THE APPLICATION OF FUZZY CONTROL TO FED-BATCH FERMENTATION

This dissertation has been submitted to the Department of Chemical Engineering, University of Cape Town, in fulfilment of the requirements of Master of Science in Engineering (Chemical).

by

Trevor Deon Hadley

October 1995

ACKNOWLEDGEMENTS

I thank my supervisor Dr Swartz for all his assistance and guidance throughout the past two years. Thanks also goes to Clifford, Julian, Mark and Roderick in the Process Systems Engineering group for their help and support.

I would like to thank FRD for funding this research.

I would also like to thank Bill Randall and Granville de la Cruz for all their computer expertise and assistance.

I would especially like to acknowledge the support of my parents throughout the duration of my post-graduate studies and the Lord for His constant love and guidance.

SYNOPSIS

Fermentation processes are highly nonlinear and subject to variability. The fermentation's states are not readily available on-line and therefore the application of closed loop control schemes have been hindered. It was decided to investigate fuzzy control as it is able to deal with systems whose operation does not easily fit into the mathematical framework of traditional control approaches such as fermentations where the systems are highly nonlinear. The fermentation of lysine is an emergent industry in South Africa and it was therefore decided to focus on this fermentation. The control of penicillin fermentation was also investigated as it closely resembles the fermentation of lysine.

A review of the types of control and estimation techniques used in the literature for biosystems was done to assess state of art in biocontrol. This covered optimal control techniques, neural networks, fuzzy controllers and adaptive control techniques.

The operation and properties of fuzzy controllers were investigated. A specific form of fuzzy controller, presented in the literature, which was shown to correspond to a PI controller with a nonlinear gain was discussed. The effect of the number of output sampling points was analysed and it was found that the number of output sampling points used has an effect on the output and input response. It was also found that a higher number of sampling points results in a nonlinear integral constant and a nonlinear gain which has more resolution. The fuzzy controller's output response equations were found to be of a PI form with a possible bias term irrespective of the number of sampling points. The fuzzy controller was shown to yield better output and input response to that of an equivalently tuned linear PI controller for a first, second and third order system because it is able to take advantage of its nonlinear form. It was also shown that it is possible to obtain less severe input action for relatively the same value of SSE (sum of squared errors) when a higher number of sampling points is used for a first order system with dead time.

The regulatory fuzzy controller was also shown to have good disturbance rejection capabilities compared to a linear PI controller when applied to the fermentations of penicillin and lysine. In both cases the control performance was evaluated through simulation. Fuzzy control was also applied in a supervisory capacity to the control of lysine fermentation where it is necessary to manipulate the fermenter's feed flow rate to ensure that product formation is favoured. The fuzzy controller was found to yield a higher profit percent despite model uncertainty in a coefficient, C, than a preset optimal trajectory obtained from the literature. The heuristic fuzzy controller was also found to have better disturbance rejection capabilities than the preset trajectory presented in the literature. This was because the fuzzy controller was able to adjust its input response whereas the trajectory presented in the literature was set *a priori*.

It was concluded that fuzzy control can be applied in both a regulatory and supervisory capacity to fed-batch fermentations and showed improved control over standard control techniques for the fermentation of lysine and penicillin.

TABLE OF CONTENTS

LIST OF DIAGRAMS

List of Figures

List of Tables

NOMENCLATURE

A_i	is the fuzzy set constituting the antecedent part of the i^{th} fuzzy rule, formed from the intersection of the input fuzzy sets of the i^{th} rule
$A_{i,j}$	are conditions which are to be met
B_i	is the fuzzy set constituting the consequent part of the i^{th} fuzzy rule
C	is the coefficient used in the equation describing the specific growth rate for lysine fermentation (C has a value of 0.125 for the ideal case)
CER	is the carbon dioxide evolution rate
cost ratio	is the ratio of the mass of substrate used (for the lysine fermentation) to the mass of product formed
DO	is the level of dissolved oxygen in the fermenter
e	is the error in the controlled variable (y_{set} - y)
E.n	stands for the grade of membership of the error in the fuzzy set *error negative*
E.p	stands for the grade of membership of the error in the fuzzy set *error positive*
error	is the error in the controlled variable (y_{set} - y)
F	is the feed volumetric flow rate
F_k	is the feed flow rate in the k^{th} time interval (L/h)
GE	is the fuzzy scaling factor for the error
GEe(t)	represents the scaled error
GM	is the value of the grade of membership
GR	is the fuzzy scaling factor for the rate
GRr(t)	represents the scaled rate
GU	is the fuzzy scaling factor for the change in output
i	is a non-negative integer used in the specification of the output sampling points
K	is a product first order decay rate [h^{-1}] (penicillin fermentation)
k	is the degradation constant of the product (used in Chapter 2)
K_1	is a constant used in the modelling equation of penicillin fermentation
K_i	is a constant used in the modelling equation of penicillin fermentation
K_p	is a constant used in the modelling equation of penicillin fermentation
K_{PI}	is the gain used for the linear PI controller
L	is the width of all fuzzy sets for the Qin style controller

m	is the number of fuzzy rules used
M_x	is the term for cell maintenance [h^{-1}] (penicillin fermentation)
n	is the number of output sampling points used for the Qin style fuzzy controller
nf	is the number of fuzzy controller inputs
NR2	is the value of the output response which was obtained by determining the weighted mean of the fuzzy output
OUR	is the oxygen utilisation rate
p	is the product concentration
PR	is the product ratio obtained via the fuzzy controller
$PR_{M\&L}$	is the product ratio obtained by Modak and Lim (1987) for the ideal model (12.661392)
profit percent	is value of the profit ratio against the profit ratio obtained by Modak and Lim (1987) for the ideal lysine fermentation
profit ratio	is the ratio of the mass of product formed (for the lysine fermentation) to the mass of substrate used
q	is any non-negative integer value
Q_p	is the specific product formation rate for lysine fermentation ($g_{prod}/g_{biomass}.h^{-1}$) (used in Chapters 4 and 5)
r	is the change in the error ($e(t+\Delta t) - e(t)$)
rate	is the change in the variable's error ($e(t+\Delta t) - e(t)$)
R.n	stands for the grade of membership of the rate in the fuzzy set *rate negative*
R.p	stands for the grade of membership of the rate in the fuzzy set *rate positive*
s	is the substrate concentration
s_F	is the feed substrate concentration (used in Chapter 2)
s_i	is the concentration of glucose in the feed stream (g/L) (used in Chapters 4 and 5)
t_1	is the time that the fed-batch stage is initiated (h) (Chapter 5)
t_2	is the time that the fed-batch stage is terminated (h) (Chapter 5)
t_f	is the final fermentation time
V	is the volume of the fermentation broth
x	is the biomass concentration
x_{ji}	is the numerical value of the j^{th} controller input of rule i.
Y	is the yield factor and taken to be 0.135 $g_{biomass}/g_{subs}$

Y_p is the yield of product from substrate (penicillin fermentation)

y_r is the r^{th} numerical value used in evaluating the controller output, termed an output sampling point

Y_x is the yield of biomass from substrate (penicillin fermentation)

Z is the fuzzy set formed from the union of each rule's fuzzy set

Z_i is the fuzzy set formed from the intersection of the antecedent and consequent part of the i^{th} fuzzy rule

β is the constant used in the determination of the rate's scaling factor (GR)

γ is negative if rate is negative and positive if rate is positive

Δb is the net conversion (in moles) of the biomass

Δc is the net conversion (in moles) of the carbon dioxide

Δp is the net conversion (in moles) of the product

Δs is the net conversion (in moles) of the substrate

Δt_{sample} is the sampling time used during integration

δ is negative if $|GRr| < |GEe|$ and positive if $|GRr| > |GEe|$

θ is the specific product formation rate for penicillin fermentation [h^{-1}] (used in Chapters 4 and 5)

θ_m is a constant used in the modelling equation of penicillin fermentation

μ is the specific cell growth rate

$\mu_A(x)$ is the grade of membership of a value x in the fuzzy set A

μ_m is a constant used in the modelling equation of penicillin fermentation

π is the product formation rate (used in Chapter 2)

σ is the substrate consumption rate (used in Chapter 2)

τ_{PI} is the integral constant used for the linear PI controller

1. INTRODUCTION

It is of great importance that bioprocess systems be operated in a cost effective manner. Previous improvements to the process involved optimising the microorganisms' characteristics through isolation and mutation techniques. Because of the expansion of the bioprocess industry and the increase in market competition, biotechnological industries are now concentrating on better monitoring and control of the bioprocesses (Morris *et al*,1991).

Traditionally industrial fermentations have been controlled by experienced operators. These operators monitor the process in order to maintain an 'ideal' state trajectory through the use of heuristics and past information. However, the fermentation process is highly nonlinear and subject to variability. Coupled with this is the fact that the fermentations' states are not readily available on-line. These factors have hindered the application of closed loop control schemes and therefore the biotechnological industries have lagged behind others in plant automation (Morris *et al*,1991).

1.1. Industrial Fermentation

Bacteria, yeasts and moulds (microorganisms) are produced in fermenters in order to yield a product which could be the microorganism itself or a product produced by the microorganism during the fermentation. The layout of a typical fermenter is shown in Figure 1.1.

The fermenter can be operated in three different modes; continuous, fed-batch and batch. The fermenter is operated in continuous mode when both the feed and outlet flow rates are nonzero (with the outlet flow rate usually the same as the feed flow rate). If the outlet stream's flow rate is zero then the fermenter is operated in the fed-batch mode whereas if both the feed and outlet stream's flow rates are zero then the fermenter is operated in the batch mode.

In aerobic fermentations the microorganisms require oxygen and produce carbon dioxide. For these systems air is bubbled through the fermenter using a sparger, and an agitator ensures good mixing and that a high mass transfer rate of the oxygen exists. On-line probes are used

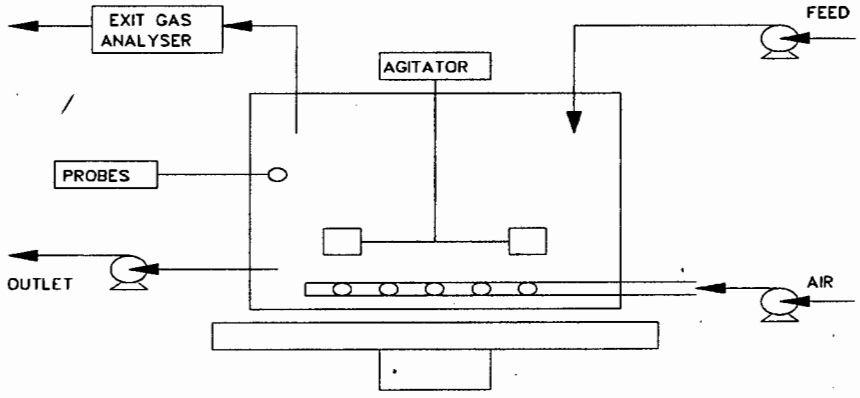

Figure 1.1. Layout of a general fermenter.

to determine the pH, temperature, level of dissolved oxygen and the progress of the fermentation in order to determine the control action required.

1.2. Control Required

The feed flow rate in continuous and fed-batch fermentations needs to be regulated in order to maximise the production of the microorganisms (biomass) and/or formation of the product. Usually the product is not the biomass itself and a trade-off is required in order to achieve a high production of product while still maintaining good cell growth.

If the microorganism is aerobic then a specific dissolved oxygen (DO) level must usually be maintained in the fermenter. This is to ensure optimum growth and performance. In the case of fed-batch lysine fermentation, a DO level of more than 10% is to be maintained (Kiss and Stephanopoulos,1991), whereas a DO level of 30% is suggested for continuous systems (Kiss and Stephanopoulos,1992; Coello *et al*,1992). The DO level is generally determined on-line via a DO probe and is maintained by either manipulating the air feed rate to the bioreactor and/or by manipulating the agitator speed or by a combination of both.

The pH should usually also be maintained at a given level to ensure optimum growth and performance of the microorganisms. If the pH deviates from the desired value, the

microorganism's proteins will start denaturing and larger pH deviations will ultimately lead to the death of the cells. The pH is controlled via the addition of a base to the fermenter. Kiss and Stephanopoulos (1991) suggest a pH of 7 for the fermentation of lysine and maintain this pH by the addition of gaseous ammonium hydroxide in the sterile air feed whereas Coello *et al* (1992) maintain the pH level for the lysine fermentation through the addition of sodium hydroxide.

An optimum fermenter temperature usually has to be maintained by manipulating the flow of coolants. If the temperature is raised, the cells will have a higher specific growth rate, however, the proteins will start denaturing. An optimum temperature of 30°C is suggested for lysine fermentation (Kiss and Stephanopoulos,1991; Coello *et al*,1992).

The pH, temperature and dissolved oxygen level in a fermenter are usually effectively controlled using simple analog controllers (San and Stephanopoulos,1984). Therefore only the regulation of the substrate addition is investigated in this thesis.

Bioprocess systems are nonlinear processes. Also, the concentrations of biomass, substrate and product (fermenter states) are not usually measurable on-line and are only observable through estimation techniques (Stephanopoulos and San,1984). The concentrations can be determined by doing off-line analysis of samples. However, the off-line analysis usually takes a long time and therefore the information is not available at the time at which it is needed.

1.3. Project Scope

The fermentation of lysine is an emergent industry in South Africa and AECI has recently built a R300-million plant in Umbogintwini to produce 10,000 tpa of lysine (Kenyon,1993). It was therefore decided to focus on the fermentation of lysine which is an important biological commodity used mainly as an amino acid feed supplement for livestock. The optimal control of lysine has been covered by many authors (Ohno *et al*,1978; Modak *et al*,1986; Lim *et al*,1986; Kiss and Stephanopoulos,1986). These optimal control techniques are, however, only formulated for the ideal fermentation model and therefore do not include the effects of model uncertainty.

Konstantinov and Yoshida (1992) point out that the great complexity and uncertainty of fermentation processes requires sophisticated logic of operation which cannot easily fit into the mathematical framework of the traditional control approach, and that knowledge-based control (which includes fuzzy control) is an alternative which is well suited to bioprocess systems due to its ability to deal with the complexities and uncertainty related to biological processes. The knowledge-based approach has already been applied to the control of processes such as waste water treatment, the production of biomass (both yeast and bacterial), enzymes, antibiotics, vitamins and amino acids (Konstantinov and Yoshida,1992). It was decided therefore to apply fuzzy control to the fed-batch fermentation of lysine and also of penicillin which closely resembles the fermentation of lysine.

The objectives of this thesis are therefore:

- To review biocontrol and estimation strategies presented in the literature.
- To investigate the use, operation and properties of fuzzy controllers.
- To investigate the differential equations describing the fed-batch fermentation of lysine and penicillin (two similar fermentations) and to determine the characteristics of the fermentation.
- To identify suitable controlled and manipulated variables for each fed-batch fermentation investigated.
- To apply fuzzy control to fermentations in both a regulatory and a supervisory capacity and to compare the results to standard control techniques.
- To investigate the effect of model uncertainty.

The dissertation is organised as follows:

Chapter 2 contains a summary of the different control and estimation techniques used for bioprocess control.

Chapter 3 introduces the concept of a fuzzy set and fuzzy operations. The fuzzy controller's operation is also investigated in terms of the fuzzy operations that are executed. From the literature it was found that a specific fuzzy controller is in fact a PI controller with a nonlinear gain. By investigating the resulting nonlinear PI controller, it was possible to propose a new controller parameter. The new fuzzy controller was implemented on a first, second and third order system and the resulting input and output responses were compared to that of a standard PI controller.

Chapter 4 contains the differential equation models of penicillin and lysine together with the results obtained from open loop simulations. These models are used in Chapter 5 to evaluate the control strategies developed. Also described is how *Corynebacterium glutamicum*, the bacteria responsible for the production of lysine, is able to overproduce lysine through mutation of the bacteria and through metabolic manipulation.

Chapter 5 deals with the application of fuzzy controllers to the control of both penicillin and lysine fermentations. The fuzzy controller which was defined in Chapter 3 is used for the regulatory type control of both lysine and penicillin production. Here the substrate level in the fermenter is controlled about a preset trajectory by manipulating the substrate feed flow rate. Heuristic type fuzzy control is thereafter used for the production of lysine. Here no trajectory is set *a priori*, but instead the controlling rules are obtained from intuition and by studying the batch fermentation.

By performing a sensitivity analysis on the placing of the fuzzy sets, it is found that the performance of the fuzzy controller depends to a great extent on the location of the sets. The effect of model uncertainty is examined by varying a coefficient in the specific growth rate equation. The disturbance rejection capabilities of the fuzzy controller are also investigated.

Chapter 6 contains a summary of conclusions drawn from the thesis.

2. CONTROL AND ESTIMATION OF BIOSYSTEMS

2.1. Optimal Feeding Profiles

Optimal substrate and/or nutrient profiles may be determined using a dynamic model of the fed-batch fermentation through optimisation of a cost function subject to certain constraints on the system. The determination of the optimal feed profile has been dealt with by numerous authors for both open-loop control and for feedback control systems.

2.1.1. Open-loop control

2.1.1.1. Optimal control theory

The optimal input trajectory is determined by introducing an objective function which needs to be minimised while the system is subjected to constraints. The general form of the modelling equations is summarised as follows (Ray,1981):

$$\dot{\underline{x}} = f(\underline{x},\underline{u}) \qquad \underline{x}(0) = \underline{x}_0$$

and the objective function is

$$I[\underline{u}(t)] = G(\underline{x}(t_f)) + \int_0^{t_f} J(\underline{x},\underline{u})dt$$

with a bounded input constraint

$$\underline{u}_* \leq \underline{u}(t) \leq \underline{u}^*$$

To solve this problem, the Hamiltonian, H, is introduced

$$H = J(\underline{x},\underline{u}) + \underline{\lambda}^T f(\underline{x},\underline{u})$$

where λ is the time-dependent Lagrange multiplier defined by

$$\frac{d\underline{\lambda}^T}{dt} = -\frac{\partial H}{\partial \underline{x}}$$

and

$$\lambda_i(t_f) = \frac{\partial G}{\partial x_i}$$

The necessary condition for minimising the objective function is that

$$\frac{\partial H}{\partial \underline{u}} = 0$$

on the unconstrained portion of the process input's trajectory and that the Hamiltonian be a minimim on the constrained portion of the path. It is also necessary that the Hamiltonian maintains a contant value along the optimal trajectory which should be zero if the final time, t_f is unspecified (Ray,1981).

2.1.1.2. Applications

In fed-batch fermentations nutrients are fed to a bioreactor with no output streams. The optimisation problem involves finding the optimal substrate feed rate for the fed-batch fermentation. The modelling equations are usually formulated with the assumption that biomass growth and product formation are limited only by one substrate.

The performance index is a function of the final fermenter conditions while constraints are imposed on the final fermenter volume as well as the substrate feed rate. The general form of the modelling equations is summarised as follows:

$$\dot{\underline{x}} = f(\underline{x},F)$$

where the state vector is given by

$$\underline{x} = (x,s,p,V)^T$$

with F = the substrate feed rate
x = the biomass concentration
s = is the substrate concentration
p = is the product concentration
V = is the volume of the fermentation broth

The following constraints are usually imposed on the fermenter volume and substrate feed rate:

$$V(t_f) = V_f$$
$$0 = F_{min} \leq F(t) \leq F_{max}$$

The objective is to determine the optimum feed profile, F(t), which will minimise a performance index which depends on the final outcome of the fermentation:

$$I[F] = G(\underline{x}(t_f)) + \int_0^{t_f} J(\underline{x},F)\,dt$$

where t_f is the final fermentation time.

In their study, Modak *et al* (1986) used the following differential equation model:

$$\frac{d}{dt}(xV) = \mu xV \qquad x(0) = x_o$$

$$\frac{d}{dt}(sV) = -\sigma xV + s_F V \qquad s(0) = s_o$$

$$\frac{d}{dt}(pV) = \pi xV - kpV \qquad p(0) = p_o$$

$$\frac{d}{dt}(V) = F \qquad V(0) = V_o$$

where μ is the specific cell growth rate
σ is the substrate consumption rate

π is the product formation rate

k is the degradation constant of the product

s_F is the feed substrate concentration

The cell mass balance states that the change in cell mass in the fermenter is equal to the growth rate of the cells. The amount of substrate in the fermenter depends on the amount of substrate taken up for utilisation by the cells and the amount which enters the fermenter in the feed stream. The amount of product depends on the quantity produced by the cells and the amount of degradation which may occur to the product. The rate of change in fermenter volume is equal to the rate of the feed stream.

The specific growth rate (μ), the specific substrate utilisation rate (σ) as well as the specific product formation rate (π) are either implicit or explicit functions of the substrate concentration. For the fermentation of antibiotics such as penicillin and amino acids such as lysine the following dependencies can be assumed [$\mu=\mu(s)$, $\sigma=\sigma(\mu)$, $\pi=\pi(\mu)$] (Modak *et al*,1986).

Modak *et al* (1986) minimised the performance index by minimising the Hamiltonian and were able to determine that the feed rate, F, is either to be set at F_{max}, $F_s(t)$ (termed the singular feed rate) or $F_{min}=0$ depending on the state of the system. The switching times at which a change is made from one feed regime (F_{max}, $F_s(t)$ or F_{min}) to the next as well as the singular feed rate were also determined.

Figure 2.1 shows their typical feed profiles for a class of fermentations which includes both lysine and penicillin fermentation.

Modak *et al* (1986) apply this optimisation technique to the fermentation of penicillin and to the production of bacterial cell mass in a subsequent paper (Lim *et al*,1986). San and Stephanopoulos (1989) also investigated the optimisation of fed-batch penicillin fermentation but utilised the feed concentration as the manipulated variable in contrast to Modak *et al* (1986) who utilise the substrate feed rate. Ohno *et al* (1976) utilise Green's theorem to maximise the production of lysine as opposed to Modak *et al*'s (1986) use of the Hamiltonian.

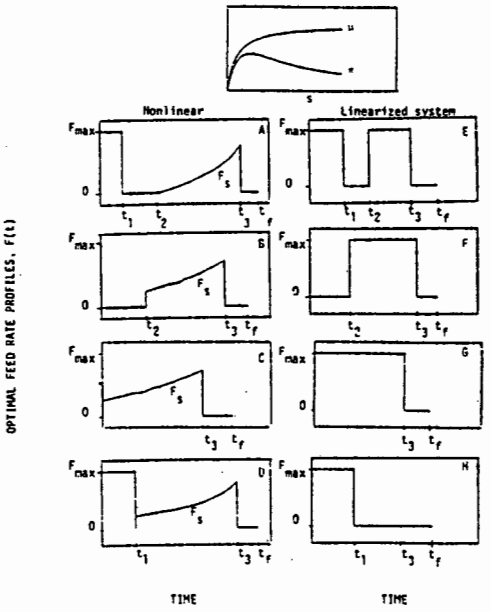

Figure 2.1. Typical substrate feed profiles: (A) low s_o and x_o, (B) high s_o, (C) appropriate s_o and x_o, (D) high x_o and low s_o (Modak *et al*,1986).

Menawat *et al* (1987) investigated the use of optimal control in the production of bakers' yeast (*Saccharomyces cerevisiae*). The fermentation was described by the following differential equation model:

$$\frac{d}{dt}(xV) = \mu xV$$

$$\frac{d}{dt}(sV) = FS_F - \frac{\mu xV}{Y_{x/s}} - m_s xV$$

$$\frac{d}{dt}(V) = F$$

where m_s is the rate of substrate consumption for maintenance

$Y_{x/s}$ is a yield factor (yield of biomass from substrate)

μ is given by

$$\mu = \frac{\mu_m s}{K_1 + s} \frac{K_2}{K_2 + s}$$

and μ_m, K_1, K_2 are constants.

The product in this case is the biomass (yeast) and the objective is to maximise the amount of biomass produced at the end of the fermentation giving a performance index (in

dimensionless terms) of

$$J[u(\cdot)] = \inf_{u(\cdot)}\{X(\tau_0)v(\tau_0) - X(\tau_1)v(\tau_1)\}$$

where

$$X = \frac{x}{K_2 Y_{x/s}} \qquad v = \frac{V}{V_0}$$

$$\tau = \mu_m t \qquad u = \frac{F}{\mu_m V_o}$$

with subscripts 0 and 1 indicating the conditions at the start and end of the fermentation respectively.

Menawat *et al* (1987) were able to obtain the optimal state trajectories via a heuristic argument. These results were also quantified by minimising the Hamiltonian. The heuristic solution is obtained by observing that certain simplifications can be made to the mass balance equations for the fermentation of baker's yeast if the fermenter is producing cells at the maximum possible rate at all times. The mass balance equations are reworked resulting in analytical expressions for the trajectories of the biomass concentration, fermenter volume and the feed stream flow rate.

Chen and Hwang (1990) investigated optimal control in the fermentation of bakers' yeast using an 'on-off' controller. The flow rate of the feed is either at its maximum or at its minimum (no flow of substrate into the fermenter) throughout the fermentation. The authors were able to show that the on-off controller yields comparable results to that obtained by Menawat *et al* (1987).

Ohno *et al* (1978) and Modak and Lim (1992) investigated the optimum sequence of operating modes (batch, continuous or fed-batch) during a fermentation i.e. where the mode of operation of the bioreactor is not set *a priori* and the fermentation is allowed to move from one mode into the next. Ohno *et al* (1978) determined the optimal sequencing of fermentation modes for lysine fermentation and found that the best operating mode depends on the fermenter's volume and the fermentation time. They found that; (i) when the operating

time becomes 'long' a batch/fed-batch stage followed by continuous fermentation is optimal, and (ii) when the fermenter volume becomes 'large' a fed-batch mode is optimal.

Modak (1993) was able to investigate which control variable (the volumetric substrate feed rate, the substrate concentration in the fermenter, the substrate concentration in the feed and the mass flow rate of the feed) would lead to the optimisation of a fed-batch fermentation. He found that the last three control variables took into account neither the dependence of the specific rates (specific growth rate, specific substrate utilisation rate and the specific product formation rate) on the concentrations of cell mass and product, nor substrate consumption dynamics. These three control variables were found to lead to suboptimal operating policies while use of the substrate feed rate as the control variable yielded the best performance. These deductions were made using models of a cell mass production and an alcohol production example (using bakers' yeast).

2.1.2. Feedback control

The control techniques presented in Section 2.1.1 all involve open loop control of the biosystems. These approaches depend critically on the quality of the mathematical models used to describe the fermentation dynamics. A closed loop technique is therefore more advantageous as it will be able to attenuate the model uncertainties.

Modak and Lim (1987) present a feedback optimisation technique based on the method of Modak *et al* (1986) and Lim *et al* (1986). The authors apply the feedback controller to the fermentation of lysine where the final fermentation time is free and not set *a priori*. The singular feed rate, F_s (discussed in Section 2.1.1), is determined to be an explicit function of the fermenter's states (ie the biomass concentration, the substrate concentration, the product concentration and the fermenter volume). Modak and Lim's (1987) approach is feedback in nature because the singular feed rate is calculated from the values of the fermenter states after each time step. Mathematically the authors were able to show a 2-5% improvement in the performance of the feedback controller over that of the open loop equivalent.

Kiss and Stephanopoulos (1991) utilised respiratory measurements as a metabolic activity indicator for the feed rate control of a lysine fermentation. The author's aims were to maximise the lysine yield and specific productivity. The authors noted that the formulation of an optimal feedback controller, such as that proposed by Modak and Lim (1987), would require extensive on-line bioreactor measurements of the fermenter's states. Due to the lack of technology and accuracy of on-line probes, the authors opted for a controller which utilises respiratory measurements (carbon dioxide evolution rate (CER) and oxygen utilisation rate (OUR)) as indirect indicators of the bacterial metabolic activity.

By investigating the results obtained from batch and continuous experiments, they determined the characteristics of the fermentation as a function of the OUR and CER. They then formulated a heuristic type controller for the fed-batch fermentation of lysine which utilises the on-line values of the OUR and CER to determine the feed flow rate of substrate that will optimise the formation of product.

2.2. Neural Networks

2.2.1. Theory

A neural network enables one to directly model nonlinear processes without a pre-specified detailed relationship. They are able to 'learn' to approximate large classes of nonlinear functions during a 'training' procedure done on the results archived from the process (Karim and Rivera,1992). While many artificial neural network architectures have been proposed, the feed forward network, commonly called backpropagation nets, has been predominant (Di Massimo *et al*,1992). A feed forward network is made up of neuron-like elements, called nodes, which are organised in layers as shown in Figure 2.2.

Scaled data enters the network via nodes in the input layer. From here the data are propagated to the output through the intermediate layer(s). Figure 2.3 is a representation of an individual node. The n inputs to the individual node, $(e_i)_{i=1,n}$, are each multiplied by their respective weighting functions, $(W_i)_{i=1,n}$. The total input, E, is formed, where E frequently

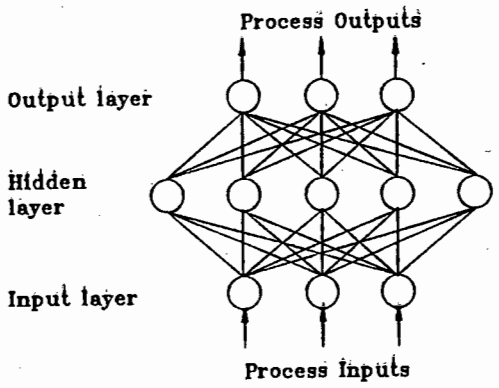

Figure 2.2. Feed forward network structure (Di Massimo *et al*,1992).

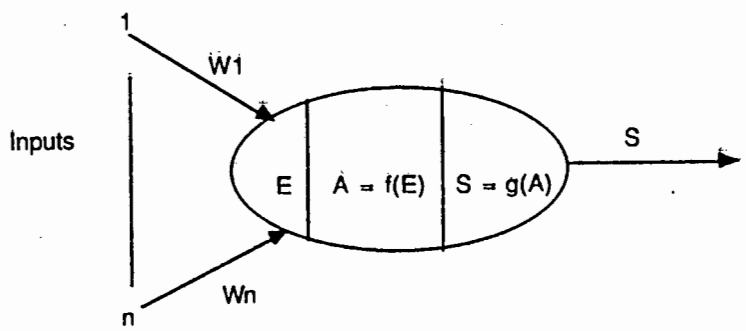

Figure 2.3. Functions of the individual node (Davalo and Naïm,1991).

takes one of the following forms (Davalo and Naïm,1991):

$$E = h(e_1,\ldots,e_n) = \sum_{i=1}^{n} W_i e_i \qquad (linear)$$

or

$$E = h(e_1,\ldots,e_n) = \sum_{i=1}^{n} (W_i e_i - a) \qquad (affine)$$

where a usually assumes a value of -1.

The state of the neuron, A, is then determined, where A is a monotonic and odd function and frequently takes one of the following forms (Davalo and Naïm,1991):

$$f(x) = \begin{cases} x & if\ u \le x \le v \\ u & if\ x < u \\ v & if\ x > v \end{cases}$$

termed the linear threshold function (SATUR) as shown in Figure 2.4(a), or

$$f(x) = a\frac{e^{kx} - 1}{e^{kx} + 1}$$

termed the sigmoid function as shown in Figure 2.4(b).

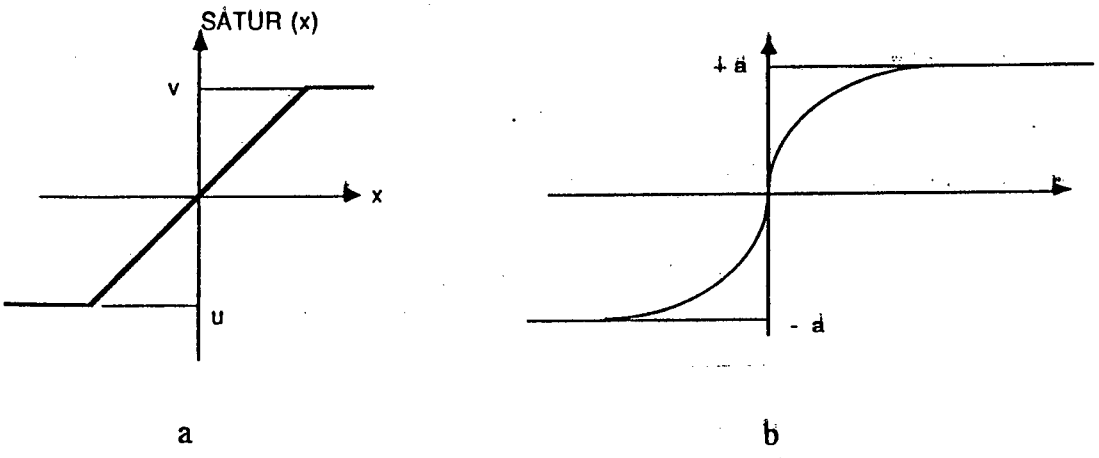

Figure 2.4. (a) Threshold and (b) sigmoid function (Davalo and Naïm, 1991).

Generally the output function, g, is taken to be the identity function therefore:

$$S = f(E) = A$$

The network is trained by successive presentations of input-output data pairs to the network obtained from previous process runs. For a network with n input neurons and m output neurons, the vector containing the inputs, $\underline{X}$, and the vector containing the desired outputs, $\underline{Y}$, are given by:

$$\underline{X} = (X_1, X_2, \ldots, X_n)^T$$
$$\underline{Y} = (Y_1, Y_2, \ldots, Y_m)^T$$

During the training period data is propagated forward through the network, which then adjusts its internal weights to minimise the error between the true output, $\underline{S}$, and the output obtained from the network, $\underline{Y}$. This is accomplished through the minimisation of the square

of the errors:

$$E(W) = \sum_{i=1}^{m} (Y_i - S_i)^2$$

The weights are modified accordingly when the data are presented for the k^{th} time (Davalo and Naïm, 1991):

$$W_{ij}(k) = W_{ij}(k-1) - e(k)d_iO_j$$

where d_i is calculated sequentially from the output layer to the input layer via

$$d_i = 2(S_i - Y_i)f'(I_i) \qquad (output\ layer)$$

$$d_i = \sum_h d_h W_{hi} f'(I_i) \qquad (hidden\ layers)$$

in which
- f is the sigmoid function used for each neuron
- f' is the derivative of f
- O_j is the output of neuron j
- I_i is the input of neuron i given by

$$I_i = \sum_j W_{ij}O_j$$

and
- e(k) is the step size used for the k^{th} iteration.

In order to supply the network with the underlying model, the data set has to be fairly rich in information. The most convenient method of testing the network is on data on which no training has been performed. An alternative routine for training the network is the chemotaxis algorithm which has been adapted by Di Massimo *et al* (1992):

1. Initialise each node's weights with small random values.
2. Use the set of weights to predict the network outputs.
3. Determine the cost function E_1 over the whole data set.
4. Generate a Gaussian distributed random vector.
5. Increment the weight vector with the random vector.
6. Determine the new cost function E_2 as in step 3.

7. If $E_2 < E_1$ retain the new weight vector, set $E_1 = E_2$ and go to step 5. If $E_1 \leq E_2$ go to step 4.

The advantage of this approach of determining network weights is that the problem of entrenchment in local minima may be avoided.

2.2.2. Applications

Di Massimo *et al* (1992) applied artificial networks to the modelling of fed-batch penicillin fermentation. The network was developed to provide estimates of the biomass concentration during the course of the fermentation. The on-line measurements of substrate feed, time and CER (carbon dioxide evolution rate) were selected as the network inputs, while biomass concentration was the network output. The network training was done off-line. It was found that although the biomass assay results were corrupted with a high level of noise, that the network was able to present the underlying characteristics of the fermentation under the two operating conditions investigated.

A neural network was also trained for the estimation of the penicillin concentration using the cell growth rate, biomass and broth age as the inputs to the network. This paper concentrated on 'modularisation' i.e. that one network predicted biomass and hence growth rate which then provides the information to the second network which predicts penicillin production. It is stated that a single network could be considered to carry out both tasks, but that a fault diagnosis investigation suggests that the modular approach may prove to be more effective.

Karim and Rivera (1992) investigated the use of neural networks in the anaerobic fermentation of ethanol utilising *Zymomonas mobilis* bacteria. The network was used to estimate the biomass, glucose and ethanol concentrations by utilising on-line measurements of the temperature, redox potential, percentage CO_2 gas in the exhaust gas and the optical density of the broth. The authors found that the neural network was able to estimate the biomass and ethanol concentrations relatively well but that the glucose estimation was not as good. They suggested that better state estimations might be obtained by including a measure of the acid production rate as an additional input to the network.

Linko and Zhu (1991) applied an artificial neural network to both the batch and the fed-batch fermentation of *Candida utilis*. The network uses current values of the fermenter states and a bias term to estimate the states one time step ahead. The data required for the training of the network was obtained by integrating a fermentation model. Latrille *et al* (1993) on the other hand utilised actual fermentation data to train a neural network for the batch fermentation of lactic acid.

2.3. Fuzzy Logic

Fuzzy estimators and fuzzy controllers are characterised by a rule base of if-then type linguistic rules which have usually been accumulated from a person with an intimate knowledge of the system and its control requirements.

The fuzzy estimator or controller is able to determine to what degree the antecedent (if) part of each fuzzy rule is met and hence determine the required estimation or control action required.

In this section only the applications of fuzzy controllers will be dealt with since the theory behind fuzzy sets and systems is presented in Chapter 3.

2.3.1. Applications

Tong *et al* (1980) successfully implemented fuzzy control on an activated sludge wastewater treatment process as shown in Figure 2.5.

Here standard P/PI controllers are used to maintain the dissolved oxygen (DO) in the aeration tank at the set point, and to maintain the recycle ratio at the set point. The fuzzy controller is used in a regulatory capacity in the control of the waste sludge rate and in a supervisory capacity to determine the change in the DO and the change in the recycle ratio set points.

Through practical experience gained operating the activated sludge plant, the authors were

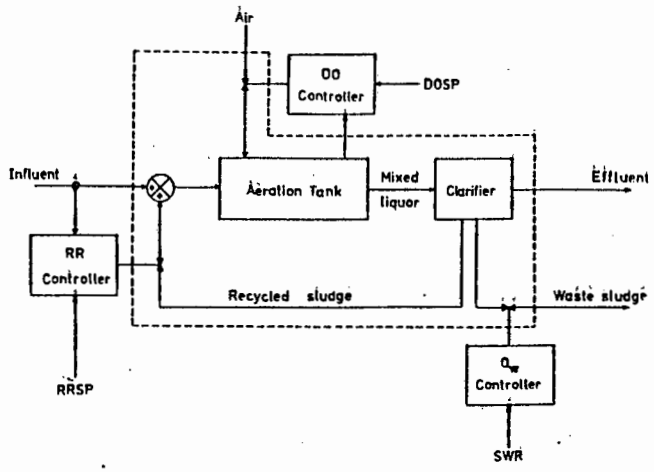

Figure 2.5. Layout of a wastewater treatment process (Tong *et al*,1980).

able to determine a number of fuzzy controller inputs. These inputs include the levels of suspended solids in the effluent, in the sludge leaving the aeration tanks and in the recycle sludge as well as the total biological oxygen demand.

While they were able to successfully implement a fuzzy controller to the activated sludge plant, it was noticed that some rules, which were not expected to be utilised that often, were being activated at low input levels which suggested the use of a threshold for the rule activation.

Postlethwaite (1989) investigated a fuzzy state estimator for the fed-batch fermentation of alcohol using bakers' yeast. This estimator was used in the determination of the yeast's growth rate by utilising readings of the fermenter dilution rate and the concentrations of the substrate, inhibitor and ethanol. The author used seven fuzzy rules in the formulation of the estimator and found that a good correlation existed between the actual and estimated values.

Whitnell *et al* (1993) utilised a fuzzy predictor to determine the fermentation time required in a commercial brewery. The fuzzy estimator determined the state of the batch and made an inference as to whether the process trajectory was normal or abnormal and then predicted the 'most likely' time of completion.

The fuzzy predictor had three predictive elements: an initial prediction of fermentation time

at the time of yeast pitching, an estimate of VDK (vicinal diketones) level during the fermentation and a revised estimate of fermentation time based on the VDK removal rate in the final stage of the fermentation. The standard way of detecting the VDK level was by doing random samples. The VDK analysis is time consuming and therefore the second predictive element was used to predict the VDK level as the fermentation progressed and thus give the quality control laboratory better direction as to the appropriate time to sample for the final VDK analysis. When the predicted VDK level reached 0.27 ppm, the final fuzzy predictive element was used to revise the estimate of fermentation time based on the predicted time to reach a VDK level of 0.05 ppm (the quality control specification for releasing the product to ageing).

Simutis *et al* (1992) also investigated the use of fuzzy logic in the beer brewing process. The authors present six models for the fermentation. Each one of these models is designed to represent a given phase of the fermentation as they consider it is easier to describe a particular fermentation phase rather than the whole fermentation with a single mathematical model. An extended Kalman filter is used to estimate the fermenter's states while updating the relevant model's parameters. Fuzzy logic is used to ascertain which model should be utilised according to the current development of the fermentation.

Konstantinov and Yoshida (1989) present a 'physiological state control approach' for fermentation systems. During a batch, fed-batch or continuously operated fermentation, the cell culture will pass through many physiological states. Each physiological state (PS) is divided into 'physiological situations' (PSNs) in which the cell population expresses stable characteristics. The purpose of the PS control is to maintain the cell population in the desired PSN (optimal PSN), or to force the population back to this PSN. The PS control system therefore is able to recognise the current PS and switch to the relevant PSN control strategy to calculate the control action required.

A fuzzy rule base is used, incorporating a weighting system, to determine the current PS. The weights are determined by a learning procedure using a sufficiently large data base. A multivariable linear regression procedure, using the least squares technique, is used in the calculation of the weights.

For this type of control, no mathematical model was required. The structure is flexible and applicable to different types of fermentation processes. However, the main disadvantage is that in some PSNs, the control method cannot guarantee the desired transfer to the optimal PSN. It is only able to provide the best condition for the transition.

2.4. Adaptive Control

Montague *et al* (1986a and b) applied an adaptive control technique to the fed-batch fermentation of penicillin. The biomass concentration was estimated from measurements of the carbon dioxide production rate and the fermenter volume by utilising an extended Kalman filter. A generalised predictive controller (GPC) with a long-range receding-horizon predictive-type algorithm was utilised. The controller assumes a process model of the controlled autoregressive integrated moving average (CARIMA) type. The estimated value of the biomass concentration is used as the controlled variable and the substrate feed rate as the manipulated variable.

In a penicillin fermentation a large substrate concentration overshoot should be avoided as it leads to the repression of the biomass production. The adaptive controller was applied to an actual penicillin fermentation and found to yield a lower overshoot in the fermenter substrate concentration than a PI controller due to a milder control input. Overall, the performance of the adaptive control technique was found to be better than that of a PI controller.

Dochain and Bastin (1984) recognised the difficulty of choosing an analytical expression for the bacterial growth rate and formulated an adaptive control technique which does not require the choice of the expression. The bacterial growth system is approximated by a discretised time-varying model while recursive least-squares estimates are obtained for the parameters. The parameter estimator was combined with a discrete-time minimum variance adaptive controller for the control of the substrate concentration. This adaptive control technique was applied to an anaerobic digestion process and shown to be successful.

Adaptive control has also been utilised for the control of the dissolved oxygen in an activated sludge process (Ko *et al*,1982) as well as for pH control in a wastewater treatment process (Mahuli *et al*,1993).

3. FUZZY LOGIC AND CONTROL

3.1. History and Applications

Fuzzy set theory was originally proposed by Zadeh in his seminal papers published in 1965, to formalise qualitative concepts that have no precise boundaries (Kandel and Langholz,1993; Pedrycz,1993).

Fuzzy controllers have become popular in Japan where they have been successfully used in household appliances such as washing machines and auto-focus systems in video equipment. Fuzzy logic has also been applied to information technology oriented fields such as databases and expert systems in the financial sector (Driankov *et al*,1993). Fuzzy control has also been applied in the field of industrial process control, an example being cement kiln operation.

3.2. Fuzzy Sets

A set in the traditional sense has definite boundaries which will determine whether or not a given object is a member thereof (lies within the bounds of the set). Therefore, a given object can be assigned a grade of membership (GM) of 1 if it lies within the set and 0 if it does not lie within the set. Fuzzy sets, however, do not have definite boundaries and values are assigned grades of membership ranging between 0 and 1 depending on the object's degree of membership in the fuzzy set. Higher values indicate a higher degree of membership in the given fuzzy set.

For a given fuzzy set, A, a value x (in a universal set X) will have a grade of membership in A given by $\mu_A(x)$. As an example, let the universal set, X, be the range of daytime temperatures with the fuzzy sets A = '*cold*', B = '*hot*' and C = '*very hot*'. Figure 3.1 shows how these fuzzy sets could be arranged over a temperature range of 0 - 50°C.

The construction of these fuzzy sets is subjective and depends on one's perception of *cold*, *hot* and *very hot*. From Figure 3.1 it can be seen that temperatures below 20°C are no longer

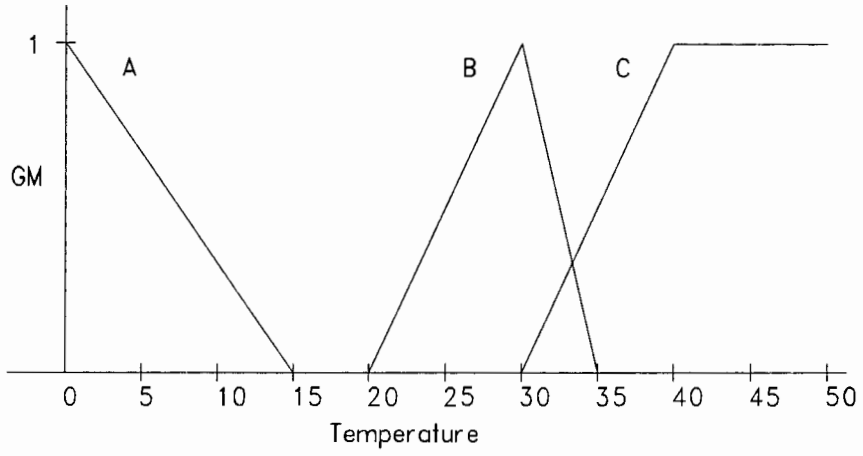

Figure 3.1. Fuzzy sets describing daytime temperatures (°C).

classified as being '*hot*', whereas temperatures below 15°C start being classified as '*cold*'. It can also be seen that a given temperature can be a member of more than one fuzzy set i.e. 32°C has a higher grade of membership in the fuzzy set '*hot*' ($\mu_B(32°C)=0.6$) than it does in the fuzzy set '*very hot*' ($\mu_C(32°C)=0.2$). The set D = '*40°C*' is a crisp set with the temperature of 40°C as its only member and this temperature has a grade of membership of 1. All other temperatures would have a grade of membership in D of 0. Therefore it can be seen that crisp sets are a special case of fuzzy sets.

As another example, the membership function for the fuzzy set (A) of *real numbers close to 0* could be defined as (Klir and Folger,1988):

$$\mu_A(x) = \frac{1}{1 + 10x^2}$$

The graph of this function is shown in Figure 3.2.

3.3. Fuzzy Operations

If the membership grade of each element of the universal set X in fuzzy set A is less than or equal to its membership grade in fuzzy set B, then A is called a **subset** of B. Therefore, if

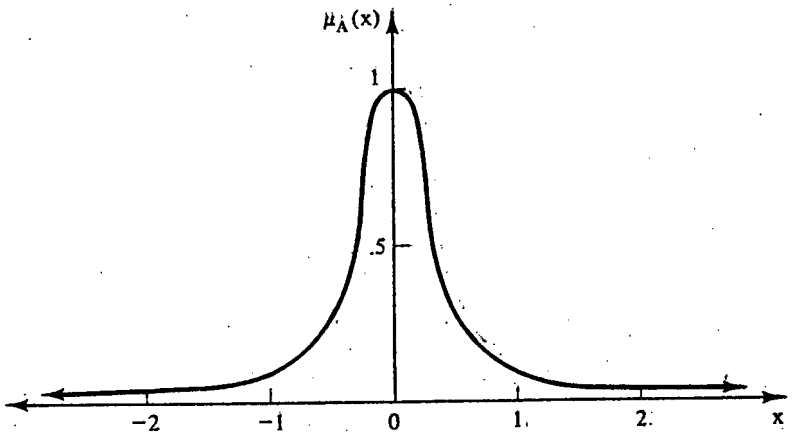

Figure 3.2. Membership function for fuzzy set of real numbers close to zero (Klir and Folger ,1988).

$\mu_A(x) \leq \mu_B(x)$ for every x ϵ X, then $A \subseteq B$. The fuzzy sets are called **equal** if $\mu_A(x) = \mu_B(x)$ for every element x ϵ X and is denoted by $A = B$. The **complement** of a fuzzy set is defined by:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$$

The **union** of two fuzzy sets A and B is a fuzzy set $A \cup B$ (often denoted as $A \vee B$ i.e. A 'or' B) such that the grade of membership of a value x in $A \cup B$ is given by:

$$\mu_{A \cup B}(x) = \max[\mu_A(x), \mu_B(x)]$$

while the **intersection** of two fuzzy sets A and B is a fuzzy set $A \cap B$ (often denoted by $A \wedge B$ i.e. A 'and' B) such that the grade of membership of a value x in $A \cap B$ is given by:

$$\mu_{A \cap B}(x) = \min[\mu_A(x), \mu_B(x)]$$

3.4. Fuzzy Logic

In this sub-section it will be explained how one can compute the membership function representing the conclusion 'Y is B' for the fuzzy rule given by 'if X is A then Y is B'.

As an example the antecedent part of the above rule, X, will be chosen as the **error** in a system's output (y_{set} - y) and the fuzzy set, *A*, is chosen to be *Negative*. The consequent part of the rule, Y, is chosen to be the **change** in the controller **output** for this example and *B* will be chosen to be *Negative*. The fuzzy sets are shown in Figure 3.3.

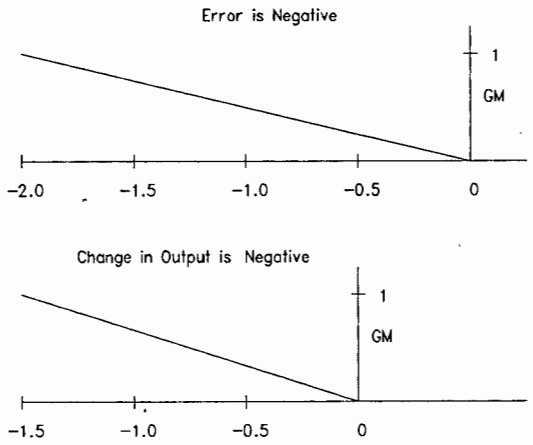

Figure 3.3. Fuzzy sets used for the error and change in output.

The statement 'X is *A*' is interpreted as the grade of membership of the value X in the fuzzy set, *A*, i.e. given by $\mu_A(X)$. An error of -1 has a grade of membership of 0.5 and is interpreted as being less *negative* than an error of -1.5 which has a grade of membership of 0.75. Similarly the statement 'Y is *B*' is given by $\mu_B(Y)$. The closer the change in the output value is to -1.5, the more it is defined to be *negative*.

The fuzzy rule:

'if the **error** is *negative* then the **change** in the **output** is *negative*'

is interpreted as the output fuzzy set given by (Driankov *et al*,1993):

$$\mu_{rule}(Y) = \{\mu_A(X)\} \cap \{\mu_B(Y)\} = \min[\mu_A(X), \mu(Y)]$$

calculated for the current value of the error and for certain values of the change in output that are a member of the *negative change in output* fuzzy set, [-1.5,0] yielding an output set as in Figure 3.4, i.e one is determining how best does the rule match values of Y for a given

X. Therefore for a given controller input, X, one obtains a fuzzy rule set covering the output range from which the final controller output response is obtained using the weighted mean as discussed in Section 3.5.3.

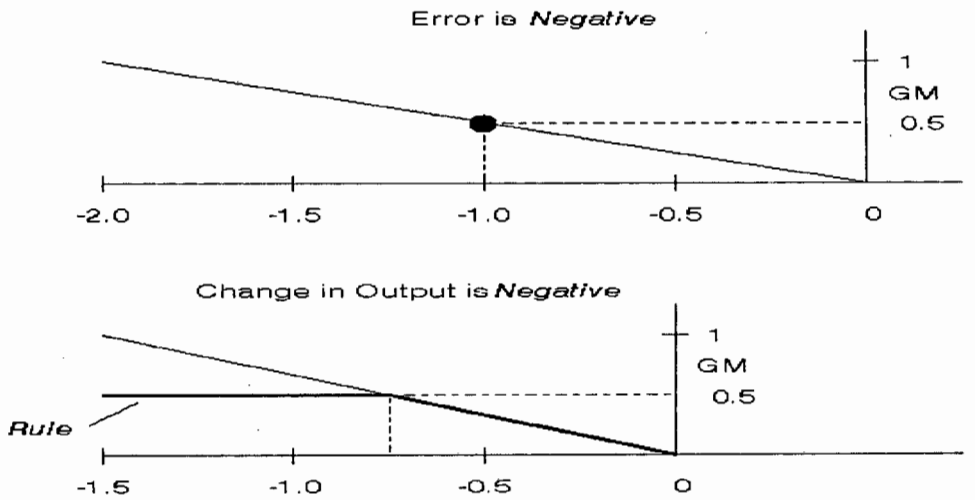

Figure 3.4. Determining the resultant fuzzy set for an if-then type rule using an error of -1.

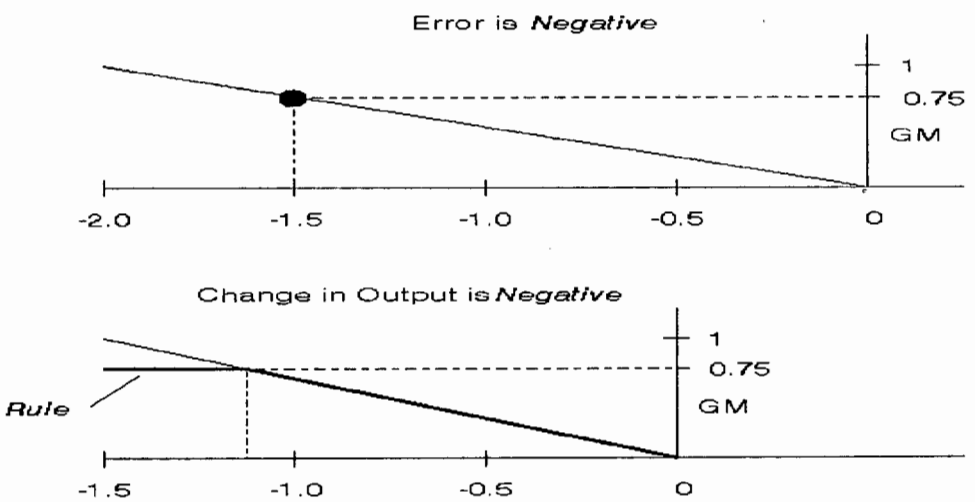

Figure 3.5. Determining the resultant fuzzy set for an error of -1.5.

In Figure 3.4 it can be seen that the output fuzzy set has been clipped at a value of 0.5 which corresponds to the grade of membership of an error of -1 in the *negative error* fuzzy set, i.e. the rule is 'fired' by 50%. This means that for an error that is only 50% *negative*, that the *negative change in the output* fuzzy sets' grade of membership may not assume a value which

is higher than 50% i.e. the change in the output's membership is at most 50% *negative*. However, if an error which is more *negative*, i.e -1.5, is used it results in a grade of membership of 0.75. Now the *negative change in output* fuzzy set is clipped at 0.75 as shown in Figure 3.5. Therefore the *negative change in output* fuzzy set is able to attain a higher limit of 0.75, i.e. the rule is now 'fired' by 75%. Therefore the consequent part of the rule has a higher degree of applicability if the antecedent part of the rule has a higher degree of applicability i.e. the more negative the error, the more negative the change in output will be.

3.5. Fuzzy Control

A simple example is presented in this section to help with the understanding of the equations used to set up the fuzzy controller. The example consists of a tank of water with two inlet streams (one hot and one cold) and one outlet stream as shown in Figure 3.6. The outlet stream is the tank overflow and this ensures that the volume of water in the tank is always constant. The aim of the fuzzy controller is to maintain a set temperature in the tank despite disturbances entering the system (cold stream). The temperature in the tank is maintained by controlling the flow of hot water into the tank.

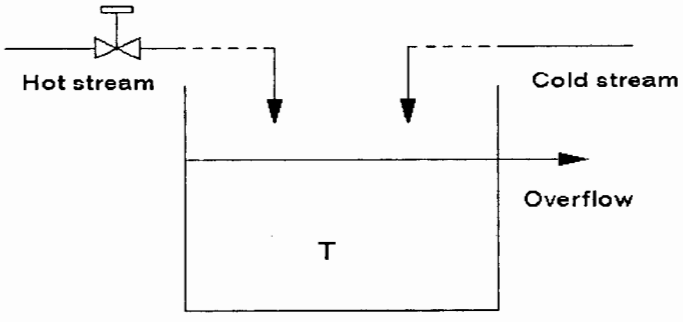

Figure 3.6. Diagram of example for fuzzy control.

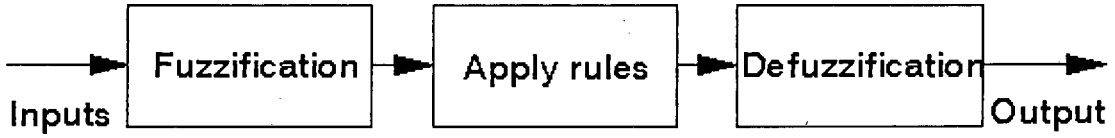

Figure 3.7. Fuzzy controller structure.

Figure 3.7 shows the fuzzy procedure which is employed in fuzzy controllers. The first step is to fuzzify the input to the controller (process outputs). This establishes the grades of membership of the input values in each input fuzzy set. The second step is the utilization of these fuzzified inputs, with the aid of the fuzzy rule base, to determine a fuzzy output. The last step is the defuzzification of the output which yields the controller output and input to the process.

3.5.1. Fuzzification

Often the input to the controller is first scaled before entering the fuzzification step. Triangular shaped membership functions are often used which can be defined by three values (a,b,c). The grade of membership of a value, x, in this fuzzy set, A, is given by (Postlethwaite,1989):

$$\mu_A(x) = \begin{cases} 0 & x \le a \\ (x-a)/(b-a) & a \le x \le b \\ (c-x)/(c-b) & b \le x \le c \\ 0 & x \ge c \end{cases}$$

where: a and c represent the left and right extremes of the fuzzy set

respectively

b represents the value of x at which $\mu_A(x) = 1$ as shown in Figure 3.8.

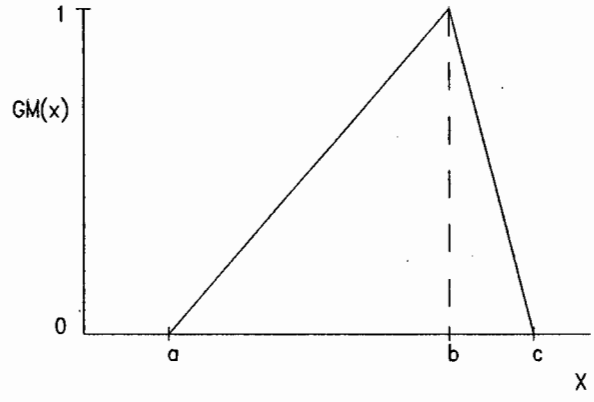

Figure 3.8. Triangular shaped membership functions.

Figure 3.9 shows an example of a fuzzy partitioning. Here the universe, X, is divided

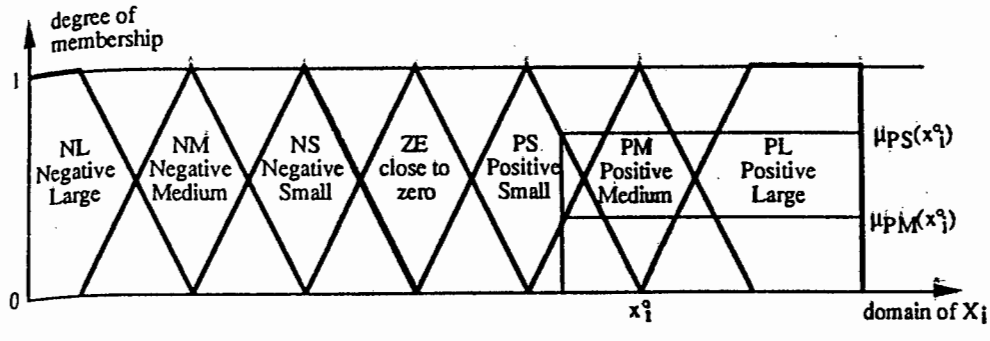

Figure 3.9. Fuzzy partition (Dubois and Prade,1993).

(partitioned) into many fuzzy sets which are usually assigned linguistic names such as *PM* (*positive medium*), *NL* (*negative large*), *ZE* (*close to zero*) etc. These fuzzy sets are assigned so that the entire range of the input variable will be covered by at least one fuzzy set. In the

tank example two controller inputs will be used - **error** (y_{set} - y) in the tank's temperature and the change in the error (**rate**). Each of the two controller inputs is assigned two fuzzy sets (by choice) - *negative* (*N*) and *positive* (*P*). The controller output is the **change** in the hot stream's **flow rate** with its range divided into three fuzzy sets (*negative*, *zero* and *positive*) as shown in Figure 3.10.

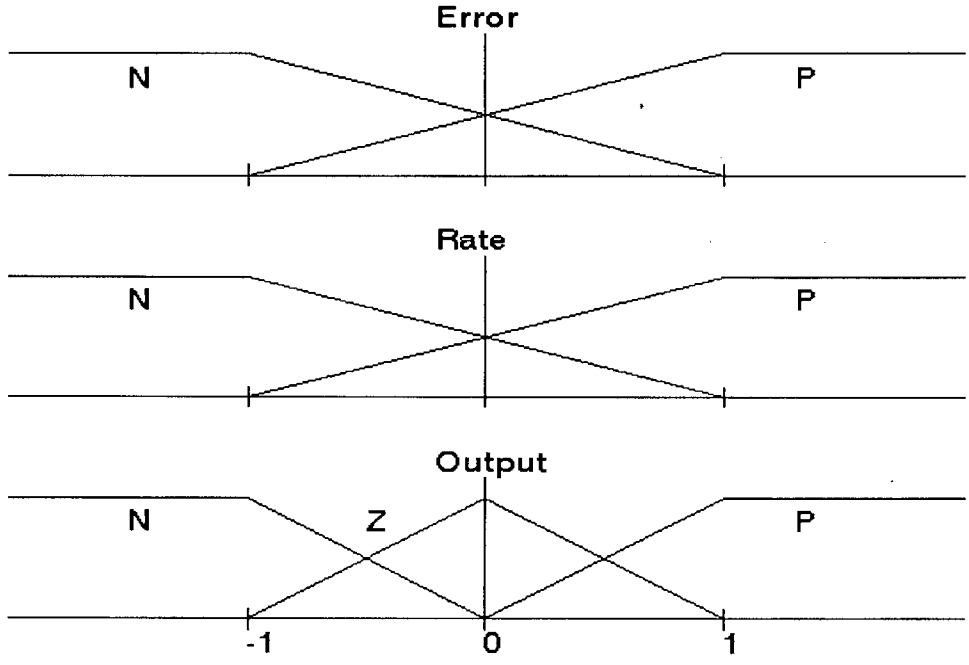

Figure 3.10. Fuzzy partitions used for the example.

During the fuzzification procedure the membership of the controller input value is determined for each input fuzzy set in which it appears, i.e. determining to what degree a value is a member of the particular fuzzy set. Therefore for a given error in the tank temperature, for example 0.8, the grades of membership over the error partitioning are given by $\mu_{EN}(0.8) = 0.1$ and $\mu_{EP}(0.8) = 0.9$ (where *EN* and *EP* stand for *error negative* and *error positive* respectively). Similarly if the rate has a value of -0.4 then the grades of membership over the rate partitioning are given by $\mu_{RN}(-0.4) = 0.7$ and $\mu_{RP}(-0.4) = 0.3$. Although only two controller inputs are used in this example, it should be noted that more inputs can be utilised.

3.5.2. Fuzzy rule base

A fuzzy rule base is a collection of antecedent-consequent type rules which are often obtained from someone who has an intimate knowledge of the system and can best impart their knowledge in the form of linguistic rules i.e.

if **condition 1** is met and **condition 2** is met then do **action B.**

The rules can also be set up by doing an analysis of the dynamics of the system which is to be controlled. The general control rule base's structure is given by:

$$
\begin{aligned}
&\textit{if } A_{1,1} \textit{ and } A_{1,2} \textit{ and } \ldots A_{1,nf} \textit{ then } B_1 \\
&\textit{or} \\
&\qquad \ldots \\
&\textit{or} \\
&\textit{if } A_{i,1} \textit{ and } A_{i,2} \textit{ and } \ldots A_{i,nf} \textit{ then } B_i \\
&\textit{or} \\
&\qquad \ldots \\
&\textit{or} \\
&\textit{if } A_{m,1} \textit{ and } A_{m,2} \textit{ and } \ldots A_{m,nf} \textit{ then } B_m
\end{aligned}
$$

Where:

$A_{i,j}$	are conditions which are to be met
B_i	are actions which need to be taken.
m	is the number of rules used (four will be used)
nf	is the number of controller inputs (two in this case)

The following four rules will be used as a simplified controller for the tank example:

1. If **error** is *positive* and **rate** is *positive* then **output** is *positive*
2. If **error** is *positive* and **rate** is *negative* then **output** is *zero*
3. If **error** is *negative* and **rate** is *positive* then **output** is *zero*
4. If **error** is *negative* and **rate** is *negative* then **output** is *negative*

Thus A_{11} and A_{12} would be the fuzzy sets '*error is positive*' and '*rate is positive*' which appear in the first rule, A_{21} and A_{22} would be the fuzzy sets '*error is positive*' and '*rate is negative*' in the second rule, A_{31} and A_{32} would be the fuzzy sets '*error is negative*' and '*rate*

is positive' in the third rule, and A_{41} and A_{42} would be the fuzzy sets '*error is negative*' and '*rate is negative*' in the fourth rule. The fuzzy consequent B_1 is the fuzzy set '*positive change*', while B_2 and B_3 are '*zero change*' and B_4 is '*negative change*'.

Step 1. Each rule is considered separately. Firstly the grade of membership of the antecedent part of each rule, A_i, is determined. From the above general structure it can be seen that the input requirements for a given rule are linked by the **and** term and therefore the fuzzy intersection, defined in Section 3.3, is used between their grades of memberships:

$$\mu_{A_i}(x_{1_i}, x_{2_i}, \ldots, x_{nf_i})$$

$$= \min[\mu_{A_{i,1}}(x_{1_i}), \mu_{A_{i,2}}(x_{2_i}), \ldots, \mu_{A_{i,nf}}(x_{nf_i})]$$

where: x_{ji} is the numerical value of the j^{th} controller input of rule i.

The first rule checks how *positive* the **error** in the tank's temperature is and how *positive* the **rate** is. This is then transferred into determining how *positive* the **change in the output** will be (**Step 2**). For the **error** of 0.8 and **rate** of -0.4, the **error** is 90% *positive* while the **rate** is 30% *positive* (Rule 1). To determine the input which least fits the conditions of the first rule, the **intersection** (**AND** or **min** operation) is taken between the two. This step is analogous to the theory that a chain is only as strong as its weakest link. Therefore the *positive rate* fuzzy set (30%) is the 'weakest link'. Similarly the 'weakest link' for rules 2, 3 and 4 are 70%, 10% and 10% respectively:

$$\mu_{A_1}(x_{1_1},x_{2_1}) = \min[\mu_{A_{1,1}}(x_{1_1}),\mu_{A_{1,2}}(x_{2_1})] = \min[0.9,0.3] = 0.3$$

$$\mu_{A_2}(x_{1_2},x_{2_2}) = \min[\mu_{A_{2,1}}(x_{1_2}),\mu_{A_{2,2}}(x_{2_2})] = \min[0.9,0.7] = 0.7$$

$$\mu_{A_3}(x_{1_3},x_{2_3}) = \min[\mu_{A_{3,1}}(x_{1_3}),\mu_{A_{3,2}}(x_{2_3})] = \min[0.1,0.3] = 0.1$$

$$\mu_{A_4}(x_{1_4},x_{2_4}) = \min[\mu_{A_{4,1}}(x_{1_4}),\mu_{A_{4,2}}(x_{2_4})] = \min[0.1,0.7] = 0.1$$

$$\textit{where}\quad x_{1_i} = 0.8 \quad \textit{and}\quad x_{2_i} = -0.4$$

Step 2. After combining the antecedent part of each rule, the rule base has the following form:

if A_1 *then* B_1
or
. . .
or
if A_i *then* B_i
or
. . .
or
if A_m *then* B_m

The output range is divided into R (R will be chosen as five for the example) sample values (y_r : r = 1,...,R), each of which are used in the calculations of step 2 and 3. The grade of membership of the i^{th} rule (Z_i = 'if A_i then B_i') is determined as in Section 3.4 by using the intersection between the antecedent, $A_i(x_{ji})$, and consequent, $B_i(y_r)$, parts of rule i:

$$\mu_{Z_i}[x_{1_i}, x_{2_i}, \ldots, x_{nf_i}, y_r] = \min[\mu_{A_i}, \mu_{B_i}(y_r)]$$

where y_r is the r^{th} numerical value used in evaluating the controller output. y_r will be referred to as an output sampling point in subsequent sections.

Figure 3.11 shows the consequent fuzzy sets obtained in each rule over the

output. The effect of matching the antecedent and consequent parts of each rule (Z_i) has been to clip the relevant output fuzzy sets at the value of $\mu_{Ai}(x_{ji})$. Because the 'weakest link' of the first rule is 30% the controller output fuzzy set corresponding to rule 1 (*positive change*) should be clipped at 0.3, the same as performing the **min** operation, as shown in Figure 3.11. Similarly the second rule's conditions only apply by 70% and therefore the *zero change* output fuzzy set should be clipped at 0.7. For the third and fourth rules the output fuzzy sets, *zero change* and *negative change*, are both clipped at 0.1. i.e. if a y_r of 0.6 is taken, then the grade of membership of Z_i is obtained as follows for the tank example:

$$\mu_{Z_1}[x_{1_1}, x_{2_1}, y_r] = \min[\mu_{A_1}, \mu_{B_1}(y_r)] = \min[0.3, 0.6] = 0.3$$

$$\mu_{Z_2}[x_{1_2}, x_{2_2}, y_r] = \min[\mu_{A_2}, \mu_{B_2}(y_r)] = \min[0.7, 0.4] = 0.4$$

$$\mu_{Z_3}[x_{1_3}, x_{2_3}, y_r] = \min[\mu_{A_3}, \mu_{B_3}(y_r)] = \min[0.1, 0.4] = 0.1$$

$$\mu_{Z_4}[x_{1_4}, x_{2_4}, y_r] = \min[\mu_{A_4}, \mu_{B_4}(y_r)] = \min[0.1, 0.0] = 0.0$$

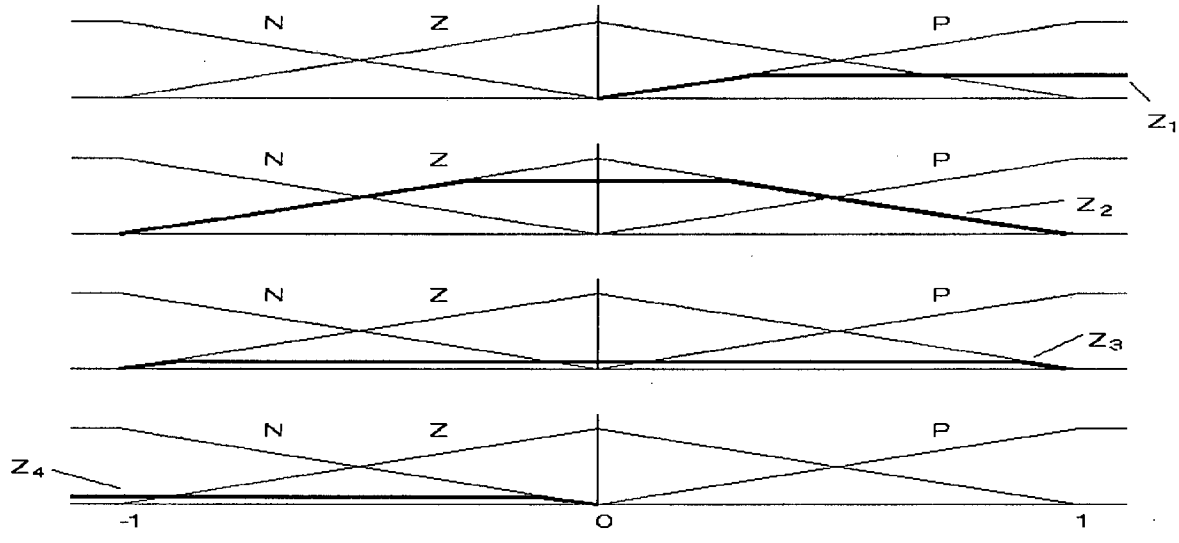

Figure 3.11. Clipping of output fuzzy sets according to degree of applicability of rule.

Step 3. After combining the antecedent and consequent part of each rule, the rule base has the following form:

$$
\begin{array}{c}
Z_1 \\
or \\
\cdots \\
or \\
Z_i \\
or \\
\cdots \\
or \\
Z_m
\end{array}
$$

Finally the fuzzy result is obtained from the rule base by combining the results obtained from each rule, Z_i. From the rule base's general structure it can be seen that each rule is linked by the **or** term and therefore the fuzzy union, defined in Section 3.3, is used between their grades of memberships. Therefore the feasible range of the fuzzy output i.e. the grade of membership of the union, Z, of all the rules, Z_i, can be obtained for each value of y_r investigated:

$$Z = \bigcup_{i=1}^{m} Z_i$$

$$\mu_Z(y_r) = \max_{1 \le i \le m} [\mu_{z_i}]$$

Figure 3.12 shows the feasible area of the controller output which has resulted from the use of the rule base.

The above steps may be combined into a single formula:

$$\mu_Z(y_r) = \max_{1 \le i \le m} \left(\min \left(\min_{1 \le j \le nf} [\mu_{A_{ij}}(x_{j,i_j})] \, , \, \mu_{B_i}(y_r) \right) \right)$$

This formula needs to be used repeatedly so that the range of the dependent variable, y_r, is covered yielding the feasible area in Figure 3.12. The grades of membership for each value of y_r considered are summarised in Table 3.1.

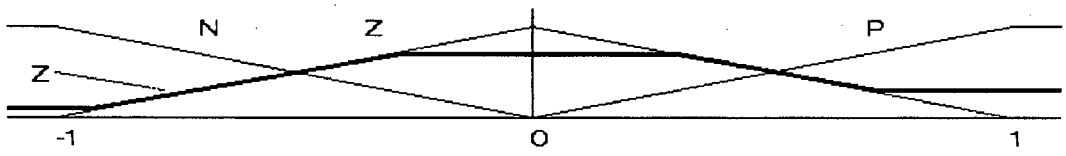

Figure 3.12. Feasible fuzzy output range.

Table 3.1. GMs of the controller output, $\mu_Z(y_r)$.

y_r	$\mu_Z(y_r)$
-1.0	0.1
-0.5	0.5
0.0	0.7
0.5	0.5
1.0	0.3

3.5.3. Defuzzification

During defuzzification the output fuzzy response obtained from the rule base is converted to a crisp numerical value which represents the controller output. The simplest numerical representation (NR1) of the fuzzy set is the numerical value with the highest grade of membership. However, the NR1 representation is rather insensitive as can be seen in Figure 3.12 where the grade of membership (GM) is found to be a maximum of 0.7 over a range of the output [-0.3,0.3]. Therefore a different method is used (NR2) which does not depend only on the highest values of GM. This numerical representation is analogous to the weighted

mean (Dohnal,1985):

$$NR2 = \frac{\sum_{r=1}^{R} y_r \mu_r}{\sum_{r=1}^{R} \mu_r}$$

Therefore the controller output for the tank example can be calculated:

$$NR2 = \frac{(-1.0) \times 0.1 + (-0.5 + 0.5) \times 0.5 + (0.0) \times 0.7 + (1.0) \times 0.3}{(0.1) \times 1 + (0.5) \times 2 + (0.7) \times 1 + (0.3) \times 1}$$

$$= 0.0952$$

This indicates that the required control action is to make a positive change of 0.0952 to the flow rate of the hot stream. It should be noted that the output might not be the same if more output sampling points (y_r) are used.

The output from defuzzification is often scaled to yield the required process input.

3.6. Equivalence with Nonlinear PI Controller

3.6.1. Literature

Qin (1994) presents a fuzzy controller which has two inputs (error and change in the variable's error) and one output (change in the manipulated variable). He utilises only three sampling points on the controller output when calculating the fuzzy implications. The controller presented by Qin (1994) is the same as the one used in the example for Section 3.5 and the same rules are used.

It is stated that this fuzzy controller, with the fuzzy partitions shown in Figure 3.13, is in fact

a nonlinear PI controller of the form:

$$Output = \frac{GU}{3 - \max[|GRr|, |GEe|]} [GRr(t) + GEe(t)]$$

where:
- r — is the change in the error (e(t) - e(t-Δt))
- e — is the error (y_{set} - y)
- GR, GE, GU — are scaling factors for the rate, error and change in output respectively.

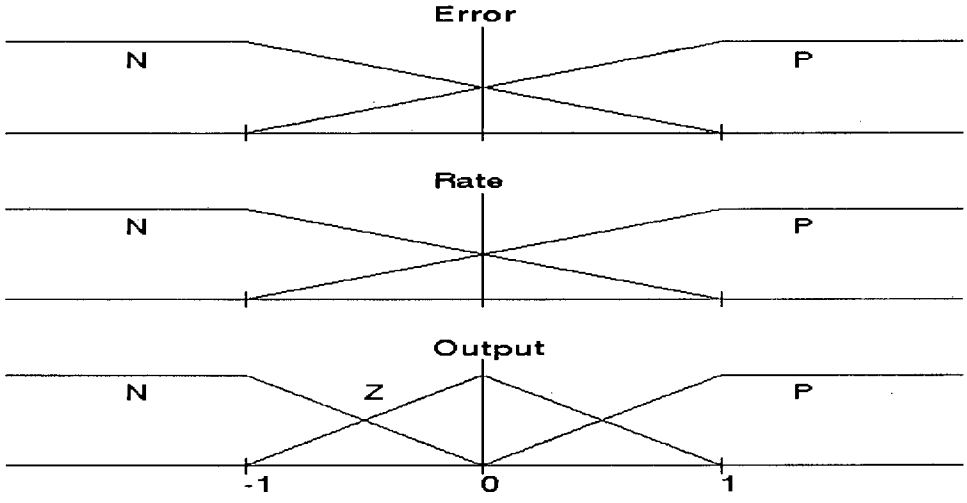

Figure 3.13. Fuzzy partitions based on Qin (1994) approach (*negative* (*N*), *positive* (*P*) and *zero* (*Z*)).

The velocity form of the PI controller's output is given by (Seborg *et al*,1989):

$$\Delta p_n = p_n - p_{n-1} = K_{PI}\left[(e_n - e_{n-1}) + \frac{\Delta t_{sample}}{\tau_{PI}} e_n\right]$$

where:
- Δp_n — is the change in the controller output for the n^{th} sampling interval
- Δt_{sample} — is the sampling time used
- K_{PI} and τ_{PI} — are the gain and integral constant

The nonlinearity is therefore introduced in the gain (analogy to a PI controller):

$$K_{fuzzy} = \frac{GU.GR}{3 - \max[|GRr|, |GEe|]}$$

for values of GRr(t) and GEe(t) lying in the range [-1,1] while the integral constant is defined to be:

$$\tau_{fuzzy} = \frac{\tau_{PI}}{\Delta t_{sample}} = \frac{GR}{GE}$$

Therefore the gain is affected by the dominant term (GEe(t) or GRr(t)). The introduction of the nonlinearity in the gain allows for higher gains to be used when the error and/or the change in error are large. As the error and change in error tend to zero, i.e. as the system stabilises (the max term becomes significantly smaller than 3), the gain becomes a constant and the fuzzy controller is effectively a linear PI controller with:

$$K_{fuzzy} = K_{PI} = \frac{GU\ GR}{3}$$

$$\tau_{fuzzy} = \frac{\tau_{PI}}{\Delta t_{sample}} = \frac{GR}{GE}$$

The tuning factors available for the fuzzy controller are the scaling factors GE, GR and GU. It is hence possible to obtain the scaling factors required for the fuzzy controller from the equivalent PI tuning factors which can be obtained for example by Ziegler-Nichols tuning methods. One more relation is required as there are only two relations and three unknown scaling factors to be determined. Qin (1993) proposes a third relation which is based on the size of the setpoint change:

$$GR = \beta\ \Delta y_{sp}$$

where β is chosen to be 0.4. Therefore, after the PI parameters have been obtained, the

scaling factors for the fuzzy controller can be obtained as follows (Qin,1993):

$$GR = \beta \, \Delta y_{sp}$$

$$GU = \frac{3K_{PI}}{GR} \qquad (1)$$

$$GE = \frac{GR}{\tau_{PI}} \Delta t_{sample}$$

3.6.2. Application to a first order system

Figure 3.14 shows the control response of a first order system with dead time using the Qin type controller. The following transfer function is used for the process:

$$g(s) = \frac{e^{-2s}}{10s + 1}$$

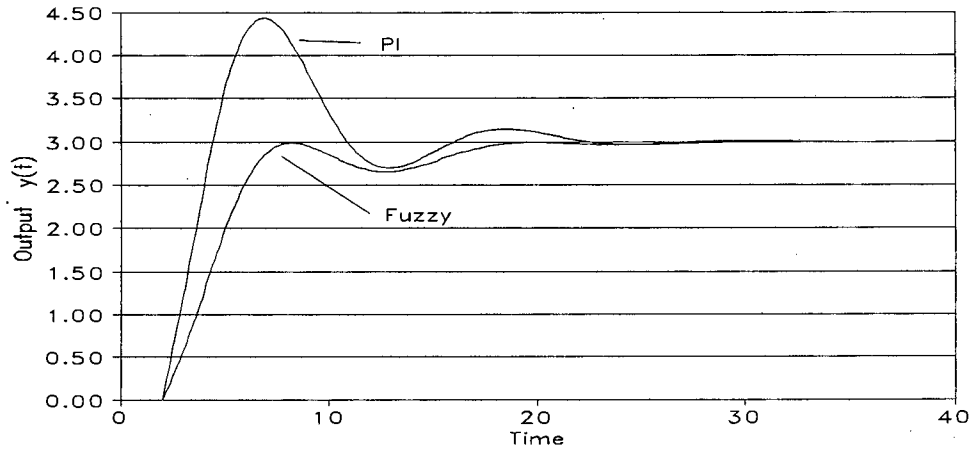

Figure 3.14. Comparison of Qin fuzzy controller with linear PI controller for first order system with dead time.

The Ziegler-Nichols tuning parameters were obtained and hence the fuzzy scaling factors could be calculated as summarized below.

$K_{PI} = 3.8647$

$\tau_{PI} = 6.2013$

giving

GR = 1.2 (for a set point change of 3.)

GU = 9.662

GE = 0.01935 (for a sampling time of 0.1)

The fuzzy controller's gain is allowed to assume higher values when the system is further away from the desired state (see Figure 3.19 in Section 3.8.1). This ensures, for the first order system, that the overshoots are reduced compared to the linear PI controller as shown in Figure 3.14. The two responses were found to settle at around the same time. The input response was found to be less severe for the fuzzy controller as seen in Figure 3.15. Appendices G and H contain the Fortran programs used for the fuzzy controller and PI controller.

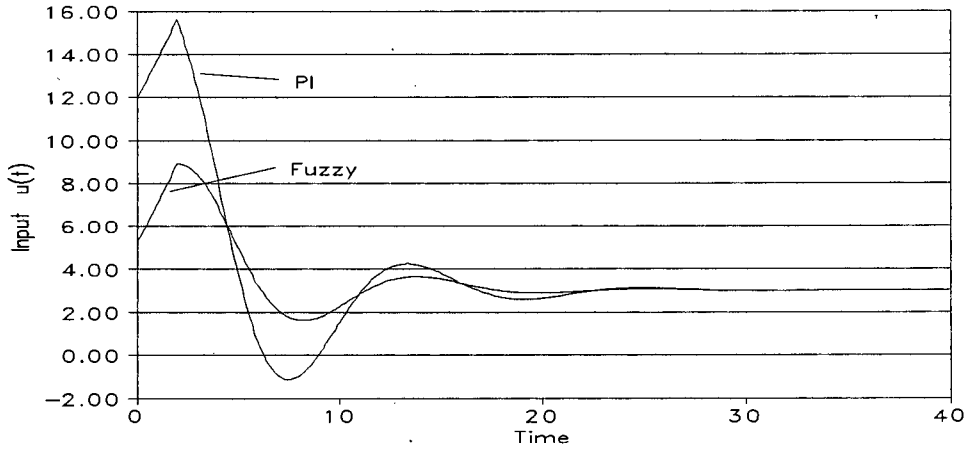

Figure 3.15. Input response of first order system with dead time.

3.7. Effect of Fuzzy Set's Width

Ying *et al* (1990) were the first to present the findings that this form of fuzzy controller yields a nonlinear PI controller. The main difference between Ying *et al* (1990) and Qin's

(1993) approaches is that Ying uses the **Łukasiewicz OR** between rules 2 and 3 when determining the union of the rules (Section 3.5.2 **Step 3**) whereas Qin uses the **Zadeh OR**. For two fuzzy sets, A and B, Zadeh logic yields

$$OR(\mu_A, \mu_B) = \max(\mu_A, \mu_B)$$

and Łukasiewicz logic yields

$$OR(\mu_A, \mu_B) = \min(1, \mu_A + \mu_B)$$

where μ_A is the grade of membership of an element in the fuzzy set A.

Ying *et al* (1990) obtain the following equation for the controller's output:

$$Output = \frac{0.5L.GU}{2L - \max\left[|GRr|, |GEe|\right]} \left[GRr(t) + GEe(t)\right]$$

where L is the width of all fuzzy sets as shown in Figure 3.16.

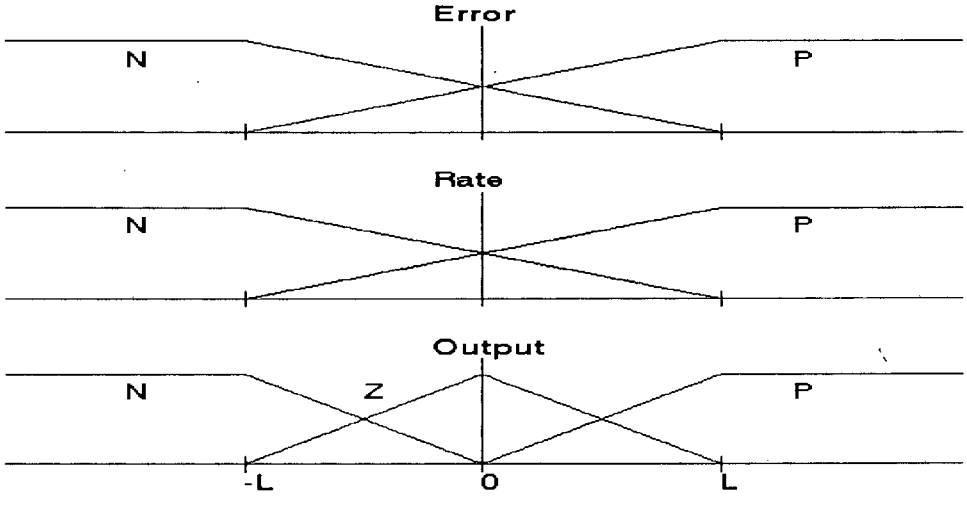

Figure 3.16. Input and output fuzzy sets used by Ying *et al* (1990).

A noticeable difference between the two equations is that Qin (1994) did not build the length, L, of the fuzzy sets into his equations. Ying *et al* (1990) does not mention the effect of a variation in this parameter. It was therefore decided to re-derive the Qin (1994) equations

yielding the following more generalized equation (see derivation in Appendix A):

$$Output = \frac{L.GU}{3L - \max\left[|GRr|, |GEe|\right]}\left[GRr(t) + GEe(t)\right] \qquad (2)$$

It is therefore evident that Qin (1994) has chosen L = 1. However, as long as the error and rate are within the [-L,L] range, L is readily absorbed into the fuzzy tuning parameters by redefining

$$GU' = \frac{GU}{L} \quad , \quad GR' = L.GR \quad and \quad GE' = L.GE$$

and hence is not an independent tuning parameter.

3.8. Effect of Number of Output Sampling Points

Up until this stage only 3 points (-L, 0 and L) have been sampled on the output fuzzy sets when calculating the fuzzy response of the controller. Figures 3.17 and 3.18 show an improved output and input response over the Ziegler-Nichols tuned PI controller as well as over a PI controller whose gain and time constant have been optimised to minimise the integral square setpoint tracking error. It can also be seen that a variation in the number of output sampling points has an effect on the output as well as the input response for the first order system.

The input response is found to be less severe for an increased number of sampling points. It can also be seen that both the output and input responses converge as the number of output sampling points is increased. Therefore it was decided to derive and investigate the output response equations for a higher number of output sampling points.

3.8.1. Three sampling points

Appendix A contains the derivation of the output response equations for three sampling points. Figure 3.19 shows how the gain is allowed to vary over the range of the rate and the

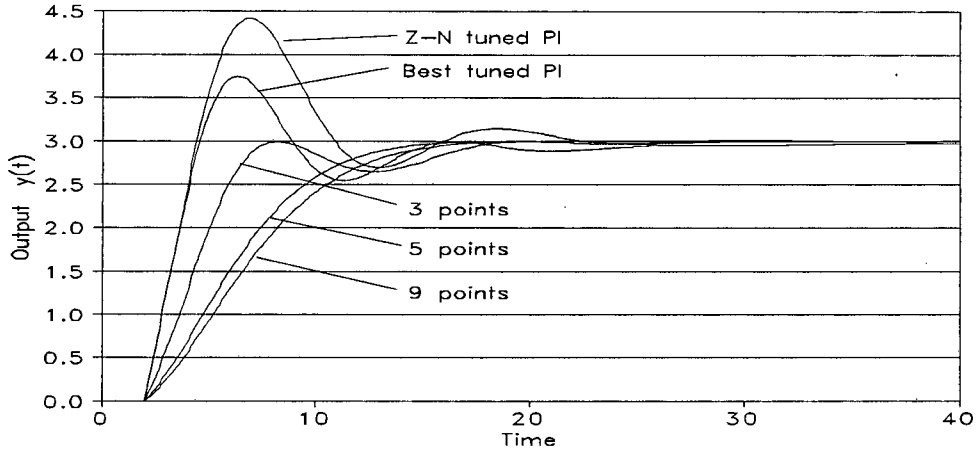

Figure 3.17. Output response for a first order system with dead time (L = 1)

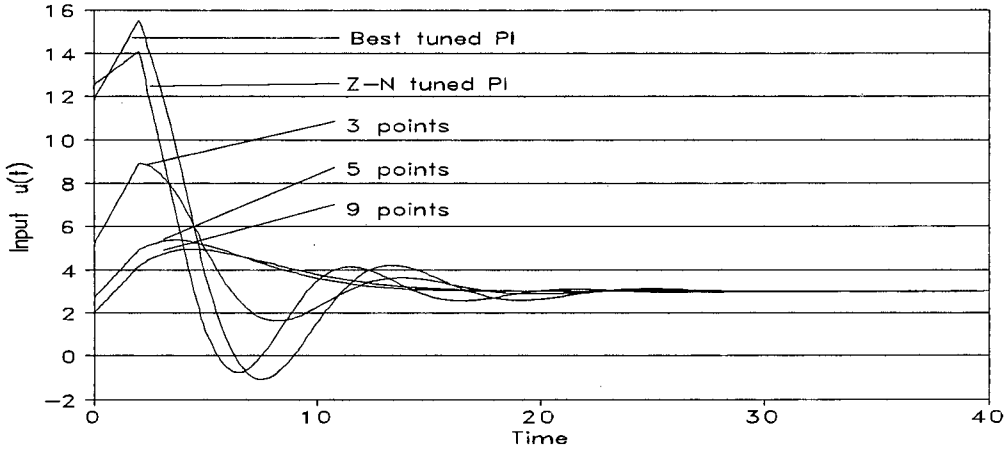

Figure 3.18. Input response for a first order system with dead time (L = 1)

error, [-L,L], with L, GE, GR and GU chosen as unity. The output response equations are also summarised in Appendix A when the rate and/or error do not lie within the [-L,L] range. As the rate and error are driven to zero by the controller, so the system is forced into the [-L,L] range and therefore only the output response equations applicable within this range will be addressed.

From Figure 3.19 it can be seen that the gain is able to vary from 0.333 (same gain used for the standard linear PI controller) to a value of 0.5 (1.5 times the linear PI gain). The gain

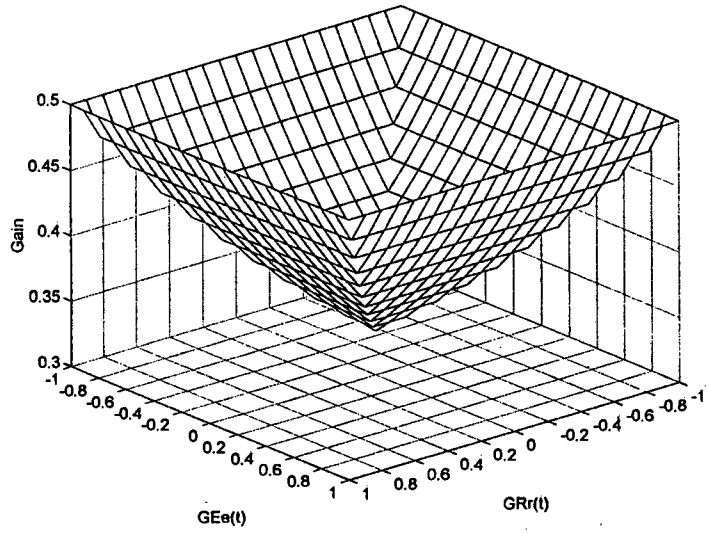

Figure 3.19. Variation of gain over input region for three sampling points.

attains the value of 0.333 only when the system is at its desired state (when both the rate and error are zero) and is allowed to assume higher values as the system is further away.

3.8.2. Five sampling points

The equations for the output response of the fuzzy controller were derived using the following sampling points: (-L, -½L, 0, ½L, L). It was decided to maintain the three original sampling points and utilise another two sampling points so as to keep the sampling symmetrical. The resultant equations once again yielded a nonlinear PI controller and are shown in Figure 3.20.

Table 3.2 summarises the controller gains and integral constants obtained. The output response equations are derived in Appendix B.

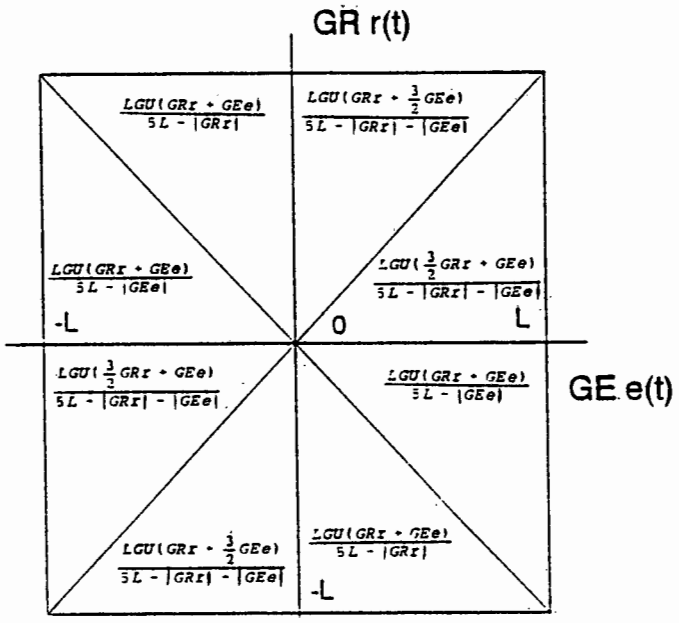

Figure 3.20. Output response equations in different regions for five output sampling points.

Table 3.2. Nonlinear gain and integral constant for five sampling points.

Parameter	Error and rate of same sign		Opposite sign
	$\lvert GRr\rvert > \lvert GEe\rvert$	$\lvert GRr\rvert < \lvert GEe\rvert$	
K_{fuzzy}	$\frac{LGUGR}{5L - \lvert GRr\rvert - \lvert GEe\rvert}$	$\frac{\frac{3}{2}LGUGR}{5L - \lvert GRr\rvert - \lvert GEe\rvert}$	$\frac{LGUGR}{5L - \max[\lvert GRr\rvert, \lvert GEe\rvert]}$
τ_{fuzzy}	$\frac{2}{3}\frac{GR}{GE}$	$\frac{3}{2}\frac{GR}{GE}$	$\frac{GR}{GE}$

From Table 3.2 it can be seen that not only the controller gain, but also the controller integral constant are now functions of the relative position of the system to its desired state. When the rate and error are of an opposite sign the gain and integral constant equations resemble the ones obtained for three sample points in Section 3.7. Here the gain is once again allowed to assume higher values determined by the dominant term (either GRr(t) or GEe(t)).

It can be seen that higher gains are used when the rate and error are of the same sign. It can also be seen that the gain attains a value 1.5 times greater when the absolute error is greater than the gain when the absolute rate is greater. Therefore the gain is discontinuous when GRr = GEe; however, the output response is continuous because the controller utilises an appropriately scaled integral constant when the gain is higher ($|GRr| < |GEe|$) and vice versa. Therefore the gain and integral constant adjust to give the same output response.

Figure 3.21 shows the values that the gain is allowed to assume with L, GE, GR and GU chosen as unity. It can be seen that the maximum gain attained corresponds to the maximum gain when three sampling points are used (0.5 when L, GE, GR and GU are unity). However, in the case of five sampling points, the gain is allowed to vary over a greater range [0.2,0.5] (cf. [0.3333,0.5] for three sampling points). It can also be seen that low gains are usually utilised and that the high gains are only used when both the rate and error are of the same sign. Therefore, whereas the controller with three sampling points utilises a high gain irrespective of the sign of the error and the rate, the controller with five sampling points does not penalise the system by using high gains when the rate and error are of an opposite sign i.e. when the system is showing signs of rectifying itself. It should also be noted that the gain is able to assume a low value of 0.2 when both the rate and error are zero.

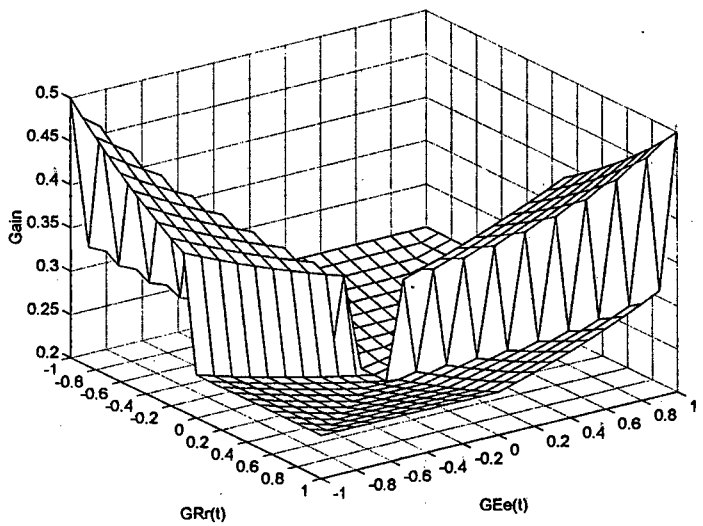

Figure 3.21. Gain values attained for five sampling points.

3.8.3. Location of Sampling Points for n ≥ 3

Figure 3.22 shows the symmetrical location of the three output sampling points.

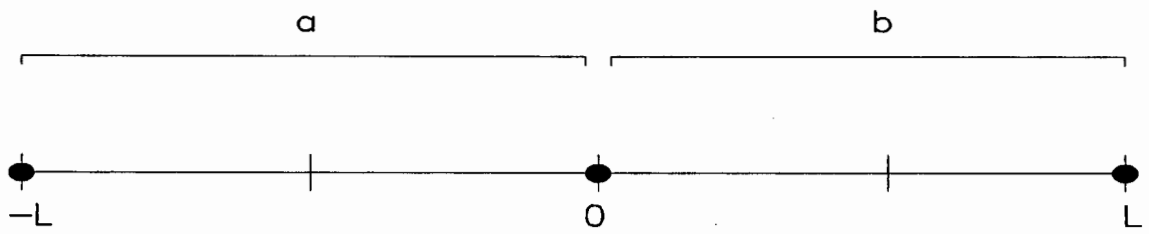

Figure 3.22. Location of output sampling points for n = 3.

Only n (for n≥3) given by the following equation will be investigated:

$$n = 5 + 2 \times q \tag{3}$$

where: q is a non-negative integer

For five sampling points one extra sampling point is placed at the centre of each region (a and b) shown in Figure 3.22. For nine sampling points three equally spaced sampling points are placed in each region and so forth. Therefore all the sampling points are always equidistant and the sampling is kept symmetrical.

3.8.4. Nine sampling points

Appendix C contains an explanation for not investigating seven output sampling points. The derivation of the output response equations for nine sampling points can be found in Appendix D. The fuzzy controller once again yielded a nonlinear PI controller. However, the equations now have an added **L** term which acts as a bias:

$$Output = K_{fuzzy}\left[r(t) + \frac{1}{\tau_{fuzzy}} e(t) + CL\right]$$

The coefficient of this **L** term is a function of the system's relative position to the desired

state.

Figure 3.23 shows the values that the gain is able to assume. The gain is now able to vary over a greater range [0.125,0.5]. Once again the highest gains are used only when both the rate and error are of the same sign. Much lower gains are utilised when they are of opposite signs, because the system is then showing signs of rectifying itself. A low gain of 0.19444 is used when the rate and error are both zero.

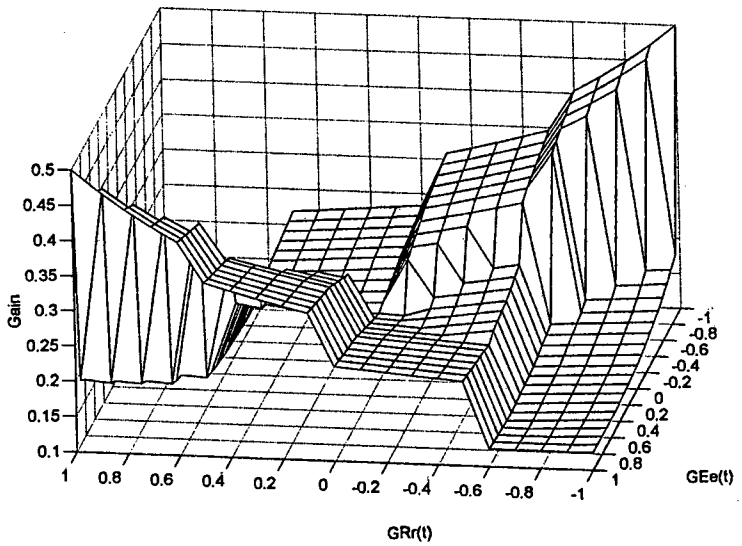

Figure 3.23. Gains utilised for nine output sampling points.

It should be stressed that the equations derived in Appendix D are only applicable when the scaled error and scaled rate are in the range [-L , L]. The equations applicable when the scaled error and rate are out of this range can be obtained in a similar fashion.

3.8.5. Generalisation and extension to more output sampling points

The fuzzy controller results shown in Figure 3.17 and Figure 3.18 indicate that the output and input responses converge to a limit as the number of output sampling points is increased. The input action is also found to be less severe for a higher value of n for the first order system with dead time. Therefore it was decided to investigate the equations describing the fuzzy output response for a higher number of sampling points.

It was observed that the output response equations can easily be determined as a function of

n for the **a** region shown in Figure 3.24. This is possible because, for each of the eight triangles making up region **a**, the same fuzzy resultant, Z, is obtained irrespective of the value of n ($n \geq 5$) satisfying Equation (3). However, it should be noted that the **a** region becomes smaller and tends towards the origin as the value of n is increased. Therefore the equations are valid for a smaller region with an increased value of n. It can easily be seen that the equations for the **a** region, summarised in Table 3.3, are applicable for both five and nine sampling points. These generalised equations for region **a** once again have the following PI form:

$$Output = K_{fuzzy}\left[r(t) + \frac{1}{\tau_{fuzzy}} e(t)\right]$$

It can be seen that there is no bias term present irrespective of n. The generalised equations are derived in Appendix E.

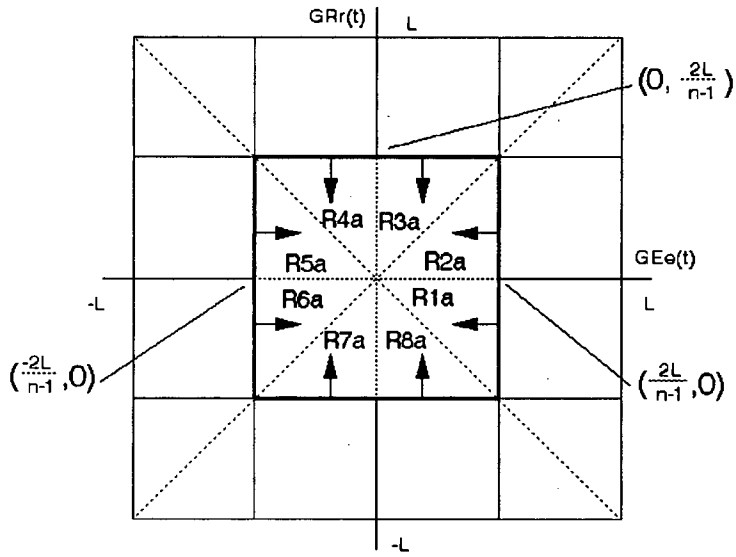

Figure 3.24. Region in which the generalised equations are applicable for higher values of n.

Table 3.3. Generalised equations for region **a**.

Region	
R1a	$\frac{L.GU\left[\frac{3n+1}{16}GRr + \frac{3n+1}{16}GEe\right]}{nL - \frac{n-5}{4}GRr - \frac{n-1}{4}\lvert GEe\rvert}$
R2a	$\frac{L.GU\left[\frac{3(n+3)}{16}GRr + \frac{3n+1}{16}GEe\right]}{nL - \frac{n-1}{4}GRr - \frac{n-1}{4}\lvert GEe\rvert}$
R3a	$\frac{L.GU\left[\frac{3n+1}{16}GRr + \frac{3(n+3)}{16}GEe\right]}{nL - \frac{n-1}{4}\lvert GRr\rvert - \frac{n-1}{4}GEe}$
R4a	$\frac{L.GU\left[\frac{3n+1}{16}GRr + \frac{3n+1}{16}GEe\right]}{nL - \frac{n-1}{4}\lvert GRr\rvert - \frac{n-5}{4}GEe}$
R5a	$\frac{L.GU\left[\frac{3n+1}{16}GRr + \frac{3n+1}{16}GEe\right]}{nL + \frac{n-5}{4}GRr - \frac{n-1}{4}\lvert GEe\rvert}$
R6a	$\frac{L.GU\left[\frac{3(n+3)}{16}GRr + \frac{3n+1}{16}GEe\right]}{nL + \frac{n-1}{4}GRr - \frac{n-1}{4}\lvert GEe\rvert}$
R7a	$\frac{L.GU\left[\frac{3n+1}{16}GRr + \frac{3(n+3)}{16}GEe\right]}{nL - \frac{n-1}{4}\lvert GRr\rvert + \frac{n-1}{4}GEe}$
R8a	$\frac{L.GU\left[\frac{3n+1}{16}GRr + \frac{3n+1}{16}GEe\right]}{nL - \frac{n-1}{4}\lvert GRr\rvert + \frac{n-5}{4}GEe}$

Figure 3.24 shows that the **a** region in which the generalised equations are applicable becomes smaller and eventually tends towards the origin as n is increased. Because the generalised **a** equations are applicable, despite the number of sampling points, in the region around the origin, it is possible to evaluate the output equations describing the response of the fuzzy controller, for GEe(t) and GRr(t) sufficiently close to the origin, as the number of sampling points becomes large.

As the number of output sampling points, n, is increased, the equations shown in Table 3.3 are found to simplify to

$$\lim_{n \to \infty} output = \frac{\frac{3}{16} L\,GU(GRr + GEe)}{L - \frac{1}{4}|\max| - \gamma\frac{1}{4}\min}$$

if both GRr(t) and GEe(t) are of the same sign and

$$\lim_{n \to \infty} output = \frac{\frac{3}{16} L\,GU(GRr + GEe)}{L - \frac{1}{4}|\max| - \delta\gamma\frac{1}{4}\min}$$

if GRr(t) and GEe(t) are of opposite signs, where

γ	is -1 if rate is negative and +1 if rate is positive
δ	is -1 if $\lvert GRr\rvert < \lvert GEe\rvert$ and +1 if $\lvert GRr\rvert > \lvert GEe\rvert$
max	is given by

$$GRr(t) \quad if \ |GRr(t)| \geq |GEe(t)|$$
$$GEe(t) \quad if \ |GRr(t)| \leq |GEe(t)|$$

min is given by

$$GRr(t) \quad if \ |GRr(t)| \leq |GEe(t)|$$
$$GEe(t) \quad if \ |GRr(t)| \geq |GEe(t)|$$

But as the number of output sampling points is increased, the region of applicability is shrunk more and more such that the value of GRr(t) and GEe(t) are close to zero. Therefore the

equations reduce to:

$$\frac{3}{16}GU(GRr + GEe)$$

which now has both a linear gain and a linear integral constant.

This shows that, for a high number of input sampling points, that the gain is allowed to assume a low value of 0.1875 when the system is close to or at its desired state. This means that small perturbations from this desired state will not yield as drastic input actuation than it would if n were smaller or if an equivalently tuned linear PI controller was used. It should also be noted that at/near a zero error and rate a gain of 0.333 was used for three sampling points, 0.2 for five sampling points and 0.19444 for nine sampling points. Therefore it can be seen that the gain, used at/near a zero rate and error, assumes a value between 0.333 and 0.1875 depending on the number of output sampling points utilised. The higher the number of sampling points, the lower the gain becomes and eventually approaches 0.1875.

During defuzzification the output fuzzy response obtained from the rule base is converted to a crisp numerical value, by utilising the weighted mean approach, which represents the controller output response as stated in Section 3.5.3:

$$NR2 = \frac{\sum_{r=1}^{R} y_r \mu_Z(y_r)}{\sum_{r=1}^{R} \mu_Z(y_r)}$$

As long as the error and rate lie within the [-L,L] range μ_Z will always be a linear function of L/L, GRr(t)/L or GEe(t)/L while y_r will always be a scaled value of L. Therefore the numerator will always be a linear combination of L, the scaled error and the scaled rate as found in Appendices A, B, D and E. Similarly the denominator will always be a linear combination of L/L, GRr(t)/L and/or GEe(t)/L. Therefore the output response equations will always have the form

$$\frac{L.GU[A.L + B.GRr(t) + C.GEe(t)]}{D.L + E.GRr(t) + F.GEe(t)}$$

irrespective of the number of output sampling points used or the position of the rate and

error in the GRr - GEe space (with GRr and GEe within [-L,L]) where A, B, C, D, E and F are real valued coefficients. In Appendix F it is shown that the B and C coefficients will always be positive ensuring a positive integral constant. Therefore it can be seen that the fuzzy controller will always yield a nonlinear PI controller with a possible bias term irrespective of the number of output sampling points utilised. This controller will always have a zero offset because of the integral term included and because the bias term is not present in the **a** region's output response equations.

3.8.6. Application to higher order systems

The Qin type controller with various numbers of output sampling points was applied earlier to a first order system with dead time. In this section the controller will be applied to both a second order system as well as to a third order system with dead time. In Section 5.1 the controller will be applied to two nonlinear systems - the fed-batch fermentation of penicillin and lysine.

3.8.6.1. Second order system

The following second order system was investigated:

$$g(s) = \frac{1}{s(s+1)}$$

The PI controller gain and integral constant were obtained according to the Ziegler-Nichols tuning technique as

$K_{PI} = 0.64011$

$\tau_{PI} = 5.21573$

From these PI tuning parameters the fuzzy controller's tuning parameters could be calculated (as summarised in Section 3.6.1):

GR = 1.2 (for a setpoint change of 3)

GU = 1.6003

GE = 0.023 (for a sampling time of 0.1)

Figure 3.25 shows the improvement in the output response showing a lower overshoot when the fuzzy controller is implemented compared to the linear PI controller. The system's rise time becomes slightly longer as the number of output sampling points is increased. The response of the controller is also found to converge to a limit with a low overshoot as the number of output sampling points is increased. These results parallel the findings for the first order system with dead time.

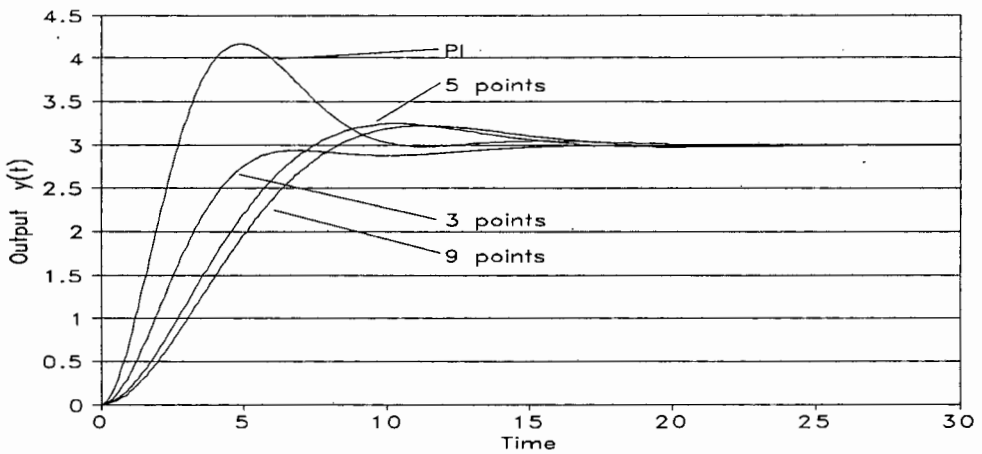

Figure 3.25. Output response of the second order system (L = 1).

Figure 3.26 shows the marked improvement of the input response when the fuzzy controller is implemented over the case when the linear PI controller is used with higher values of n yielding much less violent input action. The Fortran program listings used for the fuzzy controller and the PI controller can be found in Appendices I and J respectively.

3.8.6.2. Third order system with dead time

The following third order system was investigated:

$$g(s) = \frac{(s + 1)e^{-0.2s}}{s(s + 2)(s + 3)}$$

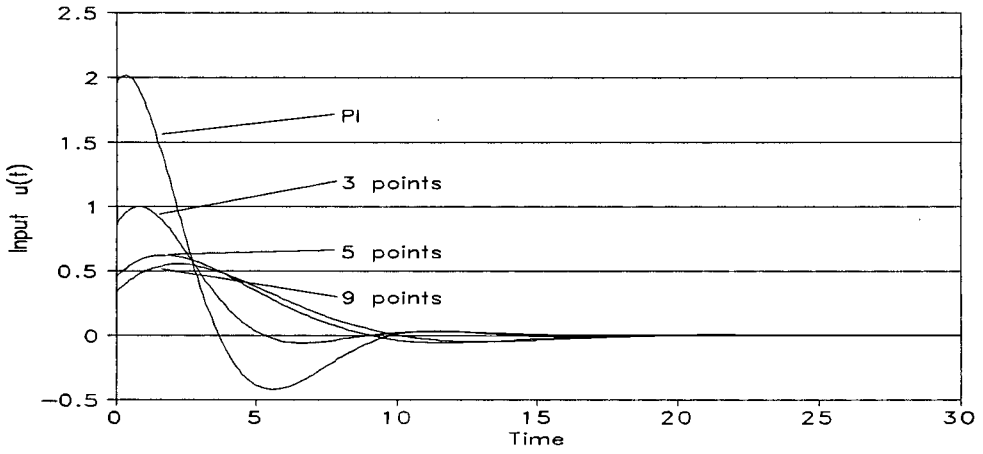

Figure 3.26. Input response of the second order system (L = 1).

The PI controller gain and integral constant were again obtained according to the Ziegler-Nichols tuning technique:

$$K_{PI} = 10.41229$$
$$\tau_{PI} = 1.254361$$

From these PI tuning parameters the fuzzy controller's tuning parameters could be calculated (as summarised in Section 3.6.1):

GR = 1.2 (for a setpoint change of 3)
GU = 26.0307
GE = 0.09567 (for a sampling time of 0.1)

Figure 3.27 shows that the fuzzy controller, with three sampling points, results in a much reduced overshoot for this third order system. However there is still a great deal of oscillation present and the system takes a long time before it settles down. As the number of output sampling points is increased the system becomes more damped and the settling time is decreased markedly.

Figure 3.28 shows the marked improvement of the input response when the fuzzy controller

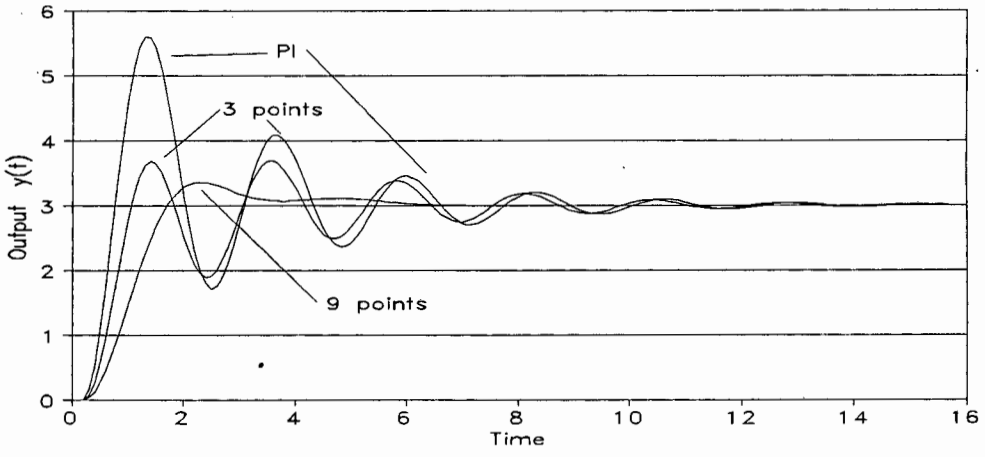

Figure 3.27. Output response of the third order system with dead time (L = 1).

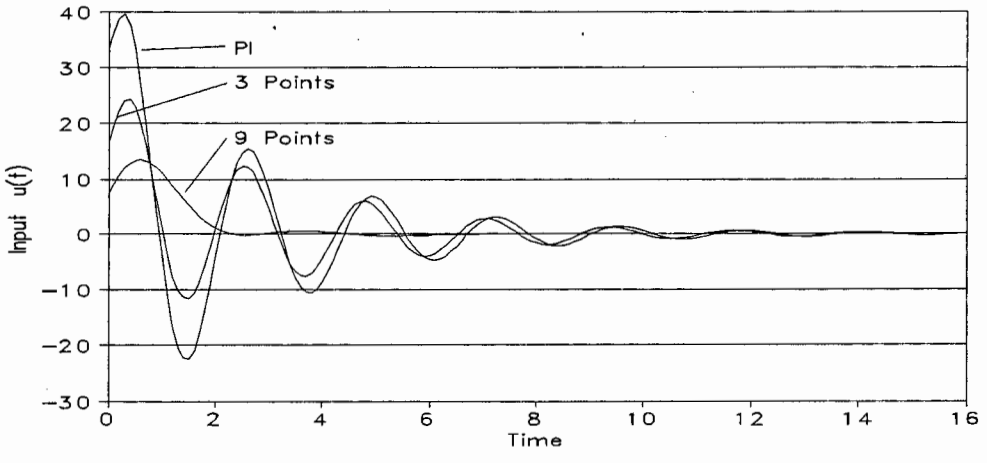

Figure 3.28. Input response of the third order system with dead time (L = 1).

is implemented, especially for a high number of output sampling points where the input response is no longer as harsh or oscillatory. The Fortran program listings used for the fuzzy controller and the PI controller can be found in Appendices K and L respectively. The results from Figures 3.27 and 3.28 show that the fuzzy controller is able to yield better control for this third order system (with dead time) especially for a high number of input sampling points.

3.8.7. Choice of n for setpoint tracking

In Section 3.8.5 it has been shown that the Qin type fuzzy controller yields a nonlinear PI controller irrespective of the number of output sampling points utilised. It has also been shown that the controller's gain and integral constant have a higher resolution as a function of the error and rate for higher values of n which should improve the performance of the controller. It is therefore hypothesised that tuning the fuzzy controller via β (used in the derivation of GR in equation 1, Section 3.6.1) with a high number of output sampling points, n, will result in better setpoint tracking than tuning the fuzzy controller with n=3. i.e. it is hypothesised that the best tuned fuzzy controller with a high value of n will yield better setpoint tracking than the best tuned fuzzy controller with n=3. This hypothesis was only applied to a first order system with dead time as many chemical engineering systems can be approximated by this type of model.

The measure of performance was chosen to be the sum of the squared errors (SSE) of the output from the setpoint

$$SSE = \sum_{i=0}^{m} (y_{set} - y_i)^2$$

where m is the number of time steps investigated

y_{set} is the setpoint which is to be attained by the output

y_i is the value of the output response during the i^{th} time step.

Table 3.4 shows the values of the sum of squared errors as well as the sum of squared change in the controller output response (SSΔU) for the best tuned fuzzy controllers for the first order system presented in Section 3.8.6. The best tuned linear PI controller's gain and integral constant were utilised to determine the remaining fuzzy tuning parameters i.e. determine GU and GE as summarised in equation 1, Section 3.6.1. Figure 3.29 shows the output response of the fuzzy controller while Figure 3.30 shows the input response.

Table 3.4. SSE and SSΔU for best tuned fuzzy controllers.

n	β	SSE	Percentage increase in SSE over PI	SSΔU	Percentage decrease in SSΔU over PI
PI	N/A	277.3392		163.1415	
3	0.005	277.2716	-0.06043	168.166	-3.07984
5	0.113	291.2811	5.027021	87.1229	46.59673
9	0.055	301.9189	8.862685	68.4161	58.06334
45	0.010	327.4603	18.07213	53.1860	67.39885

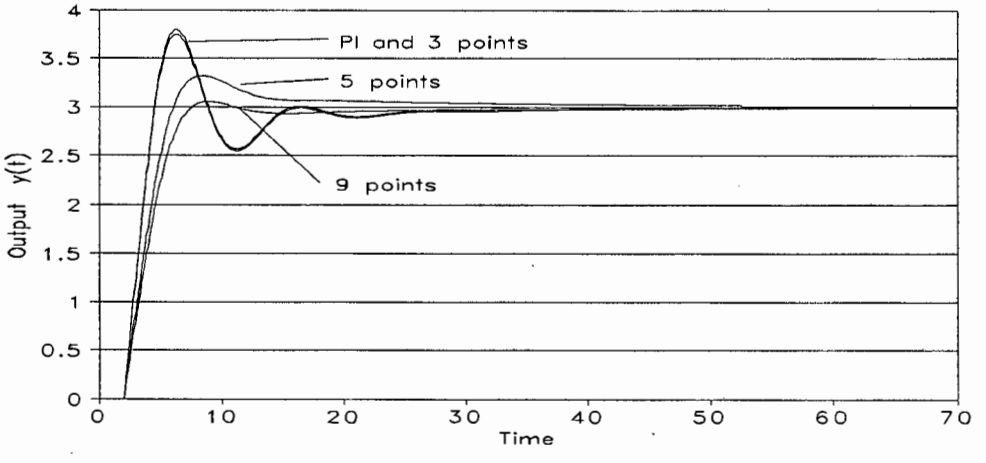

Figure 3.29. Output response of best tuned fuzzy controllers.

Table 3.4 shows that the SSE actually increased with an increase in the number of output sampling points. The fuzzy controller was able to yield a slightly lower SSE than the best tuned linear PI controller however, it also yielded a higher SSΔU. It can be seen that although the SSE increases with an increase in n, this increase is relatively small compared with the decrease in the value of SSΔU. This means that the input action becomes less violent for relatively the same value of SSE as n is increased as seen in Figures 3.29 and 3.30. The increase in SSE is due to the increase in the rise time i.e. slightly more sluggish response for

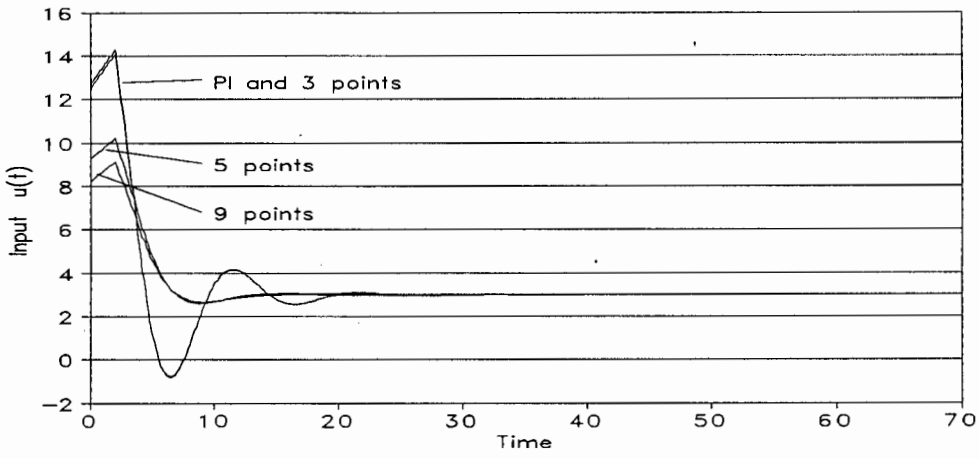

Figure 3.30. Input response of best tuned fuzzy controllers.

a higher n as shown in Figure 3.29.

4. BIOPROCESS SYSTEMS INVESTIGATED

4.1. Lysine Fermentation

4.1.1. Uses of lysine

Lysine is widely used in the animal feed industries to improve the nutritional value of the feeds. This is done because feedstocks do not always meet the essential lysine requirements for the economical growth of the animals (Kirk-Othmer,1992). Lysine is also used to enrich wheat flour and rice grain to improve human nutrition (Kirk-Othmer,1978).

Lysine exports from the United States of America totalled 4 million kilograms at a value of US$ 7.4 million for 1993 (Cintrón,1993).

4.1.2. Lysine production by *Corynebacterium glutamicum*

Lysine is one of 20 amino acids which are the monomers (basic building blocks) of proteins. The bacteria utilise the amino acids which they have formed in the production of enzymes and proteins which are to be utilised by the cell itself. The cell is able to regulate the production of the amino acids, and hence regulate its activities, by inbuilt feedback inhibition and repression and by its innate network rigidity.

Figure 4.1 shows part of the metabolic pathway of the bacteria. In the bacteria, the metabolic flux is normally channelled more favourably to produce threonine and methionine than to the production of lysine. There is also a feedback inhibition which both threonine and lysine exert on the production of the aspartate kinase enzyme which catalyses the conversion of aspartate. Aspartate kinase is only inhibited by high levels of both lysine and threonine and not inhibited to a great degree if only one of the concentrations is high.

In order for the bacteria to be able to overproduce lysine, the bacteria has been mutated to produce a strain with a metabolic blockage prior to homoserine in the metabolic pathway.

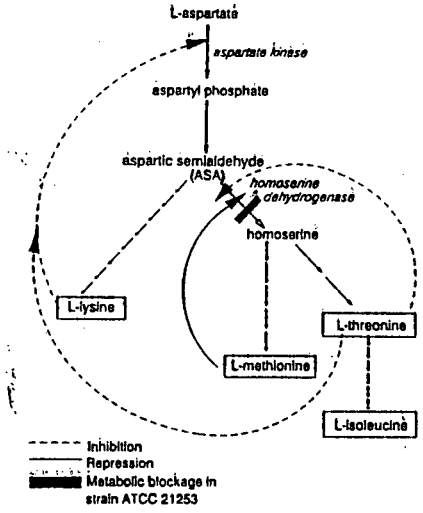

Figure 4.1. Metabolic pathways and regulation of the aspartate family of amino acids in *C. glutamicum* (Kiss and Stephanopoulos, 1992).

Therefore the cell is unable to produce methionine, threonine and isoleucine and therefore the fermenter feed has to be supplemented with these amino acids. The cells will therefore be able to overproduce lysine when low levels of threonine are maintained in the bioreactor due to the bypass of aspartate kinase inhibition by threonine and lysine (Kiss and Stephanopoulos, 1991).

4.1.3. Lysine fermentation

Lysine is produced via aerobic fed-batch bacterial fermentation. Fed-batch fermentation is used because a higher specific growth rate can be achieved than when operated in the batch mode as it is possible to control the glucose feed rate to the fermenter in order to maximise the cellular growth rate. If too much glucose is fed it will have an inhibitory effect on the cell growth. Controlling the inlet flow of substrate will lead to a higher lysine production in the fed-batch case for the same fermentation time. The glucose substrate is fed at a rate that is just high enough to ensure good cell growth. It is also operated as fed-batch instead of continuously, to lower the likelihood of contamination.

The high yield of lysine is obtained by first maximising the growth of the bacteria and then maximising the production of lysine from the bacteria. Lysine is a non-growth associated product and therefore a high yield is only obtained while the bacteria are grown with a low

specific growth rate (Kiss and Stephanopoulos,1991).

4.1.3.1. Cell production

A high concentration of cells is required in order to ensure a good yield of lysine. During the biomass production stage, a high threonine concentration is required in the bioreactor to ensure that the bacteria concentrate on the production of biomass and not on lysine production. A large enough predetermined initial charge of threonine will ensure that the concentration of threonine is always high enough in the bioreactor and hence the lysine overproduction will be suppressed during this stage. The initial charge of threonine and methionine can be predetermined so that the virtual depletion of threonine in the bioreactor coincides with the intended time for changing to lysine overproduction.

A high bacterial specific growth rate is required and therefore the rate at which the glucose is fed to the bioreactor is crucial. The initial charge of glucose does not contain all the glucose which is required throughout the fermentation due to the toxic or inhibitory effect this will have on the bacteria. Instead, the glucose is fed to the bioreactor as it is required. The glucose is fed in such a way as to control the cell growth within the available OUR (oxygen utilization rate). If the glucose feed rate is too high, the cells will become oxygen limited, while a feed rate which is too low will lead to low specific growth rates.

4.1.3.2. Lysine production

The threonine concentration has to be maintained at a low level and this is done by regulation of this amino acid feed during the lysine production stage. Similarly, the glucose concentration is to be maintained at a constant value by matching the feed to the lysine production. A low glucose level is required to maintain a low specific growth rate which will ensure a high yield of lysine.

4.1.4. Model of fermentation

Ohno *et al* (1976) modelled fed-batch lysine fermentation using the following four differential equations derived from mass balance considerations:

$$\begin{aligned} \frac{dxV}{dt} &= \mu x V \\ \frac{dsV}{dt} &= -\frac{\mu}{Y} x V + s_i F \\ \frac{dpV}{dt} &= Q_p x V \\ \frac{dV}{dt} &= F \end{aligned} \qquad (4)$$

where:

	x	is the biomass concentration in the bioreactor (g/L)
	s	is the substrate concentration (glucose) in the bioreactor (g/L)
	p	is the product concentration (lysine) in the bioreactor (g/L)
	V	is the volume of liquid in the bioreactor (L)
	F	is the flow rate of the inlet stream (L/h)
	s_i	is the concentration of glucose in the feed stream (g/L)
	μ	is the specific growth rate of the bacteria given in equation 5 (h^{-1})
	Y	is the yield factor and taken to be 0.135 $g_{biomass}/g_{subs}$
	Q_p	is the specific product formation rate given in equation 6 ($g_{prod}/g_{biomass}.h^{-1}$)

$$\mu = 0.125s \qquad (5)$$

$$Q_p = -384\mu^2 + 134\mu \qquad (6)$$

The mass balance equations (equation 4) can be written as follows:

$$\begin{aligned} \frac{dx}{dt} &= \mu x - \frac{F}{V}x \\ \frac{ds}{dt} &= \frac{F}{V}(s_i - s) - \frac{\mu}{Y}x \\ \frac{dp}{dt} &= Q_p x - \frac{F}{V}p \\ \frac{dV}{dt} &= F \end{aligned} \qquad (7)$$

The first term in the biomass differential equation represents an increase in the mass of cells due to growth. In the second differential equation the first term represents an increase in fermenter substrate due to the feeding of the substrate. The second term represents a depletion of substrate due to cell growth. The first term in the third differential equation represents an increase in the amount of product due to cellular activity.

The above differential equations are integrated over the required time period utilising a Fortran Nag Library routine, with a time step of 0.2 hours (i.e. 12 minutes) to yield the trajectories of the fermenter biomass, substrate and product concentrations and also the broth volume.

4.1.5. Open loop response

The open loop response of the model was investigated at different substrate feed concentrations as shown in Figures 4.2 and 4.3. These simulations were done for inlet flow rates of 1 and 5 L/h respectively (Initial values: x_o = 0.01 g/L, s_o = 2.8 g/L, p_o = 0 g/L and V_o = 2 L).

From Figures 4.2 and 4.3 it can be seen that the cells undergo exponential growth while a high substrate concentration is maintained in the fermenter. However, because the cells are multiplying rapidly and have a high metabolism, more substrate is required and therefore the substrate concentration starts to decrease. Due to this substrate depletion the bacteria are

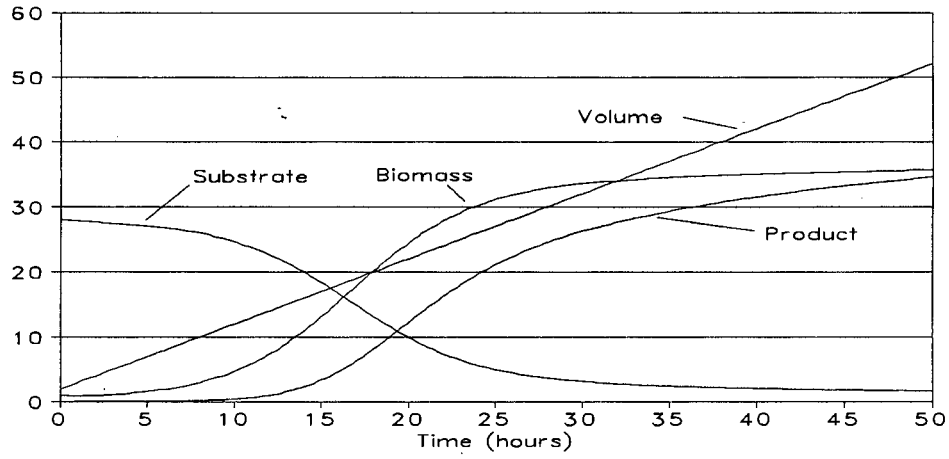

Figure 4.2. Lysine open loop response with the inlet flow rate at 1 L/h; biomass (g/L)($\times 10^{+2}$), substrate (g/L)($\times 10^{+1}$), product (g/L), volume (L). (s_i = 2.8 g/L)

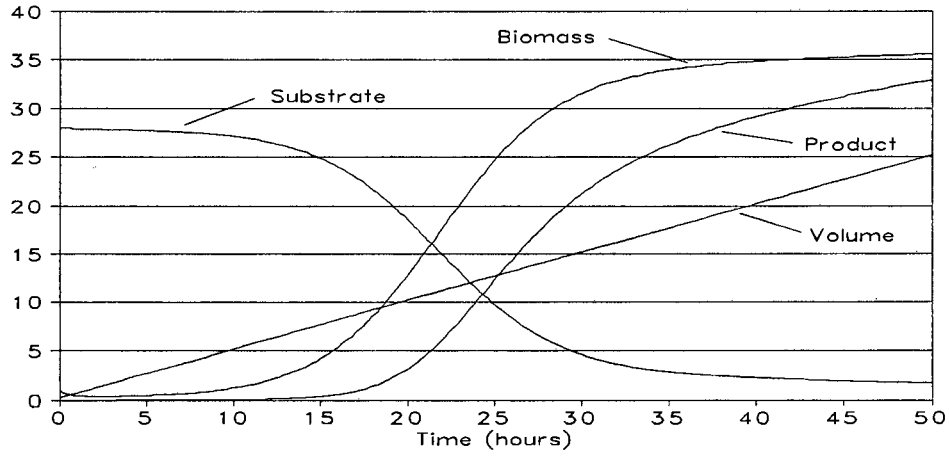

Figure 4.3. Lysine open loop response with the inlet flow rate at 5 L/h; biomass (g/L)($\times 10^{+2}$), substrate (g/L)($\times 10^{+1}$), product (g/L), volume (L)($\times 10^{-1}$). (s_i = 2.8 g/L)

forced from the exponential growth phase into a stationary growth phase.

From Figures 4.2 and 4.3 it can also be seen that the higher inlet flow rate yields a longer exponential growth phase for the biomass. The higher flow rate also leads to a longer suppression of the product formation as seen in Figure 4.3. Even though relatively the same concentrations are obtained at the end of the 50 hours for the two simulations, the second simulation results in a much larger final broth volume. Therefore the second simulation

yields a greater mass of product (lysine) at the end of the fermentation which represents a greater income. However a higher mass of glucose is fed to the fermenter over the 50 hours for the second simulation because of the higher flow rate used, and therefore the expenses are also higher.

Because a given fermenter has a finite volume, a final broth volume should be specified. It was decided to utilise 50 L in this study. Therefore the results from both simulations were compared when they had reached their final volume (48 and 9.6 hours respectively). Now it can be seen that a much lower product concentration is obtained for the higher flow rate (cf. 0.02 g/L for 5 L/h and 34.083 g/L for 1 L/h). This shows that by simply increasing the substrate feed rate will not necessarily lead to a higher production of lysine. The Fortran program listing for the open loop control can be found in Appendix M.

4.2 Penicillin Fermentation

4.2.1. Model of fermentation

Penicillin is used for the treatment of bacterial diseases in both humans and animals. It is also used as growth stimulants in pigs and poultry (Snell and Ettre,1972). Penicillin exports from the United States of America totalled 104 thousand kilograms at a value of US$ 8.3 million for 1992 (Shon,1992).

Fed-batch fermentation of penicillin closely resembles lysine fermentation by *Corynebacterium glutamicum* and in both cases the product is formed during the stationary phase of cell growth i.e. under conditions where a low specific growth rate is maintained in the fermenter. The objective is firstly to maximise the biomass concentration by utilising a high substrate concentration in the fermenter and then to maximise the production of penicillin by maintaining a low specific growth rate (lower substrate concentration in the fermenter). The fermenter substrate concentration is controlled by varying either the concentration of substrate in the feed stream or the feed stream's flow rate.

The following mass balances are used for the fed-batch penicillin fermentation (San and Stephanopoulos, 1989):

$$\frac{dxV}{dt} = \mu x V$$

$$\frac{dsV}{dt} = Fs_i - \frac{\theta x V}{Y_p} - \frac{\mu x V}{Y_x} - M_x x V$$

$$\frac{dpV}{dt} = \theta x V - KpV$$

$$\frac{dV}{dt} = F$$

where:

x	is the biomass concentration [g/L]
s	is the substrate concentration [g/L]
p	is the product concentration [g/L]
V	is the fermenter broth volume [L]
μ	is the specific growth rate of the cells (equation 8) [h^{-1}]
F	is the volumetric flow rate of the inlet stream [L/h]
s_i	is the substrate feed concentration [g/L]
θ	is the specific product formation rate (equation 9) [h^{-1}]
Y_p	is the yield of product from substrate
Y_x	is the yield of biomass from substrate
M_x	is the term for cell maintenance [h^{-1}]
K	is a product first order decay rate [h^{-1}].

The specific cell growth rate in the above equation is given by the Contois model:

$$\mu = \frac{\mu_m s}{s + K_1 x} \tag{8}$$

and the specific production rate is determined by a substrate inhibition model:

$$\theta = \frac{\theta_m}{1 + \frac{K_p}{s} + \frac{s}{K_i}} \tag{9}$$

where μ_m, K_1, θ_m, K_p and K_i are constants.

The mass balance equations can be rewritten as follows:

$$\begin{aligned}
\frac{dx}{dt} &= \mu x - \frac{F}{V}x \\
\frac{ds}{dt} &= \frac{F}{V}(s_i - s) - \frac{\theta x}{Y_p} - \frac{\mu x}{Y_x} - M_x x \\
\frac{dp}{dt} &= \theta x - Kp - \frac{F}{V}p \\
\frac{dV}{dt} &= F
\end{aligned} \tag{10}$$

The first term in the biomass equation represents an increase in the mass of cells due to growth. In the second equation, the first term represents an increase in fermenter substrate due to the feeding of the substrate while the second and third terms represent a depletion of substrate due to product formation and cell growth respectively. The last term represents a loss in the amount of substrate due to its degradation. In the third equation the first and second terms represent the formation of product due to cell growth and the loss of product due to its degradation respectively.

The values of the model parameters are summarised in Table 4.1.

Table 4.1. Model parameters (San and Stephanopoulos,1989).

μ_m	0.11 (h^{-1})
K_1	0.006
θ_m	0.004 (h^{-1})
K_p	0.0001 (g/L)
K_i	0.1 (g/L)
K	0.01 (h^{-1})
Y_x	0.47
Y_p	1.2
M_x	0.029 (h^{-1})
V_o	250,000 (L)
V_f	500,000 (L)

These modelling equations (equation 10) were integrated over a time period of 140 hours, utilising a Fortran Nag routine, with a time step of 3 minutes to yield the state trajectories.

4.2.2. Open loop response

The open loop response of the model was investigated at different substrate feed concentrations as shown in Figures 4.4, 4.5 and 4.6 These simulations were done for inlet substrate concentrations of 500, 1500 and 2500 g/L respectively (Initial values: x_o=1 g/L, s_o=1 g/L, p_o=0 g/L, V_o=250000 L).

The inlet flow rate for these simulations was taken as a constant value of 1785.7 L/h to ensure that the final volume (500,000 L) was attained at the end of the 140 hours. It can be seen for these figures that the higher the inlet substrate concentration, the higher is the final concentration of biomass and hence a higher product concentration is attained. The biomass concentration is seen to start dropping off after about 80 hours for all three cases because the cells have entered the stationary growth phase and their numbers are thus increasing much slower relative to the rate of increase in the fermenter's volume. The higher the inlet substrate concentration, the longer the exponential stage of cell growth is maintained and therefore more biomass is accumulated in Figures 4.5 and 4.6.

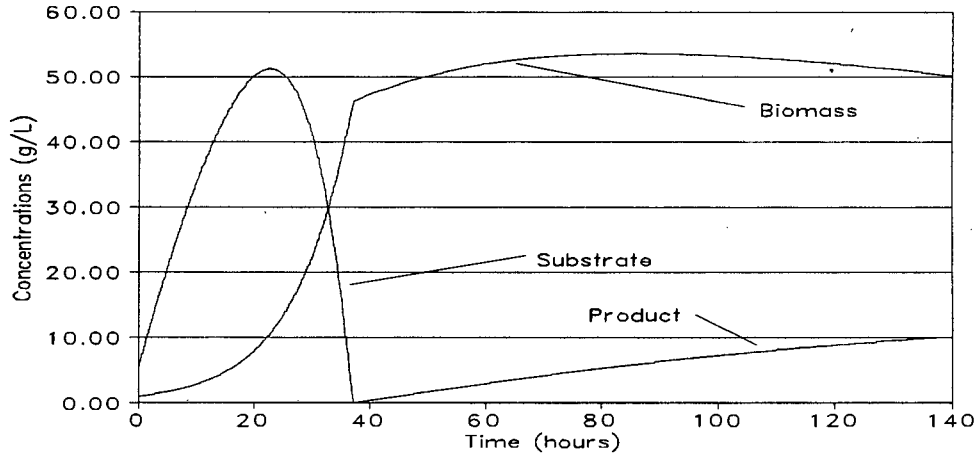

Figure 4.4. Open loop response with the inlet substrate concentration at 500 g/L.

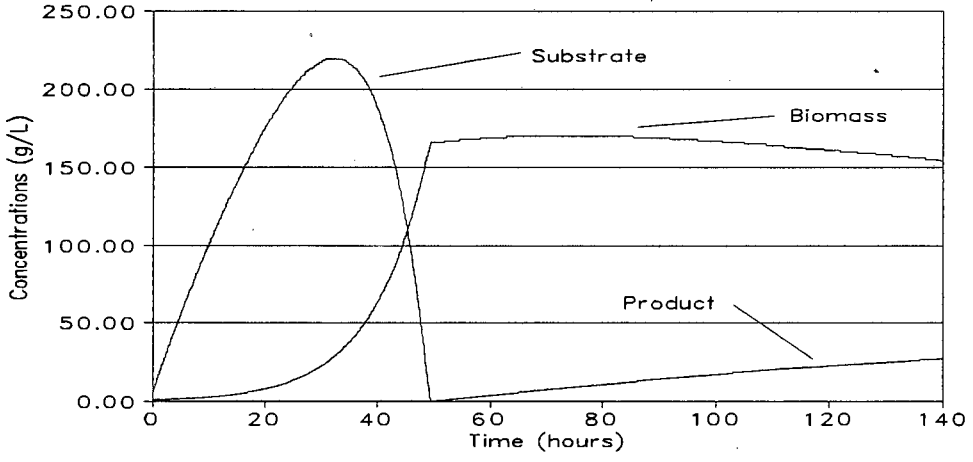

Figure 4.5. Open loop response with the inlet substrate concentration at 1500 g/L.

The fermenter substrate concentration is found to pass through a maximum during the exponential growth phase of the cells. This is because the cells in this stage are rapidly multiplying and have a high metabolism and therefore require increasing amounts of substrate. When the substrate concentration is reduced to a minimum, the cells are forced from the exponential growth phase into the stationary growth phase because their specific growth rates have been reduced. The Fortran program listing for the open loop response can be found in Appendix N.

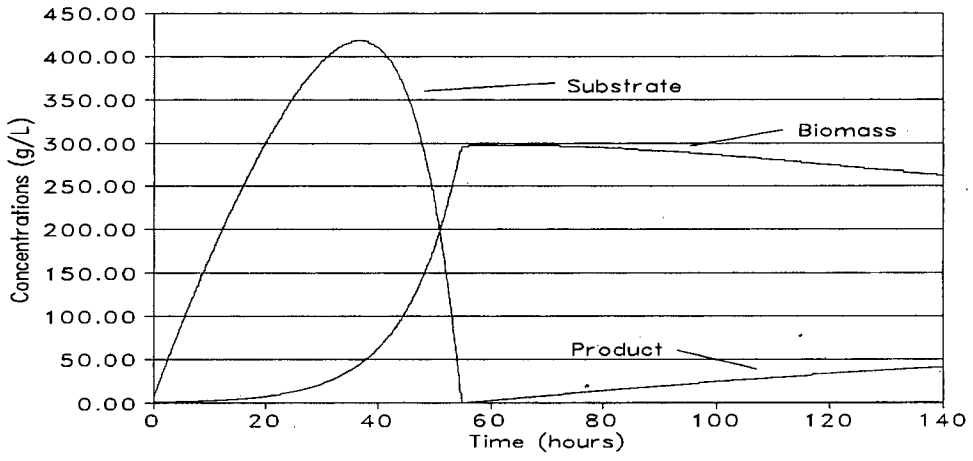

Figure 4.6. Open loop response with the inlet substrate concentration at 2500 g/L.

5. APPLICATION OF FUZZY CONTROL TO FED-BATCH FERMENTATION

In this chapter the control of lysine and penicillin fermentation is investigated. The modelling equations presented in Sections 4.1.4 and 4.2.1 are used to represent the process. Again the Fortran Nag routines were used to integrate the modelling equations. A step size of 0.1 hours was used for the penicillin fermentation and 0.2 hours for the lysine fermentation.

5.1. Regulatory Type Control

In Chapter 3 it was found that the Qin type fuzzy controller (presented in Sections 3.6 to 3.8) yielded better input as well as output response than an equivalently tuned traditional linear PI controller for a first order system with dead time, a second order system and a third order system with dead time. It was decided to implement the fuzzy controller on both the penicillin and lysine fermentation models. As these are highly nonlinear systems, the nonlinear structure of the controller might be expected to yield better control than a standard PI controller which is designed around linearised systems.

5.1.1. Lysine Fermentation

A prescribed set point trajectory as determined by Modak and Lim (1987) was specified for the fermenter substrate concentration and the inlet stream's flow rate was manipulated in order to maintain the set point value throughout the fermentation. Because of the difficulties of obtaining the Ziegler-Nichols tuning parameters for this system, the fuzzy scaling factors were optimised by trial and error to ensure good set point tracking with minimum oscillation.

Because a variation in the inlet substrate concentration could occur if its level is not constantly monitored, it was decided to treat this as a disturbance to the system. The fuzzy controller was compared to the PI controller with a disturbance of 20% in the concentration of the substrate in the feed stream for the first five hours of the fermentation i.e. an inlet

concentration of 2.24 g/L. The following PI parameters were used:

$K_{PI} = 3.$

$\tau_{PI} = 0.1$

Figures 5.1 and 5.2 show the output and input responses respectively.

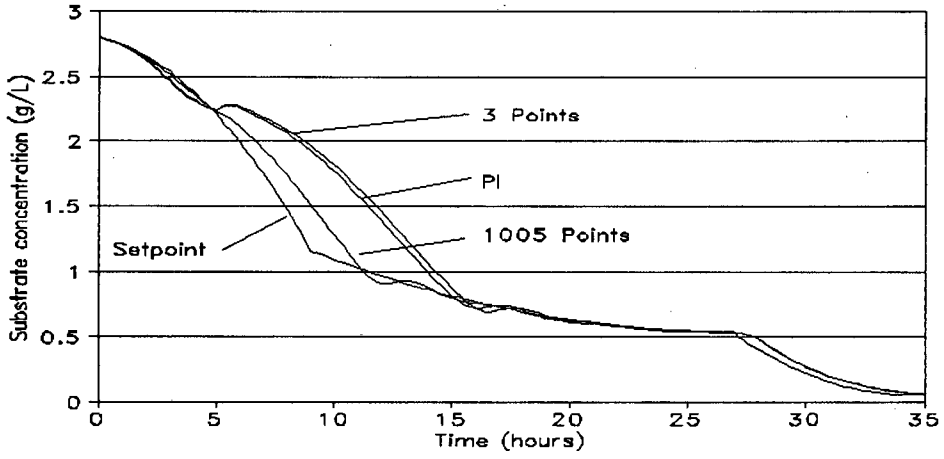

Figure 5.1. Regulatory control; output response of the lysine fermentation with a disturbance.

During the first 5 hours, when the disturbance is applied, higher flowrates are utilised to ensure that the setpoint is attained as seen in Figures 5.1 and 5.2. However when the disturbance is removed after 5 hours the mass flow rate of substrate into the fermenter is too high and therefore the substrate level in the fermenter is seen to increase in Figure 5.1. The controllers therefore decrease the inlet flow rate to zero to ensure that the fermenter substrate level is decreased as fast as possible. It can be seen that the PI controller yielded comparable control to that of the fuzzy controller with 3 output sampling points. However, a higher number of sampling points results in better input and output response. It was found that the input and output responses reach a limit as the number of output sampling points is increased. The fuzzy controller is able to deal more effectively with the disturbance with 1005 points displaying better input and output response.

The Fortran program listing for the fuzzy closed loop control of the fermentation can be found in Appendix O while the PI controlled simulation is given in Appendix P.

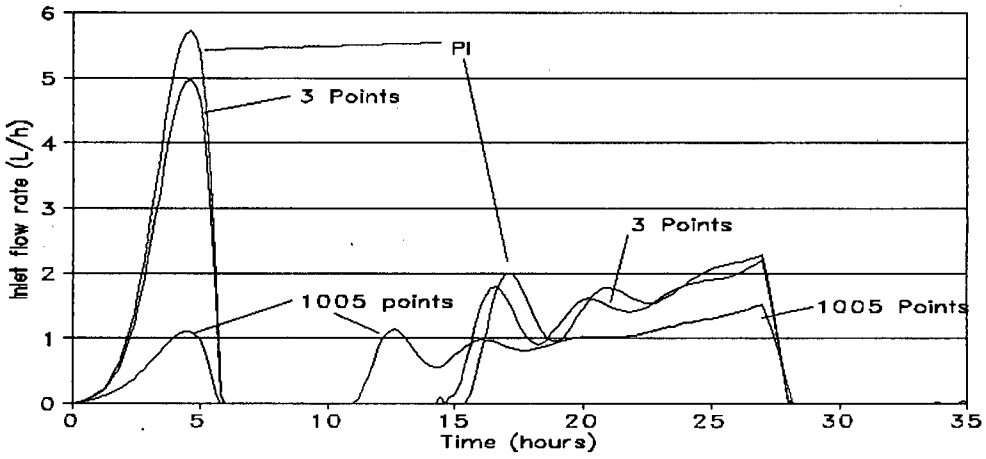

Figure 5.2. Regulatory control; input response of the lysine fermentation with a disturbance.

5.1.2. Penicillin Fermentation

A prescribed trajectory taken from San and Stephanopoulos (1989) was specified for the fermenter substrate concentration. The inlet stream's flow rate was used as the manipulated variable. A substrate feed concentration of 100 g/L was utilised and a disturbance of magnitude 65 g/L was applied to the system between 5 and 10 hours after the start of the fermentation ie. utilising an inlet concentration of 35 g/L. Once again high inlet flowrates are used during the period of the disturbance to counterbalance the decreased inlet concentration as seen in Figure 5.4. Once the disturbance is removed, at the end of 10 hours, the inlet flow rate is too high and hence the inlet substrate mass flow rate is too high. This is corrected by utilising lower inlet flowrates. A ß value of 0.04 was used to obtain GR for three output sampling points while a ß of 0.001 was found to yield the best output response for 1005 output sampling points.

In Figure 5.3 it can be seen that good set point tracking is maintained by both the PI and the fuzzy controller despite the high degree of nonlinearity inherent in the system. However, in Figure 5.4 it can be seen that, although the PI controller shows good output performance, the input response is far worse than that of the fuzzy controller. It was possible to obtain a lower overshoot after 20 hours for the fuzzy controller with 1005 output sampling points than using 3 output sampling points.

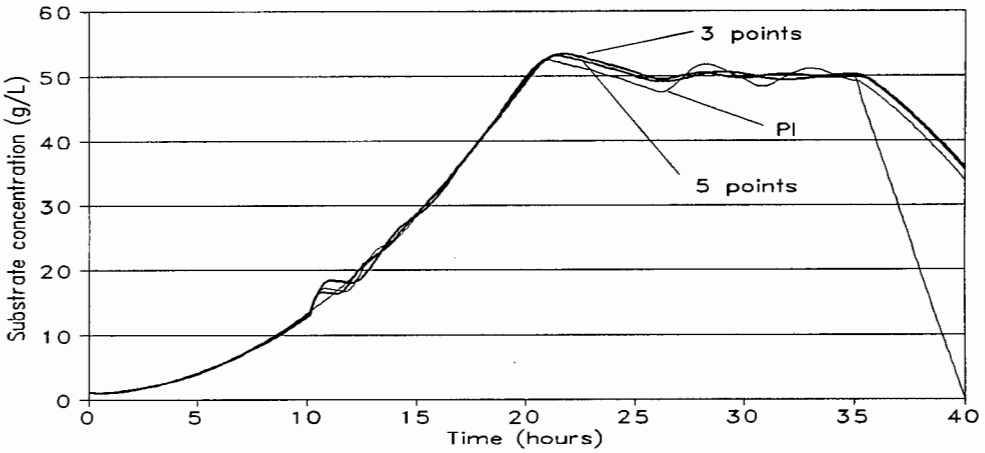

Figure 5.3. Output control response for the penicillin fermentation; ■ setpoint (L = 1).

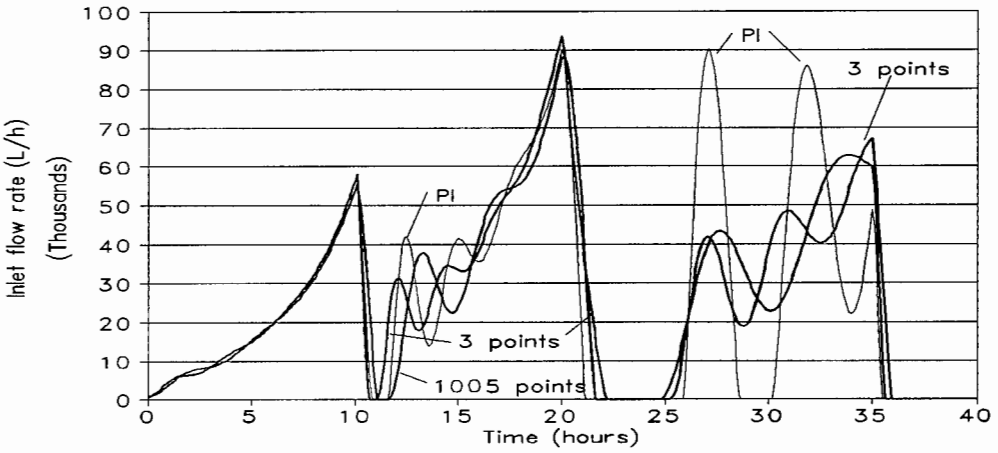

Figure 5.4. Input control response for the penicillin fermentation (L = 1).

Figure 5.5, a magnified section of Figure 5.3, shows that the fuzzy controller with three output sampling points has the lowest overshoot after the disturbance and is also the first to settle down. The response for a high number of sampling points is less desirable as it takes longer to settle down. The fuzzy controller is able to deal effectively with the high degree of nonlinearity in the system.

The Fortran program listing for the fuzzy closed loop control of the fermentation can be found in Appendix Q while the PI controlled simulation is given in Appendix R.

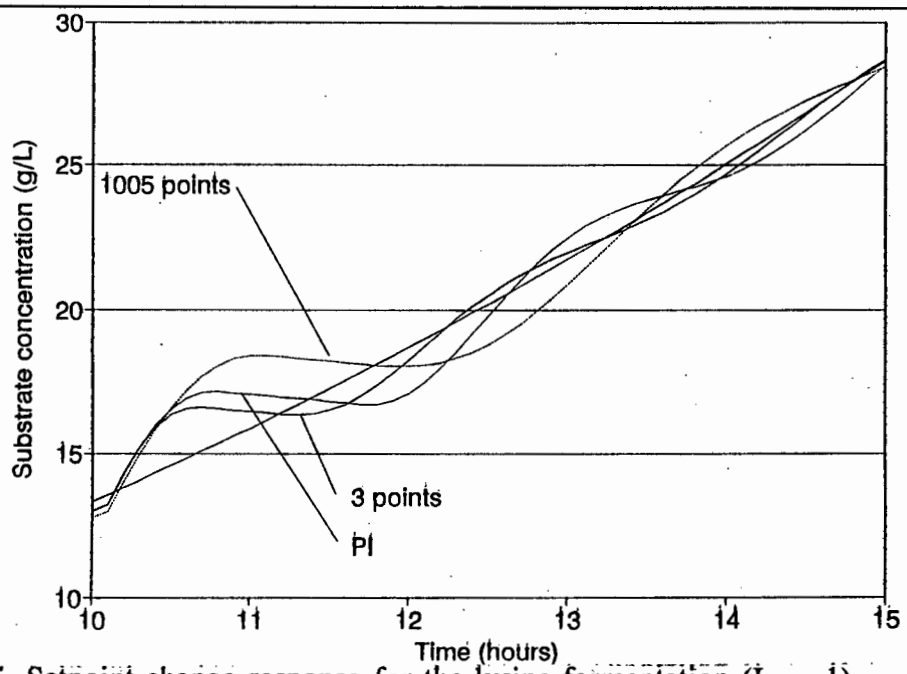

Figure 5.5. Setpoint change response for the lysine fermentation (L = 1).

5.2. Heuristic Type Control

Instead of regulatory type control about a predetermined fermentation trajectory (as investigated in the previous section), this section deals with the development of a heuristic fuzzy controller which is to maximise the ratio of the lysine produced to the amount of substrate utilised i.e. maximise the income received over the expenditure incurred. The pH, temperature and dissolved oxygen level in a fermenter are usually effectively controlled using simple analog controllers (San and Stephanopoulos,1984). Therefore only the control of the substrate addition will be investigated here.

5.2.1. Choice of fuzzy controller input

5.2.1.1. Gas analysis

Kiss and Stephanopoulos (1991) suggest using the carbon dioxide evolution rate (CER) and the oxygen utilisation rate (OUR) as the controlled variables. These values are easy to measure on-line and are able to give a good indication of the dynamics of the fermentation.

In order to determine the CER and OUR from the mathematical model, one needs to consider a carbon and oxygen balance respectively. Bioreactions are usually summarised as follows (Atkinson and Mavituna,1983; Nagai,1979):

$$\begin{aligned} &CH_{sH}N_{sN}O_{sO}P_{sP}S_{sS} + a\,H_2SO_4 + b\,H_3PO_4 + c\,NH_3 + d\,O_2 \\ &\rightarrow e\,CH_{bH}N_{bN}O_{bO}P_{bP}S_{bS} + f\,CH_{pH}N_{pN}O_{pO}P_{pP}S_{pS} + g\,CO_2 + h\,H_2O \end{aligned} \qquad (11)$$

For lysine production:

$$sH = 2,\ sN = 0,\ sO = 1,\ sP = 0,\ sS = 0$$
$$bH = 1.8,\ bN = 0.2,\ bO = 0.5,\ bP = 0,\ bS = 0$$
$$pH = 2.667,\ pN = 0.333,\ pO = 0.333,\ pP = 0,\ pS = 0$$

where the first subscripts 's', 'b' and 'p' denote the substrate, biomass and product respectively.

Equation 11 shows the conversion of substrate to biomass, product, carbon dioxide and water respectively. By performing an elemental carbon balance the net molar conversion to CO_2 can be obtained:

$$\Delta c = \Delta s - \Delta b - \Delta p$$

where: Δc, Δs, Δb and Δp are the net conversion (in moles) of the carbon dioxide, substrate, biomass and product respectively. These values are easily obtained from the fermentation model in equation 4, Section 4.1.4.

However, elemental carbon balances, using the results of the model, show that more carbon is being formed than that which is used. This indicates that carbon is also being derived from other sources (other than just glucose) such as amino acids in the feed stream and in the initial charge of the fermenter. However it is not possible to determine the dynamics of these other carbon sources as usually only the concentration of the limiting substrate (glucose) is sufficient in order to model the production of biomass and product.

Similarly the unmodelled sources of carbon also contain oxygen and therefore makes it impossible to determine either the CER or the OUR from the given fermentation model. Therefore it is not possible to utilise the CER and OUR as fuzzy controller inputs for the computer simulated control of the fermentation without a more elaborate model.

5.2.1.2. Specific growth rate

Both Ohno *et al* (1976) and Modak and Lim (1987) considered the case where the fermentation of lysine is started in the batch phase and is then followed by a fed-batch stage as summarised in Sections 2.1.1 and 2.1.2. Both sets of authors determined the optimal substrate feed trajectory which will maximise a particular cost function. A drawback with Ohno *et al's* (1976) approach is that the optimal trajectory is derived for an ideal model of the fermentation. If model imperfections are present then the trajectory is no longer optimal. A limitation of Modak and Lim's (1987) approach is that their control scheme relies on a free final fermentation time, t_f, and that the fermenter states (biomass, substrate and product concentrations) can accurately be measured on-line. The concentrations of biomass, substrate and product are not usually measurable on-line and are only observable through estimation techniques (Stephanopoulos and San,1984). The concentrations can be determined by doing off-line analysis of samples. However, the off-line analysis usually takes a long time and therefore the information is not available at the time at which it is needed. Modak and Lim (1987) also state that their control scheme is heavily dependent on the reliability of the kinetic parameters of the model and that uncertainties in the parameters can lead to deterioration in the optimisation technique.

It was therefore decided to implement a fuzzy controller which would utilise the bacteria's specific growth rate. The specific growth rate was chosen as it gives a good indication of the fermentation's development and is readily estimated from on-line readings of the specific product formation rate (Bastin and Dochain,1986) or readings of OUR and CER (Stephanopoulos and San,1984).

5.2.2. Initial conditions and constraints

The fermentation's initial values were chosen as:

x_o =	0.02 g/L	(initial biomass concentration)
s_o =	2.8 g/L	(initial substrate concentration)
p_o =	0.0 g.L	(initial product concentration)
V_o =	5 L	(initial volume)

The final volume, V_f, was taken to be 20 L, or as close to this value as possible, due to the given fermenter's size limitations. These are the same constraint and initial values used by Modak and Lim (1987). Because the initial fermenter substrate concentration is high, the fed-batch stage will follow an initial batch stage in the fermentation. This is suggested by Modak *et al* (1986) as seen in Figure 2.1 in Chapter 2. The glucose concentration in the feed stream was again chosen as 2.8 g/L while the final fermentation time, t_f, was chosen as 35 hours.

5.2.3. Fermentation mode change

In order to compare the results of one fermentation with another it was decided to introduce a profit ratio:

$$profit\ ratio = \frac{p_f V_f}{\sum_{k=t_1}^{t_2} F_k s_i \Delta t + s_o V_o}$$

where:

- f,o indicate the state values at the end and start of the fermentation respectively
- t_1 is the time that the fed-batch stage is begun (h)
- t_2 is the time that the fed-batch stage is terminated (h)
- F_k is the feed flow rate in the k^{th} time interval (L/h)
- s_i is the concentration of substrate (glucose) in the feed stream (g/L)

Δt is the time step utilised in integrating the modelling equations (0.2 h)

The numerator of the profit ratio depicts the mass of product (lysine), in grams, available at the end of the fermentation. The first term of the denominator calculates the mass of substrate (glucose) that enters the fermenter via the feed stream during the fed-batch stage of fermentation. The second term depicts the amount of substrate that forms part of the initial reactor charge.

One needs to determine when to switch from batch to fed-batch fermentation (i.e. when to start feeding, t_1). Therefore it was decided to examine the dynamics of the fermenter constituents and of the specific growth rate and product formation rate for an entirely batch fermentation. The results of computer simulations are shown in Figures 5.6 and 5.7. A low profit ratio of 9.0172 g/g was obtained whereas Modak and Lim (1987) obtain a value of 12.6614 g/g determined by utilising the optimum feed profile of Modak and Lim (1987). This indicates that a fed-batch stage is required.

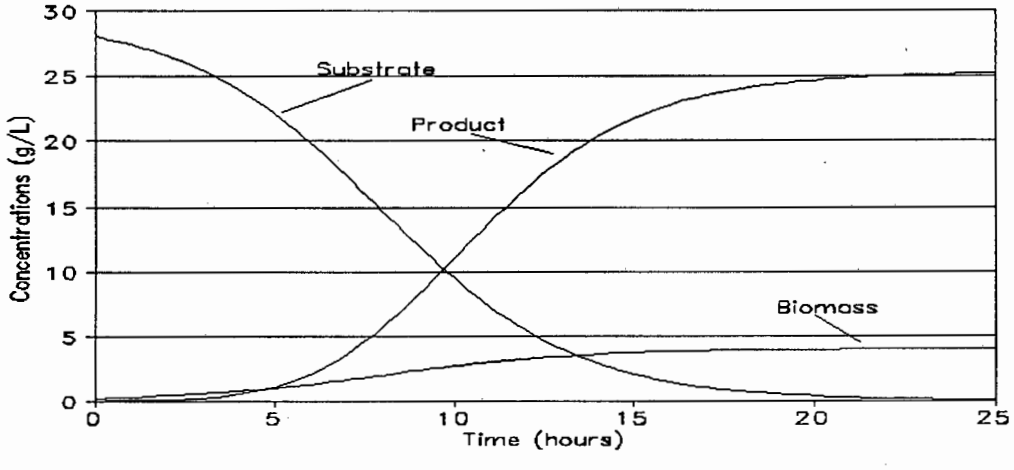

Figure 5.6. Batch lysine fermentation; biomass ($\times 10^{+1}$), substrate ($\times 10^{+1}$).

In Figure 5.7 it can be seen that the specific product formation rate, Q_p, increases, reaches a maximum and then begins to decrease. It was decided to commence the feeding of the substrate, i.e. transferring from batch to fed-batch (t_1), straight after Q_p goes through a maximum. This point was chosen as it is the point at which the production of lysine starts

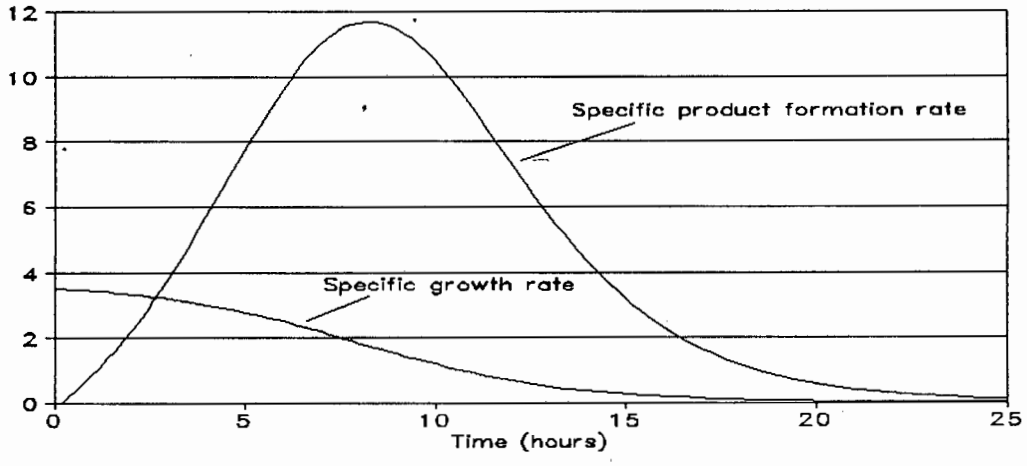

Figure 5.7. Batch lysine fermentation; specific growth rate (h^{-1}) ($\times 10^{+1}$), specific product formation rate ($g_{prod}/g_{biomass}.h^{-1}$).

to decline. It was decided also to stop feeding, i.e. return to batch fermentation (t_2), as soon as the final volume of 20 L is attained.

5.2.4. Choice of fuzzy rules and sets

5.2.4.1. Fuzzy rules

During the fed-batch stage a compromise must be drawn between the need to keep the specific growth rate, μ, nearer to 0.174 (the value of μ for which Q_p is a maximum determined from equation 6, Section 4.1.4) to achieve a good ratio of the product formed to the mass of biomass formed and maintaining μ low which will result in less substrate being used.

As soon as the fed-batch stage is started the change in the specific growth rate will be highly negative as seen at 8 hours, for the ideal model, in Figure 5.7. The change in the specific growth rate is highly negative because of the low substrate level available in the fermenter. Therefore the substrate feed rate should be high so that the drop in the μ will be attenuated and hence will attain a value which is closer to the optimum value of μ for which Q_p is

maximum. By doing this the formation of product will be favoured over the production of biomass. As the drop in the specific growth rate starts to reach an asymptote the substrate flow rate should be decreased. A positive change in the specific growth rate should be avoided by making a negative change in the substrate feed rate. The fuzzy controller was therefore structured with the following five rules:

1. If **Change** in **Specific Growth Rate** ($\Delta\mu$) is *Negative Large* (*NL*) then **Change** in **Flow rate** (Δ**F**) is *PL*.
2. If $\Delta\mu$ is *NM* then Δ**F** is *PM*.
3. If $\Delta\mu$ is *NS* then Δ**F** is *Z*.
4. If $\Delta\mu$ is *Z* then Δ**F** is *NM*.
5. If $\Delta\mu$ is *PM* then Δ**F** is *NL*.

(*Negative Large* (*NL*), *Negative Medium* (*NM*), *Negative Small* (*NS*), *Zero* (Z), *Positive Medium* (*PM*), *Positive Large* (*PL*))

These rules and fuzzy sets were chosen to attempt to attenuate the drop in μ after the start of the fed-batch stage. The attenuation of the drop in μ would then lead to an increase in the ratio of lysine to biomass concentration in the fermenter (i.e. the new value of μ should favour product formation over cell formation).

5.2.4.2. Optimization of fuzzy set locations

It was decided to utilise triangular shaped fuzzy sets. The placing (or positioning) of the input and output fuzzy sets is usually done on a trial and error basis. Athalye *et al* (1993) suggest utilising an optimisation technique. The authors utilised the Simplex Method.

Following the suggestions of Athalye *et al* (1993) it was decided to attempt to optimise the placement of the fuzzy sets using NPSOL, a Fortran optimisation routine. To reduce the number of variables which need to be varied, the following conditions were set:

1. The base of one fuzzy set (ie the point at which a value has zero membership in that

fuzzy set) corresponds with the peak of the adjacent fuzzy set as illustrated in Figure 5.8. For five input and five output fuzzy sets this means that there are 10 variables.

2. The apex of the fourth input fuzzy set (*input is zero*) should be centred at zero (as specified in Rule 4, Section 5.2.4.1). This leaves nine variables.

3. The apex of the third output fuzzy set (*change in output is zero*) should be centred at zero (as specified in Rule 3, Section 5.2.4.1). This leaves eight variables.

4. Each variable (either from the input or the output) is not allowed to assume a value which is higher than or equal to the value of the variable directly to the right of it:

$$V1 < V2 < V3 < 0 < V4$$

$$V5 < V6 < 0 < V7 < V8$$

Figure 5.8 shows the fuzzy sets used for the controller input (change in specific growth rate) and the controller output (change in the feed stream's flow rate).

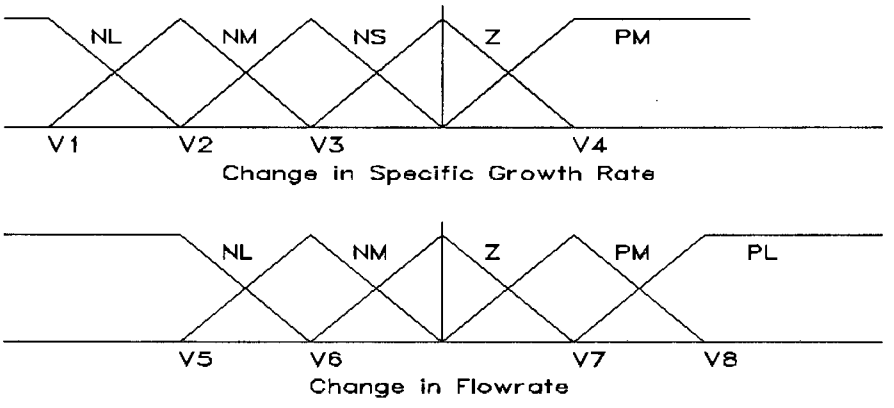

Figure 5.8. Fuzzy sets used (Negative (N), Positive (P), Large (L), Medium (M), Small (S), Zero (Z)).

The profit ratio was inverted to yield a cost ratio. The NPSOL optimisation routine was then

used to attempt to minimise this cost ratio by manipulating the position of the eight variables (positions of the input and output fuzzy sets).

Unfortunately the optimisation technique was not able to determine the optimal placing of the eight variables and repeatedly jammed while trying to minimise the cost ratio. A possible reason for this is discussed in Section 5.2.6. The Fortran program used for the optimisation can be seen in Appendix S.

5.2.4.3. Random placing of fuzzy sets

Because the optimisation technique had failed in the placing of the fuzzy sets it was decided to investigate their placing through a semi-random technique. Table 5.1 shows the minimum and maximum values assigned for V1, V4, V5 and V8 which in turn bounded the remaining variables.

Table 5.1. Minimum and maximum values for the variables.

Variable	Minimum	Maximum
V1	-0.007	-0.0001
V4	0.0001	0.001
V5	-1.0	-0.001
V8	0.001	1.0

$V1_{min}$ was chosen as -0.007 as this is the value of the change in the specific growth rate (determined by simulation) when the specific product formation rate is at its maximum as seen in Figure 5.7 (ie the time at which one should change from batch to fed-batch) and represents the lowest value that $\Delta\mu$ would attain during the fermentation (for the ideal model case). $V1_{max}$, $V4_{min}$, $V5_{max}$ and $V8_{min}$ were chosen so that condition 4 (Section 5.2.4.2) is not violated. As the specific growth rate assumes very small values, $V4_{max}$ was chosen to be a small value. $V5_{min}$ and $V8_{max}$ were chosen to be moderate values to avoid large input changes.

A True Basic program, shown in Appendix T, was used to generate random real numbers

between 0.0 and 1.0 which were then utilised as follows to determine the values of the eight variables:

$$V1 = V1_{max} - (V1_{max} - V1_{min}) \times random_1$$

$$V4 = V4_{min} + (V4_{max} - V4_{min}) \times random_2$$

$$V3 = V1 - V1 \times random_3$$

$$V2 = V1 + (V3 - V1) \times random_4$$

$$V5 = V5_{max} - (V5_{max} - V5_{min}) \times random_5$$

$$V8 = V8_{min} + (V8_{max} - V8_{min}) \times random_6$$

$$V6 = 0.0 + V5 \times random_7$$

$$V7 = 0.0 + V8 \times random_8$$

Appendix U contains a listing of the Fortran program used to find a good placing of the eight variables in order to minimise the cost ratio (maximise the profit ratio). The program enables 1000 simulations to be done. During each simulation eight different random numbers are utilised in the calculation of the fuzzy sets' positions. At the end of each simulation the profit ratio is determined and printed to a results file together with the random points used. Therefore by running the program several times an exhaustive search is carried out. The results were then analysed using Quattro Pro Ver 5 to find the optimal placing of the input and output fuzzy sets.

5.2.5. Results obtained for the ideal model

It was found that the following fuzzy set placing (derived by the semi-random technique) yielded good control of the lysine fermentation as seen in Figures 5.9 and 5.10 (for the ideal model):

V1 = -3.9368E-4	V5 = -3.6615E-2
V2 = -7.7080E-5	V6 = -2.7671E-2
V3 = -9.1880E-6	V7 = 7.3740E-3
V4 = 3.9351E-4	V8 = 2.5660E-2

30 output sampling points used

Figure 5.11 shows that by using the fuzzy controller (i.e. instead of just using batch fermentation) that the ratio of product to biomass concentrations has indeed increased as desired in Section 5.2.4.1. This shows that the fed-batch stage was successful in increasing the formation of lysine over the formation of biomass. The transfer from batch to fed-batch was after 8.6 hours (t_1) and the tank reached its final volume at 28.2 hours (t_2).

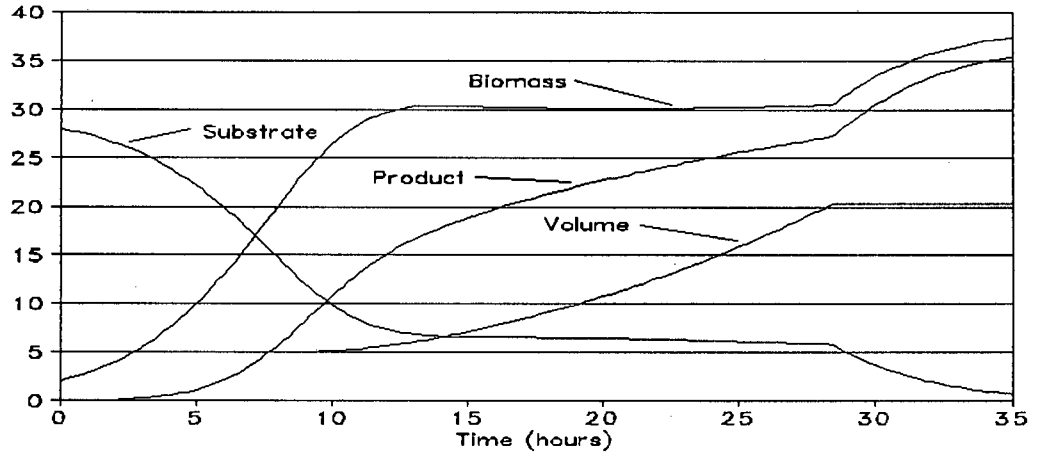

Figure 5.9. Results of fuzzy controller; biomass (g/L)($\times 10^{+2}$), substrate (g/L)($\times 10^{+1}$), product (g/L), volume (L).

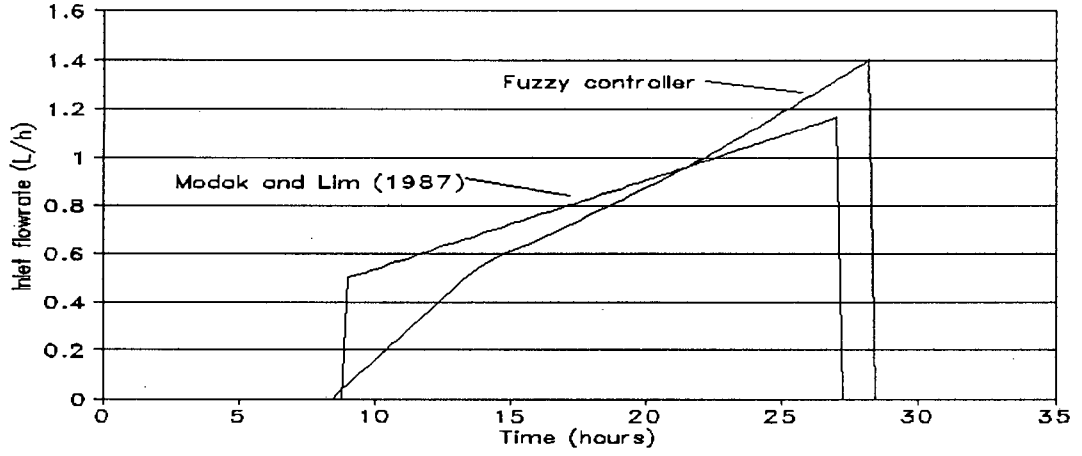

Figure 5.10. Input response for the ideal model.

Figure 5.10 shows that the input response obtained from the fuzzy controller has the same

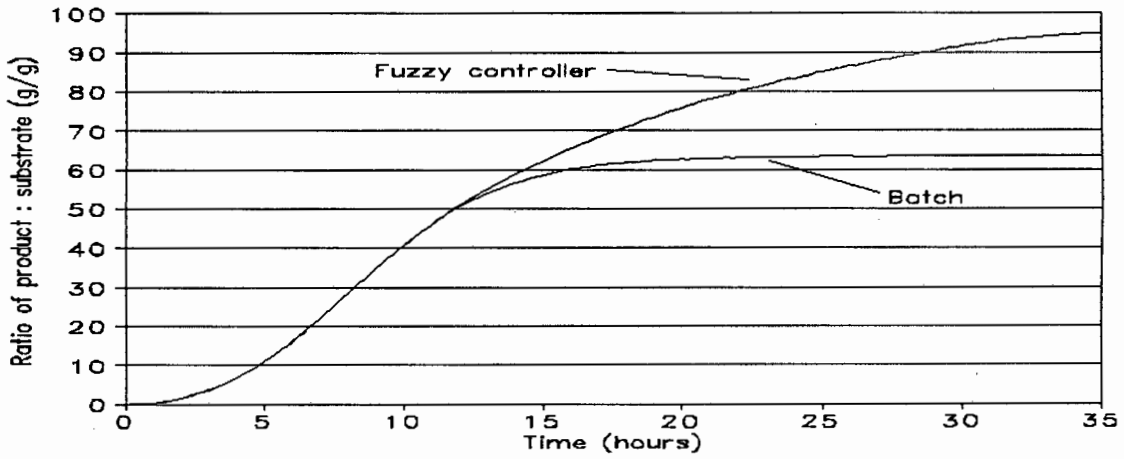

Figure 5.11. Comparison between the results of the batch fermentation and the fuzzy controlled fermentation.

characteristic shape as that suggested by Modak *et al* (1986) shown in Figure 2.1, case B, in Section 2.1.1. The input response also has the same general form to that found by Modak and Lim (1987). It was found that an improved profit ratio of 12.6844 was obtained over 12.6614 obtained by Modak and Lim (1987). This improvement is possible because the authors determined the optimum input trajectory with a different performance index which does not take into account the cost of the substrate. However, it should be stressed that, in this report, the results obtained from the fuzzy controller and from the preset trajectory of Modak and Lim were both compared using the profit ratio defined in Section 5.2.3.

5.2.6. Sensitivity analysis

5.2.6.1. Optimised variables

Because difficulties were observed when trying to find the optimal value of the variables using the NPSOL optimisation technique, it was decided to investigate the effect of slight variations in some of the variables' values on the product ratio.

It was decided to relate all subsequent product ratio results as a percentage of that obtained

by Modak and Lim (1987) for the ideal model:

$$profit\ percent = \frac{PR}{PR_{M\&L}} \times 100$$

where: PR is the product ratio obtained via the fuzzy controller

$PR_{M\&L}$ is the product ratio obtained by Modak and Lim (1987) for the ideal model (12.661392).

Figures 5.12 and 5.13 show the effect that a variation of V1 and V8 has on the profit percent respectively. Here V1 was varied between -0.007, the value of $V1_{min}$, and the value of V2 whereas V8 was varied between the value of V7 and 1.0, the value of $V8_{max}$.

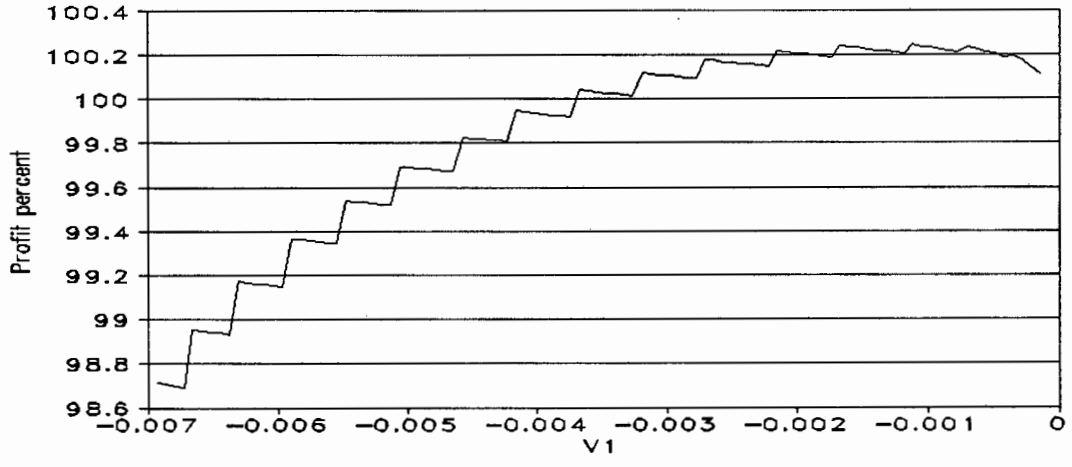

Figure 5.12. Effect of a variation in variable V1.

From Figure 5.12 it can be seen that an increase in V1 leads to a general increase in the profit percent. However, the function is not unimodal and has a number of local minima and maxima. Figure 5.13 shows that a variation in V8 does not result in a smooth profit percent function either. A variation in the other six variables delivered similar results. These results show that the profit percent (or the profit ratio) is not a simple unimodal function of the optimisation variables. The fact that there are many local maxima would explain why the NPSOL optimisation technique was unable to minimise the cost ratio and why it would often get 'stuck'.

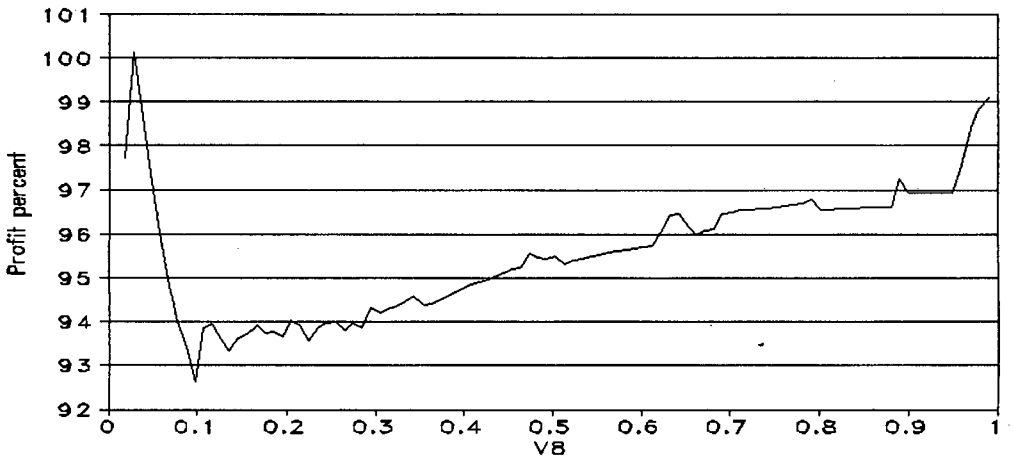

Figure 5.13. Effect of a variation in variable V8.

5.2.6.2. Number of output sampling points

Up until this point 30 output sampling points have been utilised. It was decided to investigate the effect of the number of sampling points on the profit percent obtained. This was done to determine if more or fewer sampling points should be utilised. Figure 5.14 shows that the profit percent reaches a maximum for the number of output sampling points above 25. Therefore an n of 30 was a good choice and will be maintained.

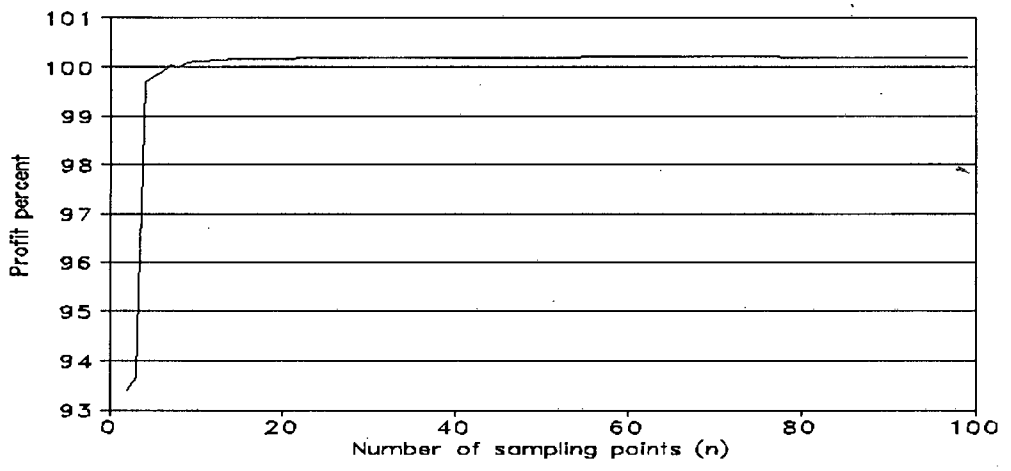

Figure 5.14. Effect of a variation in the number of output sampling points.

5.2.6.3. Number of fuzzy sets

Until this stage only five input and five output fuzzy sets have been considered. In this section the number of fuzzy sets, and hence the number of rules used, will be varied to determine the effect on the profit percent for the ideal model.

To simplify the calculations and programming it was decided to utilise equally spaced fuzzy sets between the values of V1 (-3.9368E-4) and V4 (3.9351E-4) for the input and between V5 (-3.6615E-2) and V8 (2.5660E-2) for the output (values obtained from Section 5.2.5). Once again 30 output sampling points were used for each simulation.

Figure 5.15 shows that the profit percent has a lowest value of 99.82% when between 25 and 100 fuzzy sets are utilised. The profit percent assumes a value above 100 for more than 130 sets and reaches a maximum if more than 325 fuzzy sets are used for both the input and output. This maximum is above the value of 100.182% found when five fuzzy sets were used in Section 5.2.5. Therefore it was decided to utilise the fuzzy controller with 350 fuzzy sets. Appendix V contains a listing of the Fortran program used to vary the number of input and output fuzzy sets.

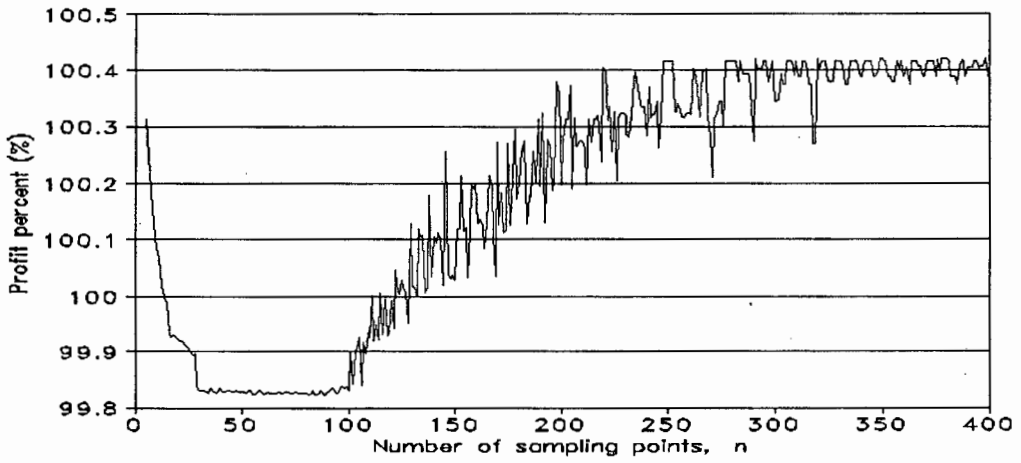

Figure 5.15. Effect of number of fuzzy sets on the profit percent.

5.2.7. Model uncertainty

An ideal model has thus far been assumed. Now the fuzzy controller specified in Section 5.2.6.3 is used to investigate the effect of a modelling error in the specific growth rate, because it (μ) serves as a good representation of the dynamic development of the fermentation. The specific growth rate is given by the following equation:

$$\mu = C \times s$$

where: C is the coefficient which will be varied and has a value of 0.125 for the ideal case

s is the concentration of the substrate in the fermenter

The coefficient, C, was varied over the range [0.1,0.15] i.e. by $\pm 20\%$. The results obtained from the fuzzy controller, at each value of C considered were compared with those obtained when the optimal feed trajectory of Modak and Lim (1987), derived for the ideal model, is utilised on the same system. The results are shown in Figure 5.16.

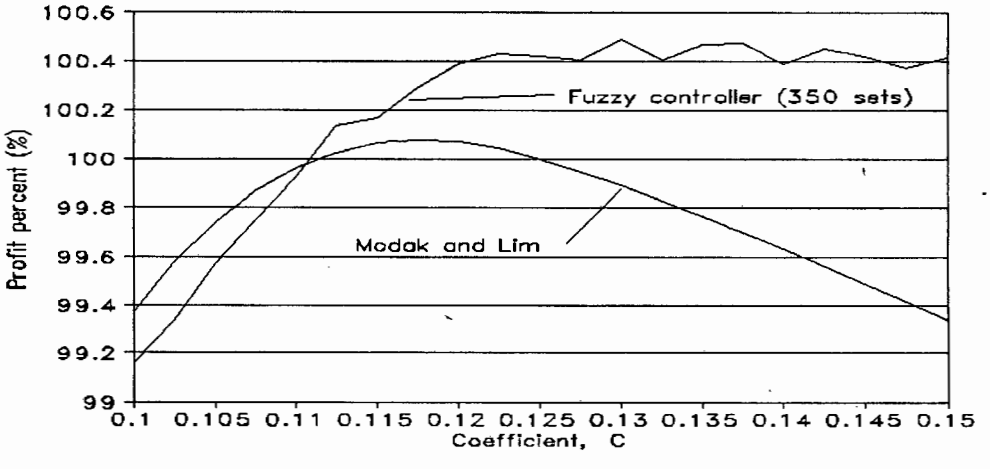

Figure 5.16. Effect of a variation in the coefficient of the specific growth rate.

In Figure 5.16 it can be seen that the fuzzy controller yields a superior result for the coefficient, C, above 0.1106. The fuzzy controller is also able to maintain a high profit percent for C above 0.125 whereas the preset trajectory of Modak and Lim results in much

lower profit percents as C is increased. The fuzzy controller is able to handle uncertainty in C better than the prescribed trajectory of Modak and Lim over a wide range of the coefficient.

5.2.8. Disturbance rejection

The feed stream's substrate concentration is not usually measured continuously and therefore any changes in this value acts as an unmeasured disturbance. The fuzzy controller specified in Section 5.2.6.3 was once again utilised and the results, obtained over the range of C specified in Section 5.2.7, were compared to that of Modak and Lim for a disturbance of 1.2 g/L in the inlet substrate concentration between 10 and 15 hours i.e. the feed stream has a substrate concentration of 4 g/L for 5 hours of the fermentation. The results are shown in Figure 5.17.

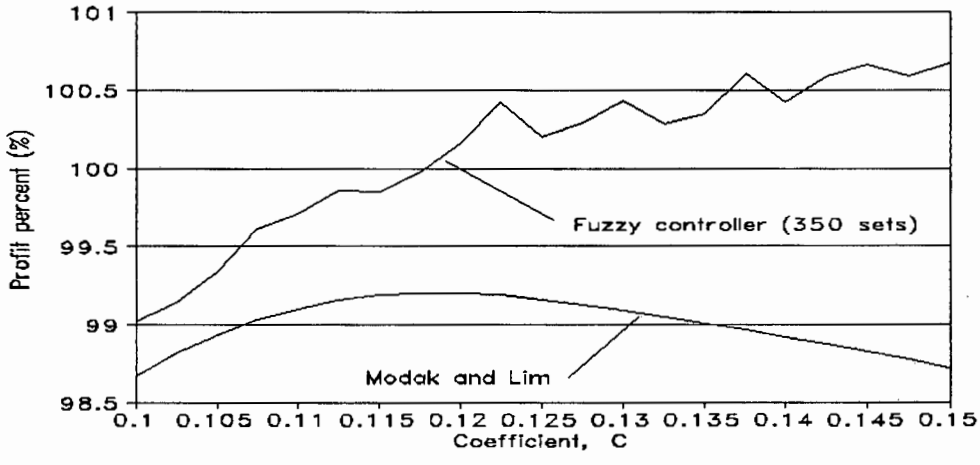

Figure 5.17. Disturbance rejection over a range of the coefficient, C.

Figure 5.17 shows that the fuzzy controller is able to deal better with the disturbance than the preset trajectory of Modak and Lim over the entire range that the coefficient, C, is investigated. It can also be seen that the fuzzy controller is still able to maintain a high profit percent over the range of C despite the disturbance whereas the trajectory of Modak and Lim yields much lower profit percents. Figure 5.18 shows the input response of the fuzzy controller for the system with a disturbance compared to the control of the fermentation

without a disturbance.

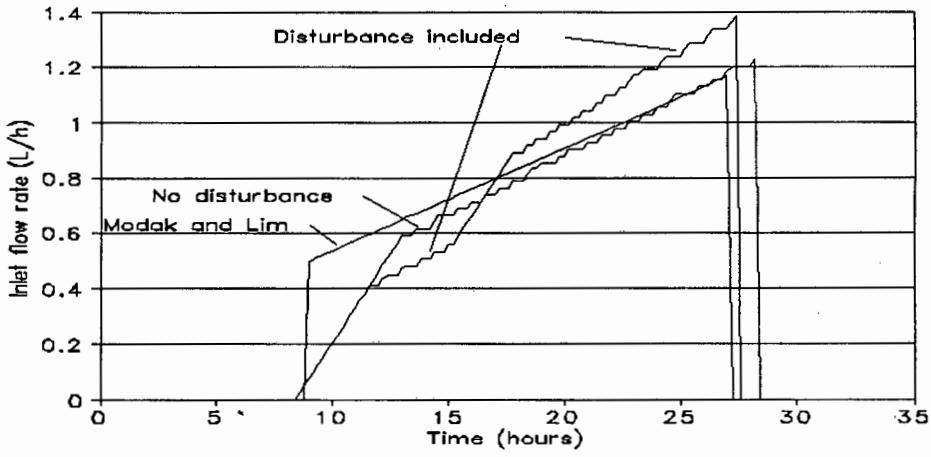

Figure 5.18. Disturbance rejection - input response of the fuzzy controller.

Figure 5.18 shows that the input response still has the same characteristic shape specified by Modak and Lim (1987). The disturbance (S_i = 4 g/L instead of 2.8 g/L) enters the system after 10 hours. The increased concentration in the feed stream leads to the increased substrate level in the fermenter as seen in Figure 5.19. The higher fermenter substrate concentration causes the specific growth rate to level off after 10 hours, as seen in Figure 5.20 and therefore the controller is able to utilise a lower flow rate than it would if no disturbance were present between 10 and 15 hours.

After 15 hours the disturbance is removed and the inlet concentration now has a lower value of 2.8 g/L. Because the inlet flow rate is low at this stage of the fermentation the mass flow rate of substrate entering the fermenter is suddenly lower. Therefore the fermenter substrate level starts dropping and hence the specific growth rate drops. To counter the effect of the drop in the specific growth rate the controller utilises a higher inlet flow rate as seen in Figure 5.18.

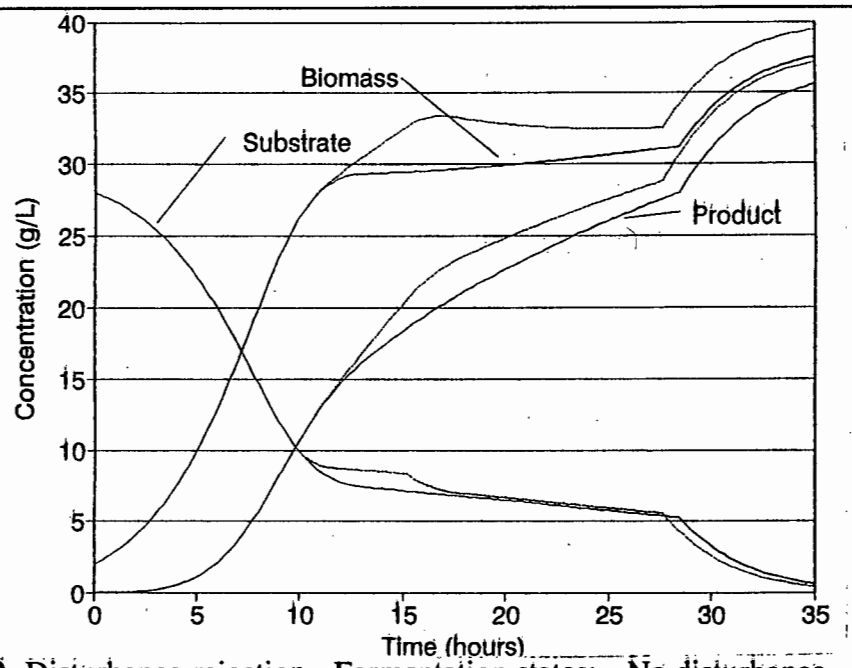

Figure 5.19. Disturbance rejection - Fermentation states; - No disturbance, --- Fermentation with disturbance.

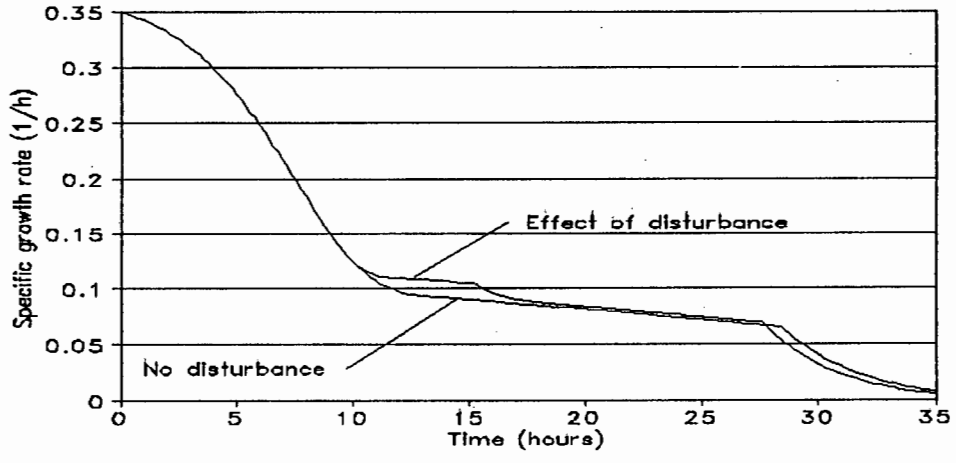

Figure 5.20. Disturbance rejection - output response of the fuzzy controller.

6. CONCLUSION

This dissertation has presented the results of an investigation into the application of fuzzy controllers to fed-batch fermentation. Fuzzy control was chosen as it is able to deal with systems whose operation does not easily fit into the mathematical framework of traditional control approaches and because it has been shown to be effective in the control of biosystems such as waste water treatment.

A literature review of the types of control and estimation techniques used for biosystems was included. This covers optimal control techniques, neural networks, fuzzy controllers and adaptive control techniques.

An overview of fuzzy sets, logic and control was presented. A specific form of fuzzy controller, presented in the literature, has been shown to correspond to a PI controller with a nonlinear gain. The gain is able to assume higher values than an equivalently tuned linear PI controller as the system is further away from its desired state. During the investigation of the controller it was found that the number of output sampling points used, n, has an effect on the output and input response. In the literature only three output sampling points were utilised. It was found that when five sampling points are used both the gain and integral constant are nonlinear functions of the system's relative position to the desired state. Nine sampling points still yields a controller which is PI in form with the gain and integral constant able to assume an even wider range of values. It was found that a higher value of n results in the gain having more resolution and that the output response equations are of a PI form with a possible bias term irrespective of the value of n.

Through extension of the derived generalised output response equations it was proven and verified that the gain utilised at, or close to, the origin approaches 0.1875 as the number of output sampling points is increased. This low gain at the origin ensures that slight deviations from steady state do not result in excessive input action.

This regulatory type fuzzy controller was successfully applied to a first order system (with

dead time), a second order system and a third order system (with dead time). In each case the fuzzy controller was able to yield superior output and input response to that of an equivalently tuned linear PI controller. It was found that slightly longer rise times result from an increased number of output sampling points. It was also found that the best tuned controller with a high value of n has a slightly larger sum of squared errors (SSE) and a much reduced sum of squared change in output (SSΔU) than the best tuned controller with n=3 for a first order system. Therefore it is possible to obtain a much less severe input action for relatively the same value of SSE as n is increased for the first order system.

The regulatory type fuzzy controller was also successfully applied to the nonlinear fermentation of penicillin and lysine; both showing good disturbance rejection capabilities. The linear PI controller is designed around linearised systems and is unable to yield good control because the fermentation is a highly nonlinear system. The fuzzy controller is able to yield better control because of its nonlinear structure.

Fuzzy control was also applied in a supervisory capacity to the control of lysine fermentation. By investigating the results of a simulated batch fermentation it was possible to determine the appropriate switching time from batch to fed-batch fermentation. The fermentation was returned to the batch stage once the fermenter's final volume was attained and the fermentation was stopped after 35 hours. The fuzzy control rules were assigned to ensure that the specific growth rate remained at a value which favours lysine production during the fed-batch stage.

The optimisation of the input and output fuzzy sets' locations by using the NPSOL optimisation technique was unsuccessful because the cost ratio was found not to be a unimodal function of the optimisation variables. The cost ratio was found to be highly dependent on the value of some variables, displaying many local maxima.

Through an exhaustive search, using random values of the variables, a good fuzzy controller was found which yields both good input and output response as well as yielding a superior profit percent than that obtained by Modak and Lim (1987). The resulting input response is

found to have the same characteristic shape as that determined by Modak and Lim (1987).

It was found that the fuzzy controller is not very sensitive to the number of output sampling points used. However, if a high number of equidistant input and output fuzzy sets are used (above 325) then it is possible to attain a slightly higher profit percent than the original fuzzy controller with 5 sets.

Model uncertainty was examined by varying a coefficient, C, in the specific growth rate equation. It was found that the fuzzy controller, with 350 equidistant sets, is able to yield a superior profit percent over the results of Modak and Lim (1987) for a range of the specific growth rate's coefficient. The fuzzy controller is able to handle uncertainty in C better than the prescribed trajectory of Modak and Lim over a wide range of the coefficient because the trajectory of Modak and Lim is determined only for the ideal model.

The fuzzy controller was also found to have better disturbance rejection capabilities than the preset trajectory of Modak and Lim over the entire range of C investigated. The input responses obtained from the fuzzy controller also have the same characteristic shape as the optimal trajectory of Modak and Lim (1987). The fuzzy controller is able to deal better with the disturbances and still maintain a high profit percent because it is allowed to adjust its input response whereas the input trajectory of Modak and Lim is set *a priori*. Therefore it was possible to implement a fuzzy controller not only for the ideal fermentation of lysine, but also in the presence of modelling errors in the form of the specific growth rate and a disturbance in the feed concentration.

In summary, this dissertation concludes that fuzzy control can be applied in both a regulatory and supervisory capacity to fed-batch fermentations and indeed has shown improved control over standard control techniques for the fermentation of lysine and penicillin.

REFERENCES

Athalye A., D. Edwards, V.S. Manoranjan and A. de Sam Lazaro, "On Designing a Fuzzy Control System Using an Optimization Algorithm", *Fuzzy Sets and Systems*,**56**,pp.281-290,(1993).

Atkinson B. and F. Mavituna, *Biochemical Engineering and Biotechnology Handbook*, MacMillan Publishers LTD, Surrey,(1983).

Bastin G. and D. Dochain, "On-Line Estimation of Microbial Specific Growth Rates", *Automatica*,**22**,pp.705-709,(1986).

Chen C. and C. Hwang, "Optimal On-Off Control for Fed-Batch Fermentation Processes", *Industrial Engineering and Chemical Research*,**29**,No.9,pp1869-1875,(1990).

Cintrón I., "Drugs and Fine Chemicals", *Chemical Marketing Reporter*,p.16,19 April(1993).

Coello N., J.G. Pan and J.M. Lebeault, "Physiological aspects of L-Lysine production: Effect of Nutritional Limitations on a Producing Strain of *Corynebacterium glutamicum*", *Applied Microbiology and Biotechnology*,**38**,pp.259-262,(1992).

Davalo E. and P. Naïm, *Neural Networks*, MacMillan Education Ltd, London,(1991).

Di Massimo C., G.A. Montague, M.J. Willis, M.T. Tham and A.J. Morris, "Towards Improved Penicillin Fermentation Via Artificial Neural Networks", *Computers and Chemical Engineering*,**16**,No.4,pp.283-291,(1992).

Dohnal M., "Fuzzy Bioengineering Models", *Biotechnology and Bioengineering*,**27**,pp.1146-1151,(1985).

Dochain D. and G. Bastin, "Adaptive Identification and Control Algorithms for Nonlinear Bacterial Growth Systems", *Automatica*,**20**,pp.621-634,(1984).

Driankov D., H. Hellendoorn and M. Reinfrank, *An Introduction to Fuzzy Control*, Springer-

Verlag, Berlin,(1993).

Dubois D. and H. Prade, "Fuzzy Sets: A Survey of Engineering Applications", *Computers and Chemical Engineering*,**17**,Suppl(Escape-2),pp.s373-s380,(1993).

Kandel A. and G. Langholz, *Fuzzy Control Systems*, CRC Press, London,(1993).

Karim M.N. and S.L. Rivera, "Comparison of Feed-Forward and Recurrent Neural Networks for Bioprocess State Estimation", *Computers and Chemical Engineering*,**16**,Suppl(Escape-1),pp.s369-s376,(1992).

Kenyon C.P., "The History of the Development of the AECI Lysine Project and the Research and Development Strategy Involved", *Chemical Technology*,pp.32-35,(August 1993).

Kirk-Othmer, *Encyclopedia of Chemical Technology*, John Wiley & Sons, New York,**2**,3rd Edition, M.F. Mark, D.F. Othmer, C.G. Overberger and G.T. Seaborg Eds.,(1978).

Kirk-Othmer, *Encyclopedia of Chemical Technology*, John Wiley & Sons, New York,**2**,4th Edition, M. Howe-Grant Ed.,(1992).

Kiss R.D. and G. Stephanopoulos, "Metabolic Activity Control of the L-Lysine Fermentation by Restrained Growth Fed-Batch Strategies", *Biotechnology Progress*,**7**,pp.501-509,(1991).

Kiss R.D. and G. Stephanopoulos, "Culture Instability of Auxotrophic Amino Acid Producers", *Biotechnology and Bioengineering*,**40**,pp.75-85,(1992).

Klir G.J. and T.A. Folger, *Fuzzy Sets, Uncertainty, and Information*, Prentice-Hall, New Jersey,(1988).

Ko K.Y., B.C. McInnis and G.C. Goodwin, "Adaptive Control and Identification of the Dissolved Oxygen Process", *Automatica*,**18**,pp.727-730,(1982).

Konstantinov K. and T. Yoshida, "Physiological State Control of Fermentation Processes", *Biotechnology and Bioengineering*,**33**,pp.1145-1156,(1989).

Konstantinov K.B. and T. Yoshida, "Real-Time Qualitative Analysis of the Temporal Shapes of (Bio)process Variables", *AIChE Journal*,**38**,No.11,pp.1703-1715,(1992).

Latrille E., G. Corrieu and J. Thibault, "pH Prediction and Final Fermentation Time Determination in Lactic Acid Batch Fermentations", *Computers and Chemical Engineering*,**17**,Suppl(Escape-2),pp.s423-s428,(1993).

Lim H.C., Y.J. Tayeb, J.M. Modak and P. Bonte, "Computational Algorithms for Optimal Feed Rates for a Class of Fed-Batch Fermentation: Numerical Results for Penicillin and Cell Mass Production", *Biotechnology and Bioengineering*,**28**,pp.1408-1420,(1986).

Linko P. and Y. Zhu, "Neural Network Programming in Bioprocess Variable Estimation and State Prediction", *Journal of Biotechnology*,**21**,pp.253-270,(1991).

Mahuli S.K., R.R. Rhinehart and J.B. Riggs, "pH Control Using a Statistical Technique for Continuous On-Line Model Adaptation", *Computers and Chemical Engineering*,**17**,No.4,pp.309-317,(1993).

Menawat N., R. Mutharasan, D.R. Coughanowr, "Singular Optimal Control Strategy for a Fed-Batch Bioreactor: Numerical Approach", *AIChE Journal*,**33**,No.5,pp.776-783,(1987).

Modak J.M., "Choice of Control Variable for Optimization of Fed-Batch Fermentation", *The Chemical Engineering and Biochemical Engineering Journal*,**52**,pp.B59-B69,(1993).

Modak J.M. and H.C. Lim, "Feedback Optimization of Fed-Batch Fermentation", *Biotechnology and Bioengineering*,**30**,pp.528-540,(1987).

Modak J.M. and H.C. Lim, "Optimal Mode of Operation of Bioreactor for Fermentation Processes", *Chemical Engineering Science*,**47**,No.15/16,pp.3869-3884,(1992).

Modak J.M., H.C. Lim and Y.J. Tayeb, "General Characteristics of Optimal Feed Rate Profiles for Various Fed-Batch Fermentation Processes", *Biotechnology and Bioengineering*,**28**,pp.1396-1407,(1986).

Montague G.A., A.J. Morris, A.R. Wright, M. Aynsley and A. Ward, "Modelling and Adaptive Control of Fed-Batch Penicillin Fermentation", *The Canadian Journal of Chemical Engineering*,**64**,pp.567-580,(1986a).

Montague G.A., A.J. Morris, A.R. Wright, M. Aynsley and A.C. Ward, "Online Estimation and Adaptive Control of Penicillin Fermentation", *IEE Proceedings*,**133**,No.5,pp.240-246,(1986b).

Morris A.J., G.A. Montague, M.T. Tham, M. Aynsley, C. Di Massimo and P. Lant, "Towards Improved Process Supervision - Algorithms and Knowledge Based Systems", *Chemical Process Control - CPIV*, Y. Arkun and W.H. Ray,eds,CACHE,pp.585-614,(1991).

Nagai S., "Mass and Energy Balances for Microbial Growth Kinetics", *Advances in Biochemical Engineering*,**11**,pp.53-83,(1979).

Ohno H., E. Nakanishi and T. Takamatsu, "Optimal Control of a Semibatch Fermentation", *Biotechnology and Bioengineering*,**18**,pp.847-867,(1976).

Ohno H., E. Nakanishi and T. Takamatsu, "Optimum Operating Mode for a Class of Fermentation", *Biotechnology and Bioengineering*,**20**,pp.625-636,(1978).

Pedrycz W., *Fuzzy Control and Fuzzy Systems*, John Wiley & Sons Inc, New York,(1993).

Postlethwaite B.E., "A Fuzzy State Estimator for Fed-Batch Fermentation", *Chemical Engineering Research and Design*,**67**,pp.267-272,(1989).

Qin S.J., "Auto-Tuned Fuzzy Logic Control", *Proceedings of the American Control Conference*,pp.2465-2469,(June,1994).

Ray W.H., *Advanced Process Control*, McGraw-Hill, New York,(1981).

San K. and G. Stephanopoulos, "Optimization of Fed-Batch Fermentation: A Case of Singular Optimal Control with State Constraints", *Biotechnology and Bioengineering*,**34**,pp.72-78,(1989).

Seborg D.E., T.F. Edgar and D.A. Mellichamp, *Process Dynamics and Control*, John Wiley & Sons, New York,(1989).

Shon M., "Drugs and Fine Chemicals", *Chemical Marketing Reporter*,p.18,6 April(1992).

Simutis R., I. Havlik and A. Lübbert, "A Fuzzy-Supported Extended Kalman Filter: A New Approach to State Estimation and Prediction Exemplified by Alcohol Formation in Beer Brewing", *Journal of Biotechnology*,**24**,pp.211-234,(1992).

Snell F.D. and L.S. Ettre, Eds., *Encyclopedia of Industrial Chemical Analysis*, John Wiley & Sons, New York,**16**,(1972).

Stephanopoulos G. and K. San, "Studies on On-Line Bioreactor Identification", *Biotechnology and Bioengineering*,**26**,pp.1176-1218,(1984).

Tong R.M., M.B. Beck and A. Latten, "Fuzzy Control of the Activated Sludge Wastewater Treatment Process", *Automatica*,**16**,pp.695-701,(1980).

Whitnell G.P., V.J. Davidson, R.B. Brown and G.L. Hayward, "Fuzzy Predictor for Fermentation Time in a Commercial Brewery", *Computers and Chemical Engineering*,**17**,No.10,pp.1025-1029,(1993).

Ying, H., W. Siler and J.J. Buckley, "Fuzzy Control Theory: A Nonlinear Case", *Automatica*,**26**,pp.513-520,(1990).

APPENDICES

Appendix A

Derivation of Output Response Equations for Three Output Sampling Points

The grade of membership of any value in either of the input or output fuzzy sets can be calculated from the following equations (due to the fuzzy sets all having regular triangular shapes in Figure 3.16, Chapter 3):

$$E.p = \frac{L + GEe(t)}{2L}$$

$$E.n = \frac{L - GEe(t)}{2L}$$

$$R.p = \frac{L + GRr(t)}{2L}$$

$$R.n = \frac{L - GRr(t)}{2L}$$

where:	E.p, E.n	stand for the grade of membership of the fuzzy sets *error positive* and *error negative*.
	R.p, R.n	stand for the grade of membership of the fuzzy sets *rate positive* and *rate negative*.
	GRr(t), GEe(t)	represent the scaled rate and error.

The controller's input range (within -L and L) can be divided into eight regions as shown in Figure A.1. The division of the input range helps in the formulation of representative output controller response equations for the input range. The fuzzy controller rules are:

Rule 1.	If **error** is *positive* and **rate** is *positive* then **output** is *positive*
Rule 2.	If **error** is *positive* and **rate** is *negative* then **output** is *zero*
Rule 3.	If **error** is *negative* and **rate** is *positive* then **output** is *zero*
Rule 4.	If **error** is *negative* and **rate** is *negative* then **output** is *negative*

In Regions R1,R2,R5 and R6 the absolute value of GEe(t) is always greater than or equal to the ablsolute value of GRr(t). The opposite is true of Regions R3,R4,R7 and R8. Therefore the **intersection** (min) operation (analogous to **step 1** in Section 3.5.2) between the two inputs can be calculated for each of the four rules as summarised in Table A.1. If, for instance, we are dealing with Region R2 then the value of GEe(t) will always be greater than or equal to GRr(t). Therefore for the first rule the value of R.p (grade of membership of GRr(t) in the *rate is positive* fuzzy set) will always be greater than or equal to the value of E.p (grade of membership of GEe(t) in the *error is positive* fuzzy set).

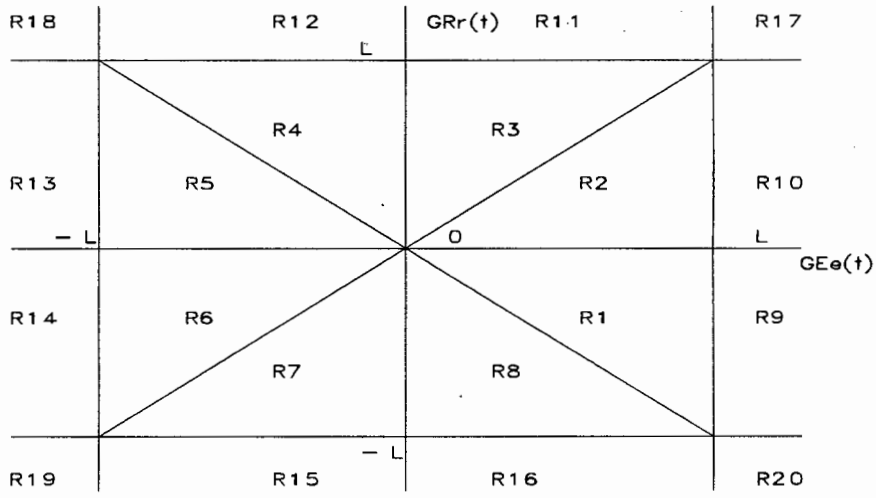

Figure A.1. Division of input range.

Table A.1. Evaluation of four fuzzy control rules in each region.

Region	Intersection of both inputs (min)			
	Rule 1	Rule 2	Rule 3	Rule 4
R1	**R.p**	**R.n**	**E.n**	**E.n**
R2	R.p	R.n	E.n	E.n
R3	E.p	R.n	E.n	R.n
R4	E.p	R.n	E.n	R.n
R5	E.p	E.p	R.p	R.n
R6	E.p	E.p	R.p	R.n
R7	R.p	E.p	R.p	E.n
R8	R.p	E.p	R.p	E.n

Consider Region R1 and assume that the values of GRr(t) and GEe(t) lie in this region. The grade of membership of Rule 1 is only affected by the value of the rate (from Table A.1). GRr(t) can only be in the range [-L , 0] (because we are dealing with Region R1) and therefore Rule 1's grade of membership (R.p) can only lie in the range [0 , 0.5L]. The grade of membership of Rule 2 is also only affected by the value of the rate. Once again GRr(t) lies in the range [-L , 0] and therefore Rule 2's grade of membership (R.n) lies in the range [0.5L , L]. Similarly the grade of membership for Rules 3 and 4 (E.n) both lie in the range [0 , 0.5L].

The **intersection** (min) operation must now be determined between each individual rule (analogous to the clipping of the output fuzzy sets summarised by **step 2** in Section 3.5.2) and each of the three output sampling points. The results are summarised in Table A.2.

Table A.2. Evaluation of intersection between the rules and the sampling points for R1.

Rules	Grades of membership at each sampling point		
	-L	0	L
Rule 1	0	0	R.p
Rule 2	0	R.n	0
Rule 3	0	E.n	0
Rule 4	E.n	0	0
Fuzzy clipping (Qin)	E.n	R.n	R.p
Fuzzy clipping (Ying *et al*)	E.n	R.n + E.n	R.p

As an example, the first rule stipulates that the **output** should be *positive*. The value of -L in the fuzzy set *output positive* is 0 and therefore the minimum between 0 and the value obtained from the R.p equation will always be 0 (ie the output fuzzy set is clipped at a grade of membership of 0) (noting that the grades of membership can only assume values between 0 and 1). Similarly the fourth rule stipulates that the **output** should be *negative*. The value of -L in the fuzzy set **output negative** is 1 and therefore the minimum between 1 and the value obtained from the E.n equation will always be E.n.

The fuzzy response of the controller is determined by calculating the **union** (max) of the four rules at each of the three output sampling points (analogous to **step 3** in Section 3.5.2) as summarised in Table A.2. When looking at the 0 output sampling point, the grade of membership of Rule 2 lies in the range [0.5L , L] while the grade of membership of Rule 3 lies in the range [0 , 0.5L]. Therefore the second rule (R.n) will yield the higher grade of membership in Region R1.

Table A.2 also summarises the results if **Łukasiewicz OR** is used between rules 2 and 3 as utilised by Ying *et al* (1990). Ying *et al*(1990) uses **Łukasiewicz OR** between rules 2 and

3 while **Zadeh OR** is used between the rest of the rules.

The weighted mean of the fuzzy outputs (using Qin's approach) at each sampling point is determined, and then scaled, to yield the controller's output:

$$Output = \frac{E.n \times (-L) + R.n \times 0 + R.p \times (+L)}{E.n + R.n + R.p} GU$$

which yields the following equation:

$$Output = \frac{LGU}{3L - GEe(t)}\left[GRr(t) + GEe(t)\right]$$

After evaluating all the regions it is shown that if the input to the controller is such that $|GEe(t)| \geq |GRr(t)|$, then the output is given by:

$$Output = \frac{LGU}{3L - |GEe(t)|}\left[GRr(t) + GEe(t)\right]$$

and if the input to the controller is such that $|GEe(t)| \leq |GRr(t)|$, then the output is given by:

$$Output = \frac{LGU}{3L - |GRr(t)|}\left[GRr(t) + GEe(t)\right]$$

Therefore the output from the controller can be summarised by the following equation:

$$Output = \frac{LGU}{3L - \max\left[|GRr|, |GEe|\right]}\left[GRr(t) + GEe(t)\right]$$

This is effectively a nonlinear PI controller with a nonlinear gain:

$$K_{fuzzy} = \frac{LGUGR}{3L - \max\left[|GRr|, |GEe|\right]}$$

and an integral constant defined by:

$$\tau_{fuzzy} = \frac{GR}{GE}$$

The output response equations for Regions R9 to R20 can be derived similarly and are summarised in Table A.3.

Table A.3. Output response equations for error and rate outside the [-L,L] region.

Region	
R9 and R10	$\frac{GU(GRr(t) + L)}{2}$
R11 and R12	$\frac{GU(GEe(t) + L)}{2}$
R13 and R14	$\frac{GU(GRr(t) - L)}{2}$
R15 and R16	$\frac{GU(GEe(t) - L)}{2}$
R17	$L.GU$
R18 and R20	0
R19	$-L.GU$

Appendix B

Derivation of Output Response Equations for Five Output Sampling Points

The equations for the output response of the fuzzy controller were derived using the following five sampling points: (-L , -0.5L , 0 , 0.5L , L). It was decided to maintain the three original sampling points and utilise another two sampling points so as to keep the sampling symmetrical. It was found that the division of the input range, as (used for three sampling points) shown in Figure A.1, Appendix A, was adequate in order to determine the equations (the choice of how to subdivide the input range is discussed in Appendix C). The results of Table A.1, Appendix A, still apply however Table B.1 now shows the revised form (for five sampling points instead of three) of Table A.2.

Table B.1. Evaluation of intersection between the rules and the sampling points for R1 (five sampling points).

Rules	Grades of membership at each sampling point				
	-L	-0.5L	0	0.5L	L
Rule 1	0	0	0	R.p	R.p
Rule 2	0	0.5	R.n	0.5	0
Rule 3	0	E.n	E.n	E.n	0
Rule 4	E.n	E.n	0	0	0
Fuzzy clipping (Qin)	E.n	0.5	R.n	0.5	R.p
Fuzzy clipping (Ying *et al*)	E.n	0.5 + E.n	R.n + E.n	0.5 + E.n	R.p

Table B.2 summarises the controller gains and integral constants obtained.

Table B.2. Nonlinear gain and integral constant for five sampling points.

Parameter	Error and rate of same sign		Opposite sign
	GRr > GEe	GRr < GEe	
K_{fuzzy}	$\frac{LGUGR}{5L - \lvert GRr\rvert - \lvert GEe\rvert}$	$\frac{\frac{3}{2}LGUGR}{5L - \lvert GRr\rvert - \lvert GEe\rvert}$	$\frac{LGUGR}{5L - \max[\lvert GRr\rvert, \lvert GEe\rvert]}$
τ_{fuzzy}	$\frac{2}{3}\frac{GR}{GE}$	$\frac{3}{2}\frac{GR}{GE}$	$\frac{GR}{GE}$

Appendix C

Choice of Division for Input Range

The extent to which the input range, as seen in Figure C.1, needs to be subdivided depends on the number of output sampling points utilised (n). The number of sub-divisions required increases with an increase in the number of output sampling points.

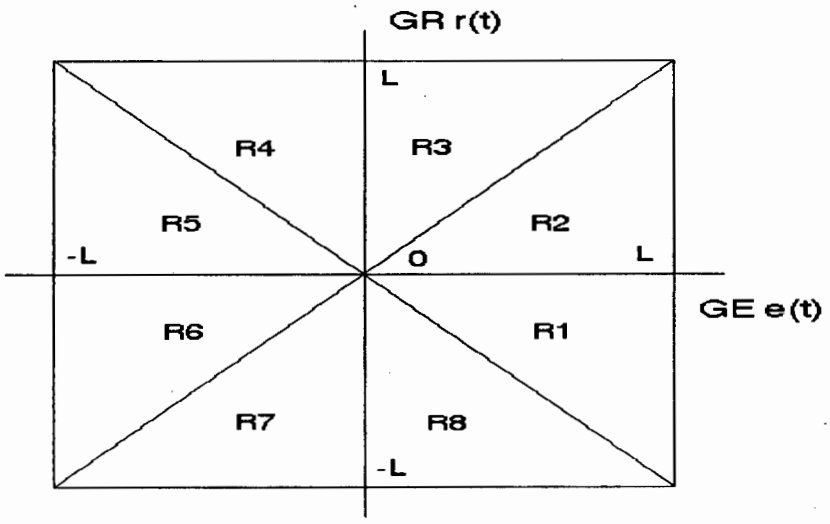

Figure C.1. Division of input range of three sampling points.

As an example consider the division of the input range as shown in Figure C.1 and consider the case where nine sampling points are used {-L, -0.75L, -0.5L, -0.25L, 0, 0.25L, 0.5L, 0.75L, L}. The first step is to calculate the **intersection** (min) between the two fuzzy controller inputs (**error** and **rate**) for each of the four rules in each major region. This is analogous to **step 1** in Section 3.5.2 and the results are summarised in Table A.1 in Appendix A. It should be stressed that the results summarised in Table A.1 are independent of the number of output sampling points used.

Step 2 summarised in Section 3.5.2 then stipulates that the output fuzzy sets need to be clipped according to the specifications of each fuzzy rule. In order to do this one has to determine the **intersection** (min) between the 'weakest link' (determined in **step 1**) and the grade of membership of the relevant output sampling points in the fuzzy set of the relevant rule. Consider the sampling point -0.75L. This point has a grade of membership of 0.75 in the *negative output* fuzzy set, 0.25 in the *zero output* fuzzy set and 0 in the *positive output* fuzzy set.

If one is attempting to derive the response for Region R2 (Figure C.1) you would have to determine the **intersection** (min) between the value of R.p (from Rule 1 in Table A.1, Appendix A) and the grade of membership of all sampling points affecting the fuzzy set *output is negative*. The problem arises when looking at the -0.75L sampling point. The

minimum has to be established between 0.75 (grade of membership of -0.75L in the fuzzy set *output is negative*) and the value of R.p, which can assume a value anywhere between 0.5 and 1 in Region R2. Therefore it becomes necessary to constrain the range of R.p by further sub-division of the input range so that the minimum operation can be performed. If 8 divisions are made on both the GRr and GEe axes then the values of R.p, R.n, E.p and E.n in each region will always be evaluated over a range of ¼ i.e. will lie between ¼ and ½ or between ¾ and 1 etc. This will ensure that the minimum operation can be performed.

Likewise, to guarantee that the output response equations can be determined, it is necessary that (n-1) divisions be made on the GRr and GEe axes for n output sampling points. Therefore the number of subdivisions required is summarised by the following equation:

$$subdivisions = (n - 1)^2 + 2(n - 1) = n^2 - 1 \qquad (1)$$

This equation is a sufficient but not a necessary condition as it was found that only 8 subdivisions (the same number required for 3 sampling points) were necessary for five sampling points and 24 for nine sampling points compared to the 24 and 80 given by equation 1. It was therefore decided to derive output response equations for the nine sampling points instead of for seven sampling points which requires 48 equations to be determined.

Appendix D

Derivation of Output Response Equations for Nine Output Sampling Points

The input range division used for nine sampling points is shown in Figure D.1. It should be pointed out that the number of rules and membership functions remains unchanged irrespective of the number of output sampling points chosen.

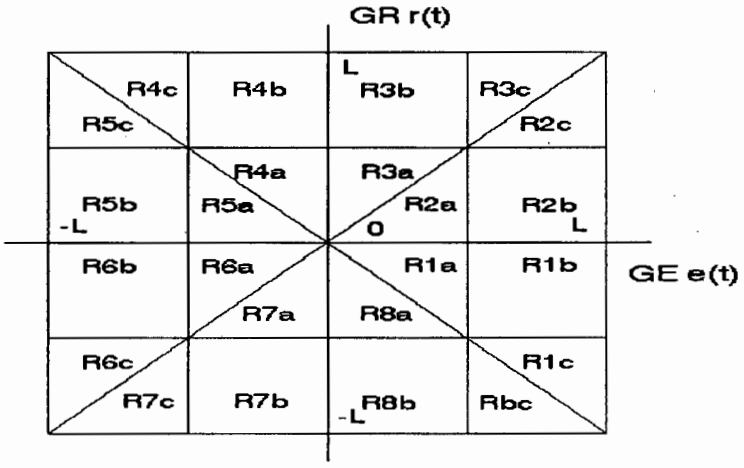

Figure D.1. Regions used in derivation of generalised equations.

It should be noted that each major input region, as used in Figure A.1 in Appendix A, has now been divided into three subregions i.e. Region R1 is now subdivided into Subregions R1a, R1b and R1c. As with the case of the three sampling points, all these subregions were dealt with separately to obtain the corresponding output response equation. Table A.1's (Appendix A) results are still applicable.

The output sampling points are situated at the following positions:

$$value = -L, -\frac{3L}{4}, -\frac{L}{2}, -\frac{L}{4}, 0, \frac{L}{4}, \frac{L}{2}, \frac{3L}{4}, L$$

Derivation of Output Response Equations for Subregion R2b.

Table D.1 shows the revised form (for nine sampling points instead of three) of Table A.2 in Appendix A.

Table D.1. Evaluation of intersection between the rules and the sampling points for R2b (nine sampling points).

Rules	Grades of membership at each sampling point								
	-L	-0.75L	-0.5L	-0.25L	0	0.25L	0.5L	0.75L	L
Rule 1	0	0	0	0	0	0.25	0.5	R.p	R.p
Rule 2	0	0.25	R.n	R.n	R.n	R.n	R.n	0.25	0
Rule 3	0	E.n	E.n	E.n	E.n	E.n	E.n	E.n	0
Rule 4	E.n	E.n	E.n	E.n	0	0	0	0	0
Fuzzy clipping (Qin)	E.n	0.25	R.n	R.n	R.n	R.n	0.5	R.p	R.p
Fuzzy clipping (Ying *et al*)	E.n	E.n + 0.25	R.n + E.n	R.n + E.n	R.n + E.n	R.n + E.n	0.5	R.p	R.p

Table D.2 summarises the controller gains and integral constants obtained for nine sampling points.

Table D.2. Nonlinear controller gain and integral constant for both nine and five sampling points.

Case 1	K_{fuzzy}	τ_{fuzzy}	C
a) $\lvert GRr\rvert > \lvert GEe\rvert$	$\frac{\frac{7}{4}L.GU.GR}{9L - 2\lvert GRr\rvert - \gamma 2GEe}$	$\frac{7}{9}\frac{GR}{GE}$	
$\lvert GRr\rvert < \lvert GEe\rvert$	$\frac{\frac{9}{4}L.GU.GR}{9L - \gamma 2GRr - 2\lvert GEe\rvert}$	$\frac{9}{7}\frac{GR}{GE}$	
b)	$\frac{\frac{7}{4}L.GU.GR}{9L - 2\lvert \max\rvert - \delta\gamma \min}$	$\frac{GR}{GE}$	

Table D.2. (continued)

Case 2	K_{fuzzy}	τ_{fuzzy}	C
a) $\lvert GRr\rvert > \lvert GEe\rvert$	$\frac{L.GU.GR}{8.5L - \lvert GRr\rvert - \gamma 2GEe}$	$\frac{4}{9}\frac{GR}{GE}$	$\frac{3}{8}\frac{\gamma}{GR}$
$\lvert GRr\rvert < \lvert GEe\rvert$	$\frac{\frac{9}{4}L.GU.GR}{8.5L - \gamma 2GRr - \lvert GEe\rvert}$	$\frac{9}{4}\frac{GR}{GE}$	$\frac{1}{6}\frac{\gamma}{GR}$
b) $\lvert GRr\rvert > \lvert GEe\rvert$	$\frac{L.GU.GR}{8.5L - \lvert GRr\rvert - \delta\gamma GEe}$	$\frac{4}{7}\frac{GR}{GE}$	$\frac{3}{8}\frac{\delta\gamma}{GR}$
$\lvert GRr\rvert < \lvert GEe\rvert$	$\frac{\frac{7}{4}L.GU.GR}{8.5L - \delta\gamma GRr - \lvert GEe\rvert}$	$\frac{7}{4}\frac{GR}{GE}$	$\frac{3}{14}\frac{\delta\gamma}{GR}$

Table D.2. (continued)

Case 3	K_{fuzzy}	τ_{fuzzy}	C
a) $\lvert GRr\rvert > \lvert GEe\rvert$	$\frac{LGUGR}{9L - \lvert GRr\rvert - \gamma 3GEe}$	$\frac{2}{5}\frac{GR}{GE}$	$\frac{1}{4}\frac{\gamma}{GR}$
$\lvert GRr\rvert < \lvert GEe\rvert$	$\frac{\frac{5}{2}L.GU.GR}{9L - \gamma 3GRr - \lvert GEe\rvert}$	$\frac{5}{2}\frac{GR}{GE}$	$\frac{1}{10}\frac{\gamma}{GR}$
b)	$\frac{LGUGR}{9L - \lvert max\rvert}$	$\frac{GR}{GE}$	

where: the output is defined as follows:

$$Output = K_{fuzzy}\left[r(t) + \frac{1}{\tau_{fuzzy}} e(t) + CL\right]$$

and:

Case 1 is when both |GRr(t)| and |GEe(t)| lie within the range [0 , 0.5L]

Case 2 is when only one of |GRr(t)| or |GEe(t)| lies within the range [0 , 0.5L] and the other in the range [0.5L , L]

Case 3 is when both |GRr(t)| and |GEe(t)| lie within the range [0.5L , L]

Part a always indicates that GRr(t) and GEe(t) are of the same sign

Part b always indicates that GRr(t) and GEe(t) are of opposite signs

γ is -1 if rate is negative and +1 if rate is positive.

δ is -1 if $|GRr| < |GEe|$ and +1 if $|GRr| > |GEe|$.

max is given by:

$$GRr(t) \quad if \ |GRr(t)| \geq |GEe(t)|$$
$$GEe(t) \quad if \ |GRr(t)| \leq |GEe(t)|$$

min is given by:

$$GRr(t) \quad if \ |GRr(t)| \leq |GEe(t)|$$
$$GEe(t) \quad if \ |GRr(t)| \geq |GEe(t)|$$

Appendix E

Derivation of More Generalised Output Response Equations

Figure E.1 shows the regions for which generalised equations were determined.

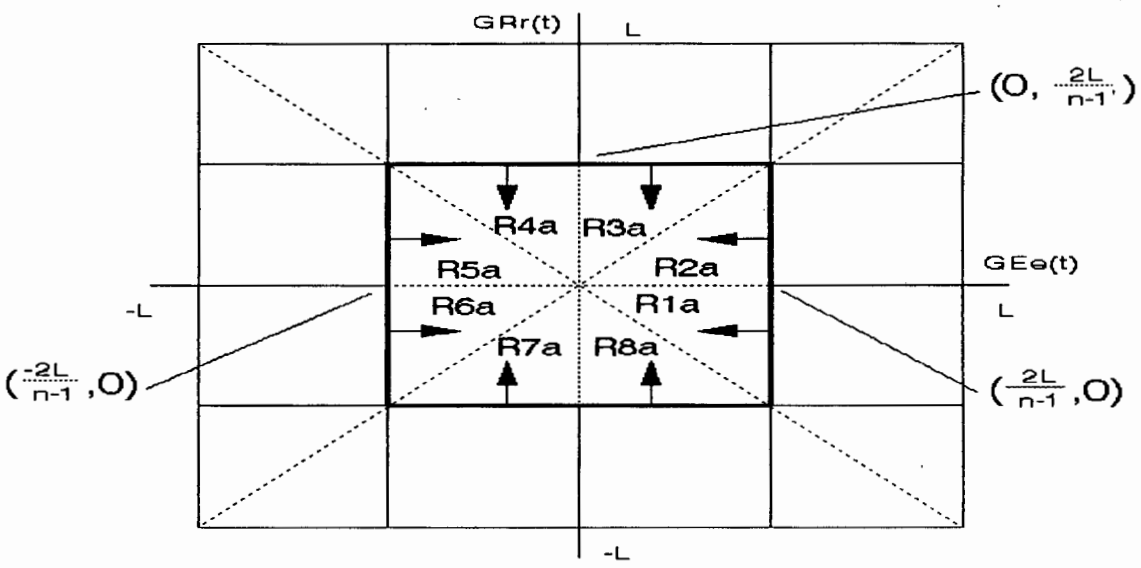

Figure E.1. Regions used in derivation of generalised equations.

It was observed that the fuzzy resultant (defined in Step 3, Section 3.5.2, Chapter 3), Z, prior to determining the weighted mean is the same for each respective region (R1a, R2a,... or R8a) regardless of n as long as the regions are bounded by $|2L/(n-1)|$ on both the GRr and GEe axes as shown in Figure E.1 i.e. region R2a will have a given fuzzy resultant which is independent of n while R1a will have a different fuzzy resultant, but one which is also independent of n. It should be noted that these regions become smaller and tend towards the origin as n is increased.

The output sampling points are situated at the following positions for 5 or more sampling points (n satisfying equation 3 in Section 3.8.3, Chapter 3):

$$value = -L + \frac{2L}{n-1}i \qquad i = 0, \ldots, \frac{n-1}{4}, \ldots, \frac{n-1}{2}, \ldots, \frac{3n-3}{4}, \ldots, n-1$$

Table E.1 shows how the fuzzy resultant, Z is determined for Region R2a.

Table E.2. Evaluation of intersection between the rules and the sampling points for R2 (five or more output sampling points).

Rules	Grades of membership at each sampling point								
	-L	(-L,-½L)	-½L	(-½L,0)	0	(0,½L)	½L	(½L,L)	L
Rule 1	0	0	0	0	0	$\mu_{\Delta UP}(y_r)$	0.5	min[μ,R.p]	R.p
Rule 2	0	min[μ,R.n]	R.n	R.n	R.n	R.n	R.n	min[μ,R.n]	0
Rule 3	0	min[μ,E.n]	E.n	E.n	E.n	E.n	E.n	min[μ,E.n]	0
Rule 4	E.n	E.n	E.n	min[μ,E.n]	0	0	0	0	0
Clipping (Qin) i.e. μ_z	E.n	E.n	R.n	R.n	R.n	R.n	0.5	R.p	R.p

Therefore the resultant grade of membership of the combined rules, μ_Z, at each output sampling point used is given by:

$$\frac{L - GEe}{2L} \quad (0 \le i < \frac{n-1}{4})$$

$$\frac{L - GRr}{2L} \quad (\frac{n-1}{4} \le i < \frac{3n-3}{4})$$

$$\frac{1}{2} \quad (i = \frac{3n-3}{4}) \qquad \frac{L + GRr}{2L} \quad (i > \frac{3n-3}{4})$$

The numerator of NR2 is therefore determined:

$$\frac{L - GEe}{2L} \sum_{i=0}^{\frac{n-5}{4}} (-L + \frac{2L}{n-1} i) + \frac{L - GRr}{2L} \sum_{i=\frac{n-1}{4}}^{\frac{3n-7}{4}} (-L + \frac{2L}{n-1} i)$$

$$+ \frac{1}{2}(\frac{1}{2}L) + \frac{L + GRr}{2L} \sum_{i=\frac{3n+1}{4}}^{n-1} (-L + \frac{2L}{n-1} i)$$

and the denominator is determined as follows:

$$\sum_{i=0}^{\frac{n-5}{4}} \frac{L - GEe}{2L} + \sum_{i=\frac{n-1}{4}}^{\frac{3n-7}{4}} \frac{L - GRr}{2L} + \frac{1}{2} + \sum_{i=\frac{3n+1}{4}}^{n-1} \frac{l + GRr}{2L}$$

Therefore the output for region R2a is given by:

$$\frac{L.GU\left[\frac{3}{16}(n+3)GRr + \frac{3n+1}{16}GEe\right]}{nL - \frac{n-1}{4}GRr - \frac{n-1}{4}GEe}$$

Table E.3 shows the values of the gain and integral constants for the generalised equations. Substitution of n=5 and n=9 correctly yields the equations for five and nine output sampling points previousy derived.

Table E.3. Nonlinear controller gain and integral constant for generalised equations.

Region a	K_{fuzzy}	τ_{fuzzy}
a) $\lvert GRr\rvert > \lvert GEe\rvert$	$\dfrac{\frac{3n+1}{16} L\, GU\, GR}{nL - \frac{n-1}{4}\lvert GRr\rvert - \gamma\frac{n-1}{4} GEe}$	$\dfrac{3n+1}{3(n+3)}\dfrac{GR}{GE}$
$\lvert GRr\rvert < \lvert GEe\rvert$	$\dfrac{\frac{3(n+3)}{16} L\, GU\, GR}{nL - \gamma\frac{n-1}{4} GRr - \frac{n-1}{4}\lvert GEe\rvert}$	$\dfrac{3(n+3)}{3n+1}\dfrac{GR}{GE}$
b)	$\dfrac{\frac{3n+1}{16} L\, GU\, GR}{nL - \frac{n-1}{4}\lvert \max\rvert - \delta\gamma\frac{n-5}{4}\min}$	$\dfrac{GR}{GE}$

where the output is defined as follows:

$$Output = K_{fuzzy}\left[r(t) + \frac{1}{\tau_{fuzzy}} e(t)\right]$$

and:

Part a	indicates that GRr(t) and GEe(t) are of the same sign
Part b	indicates that GRr(t) and GEe(t) are of opposite signs
n	is the number of output sampling points (5 or more)
γ	is -1 if rate is negative and +1 if rate is positive.
δ	is -1 if $\|GRr\| < \|GEe\|$ and +1 if $\|GRr\| > \|GEe\|$.
max	is given by:

$$GRr(t) \quad if \ |GRr(t)| \geq |GEe(t)|$$
$$GEe(t) \quad if \ |GRr(t)| \leq |GEe(t)|$$

min is given by:

$$GRr(t) \quad if \ |GRr(t)| \leq |GEe(t)|$$
$$GEe(t) \quad if \ |GRr(t)| \geq |GEe(t)|$$

Appendix F

Proof of Positive Gain and Integral Constant

It should be pointed out that the number of rules and membership functions remains unchanged irrespective of the number of output sampling points chosen.

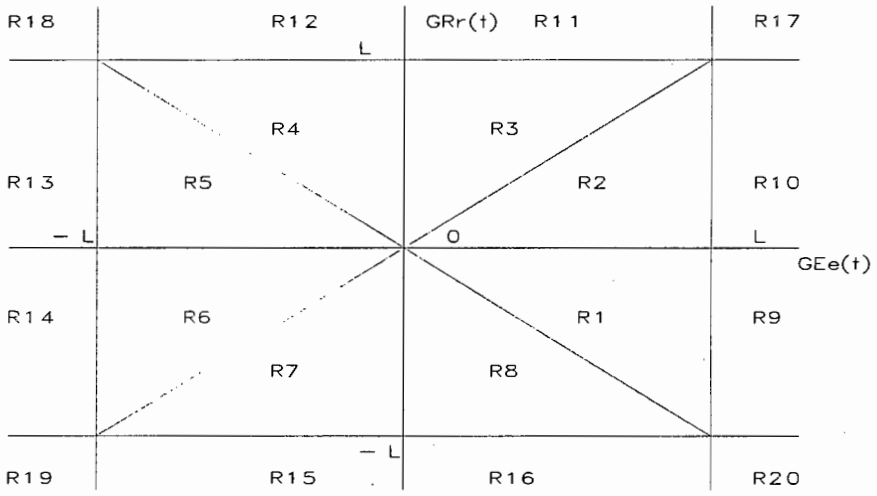

Figure F.1. Division of the input range into eight regions.

Each region shown in within the [-L,L] range in Figure F.1 is dealt with separately to obtain the possible values that the output may assume prior to defuzzification. Table A.1's (Appendix A) results are still applicable.

Derivation for Region R1.

Table F.1 shows the revised form (for five or more output sampling points with n satisfying equation 3 in Chapter 3) of Table A.2 in Appendix A.

Table F.1. Evaluation of intersection between the rules and the sampling points for R1 (five or more output sampling points).

Rules	Grades of membership at each sampling point								
	-L	(-L,-½L)	-½L	(-½L,0)	0	(0,½L)	½L	(½L,L)	L
Rule 1	0	0	0	0	0	min[μ,R.p]	R.p	R.p	R.p
Rule 2	0	$\mu_{\Delta UZ}(y_r)$	0.5	min[μ,R.n]	R.n	min[μ,R.n]	0.5	$\mu_{\Delta UZ}(y_r)$	0
Rule 3	0	min[μ,E.n]	E.n	E.n	E.n	E.n	E.n	min[μ,E.n]	0
Rule 4	E.n	E.n	E.n	min[μ,E.n]	0	0	0	0	0
Clipping (Qin) i.e. μ_Z	E.n	$\mu_{\Delta UZ}(y_r)$ or E.n	0.5	$\mu_{\Delta UZ}(y_r)$, R.n or E.n	R.n	$\mu_{\Delta UZ}(y_r)$ or R.n	0.5	$\mu_{\Delta UZ}(y_r)$ or R.p	R.p

where: $\mu_{\Delta UP}(y_r)$, $\mu_{\Delta UZ}(y_r)$ and $\mu_{\Delta UN}(y_r)$ are the grades of membership of an output sampling point, y_r, in the *change in output is positive*, *change in output is zero* and *change in output is negative* fuzzy sets respectively

μ is the grade of membership of an output sampling point, y_r, in the output fuzzy set corresponding to the rule it appears under

(a,b) denotes the output sampling points lying between the values of a and b and

$$R.n = \frac{L - GRr(t)}{2L}$$

$$R.p = \frac{L + GRr(t)}{2L}$$

$$E.n = \frac{L - GEe(t)}{2L}$$

$$E.p = \frac{L + GEe(t)}{2L}$$

The numerator of the final output response equation is determined by summing:

$$\sum_{y_r = -L}^{L} y_r \cdot \mu_Z(y_r)$$

Therefore the grades of resultant grades of membership, $\mu_Z(y_r)$ lying to the left of the 0 output sampling point will be multiplied by negative values while those to the right of the 0 output sampling point will be multiplied by positive values. It can therefore be seen that the coefficients of the rate and error will always assume positive values.

Similarly for Regions R2 - R8.

From Table F.2 to Table F.8 it can similarly be seen that the output response equations will always have positive coefficients for both the error and rate ensuring both a positive gain and integral constant.

Table F.2. Evaluation of intersection between the rules and the sampling points for R2 (five or more output sampling points).

Rules	Grades of membership at each sampling point								
	-L	(-L,-½L)	-½L	(-½L,0)	0	(0,½L)	½L	(½L,L)	L
Rule 1	0	0	0	0	0	$\mu_{\Delta UP}(y_r)$	0.5	min[μ,R.p]	R.p
Rule 2	0	min[μ,R.n]	R.n	R.n	R.n	R.n	R.n	min[μ,R.n]	0
Rule 3	0	min[μ,E.n]	E.n	E.n	E.n	E.n	E.n	min[μ,E.n]	0
Rule 4	E.n	E.n	E.n	min[μ,E.n]	0	0	0	0	0
Clipping (Qin) i.e. μ_Z	E.n	$\mu_{\Delta UZ}(y_r)$, E.n or R.n	R.n	$\mu_{\Delta UN}(y_r)$ or R.n	R.n	$\mu_{\Delta UP}(y_r)$ or R.n	0.5	$\mu_{\Delta UP}(y_r)$ or R.p	R.p

Table F.3. Evaluation of intersection between the rules and the sampling points for R3 (five or more output sampling points).

Rules	Grades of membership at each sampling point								
	-L	(-L,-½L)	-½L	(-½L,0)	0	(0,½L)	½L	(½L,L)	L
Rule 1	0	0	0	0	0	$\mu_{\Delta UP}(y_r)$	0.5	min[μ,E.p]	E.p
Rule 2	0	min[μ,R.n]	R.n	R.n	R.n	R.n	R.n	min[μ,R.n]	0
Rule 3	0	min[μ,E.n]	E.n	E.n	E.n	E.n	E.n	min[μ,E.n]	0
Rule 4	R.n	R.n	R.n	min[μ,R.n]	0	0	0	0	0
Clipping (Qin) i.e. μ_Z	R.n	$\mu_{\Delta UZ}(y_r)$, E.n or R.n	E.n	$\mu_{\Delta UN}(y_r)$ or E.n	E.n	$\mu_{\Delta UP}(y_r)$ or E.n	0.5	$\mu_{\Delta UP}(y_r)$ or E.p	E.p

Table F.4. Evaluation of intersection between the rules and the sampling points for R4 (five or more output sampling points).

Rules	Grades of membership at each sampling point								
	-L	(-L,-½L)	-½L	(-½L,0)	0	(0,½L)	½L	(½L,L)	L
Rule 1	0	0	0	0	0	min[μ,E.p]	E.p	E.p	E.p
Rule 2	0	min[μ,R.n]	R.n	R.n	R.n	R.n	R.n	min[μ,R.n]	0
Rule 3	0	$\mu_{\Delta UZ}(y_r)$	0.5	min[μ,E.n]	E.n	min[μ,E.n]	0.5	$\mu_{\Delta UZ}(y_r)$	0
Rule 4	R.n	R.n	R.n	min[μ,R.n]	0	0	0	0	0
Clipping (Qin) i.e. μ_Z	R.n	$\mu_{\Delta UZ}(y_r)$ or R.n	0.5	$\mu_{\Delta UZ}(y_r)$ or E.n	E.n	$\mu_{\Delta UZ}(y_r)$ or E.n	0.5	$\mu_{\Delta UZ}(y_r)$ or E.p	E.p

Table F.5. Evaluation of intersection between the rules and the sampling points for R5 (five or more output sampling points).

Rules	Grades of membership at each sampling point								
	-L	(-L,-½L)	-½L	(-½L,0)	0	(0,½L)	½L	(½L,L)	L
Rule 1	0	0	0	0	0	min[μ,E.p]	E.p	E.p	E.p
Rule 2	0	min[μ,E.p]	E.p	E.p	E.p	E.p	E.p	min[μ,E.p]	0
Rule 3	0	$\mu_{\Delta UZ}(y_r)$	0.5	min[μ,R.p]	R.p	min[μ,R.p]	0.5	$\mu_{\Delta UZ}(y_r)$	0
Rule 4	R.n	R.n	R.n	min[μ,R.n]	0	0	0	0	0
Clipping (Qin) i.e. μ_Z	R.n	$\mu_{\Delta UZ}(y_r)$ or R.n	0.5	$\mu_{\Delta UZ}(y_r)$ or R.p	R.p	$\mu_{\Delta UZ}(y_r)$ or R.p	0.5	$\mu_{\Delta UZ}(y_r)$ or E.p	E.p

Table F.6. Evaluation of intersection between the rules and the sampling points for R6 (five or more output sampling points).

Rules	Grades of membership at each sampling point								
	-L	(-L,-½L)	-½L	(-½L,0)	0	(0,½L)	½L	(½L,L)	L
Rule 1	0	0	0	0	0	min[μ,E.p]	E.p	E.p	E.p
Rule 2	0	min[μ,E.p]	E.p	E.p	E.p	E.p	E.p	min[μ,E.p]	0
Rule 3	0	min[μ,R.p]	R.p	R.p	R.p	R.p	R.p	min[μ,R.p]	0
Rule 4	R.n	min[μ,R.n]	0.5	$\mu_{\Delta UN}(y_r)$	0	0	0	0	0
Clipping (Qin) i.e. μ_Z	R.n	$\mu_{\Delta UN}(y_r)$ or R.n	0.5	$\mu_{\Delta UN}(y_r)$ or R.p	R.p	$\mu_{\Delta UP}(y_r)$ or R.p	R.p	$\mu_{\Delta UZ}(y_r)$, E.p or R.p	E.p

Table F.7. Evaluation of intersection between the rules and the sampling points for R7 (five or more output sampling points).

Rules	Grades of membership at each sampling point								
	-L	(-L,-½L)	-½L	(-½L,0)	0	(0,½L)	½L	(½L,L)	L
Rule 1	0	0	0	0	0	min[μ,R.p]	R.p	R.p	R.p
Rule 2	0	min[μ,E.p]	E.p	E.p	E.p	E.p	E.p	min[μ,E.p]	0
Rule 3	0	min[μ,R.p]	R.p	R.p	R.p	R.p	R.p	min[μ,R.p]	0
Rule 4	E.n	min[μ,E.n]	0.5	$\mu_{\Delta UN}(y_r)$	0	0	0	0	0
Clipping (Qin) i.e. μ_Z	E.n	$\mu_{\Delta UN}(y_r)$ or E.n	0.5	$\mu_{\Delta UN}(y_r)$ or E.p	E.p	$\mu_{\Delta UP}(y_r)$ or E.p	E.p	$\mu_{\Delta UZ}(y_r)$, E.p or R.p	R.p

Table F.8. Evaluation of intersection between the rules and the sampling points for R8 (five or more output sampling points).

Rules	Grades of membership at each sampling point								
	-L	(-L,-½L)	-½L	(-½L,0)	0	(0,½L)	½L	(½L,L)	L
Rule 1	0	0	0	0	0	min[μ,R.p]	R.p	R.p	R.p
Rule 2	0	$\mu_{\Delta UZ}(y_r)$	0.5	min[μ,E.p]	E.p	min[μ,E.p]	0.5	$\mu_{\Delta UZ}(y_r)$	0
Rule 3	0	min[μ,R.p]	R.p	R.p	R.p	R.p	R.p	min[μ,R.p]	0
Rule 4	E.n	E.n	E.n	min[μ,E.n]	0	0	0	0	0
Clipping (Qin) i.e. μ_Z	E.n	$\mu_{\Delta UZ}(y_r)$ or E.n	0.5	$\mu_{\Delta UZ}(y_r)$ or E.p	E.p	$\mu_{\Delta UZ}(y_r)$ or E.p	0.5	$\mu_{\Delta UZ}(y_r)$ or R.p	R.p

Appendix G

Fuzzy Control of a First Order System With Dead Time

```
C=====================================================
C     THIS PROGRAM IS FOR THE FUZZY RULE-BASED CONTROL
C     OF A FIRST ORDER SYSTEM WITH DEAD TIME
C=====================================================
C
      COMMON /MDLPRM/ XKP, TAUP, XKD, TAUD, YP, YD
C
650   FORMAT(1H , 5(F8.4,2X))
C
      REAL UD(0:20)
      REAL DD(0:20)
C
C     Number of rules that are used
      PARAMETER ( N = 4 ,
C     Number of output sampling points
     &  M = 3 )
C
C
C     Vectors
C
      REAL CE(N)
      REAL E(N)
      REAL OUT(N)
      REAL OUTSAMPLE(M)
      REAL XMUE(2)
      REAL XMUCE(2)
      REAL YMUV(3)
      REAL ZmuRULE(N)
C
C
C     Data input
C
C     Controller
      XKC = 3.8647
      TAUIC = 6.2013
      B = 0.4
      XL = 1.5
C
C     Process
      XKP = 1.
      TAUP = 10.
C
C     Disturbance
      XKD = 1.
      TAUD = 10.
C
C     Initial values & constraints
      TSIM = 70.
      TSMPL = 0.1
C
C     Initialise parameters
      D = 0.
      ERROR = 0.
      U = 0.
      XE = 0.
      Y = 0.
      YD = 0.
      YP = 0.
      DO 200 ICOUNTER = 0, 20
         UD( ICOUNTER ) = 0.
```



```
         DD( ICOUNTER ) = 0.
200   CONTINUE
C
C
C     Change in setpoint
      YSET = 3.
C
C     Calculation of fuzzy scaling factors
      GR = B * (YSET - Y)
      GU = 3. * XKC / GR
      GE = GR / TAUIC * TSMPL
C
C
C     Open output file
C
      OPEN( UNIT = 11, FILE='SIMOUT.TXT', STATUS='UNKNOWN')
C
C
C     Start of simulation loop
C
C
      NSTEPS = INT( TSIM / TSMPL + 1.E-5)
      DO 100 I = 1, NSTEPS + 1
          T = (I - 1) * TSMPL
C
C=======================================
C          Start of controller
C=======================================
C
           EROROLD = ERROR
           XE = (YSET - Y) * GE
           ERROR = XE / GE
           XCE = (ERROR - EROROLD) * GR
           CERROR = XCE / GR
C
           XNUMERATOR = 0.
           DENOMINATOR = 0.
C
C
C          Fuzzification of the input
C
           CALL ERRORMU( XE, XMUE, XL )
           CALL CERRORMU( XCE, XMUCE, XL )
C
C          According to the fuzzy rule base:
           E(1) = XMUE(1)
           E(2) = XMUE(1)
           E(3) = XMUE(2)
           E(4) = XMUE(2)
           CE(1) = XMUCE(1)
           CE(2) = XMUCE(2)
           CE(3) = XMUCE(1)
           CE(4) = XMUCE(2)
C
C
C          Values of Y at which fuzzy answers are to be determined
C
           SAMPLE = -XL
           DO 70 IS = 1, M
             OUTSAMPLE(IS) = SAMPLE
```

```
              SAMPLE = SAMPLE + 2. * XL / (M - 1.)
70          CONTINUE
C
C           Calculation of the membership values over the fuzzy partition
C
            DO 10 J = 1, M
C
                XMAXIMUM = 0.
C
C               Fuzzification of the output
C
                CALL OUTPUTMU( OUTSAMPLE(J), YMUV, XL )
C
C               According to the fuzzy rule base:
                OUT(1) = YMUV(1)
                OUT(2) = YMUV(2)
                OUT(3) = YMUV(2)
                OUT(4) = YMUV(3)
C
C
                DO 400 IC = 1, N
C
C                 Clipping of the output fuzzy sets
C
                  ZmuRULE(IC) = MIN( E(IC), CE(IC), OUT(IC) )
C
C                 Forming the union between the sets
C
                  XMAXIMUM = MAX( XMAXIMUM, ZmuRULE(IC) )
C
400             CONTINUE
C
C
                XNUMERATOR = XNUMERATOR + OUTSAMPLE(J) * XMAXIMUM
                DENOMINATOR = DENOMINATOR + XMAXIMUM
C
10          CONTINUE
C
C           Defuzzification of the output
C
            IF (DENOMINATOR .EQ. 0. ) THEN
                CU = 0.
            ELSE
                CU = XNUMERATOR / DENOMINATOR
            END IF
C
C===========================================
C           End of controller
C===========================================
C
C           Calculation of new input
C
            U = U + CU * GU
C
C
            WRITE(11,650) T, Y, YSET, U, D
C
C           Compensation for time delay of 2 time units in input
C           and disturbance
C
```



```
            DO 300 ICOUNT = 20, 1, -1
              UD( ICOUNT ) = UD( ICOUNT - 1 )
              DD( ICOUNT ) = DD( ICOUNT - 1 )
300         CONTINUE
C
            UD(0) = U
            DD(0) = D
C
C           Extraction of required input and disturbance values
C
            UP = UD(20)
            DP = DD(20)
C
C           Process output at end of sampling interval
C             (Y Overwritten with new value)
C
            CALL MODEL(Y, UP, DP, TSMPL)
C
100     CONTINUE
C
C
            CLOSE(11)
C
            STOP
C
            END
C
C
C
C
            SUBROUTINE MODEL( Y, UP, DP, TSMPL)
C============================================================
C           INTEGRATES DYNAMIC MODEL EQUATION UNTIL TIME = TSMPL
C           UPON INPUT:
C              Y     = INITIAL Y-VALUE
C              TSMPL = INTEGRATION INTERVAL
C              U     = VALUE OF PROCESS INPUT
C
C           UPON OUTPUT:
C              Y     = Y-VALUE AT END OF TIME INTERVAL
C============================================================
            COMMON /MDLPRM / XKP, TAUP, XKD, TAUD, YP, YD
C
C           Process
            YP = XKP * UP + (YP - XKP * UP) * EXP(-TSMPL / TAUP)
C           Disturbance
            YD = XKD * DP + (YD - XKD * DP) * EXP(-TSMPL / TAUD)
C
            Y = YD + YP
C
            RETURN
            END
C
C
C===========================================================
C           Subroutines for calculating the membership grades
C           over each fuzzy partition.
C===========================================================
C
C           Determining grade of membership for triangular
```

```
C       shaped fuzzy sets
C
C       From a to b
        SUBROUTINE UP(A, B, X, XMU)
           XMU = (X - A) / (B - A)
        RETURN
        END
C
C       From b to c
        SUBROUTINE DOWN(B, C, X, XMU)
           XMU = (C - X) / (C - B)
        RETURN
        END
C
C
C       Inputs
C
C       Error
C
        SUBROUTINE ERRORMU( X, XMUE, XL )
C
        DIMENSION XMUE(1)
C
C       Constants
C
C       Error (E)    [Error in output defined as y(set) - y(actual)]
C       Negative large (ENL)
        bENL = -XL
        cENL = XL
C       Positive large (EPL)
        aEPL = -XL
        bEPL = XL
C
C       Negative large
        IF (X .LE. bENL) THEN
          XmuENL = 1.
        ELSE IF (X .GE. bENL .AND. X .LE. cENL) THEN
          CALL DOWN( bENL, cENL, X, XMU )
          XmuENL = XMU
        ELSE
          XmuENL = 0.
        END IF
        XMUE(1) = XmuENL
C
C       Positive large
        IF (X .GE. bEPL) THEN
          XmuEPL = 1.
        ELSE IF (X .GE. aEPL .AND. X .LE. bEPL) THEN
          CALL UP( aEPL, bEPL, X, XMU)
          XmuEPL = XMU
        ELSE
          XmuEPL = 0.
        END IF
        XMUE(2) = XmuEPL
C
        RETURN
        END
C
C
C       Change in error
```



```
C
        SUBROUTINE CERRORMU( X, XMUCE, XL )
C
        DIMENSION XMUCE(1)
C
C       Constants
C
C       Change in Error (CE)    [Error(t) - Error(t-1)]
C       Negative (CEN)
        bCEN = -XL
        cCEN = XL
C       Positive (CEP)
        aCEP = -XL
        bCEP = XL
C
C       Negative
        IF (X .LE. bCEN) THEN
          XmuCEN = 1.
        ELSE IF (X .GE. bCEN .AND. X .LE. cCEN) THEN
          CALL DOWN( bCEN, cCEN, X, XMU )
          XmuCEN = XMU
        ELSE
          XmuCEN = 0.
        END IF
        XMUCE(1) = XmuCEN
C
C       Positive
        IF (X .GE. bCEP) THEN
          XmuCEP = 1.
        ELSE IF (X .GE. aCEP .AND. X .LE. bCEP) THEN
          CALL UP( aCEP, bCEP, X, XMU)
          XmuCEP = XMU
        ELSE
          XmuCEP = 0.
        END IF
        XMUCE(2) = XmuCEP
C
        RETURN
        END
C
C
C
C       Output
C
C
        SUBROUTINE OUTPUTMU( Y, YMUV, XL)
C
        DIMENSION YMUV(1)
C
C       Constants
C
C       Output (V)
C       Negative (NL)
        bVNL = -XL
        cVNL = 0.
C       Zero (Z)
        aVZ = -XL
        bVZ = 0.
        cVZ = XL
C       Positive (PL)
```

```
      aVPL = 0.
      bVPL = XL
C
C     Negative
      IF (Y .LE. bVNL) THEN
        YmuVNL = 1.
      ELSE IF (Y .GE. bVNL .AND. Y .LE. cVNL) THEN
        CALL DOWN( bVNL, cVNL, Y, YMU)
        YmuVNL = YMU
      ELSE
      YmuVNL = 0.
      END IF
      YMUV(1) = YmuVNL
C
C     Zero
      IF (Y .GE. aVZ .AND. Y .LE. bVZ) THEN
        CALL UP( aVZ, bVZ, Y, YMU)
        YmuVZ = YMU
      ELSE IF (Y .GE. bVZ .AND. Y .LE. cVZ) THEN
        CALL DOWN( bVZ, cVZ, Y, YMU)
        YmuVZ = YMU
      ELSE
        YmuVZ = 0.
      END IF
      YMUV(2) = YmuVZ
C
C     Positive
      IF (Y .GE. aVPL .AND. Y .LE. bVPL) THEN
        CALL UP( aVPL, bVPL, Y, YMU)
        YmuVPL = YMU
      ELSE IF (Y .GE. bVPL) THEN
        YmuVPL = 1.
      ELSE
        YmuVPL = 0.
      END IF
      YMUV(3) = YmuVPL
C
      RETURN
      END
```

Appendix H

Fuzzy Equations and PI Control of a First Order System With Dead Time

```
C=========================================================
C       THIS PROGRAM IS USED TO CHECK THE FUZZY EQUATIONS
C       AND FOR THE PI CASE OF THE QIN STYLE CONTROLLER
C       FOR A FIRST ORDER SYSTEM WITH DEAD TIME
C=========================================================
C
        COMMON /MDLPRM/ XKP, TAUP, XKD, TAUD, YP, YD
        COMMON /CTRPRM/ USS, TAUIC, SUM, GU, GE, XKFL, XL, XE, CU
C
650     FORMAT(1H , 5(F8.4,2X))
C
        REAL UD(0:20)
        REAL DD(0:20)
C
C
C       Data input
C
C       Controller
        XKC   = 3.8647
        TAUIC = 6.2013
        B     = 0.4
        XL    = 1.5
C
C       Process
        XKP  = 1.
        TAUP = 10.
C
C       Disturbance
        XKD  = 1.
        TAUD = 10.
C
C       Initial values & constraints
        TSIM  = 70.
        TSMPL = 0.1
        USS   = 0.
C
C       Initialise parameters
        D   = 0.
        SUM = 0.
        XE  = 0.
        Y   = 0.
        YD  = 0.
        YP  = 0.
        DO 200 COUNTER = 0, 20
           UD( COUNTER ) = 0.
           DD( COUNTER ) = 0.
200     CONTINUE
C
C
C       Change in setpoint
        YSET = 3.
C
C       Calculation of fuzzy scaling factors
        GR = B * (YSET - Y)
        GU = 3. * XKC / GR
        GE = GR / TAUIC * TSMPL
C
C
C       Open output file
C
```



```
        OPEN( UNIT = 11, FILE='SIMOUT.TXT', STATUS='UNKNOWN')

C       Start of simulation loop
C
C
        NSTEPS = INT( TSIM / TSMPL + 1.E-5)
        DO 100 I = 1, NSTEPS + 1
           T = (I - 1) * TSMPL
C
C          Calling the controller
C
           CALL CTRLR( YSET, Y, U, TSMPL, GR)
C
C          Output to data file
C
           WRITE(11,650) T, Y, YSET, U, D
C
C          Compensation for time delay of 2 time units in input
C          and disturbance
C
           DO 300 ICOUNT = 20, 1, -1
             UD( ICOUNT ) = UD( ICOUNT - 1 )
             DD( ICOUNT ) = DD( ICOUNT - 1 )
300        CONTINUE
C
           UD(0) = U
           DD(0) = D
C
C          Extraction of required values
C
           UP = UD(20)
           DP = DD(20)
C
C          Process output at end of sampling interval
C             (Y overwritten with new value)
C
           CALL MODEL(Y, UP, DP, TSMPL)
100     CONTINUE
C
C
        CLOSE(11)
C
        STOP
C
        END
C
C
C
C
        SUBROUTINE MODEL( Y, UP, DP, TSMPL)
C=========================================================
C       INTEGRATES DYNAMIC MODEL EQUATION UNTIL TIME = TSMPL
C       UPON INPUT:
C          Y     = INITIAL Y-VALUE
C          TSMPL = INTEGRATION INTERVAL
C          U     = VALUE OF PROCESS INPUT
C
C       UPON OUTPUT:
C          Y     = Y-VALUE AT END OF TIME INTERVAL
C=========================================================
```

```
      COMMON /MDLPRM / XKP, TAUP, XKD, TAUD, YP, YD
C
C     Process
      YP = XKP * UP + (YP - XKP * UP) * EXP(-TSMPL / TAUP)
C     Disturbance
      YD = XKD * DP + (YD - XKD * DP) * EXP(-TSMPL / TAUD)
C
      Y = YD + YP
C
      RETURN
      END
C
C
C
C
      SUBROUTINE CTRLR( YSET, Y, U, TSMPL, GR)
C================================================
C     PI-CONTROL LAW
C     UPON INPUT:
C        YSET  = SETPOINT
C        Y     = CURRENT MEASURED VALUE
C     UPON OUTPUT:
C        U     = CONTROL OUTPUT
C================================================
        COMMON /CTRPRM/ USS, TAUIC, SUM, GU, GE, XKFL, XL, XE, CU
C
C
      ERROROLD = XE
      ERROR = (YSET - Y) * GE
      XE = ERROR / GE
      CERROR = (XE - ERROROLD) * GR
      XCE = CERROR / GR
C
C     Calculation of controller parameters
C
C     Used for the fuzzy controller
C
      XKFL = GU * GR * XL / ((3. * XL
     &      - MAX ( (ABS(ERROR)) ,
     &        (ABS(CERROR)) ) ) )
C
C     Used for the traditional PI controller
C
C     XKFL = GU * GR * XL / (3. * XL)
C
      TFL = GR / GE
C
C     Calculation of output according to the Regions the inputs
C     lie in. If the traditional PI controller is being used then
C     this section is not utilised.
C
      IF (ABS(ERROR) .LE. XL .AND. ABS(CERROR) .LE. XL) THEN
C
      CU = XKFL * XCE
     &    + XKFL / TFL * XE
C
      ELSE IF (ABS(ERROR) .LE. XL .AND. CERROR .GT. XL) THEN
        CU = (ERROR + XL) / 2. * GU
      ELSE IF (ABS(ERROR) .LE. XL .AND. CERROR .LT. -XL) THEN
        CU = (ERROR - XL) / 2. * GU
```



```
      ELSE IF (ERROR .GT. XL .AND. ABS(CERROR) .LE. XL) THEN
        CU = (CERROR + XL) / 2. * GU
      ELSE IF (ERROR .LT. -XL .AND. ABS(CERROR) .LE. XL) THEN
        CU = (CERROR - XL) / 2. * GU
      ELSE IF (ERROR .GT. XL .AND. CERROR .GT. XL) THEN
        CU = XL * GU
      ELSE IF (ERROR .LT. -XL .AND. CERROR .LT. -XL) THEN
        CU =  -XL * GU
      ELSE
        CU = 0.
      END IF
C
C     Final output determined
C
      U = CU + U
C
      RETURN
      END
```

Appendix I

Fuzzy Control of a Second Order System

```
C=======================================================
C       THIS PROGRAM IS FOR THE FUZZY RULE-BASED CONTROL
C       OF A SECOND ORDER SYSTEM WITH NO DEAD TIME
C=======================================================
C
        COMMON /MDLPRM/ XKP, TAUP, XKD, TAUD, YP, YD
C
650     FORMAT(1H , 5(F8.4,2X))
C
        REAL UIN(2)
        REAL YOUT(2)
C
C       Number of rules that are used
        PARAMETER ( N = 4 ,
C       Number of output sampling points
     &  M = 3 )
C
C
C       Vectors
C
        REAL CE(N)
        REAL E(N)
        REAL OUT(N)
        REAL OUTSAMPLE(M)
        REAL XMUCE(2)
        REAL XMUE(2)
        REAL YMUV(3)
        REAL ZmuRULE(N)
C
C
C       Data input
C
C       Controller
        XKC   = .64011
        TAUIC = 5.21573
        B     = 0.4
        XL    = 2.
C
C       Disturbance
        XKD  = 1.
        TAUD = 10.
C
C       Initial values & constraints
        TSIM  = 30.
        TSMPL = 0.1
C
C       Initialise paramaters
        D       = 0.
        ERROR   = 0.
        U       = 0.
        XE      = 0.
        Y       = 0.
        YD      = 0.
        YP      = 0.
        YOUT(1) = 0.
        YOUT(2) = 0.
        UIN(1)  = 0.
        UIN(2)  = 0.
C
C
```



```
C       Change in setpoint
        YSET = 3.
C
C       Calculation of fuzzy scaling factors
        GR = B * (YSET - Y)
        GU = 3. * XKC / GR
        GE = GR / TAUIC * TSMPL
C
C
C       Open output file
C
        OPEN( UNIT = 11, FILE='SIMOUT.TXT', STATUS='UNKNOWN')
C
C       Start of simulation loop
C
C
        NSTEPS = INT( TSIM / TSMPL + 1.E-5)
        DO 100 I = 1, NSTEPS + 1
           T = (I - 1) * TSMPL
C
C=======================================
C          START OF CONTROLLER
C=======================================
C
           ERROROLD = ERROR
           XE = (YSET - Y) * GE
           ERROR = XE / GE
           XCE = (ERROR - ERROROLD) * GR
           CERROR = XCE / GR
C
           XNUMERATOR = 0.
           DENOMINATOR = 0.
C
C          Fuzzification of the input
C
           CALL ERRORMU( XE, XMUE, XL )
           CALL CERRORMU( XCE, XMUCE, XL )
C
C          According to the fuzzy rule base:
           E(1) = XMUE(1)
           E(2) = XMUE(1)
           E(3) = XMUE(2)
           E(4) = XMUE(2)
           CE(1) = XMUCE(1)
           CE(2) = XMUCE(2)
           CE(3) = XMUCE(1)
           CE(4) = XMUCE(2)
C
C
C          Values of Y at which fuzzy answers are to be determined
C
           SAMPLE = -XL
           DO 70 IS = 1, M
             OUTSAMPLE(IS) = SAMPLE
             SAMPLE = SAMPLE + 2. * XL / (M - 1.)
70         CONTINUE
C
C          Calculation of the membership values over the fuzzy partition
C
           DO 10 J = 1, M
```

```
C
              XMAXIMUM = 0.
C
              CALL OUTPUTMU( OUTSAMPLE(J), YMUV, XL )
C
C             According to the fuzzy rule base:
              OUT(1) = YMUV(1)
              OUT(2) = YMUV(2)
              OUT(3) = YMUV(2)
              OUT(4) = YMUV(3)
C
C
             DO 400 IC = 1, N
C
C              Clipping of the output fuzzy sets
C
               ZmuRULE(IC) = MIN( E(IC), CE(IC), OUT(IC) )
C
C              Forming the union between the sets
C
               XMAXIMUM = MAX( XMAXIMUM, ZmuRULE(IC) )
C
400           CONTINUE
C
C
              XNUMERATOR = XNUMERATOR + OUTSAMPLE(J) * XMAXIMUM
              DENOMINATOR = DENOMINATOR + XMAXIMUM
C
10          CONTINUE
C
C           Defuzzification of the output
C
            IF (DENOMINATOR .EQ. 0. ) THEN
                 CU = 0.
            ELSE
                 CU = XNUMERATOR / DENOMINATOR
            END IF
C
C===========================================
C           END OF CONTROLLER
C===========================================
C
C           Calculation of new input
C
            U = U + CU * GU
C
C
            WRITE(11,650) T, Y, YSET, U, D
C
            CALL MODEL(Y, U, D, TSMPL, YOUT, UIN)
C
100       CONTINUE
C
C
          CLOSE(11)
C
          STOP
C
          END
C
```



```
C
C
C
          SUBROUTINE MODEL( Y, U, D, TSMPL, YOUT, UIN)
C=========================================
C         INTEGRATES DYNAMIC MODEL EQUATION UNTIL TIME = TSMPL
C         UPON INPUT:
C            Y     = INITIAL Y-VALUE
C            TSMPL = INTEGRATION INTERVAL
C            U     = VALUE OF PROCESS INPUT
C
C         UPON OUTPUT:
C            Y     = Y-VALUE AT END OF TIME INTERVAL
C=========================================
          COMMON /MDLPRM / XKP, TAUP, XKD, TAUD, YP, YD
C
          DIMENSION YOUT(1)
          DIMENSION UIN(1)
C
          YOUT(2) = YOUT(1)
          UIN(2) = UIN(1)
          YOUT(1) = YP
          UIN(1) = U
C
C         Process
          YP = (1. + EXP(-TSMPL)) * YOUT(1)
     &         - EXP(-TSMPL) * YOUT(2)
     &         - (1. - TSMPL - EXP(-TSMPL)) * UIN(1)
     &         + (1. - EXP(-TSMPL) - TSMPL * EXP(-TSMPL)) * UIN(2)
C
C         Disturbance
          YD = XKD * D + (YD - XKD * D) * EXP(-TSMPL / TAUD)
C
          Y = YD + YP
C
          RETURN
          END
C
C
C=========================================================
C         Subroutines for calculating the membership grades
C         over each fuzzy partition.
C=========================================================
C
C         Determining grade of membership for triangular
C         shaped fuzzy sets
C
C         From a to b
          SUBROUTINE UP(A, B, X, XMU)
             XMU = (X - A) / (B - A)
          RETURN
          END
C
C         From b to c
          SUBROUTINE DOWN(B, C, X, XMU)
             XMU = (C - X) / (C - B)
          RETURN
          END
C
C
```

```
C       Inputs
C
C       Error
C
        SUBROUTINE ERRORMU( X, XMUE, XL )
C
        DIMENSION XMUE(1)
C
C       Constants
C
C       Error (E)    [Error in output defined as y(set) - y(actual)]
C       Negative (ENL)
        bENL = -XL
        cENL = XL
C       Positive (EPL)
        aEPL = -XL
        bEPL = XL
C
C       Negative
        IF (X .LE. bENL) THEN
          XmuENL = 1.
        ELSE IF (X .GE. bENL .AND. X .LE. cENL) THEN
          CALL DOWN( bENL, cENL, X, XMU )
          XmuENL = XMU
        ELSE
          XmuENL = 0.
        END IF
        XMUE(1) = XmuENL
C
C       Positive
        IF (X .GE. bEPL) THEN
          XmuEPL = 1.
        ELSE IF (X .GE. aEPL .AND. X .LE. bEPL) THEN
          CALL UP( aEPL, bEPL, X, XMU)
          XmuEPL = XMU
        ELSE
          XmuEPL = 0.
        END IF
        XMUE(2) = XmuEPL
C
        RETURN
        END
C
C
C       Change in error
C
        SUBROUTINE CERRORMU( X, XMUCE, XL )
C
        DIMENSION XMUCE(1)
C
C       Constants
C
C       Change in Error (CE)    [Error(t) - Error(t-1)]
C       Negative (CEN)
        bCEN = -XL
        cCEN = XL
C       Positive (CEP)
        aCEP = -XL
        bCEP = XL
C
```



```
C       Negative
        IF (X .LE. bCEN) THEN
          XmuCEN = 1.
        ELSE IF (X .GE. bCEN .AND. X .LE. cCEN) THEN
          CALL DOWN( bCEN, cCEN, X, XMU )
          XmuCEN = XMU
        ELSE
          XmuCEN = 0.
        END IF
        XMUCE(1) = XmuCEN
C
C       Positive
        IF (X .GE. bCEP) THEN
          XmuCEP = 1.
        ELSE IF (X .GE. aCEP .AND. X .LE. bCEP) THEN
          CALL UP( aCEP, bCEP, X, XMU)
          XmuCEP = XMU
        ELSE
          XmuCEP = 0.
        END IF
        XMUCE(2) = XmuCEP
C
        RETURN
        END
C
C
C
C       Outputs
C
        SUBROUTINE OUTPUTMU( Y, YMUV, XL)
C
        DIMENSION YMUV(1)
C
C       Constants
C
C       Output (V)
C       Negative (NL)
        bVNL = -XL
        cVNL = 0.
C       Zero (Z)
        aVZ = -XL
        bVZ = 0.
        cVZ = XL
C       Positive (PL)
        aVPL = 0.
        bVPL = XL
C
C       Negative
        IF (Y .LE. bVNL) THEN
          YmuVNL = 1.
        ELSE IF (Y .GE. bVNL .AND. Y .LE. cVNL) THEN
          CALL DOWN( bVNL, cVNL, Y, YMU)
          YmuVNL = YMU
        ELSE
        YmuVNL = 0.
        END IF
        YMUV(1) = YmuVNL
C
C       Zero
        IF (Y .GE. aVZ .AND. Y .LE. bVZ) THEN
```

```
          CALL UP( aVZ, bVZ, Y, YMU)
          YmuVZ = YMU
        ELSE IF (Y .GE. bVZ .AND. Y .LE. cVZ) THEN
          CALL DOWN( bVZ, cVZ, Y, YMU)
          YmuVZ = YMU
        ELSE
          YmuVZ = 0.
        END IF
        YMUV(2) = YmuVZ
C
C       Positive
        IF (Y .GE. aVPL .AND. Y .LE. bVPL) THEN
          CALL UP( aVPL, bVPL, Y, YMU)
          YmuVPL = YMU
        ELSE IF (Y .GE. bVPL) THEN
          YmuVPL = 1.
        ELSE
          YmuVPL = 0.
        END IF
        YMUV(3) = YmuVPL
C
        RETURN
        END
```

Appendix J

PI control of a Second Order System

```
C=====================================================
C       NORMAL PI CONTROL OF A SECOND ORDER SYSTEM
C       WITH NO DEAD TIME
C=====================================================
C
        COMMON /MDLPRM/ XKP, TAUP, XKD, TAUD, YP, YD
        COMMON /CTRPRM/ USS, TAUIC, SUM, GU, GE, XKFL, XL, XE, CU
C
650     FORMAT(1H , 5(F8.4,2X))
C
        REAL YOUT(2)
        REAL UIN(2)
C
C       Data input
C
C       Controller
        XKC   = .64011
        TAUIC = 5.21573
        B     = 0.4
        XL    = 2.
C
C
C       Initial values & constraints
        TSIM  = 30.
        TSMPL = 0.1
        USS   = 0.
C
C       INITIALIZE PARAMETERS
        D    = 0.
        SUM = 0.
        XE  = 0.
        Y   = 0.
        YD  = 0.
        YP  = 0.
        YOUT(1) = 0.
        UIN(1)  = 0.
        YOUT(2) = 0.
        UIN(2)  = 0.
C
C
C       Change in setpoint
        YSET = 3.
C
C       Calculation of fuzzy scaling factors
        GR = B * (YSET - Y)
        GU = 3. * XKC / GR
        GE = GR / TAUIC * TSMPL
C
C
C       Open output file
C
        OPEN( UNIT = 11, FILE='SIMOUT.TXT', STATUS='UNKNOWN')

C       Start of simulation loop
C
        NSTEPS = INT( TSIM / TSMPL + 1.E-5)
        DO 100 I = 1, NSTEPS + 1
           T = (I - 1) * TSMPL
C
C          Calling the controller
```



```
C
           CALL CTRLR( YSET, Y, U, TSMPL, GR)
C
C          Output to data file
C
           WRITE(11,650) T, Y, YSET, U, D
C
           CALL MODEL(Y, U, D, TSMPL, YOUT, UIN)
C
100     CONTINUE
C
        CLOSE(11)
C
        STOP
C
        END
C
C
C
C
        SUBROUTINE MODEL( Y, U, D, TSMPL, YOUT, UIN)
C=====================================================
C       INTEGRATES DYNAMIC MODEL EQUATION UNTIL TIME = TSMPL
C       UPON INPUT:
C          Y     = INITIAL Y-VALUE
C          TSMPL = INTEGRATION INTERVAL
C          U     = VALUE OF PROCESS INPUT
C
C       UPON OUTPUT:
C          Y     = Y-VALUE AT END OF TIME INTERVAL
C=====================================================
        COMMON /MDLPRM / XKP, TAUP, XKD, TAUD, YP, YD
C
        DIMENSION YOUT(1)
        DIMENSION UIN(1)
C
        YOUT(2) = YOUT(1)
        UIN(2) = UIN(1)
        YOUT(1) = YP
        UIN(1) = U
C
C       Process
        YP = (1. + EXP(-TSMPL)) * YOUT(1)
     &       - (EXP(-TSMPL)) * YOUT(2)
     &       - (1. - TSMPL - EXP(-TSMPL)) * UIN(1)
     &       + (1. - EXP(-TSMPL) - TSMPL * EXP(-TSMPL)) * UIN(2)
C
C       Disturbance
        YD = 1. * D + (YD - 1. * D) * EXP(-TSMPL / 1.)
C
        Y = YD + YP
C
        RETURN
        END
C
C
C
C
        SUBROUTINE CTRLR( YSET, Y, U, TSMPL, GR)
C=====================================================
```

```
C       PI-CONTROL LAW
C       UPON INPUT:
C          YSET  = SETPOINT
C          Y     = CURRENT MEASURED VALUE
C       UPON OUTPUT:
C          U     = CONTROL OUTPUT
C=================================================
        COMMON /CTRPRM/ USS, TAUIC, SUM, GU, GE, XKFL, XL, XE, CU
C
        ERROROLD = XE
        ERROR = (YSET - Y) * GE
        XE = ERROR / GE
        CERROR = (XE - ERROROLD) * GR
        XCE = CERROR / GR
C
C       Calculation of PI controller parameters
C
        XKFL = GU * GR * XL / (3. * XL)
        TFL = GR / GE
C
        CU = XKFL * XCE
     &     + XKFL / TFL * XE
C
C       Final output determined
C
        U = CU + U
C
        RETURN
        END
```

Appendix K

Fuzzy Control of a Third Order System With Dead Time

```
C=======================================================
C       THIS PROGRAM IS FOR THE FUZZY RULE-BASED CONTROL
C       OF A THIRD ORDER SYSTEM WITH DEAD TIME
C=======================================================
        COMMON /MDLPRM/ XKP, TAUP, XKD, TAUD, YP, YD
C
650     FORMAT(1H , 5(F8.4,2X))
C
        REAL UD(0:2)
        REAL DD(0:2)
        REAL YOUT(3)
        REAL UIN(3)
C
C       Number of rules that are used
        PARAMETER ( N = 4 ,
C       Number of output sampling points
     &  M = 3 )
C
C
C       Vectors
C
        REAL CE(N)
        REAL E(N)
        REAL OUT(N)
        REAL OUTSAMPLE(M)
        REAL XMUCE(2)
        REAL XMUE(2)
        REAL YMUV(3)
        REAL ZmuRULE(N)
C
C
C       Data input
C
C       Controller
        XKC   = 10.412288
        TAUIC = 1.2543632
        B     = 0.4
        XL    = 1.5
C
C       Disturbance
        XKD = 1.
        TAUD = 10.
C
C       Initial values & constraints
        TSIM  = 30.
        TSMPL = 0.1
C
C       Initialise parameters
        D     = 0.
        ERROR = 0.
        Y = 0.
        YD = 0.
        YP = 0.
        U = 0.
        XE = 0.
        DO 54 IG = 1, 3
          YOUT(IG) = 0.
          UIN(IG)  = 0.
54      CONTINUE
        DO 200 ICOUNTER = 0, 2
```



```
           UD( ICOUNTER ) = 0.
           DD( ICOUNTER ) = 0.
200     CONTINUE
C
C       Change in setpoint
        YSET = 3.
C
C       Calculation of fuzzy scaling factors
        GR = B * (YSET - Y)
        GU = 3. * XKC / GR
        GE = GR / TAUIC * TSMPL
C
C
C       Open output file
C
        OPEN( UNIT = 11, FILE='SIMOUT.TXT', STATUS='UNKNOWN')
C
C       Start of simulation loop
C

        NSTEPS = INT( TSIM / TSMPL + 1.E-5)
        DO 100 I = 1, NSTEPS + 1
           T = (I - 1) * TSMPL
C
C=======================================
C          START OF CONTROLLER
C=======================================
C
           ERROROLD = ERROR
           XE = (YSET - Y) * GE
           ERROR = XE / GE
           XCE = (ERROR - ERROROLD) * GR
           CERROR = XCE / GR
C
           XNUMERATOR = 0.
           DENOMINATOR = 0.
C
C          Fuzzification of the input
C
           CALL ERRORMU( XE, XMUE, XL )
           CALL CERRORMU( XCE, XMUCE, XL )
C
C          According to the fuzzy rule base:
           E(1) = XMUE(1)
           E(2) = XMUE(1)
           E(3) = XMUE(2)
           E(4) = XMUE(2)
           CE(1) = XMUCE(1)
           CE(2) = XMUCE(2)
           CE(3) = XMUCE(1)
           CE(4) = XMUCE(2)
C
C
C          Values of Y at which fuzzy answers are to be determined
C
           SAMPLE = -XL
           DO 70 IS = 1, M
              OUTSAMPLE(IS) = SAMPLE
                SAMPLE = SAMPLE + 2. * XL / (M - 1.)
70         CONTINUE
```

```
C
C          Calculation of the membership values over the fuzzy partition
C
           DO 10 J = 1, M
C
                XMAXIMUM = 0.
C
                CALL OUTPUTMU( OUTSAMPLE(J), YMUV, XL )
C
C               According to the fuzzy rule base:
                OUT(1) = YMUV(1)
                OUT(2) = YMUV(2)
                OUT(3) = YMUV(2)
                OUT(4) = YMUV(3)
C
C
              DO 400 IC = 1, N
C
C               Clipping of the output fuzzy sets
C
                ZmuRULE(IC) = MIN( E(IC), CE(IC), OUT(IC) )
C
C               Forming the union between the sets
C
                XMAXIMUM = MAX( XMAXIMUM, ZmuRULE(IC) )
C
400           CONTINUE
C
C
              XNUMERATOR = XNUMERATOR + OUTSAMPLE(J) * XMAXIMUM
              DENOMINATOR = DENOMINATOR + XMAXIMUM
C
10         CONTINUE
C
C          Defuzzification of the output
C
           IF (DENOMINATOR .EQ. 0. ) THEN
                CU = 0.
           ELSE
                CU = XNUMERATOR / DENOMINATOR
           END IF
C
C=========================================
C          END OF CONTROLLER
C=========================================
C
C          Calculation of new input
C
           U = U + CU * GU
C
C
           WRITE(11,650) T, Y, YSET, U, D
C
C          Compensation for time delay of 0.2 time units in input
C          and disturbance
C
           DO 300 ICOUNT = 2, 1, -1
              UD( ICOUNT ) = UD( ICOUNT - 1 )
              DD( ICOUNT ) = DD( ICOUNT - 1 )
300        CONTINUE
```



```
C
              UD(0) = U
              DD(0) = D
C
C          Extraction of required values
C
              UP = UD(2)
              DP = DD(2)
C
C          Process output at end of sampling interval
C             (Y overwritten with new value)
C
              CALL MODEL(Y, UP, DP, TSMPL, YOUT, UIN)
C
100        CONTINUE
C
C
           CLOSE(11)
C
           STOP
C
           END
C
C
C
C
           SUBROUTINE MODEL( Y, UP, DP, TSMPL, YOUT, UIN)
C=======================================
C          INTEGRATES DYNAMIC MODEL EQUATION UNTIL TIME = TSMPL
C          UPON INPUT:
C             Y     = INITIAL Y-VALUE
C             TSMPL = INTEGRATION INTERVAL
C             U     = VALUE OF PROCESS INPUT
C
C          UPON OUTPUT:
C             Y     = Y-VALUE AT END OF TIME INTERVAL
C=======================================
           COMMON /MDLPRM / XKP, TAUP, XKD, TAUD, YP, YD
C
           DIMENSION YOUT(1)
           DIMENSION UIN(1)
C
           YOUT(3) = YOUT(2)
           YOUT(2) = YOUT(1)
           YOUT(1) = YP
           UIN(3) = UIN(2)
           UIN(2) = UIN(1)
           UIN(1) = UP
C
C
           A1 = -(1. + EXP(-2. * TSMPL) + EXP(-3. * TSMPL))
           A2 = EXP(-2. * TSMPL) + EXP(-3. * TSMPL) + EXP(-5. * TSMPL)
           A3 = - EXP(-5. * TSMPL)
           B1 = 1. / 6. * (TSMPL - 1. / 6. * (1. + EXP(-2. * TSMPL)
     &          + EXP(-3. * TSMPL)) - 4. / 3. * (2. + EXP(-2. *TSMPL))
     &          + 3. / 2. * (2. + EXP(-3. * TSMPL)))
           B2 = - 1. / 6. * (TSMPL * (EXP(-2. * TSMPL)
     &          + EXP(-3. * TSMPL)) - 1. / 6. * (EXP(-2. * TSMPL)
     &          + EXP(-3. * TSMPL) + EXP(-5. * TSMPL))
     &          + 3. / 2. * (1. + 2. * EXP(-3. * TSMPL))
```

```
     &       - 4. / 3. * (1. + 2. * EXP(-2. * TSMPL)))
          B3 = 1. / 6. * ((6. * (TSMPL - 1.) + 5.) / 6.
     &       * EXP(-5. * TSMPL) - 4. / 3. * EXP(-2. * TSMPL)
     &       + 3. / 2. * EXP(-3. * TSMPL))
C
C         Process
          YP = - A1 * YOUT(1) - A2 * YOUT(2) - A3 * YOUT(3)
     &       + B1 * UIN(1) + B2 * UIN(2) + B3 * UIN(3)
C
C         Disturbance
          YD = XKD * DP + (YD - XKD * DP) * EXP(-TSMPL / TAUD)
C
          Y = YD + YP
C
          RETURN
          END
C
C
C=========================================================
C         Subroutines for calculating the membership grades
C         over each fuzzy partition.
C=========================================================
C
C         Determining the grade of membership for triangular
C         shaped fuzzy sets
C
C         From a to b
          SUBROUTINE UP(A, B, X, XMU)
             XMU = (X - A) / (B - A)
          RETURN
          END
C
C         From b to c
          SUBROUTINE DOWN(B, C, X, XMU)
             XMU = (C - X) / (C - B)
          RETURN
          END
C
C
C         Inputs
C
C         Error
C
          SUBROUTINE ERRORMU( X, XMUE, XL )
C
          DIMENSION XMUE(1)
C
C         Constants
C
C         Error (E)    [Error in output defined as y(set) - y(actual)]
C         Negative (ENL)
          bENL = -XL
          cENL = XL
C         Positive (EPL)
          aEPL = -XL
          bEPL = XL
C
C         Negative
          IF (X .LE. bENL) THEN
            XmuENL = 1.
```



```
          ELSE IF (X .GE. bENL .AND. X .LE. cENL) THEN
            CALL DOWN( bENL, cENL, X, XMU )
            XmuENL = XMU
          ELSE
            XmuENL = 0.
          END IF
          XMUE(1) = XmuENL
C
C         Positive
          IF (X .GE. bEPL) THEN
            XmuEPL = 1.
          ELSE IF (X .GE. aEPL .AND. X .LE. bEPL) THEN
            CALL UP( aEPL, bEPL, X, XMU)
            XmuEPL = XMU
          ELSE
            XmuEPL = 0.
          END IF
          XMUE(2) = XmuEPL
C
          RETURN
          END
C
C
C         Change in error
C
          SUBROUTINE CERRORMU( X, XMUCE, XL )
C
          DIMENSION XMUCE(1)
C
C         Constants
C
C         Change in Error (CE)    [Error(t) - Error(t-1)]
C         Negative (CEN)
          bCEN = -XL
          cCEN = XL
C         Positive (CEP)
          aCEP = -XL
          bCEP = XL
C
C         Negative
          IF (X .LE. bCEN) THEN
            XmuCEN = 1.
          ELSE IF (X .GE. bCEN .AND. X .LE. cCEN) THEN
            CALL DOWN( bCEN, cCEN, X, XMU )
            XmuCEN = XMU
          ELSE
            XmuCEN = 0.
          END IF
          XMUCE(1) = XmuCEN
C
C         Positive
          IF (X .GE. bCEP) THEN
            XmuCEP = 1.
          ELSE IF (X .GE. aCEP .AND. X .LE. bCEP) THEN
            CALL UP( aCEP, bCEP, X, XMU)
            XmuCEP = XMU
          ELSE
            XmuCEP = 0.
          END IF
          XMUCE(2) = XmuCEP
```

```
C
      RETURN
      END
C
C
C
C     Outputs
C
      SUBROUTINE OUTPUTMU( Y, YMUV, XL)
C
      DIMENSION YMUV(1)
C
C     Constants
C
C     Output (V)
C
C     Negative (NL)
      bVNL = -XL
      cVNL = 0.
C     Zero (Z)
      aVZ = -XL
      bVZ = 0.
      cVZ = XL
C     Positive (PL)
      aVPL = 0.
      bVPL = XL
C
C     Negative
      IF (Y .LE. bVNL) THEN
        YmuVNL = 1.
      ELSE IF (Y .GE. bVNL .AND. Y .LE. cVNL) THEN
        CALL DOWN( bVNL, cVNL, Y, YMU)
        YmuVNL = YMU
      ELSE
      YmuVNL = 0.
      END IF
      YMUV(1) = YmuVNL
C
C     Zero
      IF (Y .GE. aVZ .AND. Y .LE. bVZ) THEN
        CALL UP( aVZ, bVZ, Y, YMU)
        YmuVZ = YMU
      ELSE IF (Y .GE. bVZ .AND. Y .LE. cVZ) THEN
        CALL DOWN( bVZ, cVZ, Y, YMU)
        YmuVZ = YMU
      ELSE
        YmuVZ = 0.
      END IF
      YMUV(2) = YmuVZ
C
C     Positive
      IF (Y .GE. aVPL .AND. Y .LE. bVPL) THEN
        CALL UP( aVPL, bVPL, Y, YMU)
        YmuVPL = YMU
      ELSE IF (Y .GE. bVPL) THEN
        YmuVPL = 1.
      ELSE
        YmuVPL = 0.
      END IF
      YMUV(3) = YmuVPL
```



```
C
      RETURN
      END
```

Appendix L

PI control of a Third Order System With Dead Time

```
C=====================================================
C       NORMAL PI CONTROL OF A THIRD ORDER SYSTEM
C       WITH DEAD TIME
C=====================================================
C
        COMMON /MDLPRM/ XKP, TAUP, XKD, TAUD, YP, YD
        COMMON /CTRPRM/ USS, TAUIC, SUM, GU, GE, XKFL, XL, XE, TFL
C
650     FORMAT(1H , 5(F8.4,2X))
C
        REAL UD(0:2)
        REAL DD(0:2)
        REAL YOUT(3)
        REAL UIN(3)
C
C       Data input
C
C       Controller
        XKC   = 10.412288
        TAUIC = 1.2543632
        B     = 0.4
        XL    = 1.5
C
C
C       Initial values & constraints
        SIM   = 30.
        TSMPL = 0.1
        USS   = 0.
C
C       Initialise parameters
        D   = 0.
        SUM = 0.
        XE  = 0.
        Y   = 0.
        YD  = 0.
        DO 54 IG   = 1, 3
          YOUT(IG) = 0.
          UIN(IG)  = 0.
54      CONTINUE
        DO 200 COUNTER = 0, 2
           UD( COUNTER ) = 0.
           DD( COUNTER ) = 0.
200     CONTINUE
C
C       Change in setpoint
        YSET = 3.
C
C       Calculation of fuzzy scaling factors
        GR = B * (YSET - Y)
        GU = 3. * XKC / GR
        GE = GR / TAUIC * TSMPL
C
C
C       Open output file
C
        OPEN( UNIT = 11, FILE='SIMOUT.TXT', STATUS='UNKNOWN')
C
C       Start of simulation loop
C
        NSTEPS = INT( TSIM / TSMPL + 1.E-5)
```



```
        DO 100 I = 1, NSTEPS + 1
           T = (I - 1) * TSMPL
C
C          Calling the controller
C
           CALL CTRLR( YSET, Y, U, TSMPL, GR)
C
C          Output to data file
C
           WRITE(11,650) T, Y, YSET, U, D
C
C          Compensation for time delay of 0.2 time units in input
C          and disturbance
C
           DO 300 ICOUNT = 2, 1, -1
                UD( ICOUNT ) = UD( ICOUNT - 1 )
                DD( ICOUNT ) = DD( ICOUNT - 1 )
300        CONTINUE
C
           UD(0) = U
           DD(0) = D
C
C          Extraction of required values
C
           UP = UD(2)
           DP = DD(2)
C
C          Process output at end of sampling interval
C             (Y overwritten with new value)
C
           CALL MODEL(Y, UP, DP, TSMPL, YOUT, UIN)
C
100     CONTINUE
C
        CLOSE(11)
C
        STOP
C
        END
C
C
C
C
        SUBROUTINE MODEL( Y, UP, DP, TSMPL, YOUT, UIN)
C=====================================================
C       INTEGRATES DYNAMIC MODEL EQUATION UNTIL TIME = TSMPL
C       UPON INPUT:
C          Y     = INITIAL Y-VALUE
C          TSMPL = INTEGRATION INTERVAL
C          U     = VALUE OF PROCESS INPUT
C
C       UPON OUTPUT:
C          Y     = Y-VALUE AT END OF TIME INTERVAL
C=====================================================
        COMMON /MDLPRM / XKP, TAUP, XKD, TAUD, YP, YD
C
        DIMENSION YOUT(1)
        DIMENSION UIN(1)
C
        YOUT(3) = YOUT(2)
```

```
      YOUT(2) = YOUT(1)
      YOUT(1) = YP
      UIN(3) = UIN(2)
      UIN(2) = UIN(1)
      UIN(1) = UP
C
      A1 = - (1. + EXP(-2. * TSMPL) + EXP(-3. * TSMPL))
      A2 = EXP(-2. * TSMPL) + EXP(-3. * TSMPL) + EXP(-5. * TSMPL)
      A3 = - EXP(-5. * TSMPL)
      B1 = 1. / 6. * (TSMPL - 1. / 6. * (1. + EXP(-2. * TSMPL)
     &       + EXP(-3. * TSMPL)) - 4. / 3. * (2. + EXP(-2. * TSMPL))
     &       + 3. / 2. * (2. + EXP(-3. * TSMPL)))
      B2 = - 1. / 6. * (TSMPL * (EXP(-2. * TSMPL) + EXP(-3. * TSMPL))
     &       - 1. / 6. * (EXP(-2. * TSMPL) + EXP(-3. * TSMPL)
     &       + EXP(-5. * TSMPL)) + 3. / 2. * (1. + 2. * EXP(-3. *TSMPL))
     &       - 4. / 3. * (1. + 2. * EXP(-2. * TSMPL)))
      B3 = 1. / 6. * ((6. * (TSMPL - 1.) + 5.) / 6. * EXP(-5. * TSMPL)
     &       - 4. / 3. * EXP(-2. * TSMPL) + 3. / 2. * EXP(-3. * TSMPL))
C
C     Process
      YP = - A1 * YOUT(1) - A2 * YOUT(2) - A3 * YOUT(3)
     &       + B1 * UIN(1) + B2 * UIN(2) + B3 * UIN(3)
C
C     Disturbance
      YD = 1. * DP + (YD - 1. * DP) * EXP(-TSMPL / 1.)
C
      Y = YD + YP
C
      RETURN
      END
C
C
C
C
      SUBROUTINE CTRLR( YSET, Y, U, TSMPL, GR)
C=====================================================
C     PI-CONTROL LAW
C     UPON INPUT:
C        YSET  = SETPOINT
C        Y     = CURRENT MEASURED VALUE
C     UPON OUTPUT:
C        U     = CONTROL OUTPUT
C=====================================================
        COMMON /CTRPRM/ USS, TAUIC, SUM, GU, GE, XKFL, XL, XE, TFL
C
      ERROROLD = XE
      ERROR = (YSET - Y) * GE
      XE = ERROR / GE
      CERROR = (XE - ERROROLD) * GR
      XCE = CERROR / GR
C
C     Calculation of PI controller parameters
C
      XKFL = GU * GR * XL / (3. * XL)
      TFL = GR / GE
C
      CU = XKFL * XCE
     &    + XKFL / TFL * XE
C
C     Final output determined
```



```
C
      U = CU + U
C
      RETURN
      END
```

Appendix M

Open Loop Control of a Lysine Fermentation

```
C=====================================================================
C     THIS FORTRAN PROGRAM INTEGRATES THE FOUR DIFFERNETIAL EQUATIONS
C     FOR L-LYSINE PRODUCTION FROM THE PAPER OF OHNO ET AL (1976).
C     THE OPEN-LOOP RESPONSE IS INVESTIGATED.
C=====================================================================
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C
C     Parameters
      INTEGER N,IW
      PARAMETER (N=4, IW=14, K=4)
C
C     Local Arrays
      DIMENSION W(N,IW), C(N)
C
C     External subroutines
      EXTERNAL FCN
C
C     Executable statements
C
15    FORMAT (4 (1X,F10.3) )
C
C     Vectors
      DIMENSION Y(K)
C
      COMMON /OL/ F, S1
C
C     Initial settings and constraints (As stated in Chapter 4)
C     Biomass
      C(1) = 0.01
C     Substrate
      C(2) = 2.8
C     Product
      C(3) = 0.
C     Volume
      C(4) = 2.
C
      S1   = 2.8
      STEP = 0.2
      T    = 50.
C
      M     = 75
      YNORM = 0
      TOL   = 1.0e-4
      STIFF = 1.0e0
      IR    = 0
      XEND  = 0.
      IFAIL = 1
C
C
      OPEN( UNIT = 11, FILE = 'A.TXT', STATUS = 'UNKNOWN' )
C
      DO 100 COUNT = 0, T / STEP
C
        X = XEND
        XEND = X + STEP
C
C       Setting the inlet flowrate
        F = .5
C
```



```
C       Storing the results to file
        WRITE (11,15) C(1), C(2), C(3), C(4)
C
C       Integration of the fermentation model
C
        CALL D02BDF( X, XEND, N, C, TOL, IR,
     &   FCN, STIFF, YNORM, W, IW, M, OUT, IFAIL)
C
C
100   CONTINUE
C
      CLOSE(11)
C
      STOP
      END
C
C
C
      SUBROUTINE FCN( T, Y, Z)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C     Parameters
C
      PARAMETER (N=4)
C
C     Array arguments
      DIMENSION Z(N), Y(N)
C
      COMMON /OL/ F, S1
C
C     Parameter value
      Ys     = 0.135
C
C     Specific growth rate
      XMU    = .125 * Y(2)
C
C     Specific product formation rate
      IF (XMU .LE. 0. .OR. XMU .GE. .34896 ) THEN
        Qp   = 0.
      ELSE
        Qp   = -384. * XMU ** 2. + 134. * XMU
      END IF
C
C     Specific substrate utilisation rate
      SIGMA  = XMU / Ys
C
C     Executable statements
      Z(1)   = XMU * Y(1) - F / Y(4) * Y(1)
      Z(2)   = F / Y(4) * ( S1 - Y(2) ) - SIGMA * Y(1)
      Z(3)   = Qp * Y(1) - F * Y(3) / Y(4)
      Z(4)   = F

      RETURN
      END
```

Appendix N

Open Loop Control of a Penicillin Fermentation

```
C=====================================================================
C      THIS FORTRAN PROGRAM INTEGRATES THE FOUR DIFFERENTIAL EQUATIONS
C      FOR PENICILLIN PRODUCTION FROM THE PAPER OF SAN AND
C      STEPHANOPOULOS (1989). THE OPEN-LOOP RESPONSE IS INVESTIGATED.
C=====================================================================
C
       IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C
C      Parameters
       INTEGER N,IW
       PARAMETER (N=4, IW=14)
C
C      Local Arrays
       DIMENSION W(N,IW), C(N)
C
C      External subroutines
       EXTERNAL FCN
C
C      Executable statements
C
16     FORMAT (5 (1X,G10.3) )
C
C
       COMMON /QINMODOL/ F, S1
C
C      Initial settings and constraints (As stated in Chapter 4)
C      Biomass
       C(1) = 1.
C      Substrate
       C(2) = 5.
C      Product
       C(3) = 0.
C      Volume
       C(4) = 250000.
C
       S1   = 500.
       STEP = 0.05
       T    = 140.
       V    = 250000.
       Vf   = 500000.
C
       M     = 75
       YNORM = 0
       TOL   = 1.0e-4
       STIFF = 1.0e0
       IR    = 0
       XEND  = 0.
       IFAIL = 1
C
       OPEN( UNIT = 11, FILE = 'A.TXT', STATUS = 'UNKNOWN' )
C
       DO 100 COUNT = 0, T / STEP
C
         X = XEND
         XEND = X + STEP
C
C        Setting the inlet flowrate
C
         F = (Vf - V) / T
C
```



```
C        Storing the results to file
C
         WRITE (11,16) C(1), C(2), C(3), C(4)
C
C        Integration of the fermentation model
C
         CALL D02BDF( X, XEND, N, C, TOL, IR,
     &   FCN, STIFF, YNORM, W, IW, M, OUT, IFAIL)
C
100    CONTINUE
C
       CLOSE(11)
C
       STOP
       END
C
C
C
       SUBROUTINE FCN( T, Y, Z)
       IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C      Parameters
       PARAMETER (N=4)
       DOUBLE PRECISION MUm, K1, THETAm, Kp, Ki, K, Yx
     &                  ,Yp, Mx, Vo, Vf
C
C      Array arguments
       DIMENSION Z(N), Y(N)
C
       COMMON /QINMODOL/ F, S1
C
C      Parameter values
       MUm = 0.11
       K1 = 0.006
       THETAm = 0.004
       Kp = 0.0001
       Ki = 0.1
       K = 0.01
       Yx = 0.47
       Yp = 1.2
       Mx = 0.029
C
C      Specific growth rate
       XMU   = MUm * Y(2) / (Y(2) + K1 * Y(1))
C      Specific substrate utilisation rate
       SIGMA = XMU / Yx
C      Specific product formation rate
       Qp    = THETAm / (1. + Kp / Y(2) + Y(2) / Ki)
C
C
C      Executable statements
       Z(1) = XMU * Y(1) - F / Y(4) * Y(1)
       Z(2) = F / Y(4) * ( S1 - Y(2) ) - Qp / Yp * Y(1)
     &         - SIGMA * Y(1) - Mx * Y(1)
       Z(3) = Qp * Y(1) - K * Y(3) - F / Y(4) * Y(3)
       Z(4) = F
C
       RETURN
       END
```

Appendix O

Closed Loop Fuzzy Control of a Lysine Fermentation

```
C=====================================================================
C     THIS FORTRAN PROGRAM INTEGRATES THE FOUR DIFFERENTIAL EQUATIONS
C     FOR L-LYSINE PRODUCTION FROM THE PAPER OF OHNO ET AL (1976).
C     QIN STYLE FUZZY CONTROL USED.
C=====================================================================
C
      IMPLICIT DOUBLE PRECISION(A-H,Q-Z)
C
C     Parameters
      INTEGER N,IW
      PARAMETER (N=4, IW=14)
C
C     Local Arrays
      DIMENSION W(N,IW), C(N)
C
C     External subroutines
      EXTERNAL FCN
C
C     Executable statements
C
15    FORMAT (4 (1X,F10.3) )
16    FORMAT (5 (1X,G10.3) )
C
C     Number of rules that are used
      PARAMETER ( L = 4 ,
C     Number of output sampling points
     &            K = 3 )
C
C
C     Vectors
C
      DIMENSION CERROR(L)
      DIMENSION ERROR(L)
      DIMENSION SUBST(L)
      DIMENSION XMUCE(2)
      DIMENSION XMUE(2)
      DIMENSION Y(K)
      DIMENSION YMUS(3)
      DIMENSION ZmuRULE(L)
C
      COMMON /QINLYS/ F, S1
C
C     Initial settings and constraints (as summarised in Chapter 5)
C     Biomass
      C(1) = 0.02
C     Substrate
      C(2) = 2.8
C     Product
      C(3) = 0.
C     Volume
      C(4) = 5.
C
      ERR  = 0.
      F    = 0.
      S1   = 2.8
      STEP = .2
      T    = 35.
      V    = 5.
      Vf   = 20.
C
```



```
      M     = 75
      YNORM = 0
      TOL   = 1.0e-6
      STIFF = 1.0e0
      IR    = 0
      XEND  = 0.
      IFAIL = 1
C
C
C     Controller settings
      XK   = 3.
      XTAU = .1
      XL   = 1.
C
      OPEN( UNIT = 11, FILE = 'A.TXT', STATUS = 'UNKNOWN' )
      OPEN( UNIT = 13, FILE = 'C.TXT', STATUS = 'UNKNOWN' )
C
      DO 100 COUNT = 0, T / STEP
C
        X = XEND
        XEND = X + STEP
C
C
C
C       Determination of setpoint and GR
C
        IF (X .GE. 0. .AND. X .LE. 8.9) THEN
          SAIM = -.01581 * X ** 2. - .03938 * X + 2.798051
          GR = 0.4 * SAIM
        ELSE IF (X .GE. 8.95 .AND. X .LE. 27.1) THEN
          SAIM = .001940 * X ** 2. - .10433 * X + 1.938315
          GR = .4 * SAIM
        ELSE
          SAIM = .008235 * X ** 2. - .56845 * X + 9.860149
        END IF
C
C       Calculation of remaining fuzzy parameters
C
        GU = 3. * XK / GR
        GE = GR / XTAU * STEP
C
C
C
C=================================================
C       START OF CONTROLLER
C=================================================
C
        ERROROLD = ERR
        XE = (SAIM - C(2)) * GE
        ERR = XE / GE
        XCE = (ERR - ERROROLD) * GR
        CERR = XCE / GR
C
C
        XNUMERATOR = 0.
        DENOMINATOR = 0.
C
C       Fuzzification of the input
C
        CALL ERRORMU( XE, XMUE, XL)
```

```
      CALL CERRORMU( XCE, XMUCE, XL)
C
C     According to the fuzzy rule base:
      ERROR(1) = XMUE(1)
      ERROR(2) = XMUE(1)
      ERROR(3) = XMUE(2)
      ERROR(4) = XMUE(2)
      CERROR(1) = XMUCE(1)
      CERROR(2) = XMUCE(2)
      CERROR(3) = XMUCE(1)
      CERROR(4) = XMUCE(2)
C
C     Values of Y at which fuzzy answers are to be determined
C
      SAMPLE = -XL
      DO 70 S = 1, K
        Y(S) = SAMPLE
        SAMPLE = SAMPLE + 2. * XL / ( K - 1.)
70    CONTINUE
C
C     Calculation of the membership values over the fuzzy partition
C
      DO 10 J = 1, K
C
        XMAXIMUM = 0.
C
C       Fuzzification of the output
C
        CALL SUBSTMU( Y(J), YMUS, XL )
C
C       According to the fuzzy rule base
        SUBST(1) = YMUS(1)
        SUBST(2) = YMUS(2)
        SUBST(3) = YMUS(2)
        SUBST(4) = YMUS(3)
C
C
        DO 20 I = 1, L
C
C         Clipping of the output fuzzy sets
C
          ZmuRULE(I) = MIN( ERROR(I), CERROR(I), SUBST(I) )
C
C         Forming the union between the sets
C
          XMAXIMUM = MAX( XMAXIMUM, ZmuRULE(I) )
C
C
20        CONTINUE
C
C
        XNUMERATOR = XNUMERATOR + Y(J) * XMAXIMUM
        DENOMINATOR = DENOMINATOR + XMAXIMUM
C
10    CONTINUE
C
C     Defuzzification of the output
C
      IF (DENOMINATOR .EQ. 0. ) THEN
        XNR2 = 0.
```



```
      ELSE
       XNR2 = XNUMERATOR / DENOMINATOR
      END IF
C
C=================================================
C     END OF CONTROLLER
C=================================================
C
C     Calculation of new input
C
      F = F + XNR2 * GU
C
      IF (F .LE. 0.) THEN
        F = 0.
      ELSE
        F = F
      END IF
C
C
C     Disturbance
      IF (X .GE. 0. .AND. X .LE. 5.) THEN
        S1 = 2.24
      ELSE
        S1 = 2.8
      END IF
C
      WRITE (11,16) C(1), C(2), C(3), C(4)
      WRITE (13,15) X, C(2), SAIM, F
C
C     Integration of the fermentation model
C
      CALL D02BDF( X, XEND, N, C, TOL, IR,
     &  FCN, STIFF, YNORM, W, IW, M, OUT, IFAIL)
C
C
100   CONTINUE
C
      CLOSE(11)
      CLOSE(13)
C
      STOP
      END
C
C
C
      SUBROUTINE FCN( T, Y, Z)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C     Parameters
      PARAMETER (N=4)
C
C     Array arguments
      DIMENSION Z(N), Y(N)
C
      COMMON /QINLYS/ F, S1
C
C     Parameter values
      Ys =  0.135
C
C     Specific growth rate
      XMU   = .125 * Y(2)
```

```
C
C       Specific product formation rate
        IF (XMU .LE. 0. .OR. XMU .GE. .34896) THEN
          Qp = 0.
        ELSE
          Qp    = -384. * XMU ** 2. + 134. * XMU
        END IF
C
C       Specific substrate utilisation rate
        SIGMA = XMU / Ys
C
C
C
C       Executable statements
        Z(1) = XMU * Y(1) - F / Y(4) * Y(1)
        Z(2) = F / Y(4) * ( S1 - Y(2) ) - SIGMA * Y(1)
        Z(3) = Qp * Y(1) - F * Y(3) / Y(4)
        Z(4) = F
C
        RETURN
        END
C
C
C=======================================================
C       Subroutines for calculating the membership grades
C       over each fuzzy partition.
C=======================================================
C
C       Determining grade of membership for triangular
C       shaped fuzzy sets
C
C       From a to b
        SUBROUTINE UP(A, B, X, XMU)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
        XMU = (X - A) / (B - A)
        RETURN
        END
C
C       From b to c
        SUBROUTINE DOWN(B, C, X, XMU)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
        XMU = (C - X) / (C - B)
        RETURN
        END
C
C
C
C       Inputs
C
C       Error
C
        SUBROUTINE ERRORMU( X, XMUE, XL)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
        DIMENSION XMUE(1)
C
C       Constants
C
```



```
C       Error (E)    [Error in output defined as y(set) - y(actual)]
C
C       Negative (EN)
        bEN = -XL
        cEN = XL
C       Positive (EP)
        aEP = -XL
        bEP = XL
C
C       Negative
        IF (X .LE. bEN) THEN
         XmuEN = 1.
        ELSE IF (X .GE. bEN .AND. X .LE. cEN) THEN
         CALL DOWN( bEN, cEN, X, XMU )
         XmuEN = XMU
        ELSE
         XmuEN = 0.
        END IF
        XMUE(1) = XmuEN
C
C       Positive
        IF (X .GE. bEP) THEN
         XmuEP = 1.
        ELSE IF (X .GE. aEP .AND. X .LE. bEP) THEN
         CALL UP( aEP, bEP, X, XMU)
         XmuEP = XMU
        ELSE
         XmuEP = 0.
        END IF
        XMUE(2) = XmuEP
C
        RETURN
        END
C
C       Change in Error
C
        SUBROUTINE CERRORMU( X, XMUCE, XL)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
        DIMENSION XMUCE(1)
C
C       Constants
C
C       Change in Error (E)
C
C       Negative (CEN)
        bCEN = -XL
        cCEN = XL
C       Positive (EP)
        aCEP = -XL
        bCEP = XL
C
C       Negative
        IF (X .LE. bCEN) THEN
         XmuCEN = 1.
        ELSE IF (X .GE. bCEN .AND. X .LE. cCEN) THEN
         CALL DOWN( bCEN, cCEN, X, XMU )
         XmuCEN = XMU
        ELSE
         XmuCEN = 0.
```

```
      END IF
      XMUCE(1) = XmuCEN
C
C     Positive
      IF (X .GE. bCEP) THEN
       XmuCEP = 1.
      ELSE IF (X .GE. aCEP .AND. X .LE. bCEP) THEN
       CALL UP( aCEP, bCEP, X, XMU)
       XmuCEP = XMU
      ELSE
       XmuCEP = 0.
      END IF
      XMUCE(2) = XmuCEP
C
      RETURN
      END
C
C
C
C     Outputs
C
C     Change in Substrate feed concentration
C
      SUBROUTINE SUBSTMU( Y, YMUS, XL)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION YMUS(1)
C
C     Constants
C
C     Change in substrate feed (S)
C
C     Negative (SN)
      bSN = -XL
      cSN = 0.
C     Zero change (SZ)
      aSZ = -XL
      bSZ = 0.
      cSZ = XL
C     Positive (SP)
      aSP = 0.
      bSP = XL
C
C     Negative
      IF (Y .LE. bSN) THEN
       YmuSN = 1.
      ELSE IF (Y .GE. bSN .AND. Y  .LE. cSN) THEN
       CALL DOWN( bSN, cSN, Y, YMU)
       YmuSN = YMU
      ELSE
       YmuSN = 0.
      END IF
      YMUS(1) = YmuSN
C
C     Zero change
      IF (Y .GE. aSZ .AND. Y .LE. bSZ) THEN
       CALL UP( aSZ, bSZ, Y, YMU)
       YmuSZ = YMU
      ELSE IF (Y .GE. bSZ .AND. Y .LE. cSZ) THEN
       CALL DOWN( bSZ, cSZ, Y, YMU)
```



```
       YmuSZ = YMU
      ELSE
       YmuSZ = 0.
      END IF
      YMUS(2) = YmuSZ
C
C     Positive
      IF (Y .GE. aSP .AND. Y .LE. bSP) THEN
       CALL UP( aSP, bSP, Y, YMU)
       YmuSP = YMU
      ELSE IF (Y .GE. bSP ) THEN
       YmuSP = 1.
      ELSE
       YmuSP = 0.
      END IF
      YMUS(3) = YmuSP
C
      RETURN
      END
```

Appendix P

Closed Loop PI Control of a Lysine Fermentation

```
C=====================================================================
C    THIS FORTRAN PROGRAM INTEGRATES THE FOUR DIFFERENTIAL EQUATIONS
C    FOR L-LYSINE PRODUCTION FROM THE PAPER OF OHNO ET AL (1976).
C    PI CONTROLLER USED.
C=====================================================================
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C
      COMMON /CTRPRM/ USS, XTAU, XK, SUM
C
C     Parameters
      INTEGER N,IW
C
      PARAMETER (N=4, IW=14)
C
C     Local Arrays
      DIMENSION W(N,IW), C(N)
C
C     External subroutines
      EXTERNAL FCN
C
C     Executable statements
C
15    FORMAT (4 (1X,F10.3) )
16    FORMAT (5 (1X,G10.3) )
C
      COMMON /LYSPI/ F, S1
C
C     Initial settings and constraints (as summarised in Chapter 5)
C     Biomass
      C(1) = 0.02
C     Substrate
      C(2) = 2.8
C     Product
      C(3) = 0.
C     Volume
      C(4) = 5.
C
      F    = 0.
      S1   = 2.8
      STEP = 0.2
      SUM  = 0.
      T    = 35.
      USS  = 0.
      V    = 5.
      Vf   = 20.
C
      M     = 75
      YNORM = 0
      TOL   = 1.0e-6
      STIFF = 1.0e0
      IR    = 0
      XEND  = 0.
      IFAIL = 1
C
C
C     Controller settings
      XK = 3.
      XTAU = .1
C
```



```
      OPEN( UNIT = 11, FILE = 'A.TXT', STATUS = 'UNKNOWN' )
      OPEN( UNIT = 13, FILE = 'C.TXT', STATUS = 'UNKNOWN' )
C
      DO 100 COUNT = 0, T / STEP
C
        X = XEND
        XEND = X + STEP
C
C       Determination of setpoint
C
        IF (X .GE. 0 .AND. X .LE. 8.9) THEN
          SAIM = -.01581 * X ** 2. - .03938 * X + 2.798051
        ELSE IF (X .GE. 8.95 .AND. X .LE. 27.05) THEN
          SAIM = .001941 * X ** 2. - .10433 * X + 1.938315
        ELSE
          SAIM = .008235 * X ** 2. - .56845 * X + 9.860149
        END IF
C
C
C
C       Calling the controller
C
        CALL CTRLR (SAIM, C(2), U, STEP)
C
        IF (U .LE. 0.) THEN
          F = 0.
        ELSE
          F = U
        END IF
C
C
C       Disturbance
        IF (X .GE. 0. .AND. X .LE. 5.) THEN
          S1 = 2.24
        ELSE
          S1 = 2.8
        END IF
C
C
        WRITE (11,16) C(1), C(2), C(3), C(4)
        WRITE (13,15) X, C(2), SAIM, F
C
C       Integration of the fermentation model
C
        CALL D02BDF( X, XEND, N, C, TOL, IR,
     &    FCN, STIFF, YNORM, W, IW, M, OUT, IFAIL)
C
C
100   CONTINUE
C
      CLOSE(11)
      CLOSE(13)
C
      STOP
      END
C
C
C
      SUBROUTINE FCN( T, Y, Z)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
```

```
C       Parameters
        PARAMETER (N=4)
C
C       Array arguments
        DIMENSION Z(N), Y(N)
C
        COMMON /LYSPI/ F, S1
C
C       Parameter values
        Ys = 0.135
C
C       Specific growth rate
        XMU = .125 * Y(2)
C
C       Specific product formation rate
        IF (XMU .LE. 0. .OR. XMU .GE. .34896) THEN
          Qp = 0.
        ELSE
          Qp = -384. * XMU ** 2. + 134. * XMU
        END IF
C
C       Specific substrate utilisation rate
        SIGMA = XMU / Ys
C
C
C
C       Executable statements
        Z(1) = XMU * Y(1) - F / Y(4) * Y(1)
        Z(2) = F / Y(4) * ( S1 - Y(2) ) - SIGMA * Y(1)
        Z(3) = Qp * Y(1) - F * Y(3) / Y(4)
        Z(4) = F
C
        RETURN
        END
C
C
C
      SUBROUTINE CTRLR( YSET, Y, U, TSMPL)
C=======================================
C     PI-CONTROL LAW
C     UPON INPUT:
C     YSET   = SETPOINT
C     Y      = CURRENT MEASURED VALUE
C     UPON OUTPUT
C     U      = CONTROL OUTPUT
C=======================================
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON /CTRPRM/ USS, XTAU, XK, SUM
C
C
      ERROR = (YSET - Y)
      SUM = SUM + (ERROR) * TSMPL
      U = USS + XK * ERROR + XK / XTAU * SUM
      IF (U .LE. 0.) THEN
        SUM = SUM - ERROR * TSMPL
      ELSE
        SUM = SUM
      END IF
C
      RETURN
```



```
      END
```

Appendix Q

Closed Loop Fuzzy Control of a Penicillin Fermentation

```
C=====================================================================
C     THIS FORTRAN PROGRAM INTEGRATES THE FOUR DIFFERENTIAL EQUATIONS
C     FOR PENICILLIN PRODUCTION FROM THE PAPER OF SAN AND
C     STEPHANOPOULOS (1989). FUZZY CONTROL USED.
C=====================================================================
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C
C     Parameters
      INTEGER N,IW
      PARAMETER (N=4, IW=14)
C
C     Local Arrays
      DIMENSION W(N,IW), C(N)
C
C     External subroutines
      EXTERNAL FCN
C
C     Executable statements
C
15    FORMAT (4 (1X,F10.3) )
16    FORMAT (5 (1X,G10.3) )
C
C     Number of rules that are used
      PARAMETER ( L = 4 ,
C     Number of output sampling points
     &            K = 3 )
C
C
C     Vectors
C
      DIMENSION CERROR(L)
      DIMENSION ERROR(L)
      DIMENSION SUBST(L)
      DIMENSION XMUCE(2)
      DIMENSION XMUE(2)
      DIMENSION Y(K)
      DIMENSION YMUS(3)
      DIMENSION ZmuRULE(L)
C
      COMMON /QINMODEL/ F, S1
C
C     Initial settings and constraints (as summarised in Chapter 5)
C     Biomass
      C(1) = 1.
C     Substrate
      C(2) = 1.1
C     Product
      C(3) = 0.
C     Volume
      C(4) = 250000.
C
      ERR  = 0.
      S1   = 100.
      STEP = 0.1
      T    = 140.
      V    = 250000.
      Vf   = 500000.
      F    = 662.4
C
```



```
      M     = 75
      YNORM = 0
      TOL   = 1.0e-4
      STIFF = 1.0e0
      IR    = 0
      XEND  = 0.
      IFAIL = 1
C
C
C     Controller settings
      XK   = 4500.
      XTAU = .1
      XL   = 1.
      B    = 0.04
C
      OPEN( UNIT = 11, FILE = 'A.TXT', STATUS = 'UNKNOWN' )
      OPEN( UNIT = 13, FILE = 'C.TXT', STATUS = 'UNKNOWN' )
C
      DO 100 COUNT = 0, T / STEP
C
        X = XEND
        XEND = X + STEP
C
C       Determination of setpoint and GR
C
        IF (X .GE. 0 .AND. X .LE. 20.) THEN
          SAIM = 0.12225 * X ** 2. + 1.1
          GR = SAIM * B
        ELSE IF (X .GE. 20. .AND. X .LE. 35.) THEN
          SAIM = 50.
          GR = SAIM * B
        ELSE IF (X .GE. 35. .AND. X .LE. 40.) THEN
          SAIM = 50. * (40. - X) / 5.
          GR = SAIM * B
        ELSE
          SAIM = 0.001
          GR = SAIM * B
        END IF
C
C       Calculation of remaining fuzzy parameters
C
        GU = 3. * XK / GR
        GE = GR / XTAU * STEP
C
C
C
C=================================================
C       START OF CONTROLLER
C=================================================
C
        ERROROLD = ERR
        XE = (SAIM - C(2)) * GE
        ERR = XE / GE
        XCE = (ERR - ERROROLD) * GR
        CERR = XCE / GR
C
C
        XNUMERATOR = 0.
        DENOMINATOR = 0.
C
```

```
C       Fuzzification of the input
C
        CALL ERRORMU( XE, XMUE, XL)
        CALL CERRORMU( XCE, XMUCE, XL)
C
C       According to the fuzzy rule base:
        ERROR(1) = XMUE(1)
        ERROR(2) = XMUE(1)
        ERROR(3) = XMUE(2)
        ERROR(4) = XMUE(2)
        CERROR(1) = XMUCE(1)
        CERROR(2) = XMUCE(2)
        CERROR(3) = XMUCE(1)
        CERROR(4) = XMUCE(2)
C
C       Values of Y at which fuzzy answers are to be determined
C
        SAMPLE = -XL
        DO 70 S = 1, K
          Y(S) = SAMPLE
          SAMPLE = SAMPLE + 2. * XL / ( K - 1.)
70      CONTINUE
C
C       Calculation of the membership values over the fuzzy partition
C
        DO 10 J = 1, K
C
          XMAXIMUM = 0.
C
C         Fuzzification of the output
C
          CALL SUBSTMU( Y(J), YMUS, XL )
C
C         According to the fuzzy rule base
          SUBST(1) = YMUS(1)
          SUBST(2) = YMUS(2)
          SUBST(3) = YMUS(2)
          SUBST(4) = YMUS(3)
C
C
          DO 20 I = 1, L
C
C           Clipping of the output fuzzy sets
C
            ZmuRULE(I) = MIN( ERROR(I), CERROR(I), SUBST(I) )
C
C           Forming the union between the sets
C
            XMAXIMUM = MAX( XMAXIMUM, ZmuRULE(I) )
C
C
20          CONTINUE
C
C
          XNUMERATOR = XNUMERATOR + Y(J) * XMAXIMUM
          DENOMINATOR = DENOMINATOR + XMAXIMUM
C
10      CONTINUE
C
C       Defuzzification of the output
```



```
C
        IF (DENOMINATOR .EQ. 0. ) THEN
          XNR2 = 0.
        ELSE
         XNR2 = XNUMERATOR / DENOMINATOR
        END IF
C
C=================================================
C       END OF CONTROLLER
C=================================================
C
C       Calculation of new input
C
        F = F + XNR2 * GU
C
        IF (F .LE. 0.) THEN
          F = 0.
        ELSE
          F = F
        END IF
C
C
C       Disturbance
        IF (X .GE. 0. .AND. X .LE. 10.) THEN
          S1 = 35.
        ELSE
          S1 = 100.
        END IF
C
C
        WRITE (11,16) C(1), C(2), C(3), C(4)
        WRITE (13,15) X, C(2), SAIM, F
C
C       Integration of the fermentation model
C
        CALL D02BDF( X, XEND, N, C, TOL, IR,
     &  FCN, STIFF, YNORM, W, IW, M, OUT, IFAIL)
C
C
100     CONTINUE
C
        CLOSE(11)
        CLOSE(13)
C
        STOP
        END
C
C
C
        SUBROUTINE FCN( T, Y, Z)
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C       Parameters
        PARAMETER (N=4)
        DOUBLE PRECISION MUm, K1, THETAm, Kp, Ki, K, Yx
     &                  ,Yp, Mx, Vo, Vf
C
C       Array arguments
        DIMENSION Z(N), Y(N)
C
        COMMON /QINMODEL/ F, S1
```

```
C
C       Parameter values
        MUm = 0.11
        K1 = 0.006
        THETAm = 0.004
        Kp = 0.0001
        Ki = 0.1
        K = 0.01
        Yx = 0.47
        Yp = 1.2
        Mx = 0.029
C
C       Specific growth rate
        XMU   = MUm * Y(2) / (Y(2) + K1 * Y(1))
C
C       Specific substrate utilisation rate
        SIGMA = XMU / Yx
C
C       Specific product formation rate
        Qp    = THETAm / (1. + Kp / Y(2) + Y(2) / Ki)
C
C
C       Executable statements
        Z(1) = XMU * Y(1) - F / Y(4) * Y(1)
        Z(2) = F / Y(4) * ( S1 - Y(2) ) - Qp / Yp * Y(1)
     &         - SIGMA * Y(1) - Mx * Y(1)
        Z(3) = Qp * Y(1) - K * Y(3) - F / Y(4) * Y(3)
        Z(4) = F
C
        RETURN
        END
C
C
C=======================================================
C       Subroutines for calculating the membership grades
C       over each fuzzy partition.
C=======================================================
C
C       Determining grade of membership for triangular
C       shaped fuzzy sets
C
C       From a to b
        SUBROUTINE UP(A, B, X, XMU)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
        XMU = (X - A) / (B - A)
        RETURN
        END
C
C       From b to c
        SUBROUTINE DOWN(B, C, X, XMU)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
        XMU = (C - X) / (C - B)
        RETURN
        END
C
C
C
C       Inputs
```



```
C
C       Error
C
        SUBROUTINE ERRORMU( X, XMUE, XL)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
        DIMENSION XMUE(1)
C
C       Constants
C
C       Error (E)    [Error in output defined as y(set) - y(actual)]
C
C       Negative (EN)
        bEN = -XL
        cEN = XL
C       Positive (EP)
        aEP = -XL
        bEP = XL
C
C       Negative
        IF (X .LE. bEN) THEN
         XmuEN = 1.
        ELSE IF (X .GE. bEN .AND. X .LE. cEN) THEN
         CALL DOWN( bEN, cEN, X, XMU )
         XmuEN = XMU
        ELSE
         XmuEN = 0.
        END IF
        XMUE(1) = XmuEN
C
C       Positive
        IF (X .GE. bEP) THEN
         XmuEP = 1.
        ELSE IF (X .GE. aEP .AND. X .LE. bEP) THEN
         CALL UP( aEP, bEP, X, XMU)
         XmuEP = XMU
        ELSE
         XmuEP = 0.
        END IF
        XMUE(2) = XmuEP
C
        RETURN
        END
C
C       Change in Error
C
        SUBROUTINE CERRORMU( X, XMUCE, XL)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
        DIMENSION XMUCE(1)
C
C       Constants
C
C       Change in Error (E)
C
C       Negative (CEN)
        bCEN = -XL
        cCEN = XL
C       Positive (EP)
        aCEP = -XL
```

```
      bCEP = XL
C
C     Negative
      IF (X .LE. bCEN) THEN
       XmuCEN = 1.
      ELSE IF (X .GE. bCEN .AND. X .LE. cCEN) THEN
       CALL DOWN( bCEN, cCEN, X, XMU )
       XmuCEN = XMU
      ELSE
       XmuCEN = 0.
      END IF
      XMUCE(1) = XmuCEN
C
C     Positive
      IF (X .GE. bCEP) THEN
       XmuCEP = 1.
      ELSE IF (X .GE. aCEP .AND. X .LE. bCEP) THEN
       CALL UP( aCEP, bCEP, X, XMU)
       XmuCEP = XMU
      ELSE
       XmuCEP = 0.
      END IF
      XMUCE(2) = XmuCEP
C
      RETURN
      END
C
C
C
C     Outputs
C
C     Change in Substrate feed concentration
C
      SUBROUTINE SUBSTMU( Y, YMUS, XL)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION YMUS(1)
C
C     Constants
C
C     Change in substrate feed (S)
C
C     Negative (SN)
      bSN = -XL
      cSN = 0.
C     Zero change (SZ)
      aSZ = -XL
      bSZ = 0.
      cSZ = XL
C     Positive (SP)
      aSP = 0.
      bSP = XL
C
C     Negative
      IF (Y .LE. bSN) THEN
       YmuSN = 1.
      ELSE IF (Y .GE. bSN .AND. Y  .LE. cSN) THEN
       CALL DOWN( bSN, cSN, Y, YMU)
       YmuSN = YMU
      ELSE
```



```
       YmuSN = 0.
      END IF
      YMUS(1) = YmuSN
C
C     Zero change
      IF (Y .GE. aSZ .AND. Y .LE. bSZ) THEN
       CALL UP( aSZ, bSZ, Y, YMU)
       YmuSZ = YMU
      ELSE IF (Y .GE. bSZ .AND. Y .LE. cSZ) THEN
       CALL DOWN( bSZ, cSZ, Y, YMU)
       YmuSZ = YMU
      ELSE
       YmuSZ = 0.
      END IF
      YMUS(2) = YmuSZ
C
C     Positive
      IF (Y .GE. aSP .AND. Y .LE. bSP) THEN
       CALL UP( aSP, bSP, Y, YMU)
       YmuSP = YMU
      ELSE IF (Y .GE. bSP ) THEN
       YmuSP = 1.
      ELSE
       YmuSP = 0.
      END IF
      YMUS(3) = YmuSP
C
      RETURN
      END
```

Appendix R

Closed Loop PI Control of a Penicillin Fermentation

```
C=====================================================================
C    THIS FORTRAN PROGRAM INTEGRATES THE FOUR DIFFERENTIAL EQUATIONS
C    FOR PENICILLIN PRODUCTION FROM THE PAPER OF SAN AND
C    STEPHANOPOULOS (1989). PI CONTROLLER USED.
C=====================================================================
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C
      COMMON /CTRPRM/ USS, XTAU, XK, SUM
C
C     Parameters
      INTEGER N,IW
C
      PARAMETER (N=4, IW=14)
C
C     Local Arrays
      DIMENSION W(N,IW), C(N)
C
C     External subroutines
      EXTERNAL FCN
C
C     Executable statements
C
15    FORMAT (4 (1X,F10.3) )
16    FORMAT (5 (1X,G10.3) )
C
      COMMON /MODELPI/ F, S1
C
C     Initial settings and constraints (as summarised in Chapter 5)
C     Biomass
      C(1) = 1.
C     Substrate
      C(2) = 1.1
C     Product
      C(3) = 0.
C     Volume
      C(4) = 250000.
C
      S1   = 100.
      STEP = 0.1
      SUM  = 0.
      T    = 140.
      USS  = 662.4
      V    = 250000.
      Vf   = 500000.
      F    = 662.4
C
      M     = 75
      YNORM = 0
      TOL   = 1.0e-4
      STIFF = 1.0e0
      IR    = 0
      XEND  = 0.
      IFAIL = 1
C
C
C     Controller settings
      XK = 4500.
      XTAU = .1
C
```



```
      OPEN( UNIT = 11, FILE = 'A.TXT', STATUS = 'UNKNOWN' )
      OPEN( UNIT = 13, FILE = 'C.TXT', STATUS = 'UNKNOWN' )
C
      DO 100 COUNT = 0, T / STEP
C
        X = XEND
        XEND = X + STEP
C
C       Determination of setpoint
C
        IF (X .GE. 0 .AND. X .LE. 20.) THEN
          SAIM = 0.12225 * X ** 2. + 1.1
        ELSE IF (X .GE. 20. .AND. X .LE. 35.) THEN
          SAIM = 50.
        ELSE IF (X .GE. 35. .AND. X .LE. 40.) THEN
          SAIM = 50. * (40. - X) / 5.
        ELSE
          SAIM = 0.001
        END IF
C
C
C
C       Calling the controller
C
        CALL CTRLR (SAIM, C(2), U, STEP)
C
        IF (U .LE. 0.) THEN
          F = 0.
        ELSE
          F = U
        END IF
C
C
C       Disturbance
        IF (X .GE. 0. .AND. X .LE. 10.) THEN
          S1 = 35.
        ELSE
          S1 = 100.
        END IF
C
C
        WRITE (11,16) C(1), C(2), C(3), C(4)
        WRITE (13,15) X, C(2), SAIM, F
C
C       Integration of the fermentation model
C
        CALL D02BDF( X, XEND, N, C, TOL, IR,
     &  FCN, STIFF, YNORM, W, IW, M, OUT, IFAIL)
C
C
100   CONTINUE
C
      CLOSE(11)
      CLOSE(13)
C
      STOP
      END
C
C
C
```

```
      SUBROUTINE FCN( T, Y, Z)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C     Parameters
      PARAMETER (N=4)
      DOUBLE PRECISION MUm, K1, THETAm, Kp, Ki, K, Yx
     &                 ,Yp, Mx, Vo, Vf
C
C     Array arguments
      DIMENSION Z(N), Y(N)
C
      COMMON /MODELPI/ F, S1
C
C     Parameter values
      MUm = 0.11
      K1 = 0.006
      THETAm = 0.004
      Kp = 0.0001
      Ki = 0.1
      K = 0.01
      Yx = 0.47
      Yp = 1.2
      Mx = 0.029
C
C     Specific growth rate
      XMU = MUm * Y(2) / (Y(2) + K1 * Y(1))
C
C     Specific substrate utilisation rate
      SIGMA = XMU / Yx
C
C     Specific product formation rate
      Qp = THETAm / (1. + Kp / Y(2) + Y(2) / Ki)
C
C
C     Executable statements
      Z(1) = MUm * Y(1) - F / Y(4) * Y(1)
      Z(2) = F / Y(4) * ( S1 - Y(2) ) - Qp / Yp * Y(1)
     &       - SIGMA * Y(1) - Mx * Y(1)
      Z(3) = Qp * Y(1) - K * Y(3) - F / Y(4) * Y(3)
      Z(4) = F
C
      RETURN
      END
C
C
C
      SUBROUTINE CTRLR( YSET, Y, U, TSMPL)
C=======================================
C     PI-CONTROL LAW
C     UPON INPUT:
C     YSET   = SETPOINT
C     Y      = CURRENT MEASURED VALUE
C     UPON OUTPUT
C     U      = CONTROL OUTPUT
C=======================================
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON /CTRPRM/ USS, XTAU, XK, SUM
C
C
      ERROR = (YSET - Y)
      SUM = SUM + (ERROR) * TSMPL
```



```
      U = USS + XK * ERROR + XK / XTAU * SUM
C
      RETURN
      END
```

Appendix S

Optimisation of Fuzzy Set Locations For the Control of Lysine

```
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*     File npmain.for
*
*     OPTIMISATION OF FUZZY SET LOCATIONS, NPSOL Version 4.06-2.
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      program           npmain
      implicit          double precision (a-h,o-z)

*     ==================================================================
*     Set the declared array dimensions.
*     ldA    = the declared leading dimension of  A.
*     ldcJ   = the declared leading dimension of  cJac.
*     ldR    = the declared leading dimension of  R.
*     maxn   = maximum no. of variables allowed for.
*     maxbnd = maximum no. of variables + linear & nonlinear constrnts.
*     liwork = the length of the integer work array.
*     lwork  = the length of the double precision work array.
*     ==================================================================
      parameter         (ldA    =  10, ldcJ   =  1, ldR   =   11,
     $                   maxn   =  10, liwork =  40, lwork = 500,
     $                   maxbnd =  maxn + ldA + ldcJ)

      integer           istate(maxbnd)
      integer           iwork(liwork)

      double precision  bigbnd
      character*10      cbgbnd
      character*20      prnstr

      double precision  A(ldA,maxn)
      double precision  bl(maxbnd), bu(maxbnd)
      double precision  c(ldcJ), cJac(ldcJ,maxn), clamda(maxbnd)
      double precision  objgrd(maxn), R(ldR,maxn), xx(maxn)
      double precision  work(lwork)
      external          objfun, fncon

      parameter         (zero = 0.0, one = 1.0)

      bigbnd =  1.0d+15
      cbgbnd = '1.0d+15'

*     ==============================================================
*     Set the actual problem dimensions.
*     nn     = the number of variables.
*     nclin  = the number of general linear constraints (may be 0).
*     ncnln  = the number of nonlinear constraints (may be 0).
*     ==============================================================
      nn     = 8
      nclin  = 6
      ncnln  = 0
      nbnd   = nn + nclin + ncnln

*     --------------------------------------------------------------
*     Assign file numbers and the data arrays.
*     iPrint = the unit number for printing.
*     iOptns = the unit number for reading the options file.
*     bounds  .ge.    bigbnd  will be treated as plus  infinity.
*     bounds  .le.  - bigbnd  will be treated as minus infinity.
*     A      = the linear constraint matrix.
```



```
*     bl     = the lower bounds on  x,  a'x  and  c(x).
*     bu     = the upper bounds on  x,  a'x  and  c(x).
*     xx     = the initial estimate of the solution.
*     --------------------------------------------------------------
      iOptns = 4
      iPrint = 9
      open (unit=iOptns, file='npsol.opt', status = 'UNKNOWN' )
      open (unit=iPrint, file='npsol.out', status = 'UNKNOWN' )

*     Set the matrix  A.

      do 40, j = 1, nn
         do 30, i = 1, nclin
            A(i,j) = zero
   30    continue
   40 continue

*     Linear constraints. Ensuring that a given variable is always
*     greater than the variable to its immediated left

      A(1,1)  = -one
      A(1,2)  =  one
      A(2,2)  = -one
      A(2,3)  =  one
      A(3,3)  = -one
      A(3,4)  =  one
      A(4,5)  = -one
      A(4,6)  =  one
      A(5,6)  = -one
      A(5,7)  =  one
      A(6,7)  = -one
      A(6,8)  =  one

*     Set the bounds.

      do 50, j =  1, nbnd
         bl(j) = -bigbnd
         bu(j) =  bigbnd
   50 continue


*     Set lower bounds of zero for all 8 linear constraints.

      do 60, j =  nn+1,  nn+nclin
         bl(j) =  0.001
   60 continue

      bl(4)  = 0.0001
      bl(8)  = 0.0001
      bu(1)  = -.0001
      bu(5)  = -.0001

*     Set the initial estimate of  X.

      xx(1)  =   -.006
      xx(2)  =  -0.002
      xx(3)  =   -.0002
      xx(4)  =   0.005
      xx(5)  =   -.7
      xx(6)  =   -.2
```

```
      xx(7)   =   .05
      xx(8)   =   0.1

*     ------------------------------------------------------------------
*     Read the options file.
*     ------------------------------------------------------------------
      call npfile( iOptns, inform )
      if (inform .ne. 0) then
         write(iPrint, 3000) inform
         stop
      end if

      call npoptn( 'Infinite Bound size ='//cbgbnd )

*     ------------------------------------------------------------------
*     Solve the problem.
*     ------------------------------------------------------------------
      call npsol ( nn, nclin, ncnln, ldA, ldcJ, ldR,
     $             A, bl, bu,
     $             fncon, objfun,
     $             inform, iter, istate,
     $             c, cJac, clamda, objf, objgrd, R, xx,
     $             iwork, liwork, work, lwork )

      if (inform .gt. 0) go to 900

      stop

*     -----------
*     Error exit.
*     -----------

  900 write(iPrint, 3010) inform
      stop

 3000 format(/ ' npfile terminated with  inform =', i3)
 3010 format(/ ' npsol  terminated with  inform =', i3)

*     end of the example program for NPSOL
      end

*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*     Calling of objective function from a seperate program.

      include 'objfun.for'

*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      NOTE THAT THIS SUBROUTINE HAS NO EFFECT SINCE NO NONLINEAR
      CONSTRAINTS ARE USED
      subroutine fncon( mode, ncnln, n, ldcJ,
     $                  needc, x, c, cJac, nstate )

      implicit          double precision(a-h,o-z)
      integer           needc(*)
      double precision  x(n), c(*), cJac(ldcJ,*)

*     ==================================================================
*     fncon1  computes the values and first derivatives of the nonlinear
*     constraints.
*
```



```
*     The zero elements of Jacobian matrix are set only once.  This
*     occurs during the first call to fncon1  (nstate = 1).
*     ==================================================================
      parameter         (zero = 0.0, two = 2.0)

      if (nstate .eq. 1) then

*        First call to fncon1.  Set all Jacobian elements to zero.
*        N.B.  This will only work with `Derivative Level = 3'.

         do 120, j = 1, n
            do 110,    i = 1, ncnln
               cJac(i,j) = zero
  110       continue
  120    continue
      end if

      if (needc(1) .gt. 0) then
         c(1)       =   x(1)**2  +  x(6)**2
         cJac(1,1)  =   two*x(1)
         cJac(1,6)  =   two*x(6)
      end if

      if (needc(2) .gt. 0) then
         c(2)       =   (x(2) - x(1))**2  +  (x(7) - x(6))**2
         cJac(2,1)  = - two*(x(2) - x(1))
         cJac(2,2)  =   two*(x(2) - x(1))
         cJac(2,6)  = - two*(x(7) - x(6))
         cJac(2,7)  =   two*(x(7) - x(6))
      end if

      if (needc(3) .gt. 0) then
         c(3)       =   (x(3) - x(1))**2  +  x(6)**2
         cJac(3,1)  = - two*(x(3) - x(1))
         cJac(3,3)  =   two*(x(3) - x(1))
         cJac(3,6)  =   two*x(6)
      end if

      if (needc(4) .gt. 0) then
         c(4)       =   (x(1) - x(4))**2  +  (x(6) - x(8))**2
         cJac(4,1)  =   two*(x(1) - x(4))
         cJac(4,4)  = - two*(x(1) - x(4))
         cJac(4,6)  =   two*(x(6) - x(8))
         cJac(4,8)  = - two*(x(6) - x(8))
      end if

      if (needc(5) .gt. 0) then
         c(5)       =   (x(1) - x(5))**2  +  (x(6) - x(9))**2
         cJac(5,1)  =   two*(x(1) - x(5))
         cJac(5,5)  = - two*(x(1) - x(5))
         cJac(5,6)  =   two*(x(6) - x(9))
         cJac(5,9)  = - two*(x(6) - x(9))
      end if

      if (needc(6) .gt. 0) then
         c(6)       =   x(2)**2  +  x(7)**2
         cJac(6,2)  =   two*x(2)
         cJac(6,7)  =   two*x(7)
      end if
```

```
      if (needc(7) .gt. 0) then
         c(7)       =   (x(3) - x(2))**2  +  x(7)**2
         cJac(7,2)  = - two*(x(3) - x(2))
         cJac(7,3)  =   two*(x(3) - x(2))
         cJac(7,7)  =   two*x(7)
      end if

      if (needc(8) .gt. 0) then
         c(8)       =   (x(4) - x(2))**2  +  (x(8) - x(7))**2
         cJac(8,2)  = - two*(x(4) - x(2))
         cJac(8,4)  =   two*(x(4) - x(2))
         cJac(8,7)  = - two*(x(8) - x(7))
         cJac(8,8)  =   two*(x(8) - x(7))
      end if

      if (needc(9) .gt. 0) then
         c(9)       =   (x(2) - x(5))**2  +  (x(7) - x(9))**2
         cJac(9,2)  =   two*(x(2) - x(5))
         cJac(9,5)  = - two*(x(2) - x(5))
         cJac(9,7)  =   two*(x(7) - x(9))
         cJac(9,9)  = - two*(x(7) - x(9))
      end if

      if (needc(10) .gt. 0) then
         c(10)      =   (x(4) - x(3))**2  +  x(8)**2
         cJac(10,3) = - two*(x(4) - x(3))
         cJac(10,4) =   two*(x(4) - x(3))
         cJac(10,8) =   two*x(8)
      end if

      if (needc(11) .gt. 0) then
         c(11)      =   (x(5) - x(3))**2  +  x(9)**2
         cJac(11,3) = - two*(x(5) - x(3))
         cJac(11,5) =   two*(x(5) - x(3))
         cJac(11,9) =   two*x(9)
      end if

      if (needc(12) .gt. 0) then
         c(12)      =   x(4)**2  +  x(8)**2
         cJac(12,4) =   two*x(4)
         cJac(12,8) =   two*x(8)
      end if

      if (needc(13) .gt. 0) then
         c(13)      =   (x(4) - x(5))**2  +  (x(9) - x(8))**2
         cJac(13,4) =   two*(x(4) - x(5))
         cJac(13,5) = - two*(x(4) - x(5))
         cJac(13,8) = - two*(x(9) - x(8))
         cJac(13,9) =   two*(x(9) - x(8))
      end if

      if (needc(14) .gt. 0) then
         c(14)      =   x(5)**2  +  x(9)**2
         cJac(14,5) =   two*x(5)
         cJac(14,9) =   two*x(9)
      end if

*     end of fncon1
      end
```

```
C=====================================================================
C     THIS FORTRAN PROGRAM INTEGRATES THE FOUR DIFFERENTIAL EQUATIONS
C     FOR L-LYSINE PRODUCTION FROM THE PAPER OF OHNO ET AL (1976).
C     THIS PROGRAM IS TO BE AN INCLUSION INTO THE NPSOL PROGRAM
C     AND CALCULATES THE COST RATIO WHICH IS TO BE MINIMISED.
C=====================================================================
C
      SUBROUTINE OBJFUN(MODE, NN, XX, OBJF, OBJGRD, NSTATE)
C
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C
C     Parameters
      INTEGER N,IW
      PARAMETER (N=4, IW=14)
C
C     Local Arrays
      DIMENSION W(N,IW), C(N)
      DIMENSION XX(NN)
      DIMENSION OBJGRD(NN)
C
C     External subroutines
      EXTERNAL FCN, OUT
C
C     Executable statements
C
15    FORMAT (4 (1X,F10.3) )
C
C     Number of rules that are used
      PARAMETER ( L = 5 ,
C     Number of output sampling points
     &            K = 30)
C
C     Vectors
C
      DIMENSION XMUdFLOW(L)
      DIMENSION XMUdSGR(L)
      DIMENSION Y(K)
      DIMENSION YMU(L)
      DIMENSION ZmuRULE(L)
C
      COMMON /OBJFUN/ F, S1, SGR, Qp
C
C     Initial settings and constraints (as summarised in Chapter 5)
C     Biomass
      C(1) = 0.02
C     Substrate
      C(2) = 2.8
C     Product
      C(3) = 0.
C     Volume
      C(4) = 5.
C
      dF     = 0.
      F      = 0.
      Qp     = 0.
      QpOLD  = 0.
      S1     = 2.8
      SGR    = .125 * C(2)
      SGROLD = SGR
```



```
      STEP   = 0.2
      SUBSTOT = 0.
      T      = 35.
      V      = 5.
      Vf     = 20.
C
      M    = 75
      YNORM = 0
      TOL  = 1.0e-6
      STIFF = 1.0e0
      IR   = 0
      IFAIL = 1
      XEND  = 0.
C
C
      DO 100 COUNT = 0, T / STEP
C
        X = XEND
        XEND = X + STEP
C
        IF (C(4) .LT. 20. .AND. dQp .LT. 0.) THEN
          The Fed-batch stage
C
C=================================================
C           START OF CONTROLLER
C=================================================
C
C           Determining the change in both the specific
C           product formation rate and the specific
C           growth rate
C
            dQp  = Qp - QpOLD
            dSGR = SGR - SGROLD
C
            XNUMERATOR = 0.
            DENOMINATOR = 0.
C
C           Fuzzification of the input
C
            CALL dSGRMU( dSGR, XMUdSGR, XX)
C
C
C           Values of Y at which fuzzy answers are to be determined
C
            SAMPLE = XX(5)
            DO 70 S = 1, K
              Y(S) = SAMPLE
              SAMPLE = SAMPLE + (XX(8) - XX(5)) / (K - 1)
70          CONTINUE
C
C           Calculation of the membership values over the
C           fuzzy partition
C
            DO 10 J = 1, K
C
              XMAXIMUM = 0.
C
C             Fuzzification of the output fuzzy sets
C
              CALL dFLOWMU( Y(J), XMUdFLOW, XX)
```

```
C
              DO 20 I = 1, L
C
C               Clipping of the output fuzzy sets
C
                ZmuRULE(I) = MIN( XMUdSGR(I), XMUdFLOW(I) )
C
C               Forming the union between the sets
C
                XMAXIMUM = MAX( XMAXIMUM, ZmuRULE(I) )
C
20            CONTINUE
C
              XNUMERATOR = XNUMERATOR + Y(J) * XMAXIMUM
              DENOMINATOR = DENOMINATOR + XMAXIMUM
C
10          CONTINUE
C
C           Defuzzification of the output
C
            IF (DENOMINATOR .EQ. 0. ) THEN
              XNR2 = 0.
            ELSE
              XNR2 = XNUMERATOR / DENOMINATOR
            END IF
C
C=================================================
C           END OF CONTROLLER
C=================================================
C
            F = F + XNR2
C
C
            IF (F .LE. 0.) THEN
              F = 0.
            ELSE
              F = F
            END IF
C
          ELSE
C           The initial batch region or the final batch region
            F = 0.
            dQp = Qp - QpOLD
            dSGR = SGR - SGROLD
          END IF
C
C
          QpOLD = Qp
          SGROLD = SGR
C
C         Determining the mass of substrate entering through
C         the feed stream
C
          SUBSTOT = SUBSTOT + F * 2.8 * STEP
C
C         Integration of the fermentation model
C
          CALL D02BDF( X, XEND, N, C, TOL, IR,
     &    FCN, STIFF, YNORM, W, IW, M, OUT, IFAIL)
C
```



```
C
100       CONTINUE
C
C
C         Evaluation of cost ratio
C         Total amount of substrate used (Correlated to M&L)
          SUBSTOT = SUBSTOT + 2.8 * (20.167 - C(4) + 5.)
C         Final mass of product in fermenter
          PRODMASS = C(3) * C(4)
C
          COST = SUBSTOT / PRODMASS
C
          RETURN
C
          STOP
          END
C
C
C
          SUBROUTINE FCN( T, Y, Z)
          IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C         Parameters
          PARAMETER (N=4)
C
C         Array arguments
          DIMENSION Z(N), Y(N)
C
          COMMON /OBJFUN/ F, S1, SGR, Qp
C
C         Parameter values
          Ys      = 0.135
C
C         Specific growth rate
          SGR     = .125 * Y(2)
C
C         Specific product formation rate
          IF (SGR .LE. 0. .OR. SGR .GE. .34896) THEN
            Qp    = 0.
          ELSE
            Qp    = -384. * SGR ** 2. + 134. * SGR
          END IF
C
C         Specific substrate utilisation rate
          SIGMA = SGR / Ys
C
C         Executable statements
          Z(1)    = SGR * Y(1) - F / Y(4) * Y(1)
          Z(2)    = F / Y(4) * ( S1 - Y(2) ) - SIGMA * Y(1)
          Z(3)    = Qp * Y(1) - F * Y(3) / Y(4)
          Z(4)    = F
C
          RETURN
          END
C
C
C=======================================================
C         Subroutines for calculating the membership grades
C         over each fuzzy partition.
C=======================================================
C
```

```
C       Determining grade of membership for triangular
C       shaped fuzzy sets
C
C       From a to b
        SUBROUTINE UP(A, B, X, XMU)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        XMU = (X - A) / (B - A)
        RETURN
        END
C
C       From b to c
        SUBROUTINE DOWN(B, C, X, XMU)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        XMU = (C - X) / (C - B)
        RETURN
        END
C
C
C
C       Inputs
C
C       Change in specific growth rate
C
        SUBROUTINE dSGRMU( X, XMUdSGR, XX)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
        DIMENSION XMUdSGR(1)
        DIMENSION XX(1)
C
        GR1 = XX(1)
        GR2 = XX(2)
        GR3 = XX(3)
        GR4 = 0.
        GR5 = XX(4)
C
C       SET 1
        IF (X .LE. GR1) THEN
          XMU = 1.
        ELSE IF (X .GE. GR1 .AND. X .LE. GR2) THEN
          CALL DOWN( GR1, GR2, X, XMU)
          XMU = XMU
        ELSE
          XMU = 0.
        END IF
        XMUdSGR(1) = XMU
C
C       SET 2
        IF (X .GE. GR1 .AND. X .LE. GR2) THEN
          CALL UP( GR1, GR2, X, XMU)
          XMU = XMU
        ELSE IF (X .GE. GR2 .AND. X .LE. GR3) THEN
          CALL DOWN( GR2, GR3, X, XMU)
          XMU = XMU
        ELSE
          XMU = 0.
        END IF
        XMUdSGR(2) = XMU
C
C       SET 3
        IF (X .GE. GR2 .AND. X .LE. GR3) THEN
```



```
          CALL UP( GR2, GR3, X, XMU)
          XMU = XMU
        ELSE IF (X .GE. GR3 .AND. X .LE. GR4) THEN
          CALL DOWN( GR3, GR4, X, XMU)
          XMU = XMU
        ELSE
          XMU = 0.
        END IF
        XMUdSGR(3) = XMU
C
C       SET 4
        IF (X .GE. GR3 .AND. X .LE. GR4) THEN
          CALL UP( GR3, GR4, X, XMU)
          XMU = XMU
        ELSE IF (X .GE. GR4 .AND. X .LE. GR5) THEN
          CALL DOWN( GR4, GR5, X, XMU)
          XMU = XMU
        ELSE
          XMU = 0.
        END IF
        XMUdSGR(4) = XMU
C
C       SET 5
        IF (X .GE. GR4 .AND. X .LE. GR5) THEN
          CALL UP( GR4, GR5, X, XMU)
          XMU = XMU
        ELSE IF (X .GE. GR5) THEN
          XMU = 1.
        ELSE
          XMU = 0.
        END IF
        XMUdSGR(5) = XMU
C
        RETURN
        END
C
C
C
C       Outputs
C
C       Change in flow rate
C
        SUBROUTINE dFLOWMU( X, XMUdFLOW, XX)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
        DIMENSION XMUdFLOW(1)
        DIMENSION XX(1)
C
        F1 = XX(5)
        F2 = XX(6)
        F3 = 0.
        F4 = XX(7)
        F5 = XX(8)
C
C       SET 5
        IF (X .GE. F5) THEN
          XMU = 1.
        ELSE IF (X .GE. F4 .AND. X .LE. F5) THEN
          CALL UP( F4, F5, X, XMU)
          XMU = XMU
```

```
      ELSE
        XMU = 0.
      END IF
      XMUdFLOW(1) = XMU
C
C     SET 4
      IF (X .GE. F4 .AND. X .LE. F5) THEN
        CALL DOWN( F4, F5, X, XMU)
        XMU = XMU
      ELSE IF (X .GE. F3 .AND. X .LE. F4) THEN
        CALL UP( F3, F4, X, XMU)
        XMU = XMU
      ELSE
        XMU = 0.
      END IF
      XMUdFLOW(2) = XMU
C
C     SET 3
      IF (X .GE. F3 .AND. X .LE. F4) THEN
        CALL DOWN( F3, F4, X, XMU)
        XMU = XMU
      ELSE IF (X .GE. F2 .AND. X .LE. F3) THEN
        CALL UP( F2, F3, X, XMU)
        XMU = XMU
      ELSE
        XMU = 0.
      END IF
      XMUdFLOW(3) = XMU
C
C     SET 2
      IF (X .GE. F2 .AND. X .LE. F3) THEN
        CALL DOWN( F2, F3, X, XMU)
        XMU = XMU
      ELSE IF (X .GE. F1 .AND. X .LE. F2) THEN
        CALL UP( F1, F2, X, XMU)
        XMU = XMU
      ELSE
        XMU = 0.
      END IF
      XMUdFLOW(4) = XMU
C
C     SET 1
      IF (X .GE. F1 .AND. X .LE. F2) THEN
        CALL DOWN( F1, F2, X, XMU)
        XMU = XMU
      ELSE IF (X .LE. F1) THEN
        XMU = 1.
      ELSE
        XMU = 0.
      END IF
      XMUdFLOW(5) = XMU
C
      RETURN
      END
```

Appendix T

Determination of Random Numbers

```
! Creating Random Numbers

! Determining how many random numbers are required
INPUT PROMPT "Number of Random Numbers Required?  : ": i

! Opening the file to which the results should be printed
OPEN #1: name"f:\random.dat", create new

RANDOMIZE

! Printing random numbers to output file
FOR j = 1 to i
   PRINT #1: rnd
NEXT j

CLOSE #1
END
```

Appendix U

Random Placing of Fuzzy Sets For the Control of Lysine

```
C=====================================================================
C     THIS FORTRAN PROGRAM INTEGRATES THE FOUR DIFFERENTIAL EQUATIONS
C     FOR L-LYSINE PRODUCTION FROM THE PAPER OF OHNO ET AL (1976).
C     THIS PROGRAM CALLS IN 8 RANDOMLY GENERATED VALUES TO DETERMINE
C     THE POSITIONING OF THE FIVE INPUT AND FIVE OUTPUT FUZZY SETS.
C     THE PROFIT RATIO IS THEN CALCULATED TO DETERMINE IF THE CURRENT
C     CHOICE IS A GOOD ONE.
C=====================================================================
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C
C     Parameters
      INTEGER N,IW
      PARAMETER (N=4, IW=14)
C
C     Local Arrays
      DIMENSION W(N,IW), C(N)
C
C     External subroutines
      EXTERNAL FCN
C
C     Executable statements
C
15    FORMAT (4 (1X,F10.6) )
C
C     Number of rules that are used
      PARAMETER ( L = 5 ,
C     Number of output sampling points
     &            K = 30,
C     Number of random points read in at one time
     &            LK = 8,
C     Number of simulations
     &            ITTER = 1000)
C
C
C     Vectors
C
      DIMENSION RINPUT(LK)
      DIMENSION XMUdFLOW(L)
      DIMENSION XMUdSGR(L)
      DIMENSION XX(10)
      DIMENSION Y(K)
      DIMENSION YMU(L)
      DIMENSION ZmuRULE(L)
C
      COMMON /L/ F, S1, SGR, Qp
C
C
      OPEN( UNIT = 10, FILE = !RANDOM.DAT!, STATUS = 'UNKNOWN' )
      OPEN( UNIT = 15, FILE = !E.TXT', STATUS = 'UNKNOWN' )
      OPEN( UNIT = 16, FILE = !F1.TXT', STATUS = 'UNKNOWN' )
      OPEN( UNIT = 17, FILE = !F2.TXT!, STATUS = 'UNKNOWN' )
      OPEN( UNIT = 18, FILE = !F3.TXT!, STATUS = 'UNKNOWN' )
      OPEN( UNIT = 19, FILE = !F4.TXT!, STATUS = 'UNKNOWN' )
C
      DO 2100 IT = 1, ITTER
C
C
C       Initial settings and constraints (as summarised in Chapter 5)
C       Biomass
```



```
        C(1) = 0.02
C       Substrate
        C(2) = 2.8
C       Product
        C(3) = 0.
C       Volume
        C(4) = 5.
C
        dF      = 0.
        F       = 0.
        Qp      = 0.
        QpOLD   = 0.
        S1      = 2.8
        SGR     = .125 * C(2)
        SGROLD  = SGR
        STEP    = 0.2
        SUBSTOT = 0.
        T       = 35.
        V       = 5.
        Vf      = 20.
C
        M     = 75
        YNORM = 0
        TOL   = 1.0e-6
        STIFF = 1.0e0
        IR    = 0
        IFAIL = 1
        XEND  = 0.
C
C       Maximum and minimum values for the variables
        DMAX1  = -.0001
        DMIN1  = -.007
        DMAX5  = .001
        DMIN5  = .0001
        DMAX6  = -.001
        DMIN6  = -1.
        DMAX10 = 1.
        DMIN10 = .001
C
C
C       Reading the random data
        DO 2200 IG = 1, LK
         READ (10, *, IOSTAT = I)  RINPUT(IG)
2200    CONTINUE
C
C
C       Assigning the positions to the fuzzy sets
        XX(1)  = DMAX1 - (DMAX1 - DMIN1) * RINPUT(1)
        XX(5)  = DMIN5 + (DMAX5 - DMIN5) * RINPUT(2)
        XX(4)  = 0.
        XX(3)  = XX(1) + (XX(4) - XX(1)) * RINPUT(3)
        XX(2)  = XX(1) + (XX(3) - XX(1)) * RINPUT(4)
        XX(6)  = DMAX6 - (DMAX6 - DMIN6) * RINPUT(5)
        XX(10) = DMIN10 + (DMAX10 - DMIN10) * RINPUT(6)
        XX(8)  = 0.
        XX(7)  = XX(8) - (XX(8) - XX(6)) * RINPUT(7)
        XX(9)  = XX(8) + (XX(10) - XX(8)) * RINPUT(8)
C
        DO 100 COUNT = 0, T / STEP
          X = XEND
```

```
            XEND = X + STEP
C
            IF (C(4) .LT. 20. .AND. dQp .LT. 0.) THEN
C             The fed-batch region
C=================================================
C             START OF CONTROLLER
C=================================================
C
C             Determing the change in both the specific
C             product formation rate and the specific
C             growth rate
C
              dQp  = Qp - QpOLD
              dSGR = SGR - SGROLD
C
              XNUMERATOR = 0.
              DENOMINATOR = 0.
C
C             Fuzzification of the input
C
              CALL dSGRMU( dSGR, XMUdSGR, XX)
C
C
C             Values of Y at which fuzzy answers are to be determined
C
              SAMPLE = XX(6)
              DO 70 S = 1, K
                Y(S) = SAMPLE
                SAMPLE = SAMPLE + (XX(10) - XX(6)) / (K - 1)
70            CONTINUE
C
C             Calculation of the membership values over the
C             fuzzy partition
C
              DO 10 J = 1, K
C
                XMAXIMUM = 0.
C
C               Fuzzification of the input
C
                CALL dFLOWMU( Y(J), XMUdFLOW, XX)
C
C
                DO 20 I = 1, L
C
C                 Clipping of the output fuzzy sets
C
                  ZmuRULE(I) = MIN( XMUdSGR(I), XMUdFLOW(I) )
C
C                 Forming the union between the sets
C
                  XMAXIMUM = MAX( XMAXIMUM, ZmuRULE(I) )
C
20              CONTINUE
C
                XNUMERATOR = XNUMERATOR + Y(J) * XMAXIMUM
                DENOMINATOR = DENOMINATOR + XMAXIMUM
C
10            CONTINUE
C
```



```
C             Defuzzification of the output
C
              IF (DENOMINATOR .EQ. 0. ) THEN
                XNR2 = 0.
              ELSE
                XNR2 = XNUMERATOR / DENOMINATOR
              END IF
C
C=================================================
C             END OF CONTROLLER
C=================================================
C
              F = F + XNR2
C
C
              IF (F .LE. 0.) THEN
                F = 0.
              ELSE
                F = F
              END IF
C
            ELSE
C             The initial batch region or the final batch region
              F = 0.
              dQp = Qp - QpOLD
              dSGR = SGR - SGROLD
            END IF
C
C
            QpOLD = Qp
            SGROLD = SGR
C
C           Determining the mass of substrate entering through
C           the feed stream
C
C
            SUBSTOT = SUBSTOT + F * S1 * STEP
C
C           Integration of the fermentation model
C
            CALL D02BDF( X, XEND, N, C, TOL, IR,
     &      FCN, STIFF, YNORM, W, IW, M, OUT, IFAIL)

C
C
100       CONTINUE
C
C
C         Evaluation of profit ratio
C         Total amount of substrate used (correlated to M&L)
          SUBSTOT = SUBSTOT + 2.8 * (5.)
C         Final mass of product in fermenter
          PRODMASS = C(3) * C(4)
C
          PROFIT = PRODMASS / SUBSTOT
          PERCENT = PROFIT / 12.661395 * 100.
C
C         Storing of results to file
C         Profit ratio and profit percent
          WRITE (15,15) PROFIT, PERCENT
```

```
C         Random points used
          WRITE (16,15) RINPUT(1),RINPUT(2)
          WRITE (17,15) RINPUT(3),RINPUT(4)
          WRITE (18,15) RINPUT(5),RINPUT(6)
          WRITE (19,15) RINPUT(7),RINPUT(8)
C
2100    CONTINUE
C
        CLOSE(10)
        CLOSE(15)
        CLOSE(16)
        CLOSE(17)
        CLOSE(18)
        CLOSE(19)
C
        STOP
        END
C
C
C
        SUBROUTINE FCN( T, Y, Z)
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C       Parameters
        PARAMETER (N=4)
C
C       Array arguments
        DIMENSION Z(N), Y(N)
C
        COMMON /L/ F, S1, SGR, Qp
C
C       Parameter values
        Ys      = 0.135
C
C       Specific growth rate
        SGR     = .125 * Y(2)
C
C       Specific product formation rate
        IF (SGR .LE. 0. .OR. SGR .GE. .34896) THEN
          Qp    = 0.
        ELSE
          Qp    = -384. * SGR ** 2. + 134. * SGR
        END IF
C
C       Specific substrate utilisation rate
        SIGMA = SGR / Ys
C
C       Executable statements
        Z(1)    = SGR * Y(1) - F / Y(4) * Y(1)
        Z(2)    = F / Y(4) * ( S1 - Y(2) ) - SIGMA * Y(1)
        Z(3)    = Qp * Y(1) - F * Y(3) / Y(4)
        Z(4)    = F
C
        RETURN
        END
C
C
C=======================================================
C       Subroutines for calculating the membership grades
C       over each fuzzy partition.
C=======================================================
```



```
C
C       Determining grade of membership for triangular
C       shaped fuzzy sets
C
C       From a to b
        SUBROUTINE UP(A, B, X, XMU)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        XMU = (X - A) / (B - A)
        RETURN
        END
C
C       From b to c
        SUBROUTINE DOWN(B, C, X, XMU)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        XMU = (C - X) / (C - B)
        RETURN
        END
C
C
C
C       Inputs
C
C       Change in specific growth rate
C
        SUBROUTINE dSGRMU( X, XMUdSGR, XX)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
        DIMENSION XMUdSGR(1)
        DIMENSION XX(1)
C
        GR1 = XX(1)
        GR2 = XX(2)
        GR3 = XX(3)
        GR4 = XX(4)
        GR5 = XX(5)
C
C       SET 1
        IF (X .LE. GR1) THEN
          XMU = 1.
        ELSE IF (X .GE. GR1 .AND. X .LE. GR2) THEN
          CALL DOWN( GR1, GR2, X, XMU)
          XMU = XMU
        ELSE
          XMU = 0.
        END IF
        XMUdSGR(1) = XMU
C
C       SET 2
        IF (X .GE. GR1 .AND. X .LE. GR2) THEN
          CALL UP( GR1, GR2, X, XMU)
          XMU = XMU
        ELSE IF (X .GE. GR2 .AND. X .LE. GR3) THEN
          CALL DOWN( GR2, GR3, X, XMU)
          XMU = XMU
        ELSE
          XMU = 0.
        END IF
        XMUdSGR(2) = XMU
C
C       SET 3
```

Appendix V

Variation of The Number of Fuzzy Sets For the Control of Lysine

```
C=====================================================================
C      THIS FORTRAN PROGRAM INTEGRATES THE FOUR DIFFERENTIAL EQUATIONS
C      FOR L-LYSINE PRODUCTION FROM THE PAPER OF OHNO ET AL (1976).
C      THE EFFECT OF THE NUMBER OF FUZZY SETS IS INVESTIGATED.
C=====================================================================
C
       IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C
C      Parameters
       INTEGER N,IW
       PARAMETER (N=4, IW=14)
C
C      Local Arrays
       DIMENSION W(N,IW), C(N)
C
C      External subroutines
       EXTERNAL FCN
C
C      Executable statements
C
15     FORMAT (4 (1X,F10.6) )
C
C      Max number of rules that could be used
       PARAMETER ( L = 400,
C      Number of output sampling points
     &       K = 30,
C      Number of random points read in
     &       LK = 4)
C
C      Vectors
C
       DIMENSION FR(L)
       DIMENSION GR(L)
       DIMENSION RINPUT(LK)
       DIMENSION XMUdFLOW(L)
       DIMENSION XMUdSGR(L)
       DIMENSION XX(LK)
       DIMENSION Y(K)
       DIMENSION YMU(L)
       DIMENSION ZmuRULE(L)
C
       COMMON /LSETS/ F, S1, YI, SGR, Qp
C
C
       OPEN( UNIT = 15, FILE = 'E.TXT', STATUS = 'UNKNOWN' )
C
C      A simulation is done for every value of the specific
C      growth rate's coefficient, YI, used
C
       DO 4000 YI = .1, .15, .0025
C
C        Deciding how many input and output fuzzy sets are
C        to be used
C
         NSETS = 350
C
         OPEN( UNIT = 9 , FILE = 'D3.TXT', STATUS = 'UNKNOWN' )
C
C        Initial values and constraints (as summarised in Chapter 5)
C        Biomass
```



```
         C(1) = 0.02
C        Substrate
         C(2) = 2.8
C        Product
         C(3) = 0.
C        Volume
         C(4) = 5.
C
         dF      = 0.
         F       = 0.
         Qp      = 0.
         QpOLD   = 0.
         S1      = 2.8
         SGR     = .125 * C(2)
         SGROLD  = SGR
         STEP    = 0.2
         SUBSTOT = 0.
         T       = 35.
         V       = 5.
         Vf      = 20.
C
         M = 75
         YNORM = 0
         TOL = 1.0e-6
         STIFF = 1.0e0
         IR = 0
         IFAIL = 1
         XEND = 0.
C
C        Maximum and minimum values for the variables
         DMAX1  = -.0001
         DMIN1  = -.007
         DMAX2  = .001
         DMIN2  = .0001
         DMAX3  = -.001
         DMIN3  = -1.
         DMAX4 = 1.
         DMIN4 = .001
C
C        Reading the data
         DO 2200 IG = 1, LK
           READ (9, *, IOSTAT = I)  RINPUT(IG)
2200     CONTINUE
C
C        Assigning the boundary positions to the fuzzy sets
         XX(1)  = DMAX1 - (DMAX1 - DMIN1) * RINPUT(1)
         XX(2)  = DMIN2 + (DMAX2 - DMIN2) * RINPUT(2)
         XX(3)  = DMAX3 - (DMAX3 - DMIN3) * RINPUT(3)
         XX(4)  = DMIN4 + (DMAX4 - DMIN4) * RINPUT(4)
C
C
         DO 100 COUNT = 0, T / STEP
C
           X = XEND
           XEND = X + STEP
C
           IF (C(4) .LT. 20. .AND. dQp .LT. 0.) THEN
C            The fed-batch region
C=================================================
C              START OF CONTROLLER
```

```
C=================================================
C
C             Determining the change in both the specific
C             product formation rate and the specific
C             growth rate
C
              dQp  = Qp - QpOLD
              dSGR = SGR - SGROLD
C
              XNUMERATOR = 0.
              DENOMINATOR = 0.
C
C             Fuzzification of the input
C
              CALL dSGRMU( dSGR, XMUdSGR, XX, GR, NSETS)
C
C
C             Values of Y at which fuzzy answers are to be determined
C
              SAMPLE = XX(3)
              DO 70 S = 1, K
                Y(S) = SAMPLE
                SAMPLE = SAMPLE + (XX(4) - XX(3)) / (K - 1)
70            CONTINUE
C
C
C             Calculation of the membership values over the
C             fuzzy partition
C
              DO 10 J = 1, K
C
                XMAXIMUM = 0.
C
C               Fuzzification of the input
C
                CALL dFLOWMU( Y(J), XMUdFLOW, XX, FR, NSETS)
C
C
                DO 20 I = 1, NSETS
C
C                 Clipping of the output fuzzy sets
C
                  ZmuRULE(I) = MIN( XMUdSGR(I), XMUdFLOW(I) )
C
C                 Forming the union between the sets
C
                  XMAXIMUM = MAX( XMAXIMUM, ZmuRULE(I) )
C
20              CONTINUE
C
                XNUMERATOR = XNUMERATOR + Y(J) * XMAXIMUM
                DENOMINATOR = DENOMINATOR + XMAXIMUM
C
10            CONTINUE
C
C             Defuzzification of the output
C
              IF (DENOMINATOR .EQ. 0. ) THEN
                XNR2 = 0.
              ELSE
```



```
                XNR2 = XNUMERATOR / DENOMINATOR
              END IF
C
C=================================================
C             END OF CONTROLLER
C=================================================
C
              F = F + XNR2
C
C
              IF (F .LE. 0.) THEN
                F = 0.
              ELSE
                F = F
              END IF
C
            ELSE
C             The initial batch region or final batch region
              F = 0.
              dQp = Qp - QpOLD
              dSGR = SGR - SGROLD
            END IF
C
C
              QpOLD = Qp
              SGROLD = SGR
C
C
C             Disturbance
              IF (X .GE. 10. .AND. X .LE. 15.) THEN
                S1 = 4.
              ELSE
                S1 = 2.8
              END IF
C
C             Determining the mass of substrate entering through
C             the feed stream
C
              SUBSTOT = SUBSTOT + F * S1 * STEP
C
C             Integrating the fermentation model
C
              CALL D02BDF( X, XEND, N, C, TOL, IR,
     &        FCN, STIFF, YNORM, W, IW, M, OUT, IFAIL)
C
C
100         CONTINUE
C
C
C           Evaluation of cost/objective function
C           Total amount of substrate used
            SUBSTOT = SUBSTOT + 2.8 * 5.
C           Final mass of product in fermenter
            PRODMASS = C(3) * C(4)
C
            PROFIT = PRODMASS / SUBSTOT
            PERCENT = PROFIT / 12.661395 * 100.
C
            XNSETS = NSETS
C
```

```
C         Storing the number of sets, profit ratio and profit
C         percent to file
C
          WRITE (15,15) YI, XNSETS, PROFIT, PERCENT
C
          CLOSE(9)
C
4000    CONTINUE
C
        CLOSE(15)
C
        STOP
        END
C
C
C
        SUBROUTINE FCN( T, Y, Z)
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C       Parameters
        PARAMETER (N=4)
C
C       Array arguments
        DIMENSION Z(N), Y(N)
C
        COMMON /LSETS/ F, S1, YI, SGR, Qp
C
C       Parameter values
        Ys      = 0.135
C
C       Specific growth rate
        SGR     = YI * Y(2)
C
C       Specific product formation rate
        IF (SGR .LE. 0. .OR. SGR .GE. .34896) THEN
          Qp    = 0.
        ELSE
          Qp    = -384. * SGR ** 2. + 134. * SGR
        END IF
C
C       Specific substrate utilisation rate
        SIGMA = SGR / Ys
C
C       Executable statements
        Z(1)    = SGR * Y(1) - F / Y(4) * Y(1)
        Z(2)    = F / Y(4) * ( S1 - Y(2) ) - SIGMA * Y(1)
        Z(3)    = Qp * Y(1) - F * Y(3) / Y(4)
        Z(4)    = F
C
        RETURN
        END
C
C
C=======================================================
C       Subroutines for calculating the membership grades
C       over each fuzzy partition.
C=======================================================
C
C       Determining grade of membership for triangular
C       shaped fuzzy sets
C
```



```
C       From a to b
        SUBROUTINE UP(A, B, X, XMU)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        XMU = (X - A) / (B - A)
        RETURN
        END
C
C       From b to c
        SUBROUTINE DOWN(B, C, X, XMU)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        XMU = (C - X) / (C - B)
        RETURN
        END
C
C
C
C       Inputs
C
C       Change in specific growth rate
C
        SUBROUTINE dSGRMU( X, XMUdSGR, XX, GR, NSETS)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
        DIMENSION XMUdSGR(1)
        DIMENSION XX(1)
        DIMENSION GR(1)
C
C       Assigning numbers to the variables
        DO 400 IP = 1, NSETS
          GR(IP) = (IP - 1.) / (NSETS - 1.) * (XX(2) - XX(1)) + XX(1)
400     CONTINUE
C
C
C       Set 1
        IF (X .LE. GR(1)) THEN
          XMU = 1.
        ELSE IF (X .GE. GR(1) .AND. X .LE. GR(2)) THEN
          CALL DOWN( GR(1), GR(2), X, XMU)
          XMU = XMU
        ELSE
          XMU = 0.
        END IF
        XMUdSGR(1) = XMU
C
C       Set 2 to (NSETS - 1)
        DO 410 NCOUNT = 2, NSETS - 1
         IF (X .GE. GR(NCOUNT - 1) .AND. X .LE. GR(NCOUNT)) THEN
           CALL UP( GR(NCOUNT - 1), GR(NCOUNT), X, XMU)
           XMU = XMU
         ELSE IF (X .GE. GR(NCOUNT) .AND. X .LE. GR(NCOUNT + 1)) THEN
           CALL DOWN( GR(NCOUNT), GR(NCOUNT + 1), X, XMU)
           XMU = XMU
         ELSE
           XMU = 0.
         END IF
         XMUdSGR(NCOUNT) = XMU
410     CONTINUE
C
C       Set NSETS
        IF (X .GE. GR(NSETS - 1) .AND. X .LE. GR(NSETS)) THEN
```

```
          CALL UP( GR(NSETS - 1), GR(NSETS), X, XMU)
          XMU = XMU
        ELSE IF (X .GE. GR(NSETS)) THEN
          XMU = 1.
        ELSE
          XMU = 0.
        END IF
        XMUdSGR(NSETS) = XMU
C
        RETURN
        END
C
C
C
C       Outputs
C
C       Change in flow rate
C
        SUBROUTINE dFLOWMU( X, XMUdFLOW, XX, FR, NSETS)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
        DIMENSION XMUdFLOW(1)
        DIMENSION XX(1)
        DIMENSION FR(NSETS)
C
C       Assigning numbers to the variables
        DO 430 JP = 1, NSETS
         FR(JP) = (JP - 1.) / (NSETS - 1.) * (XX(4) - XX(3)) + XX(3)
430     CONTINUE
C
C       Set NSETS
        IF (X .GE. FR(NSETS)) THEN
          XMU = 1.
        ELSE IF (X .GE. FR(NSETS - 1) .AND. X .LE. FR(NSETS)) THEN
          CALL UP( FR(NSETS - 1), FR(NSETS), X, XMU)
          XMU = XMU
        ELSE
          XMU = 0.
        END IF
        XMUdFLOW(1) = XMU
C
C       Set (NSETS - 1) to 2
        DO 420 MCOUNT = 2, NSETS - 1
         IF (X .GE. FR(NSETS - MCOUNT + 1) .AND.
     &       X .LE. FR(NSETS - MCOUNT + 2)) THEN
           CALL DOWN( FR(NSETS - MCOUNT + 1),
     &                FR(NSETS - MCOUNT + 2), X, XMU)
           XMU = XMU
         ELSE IF (X .GE. FR(NSETS - MCOUNT) .AND.
     &            X .LE. FR(NSETS - MCOUNT + 1) ) THEN
           CALL UP( FR(NSETS - MCOUNT),
     &              FR(NSETS - MCOUNT + 1) , X, XMU)
           XMU = XMU
         ELSE
           XMU = 0.
         END IF
         XMUdFLOW(MCOUNT) = XMU
420     CONTINUE
C
C       Set 1
```



```
        IF (X .GE. FR(1) .AND. X .LE. FR(2)) THEN
          CALL DOWN( FR(1), FR(2), X, XMU)
          XMU = XMU
        ELSE IF (X .LE. FR(1)) THEN
          XMU = 1.
        ELSE
          XMU = 0.
        END IF
        XMUdFLOW(NSETS) = XMU
C
        RETURN
        END
```