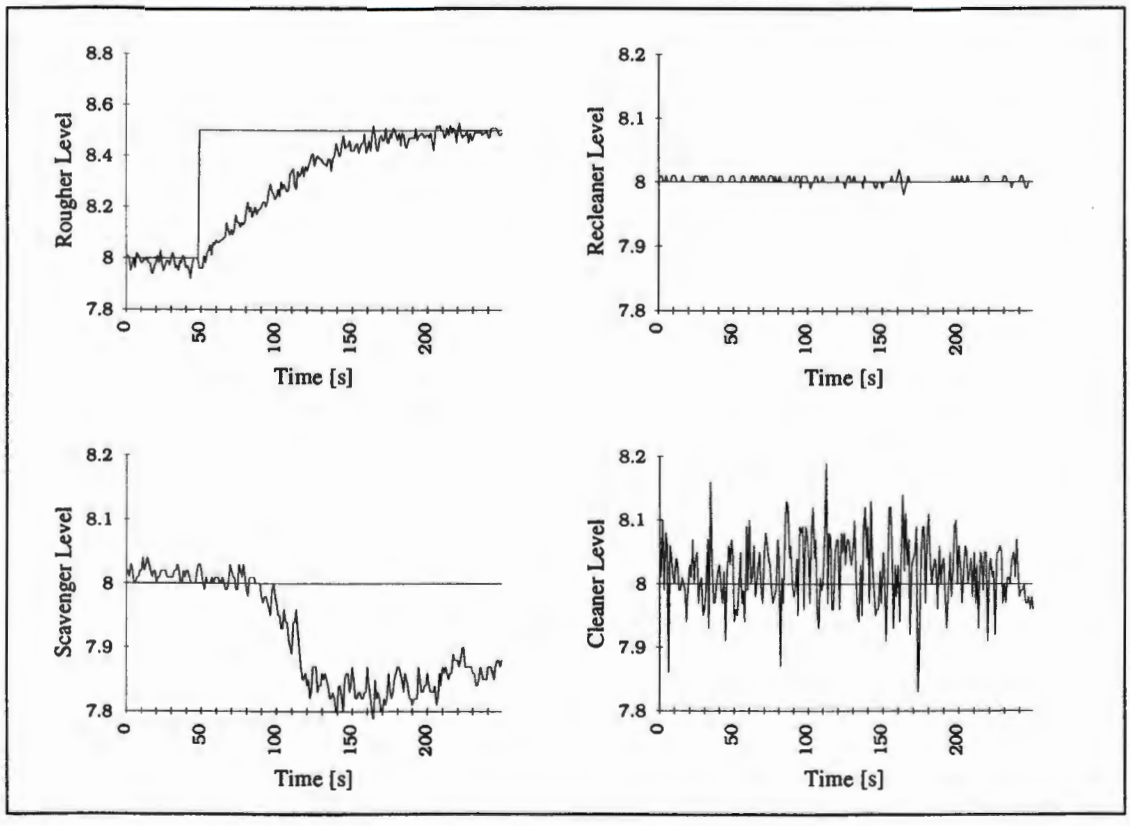

Figure 5.12 Experiment 5: Rougher stepped

Response to Signal Failure

In order to test the robustness of the implemented GPC strategy a signal failure experiment was executed. All signals to and from the flotation plant simulator were disconnected for approximately 15 seconds, after which the system was returned to normal. Fig 5.13 shows the results of this test.

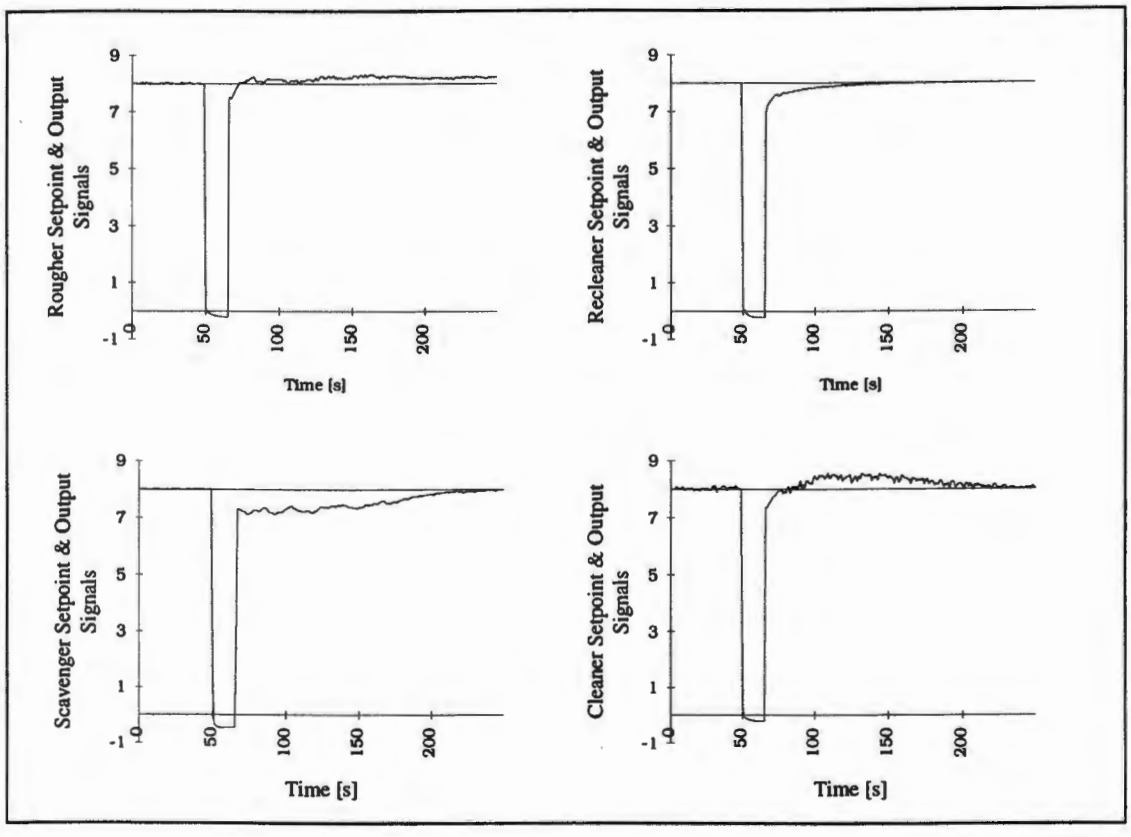

Figure 5.13 GPC response to signal failure

The GPC response to the signal failure test proved stable. Once the signal status was restored most of the plant outputs settled down to the setpoints even though this multivariable plant experienced the disturbance on all tank level signals at exactly the same time. The only problem in the results shown is the output offset of the rougher tank. Indeed, all the tanks, except the recleaner, had a slow response to the disturbance; this however can be attributed to the interaction experienced between tanks. The recleaner had a fast response since it essentially has very little interaction from the other control loops.

5.6. Long Range Predictive Control

The previous section detailed the results of the application a GPC strategy. This section demonstrates the results of using LRPC. As discussed in chapter 3, LRPC extends the GPC method by modifying the plant identification routine. The GPC parameters, therefore, remain the same and the effect of the long-range identification is observed for comparison.

The LRPC scheme was implemented in experiment 6. The same controller parameters for the GPC experiment 1 were applied as shown in Table 5.6. The resulting responses for the step tests are displayed in Appendix B, Fig. B.6a - B.6d.

Table 5.6 LRPC parameter settings for flotation plant experiments

	Experiment 6
N_1	1
N_2	50
Nu	1
λ	0.5

A summary of the ISE and ITAE values obtained for experiment 6 is given below in Table 5.7 and Table 5.8.

Table 5.7 Table summarising ISE values for experiment 6

Experiment 6				
	Tank Level Stepped			
ISE for Tank	Recleaner	Cleaner	Scavenger	Rougher
Recleaner	4.4942	0.0075	0.0064	0.0103
Cleaner	1.0211	4.2806	0.1801	0.8138
Scavenger	0.0321	0.1602	4.3527	0.1249
Rougher	0.1116	0.1062	0.0734	2.9544

Table 5.8 Table summarising ITAE values for experiment 6

Experiment 6				
	Tank Level Stepped			
ITAE for Tank	Recleaner	Cleaner	Scavenger	Rougher
Recleaner	1413.48	117.36	99.54	161.83
Cleaner	1503.09	1964.58	569.17	1317.04
Scavenger	280.15	631.20	1335.05	344.86
Rougher	488.83	578.03	410.19	1067.48

The results obtained are to be compared with those of experiment 1. One striking feature of the LRPC results is the faster responses obtained when the tanks were stepped (except for the scavenger, which was effectively as fast), which is noted from the ISE values. At the same time, the tanks that were not stepped appeared to have more active responses than those of GPC. This can be attributed to the fact that the interaction has increased because of the faster responses, clearly shown by the rougher step. The LRPC rougher response was considerably faster than the GPC response, but the interaction experienced by the scavenger was significant, taking approximately 50 seconds (about one time constant) to settle down again. Increased interaction is also seen in the cleaner level.

So while faster step responses were obtained, there was an increase in the ITAE values for all other responses, indicating more active responses. Taking a closer look at a sample of control signals in Fig. 5.14, should give an idea as to the type of responses experienced. The control signal histograms for comparison for those of experiment 1 are shown in Fig. 5.15 with the mean and standard deviation values shown in Table 5.9.

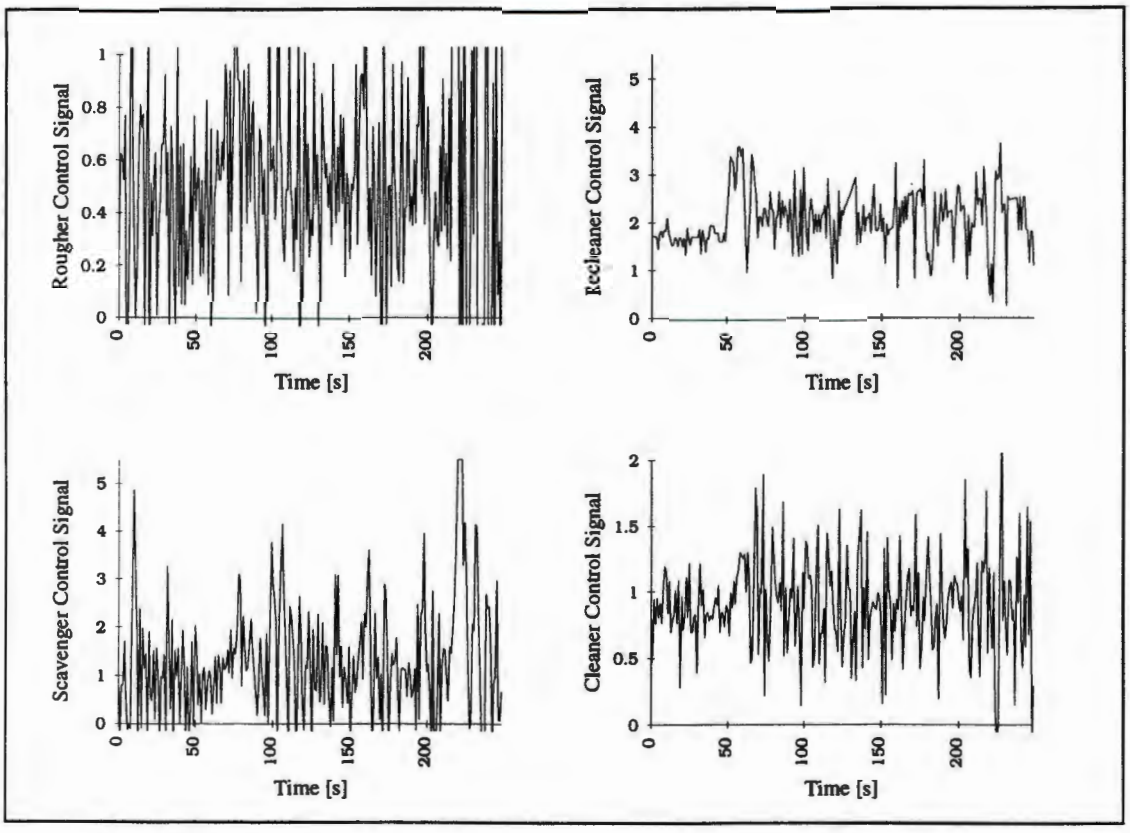

Figure 5.14 Control signal for experiment 6: Recleaner stepped

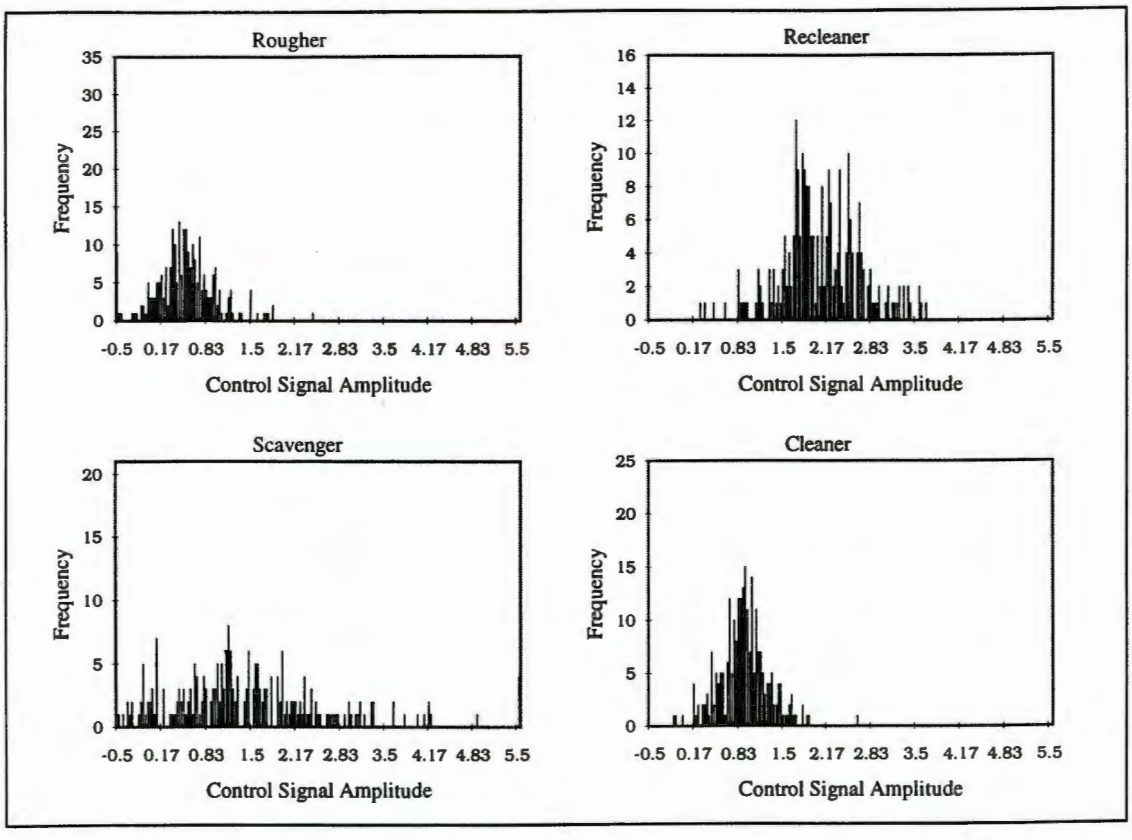

Figure 5.15 Control signal histograms for experiment 6: Recleaner stepped

Table 5.9 Mean and standard deviations for control signals

	Experiment 1		Experiment 6	
	Mean	Standard Deviation	Mean	Standard Deviation
Recleaner	2.217	0.845	2.065	0.600
Cleaner	0.911	0.289	0.906	0.369
Scavenger	0.481	0.378	1.321	1.210
Rougher	0.465	0.301	0.513	0.457

Comparing the control signal of the LRPC experiment with that of the GPC in Figure 5.1 the observations made from the output graphs can be confirmed. By comparison of the standard deviation values for the recleaner stepped it is clear that the LRPC recleaner control signal is less harsh than corresponding GPC signal. All the other control signals are more active and have harsher control signals (indicated by the increase in standard deviation of the control signals), explaining the more active output signals. Thus, LRPC has seemed to introduce a more harsh approach to control.

Response to Signal Failure

In order to test the robustness of the LRPC strategy implemented a signal failure experiment was executed. All signals to and from the flotation plant simulator were disconnected for approximately 15 seconds, after which the system was returned to normal. Fig 5.16 shows the results of this test.

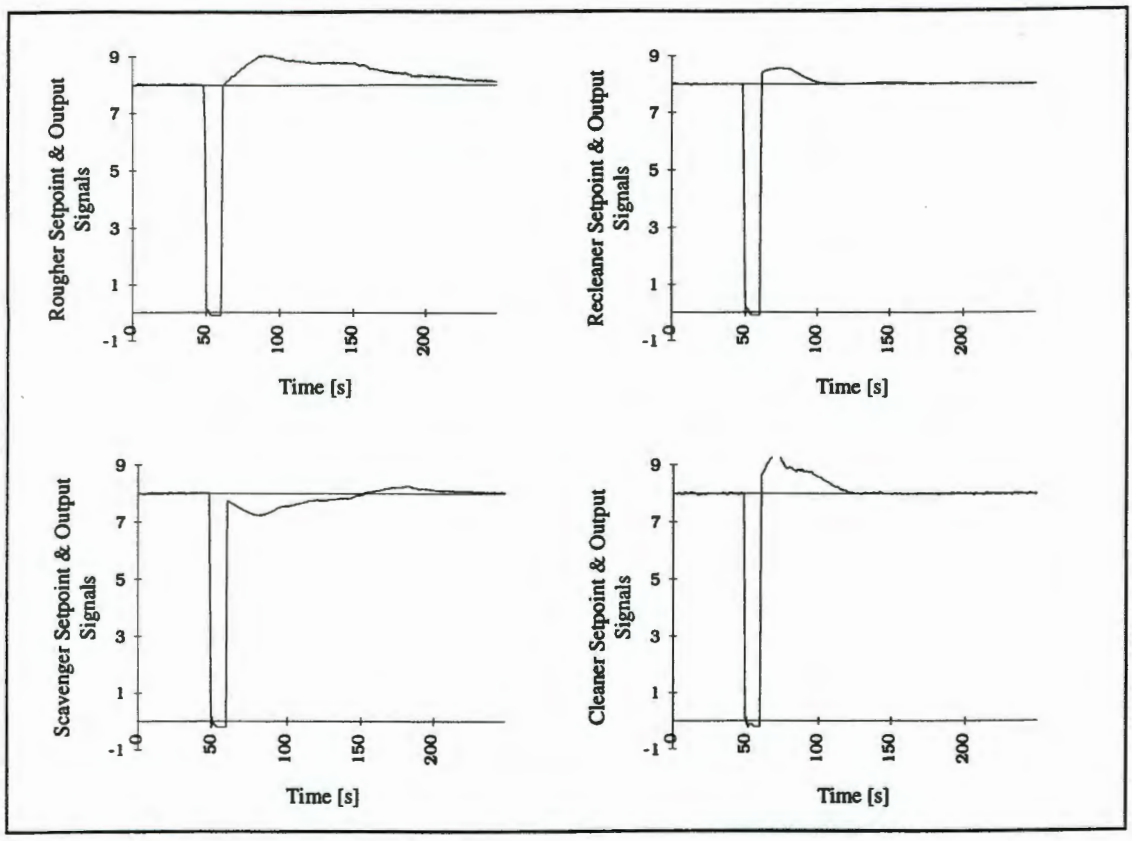

Figure 5.16 LRPC response to signal failure

The LRPC response to the signal failure test proved stable. The responses of the recleaner and cleaner tanks were fast, settling in a shorter time than the GPC responses of Fig. 5.13. The LRPC response for the rougher was very slow and had a large initial deviation from the setpoint. Unlike the GPC response, however, it eventually settles down to the setpoint. The scavenger response, although experiencing the interaction disturbance from the rougher, settled down faster than the GPC response for the scavenger.

The LRPC response to signal loss was not significantly 'better' than that of GPC providing no obvious advantage, but it is important to note that both responses were stable.

5.7. Estimated Parameters

Plots of the estimated parameters were obtained to confirm that the adaptive approaches of GPC and LRPC are adapting to the model changes. The plant was started from the same position for both GPC and LRPC. Figure 5.17 and Figure 5.18 shows the variation in parameters of the estimated plant models which have the form,

$$\frac{Water\ Level\,[\mathrm{V}]}{Valve\ Position\,[\mathrm{V}]} = \frac{b_0 + b_1 q^{-1}}{1 + a_1 q^{-1}}$$

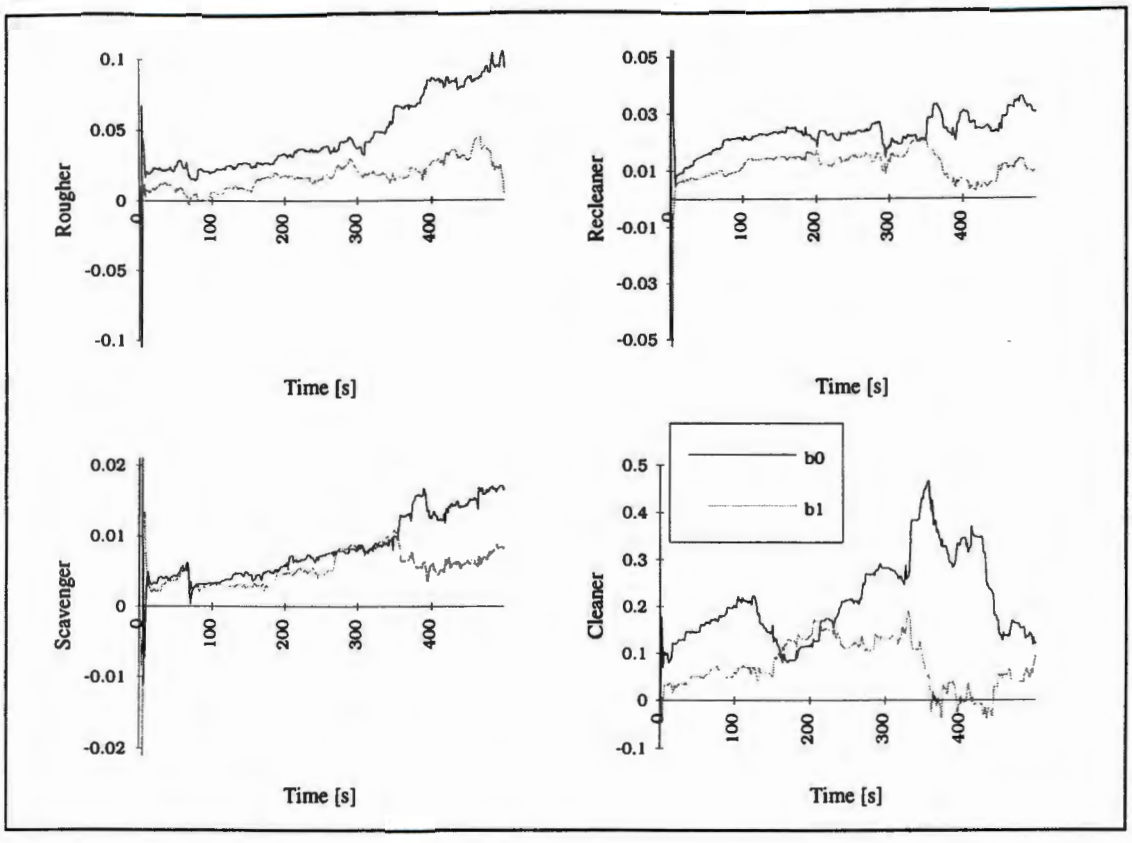

Figure 5.17 Parameter variation for GPC: b_0 & b_1

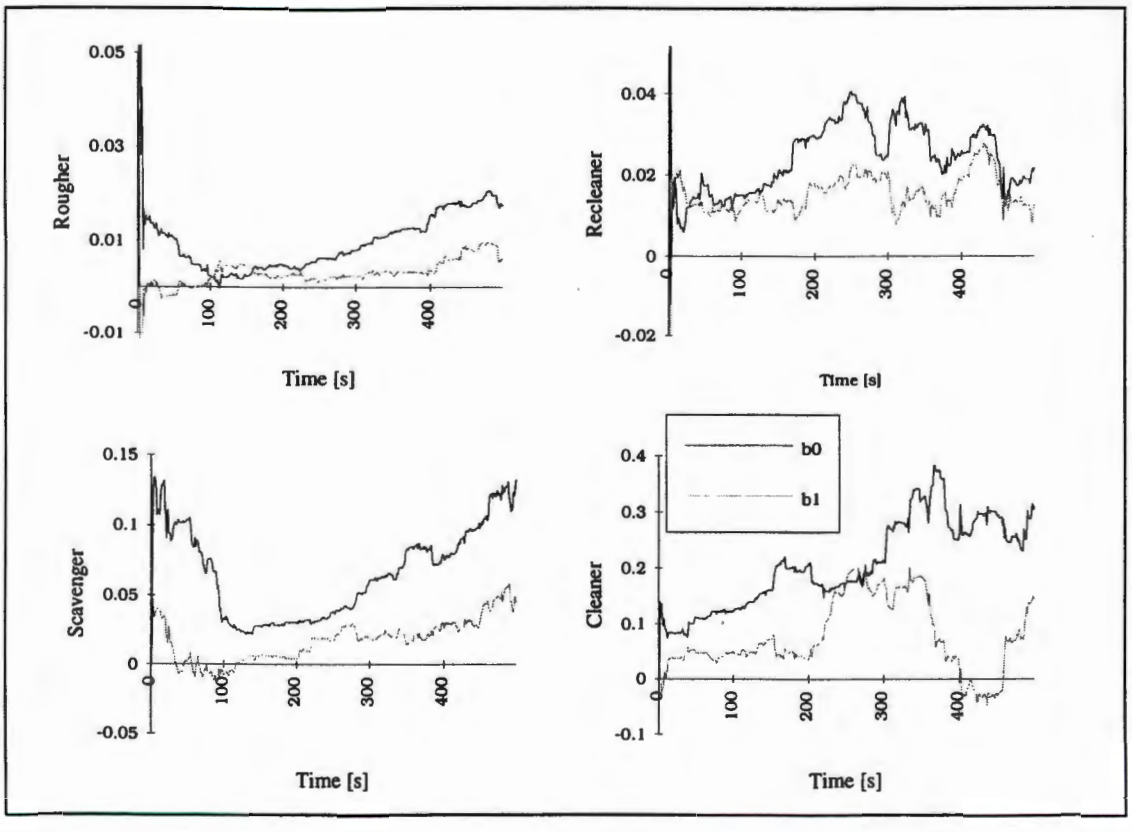

Figure 5.18 Parameter variation for LRPC: b_0 & b_1

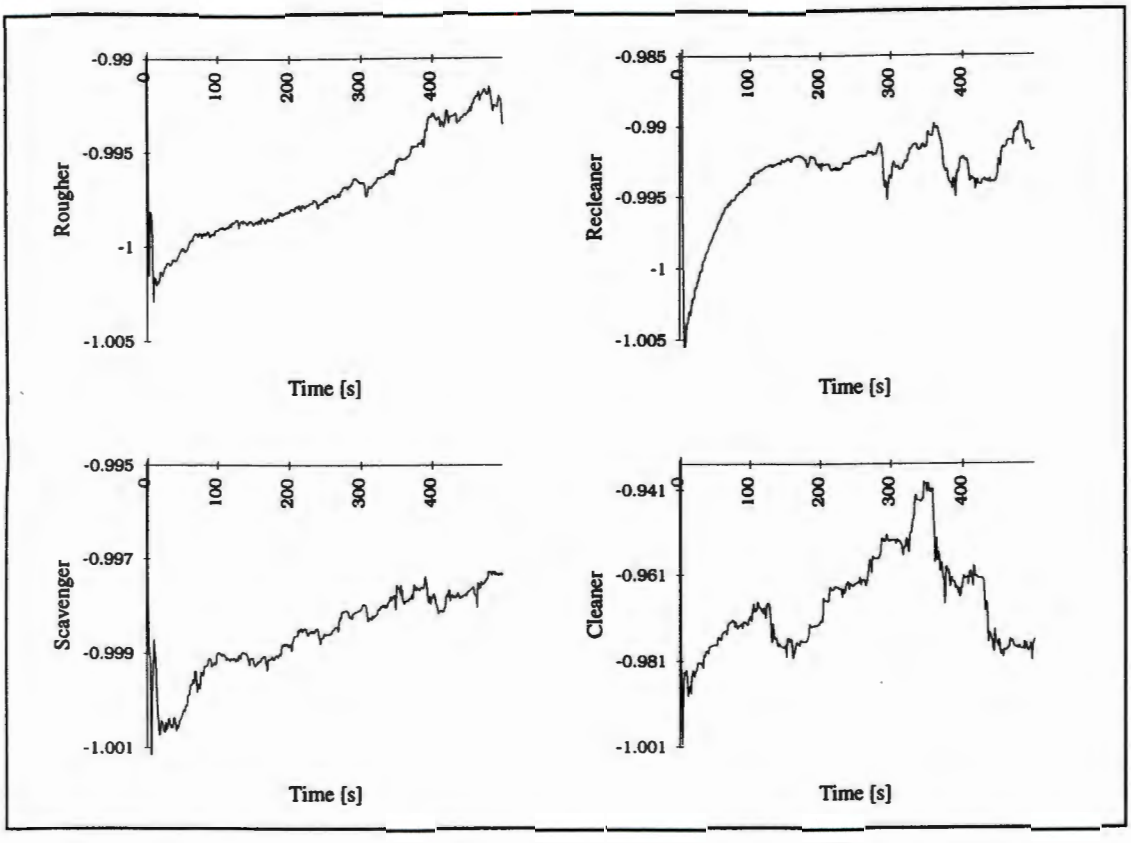

Figure 5.19 Parameter variation for GPC: a_1

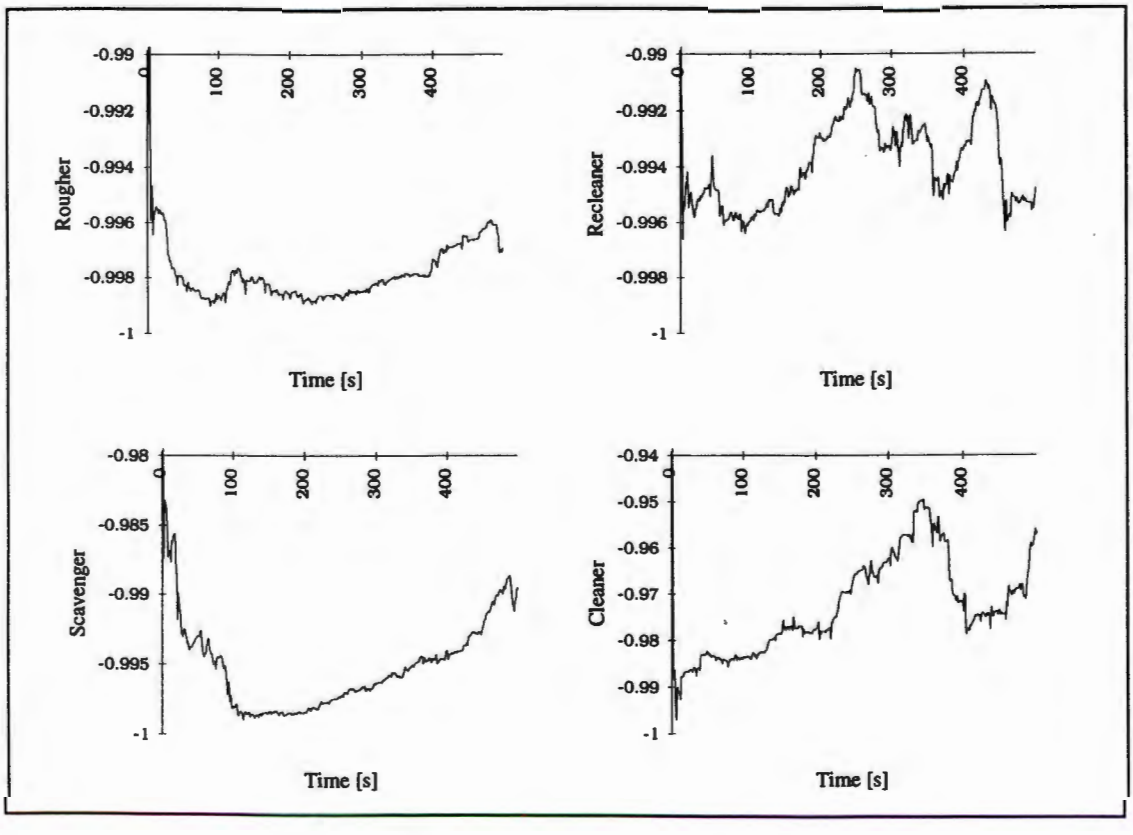

Figure 5.20 Parameter variation for LRPC: a_1

It is clear that both GPC and LRPC continually update the plant models. While not converging to constant values because of the interaction as defined in equation (5.1), it is evident that the models obtained from GPC and LRPC are essentially different.

5.8. Summary

The GPC and LRPC schemes have been applied successfully to a practical problem. The adaptive approach enabled a diagonal controller to be used without having to extend the GPC and LRPC schemes specifically for the multivariable plant. Recall that a single variable GPC/LRPC approach was taken so that the interaction itself is not addressed, rather the model changes have been taken into account. The tuning of the GPC parameters has been demonstrated and confirms the results obtained by the simulations performed in chapter 2 for the simple step responses, but allowances had to be made for the interaction in the plant. The LRPC results were satisfactory, proving faster on step responses, although not 'outperforming' by a large degree (confirming the simulation results of chapter 3).

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1. Introduction

The Generalised Predictive and Long-Range Predictive Control schemes have been studied and applied successfully in simulation and to a practical problem. A detailed study of creating a robust control system has also been investigated, with attention paid to the hardware and software components as separate and equally important entities. Based on this work completed in the thesis the following conclusions can be drawn and recommendations made.

6.2. Generalised Predictive Control

Generalised predictive control has been used on many industrial problems as a viable solution and is shown in this thesis to work well on a flotation plant simulator.

An advantage of GPC is that the control scheme is based on the CARIMA plant model. This model includes the noise model in the plant definition so that the response is adapted for particular situations. This is demonstrated by the fact that the GPC control law, based on the CARIMA model. is forced to be able to reject step output disturbances (i.e. it has inherent integral action) which is a common disturbance encountered in industry.

Another advantage of the GPC system is the flexibility derived from the tuning parameters provided, allowing a wide range of control problems to be solved. These parameters are easy to use and simplify the tuning procedure. Particular choices for the GPC parameters provided stable control of the multivariable system of the flotation plant. By tuning the parameters, various predictable responses were obtained e.g. by increasing the control horizon faster responses to step changes were obtained. It is necessary to state that there are guidelines to the choices for the tuning parameters so that typical responses can be obtained. The adaptive approach of GPC was able to cope with fairly large changes in plant model on-line and did not become unstable after transitions.

For the flotation plant simulator studied, interaction played a large role, and optimising step responses of one output would affect the control of the other outputs. The interaction experienced was not addressed directly since the GPC/LRPC controllers were implemented in a diagonal controller matrix. While not optimal a stable solution was attainable and demonstrated.

An area of GPC that needs to be investigated and applied more thoroughly is that of robustness. As outlined in section 2.8 the basic GPC method can be enhanced so that robustness is guaranteed, by the extension of an observer polynomial to higher orders.

While the importance of the observer polynomial has been recognized and demonstrated [1,10] only recent work (~1995) [9] has seen the introduction of observer design methods other than the previous intuitive approaches.

6.3. Long-Range Predictive Control

The need for a proper long-range predictive control method lead to the introduction of a modified identification scheme. The recursive least squares algorithm, employed as parameter estimator in the original GPC scheme, is only a single-step-ahead predictor. The long-range predictive identification strategy extends the GPC method by making the parameter estimator a dual of the control law as far as using its predictions and prediction horizon.

The parameter estimation scheme, therefore, has more filtering associated with it than the control scheme. The filter formulated by LRPI, based on a sound theoretical understanding of GPC, replaces previous *ad hoc* filtering methods suggested. This filter is now dependent on the GPC parameters used.

The LRPC responses, using LRPI, exhibited a slight improvement in the minimisation of the output variance than GPC with RLS. The LRPC responses also showed improvement to model changes showing that the LRPI filter has dealt better with the model changes than RLS. The LRPC scheme increased the activity of the control signal.

As for GPC, an area that needs closer study is that of robustness analysis. The influence of the observer polynomial on LRPC would require the extension of the polynomial to higher orders. The introduction of the LRPI filter has essentially allowed the observer polynomial to deal with disturbances.

6.4. The Simulator/Control Program

For the design of the overall control system, emphasis was placed on obtaining a robust system, since it was concluded that ordinary personal computers were not suited to industrial environments. The VMEbus computer system (OR Industrial Computers) decided upon has the specifications necessary for industrial applications. While it is evident that the hardware needs to be rugged it should not be forgotten that the software i.e. operating system and programming language, also need to provide a level of robustness. Windows NT by Microsoft Corporation has proved to be a suitable operating system for the control application studied.

From the work done it is clear that the design of the control system is essentially the selection of a number of components to be used as a whole. A significant conclusion which arises from this point and the thesis is the integratibility of the various components. Considerable time and finances could be spent on overcoming problems encountered interfacing different parts of the system and sufficient care must be taken in selecting compatible components.

The final simulator/control program was written in Visual C++ specifically for the Windows NT operating system. The GUI of the program makes for an attractive and visually useful interface from the plant to the user. The presentation of plant data to the operator is an important one, and with the introduction of more computer controlled systems, the graphical environment remains the clearest way to represent data.

While considerable study can be devoted to the field of ergonomics it was not the aim of this thesis to provide a complete control package. A useful, but more importantly, re-usable program has been provided in order to interface a control algorithm through the VMEbus to a plant. The program can easily be adapted and utilised as the basis for any control system.

REFERENCES

[1] D. W. Clarke & C. Mohtadi
'Properties of Generalised Predictive Control'
Automatica, vol. 25, no. 6, pp. 859-875, 1989

[2] M. Mahouf, D. A. Linkens, A. J. Ashbury, W. M. Gray & J. E. Peacock
'Generalised Predictive Control (GPC) in the Operating Theatre'
Proceedings of the IEE - Part D, vol. 139, no. 4, July 1992

[3] I. P. Fisher
'Multivariable Control of a Flotation Plant'
MSc Thesis, University of Cape Town, 1992

[4] J. E. Cohen
'The PC in Industrial Systems'
Electricity and Control, March, pp 35-41, 1993

[5] D. W. Clarke, C. Mohtadi & P. S. Tuffs
'Generalised Predictive Control: Parts 1 & 2'
Automatica, vol. 23, pp. 137-160, 1987

[6] K. J. Astrom & B. Wittenmark
'On Self-Tuning Regulators'
Automatica, vol. 9, pp. 185-199, 1973

[7] D. W. Clarke & P. J. Gawthrop
'Self-Tuning Control'
Proceedings of the IEE, vol. 123, pp. 633-640, 1979

[8] C. R. Cutler & B. L. Ramaker
'Dynamic Matrix Control: a Computer Control Algorithm'
Joint American Control Conference, San Francisco, 1980

[9] T-W. Yoon & D. W. Clarke
'Observer Design in Receding-horizon Predictive Control'
International Journal of Control, vol. 61, no. 1, pp. 171-191, 1995

[10] B. D. Robinson & D. W. Clarke
'Robustness Effects of a Prefilter in Generalised Predictive Control'
Dept. of Engineering Science, Oxford, October 27 1989

[11] D. S. Shook, C. Mohtadi, S.L. Shah
'Identification for Long-Range Predictive Control'
Proceedings of the IEE - Part D, vol. 138, no. 1, January 1991

[12] W. Lu & D. G. Fisher
'Nominal Predictive Control'
Chemical Engineering Science, vol. 47, no. 4, pp. 809-820, 1992

[13] D. S. Shook, C. Mohtadi & S. L. Shah
'A Control-Relevant Identification Strategy for GPC'
IEE Transactions on Automatic Control, vol. 37, no. 7, July 1992

[14] J. Bohm, A. Halouskova, M. Kary & V. Peterka
'Simple LQ Self-Tuning Regulators'
Proceedings of the 9th IFAC World Congress, Budapest, Hungary, 1984

[15] Microsoft Corporation
'Real-Time Systems and Micrsoft Windows NT'
Microsoft Development Library, June 29 1995

[16] J. Richter
'Advanced Windows : The developers guide to the WIN32 API for Windows NT 3.5 and Windows 95'
Microsoft Press, 1995

[17] M. Braae
'Notes on Multivariable Control Systems'
Department of Electrical Engineering
University of Cape Town, 1994

[18] J.O. O' Reilly & W. E. Leithead
'Multivariable Control by Individual Channel Design'
International Journal of Control, vol. 54, no. 1, pp. 41-46, 1991

[19] Chester L. Nachtigal (Editor)
'Instrumentation and Control: Fundamentals and applications'
John Wiley & Sons, Inc, 1990

[20] B. E. Ydstie
'Extended Horizon Adaptive Control'
Proceedings of the 9th IFAC World Congress, Budapest, Hungary, 1984

[21] R. Rouhani & R. K. Mehra
'Model Algorithmic Control: Basic theoretic perspectives'
Automatica, vol. 18, pp 401-414, 1972

[22] R. M. C. de Keyser & A. R. van Cauwenberghe
'Extended Prediction Self-Adaptive Control'
Proceedings of the 7th Symposium on Identification and System Parameter Estimation, York, UK, 1985

[23] A. J. Lynch, N. W. Johnson, E. V. Manlapig & C. G. Thorpe
'Mineral and Coal Flotation Circuits: Their simulation and control'
Elsevier Scientific Publishing Company, 1981

[24] I. P. Fisher
'Multivariable Control of a Flotation Plant Simulator'
MSc Thesis
University of Cape Town, 1988

[25] M. Stevens
'A Real Industrial PC'
Elektron, October, pp 41-42, 1993

[26] J. W. Atwood
'The Systems Analyst: How to design computer-based systems'
Hayden Book Company, Inc. 1977

[27] K. J. Astrom & B. Wittenmark
'Adaptive Control'
Addison and Wesley, Reading, 1989

APPENDIX A

Solving for the $E(q^{-1})$ and $F(q^{-1})$ polynomials

Consider the Diophantine equations (see chapter 2)

$$1 = E_j A\Delta + q^{-j} F_j \tag{A.1}$$

$$1 = E_{j+1} A\Delta + q^{-(j+1)} F_{j+1} \tag{A.2}$$

where $\Delta = 1 - q^{-1}$

Subtracting (A.1) from (A.2) produces,

$$0 = A\Delta(E_{j+1} - E_j) + q^{-j}(q^{-1}F_{j+1} - F_j) \tag{A.3}$$

Since $(E_{j+1} - E_j)$ has degree j,

$$E_{j+1} - E_j = \tilde{E}_{j+1} + e_j q^{-j} \tag{A.4}$$

Returning to (A.3)

$$0 = A\Delta(\tilde{E}_{j+1} + e_j q^{-j}) + q^{-j}(q^{-1}F_{j+1} - F_j)$$

$$0 = A\Delta\tilde{E}_{j+1} + q^{-j}(q^{-1}F_{j+1} - F_j + A\Delta e_j) \tag{A.5}$$

From (A.5) it can be deduced that

$$\tilde{E}_{j+1} = 0 \qquad \& \qquad F_{j+1} = q(F_j - A\Delta e_j)$$

Enough information is now available to solve for the polynomial coefficients by forming iteration equations. From (A.4)

$$E_{j+1} = E_j + e_j q^{-j}$$

And since $A\Delta$ is monic[1]

$$e_j = f_{(j)0}$$

$$f_{(j+1)i} = f_{(j)i+1} - A\Delta e_j$$

[1]The leading element of the polynomial is equal to one

All that is now required is to initialise the polynomials so that the iterations have a starting point. Consider the Diophantine equation of (A.1) with $j=1$,

$$1 = E_1 A\Delta + q^{-1}F_1$$

Since $A\Delta$ is monic

$$E_1 = 1$$
$$F_1 = q(1 - A\Delta)$$

APPENDIX B

STEP RESPONSE DATA FOR FLOTATION PLANT APPLICATION

Appendix B contains the plots and performance index tables that were obtained from the application of GPC and LRPC to the flotation plant simulator (see chapter 5). The information is divided into six experiments such that each experiment consists of four sets of plots and tables e.g. experiment 1 is contained on pages B-2 to B-5 and consists of figures and tables B.1a, B.1b, B.1c, B.1d. Experiments 1 to 5 show GPC responses with parameter variations and experiment 6 shows the LRPC responses. Each page shows the plots and performance indexes values of the flotation plant response when one tank level is stepped.

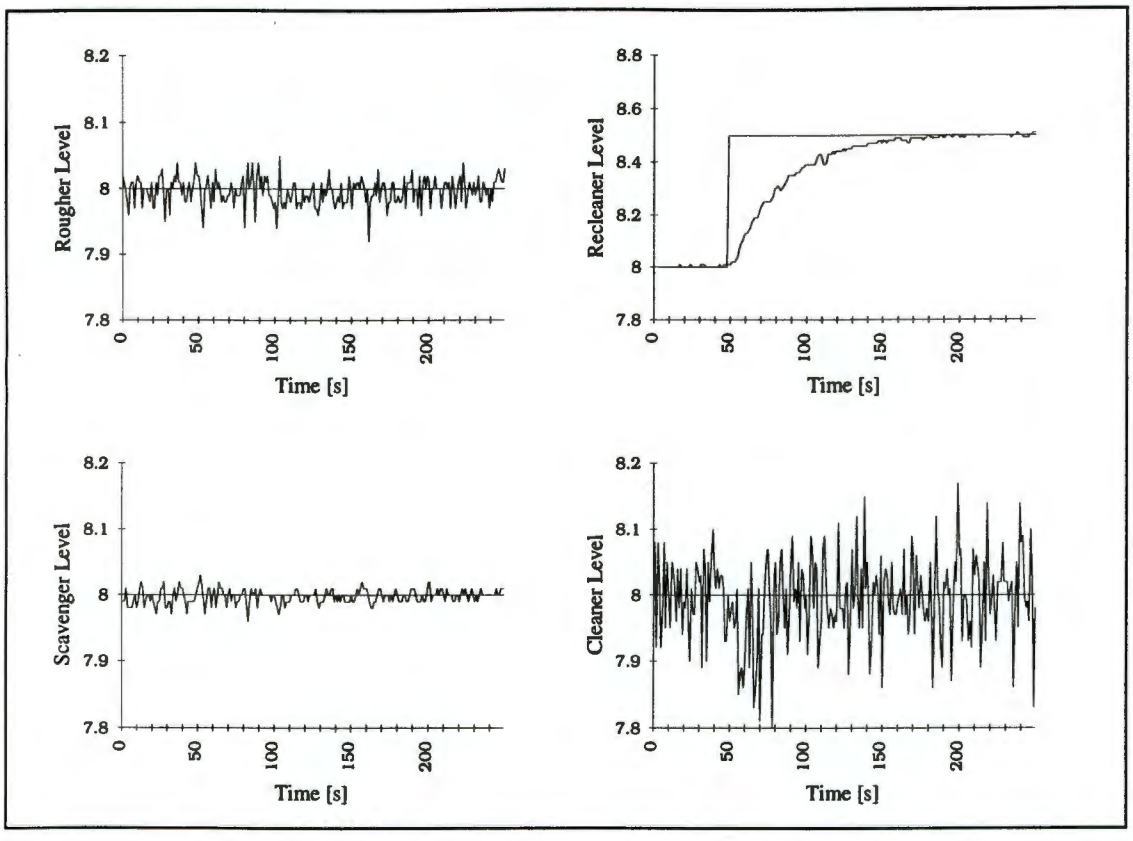

Figure B.1a Experiment 1: Recleaner stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$

Table B.1a Experiment 1: ISE and ITAE values for recleaner stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	4.7355	1426.70
Cleaner	0.9964	1484.83
Scavenger	0.0306	201.98
Rougher	0.1050	478.81

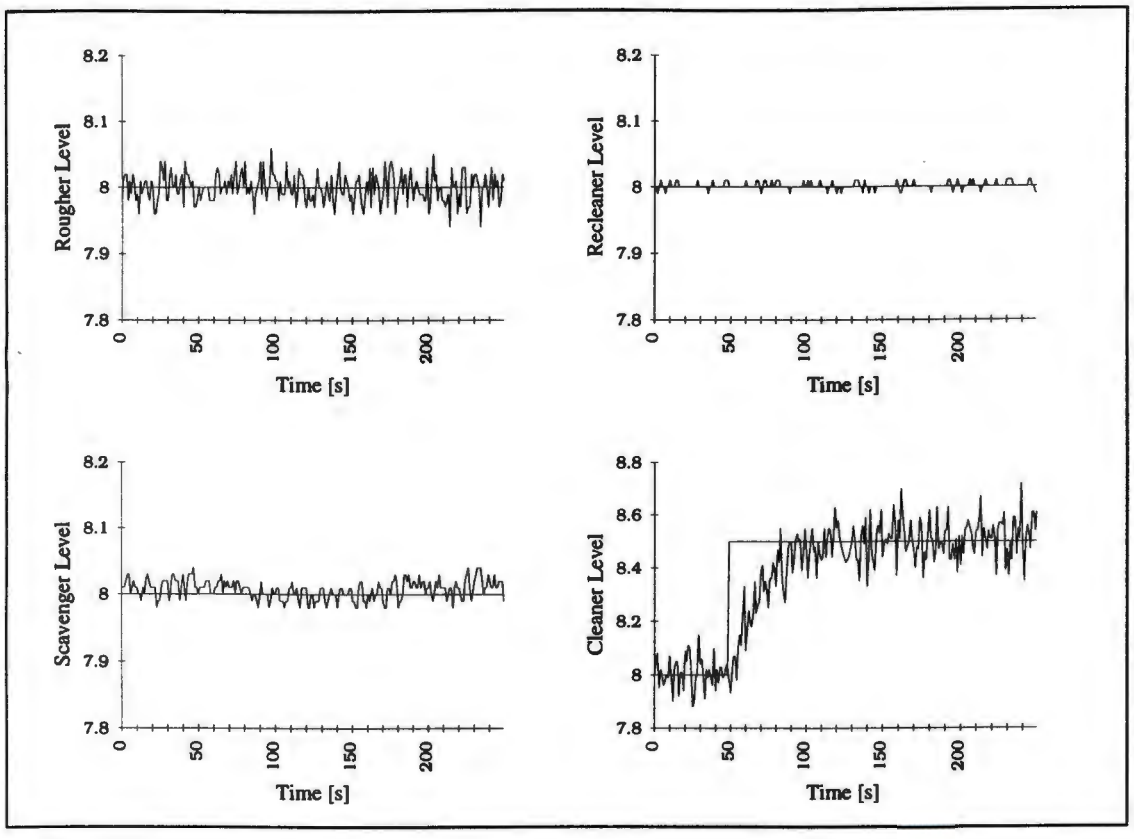

Figure B.1b Experiment 1: Cleaner stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$

Table B.1b Experiment 1: ISE and ITAE values for Cleaner stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	0.0063	79.53
Cleaner	4.2486	2240.66
Scavenger	0.0622	407.35
Rougher	0.1165	559.18

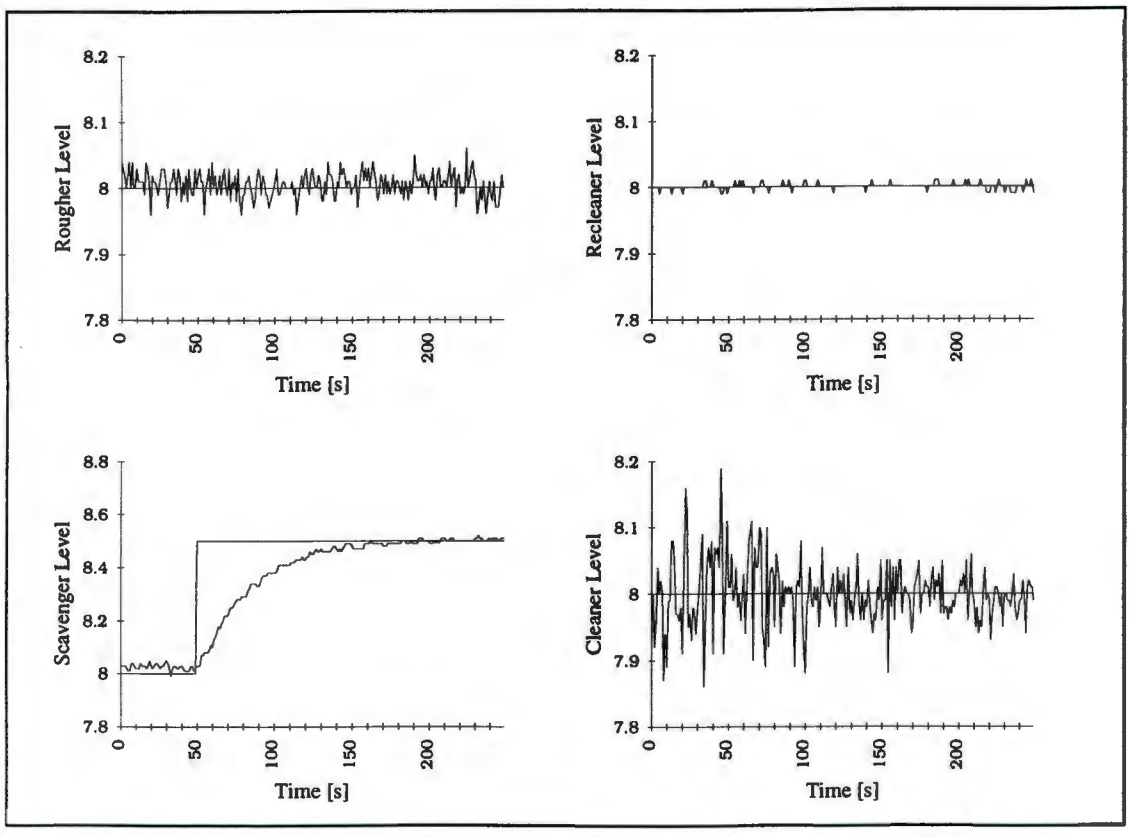

Figure B.1c Experiment 1: Scavenger stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$

Table B.1c Experiment 1: ISE and ITAE values for scavenger stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	0.0048	65.59
Cleaner	0.5638	846.15
Scavenger	4.3130	1408.31
Rougher	0.0959	479.01

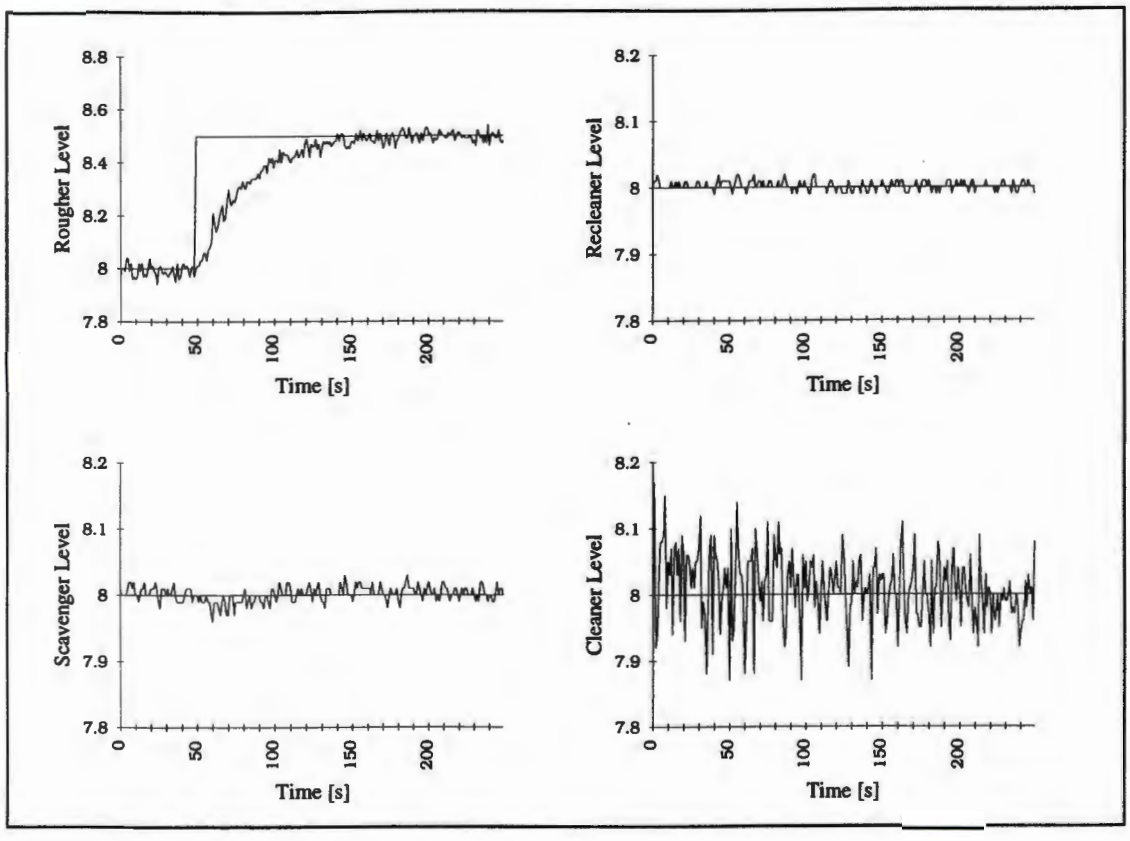

Figure B.1d Experiment 1: Rougher stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$

Table B.1d Experiment 1: ISE and ITAE values for rougher stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	0.0134	140.13
Cleaner	0.7118	1100.31
Scavenger	0.0334	248.78
Rougher	4.3819	1491.27

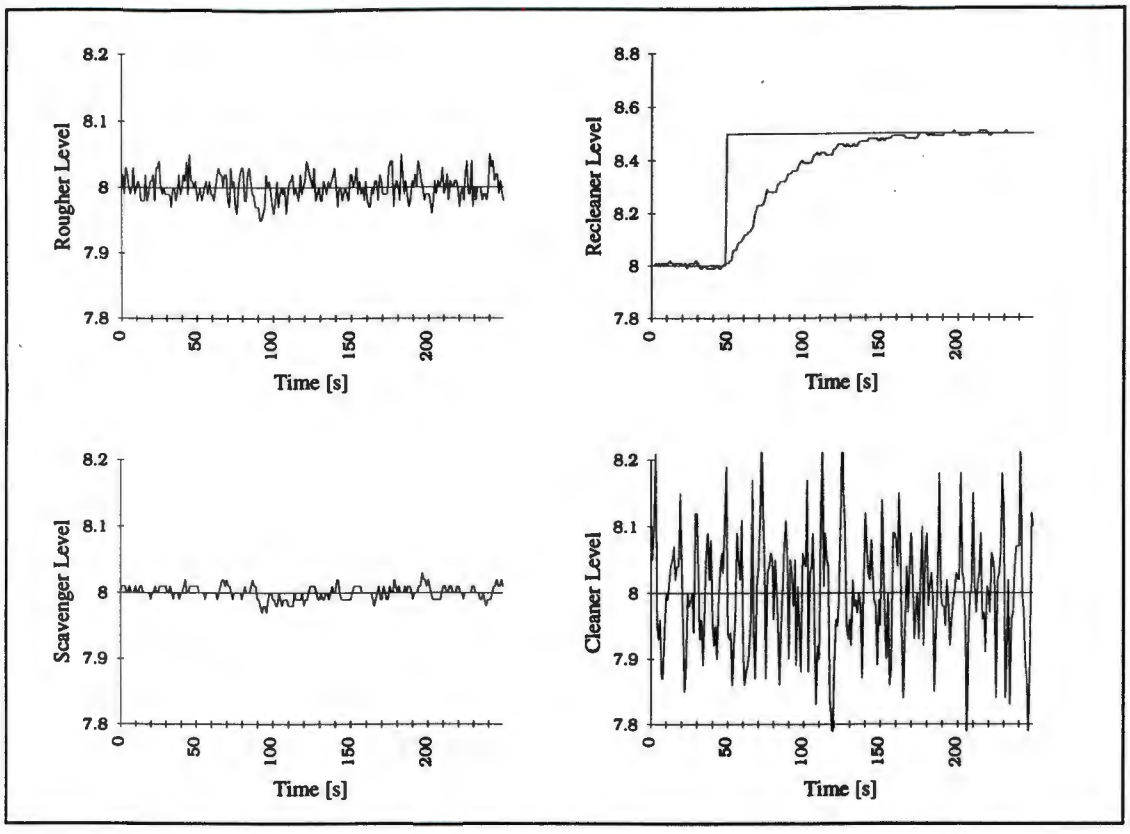

Figure B.2a Experiment 2: Recleaner stepped
$N_1 = 1, N_2 = 50, Nu = 2, \lambda = 0.5$

Table B.2a Experiment 2: ISE and ITAE values for recleaner stepped
$N_1 = 1, N_2 = 50, Nu = 2, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	4.7274	1399.66
Cleaner	1.9669	2128.76
Scavenger	0.0249	234.93
Rougher	0.0973	482.13

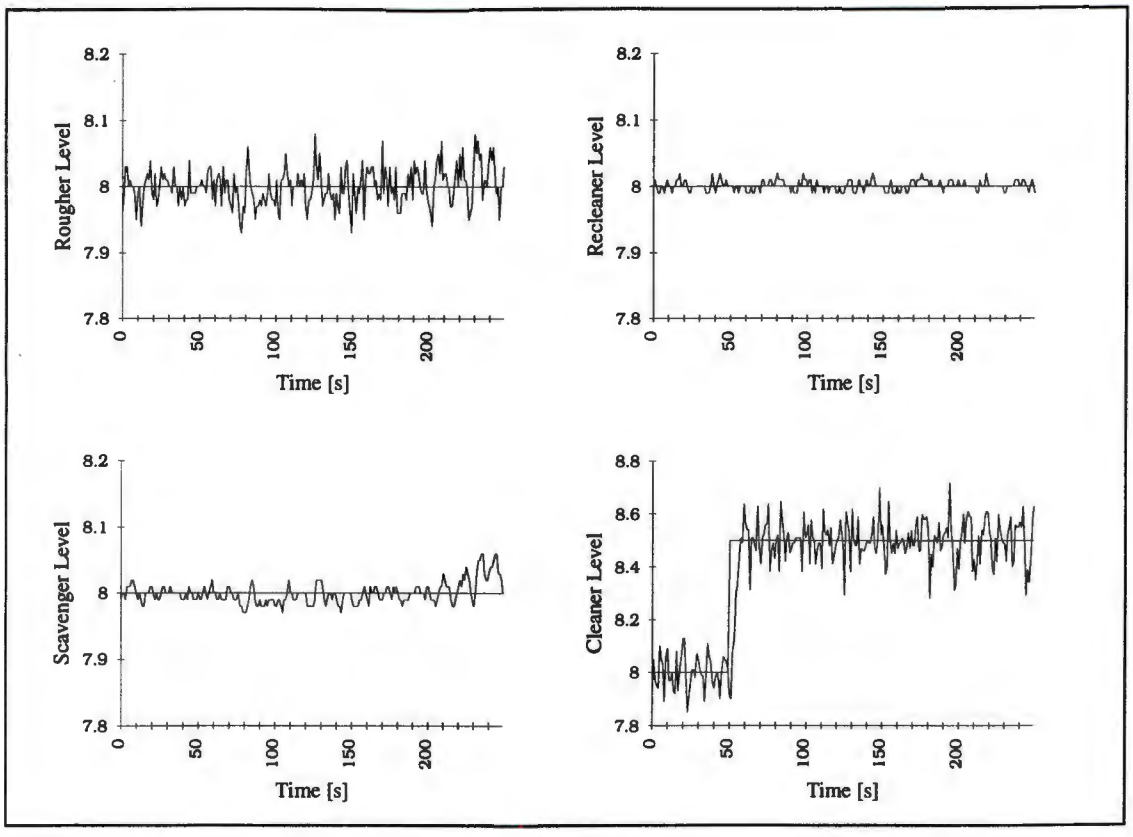

Figure B.2b Experiment 2: Cleaner stepped
$N_1 = 1, N_2 = 50, Nu = 2, \lambda = 0.5$

Table B.2b Experiment 2: ISE and ITAE values for cleaner stepped
$N_1 = 1, N_2 = 50, Nu = 2, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	0.0141	148.55
Cleaner	2.4501	1996.83
Scavenger	0.0682	440.64
Rougher	0.1971	740.375

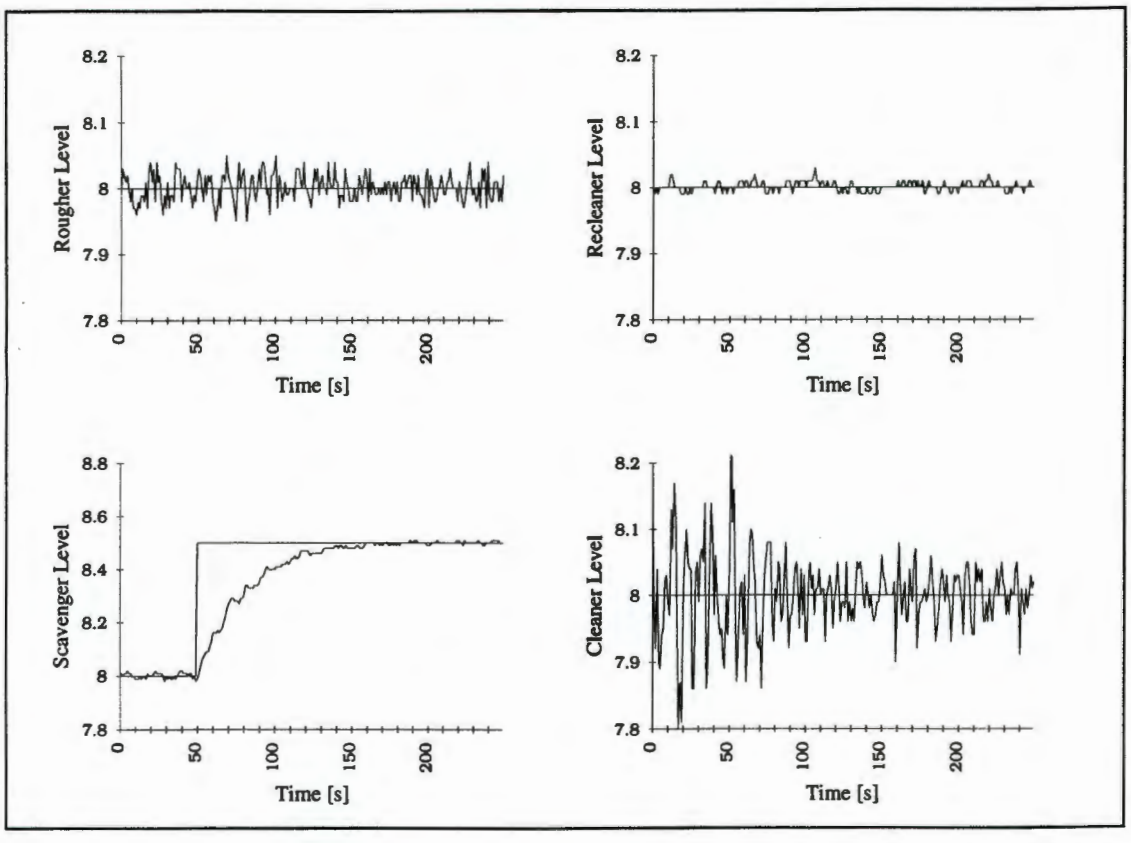

Figure B.2c Experiment 2: Scavenger stepped
$N_1 = 1, N_2 = 50, Nu = 2, \lambda = 0.5$

Table B.2c Experiment 2: ISE and ITAE values for scavenger stepped
$N_1 = 1, N_2 = 50, Nu = 2, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	0.0136	143.99
Cleaner	0.8548	978.94
Scavenger	3.9370	1184.53
Rougher	0.1072	483.43

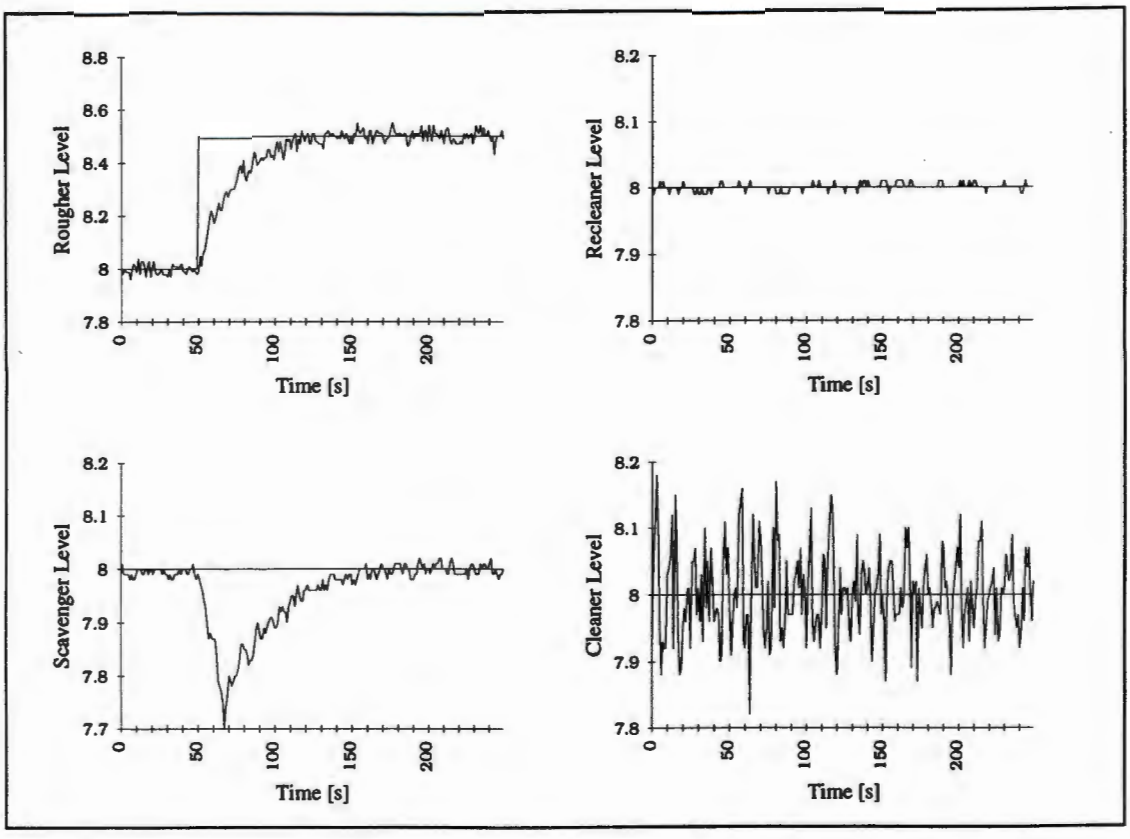

Figure B.2d Experiment 2: Rougher stepped
$N_1 = 1, N_2 = 50, Nu = 2, \lambda = 0.5$

Table B.2d Experiment 2: ISE and ITAE values for rougher stepped
$N_1 = 1, N_2 = 50, Nu = 2, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	0.0064	73.51
Cleaner	0.9788	1346.41
Scavenger	1.4144	988.76
Rougher	2.9332	1143.70

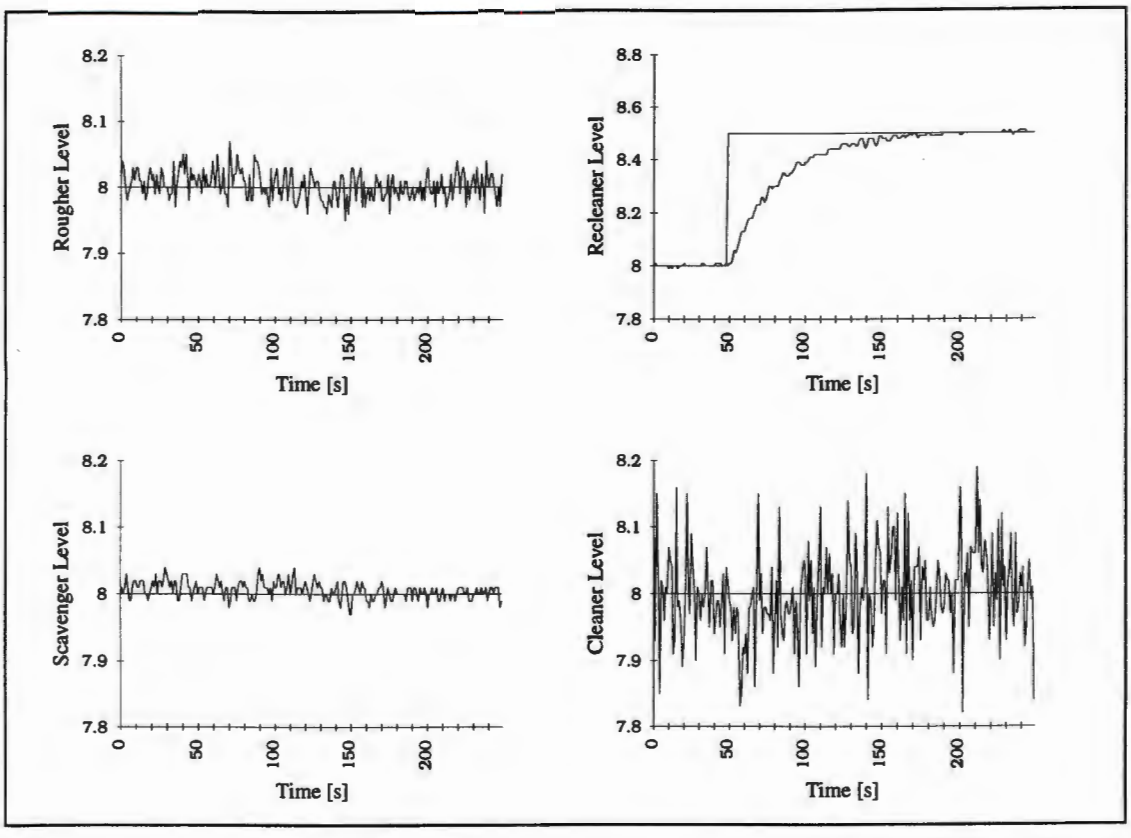

Figure B.3a Experiment 3: Recleaner stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.01$

Table B.3a Experiment 3: ISE and ITAE values for recleaner stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.01$

Tank	ISE	ITAE
Recleaner	4.3435	1332.15
Cleaner	1.1233	1614.48
Scavenger	0.0482	277.24
Rougher	0.1150	515.06

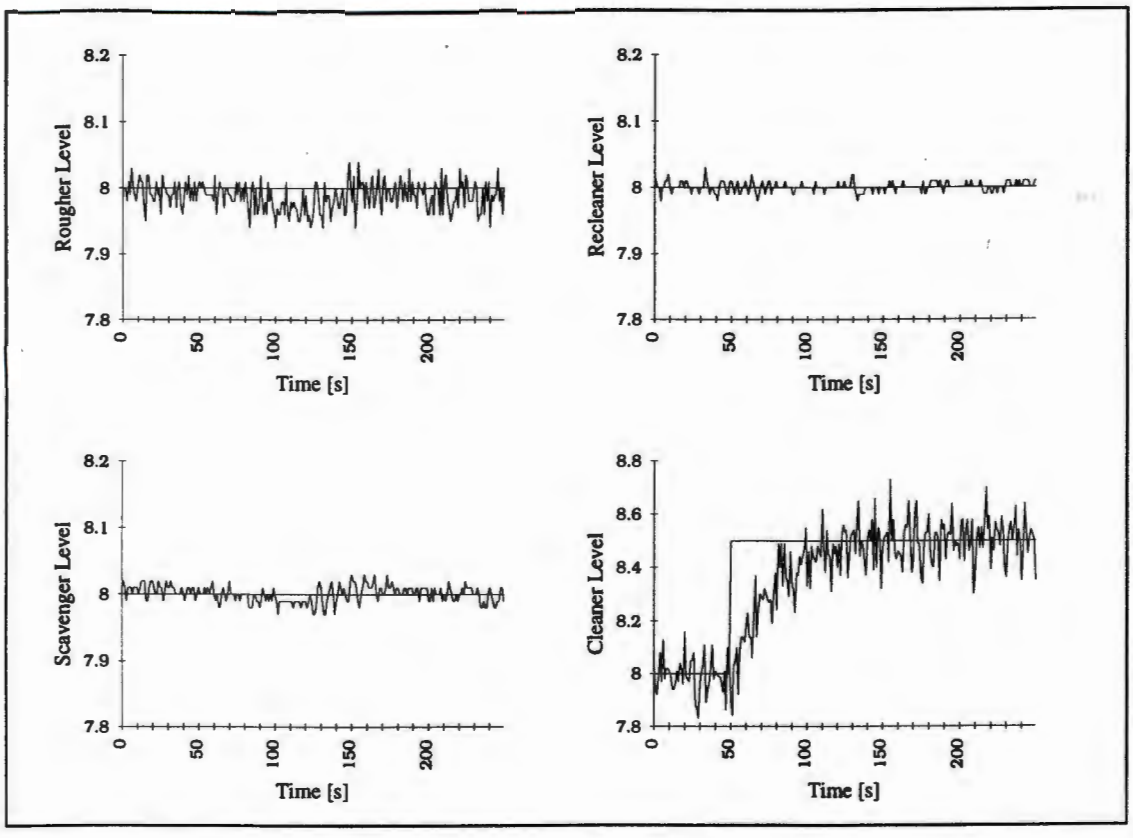

Figure B.3b Experiment 3: Cleaner stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.01$

Table B.3b Experiment 3: ISE and ITAE values for Cleaner stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.01$

Tank	ISE	ITAE
Recleaner	0.0134	125.90
Cleaner	5.3241	2520.87
Scavenger	0.0377	299.41
Rougher	0.1522	639.62

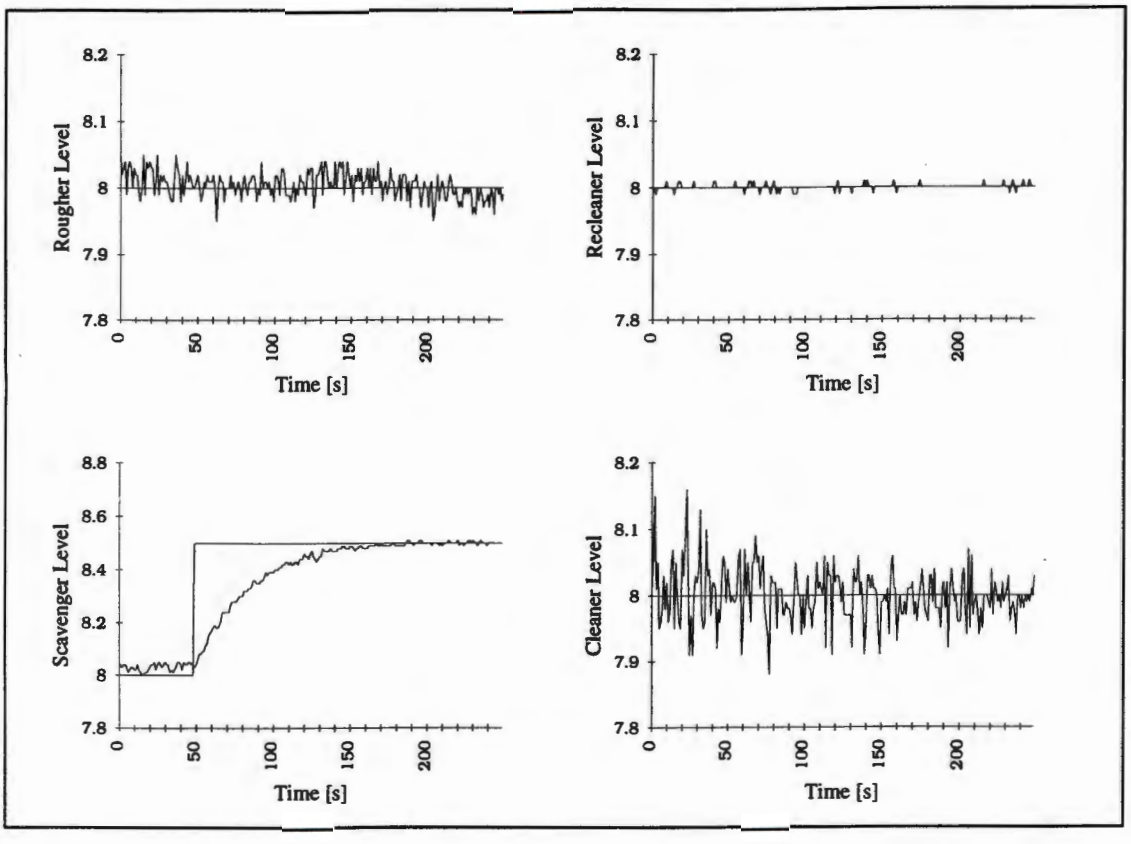

Figure B.3c Experiment 3: Scavenger stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.01$

Table B.3c Experiment 3: ISE and ITAE values for scavenger stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.01$

Tank	ISE	ITAE
Recleaner	0.0039	42.53
Cleaner	0.4277	829.58
Scavenger	3.9410	1314.93
Rougher	0.1032	471.51

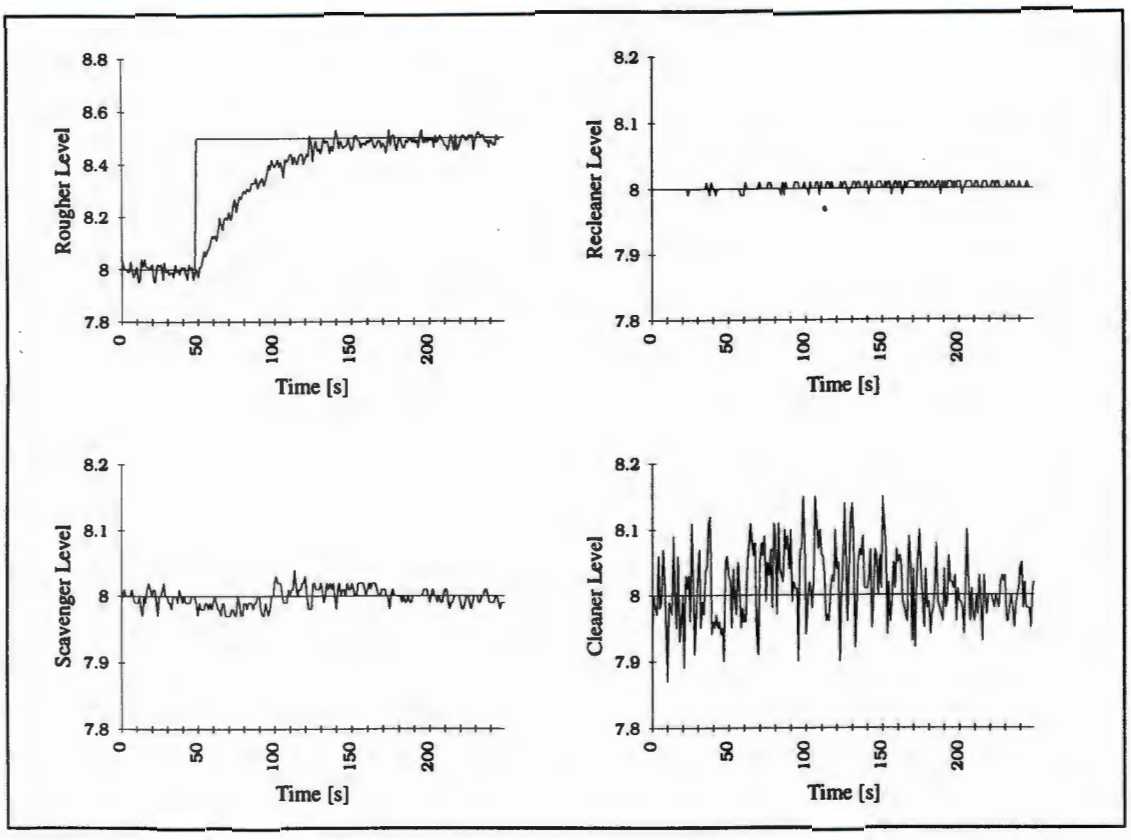

Figure B.3d Experiment 3: Rougher stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.01$

Table B.3d Experiment 3: ISE and ITAE values for rougher stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.01$

Tank	ISE	ITAE
Recleaner	0.0097	145.71
Cleaner	0.7679	1226.71
Scavenger	0.0479	295.96
Rougher	4.9375	1670.78

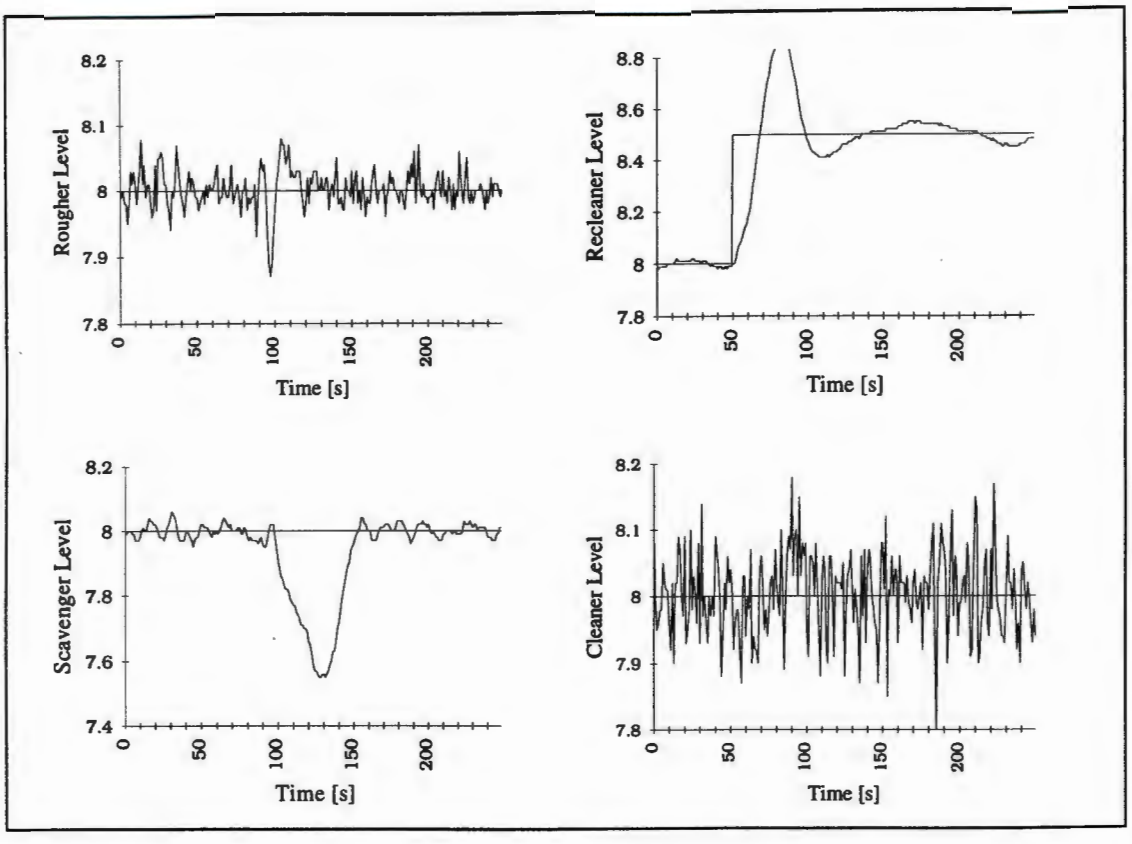

Figure B.4a Experiment 4: Recleaner stepped
$N_1 = 1, N_2 = 10, Nu = 1, \lambda = 0.5$

Table B.4a Experiment 4: ISE and ITAE values for recleaner stepped
$N_1 = 1, N_2 = 10, Nu = 1, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	4.4769	1740.35
Cleaner	0.9727	1575.48
Scavenger	4.7155	2152.43
Rougher	0.4320	582.32

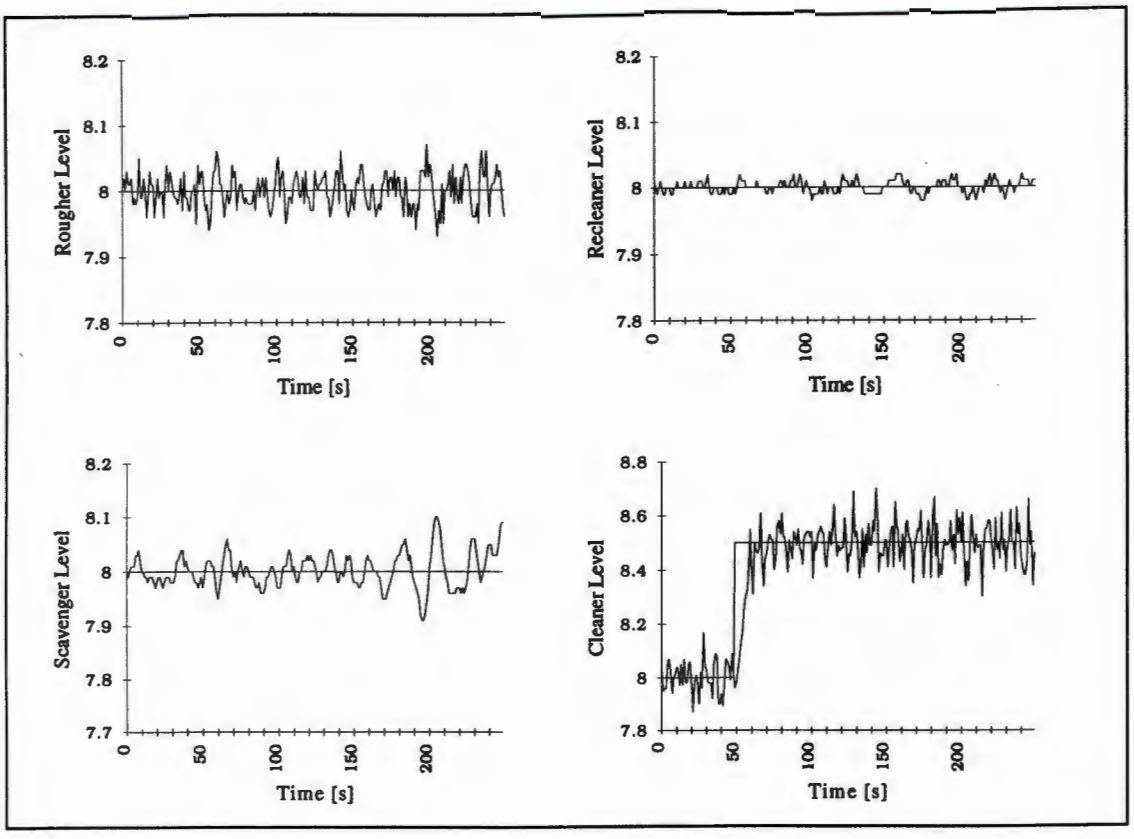

Figure B.4b Experiment 4: Cleaner stepped
$N_1 = 1, N_2 = 10, Nu = 1, \lambda = 0.5$

Table B.4b Experiment 4: ISE and ITAE values for Cleaner stepped
$N_1 = 1, N_2 = 10, Nu = 1, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	0.0212	230.22
Cleaner	2.5798	1978.22
Scavenger	0.2559	919.74
Rougher	0.1760	732.08

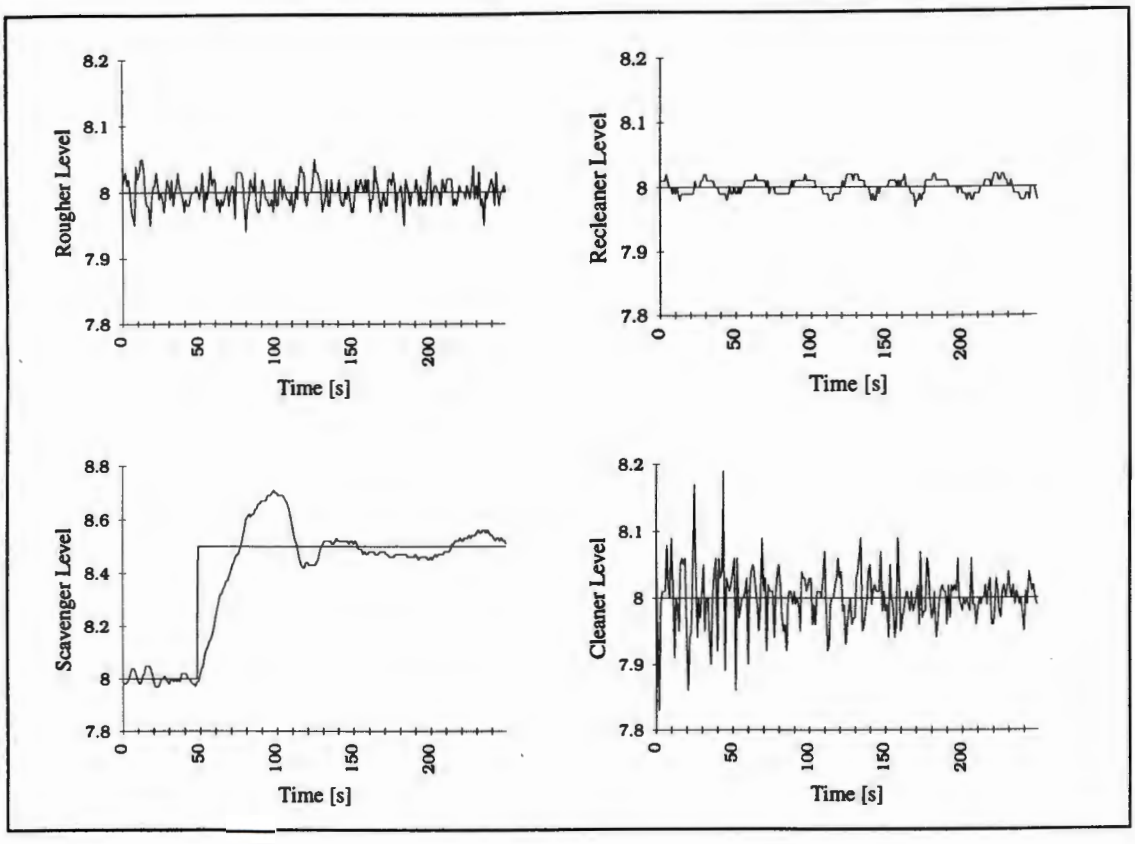

Figure B.4c Experiment 4: Scavenger stepped
$N_1 = 1, N_2 = 10, Nu = 1, \lambda = 0.5$

Table B.4c Experiment 4: ISE and ITAE values for scavenger stepped
$N_1 = 1, N_2 = 10, Nu = 1, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	0.0315	295.64
Cleaner	0.4705	771.26
Scavenger	3.2116	1675.43
Rougher	0.1006	449.89

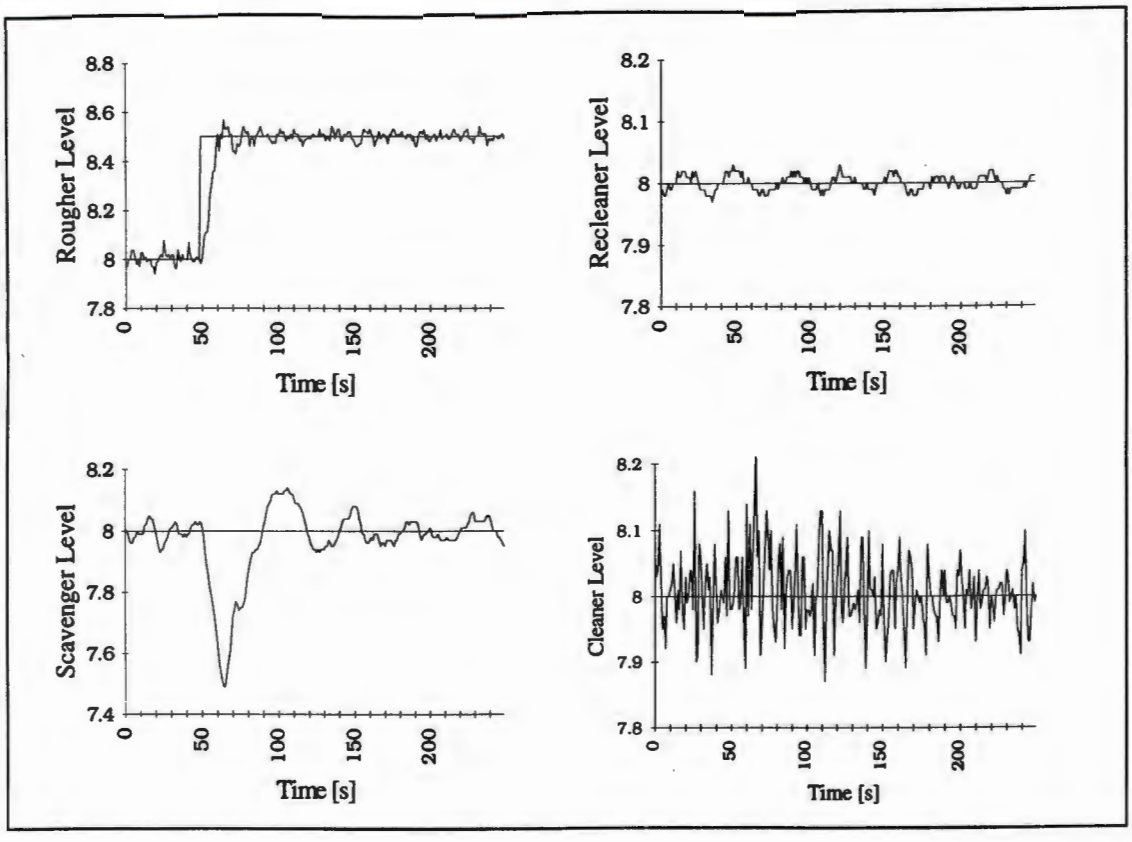

Figure B.4d Experiment 4: Rougher stepped
$N_1 = 1, N_2 = 10, Nu = 1, \lambda = 0.5$

Table B.4d Experiment 4: ISE and ITAE values for rougher stepped
$N_1 = 1, N_2 = 10, Nu = 1, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	0.0359	272.47
Cleaner	0.7771	1178.19
Scavenger	0.0025	1627.87
Rougher	1.3582	621.14

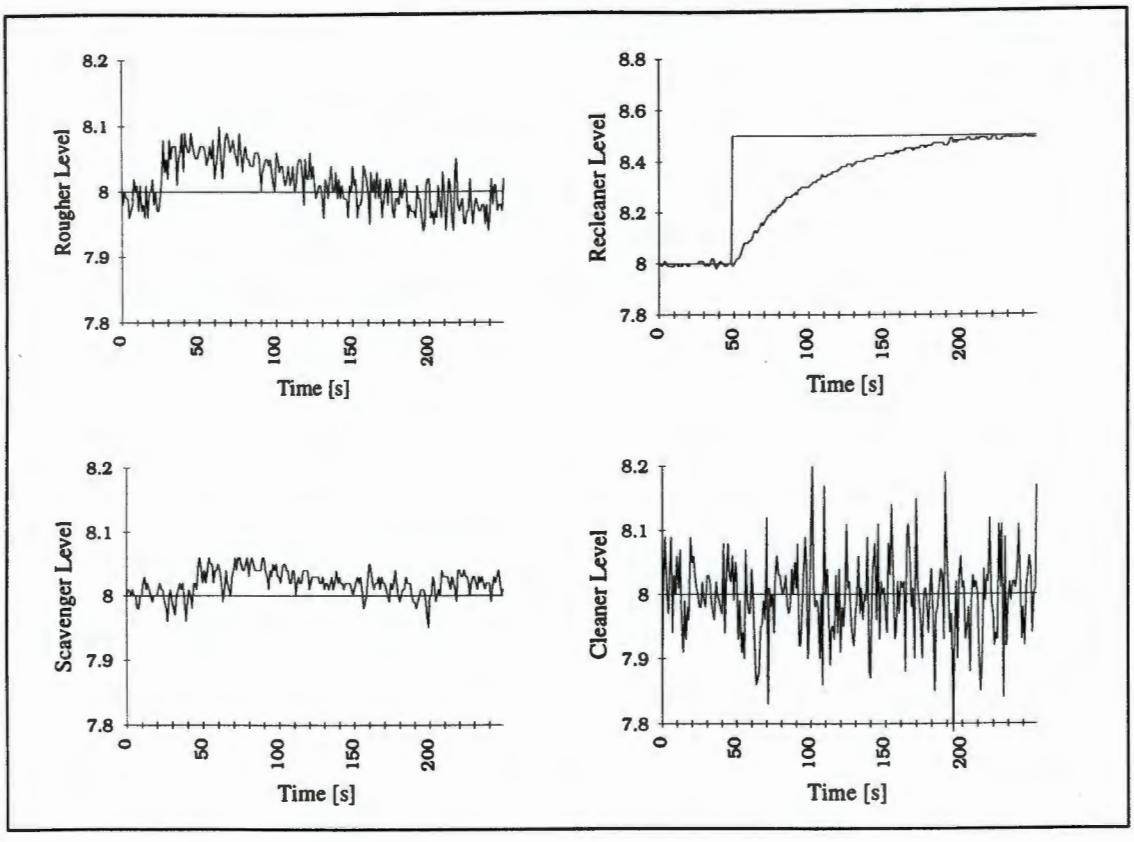

Figure B.5a Experiment 5: Recleaner stepped
$N_1 = 1, N_2 = 90, Nu = 1, \lambda = 0.5$

Table B.5a Experiment 5: ISE and ITAE values for recleaner stepped
$N_1 = 1, N_2 = 90, Nu = 1, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	6.7865	2440.01
Cleaner	1.0484	1592.82
Scavenger	0.1957	702.46
Rougher	0.3926	900.08

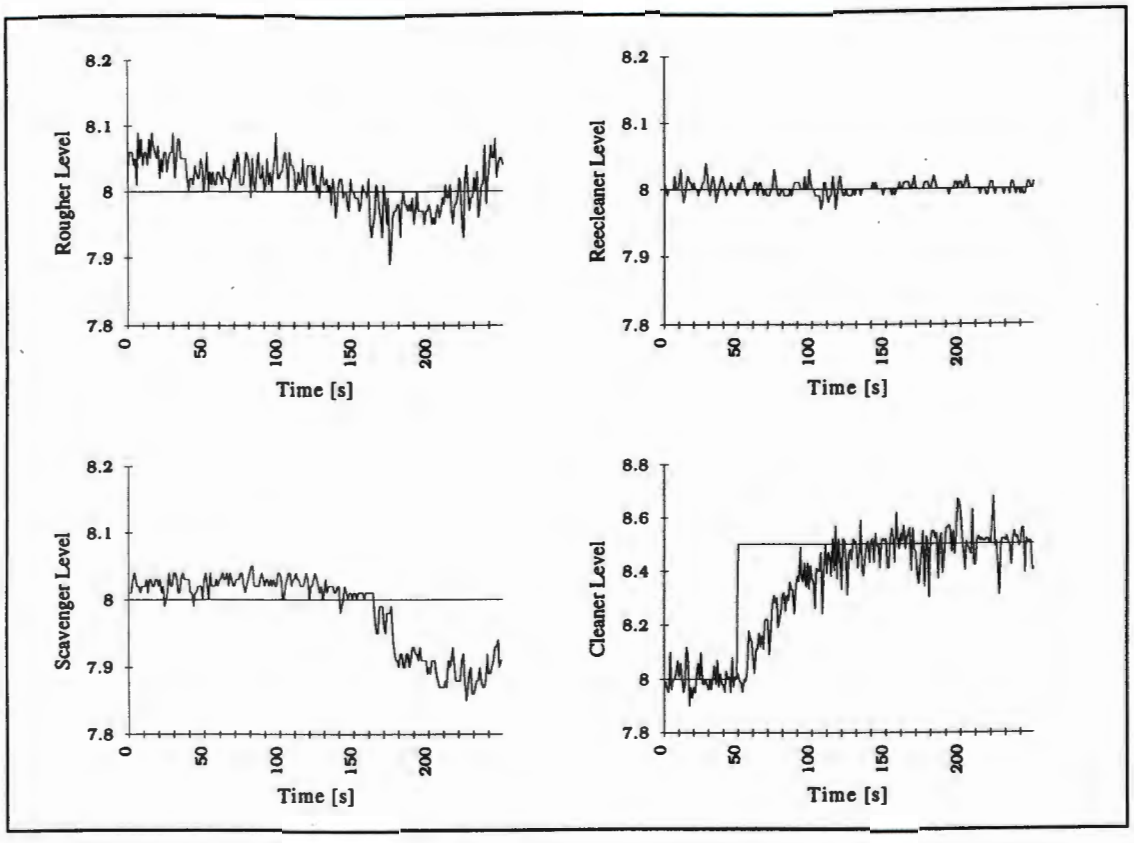

Figure B.5b Experiment 5: Cleaner stepped
$N_1 = 1, N_2 = 90, Nu = 1, \lambda = 0.5$

Table B.5b Experiment 5: ISE and ITAE values for Cleaner stepped
$N_1 = 1, N_2 = 90, Nu = 1, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	0.0237	165.47
Cleaner	5.8250	2382.83
Scavenger	0.8943	1909.48
Rougher	0.3631	873.38

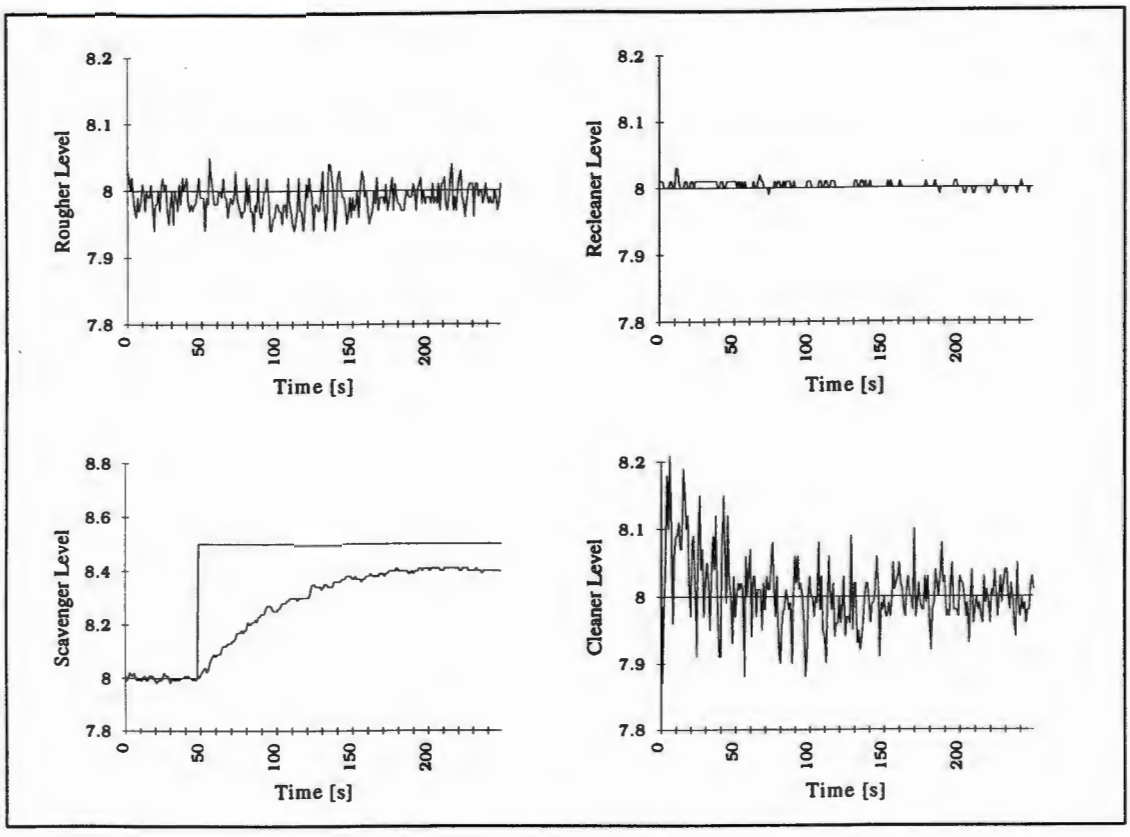

Figure B.5c Experiment 5: Scavenger stepped
$N_1 = 1, N_2 = 90, Nu = 1, \lambda = 0.5$

Table B.5c Experiment 5: ISE and ITAE values for scavenger stepped
$N_1 = 1, N_2 = 90, Nu = 1, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	0.0108	83.54
Cleaner	0.7093	904.74
Scavenger	9.1545	4342.56
Rougher	0.1604	614.20

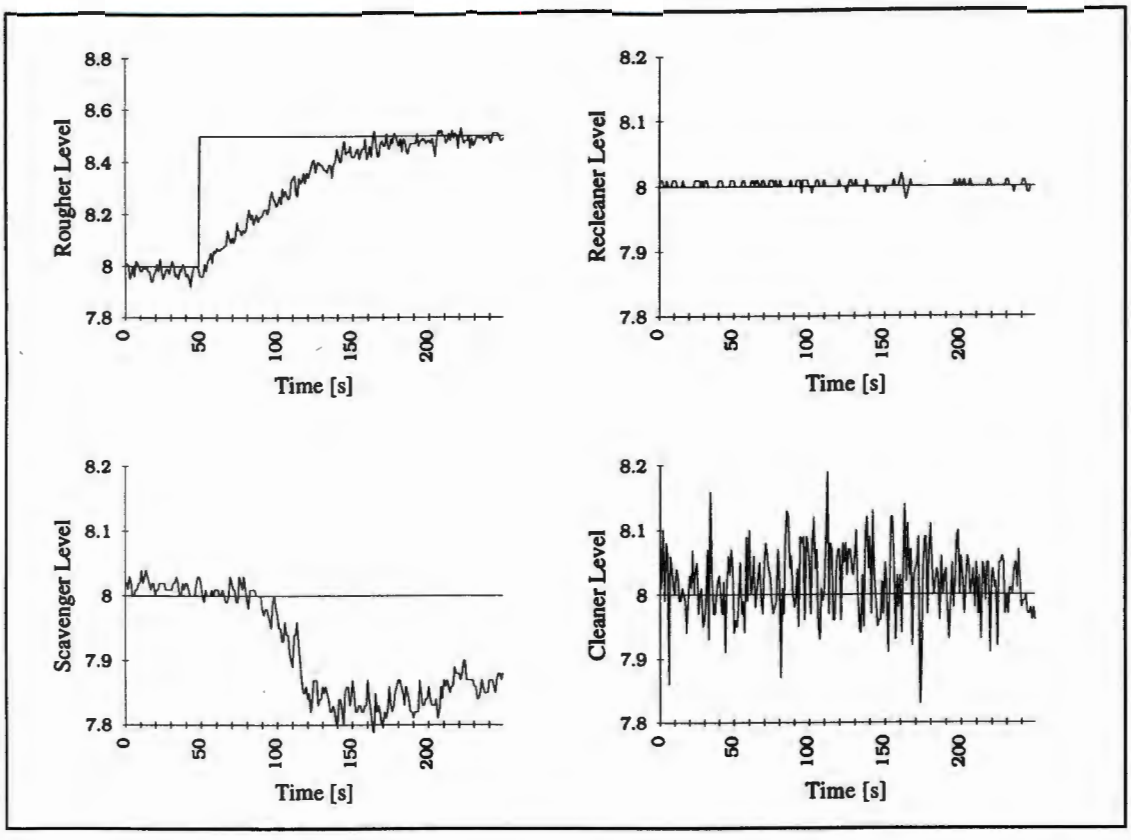

Figure B.5d Experiment 5: Rougher stepped
$N_1 = 1, N_2 = 90, Nu = 1, \lambda = 0.5$

Table B.5d Experiment 5: ISE and ITAE values for rougher stepped
$N_1 = 1, N_2 = 90, Nu = 1, \lambda = 0.5$

Tank	ISE	ITAE
Recleaner	0.0083	85.63
Cleaner	0.8319	1426.32
Scavenger	3.3798	3896.93
Rougher	8.8276	2730.71

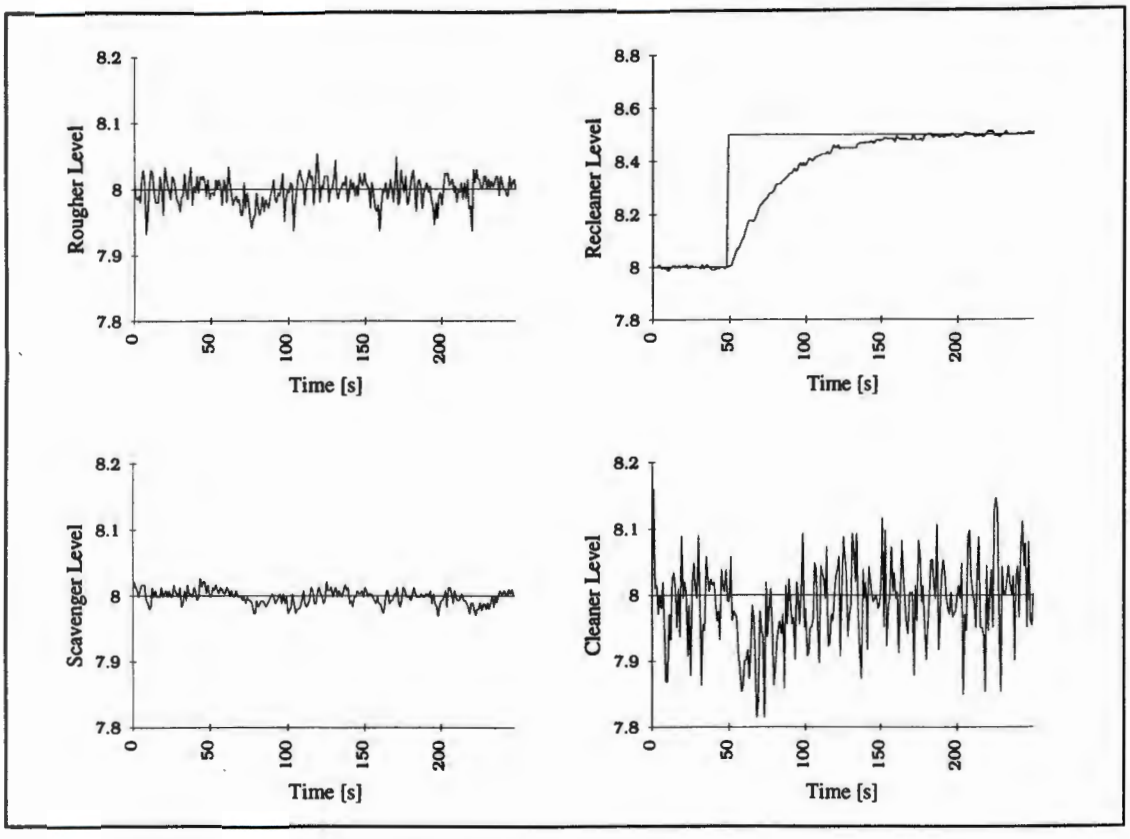

Figure B.6a Experiment 6: Recleaner stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$ using LRPC

Table B.6a Experiment 6: ISE and ITAE values for recleaner stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$ using LRPC

Tank	ISE	ITAE
Recleaner	4.4942	1413.48
Cleaner	1.0211	1503.09
Scavenger	0.0321	280.15
Rougher	0.1116	488.83

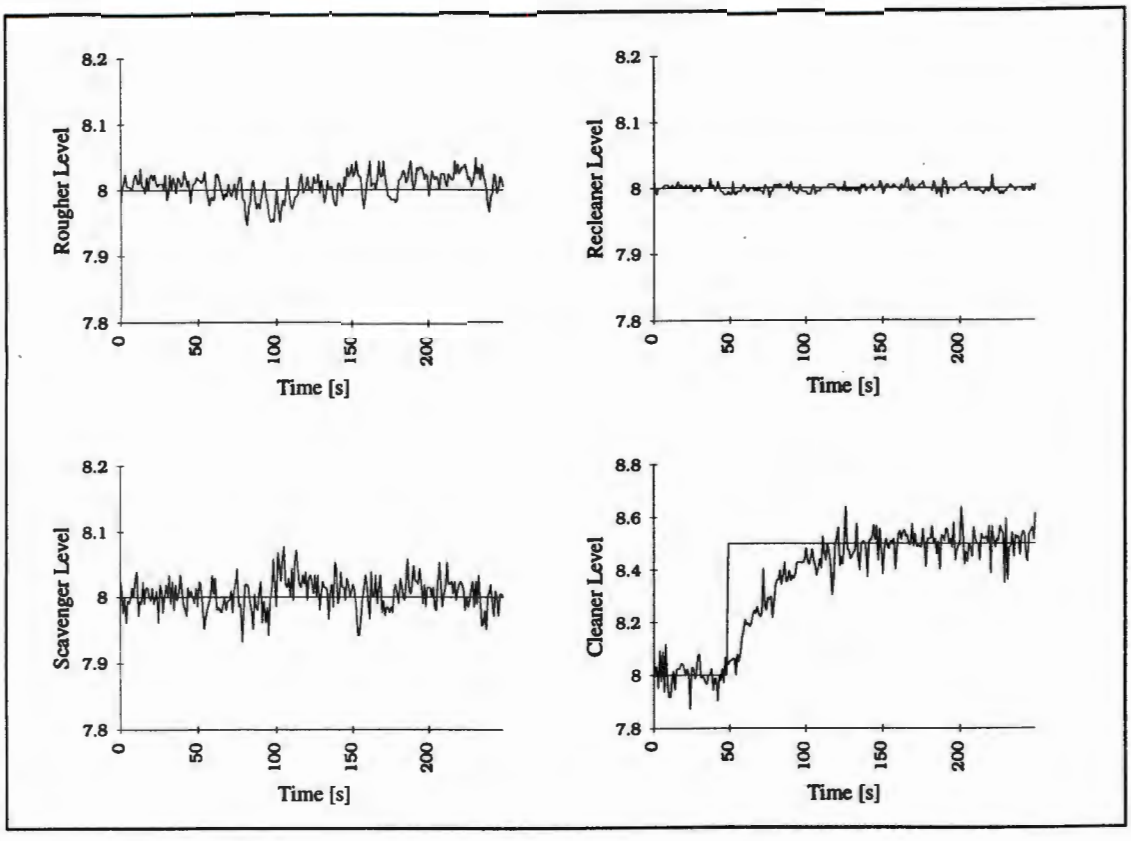

Figure B.6b Experiment 6: Cleaner stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$ using LRPC

Table B.6b Experiment 6: ISE and ITAE values for Cleaner stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$ using LRPC

Tank	ISE	ITAE
Recleaner	0.0075	117.36
Cleaner	4.2806	1964.58
Scavenger	0.1602	631.20
Rougher	0.1062	578.03

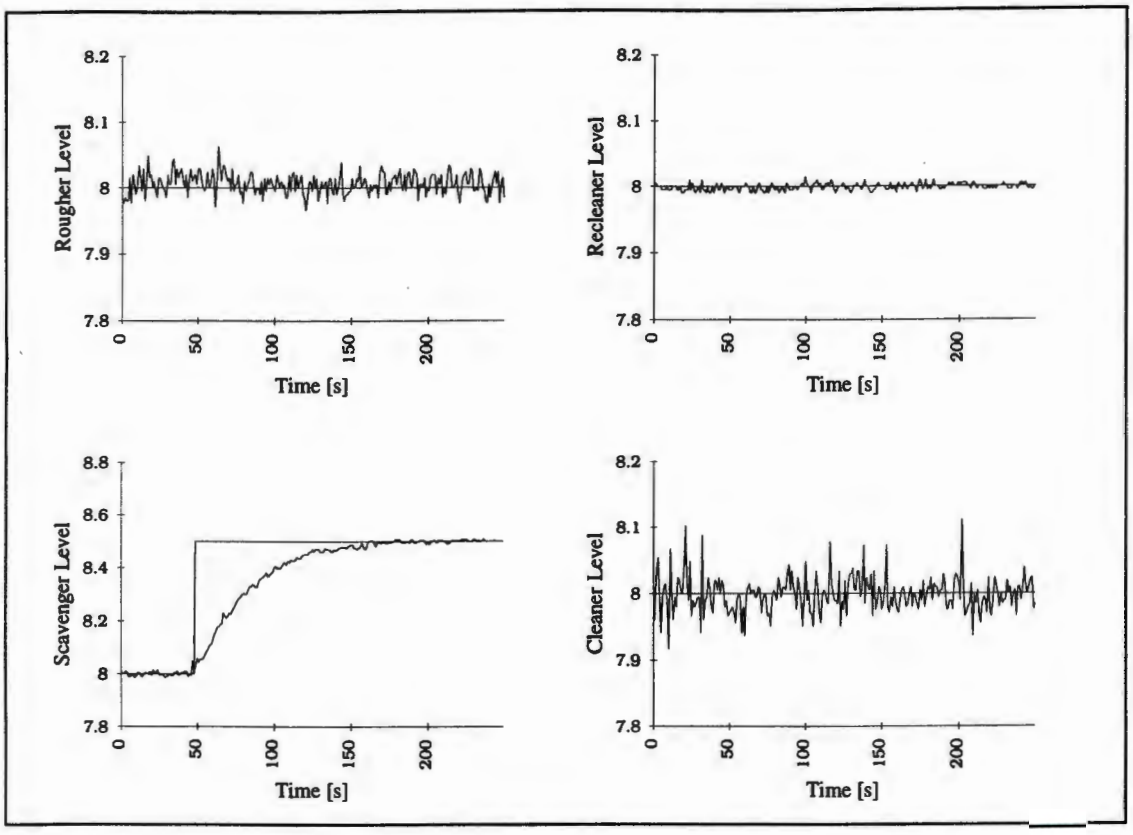

Figure B.6c Experiment 6: Scavenger stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$ using LRPC

Table B.6c Experiment 6: ISE and ITAE values for scavenger stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$ using LRPC

Tank	ISE	ITAE
Recleaner	0.0064	99.54
Cleaner	0.1801	569.17
Scavenger	4.3527	1335.05
Rougher	0.0734	410.19

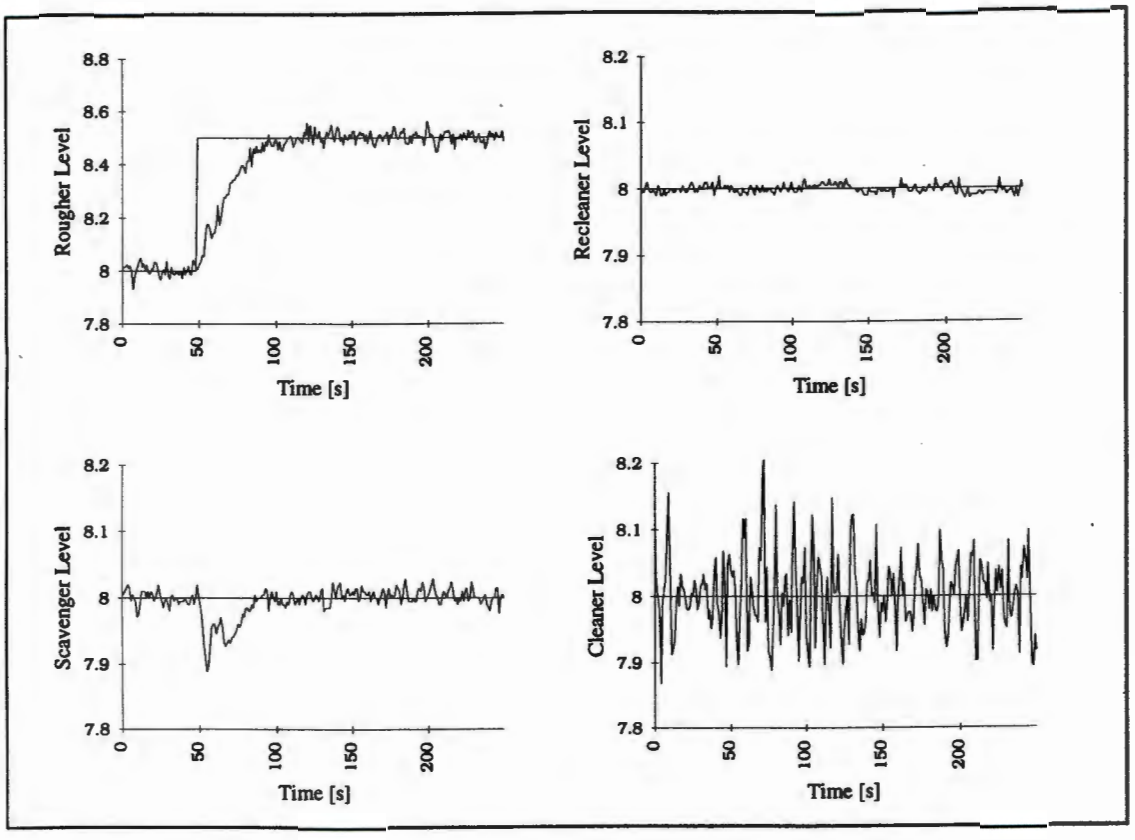

Figure B.6d Experiment 6: Rougher stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$ using LRPC

Table B.6d Experiment 6: ISE and ITAE values for rougher stepped
$N_1 = 1, N_2 = 50, Nu = 1, \lambda = 0.5$ using LRPC

Tank	ISE	ITAE
Recleaner	0.0103	161.83
Cleaner	0.8138	1317.04
Scavenger	0.1249	344.86
Rougher	2.9544	1067.48

APPENDIX C

Software Reference

Below follows a listing and description of all the basic procedures used for the development of the GPC and LRPC algorithms.

Table C.1 Polynomial functions

Function Name	Source File	Description
zerov	polutils.c	To zero the polynomial, usually used for initialisation.
polmul	polutils.c	To multiply two polynomials.
poldiv	polutils.c	To divide two polynomials
poldiv2	polutils.c	Second procedure to divide two polynomials.
poladd	polutils.c	To add two polynomials.
trnsp	polutils.c	To transpose a vector. Note that the value returned is a matrix.
dispp	polutils.c	To display a polynomial.

Table C.2 Matrix functions

Function Name	Source File	Description
zerom	matutils.c	To zero a matrix, usually used for initialisation.
invmat	matutils.c	To invert a matrix. This inversion uses the LU decomposition and back substitution method.
addmat	matutils.c	To add two matrices.
submat	matutils.c	To subtract two matrices.
multmat	matutils.c	To multiply two matrices.
smultmat	matutils.c	To multiply a matrix by a scalar number (element by element).
trns	matutils.c	To transpose a matrix.
iden	matutils.c	To create an identity matrix of any size.
dispm	matutils.c	To display a matrix.
ludcmp	ludcmp.c	To do LU decomposition.
lubksb	lubksb.c	To do LU back substitution.

Table C.3 Memory allocation functions

Function Name	Source File	Description
nrerror	nrutil.c	To provide error message indicating that memory allocation failed.
vector	nrutil.c	To allocate memory for vector of specified size and type double. Uses C function malloc().
free_vector	nrutil.c	To free memory allocated for specified vector.
ivector	nrutil.c	To allocate memory for vector of specified size and type integer. Uses C function malloc().
free_ivector	nrutil.c	To free memory allocated for specified vector.
matrix	nrutil.c	To allocate memory for matrix of specified size and type double. Uses C function malloc().
free_matrix	nrutil.c	To free memory allocated for specified matrix.

Table C.4 Signal generator functions

Function Name	Source File	Description
Step	SigGen.c	To create a single step signal.
Square	SigGen.c	To create a square wave.
Triangular	SigGen.c	To create a triangular wave.
Sine	SigGen.c	To create a sine wave.

Table C.5 Control algorithm functions

Function Name	Source File	Description
filter	filter.c	To filter data.
spectral_factorisation	SpctlFac.c	To find the LRPC filter by spectral factorisation.
rls	rls.c	To calculate plant model using recursive least squares.
recur	recur.c	To calculate GPC polynomials using the recursion method.
ctlmat	CtlMat.c	To calculate the control law matrix for GPC.
plant	plant.c	To simulate plant for simulator.
control_increment	CtlSig.c	To calculate the control increment to be used for the control signal to be output.
y_known_t	YknownT.c	To calculate predicted y (output) values.

Table C.6 Control algorithm member functions of GPC_Simulator class

Function Name	Source File	Description
GPC_Simulator	GpcClass.cpp	Constructor for the GPC_Simulator class.
Init	GpcClass.cpp	To initialise the member variables of the class.
~GPC_Simulator	GpcClass.cpp	Destructor for the GPC_Simulator class.
Free_Memory	GpcClass.cpp	To free the memory used for the vectors and matrices.
Save	GpcClass.cpp	Saving setpoint and controllable parameters.
SetLog	GpcClass.cpp	To set logging on or off.
Log	GpcClass.cpp	To log a particular set of data.
SetSpectral	GpcClass.cpp	To set the use of the spectral factorisation routine.
ResetSpectral	GpcClass.cpp	To reset the use of the spectral factorisation routine.
SimulatePlant	GpcClass.cpp	To set the program for simulation mode.
RealPlant	GpcClass.cpp	To set the program for running on physical plant.
SetupStep	GpcClass.cpp	To set the step setpoint parameters.
HardSetup	GpcClass.cpp	To setup the hardware.
Initialize	GpcClass.cpp	To initialise the GPC control algorithm.

Output	GpcClass.cpp	To output the data to a file.
Output_model	GpcClass.cpp	To output the model.
SetStore	GpcClass.cpp	To set the program for storage.
SetTracking	GpcClass.cpp	To set the tracking parameters i.e. time and tolerance.
Tracking	GpcClass.cpp	To check if signal is tracking.
Simulate	GpcClass.cpp	To execute the GPC/LRPC algorithm.
Plant_Change	GpcClass.cpp	To change plant models in order to test adaptive approach for simulation.
Load_Dist	GpcClass.cpp	To add load disturbance for simulation.

Table C.7 Member functions for VMEBus class
(needs the VMEbus device driver to work)

Function Name	**Source File**	**Description**
VMEBus	vmebus.cpp	Constructor for VMEbus class - sets up the VMEbus and addressing for I/O cards.
~VMEBus	vmebus.cpp	Destructor for VMEbus class - frees memory used.
ADC	vmebus.cpp	To input value from ADC on VMEbus.
DAC	vmebus.cpp	To output value from DAC on VMEbus.
DIG	vmebus.cpp	To output/input value from digital I/O card.

The Windows NT VMEbus Device Driver

In order to install the VMEbus device driver the driver first has to be registered in the Windows NT registry. To accomplish this one runs the following command from a command prompt:

c:\>REG.INI MAPMEM.INI

Copy the mapmem.sys file to the WINNT\SYSTEM32\DRIVERS directory.

For the driver to be able to work the machine needs to be rebooted.

The Windows NT operating system also needs to be told when to start the device driver. This is done by selecting the DRIVERS icon in the CONTROL PANEL. Select the startup option appropriate for the application e.g. if the VMEbus is to be set all the time then choose AUTOMATIC startup option, or if the driver is only needed at particular times then set the option to MANUAL.

The REG.INI file is given below.

```
\registry\machine\system\currentcontrolset\services\MapMem
   Type = REG_DWORD 0x00000001
   Start = REG_DWORD 0x00000003
   Group = Extended base
   ErrorControl = REG_DWORD 0x00000001
```

The appropriate files can be found on the disk accompanying the thesis.

To compile the code for the device driver, the Microsoft Device Driver Kit is needed. This kit is available as part of the Microsoft Developers' Platform which provides information vital to writing programs for the Microsoft operating systems and applications.