Improving the Identification of a Penicillin Fermentation Model

Mark Timothy Syddall

A thesis submitted to the Faculty of Engineering
of the University of Birmingham
for the degree of
Doctor of Philosophy

The School of Chemical Engineering
Faculty of Engineering
The University of Birmingham
Birmingham, UK
B15 2TT
October 1998

To my grandparents, LNJ and GG.

ABSTRACT

This work concentrates on the selection and improvement of differential equation based models of the penicillin G fermentation. Published penicillin fermentation models have been reviewed and compared with regard to their abilities to predict fermentation behaviour, genetic algorithms have been applied to the design of optimal experiments for model parameter estimation, and a new approach to assessing the theoretical identifiability of model structures has been proposed. When applied to the best penicillin fermenation model yet found, this new approach suggests that the model's parameters are uniquely identifiable.

The best performing model was shown to be a morphologically structured model for which measurement data related to the various morphologically distinct regions were obtained using image analysis. This model was modified to increase its speed of execution, and extended to describe fermentations where lactose was present in the inoculum.

Design criteria from the field of optimal experiment design were combined with genetic algorithms as a technique for searching through the range of possible input combinations, subject to constraints on the fermenter operation, to develop experimental feed profiles.

The theoretical identifiability of the fermentation model has been assessed for the first time, using a novel approach to identifiability testing which uses a symbolic mathematics package, along with subsequent post-processing, to determine almost at a glance whether or not a fermentation model should be uniquely identifiable.

ACKNOWLEDGEMENTS

I would like to thank Dr. C. A. Kent and Professor C. R. Thomas, my supervisors, for their guidance and advice over my three years in Birmingham, and for allowing me the freedom to pursue the occasional diversional line of enquiry. Particular thanks go to Dr. Gopal Paul, for providing me with experimental data from fermentations that he has carried out, and for his work in constructing and building on his original model. Thanks also to the collaborators from the University of Newcastle, Gary Montague, Maia Ignova and Alan Ward for insightful comments, useful discussions, and the opportunity to practise presenting material on a friendly audience.

Without the technical staff of the Biocentre, little of the work supporting this thesis would have been possible, so thanks to Hazel Jennings, Elaine Mitchell, Dave Bowden, David French and Rob Haddock for their expertise. I have also appreciated being able to chat ideas through with the occupants of G20 over the years, so thanks to Phil Cox, Jonathan Bleier, Sanja, Immanuel, Bénédicte, Grainne, Jayne, Stuart, Nigel, Peter, Trevor, Maria, Ade, Chris, Geni, Patricia, Iyad, Heike, Katja and the others that I am bound to have forgotten. G20 has been a busy place over the last three years.

I gratefully acknowledge the financial support of the EPSRC throughout my PhD studentship, and the support of both the EPSRC and the School of Chemical Engineering in helping fund my visit to the 'Computer Applications in Biotechnology 7' Conference in Osaka in June, 1998.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The University of Birmingham Centre for Bioprocess Engineering Rolling Grant Project B: 'Monitoring and Physiological Control of Productive Fermentations' proposes to combine physiological models of fermentations with Artificial Neural Networks (ANNs) to produce hybrid models and to investigate their applicability for use in monitoring and controlling the penicillin fermentation.

The work described in this thesis forms part of that larger body of work aimed at developing control and optimisation applications based on physiological models of the penicillin fermentation. It is intended that the resulting applications should be the best that can currently be achieved, based solely on physiological equation models, and that the applications developed would provide a baseline for performance against which schemes using artificial neural networks, and hybrid schemes incorporating both artificial neural networks and physiological model equations, could be compared.

1.1 The Penicillin Fermentation

The penicillin fermentation has been performed industrially since the Second World War. The original *Penicillium notatum* cultures had yields of only 2 mg/l, but searching many different varieties of *Penicillium* led to the identifi-

cation of the higher-yielding variant *Penicillium chrysogenum*. A systematic process of deliberate exposure to mutagens and screening to find high yielding mutants, along with improvements in fermentation operation and media have given an increase in titre to over 20 g/l (Primrose, 1987).

The world market in penicillin is competitive, with recent developments in India and China leading to oversupply and a consequent drop in the price of penicillin from $ 18/bu in the middle of 1996 to $ 10/bu in the second quarter of 1997, but the size of the market as a whole is still considerable, despite a drop in production from about 40,000 metric tons in 1996 to around 37,000 metric tons in 1997 (Chemical Market Reporter, 1998). With the market remaining competitive, and new producers in India and China increasing their output, the improved operation and control of fermentations remain important concerns for penicillin producers.

1.2 Fermentation Modelling

Fermentation models are produced for two main reasons: to test hypotheses about the way in which the fermentation behaves, and to provide a relatively quick and cheap way of experimenting with fermentation feed profiles and control strategies. In this thesis we are mainly concerned with being able to predict the behaviour of a fermentation, with the goal of using a model which describes the fermentation well in developing an open-loop optimal feed profile to maximise the profitability of the fermentation.

Although unstructured models, where the biomass is treated as an averaged, lumped whole have been used to model the penicillin fermentation, the validity of using such an approach to model fermentations which employ fila-

mentous fungi has been challenged (Nielsen, 1992), since "the growth mechanism of filamentous fungi is ... completely different from that of unicellular organisms" and so "for a complete description of fermentation processes it is ... important to consider the hyphal structure".

1.2.1 Types of model

Schügerl (1986) gave the following list of levels on which models used in biotechnological applications may be developed. (Both the level on which the models are developed and descriptive names for the type of models were given.)

- molecular or enzyme level (enzyme synthesis models)
- intracellular component level (structured cell models)
- cellular level (kinetic models)
- cellular environmental level (unstructured reactor models)
- dynamic cellular environmental level (structured reactor models)

Since the availability of on-line fermentation measurements is essential for models to be used as part of a control system, it is not currently possible to develop control systems based around Schügerl's first two levels of model, enzyme synthesis models and structured cell models, as on-line measurements of enzyme and intracellular concentrations are not feasible. Of the remaining three levels listed, the cellular level and the cellular environmental level are commonly combined when constructing fermentation models, with simple kinetics being used to describe biomass growth, product formation

and substrate consumption, and the assumption commonly being made that the fermenter is well-mixed and so may be approximated by a continuous stirred tank reactor (CSTR).

The models considered in this work are typically made up of terms describing processes which are known or proposed as occurring during the course of the penicillin fermentation. Historically such models have been built up using terms familiar to fermentation engineers, such as Monod, Contois and inhibition kinetics (Nielsen and Villadsen, 1994). (The equations defining the models considered in the course of this work are given in Appendix A.) Although these do not describe the underlying metabolic processes carried out by the organism, they are generally accepted as being good approximations to the gross, overall behaviour observed during the fermentation (Nielsen and Villadsen, 1994).

Attention has specifically not been focussed on models based around artificial neural networks, as one of the goals of the overall project within which the work described in this thesis lies is to provide the best exploitation of knowledge-based differential equation models of the penicillin fermentation, against which the performance of pure artificial neural network models and hybrid differential equation/artificial neural network models in control-related applications may be compared.

1.3 Model-based Control Applications

Provided that a model gives a sufficiently close description of how the fermentation proceeds, it may be possible to make practical use of it. Three of the more common applications of models in fermentation control strategies

are to estimation of states and rates, open-loop optimisation of feed profiles (and other input profiles), and to the construction of controllers themselves.

1.3.1 Estimation

The use of models in constructing state and rate estimators for use with fermentations occurs often in the published literature. A range of techniques has been applied to the on-line estimation of fermentation states, with extended Kalman filter approaches possibly being the most common (Tarbuck *et al.*, 1986; Náhlík and Burianec, 1988; Pons *et al.*, 1988; Shi and Yuan, 1988; Lee and Ricker, 1994; Myers *et al.*, 1996); recursive parameter estimation (Montesinos *et al.*, 1995) is an alternative that has been applied to a structured model describing the *Candida rugosa* fermentation. Di Massimo *et al.* (1992) applied both partially and fully adaptive estimators to the estimation of biomass concentration using offgas measurements taken from an industrial penicillin fermentation.

Estimators have also been constructed to track growth and production rates on-line (Hardwicke *et al.*, 1991; Cazzador and Lubenova, 1995; Farza *et al.*, 1997) and have been integral in the construction of control schemes associated with pre-designed, open-loop optimal fermentation trajectories (Gattu and Zafiriou, 1995; King, 1997). It is therefore not unreasonable to assume that ultimately an on-line estimator for the penicillin fermentation may be constructed using the refined models available after using the methods described later in this thesis.

1.3.2 Optimisation

The use of models in seeking optimal control trajectories for the penicillin fermentation has a long history, and many attempts at this have been made over the years (Fishman and Biryukov, 1974; Lim *et al.*, 1986; San and Stephanopoulos, 1989; van Impe *et al.*, 1992; van Impe and Bastin, 1995; Rodrigues and Filho, 1996). The models used as the bases for these designs were the model due to Ramkrishna *et al.* (1967), the widely used model of Bajpai and Reuß(1980), and the model of Nicolai *et al.* (1991).

1.3.3 Control

The use of models in the construction of controllers for fermentations is almost inextricably bound up with the previous two model applications. Since on-line measurements of fermentations tend to be limited, with offgas analysis being the most common high-rate measurement, such analysis is frequently used as input to state estimators with the state estimate produced being employed as the measured value in controllers designed to track pre-designed optimal trajectories (King, 1997).

Montague *at al.* (1986), working with models of the penicillin fermentation, combined an extended Kalman filter with a self-tuning controller. Van Impe and Bastin (1995) used the penicillin-G fermentation to illustrate the performance of adaptive controllers making use of three different sets of available measurements: biomass and substrate concentrations, the substrate concentration only, and on-line measurement of the carbon dioxide evolution rate (CER).

It is anticipated that future work, following on from that described here,

will consider the use of the best-performing differential equation based models in the optimisation, estimation and control of the penicillin fermentation.

1.4 The Importance of Good Models

The better the model fits and predicts fermentation data, the closer to the optimum any 'optimal' feed profile designs based on the model will be, the less difficulty there is likely to be in constructing on-line estimators (using the more frequent offgas, dissolved oxygen concentration and dissolved species (HPLC) measurements) to produce estimates of biomass concentrations between sample intervals for use as a part of a control scheme, and the easier it may be to produce a robust control scheme which can cope with inaccuracies in the model. For these reasons, this thesis concentrates on selecting the best performing of the existing published models of the penicillin fermentation, and on designing experiments to produce data on the basis of which confidence in the model's parameters may be increased.

1.5 Layout of this Thesis

This thesis outlines the process by which the best of the available penicillin fermentation models was identified and refined, and then goes on to describe theoretical applications concerned with model identification and model identifiability.

- Chapter two

  In order to be confident that we have found the best basis against which to compare the ANN applications, it was considered reasonable

to start with the penicillin fermentation model which best describes the behaviour of the penicillin fermentation, both in fitting to measured data and in predicting the behaviour of fermentations.

- Chapter three

  Having found the best-performing of the existing penicillin fermentation models, some work was then necessary to improve the performance of the model for our purposes. This involved simplifying part of the existing model, with consequent improvements in simulation speed, and adding a model state and relationships to describe the way in which an additional substrate, lactose, is used during the fermentation.

- Chapter four

  On the basis of this modified fermentation model, work was then carried out on optimal experiment design, with the goal of improving confidence in the model parameter estimates. This involved using genetic algorithms, which are introduced later in this chapter.

- Chapter five

  Attention was also been paid to the question of whether or not the parameters which fit the model to the data are unique or not. Existing nonlinear model identifiability techniques were reviewed, and a simple approach was developed from these, which is shown to give the same conclusions as the existing methods, but which seems simpler to use. Although limited in its power with respect to one of the existing methods, the simple approach is considered to be adequate for our purposes.

1.6 Hardware and Software

All the work described in this thesis was carried out using IBM compatible personal computers. Two different operating systems were used: Microsoft Windows 3.11 running over Microsoft DOS 6.22 and Red Hat Linux release 4.1 (Kernel 2.0.27). The modelling and experiment design work was done on a Windows PC, using MATLAB 4.2c with SIMULINK 1.3c, also using the MATLAB Optimization Toolbox, and the Genetic and Evolutionary Algorithms Toolbox (Pohlheim, 1996). Version 10.7 of the Watcom C compiler was used to produce `cmex` files and for SIMULINK model acceleration, Maple V release 4 was used to produce matricial differentials and to generate C code, and to perform manipulations and simplifications to the model equations, with Perl 5.003 (on the Linux PC) being used as the scripting language for post-processing Maple text output. This thesis was written using LaTeX2$_\epsilon$ running on a Linux PC.

2. SELECTING THE BEST MODEL

2.1 Introduction

The penicillin fermentation is an industrially important antibiotic fermentation, and is commonly studied as a model system for secondary metabolite production. A large number of models of the process have been published in the literature, ranging in complexity from simpler unstructured models such as that of Bajpai and Reuß (1981), where the biomass is modelled as a single lumped mass, to more complex morphologically structured models such as that of Megee *et al.* (1970), where the biomass is broken down into distinct fractions by association with product formation, or on the basis of observable morphological differences. (Here, models that divide the biomass solely into live and dead fractions are regarded as unstructured.)

In this chapter, we consider penicillin models with regard to their usefulness for optimisation and control studies. It is anticipated that the fermentation may be controlled by varying the substrate concentration in the fermenter, that penicillin is the product of interest, and that both the rate of biomass growth and the rate of penicillin production depend on the substrate concentration, as well as the biomass concentration. Hence a minimum requirement for us to consider a model for such uses is that it represent biomass, substrate and penicillin concentrations.

First we consider the bases of the terms from which the models are constructed, with regard to the ways in which they represent observed features of the fermentation. Then we present a comparison of the performance of the models in tuning and validation against a single, common pair of fermentation records.

The models were originally defined for a range of different fermentation conditions, both in terms of fermenter scale and of medium composition. Our comparison here is based on two sets of fed-batch fermentation data produced for identical conditions of fermenter scale and medium composition (Paul *et al.*, 1994), differing only in their input feed rate profiles. These fermentations were performed using a penicillin-G producing strain of *Penicillium chrysogenum*, and so the models are being compared particularly with reference to penicillin-G producing fermentations.

2.2 Model Structure Comparison

As mentioned earlier, models of the penicillin fermentation should include at least states representing the biomass, substrate (commonly modelled as a single limiting component, often glucose) and penicillin concentrations, and our efforts have been focussed on how accurately the collected models predict these three states.

As originally published, some of the models included additional nutrient states, such as dissolved oxygen (Bajpai and Reuß, 1980), lactose and lysed biomass concentrations (Kluge *et al.*, 1992), or additional products (Megee *et al.*, 1970), or product precursors (Nestaas and Wang, 1983), or additional expressions, such as the carbon dioxide production rate (Mon-

tague *et al.*, 1986). Since the data we have used in comparing the performance of the models do not include measurements for any of the above (the measured data are for the biomass concentrations, both as individual, morphologically distinct fractions and as total biomass concentration, and for the glucose and penicillin concentrations), we cannot consider them here. When these models were built for tuning and comparison, the influence of such terms on biomass growth, substrate consumption and penicillin formation was neglected, with these terms being replaced with constants. (The carbon dioxide production has been modelled as being dependent on biomass concentration and growth rate, and penicillin formation rate (Montague *et al.*, 1986), but no influence of the dissolved carbon dioxide concentrations on biomass, substrate or penicillin concentrations was modelled.) As a result of setting such terms constant, two of the models were effectively reduced to a common form (Bajpai and Reuß, 1981; Montague *et al.*, 1986).

The following models have been compared:

- Unstructured
  - Fishman and Biryukov (1974)
  - Heijnen *et al.* (1979)
  - Bajpai and Reuß (1980/81)
  - Nicolai *et al.* (1991)
  - Kluge *et al.* (1991)
  - Menezes *et al.* (1994)
  - Tiller *et al.* (1994)

- Morphologically structured
  - Megee *et al.* (1970)
  - Nestaas and Wang (1983)
  - Cagney *et al.* (1984)
  - Paul and Thomas (1996)

Of the above models, the majority (Fishman and Biryukov, 1974; Bajpai and Reuß, 1980; van Suijdam *et al.*, 1982; Nestaas and Wang, 1983; Cagney *et al.*, 1984; Tiller *et al.*, 1994) described generic penicillin producing fermentations. Most of the others (Heijnen *et al.*, 1979; Nicolai *et al.*, 1991; Menezes *et al.*, 1994; Paul and Thomas, 1996) referred specifically to the production of penicillin-G. Only the model of Kluge *et al.* (1992) was originally proposed as describing a penicillin-V forming fermentation.

The model of Megee *et al.* (1970) was originally proposed as a general model for mould growth, associating product formation with different fractions of the mould. Although it was originally presented with reference to *Aspergillus awamori*, it is considered here as a candidate for use in modelling the penicillin fermentation because of the model's general structure. In using this model to describe the penicillin fermentation, we have assumed that both of the non-growth-associated products given in the original model (see Appendix A.2) are penicillin. The growth-associated product in the original model has been ignored.

The penicillin fermentation has a number of distinct observed features, each of which may be represented by a term in the models. These include the following:

1. Biomass terms
   (a) Biomass growth
   (b) Conversion between biomass fractions
   (c) Biomass lysis
2. Substrate terms
   (a) Growth related consumption
   (b) Production related consumption
   (c) Maintenance related consumption
3. Penicillin terms
   (a) Formation
   (b) Hydrolysis
4. Dilution terms
   To describe the fed-batch fermentation data used here, these would be of the form $\frac{F(Z_{in}-Z)}{V}$, where F is the input feed rate, Z is the concentration of some model state in the fermenter, Z_{in} is the concentration of the same state in the feed, and V is the volume of medium present in the reactor.

The models for the penicillin fermentation that have been published in the literature have used different symbols to describe similar states found and processes occurring in the models. To ease comparison between the models, they have been collected together here, rewritten in a common form. The

model descriptions may be found in Appendix A.1 for unstructured models and Appendix A.2 for morphologically structured models.

2.2.1 Biomass terms

Biomass growth

Growth is described, for the majority of the models, by either Monod kinetics,

$$\frac{\mathrm{d}X}{\mathrm{dt}} = \frac{\mu_X S X}{K_S + S}$$

or Contois kinetics.

$$\frac{\mathrm{d}X}{\mathrm{dt}} = \frac{\mu_X S X}{K_X X + S}$$

Of these, the Monod expression was historically the first (Monod, 1942), and has been widely used, as it is simple and models using it are easily analysed. The models of Heijnen *et al.* (1979) (Eqn A.6), Nicolai *et al.* (1991) (Eqn A.18) and Kluge *et al.* (1992) (Eqn A.31) calculate the growth rate from the substrate uptake rate, which is modelled using a Michaelis-Menten kinetic (mathematically identical to the Monod expression). The morphologically structured models of Megee *et al.* (1970) (Eqns A.38,39), Cagney *et al.* (1984) (Eqns A.56,67) and Paul and Thomas (1996) (Eqns A.62.63) all model growth behind the hyphal tips as following Monod kinetics, with the formation of new hyphal tips by branching from other hyphal states also

modelled as following Monod kinetics. According to Nielsen and Villadsen (1994): "the Monod model has been shown to correlate fermentation data for many different organisms." However, "the satisfactory fit of the Monod model to many experimental data should never be misconstrued to mean that the Monod equation is a *mechanism* of fermentation processes."

The Contois expression (Contois, 1959) is an alternative to the Monod expression, used in modelling systems where the biomass increases to a concentration considered to have a significant direct effect on the specific growth rate, causing it to decrease with increasing biomass concentration. This form was first used in modelling the penicillin fermentation by Bajpai and Reuß (1980) (Eqn a.10), and has subsequently been used in models based thereon (Montague *et al.*, 1986; Nicolai *et al.*, 1991; Menezes *et al.*, 1994) (Eqns A.14, A.18 and A.21, respectively).

Nestaas and Wang (1983) divided the growth into two phases, a 'growth phase' and a 'production phase'. In the 'growth phase' the rates of growth of both tips and bulk hyphal material were linearly proportional to the concentration of tips (Eqns A.48,49). In the 'production phase' tip growth ceased and the growth rate of bulk hyphal material remained linearly proportional to the tip concentration (Eqns A.50,51).

Conversion between biomass fractions

Morphologically structured models (Megee *et al.*, 1970; Nestaas and Wang, 1983; Cagney *et al.*, 1984; Paul and Thomas, 1996) (Subsections A2.1, A2.2, A2.3 and A2.4) include terms which describe the rate at which one biomass fraction changes to another, say from being growing tips to general hyphal

material. These include expressions to describe processes which have previously been named 'differentiation', 'degeneration' and 'dormancy'. Provided that these sets of expressions are internally consistent, with material being conserved as it moves from one state to the next, there is little that can be said about them at this time.

Biomass lysis

Autolysis terms describe the rate at which the biomass is destroyed.

Van Suijdam *et al.* (1982), modelling the growth of *Penicillium chrysogenum* in pelleted form, stated that it is known that mycelia in fermentations undergo lysis at low oxygen or substrate concentration. Indeed, it is generally accepted that microorganisms will eventually lyse at low oxygen or substrate concentration. However, we assume here that the oxygen concentration in the fermentation broth does not fall to levels liable to result in lysis.

There are models which include a state representing *dead* or *dormant* biomass (Megee *et al.*, 1970; Fishman and Biryukov, 1974; Nestaas and Wang, 1983; Cagney *et al.*, 1984; Menezes *et al.*, 1994). These do not model lysis as such, but do include 'death' terms for the conversion of biomass from a live, active state into an inactive state.

A number of models include lysis terms which result in the destruction of biomass. Kluge *et al.* (1992) (Eqn A.30) and Paul and Thomas (1996) (Eqn A.64) model lysis as proceeding at a rate linearly proportional to the concentration of 'inactive' biomass. Tiller *et al.* (1994) use an age-dependent term to calculate the lysis coefficient (Eqn A.26).

2.2.2 Substrate terms

Carbon balancing suggests that there should be substrate consumption or endogenous metabolism terms expressing the utilisation of substrate material in the formation of biomass and product. It is also appreciated that viable biomass uses energy, derived from substrate, for biomass maintenance.

Growth related consumption

All of the models considered describe the growth-related consumption of substrate using a constant yield coefficient.

Production related consumption

Most of the models considered here model penicillin production related substrate consumption using a constant yield coefficient. There are only two exceptions which do not have a substrate consumption term associated with product formation (Megee *et al.*, 1970; Fishman and Biryukov, 1974).

Nicolai *et al.* (1991) included a term to describe endogenous production, occurring at low substrate concentrations (Section A 1.5). This took the form of a modifier, $(1 - e^{-S/E_P})$, multiplied by the production rate, where S is the substrate concentration and E_P is a constant.

Maintenance related consumption

The model of Fishman and Biryukov (1974) does not have a term to describe consumption of substrate associated with the maintenance requirements of the biomass.

Of the models that do consider biomass maintenance, four do so by means of a linear term proportional to the amount of biomass present (Bajpai and Reuß, 1980; Nestaas and Wang, 1983; Montague *et al.*, 1986; Tiller *et al.*, 1994) (Eqns A.11, A.55, A.15 and A.27, respectively). In a model this type of term can result in substrate being consumed even when there is no substrate present. Although this is mathematically feasible, it is clearly a biological impossibility.

Four models (Megee *et al.*, 1970; Cagney *et al.*, 1984; Menezes *et al.*, 1994; Paul and Thomas, 1996) use a Monod type (Michaelis-Menten) expression to describe the maintenance substrate consumption rate (Eqns A.43, A.59, A.22 and A.66, respectively). This term goes to zero as the substrate goes to zero, thus avoiding the feasibility problem associated with a term purely proportional to the biomass concentration.

The model of Nicolai *et al.* (1991) has a complex substrate consumption expression, which models endogenous metabolism (Eqn A.19). The maintenance term in this model is of the form $m_s * (1 - e^{-S/E_m})$, where m_s is the maintenance coefficient, S is the substrate concentration, and E_m is a constant.

Heijnen *et al.* (1979) (Eqn A.7) and Kluge *et al.* (1992) (Eqn A.33) consider the biomass maintenance as consuming a portion of the substrate taken up by the hyphae. This has the effect of reducing the growth rate, but does not affect the rate of substrate uptake itself, which is modelled using Michaelis-Menten kinetics.

The maintenance term is easier to understand for unstructured models, where it represents consumption of substrate which is not associated

with either growth or product formation. For the morphologically structured models, what has been referred to here as the maintenance term is often proportional to the terms used to describe conversion between biomass states. This conversion contributes neither to biomass growth nor to product formation, and so may be regarded as a maintenance process.

2.2.3 Penicillin terms

Penicillin is a secondary metabolite of *Penicillium chrysogenum*, and its production is known to be non-growth-associated. Its rate of formation is known to be reduced at high glucose concentrations, presumably as a result of some inhibition or repression mechanism. The mechanism of this is not yet known, and so no specific form can be indicated for the penicillin formation term.

Formation

The most common form for the production term is a type of inhibition kinetics first used in the context of penicillin fermentation modelling by (Bajpai and Reuß, 1980).

$$\text{Production rate} \quad = \quad \frac{\mu_P S X}{K_P + S(1 + (S/K_I))}$$

This is used in both unstructured and morphologically structured models. In the unstructured models (Bajpai and Reuß, 1981; Montague *et al.*, 1986; Nicolai *et al.*, 1991), production has been associated with the total amount of biomass present in the system (Eqns A.12, A.16 and A.20, respectively). The

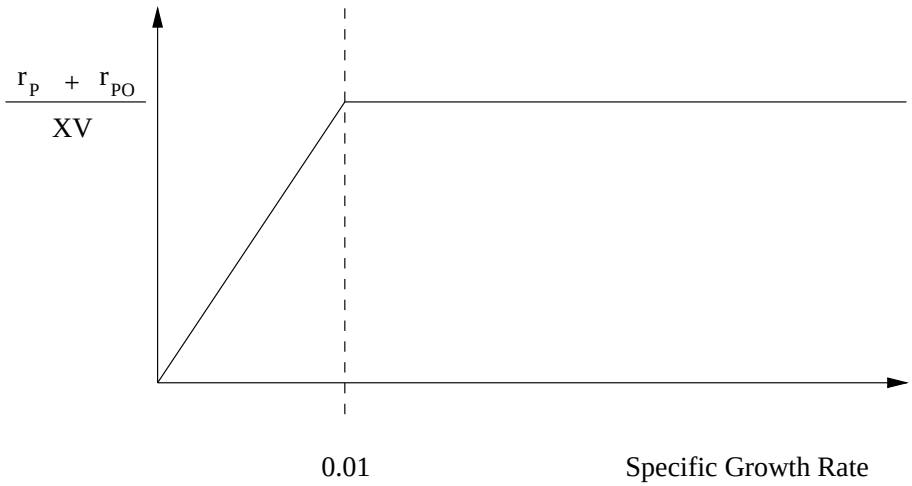

Fig. 2.1: Product formation relation used by Heijnen *et al.*, 1979

structured models (Cagney *et al.*, 1984; Paul and Thomas, 1996) associate the production with a specific portion of the biomass (Eqns A.60 and A.67). In none of the structured models is production associated with the hyphal tips, where hyphal growth is most active. Menezes *et al.* (1994) observed no catabolite repression of production in their work, and so replaced the inhibited formation term with a simpler, Monod-type product formation term (Eqn A.23).

Two models (Heijnen *et al.*, 1979; Tiller *et al.*, 1994) relate the product formation rate to the specific growth rate of the biomass (Eqns A.8 and A.29). Both assume a minimal growth rate, below which product formation decreases with decreasing growth rate. Tiller *et al.* (1994) also make use of a maximal growth rate, above which the product formation starts to decrease with increasing growth rate. These two relationships are shown in Figures 2.1 and 2.2.

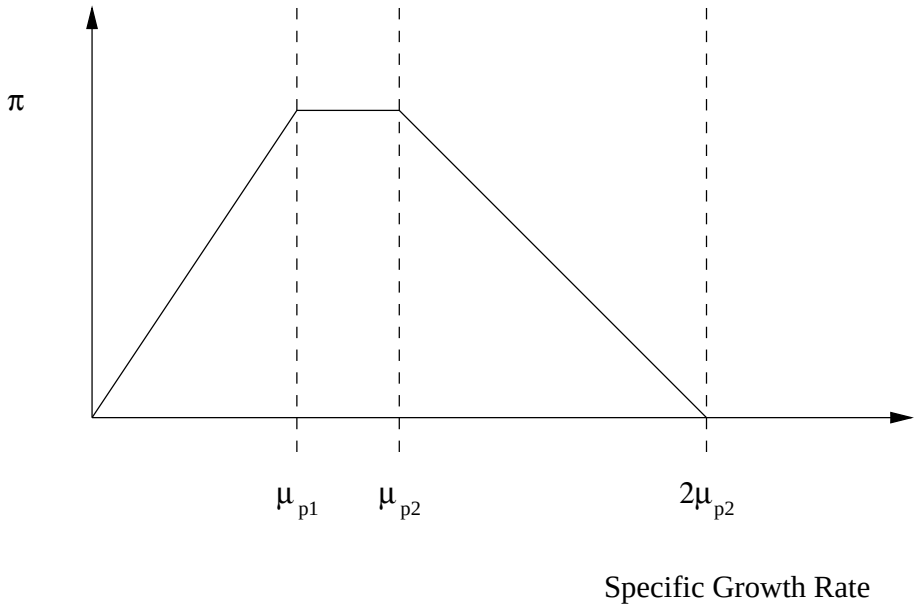

Fig. 2.2: Product formation relation used by Tiller *et al.*, 1994
(μ_{p1} is the lower limit of the maximum penicillin formation rate,
μ_{p2} is the upper limit of the maximum penicillin formation rate.)

Other models have used age-associated production (Fishman and Biryukov, 1974) (Eqn A.4), or production via a postulated intermediate (Nestaas and Wang, 1983) (Eqns A.53,54). As stated earlier, in Section 2.2, the two non-growth-associated products in the model of Megee *et al.* (1970) (Eqns A.44-46) have been lumped together and assumed to be penicillin. Their rates of formation are described using Monod kinetics. All of these expressions can represent non-growth associated kinetics, and any of these expressions could be associated either with the total biomass present, or with some distinguishable portion of the biomass.

The term used by Kluge *et al.* (1992) (Eqns A.35,36) may be shown to

be equivalent to a steady-state inhibition rate expression, with an additional constant in its numerator, subject to a first order lag.

Hydrolysis

It is known that penicillin undergoes hydrolysis to penicilloic acid in aqueous solution (Benedict *et al.*, 1945), and that the reaction is first order with respect to penicillin. Three of the models do not have hydrolysis terms (Megee *et al.*, 1970; Fishman and Biryukov, 1974; Heijnen *et al.*, 1979).

2.2.4 Dilution terms

The descriptions of the models given in Appendix A.1 do not include dilution terms, but the models as built and tuned do. The general derivation of the dilution terms is as follows.

Generally, for species X in reactor volume V.

$$\text{Accumulation} = \text{In} - \text{Out} - \text{Reaction}$$

The total quantity of species X present in the reactor is xV (concentration times volume). Rewriting the above equation we have

$$\frac{\mathrm{d}[V.x]}{\mathrm{dt}} = Q_i x_f - Q_o x_o - V.r \tag{2.1}$$

$$V\frac{\mathrm{d}x}{\mathrm{dt}} + x\frac{\mathrm{d}V}{\mathrm{dt}} = Q_i x_f - Q_o x_o - V.r \tag{2.2}$$

the volume rate expression is given by

$$\frac{\mathrm{d}V}{\mathrm{dt}} = Q_i - Q_o \tag{2.3}$$

So, dividing throughout by V, we obtain the general form

$$\frac{\mathrm{d}x}{\mathrm{dt}} + \frac{x(Q_i - Q_o)}{V} = -r + \frac{Q_i x_f - Q_o x_o}{V} \tag{2.4}$$

For a batch reactor, $Q_i = Q_o = 0$

$$\frac{\mathrm{d}x}{\mathrm{dt}} = -r \tag{2.5}$$

For a fed-batch reactor, $Q_o = 0$

$$\frac{\mathrm{d}x}{\mathrm{dt}} = -r + \frac{Q_i(x_f - x)}{V} \tag{2.6}$$

For a CSTR, $Q_i = Q_o$

$$\frac{\mathrm{d}x}{\mathrm{dt}} = -r + \frac{Q_i x_f - Q_o x_o}{V} \tag{2.7}$$

So, to model a generic reactor, useful for batch, fed-batch and continuous fermentation, we have:

$$\frac{\mathrm{d}x}{\mathrm{dt}} = -r + \frac{Q_i x_f - Q_o x_o}{V} - \frac{(Q_i - Q_o)x}{V} \tag{2.8}$$

In the above, Q_i is the feed rate to the fermenter, x_f is the concentration

	Biomass		Substrate			Penicillin	
Model Name	Growth	L.	G.r.	P.r.	M.	Production	H.
Megee *et al.*	Monod	√	√	√	√	Monod	×
Fishman and Biryukov	Monod	×	√	×	×	Age-related	×
Heijnen *et al.*	Monod	×	√	√	√	Linear	×
Bajpai and Reuß	Contois	×	√	√	√	Inhibited	√
Nestaas and Wang	Linear	√	√	√	√	Precursor	√
Cagney *et al.*	Monod	√	√	√	√	Inhibited	√
Montague *et al.*	Contois	×	√	√	√	Inhibited	√
Kluge *et al.*	Monod	√	√	√	√	See App. A	√
Nicolai *et al.*	Contois	×	√	√	√	Inhibited	√
Menezes *et al.*	Contois	√	√	√	√	Monod	√
Tiller *et al.*	Monod	√	√	√	√	See App. A	√
Paul & Thomas	Monod	√	√	√	√	Inhibited	√

Tab. 2.1: Summary of features included in the models (√– Feature present, ×–Feature absent) [L.–Lysis, G.r.–Growth related, P.r.–Production related, M.–maintenance, H.–Hydrolysis]

of species X in the feed, Q_o is the rate of removal of liquor from the fermenter, x_o is the concentration of species X in the stream leaving the fermenter, and x is the concentration of species X in the fermenter. The distinction between x_o and x is made, because in the case of online measurement of dissolved species using HPLC, the sampled stream leaving the fermenter is filtered, and so the concentration of insoluble biomass fractions in the sample stream is zero, which is not the same as that in the fermenter. The withdrawal of a filtered sample stream acts in such a way as to concentrate the insoluble species in the fermenter.

2.2.5 Summary of model features

Table 2.1 summarises the terms found in the models.

2.3 Comparing Model Performance

The models' performances in fitting to and predicting fermentation behaviour were compared using data supplied by Paul (1996).

2.3.1 Fermentation method

The data used in tuning and validating the models were taken from two 5 litre working volume fermentations carried out using a pre-production strain of *Penicillium chrysogenum* under the same conditions of fermenter scale and medium composition. The fermentation protocol used was that described by (Paul *et al.*, 1994). The two fermentations differed only in the substrate feed profile used. The first data set used was produced using a constant feed profile, whilst the second was produced using a feed profile with a step, followed by a decreasing ramp. Both feed profiles are shown in Figure 2.3. These feed profiles were used in the original work of Paul *et al.* (1994) to investigate the influence of substrate concentration on the rate of vacuole formation and hyphal differentiation.

The two data sets produced, for constant feed rate, and time-varying feed rate, comprise eighteen and nineteen measurement times respectively, with all states (four biomass states and three soluble species) being measured coincidentally at irregular intervals which vary from four to twelve hours (measurements being more frequent during the first 50 or so hours of the fermentation).

Image analysis was used to measure the relative proportions of differing morphological fractions, following the method used by Paul *et al.* (1994). The fractions identified here are those used by Paul and Thomas (1996) in

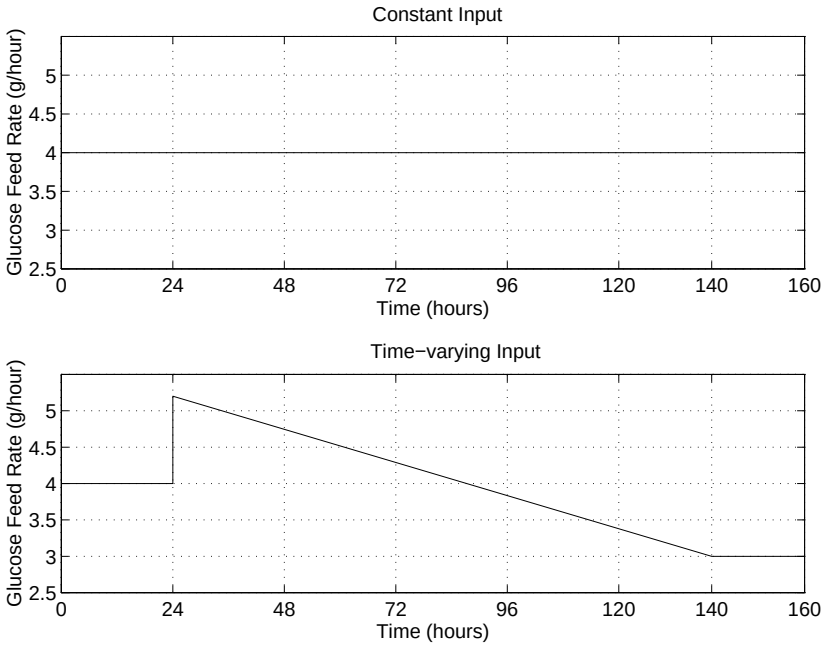

Fig. 2.3: Feed profiles used in generating the data sets used

their model. These will not necessarily be the same as, nor correlate well with, the fractions defined in other morphologically structured models, as image analysis was not used in determining the biomass fractions for the earlier models.

2.3.2 Modelling and tuning methods

The models were built using MATLAB and SIMULINK, and tuned using routines from the MATLAB Optimisation toolbox. SIMULINK is a block-diagram oriented modelling tool; the SIMULINK block diagram for the model of Paul *et al.* (1998) is shown in Figure 2.4. A least squares routine (Levenberg-Marquardt algorithm) was used to tune each model's parameters, with the target error function being calculated as follows.

- Simulate the model over the time period of the reference data set, using a fourth order Runge-Kutta algorithm.
- Interpolate linearly within the model output to obtain simulated values corresponding to the times of the experimental measurements.
- Calculate the difference between the measured and simulated values.
- Weight the differences for each model state by the inverse of the maximum value in the measured data set for that state. Weighting each state using 1/(noise variance), a commonly used approach, was considered, but the approximate assumption that the noise is Gaussian becomes less true for low state values - considering particularly the low values observed for the glucose and penicillin concentrations - with the

distribution of the noise possibly becoming skewed in favour of positive noise values. The approach used here attempts to normalise the difference values with respect to the maximum measured values of each of the states.

- Square and sum the weighted differences. (This is done by the optimisation routine–it works on the matrix of weighted differences.)

Mathematically, the target function can be expressed as follows,

$$\text{Error} = \sum_{i=1}^{n} \left[\left(\frac{X_{meas(i)} - X_{sim(i)}}{\max(X_{meas})} \right)^2 + \left(\frac{S_{meas(i)} - S_{sim(i)}}{\max(S_{meas})} \right)^2 + \left(\frac{P_{meas(i)} - P_{sim(i)}}{\max(P_{meas})} \right)^2 \right]$$

where the summation is carried out for all times corresponding to measurement times, and the subscripts $meas$ and sim denote measured and simulated values respectively.

2.3.3 Validation results

If a model were to predict perfectly the behaviour of the system, there would be a residual least squared error between simulated and measured data, related to the noise on the measurements. Assuming that the noise is Gaussian and proportional to the magnitude of the measurement (this is a simplification, as errors are likely to be larger relative to the measurement value for small values), this expected residual error can be calculated. Here we have assumed that the percentage errors on the three measured states are as follows: Biomass $\approx 10\%$, Substrate $\approx 5\%$, Penicillin $\approx 1\%$. Expected residual

errors for the constant and time-varying input data sets are given in Table 2.2. Since the magnitudes of the expected error values are proportional to the assumed percentage errors, expected error values for other percentage errors may be obtained by appropriately scaling the values given here.

State	Constant Input	Varying Input
Biomass (10%)	0.09	0.11
Substrate (5%)	0.006	0.008
Penicillin (1%)	4e-4	4e-4
Overall	≈ 0.10	≈ 0.12

Tab. 2.2: Expected residual errors for a perfect model (normalised for each state with respect to its maximum measured value, and summed to give the overall error)

For both of the sets of fermentation data, the models were tuned as described above, and then validated against the set of data not used in the tuning. The results of validating the models are shown in Figures 2.6 to 2.11. For comparison purposes, the summed least squares errors for biomass, substrate and penicillin concentrations, along with an overall value, have been tabulated. Errors for tuning using the constant feed profile and validating against the time-varying profile are given in Table 2.3. Errors for the converse are given in Table 2.4.

Model Name	Summed Squared Error			
	Biomass	Glucose	Penicillin	Overall
Paul and Thomas	0.10	0.04	0.03	0.17
Kluge *et al.*	0.03	0.19	0.29	0.51
Cagney *et al.*	0.09	0.42	0.04	0.55
Megee *et al.*	0.17	0.32	0.21	0.70
Nicolai *et al.*	0.03	0.41	0.30	0.74
Menezes *et al.*	0.10	0.33	0.42	0.85
Bajpai and Reuß	0.02	0.19	0.83	1.04
Fishman and Biryukov	0.51	0.32	0.53	1.36
Heijnen *et al.*	0.09	0.29	1.18	1.56
Tiller *et al.*	0.20	0.75	0.97	1.93
Nestaas and Wang	0.98	84.64	1.33	86.95

Tab. 2.3: Validation results for tuning using the constant feed data set (normalised for each state with respect to its maximum measured value in the time-varying feed data set, and summed to give the overall error)

2.4 Discussion

2.4.1 Difficulty in tuning models

Tuning biological models is difficult. There is no guarantee that the 'optimum' parameter sets reached using optimisation methods are global; optimisation techniques may converge to a local optimum, depending on initial conditions. There may also be difficulties in determining exact values for specific parameters as model terms may often be reduced to simpler forms for extreme values of model states. For example, consider the Contois expression for the specific growth rate, $\mu_X XS/(K_X X + S)$. At low substrate concentrations, this expression reduces to the linear form, $\mu_X S/K_X$. (These two forms are plotted for comparison in Figure 2.5.) In the data sets used

Model Name	Summed Squared Error			
	Biomass	Glucose	Penicillin	Overall
Paul and Thomas	0.06	0.09	0.04	0.18
Cagney *et al.*	0.08	0.35	0.02	0.45
Menezes *et al.*	0.11	0.07	0.38	0.55
Nicolai *et al.*	0.04	0.18	0.44	0.66
Megee *et al.*	0.15	0.12	0.45	0.73
Bajpai and Reuß	0.03	0.06	0.76	0.85
Kluge *et al.*	0.04	0.23	0.75	1.03
Heijnen *et al.*	0.04	0.18	1.22	1.44
Tiller *et al.*	0.11	0.75	0.61	1.47
Fishman and Biryukov	0.54	0.22	1.47	2.24
Nestaas and Wang	4.69	258	4.49	267

Tab. 2.4: Validation results for tuning using the time-varying feed data set (normalised for each state with respect to its maximum measured value in the constant feed data set, and summed to give the overall error)

here, the substrate concentration is reduced almost to zero after the first 40 hours of the fermentation. Whilst there is a large discrepancy between the Contois expression and its linearised form for the first 40 hours of the fermentation, this corresponds to only four or five measured data points. So, although we may be confident that we have obtained an appropriate ratio of μ_X to K_X, we should be a little less certain about the absolute values.

Given that the morphologically structured models have more states and parameters than unstructured models, they have more 'degrees of freedom' when being tuned, and so it might be supposed that by varying their parameters they may be fitted to a wider range of data. There is also the possibility that models may be 'over-fitted' to the data used in tuning, thereby modelling both the fermentation behaviour and the noise on the measurements.

The average prediction errors from Tables 2.3 and 2.4 are all greater than the expected best possible prediction errors in Table 2.2. For the assumed percentage errors used in calculating the expected errors, this suggests that none of the models has been 'over-fitted' to the experimental data.

2.4.2 Comparison of models' performances

For comparison purposes, the models are divided into three groups. The unstructured models are divided into two groups, those which are related to the model of Bajpai and Reuß (Bajpai and Reuß, 1980; Nicolai *et al.*, 1991; Menezes *et al.*, 1994; Tiller *et al.*, 1994), and those which are unrelated (Fishman and Biryukov, 1974; Heijnen *et al.*, 1979; Kluge *et al.*, 1992). The morphologically structured models (Megee *et al.*, 1970; Nestaas and Wang, 1983; Cagney *et al.*, 1984; Paul and Thomas, 1996) are treated as a single group.

Unstructured models related to the model of Bajpai and Reuß (1980)

Graphs comparing the fermentation behaviour predicted by unstructured models related to the model of Bajpai and Reuß (1980) with the measured fermentation data are given in Figures 2.6 and 2.7.

The two models which do not have explicit biomass destruction terms (Bajpai and Reuß, 1980; Nicolai *et al.*, 1991) predict the biomass profile better than the other two, neither overestimating the biomass concentration at the end of the growth phase, nor predicting an excessive decline in the biomass concentration during the production phase.

The four models have different expressions describing the rate of sub-

strate consumption associated with biomass maintenance. The poorest performance in predicting the substrate profile is that of Tiller *et al.* (1994), which uses an age-dependent maintenance term. For low substrate concentrations, both the exponential form used by Nicolai *et al.* (1991) to describe the influence of endogenous maintenance, and the Monod form used for the maintenance term by Menezes *et al.* (1994) may represent the maintenance substrate consumption better than the linear form used in Bajpai and Reuß (1980).

The model of Menezes *et al.* (1994) differs from the others in that it uses a Monod kinetic to describe the penicillin production rate, as opposed to the substrate inhibition kinetic. This causes the predicted penicillin concentration to start increasing earlier than is observed for the measured values. Using an inhibition kinetic makes the penicillin production rate extremely sensitive to changes in the substrate concentration; this is the most likely explanation for the differing penicillin profiles generated by the models of Bajpai and Reuß (1980), Nicolai *et al.* (1991) and Tiller *et al.* (1994). (Tiller *et al.* (1994) relate the penicillin production rate to the specific growth rate as shown in Figure 2.2. This may be considered equivalent to a crude approximation to the substrate inhibition kinetic.)

The divergence between the measured and predicted penicillin concentrations for the model of Bajpai and Reuß (1980), for fermentation times greater than 110 hours, tuned on constant feed rate data and validated on time-varying data (Figure 2.6) may be due to small errors in the prediction of the associated glucose concentration. The penicillin production kinetic, $\mu_P SX/(K_P + S(1 + S/K_I))$, when plotted as a function of glucose concentra-

tion, has a narrow peak, and small errors in the predicted glucose concentration therefore cause disproportionate changes in the shape of the predicted penicillin profile.

Unstructured models not related to the model of Bajpai and Reuß (1980)

Graphs comparing the fermentation behaviour predicted by unstructured models not related to the model of Bajpai and Reuß (1980) with the measured fermentation data are given in Figures 2.8 and 2.9.

The model of Fishman and Biryukov (1974) models only 'actively growing' biomass. Here we have assumed that only growing hyphal tips are 'actively growing biomass'. The models of Heijnen *et al.* (1979) and Kluge *et al.* (1992) predict their respective biomass profiles reasonably well, whilst that of Fishman and Biryukov (1974) performs badly. This may be because the assumption that only hyphal tips are 'actively growing' is poor or due to the difficulties in relating the states of Fishman and Biryukov (1974) to the reference states of Paul and Thomas (1996).

The model of Fishman and Biryukov (1974) predicts the substrate concentration during the growth phase reasonably well, but in the production phase of the fermentation the model consistently predicts substrate values which are higher than the measured values. This is probably due to the presence of a postulated inhibitor in the model. The model of Kluge *et al.* (1992) predicts the substrate concentration in the growth phase better than does the model of Heijnen *et al.* (1979) when tuning on data obtained using a constant feed rate (Table 2.3), but worse when tuned on data obtained using a time-varying feed rate (Table 2.4).

The model of Heijnen *et al.* (1979) has a function describing the penicillin production rate which relates it to the specific growth rate (illustrated in Figure 2.1). This expression saturates with increasing specific growth rate, thus approximating a Monod type kinetic, in contrast with the more common substrate inhibition kinetic, which passes through a maximum. Using this form to describe the penicillin production rate results in a predicted penicillin concentration profile of the wrong form, increasing earlier than the measured values, and levelling off after the initial growth phase.

Structured models

Graphs comparing the fermentation behaviour predicted by morphologically structured models with the measured fermentation data are given in Figures 2.10 and 2.11.

Three of the four morphologically structured models considered here are similar in form (Megee *et al.*, 1970; Cagney *et al.*, 1984; Paul and Thomas, 1996). The other morphologically structured model (Nestaas and Wang, 1983) is fundamentally different in that its description of the fermentation is divided into two distinct portions–'growth' and 'growth and production'. All three similar models include at least three distinct biomass fractions: growing tips, productive hyphae (just 'behind' the tips), and degenerating material. The model of Megee *et al.* (1970) has three productive hyphal portions, two of which are associated with non-growth-associated products.

The model of Nestaas and Wang (1983) performs far worse in predicting fermentation behaviour than the other models. In this model, the biomass growth rate is described as being independent of the substrate concentration.

The rate of penicillin production, via a precursor, is described as being dependent solely on the biomass concentration. The constructed equation describing changes in the substrate concentration is derived from these expressions, and so is also independent of the substrate concentration. This means that the predictions made by this model only depend on the data for which it is originally tuned, and are, with the exception of the predicted substrate profile, independent of the input feed profile. The substrate concentration predicted by the model is calculated from feeding and consumption terms. Since the substrate concentration influences neither the biomass growth rate nor the product formation rate, the substrate consumption term is fixed when the model is tuned. Changes in the feed profile thus directly affect the predicted substrate concentration. Therefore, however well tuned this model may be for a given data set, it is not useful for control or optimisation.

The model of Megee *et al.* (1970) does not predict the biomass concentration as well as the other two. However, as this model has five biomass states, with many terms describing transitions between them, it is hard to identify any single explanation for this. It is possible that this model has a structure for which the parameters cannot be accurately identified using only the available measurements of the total biomass, substrate and penicillin concentrations.

The models of Cagney *et al.* (1984) and Paul and Thomas (1996) both make good predictions of the biomass profile. From Figures 2.10 and 2.11 it can be seen that the separation between the predictions of Paul and Thomas (1996) and Cagney *et al.* (1984) decreases for simulation times greater than ≈ 60 hours. In Figure 2.11 the predictions are seen to cross. This observation

may be due to the presence in the model of Paul and Thomas (1996) of a biomass lysis term, which that of Cagney *et al.* (1984) does not have.

In the model of Cagney *et al.* (1984), the overall growth rate of biomass is described by one term.

$$\sum_{n=0}^{2} \frac{\mathrm{d}X_n}{\mathrm{dt}} = \frac{\mu_1 X_0 S}{K_S + S}$$

Here X_n are the individual biomass states (tips, subapical fractions, degenerate regions) in the Cagney model, X_0 represents the hyphal tips, μ_1 is the specific growth rate, by extension, of the biomass, S is the substrate concentration, and K_S is a Monod-type constant. The above expression depends on X_0 and S. The rate of change of X_0 itself is given by the following.

$$\frac{\mathrm{d}X_0}{\mathrm{dt}} = \frac{\nu X_1 S}{K_S + S} - \frac{\zeta X_0}{\lambda + S}$$

Here X_1 is the concentration of the subapical fraction, ν is a branching coefficient, and ζ and λ are differentiation coefficients. For low values of S this expression becomes negative, resulting in decreasing X_0 and thus a reduction in the overall growth rate of biomass. Eventually, the overall growth rate of biomass may be reduced to less than the effect of the dilution, at which point the total biomass concentration will start to decrease. This point is not reached with the model as tuned for either of the two data sets considered here.

As degenerated hyphae, the state assumed to undergo lysis, is involved

in neither penicillin formation nor substrate consumption, lysis terms only directly influence the biomass concentration. The addition of a lysis term to this model would introduce a rate of decrease term to the expression describing the overall biomass concentration. This may improve the prediction of overall biomass concentration towards the end of the fermentation, and may also cause a change in the proportions of the other biomass states, possibly improving the model's substrate and penicillin predictions.

The main difference between the fermentation prediction errors of Cagney *et al.* (1984) and Paul and Thomas (1996), illustrated in Figures 2.10 and 2.11 and summarised in Tables 2.3 and 2.4, is in the substrate error. This is possibly due to the fact that the model of Paul and Thomas (1996) has two additional substrate consumption states, associated with maintenance of the hyphal tips and growth by extension of the productive hyphal state.

The poorer performance of the model of Megee *et al.* (1970) in predicting the substrate concentration is associated with the differences in biomass and penicillin production and also the fact that the model of Megee *et al.* (1970) does not have production-related substrate consumption.

Both Cagney *et al.* (1984) and Paul and Thomas (1996) perform better than Megee *et al.* (1970) in predicting the penicillin concentration. The model of Megee *et al.* (1970) uses Monod kinetics to describe the rates of formation of its two non-growth-associated products, assumed here to represent penicillin, whereas Cagney *et al.* (1984) and Paul and Thomas (1996) both use a form of substrate inhibited kinetic to describe the rate of penicillin production.

It is possible for a model using Monod kinetics to reproduce the observed

penicillin production behaviour, which is known to be substrate inhibited, if the concentration of the biomass states associated with product formation varies. How well such a model can predict the penicillin profile is constrained by the requirement that the model also predict the biomass profile with reasonable accuracy.

2.4.3 Quantitative comparison of the prediction errors

Examining the graphs of Figures 2.6 to 2.11 only enables us to make a crude comparison between the performance of the models. More detailed comparison may be made with reference to the tabulated errors given in Tables 2.3 and 2.4. These error values have been calculated in the same manner as the error values used in tuning the models, being weighted for each state with respect to the inverse of the maximum value measured for that state. This scheme was used in tuning because it was considered that using the absolute error values could result in a tuning that favoured those profiles with larger absolute values, thereby resulting in tunings which fitted the biomass profile well, and the penicillin profile poorly. The maximum values were chosen for weighting so as to avoid the divide-by-zero errors that would be caused if initial or final values were used for weighting, or if normalised errors were used instead of weighted errors, and to attempt to avoid the biasing that would be caused if average values were used.

Most of the models perform better in predicting the performance of a constant feed rate fermentation, having been tuned on fermentation data obtained using a time-varying feed rate. This may be explained by considering the time-varying data as passing through a wider range of fermentation con-

ditions, and thus providing richer information on which to tune the model. The change in performance of the model of Megee *et al.* (1970) is not considered to be significant. The reason for the model of Nestaas and Wang (1983) failing to make better predictions when tuned using time-varying feed rate data has been explained above. The model of Fishman and Biryukov (1974) may fail to perform better with time-varying feed rate data tuning because it is a particularly simple model, describing penicillin formation as being age-related, not including terms to describe substrate consumption associated with biomass maintenance or penicillin formation, and also not including a penicillin hydrolysis term. The largest change in the error due to an individual state for the model of Kluge *et al.* (1992), which also performs better when tuned with constant feed rate data, is that of the penicillin state. It seems that, although this model is capable of being tuned to match either of the two sets of fermentation data considered here, its penicillin production expression is largely unaffected by the data set used in calculating prediction errors. This may be because the penicillin production expression itself is not sufficiently sensitive, particularly to changes in the glucose concentration.

After considering the variation in performance due to changing the order in which the data sets were used in tuning and validation, we consider changes in the relative performance of the models. Apart from the models which perform more poorly when tuned on time-varying data, mentioned above, the models whose relative performance changes are those of Megee *et al.* (1970), Nicolai *et al.* (1991) and Menezes *et al.* (1994). The ranking of these three models is reversed when tuned using time-varying feed rate fermentation data, as opposed to constant feed rate fermentation data. This may be due

Ranking	Model Name	Error
1	Paul and Thomas	0.18
2	Cagney *et al.*	0.50
3(=)	Nicolai *et al.*	0.70
3(=)	Menezes *et al.*	0.70
5	Megee *et al.*	0.72
6	Kluge *et al.*	0.77
7	Bajpai and Reuß	0.95
8	Heijnen *et al.*	1.50
9	Tiller *et al.*	1.70
10	Fishman and Biryukov	2.20
11	Nestaas and Wang	170

Tab. 2.5: Average summed squared error in predicting fermentation performance

to changes between the two data sets in the sensitivities of the prediction errors to variation in model parameters.

Since how well a model predicts fermentation data depends on the data set used to tune the model, we have used the average overall prediction errors to determine which models perform better (see Table 2.5).

The two best models are both morphologically structured (Cagney *et al.*, 1984; Paul and Thomas, 1996) and make good predictions of the penicillin concentration. This is likely to be a consequence of their being able to relate penicillin production specifically to one fraction of the biomass, situated between the growing tips and older, degenerating portions of the hyphae. Unstructured models are incapable of associating penicillin production so closely to a portion of the biomass. The two unstructured models which predict most closely the penicillin concentration (Nicolai *et al.*, 1991; Menezes *et al.*, 1994) include terms which decrease the concentration of biomass associated with

penicillin production. In the model of Nicolai *et al.* (1991), this is a consequence of endogenous metabolism at low substrate concentrations, whereas the model of Menezes *et al.* (1994) includes a term to describe biomass death.

2.5 Conclusions

Morphologically structured models have some practical disadvantages when compared with the simpler unstructured models. The model of Paul and Thomas (1996), along with the other morphologically structured models, divides the biomass into a number of distinct states which ideally need to be determined directly. This implies that additional equipment, e.g. an image analyser or the filtration probe, is needed if the maximum benefit is to be derived from using such models. Morphologically structured models have a greater number of states than unstructured models and as a consequence are slower to simulate than unstructured models. Morphologically structured models also tend to have a larger number of parameters than unstructured models. Combined with the greater number of states of morphological models, this means that tuning their parameters takes longer than for unstructured models.

However, the better performance of the morphologically structured models suggests that their additional complexity has benefits in terms of predictive performance. The best performing morphologically structured model has an overall prediction error less than one third of that of the best unstructured model. If this could be translated into a corresponding improvement in fermentation control, this might well make up for the premium incurred in obtaining additional equipment, e.g. an image analyser for measuring

directly distinct biomass fractions.

As engineers we are interested in using differential equation based models in designing optimal feeding strategies for the penicillin fermentation and in developing improved methods for controlling the fermentation. To do this we need models which describe the fermentation well. The rest of this thesis builds upon the best performing morphologically structured model, that of Paul and Thomas (1996).

2.6 Notation

E_M	Endogenous maintenance coefficient, g(S)l^{-1}
E_P	Endogenous production coefficient, g(S)l^{-1}
F	Feed rate to fermenter, lh^{-1}
K_I	Inhibition coefficient, g(S)l^{-1}
K_P	Inhibition coefficient, g(S)l^{-1}
K_S	Monod coefficient for glucose, g(S)l^{-1}
K_X	Contois constant, g(S)g(DW)$^{-1}$
P	Concentration of penicillin, g(P)l^{-1}
Q_i	Flow rate into fermenter, lh^{-1}
Q_o	Flow rate out of fermenter, lh^{-1}
S	Concentration of glucose, g(S)l^{-1}
V	Volume of broth in fermenter, l
X	Concentration of biomass, g(DW)l^{-1}
X_*	Concentration of biomass fraction $*$, g(DW)l^{-1}
Z	Concentration of a model state, gl^{-1}
Z_{in}	Concentration of a model state fed to the fermenter, gl^{-1}
$_{meas}$	Subscript denoting measured value
$_{sim}$	Subscript denoting simulated value
m_s	Maintenance coefficient, g(S)g(DW)$^{-1}$h^{-1}
r_P	Rate of formation of penicillin, moles h^{-1}
r_{PO}	Maximum rate of formation of penicillin, moles h^{-1}
r	Rate of consumption of some species in the fermenter, gl^{-1}h^{-1}

t	Time, h
x	Concentration of some general species X, gl^{-1}
x_f	Concentration of some general species X fed to the fermenter, gl^{-1}
x_o	Concentration of some general species X leaving the fermenter, gl^{-1}

Greek Symbols

ζ	Degeneration numerator coefficient, $\mathrm{g(S)l}^{-1}\mathrm{h}^{-1}$
λ	Differentiation denominator coefficient, $\mathrm{g(S)l}^{-1}$
μ	Specific growth rate, h^{-1}
μ_P	Penicillin production constant, h^{-1}
μ_X	Growth constant h^{-1}
μ_{p1}	Minimum specific growth rate associated with maximum rate of penicillin production, h^{-1}
μ_{p2}	Maximum specific growth rate associated with maximum rate of penicillin production, h^{-1}
μ_1	Growth rate, h^{-1}
ν	Branching numerator coefficient, h^{-1}
π	Penicillin production rate, $\mathrm{g(P)g(DW)}^{-1}\mathrm{h}^{-1}$

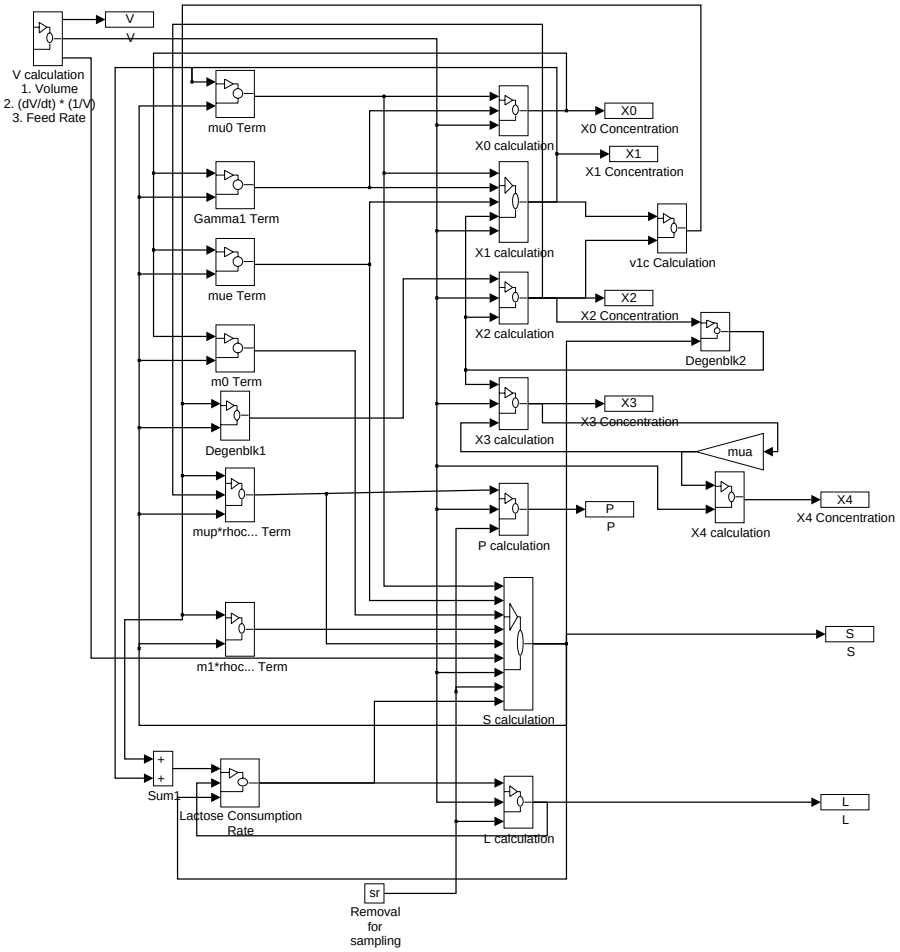

Fig. 2.4: SIMULINK block diagram for the model of Paul *et al.* (1998)

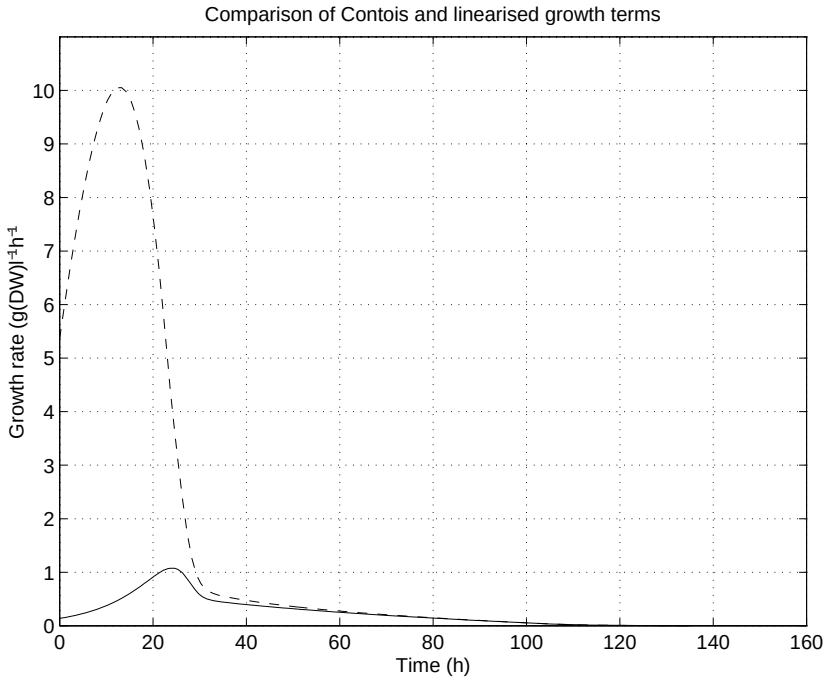

Fig. 2.5: Comparison of nonlinear and linearised expressions for specific growth rate (– nonlinear, $-\cdot$ linear) The nonlinear growth rate was taken from the model of Bajpai and Reuß, tuned for time-varying feed data. The linearised growth curve was calculated using the predicted substrate concentration from the tuned model. ($\mu_X \approx 0.10$, $K_X \approx 0.16$)

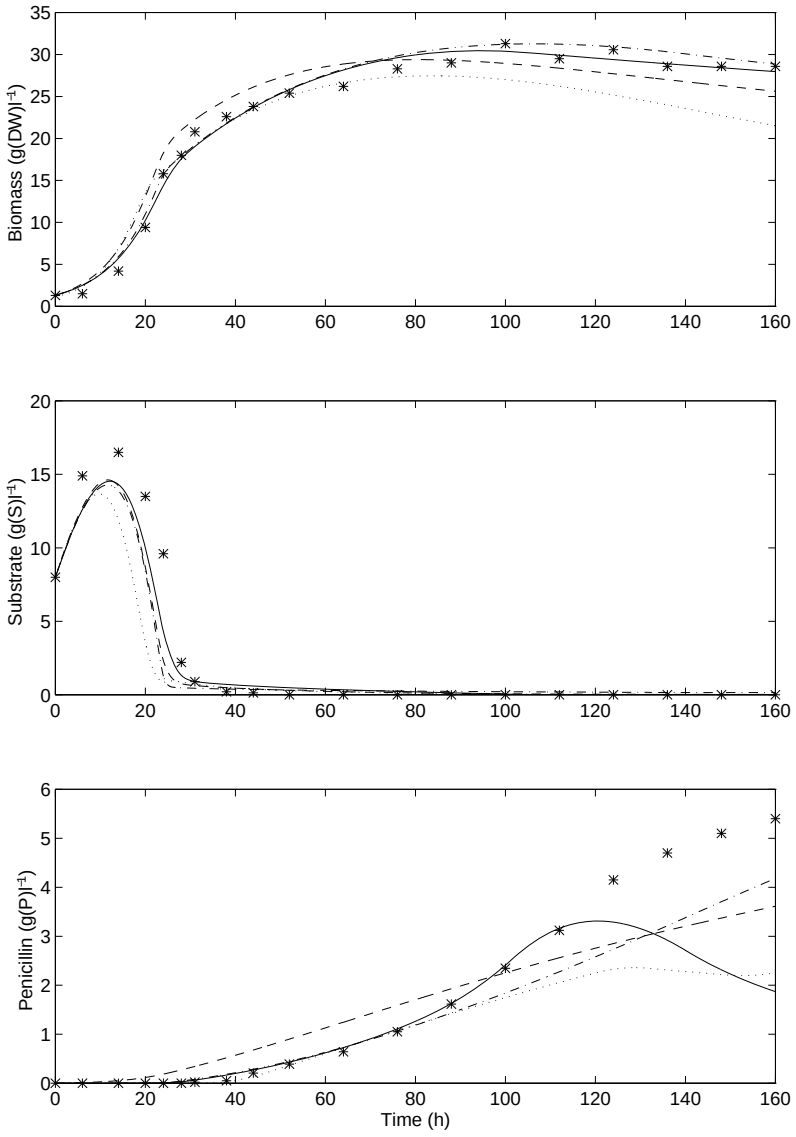

Fig. 2.6: Predicted and measured concentrations for unstructured models related to the model of Bajpai and Reuß tuned for constant feed rate fermentation data and validated against time-varying feed rate fermentation data ($*$ Measured data, — Bajpai and Reuß, - - Menezes *et al.*, $-\cdot$ Nicolai *et al.*, $\cdots$ Tiller *et al.*)

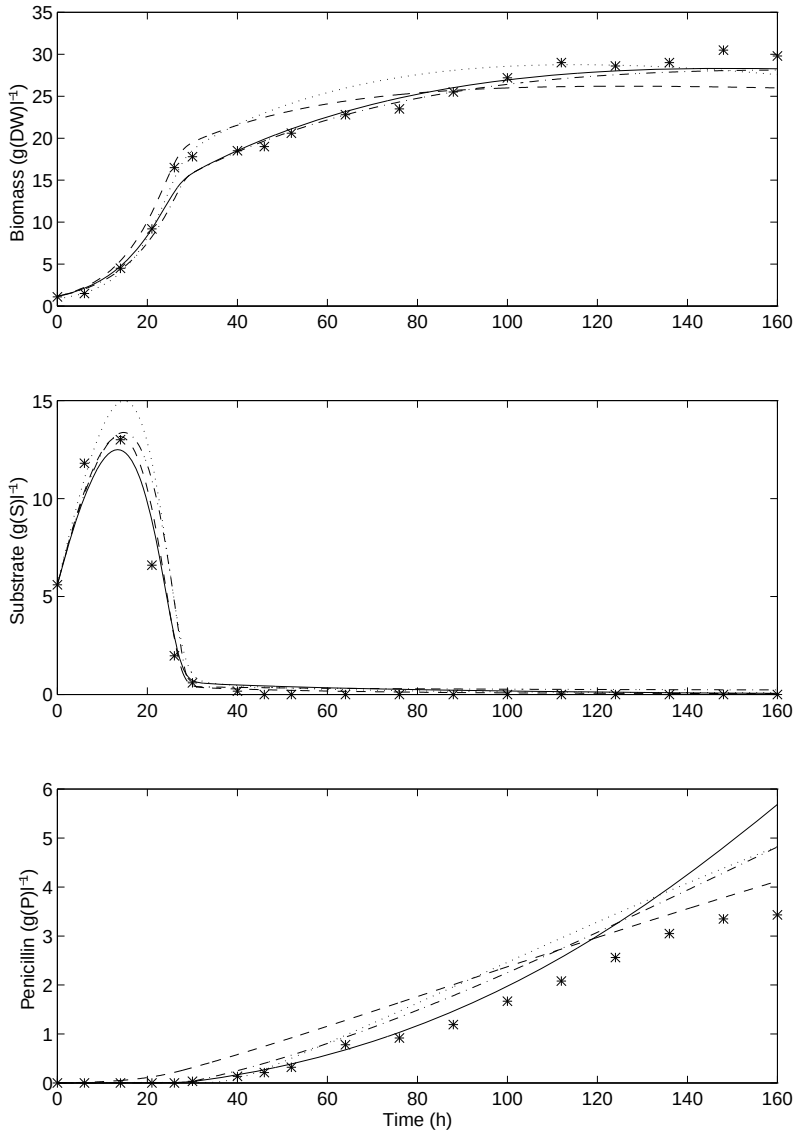

Fig. 2.7: Predicted and measured concentrations for unstructured models related to the model of Bajpai and Reuß tuned for time-varying feed rate fermentation data and validated against constant feed rate fermentation data
($*$ Measured data, — Bajpai and Reuß, - - Menezes *et al.*, $-\cdot$ Nicolai *et al.*, $\cdots$ Tiller *et al.*)

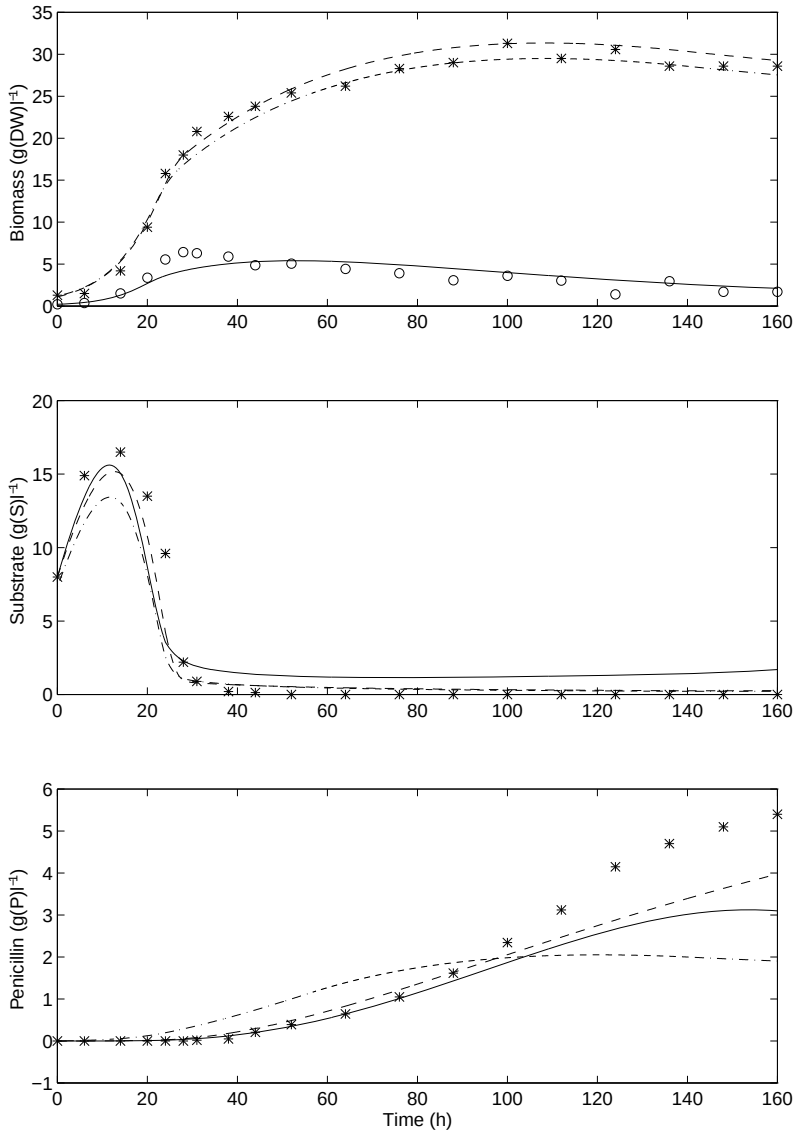

Fig. 2.8: Predicted and measured concentrations for unstructured models not related to the model of Bajpai and Reuß tuned for constant feed rate fermentation data and validated against time-varying feed rate fermentation data
($*$ Measured data, — Fishman and Biryukov, - - Kluge *et al.*, $-\cdot$ Heijnen *et al.*, $\circ$ 'actively growing biomass' (Fishman and Biryukov))

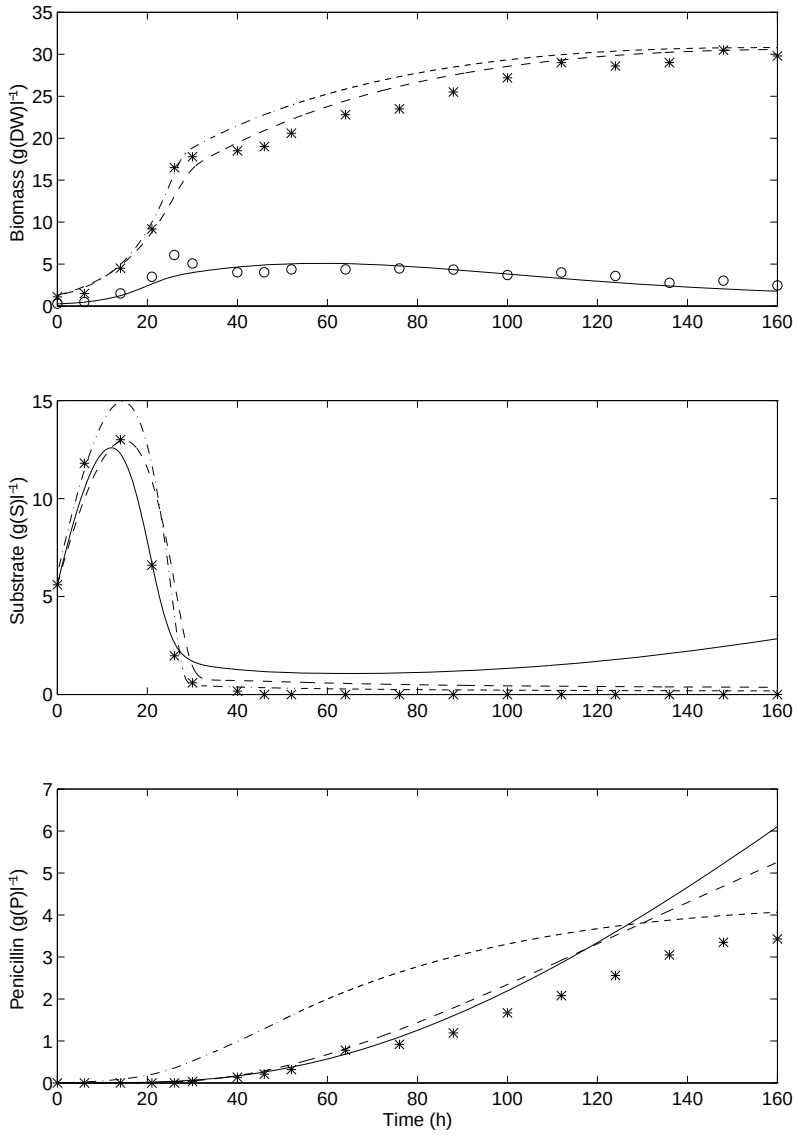

Fig. 2.9: Predicted and measured concentrations for unstructured models not related to the model of Bajpai and Reuß tuned for time-varying feed rate fermentation data and validated against constant feed rate fermentation data
($*$ Measured data, — Fishman and Biryukov, - - Kluge *et al.*, $-\cdot$ Heijnen *et al.*, $\circ$ 'actively growing biomass' (Fishman and Biryukov))

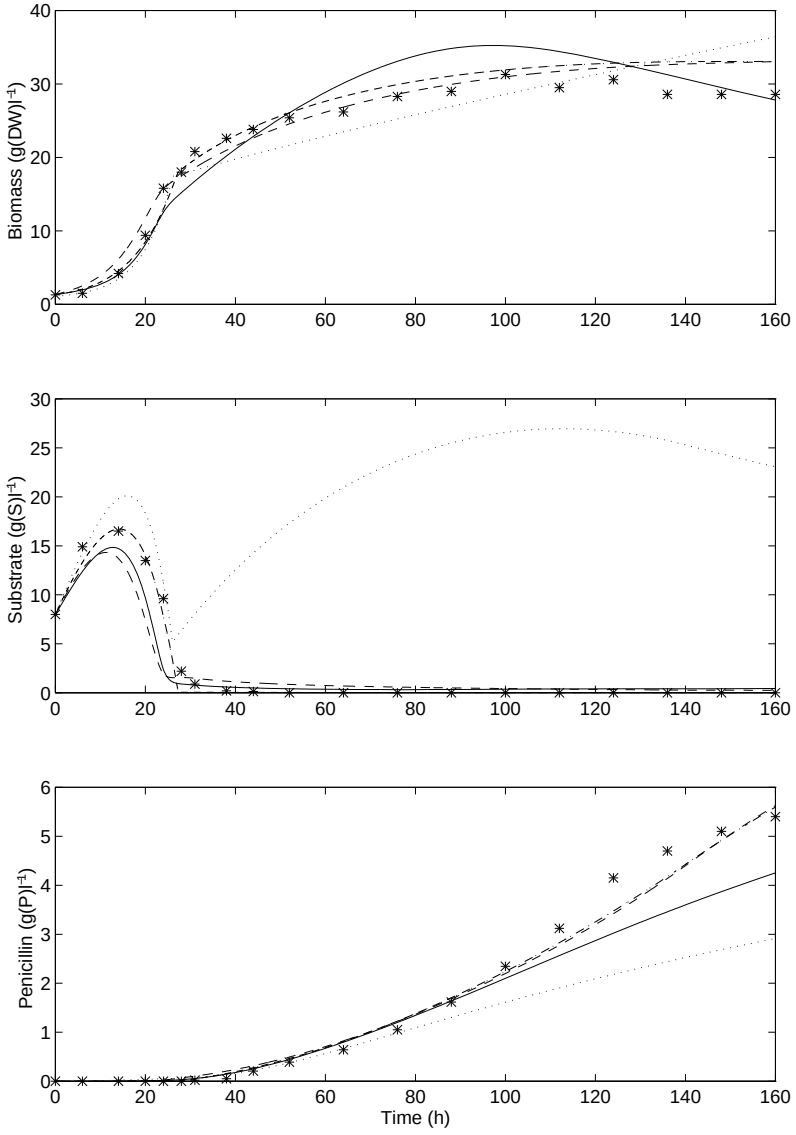

Fig. 2.10: Predicted and measured concentrations for morphologically structured models tuned for constant feed rate fermentation data and validated against constant feed rate fermentation data
($*$ Measured data, — Megee *et al.*, - - Cagney *et al.*, $-\cdot$ Paul and Thomas, $\cdots$ Nestaas and Wang)

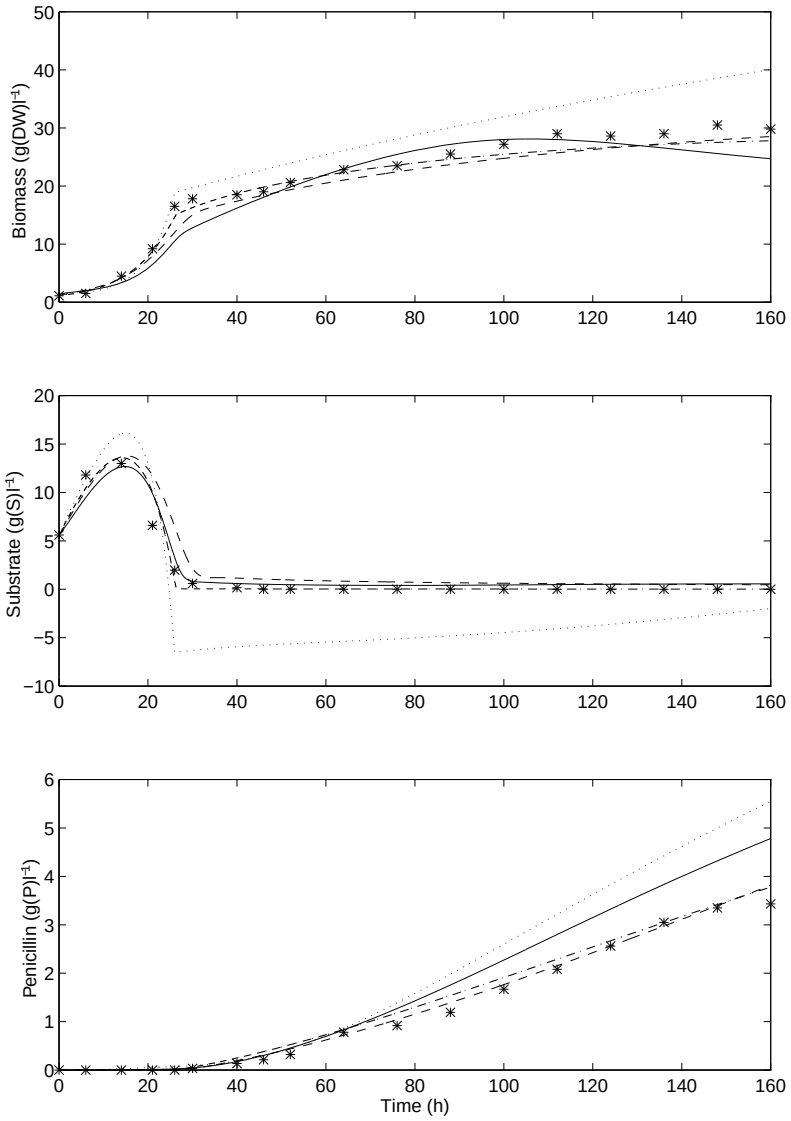

Fig. 2.11: Predicted and measured concentrations for morphologically structured models tuned for time-varying feed rate fermentation data and validated against constant feed rate fermentation data
($*$ Measured data, — Megee *et al.*, - - Cagney *et al.*, $-\cdot$ Paul and Thomas, $\cdots$ Nestaas and Wang)

3. SIMPLIFICATIONS AND EXTENSIONS TO THE PAUL AND THOMAS MODEL

3.1 Simplifying the Vacuolation Process Model

As shown in the preceding chapter, the penicillin fermentation model of Paul and Thomas (1996) is the best performing model of those considered in this thesis in predicting the behaviour of penicillin producing fermentations of *Penicillium chrysogenum*. It is a morphologically structured model which divides the biomass up into a number of distinct states:

- hyphal tips, the actively growing area of the hyphae (Region X_0)
- non-growing region of the hyphae, the region of the hyphae just behind the tips (Region X_1)
- growing vacuoles, divided by size into a number of 'bins' (Region X_2)
- fully vacuolated hyphae, in which the vacuoles have grown to fill the hyphal compartments (Region X_3)
- lysed material, formed by the destruction of fully vacuolated hyphal compartments (This model state is difficult to measure directly.) (Region X_4)

However, this model is extremely complex and could prove difficult to use in controller design or as a part of some kind of hybrid differential equation/neural network based model scheme. Most of this complexity is due to the way in which the vacuole formation and growth processes are described in the model. The distribution of vacuole sizes changes as the fermentation proceeds, with small vacuoles being formed and growing until hyphal compartments become completely vacuolated, at which point the vacuole is considered to have given rise to a degenerated hyphal compartment and ceases to be regarded as vacuole. In the model, the vacuole size distribution is discretised, with the vacuoles being divided into a number of 'bins', which correspond to distinct, non-overlapping size ranges. The number of vacuoles in each size range is represented by a model state. The rates of change of these states have relatively high values, which could cause the model as a whole to be numerically 'stiff', needing a more specialised numerical integration routine to solve it accurately. As a result of the additional states and the fact that the system is 'stiff', the original model is slow to simulate.

Here ways are considered in which the vacuolation part of the model could be replaced, or removed, with the aim of increasing the speed with which the system can be simulated, without, one hopes, too great a loss in performance as a result of using a simplified model.

3.1.1 Simplifications considered

The processes described in the vacuolation portion of the model of Paul and Thomas (1996) are shown in Figure 3.1. Vacuoles (X2) form in the non-growing region (X1), grow, and eventually reach a size where whole hyphal

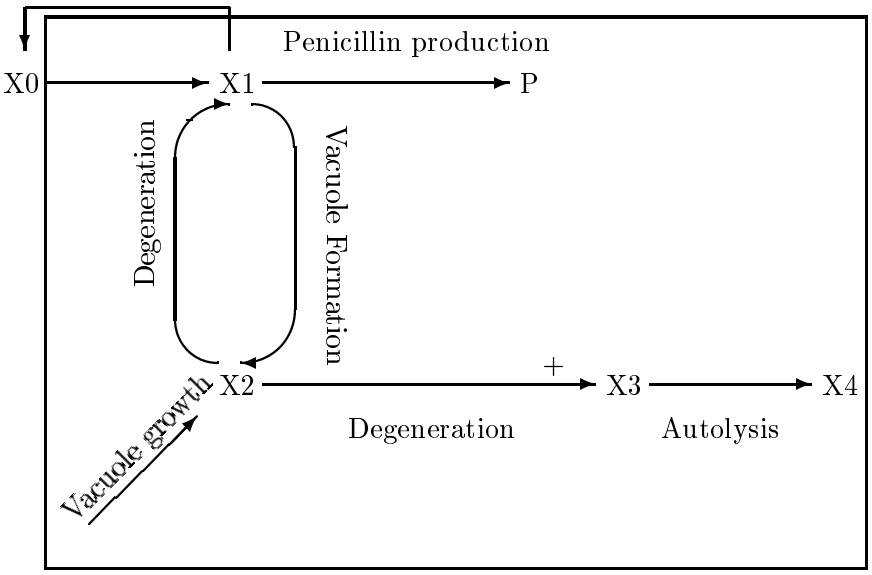

Fig. 3.1: Diagram showing the vacuolation process as modelled in Paul and Thomas (1996). The boxed area indicates those parts of the model affected by the model simplification.

compartments are vacuolated (X3). At this point, the vacuoles are regarded as having given rise to fully vacuolated hyphal compartments. In the model of Paul and Thomas (1996), penicillin production is associated with the volume of cytoplasm present in the non-growing region. The process of vacuolation therefore reduces the volume of penicillin-producing cytoplasm.

An attempt has been made to replace the existing description of the vacuolation process with much simpler terms, similar to those used elsewhere in penicillin fermentation models. Two possible candidate structures were considered.

- A structure in which material in the non-growing region gives rise to partially vacuolated material, which then gives rise to fully vacuolated

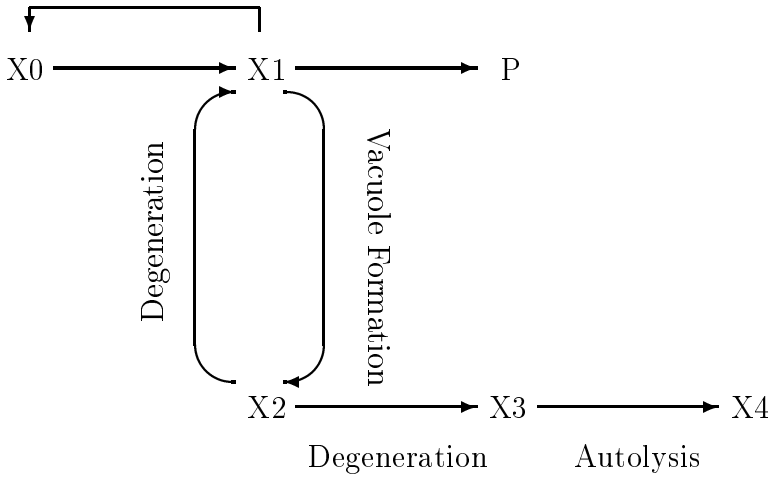

Fig. 3.2: Conversion from non-growing hyphae to fully vacuolated hyphae via an intermediate, partially vacuolated state

hyphal material (see Figure 3.2).

- A structure in which material in the non-growing region is considered as passing directly to fully vacuolated hyphal material (see Figure 3.3).

Both these structures are intended to describe only the observed gross changes in the hyphae and make no attempt to describe the mechanisms by which vacuole formation and growth take place.

Three possible types of kinetic were considered for use in describing each step in each of the candidate structures.

- a first order kinetic, kX, (F)

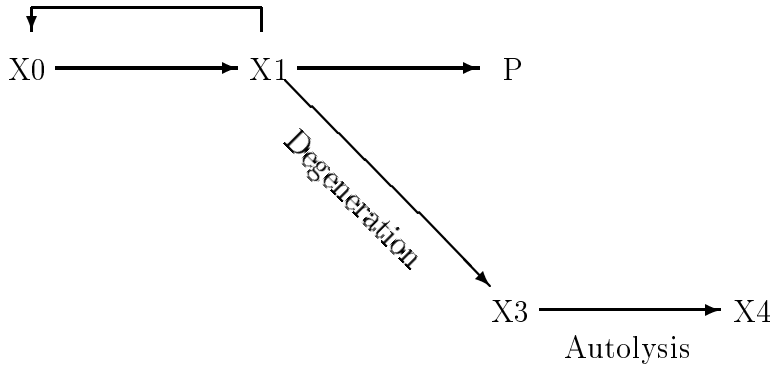

Fig. 3.3: Conversion directly from non-growing hyphae to fully vacuolated hyphae

- a conversion kinetic, $\frac{kX}{L+S}$, (C)
- a substrate inhibition kinetic, $\frac{kXS}{L+S+\frac{S^2}{M}}$, (I)

For each candidate structure, two alternative forms, with and without degeneration of the fully vacuolated hyphae (assumed to be first order), were considered.

This means that, in total, 24 different candidate model structures were tuned and compared (6 single-step models and 18 two-step models). However, it was only necessary to construct two general model structures, one for single-step models and one for two-step models. The three kinetics being considered for each step were constructed in parallel, with software 'flags' being used to determine which kinetic was operating (in any group of three parallel kinetics). This also made it possible to automate the tuning of the candidate models, by writing scripts to go through the set of candidate models sequentially, tuning each in turn and saving the resulting parameter sets to files.

The following short forms are used to refer to the candidate model structures:

- single-step models describing a transition directly from state X_1 to state X_3, are denoted by $1 \overset{F,C,I}{\rightarrow} 3$, where the transition kinetic used in the model is indicated by the superscript on the arrow, being one of the three possibilities, First order (F), Conversion (C) or Inhibited (I).

- two-step models describing transition from state X_1 to state X_2 and thence to state X_3 are denoted by $1 \overset{F,C,I}{\rightarrow} 2 \overset{F,C,I}{\rightarrow} 3$, where the superscripts on the arrows denote the transition kinetics used in the model, from the three options considered, First order (F), Conversion (C) and Inhibited (I).

Where the destruction of fully vacuolated hyphal compartments (biomass state X_3) has been modelled, the words "(with lysis)" are appended to the above short forms.

Consequential modifications

In the original model of Paul and Thomas (1996), the rates of penicillin production and of the maintenance-related substrate consumption terms are dependent on the volume concentration of active cytoplasm. In the simplified models, this dependence on the volume concentration of active cytoplasm has been replaced by dependence on the mass concentration of the non-growing regions of the hyphae. The equations defining the models considered here are given in Appendix A.3.

3.1.2 Comparing the simplified models

Each of the simplified model structures was built and tuned against fermentation data supplied by Paul (1996). Two sets of data were used, both having been obtained under the same conditions of fermenter scale and medium composition, but with different initial fermentation conditions and feed profiles being used. One set of data was obtained for a fermentation carried out with constant feed rate, and the second for a fermentation carried out with a time-varying feed rate. These data sets were the same as those used in tuning models from the published literature in the course of selecting the best performing penicillin fermentation model (see Figure 2.3 for details).

Each model was tuned using the first set of fermentation data and used to predict the performance of the fermentation for the second set of fermentation data, and *vice versa*. Because model tunings depend on the data used in tuning the model, the models were compared on the basis of how well they predicted the measured fermentation data. For the model tunings, all parameters in any given candidate model structure were optimised, with tuning being carried out using as many of the measured model states as was appropriate for the type of model (all biomass states, glucose and penicillin concentrations for two-step models, omitting only the vacuolated biomass state when considering single-step models).

The models were built using MATLAB and SIMULINK, and tuned using routines from the MATLAB Optimisation toolbox. A least squares routine (Levenberg-Marquardt algorithm) was used to tune each model's parameters, with the target error function being calculated as follows:

- Simulate the model over the time period of the reference data set,

using a fourth order Runge-Kutta algorithm or Gear's algorithm, both supplied in SIMULINK.

- Log the model output to obtain simulated values corresponding to the times of the experimental measurements.
  - Single-step models were tuned to fit the biomass states X_0, X_1 and X_3 (omitting the vacuolated state), along with the glucose and penicillin concentrations.
  - Two-step models were tuned to fit the biomass states X_0, X_1, X_2 and X_3 (all the biomass states), along with the glucose and penicillin concentrations.
- Calculate the difference between the measured and simulated values.
- Weight the differences for each model state by the inverse of the maximum value in the measured data set for that state.
- Square and sum the weighted differences. (This is done by the optimisation routine–it works on the matrix of weighted differences.)

Mathematically, the target function can be expressed as follows,

$$\text{Error} = \sum_{i=1}^{n} \sum_{all\ measured\ values} \left(\frac{value_{meas(i)} - value_{sim(i)}}{\max(value_{meas})} \right)^2 \tag{3.1}$$

where the summation is carried out for all measurement times and the subscripts *meas* and *sim* denote measured and simulated values respectively. The above equation indicates that the error expression contains contributions

from the biomass, substrate and product states modelled. For the case of single-step models, this means that summation is carried out for X_0, X_1, X_3, S and P, whilst for two-step models, summation is carried out for X_0, X_1, X_2, X_3, S and P, that is, with an additional measured biomass state, X_2.

3.1.3 Results – Single step models

The results presented here are for predicting the behaviour of a fermentation whose data have not been used in tuning the model and so obtaining a set of model parameters. Two summary tables of errors are shown (Tables 3.1 and 3.2). The tabulated data include summed squared error values for all five model states considered, three biomass states (vacuoles are ignored in the single-step models), glucose and penicillin, along with a total error value, formed by adding the entries in each row of the table. The summary tables of errors are given to four decimal places, as this is the standard format generated from MATLAB, and thus the easiest to obtain directly for inclusion in this document. The average prediction errors, provided as a summary of the calculated errors, may be found in Table 3.3. The average prediction errors were calculated by hand and are only given to two decimal places. This is sufficient for comparison between the various candidate model structures.

The best performing single-step model is that in which the conversion of X_1 to X_3 is described by an inhibited kinetic, with X_3 being considered to undergo first order degeneration (degeneration was only considered as being first order). The graphs associated with the best averaged fits to the measured fermentation data are given in Figures 3.4 and 3.5.

Model Structure	X0	X1	X3	S	P	Total
$1 \xrightarrow{F} 3$	0.3765	0.0502	0.6337	0.0480	0.0775	1.1859
$1 \xrightarrow{F} 3$ (w.l)	0.3765	0.0502	0.6337	0.0480	0.0775	1.1859
$1 \xrightarrow{C} 3$	0.3166	0.0255	0.6355	0.2146	0.4369	1.6291
$1 \xrightarrow{C} 3$ (w.l)	0.3351	0.0729	0.5829	0.1753	0.1347	1.3010
$1 \xrightarrow{I} 3$	0.2460	0.2492	0.6265	0.1095	0.0845	1.3156
$1 \xrightarrow{I} 3$ (w.l)	0.2739	0.2021	0.4834	0.1713	0.0761	1.2068

Tab. 3.1: Prediction errors for single-step models tuned on constant feed profile data, predicting time-varying feed profile data [(w.l.) denotes models with lysis considered]

The biomass fractions make the largest contributions to the summed squared error (well over half the total summed squared error value), with X_3 making the largest contribution of all the biomass states. Since X_3 does not influence growth, substrate consumption, or product formation, however, the magnitude of this error is relatively unimportant, and may even be a consequence of the lack of influence of X_3 on other states' errors. The best-performing single-step model's poor performance in fitting X_3, shown in Figures 3.4 and 3.5, is, therefore, unimportant.

Model Structure	X0	X1	X3	S	P	Total
$1 \xrightarrow{F} 3$	0.2042	0.1135	0.6292	0.0658	0.0577	1.0704
$1 \xrightarrow{F} 3$ (w.l.)	0.2106	0.0991	0.6444	0.0733	0.0760	1.1035
$1 \xrightarrow{C} 3$	0.2825	0.0718	0.7111	0.0609	0.4077	1.5340
$1 \xrightarrow{C} 3$ (w.l.)	0.2027	0.1738	0.6404	0.0954	0.1203	1.2326
$1 \xrightarrow{I} 3$	0.2937	0.1386	0.4266	0.2032	0.0973	1.1594
$1 \xrightarrow{I} 3$ (w.l.)	0.3112	0.1015	0.3048	0.1614	0.0512	0.9301

Tab. 3.2: Prediction errors for single-step models tuned on time-varying feed profile data, predicting constant feed profile data [(w.l.) denotes models with lysis considered]

Model Structure	Total
$1 \xrightarrow{F} 3$	1.13
$1 \xrightarrow{F} 3$ (with lysis)	1.14
$1 \xrightarrow{C} 3$	1.58
$1 \xrightarrow{C} 3$ (with lysis)	1.27
$1 \xrightarrow{I} 3$	1.24
$1 \xrightarrow{I} 3$ (with lysis)	1.07

Tab. 3.3: Average total prediction errors for the single-step models

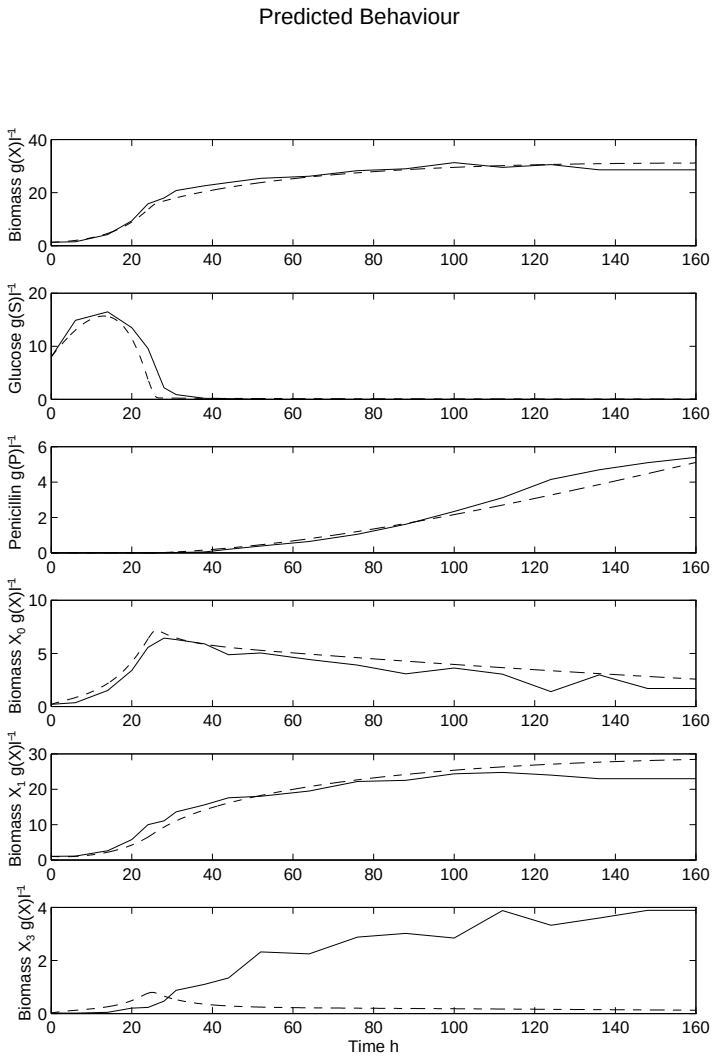

Fig. 3.4: Comparing predictions of data obtained using a constant feed rate, for models tuned with data obtained using a time-varying feed rate, validated against data obtained using a constant feed rate.
Model $1 \xrightarrow{I} 3$ (with lysis), (– Measured data, $-\cdot$ simple model)

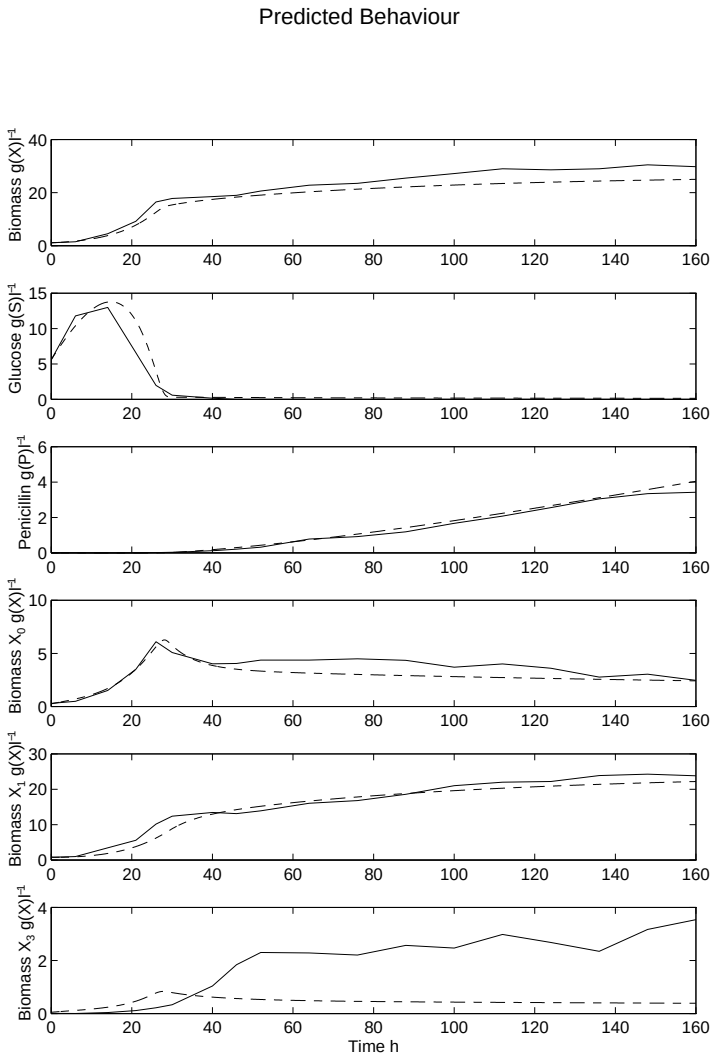

Fig. 3.5: Comparing predictions of data obtained using a time-varying feed rate, for models tuned with data obtained using a constant feed rate, validated against data obtained using a time-varying feed rate.
Model $1 \xrightarrow{I} 3$ (with lysis), (– Measured data, $-\cdot$ simple model)

3.1.4 Results – Two step models

The results presented here are for predicting the behaviour of a fermentation whose data have not been used in tuning the model and so obtaining a set of model parameters. Two summary tables of errors are shown in Tables 3.4 and 3.5, with the associated graphs being given in Figures 3.6 and 3.7.

The tabulated data include summed squared error values for all five model states considered, all four biomass states, glucose and penicillin, along with a total error value, formed by adding the entries in each row of the table. The average prediction errors, provided as a summary of the calculated errors, may be found in Table 3.6.

The best performing two-step model is that in which the conversion of X_1 to X_2 and of X_2 to X_3 are both described by first order kinetics, with X_3 being considered to undergo first order degeneration (degeneration was only considered as being first order).

Again, the biomass fractions make the largest contributions to the summed squared error (well over half the total summed squared error value), with the degenerated biomass state (X_3) often making the largest contribution of all the biomass states. Since X_3 does not influence growth, substrate consumption, or product formation, however, the magnitude of this error is relatively unimportant, and may even be a consequence of the lack of influence of X_3 on other states' errors. The best-performing two-step model's poor performance in fitting X_3, shown in Figures 3.6 and 3.7, may, therefore, be considered to be unimportant.

Model Structure	X0	X1	X2	X3	S	P	Total
$1 \xrightarrow{F} 2 \xrightarrow{F} 3$	0.2460	0.1289	0.2566	0.6900	0.1807	0.0605	1.5627
$1 \xrightarrow{F} 2 \xrightarrow{F} 3$ (w.l.)	0.2421	0.1497	0.2505	0.6805	0.1766	0.0492	1.5486
$1 \xrightarrow{F} 2 \xrightarrow{C} 3$	0.3717	0.1632	0.2369	0.6988	0.1749	1.3358	2.9814
$1 \xrightarrow{F} 2 \xrightarrow{C} 3$ (w.l.)	0.1985	0.1910	0.3944	0.7238	0.1451	0.0574	1.7102
$1 \xrightarrow{F} 2 \xrightarrow{I} 3$	0.1522	0.0992	4.7372	4.5880	0.1309	0.1087	9.8161
$1 \xrightarrow{F} 2 \xrightarrow{I} 3$ (w.l.)	0.2373	0.1836	1.6924	7.1470	0.1877	0.0438	9.4917
$1 \xrightarrow{C} 2 \xrightarrow{F} 3$	0.2387	0.1279	0.3423	0.7438	0.2203	0.0558	1.7287
$1 \xrightarrow{C} 2 \xrightarrow{F} 3$ (w.l.)	0.2330	0.1640	0.3451	0.7364	0.1981	0.0442	1.7207
$1 \xrightarrow{C} 2 \xrightarrow{C} 3$	0.2381	0.0983	0.3271	0.7600	0.1980	0.0627	1.6843
$1 \xrightarrow{C} 2 \xrightarrow{C} 3$ (w.l.)	0.0761	0.1198	3.5298	5.4979	0.1332	0.0452	9.4019
$1 \xrightarrow{C} 2 \xrightarrow{I} 3$	0.2051	0.1426	1.8380	7.1470	0.2235	0.0438	9.6000
$1 \xrightarrow{C} 2 \xrightarrow{I} 3$ (w.l.)	0.1866	0.2244	2.2195	7.1031	0.2385	0.0640	10.0360
$1 \xrightarrow{I} 2 \xrightarrow{F} 3$	0.1899	0.1956	8.6188	1.9179	0.2594	0.1403	11.3220
$1 \xrightarrow{I} 2 \xrightarrow{F} 3$ (w.l.)	0.3370	0.2634	8.0459	1.0486	0.0769	0.2312	10.0031
$1 \xrightarrow{I} 2 \xrightarrow{C} 3$	0.3283	0.2048	4.4607	1.1815	0.3758	0.4470	6.9980
$1 \xrightarrow{I} 2 \xrightarrow{C} 3$ (w.l.)	0.2198	0.2864	4.2140	3.4513	0.4126	0.2757	8.8597
$1 \xrightarrow{I} 2 \xrightarrow{I} 3$	0.4497	0.0888	0.8219	1.9767	0.1058	0.0797	3.5226
$1 \xrightarrow{I} 2 \xrightarrow{I} 3$ (w.l.)	0.3375	0.1031	0.1379	0.6385	0.2332	0.6807	2.1309

Tab. 3.4: Prediction errors for two-step models tuned on constant feed profile data, predicting time-varying feed profile data [(w.l.) denotes models with lysis considered]

Model Structure	X0	X1	X2	X3	S	P	Total
$1 \xrightarrow{F} 2 \xrightarrow{F} 3$	0.1360	0.2418	0.2091	0.7450	0.0385	0.0315	1.4020
$1 \xrightarrow{F} 2 \xrightarrow{F} 3$ (w.l.)	0.1168	0.1844	0.2317	0.7681	0.0381	0.0271	1.3662
$1 \xrightarrow{F} 2 \xrightarrow{C} 3$	0.1505	0.3012	0.7620	0.8976	0.0377	0.0489	2.1979
$1 \xrightarrow{F} 2 \xrightarrow{C} 3$ (w.l.)	0.1058	0.1476	0.4225	0.9609	0.0381	0.0200	1.6949
$1 \xrightarrow{F} 2 \xrightarrow{I} 3$	0.3930	0.2967	1.6173	5.8882	0.0415	0.0439	8.2806
$1 \xrightarrow{F} 2 \xrightarrow{I} 3$ (w.l.)	0.2917	0.3019	1.6083	5.8945	0.0592	0.0800	8.2357
$1 \xrightarrow{C} 2 \xrightarrow{F} 3$	0.1161	0.1925	0.2580	0.7612	0.0388	0.0199	1.3865
$1 \xrightarrow{C} 2 \xrightarrow{F} 3$ (w.l.)	0.1644	0.3042	0.1945	0.8911	0.0423	0.0574	1.6538
$1 \xrightarrow{C} 2 \xrightarrow{C} 3$	0.3033	0.2656	0.7155	3.5088	0.0454	0.1720	5.0105
$1 \xrightarrow{C} 2 \xrightarrow{C} 3$ (w.l.)	0.2991	0.1719	0.5477	1.3786	0.0548	0.0562	2.5084
$1 \xrightarrow{C} 2 \xrightarrow{I} 3$	0.3942	0.2311	1.7672	5.8542	0.0397	0.0643	8.3509
$1 \xrightarrow{C} 2 \xrightarrow{I} 3$ (w.l.)	0.1166	0.1244	1.7036	5.9443	0.0377	0.0185	7.9451
$1 \xrightarrow{I} 2 \xrightarrow{F} 3$	0.7691	0.4633	1.0677	0.5190	0.0573	0.0459	2.9222
$1 \xrightarrow{I} 2 \xrightarrow{F} 3$ (w.l.)	0.0951	0.0659	7.5075	1.5162	0.0609	0.0319	9.2773
$1 \xrightarrow{I} 2 \xrightarrow{C} 3$	0.4046	0.2285	0.7892	5.9075	0.1287	0.0375	7.4960
$1 \xrightarrow{I} 2 \xrightarrow{C} 3$ (w.l.)	0.3052	0.0738	0.6713	0.5210	0.0874	0.0790	1.7378
$1 \xrightarrow{I} 2 \xrightarrow{I} 3$	0.5512	0.2250	0.5307	2.3624	0.6775	0.0759	4.4227
$1 \xrightarrow{I} 2 \xrightarrow{I} 3$ (w.l.)	0.4055	0.1464	1.1847	1.1647	0.2661	0.0201	3.1874

Tab. 3.5: Prediction errors for two-step models tuned on time-varying feed profile data, predicting constant feed profile data [(w.l.) denotes models with lysis considered]

Model Name	Total
The Original Model	1.06
Two Step Models	
$1 \xrightarrow{F} 2 \xrightarrow{F} 3$	1.48
$1 \xrightarrow{F} 2 \xrightarrow{F} 3$ (w.l.)	1.46
$1 \xrightarrow{F} 2 \xrightarrow{C} 3$	2.59
$1 \xrightarrow{F} 2 \xrightarrow{C} 3$ (w.l.)	1.70
$1 \xrightarrow{F} 2 \xrightarrow{I} 3$	9.05
$1 \xrightarrow{F} 2 \xrightarrow{I} 3$ (w.l.)	8.86
$1 \xrightarrow{C} 2 \xrightarrow{F} 3$	1.56
$1 \xrightarrow{C} 2 \xrightarrow{F} 3$ (w.l.)	1.69
$1 \xrightarrow{C} 2 \xrightarrow{C} 3$	3.35
$1 \xrightarrow{C} 2 \xrightarrow{C} 3$ (w.l.)	5.96
$1 \xrightarrow{C} 2 \xrightarrow{I} 3$	8.98
$1 \xrightarrow{C} 2 \xrightarrow{I} 3$ (w.l.)	8.99
$1 \xrightarrow{I} 2 \xrightarrow{F} 3$	7.12
$1 \xrightarrow{I} 2 \xrightarrow{F} 3$ (w.l.)	10.30
$1 \xrightarrow{I} 2 \xrightarrow{C} 3$	7.25
$1 \xrightarrow{I} 2 \xrightarrow{C} 3$ (w.l.)	5.30
$1 \xrightarrow{I} 2 \xrightarrow{I} 3$	3.97
$1 \xrightarrow{I} 2 \xrightarrow{I} 3$ (w.l.)	2.66

Tab. 3.6: Average total prediction errors for the two-step models [(w.l.) denotes models with lysis considered]

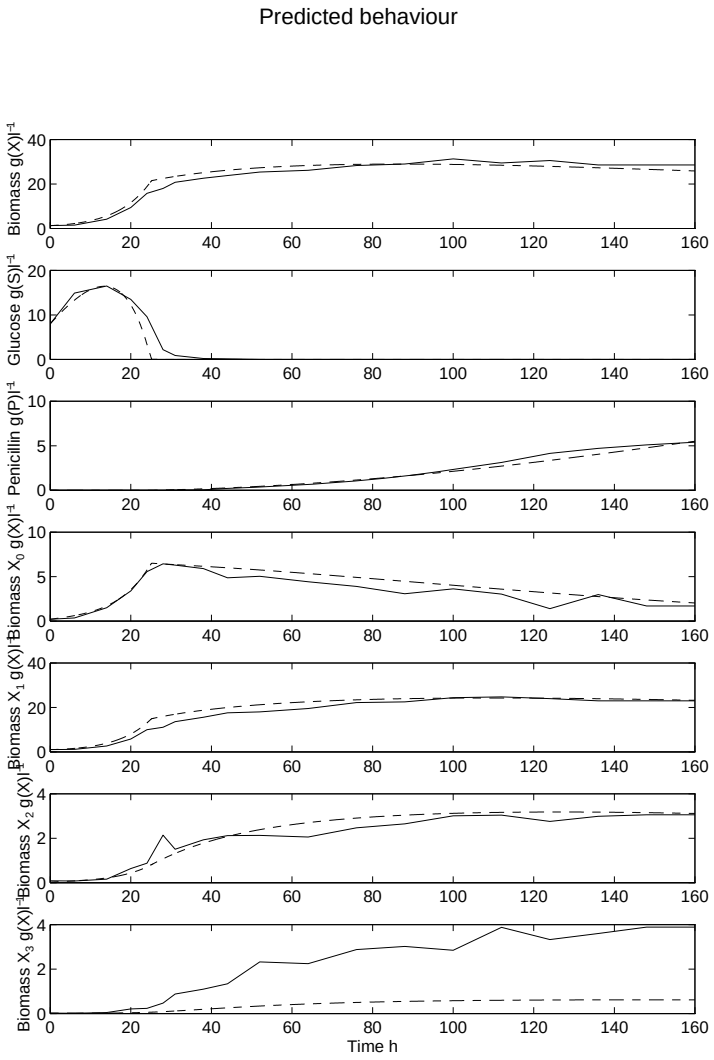

Fig. 3.6: Comparing predictions of data obtained using a time-varying feed rate, for models tuned with data obtained using a constant feed rate, validated against data obtained using a time-varying feed rate.
Model 1 $\stackrel{F}{\rightarrow}$ 2 $\stackrel{F}{\rightarrow}$ 3 with lysis, (– Measured data, $-\cdot$ simple model)

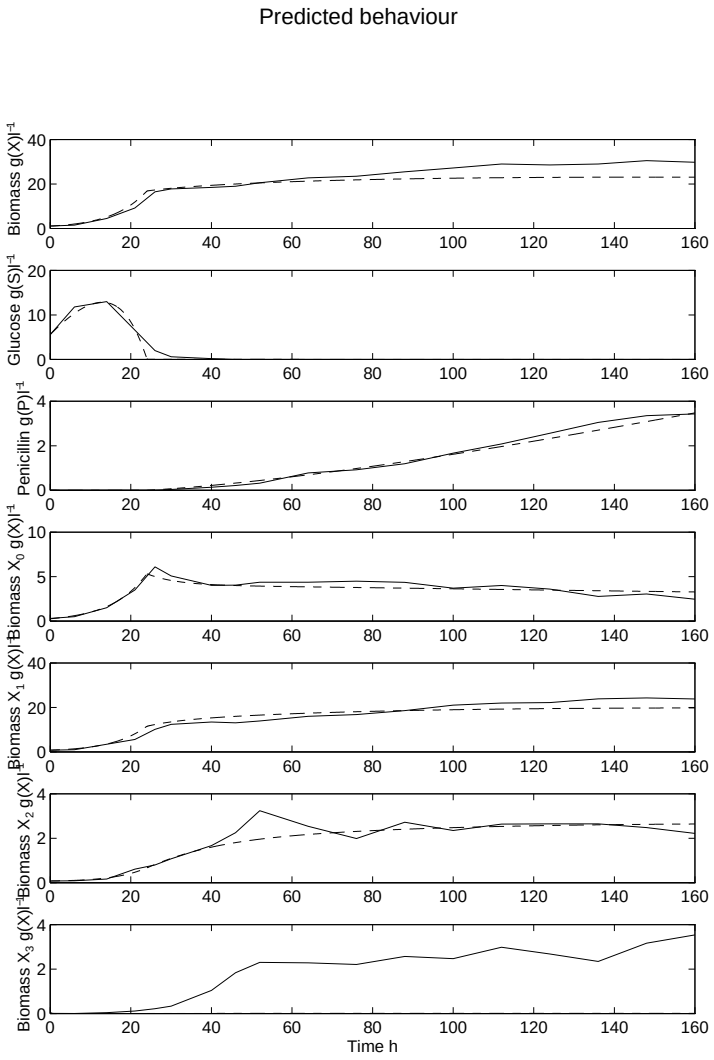

Fig. 3.7: Comparing predictions of data obtained using a constant feed rate, for models tuned with data obtained using a time-varying feed rate, validated against data obtained using a constant feed rate.
Model 1 $\stackrel{F}{\rightarrow}$ 2 $\stackrel{F}{\rightarrow}$ 3 with lysis, (– Measured data, $-\cdot$ simple model)

3.1.5 Discussion

From the tables of average summed squared prediction errors, Tables 3.3 and 3.6, it may be found that the best performing single-step model is is that in which the conversion of X_1 to X_3 is described using an inhibition kinetic and which contains a lysis term, ($1 \xrightarrow{I} 3$ (with lysis)), and that the best performing two-step model is that in which the conversion of X_1 to X_2 is described as using a first order kinetic, conversion of X_2 to X_3 is also described using a first order kinetic, and which contains a lysis term ($1 \xrightarrow{F} 2 \xrightarrow{F} 3$ (with lysis)).

It is not possible to make a direct comparison between the averaged summed squared prediction errors for these two models, as they have been tuned against differing numbers of model states.

Some of the arguments in favour of choosing the two-step model are as follows.

- It is likely to be easier to extend a two-step model to include a description of the vacuolation process than a single-step model, when the details of the vacuolation process are better understood.

- More data were used in tuning the two-step models than in tuning the single-step models (the additional vacuole state data), and so the two-step models are based on more process information.

- The two-step model makes better predictions of the glucose and penicillin concentrations. Since glucose is most likely to be the controlled variable for the fermentation, and penicillin is the product of principal

interest, it is particularly important that the model should describe these two states well.

- The two-step model describes the concentration of hyphal tips better than does the single-step model.

Against these reasons, the following arguments were advanced for considering the single-step models.

- The single-step model is simpler and has fewer states and parameters than does the two-step model.

- The single-step model fits the active cytoplasm concentration better than the two-step model does.

It is difficult to be certain that any particular model tuning is globally optimal (that the best of all possible parameter sets has been found). Parameter optimisations starting with differing initial parameter sets may terminate in local optima, or termination of the parameter estimation routine may occur because the parameter set has entered a region of the parameter space in which the summed squared error varies extremely slowly with changes in parameter values, thus being approximately 'flat'.

So, for the reasons listed above, and since the differences in the summed error values for the two best models seem to be mainly in the X_3 state, which has no influence on glucose consumption, biomass growth or penicillin formation (unlike X_0, X_1 and X_2), the best two-step model was used in the work following from model simplification.

3.2 Including Lactose as a Second Substrate

Thus far, the results presented have been based around a model of the penicillin fermentation in which only a single carbon-providing substrate, glucose, is assumed to be present. However, industrially this is not the case, as the carbon source used is frequently complex. Experimentally, lactose was found to be present in the inocula for the fermentation experiments carried out in the department at levels that can not be considered to be negligible ($\approx 6g/L$). The presence of lactose in the experimental data sets provided an opportunity to extend the model so as to consider conditions in which more than one carbon source is present.

3.2.1 A previous two substrate penicillin fermentation model

A penicillin fermentation model based on lactose and glucose as substrates has previously been published by Kluge *et al.* (1992). In this model, the focus is quite strongly on the rates of uptake of the two substrate species from the fermentation medium, with rates of growth and product formation being calculated after subtraction of the biomass's maintenance requirements from the total rate of substrate uptake.

Superficially, this approach is similar to that of Nielsen (1991), but it differs in that in the model of Kluge *et al.* (1992) no consideration is made of enzyme concentrations within the biomass that are associated with the uptake of different substrates from the medium. In the model of Nielsen (1991), these enzyme concentrations vary with time, changing according to the availability of substrates in the medium, and so the model is capable of describing the delays in converting from growth on one substrate to growth on

a second. The model of Nielsen (1991) assumes that, after uptake, the sugar substrates that it considers are converted into the same energy-providing compound and so that there is no difference in the way in which the organism obtains energy from its internal sources.

The model of Kluge *et al.* (1992) differs in that no enzyme structures are assumed, and instead considers the rate of uptake of lactose as being related to the rate of uptake of glucose. Kluge *et al.* (1992) have the following pair of equations:

$$\frac{\mathrm{d}S}{\mathrm{dt}} = -q_S X_A + [S_f - S]D \tag{3.2}$$

$$\frac{\mathrm{d}L}{\mathrm{dt}} = -q_L X_A + [L_f - L]D \tag{3.3}$$

where q_S and q_L are the specific uptake rates for glucose and lactose, respectively, S_f and L_f are the glucose and lactose concentrations in the feed, D is the dilution rate ($Feed/Volume$), and X_A is the concentration of active biomass. The specific consumption rates q_S and q_L are given by:

$$q_S = \frac{q_{S0} S}{K_S + S} \tag{3.4}$$

$$q_L = \frac{q_{L0} L}{(K_L + L)} \frac{1}{(1 + C_{LS} q_S)} \tag{3.5}$$

where q_{S0} and q_{L0} are the maximum specific uptake rates of glucose and lactose, K_S and K_L are Michaelis-Menten uptake expression coefficients, and C_{LS} is a coefficient relating the repression of lactose uptake to the glucose uptake rate. Substituting for q_S in the expression for q_L, the following is obtained:

$$q_L = \frac{q_{L0}L}{(K_L + L)} \frac{K_S + S}{(K_S + (1 + C_{LS}q_{S0})S)} \tag{3.6}$$

$$= \frac{q_{L0}L}{K_L + L} \left(1 + \frac{C_{LS}q_{S0}S}{K_S + S}\right)^{-1} \tag{3.7}$$

In Kluge *et al.* (1992), biomass growth is then described as follows.

$$\frac{\mathrm{d}X}{\mathrm{dt}} \propto \mu_A X_A \tag{3.8}$$

where

$$\mu_A = \left(q_S + \alpha q_L - m - \frac{\mu_P}{Y_{PS}}\right) Y_{XS} \tag{3.9}$$

Here μ_A is the specific growth rate of active biomass, α is a coefficient relating the nutritional value of lactose to that of glucose, m is a maintenance coefficient, μ_P is the specific penicillin production rate, and Y_{PS} and Y_{XS} are yield coefficients for the production of penicillin and biomass, respectively, from substrate.

Relating substrate and uptake in this way, using Michaelis-Menten enzyme uptake kinetics, is conceptually different from using Monod kinetics to describe the growth of an organism, and then calculating from the growth

rate the necessary growth-related rate of substrate uptake.

The Michaelis-Menten enzyme uptake kinetics based approach, as used by Kluge *et al.* (1992), starts by calculating the rate of substrate uptake, and then relates the growth rate of the biomass to the excess of uptake over requirements for maintenance and product formation. The Monod kinetic based approach starts from a correlation between substrate concentration and biomass growth rate, and calculates the growth related substrate consumption from this, usually by assuming a time-invariant yield coefficient of biomass from substrate. The substrate consumption rates related to biomass maintenance and, through a second yield coefficient, to the product formation rate are then added on, and the total is taken to be the rate of removal of substrate from the medium.

Theoretically, it would appear that the former, Michaelis-Menten uptake based, approach is more valid, because the latter, Monod growth based, approach can give rise, theoretically, to uptake rates which exceed the physical limitations of organisms being modelled.

The use of such an approach has the drawback that it is, mathematically, possible for the growth rate to become negative if the maintenance and production associated consumption of substrate should exceed the rate at which substrate is taken up by the organism, as may well be the case for low substrate concentrations such as those observed during the production phase of a penicillin fermentation. Difficulties could arise in attempting to avoid this problem, by matching the consumption rates to the uptake rates, in determining how the consumption of substrate should be split between maintenance and product formation.

3.2.2 Two substrates in the Paul and Thomas (1996) model

It was observed that the lactose present in the fermentation was carried over from the inoculum, and that the lactose concentration remained approximately constant (subject to dilution by the glucose-containing feed) until such time as the glucose concentration was significantly reduced, at which point the lactose was rapidly consumed.

Paul (1997) proposed the following form for the lactose consumption kinetic.

$$\frac{\mathrm{d}L}{\mathrm{dt}} = -\frac{\mu_L L}{(K_{SL} + L)} \frac{X_0 + X_1}{(1 + S/K_{SI})} \tag{3.10}$$

Here μ_L is the maximum specific consumption rate of lactose, K_{SL} is a Monod coefficient, and K_{SI} is an inhibition coefficient.

In the above equation, the lactose consumption rate decreases with increasing glucose concentrations, thereby giving a higher rate of lactose consumption at lower glucose concentrations. Lactose consumption, or uptake, is assumed to be associated with the active biomass fractions, the hyphal tips (X_0) and the active, penicillin-producing subapicial regions (X_1). Note that the rate of lactose consumption falls to zero as the lactose concentration falls to zero, thus satisfying the logical boundary condition that no concentration can ever become negative.

To minimise the impact of adding the lactose term to the simplified model, the consumption of lactose has been likened to converting the lactose in the medium to glucose in the medium, which is then taken up by the organism in the usual way. This is not intended to describe any physical process.

Using this approximation to describe the consumption of lactose from the medium means that the bulk of the model, the equations describing growth and product formation, may be left unchanged and solely glucose based, and that the only equation that needs to be modified is the glucose rate of change equation. As long as the rate of lactose uptake in the fermentation is small, remaining less than the total rate of substrate utilisation for growth, maintenance and product formation, then this crude approximation should be reasonable. As the two substrate model currently stands, it is probably only justified to use the model to describe fermentations where lactose is present in relatively small amounts at the start of the fermentation, and where glucose is fed throughout at a rate sufficient to account for the bulk of the substrate taken up by the organism. The model is not considered to be suitable for use in describing growth on lactose as a single substrate - such conditions are far from those for which the model has been developed and applied to date.

When the conversion of lactose to glucose is added to the equation used in the model to describe the change in the glucose concentration, the following equation is obtained.

$$\begin{aligned}\frac{\mathrm{d}S}{\mathrm{dt}} = &-\frac{\alpha_0\mu_0 X_1 S}{K_0+S} - \frac{\alpha_e\mu_e X_0 S}{K_e+S} - \frac{m_0 X_0 S}{K_1+S} - \frac{m_1\rho_c v_{ic} S}{K_2+S} \\ &- \frac{\alpha_p\mu_p\rho_c v_{ic} S}{K_P+S(1+S/K_I)} + \frac{\mu_L L}{(K_{SL}+L)}\frac{X_0+X_1}{(1+S/K_{SI})}\end{aligned} \tag{3.11}$$

3.3 Notation

C_{LS}	Constant allowing for lactose uptake repression in the presence of glucose uptake, g(DW)hg(S)$^{-1}$
D	Dilution rate, h^{-1}
K_I	Inhibition coefficient, g(S)l^{-1}
K_L	Monod coefficient for lactose, g(L)l^{-1}
K_P	Inhibition coefficient, g(S)l^{-1}
K_S	Monod coefficient for glucose, g(S)l^{-1}
K_{SI}	Inhibition coefficient for lactose conversion, g(S)l^{-1}
K_{SL}	Monod coefficient for lactose, g(L)l^{-1}
K_0	Monod type denominator term, g(S)l^{-1}
K_1	Monod type denominator term, g(S)l^{-1}
K_2	Monod type denominator term, g(S)l^{-1}
K_e	Monod type denominator term, g(S)l^{-1}
L	Concentration of lactose, g(L)l^{-1}
L_f	Concentration of lactose fed to the fermenter, g(L)l^{-1}
L	Biomass conversion kinetic coefficient, g(S)l^{-1}
M	Inhibited biomass conversion kinetic coefficient, g(S)l^{-1}
P	Concentration of penicillin, g(P)l^{-1}
S	Concentration of glucose, g(S)l^{-1}
S_f	Concentration of glucose fed to the fermenter, g(S)l^{-1}
X	Concentration of biomass, g(DW)l^{-1}
X_A	Concentration of active biomass, g(DW)l^{-1}
X_0	Concentration of biomass state 0, (hyphal tips) g(DW)l^{-1}
X_1	Concentration of biomass state 1, (subapicial regions) g(DW)l^{-1}
X_2	Effective concentration of biomass state 2, (vacuoles) g(DW)l^{-1}
X_3	Concentration of biomass state 3, (fully vacuolated regions) g(DW)l^{-1}
X_4	Effective concentration of biomass state 4, (lysed material) g(DW)l^{-1}
Y_{PS}	Yield coefficient for penicillin with respect to substrate, g(P)g(S)$^{-1}$
Y_{XS}	Yield of biomass with respect to glucose, g(DW)g(S)$^{-1}$
$_{meas}$	Subscript denoting measured value

$_{sim}$	Subscript denoting simulated value
k	First order biomass conversion coefficient, h^{-1}
m	Maintenance coefficient, $g(S)h^{-1}$
m_0	Maintenance coefficient for state 0, $g(S)g(DW)^{-1}h^{-1}$
m_1	Maintenance coefficient for state 1, $g(S)g(DW)^{-1}h^{-1}$
q_L	Uptake rate of lactose $g(L)g(DW)^{-1}h^{-1}$
q_{L0}	Uptake coefficient for lactose, $g(L)g(DW)^{-1}h^{-1}$
q_S	Uptake rate of glucose, $g(S)g(DW)^{-1}h^{-1}$
q_{S0}	Uptake coefficient for glucose, $g(S)g(DW)^{-1}h^{-1}$
t	Time, h
v_{ic}	Volume concentration of active cytoplasm, m^3l^{-1}

Greek Symbols

α	Coefficient relating nutritional value of glucose to that of lactose, $g(S)g(L)^{-1}$
α_e	Coefficient relating substrate consumption to biomass extension, $g(S)g(DW)^{-1}$
α_p	Coefficient relating substrate consumption to product formation, $g(S)g(P)^{-1}$
α_0	Coefficient relating substrate consumption to biomass growth, $g(S)g(DW)^{-1}$
μ_A	Specific growth rate of active biomass, h^{-1}
μ_L	Specific conversion rate for lactose, $g(L)g(DW)^{-1}h^{-1}$
μ_P	Penicillin production rate, $g(P)h^{-1}$
μ_P	Penicillin production constant, $g(P)g(DW)^{-1}h^{-1}$
μ_0	Specific growth rate, $g(X0)g(X1)^{-1}h^{-1}$
μ_e	Specific growth rate, $g(X1)g(X0)^{-1}h^{-1}$
ρ_c	Density of cytoplasm, gm^{-3}

4. IMPROVING PARAMETER CONFIDENCE

4.1 The Form of the Equations

The penicillin fermentation is considered here as being described by a non-linear differential equation based model of the following form.

$$\dot{x}(t) = f(x(t), \beta, u(t)) \tag{4.1}$$

$$y(t) = g(x(t), \beta, u(t)) \tag{4.2}$$

In the above, $x(t)$ is a vector of time-varying model states, β is a set of assumed time-invariant parameters, and $u(t)$ is some time-varying input to the model (such as the feed rate of substrate to the fermenter). The output of the model is $y(t)$; this second equation may be used to relate measurements to the model states. Frequently the measurements *are* the model states, such as biomass, substrate and product concentrations, and volume in the fermenter, but other measurements are possible, for example, carbon dioxide production rate (CPR), which has been modelled previously (Montague *et al.*, 1986) as being related to biomass growth and maintenance, and to penicillin production, ie. a relationship of the form $CPR \propto \alpha\dot{X} + \beta X + \gamma\dot{P}$. For simplicity, we will here assume that only the states are measured, that is, that $y(t) = x(t)$.

Typically, there exists no analytical solution to a model of this type, and so numerical integration is needed to calculate the state trajectories over time. Here the models were numerically integrated using SIMULINK, a block-diagram oriented modelling tool associated with MATLAB, which provides a number of numerical integration algorithms. The two algorithms most frequently used in this work were a fourth order Runge-Kutta method, and Gear's algorithm.

4.2 Tuning the Model Parameters

In order to use the model for practical purposes, it must first be tuned so as to accurately represent the fermentation. This was done using the MATLAB least squares based optimisation routine `leastsq` to adjust the parameters so as to minimise the summed weighted squared error between the fermentation data measured at a number of sample intervals, and values generated using the model.

The least squares error between the measured and simulated values may be expressed as follows.

$$E = \sum_{i=1}^{n} (m(t_i) - x(t_i))' W (m(t_i) - x(t_i)) \tag{4.3}$$

In the above, E is the error value, $m(t)$ is a vector of measurement values at some time t, $x(t)$ is the corresponding vector of simulated values, and the summation is carried out for n sample times. The matrix W is a time-invariant weighting matrix, frequently a diagonal matrix with one weight per state along the leading diagonal.

To avoid possible bias in the parameter set obtained, as a result of a state with large absolute values dominating the tuning of the parameters, the error value at each measurement interval, for each model state, was divided by the maximum measured value of the model state in question. This is equivalent to using a diagonal W matrix, with $1/(\max(x))^2$ along the diagonal. Weighting on the basis of initial, final or minimum values would have resulted in divide-by-zero errors, and it was considered that weighting on the basis of average values would have been biased, since zero values would depress the average value of a state's measurements. (The measured data included zero values for some initial, final and intermediate values.)

4.2.1 A geometrical interpretation of the errors

The error function E can be considered as a hypersurface given by

$$E = E(\beta) - E_0 \tag{4.4}$$

where $E(\beta)$ denotes a general error value for some parameter set β, and E_0 is the minimum value of the error function that we are seeking.

$$E_0 = E(b) \tag{4.5}$$

In the above, b denotes the parameter set at the optimal tuning. Expressed as a Taylor expansion around the optimum ...

$$E(\beta) = E_0 + \left.\frac{\partial E}{\partial \beta'}\right|_{\beta=b} (\beta - b) + \frac{1}{2}(\beta - b)' \left.\frac{\partial^2 E}{\partial \beta \, \partial \beta'}\right|_{\beta=b} (\beta - b) + \cdots \tag{4.6}$$

First and second derivatives of scalars with respect to vectors are defined as follows:

$$\frac{\partial E}{\partial \beta} = \begin{bmatrix} \frac{\partial E}{\partial \beta_1} \\ \frac{\partial E}{\partial \beta_2} \\ \frac{\partial E}{\partial \beta_3} \end{bmatrix} \tag{4.7}$$

$$\frac{\partial E}{\partial \beta'} = \begin{bmatrix} \frac{\partial E}{\partial \beta_1} \frac{\partial E}{\partial \beta_2} \frac{\partial E}{\partial \beta_3} \end{bmatrix} \tag{4.8}$$

$$\frac{\partial^2 E}{\partial \beta \partial \beta'} = \begin{bmatrix} \frac{\partial^2 E}{\partial \beta_1 \beta_1} \frac{\partial^2 E}{\partial \beta_1 \beta_2} \frac{\partial^2 E}{\partial \beta_1 \beta_3} \\ \frac{\partial^2 E}{\partial \beta_2 \beta_1} \frac{\partial^2 E}{\partial \beta_2 \beta_2} \frac{\partial^2 E}{\partial \beta_2 \beta_3} \\ \frac{\partial^2 E}{\partial \beta_3 \beta_1} \frac{\partial^2 E}{\partial \beta_3 \beta_2} \frac{\partial^2 E}{\partial \beta_3 \beta_3} \end{bmatrix} \tag{4.9}$$

The derivative of a vector with respect to another vector, for example, $\partial x / \partial \beta$, is as follows:

$$\frac{\partial x}{\partial \beta} = \begin{bmatrix} \frac{\partial x_1}{\partial \beta_1} \frac{\partial x_2}{\partial \beta_1} \\ \frac{\partial x_1}{\partial \beta_2} \frac{\partial x_2}{\partial \beta_2} \\ \frac{\partial x_1}{\partial \beta_3} \frac{\partial x_2}{\partial \beta_3} \end{bmatrix} \tag{4.10}$$

Because E has a minimum at $\beta = b$ ...

$$\left.\frac{\partial E}{\partial \beta'}\right|_{\beta=b} = 0$$

and

$$\frac{\partial^2 E}{\partial \beta \, \partial \beta'} \quad \text{is positive definite}$$

So, neglecting higher terms, we have ...

$$E(\beta) \approx E_0 + \frac{1}{2}(\beta - b)' \left.\frac{\partial^2 E}{\partial\beta\,\partial\beta'}\right|_{\beta=b} (\beta - b) \tag{4.11}$$

Hence,

$$E \approx \frac{1}{2}(\beta - b)' \left.\frac{\partial^2 E}{\partial\beta\,\partial\beta'}\right|_{\beta=b} (\beta - b) \tag{4.12}$$

which describes a hyperparaboloid. Curves of constant E are hence hyperellipsoids.

For the error function in Equation 4.3, the first two derivatives of the error function with respect to the parameters may be expressed as follows (Eykhoff, 1974).

$$\frac{\partial E}{\partial\beta} = -2\sum_{i=1}^{n}\left(\frac{\partial x(t_i)}{\partial\beta}\right)' W(m(t_i) - x(t_i)) \tag{4.13}$$

$$\frac{\partial^2 E}{\partial\beta\,\beta'} = 2\sum_{i=1}^{n}\left(\frac{\partial x(t_i)}{\partial\beta}\right)' W \left(\frac{\partial x(t_i)}{\partial\beta}\right) - 2\sum_{i=1}^{n}\left(\frac{\partial^2 x(t_i)}{\partial\beta\,\partial\beta'}\right) W(m(t_i) - x(t_i)) \tag{4.14}$$

The second of the above equations is not strictly correct, as $\partial^2 x(t_i)/\partial\beta\,\partial\beta'$ is a tensor. However, close to the optimal parameter set, for noise-free measurements perfectly described by the model, the error between measured and simulated values goes to zero, $[\forall t_i, \lim_{\beta\to b}(m(t_i) - x(t_i)) = 0]$, and so this second term vanishes. (For cases where the model structure is not capable of matching the data, or where there is a relatively large noise contribution to the error in fitting the model, this will, however, not be the case, and it

may be worthwhile comparing the relative magnitudes of the the two terms in Equation 4.14 in such circumstances.)

The sensitivity of the model states to the parameter values ($\partial x(t)/\partial\beta$) is described by the following equation, obtained by differentiating Equation 4.1 with respect to the parameter vector β (Holmberg, 1982).

$$\frac{\mathrm{d}\frac{\partial \mathrm{x(t)}}{\partial\beta}}{\mathrm{dt}} = \frac{\partial f}{\partial x}\frac{\partial x(t)}{\partial\beta} + \frac{\partial f(t)}{\partial\beta} \tag{4.15}$$

In optimisation, it is generally assumed that, close to the optimal point, the errors of a system vary in a quadratic manner (Norton, 1986). That is to say, that contours of constant error around the tuning point form ellipsoids (ellipses in the case of a two-parameter system). For a least squares objective function of the type given in Equation 4.3, the above equations show that close to an optimum, where the errors may be approximately described by a Taylor series expansion, this is always the case.

4.2.2 Considering ellipsoids

Substituting for $\partial^2 E/\partial\beta\partial\beta'$ in Equation 4.12 from Equation 4.14, ignoring the second term which vanishes close to the optimal parameter set we obtain the following,

$$E \approx \frac{1}{2}(\beta - b)' \left[\sum_{i=1}^{n} \left(\frac{\partial x(t_i)}{\partial\beta}\right)' W \left(\frac{\partial x(t_i)}{\partial\beta}\right)\right] (\beta - b) \tag{4.16}$$

which is a quadratic equation.

Consider a general quadratic equation of the following form.

$$E = X'AX \tag{4.17}$$

If we assume that A is a diagonal matrix, and that the X vector has only two elements, x_1 and x_2, then this may be rewritten in the simpler form

$$E = a_{11}x_1^2 + a_{22}x_2^2 \tag{4.18}$$

which is the equation describing a three-dimensional paraboloid.

For fixed values of E, the above equation is analogous to that for an ellipse, written in terms of the major and minor axis lengths.

$$E = x_1^2/a^2 + x_2^2/b^2 \tag{4.19}$$

It can be seen that a_{11} is analogous to $1/a^2$ and, similarly, a_{22} is analogous to $1/b^2$.

A quadratic function is plotted in Figure 4.1, which shows the paraboloid for the function, two elliptical contours of constant error value, and the major and minor axes corresponding to one of the error contour ellipses.

4.3 Optimal Experiment Design

The field of optimal experiment design aimed at improving the quality of system models is well established (Walter and Pronzato, 1990). Most of the work published in this field is concerned with the improvement of models' system predictions by improving the confidence with which model parame-

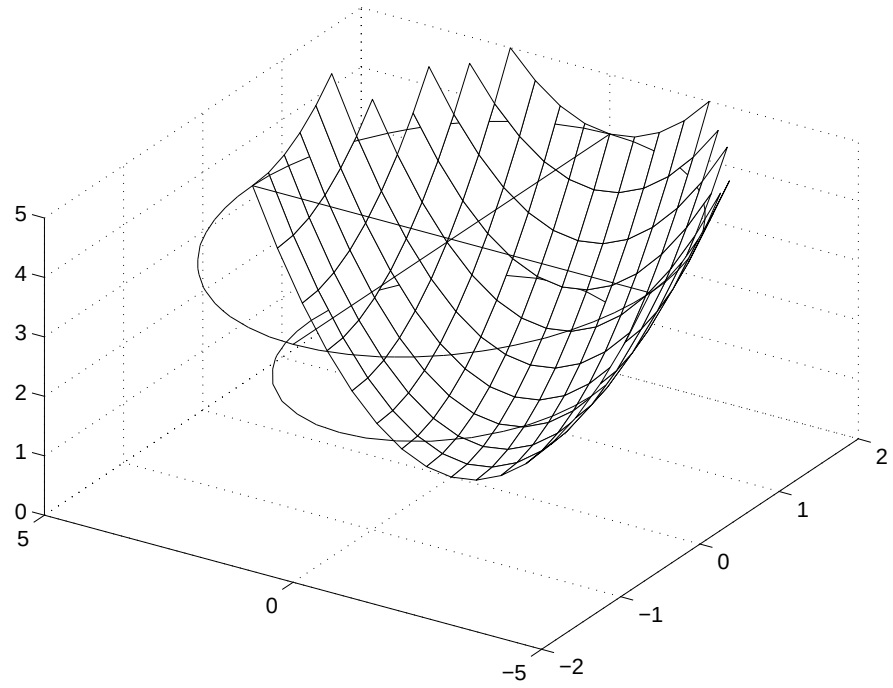

Fig. 4.1: Paraboloid surface, two elliptical error contours and specimen major and minor axes for the quadratic function $E = x_1^2 + x_2^2/4$

ters are estimated. Optimal experiment works have focussed on selecting the conditions under which experiments should be carried out (Hosten, 1974; Hosten and Emig, 1975; Pinto *et al.*, 1990), designing the inputs used to excite systems being modelled (Murray and Reiff, 1984; Espie and Macchietto, 1989; Versyck *et al.*, 1997), improving the positioning of sensors, and on selecting portions of the measured data (Yoo *et al.*, 1986; Kalogerakis and Luus, 1984) or sampling rates (Murray and Reiff, 1984; Jacquez and Greif, 1985) so as to maximise parameter confidence.

Various optimisation criteria have been advanced, most of which are re-

lated to the information matrix (Fisher Information Matrix) or its inverse, the parameter variance-covariance matrix (Hosten, 1974; Pinto *et al.*, 1990; Walter and Pronzato, 1990). Sequential schemes, in which data are gradually accumulated over a number of experiments and model tuning is based on the total data available, as well as single-shot, best next experiment approaches are described. The use of reparameterisation (Agarwal and Brisk, 1985; Bilardello *et al.*, 1993) and of rescaling the parameters (Pinto *et al.*, 1991) as means of improving confidence in the parameters to be estimated also appears in the literature. Examples of dynamic systems used to illustrate the design of experimental inputs based on information matrix related design criteria include the continuous yeast fermentation (Espie and Macchietto, 1989), a batch fermentation of *Trichosporon cutaneum* (Baltes *et al.*, 1994), model Monod and Haldane processes (Versyck *et al.*, 1997) and fixed bed heat transport (Murray and Reiff, 1984).

Improved experiment design for parameter estimation is particularly important with respect to fermentation modelling, as performing fermentations to generate data for model tuning is both costly and time-consuming. The data sets supplied by Dr. Gopal Paul, used in this thesis, are typically taken from week-long fermentations, with additional time being taken to prepare inocula, media and equipment, and for biomass sampling, image analysis measurements and subsequent post-processing of the data obtained so as to get it into a form suitable for use in modelling.

An alternative approach, which has received attention in recent years, is the more goal-oriented approach of designing only experiments which are optimal with regard to economic or productivity criteria, and using the data

from these as a basis for sequential model refinement (Galvanauskas *et al.*, 1997; Galvanauskas *et al.*, 1998). This leads to repeated passes through a cycle of designing an economically optimal input, performing an experiment using this input, refining the model parameters with emphasis on those parameters to which the economically optimal profile is most sensitive, and returning to the design of an economically optimal input. This approach may have the advantage that, since it concentrates primarily on the region around the economically optimal trajectory, a simpler model may be applied over this subset of fermentation conditions and be capable of predicting the behaviour of the process in this range just as well as a more complex fermentation model, tuned over a wider range of fermentation conditions. If a model is to be used to improve understanding of the system, as well as for improving the performance of a production process, then this highly goal-centred approach may well focus on too narrow a set of fermentation conditions. Example processes used as illustrations (Galvanauskas *et al.*, 1997; Galvanauskas *et al.*, 1998) are maximising the biomass in an *Escherichia coli* fermentation, maximising the amount of biomass produced with respect to glucose supplied in a baker's yeast fermentation, and maximising the concentration of penicillin at a predefined final fermentation time for a *Penicillium chrysogenum* fermentation.

4.3.1 Criteria for experiment design

Two criteria are commonly used to determine convergence of calculus-based techniques:

- the change in function value from one iteration to the next is less than some small value
- the change in the parameter set from one iteration to the next (the 'distance' moved) is less than some small value

It is also common for these techniques to be abandoned if a minimum is not found within a given number of iterations. One of the goals of experiment design is to improve the quality of existing parameter estimates. If the parameter optimisation algorithm has terminated prematurely, as a result of exceeding a fixed number of iterations, then the parameter set thus obtained may not be suitable for use in designing experiments, because it may not be located at a minimum of the objective function.

Since one of the criteria commonly used to determine convergence in optimisation routines is that the change in error value for an update to the parameter values be less than some specified value, surfaces that are 'steeper' close to the optimal point are likely to get closer than those that are 'shallower'.

If we consider also the *shape* of the ellipsoids around the optimal point, it seems reasonable that we would wish the ellipsoids to be close to spherical. Since we are considering the region close to the optimum parameter set, in which we have assumed that the Taylor series expansion provides a good approximation to the error surface, the surfaces of constant error close to the optimum parameter set will always be ellipsoids. (For the majority of the gradient descent methods described below in Section 4.5, progress directed towards the minimum only occurs when the error contours around the minimum form hyperspheres.) In ellipsoids with relatively long axes the

optimisation routine is likely to reach a point on said long axis, and then to attempt to progress along the 'bottom of the valley' towards the optimal point. Since the error values are likely to change slowly along such a 'valley', we find ourselves in a position where, although we may be sure of having obtained the correct ratio between some of the parameters being tuned, we cannot be as certain of their absolute values. It should be noted that the principal axes of the ellipsoids are almost certainly not going to lie parallel to the axes along which the parameters vary.

Much work in the area of optimal experiment design has been based on criteria derived from the Fisher Information Matrix (FIM). In its continuous form, the FIM may be defined as follows (Munack, 1989). (Alternative definitions based on the logarithm of the sensitivities exist, but are less immediately comprehensible.)

$$\text{FIM} = \int_0^T \left(\frac{\partial \mathbf{x}(\mathrm{t})}{\partial \boldsymbol{\beta}}\right)' \mathbf{W} \left(\frac{\partial \mathbf{x}(\mathrm{t})}{\partial \boldsymbol{\beta}}\right) \mathrm{dt} \tag{4.20}$$

In discrete form, this becomes the following:

$$\text{FIM} = \sum_{i=1}^{n} \left(\frac{\partial \mathbf{x}(t_i)}{\partial \boldsymbol{\beta}}\right)' \mathbf{W} \left(\frac{\partial \mathbf{x}(t_i)}{\partial \boldsymbol{\beta}}\right) \tag{4.21}$$

Comparing Equation 4.21 with Equations 4.12 and 4.14, we see that the FIM is an approximation to the second derivative of the expression for the error surface, which is the dominant term in the Taylor series expansion defining the error surface in the neighbourhood of an optimal parameter set.

Hence

$$E \propto (\beta - b)' FIM (\beta - b) \tag{4.22}$$

The FIM is related to the inverse of expectation of the square of the error in the parameter set, that is to say ...

$$\mathcal{E}[(\hat{\beta} - b)(\hat{\beta} - b)'] \quad \geq \quad FIM^{-1} \tag{4.23}$$

(The FIM is the inverse of the covariance matrix for the modelling errors.) For a derivation of the FIM, see Eykhoff (1974).

The FIM in Equation 4.21 is defined as the sum of a series of product terms, each of which comprises the transposed sensitivity matrix, multiplied by the diagonal weighting matrix, multiplied finally by the sensitivity matrix. Thus we can see that the FIM is always going to be a diagonally symmetric matrix.

Diagonally symmetric matrices can always be decomposed in the following manner (Bronshtein and Semandyayev, 1985).

$$FIM = V' D V \tag{4.24}$$

where FIM is the diagonally symmetric Fisher Information Matrix, V is a matrix made up row-wise of the eigenvectors of FIM and D is a diagonal matrix with the eigenvalues of FIM along the diagonal.

Substituting from Equation 4.24 into Equation 4.22, we obtain

$$E = (\beta - b)' V' D V (\beta - b) \tag{4.25}$$

rearranging this becomes

$$E = [(\beta - b)' V'] D [V(\beta - b)] \tag{4.26}$$

which is equivalent to

$$E = (\Delta\beta^*)' D (\Delta\beta^*) \tag{4.27}$$

where

$$\Delta\beta^* = [V(\beta - b)] \tag{4.28}$$

This form of the equation for the error is the same as that in Equation 4.17, and so the properties of the general error (hyper)ellipsoids centred on the optimal parameter set are simply related to those for the two-dimensional ellipses defined by Equation 4.18 and Equation 4.19. Since elements on the diagonal of a diagonal matrix are related to the lengths of the axes of an ellipsoid, $a_{11} \propto 1/a^2$, (see Subsection 4.2.2 for more details), and D is a diagonal matrix with the eigenvalues of the FIM along the diagonal, then the eigenvalues of the FIM may be related to the lengths of the axes of error ellipsoids, $\lambda \propto 1/l^2$, where λ is some eigenvalue of the FIM, and l is the length of the corresponding axis of the error ellipsoid.

A number of criteria based on the FIM, and having geometric interpretations based around the ellipsoids of constant error value, have been advanced for use in the optimal design of identification experiments (Hosten and Emig, 1975; Pinto *et al.*, 1990; Walter and Pronzato, 1990). The most common of these are summarised, along with their conventional names and one possible geometrical interpretation, in Table 4.1.

Criterion	Formula	Interpretation
A	$\min(\mathrm{tr}(\mathrm{FIM}^{-1}))$	minimise mean variance
simplified A	$\max(\mathrm{tr}(\mathrm{FIM}))$	minimise mean variance
C	$\min(\mathrm{tr}(\mathrm{FIM}))$	minimises relative (mean) volume
D	$\max(\det(\mathrm{FIM}))$	minimises ellipsoid volume
E	$\max(\lambda_{\min}(\mathrm{FIM}))$	minimises longest axis
modified E	$\min(\mathrm{cond}(\mathrm{FIM}) = \frac{\lambda_{\max}(\mathrm{FIM})}{\lambda_{\min}(\mathrm{FIM})})$	spherical as possible

Tab. 4.1: Criteria for optimal experiment design derived from the Fisher Information Matrix (FIM). $\lambda_{\min}$ and $\lambda_{\max}$ are the minimum and maximum eigenvalues of the FIM. The above definitions are taken from Walter and Pronzato (1990)

The criteria given in Table 4.1 are not entirely independent of one another. (This is to be expected, as they are related to the properties of the same geometric structures.) Consider the A, D and modified E-optimality criteria.

$$\begin{aligned} \mathrm{tr}\ (FIM) &= \sum_{\lambda_{min}}^{\lambda_{max}} \lambda \\ \det(FIM) &= \prod_{\lambda_{min}}^{\lambda_{max}} \lambda \\ \mathrm{cond}\ (FIM) &= \lambda_{max}/\lambda_{min} \end{aligned}$$

(λ_{min} denotes the minimum eigenvalue and λ_{max} denotes the maximum eigenvalue.) If the desired reduction in the modified E-optimality criterion were to be achieved solely by decreasing the maximum eigenvalue (λ_{max}), then the desired reduction in the modified E-optimality criterion would lead to corresponding, undesirable decreases in the A and D-optimality criterion.

4.4 Multi-rate Extension to the Information Matrix

It may be possible to construct the information matrix for cases where data are measured at more than one sample rate, as is the case for fermentation data, measuring biomass concentrations at a relatively low sample rate and the concentrations of soluble species (by means of online HPLC, for example) at a higher rate.

If we consider the information matrix as an approximation to the second derivative of the error surface, then we can quickly calculate its form. Consider a simple least squares error, based on data measured at two different sampling rates. (The sampling rates do not need to be regular.) Then the error value is given by the following expression

$$\begin{aligned} E = & \sum_{i=1}^{n_1} (m_1(t_i) - x_1(t_i))' W_1 (m_1(t_i) - x_1(t_i)) \\ & + \sum_{i=1}^{n_2} (m_2(t_i) - x_2(t_i))' W_2 (m_2(t_i) - x_2(t_i)) \end{aligned} \tag{4.29}$$

where the measurements m_1 and m_2 (of states x_1 and x_2 respectively) are measured at n_1 and n_2 sample intervals over the measurement period. (The subscripts 1 and 2 refer to two sets of measurement data, taken from the

same experiment at two different sample rates for distinct sets of measured variables. There should be no duplication of measurements between sets 1 and 2.)

The two summed terms in the above expression are independent of each other, and can be differentiated separately, giving

$$\frac{\partial^2 E}{\partial\beta\,\beta'} = \sum_{i=1}^{n_1} \frac{\partial x_1(t_i)'}{\partial\beta} W_1 \frac{\partial x_1(t_i)}{\partial\beta} - \sum_{i=1}^{n_1} \frac{\partial^2 x(t_i)}{\partial\beta\,\beta'} W_1 (m_1(t_i) - x_1(t_i)) \quad (4.30)$$
$$+ \sum_{i=1}^{n_2} \frac{\partial x_2(t_i)'}{\partial\beta} W_2 \frac{\partial x_2(t_i)}{\partial\beta} - \sum_{i=1}^{n_2} \frac{\partial^2 x_2(t_i)}{\partial\beta\,\beta'} W_2 (m_2(t_i) - x_2(t_i))$$

In the above second derivative expression, the second term on each line vanishes close to the optimal parameter set, with the simulated values approximating the measured values, and we are left with the summation of two information matrix expressions, after the pattern of Equation 4.21.

$$\frac{\partial^2 E}{\partial\beta\,\beta'} = \sum_{i=1}^{n_1} \frac{\partial x_1(t_i)'}{\partial\beta} W_1 \frac{\partial x_1(t_i)}{\partial\beta} + \sum_{i=1}^{n_2} \frac{\partial x_2(t_i)'}{\partial\beta} W_2 \frac{\partial x_2(t_i)}{\partial\beta} \quad (4.31)$$

4.4.1 Example of multi-rate information matrix expression

The calculation of two information matrix expressions may be illustrated using the example of a model having two states (x and y) and three parameters (a, b, and c). Denoting $\partial x/\partial a$ by x_a for simplicity, we obtain the following expression.

$$FIM_{(1+2)} = \begin{bmatrix} x_a & y_a \\ x_b & y_b \\ x_c & y_c \end{bmatrix} \begin{bmatrix} W_1 & 0 \\ 0 & W_2 \end{bmatrix} \begin{bmatrix} x_a & x_b & x_c \\ y_a & y_b & y_c \end{bmatrix} \quad (4.32)$$

or, as a summation

$$FIM_1 + FIM_2 = \begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} \begin{bmatrix} W_1 \end{bmatrix} \begin{bmatrix} x_a & x_b & x_c \end{bmatrix} + \begin{bmatrix} y_a \\ y_b \\ y_c \end{bmatrix} \begin{bmatrix} W_2 \end{bmatrix} \begin{bmatrix} y_a & y_b & y_c \end{bmatrix} \tag{4.33}$$

Multiplying through in either case gives us

$$FIM_{(1+2)} = \begin{bmatrix} W_1 x_a x_a + W_2 y_a y_a & W_1 x_a x_b + W_2 y_a y_b & W_1 x_a x_c + W_2 y_a y_c \\ W_1 x_a x_b + W_2 y_a y_b & W_1 x_b x_b + W_2 y_b y_b & W_1 x_b x_c + W_2 y_b y_c \\ W_1 x_a x_c + W_2 y_a y_c & W_1 x_b x_c + W_2 y_b y_c & W_1 x_c x_c + W_2 y_c y_c \end{bmatrix} \tag{4.34}$$

In order to calculate a combined information matrix for data sampled at multiple rates, therefore, we simply calculate the individual information matrices for each distinct sampling rate, and add them together.

As an example of this, information matrices were calculated for the model of Paul *et al.* (1998), using an experimental input designed to be optimal for the sampling of all measured states at a uniform 5-hour sample interval, considering instead biomass states to be measured at 5-hour intervals and the soluble states to be measured at the much higher rate (online HPLC) of 5-minute intervals. The information matrix based solely on data obtained for all measured states, both biomass and soluble species, at the 5-hour sampling rate had a determinant of the order $1e27$, whilst that obtained by combining data obtained for the biomass states at the 5-hour sampling rate with data

obtained for the soluble species at the higher 5-minute sampling rate had a determinant of the order $1e47$.

Taking into consideration the higher rate at which the soluble species were measured results in a joint confidence ellipsoid for parameter estimation which is significantly smaller than is the case for the uniform 5-hour sample rate. When the individual eigenvalues of the two information matrices were compared, those for the information matrix formed by summating the pair of matrices for the two sampling rates were found to be, with a single exception, smaller than those for the single-rate information matrix.

It should be noted that the information matrices corresponding to the two sampling rates, obtained for biomass states alone, and for soluble species alone, both had determinants of zero, indicating that the volumes of the joint confidence ellipsoids for parameter estimation were infinite. This may be explained by referring to the sensitivities of the states considered in each case to the complete set of parameters. If there was a parameter to which all the states in a particular subset were insensitive, then the corresponding column in the sensitivity matrix would contain only zeros, and hence the row and column in the resulting information matrix would be all zeros, and the determinant of the information matrix would be zero. As modelled, the biomass states have no dependence on the penicillin hydrolysis rate, μ_h, and the soluble states have no dependence on the rate of lysis of degenerated hyphal compartments, μ_a. Hence the individual information matrices for the two sampling rates have determinants of zero.

4.5 Calculus Based Optimisation Techniques

There are a range of methods, broadly described as 'hill-climbing' (or 'valley-seeking'), used in optimisation. At each iteration they use gradient information supplied either as explicit derivatives of the objective function, or calculated from numerical 'experiments', small perturbations around the current parameter set, to determine in which 'direction' the function value decreases most rapidly, and then move in that direction. In this way, the algorithm progresses until a local minimum is reached.

The following descriptions of calculus based optimisation techniques are taken from Eykhoff (1974).

4.5.1 Steepest descent

In this method, update proceeds according to the following equation.

$$\beta(i+1) \;=\; \beta(i) - \Gamma \left.\frac{\partial E}{\partial \beta}\right|_{\beta=\beta(i)} \tag{4.35}$$

Here Γ is a positive constant. Some compromise must be sought between speed of convergence and size of Γ. Too large a value will cause the optimisation routine to oscillate around the optimal point; too small a value will take an inordinately long time to come close to it.

4.5.2 Steepest descent with minimisation along a line

This method proceeds exactly as for the preceding steepest descent method, with Γ being chosen so as to minimise the objective function in the direction $\partial E/\partial \beta$ for each parameter update. Typically, points are evaluated along the

direction of steepest descent until an increase is detected between values, a quadratic function is fitted through the evaluated points, and the minimum of the quadratic is used as starting point for the next iteration.

4.5.3 Newton-Raphson

According to Section 4.2.1, curves of constant error value (E) form hyperellipsoids. For an ellipsoid, lines of steepest descent do not necessarily point towards the minimum, unless the ellipsoid happens to be circular (see Figures 4.2 and 4.3).

It seems reasonable to suppose that a steepest descent method will be optimal when the surfaces of constant E are (hyper)spheres, such that progress in the direction of steepest descent is also progress towards the minimum. This may be achieved by transforming the ellipsoids into spheres in another space (with respect to a modified parameter vector) and then using the steepest descent method.

Since

$$\left.\frac{\partial^2 E}{\partial\beta\,\partial\beta'}\right|_{\beta=b} \tag{4.36}$$

is symmetric and positive definite, it can always be decomposed into a product made up a diagonal matrix with its eigenvalues on the leading diagonal (D) and a matrix made up, row-wise, of its eigenvectors (V).

$$\left.\frac{\partial^2 E}{\partial\beta\,\partial\beta'}\right|_{\beta=b} = V'DV \tag{4.37}$$

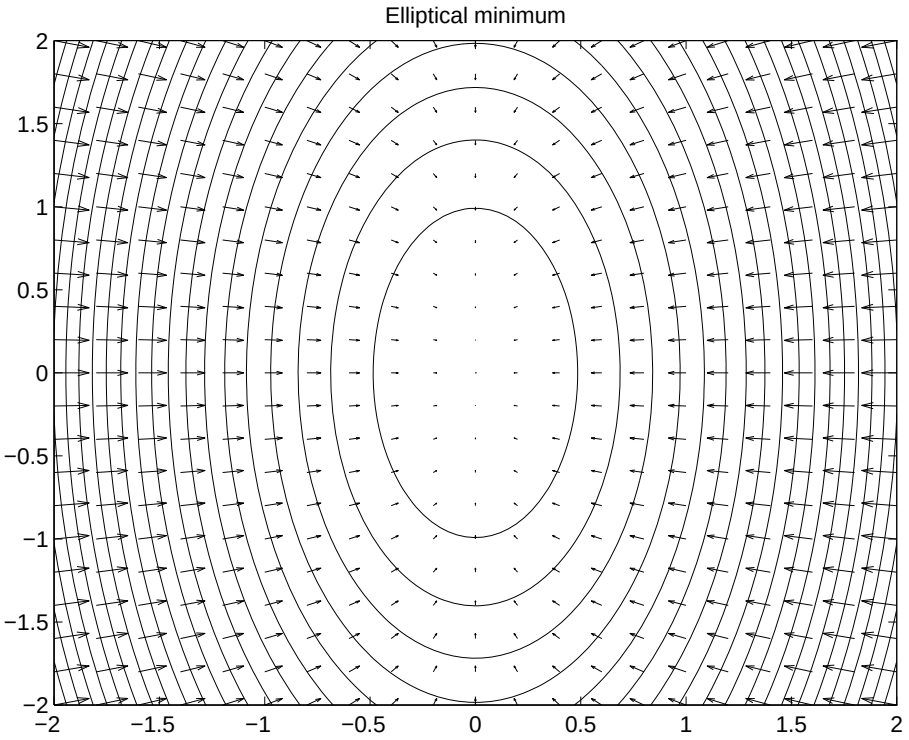

Fig. 4.2: Contours and arrows indicating steepest descent directions for an ellipsoidal objective function

Substituting Equation 4.37 into Equation 4.11 leads to

$$E(z^*) \quad = \quad E_0 + \frac{1}{2} z^{*'} D z^* \tag{4.38}$$

where

$$z^* \quad = \quad V(\beta - b)$$

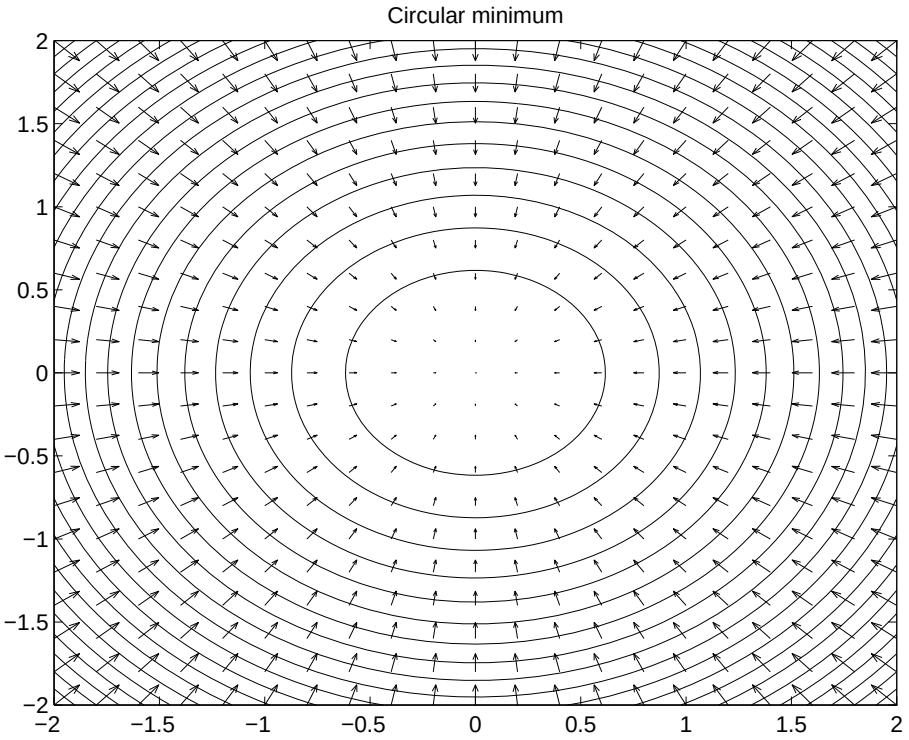

Fig. 4.3: Contours and arrows indicating steepest descent directions for a circular objective function

Defining

$$z = D^{\frac{1}{2}} z^* \quad \text{and} \quad z^* = D^{-\frac{1}{2}} z$$

we get

$$E(z) \;=\; E_0 + \frac{1}{2} z' z \tag{4.39}$$

In this z-space, the surfaces of constant E are hyperspheres. Performing

steepest descent with $\Gamma = 1$,

$$z(i+1) = z(i) - \left.\frac{\partial E}{\partial z}\right|_{z=z(i)}$$

Differentiating Equation 4.39

$$\left.\frac{\partial E}{\partial z}\right|_{z=z(i)} = z(i)$$

and so $z(i+1) = z(i) - z(i) = 0$. Because $\beta - b = V'D^{-\frac{1}{2}}z$ with V' and D positive definite, $\beta - b = 0$ also holds.

Transforming back from z-space to β-space gives,

$$\beta(i+1) = \beta(i) - \left[\frac{\partial^2 E}{\partial\beta\partial\beta'}\right]^{-1}_{\beta=b} \left.\frac{\partial E}{\partial\beta}\right|_{\beta=\beta(i)} \tag{4.40}$$

In most cases the second derivative will be a function of β, however it is often assumed to be almost independent of β, and so Equation 4.40 may be taken as defining the Newton-Raphson method, provided that the second derivative is calculated for the current values of β.

$$\beta(i+1) = \beta(i) - \left[\frac{\partial^2 E}{\partial\beta\partial\beta'}\right]^{-1}_{\beta=\beta(i)} \left.\frac{\partial E}{\partial\beta}\right|_{\beta=\beta(i)} \tag{4.41}$$

4.5.4 Gauss-Newton

Calculating second derivatives using small perturbations around the current parameter set is likely to be difficult and time-consuming, but it is possible to simplify the expression given above in Equation 4.41 for least-squares criteria. Differentiating Equation 4.3 twice gives the following expressions for its first

and second derivatives.

$$\begin{aligned}\frac{\partial E}{\partial \beta} &= -2\sum_{n=1}^{T}\frac{\partial x(t_n)'}{\partial \beta}W(m(t_n)-x(t_n)) \\ \frac{\partial^2 E}{\partial \beta\,\beta'} &= 2\sum_{n=1}^{T}\frac{\partial x(t_n)'}{\partial \beta}W\frac{\partial x(t_n)}{\partial \beta} - 2\sum_{n=1}^{T}\frac{\partial^2 x(t_n)}{\partial \beta\,\partial \beta'}W(m(t_n)-x(t_n))\end{aligned}$$

Close to the minimum, the second term in the second derivative term can be neglected, as $x \approx m$. Substituting the above simplifications into Equation 4.40 gives us the following.

$$\begin{aligned}&\beta(i+1) = \beta(i) + \\ &\left[\sum_{n=1}^{T}\frac{\partial x(t_n)'}{\partial \beta}W\frac{\partial x(t_n)}{\partial \beta'}\right]^{-1}_{\beta=\beta(i)}\left[\sum_{n=1}^{T}\frac{\partial x(t_n)'}{\partial \beta}W(m(t_n)-x(t_n))\right]^{-1}_{\beta=\beta(i)}\end{aligned} \quad (4.42)$$

Eykhoff (1974) states that:

> The Gauss-Newton method is preferable to the steepest descent on account of the quadratic convergence, although, contrary to the steepest descent method, the convergence is not guaranteed.

4.5.5 Marquardt method

The Marquardt method is, in effect, a compromise between the steepest descent method and the Gauss-Newton method.

Let $\beta(i)$ be the centre of a hypersphere in the parameter space. We seek the minimum on the hypersphere according to Lagrange minimising $E(\beta)$,

subject to the restriction.

$$\begin{aligned} \Delta\beta &= \beta(i+1) - \beta(i) \\ \Delta\beta'\Delta\beta &= \text{constant} \end{aligned}$$

Hence

$$\frac{\partial E}{\partial \Delta\beta} + \mu\Delta\beta = 0 \tag{4.43}$$

Taking a Taylor expansion of x (implies that the assumed hypersphere need not be a pure hypersphere)

$$\begin{aligned} x(i+1) &= x(i) + \left.\frac{\partial x}{\partial \beta'}\right|_{\beta=\beta(i)} \Delta\beta \\ &\text{or} \\ \frac{\partial x(i+1)}{\partial \Delta\beta'} &\approx \left.\frac{\partial x}{\partial \beta'}\right|_{\beta=\beta(i)} \end{aligned} \tag{4.44}$$

So Equation 4.43 becomes

$$-\sum_{n=1}^{T} \left.\frac{\partial x(t_n)'}{\partial \beta}\right|_{\beta=\beta(i)} W \left(m(t_n) - x(t_n) - \left.\frac{\partial x(t_n)}{\partial \beta'}\right|_{\beta=\beta(i)} \Delta\beta \right) + \mu\Delta\beta = 0$$

which implies

$$\begin{aligned} &\beta(i+1) = \beta(i) + \\ &\left[\sum_{n=1}^{T} \left.\frac{\partial x(t_n)'}{\partial \beta}\right|_{\beta=\beta(i)} W \left.\frac{\partial x(t_n)}{\partial \beta'}\right|_{\beta=\beta(i)} + \mu\mathbf{I} \right]^{-1} \sum_{n=1}^{T} \left.\frac{\partial x(t_n)'}{\partial \beta}\right|_{\beta=\beta(i)} W(m(t_n) - x(t_n)) \end{aligned} \tag{4.45}$$

In the above, if $\mu = 0$, then the update is as per the Gauss-Newton method; if $\mu = \infty$, then the update is effectively as per the steepest descent method, but with a step size of 0!

By varying μ, the convergence properties may be altered.

4.6 Introduction to Genetic Algorithms

The study of genetic algorithms (GAs) within the fields of computer science and engineering has its roots in the recently re-published monograph by John Holland (1992), first published in 1975. In that volume, GAs are presented as modelling the processes which occur during the evolution of a population of individuals under the action of what Holland describes as 'reproductive plans'. Emphasis is placed on the abilities of such plans in maximising returns (minimising losses), with the two-armed bandit problem (pp.75-88) being used as an example. It is, perhaps, important to note that GAs are not explicitly treated as function optimisers, as pointed out by De Jong (1993), although modifications have been made over the years which have rendered the basic GA structure applicable to a range of optimisation problems.

The oft-cited volume written by David E. Goldberg (1989) provides an excellent introduction to the theory and practice of genetic algorithms, giving details of both the theoretical foundations of GAs and citing examples of applications to a wide range of problems. (The title of the book in itself, *"Genetic Algorithms in Search, Optimisation, and Machine Learning"*, betrays the broad scope of GAs.) Following this book, the number of people working with GAs seems to have increased dramatically, possibly due in part to the decreasing cost of computer hardware on which to run GA programs.

Work continues on both the theoretical underpinnings of GAs, as described in conference proceedings such as the Foundations of Genetic Algorithms series (Whitley, 1993; Whitley and Vose, 1995), and on the practical applications of GAs to engineering and other problems (Davidor, 1990; GALESIA, 1995; GALESIA, 1997). The vast majority of the work done to date using GAs has been based on computer simulations, but more recently work has started on the direct application of GAs to problems in science and engineering, using GAs to search for regions of optimal experimental performance, for example, (Weuster-Botz *et al.*, 1995).

More importantly, because they make use of a population of values, genetic algorithms offer an implicitly parallel approach to the minimisation of complex, potentially multimodal, functions which is not prone to terminating at a local minimum, as gradient descent based minimisation techniques are.

The fundamental concepts of genetic algorithms are loosely based on ideas taken from biology. The parameters of the problem are encoded in a *population* of *genes* (vectors of numbers), and, over the course of a number of generations, *selection* is applied with the result that these genes *evolve* so as to progress towards optimal solutions. GAs work with a coding of the problem parameters, the *genotype*, as opposed to the parameters themselves, the *phenotype*, starting from an initial population of *chromosomes* and at each *generation* allowing the fittest chromosomes to *mate* and produce offspring in the subsequent generation. The fitness values of the chromosomes provide the only problem-specific information used by the algorithm, and so the algorithm may be applied to discontinuous as well as continuous functions, unlike calculus based optimisation techniques. In addition, the implicitly parallel

nature of the algorithm, using a population of individuals, reduces the risk of the problem becoming trapped in a local, non-global, minimum. However, there are some problems which are termed GA-deceptive, which are difficult for GAs to optimise.

Various attempts have been made to combine the best features of GAs with those of calculus based, or simulated annealing based optimisation methods. Such hybrid schemes are not considered here, as we are treating the GAs simply as a means of finding the neighbourhoods in which the optimal solutions to the various problems treated here lie. We have applied GAs to the estimation of parameters of complex, nonlinear, morphologically structured models of the penicillin fermentation, and also to searches for optimal fermentation feed profiles for model parameter estimation (based on criteria related to the Fisher Information Matrix) and for maximising an economic performance criterion for the fermentation.

4.6.1 Genetic algorithms, a HOWTO

Populations of chromosomes evolve through a number of generations; with genetic operators to perform selection for 'fitter' individuals, the population gradually drifts in the direction of an optimal point.

A *canonical genetic algorithm* as described by De Jong (1993) is given in Figure 4.4.

This canonical algorithm forms the basis for most simple GAs. It includes two of the three basic genetic operators involved in the execution of a genetic algorithm - selection and recombination. The third common operator, mutation, is applied after the recombination stage is completed.

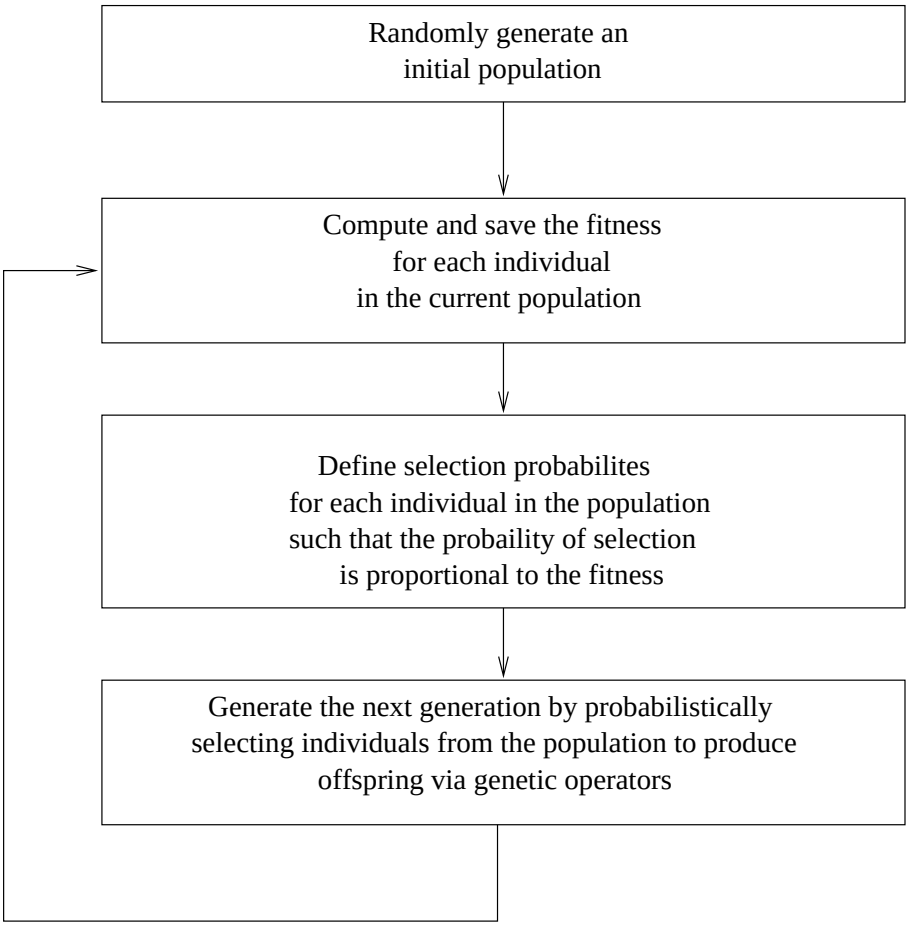

Fig. 4.4: A Canonical Genetic Algorithm

GA operators

In coding and analysing GAs, the primary focus of attention is on the operators used in the course of the algorithm. Tbe most commonly used operators are those which mimic most closely the behaviour of genetic operators in nature and are as follows.

- Selection, whereby chromosomes with higher fitness values produce more offspring in the breeding population than chromosomes with lower fitness values.
- Recombination, in which pairs of chromosomes in the breeding population exchange information by swapping portions of the chromosomes beyond a randomly determined cutting point.
- Mutation, which acts randomly, normally with a low probability, changing the value of a single bit in binary codings. In this way, schemata which have been lost from the population may be reintroduced to it, albeit with low probability. High mutation rates tend to overwhelm the convergence abilities of the GA, leading to the need for increased numbers of generations for optima to be found.

Other, less commonly used operators include the *elitist* operators which act to replace a random individual in the new population with the best individual found thus far. (Normally this individual would not have its fitness recalculated, unless the GA were being applied in conditions with dynamically varying fitness values.) Elitist algorithms, in which the GA retains the best individual found to date within the current population, have been shown to converge eventually to the global optimum (Yao and Sethares, 1994).

Another less common operator is the inversion operator, in which the worst individual in the population is replaced with the bitwise complement of an individual selected at random from the whole population. This mechanism preserves the number of schemata present in the population.

4.6.2 Analysis of convergence

Regardless of whether or not a Genetic Algorithm finds the neighbourhood of the global optimum for a given problem (which is, after all, what we are using them for), without mutation a GA will eventually converge to the point where every individual in the population is identical. Louis and Rawlins (1993) give an exposition of an analysis of the time to convergence for a population of binary strings, based on the mean Hamming distance between members of the population.

The average Hamming distance of a population is the average distance between all its members, with the Hamming distance being the summed bitwise difference between each pair of strings considered. For a population of N individuals, each individual is half of $N-1$ pairs (distance calculations), and the total number of interpair distances from which the average distance is calculated is $N(N-1)/2$: the distance from individual A to individual B is the same as that from B to A.

If the strings in the population are l bits long, then, for an initially random population, the mean Hamming distance may be approximated by a normal distribution with mean h_0 ($h_0 = l/2$) and standard deviation s_0 ($s_0 = \sqrt{l}/2$). As mentioned before, in the absence of mutation, the average Hamming distance of the converged population is zero. Over the time to convergence,

the Hamming distance drops from $l/2$ to 0, and for a simple GA this can only be due to the influences of selection and crossover.

The influence of crossover on Hamming distance

For a simple, single-point crossover operator, whereby the two parents are replaced by their offspring, the Hamming distance will be unaffected. As Louis and Rawlins (1993) point out, the sole effect of simple crossover is to change the order in which the bitwise contributions of points in each pair of strings are summed.

The influence of selection on Hamming distance

How individuals are selected for mating in the next generation depends on the problem to which the Genetic Algorithm is being applied. Louis and Rawlins (1993) suggest that selection with probability greater than 1/2 reduces the average Hamming distance from generation to generation, and attempt to obtain an upper bound for the time to convergence. Said upper bound is assumed to occur for a completely flat fitness function, which is regarded as the worst possible. On such a surface, they state that a GA can do no better than random search, as, for the function $f(x) =$ constant, no useful information may be obtained to aid any algorithm in searching for an optimum.

The convergence of a GA on a flat function is stated to be caused by *genetic drift*, whereby small random variations in the initial distribution of alleles result in their gradual accumulation and eventual convergence. The proof (Louis and Rawlins, 1993) begins by calculating the time for a single

allele to become fixed due to so-called genetic drift. The probability that k copies of an allele i are produced in the next generation is given by the binomial probability distribution

$$\left(\begin{array}{c} N \\ k \end{array} \right) p_i^k (1 - p_i)^{N-k} \tag{4.46}$$

where N is the number of individuals in the population, p_i is the proportion of allele i present in the current population.

Using this distribution, it is possible to calculate the probability of a particular frequency of occurence of allele i in subsequent generations. The solution to this problem (a classical problem in population genetics) can be approximated, for intermediate allele frequencies and population sizes. If $f(p,t)$ is the probability that the frequency of an allele has the value p in generation t $(0 < p < 1)$, then

$$f(p,t) \quad = \quad \frac{6p_0(1-p_0)}{N} \left(1 - \frac{2}{N}\right)^t$$

This specifies the probability that the allele has not converged. The probability that an allele is fixed (has converged) at generation t is obtained simply, as follows.

$$\mathcal{P}(t) = 1 - f(p,t) \tag{4.47}$$

Combining the two preceding equations, we obtain

$$\mathcal{P}(t) = 1 - \frac{6p_0(1-p_0)}{N}\left(1-\frac{2}{N}\right)^t$$

and assuming that alleles are independent of one another, which seems to be a reasonable assumption for a flat fitness function, the probability that all alleles are fixed at generation t for chromosomes of length l is given by

$$\mathcal{P}(t,l) = \left[1 - \frac{6p_0(1-p_0)}{N}\left(1-\frac{2}{N}\right)^t\right]^l$$

The above equation gives the probability of convergence of a genetic algorithm on a flat function. From it may be obtained an estimate of the upper bound on the time to convergence for a binary GA, given the population size (N) and the chromosome length (l). Since this is the probability of convergence for a function in which selection plays no role in directing the search by the GA, the equation may also be considered as providing a *lower bound* on the likelihood of the GA having converged after t generations. It should be noted, however, that this approximation is valid for 'intermediate allele frequencies and population sizes'. The exact meaning of this phrase is worth some consideration.

The probability surface described by the above equation, as a function of population size (N) and number of generations (t) is given in Figure 4.5.

Louis and Rawlins (1993) extend the above approximation for a flat fitness function, and apply it to predicting the time to convergence for more realistic problems.

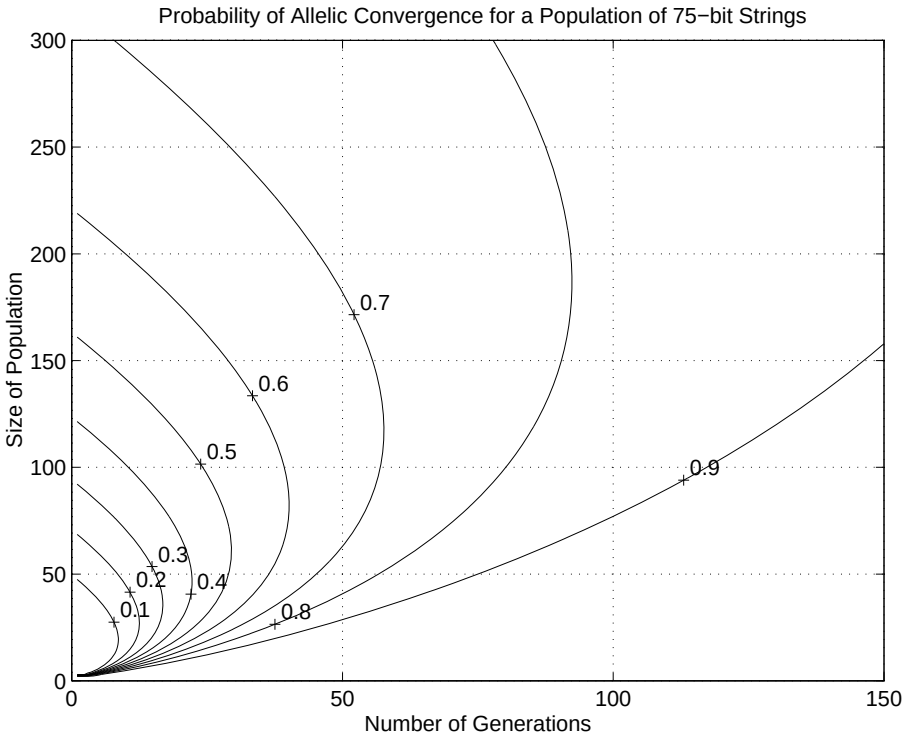

Fig. 4.5: Probability of convergence for a GA as a function of population size and number of generations (all chromosomes were 33 bits long)

> computing the rate of decrease in the Hamming average while a GA is working on a particular problem allows us to predict roughly the time to Hamming convergence. (Louis and Rawlins, 1993)

It is assumed that similarity between chromosomes implies similarity between fitness values (the similarity assumption) – this is true for unimodal functions, and for multimodal functions genetic drift is alleged, ultimately, to cause behaviour which may be predicted by the following model, unless countered by niching or other diversity-preserving schemes.

Generally, the change in average Hamming distance from generation to generation is given by

$$h_{t+1} = f(h_t)$$

which relates h_t, the Hamming average in generation t, to the Hamming average in the subsequent generation.

Assuming that $f(h_t)$ is linear, and given that, without mutation, the final Hamming average is zero, Louis and Rawlins (1993) obtain the equation

$$h_{t+1} = ah_t$$

Solving this recurrence, we obtain

$$h_t = a^t h_0$$

where h_0 is the initial Hamming average, $l/2$ for a randomly initialised population.

4.7 Selecting a Search Method

The experiment design problem addressed here is that of designing an input feed profile for the fermentation such that the best possible experiment can be performed, subject to a fixed, pre-specified measurement sample rate, and assuming that the optimal parameter set to be found will be in the neighbourhood of the parameter set thus far obtained. Input profiles may be parameterised in a number of ways:

- piecewise linear interpolation between specified values
- spline interpolation between specified points
- piecewise constant, 'stairstep', profile
- sum of exponential or periodic waveforms
- polynomial of variable order
- output of a neural network

For the work done here, the input profile has been specified as a piecewise constant, 'stairstep', profile, as such a profile may simply be obtained manually, making adjustments to the feed rate at each measurement sample time. An approach for which manual input is feasible was chosen to avoid possible complications and delays in producing a computer-controlled profile, as could be required for any of the other candidate parameterisation methods.

Having chosen the way in which the input profile is to be parameterised, it is necessary to search for profile parameters which give rise to the optimal experiment design. It has previously been stated (Munack, 1989) that using gradient descent techniques may give rise to suboptimal experiment designs, because gradient descent techniques can become entrapped in local minima, and thus terminate without finding the global optimum. Genetic algorithms (GAs) have been found to perform well on (potentially) multimodal objective functions, and have a high probability of finding the neighbourhood of the global optimum on such objective functions (Goldberg, 1989). The performances of gradient descent methods and genetic algorithms were compared, for a simple function having two minima.

4.7.1 Gradient descent-based optimisation algorithms

As an illustration of the way in which gradient descent-based optimisation routines may become trapped in local minima, consider the minimisation of the function shown in Figure 4.6. This function is based on a quartic polynomial in x, added to a minimum quadratic in y, which give a 'valley' shaped function with two minima along the 'valley'. The function has been rotated so that the lines of steepest descent are not parallel to the parameter axes. This figure has two minima, a global minimum at approximately (-3,-1) and a local minimum at approximately (3.5,1). When seeking the minimum of this function using a gradient descent technique (see Section 4.5 for details), which minimum is found depends on where the minimisation is started from. Figure 4.7 shows a contour plot of this function, with a line drawn across it, dividing those points from which the global optimum is obtained from those from which the weaker local minimum is obtained.

4.7.2 Genetic algorithms

As an alternative to gradient descent-based optimisation methods, consider the progress made using genetic algorithms (GAs) to minimise the function shown in Figure 4.6.

Applying a genetic algorithm to the simple function mentioned above, and plotted in Figure 4.6, the population of points is found to rapidly 'cluster' around the lower of the two minima of the function. The populations evaluated in the first nine generations are shown in Figure 4.8. The spread of the points is initially random, with values constrained to lie between ± 5 for all parameter pairs (subplot 1), converges first into an area covering both

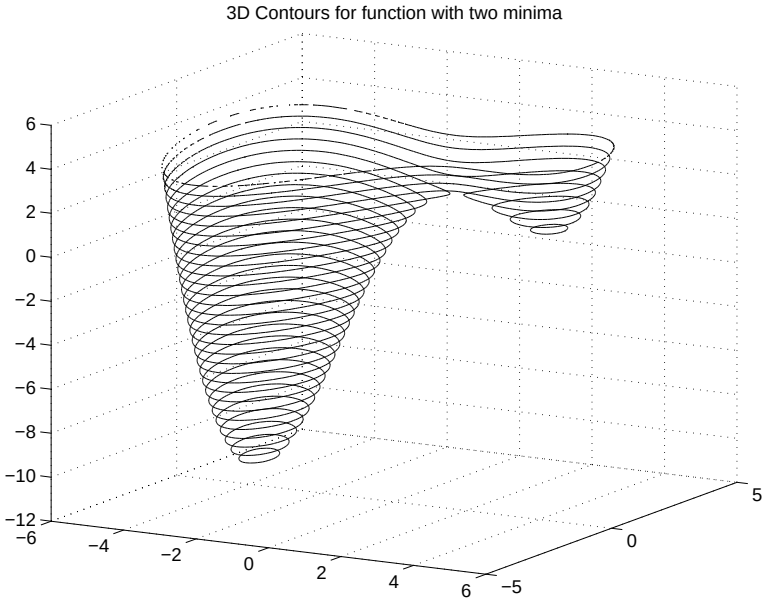

Fig. 4.6: Contours in 3 dimensions for a function with two minima

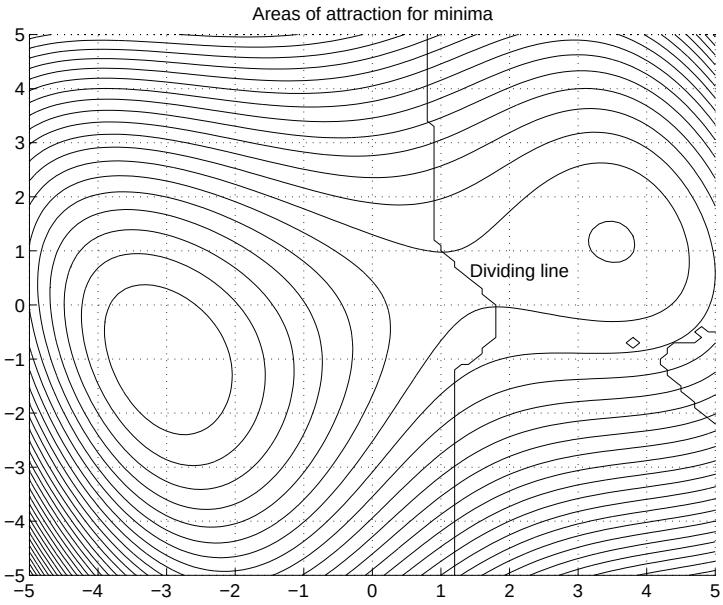

Fig. 4.7: Contours in 2 dimensions for a function with two minima, with lines added to divide areas containing points from which the two minima are reached using gradient descent minimisation methods

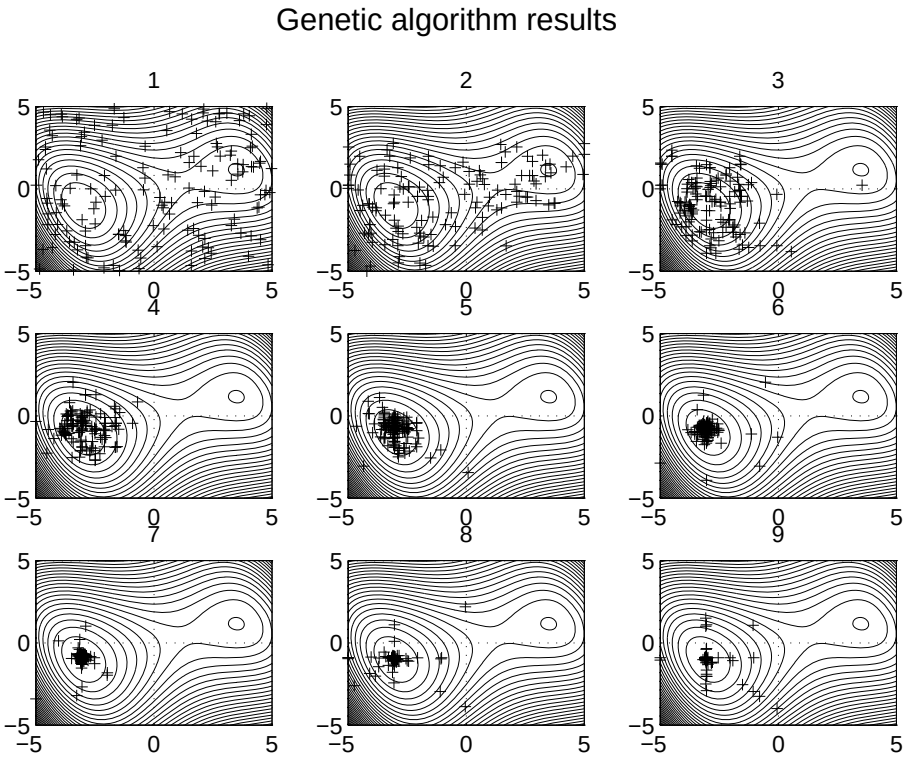

Fig. 4.8: Progress of genetic algorithm over nine generations; each '+' is a parameter pair evaluated in the generation indicated by the figure above the subplot

minima (subplot 2), and over the subsequent generations the space being searched closes down around the lower of the two minima of the objective function.

4.8 Designing Optimal Experiments Using Genetic Algorithms

Table 4.1 lists several commonly used optimal experiment design criteria, along with geometrical interpretations of their effects on ellipsoids of constant error value, centred on the optimal parameter set. Of these, the D optimality criterion, corresponding to minimising the volume of these error ellipsoids, and the E optimality criterion, corresponding to making the error ellipsoids as 'spherical' as possible, are, perhaps the most common. Of these two, D optimality was considered to be more appropriate, with smaller error ellipsoids being thought preferable to 'rounder' errror ellipsoids. For widely ranging parameter values, with a range of sensitivities of error value to parameters, the E optimality criterion could result in an experimental design for which the error ellipsoids, although 'rounder', were larger than for the undesigned case, thus implying that fixing the model parameters to lie within a larger range after a designed experiment was an improvement. A D optimal experiment design, on the other hand, should produce smaller error ellipsoids, and hence less variation in the parameters for given error values.

Optimal experiment designs have been produced based on the D optimality criterion using genetic algorithms as the search method. This section describes how the input feed profile was parameterised and how the genetic algorithm search was implemented, listing the parameters used to control the GA's operation, the range of feed values considered and the constraints applied to the optimisation, and gives details of the objective function used in the search.

4.8.1 Input feed profile parameterisation

As mentioned above in Section 4.7, the 'stairstep' profile was chosen as the way of defining the input feed profile, as it is easy to use as a part of a genetic algorithm search, and it is possible to apply the resulting designed inputs manually. After discussions with Paul (1998), the input profile was divided into piecewise constant portions each 8 hours long. The original experiment design work (Syddall *et al.*, 1998) was done on the basis of piecewise constant input profiles with 5 hour long portions, but this was felt to be infeasible, should the adjustments need to be made by hand during the fermentation. To ease manual implementation of the feed profile, each 'step' in the profile was specified as being eight hours long. The length of the fermentation was fixed at 120 hours and so the input profile was defined by fifteen parameter values. (The initial conditions for the fermentation were fixed at an 'average' set of values.) It was assumed that biomass samples are taken at eight hour intervals, coincident with the changes in the input feed rate, and that the soluble components in the fermentation broth are measured, using online HPLC, at half-hourly intervals. The information matrix considered for these designs was a multi-rate information matrix as described in Section 4.4.

4.8.2 Genetic algorithm parameters

Two different genetic algorithm implementations provided in the Genetic and Evolutionary Algorithms Toolbox (GEAT) (Pohlheim, 1996) were used, one for 'real-valued' problems (20 bit accuracy over the search range), and one for binary valued problems (using which the search was implemented using 5 bit accuracy over the same search range). The sizes of the spaces being searched

by the two methods differ greatly. For a stairstep input defined by fifteen parameters, 20 bit accuracy implies a search through 2^{20} combinations for each parameter, resulting in a search through some 2^{300} possible combinations (approximately 10^{90}). The case of 5 bit accuracy, with 2^5 combinations per parameter gives a total of 2^{75} combinations (approximately 10^{22}).

Since the accuracy with which the inputs to the fermenter can be applied is limited, and since the smaller 5 bit search space can be searched effectively using smaller genetic algorithm populations running over fewer generations than for the 20 bit search space, the 5 bit binary coded genetic algorithm was chosen for the final experiment designs.

The population sizes and numbers of generations needed for convergence were determined experimentally, running repeated designs until approximate correlation between successive results was obtained. The genetic algorithm's operating parameters determined in this way are given in Table 4.2 for both the 20 bit and 5 bit codings. Also given in the table are the probabilities of crossover and mutation used by the algorithm. Examples of scripts and functions to be used with the Genetic and Evolutionary Algorithm Toolbox (Pohlheim, 1996) are given in Tables B.5 and B.6 in Appendix B.

4.8.3 Feed rate limits and constraints

The range of input values searched using the genetic algorithms was bounded such that 0 l/hr $<$ feed rate $<$ 0.0448 l/hr. The lower bound is no feed rate, and the upper bound corresponds to a feed rate of approximately 22 g(glucose)/hr for glucose fed at a concentration of 500 g/l. This upper limit is in excess of the feed rates used by Paul (1996) in the course of the

Discretisation	20 bits	5 bits
Population size	100	50
Number of subpopulations	3	3
Percentage retained between generations	10%	10%
Number of generations	150	50
Probability of mutation, per bit	0.0005	0.025
Probability of crossover	0.7	1

Tab. 4.2: Genetic algorithm operating parameters

previous work here in the department, and the optimal experiment designs remain below this limit throughout the designed feed profiles.

In addition to defining the range of permissible input feed rates to be searched for an optimal experiment design, constraints were imposed on the fermentation behaviour. The volume in the fermenter was initially constrained to remain within the working volume of the fermenter (5l in a 6l fermenter, initial volume 4l). Experiment designs produced using this constraint gave rise to total biomass concentrations that were considered, in the judgement of Paul (1998), to be liable to cause the dissolved oxygen concentration in the fermenter to fall to levels at which oxygen availability to the organism would become limiting. Under such circumstances, the model, which does not describe changes in the dissolved oxygen concentration, nor the influence of dissolved oxygen concentration on biomass growth and product formation, would become inappropriate, and the experiment designs would no longer be valid.

A second constraint was defined in the hope of avoiding excessively low dissolved oxygen concentrations. This was a constraint on the total biomass

concentration, which was to remain below 30 g/l throughout the fermentation. Limiting the total biomass concentration should limit the biomass growth and maintenance rates, and the penicillin production rate, all of which processes consume energy and therefore oxygen. Restricting all of these rates should therefore also restrict the total oxygen demand of the organism, hopefully avoiding an excessive decrease in the dissolved oxygen tension.

4.8.4 Objective function for optimal experiment design

For the problem of finding a D optimal experiment design, the genetic algorithm is searching for that input profile which maximises the determinant of the information matrix. Since the GEAT (Pohlheim, 1996) is designed to minimise objective functions supplied by the user and uses the ranking of individuals within the population to determine their fitness rather than their absolute values, maximising a value can be replaced by either minimising the negated value or minimising the inverse of the value. Minimising the negated value was chosen here. That is to say:

$$\text{To maximise } \det(FIM)$$
$$\text{minimise } -\det(FIM)$$

When using genetic algorithms, constraints are most commonly dealt with by applying some form of penalty function to the basic objective function, so as to render those individuals which breach constraints less fit than those that do not, thereby reducing the probability that offspring of constraint-breaching individuals will be produced in the next generation.

Goldberg (1989) gives an example of a penalty function (pp 85-86).

$$\text{To minimise } g(x)$$
$$\text{subject to } h_i(x) \geq 0, i = 1, 2, \ldots, n$$

the constrained form of the objective function becomes

$$\text{minimise } g(x) + r \sum_{i=1}^{n} \Phi\left[h_i(x)\right]$$

where Φ is a penalty function (for example, the square of the violation) and r is a penalty coefficient.

For the case of constraining the experiment design criteria, such a penalty function was difficult to define, due to the great difference between the magnitudes of the constraint violations (≈ 200) and the penalty-free objective functions ($\geq$ 1e25). Since the GEAT (Pohlheim, 1996) used in this work uses a ranking-based selection method, with lower values having greater probabilities of being selected for producing the next generation, the absolute values for the chromosomes are less important than their relative order. This means that a penalty function could be constructed for the initial case of a single constraint in which the objective function returns a positive value associated with the amount by which the constraint is broken, for cases where the constraint is broken, and $-\det(FIM)$ for the others. For broken constraints, the objective function returns larger values for larger constraint breaches, so that individuals which only just breach the constraint rank better than those which grossly breach the constraint.

For the case of D optimal experiment design, attempting to maximise $\det(FIM)$, subject to a constraint that the volume remain below 5 litres, the objective function becomes

$$F = \begin{cases} \int V(t)dt & \forall V(t) > 5.0, \text{if any} V(t) > 5.0 \\ -\det(FIM) & \text{if constraints satisfied} \end{cases}$$

This penalised objective function returns negative values consistently for valid input profiles, with the objective function values becoming more negative for better experiment designs.

When the total biomass constraint was added to the experiment design criteria, the objective function was only penalised for exceeding the biomass concentration if the volume constraint was satisfied. With this ordering of penalty criteria, the model equations are only integrated numerically when it is known that the volume constraint will not be broken. Since calculating whether or not the volume constraint would be breached can be done quickly, using the net feed rate and the initial volume, without needing to numerically integrate any model equations, this ordering of penalties saves time in executing the genetic algorithm. As executing the objective function for a genetic algorithm is frequently the most time-consuming step, this ordering was considered to be more time-efficient. For example, the constrained

D-optimal objective function became:

$$F = \begin{cases} \int V(t)dt & \forall V(t) > 5.0, \text{if any} V(t) > 5.0 \\ \int X_t(t)dt & \forall X_t(t) > X_{t,MAX}, \text{if any} X_t(t) > X_{t,MAX} \\ -\det(FIM) & \text{if all constraints satisfied} \end{cases}$$

where $X_t(t)$ is the total biomass concentration at time t and $X_{t,MAX}$ is the biomass constraint value (30g/l for our purposes). In practice, the integrated values for $V(t)$ and $X_t(t)$ were approximated by summating over all values for which their respective constraints were exceeded.

Implementation detail

Since objective function evaluation is the major contributor to the time taken to carry out a genetic algorithm search, the objective function was coded in such a way as to avoid carrying out the time consuming numerical integration of the model and its associated information matrix calculations in cases where constraints were violated.

Firstly, the feed profile values were checked to see whether or not the volume constraint would be breached as follows:

- calculate the net feed rate in each 8 hour period, by adding the precursor addition rate to the glucose feed rate and subtracting the sample removal rate
- calculate the total volume change in each 8 hour period (by multiplying the net feed rate by 8)

- calculate the cumulative sum of the volume changes

The resulting vector of volume changes was then compared to the permissible change in volume (1l), and only if all elements of the vector satisfied the constraint was the model numerically integrated. In the early stages of the genetic algorithm search, this quick check saves a lot of time, since the proportion of individuals which breach the volume constraint is initially high.

The possibility of numerically integrating the model without its associated information matrix calculations was considered, but was not implemented, due to time constraints.

4.9 Results

Three sets of optimal experiment designs were produced for three differing sets of search criteria, all using two-rate FIMs as the basis for determinant calculations.

- 20 bit ('real-valued') search constrained only on feed and working volume
- 5 bit search constrained only on feed and working volume
- 5 bit search constrained on feed, working volume and on total biomass concentration

Five replicate genetic algorithm searches were run for each design criterion, using genetic algorithm routines from the GEAT (Pohlheim, 1996), with the genetic algorithm operating parameters specified in Table 4.2.

The values used for these constraints are described above. Although the 20 bit search was superseded, for reasons of speed and achievability, by the 5 bit search, its results are presented for comparison with those achieved using the 5 bit search method. The results obtained using the first two sets of search criteria are given briefly, with those obtained using the third, most realistic, set of search criteria being given in more detail. In the summary table of determinant values found, Table 4.3, determinant values based on a single-rate FIM sampling at 8 hour intervals are given for the same experiment designs as were produced for the 5 bit constrained design criterion.

4.9.1 Experiment designs – real-valued

The experiments were designed to be optimal with respect to the D optimal experiment design criterion given above in Table 4.1. The only constraints on the GA search were the minimum and maximum input feed rates, and the total volume permissible in the fermenter. The resulting determinant values are given in Table 4.3. The corresponding feed profiles and volumes over the course of fermentation are shown in Figure 4.9, and the simulated values for the biomass states and for the modelled soluble species (glucose, lactose and penicillin) are shown in Figure 4.10.

4.9.2 Experiment designs – 5 bit binary-valued, unconstrained

The only constraints on the GA search were the minimum and maximum input feed rates, and the total volume permissible in the fermenter. The experiments were designed to be optimal with respect to the D optimal experiment design criterion given above in Table 4.1. The resulting determinant

Coding	Determinant				
	Run 1	Run 2	Run 3	Run 4	Run 5
20 bit	0.5e31	1.9e31	2.8e31	3.1e31	4.5e31
5 bit (unconstrained)	8.4e26	1.0e28	7.1e29	8.6e29	1.3e30
5 bit (constrained)	1.0e26	1.9e26	3.9e26	6.9e26	1.9e27
5 bit (single-rate)	0.2e11	1.1e11	2.0e11	2.8e11	1.7e12

Tab. 4.3: Determinant values for five experiment designs for the 20 bit ('real-valued') coding, and for the 5 bit binary valued genetic algorithm coding, both with and without the constraint on total biomass concentration, sorted into ascending order

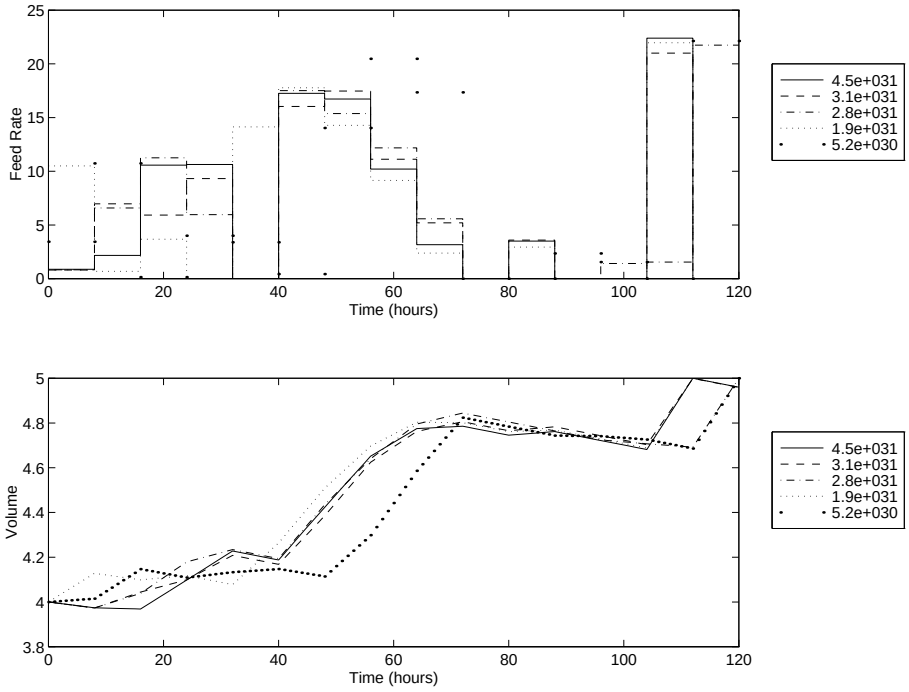

Fig. 4.9: Input feed rates and simulated volume profiles for 20 bit D optimal experiment designs based on the model of Paul *et al.* (1998) (Feed rate in g/hr at 500g(glucose)/L, Volume in L)

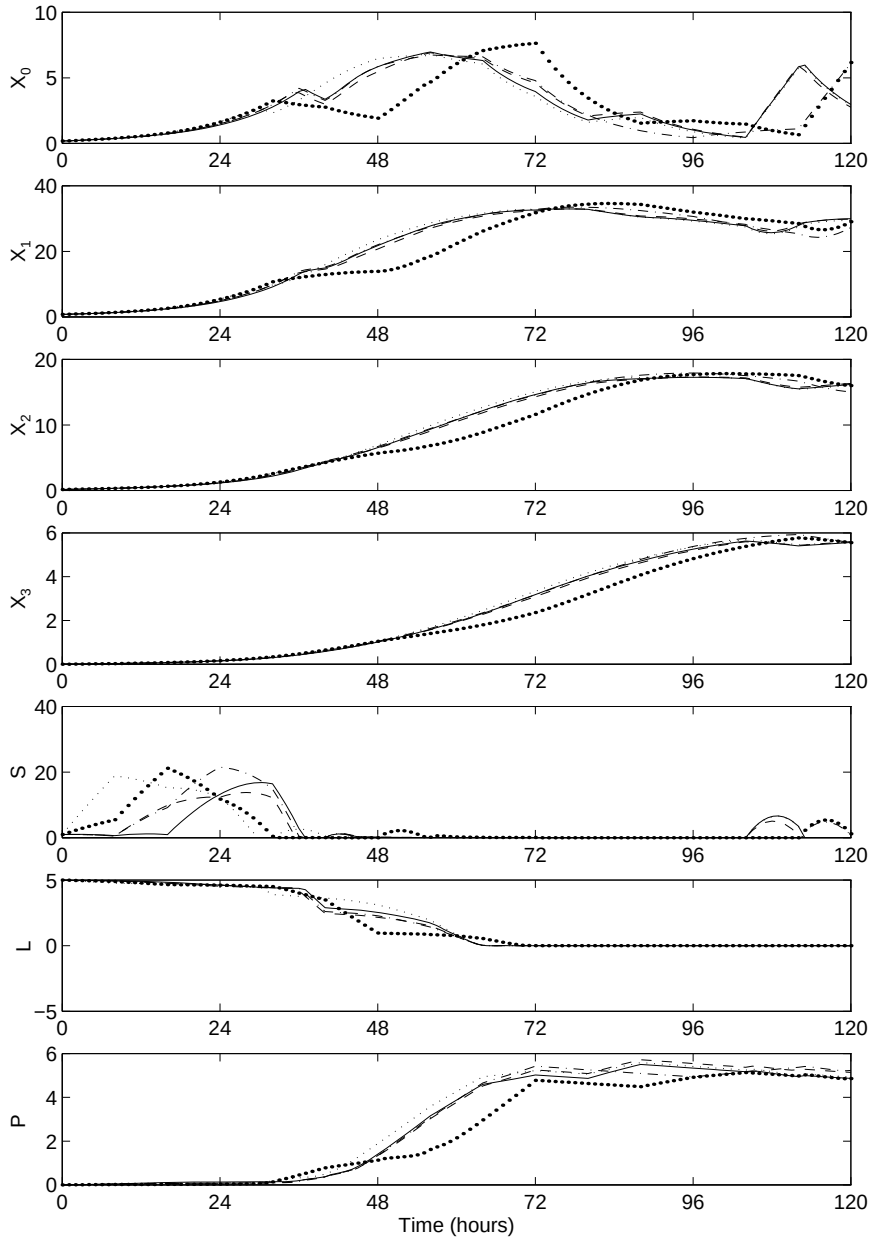

Fig. 4.10: Simulated biomass and soluble species concentrations for a 20 bit D optimal experiment design based on the model of Paul *et al.* (1998)

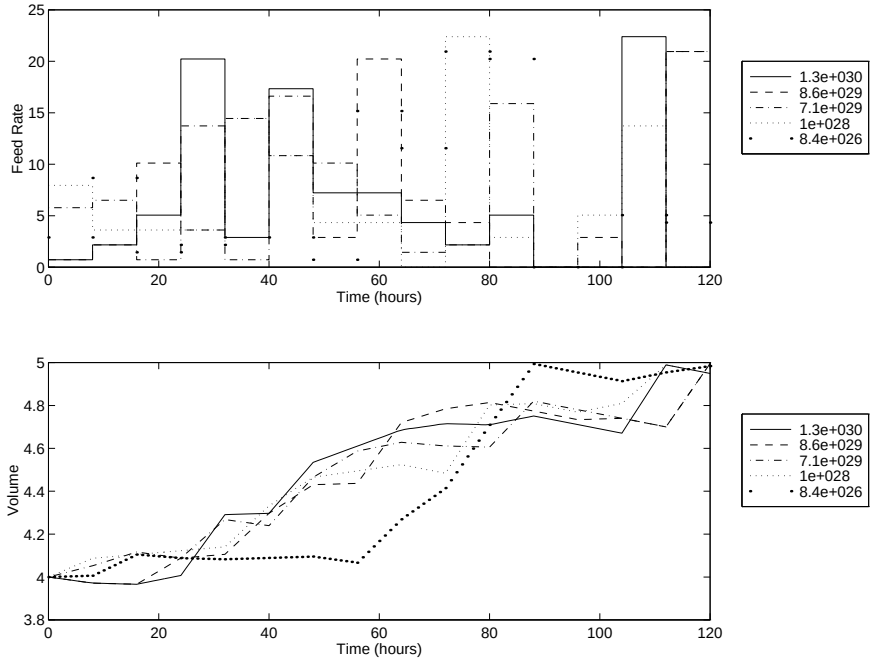

Fig. 4.11: Input feed rates and simulated volume profiles for unconstrained 5 bit D optimal experiment designs based on the model of Paul *et al.* (1998) (Feed rate in g/hr at 500g(glucose)/L, Volume in L)

values are given in Table 4.3. The corresponding feed profiles and volumes over the course of fermentation are shown in Figure 4.11, and the simulated values for the biomass states and for the modelled soluble species (glucose, lactose and penicillin) are shown in Figure 4.12.

4.9.3 Experiment designs – 5 bit binary-valued, constrained

In addition to the minimum and maximum input feed rate constraint, and the total permissible volume constraint, an additional constraint on simulated total biomass concentration was added in the hope of avoiding low dissolved

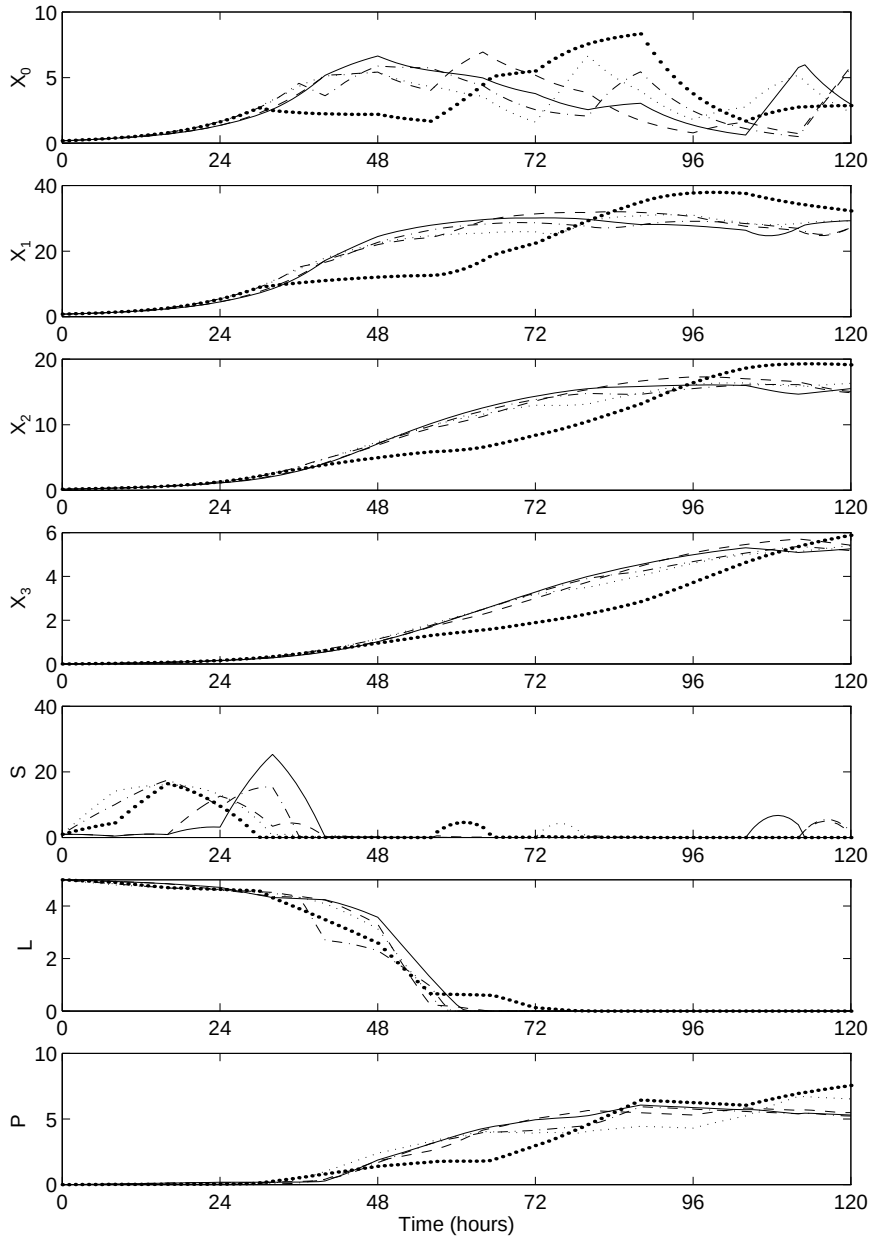

Fig. 4.12: Simulated biomass and soluble species concentrations for unconstrained 5 bit D optimal experiment designs based on the model of Paul *et al.* (1998)

oxygen concentrations during the practical experiment. The experiments were designed to be optimal with respect to the D optimal experiment design criterion given above in Table 4.1. The resulting determinant values are given in Table 4.3, and the input profile corresponding to the best determinant value is given in Table 4.4. The corresponding feed profiles and volumes over the course of fermentation are shown in Figure 4.13, and the simulated values for the biomass states and for the modelled soluble species (glucose, lactose and penicillin) are shown in Figure 4.14.

The D optimal experiment design criterion is derived from the Fisher Information Matrix, which in turn depends on the sensitivities of the states to the model parameters at the sample intervals. Graphs showing the sensitivities of the model states $\{X_0, X_1, X_2, X_3, S, L, P\}$ are shown in Figures 4.15 to 4.21. These state sensitivity profiles were calculated using Equation 4.15 from Section 4.2.1 and so depend on the variation of the glucose concentration with time during the simulation.

Indirect effects may be observed in, for example, the sensitivity profile for $\partial X_0/\partial \mu_3$ (the sensitivity of the concentration of hyphal tips to a vacuole degeneration coefficient). The equation describing the change in X_0 with time does not contain μ_3; μ_3 only occurs in the equations describing the variation with time of X_1, X_2 and X_3. However, the change in X_0 does depend on X_1, which depends more directly on the value of μ_3, and so changes in the value of μ_3 have an indirect effect on X_0.

The graphs of state sensitivities to parameters shown in Figures 4.15 to 4.21 may be divided into three categories.

1. those in which the state is largely insensitive to the parameter, such as

$\partial X_0/\partial \mu_3$, $\partial S/\partial \mu_1$, and $\partial P/\partial \mu_a$

2. those in which the sensitivity of the state to the parameter changes gradually throughout the fermentation, such as $\partial X_1/\partial \alpha_0$, $\partial X_2/\partial m_0$, and $\partial X_3/\partial m_1$

3. those in which the graph shows three phases, corresponding, broadly, to changes in the glucose concentrations

   (a) the initial, high glucose concentration (for times less than 30 hours)

   (b) intermediate values (for times between 30 and 90 hours)

   (c) the final, low glucose concentration (for times greater than 90 hours)

   (The sensitivities of the lactose concentration to the parameters depend more strongly on the lactose concentration itself, with the lactose concentration having 'intermediate' values for fermentation times between 30 and 60 hours.)

The sensitivity traces are related to the substrate concentrations. Were this not the case, attempting to design improved experiments for model parameter estimation by modifying the glucose feed profile to the fermentation would be impossible.

4.10 Discussion

Although the determinant values obtained for the 20 bit design are greater (better) than those obtained for the corresponding 5 bit design, it is not

Tstart (hr)	Tstop (hr)	Feed Rate (g)/hr)
0	8	13.7
8	16	1.4
16	24	0.0
24	32	0.0
32	40	13.0
40	48	9.4
48	56	7.9
56	64	3.6
64	72	0.7
72	80	2.9
80	88	7.9
88	96	0.0
96	104	0.0
104	112	0.0
112	120	18.1

Tab. 4.4: Feed profile specification for the best 5 bit constrained D optimal experiment design (glucose fed at a concentration of 500g/L)

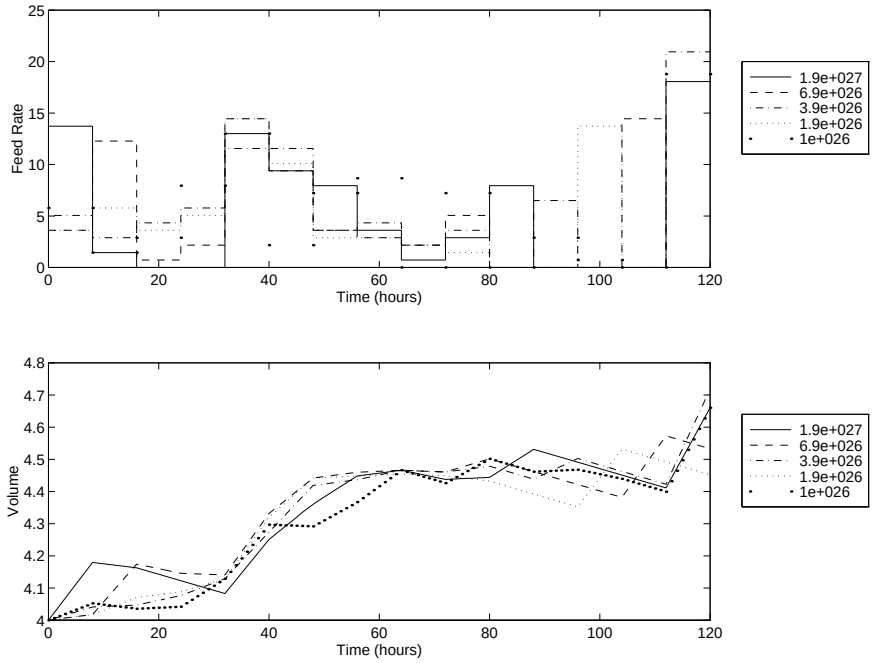

Fig. 4.13: Input feed rates and simulated volume profiles for constrained 5 bit D optimal experiment designs based on the model of Paul *et al.* (1998) (Feed rate in g/hr at 500g(glucose)/L, Volume in L)

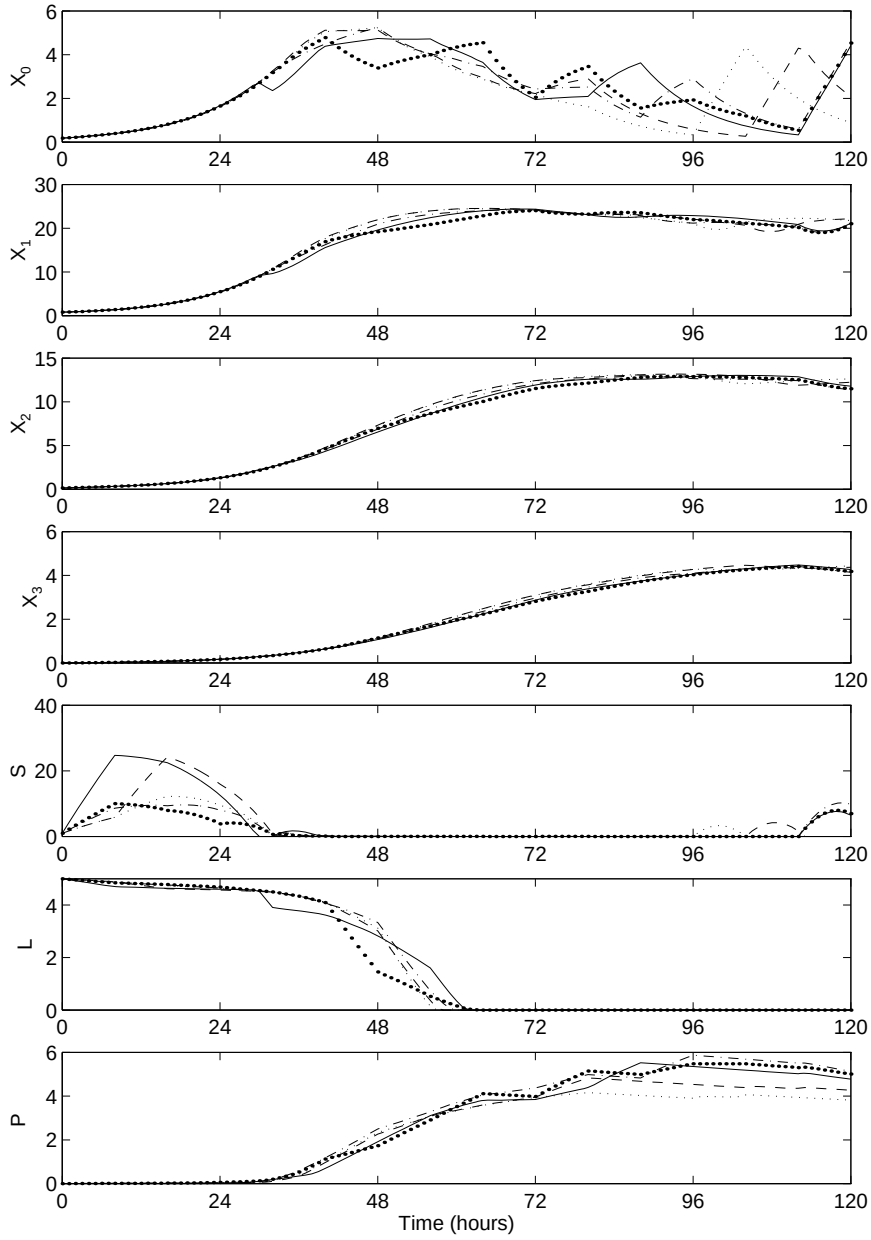

Fig. 4.14: Simulated biomass and soluble species concentrations for constrained 5 bit D optimal experiment designs based on the model of Paul *et al.* (1998)

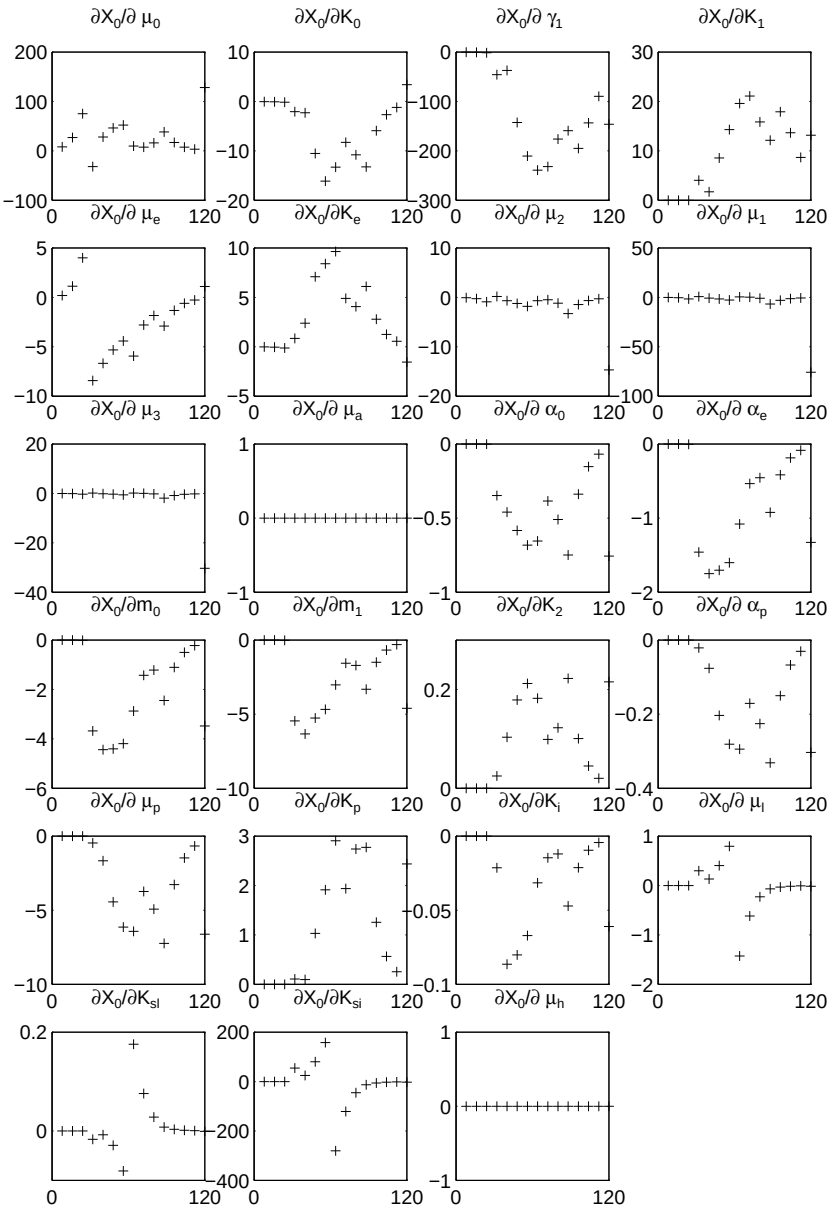

Fig. 4.15: Sensitivity of X_0 to the model parameters for a 5 bit constrained D optimal experiment design based on the model of Paul *et al.*(1998). The '+' correspond to the 8-hour sample intervals.

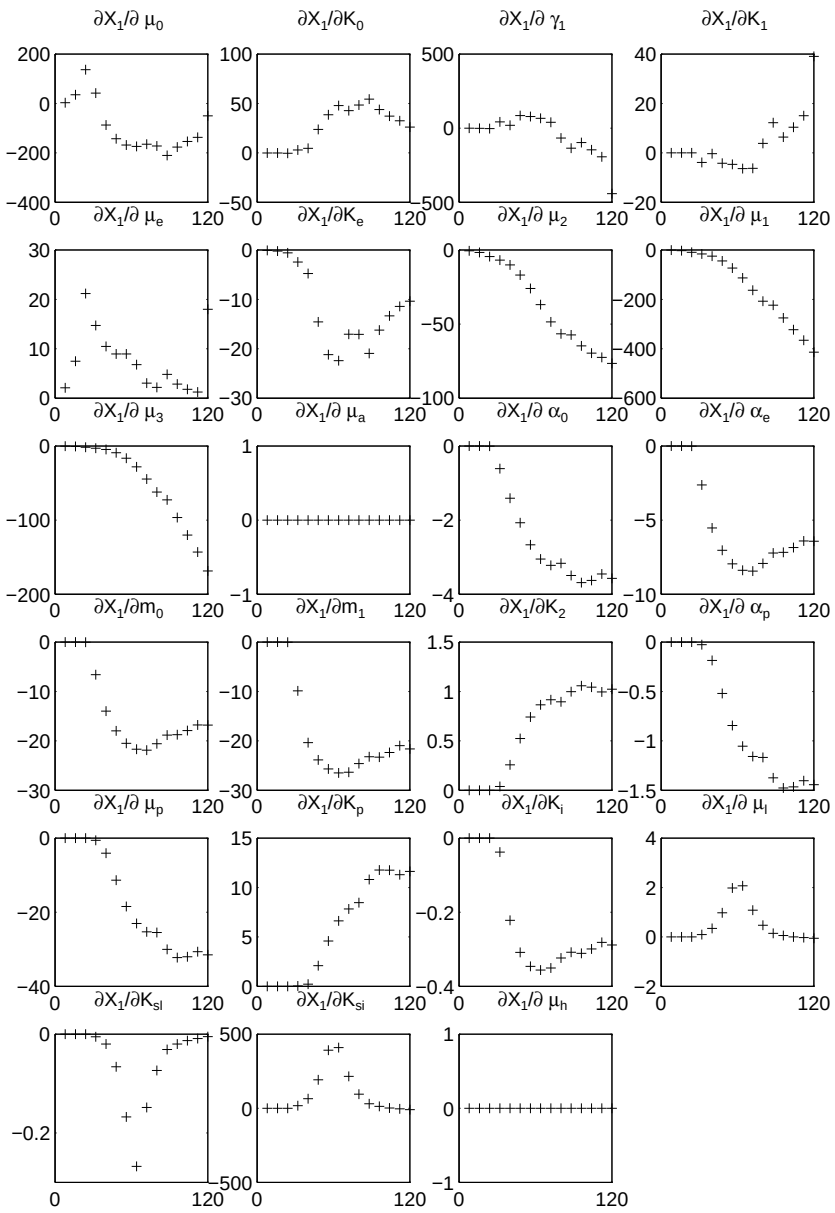

Fig. 4.16: Sensitivity of X_1 to the model parameters for a 5 bit constrained D optimal experiment design based on the model of Paul *et al.*(1998). The '+' correspond to the 8-hour sample intervals.

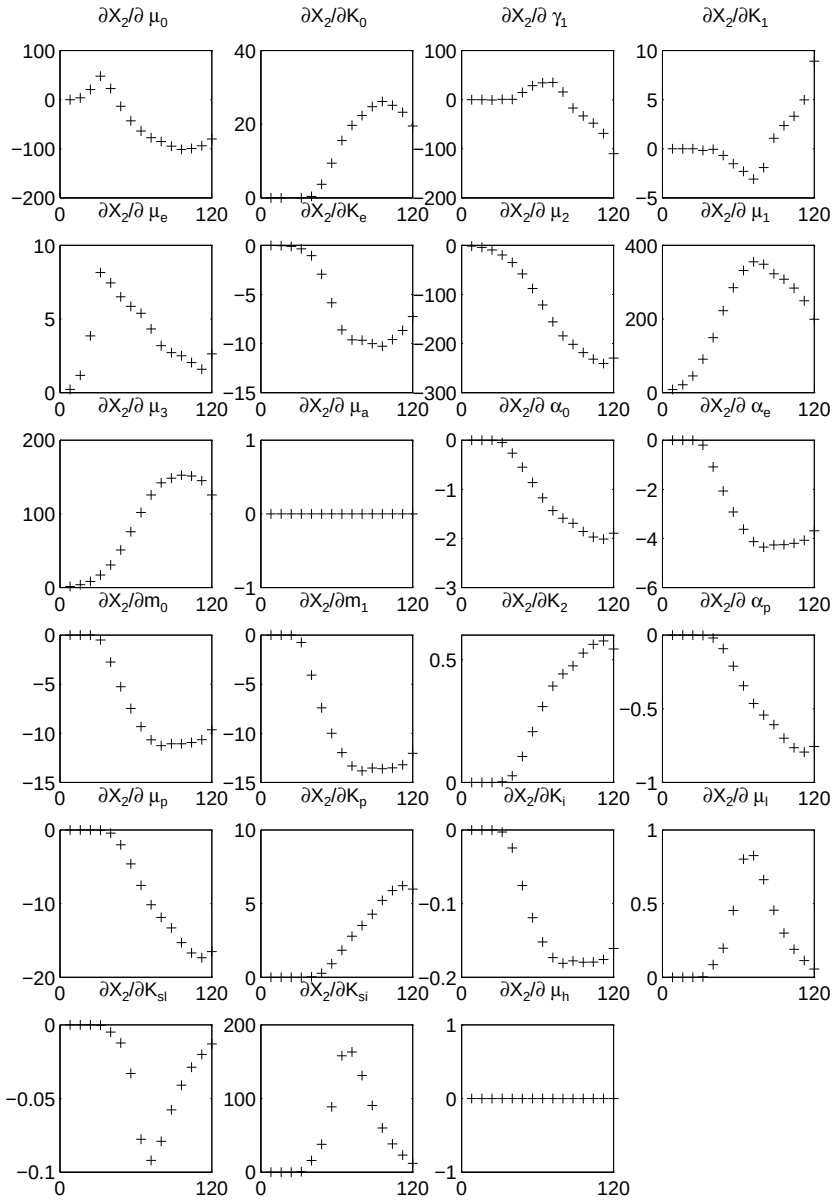

Fig. 4.17: Sensitivity of X_2 to the model parameters for a 5 bit constrained D optimal experiment design based on the model of Paul *et al.*(1998). The '+' correspond to the 8-hour sample intervals.

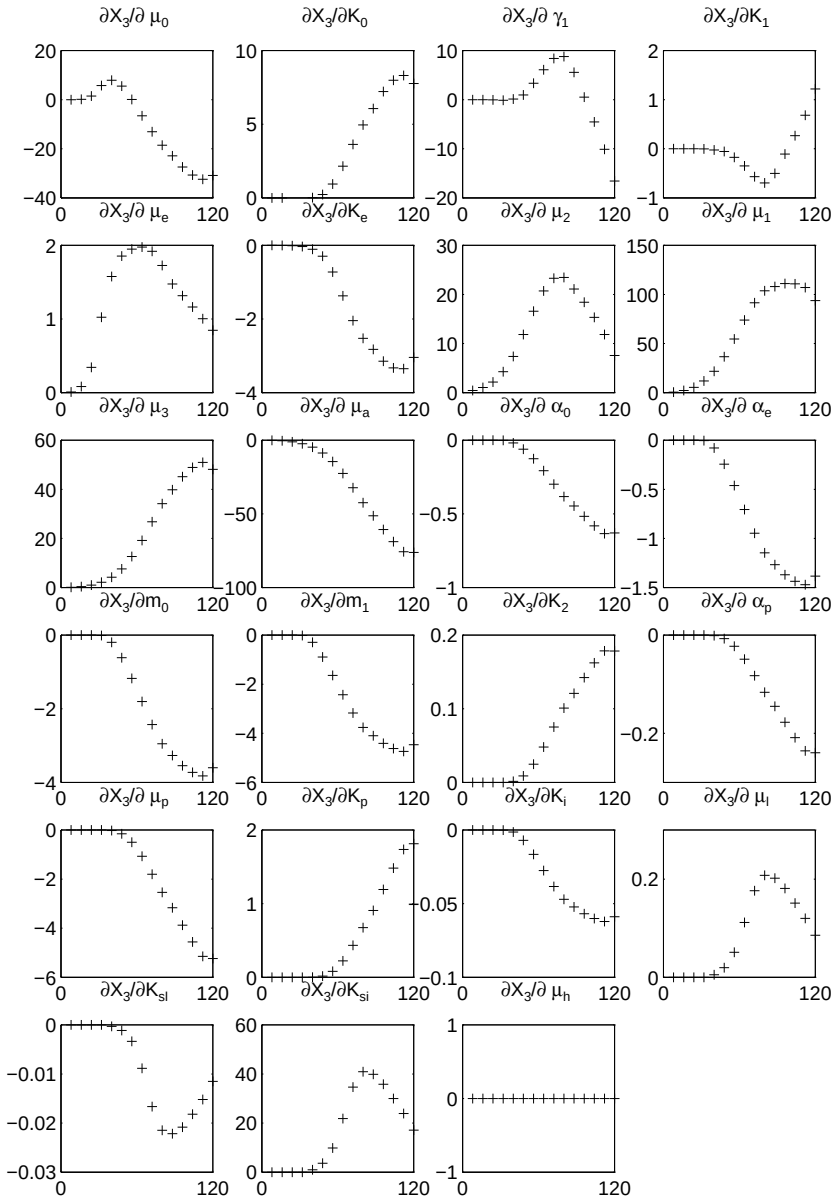

Fig. 4.18: Sensitivity of X_3 to the model parameters for a 5 bit constrained D optimal experiment design based on the model of Paul *et al.*(1998). The '+' correspond to the 8-hour sample intervals.

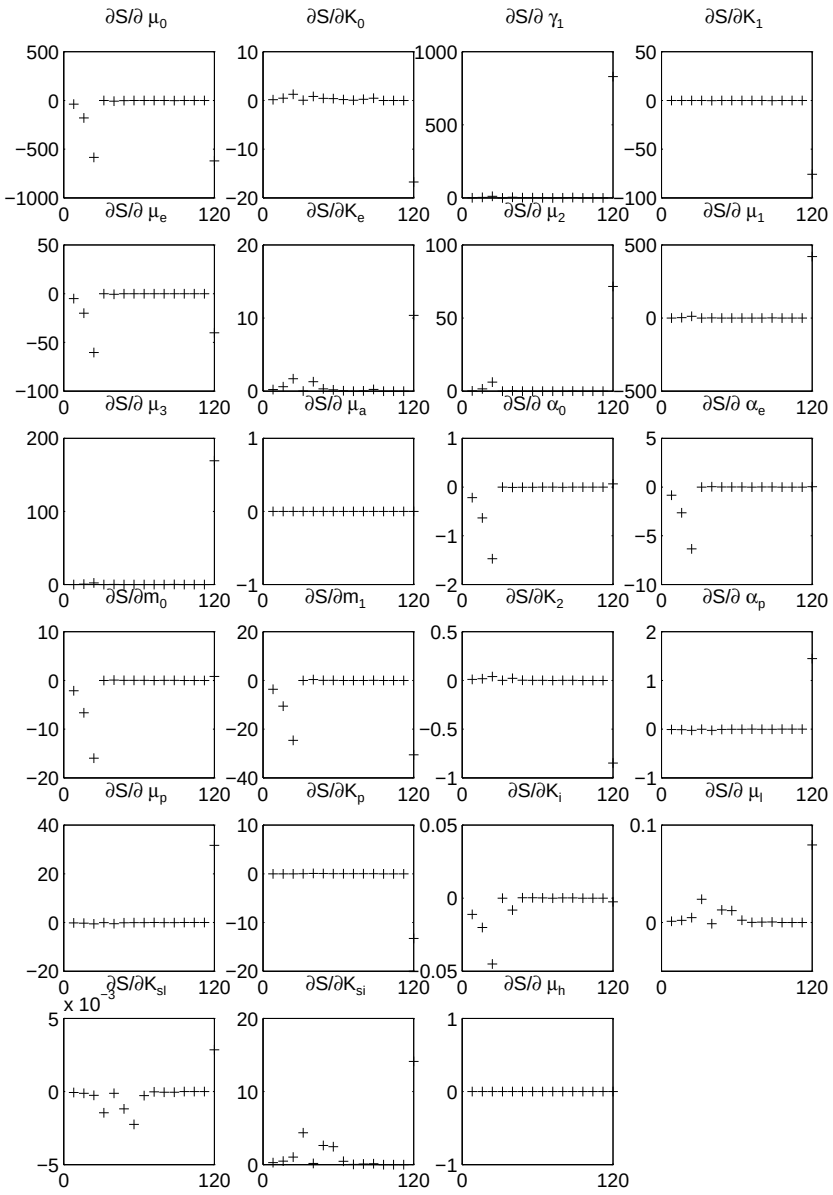

Fig. 4.19: Sensitivity of S to the model parameters for a 5 bit constrained D optimal experiment design based on the model of Paul *et al.*(1998). The '+' correspond to the 8-hour sample intervals.

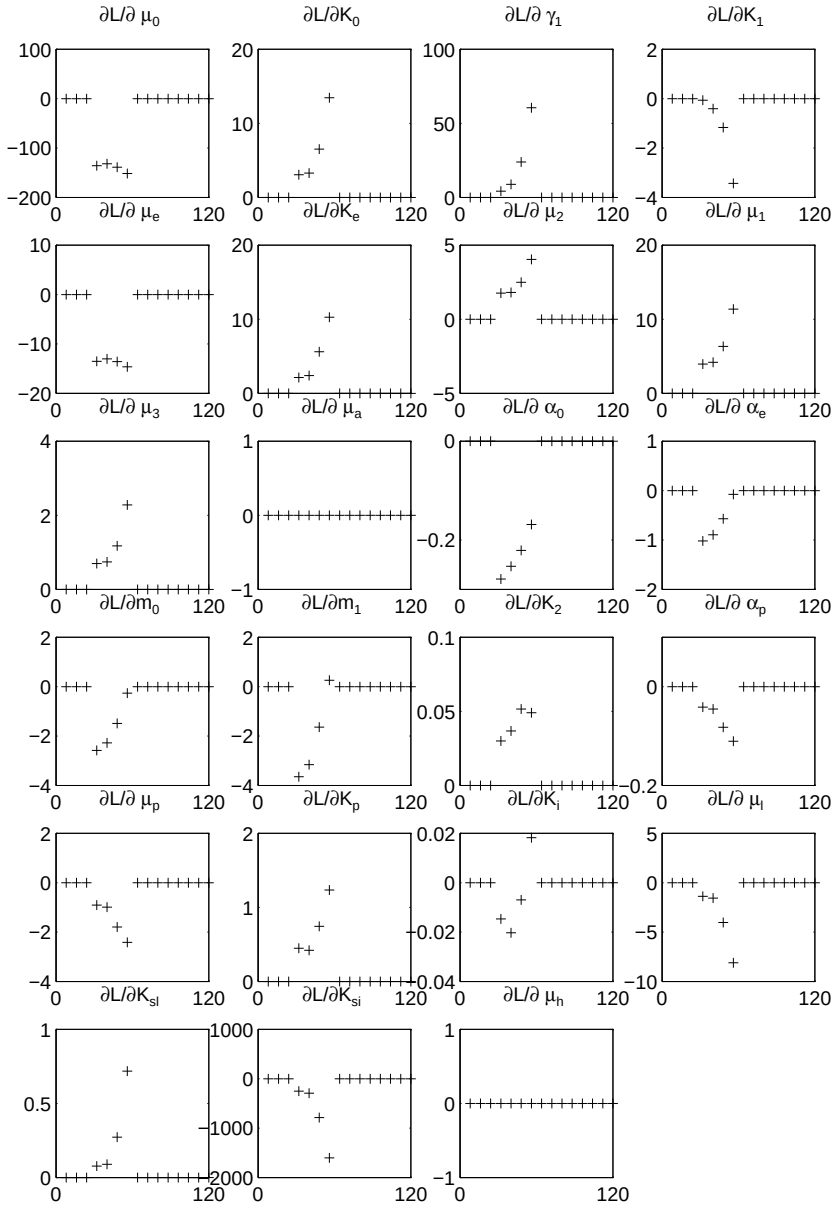

Fig. 4.20: Sensitivity of L to the model parameters for a 5 bit constrained D optimal experiment design based on the model of Paul *et al.*(1998). The '+' correspond to the 8-hour sample intervals.

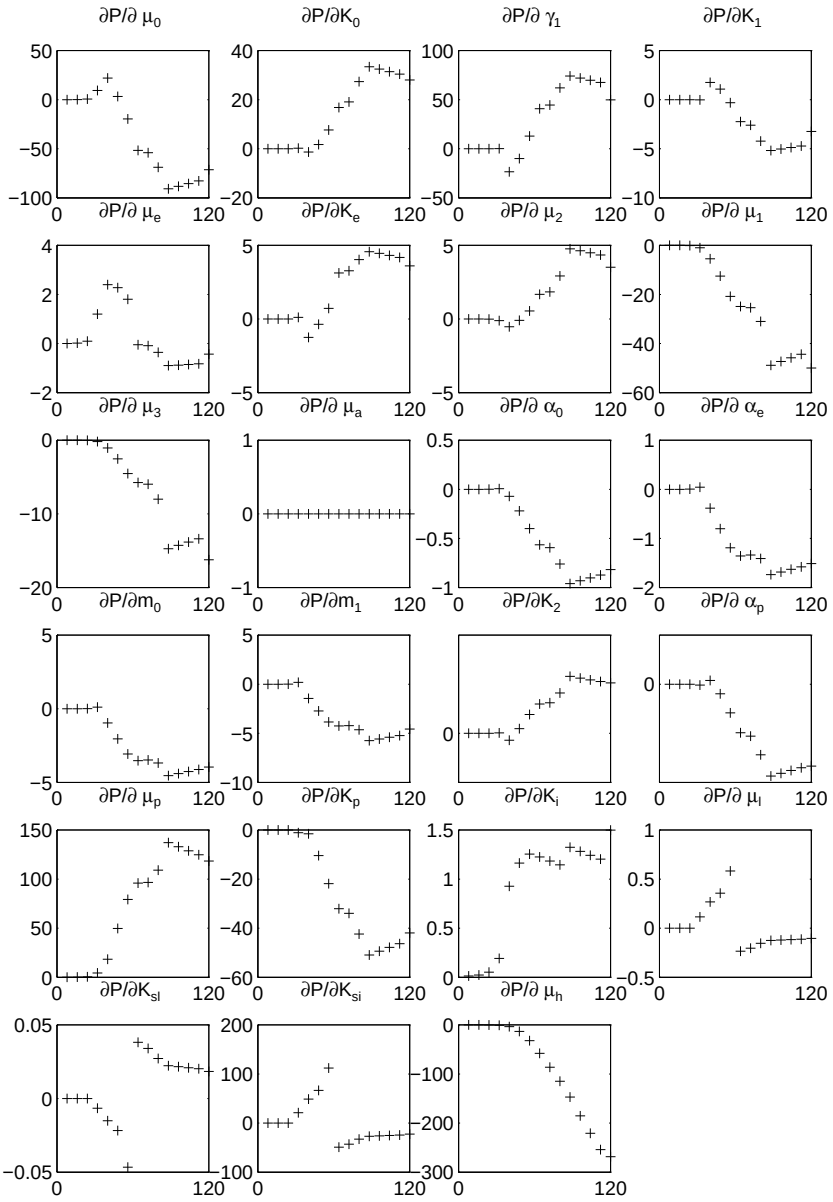

Fig. 4.21: Sensitivity of P to the model parameters for a 5 bit constrained D optimal experiment design based on the model of Paul *et al.*(1998). The '+' correspond to the 8-hour sample intervals.

possible to precisely implement the designed 20 bit feed profile. Discussions with Paul (1998) suggested that the feed rate could be controlled in steps of around 0.5 to 1.0 g(glucose)/hr. The upper bound on the input feed profile corresponds approximately to a feed rate of 22g(glucose)/hr, and so dividing the range into 22 to 44 divisions would correspond to the achievable intervals. On this basis, using 5 bit designs, having $2^5(= 32)$ divisions of the input range was considered to be adequate for practical purposes.

The difference of 5 orders of magnitude between the 5 bit and 20 bit designs corresponds, on average, to reducing the range over which each of the 23 model parameters may vary for a given error value, for the 20 bit case, to two thirds of the corresponding range for the 5 bit case. The addition of the total biomass concentration constraint to the 5 bit design crierion results in a further decrease in the determinant values obtained.

The difference in the range of determinant values obtained for the constrained and unconstrained 5 bit experiment designs may be due to changes in the size and shape of the search space containing acceptable feed profiles. For the unconstrained design, the whole of the search space is acceptable, but adding constraints reduces the size of the search space and may produce an acceptable search space which is non-convex or discontinuous. Since the search space for the constrained case is smaller than for the unconstrained case, the range of possible determinant values is narrower and this may go some way towards explaining the narrower range of determinant values found in the constrained case.

The determinant values obtained for the experiment designs have been compared with those obtained for a number of simpler input profiles. These

are:

- a constant feed rate, feeding the same total volume to the fermenter as the best 5 bit constrained designed profile
- two square profiles, stepping between values that are half of and one and a half times the constant feed rate value:
  - low-high, starting with the feed rate set to half the constant feed rate value
  - high-low, starting with the feed rate set to one and a half times the constant feed rate value
- a ramped feed rate, starting from zero feed and rising over the 120 hours of the fermentation to a final value double the constant feed rate (therefore delivering the same total feed to the fermenter)

All these simple feed profiles result in determinant values which are less than those for the designed experiments (see Table 4.5). The square wave profiles are the best performing of the simple feed profiles with respect to the D optimal experiment design criterion, but even they give rise to determinant values which are several orders of magnitude smaller than the designed feed profiles. The reason for the extremely poor performance of the ramped feed profile is not clear, but may be due to such a profile causing much smaller changes in the values of the biomass, substrate and penicillin concentrations during the fermentation than the other inputs, and so producing data over a smaller portion of the model's state space than the other input profiles.

Feed Profile	Determinant	
	two-rate FIM	single-rate FIM
Constant feed	8.2e12	6.7e-3
Square wave (low-high)	1.7e18	2.7e5
Square wave (high-low)	8.6e18	4.9e5
Ramped feed	7.0e-2	2.4e-15

Tab. 4.5: Determinant values for simple feed profiles, calculated for both single-rate and two-rate information matrices

One way of assessing the effectiveness of this experiment design process would be by examining the parameter values obtained, and their confidence intervals, on tuning to data obtained from actually running the optimal experiment designs. Work on running such a design has been planned, and is intended to be carried out in the near future. Parameter estimation based on the data obtained should be carried out using SIMUSOLV and an existing ACSL model, since SIMUSOLV may be used to produce estimates of the parameter confidence intervals for the model parameters. Comparing the parameter confidence intervals obtained in tuning to data from the experiment design with those for the parameters tuned from earlier experiments will provide a means of assessing whether or not the designed experiments may lead to an improvement in the parameter confidence intervals.

4.11 Notation

A	diagonal matrix
CPR	carbon dioxide production rate, $g(CO_2)h^{-1}$
D	diagonal matrix with eigenvalues along the diagonal
E	summed squared error

E_0	minimum value of the summed squared error
FIM	Fisher Information Matrix
H	a schema
L	lactose concentration, g(L)l^{-1}
N	number of individuals in a genetic algorithm's population
P	concentration of Penicillin, g(P)l^{-1}
$\mathcal{P}(t)$	probability that an allele in a chromosome is fixed
S	glucose concentration, g(S)l^{-1}
V	matrix made up, row-wise, of eigenvectors of the FIM
V	volume in the fermenter, l
V	binary alphabet of alleles, {0,1}
V^+	augmented binary alphabet, $\{0, 1, *\}$
W	weighting matrix
W_1	weighting matrix for measurement rate 1
W_2	weighting matrix for measurement rate 2
X	biomass concentration, g(DW)l^{-1}
X_t	total biomass concentration, g(DW)l^{-1}
X_0	concentration of hyphal tips, g(DW)l^{-1}
X_1	concentration of subapicial regions, g(DW)l^{-1}
X_2	effective concentration of vacuoles, g(DW)l^{-1}
X_3	concentration of degenerated regions, g(DW)l^{-1}
a	length of axis of a confidence ellipsoid
a	nominal model parameter
a_{11}	first element on the diagonal of matrix A
a_{22}	second element on the diagonal of matrix A
$a_* * = 1, 2 \ldots n$	position along a chromosome
b	optimum parameter set
b	length of axis of a confidence ellipsoid
b	nominal model parameter
c	nominal model parameter
$f(x(t), \beta, u(t))$	function relating rate of change of the model states $x(t)$ to the states, $x(t)$, parameters, β, and inputs, $u(t)$
$f(i)$	fitness of individual i
$f(p, t)$	frequency of allele having value p in generation t
$g(x(t), \beta, u(t))$	function relating the model output $y(t)$ to

	the states, $x(t)$, parameters, β, and inputs, $u(t)$
$g(x)$	general genetic algorithm fitness function
h_0	mean Hamming distance of initial population
$h_i(x)$	constraint function
k	cardinality of alphabet (number of characters)
l	length of axis of error ellipsoid
l	length of chromosome
m	measured value
m_1	value measured at rate 1
m_2	value measured at rate 2
$m(H,t)$	number of members of schema H in generation t
n	number of individuals in genetic algorithm's population
$o(H)$	order of schema H
$p(i)$	probability of string i being selected for reproduction
p_c	probability of crossover at a given point
p_{sc}	probability of a schema surviving single point crossover
p_m	probability of mutation occuring at a given point
p_{sm}	probability of a schema surviving mutation
s_0	standard deviation of Hamming distances in initial population
t	time, h
t_i	measurement time i
u	inputs
x	model states
x	first model state
x_1	first element of x vector
x_2	second element of x vector
x_1	simulated values sample at measurement rate 1
x_2	simulated values sample at measurement rate 2
x_a, x_b, x_c	derivatives of first model state wrt. nominal parameters
y_a, y_b, y_c	derivatives of second model state wrt. nominal parameters
y	model outputs
y	second model state
z^*	modified parameter space in Newton-Raphson method

Greek Symbols

Γ	constant modifying length of step in gradient descent

$\Phi[h_i(x)]$	penalty function applied for constrained genetic algorithm
α	coefficient relating CPR to $\dot{X}$
β	coefficient relating CPR to X
β	parameter set
$\hat{\beta}$	estimate of parameters, β
$\Delta\beta^*$	modified distance from optimum parameter set
γ	coefficient relating CPR to $\dot{P}$
$\delta(H)$	length of schema H
λ	eigenvalue of the FIM
μ	variable modifier used in Marquardt's method
μ_3	vacuole degeneration coefficient in model of Paul *et al.* (1998)
$*$	wildcard

5. CONSIDERING MODEL IDENTIFIABILITY

The problem of model identifiability is that of determining whether or not, for a given model structure, there is in theory only one set of parameters for which given model input(s) will generate given model output(s). This is important as the model parameters may be physically significant, and there may be interest in knowing whether or not they can be estimated uniquely for a given set of experimental measurements, or there may be difficulties in using numerical search techniques if there is more than one possible set of parameters for which the model will fit the data (Ljung and Glad, 1994). It may also be true that only a few variables are available for measurement; when fitting a model to even a limited set of data, parameter estimation will always give some kind of answer. However, if there exist more than one possible set of parameters (possibly even an infinite number of sets of parameters), the parameter estimates obtained may be of little practical use (Vajda *et al.*, 1989).

In this chapter, the problem of model identifiability is introduced, approaches to tackling the identifiability problem taken from the engineering literature are described, and a new approach to assessing the global identifiability of models is introduced. Examples then demonstrate that the new approach produces results comparable with those obtained using the approaches

taken from the literature, and this chapter concludes with an identifiability analysis, using the new approach, of the model of Paul *et al.* (1998). The analysis shows that the model is theoretically globally identifiable.

5.1 The Problem

Assuming a typical structure for the nonlinear model:

$$\dot{x}(t) = f(x(t), u(t), t, \theta) \qquad x(t), u(t), \theta \in \Re^n, t \in [0, T]$$
$$y(t) = g(x(t), \theta)$$

with $x(t), y(t)$ and $u(t)$ being vector-valued, time-varying model states, outputs and inputs and θ being a vector of parameter values. The derivatives of the states, $\dot{x}(t)$ are given by the set of nonlinear differential equations $f(x(t), u(t), t\theta)$ and the outputs by the nonlinear relations $g(x(t), \theta)$. $\Re^n$ denotes a real-valued, n-dimensional vector space.

For this model structure, the parameter vector θ is *locally identifiable* if for almost any solution $\hat{\theta}$ the solution is unique in some neighbourhood of $\hat{\theta}$. A model is *globally identifiable* if the conditions for local identifiability apply over the whole of the parameter space, not just a neighbourhood of $\hat{\theta}$ (Jacquez and Greif, 1985).

For a model to be of use in describing a fermentation process, or in the development of estimators and controllers, or in the optimisation of the economic performance of a process, it must first be tuned against measured experimental data. There are three possible outcomes of this process.

- The model is globally identifiable; there is a single, unique set of pa-

rameters for which the model fits the data.

- The model is locally identifiable; within a given range of parameter values (all positive, for example) there is only one set for which the model fits the data.

- The model is unidentifiable; there exist more than one set of parameter combinations for which the model can fit the data (possibly an infinite number).

The range of feasible parameter values can often be restricted, especially in the case of fermentation models, where parameters are related to physical concepts and properties of the process. Specific growth rates and yield coefficients must have positive values, for example. That being the case, the goal is to show that the models being used here to describe the fermentation are either locally identifiable within the range of feasible parameter values or globally identifiable, and hence suitable for our purposes.

5.2 Theoretical Identifiability

There are a number of techniques available in the engineering literature for assessing the identifiability properties of a model. Three of these are the Taylor series expansion approach (Pohjanpalo, 1978), the state isomorphism method (Vajda and Rabitz, 1989), and a differential algebra based method (Ljung and Glad, 1994).

5.2.1 The Taylor series approach

This method is based on solving the set of equations generated by taking successively higher derivatives of the basic model equations to give expressions for the parameters in terms purely of measurable quantities - typically the measured inputs and outputs of the process.

Assuming a typical structure for the nonlinear model:

$$\dot{x}(t) = f(x(t), u(t), t, \theta) \qquad x(t) \in \Re^n, t \in [0, T]$$
$$y(t) = g(x(t), \theta)$$

then taking derivatives and inserting assumed 'known' values for their initial conditions, the following set of equations is obtained:

$$g^{(k)}(x(0), \theta) = a_k(0) \qquad k = 0, \ldots, \infty$$

($g^{(k)}$ is the kth time derivative of g, and $a_k(0)$ is the (theoretically obtainable) initial value for the kth derivative.)

This method, based on the Taylor series expansion of the model equations may, in theory at least, be easily performed (Chappell *et al.*, 1990).

1. Differentiate $x(t, \theta)$ and $y(t, \theta)$.

2. Evaluate $y^{(i)}(0^+, \theta)$ by substitution of quantities already known from $y(0^+, \theta)$ and lower derivatives of x $(< i)$.

3. Check on the independence of the equations in successive derivatives and on what parameters, if any, can be identified at each stage in the

differentiation.

4. If not all parameters have been identified, then repeat the procedure from step 1.

Although the above method seems simple enough, in practice it rapidly becomes algebraically involved, especially where models are nonlinear in the parameters. The use of computational algebra packages, such as Maple, helps to a certain degree, but even so this method is limited in practice to simpler models with few state equations and few parameters.

5.2.2 The state isomorphism approach

The following is taken from the paper of Vajda and Rabitz (1989).

Consider a parametrised nonlinear system.

$$\Sigma_{\theta}^{x_0(\theta)} : \begin{cases} \dot{x}(t,\theta) = f(x(t,\theta),\theta) + uh(x(t,\theta),\theta) \\ y(t,\theta) = g(x(t,\theta),\theta),\ x(0,\theta) = x_0(\theta) \end{cases} \tag{5.1}$$

Let M and Ω be bounded, connected and open sets in $\Re^n$ and $\Re^q$, respectively, such that $x \in M$ and $\theta \in \Omega$, where θ represents the constant parameter vector. It is assumed that the vector fields $f(\cdot,\theta)$ and $h(\cdot,\theta)$, and the function $g(\cdot,\theta) : M \to \Re^m$ are real analytic on M for all $\theta \in \Omega$. Vajda and Rabitz (1989) considered the problem of identifiability of the above model system in the experiments with given initial condition $x_0(\theta)$ and bounded inputs $U[0,t_1]$, defined over the range $[0,t_1]$. Let $\Sigma_{\theta}^{x_0(\theta)}$ denote the input-output map of the system. Then parameter values $\theta, \tilde{\theta} \in \Omega$ are said to be indistinguishable (denoted by $\theta \sim \tilde{\theta}$) in the set of possible experiments

$(x_0(\theta), U[0,t_1])$, if $\Sigma_\theta^{x_0(\theta)}(u) = \Sigma_{\tilde{\theta}}^{x_0(\tilde{\theta})}(u)$ for all $u \in U[0,t_1]$. (That is to say, two experiments with different parameter sets are indistinguishable if the same inputs produce the same measured outputs for all permissible inputs.)

The system is *globally identifiable* at θ if $\tilde{\theta} \sim \theta, \tilde{\theta} \in \Omega$ implies $\tilde{\theta} = \theta$. The system is *locally identifiable* at θ if there exists an open neighbourhood W of θ in Ω such that $\tilde{\theta} = \theta, \tilde{\theta} \in W$ implies $\tilde{\theta} = \theta$.

Vajda and Rabitz (1989) referred to previous work on identifiability, in which three factors had been considered:

1. the relationship between the local identifiability and the local observability of a system

2. the functional expansion of the input-output map, eg. using a Taylor series expansion

3. the local state isomorphism approach of nonlinear realisation theory

Of these three, the first was discounted as being explicitly local, the second was considered to result in conditions for identifiability which were sufficient, but not necessary, from a practical point of view (for Vajda and Rabitz), and so Vajda and Rabitz (1989) concentrated on the third approach, aiming to extend the state isomorphism approach to the global identifiability of nonlinear systems.

In addition to assuming the analyticity of the system, it is assumed that the system satisfies both the controllability rank criterion (CRC) and the observability rank criterion (ORC). (The CRC and ORC are covered well in (Ray, 1989).) The problem of global identifiability may then be summarised

as follows: Given the model system (5.1) and $\theta \in \Omega$, find all $\tilde{\theta} \in \Omega$ and systems of the form

$$\Sigma_{\tilde{\theta}}^{\tilde{x}_0(\tilde{\theta})} : \begin{cases} \dot{\tilde{x}}_0(t,\tilde{\theta}) = f(\tilde{x}_0(t,\tilde{\theta}),\tilde{\theta}) + uh(\tilde{x}_0(t,\tilde{\theta}),\tilde{\theta}) \\ y(t,\tilde{\theta}) = g(\tilde{x}_0(t,\tilde{\theta}),\tilde{\theta}),\ \tilde{x}_0(0,\tilde{\theta}) = \tilde{x}_0(\tilde{\theta}) = x_0(\tilde{\theta}) \end{cases} \tag{5.2}$$

such that

$$\Sigma_{\theta}^{x_0(\theta)}(u) = \Sigma_{\tilde{\theta}}^{\tilde{x}_0(\tilde{\theta})} \tag{5.3}$$

for all $u \in U[0,t_1]$.

Vajda and Rabitz (1989) describe this as a highly restricted problem of system equivalence. First, both (5.1) and (5.2) are locally reduced and have the same subset M in $\Re^n$ as their state spaces. Second, in addition to the input-output map, the known system structure is also invariant under the feasible class of local state isomorphisms. The analysis is based on the construction of all such transformations. This idea had previously been applied to linear systems, where equivalence transformations are linear; although local state isomorphisms between (5.1) and (5.2) generally are solutions of a set of partial differential equations, their construction is relatively simple for certain locally identifiable systems of practical interest. Vajda and Rabitz (1989) also showed that any local state isomorphism preserving the structure of a homogeneous system is linear, and suggested that, because of this, the local state isomorphism method is very simple for this class of systems, and that the known conditions for global identifiability of linear and bilinear systems are special cases of their results.

The condition for identifiability advanced by Vajda and Rabitz (1989) was as follows.

Consider $\theta, \tilde{\theta} \in \Omega$, an open neighbourhood V of $x_0(\theta)$ in $\Re^n$, and any analytic map $\lambda : V \to \Re^n$ defined on V such that

$$\lambda(x_0(\tilde{\theta})) = x_0(\theta) \tag{5.4}$$

$$\text{rank}\frac{\partial \lambda}{\partial \tilde{x}} = n \text{ for all } \tilde{x} \in V \tag{5.5}$$

$$f(\lambda(\tilde{x}), \theta) = \frac{\partial \lambda}{\partial \tilde{x}} f(\tilde{x}, \tilde{\theta}) \tag{5.6}$$

$$h(\lambda(\tilde{x}), \theta) = \frac{\partial \lambda}{\partial \tilde{x}} h(\tilde{x}, \tilde{\theta}) \tag{5.7}$$

$$g(\lambda(\tilde{x}), \theta) = g(\tilde{x}, \tilde{\theta}) \tag{5.8}$$

for all $\tilde{x} \in V$. Then there exists $t_1 > 0$ such that (5.1) is globally identifiable at θ in the experiments $(x_0(\theta), U[0, t_1])$ if and only if the above conditions imply $\tilde{\theta} = \theta$.

5.2.3 The differential algebraic approach

A recent paper (Ljung and Glad, 1994) advances a technique for assessing the identifiability of nonlinear models by using methods pioneered in the field of differential algebra, in particular an algorithm indicated by Ritt (Ritt, 1950; Kolchin, 1973). This algorithm enables the generation of a set of *characteristic sets* of *prime differential ideals* from a set of *differential polynomials*. In the paper (Ljung and Glad, 1994), a naive description of the method is provided, which gives an overview of how this differential algebra based technique works. A summary of that is given here, in preference to the far more

involved and abstruse more mathematically complete and correct version, also offered in Ljung and Glad (1994).

Start from a set of differential polynomials describing the model

$$g_i(u, x, y, \theta, p) = 0 \quad i = 1, 2, \ldots, r \tag{5.9}$$

where $u(t)$ and $y(t)$ are the measured input and output values, $x(t)$ are non-measurable state variables, θ is a vector of time-invariant parameters and p is a differentiation operator. (**Note** that the formal development of the method is based around polynomial descriptions. Other types of relationship would need to be replaced with a suitable polynomial approximation. Ljung and Glad (1994) give the illustration of replacing $x = \sin(y)$ with $\dot{x}^2 = \dot{y}^2(1-x^2)$.)

From this set of differential polynomials an infinite number of other polynomials may be formed by differentiating, adding, scaling and multiplying the original differential polynomials. If the infinite set of all these expressions is denoted by G, then any solution u, y satisfying the original set of differential polynomials will satisfy all equations in G. Ljung and Glad (1994) state that it is sufficient to select a finite subset of G that has the same solution set as the original set of differential polynomials, and ask whether there might exist a set of differential polynomials that would make it easier to establish identifiability.

An analogy may be drawn with linear algebra and linear spaces and bases. A finite basis is sufficient to describe infinitely many vectors in the space, and there are infinitely many bases, each defined by a finite set of vectors. Certain questions are more easily answered in one basis than another Three criteria are given to define a 'good' basis (Ljung and Glad, 1994).

- Expressions should not contain the variable x, since it is not known to us.
- Expressions should not contain higher powers of θ, since this would make it more difficult to assess the identifiability.
- It is OK if the expressions contain powers and derivatives of u and y, since these anyway are known to us.

It was suggested that the 'best' expression in G would take the following form:

$$\Psi_0(y, u, p) = 0 \tag{5.10}$$

i.e. a differential algebraic expression in terms only of u and y. The next best would be of the type:

$$\Phi_1(y, u, p) + \theta\Psi_1(y, u, p) = 0 \tag{5.11}$$

from which it was suggested that the value of θ could be uniquely determined only if the function

$$\Psi_1(y, u, p)$$

has full rank.

Determining whether or not equations of the form of Equation 5.11 may be found is determined by searching over G with a procedure reminiscent of the Gauss-elimination algorithm. Take an arbitrary element. If it contains

unwanted features, try to eliminate them by the allowed algebraic manipulations. In this way, a 'better' element is created in each step, and finally the existence or otherwise of equations of the type shown in Equation 5.11 is determined. Formally, this is Ritt's algorithm.

In the original paper, step-by-step details of the working of the version of Ritt's algorithm used are not given (Ljung and Glad, 1994), but a few simple illustrations of the inputs and outputs of the algorithm are shown. One of these is shown later, in Section 5.5.

5.3 A New Approach to Identifiability

This section outlines a new approach to assessing the identifiability of non-linear models, based broadly on the notion of finding a 'linear regression' for each parameter in terms of measurable states and inputs only, as advanced by Ljung and Glad (1994). The method outlined here is similar to that first advanced by Pohjanpalo (1978), with the exception that here no attempt is necessarily made to solve explicitly for each parameter for which an attempt is being made to assess identifiability. The approach here may also be considered as marginally more generic than that of Pohjanpalo (1978), as it does not focus exclusively on the information contained in the 'germ' of the dynamic system behaviour (its initial conditions and derivative values).

5.3.1 The test

A model system is considered to be globally identifiable if the following statements are true. Starting from the model equations, a set of expressions can be obtained in which all the parameters present in the original model are

found, and in which the only states, inputs and outputs present are measurable, each expression including at least one term made up only of measurable states, inputs and outputs. Assuming that the groups of parameters present in each expression may be equated to distinct, non-zero constants, the set of simultaneous equations which can be formed, relating the parameters to the constants may be solved uniquely to give each parameter as being linearly related to a function made up solely of constants.

One possible procedure to follow to determine identifiability according to the above is as follows.

1. multiply throughout each equation in turn, so as to eliminate divisor terms

2. substitute between expressions to eliminate unmeasurable quantities

3. collect together the terms in each expression, grouping them by unique groups of measurable states and outputs

4. assuming that the groups of parameters associated with each unique group of measurable states and outputs can be equated to distinct, non-zero constants, generate a set of simultaneous equations

5. attempt to solve the resulting simultaneous equations for the individual parameters (as solutions are generated for each parameter in turn, substitute an assumed constant value for the parameter)

If any parameters are present in the set of simultaneous equations only as powers other than unity, then there exists the possibility of more than one

feasible set of equations, and global identifiability may not be shown. However, the system may still have only a single feasible parameter set.

This approach to determining model identifiability has been based on the assumption that the measurable quantities may be approximated, as functions of time, by high order polynomials, differentiable at least one more time than there are terms in the longest expression.

This new approach to determining the identifiability has advantages over the existing approaches. The theory on which it is based is simpler than that underlying the approaches of Vajda *et al* (1989) or Ljung and Glad (1994), and the approach does not call for the repeated calculations which are a feature of the approach of Pohjanpalo (1978).

The approach of Pohjanpalo (1978) suggests that constant values be defined for all measurable initial conditions, and their progressive derivatives, whilst the new approach merely assumes that expressions made up solely of measurable states and their derivatives will evaluate to constants, which assumption may be justified, since such expressions are to be equated to groups made up solely of parameters, which are assumed to be time-invariant.

5.3.2 The problem

What are sufficient conditions for it to be possible to obtain independent expressions for parameters in terms solely of states and inputs/outputs?

Consider a general differential polynomial expression:

$$k_1\zeta_1 + k_2\zeta_2 + \cdots + k_n\zeta_n \quad = \quad 0 \tag{5.12}$$

where the k_i are expressions made up only of parameters (assumed constants), and the ζ_i are distinct groups of states and their derivatives (or inputs or outputs), i.e. no two terms $k_i\zeta_i, k_j\zeta_j$ have the same ζ_*, assumed to be multiply differentiable. (A differential polynomial of the form given above may almost always be formed from the equations given in a fermentation model. Multiplying through by divisor terms, such as $(K_m + S)$ from the Monod expression, will eventually give a 'simple' differential polynomial of the form above.)

Differentiating a differential polynomial containing distinct groups of states produces a new differential polynomial containing distinct groups of states and their derivatives.

$$k_1\dot{\zeta}_1 + k_2\dot{\zeta}_2 + \cdots + k_n\dot{\zeta}_n = 0 \tag{5.13}$$

When any differential polynomial in which the groups of states are distinct is multiplied by any group of states, the groups of states in the resulting differential polynomial remain distinct. Multiplying Equation 5.12 by $\dot{\zeta}_1$ and Equation 5.13 by ζ_1 gives the following pair of equations.

$$k_1\zeta_1\dot{\zeta}_1 + k_2\zeta_2\dot{\zeta}_1 + \cdots + k_n\zeta_n\dot{\zeta}_1 = 0 \tag{5.14}$$

$$k_1\zeta_1\dot{\zeta}_1 + k_2\zeta_1\dot{\zeta}_2 + \cdots + k_n\zeta_1\dot{\zeta}_n = 0 \tag{5.15}$$

Subtracting the second of these equations from the first results in a differential polynomial with one fewer parameter group than the original, in which

all the groups of states are distinct.

$$k_2(\dot{\zeta}_1\zeta_2 - \zeta_1\dot{\zeta}_2) + \cdots + k_n(\dot{\zeta}_1\zeta_n - \zeta_1\dot{\zeta}_n) = 0 \tag{5.16}$$

It is not possible for more than parameter group to be eliminated in one step following this procedure, since the groups of states are distinct. Given a differential polynomial with distinct groups of states:

$$at^m + bt^n = 0$$

Differentiating with respect to time gives:

$$mat^{m-1} + nbt^{n-1} = 0$$

Multiplication and subtraction give:

$$mbt^{n+(m-1)} - nbt^{m+(n-1)} = 0$$

Assuming that m, n and b are all non-zero, the only case for which the left-hand side of this expression equals zero is that for which m and n are equal, which cannot be true if the groups of states are distinct, as was assumed.

The procedure of differentiation, multiplication and subtraction may be repeated, eliminating one parameter group at a time until only a single parameter group remains, at which point an expression may be formed for that parameter solely in terms of states, inputs, outputs and their derivatives, provided that a term consisting only of groups of states is present in the

original expression.

For a differential polynomial of the type in Equation 5.12, there are four possible structures:

- **Case 1** all terms in the differential polynomial are made up of groups of parameters multiplied by groups of inputs/outputs/states
- **Case 2** one term in the differential polynomial is made up only of parameters
- **Case 3** one term in the differential polynomial is made up only of inputs/outputs/states
- **Case 4** there is one term in the differential polynomial made up only of parameters, and one made up only of inputs/outputs/states

The four cases are not all identifiable. A simple illustration of each case is given here.

Case 1

Consider the simple Case 1 system:

$$aX + bY = 0 \tag{5.17}$$

where (a, b) are parameters and (X, Y) are inputs/outputs/states.
Differentiating:

$$a\dot{X} + b\dot{Y} = 0 \tag{5.18}$$

Cross-multiplying by the inputs/outputs/states associated with a:

$$aX\dot{X} + bY\dot{X} = 0 \tag{5.19}$$

$$aX\dot{X} + bX\dot{Y} = 0 \tag{5.20}$$

Subtracting the second from the first:

$$b(Y\dot{X} - \dot{Y}X) = 0 \tag{5.21}$$

This last equation is only true if either $b = 0$ or $(Y\dot{X} - \dot{Y}X) = 0$, and so this simple Case 1 system is not identifiable.

Case 2

Consider the simple Case 2 system:

$$aX + bY + c = 0 \tag{5.22}$$

where (a, b, c) are parameters and (X, Y) are inputs/outputs/states.
Differentiating:

$$a\dot{X} + b\dot{Y} = 0 \tag{5.23}$$

which is equivalent to a Case 1 system and so, again, the system is not identifiable.

Case 3

Consider the simple Case 3 system:

$$aX + bY + Z = 0 \tag{5.24}$$

where (a, b) are parameters and (X, Y, Z) are inputs/outputs/states. Differentiating:

$$a\dot{X} + b\dot{Y} + \dot{Z} = 0 \tag{5.25}$$

Cross-multiplying by the inputs/outputs/states associated with a:

$$aX\dot{X} + bY\dot{X} + Z\dot{X} = 0 \tag{5.26}$$

$$a\dot{X}X + b\dot{Y}X + \dot{Z}X = 0 \tag{5.27}$$

Subtracting the second from the first, and solving for b:

$$b = \frac{\dot{Z}X - Z\dot{X}}{Y\dot{X} - \dot{Y}X} \tag{5.28}$$

and solving for a:

$$a = \left(\frac{(Z\dot{X} - \dot{Z}X)Y\dot{X}}{Y\dot{X} - \dot{Y}X} - Z\dot{X} \right) / X\dot{X} \tag{5.29}$$

So, with the exception of those cases for which any of the denominators in the expressions of a and b is zero, this Case 3 system is identifiable.

For Case 3 systems, if all ζ_i are unique and differentiably analytic, then each k_i may be found as an expression made up solely of the ζ_i and their derivatives.

Case 4

Consider the simple Case 4 system:

$$aX + b + Y = 0 \tag{5.30}$$

where (a, b) are parameters and (X, Y) are inputs/outputs/states. Differentiating the above produces a Case 3 system,

$$a\dot{X} + 0 + \dot{Y} = 0 \tag{5.31}$$

and so can be solved for a,

$$a = -\frac{\dot{Y}}{\dot{X}} \tag{5.32}$$

and, substituting for a, can be solved for b,

$$b = -Y + \frac{\dot{Y}X}{\dot{X}} \tag{5.33}$$

and so Case 4 systems are identifiable.

5.3.3 What if not all ζ_i are measurable?

Clearly the chances of identifying all k_i are less if all the data described by the model are not available. Can ζ_i be eliminated without eliminating k_i?

Consider a differential polynomial expression made up of n terms, as in Equation 5.12. For some ζ_i, $1 \leq i \leq n$,

$$-k_i\zeta_i = \sum_{j=1}^{i-1} k_j\zeta_j + \sum_{j=i+1}^{n} k_j\zeta_j \tag{5.34}$$

$$\zeta_i = -\frac{\left(\sum_{j=1}^{i-1} k_j\zeta_j + \sum_{j=i+1}^{n} k_j\zeta_j\right)}{k_i} \tag{5.35}$$

(**Note** that $k_n = 0$ for a Case 3 system.)

With a single differential polynomial, this gets us nowhere. However, with a set of differential polynomials, such as those which make up a typical fermentation model, it may be possible to substitute progressively for the unknown states, provided that the differential polynomials are linearly independent. In that case, given at least one more differential polynomial than there are unmeasurable states, it should be possible to obtain a differential polynomial which contains only parameters and ζ expressions made up only of measurable states, inputs and outputs.

At this point the original differential polynomial expressions have been converted, by substituting for unmeasurable quantities, into a set of equivalent expressions, now made up of groups of the original k_i and ζ_i,

$$k_{1G}\zeta_{1G} + k_{2G}\zeta_{2G} + \cdots + k_{nG}\zeta_{nG} = 0 \tag{5.36}$$

where the k_{iG} are the new 'groups' of parameters, and the ζ_{iG} are the new 'groups' of measurable quantities and their derivatives. Again, $k_{nG} = 0$ is necessary for an identifiable, Case 3, system.

Given a differential polynomial made up of parameters and measurable quantities only, expressions can be obtained for each group of parameters, k_{iG}, Thus:

$$k_{iG} = f_{iG}(\zeta_*, p) \tag{5.37}$$

for all the k_{iG} present in the differential polynomial used to obtain the parameter group expressions. p is the differential operator. Expressions may be obtained for each of the k_{iG} in terms only of groups of measurable states, inputs and derivatives (ζ_{iG}).

It is not necessary to evaluate the $f_{iG}(\zeta_*, p)$. These expressions are only needed to check for conditions for which the k_{iG} have undetermined values. Since all the k_{iG} expressions will be expressed as the quotient of one expression made up of measurable quantities and another expression of the same type, there will be conditions for which the divisors have the value zero, and for those cases it is not possible to solve for the k_{iG}.

5.3.4 Finding k_i from k_{iG}

This is not enough, yet, to ensure that unique expressions can be obtained for all the parameters present in the original model equations (k_i). That can only be done if all the original parameters are present in the k_{iG} groups, and they are either present singly in the k_{iG} expressions, or present only

in conjunction with other parameters that may be obtained individually by substitutions from other k_{iG} groups.

After obtaining a differential polynomial with the form of Equation 5.36, it is necessary to be able to solve the set of polynomials given by setting the k_{iG} expressions present in the differential polynomial equal to constant values which would be obtained by evaluating the $f_{iG}(\zeta_*, p)$. These constant values are denoted by the Φ_i in the expression below. In other words it is necessary to solve the set of equations:

$$g_i(k_1, k_2, \cdots, k_{n-1}) \quad = \quad \Phi_i \tag{5.38}$$

where there exists one equation for each group of parameters in Equation 5.36, for the assorted k_i parameters from the original model equations. (The above expression is simply a rewritten form of Equation 5.37). The types of solutions obtained from these expressions will determine the number of sets of parameter values for which the model may display identical behaviour. If all (k_i) are linearly related to the Φ_i, then there will be single unique solutions for each; higher powers will have more roots, but these may lie outside the range of valid parameter values. Most fermentation model parameters are defined to be positive, for example.

If it is not possible to obtain distinct linear expressions for all (k_i) independent of the other model parameters, then there will exist a set of possible parameter values, possibly infinite, although bounded by physical feasibility. For example, if $k_1 + k_8 = \Phi_{10}$, then there are an infinite number of solutions for k_1 and k_8, but the constraint that $k_1 > 0, k_8 > 0$ means that both k_1 and k_8 must lie in the range $0 \leq \text{value} \leq \Phi_{10}$. In this case, although the system

does not exhibit *global* identifiability, it does exhibit *bounded* identifiability.

5.4 Pohjanpalo's Compartmental Model

Pohjanpalo (1978) gave the example of a compartmental model which was unidentifiable when all rates were considered to be linear, but which became identifiable when one of the compartments was considered to possess only a limiting number of binding sites for entering molecules, giving rise to Langmuir saturation. This example is used here to check that the new identifiability test gives results consistent with the approaches used before, showing unidentifiability for the linear model and identifiability for the non-linear model.

5.4.1 The linear model

The linear model was as follows.

$$\dot{x}_1(t) = -(\lambda_{10} + \lambda_{12})x_1(t) \tag{5.39}$$

$$\dot{x}_2(t) = \lambda_{12}x_1(t) - \lambda_{20}x_2(t) \tag{5.40}$$

$$y(t) = x_1(t) \tag{5.41}$$

In this case, only one of the two compartments ($x_1(t)$ and $x_2(t)$) is considered to be measurable, and an attempt is being made to demonstrate identifiability for $\{\lambda_{10}, \lambda_{12}, \lambda_{20}\}$.

However, there is no immediately evident way of eliminating $x_2(t)$ from the set of expressions, and so only the summed pair of parameters present in the $\dot{x}_1(t)$ expression, $\lambda_{10} + \lambda_{12}$, may be identified.

5.4.2 The nonlinear model

Adapting the linear equations to describe Langmuir saturation in the second compartment, the model equations become:

$$\dot{x}_1(t) = -\lambda_{10}x_1(t) - \lambda_{12}[1 - s_2x_2(t)]x_1(t) \tag{5.42}$$

$$\dot{x}_2(t) = \lambda_{12}[1 - s_2x_2(t)]x_1(t) - \lambda_{20}x_2(t) \tag{5.43}$$

$$y(t) = x_1(t) \tag{5.44}$$

In the case of the nonlinear model, there is one additional parameter to identify, s_2, the saturation factor for compartment 2.

Solving Equation 5.42 for $x_2(t)$, the following is obtained:

$$x_2 = \frac{\dot{x}_1 + (\lambda_{10} + \lambda_{12})x_1}{\lambda_{12}s_2x_1} \tag{5.45}$$

The explicit time dependence of the states x_1 and x_2 has been omitted from here on to attempt to make the equations simpler and clearer.

Differentiating this expression gives:

$$\dot{x}_2 = \frac{\lambda_{12}s_2x_1\ddot{x}_1 - \lambda_{12}s_2\dot{x}_1^2}{(\lambda_{12}s_2x_1)^2} \tag{5.46}$$

Substituting for x_2 and $\dot{x}_2$ in Equation 5.43, and simplifying, gives:

$$x_1\ddot{x}_1 - \dot{x}_1^2 = \lambda_{12}^2s_2x_1^3 - \lambda_{12}s_2x_1^2\dot{x}_1 - \lambda_{12}\lambda_\Sigma s_2x_1^3 - \lambda_{20}x_1\dot{x}_1 - \lambda_{20}\lambda_\Sigma x_1^2 \tag{5.47}$$

where $\lambda_\Sigma = (\lambda_{10} + \lambda_{12})$. Since the measurement vector y is identical to state vector x_1, expressions in x_1 and its derivatives are equivalent to expressions

in the measured variable y and its derivatives, and so no substitution has been made for x_1.

Collecting together terms in the same powers of x_1 and its derivatives, the following table is generated.

$$x_1^3 \qquad -\lambda_{12} s_2(\lambda_{12} + \lambda_\Sigma) \tag{5.48}$$

$$x_1^2 \qquad \lambda_{20}\lambda_\Sigma \tag{5.49}$$

$$x_1^2\dot{x}_1 \qquad \lambda_{12} s_2 \tag{5.50}$$

$$x_1\dot{x}_1 \qquad \lambda_{20} \tag{5.51}$$

$$x_1\ddot{x}_1 \qquad 1 \tag{5.52}$$

$$\dot{x}_1^2 \qquad -1 \tag{5.53}$$

The last two rows in this table show that the nonlinear model is a Case 3 system, and so is identifiable, provided that expressions for each of the parameters in the original model can be formed from the parameter groups in the above table.

Since successive groups of terms can be eliminated, expressions can be obtained for each group of associated parameters in terms only of the measurable state (x_1) and its derivatives. Attempting then to obtain expressions for the individual parameters, it would be possible to proceed as follows, working with the right-hand expressions from the rows mentioned in the following:

1. from expression 5.51 an expression may be obtained for λ_{20}

2. dividing expression 5.49 by expression 5.51 it is possible to solve for λ_Σ

3. dividing expression 5.48 by expression 5.50 it is possible to obtain an expression for $(\lambda_{12} + \lambda_{\Sigma})$ from which it is possible to solve for λ_{12}

4. since there now exist expressions for λ_{Σ} and λ_{12}, it is possible to solve for λ_{10}, as $\lambda_{\Sigma} = (\lambda_{10} + \lambda_{12})$

5. given λ_{12} , it is possible to solve for s_2 from expression 5.50

And so solutions have been found for all parameters in the nonlinear model,
$\{\lambda_{10}, \lambda_{12}, \lambda_{20}, s_2\}$. The new method for assessing model identifiability has been shown to give the same results for this pair of related examples as did the earlier method of Pohjapalo (1978).

5.5 An Example from Ljung and Glad (1994)

The following example, taken from an earlier paper on identifiability (Chappell *et al.*, 1990), was used in Ljung and Glad (1994) to illustrate the differential algebraic approach(using Ritt's algorithm). Chappell *et al.* (1980) applied the Taylor series expansion approach of Pohjanpalo (1978) and the similarity transform method of Vajda and Rabitz (1989) to the example of a biological system modelled as having biomass growth described by Monod kinetics and a first order death kinetic, and showed that both methods found

the system to be theoretically identifiable.

$$\dot{x}(t) = -\frac{V_m x(t)}{k_m + x(t)} - k_{01} x(t) \tag{5.54}$$

$$x(0) = D \tag{5.55}$$

$$y(t) = cx(t) \tag{5.56}$$

Here the goal is to identify the set of parameters $\{V_m, k_m, k_{01}, c\}$. D, the initial biomass concentration, is initially assumed to be unknown. Both the Ritt algorithm method and the new approach given above in Section 5.3 lead to the same conclusions regarding the identifiability or otherwise of the system as did the methods applied by Chappell *et al.*(1980).

5.5.1 The Ritt's algorithm results

The results from applying Ritt's algorithm to this problem are somewhat lengthy. Here $y^{(n)}$ denotes the n^{th} derivative of y, with $\dot{y}$ and $\ddot{y}$ being the first and second derivatives as normal.

$$\begin{aligned} -3y^2\ddot{y}^4 + 2y^2\dot{y}y^{(3)}\ddot{y}^2 - 2y^2\dot{y}^2y^{(4)}\ddot{y} + 3y^2\dot{y}^2y^{(3)^2} \\ +12y\dot{y}^2\ddot{y}^3 - 14y\dot{y}^3y^{(3)}\ddot{y} + 2y\dot{y}^4y^{(4)} - 6\dot{y}^4\ddot{y}^2 \\ +6\dot{y}^5y^{(3)} = 0 \end{aligned} \tag{5.57}$$

$$\dot{V}_m = 0 \tag{5.58}$$

$$\begin{aligned}(4V_m\dot{y}^8 - 8yV_m\ddot{y}\dot{y}^6 + 4\dot{y}^5y^2y^{(3)}V_m \\ -4\dot{y}^3y^3y^{(3)}V_m + \dot{y}^2y^{(3)}V_my^4 + 4\dot{y}^2\ddot{y}^3V_my^3 \\ -2y^{(3)}\ddot{y}^2V_my^4\dot{y} + \ddot{y}^4V_my^4)c \\ +4\dot{y}^9 - 12y\ddot{y}\dot{y}^7 + 12\ddot{y}^2y^2\dot{y}^5 - 4\ddot{y}^3y^3\dot{y}^3 = 0\end{aligned} \tag{5.59}$$

$$\begin{aligned}(y^2y^{(3)}\dot{y} - y^2\ddot{y}^2 - 2y\dot{y}^2\ddot{y} + 2\dot{y}^4)k_{01} \\ +y\dot{y}^2y^{(3)} - 3y\dot{y}\ddot{y}^2 + 2\dot{y}^3\ddot{y} = 0\end{aligned} \tag{5.60}$$

$$\begin{aligned}(4\dot{y}^9 - 12\ddot{y}y\dot{y}^7 + 12\dot{y}^5y^2\ddot{y}^2 - 4\dot{y}^3y^3\ddot{y}^3)k_m \\ +2\dot{y}^3y^4\ddot{y}y^{(3)}V_m - 2\dot{y}^5y^3y^{(3)}V_m + 2y^3\ddot{y}^2V_m\dot{y}^4 \\ -\dot{y}^2{y^{(3)}}^2V_my^5 - 2\dot{y}^2y^4\ddot{y}^3V_m + 2\ddot{y}^2y^5y^{(3)}V_m\dot{y} \\ -\ddot{y}^4y^5V_m = 0\end{aligned} \tag{5.61}$$

From the above equations, the following results were deduced (Ljung and Glad, 1994).

- The structure is neither locally nor globally identifiable, as a consequence of the $\dot{V}_m = 0$ expression. (This expression shows that V_m is a constant, but does not fix its value. Since V_m is present in Equations 5.59 and 5.61, the values of c and k_m depend on the value of V_m, and so the system is not identifiable.)
- k_{01} **is** globally identifiable, from Equation 5.60.
- If V_m were known, then c would be globally identifiable, according to Equation 5.59.
- If V_m were known, then k_m would be globally identifiable, according to Equation 5.61.

In other words, the system as it stands is not globally identifiable, but would be, if the value of any of $\{c, k_m, V_m\}$ were known.

5.5.2 Results following the new approach to identifiability

After substituting $y(t)/c$ throughout for $x(t)$, and rearranging the original equations, the following equation is obtained.

$$ck_m\dot{y} + y\dot{y} + cV_m y + ck_m k_{01} y + k_{01} y^2 = 0 \tag{5.62}$$

Collecting together groups of terms in the same powers of y and its derivatives, the following table can be constructed.

$$\dot{y}y \qquad +1 \tag{5.63}$$

$$y^2 \qquad k_{01} \tag{5.64}$$

$$\dot{y} \qquad c * k_m \tag{5.65}$$

$$y \qquad c * V_m + c * k_m * k_{01} \tag{5.66}$$

The first row in the above table shows that this is a Case 3 system, and so could be identifiable. From the table, it can immediately be seen that k_{01} is globally identifiable. It may also be seen that it should be possible to derive expressions for $c * k_m$, and, hence, for $c * V_m$. Given the value of any one of the three remaining unidentifiable parameters $\{c, k_m, V_m\}$, it should be possible to uniquely identify the other two.

If the initial x concentration, D had been known, then a value for c could immediately be obtained from the ratio of $y(0)$ to $x(0)$. Thus, for known ini-

tial conditions, the system is theoretically identifiable. This is the conclusion reached with all methods applied; the Ritt's algorithm approach (Ljung and Glad, 1994), the Taylor series and similarity transform approaches considered in Chappell *et al.* (1990) and the new approach given above in Section 5.3.

5.6 Identifiability Analysis of the Model of Paul et al. (1998)

The new method for assessing identifiability is here applied to the model of Paul *et al.* (1998). This method involves attempting to find expressions in terms only of states, outputs and their derivatives for the following 23 parameters.

$$\{\mu 0, mue, m0, m1, \gamma 1, mup, muh, \alpha 0, alphae, alphap,$$
$$K0, Ke, K1, K2, Kp, Ki, \mu 1, \mu 2, mua, Ksl, Ksi, mul, \mu 3\}$$

Maple has been used to perform algebraic manipulations on the model equations, and output taken from Maple was then fed into a program written in the programming language Perl, which collected together terms containing identical groups of measured quantities (inputs and outputs) and produced tabular output with the groups of measured quantities in one column and the associated groups of parameters in the other. The results of the Maple session are shown in the body of the following text, with the Perl output being used as the basis for the tables also given in the following. (Details of the Perl program are given in Appendix C.)

The Maple commands used to perform what would otherwise be tedious hand-cranked algebra are 'simplify', to rearrange the original equations, 'col-

lect', to group equations by parameter-containing expressions and 'sort', to make things easier to make sense of.

It should be noted that the following simple Maple examples do not use subscripts consistently. What is shown here is how Maple interprets the commands typed, without too much concern for the cosmetic appearance of the mathematics being performed. Commands typed at the Maple prompt will be shown as in the following example:

```
> restart;
```

The equations that define the model were set up one by one, checking each to see which parameters are identifiable, based on that particular equation. The expressions here are based on those given in Paul *et al.*.

5.6.1 The X0 expression

The illustration of the new method starts with the equation describing the rate of change of concentration of the growing tips, X0(t) ($X0expr$). The expressions for $rho1(t)$ and $v1c(t)$ are defined first, to ensure their availability later.

```
> rho1(t) := X1(t)/((X1(t)/rho)+X2(t)):

> v1c(t) := X1(t)/(2*rho1(t)) - X2(t):

> X0expr := - diff(X0(t),t) + mu0*S(t)*X1(t)/(K0+S(t))
- gamma1*X0(t)/(K1+S(t)) - Sigma1*X0(t)/V(t):

> X0expr := sort(collect(simplify(X0expr * (K0+S(t)) * (K1+S(t))
* V(t)),[mu0,gamma1,K0,K1],distributed),{X0(t),X1(t),S(t),V(t)});
```

$$
\begin{aligned}
X0expr := {} & \mu 0\,\mathrm{X1}(t)\,\mathrm{S}(t)^2\,\mathrm{V}(t) + \mu 0\,K1\,\mathrm{X1}(t)\,\mathrm{S}(t)\,\mathrm{V}(t) \\
& - \Sigma 1\,\mathrm{X0}(t)\,\mathrm{S}(t)^2 - \gamma 1\,\mathrm{X0}(t)\,\mathrm{S}(t)\,\mathrm{V}(t) - (\frac{\partial}{\partial t}\,\mathrm{X0}(t))\,\mathrm{S}(t)^2\,\mathrm{V}(t) \\
& - \gamma 1\,K0\,\mathrm{X0}(t)\,\mathrm{V}(t) + (-\Sigma 1\,\mathrm{X0}(t) - (\frac{\partial}{\partial t}\,\mathrm{X0}(t))\,\mathrm{V}(t))\,K0\;K1 \\
& + (-\Sigma 1\,\mathrm{X0}(t)\,\mathrm{S}(t) - (\frac{\partial}{\partial t}\,\mathrm{X0}(t))\,\mathrm{S}(t)\,\mathrm{V}(t))\,K0 \\
& + (-\Sigma 1\,\mathrm{X0}(t)\,\mathrm{S}(t) - (\frac{\partial}{\partial t}\,\mathrm{X0}(t))\,\mathrm{S}(t)\,\mathrm{V}(t))\,K1
\end{aligned}
$$

The term $\Sigma 1$ is the summed feeding and sampling terms, all of which affect the concentrations of the insoluble states.

At this point, the equation describing the rate of change of the concentration of the growing tips has been entered, multiplied throughout by the denominator terms from the Monod expressions in order to form a polynomial without quotient terms (which is easier to manipulate, and easier to interpret), collected together terms containing particular parameters, and finally sorted the terms in the equation according to the measurable states that they contain.

Using the Perl program from Appendix C to combine groups in like states and inputs, a table is produced relating the groups of states to their associated collections of parameters.

$$
\begin{aligned}
S(t)^2 * \Sigma 1 * X0(t) & \qquad -1 \\
S(t)^2 * V(t) * \left(\frac{\partial}{\partial t} X0(t)\right) & \qquad -1 \\
S(t)^2 * V(t) * X1(t) & \qquad +\mu 0 \\
-\left(\frac{\partial}{\partial t} X0(t)\right) * V(t) * S(t) - \Sigma 1 * X0(t) * S(t) & \qquad +K0 + K1
\end{aligned}
$$

$$
\begin{array}{rr}
-\left(\frac{\partial}{\partial t}X0(t)\right) * V(t) - \Sigma 1 * X0(t) & +K0 * K1 \\
S(t) * V(t) * X0(t) & -\gamma 1 \\
S(t) * V(t) * X1(t) & +K1 * \mu 0 \\
V(t) * X0(t) & -K0 * \gamma 1
\end{array}
$$

The first two rows of the above table show that the $X0expr$ expression defines a Case 3 system, and so by differentiating the original rate expression, a series of expressions can be obtained, containing successive derivatives of the states and inputs, with the original collections of parameters being associated with derivatives of the original groups of states.

By performing Gauss-elimination on the set of derivative expressions thus obtained (multiplying equations by groupings of states, inputs and derivatives), it is possible to eliminate successively the original groups, so as to obtain expressions in which the collections of parameters in the right-hand column above are expressed in terms only of states, inputs and derivatives.

To determine whether or not the individual parameters are uniquely identifiable, it is necessary to determine whether or not expressions can be found for each parameter, solely in terms of states, inputs and derivatives. For the above case, expressions can be obtained for each collection of parameters in the right-hand column. Since the model parameters are assumed to be constants, independent of the fermentation time, each of the expressions to which groups of parameters are equated must also be constant. These constants should not be evaluated, however, as they may contain high-order derivatives of measurable quantities, whose estimation is likely to be prone to

affected by the noise generated in calculating derivatives. If an attempt were to be made to estimate parameter values (say, for use as an initial guess to be used in parameter estimation) then calculating the values of these constants at a number of measurement times should reduce the error in the parameter estimates thus obtained.

$$\mu 0$$
$$\gamma 1$$
$$\mu 0 K1$$
$$\gamma 1 K0$$
$$K0 + K1$$
$$K0K1$$

Treating these as equations with each of the above equal to a constant, these become:

$$\mu 0 = c_1$$
$$\gamma 1 = c_2$$
$$\mu 0 K1 = c_3$$
$$\gamma 1 K0 = c_4$$
$$K0 + K1 = c_5$$
$$K0K1 = c_6$$

Expressions have immediately be obtained for $\mu 0$ and $\gamma 1$, and using these

expressions for $K0$ and $K1$ can be derived,

$$K1 = c_3/c_1$$
$$K0 = c_4/c_2$$

Hopefully, these will be consistent with the expressions obtained for $K0 + K1$ and $K0K1$.

From the X0 expression, it has been possible to obtain expressions for the four parameters $\{\mu 0, K0, \gamma 1, K1\}$ and so these parameters may now be reasonably considered to be 'known', and so it is not necessary to solve for them again. Thus there now remain 19 parameters for which expressions are sought.

$$\{\boldsymbol{mue}, \boldsymbol{m0}, \boldsymbol{m1}, \boldsymbol{mup}, \boldsymbol{muh}, \boldsymbol{\alpha 0}, \boldsymbol{alphae}, \boldsymbol{alphap},$$
$$\boldsymbol{Ke}, \boldsymbol{K2}, \boldsymbol{Kp}, \boldsymbol{Ki}, \boldsymbol{\mu 1}, \boldsymbol{\mu 2}, \boldsymbol{mua}, \boldsymbol{Ksl}, \boldsymbol{Ksi}, \boldsymbol{mul}, \boldsymbol{\mu 3}\}$$

5.6.2 The X1 expression

The sequence of operations is repeated for the expression describing the rate of change of the concentration of the subapical regions ($X1expr$), starting by multiplying through, collecting and sorting the terms from the original expression.

```
> X1expr := - diff(X1(t),t) + mue*S(t)*X0(t)/(Ke+S(t))
-mu0*S(t)*X1(t)/(K0+S(t)) + gamma1*X0(t)/(K1+S(t))
- mu2*rho*X2(t) - Sigma1*X1(t)/V(t):
```

```
> X1expr := sort(collect(simplify(X1expr * (Ke+S(t)) * (K0+S(t))
* (K1+S(t)) * V(t)),[mue,Ke,mu0,K0,gamma1,K1,mu2],distributed),
{X0(t),X1(t),X2(t),S (t),V(t)});
```

$$
\begin{aligned}
X1expr := & -\mu 0\,\mathrm{X1}(t)\,\mathrm{S}(t)^3\,\mathrm{V}(t) - \mu 2\,\rho\,\mathrm{X2}(t)\,\mathrm{S}(t)^3\,\mathrm{V}(t) \\
& + mue\,\mathrm{X0}(t)\,\mathrm{S}(t)^3\,\mathrm{V}(t) - \Sigma 1\,\mathrm{X1}(t)\,\mathrm{S}(t)^3 \\
& - \mu 0\,K1\,\mathrm{X1}(t)\,\mathrm{S}(t)^2\,\mathrm{V}(t) - \mu 0\,Ke\,\mathrm{X1}(t)\,\mathrm{S}(t)^2\,\mathrm{V}(t) \\
& - \mu 2\,\rho\,K1\,\mathrm{X2}(t)\,\mathrm{S}(t)^2\,\mathrm{V}(t) - \mu 2\,\rho\,K0\,\mathrm{X2}(t)\,\mathrm{S}(t)^2\,\mathrm{V}(t) \\
& - \mu 2\,\rho\,Ke\,\mathrm{X2}(t)\,\mathrm{S}(t)^2\,\mathrm{V}(t) + mue\,K1\,\mathrm{X0}(t)\,\mathrm{S}(t)^2\,\mathrm{V}(t) \\
& + mue\,K0\,\mathrm{X0}(t)\,\mathrm{S}(t)^2\,\mathrm{V}(t) + \gamma 1\,\mathrm{X0}(t)\,\mathrm{S}(t)^2\,\mathrm{V}(t) \\
& - (\frac{\partial}{\partial t}\,\mathrm{X1}(t))\,\mathrm{S}(t)^3\,\mathrm{V}(t) - \mu 0\,Ke\,K1\,\mathrm{X1}(t)\,\mathrm{S}(t)\,\mathrm{V}(t) \\
& - \mu 2\,\rho\,Ke\,K0\,\mathrm{X2}(t)\,\mathrm{S}(t)\,\mathrm{V}(t) - \mu 2\,\rho\,K0\,K1\,\mathrm{X2}(t)\,\mathrm{S}(t)\,\mathrm{V}(t) \\
& - \mu 2\,\rho\,Ke\,K1\,\mathrm{X2}(t)\,\mathrm{S}(t)\,\mathrm{V}(t) + \gamma 1\,Ke\,\mathrm{X0}(t)\,\mathrm{S}(t)\,\mathrm{V}(t) \\
& + \gamma 1\,K0\,\mathrm{X0}(t)\,\mathrm{S}(t)\,\mathrm{V}(t) + mue\,K0\,K1\,\mathrm{X0}(t)\,\mathrm{S}(t)\,\mathrm{V}(t) \\
& - \mu 2\,\rho\,Ke\,K0\,K1\,\mathrm{X2}(t)\,\mathrm{V}(t) + \gamma 1\,Ke\,K0\,\mathrm{X0}(t)\,\mathrm{V}(t) \\
& + \%2\,Ke + \%2\,K1 + \%2\,K0 + \%1\,Ke\,K0 + \%1\,Ke\,K1 \\
& + (-\Sigma 1\,\mathrm{X1}(t) - (\frac{\partial}{\partial t}\,\mathrm{X1}(t))\,\mathrm{V}(t))\,Ke\,K0\,K1 + \%1\,K0\,K1 \\
\%1 := & -\Sigma 1\,\mathrm{X1}(t)\,\mathrm{S}(t) - (\frac{\partial}{\partial t}\,\mathrm{X1}(t))\,\mathrm{S}(t)\,\mathrm{V}(t) \\
\%2 := & -\Sigma 1\,\mathrm{X1}(t)\,\mathrm{S}(t)^2 - (\frac{\partial}{\partial t}\,\mathrm{X1}(t))\,\mathrm{S}(t)^2\,\mathrm{V}(t)
\end{aligned}
$$

This is a more complicated result than that obtained by manipulations on the first expression, but the same sequence of operations can still be carried out. Starting by combining groups containing like groups of states, inputs and derivatives, a table can be made as before. Since the %1 and %2, returned by Maple, contain no parameters, they are treated as single units for the purposes of the following. This makes the task a little easier.

Running the Perl parsing program, the following are obtained.

$$
\begin{array}{rl}
S(t)^3 * \Sigma 1 * X1(t) & -1 \\
S(t)^3 * V(t) * \left(\frac{\partial}{\partial t} X1(t)\right) & -1 \\
S(t)^3 * V(t) * X0(t) & +\mu e \\
S(t)^3 * V(t) * X1(t) & -\mu 0 \\
S(t)^3 * V(t) * X2(t) & -\mu 2 * \rho \\
-\left(\frac{\partial}{\partial t} X1(t)\right) * V(t) * S(t)^2 & +K0 + K1 + Ke \\
-\Sigma 1 * S(t)^2 * X1(t) & \\
-\left(\frac{\partial}{\partial t} X1(t)\right) * V(t) - \Sigma 1 * X1(t) & +K0 * K1 * Ke \\
S(t) * V(t) * X1(t) & -K1 * Ke * \mu 0 \\
V(t) * X0(t) & +K0 * Ke * \gamma 1 \\
S(t)^2 * V(t) * X1(t) & -K1 * \mu 0 - Ke * \mu 0 \\
V(t) * X2(t) & -K0 * K1 * Ke * \mu 2 * \rho \\
-\left(\frac{\partial}{\partial t} X1(t)\right) * V(t) * S(t) & +K0 * K1 + K0 * Ke + K1 * Ke \\
-\Sigma 1 * S(t) * X1(t) & \\
S(t)^2 * V(t) * X0(t) & +K0 * \mu e + K1 * \mu e + \gamma 1 \\
S(t) * V(t) * X0(t) & +K0 * K1 * \mu e + K0 * \gamma 1 + Ke * \gamma 1 \\
S(t)^2 * V(t) * X2(t) & -\mu 2 * \rho * (K0 + K1 + Ke) \\
S(t) * V(t) * X2(t) & -\mu 2 * \rho * (K0 * K1 + K0 * Ke + K1 * Ke)
\end{array}
$$

Again, the first two rows of the above table shows that the $X1expr$ is a Case 3 system. Assuming that each of the right hand sides can be equated to some parameter-free expression obtained by repeatedly differentiating the original expression and performing Gauss-elimination on the resulting set of expressions, each element of the right-hand column can be set equal to a constant.

$$mue = d_1 \tag{5.67}$$

$$-\mu 0 = d_2 \tag{5.68}$$

$$-\mu 2\rho = d_3 \tag{5.69}$$

$$mue(K0 + K1) + \gamma 1 = d_4 \tag{5.70}$$

$$-\mu 0(Ke + K1) = d_5 \tag{5.71}$$

$$-\mu 2\rho(K0 + Ke + K1) = d_6 \tag{5.72}$$

$$\gamma 1(K0 + Ke) + mueK0K1 = d_7 \tag{5.73}$$

$$-\mu 0KeK1 = d_8 \tag{5.74}$$

$$-\mu 2\rho(K0K1 + K1Ke + KeK0) = d_9 \tag{5.75}$$

$$\gamma 1KeK0 = d_{10} \tag{5.76}$$

$$-\mu 2\rho KeK0K1 = d_{11} \tag{5.77}$$

$$(K0 + Ke + K1) = d_{12} \tag{5.78}$$

$$(K0K1 + K1Ke + KeK0) = d_{13} \tag{5.79}$$

$$-KeK0K1 = d_{14} \tag{5.80}$$

Assuming that ρ, the biomass density, has been measured by some other method, independent of the fermentation dynamics, then expressions for mue, $\mu 0$ and $\mu 2$ may immediately be obtained.

Thereafter, a little more work is needed. $K0$ may be obtained by dividing Equation 5.77 by Equation 5.74, giving $K0 = (\mu 0 \rho d_{11})/(\mu 2 d_8)$.

Obtaining an expression for Ke is only a little more complicated. Multiplying Equation 5.70 by $(K0 + Ke)$, and subtracting Equation 5.73 gives us the following expression.

$$\mu eK0^2 + \mu eK0Ke + \mu eK1Ke = d_4(K0 + Ke) - d_7$$

Substituting into this for $K1Ke$ from Equation 5.74 ($K1Ke = -d_8/\mu 0$), an expression can be obtained solely in terms of Ke and the already known $\mu 0$, mue and $K0$. Since this expression is linear in Ke, there can only be one solution for Ke.

From this point on, it is relatively simple to obtain expressions for the remaining parameters $K1$ and $\gamma 1$, using Equations 5.71 and 5.76.

If the expressions obtained for $\mu 0$, $\gamma 1$, $K0$, and $K1$ from working on the expression $X0expr$ had been reused, then it would only be necessary to have obtained expressions for μe, $\mu 2$ and Ke from the expression $X1expr$. If it were not possible to assume prior knowledge of the value of ρ, then it would only have been possible to obtain an expression for $\mu 2\rho$, but this would not have been important as $\mu 2$ and ρ only occur in the $X1expr$ expression combined as $\mu 2\rho$.

From the X1 expression, further expressions for the three parameters $\{mue, \mu 2, Ke\}$ have been obtained, and so these parameters may now rea-

sonably be considered to be 'known', and so there is no need to solve for them again. Thus there are now 16 parameters remaining for which expressions are sought.

$$\{\boldsymbol{m0}, \boldsymbol{m1}, \boldsymbol{mup}, \boldsymbol{muh}, \boldsymbol{\alpha 0}, \boldsymbol{alphae}, \boldsymbol{alphap},$$
$$\boldsymbol{K2}, \boldsymbol{Kp}, \boldsymbol{Ki}, \boldsymbol{\mu 1}, \boldsymbol{mua}, \boldsymbol{Ksl}, \boldsymbol{Ksi}, \boldsymbol{mul}, \boldsymbol{\mu 3}\}$$

5.6.3 The X2 expression

The next expression illustrates a difficulty which may arise in attempting to use this technique to show identifiability of a model structure.

```
> X2expr := - diff(X2(t),t) + mu1*X1(t) - mu2*X2(t) + mu3*X2(t)
- Sigma1*X2(t)/V(t):

> X2expr := sort(collect(simplify(X2expr * V(t)),[mu1,mu2,mu3],
distributed),{X0(t),X1(t),X2(t),S(t),V(t)});
```

$$X2expr := \mu 1\,\mathrm{X1}(t)\,\mathrm{V}(t) - \mu 2\,\mathrm{X2}(t)\,\mathrm{V}(t) + \mu 3\,\mathrm{X2}(t)\,\mathrm{V}(t) - \Sigma 1\,\mathrm{X2}(t) - (\frac{\partial}{\partial t}\,\mathrm{X2}(t))\,\mathrm{V}(t)$$

Collecting together terms with like groups of states and inputs, the following table can be produced:

$$\begin{array}{rr} \Sigma 1 * X2(t) & -1 \\ V(t) * \left(\dfrac{\partial}{\partial t} X2(t)\right) & -1 \\ V(t) * X1(t) & +\mu 1 \\ V(t) * X2(t) & +\mu 3 - \mu 2 \end{array}$$

It may be immediately seen that it is possible to obtain an expression for $\mu 1$. However, using only this one equation, it would not be possible to obtain independent expressions for $\mu 2$ and $\mu 3$. However, an expression for $\mu 2$ was obtained by analysing the X1 expression, and so an expression can now be obtained for $\mu 3$.

Thus there now remain 14 parameters for which expressions are sought.

$$\{\boldsymbol{m0}, \boldsymbol{m1}, \boldsymbol{mup}, \boldsymbol{muh}, \boldsymbol{\alpha 0}, \boldsymbol{alphae}, \boldsymbol{alphap},$$
$$\boldsymbol{K2}, \boldsymbol{Kp}, \boldsymbol{Ki}, \boldsymbol{mua}, \boldsymbol{Ksl}, \boldsymbol{Ksi}, \boldsymbol{mul}\}$$

5.6.4 The X3 expression

Starting in the same way as for the previous expressions, with the following:

```
> X3expr := - diff(X3(t),t) + mu3*rho*X2(t) - mua*X3(t)
- Sigma1*X3(t)/V(t):

> X3expr := sort(collect(simplify(X3expr * V(t)),[mu3,mua],
distributed),{X0(t),X1(t),X2(t),X3(t),S(t),V(t)});
```

$$X3expr := \mu 3\, \rho\, \mathrm{X2}(t)\, \mathrm{V}(t) - mua\, \mathrm{V}(t)\, \mathrm{X3}(t)$$
$$- (\frac{\partial}{\partial t}\, \mathrm{X3}(t))\, \mathrm{V}(t) - \Sigma 1\, \mathrm{X3}(t)$$

Collecting together terms in like groups of states and inputs, and producing the following table:

$$\Sigma 1 * X3(t) \qquad -1$$
$$V(t) * \left(\frac{\partial}{\partial t} X3(t)\right) \qquad -1$$

$$V(t) * X3(t) \qquad -\mu a$$
$$V(t) * X2(t) \qquad +\mu 3 * \rho$$

From which it can be seen that it should be possible to obtain independent expressions for mua and $\mu 3$, assuming that ρ is known. Using the expression thus obtained for $\mu 3$, an expression may then be obtained for $\mu 2$.

As an example, the steps involved in obtaining expressions for mua and $\mu 3$ are shown. First the original expression ($X3expr$) is differentiated, obtaining two equations in the two 'unknowns'.

$$-mua[VX3] + \mu 3[VX2] - [\Sigma 1X3 - \left(\frac{\partial}{\partial t}X3\right)V] = 0 \tag{5.81}$$

$$-mua\dot{\overline{[VX3]}} + \mu 3\dot{\overline{[VX2]}} - \dot{\overline{[\Sigma 1X3 - \left(\frac{\partial}{\partial t}X3\right)V]}} = 0 \tag{5.82}$$

The (t) have been dropped, and dot notation has been used to denote differentiation, for compactness. The $\dot{\overline{\quad}}$ denotes differentiation of the term(s) under the line.

Multiplying Equation 5.81 by $\dot{\overline{[VX2]}}$ and Equation 5.82 by $[VX2]$, the following are obtained:

$$-mua[VX3]\dot{\overline{[VX2]}} + \mu 3[VX2]\dot{\overline{[VX2]}} - [\Sigma 1X3 - \left(\frac{\partial}{\partial t}X3\right)V]\dot{\overline{[VX2]}} = 0 \tag{5.83}$$

$$-mua\dot{\overline{[VX3]}}[VX2] + \mu 3\dot{\overline{[VX2]}}[VX2] - \dot{\overline{[\Sigma 1X3 - \left(\frac{\partial}{\partial t}X3\right)V]}}[VX2] = 0 \tag{5.84}$$

Subtracting Equation 5.84 from Equation 5.83, an expression is obtained containing only terms with mua as a parameter.

$$-mua[VX3]\dot{\overline{[VX2]}} + mua\dot{\overline{[VX3]}}[VX2] - [\Sigma 1X3 + \left(\frac{\partial}{\partial t}X3\right)V]\dot{\overline{[VX2]}} + \dot{\overline{[\Sigma 1X3 + \left(\frac{\partial}{\partial t}X3\right)V]}}[VX2] = 0 \tag{5.85}$$

Rearranging the above expression, the following expression is obtained for mua:

$$mua = \frac{+[\Sigma 1X3 + \left(\frac{\partial}{\partial t}X3\right)V]\dot{\overline{[VX2]}} - \dot{\overline{[\Sigma 1X3 + \left(\frac{\partial}{\partial t}X3\right)V]}}[VX2]}{\dot{\overline{[VX3]}}[VX2] - [VX3]\dot{\overline{[VX2]}}} \tag{5.86}$$

To obtain an expression for $\mu 3$, substitute for mua in Equation 5.81.

$$-\frac{+[\Sigma 1X3 + \left(\frac{\partial}{\partial t}X3\right)V]\dot{\overline{[VX2]}} - \dot{\overline{[\Sigma 1X3 + \left(\frac{\partial}{\partial t}X3\right)V]}}[VX2]}{\dot{\overline{[VX3]}}[VX2] - [VX3]\dot{\overline{[VX2]}}}[VX3] + \mu 3[VX2] - [\Sigma 1X3 - \left(\frac{\partial}{\partial t}X3\right)V] = 0 \tag{5.87}$$

and hence:

$$\mu 3 = \frac{[\Sigma 1X3 - \left(\frac{\partial}{\partial t}X3\right)V]\frac{+[\Sigma 1X3+\left(\frac{\partial}{\partial t}X3\right)V]\dot{\overline{[VX2]}} - \dot{\overline{[\Sigma 1X3+\left(\frac{\partial}{\partial t}X3\right)V]}}[VX2]}{\dot{\overline{[VX3]}}[VX2]-[VX3]\dot{\overline{[VX2]}}}[VX3]}{[VX2]} \tag{5.88}$$

From the X3 expression, the only parameter for which an expression can be obtained, for which an expression had not already been obtained, is mua. Thus there now remain 13 parameters for which expressions are sought.

$$\{\boldsymbol{m0}, \boldsymbol{m1}, \boldsymbol{mup}, \boldsymbol{muh}, \boldsymbol{\alpha 0}, \boldsymbol{alphae}, \boldsymbol{alphap},$$
$$\boldsymbol{K2}, \boldsymbol{Kp}, \boldsymbol{Ki}, \boldsymbol{Ksl}, \boldsymbol{Ksi}, \boldsymbol{mul}\}$$

5.6.5 The X4 expression

The following $X4expr$ contains only a single parameter, but cannot be used in determining the identifiability of the model, as the state $X4(t)$, the degenerated biomass, is not measurable.

```
> X4expr := - diff(X4(t),t) + mua*X3(t) - Sigma1*X4(t)/V(t):
> X4expr := simplify(X4expr*V(t));
```

$$X4expr := -(\frac{\partial}{\partial t}\,\mathrm{X4}(t))\,\mathrm{V}(t) + mua\,\mathrm{V}(t)\,\mathrm{X3}(t) - \Sigma 1\,\mathrm{X4}(t)$$

5.6.6 The S expression

When evaluated, the glucose expression ($Sexpr$) runs to several pages (listed in Appendix C). Perhaps some conclusions may be drawn about its likely identifiability by looking at the initial differential expression.

```
> Sexpr := - diff(S(t),t) -alpha0*mu0*S(t)*X1(t)/(K0+S(t))
- alphae*mue*S(t)*X0(t)/(Ke+S(t)) - m0*X0(t)*S(t)/(K1+S(t))
- m1*rho*v1c(t)*S(t)/(K2+S(t))
- alphap*mup*rho*v1c(t)*S(t)/(Kp+S(t)*(1+S(t)/Ki))
+ mul*L(t)*(X0(t)+X1(t))/((Ksl+L(t))*(1+(S(t)/Ksi)))
```

```
- Sigma2*S(t)/V(t) + (Ff*sf)/V(t):

>  Sexpr := sort(collect(simplify(Sexpr * (K0+S(t))* (Ke+S(t))
* (K1+S(t)) * (K2+S(t)) * (Kp+S(t)*(1+S(t)/Ki))
* ((Ksl+L(t))*(1+(S(t)/Ksi))) * V(t)),[alpha0,mu0,K0,alphae,mue,
Ke,m0,m1,K1,K2,alphap,mup,Kp,Ki,mul,Ksl,Ksi],distributed),
{X0(t),X1(t),S(t),L(t),V(t)}):
```

$$
\begin{aligned}
&-\frac{\partial S(t)}{\partial t} - \frac{\alpha_0\,\mu_0\,S(t)\,X1(t)}{K0+S(t)} - \frac{alphae\,mue\,S(t)\,X0(t)}{Ke+S(t)} \\
&-\frac{m_0\,X0(t)\,S(t)}{K1+S(t)} - \frac{m_1\,\rho\,v1c(t)\,S(t)}{K2+S(t)} - \frac{alphap\,mup\,\rho\,v1c(t)\,S(t)}{Kp+S(t)(1+S(t)/Ki)} \\
&+\frac{mul\,L(t)\,(X0(t)+X1(t))}{(Ksl+L(t))\,(1+(S(t)/Ksi))} - \frac{\Sigma_2\,S(t)}{V(t)} + \frac{Ff\,sf}{V(t)} = 0
\end{aligned}
\tag{5.89}
$$

Examining the above equation with regard for how the model's parameters are distributed among the terms reveals that there are three pairs of parameters which do not occur singly, and which, therefore, cannot occur singly when the expression is multiplied throughout by the quotient expressions, so as to obtain a differential polynomial without quotients. These are $(\alpha_0\,\mu_0)$, $(alphae\,mue)$ and $(alphap\,mup)$. Unless expressions can be obtained for one parameter from each of these three pairings, it will not be possible to uniquely identify a parameter set, and it will only be possible to derive an expression for the product of two parameter terms. That being the case, there will be an infinite number of possible solutions for each pair of parameters.

5.6.7 The L expression

The lactose expression ($Lexpr$), however, has been evaluated.

```
> Lexpr := - diff(L(t),t) - mul*L(t)*(X0(t)+X1(t))/((Ksl+L(t))
*(1+(S(t)/Ksi)))- Sigma2*L(t)/V(t):

> Lexpr := sort(collect(simplify(Lexpr
* ((Ksl+L(t))*(1+(S(t)/Ksi))) * Ksi * V(t)),[mul,Ksl,Ksi],
distributed),{X0(t),X1(t),S(t),L(t),V(t)});
```

$$
\begin{aligned}
Lexpr := & -(\frac{\partial}{\partial t}\,\mathrm{L}(t))\,\mathrm{S}(t)\,\mathrm{V}(t)\,\mathrm{L}(t) - \Sigma 2\,\mathrm{S}(t)\,\mathrm{L}(t)^2 \\
& + (-(\frac{\partial}{\partial t}\,\mathrm{L}(t))\,\mathrm{S}(t)\,\mathrm{V}(t) - \Sigma 2\,\mathrm{S}(t)\,\mathrm{L}(t))\,Ksl \\
& + (-(\frac{\partial}{\partial t}\,\mathrm{L}(t))\,\mathrm{V}(t)\,\mathrm{L}(t) - \Sigma 2\,\mathrm{L}(t)^2)\,Ksi \\
& + (-(\frac{\partial}{\partial t}\,\mathrm{L}(t))\,\mathrm{V}(t) - \Sigma 2\,\mathrm{L}(t))\,Ksi\,Ksl \\
& + (-\mathrm{X1}(t)\,\mathrm{V}(t)\,\mathrm{L}(t) - \mathrm{X0}(t)\,\mathrm{V}(t)\,\mathrm{L}(t))\,Ksi\,mul
\end{aligned}
$$

Collecting together groups of like states, inputs and derivatives, the following table can be constructed:

$$
\begin{aligned}
L(t) * S(t) * V(t) * \left(\frac{\partial}{\partial t} L(t)\right) & \quad -1 \\
L(t)^2 * S(t) * \Sigma 2 & \quad -1 \\
-\left(\frac{\partial}{\partial t} L(t)\right) * V(t) * L(t) - \Sigma 2 * L(t)^2 & \quad +Ksi \\
-\left(\frac{\partial}{\partial t} L(t)\right) * V(t) * S(t) - \Sigma 2 * S(t) * L(t) & \quad +Ksl \\
-V(t) * X0(t) * L(t) - V(t) * X1(t) * L(t) & \quad +Ksi * \mu l \\
-\left(\frac{\partial}{\partial t} L(t)\right) * V(t) - \Sigma 2 * L(t) & \quad +Ksi * Ksl
\end{aligned}
$$

All parameters are present independent of one another and so expressions may be obtained from $Lexpr$ for each of Ksl, Ksi, and mul. It may well be

simpler, however, to obtain an expression for $1/Ksi$, rather than for Ksi.

Thus there now remain 10 parameters for which expressions are sought.

$$\{\boldsymbol{m0}, \boldsymbol{m1}, \boldsymbol{mup}, \boldsymbol{muh}, \boldsymbol{\alpha 0}, \boldsymbol{alphae}, \boldsymbol{alphap}, \boldsymbol{K2}, \boldsymbol{Kp}, \boldsymbol{Ki}\}$$

5.6.8 The P expression

Finally, the identifiability of parameters in the penicillin expresion ($Pexpr$) is investigated.

```
> Pexpr := - diff(P(t),t) + mup*rho*v1c(t)*S(t)/(Kp+S(t)
*(1+S(t)/Ki)) - muh*P(t)- Sigma2*P(t)/V(t):

> Pexpr := sort(collect(simplify(Pexpr * (Kp+S(t)
*(1+S(t)/Ki)) * Ki * V(t)),[mup,Kp,Ki,muh], distributed),
{X0(t),X1(t),S(t),P(t),V(t)});
```

$$\begin{aligned} Pexpr := & -muh\,\mathrm{S}(t)^2\,\mathrm{V}(t)\,\mathrm{P}(t) - (\frac{\partial}{\partial t}\mathrm{P}(t))\,\mathrm{S}(t)^2\,\mathrm{V}(t) \\ & - \Sigma 2\,\mathrm{S}(t)^2\,\mathrm{P}(t) - muh\,Ki\,\mathrm{S}(t)\,\mathrm{V}(t)\,\mathrm{P}(t) \\ & - muh\,Kp\,Ki\,\mathrm{V}(t)\,\mathrm{P}(t) \\ & + (-(\frac{\partial}{\partial t}\mathrm{P}(t))\,\mathrm{S}(t)\,\mathrm{V}(t) - \Sigma 2\,\mathrm{S}(t)\,\mathrm{P}(t))\,Ki \\ & + (\frac{1}{2}\,\mathrm{X1}(t)\,\mathrm{S}(t)\,\mathrm{V}(t) - \frac{1}{2}\,\mathrm{X2}(t)\,\rho\,\mathrm{S}(t)\,\mathrm{V}(t))\,Ki\,mup \\ & + (-(\frac{\partial}{\partial t}\mathrm{P}(t))\,\mathrm{V}(t) - \Sigma 2\,\mathrm{P}(t))\,Kp\,Ki \end{aligned}$$

Collecting together terms containing like groups of states, inputs and derivatives, the following is obtained:

$$P(t) * S(t)^2 * \Sigma 2 \qquad -1$$

$$
\begin{array}{rr}
S(t)^2 * V(t) * \left(\frac{\partial}{\partial t} P(t)\right) & -1 \\
-\Sigma 2 * P(t) * S(t) - \left(\frac{\partial}{\partial t} P(t)\right) * V(t) * S(t) & +Ki \\
P(t) * S(t)^2 * V(t) & -\mu h \\
-\Sigma 2 * P(t) - \left(\frac{\partial}{\partial t} P(t)\right) * V(t) & +Ki * Kp \\
1/2 * V(t) * S(t) * X1(t) - 1/2 * X2(t) * \rho * V(t) * S(t) & +Ki * \mu p \\
P(t) * S(t) * V(t) & -Ki * \mu h \\
P(t) * V(t) & -Ki * Kp * \mu h
\end{array}
$$

Again, all parameters are associated independently with different groups of states, inputs and derivatives, and so expressions may be obtained for each in terms only of measurable quantities. (Assuming that ρ is known.) That is to say, expressions can be obtained for the parameters $\{muh, mup, Kp, Ki\}$ from the P expression.

Thus there now remain 6 parameters for which expressions are sought.

$$\boldsymbol{\{m0, m1, \alpha 0, alphae, alphap, K2\}}$$

5.6.9 Identifiability result for the model of Paul et al. (1998)

Having examined all bar one of the expressions in the model of Paul *et al.* (1998) to see for which parameters expressions can be obtained solely in terms of states, outputs and their derivatives, six parameters remain, all of which are found in the S expression.

$$\{\boldsymbol{m0}, \boldsymbol{m1}, \boldsymbol{\alpha 0}, \boldsymbol{alphae}, \boldsymbol{alphap}, \boldsymbol{K2}\}$$

By examining the output of a Perl program written to decompose the expressions generated from Maple into unique groups of states, outputs and derivatives, and their associated groups of parameters, it is possible to determine whether or not these remaining six parameters are identifiable (see Appendix C).

Upon analysing the S expression, the parameter $m1$ is found to be the lone parameter associated with one group of states: the parameter $\alpha 0$ is found associated only with the known parameter $\mu 0$; $K2$ is found to be the only 'unknown' parameter in a number of expressions, as is *alphap*; thus two parameters remain, $m0$ and *alphae*. These two may be solved by eliminating between a pair of linearly independent expressions in which they are the only unknown parameters. Thus it has been shown that it is possible to form expressions for each of the model's parameters in terms solely of measurable states and inputs, and their derivatives, and so the model of Paul *et al.* (1998) is theoretically globally identifiable.

5.7 Notation

D	initial biomass concentration in example from Ljung and Glad (1994)
K_m	Monod denominator term, g(S)l^{-1}
$K0$	Monod denominator coefficient, g(S)l^{-1}
$K1$	denominator coefficient, g(S)l^{-1}
$K2$	denominator coefficient, g(S)l^{-1}
Ke	differentiation denominator coefficient, g(S)l^{-1}
Ki	inhibited penicillin production coefficient, g(S)l^{-1}

Kp	inhibited penicillin production coefficient, g(S)l^{-1}
Ksi	inhibited lactose conversion coefficient, g(S)l^{-1}
Ksl	inhibited lactose conversion coefficient, g(S)l^{-1}
L	concentration of lactose $*$, g(L)l^{-1}
M	system's state space
P	concentration of penicillin $*$, g(P)l^{-1}
S	concentration of glucose $*$, g(S)l^{-1}
$U[0, t_1]$	set of possible experimental inputs
V	neighbourhood around a parameter set
V	volume in the fermenter, l
V_m	Monod numerator coefficient in example from Ljung and Glad (1994)
W	neighbourhood around a parameter set
X_*	concentration of biomass fraction $*$, g(DW)l^{-1}
X, Y, Z	groups of states, inputs and outputs for the novel identifiability method
a, b, c	nominal constant parameters for the novel identifiability method
$a_k(0)$	value of k-the derivative at time 0
c	constant relating output to state in example from Ljung and Glad (1994)
c_*	nominal constant
$f()$	function defining rate of change of model states
$f_{*G}()$	expression relating some group of group of constant parameters to system states, input and outputs, and the differentiation operator in the novel identifiability method
$g()$	function defining model outputs
$g()$	function defining system output for Vajda *et al.* (1989)
$g_*(u, x, y, \theta, p)$	differential polynomial as defined in the method of Ljung and Glad (1994)
$g^{(k)}$	k-th derivative of output function $g()$
$h()$	function defining system input influence on rate of of change of model states after Vajda *et al.* (1989)
k_*	group of constant parameters in the novel identifiability method
k_{*G}	group of groups of constant parameters in the novel

	identifiability method
k_m	Monod denominator coefficient in example from Ljung and Glad (1994)
k_{01}	death rate coefficient in example from Ljung and Glad (1994)
m, n	nominal powers of time
$m0$	maintenance coefficient, g(S)g(DW)$^{-1}$h^{-1}
$m1$	maintenance coefficient, g(S)g(DW)$^{-1}$h^{-1}
p	differential operator
$\tilde{p}$	parameter vector
s_2	constant in Pohjanpalo's compartmental model
t	time, h
$u(t)$	inputs to model
$x(t)$	model state vector
$x_1(t)$	first model state in Pohjanalo's compartmental model
$x_2(t)$	second model state in Pohjanalo's compartmental model
$y(t)$	outputs from model
$y^{(i)}(0^+, \theta)$	i-the derivative of the output vector at time marginally greater than zero (limiting case)
$\Re^n$	n-dimensional space of real-valued numbers

Greek Symbols

$\Sigma_p^{x_0(p)}$	system as defined by Vajda *et al.* (1989)
Σ_1	summed feeds and abstractions, lh^{-1}
Φ_*	nominal constant in novel identifiability method
$\Phi_*(y, u, p)$	differential algebraic expression in the method of Ljung and Glad (1994)
$\Psi_*(y, u, p)$	differential algebraic expression in the method of Ljung and Glad (1994)
Ω	system's parameter space
$\alpha 0$	inverse yield coefficient for biomass on glucose, g(S)g(DW)$^{-1}$
$alphae$	inverse yield coefficient for biomass on glucose, g(S)g(DW)$^{-1}$
$alphap$	inverse yield coefficient for penicillin on glucose, g(S)g(P)$^{-1}$
$gamma1$	differentiation numerator coefficient, g(S)h^{-1}
ζ_*	group of time-varying states, input and output in the novel identifiability method
ζ_{*G}	group of groups of time-varying states, input and output

	in the novel identifiability method
θ	vector of model parameters
λ	analytic map in state isomorphism method
λ_{10}	constant in Pohjanpalo's compartmental model
λ_{12}	constant in Pohjanpalo's compartmental model
λ_{20}	constant in Pohjanpalo's compartmental model
λ_{Σ}	$\lambda_{10} + \lambda_{12}$
$\mu 0$	Monod numerator coefficient, h^{-1}
$\mu 1$	vacuole formation coefficient, $m^3 g^{-1} h^{-1}$
$\mu 2$	vacuolation rate coefficient,h^{-1}
$\mu 3$	vacuole formation coefficient, h^{-1}
mua	autolysis rate coefficient, h^{-1}
mue	Monod numerator coefficient, h^{-1}
muh	hydrolysis rate coefficient, h^{-1}
ρ	biomass density, gm^{-3}

6. CONCLUSIONS AND FURTHER WORK

The work described in this thesis has contributed to the aims of the University of Birmingham Biochemical Engineering Centre Rolling Grant Project B: 'Monitoring and Physiological Control of Productive Fermentations', and towards the goal of providing a control implementation, designed on the basis of differential-equation based fermentation models which may then serve as a base case against which the performance of Artificial Neural Network and hybrid models may be compared.

6.1 This Thesis

The literature was examined for existing models of the penicillin fermentation, and these were built and tuned using data supplied by Paul (1996), and their abilities to predict fermentation data were compared (Chapter 2). The best performing of these models, that of Paul and Thomas (1996), was simplified to increase its simulation speed, removing, for the time being, a description of the vacuolation process which involved a number of states and made the model numerically 'stiff', and extended to include a description of the way in which lactose present in the inoculum is consumed in the fermentation, thus producing the model of Paul *et al.*(1998) (Chapter 3).

Attention then switched to the confidence with which the parameters

of the model are known, and to ways of designing experiments to improve confidence in the parameter values. In this work, genetic algorithms were applied to the problem of finding feed profiles for the fermentation which would give rise to data that would, when used in parameter estimation, decrease the size of the joint confidence volume (the region of parameter space across which parameters may vary with the error remaining below a certain value). It was shown that the addition of constraints derived from practical considerations reduced the extent to which the parameter confidence could be improved (Chapter 4).

In parallel with this work, the problem of the identifiability of model parameters was considered. Reviewing the available literature on identifiability criteria for nonlinear models suggested a novel approach to assessing the theoretical identifiability of models, closely related to and inspired by existing approaches (Chapter 5). This approach was compared with existing approaches, and found to give the same results for a number of specimen problems, before being applied to the problem of assessing the identifiability of the model of Paul *et al.*(1998), which, according to the new approach, turns out to be theoretically identifiable, provided that the density of the biomass is known, and all model states are measurable.

6.2 Future Work

Now that the best existing model describing the penicillin fermentation has been identified, this model's parameters have been shown to be theoretically identifiable, and experiments have been designed to improve the confidence with which the model's parameters are known, attention should focus on the

use of the model in developing control-related applications: the open-loop economic optimisation of the model, the construction of estimators and the design of controllers.

6.2.1 Open-loop economic optimisation

Searching for a feed profile that maximises the profitability of the fermentation, subject to practical constraints is a task to which genetic algorithms seem to be well suited. Recent work (Simutis and Lübbert, 1997) has suggested that:

> "it does not make much sense to use the very complicated classical optimisation procedures like Pontryagin's maximum principle for most optimisation tasks in practical bioengineering"

and showed that the chemotaxis algorithm, simulated annealing and evolutionary programming gave results that were comparable with those obtained using classical approaches in earlier work. Iterative dynamic programming has also been applied to the optimisation of fermentation feed profiles (Luus, 1992), with the results obtained over a range of fermentation durations suggesting that the function relating performance index (total mass of product at the end of the fermentation) to time passes through a number of maxima. It may well be the case that genetic algorithms also produce comparable results for open-loop optimisation, given a similar input feed profile parameterisation. Applying genetic algorithms to searching for an economically optimal input feed profile should involve only minor modifications to existing MATLAB routines.

6.2.2 Estimators

Extended Kalman filters have often been used in constructing estimators for bioprocess applications, particularly in estimating biomass concentrations between sample intervals. For the penicillin fermentation as described by the model of Paul *et al.* (1998), these would presumably be attempting to estimate the individual biomass fractions, X_0, X_1, X_2 and X_3 from available online measurements of offgas composition and, via HPLC, of the concentrations of the soluble states S, L, and P. (The model of Paul *et al.* (1998) would need to be augmented with the addition of a carbon dioxide production rate term before offgas composition measurements were useful.)

One problem with the construction of extended Kalman filters for nonlinear systems is that of determining the identifiability of the nonlinear systems in question (Ray, 1989). In Ray (1989), the observability is defined in the following way. Recalling the general noise-free model structure from Chapter 4:

$$\dot{x}(t) = f(x(t), \beta, u(t)) \tag{6.1}$$

$$y(t) = g(x(t), \beta, u(t)) \tag{6.2}$$

If these model equations are linearised about a nominal state trajectory $\bar{x}(t)$, which satisfies the model equations and has the initial conditions $\bar{x}(0) = \bar{x}_0$, defining:

$$\delta x(t) = x(t) - \bar{x}(t) \qquad \delta y(t) = y(t) - \bar{y}(t) \tag{6.3}$$

$$A(t) = \left.\frac{\partial f}{\partial x}\right|_{\bar{x}(t)} \qquad C(t) = \left.\frac{\partial g}{\partial x}\right|_{\bar{x}(t)} \tag{6.4}$$

then the noise-free linearised system becomes:

$$\delta\dot{x}(t) = A(t)\delta x(t) \qquad \delta x(0) = \delta x_0 \tag{6.5}$$

$$\delta y(t) = C(t)\delta x(t) \tag{6.6}$$

Defining the fundamental matrix solution (mapping from the initial states x_0 to the current states $x(t)$:

$$\dot{\Phi}(t, t_0) = A(t)\Phi(t, t_0) \qquad \Phi(t_0, t_0) = I \tag{6.7}$$

the criterion for the observability of our general nonlinear system is the matrix $M(0, t_f)$, given by:

$$M(0, t_f) = \int_0^{t_f} \Phi(t, 0)' C'(t) C(t) \Phi(t, 0) dt \tag{6.8}$$

is positive definite for $t_f > 0$, i.e. that all the eigenvalues of $M(0, t_f)$ are positive.

This observability test depends on the model's parameter set, and, more critically, on the input to the system. Ray (1989) states that: "simple linearised observability tests are usually adequate for nonlinear problems", but perhaps an alternative approach could be taken.

Genetic algorithms could be applied to the problem of searching for inputs, constrained as for experiment design in Chapter 4, that gave rise to conditions where $M(0, t)$ had the minimum eigenvalues over as much of the fermentation as possible. The results of such a search would either find input scenarios for which the fermentation model is not observable, or increase

confidence in the model being observable. (Due to their probabilistic nature, genetic algorithms cannot show that the model is observable.) Whether or not the existence of feed profiles for which the model was not observable was important or not would depend on how close the unobservable profiles were to any open-loop economic optimum feed profiles that were developed, as optimal feed profiles are the ones of greatest potential interest and benefit.

6.2.3 Controllers

Having designed open-loop optimal feed profiles, and constructed estimators, the task of controller development remains. Presumably such controllers would regulate the fermentation so as to follow a predefined optimal state trajectory.

Montague *et al.* (1986) presented the combination of estimators and controllers in following predefined state trajectories and showed that adaptive controllers performed better than proportional plus integral control. In their conclusions, they suggested that the generalised predictive control law is particularly applicable to fermentation systems, and indicated the need for an optimised biomass profile, based on applying optimisation methods to a process model.

The best performing differential equation based physiological model of the penicillin fermentation has been identified. It has been shown that this model has, theoretically, uniquely identifiable model parameters, and experiments have been designed to improve the confidence with which the model parameters have been estimated. Thus the best currently achievable position has been reached, from which optimisation, estimator construction and controller

design may be done on the basis of differential equation based physiological fermentation models. The provision of a base case against which the performance of artificial neural networks and hybrid models may be compared is now simply a matter of time.

6.3 Notation

A	$\partial f/\partial x$
C	$\partial g/\partial x$
I	identity matrix
$M(0, t_f)$	observability test matrix
$f()$	model state derivative function
$g()$	model output function
t	time, h
u	model inputs
x	model states
$\bar{x}$	model states on nominal state trajectory
y	model outputs

Greek Symbols

$\Phi(t, t_0)$	state transition matrix from time t_0 to time t $x(t) = \Phi(t, t_0)x(t_0)$
β	model parameter vector

APPENDIX

A. MODELS CONSIDERED IN THIS WORK

A.1 Unstructured Models

Symbols used in the models are defined in the above notation list.

A.1.1 Fishman and Biryukov (1974)

Fishman and Biryukov (1974) used an extended version of the model of Ramkrishna *et al.* (1967) in a theoretical study of optimal control of the penicillin fermentation. This is the first study of this type that we have been able to find. The extension to the original model was an additional term describing a postulated penicillin production mechanism, relating penicillin production to both the amount of biomass present and its mean age. This model does not contain a expression to describe the rate of consumption of substrate related to penicillin production.

$$\frac{\mathrm{d}X}{\mathrm{dt}} = \frac{\mu_X S X}{K_S + S} - K I X \tag{A.1}$$

$$\frac{\mathrm{d}S}{\mathrm{dt}} = -\frac{1}{Y_{XS}} \frac{\mu_X S X}{K_S + S} \tag{A.2}$$

$$\frac{\mathrm{d}I}{\mathrm{dt}} = \frac{a_T \mu_X S X}{K_S + S} + K a_{T_1} I X \tag{A.3}$$

$$\frac{\mathrm{d}P}{\mathrm{dt}} = X\left[a_0 + a_1\frac{\kappa_1}{X} + a_2\frac{(\kappa_1)^2}{X}\right] \tag{A.4}$$

$$\frac{\mathrm{d}\kappa_1}{\mathrm{dt}} = X \tag{A.5}$$

A.1.2 *Heijnen* et al. *(1979)*

Heijnen *et al.* (1979) took a different approach to the modelling of the system from that used in the other models considered thus far. Their model development method concentrated on consideration of reactions involving chemical species known to be involved in the fermentation. This resulted in the following set of equations, expressed in terms of concentration of species in moles per kg of broth.

$$\begin{aligned}
\frac{\mathrm{d}(V_H \cdot X_H)}{\mathrm{dt}} &= X_H \frac{\mathrm{d}V_H}{\mathrm{dt}} + V_H \frac{\mathrm{d}X_H}{\mathrm{dt}} \\
&= r_X && \text{(A.6)} \\
\frac{\mathrm{d}(V_H \cdot S_H)}{\mathrm{dt}} &= S_H \frac{\mathrm{d}V_H}{\mathrm{dt}} + V_H \frac{\mathrm{d}S_H}{\mathrm{dt}} \\
&= r_S + \phi_S && \text{(A.7)} \\
\frac{\mathrm{d}(V_H \cdot P_H)}{\mathrm{dt}} &= P_H \frac{\mathrm{d}V_H}{\mathrm{dt}} + V_H \frac{\mathrm{d}P_H}{\mathrm{dt}} \\
&= r_P && \text{(A.8)} \\
\frac{\mathrm{d}V_H}{\mathrm{dt}} &= \sum(feeds - evaporation) + 0.032 r_O + 0.044 r_C && \text{(A.9)} \\
r_S &= -q_{s,max} \frac{S_H X_H V_H}{(K_S + S_H)} \\
(r_P + r_{PO}) &= \begin{cases} 3.3 \times 10^{-4} \cdot X_H V_H & \mu \geq 0.01 \text{ h}^{-1} \\ 3.3 \times 10^{-4} \cdot X_H V_H \frac{\mu}{0.01} & \mu < 0.01 \text{ h}^{-1} \end{cases} \\
r_{PO} &= 0.002 P_H V_H \\
r_X &= \left[-r_S - m_{s_H} X_H V_H - \frac{(r_P + r_{PO})}{Y_{PS_H}} \right] \cdot Y_{XS_H} \\
r_O &= \left(\frac{6}{Y_{XS_H}} - 1.044 \right) r_X + 6 m_{s_H} X_H V_H
\end{aligned}$$

$$
\begin{aligned}
& + \left(\frac{6}{Y_{PS_H}} - 9.5 \right) (r_P + r_{PO}) \\
r_C \quad = \quad & \left(\frac{6}{Y_{XS_H}} - 1. \right) r_X + 6 m_{s_H} X_H V_H \\
& + \left(\frac{6}{Y_{PS_H}} - 8 \right) (r_P + r_{PO})
\end{aligned}
$$

Here, V_H represents the mass of broth present in the reactor, whilst X_H, S_H, and P_H are concentrations in moles kg^{-1}. ϕ_S is the rate of glucose feed to the reactor (moles h^{-1}). The numbers used in the equations are values taken from Heijnen's paper. These numbers are *not* dimensionless, but have the dimensions of the ratios which they represent, obtained from elemental balancing.

The form of the expression for the penicillin production rate shows an increase in penicillin production rate with increasing substrate concentration, rising to a maximum (Figure 2.1).

A.1.3 Bajpai and Reuß(1980/1981)

This model contains differential equations for the biomass, substrate, product and dissolved oxygen concentrations. The biomass growth rate is described by Contois type kinetics, giving a growth rate dependent on the concentration of biomass as well as substrate and oxygen concentrations. This could be important for high concentrations of biomass, where diffusional limitations in transport of substrate to the surface of the hyphae, could possibly limit the growth rate.

The equations used by Bajpai and Reuß to describe the system are as follows.

$$\frac{\mathrm{d}X}{\mathrm{dt}} = \frac{\mu_X S}{(K_X X + S)} \frac{O_2}{(K_{O_2} X + O_2)} X \tag{A.10}$$

$$\frac{\mathrm{d}S}{\mathrm{dt}} = -\frac{\mu_X S O_2 X}{Y_{XS}(K_X X + S)(K_{O_2} X + O_2)} - \frac{\mu_P S O_2^P X}{Y_{PS}(K_P + S(1 + (S/K_I))(K_{OP} X + O_2^P)} - m_s X \tag{A.11}$$

$$\frac{\mathrm{d}P}{\mathrm{dt}} = \frac{\mu_P S O_2^P X}{(K_P + S(1 + (S/K_I))(K_{OP} X + O_2^P)} - K_h P \tag{A.12}$$

$$\frac{\mathrm{d}O_2}{\mathrm{dt}} = -\frac{\mu_X S O_2 X}{Y_{XO}(K_X X + S)(K_{O_2} X + O_2)} - \frac{\mu_P S O_2^P X}{Y_{PO}(K_P + S(1 + (S/K_I))(K_{OP} X + O_2^P)} - m_O X + k_L a(O_2^* - O_2) \tag{A.13}$$

A.1.4 *Montague* et al. *(1986)*

Montague *et al.* (1986) took the model proposed by Bajpai and Reuß(1980), and used it as a basis for the development of a form of parameter adaptive control. They used the original model, removing the dissolved oxygen concentration term, and including a term for the generation of carbon dioxide by the system. Measurements of the carbon dioxide production rate were used as a part of an inferential scheme for estimating the biomass concentration. Their equations, unsurprisingly, strongly resemble those of Bajpai and Reuß.

$$\frac{\mathrm{d}X}{\mathrm{dt}} = \frac{\mu_X S}{(K_X X + S)} X \tag{A.14}$$

$$\frac{\mathrm{d}S}{\mathrm{dt}} = -\frac{\mu_X S X}{Y_{XS}(K_X X + S)} - \frac{\mu_P S X}{Y_{PS}(K_P + S(1 + (S/K_I))} - m_s X \tag{A.15}$$

$$\frac{\mathrm{d}P}{\mathrm{dt}} = \frac{\mu_P S X}{(K_P + S(1 + (S/K_I))} - K_h P \tag{A.16}$$

$$\frac{\mathrm{d}CO_2}{\mathrm{dt}} = \left[\frac{1}{k_4}\frac{\mathrm{d}X}{\mathrm{dt}} + m_c X + k_5 \frac{\mu_P S X}{(K_P + S(1 + (S/K_I))}\right] V \tag{A.17}$$

In the original paper by Montague *et al.*, it is suspected that there is a misprint, as the given expression for CO_2 generation in the paper is described as including a term proportional to the rate of penicillin synthesis. The term given in equation A.17 as $k_5 \frac{\mu_P S X}{(K_P + S(1+(S/K_I))}$ is given in the paper as k_5.

A.1.5 *Nicolai* et al. *(1991)*

This model was presented as an update to the models of Heijnen *et al.* (1979) and Bajpai and Reuß(1980), attempting to fuse the two and account for both endogenous and maintenance metabolism, with a smooth transition between the two.

The specific growth rate of the biomass in this model is calculated from the substrate uptake rate, after deductions for biomass maintenance and penicillin production substrate requirements. At low substrate concentrations, the resulting low substrate uptake rate leads to a negative growth rate for the biomass, equivalent to the consumption of part of the biomass to meet maintenance and production requirements.

$$\frac{\mathrm{d}X}{\mathrm{dt}} = \mu X \tag{A.18}$$

$$\frac{\mathrm{d}S}{\mathrm{dt}} = -\sigma X \tag{A.19}$$

$$\frac{\mathrm{d}P}{\mathrm{dt}} = \pi X - K_h P \tag{A.20}$$

$$\pi = \mu_p \frac{S}{K_P + S + S^2/K_I}$$

$$\mu = \mu_{substr} - Y_{XS}\left(exp(-S/E_M)m_s + \frac{exp(-S/E_P)\pi}{Y_{PS}}\right)$$

$$\mu_{substr} = \frac{\mu_X S}{K_X X + S}$$

$$\sigma = \frac{\mu_{substr}}{Y_{XS}} + m_s(1 - exp(-S/E_M)) + \frac{\pi(1 - exp(-S/E_P))}{Y_{PS}}$$

A.1.6 *Menezes* et al. *(1994)*

This is another model based on that of Bajpai and Reuß(1980). The model uses the same Contois kinetics to describe the biomass growth as did the earlier model, and has the same inhibited penicillin production kinetics. However, there is an additional term to describe the conversion of biomass from a live to a dead state. Both biomass states are modelled, total biomass being the sum of the live and dead fractions.

The equations used are as follows.

$$\frac{\mathrm{d}X}{\mathrm{dt}} = \frac{\mu_X S}{(K_X X + S)} X - K_d X \tag{A.21}$$

$$\frac{\mathrm{d}S}{\mathrm{dt}} = -\frac{\mu_X S X}{Y_{XS}(K_X X + S)} - \frac{\mu_P S X}{Y_{PS}(K_S + S)} - \frac{m_s S X}{K_{ml} + S} \tag{A.22}$$

$$\frac{\mathrm{d}P}{\mathrm{dt}} = \frac{\mu_P S X}{K_S + S} - K_h P \tag{A.23}$$

$$\frac{\mathrm{d}X_{dead}}{\mathrm{dt}} = K_d X \tag{A.24}$$

A.1.7 *Tiller* et al. *(1994)*

The model proposed by Tiller *et al.* (1994) is based on that of Bajpai and Reuß(1980), distinguishing between 'growing and producing' X_1 and 'non-growing and producing' X_2 cells. It also includes a term for cell lysis, by which the concentration of the non-growing biomass state is decreased.

The given equations are as follows.

$$\frac{\mathrm{d}X_1}{\mathrm{dt}} = \left(\frac{\mu_{S,max}S}{K_S+S} + \frac{\mu_{PM,max}PM}{K_{PM}+PM}\right)X_1 - k_{12}X_1 \tag{A.25}$$

$$\frac{\mathrm{d}X_2}{\mathrm{dt}} = k_{12}X_1 - k_{ly}X_2 \tag{A.26}$$

$$\frac{\mathrm{d}S}{\mathrm{dt}} = -\frac{\mu_{S,max}SX_1}{Y_{XS}(K_S+S)} - \frac{\pi(X_1+X_2)}{Y_{PS}} - m_s(X_1+X_2) \tag{A.27}$$

$$\frac{\mathrm{d}PM}{\mathrm{dt}} = -\frac{1}{Y_{PM}}\frac{\mu_{PM,max}PM}{K_{PM}+PM}X_1 + R_1k_{ly}X_2 \tag{A.28}$$

$$\frac{\mathrm{d}P}{\mathrm{dt}} = \pi(X_1+X_2) - K_hP \tag{A.29}$$

In the above set of equations, the coefficients k_{ly}, k_{12} and m are dependent on the mean age of the hypha, as follows.

$$\begin{aligned} k_{ly} &= a_{ly} + b_{ly}\kappa_2 \\ k_{12} &= f_{12}\kappa_2 \\ m_s &= b_m + a_m\kappa_2 \\ \kappa_2 &= \frac{1}{X(t)}\int_0^t X(\tau)\mathrm{d}\tau \end{aligned}$$

As Tiller *et al.* did not observe glucose inhibition, they described the

product formation rate π as a function of the specific growth rate μ. The shape of the relationship used is plotted in Figure 2.2.

A.1.8 *Kluge* et al. *(1992)*

Kluge *et al.* (1992) outline a model for penicillin production which is unstructured, and has a complex expression for penicillin production. It considers nutrient uptake for multiple substrates (glucose, lactose and lysed biomass). Nutrient uptake is modelled using Michaelis-Menten kinetics, with maintenance and penicillin production substrate requirements being subtracted from the uptake rate, and the result used to calculate the biomass growth rate.

The biomass is divided into active and inactive portions. Biomass deactivation is modelled as being linearly proportional to the concentration of active biomass. The lysis rate is modelled similarly, being linearly proportional to the concentration of inactive biomass.

The rate of change of the penicillin production rate is subject to a first order lag term, thus delaying the production of penicillin.

The given equations are as follows.

$$\frac{\mathrm{d}X_I}{\mathrm{dt}} = \mu_I X_A - k_{ly} X_I \tag{A.30}$$

$$\frac{\mathrm{d}X_A}{\mathrm{dt}} = q_T Y_{XS} X_A - \mu_I X_A \tag{A.31}$$

$$\frac{\mathrm{d}Y}{\mathrm{dt}} = R_2 k_{ly} X_I - q_Y X_A \tag{A.32}$$

$$\frac{\mathrm{d}S}{\mathrm{dt}} = -q_S X_A \tag{A.33}$$

$$\frac{\mathrm{d}L}{\mathrm{dt}} = -q_L X_A \tag{A.34}$$

$$\frac{\mathrm{d}P}{\mathrm{dt}} = \mu_P X_A - K_h P \tag{A.35}$$

$$\frac{\mathrm{d}\mu_P}{\mathrm{dt}} = \frac{\left(\frac{K_R}{K_R+S}\mu_{P0}\left(q_T Y_{XS} + K_A\right) - \mu_P\right)}{T_P} \tag{A.36}$$

where

$$\begin{aligned} q_T &= q_S + \alpha q_L + \beta q_Y - m_s - \frac{\mu_p}{Y_{PS}} \\ q_Y &= \frac{q_{Y0} Y}{K_Y + Y} \\ q_S &= \frac{q_{S0} S}{K_S + S} \\ q_L &= \frac{q_{L0} L}{(K_L + L)(1 + C_{L,s} q_S)} \end{aligned} \tag{A.37}$$

The T_P acts as the time constant in a first order lag. If allowed to come to steady state, the value of the specific production rate would be (neglecting lactose and lysed biomass terms):

$$\mu_{P_{SS}} = \frac{\mu_P S + A}{K_P + S(1 + S/K_I)}$$

where

$$\begin{aligned} \mu_P &= \frac{K_R Y_{PS} \mu_{PO} (Y_{XS}(q_{SO} - m_s) + K_A)}{K_R(Y_{PS} + Y_{XS}\mu_{PO}) + K_S Y_{PS}} \\ K_P &= \frac{K_R K_S (Y_{PS} + Y_{XS}\mu_{PO})}{K_R(Y_{PS} + Y_{XS}\mu_{PO}) + K_S Y_{PS}} \\ K_I &= \frac{K_R(Y_{PS} + Y_{XS}\mu_{PO}) + K_S Y_{PS}}{Y_{PS}} \\ A &= \frac{K_R Y_{PS} \mu_{PO} (K_A K_S - Y_{XS} m_s K_S)}{K_R(Y_{PS} + Y_{XS}\mu_{PO}) + K_S Y_{PS}} \end{aligned}$$

A.2 Morphologically Structured Models

A.2.1 Megee et al. (1970)

Megee *et al.* (1970) were the first to present a morphologically structured model, with four hyphal states being identified by association with different products (X_1, X_2, X_3, X_4). In addition, the growing hyphal tips were modelled as a distinct hyphal state X_0, as was a dormant state$Q$.

In using this model, originally applied to *Aspergillus awamori*, to describe the penicillin fermentation we have assumed that product states P_3 and P_4 represent penicillin. Product state P_1 is growth associated, and so is unsuited to representing penicillin.

The equations given in the paper are as follows.

$$\frac{\mathrm{d}X_0}{\mathrm{dt}} = \sum_{i=1}^{3} \frac{\nu_i X_i S}{\xi_i + S} - \frac{\zeta_0 X_0}{\lambda_0 + S} \tag{A.38}$$

$$\begin{aligned}\frac{\mathrm{d}X_1}{\mathrm{dt}} = {} & \frac{\mu_1 X_0 S}{K_1 + S} + \frac{\mu_2 X_1 S P_1}{(K_2 + S)(K_4 + P_1)} - \frac{\nu_1 X_1 S}{\xi_1 + S} \\ & - \frac{\zeta_1 X_1}{\lambda_1 + S} - \frac{\omega_1 S X_1}{\lambda_1 + S} + \frac{\zeta_0 X_0}{\lambda_0 + S}\end{aligned} \tag{A.39}$$

$$\begin{aligned}\frac{\mathrm{d}X_2}{\mathrm{dt}} = {} & \frac{\mu_3 X_2 S P_1}{(K_3 + S)(K_5 + P_1)} - \frac{\nu_2 X_2 S}{\xi_2 + S} + \frac{\omega_1 S X_1}{\lambda_1 + S} \\ & - \frac{\zeta_2 X_2}{\lambda_2 + S} - \frac{\omega_2 S X_2}{\lambda_2 + S}\end{aligned} \tag{A.40}$$

$$\frac{\mathrm{d}X_3}{\mathrm{dt}} = \frac{\nu_3 X_3 S}{\xi_3 + S} + \frac{\omega_2 S X_2}{\lambda_2 + S} - \frac{\zeta_3 X_3}{\lambda_3 + S} - \frac{\omega_3 S X_3}{\lambda_3 + S} \tag{A.41}$$

$$\frac{\mathrm{d}X_4}{\mathrm{dt}} = \frac{\omega_3 S X_3}{\lambda_3 + S} \tag{A.42}$$

$$\begin{aligned}\frac{\mathrm{d}S}{\mathrm{dt}} = & -\frac{1}{Y_{XS}}\frac{\mu_1 X_0 S}{K_1 + S} - \sum_{i=1}^{3}\frac{\varepsilon_{i+1}\eta_i X_i S}{\lambda i + S} \\ & - \sum_{i=1}^{2}\frac{\alpha_{i+1}\mu_{i+1} X_i S P_1}{(K_{i+1} + S)(K_{i+3} + P_1)}\end{aligned} \tag{A.43}$$

$$\frac{\mathrm{d}P_1}{\mathrm{dt}} = \frac{\gamma_1\mu_1 X_0 S}{K_1 + S} - \frac{\sigma_2 X_1 P_1}{K_6 + P_1} - \frac{\sigma_3 X_2 P_1}{K_7 + P_1} \tag{A.44}$$

$$\frac{\mathrm{d}P_3}{\mathrm{dt}} = \frac{\gamma_3\omega_2 X_2 S}{\lambda_2 + S} \tag{A.45}$$

$$\frac{\mathrm{d}P_4}{\mathrm{dt}} = \frac{\gamma_4\omega_3 X_3 S}{\lambda_3 + S} \tag{A.46}$$

$$\frac{\mathrm{d}Q}{\mathrm{dt}} = \sum_{i=1}^{3}\frac{\zeta_i X_i}{\lambda_i + S} \tag{A.47}$$

The terms in the differential equations for the biomass states may be thought of as describing the following processes.

- Branching – the formation of new hyphal tips.
- Differentiation – the 'aging' of the hyphae from one state to another.
- Dormancy – the process by which hyphal material enters the dormant state. (Here material may pass from any but the hyphal tip state directly to the dormant state.)
- Assimilation – this process is ill-defined.

Of these processes, all but assimilation are found in the other morphologically structured models.

Note that the dormant biomass state Q is regarded, for our purposes, as being lysed biomass, and is not included when calculating the total biomass concentration.

A.2.2 Nestaas and Wang (1983)

Nestaas and Wang (1983) proposed a model which built on the foundations of the model of Megee *et al.* (1970), applying Megee *et al.*'s proposed morphological structure to measurable quantities. In this model, three hyphal states were identified:

1. tips (X_0)
2. producing cells (X_1)
3. degenerated cells (X_2)

The substrate concentration during the fermentation is not modelled in the paper. Instead, as glucose is 'never allowed to accumulate in the broth', an expression is given for the rate of glucose consumption. This expression is independent of the glucose concentration in the medium.

Penicillin production is modelled as being formed via a precursor. 'The precursor conversion is expressed by 'masked' second-order kinetics $[pP^2/X]$ in order to minimize the role of this postulated component in the overall carbon balance' (Nestaas and Wang, 1983). Hydrolysis of penicillin in the medium is assumed to follow first order kinetics.

The model is divided into two sets of equations, one for use during an initial rapid growth phase, and one for use during a subsequent production phase.

The model equations are as follows.

For rapid growth.

$$\frac{\mathrm{d}X_0}{\mathrm{dt}} = \mu_0 X_0 \tag{A.48}$$

$$\frac{\mathrm{d}X_1}{\mathrm{dt}} = \mu_{1,max} X_0 \tag{A.49}$$

For the production phase.

$$\frac{\mathrm{d}X_0}{\mathrm{dt}} = 0 \tag{A.50}$$

$$\frac{\mathrm{d}X_1}{\mathrm{dt}} = \mu_1 X_0^* - k_2 X_1 \tag{A.51}$$

$$\frac{\mathrm{d}X_2}{\mathrm{dt}} = k_2 X_1 \tag{A.52}$$

$$\frac{\mathrm{d}P}{\mathrm{dt}} = k_{pen}\frac{(pP)^2}{X_0 + X_1 + X_2} - K_h P \tag{A.53}$$

$$\frac{\mathrm{d}pP}{\mathrm{dt}} = k_p X_1 - R_3 k_{pen}\frac{(pP)^2}{X_0 + X_1 + X_2} \tag{A.54}$$

The following equation describing the substrate concentration in the fermentation has been constructed using expressions given by Nestaas and Wang. This differs slightly from the formulation given by Nestaas and Wang in that the product formation related substrate consumption term has been corrected from that given in the original paper.

$$\frac{\mathrm{d}S}{\mathrm{dt}} = -\frac{\mu(X_0 + X_1 + X_2)}{Y_{XS}} - m_s(X_0 + X_1) - \frac{k_p X_1}{Y_{pPS}} \tag{A.55}$$

A.2.3 Cagney et al. *(1983)*

Cagney *et al.* (1983) used an updated form of the model described by Nestaas and Wang (1983), in conjunction with a filtration probe, to attempt to provide additional information regarding the different morphological states defined in the model. The Cagney model represents the whole course of the fermentation, and is not divided into 'growth' and 'production' phases.

Again the biomass is divided into three fractions:

1. tips (X_0)
2. producing cells (X_1)
3. degenerated cells (X_2)

The model equations are as follows.

$$\frac{\mathrm{d}X_0}{\mathrm{dt}} = \frac{\nu X_1 S}{K_S + S} - \frac{\zeta_0 X_0}{\lambda + S} \tag{A.56}$$

$$\frac{\mathrm{d}X_1}{\mathrm{dt}} = \frac{\mu_1 X_0 S}{K_S + S} - \frac{\nu X_1 S}{K_S + S} + \frac{\zeta_0 X_0}{\lambda + S} - \frac{\zeta_1 X_1}{\lambda + S} \tag{A.57}$$

$$\frac{\mathrm{d}X_2}{\mathrm{dt}} = \frac{\zeta_1 X_1}{\lambda + S} \tag{A.58}$$

$$\frac{\mathrm{d}S}{\mathrm{dt}} = -\frac{1}{Y_{XS}} \frac{\mu_1 X_0 S}{K_S + S} - \frac{1}{Y_{PS}} \frac{\mu_P X_1 S}{K_p + S(1 + S/K_I)} - \frac{m_s X_1 S}{K_{ml} + S} \tag{A.59}$$

$$\frac{\mathrm{d}P}{\mathrm{dt}} = \frac{\mu_P X_1 S}{K_p + S(1 + S/K_I)} - K_h P \tag{A.60}$$

(A.61)

A.2.4 Paul and Thomas (1996)

Paul and Thomas (1996) proposed a morphologically structured model similar in format to the original model of Megee *et al.* (1970).

The following morphological states are distinguished in the model.

1. growing tips (X_0)
2. non-growing regions (X_1)
3. vacuoles (notionally X_2–not shown here)
4. degenerated regions (X_3)
5. autolysed biomass (X_4)

The relative quantities of the first four portions of the biomass were assessed using image analysis techniques, thus facilitating the validation of the model. The autolysed biomass state is included to keep track of the amount of biomass lysed.

The equations given are as follows.

$$\frac{\mathrm{d}X_0}{\mathrm{d}t} = \frac{\nu_0 X_1 S}{\xi_0 + S} - \frac{\zeta_0 X_0}{\lambda_0 + S} \tag{A.62}$$

$$\frac{\mathrm{d}X_1}{\mathrm{d}t} = \frac{\mu_s X_0 S}{K_S + S} - \frac{\nu_0 X_1 S}{\xi_0 + S} + \frac{\zeta_0 X_0}{\lambda_0 + S} - \frac{\pi(r_k + r_l)^3}{6}\rho_3 k_s \bar{n}_k \tag{A.63}$$

$$\frac{\mathrm{d}X_3}{\mathrm{d}t} = \frac{\pi(r_k + r_l)^3}{6}\rho_3 k_s \bar{n}_k - \mu_a X_3 \tag{A.64}$$

$$\frac{\mathrm{d}X_4}{\mathrm{d}t} = \mu_a X_3 \tag{A.65}$$

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\frac{1}{Y_{XS_0}}\frac{\nu_0 X_1 S}{\xi_0 + S} - \frac{1}{Y_{XS_e}}\frac{\mu_s X_0 S}{K_S + S} - \frac{m_{s0} X_0 S}{\lambda_0 + S} - \frac{m_{s1}\rho_c v_{ic} S}{\lambda_1 + S}$$

$$-\frac{1}{Y_{PS}}\frac{\mu_p \rho_c v_{ic} S}{K_p + S(1 + S/K_I)} \tag{A.66}$$

$$\frac{\mathrm{d}P}{\mathrm{dt}} = \frac{\mu_p \rho_c v_{ic} S}{K_p + S(1 + S/K_I)} - K_h P \tag{A.67}$$

$$\frac{\mathrm{d}V}{\mathrm{dt}} = F \tag{A.68}$$

This model also incorporates a representation of the process by which vacuoles form and grow, giving rise to the inactive biomass state, X_3. Vacuoles were not considered to contribute to the overall biomass concentration, but are significant in estimating the volume (and therefore the mass) of non-growing regions X_1 from the total hyphal volume. For full details of this see Paul and Thomas (1996).

A.3 Model Simplification

The original model of Paul and Thomas (1996), given in appendix A.2.4 has been simplified in this work so as to reduce the time taken per simulation, and also to make the model easier to analyse.

Two types of simplification were considered;

- two step models, in which subapical regions are modelled as forming vacuoles, which subsequently give rise to degenerated regions of the biomass
- one step models, in which subapical regions are modelled as forming degenerated regions directly

A.3.1 The two step models

The two step models retain the division of biomass into distinct fractions that was used in the original model.

The equations are as follows.

$$\frac{\mathrm{d}X_0}{\mathrm{dt}} = \frac{\nu_0 X_1 S}{\zeta_0 + S} - \frac{\omega_0 X_0}{\lambda_0 + S} \tag{A.69}$$

$$\frac{\mathrm{d}X_1}{\mathrm{dt}} = \frac{\mu_s X_1 S}{K_s + S} - \frac{\nu_0 X_1 S}{\zeta_0 + S} + \frac{\omega_0 X_0}{\lambda_0 + S} - \mathrm{Kinetic2}X_2 \tag{A.70}$$

$$\frac{\mathrm{d}X_2}{\mathrm{dt}} = \mathrm{Kinetic1}X_1 - \mathrm{Kinetic2}X_2 \tag{A.71}$$

$$\frac{\mathrm{d}X_3}{\mathrm{dt}} = \mathrm{Kinetic2}X_2 - \mu_a X_3 \tag{A.72}$$

$$\frac{\mathrm{d}X_4}{\mathrm{dt}} = \mu_a X_3 \tag{A.73}$$

$$\frac{\mathrm{d}S}{\mathrm{dt}} = -\frac{1}{Y_{XS_0}}\frac{\nu_0 X_1 S}{\zeta_0 + S} - \frac{1}{Y_{XS_e}}\frac{\mu_s X_0 S}{K_s + S} - \frac{m_{s0} X_0 S}{\lambda_0 + S} - \frac{m_{s1} X_1 S}{\lambda_1 + S}$$

$$-\frac{1}{Y_{PS}}\frac{\mu_p X_1 S}{K_p + S(1 + S/K_I)} \tag{A.74}$$

$$\frac{\mathrm{d}P}{\mathrm{dt}} = \frac{\mu_p X_1 S}{K_p + S(1 + S/K_I)} - K_h P \tag{A.75}$$

$$\frac{\mathrm{d}V}{\mathrm{dt}} = F \tag{A.76}$$

In the above set of equations, Kinetics 1 and 2 are those describing vacuole formation and destruction, respectively. These are chosen from the three candidate kinetics, first order, k, conversion, $k/(L+S)$, and inhibition kinetic, $kS/(L + S + (S^2/M))$.

The symbol X_2 represents some numerical measure of the total amount of vacuoles present. (This state was *not* considered in tuning the simplified models.)

A.3.2 The one step models

The one step models retain the division of biomass into distinct fractions that was used in the original model.

The equations are as follows.

$$\frac{\mathrm{d}X_0}{\mathrm{dt}} = \frac{\nu_0 X_1 S}{\zeta_0 + S} - \frac{\omega_0 X_0}{\lambda_0 + S} \tag{A.77}$$

$$\frac{\mathrm{d}X_1}{\mathrm{dt}} = \frac{\mu_s X_1 S}{K_s + S} - \frac{\nu_0 X_1 S}{\zeta_0 + S} + \frac{\omega_0 X_0}{\lambda_0 + S} - \mathrm{Kinetic3}X_2 \tag{A.78}$$

$$\frac{\mathrm{d}X_3}{\mathrm{dt}} = \mathrm{Kinetic3}X_2 - \mu_a X_3 \tag{A.79}$$

$$\frac{\mathrm{d}X_4}{\mathrm{dt}} = \mu_a X_3 \tag{A.80}$$

$$\frac{\mathrm{d}S}{\mathrm{dt}} = -\frac{1}{Y_{XS_0}}\frac{\nu_0 X_1 S}{\zeta_0 + S} - \frac{1}{Y_{XS_e}}\frac{\mu_s X_0 S}{K_s + S} - \frac{m_{s0} X_0 S}{\lambda_0 + S} - \frac{m_{s1} X_1 S}{\lambda_1 + S} - \frac{1}{Y_{PS}}\frac{\mu_p X_1 S}{K_p + S(1 + S/K_I)} \tag{A.81}$$

$$\frac{\mathrm{d}P}{\mathrm{dt}} = \frac{\mu_p X_1 S}{K_p + S(1 + S/K_I)} - K_h P \tag{A.82}$$

$$\frac{\mathrm{d}V}{\mathrm{dt}} = F \tag{A.83}$$

In the above set of equations, Kinetics 3 describes the conversion of biomass from non-growing regions to degenerated regions. This is chosen from the three candidate kinetics, first order, k, conversion, $k/(L + S)$, and inhibition kinetic, $kS/(L + S + (S^2/M))$.

There is no model state in the one step model representing vacuoles.

A.4 Paul and Thomas (1998)

The original model of Paul and Thomas (1996), has been simplified and extended in the course of the work described in this thesis, so as to give a new form of the model, that published in Paul *et al.* (1998).

The same morphological states are distinguished in this model as in the original model of Paul and Thomas (1996).

1. growing tips (X_0)
2. non-growing regions (X_1)
3. vacuoles (notionally X_2–not shown here)
4. degenerated regions (X_3)
5. autolysed biomass (X_4)

In addition to the soluble species modelled in the original model, terms have been added to describe the consumption of lactose.

The equations are as follows.

$$\frac{\mathrm{d}X_0}{\mathrm{dt}} = \frac{\nu_0 X_1 S}{\xi_0 + S} - \frac{\zeta_0 X_0}{\lambda_0 + S} \tag{A.84}$$

$$\frac{\mathrm{d}X_1}{\mathrm{dt}} = \frac{\mu_s X_0 S}{K_S + S} - \frac{\nu_0 X_1 S}{\xi_0 + S} + \frac{\zeta_0 X_0}{\lambda_0 + S} - \mu_2 X_2 \rho \tag{A.85}$$

$$\frac{\mathrm{d}X_2}{\mathrm{dt}} = \mu_1 X_1 - \mu_2 X_2 + \mu_3 X_2 \tag{A.86}$$

$$\frac{\mathrm{d}X_3}{\mathrm{dt}} = \mu_2 X_2 \rho - \mu_a X_3 \tag{A.87}$$

$$\frac{\mathrm{d}X_4}{\mathrm{dt}} = \mu_a X_3 \tag{A.88}$$

$$\frac{\mathrm{d}S}{\mathrm{dt}} = -\frac{1}{Y_{XS_0}} \frac{\nu_0 X_1 S}{\xi_0 + S} - \frac{1}{Y_{XS_e}} \frac{\mu_s X_0 S}{K_S + S} - \frac{m_{s0} X_0 S}{\lambda_0 + S} - \frac{m_{s1} \rho_c v_{ic} S}{\lambda_1 + S}$$

$$-\frac{1}{Y_{PS}}\frac{\mu_p \rho_c v_{ic} S}{K_p + S(1+S/K_I)} + \frac{\mu_L L}{(K_{SL}+L)}\frac{(X_0+X_1)}{(1+S/K_{SI})} \quad \text{(A.89)}$$

$$\frac{\mathrm{d}L}{\mathrm{dt}} = -\frac{\mu_L L}{(K_{SL}+L)}\frac{(X_0+X_1)}{(1+S/K_{SI})} \quad \text{(A.90)}$$

$$\frac{\mathrm{d}P}{\mathrm{dt}} = \frac{\mu_p \rho_c v_{ic} S}{K_p + S(1+S/K_I)} - K_h P \quad \text{(A.91)}$$

$$\frac{\mathrm{d}V}{\mathrm{dt}} = F \quad \text{(A.92)}$$

A.5 Notation

A	Numerator term in steady state simplification of the penicillin production term in Kluge *et al.* (1992), g(P)g(S)g(DW)$^{-1}$h^{-1}l^{-1}
CO_2	Volume of CO_2,l
$C_{L,s}$	Constant allowing for lactose uptake repression in the presence of glucose uptake, g(DW)hg(S)$^{-1}$
E_M	Endogenous maintenance coefficient, g(S)l^{-1}
E_P	Endogenous production coefficient, g(S)l^{-1}
F	Feed rate to fermenter, lh^{-1}
I	Concentration of inhibitor, g(I)l^{-1}
K	Deactivation constant, lg(I)$^{-1}$h^{-1}
K_A	Biomass growth offset term, h^{-1}
K_I	Inhibition coefficient, g(S)l^{-1}
K_L	Monod coefficient for lactose, g(L)l^{-1}
K_{O_2}	Contois constant for O_2, g(O_2)l^{-1}
K_{OP}	Contois coefficient for penicillin production, g(S)g(DW)$^{-1}$
K_P	Inhibition coefficient, g(S)l^{-1}
K_{PM}	Monod coefficient for pharmamedia, g(PM)l^{-1}
K_R	Constant for catabolite repression by glucose, g(S)l^{-1}
K_S	Monod coefficient for glucose, g(S)l^{-1}
K_{SI}	inhibited lactose conversion coefficient, g(S)l^{-1}
K_{SL}	inhibited lactose conversion coefficient, g(S)l^{-1}
K_X	Contois constant, g(S)g(DW)$^{-1}$
K_Y	Monod coefficient for lysed material, g(Y)l^{-1}
K_d	Death coefficient for active biomass, h^{-1}
K_e	Monod coefficient for biomass growth, h^{-1}
K_h	Penicillin hydrolysis coefficient, h^{-1}
K_{ml}	Michaelis-Menten maintenance coefficient, g(S)l^{-1}
K_1, K_2, K_3	Monod type denominator (glucose) terms, g(S)l^{-1}
K_4, K_5, K_6, K_7	Monod type denominator (P_1) terms, g(P_1)l^{-1}
L	second order biomass conversion coefficient
L	Concentration of lactose, g(L)l^{-1}
M	inhibition coefficient for biomass conversion
O_2	Concentration of O_2, g(O_2)l^{-1}
O_2^*	Interfacial oxygen concentration O_2, g(O_2)l^{-1}
O_2^P	Concentration of O_2 affecting penicillin production, g(O_2)l^{-1}

P	Concentration of penicillin, g(P)l^{-1}
P_H	Concentration of penicillin in the model of Heijnen *et al.* (1979), moles(P)kg^{-1}
P_n	Concentration of product P_n, g(P_n)l^{-1}
PM	Concentration of pharmamedia, g(PM)l^{-1}
Q	Concentration of dormant biomass state, g(DW)l^{-1}
R_1	Effective Pharmamedia:lysed biomass ratio, g(PM)g(DW)$^{-1}$
R_2	Effective lysed biomass:viable biomass ratio, g(Y)g(DW)$^{-1}$
R_3	Precursor to penicillin conversion coefficient, g(pP)g(P)$^{-1}$ *Note that the three ratios R_* were not present in the original models. but are included here for dimensional consistency.*
S	Concentration of glucose, g(S)l^{-1}
S_H	Concentration of glucose in the model of Heijnen *et al.* (1979), moles(S)kg^{-1}
T_p	Time constant for rate of change of penicillin production rate, h
V	Volume of broth in fermenter, l
V_H	Mass of broth in fermenter in the model of Heinen *et al.* (1979), kg
X	Concentration of biomass, g(DW)l^{-1}
X_A	Concentration of active biomass, g(DW)l^{-1}
X_H	Concentration of biomass in the model of Heijnen *et al.* (1979), moles(X)kg^{-1}
X_I	Concentration of inactive biomass, g(DW)l^{-1}
X_{dead}	Concentration of dead biomass, g(DW)l^{-1}
X_*	Concentration of biomass fraction $*$, g(DW)l^{-1}
X_0^*	Concentration of biomass fraction 0 at the end of the growth phase in the model of Nestaas and Wang (1983), g(DW)l^{-1}
Y	Concentration of lysed material, g(Y)l^{-1}
Y_{PM}	Yield of biomass with respect to pharmamedia, g(DW)g(PM)$^{-1}$
Y_{PO}	Yield coefficient for penicillin with respect to oxygen, g(P)g(O_2)$^{-1}$
Y_{PS}	Yield coefficient for penicillin with respect to substrate, g(P)g(S)$^{-1}$
Y_{PS_H}	Yield coefficient for penicillin with respect to substrate in

	the model of Heijnen *et al.* (1979), moles(P)moles(S)$^{-1}$
Y_{XO}	Yield coefficient for biomass with respect to oxygen, g(DW)g(O_2)$^{-1}$
Y_{XS}	Yield of biomass with respect to glucose, g(DW)g(S)$^{-1}$
Y_{XS_H}	Yield of biomass with respect to glucose in the model of Heijnen *et al.* (1979), moles(DW)moles(S)$^{-1}$
Y_{XS_0}	Yield of biomass with respect to glucose for state 0, g(DW)g(S)$^{-1}$
Y_{XS_e}	Yield of biomass with respect to glucose for state e, g(DW)g(S)$^{-1}$
Y_{pPS}	Yield coefficient for penicillin precursor with respect to substrate, g(pP)g(S)$^{-1}$
Z	Concentration of a model state, gl^{-1}
$_{meas}$	Subscript denoting measured value
$_{sim}$	Subscript denoting simulated value
a_0	Coefficient of age function, g(P)g(DW)$^{-1}$h^{-1}
a_1	Coefficient of age function, g(P)(g(DW)$^{-1}$h^{-2}
a_2	Coefficient of age function, g(P)lg(DW)$^{-2}$h^{-3}
a	Area for gas→liquid mass transfer, m^2
a_{ly}	Polynomial coefficient for lysis rate, h^{-1}
b_{ly}	Polynomial coefficient for lysis rate, h^{-2}
a_m	Polynomial coefficient for maintenance rate, h^{-2}
b_m	Polynomial coefficient for maintenance rate, h^{-1}
a_T	Stoichiometric coefficient, g(I)g(DW)$^{-1}$
a_{T1}	Stoichiometric coefficient, g(I)g(DW)$^{-1}$
f_{12}	Inactivation rate parameter, h^{-2}
k	first order biomass conversion coefficient
k_2	Degeneration coefficient, h^{-1}
k_4	CO_2 'yield' on biomass growth, g(DW)l^{-2}
k_5	Penicillin production related coefficient for CO_2 formation, l^{2}g(P)$^{-1}$
k_{12}	Inactivation coefficient, h^{-1}
k_l	Gas→liquid mass transfer coefficient, m^{-2}h^{-1}
k_{ly}	Lysis coefficient, h^{-1}
k_p	Precursor formation coefficient, g(pP)g(DW)$^{-1}$h^{-1}
k_{pen}	Coefficient for rate of formation of penicillin from precursor, g(P)g(DW)g(pP)$^{-2}$h^{-1})
k_s	Vacuole growth rate constant, l^{-1}h^{-1}

m_O	Maintenance coefficient for oxygen, g(O_2)g(DW)$^{-1}$h^{-1}
m_c	Maintenance-related coefficient for CO_2 formation, g(CO_2)g(DW)$^{-1}$h^{-1}
m_s	Maintenance coefficient, g(S)g(DW)$^{-1}$h^{-1}
m_{s_H}	Maintenance coefficient in the model of Heijnen *et al.* (1979), moles(S)moles(DW)$^{-1}$h^{-1}
m_{s0}	Maintenance coefficient for state 0, g(S)g(DW)$^{-1}$h^{-1}
m_{s1}	Maintenance coefficient for state 1, g(S)g(DW)$^{-1}$h^{-1}
$\bar{n}_k$	Number of vacuoles in bin k
pP	Concentration of penicillin precursor, g(pP)l^{-1}
q_L	Uptake rate of lactose g(L)g(DW)$^{-1}$h^{-1}
q_{L0}	Uptake coefficient for lactose, g(L)g(DW)$^{-1}$h^{-1}
q_S	Uptake rate of glucose, g(S)g(DW)$^{-1}$h^{-1}
q_{S0}	Uptake coefficient for glucose, g(S)g(DW)$^{-1}$h^{-1}
$q_{S,max}$	Maximum specific glucose consumption rate, moles(S)moles(X)$^{-1}$h^{-1}
q_T	Total rate of use of glucose and equivalents for biomass growth in the model of Kluge *et al.* (1992), g(S)h^{-1}
q_Y	Uptake rate of lysed material, g(Y)g(DW)$^{-1}$h^{-1}
q_{Y0}	Uptake coefficient for lysed material, g(Y)g(DW)$^{-1}$h^{-1}
r_C	Rate of consumption of carbon, moles h^{-1}
r_O	Rate of consumption of oxygen, moles h^{-1}
r_S	Rate of consumption of glucose, moles h^{-1}
r_P	Rate of formation of penicillin, moles h^{-1}
r_{PO}	Maximum rate of formation of penicillin, moles h^{-1}
r_X	Rate of formation of biomass, moles h^{-1}
r_k	Radius of smallest vacuoles, m
r_l	Radius of largest vacuoles, m
t	Time, h
v_{ic}	Volume of active cytoplasm, m^3

Greek Symbols

α	Coefficient relating nutritional value of lysed material to that of glucose, g(Y)g(S)$^{-1}$
α_n	Coefficient relating substrate consumption to P_1 assimilation, g(S)g(DW)$^{-1}$
β	Coefficient relating nutritional value of lactose to that

	of glucose, g(L)g(S)$^{-1}$
γ_n	Product n production coefficient, g(P_n)g(DW)$^{-1}$
ε_n	Maintenance coefficient n (for conversion between biomass states), g(S)g(DW)$^{-1}$
ζ_n	Degeneration numerator coefficient n, g(S)l^{-1}h^{-1}
η_n	Degeneration numerator coefficient n, h^{-1}
κ_1	Biomass age function, g(DW)hl^{-1}
κ_2	Mean age of biomass, h
λ	Differentiation denominator coefficient, g(S)l^{-1}
λ_n	Differentiation denominator coefficient n, g(S)l^{-1}
μ	Specific growth rate, h^{-1}
μ_I	Specific inactivation rate, $^{-1}$
μ_L	maximum lactose conversion rate
μ_P	Penicillin production constant, h^{-1}
$\mu_{PM,max}$	Growth rate with respect to Pharmamedia, h^{-1}
μ_{PO}	Penicillin production coefficient, g(P)g(DW)$^{-1}$
$\mu_{P_{SS}}$	Steady state penicillin production constant, g(P)g(DW)$^{-1}$h^{-1}
$\mu_{S,max}$	Growth rate with respect to glucose, h^{-1}
μ_X	Growth constant h^{-1}
μ_1	Growth rate, h^{-1}
μ_1	vacuole formation coefficient
μ_2	vacuole growth coefficient
μ_3	vacuole loss coefficient
μ_a	Autolysis coefficient, h^{-1}
μ_n	Specific growth rate n, h^{-1}
μ_{substr}	Specific growth rate on substrate, h^{-1}
ν	Branching numerator coefficient, h^{-1}
ν_n	Branching numerator coefficient n, h^{-1}
ξ_n	Differentiation denominator coefficient n, g(S)l^{-1}
π	Penicillin production rate, g(P)g(DW)$^{-1}$h^{-1}
π	the constant
ρ_3	Density of biomass fraction 3, gm^{-3}
ρ_c	Density of cytoplasm, gm^{-3}
σ	Substrate consumption rate, g(S)l^{-1}
σ_n	P_1 formation rate coefficient, g(P_1)g(DW)$^{-1}$h^{-1}
τ	Time, h
ϕ_S	Rate of feeding of glucose, moles h^{-1}

ω_n	Differentiation coefficient n, h^{-1}

B. MATLAB ROUTINES

This appendix contains sample listings of MATLAB programs typical of those used in the course of the work described in this thesis. Examples of the following programs are given.

- `eg_tune.m`—script to run least squares optimisation of model parameters
- `eg_targ.m`—MATLAB function, called as the objective function for least squares optimisation
- `eg_data.m`—script to set up initial values for parameters and a typical feed profile
- `fb14.m`—script containing measured fermentation data, called to specify reference values for `eg_targ.m`
- `eg_scrga.m`—script to run Genetic Algorithm based optimisation
- `eg_objga.m`—MATLAB function to serve as the objective function for Genetic Algorithm based optimisation (this function calls `eg_targ.m`, and therefore the function being minimised by both least squares routine and Genetic Algorithm is the same

A SIMULINK block diagram for the model used by the above example scripts and functions is shown in Figure 2.4. This example model is that of Paul *et al.* (1998).

```
% This script file runs a least squares optimisation on
% an example model.
%
% MTS July 1998

% Declaring the parameters to be tuned to be global,
% so that changes made to their values by the optimisation
% routine, when calling the objective function, are reflected
% in their values, used by SIMULINK, in the MATLAB workspace.
global mu0 mue m0 m1 gamma1 mup muh
global alpha0 alphae alphap K0 Ke K1 K2 Kp Ki
global mu1 mu2 mua Ksl Ksi mul mu3

% Setting globals for initial conditions and input feed rate
% so that they may be modified along with the parameter values.
global Qi
global x0init x1init x2init x3init x4init sinit linit vinit pinit

% Creating a vector of control options
% for the optimisation routine
options = foptions;                    % The default vector
options(1) = 1;                        % Report progress
options(16) = 0.01*sqrt(eps);          % Minimum step length for df/dx
options(17) = 0.01*sqrt(eps);          % Maximum step length for df/dx

% Initialising parameter values for the optimisation
eg_data

% Creating the initial parameters vector for the optimisation
x0 = [mu0 mue m0 m1 gamma1 mup muh alpha0 ...
      alphae alphap K0 Ke K1 K2 Kp Ki ...
      mu1 mu2 mua Ksl Ksi mul mu3];

% Creating the variable 'dataname'
dataname = 'fb14';

% Calling the MATLAB least squares optimisation routine
[x] = leastsq('eg_targ',[x0],options,[],dataname);
```

Tab. B.1: Listing for `eg_tun.m`

```
function [error] = eg_targ(x,dataname)
% Function to simulate the updated Paul/Thomas model
% and return the errors between measured and simulated state values.
%
% Inputs
% x           vector containing parameter values for optimisation
% dataname    string containing the name of the data set to be used
%
% MTS July 1998

% Declare globals for optimisation
% The parameters to be optimised need to be declared global
% so that any changes to the parameter values made in this
% program (as a consequence of the optimisation algorithm)
% affect the corresponding parameter values in the MATLAB
% workspace which are used in simulating the SIMULINK model.
global mu0 mue m0 m1 gamma1 mup muh
global alpha0 alphae alphap K0 Ke K1 K2 Kp Ki
global mu1 mu2 mua Ksl Ksi mul mu3

% Setting globals for initial conditions and input feed rate
% Again, this is so that values changed here (dependent on
% the data set used) are reflected in the MATLAB workspace.
global Qi
global x0init x1init x2init x3init x4init sinit linit vinit pinit

% Running a script file to load in data values
eval(dataname)

% Avoiding negative parameter values
x=abs(x);

% Creating parameter values from the input x vector
mu0    = x(1);   mue    = x(2);    m0  = x(3);    m1     = x(4);
gamma1 = x(5);   mup    = x(6);    muh = x(7);    alpha0 = x(8);
alphae = x(9);   alphap = x(10);   K0  = x(11);   Ke     = x(12);
K1     = x(13);  K2     = x(14);   Kp  = x(15);   Ki     = x(16);
mu1    = x(17);  mu2    = x(18);   mua = x(19);   Ksl    = x(20);
Ksi    = x(21);  mul    = x(22);   mu3 = x(23);
```

```
% Call the simulation
% Here Gear's algorithm is being used on the model 'eg_modl'
% running from time = 0 to the maximum in the reference times
% vector (T_ref), with no specified initial conditions
% ([] - initial conditions have specified above as x0init, etc
%  since the order of the states may change on resaving the model),
% and with tolerance 1e-6, minimum step length 1e-6, and
% maximum step length 10. (Time units for simulations are hours.)
% "fred" is used as a dummy variable to store the returned
% simulation time. If SIMULINK integration routines are called
% from the command line without output variables, the states are
% automatically plotted graphically, which takes time, and so
% is undesirable as well as unnecessary for parameter optimisation.
[fred]=gear('eg_modl',[0 max(T_ref)],[],[1e-6 1e-10 10]);

% Interpolate to find the simulated values at measurement times
% Here we are tuning for glucose (S), penicillin (P), lactose (L),
% and all biomass states (X0, X1, X2, X3) excluding the
% unmeasurable lysed state (X4).
x0_value = interp1(fred, X0, T_ref);
x1_value = interp1(fred, X1, T_ref);
x2_value = interp1(fred, X2, T_ref);
x3_value = interp1(fred, X3, T_ref);

glucose_value = interp1(fred, S, T_ref);
pen_value = interp1(fred, P, T_ref);
l_value = interp1(fred,L,T_ref);

%% Obtaining simulated values at measurement times
%% Alternative form, typically for models with hourly output
%% sampling times and reference data at hourly intervals.
% x0_value = X0(T_ref+1);
% x1_value = X1(T_ref+1);
% x2_value = X2(T_ref+1);
% x3_value = X3(T_ref+1);
% glucose_value = S(T_ref+1);
% pen_value = P(T_ref+1);
% l_value = L(T_ref+1);

% Calculate the absolute errors between measured and simulated values
```

```
x0_error = (x0_value - X0_ref)/max(X0_ref);
x1_error = (x1_value - X1_ref)/max(X1_ref);
x2_error = (x2_value - X2_ref)/max(X2_ref);
x3_error = (x3_value - X3_ref)/max(X3_ref);

glucose_error = (glucose_value - S_ref)/max(S_ref);
pen_error = (pen_value - P_ref)/max(P_ref);
l_error = (l_value - L_ref)/max(L_ref);

% Concatenate values to form a matrix so as to return a
% single error variable to the least squares routine.
% ... indicates continuation from line to line in MATLAB
error = [x0_error,x1_error,x2_error,x3_error,...
         glucose_error, pen_error, l_error];
```

Tab. B.2: Listing for `eg_targ.m`

```
% This is a script to set up parameters for use in the
% SIMULINK simulation of the example model.

% Model parameters to be optimised
mu0    = 0.065;    mue    = 0.28;    m0  = 0.029;    m1     = 0.029;
gamma1 = 0.01042;  mup    = 0.02985; muh = 0.003;    alpha0 = 2.1;
alphae = 1.00;     alphap = 1.00;    K0  = 0.05;     Ke     = 0.05;
K1     = 0.05;     K2     = 1.041;   Kp  = 0.0002;   Ki     = 0.024;
mu1    = 0.012;    mu2    = 0.023;   mua = 5.142e-3; Ksl    = 0.7371;
Ksi    = 1.353e-3; mul    = 0.1015;  mu3 = 4.461e-3;

% Fixed parameters (biomass density)
rho = 0.35;
rho3 = rho;

% Feed rate parameters
f  = 0.008;
fx = 0.004;
sf = 500;
t_ref = [0 24 24 160 161]';
Qi    = 6/sf*[1 1 1 1 1]';              %% Feed + precursor dilution
Qo    = [0 0 0 0 0]';
sr = .009;

% NOTE that all SIMULINK model parameters relevant to a specific
% data set are set up in that data set's script file, called from
% within the optimisation's objective function. The file 'fb14.m'
% is given as an example.
```

Tab. B.3: Listing for `eg_data.m`

```
% Data obtained from a fixed feed rate fermentation
% FB14:
% TIME  A0      A1    A2    A3    AT    S     SL     P
data = [
   1.0  0.15   0.89  0.15  0.02  1.06  1.52  5.80  0.03
  15.0  1.04   3.87  0.55  0.00  4.91  9.58  5.81  0.05
  21.0  1.39   7.42  0.80  0.03  8.84  8.57  5.76  0.08
  25.0  2.66  10.04  0.95  0.04 12.73  6.47  5.67  0.12
  29.0  2.60  11.34  1.63  0.09 14.03  2.76  5.51  0.20
  39.0  2.34  14.56  2.57  0.31 17.20  0.0   4.81  0.72
  51.0  2.34  15.99  4.22  0.74 19.08  0.0   2.79  1.70
  63.0  2.21  17.35  5.46  1.56 21.12  0.0   0.78  2.70
  73.0  2.70  18.42  6.27  1.56 22.68  0.0   0.0   3.40
  87.0  2.06  18.35  8.65  2.39 22.80  0.0   0.0   4.28
  97.0  3.54  16.52  8.17  3.40 23.45  0.0   0.0   4.86
 111.0  2.15  21.28  8.67  2.36 25.79  0.0   0.0   5.43
 121.0  2.25  18.88  8.98  5.16 26.29  0.0   0.0   5.57
];

% 'Unpacking' reference vectors from the data matrix
% (Although not necessary, this makes things easier to handle)
T_ref = data(:,1);
X0_ref = data(:,2);
X1_ref = data(:,3);
X2_ref = data(:,4);
X3_ref = data(:,5);
Xt_ref = data(:,6);
S_ref  = data(:,7);
L_ref  = data(:,8);
P_ref  = data(:,9);

% Initial Conditions and feed/sample rate information
x0init=.15;   x1init=.89;   x2init=.15;   x3init=.02;  x4init=0.0;
sinit =1.52;  linit=5.8;  pinit=0.0;
vinit =4.00;  FI=0.008;  SRI=0.009;  FXI=0.004;
Qi = (FI+FXI-SRI)*ones(size(Qi));
```

Tab. B.4: Listing for `fb14.m`

```
% Script file to run Genetic Algorithm optimisation of parameter
% values in the example model.

% Declaring the parameters to be tuned to be global,
% so that changes made to their values by the optimisation
% routine, when calling the objective function, are reflected
% in their values, used by SIMULINK, in the MATLAB workspace.
global mu0 mue m0 m1 gamma1 mup muh
global alpha0 alphae alphap K0 Ke K1 K2 Kp Ki
global mu1 mu2 mua Ksl Ksi mul mu3

% Setting globals for initial conditions and input feed rate
% so that they may be modified along with the parameter values.
global Qi
global x0init x1init x2init x3init x4init sinit linit vinit pinit

% Customising the options for the Genetic Algorithm
goptions(1,1) = 2;                % Output
goptions(14,1) = 100;             % MAXGEN
goptions(20,1) = 50;              % NIND
goptions(21,1) = 3;               % SUBPOP
goptions(23,1) = 2;               % SP
goptions(29,1) = 0;               % INITFUNCTION
goptions(30,1) = 6;               % STATEPLOT
goptions(5,1:5) = [0,0,3,3,3];    % selection function
goptions(6,1) = 0;                % mutation function
goptions(7,1:5) = [0,1,2,3,0];    % recombination function
goptions(28,1) = 0;               % SAVE2FILE

% Initialising parameter values for the optimisation
eg_data

% Creating the initial parameters vector for the optimisation
x0 = [mu0 mue m0 m1 gamma1 mup muh alpha0 ...
      alphae alphap K0 Ke K1 K2 Kp Ki ...
      mu1 mu2 mua Ksl Ksi mul mu3];

% Obtaining the bounds for our Genetic Algorithm search
objfun = 'eg_objga';
bounds = feval(objfun, [], 1,x0);
VLB = bounds(1,:); VUB = bounds(2,:);

method = 12;
Tstart = 0; Tend = 2; Fmax = 5/160; Raw_Time = t_ref;

% Calling the Genetic Algorithm optimsation routine
[xnew, GOPTIONS] = tbxdbga(objfun, goptions, VLB, VUB, method);
```

Tab. B.5: Listing for `eg_scrga.m`

```
function [ObjVal, t, x] = eg_objga(Chrom, switch,x0)
% This function is based on those provided with the
% Genetic and Evolutionary Algorithm Toolbox (GEAT)
% of Harmut Pohlheim
% Technical University Ilmenau, 1996
% http://www.systemtechnik.tu-ilmenau.de/~pohlheim/GA_Toolbox/index.html
%
% MTS July 1998

% global variable for setting initial bounds
% global x0

% Compute population parameters
   [Nind, Nvar] = size(Chrom);

% Check size of Chrom and do the appropriate thing
   % if Chrom is [], then
   if Nind == 0
      % lower and upper bound, identical for all n variables
      ObjVal = [1e-3*x0;10*x0];
   % compute values of function
   else
      % Start computation of objective function
      ObjVal = zeros(Nind,1);
      for indrun = 1:Nind
         % Convert vector from GA into parameter vector
         x = Chrom(indrun,:)';
         % Call the least squares objective function
         % from conventional model tuning
         % The (sum(sum(###.^2)) calculates the SSE
         ObjVal(indrun) = sum(sum(eg_targ(x,'fb14').^2));
      end
   end
```

Tab. B.6: Listing for `eg_objga.m`

C. PERL PROGRAM FOR PARSING MODEL EQUATIONS

A program written in Perl was used to automate the processing of the output generated from algebraic manipulations in Maple. Although distinguishing groups of terms on the basis of the measured quantities in them is relatively simple for small model systems, the large volume of output generated in analysing the simplified lactose-incorporating model of Paul and Thomas (1996) was such that the effort involved in writing a program to carry out the sorting operation was less than that which would have been needed to perform the sorting manually.

To generate the input for the Perl program, the model equations must be entered into Maple, multiplied throughout by their divisor terms and the resulting expressions sorted. Exporting the resulting Maple session as plain text produces a `.txt` file which may then be used as input to the Perl program. (Note that the output style option in Maple should be set to linetype notation.)

The Perl program prompts for the name of a `.txt` file, which is then used as input. Each Maple output line is split into its individual terms, sorting states and parameters, and output is then written to three sets of files.

- a `.out` file, containing tabulated output for each expression, consisting of a column of parameters and one of associated, distinct groups of

states

- a `.all` file, containing a sorted list of all parameter groups found in the Maple output within the `.txt` file
- a group of `.tex` files, one for each expression in the Maple output, containing tables suitable for use with LaTeX

C.1 The Perl Program

The Perl program is listed here.

```
#!/usr/bin/perl
# This script is intended to read in a text file containing
# all the parameter groups generated by a differential equation,
# reorder the terms in each group alphabetically,
# sort the reordered groups alphabetically,
# and then write them to an output file.

# Where are we reading from and writing to?
print "Enter file to be sorted: ";chop($sourcename=<STDIN>);

# Check that we have a superficially valid filename
unless ($sourcename=~/txt/){
   die "The input file should be a .txt file."
}

# Create the various output file names
$outname = $sourcename; $outname =~ s/txt/out/;    # Full output
$allname = $sourcename; $allname =~ s/txt/all/;    # All parameter groups
$texname = $sourcename; $texname =~ s/txt//;       # Latex tables

# Keep the user informed about progress
print "Reading from $sourcename and writing to $outname.\n";
print "Also writing parameters to $allname.\n";
print "This could take a while. Please be patient.\n";
```

```
# A little light formatting
format SORTED =
^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<  ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
$currentlist{$ckey}                    $key
~~   ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<  ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
     $currentlist{$ckey}               $key
.

# The format lines for LISTALL originally had $listall{$key}, NOT $fred
format LISTALL =
   ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
   $fred
~~    ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
   $fred
.

# Attempting to cheat on the permissible word breaks
$: = $:."+";     # We add "+" as another mathematically permissible line-break

# Open the files involved
open(RAWLIST, "$sourcename");
open(SORTED, ">$outname");
open(LISTALL, ">$allname");
# open(TEXOUT, ">$texname");

# The main workhorse loop of the program
while ($nextgroup = <RAWLIST>) {              # Reading a line at a time
   %currentlist = ();                         # Clearing the current list

   # We only bother to process the line if it's Maple output
   # ie, if it starts with a ">" and contains a ":="
   if ((index($nextgroup,">") != 0) && ($nextgroup=~/:=/)){
      ($name, $expr) = split(/:=/,$nextgroup); # Split the name and expression
      $name  =~ s/\s//g;                      # Remove spaces from the name
      $expr  =~ s/\s//g;                      # Remove spaces from the expr
      print SORTED "$name := \n";             # Write the name to file
      print "Parsing $name\n";                # Say what we're doing
```

```
# Break up the expression into terms, go through element by element
foreach $elem (@barray=&break_expr($expr)) {       # subroutines at end

   # Break each element into parameters and states
   ($params, $states) = &split_by_stars($elem);    # subroutines at end

   # We need to see something in the file if a group has no parameter
   if (($params eq "+") || ($params eq "-") && ($states ne "")) {
      $params = $params."1";
   }

   # Add the parameter group to a hash indexed by state groupings
   $currentlist{$states} = $currentlist{$states} . $params;
   $listall{join(": ",$name,$states)} = $listall{join(": ",$name,$states)} . $params;
}

# Sort the collated parameter groups in the currentlist
foreach $key (keys(%currentlist)) {  # Each parameter group in turn
   @parray = sort(&break_expr($currentlist{$key}));  # Split and sort
   $currentlist{$key} = join("",@parray);            # Reassemble
}

# Sort the keys of the hash by entry length (or alphabetically)
# by_current_length is subroutine at end
@sortedlist = sort by_current_length keys(%currentlist);

# Write suitably modified output to the TEXOUT file
open(TEXOUT, ">$texname$name.tex");
print "$texname$name.tex\n";
print TEXOUT "\\subsection{$name} \n";
print TEXOUT "\\begin{align*} \n";
foreach $key (@sortedlist) {
   $texentry = &slash_greek($currentlist{$key});
   $texkey = &slash_greek(&tex_key($key));
   print TEXOUT $texkey . "  &&  " . $texentry . "\\\\\n";
}
print TEXOUT "\\end{align*} \n\n\n";
```

```
      close(TEXOUT);

      # Write each key and entry to the SORTED output file
      foreach $key (@sortedlist) {
         $ckey = $key;   # Dummy variable to allow array referencing
         write SORTED;   # Generate the .out file
      }

      # Spacing out the various sections in the .out file
      print SORTED "\n\n";
   }

}

# Sort the collated parameter groups in the overall list
foreach $key (keys(%listall)) {                  # Each parameter group in turn
   @parray = sort(&break_expr($listall{$key}));  # Split and sort
   $listall{$key} = join("",@parray);            # Reassemble
}

# Sort the keys of the overall list by entry length (or alphabetically)
@longsort = sort by_length keys(%listall);       # by_length is subroutine
foreach $key (@longsort) {                       # Produce the .all file
   ($name,$states) = split(": ",$key);
   $fred = $name . "\t" . $listall{$key};        # Dummy variable for LISTALL
   write LISTALL;
}

# Close the files involved
close(RAWLIST);
close(SORTED);
close(LISTALL);
# close(TEXOUT);

################################################################################
######################### SUBROUTINES FOLLOW ###################################
################################################################################
```

```
sub by_current_length {
    # Compare the lengths of the two strings, or their names, alphabetically
    (($currentlist{$a} =~ tr///c) <=> ($currentlist{$b} =~ tr///c)) || ($a cmp $b)
}

sub by_length {
    # Compare the lengths of the two strings, or their names, alphabetically
    (($listall{$a} =~ tr/-\+\*/-\+\*/) <=> ($listall{$b} =~ tr/-\+\*/-\+\*/)) ||
    (($listall{$a} =~ tr///c) <=> ($listall{$b} =~ tr///c)) ||
    (($listall{$a} =~ tr/-\+/-\+/) <=> ($listall{$b} =~ tr/-\+/-\+/)) ||
    ($a cmp $b)
}

sub break_expr {         # To break an expression into terms at + or -
                         # but only if +|- is outside ()
   # Declare some local variables...
   local($expr) = @_;
   local(@barray);
   local($counter,$breakable,$arraycount);

   # A more intelligent "split"
   # NOTE: Since the expression will always start with
   # a (+|-), then $arraycount needs to be set to -1
   # to ensure that the first term is stored in $barray[0].
   # Perl numbers from 0, and we increment $arraycount
   # whenever we encounter a (+|-).
   $counter = 0; $breakable = 1; $arraycount = -1;

   # In an expression, the first term may be unsigned.
   # Since we wish all terms to be signed, we prepend a "+" if needed.
   if ((substr($expr,0,1) ne "-") && (substr($expr,0,1) ne "+")) { # If no +|-
      $expr = "+" . $expr;                                        # Prepend +
   }

while (($test = substr($expr,$counter++,1)) ne "") {    # Split at +|- not in ()
   if (($test eq "+" || $test eq "-") && $breakable) {  # If +|- & not in ()
      $arraycount++;                                    # New array element
   }  elsif ($test eq "(") {                            # If an opening bracket
```

```
      $breakable--;                                    # Breakable false (0)
   }  elsif ($test eq ")") {                           # If a closing bracket
      $breakable++;                                    # Breakable true (1)
   }
   $barray[$arraycount] = join("",$barray[$arraycount], $test); # Add character
}
@barray;                                               # Return array of terms
}  # End of subroutine...

sub split_by_stars { # To split a term into its component states and parameters
    # Declare some local variables...
    local($expr) = @_;
    local($counter,$breakable,$arraycount,$sign);
    local(@barray,@statelist,@paramlist);

# A more intelligent "split"
$counter = 0; $breakable = 1; $arraycount = 0;

# First remove the +|- from the term
if (substr($expr,0,1) eq "-") {
   $sign = "-";
   $counter++;
} elsif (substr($expr,0,1) eq "+") {
   $sign = "+";
   $counter++;
} else {
   $sign = "+";
}

while (($test = substr($expr,$counter++,1)) ne "") { # Splitting at * outside ()
   if (($test eq "*") && $breakable) {
      $arraycount++;
   }  elsif ($test eq "(") {
      $breakable--;
      $barray[$arraycount] = join("",$barray[$arraycount], $test);
   }  elsif ($test eq ")") {
      $breakable++;
      $barray[$arraycount] = join("",$barray[$arraycount], $test);
```

```
   }  else {
      $barray[$arraycount] = join("",$barray[$arraycount], $test);
   }
}

foreach $term (@barray) {                 # Going through the terms
   if ((index($term,"(")>=0) || (index($term,"Sigma")>=0)) {
      if (substr($term,0,1) eq "(") {     # Bracketed term, special sort
         $sortedterm = "";
         substr($term,0,1) = "";          # Get rid of opening bracket
         substr($term,-1,1) = "";         # Get rid of closing bracket
         @termarray = &break_expr($term);  # Split it up...
         foreach $termkey (@termarray) {  # Go through the array
            ($tparams,$tstates) = &split_by_stars($termkey);
            if (($tparams eq "-") | ($tparams eq "+")) {
               # Empty parameter group
               $sortedterm = $sortedterm . $tparams . $tstates;
            } else {
               # We have some parameters
               $sortedterm = $sortedterm . $tparams . "*" . $tstates;
            }
         }
         #$term = "(" . $sortedterm . ")"; # Replace brackets around term
      }
      @statelist = (@statelist,$term);    # It's a state term
   }  else {
      $term =~ s/(\+|-)//g;               # It's a parameter term
      @paramlist = (@paramlist,$term);
   }
}

# Rejoining the components, with the sign joining the params
$states = join("*",sort(@statelist));
$params = $sign . join("*",sort(@paramlist));

@returnarray = ($params, $states);
}  # End of subroutine...
```

```
sub slash_greek {
   # Subroutine to replace greek letter names with their LaTeX equivalents
   local($new) = @_;
   $new =~ s/(alpha)/\\$1 /g;
   $new =~ s/(gamma)/\\$1 /g;
   $new =~ s/(mu)/\\$1 /g;
   $new =~ s/(rho)/\\$1 /g;
   $new =~ s/(Sigma)/\\$1 /g;
   $new;
}  # End of subroutine...

sub tex_key {
   # Subroutine to convert the array key into LaTeX
   local($new) = @_;
   $new =~ s/diff\(/\\left\( \\frac\{\\partial\}\{\\partial t\}/;
   $new =~ s/,t\)/\\right\)/;
   $new;
}  # End of subroutine...
```

C.2 A Maple Session

The portion of a Maple session related to the glucose concentration's differential equation is given here.

```
> Sexpr := - diff(S(t),t) -alpha0*mu0*S(t)*X1(t)/(K0+S(t)) -
alphae*mue*S(t)*X0(t)/(Ke+S(t)) - m0*X0(t)*S(t)/(K1+S(t)) -
m1*rho*v1c(t)*S(t)/(K2+S(t)) - alphap*mup*rho*v1c(t)*S(t)/(Kp+S(t)*(1+
S(t)/Ki)) + mul*L(t)*(X0(t)+X1(t))/((Ksl+L(t))*(1+(S(t)/Ksi))) -
Sigma2*S(t)/V(t) + (Ff*sf)/V(t):
> Sexpr := sort(collect(simplify(Sexpr * (K0+S(t))* (Ke+S(t)) * (K1+
S(t)) * (K2+S(t)) * (Kp+S(t)*(1+S(t)/Ki)) * ((Ksl+L(t))*(1+
(S(t)/Ksi))) * Ki * Ksi *
V(t)),[alpha0,mu0,K0,alphae,mue,Ke,m0,m1,K1,K2,alphap,mup,Kp,Ki,mul,Ks
l,Ksi],distributed),{X0(t),X1(t),S(t),L(t),V(t)});
```

```
Sexpr := -alphae*mue*V(t)*X0(t)*S(t)^7*L(t)-m0*V(t)*X0(t)*S(t)^7*L(t)-
alpha0*mu0*V(t)*S(t)^7*X1(t)*L(t)-alphae*mue*Ksl*V(t)*X0(t)*S(t)^7-
m0*Ksl*V(t)*X0(t)*S(t)^7-alphae*mue*K2*V(t)*X0(t)*S(t)^6*L(t)-
m0*Ksi*V(t)*X0(t)*S(t)^6*L(t)-alphae*mue*Ki*V(t)*X0(t)*S(t)^6*L(t)-
alphae*mue*K0*V(t)*X0(t)*S(t)^6*L(t)-m0*K2*V(t)*X0(t)*S(t)^6*L(t)-
alphae*mue*Ksi*V(t)*X0(t)*S(t)^6*L(t)-m0*Ke*V(t)*X0(t)*S(t)^6*L(t)-
m0*K0*V(t)*X0(t)*S(t)^6*L(t)-m0*Ki*V(t)*X0(t)*S(t)^6*L(t)-
alphae*mue*K1*V(t)*X0(t)*S(t)^6*L(t)-alpha0*mu0*Ksl*V(t)*S(t)^7*X1(t)-
diff(S(t),t)*V(t)*S(t)^7*L(t)-alpha0*mu0*K2*V(t)*S(t)^6*X1(t)*L(t)-
alpha0*mu0*Ksi*V(t)*S(t)^6*X1(t)*L(t)-
alpha0*mu0*Ke*V(t)*S(t)^6*X1(t)*L(t)-
alpha0*mu0*K1*V(t)*S(t)^6*X1(t)*L(t)-
alpha0*mu0*Ki*V(t)*S(t)^6*X1(t)*L(t)-Sigma2*S(t)^8*L(t)-
alphae*mue*Ksl*Ksi*V(t)*X0(t)*S(t)^6-m0*K0*Ksl*V(t)*X0(t)*S(t)^6-
alphae*mue*K2*Ksl*V(t)*X0(t)*S(t)^6-
alphae*mue*Ki*Ksl*V(t)*X0(t)*S(t)^6-m0*Ksl*Ksi*V(t)*X0(t)*S(t)^6-
alphae*mue*K0*Ksl*V(t)*X0(t)*S(t)^6-m0*K2*Ksl*V(t)*X0(t)*S(t)^6-
m0*Ke*Ksl*V(t)*X0(t)*S(t)^6-m0*Ki*Ksl*V(t)*X0(t)*S(t)^6-
alphae*mue*K1*Ksl*V(t)*X0(t)*S(t)^6-m0*K2*Ksi*V(t)*X0(t)*S(t)^5*L(t)-
m0*Ke*K2*V(t)*X0(t)*S(t)^5*L(t)-
alphae*mue*Kp*Ki*V(t)*X0(t)*S(t)^5*L(t)-
m0*Kp*Ki*V(t)*X0(t)*S(t)^5*L(t)-m0*K0*K2*V(t)*X0(t)*S(t)^5*L(t)-
m0*Ke*Ki*V(t)*X0(t)*S(t)^5*L(t)-m0*K0*Ki*V(t)*X0(t)*S(t)^5*L(t)-
alphae*mue*K2*Ksi*V(t)*X0(t)*S(t)^5*L(t)-
alphae*mue*K1*Ksi*V(t)*X0(t)*S(t)^5*L(t)-
alphae*mue*K1*K2*V(t)*X0(t)*S(t)^5*L(t)-
alphae*mue*K0*Ki*V(t)*X0(t)*S(t)^5*L(t)-
m0*K0*Ke*V(t)*X0(t)*S(t)^5*L(t)-
alphae*mue*Ki*Ksi*V(t)*X0(t)*S(t)^5*L(t)-
m0*Ke*Ksi*V(t)*X0(t)*S(t)^5*L(t)-m0*Ki*Ksi*V(t)*X0(t)*S(t)^5*L(t)-
```

```
alphae*mue*K0*K2*V(t)*X0(t)*S(t)^5*L(t)-
alphae*mue*K2*Ki*V(t)*X0(t)*S(t)^5*L(t)-
m0*K2*Ki*V(t)*X0(t)*S(t)^5*L(t)-m0*K0*Ksi*V(t)*X0(t)*S(t)^5*L(t)-
alphae*mue*K0*Ksi*V(t)*X0(t)*S(t)^5*L(t)-
alphae*mue*K0*K1*V(t)*X0(t)*S(t)^5*L(t)-
alphae*mue*K1*Ki*V(t)*X0(t)*S(t)^5*L(t)-
alpha0*mu0*Ki*Ksl*V(t)*S(t)^6*X1(t)-
alpha0*mu0*K1*Ksl*V(t)*S(t)^6*X1(t)-
alpha0*mu0*K2*Ksl*V(t)*S(t)^6*X1(t)-
alpha0*mu0*Ksl*Ksi*V(t)*S(t)^6*X1(t)-
alpha0*mu0*Ke*Ksl*V(t)*S(t)^6*X1(t)-
alpha0*mu0*Ke*K2*V(t)*S(t)^5*X1(t)*L(t)-
alpha0*mu0*Kp*Ki*V(t)*S(t)^5*X1(t)*L(t)-
alpha0*mu0*Ke*Ksi*V(t)*S(t)^5*X1(t)*L(t)-
alpha0*mu0*K1*Ki*V(t)*S(t)^5*X1(t)*L(t)-
alpha0*mu0*K1*Ksi*V(t)*S(t)^5*X1(t)*L(t)-
alpha0*mu0*K2*Ki*V(t)*S(t)^5*X1(t)*L(t)-
alpha0*mu0*Ki*Ksi*V(t)*S(t)^5*X1(t)*L(t)-
alpha0*mu0*K1*K2*V(t)*S(t)^5*X1(t)*L(t)-
alpha0*mu0*Ke*Ki*V(t)*S(t)^5*X1(t)*L(t)-
alpha0*mu0*Ke*K1*V(t)*S(t)^5*X1(t)*L(t)-
alpha0*mu0*K2*Ksi*V(t)*S(t)^5*X1(t)*L(t)+Ff*sf*S(t)^7*L(t)-
m0*K2*Ki*Ksl*V(t)*X0(t)*S(t)^5-m0*K0*Ke*Ksl*V(t)*X0(t)*S(t)^5-
alphae*mue*K0*Ksl*Ksi*V(t)*X0(t)*S(t)^5-
alphae*mue*K1*Ki*Ksl*V(t)*X0(t)*S(t)^5-m0*Ke*K2*Ksl*V(t)*X0(t)*S(t)^5-
m0*K0*Ksl*Ksi*V(t)*X0(t)*S(t)^5-m0*K2*Ksl*Ksi*V(t)*X0(t)*S(t)^5-
m0*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^5-m0*Ke*Ksl*Ksi*V(t)*X0(t)*S(t)^5-
alphae*mue*K0*Ki*Ksl*V(t)*X0(t)*S(t)^5-
alphae*mue*K1*K2*Ksl*V(t)*X0(t)*S(t)^5-
alphae*mue*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^5-
```

```
alphae*mue*K0*K2*Ksl*V(t)*X0(t)*S(t)^5-
alphae*mue*K2*Ksl*Ksi*V(t)*X0(t)*S(t)^5-
alphae*mue*K0*K1*Ksl*V(t)*X0(t)*S(t)^5-m0*K0*K2*Ksl*V(t)*X0(t)*S(t)^5-
alphae*mue*K1*Ksl*Ksi*V(t)*X0(t)*S(t)^5-
m0*K0*Ki*Ksl*V(t)*X0(t)*S(t)^5-alphae*mue*K2*Ki*Ksl*V(t)*X0(t)*S(t)^5-
m0*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^5-alphae*mue*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^5-
m0*Ke*Ki*Ksl*V(t)*X0(t)*S(t)^5-m0*K0*Ke*K2*V(t)*X0(t)*S(t)^4*L(t)-
m0*K0*Ki*Ksi*V(t)*X0(t)*S(t)^4*L(t)-
m0*Kp*Ki*Ksi*V(t)*X0(t)*S(t)^4*L(t)-
alphae*mue*K0*Ki*Ksi*V(t)*X0(t)*S(t)^4*L(t)-
alphae*mue*K0*K1*K2*V(t)*X0(t)*S(t)^4*L(t)-
alphae*mue*K0*K2*Ki*V(t)*X0(t)*S(t)^4*L(t)-
m0*K0*Kp*Ki*V(t)*X0(t)*S(t)^4*L(t)-
m0*K0*Ke*Ksi*V(t)*X0(t)*S(t)^4*L(t)-
alphae*mue*K0*Kp*Ki*V(t)*X0(t)*S(t)^4*L(t)-
m0*K0*K2*Ksi*V(t)*X0(t)*S(t)^4*L(t)-
alphae*mue*K0*K2*Ksi*V(t)*X0(t)*S(t)^4*L(t)-
m0*K2*Kp*Ki*V(t)*X0(t)*S(t)^4*L(t)-
alphae*mue*K2*Kp*Ki*V(t)*X0(t)*S(t)^4*L(t)-
alphae*mue*K0*K1*Ki*V(t)*X0(t)*S(t)^4*L(t)-
alphae*mue*K1*K2*Ki*V(t)*X0(t)*S(t)^4*L(t)-
m0*K0*K2*Ki*V(t)*X0(t)*S(t)^4*L(t)-
alphae*mue*K1*K2*Ksi*V(t)*X0(t)*S(t)^4*L(t)-
m0*K2*Ki*Ksi*V(t)*X0(t)*S(t)^4*L(t)-
alphae*mue*K0*K1*Ksi*V(t)*X0(t)*S(t)^4*L(t)-
alphae*mue*K1*Ki*Ksi*V(t)*X0(t)*S(t)^4*L(t)-
alphae*mue*K1*Kp*Ki*V(t)*X0(t)*S(t)^4*L(t)-
m0*Ke*Ki*Ksi*V(t)*X0(t)*S(t)^4*L(t)-
m0*K0*Ke*Ki*V(t)*X0(t)*S(t)^4*L(t)-m0*Ke*K2*Ki*V(t)*X0(t)*S(t)^4*L(t)-
alphae*mue*Kp*Ki*Ksi*V(t)*X0(t)*S(t)^4*L(t)-
```

```
alphae*mue*K2*Ki*Ksi*V(t)*X0(t)*S(t)^4*L(t)-
m0*Ke*Kp*Ki*V(t)*X0(t)*S(t)^4*L(t)-
m0*Ke*K2*Ksi*V(t)*X0(t)*S(t)^4*L(t)-
alpha0*mu0*Ki*Ksl*Ksi*V(t)*S(t)^5*X1(t)-
alpha0*mu0*K1*Ksl*Ksi*V(t)*S(t)^5*X1(t)-
alpha0*mu0*K1*Ki*Ksl*V(t)*S(t)^5*X1(t)-
alpha0*mu0*Kp*Ki*Ksl*V(t)*S(t)^5*X1(t)-
alpha0*mu0*Ke*Ksl*Ksi*V(t)*S(t)^5*X1(t)-
alpha0*mu0*K2*Ksl*Ksi*V(t)*S(t)^5*X1(t)-
alpha0*mu0*K2*Ki*Ksl*V(t)*S(t)^5*X1(t)-
alpha0*mu0*Ke*K2*Ksl*V(t)*S(t)^5*X1(t)-
alpha0*mu0*Ke*Ki*Ksl*V(t)*S(t)^5*X1(t)-
alpha0*mu0*K1*K2*Ksl*V(t)*S(t)^5*X1(t)-
alpha0*mu0*Ke*K1*Ksl*V(t)*S(t)^5*X1(t)-
alpha0*mu0*K2*Ki*Ksi*V(t)*S(t)^4*X1(t)*L(t)-
alpha0*mu0*K1*K2*Ksi*V(t)*S(t)^4*X1(t)*L(t)-
alpha0*mu0*Ke*K1*K2*V(t)*S(t)^4*X1(t)*L(t)-
alpha0*mu0*K1*K2*Ki*V(t)*S(t)^4*X1(t)*L(t)-
alpha0*mu0*Ke*K1*Ksi*V(t)*S(t)^4*X1(t)*L(t)-
alpha0*mu0*Kp*Ki*Ksi*V(t)*S(t)^4*X1(t)*L(t)-
alpha0*mu0*K2*Kp*Ki*V(t)*S(t)^4*X1(t)*L(t)-
alpha0*mu0*Ke*Ki*Ksi*V(t)*S(t)^4*X1(t)*L(t)-
alpha0*mu0*Ke*K2*Ki*V(t)*S(t)^4*X1(t)*L(t)-
alpha0*mu0*Ke*K1*Ki*V(t)*S(t)^4*X1(t)*L(t)-
alpha0*mu0*K1*Kp*Ki*V(t)*S(t)^4*X1(t)*L(t)-
alpha0*mu0*Ke*K2*Ksi*V(t)*S(t)^4*X1(t)*L(t)-
alpha0*mu0*K1*Ki*Ksi*V(t)*S(t)^4*X1(t)*L(t)-
alpha0*mu0*Ke*Kp*Ki*V(t)*S(t)^4*X1(t)*L(t)-
alphae*mue*K2*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^4-
m0*K0*K2*Ksl*Ksi*V(t)*X0(t)*S(t)^4-m0*Ke*K2*Ksl*Ksi*V(t)*X0(t)*S(t)^4-
```

```
alphae*mue*K1*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^4-
m0*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^4-
alphae*mue*K0*K1*Ki*Ksl*V(t)*X0(t)*S(t)^4-
m0*K0*Ke*Ksl*Ksi*V(t)*X0(t)*S(t)^4-
alphae*mue*K0*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^4-
alphae*mue*K0*K2*Ksl*Ksi*V(t)*X0(t)*S(t)^4-
alphae*mue*K0*K1*K2*Ksl*V(t)*X0(t)*S(t)^4-
m0*K0*Ke*K2*Ksl*V(t)*X0(t)*S(t)^4-m0*K2*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^4-
alphae*mue*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^4-
m0*K0*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^4-
alphae*mue*K0*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^4-
alphae*mue*K1*K2*Ksl*Ksi*V(t)*X0(t)*S(t)^4-
m0*K0*Ke*Ki*Ksl*V(t)*X0(t)*S(t)^4-m0*K0*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^4-
m0*Ke*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^4-m0*K2*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^4-
alphae*mue*K0*K1*Ksl*Ksi*V(t)*X0(t)*S(t)^4-
alphae*mue*K1*K2*Ki*Ksl*V(t)*X0(t)*S(t)^4-
m0*Ke*K2*Ki*Ksl*V(t)*X0(t)*S(t)^4-m0*Ke*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^4-
m0*K0*K2*Ki*Ksl*V(t)*X0(t)*S(t)^4-
alphae*mue*K2*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^4-
alphae*mue*K0*K2*Ki*Ksl*V(t)*X0(t)*S(t)^4-
alphae*mue*K1*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^4-
alphae*mue*K0*K2*Kp*Ki*V(t)*X0(t)*S(t)^3*L(t)-
alphae*mue*K0*K2*Ki*Ksi*V(t)*X0(t)*S(t)^3*L(t)-
m0*K0*Ke*Kp*Ki*V(t)*X0(t)*S(t)^3*L(t)-
alphae*mue*K0*K1*Kp*Ki*V(t)*X0(t)*S(t)^3*L(t)-
m0*K0*Ke*K2*Ki*V(t)*X0(t)*S(t)^3*L(t)-
alphae*mue*K0*K1*K2*Ki*V(t)*X0(t)*S(t)^3*L(t)-
m0*K0*Kp*Ki*Ksi*V(t)*X0(t)*S(t)^3*L(t)-
m0*K0*K2*Ki*Ksi*V(t)*X0(t)*S(t)^3*L(t)-
m0*K0*Ke*Ki*Ksi*V(t)*X0(t)*S(t)^3*L(t)-
```

```
alphae*mue*K0*K1*Ki*Ksi*V(t)*X0(t)*S(t)^3*L(t)-
m0*K2*Kp*Ki*Ksi*V(t)*X0(t)*S(t)^3*L(t)-
m0*Ke*Kp*Ki*Ksi*V(t)*X0(t)*S(t)^3*L(t)-
alphae*mue*K1*K2*Kp*Ki*V(t)*X0(t)*S(t)^3*L(t)-
alphae*mue*K1*Kp*Ki*Ksi*V(t)*X0(t)*S(t)^3*L(t)-
m0*K0*Ke*K2*Ksi*V(t)*X0(t)*S(t)^3*L(t)-
m0*K0*K2*Kp*Ki*V(t)*X0(t)*S(t)^3*L(t)-
alphae*mue*K1*K2*Ki*Ksi*V(t)*X0(t)*S(t)^3*L(t)-
m0*Ke*K2*Kp*Ki*V(t)*X0(t)*S(t)^3*L(t)-
m0*Ke*K2*Ki*Ksi*V(t)*X0(t)*S(t)^3*L(t)-
alphae*mue*K2*Kp*Ki*Ksi*V(t)*X0(t)*S(t)^3*L(t)-
alphae*mue*K0*K1*K2*Ksi*V(t)*X0(t)*S(t)^3*L(t)-
alphae*mue*K0*Kp*Ki*Ksi*V(t)*X0(t)*S(t)^3*L(t)-
alpha0*mu0*Ke*K2*Ksl*Ksi*V(t)*S(t)^4*X1(t)-
alpha0*mu0*Ke*Kp*Ki*Ksl*V(t)*S(t)^4*X1(t)-
alpha0*mu0*K1*Kp*Ki*Ksl*V(t)*S(t)^4*X1(t)-
alpha0*mu0*Kp*Ki*Ksl*Ksi*V(t)*S(t)^4*X1(t)-
alpha0*mu0*Ke*K1*Ksl*Ksi*V(t)*S(t)^4*X1(t)-
alpha0*mu0*Ke*K1*Ki*Ksl*V(t)*S(t)^4*X1(t)-
alpha0*mu0*K1*K2*Ki*Ksl*V(t)*S(t)^4*X1(t)-
alpha0*mu0*K2*Kp*Ki*Ksl*V(t)*S(t)^4*X1(t)-
alpha0*mu0*Ke*K1*K2*Ksl*V(t)*S(t)^4*X1(t)-
alpha0*mu0*K2*Ki*Ksl*Ksi*V(t)*S(t)^4*X1(t)-
alpha0*mu0*Ke*K2*Ki*Ksl*V(t)*S(t)^4*X1(t)-
alpha0*mu0*K1*Ki*Ksl*Ksi*V(t)*S(t)^4*X1(t)-
alpha0*mu0*K1*K2*Ksl*Ksi*V(t)*S(t)^4*X1(t)-
alpha0*mu0*Ke*Ki*Ksl*Ksi*V(t)*S(t)^4*X1(t)-
alpha0*mu0*Ke*K2*Kp*Ki*V(t)*S(t)^3*X1(t)*L(t)-
alpha0*mu0*K1*K2*Kp*Ki*V(t)*S(t)^3*X1(t)*L(t)-
alpha0*mu0*Ke*K1*K2*Ki*V(t)*S(t)^3*X1(t)*L(t)-
```

```
alpha0*mu0*Ke*K1*Ki*Ksi*V(t)*S(t)^3*X1(t)*L(t)-
alpha0*mu0*Ke*K1*K2*Ksi*V(t)*S(t)^3*X1(t)*L(t)-
alpha0*mu0*K2*Kp*Ki*Ksi*V(t)*S(t)^3*X1(t)*L(t)-
alpha0*mu0*K1*Kp*Ki*Ksi*V(t)*S(t)^3*X1(t)*L(t)-
alpha0*mu0*K1*K2*Ki*Ksi*V(t)*S(t)^3*X1(t)*L(t)-
alpha0*mu0*Ke*K2*Ki*Ksi*V(t)*S(t)^3*X1(t)*L(t)-
alpha0*mu0*Ke*K1*Kp*Ki*V(t)*S(t)^3*X1(t)*L(t)-
alpha0*mu0*Ke*Kp*Ki*Ksi*V(t)*S(t)^3*X1(t)*L(t)-
m0*K0*Ke*K2*Ki*Ksl*V(t)*X0(t)*S(t)^3-
alphae*mue*K0*K2*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^3-
m0*K0*Ke*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^3-
alphae*mue*K1*K2*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^3-
alphae*mue*K2*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^3-
m0*Ke*K2*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^3-
alphae*mue*K1*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^3-
m0*Ke*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^3-
m0*K0*K2*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^3-
alphae*mue*K1*K2*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^3-
m0*K0*K2*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^3-
alphae*mue*K0*K1*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^3-
alphae*mue*K0*K1*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^3-
m0*K0*Ke*K2*Ksl*Ksi*V(t)*X0(t)*S(t)^3-
alphae*mue*K0*K1*K2*Ksl*Ksi*V(t)*X0(t)*S(t)^3-
alphae*mue*K0*K1*K2*Ki*Ksl*V(t)*X0(t)*S(t)^3-
alphae*mue*K0*K2*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^3-
alphae*mue*K0*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^3-
m0*Ke*K2*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^3-
m0*K0*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^3-
m0*K2*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^3-
m0*K0*Ke*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^3-
```

```
m0*K0*Ke*K2*Ki*Ksi*V(t)*X0(t)*S(t)^2*L(t)-
alphae*mue*K0*K2*Kp*Ki*Ksi*V(t)*X0(t)*S(t)^2*L(t)-
m0*K0*K2*Kp*Ki*Ksi*V(t)*X0(t)*S(t)^2*L(t)-
alphae*mue*K0*K1*Kp*Ki*Ksi*V(t)*X0(t)*S(t)^2*L(t)-
m0*K0*Ke*K2*Kp*Ki*V(t)*X0(t)*S(t)^2*L(t)-
m0*K0*Ke*Kp*Ki*Ksi*V(t)*X0(t)*S(t)^2*L(t)-
m0*Ke*K2*Kp*Ki*Ksi*V(t)*X0(t)*S(t)^2*L(t)-
alphae*mue*K0*K1*K2*Ki*Ksi*V(t)*X0(t)*S(t)^2*L(t)-
alphae*mue*K0*K1*K2*Kp*Ki*V(t)*X0(t)*S(t)^2*L(t)-
alphae*mue*K1*K2*Kp*Ki*Ksi*V(t)*X0(t)*S(t)^2*L(t)-
alpha0*mu0*Ke*K1*K2*Ksl*Ksi*V(t)*S(t)^3*X1(t)-
alpha0*mu0*Ke*K1*Kp*Ki*Ksl*V(t)*S(t)^3*X1(t)-
alpha0*mu0*K1*K2*Kp*Ki*Ksl*V(t)*S(t)^3*X1(t)-
alpha0*mu0*Ke*K2*Kp*Ki*Ksl*V(t)*S(t)^3*X1(t)-
alpha0*mu0*Ke*K2*Ki*Ksl*Ksi*V(t)*S(t)^3*X1(t)-
alpha0*mu0*K2*Kp*Ki*Ksl*Ksi*V(t)*S(t)^3*X1(t)-
alpha0*mu0*Ke*K1*K2*Ki*Ksl*V(t)*S(t)^3*X1(t)-
alpha0*mu0*K1*K2*Ki*Ksl*Ksi*V(t)*S(t)^3*X1(t)-
alpha0*mu0*Ke*K1*Ki*Ksl*Ksi*V(t)*S(t)^3*X1(t)-
alpha0*mu0*K1*Kp*Ki*Ksl*Ksi*V(t)*S(t)^3*X1(t)-
alpha0*mu0*Ke*Kp*Ki*Ksl*Ksi*V(t)*S(t)^3*X1(t)-
alpha0*mu0*Ke*K1*Kp*Ki*Ksi*V(t)*S(t)^2*X1(t)*L(t)-
alpha0*mu0*Ke*K2*Kp*Ki*Ksi*V(t)*S(t)^2*X1(t)*L(t)-
alpha0*mu0*K1*K2*Kp*Ki*Ksi*V(t)*S(t)^2*X1(t)*L(t)-
alpha0*mu0*Ke*K1*K2*Ki*Ksi*V(t)*S(t)^2*X1(t)*L(t)-
alpha0*mu0*Ke*K1*K2*Kp*Ki*V(t)*S(t)^2*X1(t)*L(t)-
alphae*mue*K1*K2*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^2-
m0*K0*Ke*K2*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^2-
alphae*mue*K0*K1*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^2-
alphae*mue*K0*K1*K2*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^2-
```

```
m0*K0*K2*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^2-
alphae*mue*K0*K1*K2*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^2-
alphae*mue*K0*K2*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^2-
m0*Ke*K2*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^2-
m0*K0*Ke*K2*Kp*Ki*Ksl*V(t)*X0(t)*S(t)^2-
m0*K0*Ke*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)^2-
alphae*mue*K0*K1*K2*Kp*Ki*Ksi*V(t)*X0(t)*S(t)*L(t)-
m0*K0*Ke*K2*Kp*Ki*Ksi*V(t)*X0(t)*S(t)*L(t)-
alpha0*mu0*Ke*K1*K2*Ki*Ksl*Ksi*V(t)*S(t)^2*X1(t)-
alpha0*mu0*K1*K2*Kp*Ki*Ksl*Ksi*V(t)*S(t)^2*X1(t)-
alpha0*mu0*Ke*K1*K2*Kp*Ki*Ksl*V(t)*S(t)^2*X1(t)-
alpha0*mu0*Ke*K2*Kp*Ki*Ksl*Ksi*V(t)*S(t)^2*X1(t)-
alpha0*mu0*Ke*K1*Kp*Ki*Ksl*Ksi*V(t)*S(t)^2*X1(t)-
alpha0*mu0*Ke*K1*K2*Kp*Ki*Ksi*V(t)*S(t)*X1(t)*L(t)-
alphae*mue*K0*K1*K2*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)-
m0*K0*Ke*K2*Kp*Ki*Ksl*Ksi*V(t)*X0(t)*S(t)-
alpha0*mu0*Ke*K1*K2*Kp*Ki*Ksl*Ksi*V(t)*S(t)*X1(t)+(-
diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ksi*Ki*Kp*Ke+(-1/2*V(t)*S(t)^4*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^4)*Ksl*Ki*Kp*Ke*m1+(-
diff(S(t),t)*V(t)*S(t)^2*L(t)-Sigma2*S(t)^3*L(t)+
Ff*sf*S(t)^2*L(t))*Ksi*Ki*Kp*K2*K1+(-diff(S(t),t)*V(t)*S(t)^3*L(t)-
Sigma2*S(t)^4*L(t)+Ff*sf*S(t)^3*L(t))*Ksi*Ki*K2*K0+(-
diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksi*Ksl*Ki*K2*K1+
(-1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ki*Ke*K0*m1+(-1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksl*Ki*Kp*K1*K0*m1+(-1/2*V(t)*S(t)^7*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^7)*Ksl*m1+(-diff(S(t),t)*V(t)*S(t)^3-
Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksi*Ksl*Ki*Kp*Ke+(-
diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
```

```
Ff*sf*S(t)^3*L(t))*Ksi*Ki*Kp*K1+(-diff(S(t),t)*V(t)*S(t)^3-
Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksi*Ksl*Ki*Kp*K1+(-
diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksi*Ksl*Ki*K1*Ke+
(-diff(S(t),t)*V(t)*S(t)^2-Sigma2*S(t)^3+
Ff*sf*S(t)^2)*Ksi*Ksl*Ki*Kp*K1*Ke+(-diff(S(t),t)*V(t)*S(t)^3-
Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksi*Ksl*Ki*Kp*K2+(-
diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+Ff*sf*S(t)^5)*Ksl*Ki*K0+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*K2*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^5*X1(t)*L(t)+1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*Ke*K0*m1+(-
diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksi*Ksl*Ki*K1*K0+
(-1/2*V(t)*S(t)^5*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^5)*Ksl*Ke*K0*m1+(-
1/2*V(t)*S(t)^4*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^4)*Ksl*Ki*Ke*K0*m1+(-
diff(S(t),t)*V(t)*S(t)^2-Sigma2*S(t)^3+
Ff*sf*S(t)^2)*Ksi*Ksl*Ki*Kp*K2*Ke+(-diff(S(t),t)*V(t)*S(t)^2-
Sigma2*S(t)^3+Ff*sf*S(t)^2)*K0*Ke*K2*Kp*Ki*Ksl+(-
diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
Ff*sf*S(t)^4*L(t))*Ksi*Ki*K0+(-diff(S(t),t)*V(t)-Sigma2*S(t)+
Ff*sf)*K0*Ke*K1*K2*Kp*Ki*Ksl*Ksi+(-diff(S(t),t)*V(t)*S(t)^5*L(t)-
Sigma2*S(t)^6*L(t)+Ff*sf*S(t)^5*L(t))*K0*K1+(-
diff(S(t),t)*V(t)*S(t)^5*L(t)-Sigma2*S(t)^6*L(t)+
Ff*sf*S(t)^5*L(t))*Kp*Ki+(-diff(S(t),t)*V(t)*S(t)^5*L(t)-
Sigma2*S(t)^6*L(t)+Ff*sf*S(t)^5*L(t))*K0*Ke+(-
diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ksi*Ki*K2*K1+(-1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ki*Kp*K1*K0*m1+(-
diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ksi*Ki*K1*K0+(-diff(S(t),t)*V(t)*S(t)^4-
Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksi*Ksl*Ki*K2+(-
diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
```

```
Ff*sf*S(t)^4*L(t))*K0*Ke*K2+(-diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+
Ff*sf*S(t)^3)*Ksl*Ki*Kp*K1*Ke+(-diff(S(t),t)*V(t)*S(t)^2*L(t)-
Sigma2*S(t)^3*L(t)+Ff*sf*S(t)^2*L(t))*Ksi*Ki*Kp*K2*Ke+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksi*Ksl*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)^2-Sigma2*S(t)^3+
Ff*sf*S(t)^2)*K0*K1*K2*Kp*Ki*Ksl+(-diff(S(t),t)*V(t)*S(t)^2-
Sigma2*S(t)^3+Ff*sf*S(t)^2)*Ksi*Ksl*Ki*Kp*K2*K1+(-
diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksi*Ksl*Ki*Kp*K0+
(-1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksl*K2*K1*K0*Ki*mup*alphap+(-
diff(S(t),t)*V(t)*S(t)^2*L(t)-Sigma2*S(t)^3*L(t)+
Ff*sf*S(t)^2*L(t))*Ksi*Ki*Kp*K2*K0+(-diff(S(t),t)*V(t)*S(t)^3-
Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksi*Ksl*Ki*K2*K0+(V(t)*X0(t)*S(t)^2*L(t)+
V(t)*S(t)^2*X1(t)*L(t))*Ki*K1*Ke*K0*Ksi*mul+(-
diff(S(t),t)*V(t)*S(t)^2-Sigma2*S(t)^3+
Ff*sf*S(t)^2)*Ksi*Ksl*Ki*K2*K1*K0+(-diff(S(t),t)*V(t)*S(t)-
Sigma2*S(t)^2+Ff*sf*S(t))*K0*K1*K2*Kp*Ki*Ksl*Ksi+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ki*Kp*Ke*K0*m1+(-
diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ksi*Ki*Ke*K0+(-diff(S(t),t)*V(t)*S(t)^2*L(t)-
Sigma2*S(t)^3*L(t)+Ff*sf*S(t)^2*L(t))*K0*Ke*K2*Kp*Ki+(-
diff(S(t),t)*V(t)*S(t)-Sigma2*S(t)^2+
Ff*sf*S(t))*K0*Ke*K2*Kp*Ki*Ksl*Ksi+(-diff(S(t),t)*V(t)*S(t)^2*L(t)-
Sigma2*S(t)^3*L(t)+Ff*sf*S(t)^2*L(t))*Ke*K1*K2*Kp*Ki+(-
diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ksi*Ki*K1*Ke+(-diff(S(t),t)*V(t)*S(t)^2*L(t)-
Sigma2*S(t)^3*L(t)+Ff*sf*S(t)^2*L(t))*Ksi*Ki*Kp*K1*Ke+(-
diff(S(t),t)*V(t)*S(t)^5*L(t)-Sigma2*S(t)^6*L(t)+
Ff*sf*S(t)^5*L(t))*Ke*K1+(-diff(S(t),t)*V(t)*S(t)^6*L(t)-
```

```
Sigma2*S(t)^7*L(t)+Ff*sf*S(t)^6*L(t))*Ksi+(-
diff(S(t),t)*V(t)*S(t)^6*L(t)-Sigma2*S(t)^7*L(t)+
Ff*sf*S(t)^6*L(t))*Ki+(-1/2*V(t)*S(t)^5*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*Ki*Ke*m1+(V(t)*X0(t)*S(t)^4*L(t)+
V(t)*S(t)^4*X1(t)*L(t))*K2*Ke*Ksi*mul+(-diff(S(t),t)*V(t)*S(t)^2*L(t)-
Sigma2*S(t)^3*L(t)+Ff*sf*S(t)^2*L(t))*Ksi*Ki*K2*K1*K0+(-
diff(S(t),t)*V(t)*S(t)^2*L(t)-Sigma2*S(t)^3*L(t)+
Ff*sf*S(t)^2*L(t))*K0*K1*K2*Kp*Ki+(V(t)*X0(t)*S(t)^4*L(t)+
V(t)*S(t)^4*X1(t)*L(t))*Ki*Kp*Ksi*mul+(-diff(S(t),t)*V(t)*S(t)^2*L(t)-
Sigma2*S(t)^3*L(t)+Ff*sf*S(t)^2*L(t))*Ksi*Ki*Kp*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)-Sigma2*S(t)^2+
Ff*sf*S(t))*Ksi*Ksl*Ki*K2*K1*Ke*K0+(-diff(S(t),t)*V(t)*S(t)*L(t)-
Sigma2*S(t)^2*L(t)+Ff*sf*S(t)*L(t))*Ksi*Ki*Kp*K1*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)^2*L(t)-Sigma2*S(t)^3*L(t)+
Ff*sf*S(t)^2*L(t))*Ksi*K2*K1*Ke*K0+(-diff(S(t),t)*V(t)*S(t)^2-
Sigma2*S(t)^3+Ff*sf*S(t)^2)*Ksi*Ksl*K2*K1*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksi*Ksl*K1*Ke*K0+
(-diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ksi*K1*Ke*K0+(-diff(S(t),t)*V(t)*S(t)^2*L(t)-
Sigma2*S(t)^3*L(t)+Ff*sf*S(t)^2*L(t))*Ksi*Ki*K1*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)-Sigma2*S(t)^2+
Ff*sf*S(t))*K0*Ke*K1*K2*Kp*Ki*Ksl+(-1/2*V(t)*S(t)^7*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^7*L(t))*m1+(-diff(S(t),t)*V(t)*S(t)*L(t)-
Sigma2*S(t)^2*L(t)+Ff*sf*S(t)*L(t))*Ksi*Ki*Kp*K2*K1*Ke+(-
diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksi*Ksl*K2*K1*K0+
(-diff(S(t),t)*V(t)*S(t)^2-Sigma2*S(t)^3+
Ff*sf*S(t)^2)*Ksi*Ksl*Ki*Kp*K1*K0+(-diff(S(t),t)*V(t)*S(t)^3*L(t)-
Sigma2*S(t)^4*L(t)+Ff*sf*S(t)^3*L(t))*Ksi*K2*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksi*Ksl*K1*K0+(-
diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+Ff*sf*S(t)^5)*Ksi*Ksl*K0+(-
```

```
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ksi*K2*K1*Ki*mup*alphap+(-
diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
Ff*sf*S(t)^4*L(t))*Ksi*Ke*K0+(-1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ksi*K2*Ki*mup*alphap+(-
1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksi*Ksl*K2*Ke*Ki*mup*alphap+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ksi*K2*Ke*Ki*mup*alphap+(-
1/2*V(t)*S(t)^2*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^2*L(t))*Ksi*K2*K1*Ke*Ki*mup*alphap+(-
1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksi*Ksl*K2*K1*Ki*mup*alphap+(-
diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
Ff*sf*S(t)^4*L(t))*Ke*K1*K2+(-1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ksi*K2*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^2*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^2)*Ksi*Ksl*K2*K1*Ke*Ki*mup*alphap+(-
1/2*V(t)*S(t)^2*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^2)*Ksi*Ksl*K2*K1*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^2*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^2*L(t))*Ksi*K2*K1*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^2*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^2)*Ksi*Ksl*K1*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^2*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^2*L(t))*Ksi*K1*Ke*K0*Ki*mup*alphap+(-
diff(S(t),t)*V(t)*S(t)^2-Sigma2*S(t)^3+
Ff*sf*S(t)^2)*Ke*K1*K2*Kp*Ki*Ksl+(-diff(S(t),t)*V(t)*S(t)-
Sigma2*S(t)^2+Ff*sf*S(t))*Ke*K1*K2*Kp*Ki*Ksl*Ksi+(-
diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksi*Ksl*K2*K1*Ke+
```

```
(-diff(S(t),t)*V(t)*S(t)^2*L(t)-Sigma2*S(t)^3*L(t)+
Ff*sf*S(t)^2*L(t))*Ksi*Ki*Kp*K1*K0+(-diff(S(t),t)*V(t)*S(t)^4*L(t)-
Sigma2*S(t)^5*L(t)+Ff*sf*S(t)^4*L(t))*Ksi*K1*Ke+(-
diff(S(t),t)*V(t)*S(t)^7-Sigma2*S(t)^8+Ff*sf*S(t)^7)*Ksl+(-
diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ksi*K2*K1*K0+(-1/2*V(t)*S(t)^4*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^4)*Ksi*Ksl*K1*Ki*mup*alphap+(-
1/2*V(t)*S(t)*X1(t)+
1/2*X2(t)*rho*V(t)*S(t))*Ksi*Ksl*K2*K1*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^2*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^2)*Ksl*K2*K1*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)*L(t))*Ksi*K2*K1*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ksi*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksi*Ksl*K1*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ksi*K1*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^4*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^4)*Ksi*Ksl*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksi*Ksl*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ksi*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^5*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*K2*Ki*mup*alphap+(-
1/2*V(t)*S(t)^5*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^5)*Ksi*Ksl*Ki*mup*alphap+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
```

```
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ksi*K1*Ke*Ki*mup*alphap+(-
diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
Ff*sf*S(t)^4*L(t))*K0*Ke*K1+(-diff(S(t),t)*V(t)*S(t)^4*L(t)-
Sigma2*S(t)^5*L(t)+Ff*sf*S(t)^4*L(t))*Ki*Kp*K1+(-
diff(S(t),t)*V(t)*S(t)*L(t)-Sigma2*S(t)^2*L(t)+
Ff*sf*S(t)*L(t))*K0*Ke*K1*K2*Kp*Ki+(-diff(S(t),t)*V(t)*S(t)^3*L(t)-
Sigma2*S(t)^4*L(t)+Ff*sf*S(t)^3*L(t))*K0*Ke*K1*K2+(-
diff(S(t),t)*V(t)*S(t)^6*L(t)-Sigma2*S(t)^7*L(t)+
Ff*sf*S(t)^6*L(t))*K2+(-diff(S(t),t)*V(t)*S(t)^6*L(t)-
Sigma2*S(t)^7*L(t)+Ff*sf*S(t)^6*L(t))*K1+(-
diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
Ff*sf*S(t)^4*L(t))*K0*K1*K2+(-diff(S(t),t)*V(t)*S(t)^6*L(t)-
Sigma2*S(t)^7*L(t)+Ff*sf*S(t)^6*L(t))*K0+(-1/2*V(t)*S(t)^5*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*K1*Ki*mup*alphap+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*K1*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ki*Kp*Ke*m1+(-1/2*V(t)*S(t)^5*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^5)*Ksl*K2*Ki*mup*alphap+(-
1/2*V(t)*S(t)^4*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^4)*Ksl*K2*K1*Ki*mup*alphap+(-
1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksl*K2*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^5*X1(t)*L(t)+1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*K1*Ke*m1+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*K2*K1*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksl*K2*K1*Ke*Ki*mup*alphap+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*K2*K1*Ke*Ki*mup*alphap+(-
```

```
1/2*V(t)*S(t)^4*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^4)*Ksl*K2*Ke*Ki*mup*alphap+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*K2*Ke*Ki*mup*alphap+(-
1/2*V(t)*S(t)^2*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^2*L(t))*K2*K1*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^2*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^2)*Ksi*Ksl*K2*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^2*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^2*L(t))*Ksi*K2*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ksi*K1*Ki*mup*alphap+(-
1/2*V(t)*S(t)^4*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^4)*Ksi*Ksl*Ke*Ki*mup*alphap+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ksi*Ke*Ki*mup*alphap+(-
1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksi*Ksl*K1*Ke*Ki*mup*alphap+
(V(t)*X0(t)*S(t)^2*L(t)+V(t)*S(t)^2*X1(t)*L(t))*Ki*K2*Ke*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^3*L(t)+V(t)*S(t)^3*X1(t)*L(t))*K2*Ke*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^3*L(t)+V(t)*S(t)^3*X1(t)*L(t))*K1*Ke*K0*Ksi*mul+
(V(t)*X0(t)*S(t)*L(t)+V(t)*S(t)*X1(t)*L(t))*Ki*Kp*K2*Ke*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^6*L(t)+V(t)*S(t)^6*X1(t)*L(t))*Ksi*mul+
(V(t)*X0(t)*S(t)*L(t)+V(t)*S(t)*X1(t)*L(t))*Ki*K2*K1*Ke*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^2*L(t)+V(t)*S(t)^2*X1(t)*L(t))*K2*K1*Ke*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^4*L(t)+V(t)*S(t)^4*X1(t)*L(t))*K1*Ke*Ksi*mul+
(V(t)*X0(t)*S(t)^3*L(t)+V(t)*S(t)^3*X1(t)*L(t))*Ki*K1*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^4*L(t)+V(t)*S(t)^4*X1(t)*L(t))*K1*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^2*L(t)+V(t)*S(t)^2*X1(t)*L(t))*Ki*Kp*K1*Ke*Ksi*mul+
(V(t)*X0(t)*S(t)^4*L(t)+V(t)*S(t)^4*X1(t)*L(t))*Ki*K2*Ksi*mul+
```

```
(V(t)*X0(t)*S(t)^2*L(t)+V(t)*S(t)^2*X1(t)*L(t))*Ki*Kp*Ke*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^3*L(t)+V(t)*S(t)^3*X1(t)*L(t))*Ki*Ke*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^4*L(t)+V(t)*S(t)^4*X1(t)*L(t))*Ke*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^3*L(t)+V(t)*S(t)^3*X1(t)*L(t))*Ki*Kp*Ke*Ksi*mul+
(V(t)*X0(t)*S(t)^4*L(t)+V(t)*S(t)^4*X1(t)*L(t))*Ki*Ke*Ksi*mul+
(V(t)*X0(t)*S(t)^3*L(t)+V(t)*S(t)^3*X1(t)*L(t))*Ki*Kp*K1*Ksi*mul+(-
1/2*V(t)*S(t)^4*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^4)*Ksl*K2*K0*Ki*mup*alphap+
(V(t)*X0(t)*S(t)^4*L(t)+V(t)*S(t)^4*X1(t)*L(t))*Ki*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^4*L(t)+V(t)*S(t)^4*X1(t)*L(t))*Ki*K1*Ksi*mul+
(V(t)*X0(t)*S(t)^3*L(t)+V(t)*S(t)^3*X1(t)*L(t))*Ki*Kp*K2*Ksi*mul+
(V(t)*X0(t)*S(t)^2*L(t)+V(t)*S(t)^2*X1(t)*L(t))*Ki*Kp*K1*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^3*L(t)+V(t)*S(t)^3*X1(t)*L(t))*Ki*K1*Ke*Ksi*mul+
(V(t)*X0(t)*L(t)+V(t)*X1(t)*L(t))*Ki*Kp*K2*K1*Ke*K0*Ksi*mul+
(V(t)*X0(t)*S(t)*L(t)+V(t)*S(t)*X1(t)*L(t))*Ki*Kp*K1*Ke*K0*Ksi*mul+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ki*K1*Ke*m1+(-1/2*V(t)*S(t)^5*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^5)*Ksl*K1*Ke*m1+(V(t)*X0(t)*S(t)^3*L(t)+
V(t)*S(t)^3*X1(t)*L(t))*Ki*K2*Ke*Ksi*mul+(V(t)*X0(t)*S(t)^2*L(t)+
V(t)*S(t)^2*X1(t)*L(t))*Ki*Kp*K2*K1*Ksi*mul+(-
1/2*V(t)*S(t)^5*X1(t)*L(t)+1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*K1*K0*m1+
(V(t)*X0(t)*S(t)^3*L(t)+V(t)*S(t)^3*X1(t)*L(t))*Ki*Kp*K0*Ksi*mul+(-
diff(S(t),t)*V(t)*S(t)^2-Sigma2*S(t)^3+
Ff*sf*S(t)^2)*Ksi*Ksl*Ki*K1*Ke*K0+(-1/2*V(t)*S(t)^5*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^5)*Ksl*Ki*Kp*m1+(-1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksl*Ki*K1*Ke*K0*m1+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ki*K1*Ke*K0*m1+(-
1/2*V(t)*S(t)^4*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^4)*Ksl*K1*Ke*K0*m1+(-
1/2*V(t)*S(t)^5*X1(t)*L(t)+1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*Ki*K1*m1+(-
```

```
diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ksi*Ki*Kp*K2+(-1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*K1*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^4*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^4)*Ksl*K1*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^5*X1(t)*L(t)+1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*Ki*Kp*m1+(-
1/2*V(t)*S(t)^6*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^6)*Ksl*Ki*m1+(-
1/2*V(t)*S(t)^5*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^5)*Ksl*Ki*K1*m1+
(V(t)*X0(t)*S(t)^3*L(t)+V(t)*S(t)^3*X1(t)*L(t))*Ki*K2*K1*Ksi*mul+
(V(t)*X0(t)*S(t)^4*L(t)+V(t)*S(t)^4*X1(t)*L(t))*K2*K1*Ksi*mul+(-
1/2*V(t)*S(t)^5*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^5)*Ksl*K1*Ki*mup*alphap+
(-1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksl*K1*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^4*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^4)*Ksl*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ke*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^6*X1(t)*L(t)+1/2*X2(t)*rho*V(t)*S(t)^6*L(t))*Ki*m1+(-
1/2*V(t)*S(t)^5*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*K0*Ki*mup*alphap+(-
1/2*V(t)*S(t)^4*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^4)*Ksl*K1*Ke*Ki*mup*alphap+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*K1*Ke*Ki*mup*alphap+(-
1/2*V(t)*S(t)^5*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^5)*Ksl*K0*Ki*mup*alphap+
(-1/2*V(t)*S(t)^6*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^6)*Ksl*Ki*mup*alphap+
(-1/2*V(t)*S(t)^5*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*Ksi*Ki*mup*alphap+(-
1/2*V(t)*S(t)^6*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^6*L(t))*Ki*mup*alphap+(-1/2*V(t)*S(t)^5*X1(t)+
```

```
1/2*X2(t)*rho*V(t)*S(t)^5)*Ksl*Ke*Ki*mup*alphap+(-
1/2*V(t)*S(t)^5*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*Ke*Ki*mup*alphap+(-
1/2*V(t)*S(t)^5*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^5)*Ksl*K1*K0*m1+(-
1/2*V(t)*S(t)^4*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^4)*Ksi*Ksl*K2*Ki*mup*alphap+(-
1/2*V(t)*S(t)^4*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^4)*Ksl*Ki*K1*Ke*m1+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ki*Kp*K1*Ke*m1+(-
1/2*V(t)*S(t)^6*X1(t)*L(t)+1/2*X2(t)*rho*V(t)*S(t)^6*L(t))*Ke*m1+(-
diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+Ff*sf*S(t)^5)*Ksl*Ki*Kp+(-
diff(S(t),t)*V(t)*S(t)^6-Sigma2*S(t)^7+Ff*sf*S(t)^6)*Ksi*Ksl+(-
1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksi*Ksl*K2*K0*Ki*mup*alphap+(-
diff(S(t),t)*V(t)*S(t)^5*L(t)-Sigma2*S(t)^6*L(t)+
Ff*sf*S(t)^5*L(t))*Ki*K2+(-diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+
Ff*sf*S(t)^5)*Ksl*Ki*K2+(-diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+
Ff*sf*S(t)^3)*Ksl*Ki*K2*K1*K0+(-diff(S(t),t)*V(t)*S(t)^4*L(t)-
Sigma2*S(t)^5*L(t)+Ff*sf*S(t)^4*L(t))*Ki*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksl*Ki*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksl*Ki*K2*Ke+(-
diff(S(t),t)*V(t)*S(t)^2*L(t)-Sigma2*S(t)^3*L(t)+
Ff*sf*S(t)^2*L(t))*Ki*Kp*K1*Ke*K0+(-diff(S(t),t)*V(t)*S(t)^3-
Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksl*K2*K1*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksl*K1*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)*L(t)-Sigma2*S(t)^2*L(t)+
Ff*sf*S(t)*L(t))*Ksi*Ki*Kp*K2*K1*K0+(-diff(S(t),t)*V(t)*S(t)^5*L(t)-
Sigma2*S(t)^6*L(t)+Ff*sf*S(t)^5*L(t))*Ksi*K0+(-
diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ksi*K2*K1*Ke+(-diff(S(t),t)*V(t)*S(t)^5*L(t)-
```

```
Sigma2*S(t)^6*L(t)+Ff*sf*S(t)^5*L(t))*Ksi*Ke+(-
diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+Ff*sf*S(t)^5)*Ksi*Ksl*Ke+(-
diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ki*K2*K1*Ke+(-diff(S(t),t)*V(t)*S(t)^3-
Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksl*Ki*K2*K1*Ke+(-
diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+Ff*sf*S(t)^5)*Ksl*Ki*Ke+(-
diff(S(t),t)*V(t)*S(t)^5*L(t)-Sigma2*S(t)^6*L(t)+
Ff*sf*S(t)^5*L(t))*K2*Ke+(-diff(S(t),t)*V(t)*S(t)^4*L(t)-
Sigma2*S(t)^5*L(t)+Ff*sf*S(t)^4*L(t))*Ki*K2*Ke+(-
diff(S(t),t)*V(t)*S(t)^6*L(t)-Sigma2*S(t)^7*L(t)+
Ff*sf*S(t)^6*L(t))*Ke+(-diff(S(t),t)*V(t)*S(t)^5*L(t)-
Sigma2*S(t)^6*L(t)+Ff*sf*S(t)^5*L(t))*Ki*K1+(-
diff(S(t),t)*V(t)*S(t)^5*L(t)-Sigma2*S(t)^6*L(t)+
Ff*sf*S(t)^5*L(t))*K2*K1+(-diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+
Ff*sf*S(t)^5)*Ksl*Ki*K1+(-diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+
Ff*sf*S(t)^5)*Ksl*K2*K1+(-diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+
Ff*sf*S(t)^4)*Ksi*Ksl*Ki*K0+(-diff(S(t),t)*V(t)*S(t)^4*L(t)-
Sigma2*S(t)^5*L(t)+Ff*sf*S(t)^4*L(t))*Ksi*Ki*Kp+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksi*Ksl*Ki*Kp+(-
diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
Ff*sf*S(t)^4*L(t))*Ksi*Ki*K2+(-1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ki*K1*K0*m1+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksi*Ksl*K2*Ke+(-
diff(S(t),t)*V(t)*S(t)^2-Sigma2*S(t)^3+
Ff*sf*S(t)^2)*Ksi*Ksl*Ki*Kp*Ke*K0+(-diff(S(t),t)*V(t)*S(t)^6-
Sigma2*S(t)^7+Ff*sf*S(t)^6)*Ksl*Ki+(-diff(S(t),t)*V(t)*S(t)^5*L(t)-
Sigma2*S(t)^6*L(t)+Ff*sf*S(t)^5*L(t))*K2*K0+(-
diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+Ff*sf*S(t)^5)*Ksl*K2*K0+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksl*K2*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+Ff*sf*S(t)^5)*Ksl*Ke*K0+(-
```

```
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksl*Ki*Kp*K1+(-
diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
Ff*sf*S(t)^4*L(t))*Ki*K2*K1+(-diff(S(t),t)*V(t)*S(t)^3*L(t)-
Sigma2*S(t)^4*L(t)+Ff*sf*S(t)^3*L(t))*Ki*Kp*K2*K1+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksl*Ki*K2*K1+(-
diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
Ff*sf*S(t)^4*L(t))*Ki*Kp*K0+(-diff(S(t),t)*V(t)*S(t)^4*L(t)-
Sigma2*S(t)^5*L(t)+Ff*sf*S(t)^4*L(t))*Ki*K1*K0+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksl*Ki*K1*K0+(-
diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
Ff*sf*S(t)^4*L(t))*Ki*Kp*Ke+(-diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+
Ff*sf*S(t)^4)*Ksl*Ki*Kp*K2+(-diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+
Ff*sf*S(t)^4)*Ksl*Ki*Kp*Ke+(-diff(S(t),t)*V(t)*S(t)^4*L(t)-
Sigma2*S(t)^5*L(t)+Ff*sf*S(t)^4*L(t))*Ki*Kp*K2+(-
diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksl*Ki*Kp*K2*K1+
(-diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ki*Kp*K1*K0+(-diff(S(t),t)*V(t)*S(t)^3-
Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksl*Ki*Kp*K2*K0+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksl*Ki*Kp*K0+(-
diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
Ff*sf*S(t)^4*L(t))*Ki*K2*K0+(-diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+
Ff*sf*S(t)^4)*Ksl*Ki*K2*K0+(-diff(S(t),t)*V(t)*S(t)^3*L(t)-
Sigma2*S(t)^4*L(t)+Ff*sf*S(t)^3*L(t))*Ki*Kp*K2*K0+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksl*K2*K1*Ke+(-
diff(S(t),t)*V(t)*S(t)^5*L(t)-Sigma2*S(t)^6*L(t)+
Ff*sf*S(t)^5*L(t))*Ki*K0+(-diff(S(t),t)*V(t)*S(t)^3*L(t)-
Sigma2*S(t)^4*L(t)+Ff*sf*S(t)^3*L(t))*Ki*Kp*K2*Ke+(-
diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+Ff*sf*S(t)^5)*Ksl*K2*Ke+(-
diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksl*Ki*Kp*K2*Ke+
(-diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
```

```
Ff*sf*S(t)^4*L(t))*Ki*K1*Ke+(-diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+
Ff*sf*S(t)^4)*Ksl*Ki*K1*Ke+(-diff(S(t),t)*V(t)*S(t)^3*L(t)-
Sigma2*S(t)^4*L(t)+Ff*sf*S(t)^3*L(t))*Ki*Kp*K1*Ke+(-
diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ksi*Ki*K2*Ke+(-diff(S(t),t)*V(t)*S(t)^5-
Sigma2*S(t)^6+Ff*sf*S(t)^5)*Ksl*K1*Ke+(-diff(S(t),t)*V(t)*S(t)^3*L(t)-
Sigma2*S(t)^4*L(t)+Ff*sf*S(t)^3*L(t))*Ki*K2*K1*K0+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksl*K2*K1*K0+(-
diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksl*Ki*Kp*K1*K0+
(-diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+
Ff*sf*S(t)^3)*Ksl*Ki*Kp*Ke*K0+(-diff(S(t),t)*V(t)*S(t)^6-
Sigma2*S(t)^7+Ff*sf*S(t)^6)*Ksl*K2+(-diff(S(t),t)*V(t)*S(t)^3*L(t)-
Sigma2*S(t)^4*L(t)+Ff*sf*S(t)^3*L(t))*Ki*Kp*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)^6-Sigma2*S(t)^7+Ff*sf*S(t)^6)*Ksl*K0+(-
diff(S(t),t)*V(t)*S(t)^6-Sigma2*S(t)^7+Ff*sf*S(t)^6)*Ksl*K1+(-
diff(S(t),t)*V(t)*S(t)^6-Sigma2*S(t)^7+Ff*sf*S(t)^6)*Ksl*Ke+(-
diff(S(t),t)*V(t)*S(t)^2*L(t)-Sigma2*S(t)^3*L(t)+
Ff*sf*S(t)^2*L(t))*Ksi*Ki*K2*K1*Ke+(-diff(S(t),t)*V(t)*S(t)^2-
Sigma2*S(t)^3+Ff*sf*S(t)^2)*Ksi*Ksl*Ki*K2*K1*Ke+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksi*Ksl*Ki*Ke+(-
diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
Ff*sf*S(t)^4*L(t))*Ksi*K2*Ke+(-diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+
Ff*sf*S(t)^5)*Ksi*Ksl*K1+(-diff(S(t),t)*V(t)*S(t)^5*L(t)-
Sigma2*S(t)^6*L(t)+Ff*sf*S(t)^5*L(t))*Ksi*K1+(-
diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
Ff*sf*S(t)^4*L(t))*Ksi*Ki*K1+(-diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+
Ff*sf*S(t)^4)*Ksi*Ksl*Ki*K1+(-diff(S(t),t)*V(t)*S(t)^4*L(t)-
Sigma2*S(t)^5*L(t)+Ff*sf*S(t)^4*L(t))*Ksi*K2*K1+(-
diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ki*K2*Ke*K0+(-diff(S(t),t)*V(t)*S(t)^3-
```

```
Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksl*Ki*K2*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+Ff*sf*S(t)^5)*Ksl*K1*K0+(-
diff(S(t),t)*V(t)*S(t)^2-Sigma2*S(t)^3+
Ff*sf*S(t)^2)*Ksl*Ki*Kp*K1*Ke*K0+(-diff(S(t),t)*V(t)*S(t)^2*L(t)-
Sigma2*S(t)^3*L(t)+Ff*sf*S(t)^2*L(t))*Ki*K2*K1*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)^5*L(t)-Sigma2*S(t)^6*L(t)+
Ff*sf*S(t)^5*L(t))*Ki*Ke+(-diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+
Ff*sf*S(t)^3)*Ksl*Ki*K1*Ke*K0+(-diff(S(t),t)*V(t)*S(t)^2-
Sigma2*S(t)^3+Ff*sf*S(t)^2)*Ksl*Ki*K2*K1*Ke*K0+(-
1/2*V(t)*S(t)^3*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^3)*Ksl*Ki*Kp*K1*Ke*m1+(-
1/2*V(t)*S(t)^3*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^3)*Ksl*Ki*Kp*Ke*K0*m1+(-
1/2*V(t)*S(t)^6*X1(t)*L(t)+1/2*X2(t)*rho*V(t)*S(t)^6*L(t))*K0*m1+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksi*Ksl*K1*Ke+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*K1*Ke*K0*m1+(-1/2*V(t)*S(t)^4*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^4)*Ksl*Ki*K1*K0*m1+(-1/2*V(t)*S(t)^6*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^6)*Ksl*K1*m1+(-1/2*V(t)*S(t)^6*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^6)*Ksl*K0*m1+(-1/2*V(t)*S(t)^5*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^5)*Ksl*Ki*Ke*m1+(-1/2*V(t)*S(t)^6*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^6*L(t))*K1*m1+(-1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ki*Kp*K1*m1+(-1/2*V(t)*S(t)^4*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^4)*Ksl*Ki*Kp*K1*m1+(-1/2*V(t)*S(t)^6*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^6)*Ksl*Ke*m1+(-1/2*V(t)*S(t)^5*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*Ksi*Ki*m1+(-1/2*V(t)*S(t)^6*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^6)*Ksi*Ksl*m1+(-1/2*V(t)*S(t)^5*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*Ki*K0*m1+(-
diff(S(t),t)*V(t)*S(t)^4*L(t)-Sigma2*S(t)^5*L(t)+
Ff*sf*S(t)^4*L(t))*Ksi*K2*K0+(-diff(S(t),t)*V(t)*S(t)^4*L(t)-
Sigma2*S(t)^5*L(t)+Ff*sf*S(t)^4*L(t))*Ksi*K1*K0+(-
diff(S(t),t)*V(t)*S(t)^4-Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksi*Ksl*K2*K0+(-
```

```
diff(S(t),t)*V(t)*L(t)-Sigma2*S(t)*L(t)+
Ff*sf*L(t))*Ksi*Ki*Kp*K2*K1*Ke*K0+(-diff(S(t),t)*V(t)*S(t)^2*L(t)-
Sigma2*S(t)^3*L(t)+Ff*sf*S(t)^2*L(t))*Ksi*Ki*K2*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksi*Ksl*K2*Ke*K0+
(-diff(S(t),t)*V(t)*S(t)^2-Sigma2*S(t)^3+
Ff*sf*S(t)^2)*Ksi*Ksl*Ki*K2*Ke*K0+(-diff(S(t),t)*V(t)*S(t)*L(t)-
Sigma2*S(t)^2*L(t)+Ff*sf*S(t)*L(t))*Ksi*Ki*Kp*K2*Ke*K0+(-
diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksi*Ksl*Ki*Ke*K0+
(-diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ksi*Ki*Kp*K0+(-diff(S(t),t)*V(t)*S(t)^2-
Sigma2*S(t)^3+Ff*sf*S(t)^2)*Ksi*Ksl*Ki*Kp*K2*K0+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ki*Kp*K0*m1+(-1/2*V(t)*S(t)^5*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^5)*Ksl*Ki*K0*m1+(-1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksi*Ksl*Ki*K1*K0*m1+(-
1/2*V(t)*S(t)^2*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^2*L(t))*Ksi*Ki*Kp*K1*K0*m1+(-
1/2*V(t)*S(t)^2*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^2)*Ksi*Ksl*Ki*Kp*K1*K0*m1+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ksi*Ki*Ke*K0*m1+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ksi*Ke*K0*m1+(-1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksi*Ksl*Ki*Ke*K0*m1+(-
1/2*V(t)*S(t)^2*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^2*L(t))*Ksi*Ki*Kp*Ke*K0*m1+(-
1/2*V(t)*S(t)^2*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^2)*Ksi*Ksl*Ki*Kp*Ke*K0*m1+(-
1/2*V(t)*S(t)^4*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^4)*Ksi*Ksl*K1*Ke*m1+(-
1/2*V(t)*S(t)^4*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^4)*Ksl*Ki*Kp*K0*m1+(-
```

```
1/2*V(t)*S(t)^3*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^3)*Ksi*Ksl*Ki*K1*Ke*m1+
(-1/2*V(t)*S(t)^2*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^2*L(t))*Ksi*Ki*Kp*K1*Ke*m1+(-
1/2*V(t)*S(t)^5*X1(t)*L(t)+1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*Ksi*Ke*m1+
(-1/2*V(t)*S(t)^5*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^5)*Ksi*Ksl*Ke*m1+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ksi*Ki*Ke*m1+(-
1/2*V(t)*S(t)^6*X1(t)*L(t)+1/2*X2(t)*rho*V(t)*S(t)^6*L(t))*Ksi*m1+(-
1/2*V(t)*S(t)^2*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^2)*Ksl*Ki*Kp*K1*Ke*K0*m1+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ksi*Ki*Kp*Ke*m1+(-
1/2*V(t)*S(t)^3*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^3)*Ksi*Ksl*K1*Ke*K0*m1+
(-1/2*V(t)*S(t)*X1(t)+
1/2*X2(t)*rho*V(t)*S(t))*Ksi*Ksl*Ki*Kp*K1*Ke*K0*m1+(-
1/2*V(t)*S(t)^2*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^2*L(t))*Ksi*Ki*K1*Ke*K0*m1+(-
1/2*V(t)*S(t)^2*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^2*L(t))*Ki*Kp*K1*Ke*K0*m1+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ksi*Ki*Kp*m1+(-1/2*V(t)*S(t)^5*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^5)*Ksi*Ksl*Ki*m1+(-1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ksi*Ki*K1*Ke*m1+(-
1/2*V(t)*S(t)^5*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^5)*Ksi*Ksl*K1*m1+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ksi*Ki*K1*m1+(-1/2*V(t)*S(t)^4*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^4)*Ksi*Ksl*Ki*K1*m1+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ksi*Ki*Kp*K1*m1+(-
1/2*V(t)*S(t)^3*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^3)*Ksi*Ksl*Ki*Kp*K1*m1+
```

```
(-1/2*V(t)*S(t)^3*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^3)*Ksi*Ksl*Ki*Kp*Ke*m1+(-
1/2*V(t)*S(t)^5*X1(t)*L(t)+1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*Ksi*K1*m1+
(-1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ksi*K1*Ke*m1+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ksi*K1*K0*m1+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ksi*K1*Ke*K0*m1+(-
1/2*V(t)*S(t)^4*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^4)*Ksi*Ksl*K1*K0*m1+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ksi*Ki*K1*K0*m1+(-
1/2*V(t)*S(t)^4*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^4)*Ksi*Ksl*Ki*Kp*m1+(-
1/2*V(t)*S(t)^4*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^4)*Ksi*Ksl*Ki*K0*m1+(-
1/2*V(t)*S(t)^3*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^3)*Ksi*Ksl*Ki*Kp*K0*m1+
(-1/2*V(t)*S(t)^4*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^4)*Ksi*Ksl*Ki*Ke*m1+(-
1/2*V(t)*S(t)^3*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^3*L(t))*Ksi*Ki*Kp*K0*m1+(-
1/2*V(t)*S(t)^5*X1(t)*L(t)+1/2*X2(t)*rho*V(t)*S(t)^5*L(t))*Ksi*K0*m1+
(-1/2*V(t)*S(t)^5*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^5)*Ksi*Ksl*K0*m1+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*Ksi*Ki*K0*m1+(-1/2*V(t)*S(t)^2*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^2)*Ksi*Ksl*Ki*Kp*K1*Ke*m1+(-
1/2*V(t)*S(t)^2*X1(t)+
1/2*X2(t)*rho*V(t)*S(t)^2)*Ksi*Ksl*Ki*K1*Ke*K0*m1+(-
1/2*V(t)*S(t)^4*X1(t)+1/2*X2(t)*rho*V(t)*S(t)^4)*Ksi*Ksl*Ke*K0*m1+(-
1/2*V(t)*S(t)*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)*L(t))*Ksi*Ki*Kp*K1*Ke*K0*m1+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*K2*K1*Ki*mup*alphap+
```

```
(V(t)*X0(t)*S(t)^5*L(t)+V(t)*S(t)^5*X1(t)*L(t))*K2*Ksi*mul+(-
1/2*V(t)*S(t)^4*X1(t)*L(t)+
1/2*X2(t)*rho*V(t)*S(t)^4*L(t))*K2*K0*Ki*mup*alphap+
(V(t)*X0(t)*S(t)^5*L(t)+V(t)*S(t)^5*X1(t)*L(t))*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^5*L(t)+V(t)*S(t)^5*X1(t)*L(t))*Ke*Ksi*mul+
(V(t)*X0(t)*S(t)^5*L(t)+V(t)*S(t)^5*X1(t)*L(t))*K1*Ksi*mul+
(V(t)*X0(t)*S(t)^5*L(t)+V(t)*S(t)^5*X1(t)*L(t))*Ki*Ksi*mul+
(V(t)*X0(t)*S(t)*L(t)+V(t)*S(t)*X1(t)*L(t))*Ki*Kp*K2*K1*K0*Ksi*mul+(-
diff(S(t),t)*V(t)*S(t)^3-Sigma2*S(t)^4+Ff*sf*S(t)^3)*Ksi*Ksl*Ki*K2*Ke+
(-diff(S(t),t)*V(t)*S(t)^3*L(t)-Sigma2*S(t)^4*L(t)+
Ff*sf*S(t)^3*L(t))*Ki*K1*Ke*K0+(-diff(S(t),t)*V(t)*S(t)^4-
Sigma2*S(t)^5+Ff*sf*S(t)^4)*Ksi*Ksl*K2*K1+(-
diff(S(t),t)*V(t)*S(t)^5*L(t)-Sigma2*S(t)^6*L(t)+
Ff*sf*S(t)^5*L(t))*Ksi*K2+(-diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+
Ff*sf*S(t)^5)*Ksi*Ksl*Ki+(-diff(S(t),t)*V(t)*S(t)^4*L(t)-
Sigma2*S(t)^5*L(t)+Ff*sf*S(t)^4*L(t))*Ksi*Ki*Ke+(-
diff(S(t),t)*V(t)*S(t)^5-Sigma2*S(t)^6+Ff*sf*S(t)^5)*Ksi*Ksl*K2+(-
diff(S(t),t)*V(t)*S(t)-Sigma2*S(t)^2+
Ff*sf*S(t))*Ksi*Ksl*Ki*Kp*K1*Ke*K0+(-diff(S(t),t)*V(t)*S(t)*L(t)-
Sigma2*S(t)^2*L(t)+Ff*sf*S(t)*L(t))*Ksi*Ki*K2*K1*Ke*K0+
(V(t)*X0(t)*S(t)^2*L(t)+V(t)*S(t)^2*X1(t)*L(t))*Ki*K2*K1*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^3*L(t)+V(t)*S(t)^3*X1(t)*L(t))*K2*K1*K0*Ksi*mul+
(V(t)*X0(t)*S(t)^2*L(t)+V(t)*S(t)^2*X1(t)*L(t))*Ki*Kp*K2*K0*Ksi*mul+(-
diff(S(t),t)*V(t)*S(t)^5*L(t)-Sigma2*S(t)^6*L(t)+
Ff*sf*S(t)^5*L(t))*Ksi*Ki+(V(t)*X0(t)*S(t)*L(t)+
V(t)*S(t)*X1(t)*L(t))*Ki*Kp*K2*K1*Ke*Ksi*mul+(V(t)*X0(t)*S(t)^4*L(t)+
V(t)*S(t)^4*X1(t)*L(t))*K2*K0*Ksi*mul+(V(t)*X0(t)*S(t)^2*L(t)+
V(t)*S(t)^2*X1(t)*L(t))*Ki*K2*K1*Ke*Ksi*mul+(V(t)*X0(t)*S(t)^3*L(t)+
V(t)*S(t)^3*X1(t)*L(t))*K2*K1*Ke*Ksi*mul+(V(t)*X0(t)*S(t)^3*L(t)+
V(t)*S(t)^3*X1(t)*L(t))*Ki*K2*K0*Ksi*mul+(V(t)*X0(t)*S(t)^2*L(t)+
```

```
V(t)*S(t)^2*X1(t)*L(t))*Ki*Kp*K2*Ke*Ksi*mul
```

C.3 Output from the Perl program

The corresponding LaTeX output from the Perl program was used to produce the following.

```
Sexpr :=
-1                                  L(t)*S(t)^7*V(t)*diff(S(t),t)
-1                                  L(t)*S(t)^8*Sigma2
+m1                                 -1/2*V(t)*S(t)^7*X1(t)*L(t)+
                                    1/2*X2(t)*rho*V(t)*S(t)^7*L(t)
+Ksl                                -diff(S(t),t)*V(t)*S(t)^7-
                                    Sigma2*S(t)^8+Ff*sf*S(t)^7
+Ff*sf                              L(t)*S(t)^7
+Ksl*m1                             -1/2*V(t)*S(t)^7*X1(t)+
                                    1/2*X2(t)*rho*V(t)*S(t)^7
+Ksi*mul                            V(t)*X0(t)*S(t)^6*L(t)+
                                    V(t)*S(t)^6*X1(t)*L(t)
-alpha0*mu0                         L(t)*S(t)^7*V(t)*X1(t)
-alphae*mue-m0                      L(t)*S(t)^7*V(t)*X0(t)
-Ksl*alpha0*mu0                     S(t)^7*V(t)*X1(t)
+K0+K1+K2+Ke+Ki+Ksi                 -diff(S(t),t)*V(t)*S(t)^6*L(t)-
                                    Sigma2*S(t)^7*L(t)+
                                    Ff*sf*S(t)^6*L(t)
+K0*K1*K2*Ke*Ki*Kp*Ksi              -diff(S(t),t)*V(t)*L(t)-
                                    Sigma2*S(t)*L(t)+Ff*sf*L(t)
-Ksl*alphae*mue-Ksl*m0              S(t)^7*V(t)*X0(t)
+K0*K1*K2*Ke*Ki*Kp*Ksi*Ksl          -diff(S(t),t)*V(t)-Sigma2*S(t)+
                                    Ff*sf
```

```
+K0*K1*K2*Ke*Ki*Kp*Ksi*mul              V(t)*X0(t)*L(t)+V(t)*X1(t)*L(t)
-K1*K2*Ke*Ki*Kp*Ksi*alpha0*mu0          L(t)*S(t)*V(t)*X1(t)
-K1*K2*Ke*Ki*Kp*Ksi*Ksl*alpha0*mu0      S(t)*V(t)*X1(t)
+K0*Ksl+K1*Ksl+K2*Ksl+Ke*Ksl+Ki*Ksl+    -diff(S(t),t)*V(t)*S(t)^6-
     Ksi*Ksl                            Sigma2*S(t)^7+Ff*sf*S(t)^6
+K0*m1+K1*m1+Ke*m1+Ki*alphap*mup+        -1/2*V(t)*S(t)^6*X1(t)*L(t)+
     Ki*m1+Ksi*m1                       1/2*X2(t)*rho*V(t)*S(t)^6*L(t)
+K0*K1*K2*Ke*Ki*Ksi*alphap*mup+         -1/2*V(t)*S(t)*X1(t)*L(t)+
     K0*K1*Ke*Ki*Kp*Ksi*m1              1/2*X2(t)*rho*V(t)*S(t)*L(t)
-K0*K1*K2*Ki*Kp*Ksi*alphae*mue-         L(t)*S(t)*V(t)*X0(t)
     K0*K2*Ke*Ki*Kp*Ksi*m0
+K0*Ksi*mul+K1*Ksi*mul+K2*Ksi*mul+      V(t)*X0(t)*S(t)^5*L(t)+
     Ke*Ksi*mul+Ki*Ksi*mul              V(t)*S(t)^5*X1(t)*L(t)
+K0*K1*K2*Ke*Ki*Ksi*Ksl*alphap*mup+     -1/2*V(t)*S(t)*X1(t)+
     K0*K1*Ke*Ki*Kp*Ksi*Ksl*m1          1/2*X2(t)*rho*V(t)*S(t)
-K0*K1*K2*Ki*Kp*Ksi*Ksl*alphae*mue-     S(t)*V(t)*X0(t)
     K0*K2*Ke*Ki*Kp*Ksi*Ksl*m0
+K0*Ksl*m1+K1*Ksl*m1+Ke*Ksl*m1+         -1/2*V(t)*S(t)^6*X1(t)+
     Ki*Ksl*alphap*mup+Ki*Ksl*m1+       1/2*X2(t)*rho*V(t)*S(t)^6
     Ksi*Ksl*m1
-K1*alpha0*mu0-K2*alpha0*mu0-           L(t)*S(t)^6*V(t)*X1(t)
     Ke*alpha0*mu0-Ki*alpha0*mu0-
     Ksi*alpha0*mu0
-K1*Ksl*alpha0*mu0-K2*Ksl*alpha0*mu0-   S(t)^6*V(t)*X1(t)
     Ke*Ksl*alpha0*mu0-
     Ki*Ksl*alpha0*mu0-
     Ksi*Ksl*alpha0*mu0
+K0*K1+K0*K2+K0*Ke+K0*Ki+K0*Ksi+         -diff(S(t),t)*V(t)*S(t)^5*L(t)-
     K1*K2+K1*Ke+K1*Ki+K1*Ksi+K2*Ke+    Sigma2*S(t)^6*L(t)+
     K2*Ki+K2*Ksi+Ke*Ki+Ke*Ksi+Ki*Kp+   Ff*sf*S(t)^5*L(t)
```

```
     Ki*Ksi
-K0*alphae*mue-K0*m0-K1*alphae*mue-    L(t)*S(t)^6*V(t)*X0(t)
     K2*alphae*mue-K2*m0-Ke*m0-
     Ki*alphae*mue-Ki*m0-
     Ksi*alphae*mue-Ksi*m0
+K0*K1*K2*Ke*Ki*Kp+                    -diff(S(t),t)*V(t)*S(t)*L(t)-
     K0*K1*K2*Ke*Ki*Ksi+               Sigma2*S(t)^2*L(t)+Ff*sf*S(t)*L(t)
     K0*K1*K2*Ki*Kp*Ksi+
     K0*K1*Ke*Ki*Kp*Ksi+
     K0*K2*Ke*Ki*Kp*Ksi+
     K1*K2*Ke*Ki*Kp*Ksi
+K0*K1*K2*Ke*Ki*Ksi*mul+               V(t)*X0(t)*S(t)*L(t)+
     K0*K1*K2*Ki*Kp*Ksi*mul+          V(t)*S(t)*X1(t)*L(t)
     K0*K1*Ke*Ki*Kp*Ksi*mul+
     K0*K2*Ke*Ki*Kp*Ksi*mul+
     K1*K2*Ke*Ki*Kp*Ksi*mul
-K1*K2*Ke*Ki*Kp*alpha0*mu0-            L(t)*S(t)^2*V(t)*X1(t)
     K1*K2*Ke*Ki*Ksi*alpha0*mu0-
     K1*K2*Ki*Kp*Ksi*alpha0*mu0-
     K1*Ke*Ki*Kp*Ksi*alpha0*mu0-
     K2*Ke*Ki*Kp*Ksi*alpha0*mu0
+K0*K1*K2*Ke*Ki*Kp*Ksl+                -diff(S(t),t)*V(t)*S(t)-
     K0*K1*K2*Ke*Ki*Ksi*Ksl+           Sigma2*S(t)^2+Ff*sf*S(t)
     K0*K1*K2*Ki*Kp*Ksi*Ksl+
     K0*K1*Ke*Ki*Kp*Ksi*Ksl+
     K0*K2*Ke*Ki*Kp*Ksi*Ksl+
     K1*K2*Ke*Ki*Kp*Ksi*Ksl
-K0*Ksl*alphae*mue-K0*Ksl*m0-          S(t)^6*V(t)*X0(t)
     K1*Ksl*alphae*mue-
     K2*Ksl*alphae*mue-K2*Ksl*m0-
```

```
     Ke*Ksl*m0-Ki*Ksl*alphae*mue-
     Ki*Ksl*m0-Ksi*Ksl*alphae*mue-
     Ksi*Ksl*m0
-K1*K2*Ke*Ki*Kp*Ksl*alpha0*mu0-        S(t)^2*V(t)*X1(t)
     K1*K2*Ke*Ki*Ksi*Ksl*alpha0*mu0-
     K1*K2*Ki*Kp*Ksi*Ksl*alpha0*mu0-
     K1*Ke*Ki*Kp*Ksi*Ksl*alpha0*mu0-
     K2*Ke*Ki*Kp*Ksi*Ksl*alpha0*mu0
+K0*K1*Ksi*mul+K0*K2*Ksi*mul+           V(t)*X0(t)*S(t)^4*L(t)+
     K0*Ke*Ksi*mul+K0*Ki*Ksi*mul+      V(t)*S(t)^4*X1(t)*L(t)
     K1*K2*Ksi*mul+K1*Ke*Ksi*mul+
     K1*Ki*Ksi*mul+K2*Ke*Ksi*mul+
     K2*Ki*Ksi*mul+Ke*Ki*Ksi*mul+
     Ki*Kp*Ksi*mul
+K0*K1*Ksl+K0*K2*Ksl+K0*Ke*Ksl+        -diff(S(t),t)*V(t)*S(t)^5-
     K0*Ki*Ksl+K0*Ksi*Ksl+K1*K2*Ksl+   Sigma2*S(t)^6+Ff*sf*S(t)^5
     K1*Ke*Ksl+K1*Ki*Ksl+K1*Ksi*Ksl+
     K2*Ke*Ksl+K2*Ki*Ksl+K2*Ksi*Ksl+
     Ke*Ki*Ksl+Ke*Ksi*Ksl+Ki*Kp*Ksl+
     Ki*Ksi*Ksl
+K0*K1*m1+K0*Ke*m1+K0*Ki*alphap*mup+   -1/2*V(t)*S(t)^5*X1(t)*L(t)+
     K0*Ki*m1+K0*Ksi*m1+K1*Ke*m1+      1/2*X2(t)*rho*V(t)*S(t)^5*L(t)
     K1*Ki*alphap*mup+K1*Ki*m1+
     K1*Ksi*m1+K2*Ki*alphap*mup+
     Ke*Ki*alphap*mup+Ke*Ki*m1+
     Ke*Ksi*m1+Ki*Kp*m1+
     Ki*Ksi*alphap*mup+Ki*Ksi*m1
-K1*K2*alpha0*mu0-K1*Ke*alpha0*mu0-    L(t)*S(t)^5*V(t)*X1(t)
     K1*Ki*alpha0*mu0-
     K1*Ksi*alpha0*mu0-
```

```
     K2*Ke*alpha0*mu0-
     K2*Ki*alpha0*mu0-
     K2*Ksi*alpha0*mu0-
     Ke*Ki*alpha0*mu0-
     Ke*Ksi*alpha0*mu0-
     Ki*Kp*alpha0*mu0-
     Ki*Ksi*alpha0*mu0
+K0*K1*K2*Ke*Ksi*mul+                V(t)*X0(t)*S(t)^2*L(t)+
     K0*K1*K2*Ki*Ksi*mul+            V(t)*S(t)^2*X1(t)*L(t)
     K0*K1*Ke*Ki*Ksi*mul+
     K0*K1*Ki*Kp*Ksi*mul+
     K0*K2*Ke*Ki*Ksi*mul+
     K0*K2*Ki*Kp*Ksi*mul+
     K0*Ke*Ki*Kp*Ksi*mul+
     K1*K2*Ke*Ki*Ksi*mul+
     K1*K2*Ki*Kp*Ksi*mul+
     K1*Ke*Ki*Kp*Ksi*mul+
     K2*Ke*Ki*Kp*Ksi*mul
+K0*K1*K2*Ke*Ki*alphap*mup+           -1/2*V(t)*S(t)^2*X1(t)*L(t)+
     K0*K1*K2*Ki*Ksi*alphap*mup+     1/2*X2(t)*rho*V(t)*S(t)^2*L(t)
     K0*K1*Ke*Ki*Kp*m1+
     K0*K1*Ke*Ki*Ksi*alphap*mup+
     K0*K1*Ke*Ki*Ksi*m1+
     K0*K1*Ki*Kp*Ksi*m1+
     K0*K2*Ke*Ki*Ksi*alphap*mup+
     K0*Ke*Ki*Kp*Ksi*m1+
     K1*K2*Ke*Ki*Ksi*alphap*mup+
     K1*Ke*Ki*Kp*Ksi*m1
-K0*K1*K2*Ki*Kp*alphae*mue-          L(t)*S(t)^2*V(t)*X0(t)
     K0*K1*K2*Ki*Ksi*alphae*mue-
```

```
     K0*K1*Ki*Kp*Ksi*alphae*mue-
     K0*K2*Ke*Ki*Kp*m0-
     K0*K2*Ke*Ki*Ksi*m0-
     K0*K2*Ki*Kp*Ksi*alphae*mue-
     K0*K2*Ki*Kp*Ksi*m0-
     K0*Ke*Ki*Kp*Ksi*m0-
     K1*K2*Ki*Kp*Ksi*alphae*mue-
     K2*Ke*Ki*Kp*Ksi*m0
-K1*K2*Ksl*alpha0*mu0-                S(t)^5*V(t)*X1(t)
     K1*Ke*Ksl*alpha0*mu0-
     K1*Ki*Ksl*alpha0*mu0-
     K1*Ksi*Ksl*alpha0*mu0-
     K2*Ke*Ksl*alpha0*mu0-
     K2*Ki*Ksl*alpha0*mu0-
     K2*Ksi*Ksl*alpha0*mu0-
     Ke*Ki*Ksl*alpha0*mu0-
     Ke*Ksi*Ksl*alpha0*mu0-
     Ki*Kp*Ksl*alpha0*mu0-
     Ki*Ksi*Ksl*alpha0*mu0
+K0*K1*K2+K0*K1*Ke+K0*K1*Ki+          -diff(S(t),t)*V(t)*S(t)^4*L(t)-
     K0*K1*Ksi+K0*K2*Ke+K0*K2*Ki+     Sigma2*S(t)^5*L(t)+
     K0*K2*Ksi+K0*Ke*Ki+K0*Ke*Ksi+    Ff*sf*S(t)^4*L(t)
     K0*Ki*Kp+K0*Ki*Ksi+K1*K2*Ke+
     K1*K2*Ki+K1*K2*Ksi+K1*Ke*Ki+
     K1*Ke*Ksi+K1*Ki*Kp+K1*Ki*Ksi+
     K2*Ke*Ki+K2*Ke*Ksi+K2*Ki*Kp+
     K2*Ki*Ksi+Ke*Ki*Kp+Ke*Ki*Ksi+
     Ki*Kp*Ksi
+K0*K1*K2*Ksi*mul+K0*K1*Ke*Ksi*mul+   V(t)*X0(t)*S(t)^3*L(t)+
     K0*K1*Ki*Ksi*mul+                V(t)*S(t)^3*X1(t)*L(t)
```

```
     K0*K2*Ke*Ksi*mul+
     K0*K2*Ki*Ksi*mul+
     K0*Ke*Ki*Ksi*mul+
     K0*Ki*Kp*Ksi*mul+
     K1*K2*Ke*Ksi*mul+
     K1*K2*Ki*Ksi*mul+
     K1*Ke*Ki*Ksi*mul+
     K1*Ki*Kp*Ksi*mul+
     K2*Ke*Ki*Ksi*mul+
     K2*Ki*Kp*Ksi*mul+
     Ke*Ki*Kp*Ksi*mul
+K0*K1*K2*Ke*Ki+K0*K1*K2*Ke*Ksi+       -diff(S(t),t)*V(t)*S(t)^2*L(t)-
     K0*K1*K2*Ki*Kp+K0*K1*K2*Ki*Ksi+   Sigma2*S(t)^3*L(t)+
     K0*K1*Ke*Ki*Kp+K0*K1*Ke*Ki*Ksi+   Ff*sf*S(t)^2*L(t)
     K0*K1*Ki*Kp*Ksi+K0*K2*Ke*Ki*Kp+
     K0*K2*Ke*Ki*Ksi+K0*K2*Ki*Kp*Ksi+
     K0*Ke*Ki*Kp*Ksi+K1*K2*Ke*Ki*Kp+
     K1*K2*Ke*Ki*Ksi+K1*K2*Ki*Kp*Ksi+
     K1*Ke*Ki*Kp*Ksi+K2*Ke*Ki*Kp*Ksi
+K0*K1*Ksl*m1+K0*Ke*Ksl*m1+            -1/2*V(t)*S(t)^5*X1(t)+
     K0*Ki*Ksl*alphap*mup+             1/2*X2(t)*rho*V(t)*S(t)^5
     K0*Ki*Ksl*m1+K0*Ksi*Ksl*m1+
     K1*Ke*Ksl*m1+
     K1*Ki*Ksl*alphap*mup+
     K1*Ki*Ksl*m1+K1*Ksi*Ksl*m1+
     K2*Ki*Ksl*alphap*mup+
     Ke*Ki*Ksl*alphap*mup+
     Ke*Ki*Ksl*m1+Ke*Ksi*Ksl*m1+
     Ki*Kp*Ksl*m1+
     Ki*Ksi*Ksl*alphap*mup+
```

```
     Ki*Ksi*Ksl*m1
-K1*K2*Ke*Ki*alpha0*mu0-              L(t)*S(t)^3*V(t)*X1(t)
     K1*K2*Ke*Ksi*alpha0*mu0-
     K1*K2*Ki*Kp*alpha0*mu0-
     K1*K2*Ki*Ksi*alpha0*mu0-
     K1*Ke*Ki*Kp*alpha0*mu0-
     K1*Ke*Ki*Ksi*alpha0*mu0-
     K1*Ki*Kp*Ksi*alpha0*mu0-
     K2*Ke*Ki*Kp*alpha0*mu0-
     K2*Ke*Ki*Ksi*alpha0*mu0-
     K2*Ki*Kp*Ksi*alpha0*mu0-
     Ke*Ki*Kp*Ksi*alpha0*mu0
+K0*K1*K2*Ke*Ki*Ksl*alphap*mup+        -1/2*V(t)*S(t)^2*X1(t)+
     K0*K1*K2*Ki*Ksi*Ksl*alphap*mup+  1/2*X2(t)*rho*V(t)*S(t)^2
     K0*K1*Ke*Ki*Kp*Ksl*m1+
     K0*K1*Ke*Ki*Ksi*Ksl*alphap*mup+
     K0*K1*Ke*Ki*Ksi*Ksl*m1+
     K0*K1*Ki*Kp*Ksi*Ksl*m1+
     K0*K2*Ke*Ki*Ksi*Ksl*alphap*mup+
     K0*Ke*Ki*Kp*Ksi*Ksl*m1+
     K1*K2*Ke*Ki*Ksi*Ksl*alphap*mup+
     K1*Ke*Ki*Kp*Ksi*Ksl*m1
-K0*K1*K2*Ki*Kp*Ksl*alphae*mue-        S(t)^2*V(t)*X0(t)
     K0*K1*K2*Ki*Ksi*Ksl*alphae*mue-
     K0*K1*Ki*Kp*Ksi*Ksl*alphae*mue-
     K0*K2*Ke*Ki*Kp*Ksl*m0-
     K0*K2*Ke*Ki*Ksi*Ksl*m0-
     K0*K2*Ki*Kp*Ksi*Ksl*alphae*mue-
     K0*K2*Ki*Kp*Ksi*Ksl*m0-
     K0*Ke*Ki*Kp*Ksi*Ksl*m0-
```

```
     K1*K2*Ki*Kp*Ksi*Ksl*alphae*mue-
     K2*Ke*Ki*Kp*Ksi*Ksl*m0
-K1*K2*Ke*alpha0*mu0-                  L(t)*S(t)^4*V(t)*X1(t)
     K1*K2*Ki*alpha0*mu0-
     K1*K2*Ksi*alpha0*mu0-
     K1*Ke*Ki*alpha0*mu0-
     K1*Ke*Ksi*alpha0*mu0-
     K1*Ki*Kp*alpha0*mu0-
     K1*Ki*Ksi*alpha0*mu0-
     K2*Ke*Ki*alpha0*mu0-
     K2*Ke*Ksi*alpha0*mu0-
     K2*Ki*Kp*alpha0*mu0-
     K2*Ki*Ksi*alpha0*mu0-
     Ke*Ki*Kp*alpha0*mu0-
     Ke*Ki*Ksi*alpha0*mu0-
     Ki*Kp*Ksi*alpha0*mu0
-K0*K1*alphae*mue-K0*K2*alphae*mue-     L(t)*S(t)^5*V(t)*X0(t)
     K0*K2*m0-K0*Ke*m0-
     K0*Ki*alphae*mue-K0*Ki*m0-
     K0*Ksi*alphae*mue-K0*Ksi*m0-
     K1*K2*alphae*mue-
     K1*Ki*alphae*mue-
     K1*Ksi*alphae*mue-K2*Ke*m0-
     K2*Ki*alphae*mue-K2*Ki*m0-
     K2*Ksi*alphae*mue-K2*Ksi*m0-
     Ke*Ki*m0-Ke*Ksi*m0-
     Ki*Kp*alphae*mue-Ki*Kp*m0-
     Ki*Ksi*alphae*mue-Ki*Ksi*m0
-K1*K2*Ke*Ki*Ksl*alpha0*mu0-           S(t)^3*V(t)*X1(t)
     K1*K2*Ke*Ksi*Ksl*alpha0*mu0-
```

```
     K1*K2*Ki*Kp*Ksl*alpha0*mu0-
     K1*K2*Ki*Ksi*Ksl*alpha0*mu0-
     K1*Ke*Ki*Kp*Ksl*alpha0*mu0-
     K1*Ke*Ki*Ksi*Ksl*alpha0*mu0-
     K1*Ki*Kp*Ksi*Ksl*alpha0*mu0-
     K2*Ke*Ki*Kp*Ksl*alpha0*mu0-
     K2*Ke*Ki*Ksi*Ksl*alpha0*mu0-
     K2*Ki*Kp*Ksi*Ksl*alpha0*mu0-
     Ke*Ki*Kp*Ksi*Ksl*alpha0*mu0
+K0*K1*K2*Ke+K0*K1*K2*Ki+              -diff(S(t),t)*V(t)*S(t)^3*L(t)-
     K0*K1*K2*Ksi+K0*K1*Ke*Ki+         Sigma2*S(t)^4*L(t)+
     K0*K1*Ke*Ksi+K0*K1*Ki*Kp+         Ff*sf*S(t)^3*L(t)
     K0*K1*Ki*Ksi+K0*K2*Ke*Ki+
     K0*K2*Ke*Ksi+K0*K2*Ki*Kp+
     K0*K2*Ki*Ksi+K0*Ke*Ki*Kp+
     K0*Ke*Ki*Ksi+K0*Ki*Kp*Ksi+
     K1*K2*Ke*Ki+K1*K2*Ke*Ksi+
     K1*K2*Ki*Kp+K1*K2*Ki*Ksi+
     K1*Ke*Ki*Kp+K1*Ke*Ki*Ksi+
     K1*Ki*Kp*Ksi+K2*Ke*Ki*Kp+
     K2*Ke*Ki*Ksi+K2*Ki*Kp*Ksi+
     Ke*Ki*Kp*Ksi
+K0*K1*K2*Ke*Ki*Ksl+                   -diff(S(t),t)*V(t)*S(t)^2-
     K0*K1*K2*Ke*Ksi*Ksl+              Sigma2*S(t)^3+Ff*sf*S(t)^2
     K0*K1*K2*Ki*Kp*Ksl+
     K0*K1*K2*Ki*Ksi*Ksl+
     K0*K1*Ke*Ki*Kp*Ksl+
     K0*K1*Ke*Ki*Ksi*Ksl+
     K0*K1*Ki*Kp*Ksi*Ksl+
     K0*K2*Ke*Ki*Kp*Ksl+
```

```
     K0*K2*Ke*Ki*Ksi*Ksl+
     K0*K2*Ki*Kp*Ksi*Ksl+
     K0*Ke*Ki*Kp*Ksi*Ksl+
     K1*K2*Ke*Ki*Kp*Ksl+
     K1*K2*Ke*Ki*Ksi*Ksl+
     K1*K2*Ki*Kp*Ksi*Ksl+
     K1*Ke*Ki*Kp*Ksi*Ksl+
     K2*Ke*Ki*Kp*Ksi*Ksl
+K0*K1*K2*Ksl+K0*K1*Ke*Ksl+           -diff(S(t),t)*V(t)*S(t)^4-
     K0*K1*Ki*Ksl+K0*K1*Ksi*Ksl+      Sigma2*S(t)^5+Ff*sf*S(t)^4
     K0*K2*Ke*Ksl+K0*K2*Ki*Ksl+
     K0*K2*Ksi*Ksl+K0*Ke*Ki*Ksl+
     K0*Ke*Ksi*Ksl+K0*Ki*Kp*Ksl+
     K0*Ki*Ksi*Ksl+K1*K2*Ke*Ksl+
     K1*K2*Ki*Ksl+K1*K2*Ksi*Ksl+
     K1*Ke*Ki*Ksl+K1*Ke*Ksi*Ksl+
     K1*Ki*Kp*Ksl+K1*Ki*Ksi*Ksl+
     K2*Ke*Ki*Ksl+K2*Ke*Ksi*Ksl+
     K2*Ki*Kp*Ksl+K2*Ki*Ksi*Ksl+
     Ke*Ki*Kp*Ksl+Ke*Ki*Ksi*Ksl+
     Ki*Kp*Ksi*Ksl
-K1*K2*Ke*Ksl*alpha0*mu0-             S(t)^4*V(t)*X1(t)
     K1*K2*Ki*Ksl*alpha0*mu0-
     K1*K2*Ksi*Ksl*alpha0*mu0-
     K1*Ke*Ki*Ksl*alpha0*mu0-
     K1*Ke*Ksi*Ksl*alpha0*mu0-
     K1*Ki*Kp*Ksl*alpha0*mu0-
     K1*Ki*Ksi*Ksl*alpha0*mu0-
     K2*Ke*Ki*Ksl*alpha0*mu0-
     K2*Ke*Ksi*Ksl*alpha0*mu0-
```

```
     K2*Ki*Kp*Ksl*alpha0*mu0-
     K2*Ki*Ksi*Ksl*alpha0*mu0-
     Ke*Ki*Kp*Ksl*alpha0*mu0-
     Ke*Ki*Ksi*Ksl*alpha0*mu0-
     Ki*Kp*Ksi*Ksl*alpha0*mu0
+K0*K1*Ke*m1+K0*K1*Ki*alphap*mup+      -1/2*V(t)*S(t)^4*X1(t)*L(t)+
     K0*K1*Ki*m1+K0*K1*Ksi*m1+         1/2*X2(t)*rho*V(t)*S(t)^4*L(t)
     K0*K2*Ki*alphap*mup+
     K0*Ke*Ki*alphap*mup+K0*Ke*Ki*m1+
     K0*Ke*Ksi*m1+K0*Ki*Kp*m1+
     K0*Ki*Ksi*alphap*mup+
     K0*Ki*Ksi*m1+
     K1*K2*Ki*alphap*mup+
     K1*Ke*Ki*alphap*mup+K1*Ke*Ki*m1+
     K1*Ke*Ksi*m1+K1*Ki*Kp*m1+
     K1*Ki*Ksi*alphap*mup+
     K1*Ki*Ksi*m1+
     K2*Ke*Ki*alphap*mup+
     K2*Ki*Ksi*alphap*mup+
     Ke*Ki*Kp*m1+
     Ke*Ki*Ksi*alphap*mup+
     Ke*Ki*Ksi*m1+Ki*Kp*Ksi*m1
-K0*K1*Ksl*alphae*mue-                 S(t)^5*V(t)*X0(t)
     K0*K2*Ksl*alphae*mue-
     K0*K2*Ksl*m0-K0*Ke*Ksl*m0-
     K0*Ki*Ksl*alphae*mue-
     K0*Ki*Ksl*m0-
     K0*Ksi*Ksl*alphae*mue-
     K0*Ksi*Ksl*m0-
     K1*K2*Ksl*alphae*mue-
```

```
     K1*Ki*Ksl*alphae*mue-
     K1*Ksi*Ksl*alphae*mue-
     K2*Ke*Ksl*m0-
     K2*Ki*Ksl*alphae*mue-
     K2*Ki*Ksl*m0-
     K2*Ksi*Ksl*alphae*mue-
     K2*Ksi*Ksl*m0-Ke*Ki*Ksl*m0-
     Ke*Ksi*Ksl*m0-
     Ki*Kp*Ksl*alphae*mue-
     Ki*Kp*Ksl*m0-
     Ki*Ksi*Ksl*alphae*mue-
     Ki*Ksi*Ksl*m0
+K0*K1*K2*Ki*alphap*mup+              -1/2*V(t)*S(t)^3*X1(t)*L(t)+
     K0*K1*Ke*Ki*alphap*mup+         1/2*X2(t)*rho*V(t)*S(t)^3*L(t)
     K0*K1*Ke*Ki*m1+K0*K1*Ke*Ksi*m1+
     K0*K1*Ki*Kp*m1+
     K0*K1*Ki*Ksi*alphap*mup+
     K0*K1*Ki*Ksi*m1+
     K0*K2*Ke*Ki*alphap*mup+
     K0*K2*Ki*Ksi*alphap*mup+
     K0*Ke*Ki*Kp*m1+
     K0*Ke*Ki*Ksi*alphap*mup+
     K0*Ke*Ki*Ksi*m1+K0*Ki*Kp*Ksi*m1+
     K1*K2*Ke*Ki*alphap*mup+
     K1*K2*Ki*Ksi*alphap*mup+
     K1*Ke*Ki*Kp*m1+
     K1*Ke*Ki*Ksi*alphap*mup+
     K1*Ke*Ki*Ksi*m1+K1*Ki*Kp*Ksi*m1+
     K2*Ke*Ki*Ksi*alphap*mup+
     Ke*Ki*Kp*Ksi*m1
```

```
+K0*K1*K2*Ke*Ksl+K0*K1*K2*Ki*Ksl+
     K0*K1*K2*Ksi*Ksl+
     K0*K1*Ke*Ki*Ksl+
     K0*K1*Ke*Ksi*Ksl+
     K0*K1*Ki*Kp*Ksl+
     K0*K1*Ki*Ksi*Ksl+
     K0*K2*Ke*Ki*Ksl+
     K0*K2*Ke*Ksi*Ksl+
     K0*K2*Ki*Kp*Ksl+
     K0*K2*Ki*Ksi*Ksl+
     K0*Ke*Ki*Kp*Ksl+
     K0*Ke*Ki*Ksi*Ksl+
     K0*Ki*Kp*Ksi*Ksl+
     K1*K2*Ke*Ki*Ksl+
     K1*K2*Ke*Ksi*Ksl+
     K1*K2*Ki*Kp*Ksl+
     K1*K2*Ki*Ksi*Ksl+
     K1*Ke*Ki*Kp*Ksl+
     K1*Ke*Ki*Ksi*Ksl+
     K1*Ki*Kp*Ksi*Ksl+
     K2*Ke*Ki*Kp*Ksl+
     K2*Ke*Ki*Ksi*Ksl+
     K2*Ki*Kp*Ksi*Ksl+
     Ke*Ki*Kp*Ksi*Ksl
-K0*K1*K2*Ki*alphae*mue-
     K0*K1*K2*Ksi*alphae*mue-
     K0*K1*Ki*Kp*alphae*mue-
     K0*K1*Ki*Ksi*alphae*mue-
     K0*K2*Ke*Ki*m0-K0*K2*Ke*Ksi*m0-
     K0*K2*Ki*Kp*alphae*mue-
-diff(S(t),t)*V(t)*S(t)^3-
Sigma2*S(t)^4+Ff*sf*S(t)^3
L(t)*S(t)^3*V(t)*X0(t)
```

```
     K0*K2*Ki*Kp*m0-
     K0*K2*Ki*Ksi*alphae*mue-
     K0*K2*Ki*Ksi*m0-K0*Ke*Ki*Kp*m0-
     K0*Ke*Ki*Ksi*m0-
     K0*Ki*Kp*Ksi*alphae*mue-
     K0*Ki*Kp*Ksi*m0-
     K1*K2*Ki*Kp*alphae*mue-
     K1*K2*Ki*Ksi*alphae*mue-
     K1*Ki*Kp*Ksi*alphae*mue-
     K2*Ke*Ki*Kp*m0-K2*Ke*Ki*Ksi*m0-
     K2*Ki*Kp*Ksi*alphae*mue-
     K2*Ki*Kp*Ksi*m0-Ke*Ki*Kp*Ksi*m0
-K0*K1*K2*alphae*mue-                  L(t)*S(t)^4*V(t)*X0(t)
     K0*K1*Ki*alphae*mue-
     K0*K1*Ksi*alphae*mue-
     K0*K2*Ke*m0-K0*K2*Ki*alphae*mue-
     K0*K2*Ki*m0-
     K0*K2*Ksi*alphae*mue-
     K0*K2*Ksi*m0-K0*Ke*Ki*m0-
     K0*Ke*Ksi*m0-
     K0*Ki*Kp*alphae*mue-K0*Ki*Kp*m0-
     K0*Ki*Ksi*alphae*mue-
     K0*Ki*Ksi*m0-
     K1*K2*Ki*alphae*mue-
     K1*K2*Ksi*alphae*mue-
     K1*Ki*Kp*alphae*mue-
     K1*Ki*Ksi*alphae*mue-
     K2*Ke*Ki*m0-K2*Ke*Ksi*m0-
     K2*Ki*Kp*alphae*mue-K2*Ki*Kp*m0-
     K2*Ki*Ksi*alphae*mue-
```

```
     K2*Ki*Ksi*m0-Ke*Ki*Kp*m0-
     Ke*Ki*Ksi*m0-
     Ki*Kp*Ksi*alphae*mue-
     Ki*Kp*Ksi*m0
+K0*K1*Ke*Ksl*m1+                    -1/2*V(t)*S(t)^4*X1(t)+
     K0*K1*Ki*Ksl*alphap*mup+        1/2*X2(t)*rho*V(t)*S(t)^4
     K0*K1*Ki*Ksl*m1+
     K0*K1*Ksi*Ksl*m1+
     K0*K2*Ki*Ksl*alphap*mup+
     K0*Ke*Ki*Ksl*alphap*mup+
     K0*Ke*Ki*Ksl*m1+
     K0*Ke*Ksi*Ksl*m1+
     K0*Ki*Kp*Ksl*m1+
     K0*Ki*Ksi*Ksl*alphap*mup+
     K0*Ki*Ksi*Ksl*m1+
     K1*K2*Ki*Ksl*alphap*mup+
     K1*Ke*Ki*Ksl*alphap*mup+
     K1*Ke*Ki*Ksl*m1+
     K1*Ke*Ksi*Ksl*m1+
     K1*Ki*Kp*Ksl*m1+
     K1*Ki*Ksi*Ksl*alphap*mup+
     K1*Ki*Ksi*Ksl*m1+
     K2*Ke*Ki*Ksl*alphap*mup+
     K2*Ki*Ksi*Ksl*alphap*mup+
     Ke*Ki*Kp*Ksl*m1+
     Ke*Ki*Ksi*Ksl*alphap*mup+
     Ke*Ki*Ksi*Ksl*m1+
     Ki*Kp*Ksi*Ksl*m1
+K0*K1*K2*Ki*Ksl*alphap*mup+         -1/2*V(t)*S(t)^3*X1(t)+
     K0*K1*Ke*Ki*Ksl*alphap*mup+     1/2*X2(t)*rho*V(t)*S(t)^3
```

```
     K0*K1*Ke*Ki*Ksl*m1+
     K0*K1*Ke*Ksi*Ksl*m1+
     K0*K1*Ki*Kp*Ksl*m1+
     K0*K1*Ki*Ksi*Ksl*alphap*mup+
     K0*K1*Ki*Ksi*Ksl*m1+
     K0*K2*Ke*Ki*Ksl*alphap*mup+
     K0*K2*Ki*Ksi*Ksl*alphap*mup+
     K0*Ke*Ki*Kp*Ksl*m1+
     K0*Ke*Ki*Ksi*Ksl*alphap*mup+
     K0*Ke*Ki*Ksi*Ksl*m1+
     K0*Ki*Kp*Ksi*Ksl*m1+
     K1*K2*Ke*Ki*Ksl*alphap*mup+
     K1*K2*Ki*Ksi*Ksl*alphap*mup+
     K1*Ke*Ki*Kp*Ksl*m1+
     K1*Ke*Ki*Ksi*Ksl*alphap*mup+
     K1*Ke*Ki*Ksi*Ksl*m1+
     K1*Ki*Kp*Ksi*Ksl*m1+
     K2*Ke*Ki*Ksi*Ksl*alphap*mup+
     Ke*Ki*Kp*Ksi*Ksl*m1
-K0*K1*K2*Ki*Ksl*alphae*mue-          S(t)^3*V(t)*X0(t)
     K0*K1*K2*Ksi*Ksl*alphae*mue-
     K0*K1*Ki*Kp*Ksl*alphae*mue-
     K0*K1*Ki*Ksi*Ksl*alphae*mue-
     K0*K2*Ke*Ki*Ksl*m0-
     K0*K2*Ke*Ksi*Ksl*m0-
     K0*K2*Ki*Kp*Ksl*alphae*mue-
     K0*K2*Ki*Kp*Ksl*m0-
     K0*K2*Ki*Ksi*Ksl*alphae*mue-
     K0*K2*Ki*Ksi*Ksl*m0-
     K0*Ke*Ki*Kp*Ksl*m0-
```

```
     K0*Ke*Ki*Ksi*Ksl*m0-
     K0*Ki*Kp*Ksi*Ksl*alphae*mue-
     K0*Ki*Kp*Ksi*Ksl*m0-
     K1*K2*Ki*Kp*Ksl*alphae*mue-
     K1*K2*Ki*Ksi*Ksl*alphae*mue-
     K1*Ki*Kp*Ksi*Ksl*alphae*mue-
     K2*Ke*Ki*Kp*Ksl*m0-
     K2*Ke*Ki*Ksi*Ksl*m0-
     K2*Ki*Kp*Ksi*Ksl*alphae*mue-
     K2*Ki*Kp*Ksi*Ksl*m0-
     Ke*Ki*Kp*Ksi*Ksl*m0
-K0*K1*K2*Ksl*alphae*mue-              S(t)^4*V(t)*X0(t)
     K0*K1*Ki*Ksl*alphae*mue-
     K0*K1*Ksi*Ksl*alphae*mue-
     K0*K2*Ke*Ksl*m0-
     K0*K2*Ki*Ksl*alphae*mue-
     K0*K2*Ki*Ksl*m0-
     K0*K2*Ksi*Ksl*alphae*mue-
     K0*K2*Ksi*Ksl*m0-
     K0*Ke*Ki*Ksl*m0-
     K0*Ke*Ksi*Ksl*m0-
     K0*Ki*Kp*Ksl*alphae*mue-
     K0*Ki*Kp*Ksl*m0-
     K0*Ki*Ksi*Ksl*alphae*mue-
     K0*Ki*Ksi*Ksl*m0-
     K1*K2*Ki*Ksl*alphae*mue-
     K1*K2*Ksi*Ksl*alphae*mue-
     K1*Ki*Kp*Ksl*alphae*mue-
     K1*Ki*Ksi*Ksl*alphae*mue-
     K2*Ke*Ki*Ksl*m0-
```

```
K2*Ke*Ksi*Ksl*m0-
K2*Ki*Kp*Ksl*alphae*mue-
K2*Ki*Kp*Ksl*m0-
K2*Ki*Ksi*Ksl*alphae*mue-
K2*Ki*Ksi*Ksl*m0-
Ke*Ki*Kp*Ksl*m0-
Ke*Ki*Ksi*Ksl*m0-
Ki*Kp*Ksi*Ksl*alphae*mue-
Ki*Kp*Ksi*Ksl*m0
```

D. GENERATING THE FISHER INFORMATION MATRIX USING MAPLE

The symbolic mathematics package Maple was used to simplify the calculation of the Fisher Information Matrices used in searching for optimal experiment designs. This appendix contains a listing, automatically generated from Maple (in LaTeX format), which describes how the Fisher Information Matrix was first calculated symbolically from the model equations describing the simplified and lactose-incorporating version of the model of (Paul and Thomas, 1996), and then illustrates the use of Maple's C code generation facility to produce snippets of code that were subsequently spliced into *S*-functions for use with SIMULINK. The raw output as obtained from Maple has been modified slightly so as to improve its layout on the printed page.

D.1 Start of Maple Session, and Introductory Comments

```
> restart;
```

This file needed to be modified on 8/1/98 to take into consideration the fact that the model proper does not just make use of the feed rate (of glucose),

but also takes into consideration the rate of addition of PAA precursor, and the rate of abstraction of filtered liquor for HPLC analysis.

The practical upshot of this is that the F in the insoluble species equations (X0,X1,X2,X3,X4 and V) is replaced by (F+FX-SR),

and the F in the soluble species equations (S,L,P) is replaced by (F+FX).

D.2 Model Equations

We start by entering the equations defining the model.

```
> rho1 := X1/((X1/rho)+X2):

> v1c := X1/(2*rho1) - X2:

> X0expr := mu0*X1*S/(K0+S) - gamma1*X0/(K1+S)
- (F+FX-SR)*X0/V:

> X1expr := mue*X0*S/(Ke+S) -mu0*X1*S/(K0+S)
+ gamma1*X0/(K1+S) - mu2*X2*rho - (F+FX-SR)*X1/V:

> X2expr := mu1*v1c - mu2*X2 + mu3*X2 - (F+FX-SR)*X2/V:

> X3expr := mu2*X2*rho - mua*X3 - (F+FX-SR)*X3/V:

> X4expr := mua*X3 - (F+FX-SR)*X4/V:

> Sexpr := -alpha0*mu0*X1*S/(K0+S) - alphae*mue*X0*S/(Ke+S)
- m0*X0*S/(K1+S) - m1*rho*v1c*S/(K2+S)
- alphap*mup*rho*v1c*S/(Kp+S*(1+S/Ki))
+ mul*L*(X0+X1)/((Ksl+L)*(1+(S/Ksi))) + F*sf/V - (F+FX)*S/V:

> Lexpr := - mul*L*(X0+X1)/((Ksl+L)*(1+(S/Ksi)))
- (F+FX)*L/V:

> Pexpr := mup*rho*v1c*S/(Kp+S*(1+S/Ki)) - muh*P
- (F+FX)*P/V:

> Vexpr := (F+FX-SR):
```

D.3 Nonlinear State Derivative Vector

Next we construct the nonlinear state derivative vector from the model equations.

```
> stateqs := array(1..8):
> stateqs[1] := X0expr:
> stateqs[2] := X1expr:
> stateqs[3] := X2expr:
> stateqs[4] := X3expr:
> stateqs[5] := Sexpr:
> stateqs[6] := Lexpr:
> stateqs[7] := Pexpr:
> stateqs[8] := Vexpr:
> eval(stateqs);
```

$$\left[\frac{\mu 0\, X1\, S}{K0+S}-\frac{\gamma 1\, X0}{K1+S}-\frac{(F+FX-SR)\, X0}{V},\ \frac{mue\, X0\, S}{Ke+S}\right.$$
$$-\frac{\mu 0\, X1\, S}{K0+S}+\frac{\gamma 1\, X0}{K1+S}-\mu 2\, X2\, \rho-\frac{(F+FX-SR)\, X1}{V},$$
$$\mu 1\, \%1-\mu 2\, X2+\mu 3\, X2-\frac{(F+FX-SR)\, X2}{V},$$
$$\mu 2\, X2\, \rho-mua\, X3-\frac{(F+FX-SR)\, X3}{V},-\frac{\alpha 0\, \mu 0\, X1\, S}{K0+S}$$
$$-\frac{alphae\, mue\, X0\, S}{Ke+S}-\frac{m0\, X0\, S}{K1+S}-\frac{m1\, \rho\, \%1\, S}{K2+S}$$
$$\left.-\frac{alphap\, mup\, \rho\, \%1\, S}{Kp+S\,(1+\frac{S}{Ki})}+\frac{mul\, L\,(X0+X1)}{(Ksl+L)\,(1+\frac{S}{Ksi})}+\frac{F\, sf}{V}\right.$$

$$-\frac{(F+FX)\,S}{V},\ -\frac{mul\,L\,(X0+X1)}{(Ksl+L)\,(1+\frac{S}{Ksi})}-\frac{(F+FX)\,L}{V},$$
$$\frac{mup\,\rho\,\%1\,S}{Kp+S\,(1+\frac{S}{Ki})}-muh\,P-\frac{(F+FX)\,P}{V},\ F+FX-SR\Big]$$
$$\%1:=\frac{1}{2}\,\frac{X1}{\rho}-\frac{1}{2}\,X2$$

D.4 States Vector

Then we construct the states vector; this must be in the same order as the nonlinear state derivative vector.

```
> states := array(1..8):
> states[1] := X0:
> states[2] := X1:
> states[3] := X2:
> states[4] := X3:
> states[5] := S:
> states[6] := L:
> states[7] := P:
> states[8] := V:
> eval(states);
```

$$[X0,\ X1,\ X2,\ X3,\ S,\ L,\ P,\ V]$$

D.5 Calculating the Derivatives wrt the States

Generating the delf_delx matrix.

```
> delf_delx := array(1..8,1..8):
> for i to 8 do for j to 8 do delf_delx[i,j] :=
diff(stateqs[i],states[j]) od od:
> delf_delx:
> eval(delf_delx);
```

$$\left[-\frac{\gamma 1}{K1+S}-\frac{F+FX-SR}{V},\ \frac{\mu 0\,S}{K0+S},\ 0,\ 0,\right.$$
$$\left.\frac{\mu 0\,X1}{K0+S}-\frac{\mu 0\,X1\,S}{(K0+S)^2}+\frac{\gamma 1\,X0}{(K1+S)^2},\ 0,\ 0,\ \frac{(F+FX-SR)\,X0}{V^2}\right]$$
$$\left[\frac{mue\,S}{Ke+S}+\frac{\gamma 1}{K1+S},\ -\frac{\mu 0\,S}{K0+S}-\frac{F+FX-SR}{V},\ -\mu 2\,\rho,\ 0,\right.$$
$$\frac{mue\,X0}{Ke+S}-\frac{mue\,X0\,S}{(Ke+S)^2}-\frac{\mu 0\,X1}{K0+S}+\frac{\mu 0\,X1\,S}{(K0+S)^2}-\frac{\gamma 1\,X0}{(K1+S)^2},\ 0$$
$$\left.,\ 0,\ \frac{(F+FX-SR)\,X1}{V^2}\right]$$
$$\left[0,\ \frac{1}{2}\,\frac{\mu 1}{\rho},\ -\frac{1}{2}\,\mu 1-\mu 2+\mu 3-\frac{F+FX-SR}{V},\ 0,\ 0,\ 0,\ 0,\right.$$
$$\left.\frac{(F+FX-SR)\,X2}{V^2}\right]$$
$$\left[0,\ 0,\ \mu 2\,\rho,\ -mua-\frac{F+FX-SR}{V},\ 0,\ 0,\ 0,\right.$$
$$\left.\frac{(F+FX-SR)\,X3}{V^2}\right]$$
$$\left[-\frac{alphae\,mue\,S}{Ke+S}-\frac{m0\,S}{K1+S}+\frac{mul\,L}{(Ksl+L)\,(1+\frac{S}{Ksi})},\ -\frac{\alpha 0\,\mu 0\,S}{K0+S}\right.$$
$$\left.-\frac{1}{2}\,\frac{m1\,S}{K2+S}-\frac{1}{2}\,\frac{alphap\,mup\,S}{Kp+S\,(1+\frac{S}{Ki})}+\frac{mul\,L}{(Ksl+L)\,(1+\frac{S}{Ksi})},\right.$$

$$\frac{1}{2}\,\frac{m1\,\rho\,S}{K2+S}+\frac{1}{2}\,\frac{alphap\,mup\,\rho\,S}{Kp+S\,(1+\frac{S}{Ki})}\,,\,0\,,-\frac{\alpha0\,\mu0\,X1}{K0+S}$$
$$+\frac{\alpha0\,\mu0\,X1\,S}{(K0+S)^2}-\frac{alphae\,mue\,X0}{Ke+S}+\frac{alphae\,mue\,X0\,S}{(Ke+S)^2}$$
$$-\frac{m0\,X0}{K1+S}+\frac{m0\,X0\,S}{(K1+S)^2}-\frac{m1\,\rho\,\%1}{K2+S}+\frac{m1\,\rho\,\%1\,S}{(K2+S)^2}$$
$$-\frac{alphap\,mup\,\rho\,\%1}{Kp+S\,(1+\frac{S}{Ki})}+\frac{alphap\,mup\,\rho\,\%1\,S\,(1+2\,\frac{S}{Ki})}{(Kp+S\,(1+\frac{S}{Ki}))^2}$$
$$-\frac{mul\,L\,(X0+X1)}{(Ksl+L)\,(1+\frac{S}{Ksi})^2\,Ksi}-\frac{F+FX}{V},$$
$$\frac{mul\,(X0+X1)}{(Ksl+L)\,(1+\frac{S}{Ksi})}-\frac{mul\,L\,(X0+X1)}{(Ksl+L)^2\,(1+\frac{S}{Ksi})}\,,\,0\,,$$
$$-\frac{F\,sf}{V^2}+\frac{(F+FX)\,S}{V^2}\Bigg]$$

$$\Bigg[-\frac{mul\,L}{(Ksl+L)\,(1+\frac{S}{Ksi})}\,,\,-\frac{mul\,L}{(Ksl+L)\,(1+\frac{S}{Ksi})}\,,\,0\,,\,0\,,$$
$$\frac{mul\,L\,(X0+X1)}{(Ksl+L)\,(1+\frac{S}{Ksi})^2\,Ksi}\,,$$
$$-\frac{mul\,(X0+X1)}{(Ksl+L)\,(1+\frac{S}{Ksi})}+\frac{mul\,L\,(X0+X1)}{(Ksl+L)^2\,(1+\frac{S}{Ksi})}-\frac{F+FX}{V}\,,\,0\,,$$
$$\frac{(F+FX)\,L}{V^2}\Bigg]$$

$$\Bigg[0\,,\,\frac{1}{2}\,\frac{mup\,S}{Kp+S\,(1+\frac{S}{Ki})}\,,\,-\frac{1}{2}\,\frac{mup\,\rho\,S}{Kp+S\,(1+\frac{S}{Ki})}\,,\,0\,,$$
$$\frac{mup\,\rho\,\%1}{Kp+S\,(1+\frac{S}{Ki})}-\frac{mup\,\rho\,\%1\,S\,(1+2\,\frac{S}{Ki})}{(Kp+S\,(1+\frac{S}{Ki}))^2}\,,\,0\,,$$

$$-muh - \frac{F + FX}{V}, \frac{(F + FX)\,P}{V^2}\Bigg]$$

$$[0, 0, 0, 0, 0, 0, 0, 0]$$

$$\%1 := \frac{1}{2}\,\frac{X1}{\rho} - \frac{1}{2}\,X2$$

D.6 Parameters Vector

This is an arbitrary ordering of the parameters, but the same ordering should be used in the SIMULINK model. (By this, I mean that the order in which parameter arguments are passed to the S-function calculating the sensitivity matrix should be the same order as that used in calculating the sensitivity matrix in Maple. Although this is not strictly necessary, it is simpler to use the same parameter order in both places than it is to convert from one list order to another in moving from Maple to SIMULINK.)

```
> parameters := array(1..23):
> parameters[1] := mu0:
> parameters[2] := K0:
> parameters[3] := gamma1:
> parameters[4] := K1:
> parameters[5] := mue:
> parameters[6] := Ke:
> parameters[7] := mu2:
> parameters[8] := mu1:
```

```
> parameters[9] := mu3:
> parameters[10] := mua:
> parameters[11] := alpha0:
> parameters[12] := alphae:
> parameters[13] := m0:
> parameters[14] := m1:
> parameters[15] := K2:
> parameters[16] := alphap:
> parameters[17] := mup:
> parameters[18] := Kp:
> parameters[19] := Ki:
> parameters[20] := mul:
> parameters[21] := Ksl:
> parameters[22] := Ksi:
> parameters[23] := muh:
> eval(parameters);
```

$$[\mu 0,\ K0,\ \gamma 1,\ K1,\ mue,\ Ke,\ \mu 2,\ \mu 1,\ \mu 3,\ mua,\ \alpha 0,\ alphae,\ m0,\ m1,\ K2,\ alphap,\ mup,\ Kp,\ Ki,\ mul,\ Ksl,\ Ksi,\ muh]$$

D.7 Calculating the Derivatives wrt the Parameters

Generating the delf_delp matrix.

```
> delf_delp := array(1..8,1..23):
> for i to 8 do for j to 23 do delf_delp[i,j] :=
diff(stateqs[i],parameters[j]) od od:
> delf_delp:
> eval(delf_delp);
```

$$\left[\frac{X1\,S}{K0+S}, -\frac{\mu 0\,X1\,S}{(K0+S)^2}, -\frac{X0}{K1+S}, \frac{\gamma 1\,X0}{(K1+S)^2}, 0, 0, 0, 0, 0,\right.$$
$$\left. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\right]$$
$$\left[-\frac{X1\,S}{K0+S}, \frac{\mu 0\,X1\,S}{(K0+S)^2}, \frac{X0}{K1+S}, -\frac{\gamma 1\,X0}{(K1+S)^2}, \frac{X0\,S}{Ke+S},\right.$$
$$-\frac{mue\,X0\,S}{(Ke+S)^2}, -X2\,\rho, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,$$
$$\left. 0, 0, 0, 0\right]$$
$$[0, 0, 0, 0, 0, 0, -X2, \%1, X2, 0, 0, 0, 0, 0, 0, 0, 0,$$
$$0, 0, 0, 0, 0, 0]$$
$$[0, 0, 0, 0, 0, 0, X2\,\rho, 0, 0, -X3, 0, 0, 0, 0, 0, 0, 0,$$
$$0, 0, 0, 0, 0, 0]$$
$$\left[-\frac{\alpha 0\,X1\,S}{K0+S}, \frac{\alpha 0\,\mu 0\,X1\,S}{(K0+S)^2}, 0, \frac{m0\,X0\,S}{(K1+S)^2}, -\frac{alphae\,X0\,S}{Ke+S},\right.$$
$$\frac{alphae\,mue\,X0\,S}{(Ke+S)^2}, 0, 0, 0, 0, -\frac{\mu 0\,X1\,S}{K0+S}, -\frac{mue\,X0\,S}{Ke+S},$$
$$-\frac{X0\,S}{K1+S}, -\frac{\rho\,\%1\,S}{K2+S}, \frac{m1\,\rho\,\%1\,S}{(K2+S)^2}, -\frac{mup\,\rho\,\%1\,S}{Kp+S\,(1+\frac{S}{Ki})},$$
$$-\frac{alphap\,\rho\,\%1\,S}{Kp+S\,(1+\frac{S}{Ki})}, \frac{alphap\,mup\,\rho\,\%1\,S}{(Kp+S\,(1+\frac{S}{Ki}))^2},$$
$$\left.-\frac{alphap\,mup\,\rho\,\%1\,S^3}{(Kp+S\,(1+\frac{S}{Ki}))^2\,Ki^2}, \frac{L\,(X0+X1)}{(Ksl+L)\,(1+\frac{S}{Ksi})},\right.$$

$$-\frac{mul\,L\,(X0+X1)}{(Ksl+L)^2\,(1+\frac{S}{Ksi})},\ \frac{mul\,L\,(X0+X1)\,S}{(Ksl+L)\,(1+\frac{S}{Ksi})^2\,Ksi^2},\ 0\Bigg]$$

$$\Bigg[0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,$$

$$-\frac{L\,(X0+X1)}{(Ksl+L)\,(1+\frac{S}{Ksi})},\ \frac{mul\,L\,(X0+X1)}{(Ksl+L)^2\,(1+\frac{S}{Ksi})},$$

$$-\frac{mul\,L\,(X0+X1)\,S}{(Ksl+L)\,(1+\frac{S}{Ksi})^2\,Ksi^2},\ 0\Bigg]$$

$$\Bigg[0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,$$

$$\frac{\rho\,\%1\,S}{Kp+S\,(1+\frac{S}{Ki})},\ -\frac{mup\,\rho\,\%1\,S}{(Kp+S\,(1+\frac{S}{Ki}))^2},$$

$$\frac{mup\,\rho\,\%1\,S^3}{(Kp+S\,(1+\frac{S}{Ki}))^2\,Ki^2},\ 0,\ 0,\ 0,\ -P\Bigg]$$

$$[0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,$$

$$0,\ 0,\ 0,\ 0]$$

$$\%1 := \frac{1}{2}\,\frac{X1}{\rho} - \frac{1}{2}\,X2$$

D.8 Generating C code for use in the SIMULINK S-function

Maple's C code utilities were used to automatically produce C code describing the calculation of the delf_delp and delf_delx matrices in terms of the stated parameters and states. This code was then be copied into a template *S*-function, with the parameters used here being related to a vector of parameters passed into the *S*-function from SIMULINK and the states being related to the inputs passed into the *S*-function in SIMULINK. This required

a few small modifications to the standard *S*-function template.

The Maple command to include the C utilities in the current session, making them available for subsequent use, is `readlib(C):`, and the command used to generate the two sets of optimised C code for the two derivative matrices were `C(delf_delp, optimized):` for the derivative with respect to the parameters, and `C(delf_delx, optimized):` for the derivative with respect to the states. The outputs of these two commands, and the C *S*-function produced using the Maple-generated C are not shown here, for reasons of brevity.

E. CONFERENCE PAPER

The following paper was presented at the 7th International Conference on Computer Applications in Biotechnology (CAB7), held in Osaka, Japan, in June 1998.

IMPROVING THE ESTIMATION OF PARAMETERS OF PENICILLIN FERMENTATION MODELS

M.T. Syddall * **G.C. Paul** * **C.A. Kent** *,1

* *Centre for Bioprocess Engineering, School of Chemical Engineering, The University of Birmingham, Edgbaston, Birmingham, B15 2TT, United Kingdom*

Abstract: Models for use in control and estimation applications should match the process as closely as possible. Fermentation process models are usually complex, containing many states and parameters. Obtaining accurate estimates of the parameters of such models is a costly and time-consuming process. Here we show a way of reducing the time and cost by designing optimal experiments for parameter identification. The method presented uses genetic algorithms to search for input profiles which optimise scalar functions of the Fisher information matrix, thus maximising the improvement in the parameter estimates that may be obtained from each experiment performed. The *Penicillium chrysogenum* penicillin-G fermentation, a secondary metabolite fermentation, is used as an example.

Keywords: Fermentation processes, Modelling, Parameter estimation, Genetic algorithms, Optimal experiment design

1. INTRODUCTION

The use of differential equation based physiological models in the design of optimal production strategies for penicillin fermentations (Lim *et al.* 1986, San and Stephanopoulos 1989) and in the design of advanced controllers for the fermentation (di Massimo *et al.* 1992, van Impe and Bastin 1995) has previously been reported in the literature.

The models on which these approaches were based contain parameters which must be estimated from experimental data, which can often cause difficulties. Nihtilä and Virkunnen (1977) reported lack of confidence in parameter estimates obtained tuning bacterial models with data from batch fermentations. Holmberg (1982) demonstrated the theoretical identifiability of a model incorporating Michaelis-Menten kinetics, but went on to show that, given limited samples of noisy data, this same model was not practically identifiable, although more frequent sampling did help to improve confidence in the estimates obtained. The fact that algorithmic parameter estimation methods such as Marquardt methods do not necessarily lead to global optima was also mentioned.

These difficulties may be related to the geometry of the least squares error surface on which the parameter estimation is performed. Holmberg and Ranta (1982) showed that the niveau curves (contours of constant error) about the optimal parameter set were long and narrow.

Series of experiments are often performed, generating incremental improvements in the quality of the parameter estimates. This is typically costly and time-consuming, particularly in the case of fermentation modelling, where a single experiment may take a week or more. Designing experiments so as to gain the maximum improvement in the parameter estimates could save time and

[1] Author to whom correspondence should be addressed

money in the development of models for use in advanced fermentation control.

2. TUNING THE MODEL PARAMETERS

Throughout this work, the penicillin fermentation is considered as being described by a nonlinear differential equation based model of the following form.

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \boldsymbol{\beta}, \mathbf{u}(t)) \quad (1)$$

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \boldsymbol{\beta}, \mathbf{u}(t)) \quad (2)$$

In the above, $\mathbf{x}(t)$ is a vector of model states, $\boldsymbol{\beta}$ is a set of time-invariant parameters, and $\mathbf{u}(t)$ is the vector of inputs to the model. The output of the model is $\mathbf{y}(t)$; this second equation is used to relate measurements to the model states. (In our case, for simplicity, we assume $\mathbf{y}(t) = \mathbf{x}(t)$.) The model structure used in this work is given in section 4.1.

In order to be able to make use of the model for practical purposes, it must first be tuned so as to most accurately represent the fermentation. This may be done using a least squares based optimisation routine, with an objective function of the following form.

$$E = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{m}(t_i) - \mathbf{x}(t_i))'\mathbf{W}(\mathbf{m}(t_i) - \mathbf{x}(t_i)) \quad (3)$$

In the above, E is the error value, $\mathbf{m}(t_i)$ is a vector of measurement values at times t_i, the summation is carried out for n sample times, and $\mathbf{W}$ is a weighting matrix. In our case, $\mathbf{W}$ is a diagonal matrix, with the maximum values of the measured states along the diagonal. The prime $'$ denotes vector or matrix transposition.

2.1 *A geometrical interpretation of the errors*

The error function E can be considered as a hypersurface given by

$$E = E(\boldsymbol{\beta}) - E|_{\boldsymbol{\beta}=\mathbf{b}} \quad (4)$$

where $\mathbf{b}$ denotes the optimal parameter set, and hence $E|_{\boldsymbol{\beta}=\mathbf{b}}$ gives the minimum value for E.

If we assume that the error surface is smooth and continuous with respect to the parameter values around the optimal parameter set, we can approximate the surface using a Taylor expansion around the optimum.

$$\begin{aligned} E(\boldsymbol{\beta}) = {} & E|_{\boldsymbol{\beta}=\mathbf{b}} + \left.\frac{\partial E}{\partial \boldsymbol{\beta}'}\right|_{\boldsymbol{\beta}=\mathbf{b}} (\boldsymbol{\beta} - \mathbf{b}) \\ & + \frac{1}{2}(\boldsymbol{\beta} - \mathbf{b})' \left.\frac{\partial^2 E}{\partial \boldsymbol{\beta}\, \partial \boldsymbol{\beta}'}\right|_{\boldsymbol{\beta}=\mathbf{b}} (\boldsymbol{\beta} - \mathbf{b}) \\ & + \text{higher order terms} \end{aligned} \quad (5)$$

Because E has a minimum at $\boldsymbol{\beta} = \mathbf{b}$ ...

$$\left.\frac{\partial E}{\partial \boldsymbol{\beta}}\right|_{\boldsymbol{\beta}=\mathbf{b}} = 0$$

and

$$\frac{\partial^2 E}{\partial \boldsymbol{\beta}\, \partial \boldsymbol{\beta}'} \quad \text{is positive definite}$$

Neglecting terms above the second derivative, and substituting equation 4 into equation 5, we have ...

$$E \approx \frac{1}{2}(\boldsymbol{\beta} - \mathbf{b})' \left.\frac{\partial^2 E}{\partial \boldsymbol{\beta}\, \partial \boldsymbol{\beta}'}\right|_{\boldsymbol{\beta}=\mathbf{b}} (\boldsymbol{\beta} - \mathbf{b}) \quad (6)$$

which describes a hyperparaboloid. Surfaces of constant E are hence hyperellipsoids.

The second derivative of the error value, given in equation 3, with respect to the parameters, is given by the following equation (Eykhoff 1974).

$$\begin{aligned} \frac{\partial^2 E}{\partial \boldsymbol{\beta}\, \partial \boldsymbol{\beta}'} = {} & \sum_{i=1}^{n} \left(\frac{\partial \mathbf{x}(t_i)}{\partial \boldsymbol{\beta}}\right)' \mathbf{W} \left(\frac{\partial \mathbf{x}(t_i)}{\partial \boldsymbol{\beta}}\right) \\ & - \sum_{i=1}^{n} \left(\frac{\partial^2 \mathbf{x}(t_i)}{\partial \boldsymbol{\beta}\, \boldsymbol{\beta}'}\right) \mathbf{W}(\mathbf{m}(t_i) - \mathbf{x}(t_i)) \end{aligned} \quad (7)$$

The second term vanishes close to the optimal parameter set, as $\lim_{\boldsymbol{\beta}\to\mathbf{b}}(\mathbf{m}(t_i) - \mathbf{x}(t_i)) = 0$.

$\partial \mathbf{x}(t)/\partial \boldsymbol{\beta}$ is given by the following equation, obtained by differentiating equation 1.

$$\frac{\mathrm{d}\frac{\partial \mathbf{x}(t)}{\partial \boldsymbol{\beta}}}{\mathrm{dt}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}(t)}{\partial \boldsymbol{\beta}} + \frac{\partial \mathbf{f}(t)}{\partial \boldsymbol{\beta}} \quad (8)$$

3. THE FISHER INFORMATION MATRIX AND OPTIMAL EXPERIMENT DESIGN

The Fisher Information Matrix (FIM), forms the basis of several criteria used in the design of optimal experiments for model identification (see Table 1.) For a derivation of the FIM, see Eykhoff (1974). In its discrete form, applicable for cases where measurements are taken at discrete sample intervals rather than continuously, the FIM may be defined as follows.

$$\mathrm{FIM} = \sum_{i=1}^{n} \left(\frac{\partial \mathbf{x}(t_i)}{\partial \boldsymbol{\beta}}\right)' \mathbf{W} \left(\frac{\partial \mathbf{x}(t_i)}{\partial \boldsymbol{\beta}}\right) \quad (9)$$

Comparing equations 7 and 9 shows that the the FIM is an approximation to the second derivative

Criterion	Formula	Interpretation
A	$\min(\mathrm{tr}(\mathrm{FIM}^{-1}))$	minimise mean variance
simplified A	$\max(\mathrm{tr}(\mathrm{FIM}))$	minimise mean variance
C	$\min(\mathrm{tr}(\mathrm{FIM}))$	minimises relative (mean) volume
D	$\max(\det(\mathrm{FIM}))$	minimises ellipsoid volume
E	$\max(\lambda_{\min}(\mathrm{FIM}))$	minimises longest axis
modified E	$\min(\mathrm{cond}(\mathrm{FIM}) = \frac{\lambda_{\max}(\mathrm{FIM})}{\lambda_{\min}(\mathrm{FIM})})$	spherical as possible

Table 1. Criteria for optimal experiment design, derived from the Fisher Information Matrix (FIM) $\lambda_{\min}$ and $\lambda_{\max}$ are the minimum and maximum eigenvalues of the FIM. The above definitions are taken from Walter and Pronzato (1990)

of the error surface, being the first term in the expression for $\frac{\partial^2 \mathbf{x}(t_i)}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}'}$.

The FIM has been used in the design of experimental conditions for estimating parameters of batch fermentations (Yoo *et al.* 1986) and fed-batch fermentations (Kalogerakis and Luus 1984). Munack (1989) has shown that fed-batch fermentations are better, from the point of view of identifiability, than batch fermentations and his work was focussed on seeking out an input trajectory that made identification as robust as possible.

4. USING GENETIC ALGORITHMS TO SEARCH FOR OPTIMAL INPUTS

The FIM depends on the model structure and parameter set used, on the input applied to the fermentation, and, in its discrete form, on the sampling interval used in obtaining measurements. Given a model structure and an estimate of its parameters, if the sampling rate is fixed, an optimal experiment may be designed by seeking out the input profile which maximises one of the design criteria given in Table 1.

Munack (1989) stated that the gradient technique used in searching for an optimal input profile may have stopped in a suboptimal point, thereby finding a good local optimum, but not necessarily a global optimum. Genetic algorithms (GAs) have been shown to behave well on multimodal functions (Goldberg 1989), being less likely to become stuck in local optima than conventional optimisation techniques. GAs only need to calculate the objective function in the course of their search - no use is made of derivatives, and GAs may be used where the search surface is neither smooth nor differentiable.

In order to use genetic algorithms, the problem to be solved must first be encoded as a string, which the genetic algorithm acts on as it searches. In each generation, the algorithm evaluates the fitness of every string in the population, mates the strings according to their fitnesses (reproduction), exchanges information between pairs of strings randomly (crossover), and finally changes a small number of string elements with a low probability (mutation).

Here we are using GAs to search for fermentation inputs which optimise the experiment design according to the D and modified E criteria, two of the more commonly used criteria. (In our work, we have attempted to maximise the reciprocal of the condition number, rather than minimising the condition number itself–the two approaches are equivalent.) For use with the genetic algorithm, our input profile has been divided into a stepped input, with discrete portions having constant value. In this way, the input pattern is only determined by the values of the alleles being modified by the genetic algorithm. This stepped input profile is simple to specify using computer control, and may be applied by hand. The length of each 'step' in the input profile was chosen to be five hours as this was considered reasonable for manual implementation, should that be needed. The choice of a five hour interval also reduces the length of the strings and hence the size of the search space being searched by the genetic algorithm.

4.1 *Example - the Penicillin Fermentation*

The results of computer studies aimed at finding the best input for use in identifying the parameters of a penicillin fermentation model derived from that of Paul and Thomas (1996), given in Table 2, are presented here. This model is used to illustrate the application of the GA input optimisation technique to a complex fermentation model, for which conventional optimal control related methods would prove mathematically involved.

The parameter set used in the model is given in Table 3.

In this work, the Genetic and Evolutionary Algorithm Toolbox for MATLAB (Pohlheim 1996) was used. An initial population size of 100, with 3 subpopulations was selected, running over a maximum of 100 generations, with the feasible input profiles bounded between 0 and 3.5/160 dm^3/hr. The upper input limit was set to be the rate which would fill the working volume of the fermenter ($7\mathrm{dm}^3$ in a $10\mathrm{dm}^3$ fermenter) over the course

$$\frac{dX_0}{dt} = \frac{\mu_0 X_1 S}{K_0 + S} - \frac{\gamma_1 X_0}{K_1 + S} \quad (10)$$

$$\frac{dX_1}{dt} = \frac{\mu_e X_1 S}{K_e + S} - \frac{\mu_0 X_1 S}{K_0 + S} + \frac{\gamma_1 X_0}{K_1 + S} - \mu_2 X_2 \rho \quad (11)$$

$$\frac{dX_2}{dt} = \mu_1 v_{ic} - \mu_2 X_2 + \mu_3 X_2 \quad (12)$$

$$\frac{dX_3}{dt} = \mu_2 X_2 \rho - \mu_a X_3 \quad (13)$$

$$\frac{dX_4}{dt} = \mu_a X_3 \quad (14)$$

$$\frac{dS}{dt} = -\frac{\alpha_0 \mu_0 X_1 S}{K_0 + S} - \frac{\alpha_e \mu_e X_0 S}{K_e + S} - \frac{m_0 X_0 S}{K_1 + S} - \frac{m_1 \rho_c v_{ic} S}{K_2 + S} - \frac{\alpha_p \mu_p \rho_c v_{ic} S}{K_P + S(1 + S/K_I)} - \frac{dL}{dt} \quad (15)$$

$$\frac{dL}{dt} = -\frac{\mu_L L(X_0 + X_1)}{(K_{SL} + L)(1 + S/K_{SI})} \quad (16)$$

$$\frac{dP}{dt} = \frac{\mu_p \rho_c v_{ic} S}{K_P + S(1 + S/K_I)} - \mu_h P \quad (17)$$

$$\frac{dV}{dt} = F \quad (18)$$

Table 2. Model of Paul and Thomas (1996), simplified to increase simulation speed (equations 11 to 13), and extended to consider lactose present at the start of the fermentation (equations 15 and 16), X_* – morphologically distinguished biomass fractions, S – glucose concentration, L – glucose concentration, P – penicillin concentration, V – broth volume, F – input feed rate, for clarity, dilution terms have been omitted

Parameter	Valu e	Parameter	Value
μ_0	0.0333	K_e	0.0788
μ_1	0.0193	K_p	0.0413
μ_2	0.0535	K_i	0.3656
μ_3	2.026e-3	α_0	1.85
μ_e	0.4092	α_e	1.83
μ_a	0.0437	α_p	0.85
μ_p	0.0287	m_0	0.0256
μ_l	0.241	m_1	0.0242
μ_h	0.0028	K_{SL}	0.0864
K_0	0.0352	K_{SI}	1.099e-3
K_1	0.0905	γ_1	0.0122
K_2	0.3017		

Table 3. Table of parameters used in simulating the model

of the fermentation (assumed to be 160 hours), starting from a typical initial volume ($3.5 dm^3$).

5. RESULTS

Both D and modified E optimal experiment designs improved over the course of the 100 generations used, giving quite distinct input profiles at the end. Figure 1 shows the D and modified E input designs produced, along with results of simulations performed using these input profiles. In Table 4, the designed inputs' values for both

Criterion	Constant Input	D-optimal Input	E-optimal Input
D	1.64e4	5.18e24	3.00e19
Modified E	1.30e20	1.55e19	1.30e19

Table 4. Values of design criteria obtained using GA-designed input profiles

criteria are compared with the values for a typical constant input feed profile. The two designed inputs produce better values for the criterion for which they were designed than does the constant feed profile (a greater D criterion value for the D-optimal design, and a greater E criterion value for the E-optimal design).

However, the graphs of the simulated fermentation results suggest that a practical fermentation carried out using the D-optimal input design could run into difficulties with low oxygen concentration from around 40 hours on. That the 'optimal' design gives rise to what may be a practically impossible situation could be because either the model does not describe the dissolved oxygen concentration or the maximum feed profile permitted is excessive.

The fact that both designed inputs produce 'better' values for the criterion for which they were not designed than for the constant input feed profile may be because the two experiment design criteria compared in this work are not entirely independent ($\det \mathrm{FIM} = \prod_{\lambda=\lambda_{\min}}^{\lambda=\lambda_{\max}} \lambda$). Geometrically, the D criterion is attempting to minimise the volume of the confidence ellipsoids, whilst the E criterion is attempting to improve the 'roundness' of the same ellipsoids.

6. CONCLUSIONS

Genetic algorithms have been shown to be useful in designing optimal experiments for parameter estimation for complex, nonlinear fermentation models, for which optimal control based approaches to experiment design could prove involved.

In the future it is intended that the E-optimal experiment design will be implemented; as the current D-optimal design may encounter practical difficulties, this should not be implemented. There is also scope for work investigating the design of experiments using scaled parameter values, so as to obtain designs with equal percentage errors, as opposed to equal absolute error magnitudes, for all of the parameters.

Acknowledgement: The financial support of the BBSRC is gratefully acknowledged.

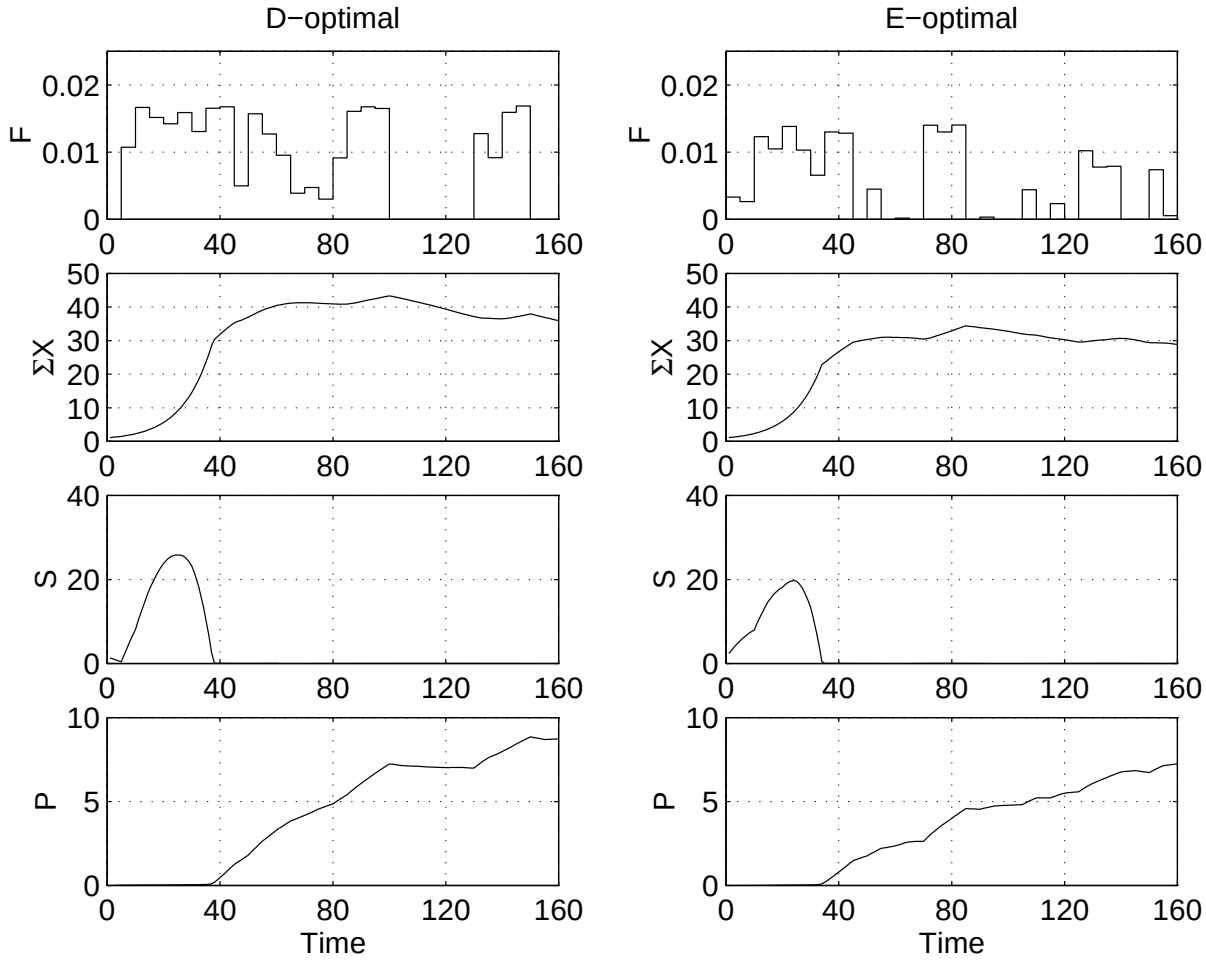

Fig. 1. Simulation results for D and modified E designs, F – input feed rate in dm^3/hr, ΣX_* – total biomass concentration, S – glucose concentration, P – penicillin concentration, all concentrations in gm/l

7. REFERENCES

di Massimo, Christine, Paul A. Lant, Aidan Saunders, Gary A. Montague, Ming T. Tham and A. Julian Morris (1992). Bioprocess applications of model-based estimation techniques. *Journal of Chemical Technology and Biotechnology* **53**, 265–277.

Eykhoff, Pieter (1974). *System Identification*. Wiley.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimisation, and Machine Learning*. Addison-Wesley. New York.

Holmberg, Andrea (1982). On the practical identifiability of microbial growth models incorporating Michaelis-Menten type nonlinearities. *Mathematical Biosciences* **62**, 23–43.

Holmberg, Andrea and Jukka Ranta (1982). Procedures for parameter and state estimation of microbial growth process models. *Automatica* **18**, 181–193.

Kalogerakis, Nicolas and Rein Luus (1984). Sequential experimental design of dynamic systems through the use of information index. *Canadian Journal of Chemical Engineering* **62**, 730–737.

Lim, H. C., Y. J. Tayelo, J. M. Modak and P. Bonte (1986). 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**, 1408–1420.

Munack, Axel (1989). Optimal feeding strategy for identification of Monod-type models by fed-batch experiments. In: *Computer Applications in Fermentation Technology*. Elsevier. pp. 195–203.

Nihtilä, Markku and Jouko Virkunnen (1977). Practical identifiability of growth and substrate consumption models. *Biotechnology and Bioengineering* **19**, 1831–1850.

Paul, G. C. and C. R. Thomas (1996). A structured model for hyphal differentiation and penicillin production using *Penicillium chrysogenum*. *Biotechnology and Bioengineering* **51**, 558–572.

Pohlheim, H. (1996). Genetic and evolutionary algorithm toolbox for use with MATLAB - documentation. Technical report. Technical Uni-

versity Ilmenau.
(http://www.systemtechnik.tu-ilmenau.de/~pohlheim/GA_Toolbox/index.html).

San, Ka-Yiu and Gregory Stephanopoulos (1989). Optimization of fed-batch penicillin fermentation: A case of singular optimal control with state constraints. *Biotechnology and Bioengineering* **34**, 72–78.

van Impe, J. F. and G. Bastin (1995). Optimal adaptive control of fed-batch fermentation processes. *Control Engineering Practice* **3**, 939–954.

Yoo, Young J., M. Marino-Galarraga, J. Hong and R.T. Hatch (1986). Experimental design for parameter estimation from batch culture. *Biotechnology and Bioengineering* **28**, 836–841.

BIBLIOGRAPHY

Agarwal, A. K. and Brisk, M. L. (1985). Sequential experimental design for precise parameter estimation 1. use of reparameterization. *Industrial Engineering Chemistry Process Design and Development*, 24:203–207.

Bajpai, R. and Reuß, M. (1980). A mechanistic model for penicillin production. *Journal of Chemical Technology and Biotechnology*, 30:332–344.

Bajpai, R. and Reuß, M. (1981). Evaluation of feeding strategies in carbon-regulated secondary metabolite production through mathematical modelling. *Biotechnology and Bioengineering*, 23:717–738.

Baltes, M., Schneider, R., Sturm, C., and Reuss, M. (1994). Optimal experimental design for parameter estimation in unstructured growth models. *Biotechnology Progress*, 10:480–488.

Benedict, R., Schmidt, W., Coghill, R., and Oleson, A. (1945). Penicillin iii, the stability of penicillin in aqueous solution. *Journal of Bacteriology*, 49:85–95.

Bilardello, P., Joulia, X., Lann, M. L., Delmas, H., and B.Koehret (1993). A general strategy for parameter estimation in differential-algebraic systems. *Computers in Chemical Engineering*, 17:517–525.

Bronshtein, I. and Semandyayev, K. (1985). *Handbook of Mathematics*. van Nostrand Reinhold, english third edition.

Cagney, J. W., Chittur, V. K., and Lim, H. C. (1984). Use of filtration measurements for estimation of cellular activity in penicillin production. *Biotechnology and Bioengineering Symposium*, 14:619–634.

Cazzador, L. and Lubenova, V. (1995). Nonlinear estimation of specific growth rate for aerobic fermentation processes. *bb*, 47:626–632.

Chappell, M., Godfrey, K., and Vajda, S. (1990). Global identifiability of the parameters of nonlinear systems with specified inputs: A comparison of methods. *Mathematical Biosciences*, 102:41–73.

Chemical Market Reporter (1998). Vol. 153, No.10.

Contois, D. (1959). Kinetics of bacterial growth: Relationship between population density and specific growth rate of continuous culture. *Journal of General Microbiology*, 21:40–50.

Davidor, Y. (1990). *Genetic Algorithms and Robotics*. Addison-Wesley.

De Jong, K. A. (1993). Genetic algorithms are NOT function optimisers. In Whitley, L. D., editor, *Foundations of Genetic Algorithms 2*, San Mateo, CA. Morgan Kaufmann.

di Massimo, C., Lant, P. A., Saunders, A., Montague, G. A., Tham, M. T., and Morris, A. J. (1992). Bioprocess applications of model-based estimation techniques. *Journal of Chemical Technology and Biotechnology*, 53:265–277.

Espie, D. and Macchietto, S. (1989). The optimal design of dynamic experiments. *American Institute of Chemical Engineers*, 35:223–229.

Eykhoff, P. (1974). *System Identification*. Wiley.

Farza, M., Othman, S., Hammouri, H., and Bilton, J. (1997). A nonlinear approach for the on-line estimation of the kinetic rates in bioreactors. *Bioprocess Engineering*, 17:143–150.

Fishman, V. M. and Biryukov, V. V. (1974). Kinetic model of secondary metabolite production and its use in computation of optimal conditions. *Biotechnology and Bioengineering Symposium*, 4:647–662.

GALESIA (1995). *Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA '95*. Institution of Electrical Engineers.

GALESIA (1997). *Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA '97*. Institution of Electrical Engineers.

Galvanauskas, V., Simutis, R., and Lübbert, A. (1997). Model-based design of a biochemical processes: Simulation studies and experimental tests. *Biotechnology Letters*, 19:1043–1047.

Galvanauskas, V., Simutis, R., Volk, N., and Lübbert, A. (1998). Model based design of a biochemical cultivation process. *Bioprocess Engineering*, 18:227–234.

Gattu, G. and Zafiriou, E. (1995). Observer based nonlinear quadratic dynamic matrix control for state space and input/output models. *Canadian Journal of Chemical Engineering*, 73:883–895.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimisation, and Machine Learning*. Addison-Wesley, New York.

Hardwicke, P. I., Kent, C. A., Norton, J. P., and Veres, S. M. (1991). A robust procedure for trend and model estimation during fermentation. *Process Biochemistry*, 26:269–273.

Heijnen, J. J., Roels, J. A., and Stouthamer, A. H. (1979). Application of balancing methods in modeling the penicillin fermentation. *Biotechnology and Bioengineering*, 21:2175–201.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press. First Edition ©1975 Massachusetts Institute of Technology.

Holmberg, A. (1982). On the practical identifiability of microbial growth models incorporating michaelis-menten type nonlinearities. *Mathematical Biosciences*, 62:23–43.

Hosten, L. and Emig, G. (1975). Sequential experimental design procedures for precise parameter estimation in ordinary differential equations. *Chemical Engineering Science*, 30:1357–1364.

Hosten, L. H. (1974). A sequential experimental design procedure for precise parameter estimation based upon the shape of the joint confidence region. *Chemical Engineering Science*, 29:2247–2252.

Jacquez, J. A. and Greif, P. (1985). Numerical parameter identifiability and estimability: Integrating identifiability, estimability, and optimal sampling design. *Mathematical Biosciences*, 77:201–227.

Kalogerakis, N. and Luus, R. (1984). Sequential experimental design of dynamic systems through the use of information index. *Canadian Journal of Chemical Engineering*, 62:730–737.

King, R. (1997). A structured mathematical model for a class of organisms: 1. Development of a model for *streptomyces tendae* and application of model-based control. *Journal of Biotechnology*, 52:219–234.

Kluge, M., Siegmund, D., Diekman, H., and Thomas, M. (1992). A model for penicillin production with and without temperature shift after the growth phase. *Applied Microbiology and Biotechnolog and Biotechnologyy*, 36:446–451.

Kolchin, E. R. (1973). *Differential Algebra and Algebraic Groups*. Academic Press.

Lee, J. H. and Ricker, N. L. (1994). Extended Kalman filter based nonlinear model predictive control. *Industrial and Engineering Chemical Research*, 33:1530–1541.

Lim, H. C., Tayeb, Y. J., Modak, J. M., and Bonte, P. (1986). Computational algorithms for optimal feed rates for a class of fed-batch fermentation: Numerical results for penicillin and cell mass production. *Biotechnology and Bioengineering*, 28:1408–1420.

Ljung, L. and Glad, T. (1994). On global identifiability for arbitrary model parametrizations. *Automatica*, 30(2):265–276.

Louis, S. J. and Rawlins, G. J. E. (1993). Syntactic analysis of convergence in genetic algorithms. In Whitley, L. D., editor, *Foundations of Genetic Algorithms*, San Mateo, CA. Morgan Kaufmann.

Luus, R. (1992). Optimization of fed-batch fermentors by iterative dynamic programming. *Biotechnology and Bioengineering*, 41:599–602.

Megee, R. D., Kinoshita, S., Fredrickson, A. G., and Tsuchiya, H. M. (1970). Differentiation and product formation in molds. *Biotechnology and Bioengineering*, 12:771–801.

Menezes, J. C., Alves, S. S., Lemos, J. M., and de Azevedo, S. F. (1994). Mathematical modelling if industrial pilot-plant penicillin-g fed-batch fermentations. *Journal of Chemical Technology and Biotechnology*, 61:123–138.

Monod, J. (1942). *Recherches sur la Croissance des Cultures Bactériennes*. Hermann, Paris.

Montague, G. A., Morris, A. J., Wright, A. R., Aynsley, M., and Ward, A. (1986). Modelling and adaptive control of fed-batch penicillin fermentation. *Canadian Journal of Chemical Engineering*, 64:567–580.

Montesinos, J., Lafuente, J., Gordillo, M., Valero, F., Folà, C., Charbonnier, E., and Cheruy, A. (1995). Structured modelling and state estimation in a fermentation process: lipase production by *candida rugosa*. *Biotechnology and Bioengineering*, 48:573–584.

Munack, A. (1989). Optimal feeding strategy for identification of monod-type models by fed-batch experiments. In *Computer Applications in Fermentation Technology*, pages 195–203. Elsevier.

Murray, L. E. and Reiff, Jr., E. K. (1984). Design of transient experiments for identification of fixed bed thermal transport propoerties. *Canadian Journal of Chemical Engineering*, 62:55–61.

Myers, M. A., Kang, S., and Luecke, R. H. (1996). State estimation and control for systems with delayed off-line measurements. *Computers in Chemical Engineering*, 5:585–588.

Náhlík, J. and Burianec, Z. (1988). On-line parameter and state estimation of continuous cultivation by extended Kalman filter. *Applied Microbiology and Biotechnology*, 28:128–134.

Nestaas, E. and Wang, D. I. C. (1983). Computer control of the penicillin fermentation using the filtration probe in conjunction with a structured process model. *Biotechnology and Bioengineering*, 25:781–796.

Nicolai, B. M., van Impe, J. F., Vanrolleghem, P. A., and Vandewalle, J. (1991). A modified unstructured mathematical model for the penicillin-G fed-batch fermentation. *Biotechnology Letters*, 13:489–494.

Nielsen, J. (1992). Modelling of microbial kinetics. *Chemical Engineering Science*, 47:4225–4270.

Nielsen, J., Nikolajsen, K., and Villadsen, J. (1991). Structured modelling of a microbial system: 2. experimental verification of a structured lactic-acid fermentation model. *Biotechnology and Bioengineering*, 38(1):11–23.

Nielsen, J. and Villadsen, J. (1994). *Bioreaction Engineering Principles.* Plenum Press, New York.

Norton, J. P. (1986). *An Introduction to Identification.* Academic Press, first edition.

Paul, G. C. (1996). Personal communication.

Paul, G. C. (1998). Personal communication.

Paul, G. C., Kent, C. A., and Thomas, C. R. (1994). Image analysis for characterising differentiation of *penicillium chrysogenum. Transactions of the Institute of Chemical Engineers (Part C)*, 72:95–105.

Paul, G. C., Syddall, M. T., Kent, C. A., and Thomas, C. R. (1998). A structured model for penicillin production on mixed substrates. *Biochemical Engineering Journal.* In press.

Paul, G. C. and Thomas, C. R. (1996). A structured model for hyphal differentiation and penicillin production using *penicillium chrysogenum. Biotechnology and Bioengineering*, 51:558–572.

Pinto, J. C., Lobão, M. W., and Monteiro, J. L. (1990). Sequential experimental design for parameter estimation: A different approach. *Chemical Engineering Science*, 45:883–892.

Pinto, J. C., Lobão, M. W., and Monteiro, J. L. (1991). Sequential experimental design for parameter estimation: Analysis of relatvie deviations. *Chemical Engineering Science*, 46:3129–3138.

Pohjanpalo, H. (1978). System identifiability based on the power series expansion of the solution. *Mathematical Biosciences*, 41:21–33.

Pohlheim, H. (1996). Genetic and evolutionary algorithm toolbox for use with MATLAB - documentation. Technical report, Technical University Ilmenau. (http://www.systemtechnik.tu-ilmenau.de/~pohlheim/GA_Toolbox/index.html) .

Pons, M. N., Rajab, A., Flaus, J. M., Engasser, J. M., and Cheruy, A. (1988). Comparison of estimation methods for biotechnological processes. *Chemical Engineering Science*, 43:1909–1914.

Primrose, S. B., editor (1987). *Modern Biotechnology*. Blackwell Scientific Publications.

Ramkrishna, D., Fredrickson, A. G., and Tsuchiya, H. M. (1967). Dynamics of microbial propagation: Models considering inhibitors and variable cell composition. *Biotechnology and Bioengineering*, 9:129–170.

Ray, W. H. (1989). *Advanced Process Control*. Butterworth.

Ritt, J. F. (1950). *Differential Algebra*. American Mathematical Society.

Rodrigues, J. A. D. and Filho, R. M. (1996). Optimal feed rates strategies with operating constraints for the penicillin production process. *Chemical Engineering Science*, 51:2859–2864.

San, K.-Y. and Stephanopoulos, G. (1989). Optimization of fed-batch penicillin fermentation: A case of singular optimal control with state constraints. *Biotechnology and Bioengineering*, 34:72–78.

Schügerl, K. (1986). Modelling of biotechnical processes. In *Modelling and Control of Biotechnological Processes*, IFAC Proceedings Series, pages 13–32. International Fedaration of Automatic Control, Pergamon Press.

Shi, Y. and Yuan, W.-K. (1988). Application of adaptive estimation in microbial fermentation processes. *Chemical Engineering Science*, 43:1915–1920.

Simutis, R. and Lübbert, A. (1997). A comparative study on random search algorithms for biotechnical process optimization. *Journal of Biotechnology*, 52:245–256.

Syddall, M., Paul, G., and Kent, C. (1998). Improving the estimation of parameters of penicillin fermentation models. In *Proceedings of the 7th Conference on Computer Applications in Biotechnology*.

Tarbuck, L. A., Ng, M. H., Tampion, J., and Leigh, J. R. (1986). Development of strategies for online estimation of biomass and secondary product formation in growth-limited batch fermentations. *IEE Proceedings Part D*, 5:235–239.

Tiller, V., Meyerhoff, J., Sziele, D., Schügerl, K., and Bellgardt, K.-H. (1994). Segregated mathematical model for the fed-batch cultivation of a high-producing strain of *penicillium chrysogenum*. *Journal of Biotechnology*, 34:119–131.

Vajda, S. and Rabitz, H. (1989). State isomorphism approach to global identifiability of nonlinear systems. *IEEE Transactions on Automatic Control*, 34:220–223.

Vajda, S., Rabitz, H., Walter, E., and Lecourtier, Y. (1989). Qualitative and quantative identifiability analysis of nonlinear chemical kinetic models. *Chemical Engineering Communications*, 83:191–219.

van Impe, J. F. and Bastin, G. (1995). Optimal adaptive control of fed-batch fermentation processes. *Control Engineering Practice*, 3:939–954.

van Impe, J. F., Nicolai, B. M., Vanrolleghem, P. A., Spriet, J. A., de Moor, B., and Vandewalle, J. (1992). Optimal control of the penicillin G fed-batch fermentation: An analysis of a modified unstructured model. *Chemical Engineering Communications*, 117:337–353.

van Suijdam, J. C., Hols, H., and Kossen, N. W. F. (1982). Unstructured model for growth of mycelial pellets in submerged cultures. *Biotechnology and Bioengineering*, 24:177–191.

Versyck, K. J., Claes, J. E., and Impe, J. F. V. (1997). Practical identification of unstructured growth kinetics by application of optimal experiment design. *Biotechnology Progress*, 13:524–531.

Walter, E. and Pronzato, L. (1990). Qualitative and quantitative experiment design for phenomenological models, a survey. *Automatica*, 25:195–213.

Weuster-Botz, D., Pramatarova, V., Spassov, G., and Wandrey, C. (1995). Use of a genetic algorithm in the development of a synthetic

growth medium for *arthrobacter simplex* with high hydrocortisone Δ-dehydrogenase activity. *Journal of Chemical Technology and Biotechnology*, 64:386–392.

Whitley, L. D., editor (1993). *Foundations of Genetic Algorithms 2*, San Mateo, CA. Morgan Kaufmann.

Whitley, L. D. and Vose, M. D., editors (1995). *Foundations of Genetic Algorithms 3*, San Francisco, CA. Morgan Kaufmann.

Yao, L. and Sethares, W. A. (1994). Nonlinear parameter estimation via the genetic algorithm. *IEEE Transactions on Signal Processing*, 42:927–937.

Yoo, Y. J., Marino-Galarraga, M., Hong, J., and Hatch, R. (1986). Experimental design for parameter estimation from batch culture. *Biotechnology and Bioengineering*, 28:836–841.