

# **Optimization and control of Fed-batch Fermentation Processes by Using Artificial Neural Systems**

Catalina Valencia Peroni



**Departament D'Enginyeria Química  
Escola Tècnica Superior D'Enginyeria Química  
Universitat Rovira I Virgili**

The copyright of the preprints in the annex of this report belongs to the journals where they will be published. Anyone wishing to reproduce partially or totally those preprints should ask permission to the copyright owner.

© 2002

Programa de doctorat: *Enginyeria Química*

Bienni: 1998-2000

**Control of Fed-batch Fermentation Processes by Using Artificial Neural Systems**

Memòria presentada per na Catalina Valencia Peroni per a optar al títol de Doctor en  
Enginyeria Química

**Certifiquem** que la present memòria ha estat realitzada sota la nostra direcció i que tots els resultats presentats i la seva anàlisi son fruit de la investigació realitzada per l'esmentat doctorant. Tarragona, Catalunya, Spain, Octubre 2002.

Francesc Giralt Prat  
Catedràtic  
Departament D'Enginyeria Química  
Universitat Rovira i Virgili

Jaume Giralt Marcé  
Catedràtic  
Departament D'Enginyeria Química  
Universitat Rovira i Virgili

## Abstract

This work focuses on the application of neural networks in the areas of modelling, identification, control and optimization of biotechnology processes, mainly fed-batch bioreactors. The basic ideas and techniques of artificial neural networks are presented with the notation familiar to control engineers. The applications of a variety of neural network architectures in control and control schemes are first surveyed. Some specific fed-batch bioreactor processes are mentioned to illustrate particular control cases to be examined in detail and solved. Specifically, a non-linear multivariable bioreactor control problem is used as a case study for model based control techniques. An implementation of direct and inverse process control models based on neural networks that considers biological, thermal and pH effects for this multivariable fed-batch bioreactor is performed and tested. Multilayer perceptrons and radial basis functions neural networks are considered to model this type of non-linear multi-input multi-output (MIMO) dynamic process. The direct models are successfully tested under steady state, dynamic process operation and when a acid disturbance in the process causes a plant/model mismatch. The inverse process model is also successfully tested at the set-point input with a random series of perturbations around the plant operation state. The RBF architecture with goal 3.0 is the best architecture for the direct model of this multivariable process while the best inverse model is based on a MLP 19-11-7-1 trained including past information of the steady states of the process.

On the other hand, optimal control techniques that employ neural networks are studied to optimize the production of invertase in a fed-batch bioreactor. The controlled addition of substrates is used in this bioreactor process to increase productivity when end-product inhibition or catabolite repression are present. Cloned invertase production in *Saccharomyces cerevisiae* yeast is carried out in fed-batch mode of operation because the enzyme expression is repressed at high glucose concentrations. An optimal glucose feed rate profile is needed to achieve the highest fermentation profit. The controller has to find at each time step an optimal control action that increments the fed-batch bioreactor profitability, even when a disturbance or a set-point change arise. This optimal control action increases the productivity and, within the same optimization process, finds the optimal fermentation ending time. This double optimization is a novelty not met by previous optimization schemes published in the literature. A neuro dynamic programming (NDP) approach coupled with MLP neural networks or fuzzy ARTMAP systems is employed to accomplish these optimization objectives. Fuzzy ARTMAP creates multidimensional category maps by incremental supervised learning. The optimization method utilizes suboptimal control policies as a starting guess. The neural networks are used to build a cost surface in the state space visited by the process. Bellman's iteration is used to improve the cost approximation. The cost surface obtained is implemented into a control system. The controller is tested for different fermentation processes started

with different initial fermentation volumes. NDP outperforms other optimization methods employed to find an optimal feeding profile. Besides, it can be used to optimize any fermentation process (starting at different initial conditions) because the future costs (profits) are characterized as a function of system states. The optimal control trajectories found by the controller are similar to the best suboptimal policy for each initial volume. MLP-NDP controllers yield the highest profits, but the manipulated variable trajectories are not smooth. Fuzzy ARTMAP-NDP overcomes this limitation. The best fuzzy ARTMAP-NDP based control system is also tested when an abrupt death of yeast cells occurs. In this case, the controller performance is better than the performance of the fermentation using the best suboptimal policy for the given initial volume.

The integration of control science with neural networks in a unified presentation and identification key areas is a path to follow in future research. Artificial neural networks techniques can be successfully applied to control fed-batch bioreactors.

## **Resumen**

### **Optimización y Control de Procesos de Fermentación Fed-batch a través de Sistemas Neuronales Artificiales**

Los procesos de fermentación son ampliamente utilizados en la industria química, farmacéutica y alimentaria. La producción de comida para animales, yogures, quesos, cerveza, colorantes para comida, fertilizantes, medicinas terapéuticas y penicilina, entre otros, son algunos ejemplos de procesos biotecnológicos. En una búsqueda rápida en Internet, cerca de 113000 sitios fueron encontrados, todos ellos relacionados con productos de procesos de fermentación.

El campo de aplicación de la bioingeniería abarca desde los procesos tradicionales de fermentación de vinos hasta la industrialización de no solo la producción de cerveza, queso y leche; sino también de nuevos productos biotecnológicos como son los antibióticos, enzimas, hormonas, vitaminas, azúcares y ácidos orgánicos. Desde tiempos remotos los microorganismos han sido utilizados por el hombre en la producción de alimentos esenciales, como el pan o el queso. El arte de hacer vino ha pasado de padre a hijo desde el antiguo Egipto. Solo hasta finales del siglo XIX, gracias a los estudios del químico y microbiólogo Louis Pasteur, nació la biotecnología como ciencia. La definición actual de biotecnología, según la OCDE, es la aplicación de la ciencia y la tecnología tanto a organismos vivos como a partes, productos y modelos de ellos, con el propósito de modificar tanto materia viva como la no viva con el fin de producir conocimiento, bienes y servicios.

Un proceso de fermentación es un proceso químico en el cual se emplean microorganismos para obtener un producto en particular, aprovechando la selectividad de los microorganismos para producir un determinado compuesto. Los procesos de fermentación son llevados a cabo en un bioreactor. Un bioreactor es un recipiente en el cual microorganismos son cultivados de manera controlada y/o materia prima es convertida o transformada debido a reacciones biológicas.

En una fermentación, es necesario un control adecuado de todas las variables de proceso, debido a que cualquier cambio inesperado en el valor de alguna de ellas puede afectar el desarrollo de los microorganismos y en consecuencia disminuir la productividad del bioreactor. El principal objetivo de controlar un proceso de fermentación es maximizar la producción de microorganismos u otros compuestos metabólicos. Avances recientes en ingeniería genética han aumentado la importancia del adecuado control de los procesos biotecnológicos. El uso de células de mamíferos o microorganismos en la producción de moléculas complejas requiere el análisis y control de todas las variables de proceso, tales como temperatura, concentración de oxígeno y pH.

El presente trabajo se centra en la aplicación de redes neuronales artificiales en las áreas de modelado, identificación, control y optimización de procesos biotecnológicos, principalmente

en bioreactores del tipo fed-batch. Un bioreactor fed-batch se emplea cuando la producción de determinado compuesto de interés, es inhibida debido a la alta concentración de sustrato. En un proceso fed-batch, la fermentación empieza con un volumen, concentración de microorganismos y sustrato determinados y, a medida que transcurre el proceso de fermentación, el sustrato se añade poco a poco, hasta que se consigue llenar el bioreactor.

En este trabajo las ideas y técnicas utilizadas por las redes neuronales artificiales son presentadas con la notación familiar para un ingeniero de control. Diferentes estructuras de redes neuronales artificiales y su posible aplicación a diferentes sistemas de control son resumidas. También son presentados algunos procesos de fermentación fed-batch. Dichos procesos son empleados para ilustrar casos específicos de problemas de control. Específicamente, un modelo no lineal y multivariable de un bioreactor es empleado para ilustrar las técnicas de control basadas en el modelo del proceso. Un modelo para la producción de invertasa a través de la levadura *Saccharomyces cerevisiae* es empleado para ilustrar las técnicas de optimización y control.

Dentro de las técnicas de control basadas en el modelo del proceso, se implementó un modelo directo y uno inverso de la fermentación multivariable antes mencionada. Ambos modelos, basados en redes neuronales artificiales, consideran efectos biológicos, térmicos y de pH. Multilayer perceptron y Radial Basis Function son las redes neuronales empleadas para la construcción de los dos modelos. Para ilustrar la fiabilidad de estos modelos, diferentes pruebas les fueron realizadas. El modelo directo del proceso de fermentación, basado en redes neuronales, fue probado en operación en estado estacionario, en estado dinámico y cuando una perturbación en el ácido causa que el pH del proceso sea diferente. El modelo inverso del proceso de fermentación fue probado haciendo cambios aleatorios en el punto de referencia. Con la arquitectura Radial Basis Function se obtuvo el mejor modelo directo. Para el modelo inverso del proceso de fermentación, se encontró que la mejor arquitectura es la multilayer perceptron 11-7-1, entrenada con información de los estados estacionarios del proceso.

Por otro lado, para la optimización de la producción de invertasa es necesario encontrar el perfil de alimentación óptimo, de manera que la productividad del bioreactor sea máxima y el tiempo de fermentación sea mínimo. Este doble objetivo de optimización es una novedad y no ha sido antes obtenido por otros esquemas de optimización previamente publicados. El objeto del controlador debe ser hallar a cada instante de tiempo la acción óptima de control, es decir, cada vez encontrar cual es el flujo de alimentación adecuado para cumplir el doble objeto de la optimización. En este trabajo se utiliza la programación dinámica neuronal (NDP) con el fin de implementar dicho controlador. Esta técnica emplea redes multilayer perceptron o fuzzy ARTMAP para realizar la optimización del proceso. NDP utiliza perfiles de alimentación subóptimos como suposición inicial. A través de esta suposición, una red neuronal es empleada para construir la superficie de costos en el espacio de los estados del proceso. Esta

superficie de costos se mejora a través de la iteración de Bellman. Una vez obtenida una buena aproximación a la superficie de costos óptima, esta es implementada en un sistema de control que hace uso también de la ecuación de Bellman. El controlador es probado en diferentes condiciones de operación del proceso de fermentación, específicamente cuando la fermentación comienza con diferentes volúmenes iniciales. Al comparar la metodología empleada se encontró que esta es mejor que otros métodos de optimización utilizados con el mismo fin, debido a que la metodología NDP puede ser usada en diferentes procesos de fermentación sin necesidad de realizar una optimización on-line. Las trayectorias óptimas encontradas por el controlador son similares a la trayectoria seguida por el mejor de los perfiles subóptimos. Con multilayer perceptron- NDP se obtienen los mas altos rendimientos pero la trayectoria de variable manipulada es muy abrupta. Con fuzzy ARTMAP-NDP no se presenta este problema. El controlador que implementa fuzzy ARTMAP-NDP es probado también cuando hay un cambio brusco en la concentración de células. El 50% mueren. En este caso el desempeño del controlador es mejor que el rendimiento de la fermentación cuando la mejor de los perfiles de alimentación subóptimas es utilizado. Por último se puede decir que la integración de la ingeniería de control con las redes neuronales es un fructífero camino a seguir por futuras líneas de investigación ya que las redes neuronales artificiales pudieron ser empleadas con éxito en el control de bioreactores fed-batch.

### **Publicaciones**

Valencia C., Giralt J., Arenas A., Giralt F., Implementation of a non-linear multivariable (MIMO) process control model of a fed-batch bioreactor with neural networks. Poster Session: Topics in systems and process control, AIChE annual meeting Reno 2001.

Valencia C., Giralt J., Arenas A., Giralt F., Non-linear multivariable (MIMO) process control model of a fed-batch bioreactor with neural networks. Submitted to Chemical Engineering Science. 2002

Valencia C., Lee J.H., Kaisare N.S., Final time and productivity optimization of a fed-batch bioreactor for invertase production. Presented at Control of Pharmaceutical and Biological Processes Session 347, AIChE annual meeting, Indianapolis 2002.

Valencia C., Giralt J., Arenas A., Giralt F., Optimization of invertase production in a fed-batch bioreactor using dynamic programming coupled with fuzzy ARTMAP, to be submitted to Biotechnology and Bioengineering 2002



## **Resumen**

### **Optimització i Control dels Processos de Fermentació Fed-batch a través de Sistemes Neuronals Artificials**

Els processos de fermentació són amplament utilitzats en l'indústria química, farmacèutica i alimentaria. La producció de menjar per a animals, iogurts, formatge, cervesa, colorants per aliments, fertilitzants, medicines terapèutiques i penicil·lina entre altres, són alguns exemples de processos biotecnològics. En una recerca ràpida a Internet, cerca de 113000 llocs van ser trobats, tots ells relacionats amb productes de processos de fermentació.

El camp d'aplicació de la bioenginyeria avarca des dels tradicionals processos de fermentació de vi fins a l'industrialització de no solament la producció de cervesa, formatge i llet; sinó també de noves productes biotecnològics com són els antibiòtics, enzims, hormones, vitamines, sucres i àcids orgànics. Des de temps remots els microorganismes van ser utilitzats per l'home en la producció d'aliments bàsics, com el pa o el formatge. L'art de fer vi a passat de pares a fills des d'1 antic Egipte. Només fins a finals del segle XIX, gràcies als estudis del químic i microbiòleg Louis Pasteur, va nàixer la biotecnologia com a ciència. La definició moderna de biotecnologia, segons la OCDE, és l'aplicació de la ciència i la tecnologia tant als organismes vius com a les seves parts, productes i models d'ells, amb el propòsit de modificar tant la matèria viva com la no viva amb l'objecte de produir coneixements, bens i serveis.

Un procés de fermentació és un procés químic que fa servir microorganismes per a obtenir un producte en particular, aprofitant la selectivitat dels microorganismes per a produir un determinat compost. Els processos de fermentació es realitzen en un bioreactor. Un bioreactor és un vaixell on els microorganismes són cultivats de forma controlada i/o matèria primera és convertida o transformada per reaccions biològiques.

En una fermentació, es necessita un adequat control de totes les variables de procés, per tal que qualsevol canvi inesperat en el valor d'alguna de elles pot afectar el desenvolupament de els microorganismes i en conseqüència disminuir la productivitat del bioreactor. El principal objectiu de controlar un procés de fermentació, és maximitzar la producció de microorganismes o altres compostos metabòlics. Recents avanços en enginyeria genètica han augmentat la importància de l'adequat control dels processos biotecnològics. L'ús de cèl·lules de mamífers o microorganismes en la producció de molècules complexes necessita de l'anàlisi i el control de totes las variables de procés, tal com temperatura, concentració d'oxigen i pH.

El present treball es centra en l'aplicació de xarxes neuronals artificials en les àrees de modelat, identificació, control i optimització de processos biotecnològics, principalment en

bioreactors de tipus fed-batch. Un bioreactor fed-batch es fa servir quan la producció d'un determinat compost d'interès, és inhibida per l'alta concentració de substrat. En un procés fed-batch, la fermentació comença amb un volum, concentració de microorganismes i substrat determinats i a mida que transcorre el procés de fermentació, el substrate s'agrega poc a poc, fins que el bioreactor és ple.

En aquell treball les idees i tècniques utilitzades per las xarxes neuronals artificials són presentades amb la notació familiar per a un enginyer de control. Diferents estructures de xarxes neuronals artificials i la seva possible aplicació a diferents sistemes de control van ser resumides. També s'han presentat alguns processos de fermentació fed-batch. Aquests processos es fan servir per il·lustrar casos específics de problemes de control. Específicament, un model no lineal i multivariable d'un bioreactor es fa servir per il·lustrar les tècniques de control basades en el model del procés. Un model per a la producció de invertasa a través del llevat *Saccharomyces cerevisiae* es fa servir per il·lustrar les tècniques d'optimització i control.

Dins les tècniques de control basades en el model del procés, es va a implementar un model directe i un invers de la fermentació multivariable. Els dos models, basats en xarxes neuronals artificials, consideren efectes biològics, tèrmics i de pH. Multilayer perceptron i Radial Basis Function són las xarxes neuronals que es van a fer servir per a la construcció d'ambdós models. Per il·lustrar la fiabilitat d'aquests models, diferents proves van ser realitzades. El model directe del processo de fermentació, basat en xarxes neuronals, va ser provat quan el procés opera en estat estacionari, en estat dinàmic i quan una perturbació en l'àcid causa que el pH del procés sigui un altre. El model invers del procés de fermentació també va ser provat fent canvis aleatoris del punt de consigna. L'arquitectura Radial Basis Function va a ser el millor model directe que es va a trobar. Pel model invers del procés de fermentació, es va a trobar que la millor arquitectura es la Multilayer Perceptron 11-7-1, que va ser entrenada amb informació dels estats estacionaris del procés.

Per una altra banda, per a l'optimització de la producció de invertasa es necessita trobar el perfil d'alimentació òptim, de manera que la productivitat del bioreactor sigui màxima i el temps de fermentació sigui mínim. Aquest doble objectiu de optimització constitueix una novetat i no ha estat obtingut per altres mètodes d'optimització prèviament publicats. L'objectiu del controlador és trobar a cada instant de temps l'acció òptima de control, és dir, cada vegada trobar quin és el flux d'alimentació correcte per complir el doble objecte de l'optimització. Aquest treball va fer servir la programació dinàmica neuronal (NDP) amb l'objectiu de implementar aquell controlador. Aquesta tècnica fa servir xarxes multilayer perceptron o fuzzy ARTMAP. Aquest mètode d'optimització utilitza perfils d'alimentació subòptims com a suposició inicial. A través d'ella, una xarxa neuronal es utilitza per construir la superfície de costos en l'espai dels estats del procés. Aquesta superfície de costos es millora a través de la iteració de Bellman. Una vegada obtinguda una bona aproximació a la superfície de costos òptima, aquesta es implementada en un sistema de control que fa ús

de l'equació de Bellman. Aquest controlador és provat dins diferents condicions d'operació del procés de fermentació, específicament quan la fermentació comença amb diferents volums inicials. S'ha trobat que la metodologia emprada és millor que altres mètodes de optimització ja que es pot utilitzar en altres processos de fermentació sense la necessitat de fer una optimització on-line. Les trajectòries òptimes trobades pel controlador son similars a la trajectòria seguida pel millor dels perfils subòptims. Amb Multilayer Perceptron- NDP s'han obtingut els millors rendiments, però la trajectòria de variable manipulada és força abrupta. Amb Fuzzy ARTMAP-NDP no es presenta aquest problema. El controlador que implementa fuzzy ARTMAP-NDP és provat també quan es presenta un canvi brusca en la concentració de cèl·lules. El 50% moren. En aquell cas el desenvolupament del controlador és millor que el rendiment de la fermentació quan el millor dels perfils d'alimentació subòptims es fan servir.

Per últim es pot dir que la integració de l'enginyeria de control amb les xarxes neuronals és un camí a seguir per futures línees d'investigació. Las xarxes neuronals artificials poden, amb èxit, fer-se servir en el control de bioreactors fed-batch.

*A los físicos y matemáticos por hacer del mundo un lugar mas entretenido*

## Acknowledgements

I would like to acknowledge my thesis advisors Francesc Giralte and Jaume Giralte Marcé for their ideas and support and to the Spanish government for the scholarship that made possible the realization of this work.

Someone said that each person is a brut diamond that is polished little by little by every one that she or he meets. I agree with that thought and I would like to acknowledge all people that have polished my mind during these four years. My thesis advisors and Alex Arenas. Also, Prof. Venkat Venkatasubramanian from Purdue University that kindly accepted me in his research group during the summer of 2000 and introduced me to MATLAB. To Prof. Jay H. Lee from Georgia Institute of Technology that introduced me to optimal control and gently accepted my visit to his research group during the fall of 2001.

To my teachers at the Rovira I Virgili University: Lourdes Vega, Josep Bonet, Allan Mackie, Azael Fabregat, Ildelfonso Cuesta, Joan Herrero, Jim Keffer, Jordi Grifoll, Rene Bañares, Srinivasa Murthy and Lidia Quinzani. The teachers I kindly had at UPC: Robert Griño for his robust control course, Joan Cabestany and Rene Alquezar for their introductions to neural nets. To the teachers that accepted me at their courses at Gatech: Nader Sadegh for his linear control course, Ashok Goel from Computer Science, Prof Hayes and his ultra-tech classes of adaptive filters, and specially to Prof. Anton J. Kleywegt and his introduction to the optimization world through the stochastic optimization course at the school of industrial and systems engineering.

To the researches of the Fenòmens de transport group that I did not mentioned before: Joan Ferré, Robert Rallo and Dolors Puigjaner.

Also to the people from the Intelligent Process Systems Laboratory of Purdue University, especially to Sourab Dash and to the people from ISSICL group at Georgia Institute of Technology: Niket Kaisare, Jong Min Lee, Dr. Kangwook Lee, Yandog Pan, Andrew Dorsey and Jaein Choi.

Because a person is composed by both mind and soul and one cannot develop without the other, I would like also to mention the people that polish my soul, and my mind in some degree.

Firstly my Tarragona friends: Vesselina P. Pashova, Gabriela Espinosa, Jorge Velasquez, Eliana Arango. Camilo Zapata for his constant support. Pedro Zapata the best roommate I ever had even with Paula's story. My *despatx* partners Josep Maria Gasto, Roger Guimerà, Orlando Silva, Gabriela again and Paulo Louvo. Among them all the people who saw me cry and laugh and helped me in the hardest times. Alejandro Gomez and his directions, and la Encuesta de la semana y el espacio muestral that keep me up to date with Medellin news. The Purdue crew: Erika Hernandez, Luis Roman, Gopal Natajaran and Daphne and the

people I met at Gatech: Helene Simone, Juan Pablo Hinestroza and Jennifer. Precaris: Susana Figueroa, Carlos Nieto, Paula Pescador, Silvia Diez, Tanacis Aeftaxias, Angel Jimenez, Samira Elboudamousi and our Morocco trip and the Arabic culture immersion. Mohammed Alshang, Antonio Rodriguez, Israel Herrera, Josep Pàmies, Zaid Alamber and his emails, Oliver Contreras and Claudia Barba, Anton Davidoff, Leonardo Valencia, Edgar Soto, Nuria Suarez, Felipe Osorio, Guillermo, Cathy and those who I forget now to mention. Among those, the people at la penya who drank one vermouth or two with me and taught me the facts of life: Alvarito Morato, Albert Manyes, Roger Guimerà, Montse Meneses, Magdalena Paradowska, Frank Dounabel and the rest of Erasmus Germans, among them Ralph Rosenbaum, Ralph Hartanatan and Michael. Once in la penya how not to mention la *comision permanente* at *el candil* and their constant reminder of what not to become.

Finally, and to close a chapter of my life, I would like to acknowledge Catalunya and the Catalans and their castells, sardanes, Barça, correfocs, Sant Joan and Sant Jordi diades, for let me grow a little more among them.

And last but not least I like to acknowledge my family for their constant support: Edilberto my father y Maria Paola, Voqui y Esteban, Adriana, Nia, Pablo, Valen y los nuevos miembros. La tita y los Peroni. De no ser por todos ellos este escrito nunca hubiera sido una realidad.

Cata

*The proper and immediate object of science is the acquisition, or communication, of  
truth...*

Samuel Taylor Coleridge, Definitions of Poetry, 1811





## **Contents**

<b>LIST OF TABLES</b>	<b>IV</b>
<b>LIST OF FIGURES</b>	<b>V</b>
<b>LIST OF FIGURES</b>	<b>V</b>
<b>NOTATION</b>	<b>VIII</b>
<b>CHAPTER 1: INTRODUCTION</b>	<b>1</b>
<b>1.1 Motivation</b>	<b>1</b>
<b>1.2 Background</b>	<b>3</b>
<b>1.3 Objectives and structure</b>	<b>5</b>
<b>CHAPTER 2 : FED-BATCH FERMENTATION PROCESSES</b>	<b>7</b>
<b>2.1 Description</b>	<b>7</b>
<b>2.2 Fed-batch fermentation models</b>	<b>7</b>
2.2.1 Multivariable Model	8
2.2.2 Invertase production model	13
2.2.3 Penicillin production model	17
2.2.4 Other models	19
<b>CHAPTER 3 : ARTIFICIAL NEURAL NETWORKS</b>	<b>21</b>
<b>3.1 Introduction</b>	<b>21</b>
3.1.1 Where they come from? Its relation with biological neural systems	21
3.1.2 Mathematical modeling of biological neural systems: Artificial neural networks	23
3.1.3 Neural network classification: types of artificial neural networks	24

<b>3.2 Neural networks history : Major milestones in the development of neural computation</b>	<b>27</b>
<b>3.2 Advantages and disadvantages of neural networks : when they can be used</b>	<b>29</b>
<b>3.3 Some neural networks types</b>	<b>31</b>
3.3.1 Multilayer Perceptron	33
3.3.2 Radial Basis Function	37
3.3.3 Self Organized Maps	39
3.3.4 Fuzzy Neural Networks	41
3.3.5 Fuzzy ARTMAP	47
<b>3.4 Building a NN control model: multivariable fed-batch bioreactor case study</b>	<b>51</b>
3.4.1 Data acquisition and control models	52
3.4.2 Data pre-processing	54
3.4.3 Tuning the network architecture	57
3.4.4 Tuning the training algorithm	58
<b>CHAPTER 4 : PROCESS CONTROL</b>	<b>61</b>
<b>4.1 Introduction</b>	<b>61</b>
<b>4.2 Objectives of process control</b>	<b>62</b>
<b>4.3 Non-linear vs. linear process control</b>	<b>63</b>
<b>4.4 Some structures for model based process control</b>	<b>63</b>
4.4.1 Inverse model control	64
4.4.2 Internal model control	64
4.4.3 Model predictive control	67
<b>4.5 Neural networks in process control</b>	<b>69</b>
4.5.1 Neural networks in identification	72
4.5.2 Fault diagnosis and neural networks	75

<b>4.6 Optimal control</b>	<b>76</b>
4.6.1 Methods for optimal control used for fermentation processes : invertase production case study	76
4.6.2 Dynamic programming and reinforcement learning	79
4.6.3 Neuro dynamic programming approach	82
4.6.4 Optimal control trough NDP: invertase production case study	84
<b>CHAPTER 5 : RESULTS</b>	<b>87</b>
<b>5.1 Model based control</b>	<b>87</b>
5.1.1 Direct model	88
5.1.2 Inverse model	93
<b>5.2 Optimal control</b>	<b>94</b>
5.2.1 MLP dynamic programming	99
5.2.2 Fuzzy ARTMAP dynamic programming	107
<b>CHAPTER 6: CONCLUSIONS AND WORK PERSPECTIVES</b>	<b>115</b>
<b>7. BIBLIOGRAPHY</b>	<b>119</b>
<b>APPENDIX A</b>	<b>130</b>
<b>APPENDIX B</b>	<b>180</b>

## List of Tables

<b>Table 1.1</b> Economic impact of biotechnology in the OCDE countries	1
<b>Table 2.1</b> Initial values of the fermentation process, microorganism parameters and operation and control variables values at the operation point used to simulate the multivariable fermentation system	12
<b>Table 2.2</b> Initial values of the fermentation process, the microorganism parameters and the values of the operation and control variables at the operation point used to simulate the invertase production fermentation process	16
<b>Table 2.3</b> Initial values of the fermentation process, microorganism parameters and operation and control variables constraints for penicillin biosynthesis	18
<b>Table 3.1</b> Neural network classification scheme adapted from Sarle (1997)	25
<b>Table 3.2</b> Fuzzy ARTMAP map field output based on which one of the fuzzy ART modules is active	50
<b>Table 3.3</b> Disturbances applied to the substrate flow rate (input variable) in a multivariable fed-batch bioreactor for process simulation	53
<b>Table 3.4</b> Normalization limits for the NN input data in the multivariable bioreactor model	56
<b>Table 5.1</b> Average relative errors of the direct model outputs with respect to the final process values for steady state operation of the multivariable fed-batch bioreactor	89
<b>Table 5.2</b> Average relative errors of the direct model outputs with respect to the process expected values for a positive step in the manipulated variable of the multivariable fed-batch bioreactor	90
<b>Table 5.3</b> Average relative errors of the direct model outputs with respect to the process expected values for multiple random steps in the manipulated variable for the multivariable fed-batch bioreactor	91
<b>Table 5.4</b> Average relative errors of the direct model outputs with respect to the expected process values for a steady state process operation with a pH perturbation of the multivariable fed-batch bioreactor	92
<b>Table 5.5</b> Average relative errors of the direct model outputs with respect to the expected process values for a positive step in the manipulated variable with a pH perturbation of the multivariable fed-batch bioreactor	92
<b>Table 5.6</b> Ranking of controller performance for direct process models of the multivariable fed-batch bioreactor	93
<b>Table 5.7</b> Average relative errors of the inverse model output with respect to the expected process value for random values in the set point for the multivariable fed-batch bioreactor	93
<b>Table 5.8</b> Invertase production optimization results for an initial fermentation volume of 0.6 liters. The profit depends on the productivity and the total fermentation time	102
<b>Table 5.9</b> Profits obtained for unknown initial fermentation volumes $V_0$ when the MLP-NDP controller is used. Profit results are compared against results obtained by applying the best policy for the given volume	105
<b>Table 5.10</b> Best Fuzzy ARTMAP structures used into Bellman's iteration for the invertase production optimization	107
<b>Table 5.11</b> Invertase production optimization results for an initial fermentation volume of 0.6 liters. The profit depends on the productivity and the total fermentation time	108
<b>Table 5.12</b> Profits obtained for different initial fermentation volumes for invertase production	110

## List of Figures

- Figure 1.1** Summary of the control objectives and methodologies followed by the current work
- Figure 2.1** Scheme of MIMO fermentation process
- Figure 2.2** Bioreactor control model with 6 inputs  $\Phi^{\text{in}} = \{F, F_{\text{steam}}, F_a, F_b, F_{\text{air}}, \text{RPM}\}$  and 6 outputs  $V = (x, s, p, T, \text{pH}, [\text{O}_2])$
- Figure 2.3** Schematic representation of fermentative and respiratory fluxes inside cloned *Saccharomyces cerevisiae* cell. Pyruvate is the bifurcation point at which the cell decides the fractions of glycolytic flux to be diverted toward fermentative and respiratory metabolic pathways in invertase fermentation
- Figure 3.1** Schematic drawing of a biological neuron
- Figure 3.2** Ramon y Cajal drawing of the cerebellar cortex and neural tissue for a cat. From *Histologie du systeme nerveux de l'homme et des vertebres* 1909
- Figure 3.3** **Upper picture:** Hippocampus neurons with permission of *Slice of Life* project. **Lower picture:** Pyramid neuron grown in culture. From Vision Concepts (© information in [www.4colorvision.com/citation.htm](http://www.4colorvision.com/citation.htm))
- Figure 3.4** Schematic drawing of a typical artificial neuron. The output is a function of the inner product of the input vector and the weight vector of the connections  $\mathbf{w}$
- Figure 3.5** Scheme of multilayer perceptron architecture. The activation function of each neuron is  $f(y) = 1/(1 + e^{-y})$  where  $y$  is the inner product of the input vector and the weight vector of the neuron input connections.
- Figure 3.6** Scheme of a multilayer perceptron architecture showing the notation for neurons and weights
- Figure 3.7** Scheme of a Radial Basis Function network architecture. The activation function of each RBF neuron is  $f(y) = \exp\left(-\|y - \mu_j\|^2 / \sigma_j\right)$  where  $y$  is the input vector,  $\mu_j$  is the center of the RBF neuron and  $\sigma_j$  is a measure of its width
- Figure 3.8** Example of a topological neighborhood  $N_c(t)$  in an hexagonal lattice of a SOM, where  $t_1 < t_2 < t_3$
- Figure 3.9** Fuzzy neural network architecture composed by a fuzzification layer, a fuzzy inference layer and a defuzzification layer
- Figure 3.10** Combination of fuzzy grid and fuzzy ellipsoid
- Figure 3.11** Scheme followed to obtain the NN control models
- Figure 3.12** Input-Output scheme for the neural network models of the multivariate bioreactor: (a) Current direct process model with  $\mathbf{V} = \{x, s, p, T, \text{pH}, [\text{O}_2]\}$ ; (b) literature inverse process models; (c) modified literature inverse models and current proposal
- Figure 3.13** Influence of the normalization procedure when the 404 patterns of data from the multivariable bioreactor used to train the neural model contain noise. Comparison between predicted and expected outputs for three different normalization limits: (a) SSE=6.88; (b) SSE=2.50; (c) SSE=1.65

- Figure 3.14** PCA of the multivariable bioreactor data (a) Process outputs; (b) neural network input training data
- Figure 4.1** Inverse model control scheme
- Figure 4.2** Complete scheme of non-linear internal model control structure
- Figure 4.3** Scheme for model based predictive control. The  $i$ th control action is based on the prediction of future process outputs over a prediction horizon  $p$ . An optimal control action trajectory is found and only the first control move is implemented
- Figure 4.4** Direct process identification scheme.  $M$  learns the direct model of the plant  $P$
- Figure 4.5** Indirect process identification scheme.  $C$  learns the inverse model of the plant  $P$
- Figure 5.1** Test schemes for the obtained direct and inverse process models of the multivariable fed-batch bioreactor. The models prediction error (deviation) is computed as shown: (a) Test for direct process models (b) Test for the inverse process models
- Figure 5.2** Different policies used for an invertase bioreactor with initial volume of 0.6 liters. (–) Suboptimal policies given by eq. 5.1 with initial times of 3, 5 and 7 (– –) Optimal policy by Chaudhuri and Modak (1998) (...) Optimal policy by Patkar et al. (1993)
- Figure 5.3** Fermentation dynamics for a policy  $u(t,5,0.13)$  given by eq. 5.1. Cells, glucose and invertase concentrations and fermentation volume are shown. Also shown is the associated profit when  $\lambda = 0.3$
- Figure 5.4** Effect of initial conditions on the productivity of a fermentation process. Solid lines: initial state  $(0.15, 5, 0.1, V_0)$  where  $V_0 = \{0.2, 0.4, 0.6, 0.8, 1\}$ . + lines: initial state  $(0.15, S_0, 0.1, 0.6)$  where  $S_0 = \{1, 1.5, 2, 3\}$
- Figure 5.5** Explored state space 9328 points of the invertase fermentation process by using the suboptimal policies described by eq. 5.1
- Figure 5.6** Optimum final time  $t_f^*$  of the invertase fermentation for each suboptimal policy  $u(t, t_i, b)$  and different initial volumes of the bioreactor. The final optimum fermentation time depends on the value of the initial fermentation volume  $V_0$  and on the policy parameters  $t_i$  and  $b$
- Figure 5.7** State space representation of the trajectories followed by the controlled fermentation process for initial fermentation volumes of 0.4 ( $\cdot$ ), 0.6 ( $+$ ), 0.8 ( $\times$ ). The controller is implemented with the cost-to-go approximator obtained from the second Bellman iteration. The new visited states and its associated cost-to-go are used into a new approximator
- Figure 5.8** Explored state space of the invertase fermentation process after policy iteration procedure. The controlled process followed a different trajectory and visited new states
- Figure 5.9** Optimal policy for an invertase fermentation process with an initial volume of 0.6. The obtained NDP policy is compared against other optimization procedures results. (–) NDP methodology (– –) Chaudhuri et al. (1998), (– – –) Patkar et al. (1993)
- Figure 5.10** Fourier analysis of the control signal from the optimizer
- Figure 5.11** Control policy obtained when the optimizer output is filtered during a fermentation process with initial volume of 0.6 l. Upper figure. Control action average is over five consecutive times. Lower picture, when the average is over six consecutive times
- Figure 5.12.** Optimal NDP policies for different invertase fermentation processes obtained with the MLP-NDP controller
- Figure 5.13** State space representation of the optimal trajectories followed by the invertase fermentation process when MLP-NDP controller is used for different initial volumes: 0.5 ( $\cdot$ ), 0.6 ( $+$ ), 0.7 ( $\times$ )
- Figure 5.14** Optimal policy for invertase fermentation process with an initial volume of 0.6. The obtained fuzzy ARTMAP-NDP policy is compared against other optimization results. (...) Fuzzy ARTMAP methodology, (–) MLP-NDP methodology, (– –) Chaudhuri and Modak (1998), (– – –) Patkar et al. (1993)

- Figure 5.15** State space representation of the optimal trajectories followed by the invertase fermentation process under control with fuzzy ARTMAP-NDP, for different initial volumes: 0.4(x), 0.5 (.), 0.6 (+), 0.7 (^), 0.8 (v)
- Figure 5.16** Optimal policies for different invertase fermentation initial volumes obtained through the control of the process by the fuzzy ARTMAP-NDP based controller
- Figure 5.17** Invertase fermentation process behavior when an abrupt death of microorganisms occurs at  $t=9$ h. The evolution of the controlled process is shown in figure 5.14(a). The control variable, glucose feed rate is shown in figure 5.14(b). The optimal policy is compared against the best suboptimal policy for that given initial volume (0.4 l)
- Figure 5.18** Optimal trajectory of a controlled abrupt microorganisms death. The invertase fermentation process has an initial volume of 0.4 l. When fermentation time equals 9 hours, cells concentration decreases 50%. The controller senses this new bioreactor state and finds both the optimal control action and the new optimal control policy

## Notation

$C_a$	$\text{Kg/m}^3$	Acid concentration in the acid inlet stream
$C_b$	$\text{Kg/m}^3$	Base concentration in the base inlet stream
$C_{\text{pair}}$	$\text{J/kg K}$	Heat capacity of air
$C_{\text{pR}}$	$\text{J/kg K}$	Heat capacity of the bioreactor contents
$C_{\text{ps}}$	$\text{J/kg K}$	Heat capacity of the subtract stream
$E$	$\text{J/gmol}$	Activation energy
$f_d$	$1/\text{s}$	Microorganism mortality rate
$F$	$\text{m}^3/\text{s}$	Flow rate of the subtract stream
$F_a$	$\text{m}^3/\text{s}$	Flow rate of the acid stream
$F_{\text{air}}$	$\text{Kg/s}$	Flow rate of the air stream
$F_b$	$\text{m}^3/\text{s}$	Flow rate of the base stream
$F_{\text{steam}}$	$\text{Kg/s}$	Flow rate of the thermal fluid in the bioreactor jacket
$H$	$\text{m}^3/\text{kg}$	Henry's constant
$K_a$		Acetic acid dissociation constant
$k_d$	$1/\text{s}$	Microorganism death constant
$K_{\text{O}_2}$	$\text{Kg/m}^3$	Oxygen Monod's constant
$K_s$	$\text{Kg/m}^3$	Subtract Monod's constant
$\text{O}_2$	$\text{kg/m}^3$	Dissolved oxygen
OCR	$\text{Kg/m}^3 \text{ s}$	Oxygen consumption rate
OTR	$\text{Kg/m}^3 \text{ s}$	Oxygen Transfer Rate
$p$	$\text{kg/m}^3$	Concentration of the bioreactor product
$p_f$	$\text{kg/m}^3$	Concentration of the product inlet stream
$R$	$\text{J/gmol K}$	Ideal gas constant
RPM		Bioreactor mixer revolutions per minute
$S$	$\text{Kg/m}^3$	Concentration of the bioreactor subtract
$s_f$	$\text{Kg/m}^3$	Concentration of the subtract inlet stream
$t$	$\text{s}$	Time
$T$	$\text{K}$	Bioreactor temperature
$T_{\text{inair}}$	$\text{K}$	Temperature of the air stream
$T_{\text{ins}}$	$\text{K}$	Temperature of the inlet stream
$V_R$	$\text{m}^3$	Bioreactor effective volume
$\mathbf{V}$		State vector. $\mathbf{V}=(x, s, p, T, \text{pH}, [\text{O}_2])$
$X$	$\text{Kg/m}^3$	Bioreactor microorganism (biomass) concentration
$x_a$	$\text{Kg/m}^3$	Bioreactor acid concentration
$x_b$	$\text{Kg/m}^3$	Bioreactor base concentration



$x_f$	Kg/m <sup>3</sup>	Inlet stream biomass concentration
$Y_{O_2}$	kg of m.o./kg of O <sub>2</sub>	Microorganism oxygen consumption
$Y_{p/x}$	kg of p./kg of m.o.	Product yield
$Y_{x/s}$	kg of m.o./kg of s	Substrate yield
$\Delta H_\mu$	J/ kg de m.o.	Heat generated by the microorganism
$\Delta G_d$	J	Gibbs's free energy
$\beta$	kg of p./ (kg of m.o min).	Microorganism non-growth production parameter
$\gamma$	1/s K	Maximum microorganism growth rate per K
$\lambda$	J/kg	Latent heat of thermal fluid
$\mu$	1/s	Microorganism growth rate
$\rho_R$	Kg/m <sup>3</sup>	Density of bioreactor contents



# Chapter 1: Introduction

## 1.1 Motivation

Fermentation processes are widely used in chemical, alimentary and pharmaceutical industries. The production of animal feed and supplements, diary food (yogurts and cheese), beer, food colorants, fertilizers, therapeutic drugs and penicillin are some examples of biotechnological processes. Over 113000 sites related to fermentation products have been found in a quick Internet search (www.google.com 28-02-2001). In recent years, the implementation of biotechnological processes in the chemical and pharmaceutical industry has grown and more knowledge and techniques have been acquired and developed (Williams, 2002). The economic impact of biotechnology on different industrial activities over the last two decades in the OCDE (Organization for Economic Co-operation and Development) countries is shown in Table 1.1. Its increasing economic importance is evident in the agriculture and food industries.

**Table 1.1** Economic impact of biotechnology in the OCDE countries

Year	Agriculture and Food industries	Pharmaceutical industries	Chemical industries	Energy Industries	Total (Millions of US\$)
1980	37%	37%	12%	11%	5-20.000
1990	21%	29%	13%	37%	20-40.000
2000	48%	22%	12%	18%	45-200.000

The scope of bioengineering has grown from simple wine-bottle microbiology to the industrialization of not only beer, wine, cheese and milk production, but also of newer products – antibiotics, enzymes, steroidal hormones, vitamins, sugars and organic acids. Since remote times, microbiological systems have been used to produce several basic products. The art of making wine, beer and cheese has passed from father to son since ancient Egypt. Also, yeast was needed in bread cooking since an earliest age. It was until the French chemist and microbiologist Louis Pasteur (1822-1895) when the first fermentation process was studied. French vintners, who found they were producing vinegar instead of wine, requested the study of wine fermentation process. Pasteur found a microscopic plant that he called yeast and since then, the art of making wine became a science and gave birth to biotechnology.

The actual definition of biotechnology is given by OCDE as "The application of Science and Technology to living organisms as well as parts, products and models thereof, to alter living or non-living materials for the production of knowledge, goods and services". Biotechnology and bioengineering can be seen as the branches of chemical engineering specialized in the study of the complete fermentation process. Fermentation is a chemical process that uses microorganisms to obtain a particular product, taking advantage of the microorganism selectivity for the specific product or for a reactant. The fermentation process is carried out in a bioreactor or a system where a biological conversion is effected. The bioreactor is a mechanical vessel in which a) organisms are cultivated in a controlled manner and/or b) materials are converted or transformed via specific reactions. Quite similar to conventional chemical reactors, bioreactors differ in that they are specifically designed to influence metabolic pathways. The accurate control of all variables in a fermentation process is needed since any unexpected change could affect the microorganisms and consequently the bioreactor productivity due to the fragility of biological systems.

Recent advances in genetic engineering have increased the importance of biotechnological processes. The use of mammalian cells or microorganisms for the production of complex molecules requires the analysis and control of many process variables, like temperature, oxygen concentration, pH and mixer shear stress. The objective of controlling a fermentation process is to maximize the output of microorganisms or other metabolic products while keeping other outputs with a deleterious effect (such as ethanol, acetic acid or penicillin) at low values.

A useful kind of biotechnological process is the fed-batch fermentation. Fed-batch fermentation is used to prevent or reduce substrate-associated growth inhibition by controlling nutrient supply. In a fed-batch fermentation process, fermentation starts with some initial volume, concentration of microorganisms and substrate, and, as the process evolves, the substrate is continuously fed into the fermenter. This process is distinctly non-linear and multivariable, and its dynamics and reaction kinetics are not well known. Changes in initial conditions, parameter variations, input saturation, external disturbances, and unmodeled dynamics are all encountered in practice, making the control problem very difficult. It is typical of complex processes where a small improvement in performance results in substantial economic benefits.

The control of a fed-batch fermentation process is a problem with increased industrial interest. With an effective control strategy the production of the desired product increases, and the cost of further purification decreases remarkably. An effective control system should be able to learn how to improve its performance and adapt itself to changes during a cultivation process. Also errors in the online analysis should be identified and ignored by the controlling algorithms, through a data reconciliation method. The aim of control engineers should be to develop intelligent controllers with better performance, such as those based on

neural networks or fuzzy logic, which allow predictive control of the fermentation (Ritzka et al., 1997). On the other side, the objective of any control system is to influence the behavior of a dynamic system. The later includes maintaining the outputs of systems at constant values (regulation) or forcing them to follow prescribed time functions (tracking). The control problem is to determine the control inputs to the process using all available data. Achieving fast and accurate control even while assuring stability and robustness is the aim of all control systems design.

The present work assumes that the behavior of a fed-batch fermentation process, as any causal process, can be influenced through a control system if a reliable model of it is at hand. A reliable model presents a balance between complexity and accuracy. It is assumed also that this model can be built based on some mapping of the input-output fermentation process historical data

## 1.2 Background

Lee et al. (1999) reviewed the advances in control of fed-batch fermentations. They examined both simple exponential feeding and inferential methods with classical control action and knowledge-based control systems. They concluded that the newer knowledge-based control systems, e.g., those relying on fuzzy logic and neural networks concepts, are receiving considerable research attention and hold promise for optimizing fed-batch techniques for complex fermentation systems. Sargantis and Karim (1998) used a model-based control for dissolved oxygen in  $\beta$ -lactamase production. They based his work on an autoregressive with exogenous inputs (ARX) model and on an autoregressive moving average with exogenous inputs model (ARMAX) to adaptively place system poles. A similar control technique could be used in fed-batch fermentation if an accurate model of this process is available. Tholodur and Ramirez (1996) studied a neural network-based scheme of parameter function modeling used to model the non-linear dynamics of a yeast and bacterial fed-batch bioreactors. They used the obtained models into dynamic programming optimization to generate optimal feeding policies, although the lack of complete state information resulted in suboptimal control policies. A way to use dynamic programming and overcome this problem is used in the present work. It would be discussed in detail later in the process control chapter.

The adaptive control of a fed-batch fermentation plant was discussed by Boskovic and Narendra (1995). In adaptive control, a filter is used to realize the controller components. Neural networks can be viewed as non-linear adaptive filters. Thus, a neural network based controller is a non-linear adaptive control system. These authors compared the performance of linear, linear-adaptive non-linear and non-linear adaptive control methods, using simulation studies. They found that neural networks are distinctly preferable if accuracy and robustness are critical issues. Efe et al. (2001 a and b) designed a model reference control for a

benchmark bioreactor problem (Agrawal et al. 1982). They used a neural network to model the dynamics of the fermentation process in a control-affine process model. As a control-affine model approximates the process, the inverse process model can be obtained by algebraically inverting the process model. Thus, the desired process input could be easily obtained at any time step. Efe et al. (2001 a and b) also showed that the obtained controller is stable in the sense of bounded inputs/bounded outputs criterion as well as it is locally stabilizing in the overall control system in the sense of Lyapunov. Later these authors (Efe and Kaynak, 2001c) used the same bioreactor benchmark to test three neural network control strategies: an inverse model of the plant used as a controller, a self-learning controller where a neural network learns on-line the process inputs for a desired process trajectory, and dynamical neural unit (recurrent nodes) based controller. They used engineering criteria to conclude that the best controller is the one based on a dynamical neural unit layer.

Besides process control, neural networks have been used also in the biotechnology field in the prediction of some bioreactor variables that are difficult to measure. Karim and Riviera (1992) used a feedforward neural network for process state estimation of difficult measure primary process variables, such as biomass, substrate and product concentrations in an ethanol fermentation of *Z. mobilis*. They concluded that the obtained model could be used for diagnosis and control of the fermentation process. Linko and Zhu (1992) used a neural network for real-time estimation and multi-step ahead prediction of enzyme activity and biomass dry matter in fungal *A. Niger* glucoamylase fermentation. Tsaptinos and Leigh (1993) discussed various issues to consider in feedforward neural network learning of fermentation process dynamics. Glassey et al. (1994) also used neural networks as on-line estimators for biomass, recombinant protein concentration and plasmid structural instability during the production of some proteins expressed by constitutive, chemo and thermo-inducible vectors in recombinant *E. coli*. They explored several feedforward neural network architectures using genetic algorithms. Chemotaxis algorithm was used for network learning. They successfully developed also dynamic neural networks and auto-associative neural networks as estimators. Syu and Hu (1997) employed a time-delayed neural network to predict metabolic products of the 2,3-butanediol fermentation by *K. oxytoca* over different fermentation batches. They concluded that the success of his work could be further extended to other bioreactor systems. Finally, van Can et al. (1999) used a neural network to build a gray box model of the enzymatic conversion of penicillin G. In a gray box model some unknown parameters of a first principle model are modeled with a neural network. They concluded that his methodology could be used to considerably reduce the number of identification experiments and it can be applied to a wide range of biochemical processes to obtain a model with reliable frequency extrapolation properties to be used under dynamic conditions not present in the identification experiments. An introduction to neural networks and to some of the more used neural network architectures and types is presented in the third chapter.

### 1.3 Objectives and structure

The objective of the current work is to study and solve, by using neural networks systems, the control problem of a fed-batch fermentation process. Based on previous work in this field, it can be assumed that a neural network fermentation model can be obtained in a manner suitable for control. Identification and modeling are the first step in the search of a control system. There are several model-based control schemes that can be used when a reliable process model is available. These control systems will be discussed in detail in the process control chapter. The specific objectives of the present work can be divided in two categories: fed-batch bioreactor identification and modeling, and fed-batch bioreactor optimization. Both categories deal with the control problem of a fermentation process but the objective of each one of them is different. In fed-batch bioreactor identification and modeling the objective is to find a suitable control model to be used in a control scheme. In the optimization of a fed-batch bioreactor system, the objective is to find the optimal control action at each time step of the fermentation process. The optimal control action is given by an objective function based on the fermentation profitability. In both categories, neural networks systems are employed due its non-linear approximation capability of mapping data-based functions. The specific objectives of the present work can be stated as:

- Obtain a control model of a class of fed-batch bioreactor, that is, a non-linear multi input-multi output (MIMO) dynamic process, using feedforward and Radial Basis Function artificial neural networks. The model is applied to the fermentation of glucose with *Saccharomyces cerevisiae* yeast to obtain ethanol.
- Study and compare the methodology and the resulting models obtained from the different paths used to find a neural network process model for *Saccharomyces cerevisiae* glucose fermentation system.
- Study different model-based control schemes for the above fermentation process.
- Solve the optimal control problem of cloned invertase expression in *Saccharomyces cerevisiae* fermentation using neural network systems.
- Find an optimal control action that maximizes the productivity and minimizes the invested time of cloned invertase expression in *Saccharomyces cerevisiae* fermentation. To accomplish this goal neuro dynamic programming methodology coupled with feedforward neural networks and fuzzy ARTMAP systems will be used.
- Compare the neural network-optimization methodology employed in the present work with previous optimization methods used to solve the same fed-batch fermentation optimization problem.

- Develop an optimal controller able to find the optimal feeding profile even when disturbances arise in the invertase fermentation process by *Saccharomyces cerevisiae*.
- Explore and compare different possibilities of combine neural network systems into control schemes for fed-batch bioreactor process.

In Figure 1.1 are summarized the control objectives and the methodologies used in the current work.

In the next chapter fed-batch fermentation processes are treated in detail. Some common structured models and their drawbacks for control systems are described. Also a first principle multivariable model is proposed to generate data for simulation. Moreover, some of these models are used in the simulation of the fermentation process control studies. Chapter 3 is dedicated to neural networks. After an introduction, the advantages and disadvantages of these algorithms are discussed. Then, some types of neural networks are explained in detail and finally, several issues of a neural network model construction are reviewed. Chapter 4 describes relevant aspects of process control. Its objectives and the problems of non-linear control are discussed. Some structures of model-based process control are described as well as the role of neural network systems in process control and optimal control for fed-batch fermentation process. Chapter 5 presents the results of applying neural networks to different control schemes for a *Saccharomyces cerevisiae* fed-batch fermentation process. Last chapter contains the conclusions of the present work and the lines of future development.

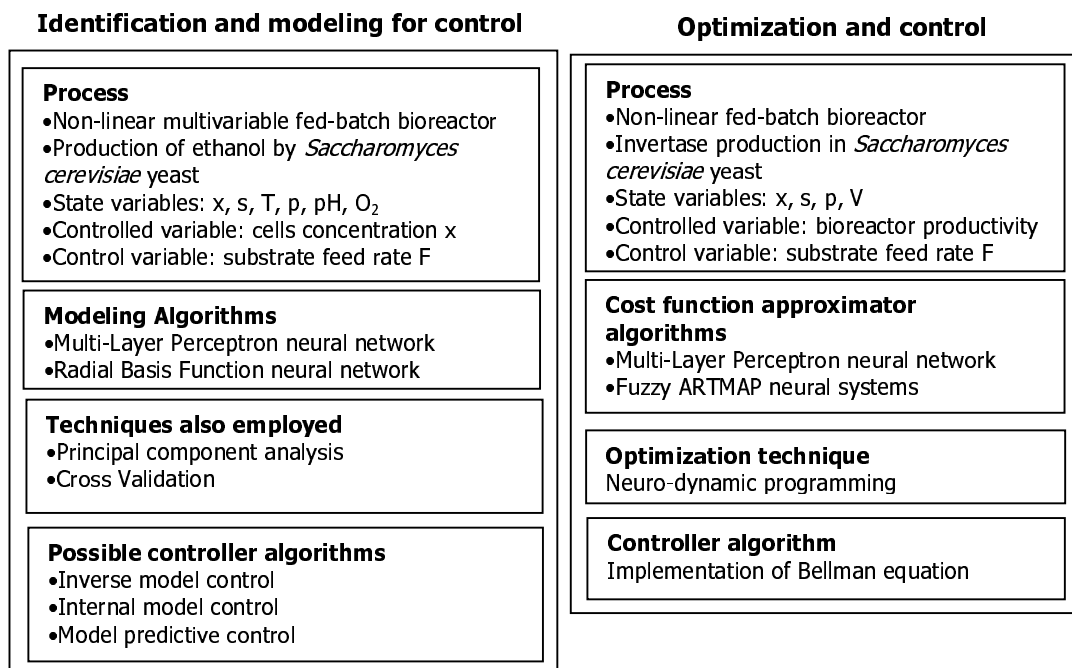


Figure 1.1 Summary of the control objectives and methodologies followed by the current work



## **Chapter 2 : Fed-batch fermentation processes**

### **2.1 Description**

Many industrially important fermentation processes involving production of antibiotics, enzymes and organic acids are carried out in fed-batch mode of operation. In this mode of action the substrate(s) is (are) added continuously in an otherwise batch operation. Fed-batch bioreactors are particularly useful when, in a fermentation process, the growth and/or metabolite production is inhibited at certain substrate or end product concentration or due a catabolite repression. In those cases, the controlled addition of substrate is essential to achieve maximum production of the desired product. Therefore, the fermentation process productivity depends on the substrate feed rate profile.

Usually, the relevant process variables are substrate concentration, product concentration and some measure of the microorganism cell concentration. Also, mixer shear stress, pH and temperature are considered relevant because they should be maintained within certain limits. Fermentation volume is considered as a process state, because it changes during the fermentation until the vessel volume is reached. The common variable used for biomass control is the substrate feed rate. Airflow rate and thermal fluid rate in the heat exchanger can be used also as manipulated variables. The relations among all these variables could be given by first principles, but often the growth rate of microorganisms and/or the metabolites production mechanism is not well known. Thus, the resulting first principles model has unknown and/or uncertain parameters.

### **2.2 Fed-batch fermentation models**

Fermentation is one of the most widespread bioprocess. It has been widely studied but there is not a unified way of modeling this kind of chemical process. To effectively control fermentation it is necessary to represent the underlying dynamics with a suitable model that effectively balances accuracy and complexity. As it was said before, the dynamics of this biotechnological process is strongly non-linear and multivariable. In addition, fermentation reactions are difficult to model. Often reaction mechanisms are unknown and kinetic equations have to be postulated and experimentally verified. The application of first principle non-linear models, with multi-input multi-output (MIMO) variables, often results in a complex estimator/controller design that increases the on-line computational requirements of the control system. Since detailed mechanistic models are too complex for direct use in many

model-based control schemes, it is necessary to seek simpler approximations. Some of the more used approximations for non-linear models are analyzed by Pearson (2000). Efficient model strategies for biochemical processes are needed to obtain reliable mathematical models of these processes over a limited period of time and with limited experimental effort. This means that the bioprocess model should be based on easily obtainable knowledge. Besides, the resulting models should have good interpolation capacities, i.e., that they should be applicable under conditions that differ from the conditions of the identification experiments. Neural networks have been used successfully to model several non-linear multivariable systems, including bioreactors, (Karim and Riviera, 1992; Linko and Zhu, 1992; Tsapinos and Leigh, 1993; Glassey et al., 1994; Boskovic and Narendra, 1995; Syu and Hu, 1997; van Can et al., 1999, Efe et al., 2001a,b,c;) as it have been previously mentioned. Some other examples of neural networks non-linear modeling are given in the process control chapter.

The following fed-batch fermentation models are developed from first principles. They are solved through simulation to obtain the historical data necessary to develop a neural network control model. The following model parameters were obtained through experimental data fitting. The multivariable model is used to obtain all dynamical data needed for neural network learning about the bioreactor system for the fermentation process identification procedure. While the invertase production model is used into the bioreactor optimization procedure. Other models have been studied before and are presented here to compare performances with the same techniques proposed in the current work.

### 2.2.1 Multivariable Model

The considered system is a perfectly mixed fed-batch bioreactor using *Saccharomyces cerevisiae* yeast to produce mainly ethanol from glucose, witch is widely used in the brewing industry. A bioreactor scheme is shown in Figure 2.1. In this kind of process, a substrate, containing glucose and other metabolites necessary to the yeast, is added continuously to the bioreactor and no output is removed until the reaction time is reached. Independently, an airflow trough the bioreactor is maintained in order to supply the oxygen necessary for the microorganisms. The operation temperature is maintained at the desired value by adding or removing heat to or from the reactor. Additionally, the pH of the reaction mixture is

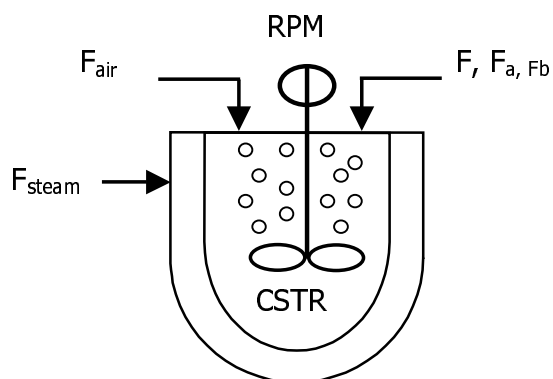


Figure 2.1 Scheme of MIMO fermentation process

regulated by adding the amount of acid or base needed. The control model of this process is considered as a MIMO system with 6 inputs and 6 outputs. The inputs to the control system are the flow rates of substrate, air, thermal fluid, acid and base solution, and the stirring RPM of the mixer, which assures complete mixing and the uniform oxygen transfer from the air phase. The outputs of the control model are the vessel bulk concentrations of cells  $x$ , substrate (glucose)  $s$ , and product (mainly ethanol)  $p$ , the temperature  $T$ , pH and dissolved oxygen concentration  $[O_2]$ . The state vector  $(x, s, p, T, \text{pH}, [O_2])$  herein after is denoted  $\mathbf{V}$ . The bioreactor system is simulated with the equations and relations between the principal output process variables and the input variables usually used to control. A process scheme is given in Figure 2.2. The value of each output variable at time  $t$  depends on the values of the input variables at time  $t$  and on all the values of the output variables at time  $t-1$  derived solving the model process. The microorganism, product and substrate kinds, product and substrate yields, and other operation parameters are set into the above relations by means of specific constants. This provides the flexibility needed to emulate different fermentation processes.

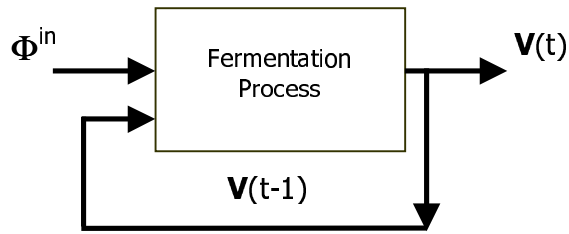


Figure 2.2 Bioreactor control model with 6 inputs  
 $\Phi^{\text{in}} = \{ F, F_{\text{steam}}, F_a, F_b, F_{\text{air}}, \text{RPM} \}$   
 and 6 outputs  
 $\mathbf{V} = ( x, s, p, T, \text{pH}, [O_2] )$

### 2.2.1.1 Fermentation Model

The fermentation model considers the cells as a collection of discrete entities forming a whole homogeneous unit with multicomponent description of cell-to-cell heterogeneity (structured model of population). This model has the following characteristics:

The growth rate  $\mu$  of cells is a function of the substrate ( $s$ ) and oxygen ( $[O_2]$ ) concentration which are assumed to follow the Monod law, the fermentation temperature as an enzymatic activity, and the pH, as described by:

$$\mu = \frac{s}{K_s + s} \cdot \frac{[O_2]}{K_{O_2} + [O_2]} \cdot \frac{T \cdot e^{-E/RT}}{1 + e^{-\Delta G_d/RT}} \cdot f(\text{pH}) \cdot \gamma \quad (2.1)$$

with

$$f(\text{pH}) = \begin{cases} 1 & \text{if } 4.5 < \text{pH} < 9 \\ 0 & \text{otherwise} \end{cases} \quad (2.2)$$

All variables and constants with its units are listed in the nomenclature section.

The Oxygen Transfer Rate (OTR) is given by the following empirical correlation between the airflow, the bioreactor volume and the mixer RPM (Atkinson and Marituaña, 1991).

$$OTR = 5.717 \cdot 10^{-8} \cdot RPM^{1.2} \cdot \left( \frac{F_{air}}{V_R} \right)^{0.2} \quad (2.3)$$

The following equations, that describe the pH behavior, are similar to those proposed by Yeo and Kwon (1999) for a CSTR where pH is a function only of the flow rates of the base and acid streams,

$$x_b + 10^{-pH} - 10^{pH-14} - \frac{x_a}{1 + 10^{pK_a - pH}} = 0 \quad (2.4)$$

$$\frac{dx_a}{dt} = \frac{F_a}{V_R} \cdot C_a - \frac{F_a + F_b + F}{V_R} \cdot x_a \quad (2.5)$$

$$\frac{dx_b}{dt} = \frac{F_b}{V_R} \cdot C_b - \frac{F_a + F_b + F}{V_R} \cdot x_b \quad (2.6)$$

It is assumed that neither any product nor the substrate could change the pH value at equilibrium.

The fermenter temperature depends on the flow rate of the thermal fluid stream  $F_{steam}$ , on the bioreactor jacket, the microorganism generated heat and on the flows and temperatures of all the inlet and outlet streams. In the present work, all heat contribution due to the addition of acid and base streams are neglected because of their low and sporadic flow. The heat balance is summarized by

$$\frac{dT}{dt} = \frac{\lambda}{C_{pR} \rho_R V_R} \cdot F_{steam} + \frac{\mu x \Delta H_\mu}{C_{pR} \rho_R V_R} + \frac{C_{pair} \cdot F_{air}}{C_{pR} \rho_R V_R} \cdot (T_{inair} - T) + \frac{C_{ps} \cdot F}{C_{pR} \rho_R V_R} \cdot (T_{ins} - T) \quad (2.7)$$

### 2.2.1.2 Mass balances

The balance of the other species considered depends on the specific bioreactor used. A fed-batch bioreactor is chosen because it is widely used in industry and its equations are exactly the same as those for a continuous stirred tank (CST) bioreactor assuming that the reactor volume change can be neglected when a subtract stream is added.

A balance for the biomass concentration yields

$$\frac{dx}{dt} = \frac{F}{V_R} \cdot (x_f - x) + \mu(s, [O_2], pH, T) \cdot x - f_d(T) \cdot x - x \cdot \frac{(F_a + F_b)}{V_R} \quad (2.8)$$

with

$$f_d(T) = \frac{(T - 303.15)^{10}}{1.78 \cdot 10^{10}} + k_d \quad (2.9)$$

where  $f_d$  in  $\text{min}^{-1}$  represents the microorganism's mortality as a function of temperature (K). When the temperature is between 297.15 (24°C) and 310.15 (37°C),  $f_d$  is the inverse of the average lifetime of the cells ( $k_d$ ).

The substrate concentration balance is given by

$$\frac{ds}{dt} = \frac{F}{V_R} \cdot (s_f - s) - \mu(s, [O_2], pH, T) \cdot \frac{x}{Y_{x/s}} - s \cdot \frac{(F_a + F_b)}{V_R} \quad (2.10)$$

and the product concentration balance yields

$$\frac{dp}{dt} = \frac{F}{V_R} \cdot (p_f - p) + \mu(s, [O_2], pH, T) \cdot Y_{p/x} \cdot x + \beta x - p \cdot \frac{(F_a + F_b)}{V_R} \quad (2.11)$$

Finally, the dissolved oxygen is given by one of the following two equations depending on the oxygen availability:

$$\text{if } OTR < OCR \text{ Then } \frac{d[O_2]}{dt} = OTR - OCR - [O_2] \cdot \frac{(F_a + F_b + F)}{V_R} \quad (2.12)$$

$$\text{Otherwise } [O_2] = \frac{0.21}{H} \quad (2.13)$$

Thus, liquid oxygen concentration at saturation level is calculated using Henry's law. The Oxygen Consumption Rate (OCR) is given by

$$OCR = \mu(s, [O_2], pH, T) \cdot \frac{x}{Y_{O_2}} \quad (2.14)$$

Table 2.1 shows the values of the operation and control variables on the operation point, the microorganism parameters used and the initial values of the system.

**Table 2.1** Initial values of the fermentation process, microorganism parameters and operation and control variables values at the operation point used to simulate the multivariable fermentation system

Initial Values	Variable	Value	Control Variable	Value
	x	$2.47 \cdot 10^{-1} \text{ kg/m}^3$	F	$8.3 \cdot 10^{-9} \text{ m}^3/\text{s}$
s	$4.63 \cdot 10^{-7} \text{ kg/m}^3$	$F_{\text{steam}}$	0 kg/s	
p	$1.74 \text{ kg/m}^3$	$F_a$	0 $\text{m}^3/\text{s}$	
T	303.15 K	$F_b$	0 $\text{m}^3/\text{s}$	
$x_a$	$0.5 \text{ kg/m}^3$	$F_{\text{air}}$	$1.67 \cdot 10^{-6} \text{ kg/s}$	
$x_b$	$0.5 \text{ kg/m}^3$	RPM	85	
$\text{O}_2$	$0.2 \text{ kg/m}^3$			
Microorganism Parameters	Parameter	Value	Parameter	Value
	$k_d$	$2.67 \cdot 10^{-5} \text{ 1/s}$	$\gamma$	$7.17 \cdot 10^{-3} \text{ 1/(s}\cdot\text{K)}$
	$K_s$	$2.5 \cdot 10^{-2} \text{ kg/m}^3$	$Y_{\text{O}_2}$	1.46 kg cells/kg $\text{O}_2$
	$K_{\text{O}_2}$	$1.3 \cdot 10^{-2} \text{ kg/m}^3$	$Y_{x/s}$	0.5 kg cells/kg s
	E	19238.35 J/gmol	$Y_{p/x}$	0.35 kg p/kg cells
	$\Delta G_d$	2093.4 J	$Y_{\Delta}$	$2.38 \cdot 10^3 \text{ J/kg cells}$
	$\beta$	0 kg p/(kg cells•s)		
Operation Variables	Variable	Value	Variable	Value
	$\rho_R$	$1000 \text{ kg/m}^3$	$x_f$	0 $\text{kg/m}^3$
	$K_a$	$1.978 \cdot 10^{-5}$	$s_f$	10 $\text{kg/m}^3$
	H	$0.704 \text{ m}^3/\text{kg}$	$p_f$	0 $\text{kg/m}^3$
	$\lambda$	519 J/kg	$C_a$	$0.001 \text{ kg/m}^3$
	$T_{\text{ins}}$	303.59 K	$C_b$	$0.2 \text{ kg/m}^3$
	$T_{\text{inair}}$	298.15 K	$C_{ps}$	$3.1531 \cdot 10^3 \text{ (J/kg}\cdot\text{K)}$
	$V_R$	$6 \cdot 10^{-3} \text{ m}^3$	$C_{\text{pair}}$	$2.38846 \cdot 10^{-1}$ (J/kg•K)
			$C_{pR}$	$3.1531 \cdot 10^3 \text{ (J/kg}\cdot\text{K)}$

### 2.2.2 Invertase production model

The system considered is a fed-batch fermentation of *Saccharomyces cerevisiae* containing the plasmid pRB58. This plasmid contains the yeast SUC2 gene, which codes for the enzyme invertase. The expression of SUC2 gene is repressed when the glucose concentration in the medium is high. This characteristic allows the regulation of invertase production by manipulating the glucose feed rate in a fed-batch mode of operation. An important feature of the cell growth kinetics is the variation in the cell yield of glucose as a function of glucose concentration in the medium. At high glucose levels, the less energy efficient fermentative growth metabolism predominates and the cell yield is understandably low. A large part of the consumed glucose is directed toward ethanol production. On the contrary, at low glucose concentrations, the respiratory pathway becomes predominant. This shift from fermentative to respiratory metabolism, characterized by a drastic increase in cell yield, must be incorporated as an integral part of the cell growth model.

The experimental results needed to model development, including the kinetics of cell growth, glucose utilization, ethanol formation, and invertase production in batch and fed-batch fermentations, are described by Patkar and Seo (1992). Patkar et al. 1993 proposed a model for invertase expression based on his experimental results. The fermentation model proposed is a cybernetic model where the cells are viewed as optimal strategists involved in the maximization of a performance index; say the specific cell growth rate. This goal-oriented approach supposes that all the glucose is utilized for growth purposes and that the cell

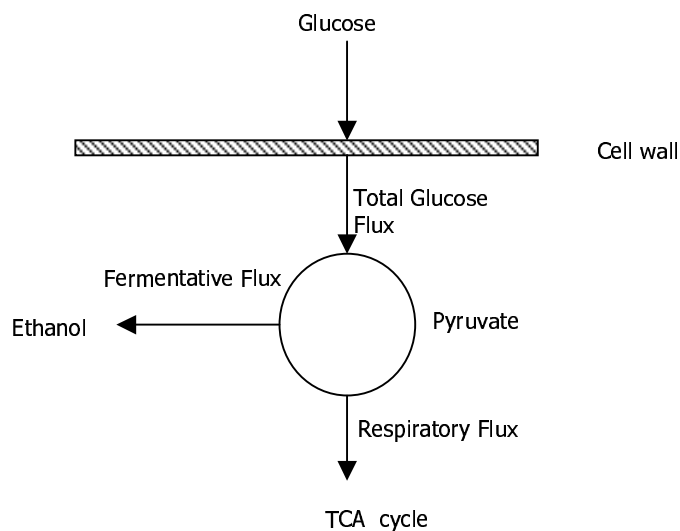


Figure 2.3 Schematic representation of fermentative and respiratory fluxes inside cloned *Saccharomyces cerevisiae* cell. Pyruvate is the bifurcation point at which the cell decides the fractions of glycolytic flux to be diverted toward fermentative and respiratory metabolic pathways in invertase fermentation

maintenance requirements can be neglected. Thus, the objective of the multitude of reactions within the cell is to maximize the cell growth rate. This occurs when the respiratory pathway dominates. Nevertheless this respiratory pathway gets saturated at glucose levels lower than those required to saturate the glucose uptake rate. The saturated respiratory flux is less than the saturated glucose uptake rate. The approach taken by this model, focuses on glucose fluxes

inside the cell rather than glucose concentration. The complex metabolic pathways are reduced to bare essentials considering only the respiratory ( $R_r$ ) and fermentative ( $R_f$ ) fluxes. A scheme of this metabolic pathway is shown in Figure 2.3. The glucose consumed by the cell first enters the glycolytic chain. There is a bifurcation point at pyruvate. Using the fermentative pathway, pyruvate can be converted to ethanol. Alternately, using the respiratory pathway, more energy yield is obtained through its carboxylation to oxaloacetate in the tricarboxylic-acid cycle. At low glucose concentrations, the respiratory pathway is not saturated and all the pyruvate produced is carboxylated. As the glucose concentration increases, the respiratory flux remains almost constant. Thus, increased glucose uptake rate results in an increased fermentative flux leading to a higher ethanol production rate.

The control model of this fermentation process is considered to have four outputs and one input. The input of the control system is the glucose feed rate ( $F$ ). The outputs are the biomass density (expressed as optical density), the glucose concentration ( $s$ ), the specific invertase activity ( $p$ ) and volume ( $V$ ). The state vector of this process is ( $x, s, p, V$ ). The fermentation process is simulated solving the model equations given below and the relations between the output process variables and the input used to control.

#### 2.2.2.1 Fermentation model

The above model framework can be written mathematically as follows. The specific growth rate  $\mu$  can be determined by assuming that the cell mass yield is constant for both the respiratory and the fermentative fluxes:

$$\mu = R_r Y_{xr} + R_f Y_{xf} \quad (2.15)$$

Simple Monod-type saturation forms are used to describe both the glucose uptake rate ( $R_t$ ) and the respiratory flux ( $R_r$ ) as a function of the bioreactor substrate concentration ( $s$ ). The uptake rate must be large enough to sustain the respiratory flux. Otherwise, the respiratory flux should be equal to the uptake flux rate,

$$R_t = \frac{k_t s}{K_t + s} \quad (2.16)$$

$$R_r = \min\left(\frac{k_r s}{K_r + s}, R_t\right) \quad (2.17)$$

where  $k_t$  is the maximum glucose uptake rate,  $k_r$  is the maximum respiratory flux and  $K_t$  and  $K_r$  are model parameters.

The fermentative flux is expressed as,

$$R_f = R_t - R_r \quad (2.18)$$



### 2.2.2.2 Mass balances

The overall mass balance for a fermentation process in fed-batch mode of operation, is given by,

$$\frac{dV}{dt} = F \quad (2.19)$$

since the fermentation volume (V), changes with time according to the substrate feed rate (F).

A balance of the biomass concentration yields

$$\frac{d(xV)}{dt} = \mu xV = (R_r Y_{xr} + R_f Y_{xf}) xV \quad (2.20)$$

While the substrate concentration balance is given by

$$\frac{d(sV)}{dt} = F s_F - R_t xV \quad (2.21)$$

The process of regulating the gene SUC2 expression is very complex and only understood partially. It is impossible to formulate a detailed mathematical model that takes into account the complete interplay among genes. Even if the mechanisms were known, the number of parameters in such a detailed, structured model, would be quite large. An empirical approach is more suitable. Invertase production is derepressed at low glucose concentrations. Also, the specific invertase activity responds almost instantaneously to changes in the extracellular glucose concentration. So, it can be assumed that the rate of transcription of SUC2 gene is an explicit function of the extracellular glucose concentration. On the basis of this assumption a simple substrate inhibition type of expression is used to describe the invertase expression (p) as a function of glucose concentration in the medium,

$$\frac{d(pxV)}{dt} = \left[ \frac{k_p s}{K_p + s + K_i s^2} - k_d p \right] xV \quad (2.22)$$

The yields and the kinetic parameters used in the above equations, the initial values of the fermentation process and the operating conditions are shown in Table 2.2.

**Table 2.2** Initial values of the fermentation process, the microorganism parameters and the values of the operation and control variables at the operation point used to simulate the invertase production fermentation process

Initial Values	Variable	Value
	x	0.15 OD
	s	1, 1.5, 2, 3, 5 kg/m <sup>3</sup>
	p	100000 units/(OD•m <sup>3</sup> )
	V 10 <sup>3</sup>	0.2, 0.4, 0.5 0.6, 0.7, 0.8, 0.9, 1 m <sup>3</sup>
	F 10 <sup>7</sup>	0-0.75611 m <sup>3</sup> /s
Microorganism Parameters	Parameter	Value
	k <sub>t</sub>	1.25 g/(h•OD)
	K <sub>t</sub>	0.95 kg/m <sup>3</sup>
	k <sub>r</sub>	0.55 g/(h•OD)
	K <sub>r</sub>	0.05 kg/m <sup>3</sup>
	Y <sub>sr</sub>	600 OD/kg
	k <sub>p</sub>	6.25 units/(l•OD•h)
	K <sub>p</sub>	0.1 kg/m <sup>3</sup>
	K <sub>i</sub>	2.0 m <sup>3</sup> /kg
	k <sub>d</sub>	1.85 1/h
	Y <sub>xf</sub>	150 OD/kg
Operation Variables	Variable	Value
	S <sub>F</sub>	10 kg/m <sup>3</sup>
	Vessel Volume	1.2 10 <sup>-3</sup> m <sup>3</sup>
	Max. fermentation time (h)	24 h

### 2.2.3 Penicillin production model

Penicillin production in a fed-batch bioreactor was studied by Cuthrell and Biegler (1989). These authors tried to find the optimal feed-rate profile for the biosynthesis of penicillin. The penicillin fed-batch bioreactor model was first proposed by Lim et al. (1986). This model has been used by several authors, including Luus (1993a) and Banga et al. (1997), to test optimal control techniques. Cuthrell and Biegler. (1989) based the solution of the optimal control problem stated by this fed-batch fermentation in a successive quadratic programming and orthogonal collocation on finite elements. Luus (1993a) used iterative dynamic programming to find this optimal feed-rate profile. Finally Banga et al. (1993) presented a fast stochastic dynamic optimization method to solve the same optimal control problem. He called his method integrated controlled random search for dynamic systems.

The studied fermentation system is a bioreactor that contains biomass ( $x$ ), substrate ( $s$ ), and product ( $p$ ) at certain concentrations and has a volume ( $V$ ). The control variable of this process is the mass feed rate of substrate ( $F$ ). The state variables are  $x$ ,  $s$ ,  $p$  and  $V$ , as before. The overall fermentation process lead-time is between 72 to 200 h.

#### 2.2.3.1 Fermentation model

The differential equations used to describe the penicillin biosynthesis are stated below. As before, the fermentation process is carried out in a fed-batch mode of operation, and the mass substrate feed rate determined the volume change with time,

$$\frac{dV}{dt} = \frac{F}{s_F} \quad (2.23)$$

where  $s_F$  is the substrate density. A biomass balance results in

$$\frac{dx}{dt} = \mu x - \frac{x F}{s_F V} \quad (2.24)$$

where  $\mu$  is a Monod type growth rate of biomass given by

$$\mu(x, s) = \mu_{\max} \frac{s}{K_s + s} \quad (2.25)$$

A substrate concentration mass balance yields

$$\frac{ds}{dt} = -\frac{\mu x}{Y_{x/s}} - \frac{\rho x}{Y_{p/x}} - \frac{m_s s x}{K_m + s} + \left(1 - \frac{s}{s_F}\right) \frac{F}{V} \quad (2.26)$$

where  $Y_{x/s}$  and  $Y_{p/x}$  are the substrate and product yields, respectively, and  $m_s$  is a composite constant for the Monod type microorganism maintenance term. The production rate of penicillin ( $\rho$ ) is given by:

$$\rho(s) = \rho_{\max} \frac{s}{K_p + s(1 + s/K_{in})} \quad (2.27)$$

Finally, the penicillin mass balance considers a degradation constant rate

$$\frac{dp}{dt} = \rho x - K_{\text{degr}} r p - \frac{pF}{s_F V} \quad (2.28)$$

The yields and the kinetic parameters used in the above equations, the initial values of the fermentation process and the operating conditions are shown in Table 2.3.

**Table 2.3** Initial values of the fermentation process, microorganism parameters and operation and control variables constraints for penicillin biosynthesis

Initial Values	Variable	Value
	x	1.5 kg/m <sup>3</sup>
	s	0 kg/m <sup>3</sup>
	p	0 kg/m <sup>3</sup>
	V	7 · 10 <sup>-3</sup> m <sup>3</sup>
Microorganism Parameters	Parameter	Value
	μ <sub>max</sub>	0.11 1/h
	K <sub>s</sub>	0.006 kg s/kg cells
	Y <sub>x/s</sub>	0.47kg cells/kg s
	Y <sub>p/x</sub>	1.2 kg p/kg s
	m <sub>s</sub>	0.029 kg s/ (kg cells•h)
	ρ <sub>max</sub>	0.0055 kg p/kg cells•h)
	K <sub>p</sub>	0.0001 kg/m <sup>3</sup>
	K <sub>in</sub>	0.1 kg/m <sup>3</sup>
	K <sub>degr</sub>	0.0001 1/h
Operation variables and process constraints	Variable	Value
	S <sub>F</sub>	500 kg/m <sup>3</sup>
	x	< 40 kg/m <sup>3</sup>
	s	< 100 or < 25 kg/m <sup>3</sup>
	Vessel volume	< 1 · 10 <sup>-2</sup> m <sup>3</sup>
	F	< 50 g/h

### 2.2.4 Other models

A simple fermentation model was studied by Agrawal et al. (1982). He considered a pure culture in a constant-volume continuous stirred tank reactor, which is fed with sterile nutrient medium. The mass balances equations on cells ( $x$ ) and limiting substrate ( $s$ ) are

$$\frac{dx}{dt} = -\frac{Fx}{V} + \mu(s)x \quad (2.29)$$

$$\frac{ds}{dt} = -\frac{F(s_F - s)}{V} + \sigma(s)x \quad (2.30)$$

where  $\mu$  is the specific growth rate,  $s_F$  the feed substrate concentration,  $F$  the volumetric feed flow rate, and  $V$  the bioreactor volume. The specific substrate consumption rate ( $\sigma$ ) is given by

$$\sigma(s) = \mu(s)/Y(s) \quad (2.31)$$

where  $Y(s)$  is the substrate yield.

Agrawal et al. (1982) carried out studies about the stability and multiplicity of the possible steady states assuming a linear variable yield coefficient with a Monod type growth model. Also, he studied the stability of this system using a substrate inhibition model for the growth rate and a linear variable yield coefficient. The substrate inhibition model is given by

$$\mu(s) = kse^{-s/K} \quad (2.32)$$

In dimensionless form and assuming a linear equation for substrate yield, equations (2.29) and (2.30) using (2.31) and (2.32) can be written as

$$\frac{dC_1}{dt} = -C_1w + C_1(1 - C_2)e^{C_2/\gamma} \quad (2.33)$$

$$\frac{dC_2}{dt} = -C_2w + C_1(1 - C_2)e^{C_2/\gamma} \frac{1 + \beta}{1 + \beta - C_2} \quad (2.34)$$

where  $w$  is the dilution rate  $F/V$  and is used as a control parameter,  $C_1$  is the dimensionless cell mass and  $C_2$  is the dimensionless substrate concentration defined as

$$C_2 = \frac{(s_F - s)}{s_F} \quad (2.35)$$

In equations (2.33) and (2.34)  $\beta$  and  $\gamma$  are the model parameters. This author found that necessary and sufficient conditions for the existence of limit cycles are satisfied if

$$0.3 < \gamma < 0.5 \text{ and } \beta < 1/(4 - 1/\gamma) - 0.5 \quad (2.36)$$

Through simulation it was found that if  $\gamma=0.48$ ,  $\beta=0.02$  and the initial conditions are  $C_1=0.1207$  and  $C_2=0.881$ , the system reaches a stable steady state for  $w < 0.8$ . The bioreactor oscillates when  $0.8 < w < 1.205$  and there is a washout of the microorganisms in the bioreactor when  $w=1.205$ .

The above system has been studied extensively (see for example Narendra, 1995) because it is difficult to control for several reasons. The uncontrolled equations are highly non-linear and exhibit limit cycles. Optimal behavior occurs in or near an unstable region. Besides, the problem exhibits multiplicity since two different values of the control parameter- dilution rate- can lead to the same desired set point in cell mass yield.

Another fermentation process of current research interest is the ethanol production from cellulose. Ethanol can be used as the basis for fuels since cellulose is an abundant renewable resource. Philippidis et al. (1993, 1997) studied the continuous production of ethanol from cellulose using *Saccharomyces cerevisiae* yeast. He proposed a detailed model of the simultaneous saccharification and fermentation process. Current efforts are focused on the reduction of the overall production cost to make ethanol a non oil-based competitive fuel. The production cost could be lowered with an effective process control strategy. This research topic is still open.

## Chapter 3 : Artificial Neural Networks

### 3.1 Introduction

#### 3.1.1 Where they come from? Its relation with biological neural systems

An artificial neural network (NN) is a mathematical algorithm built in such a way that it tries to emulate the human brain. These mathematical models of the brain have been studied extensively with the hope of achieving human-like performance, especially in the fields of speech and image recognition, perception and motor control (Quantrille and Liu, 1991; Hertz and Krogh, 1993; Bishop, 1995; Randall et al. 2000) A human being can discern a whisper on a noisy room, a face in a dimly lit alley and a hidden agenda in a political statement. Tasks of lesser complexity will take days on a huge conventional computer. Most impressive of all, the brain learns -without any explicit instruction- to create the internal representations that make these skills possible. Also, the brain is very efficient in energy consumption. It has been calculated that  $10^{-16}$  J/s are needed per brain operation, while  $10^{-6}$  J/s are needed in a computer operation. However, the human brain is relatively slow. Brain events take milliseconds against the nanoseconds of any event on a silicon chip. Although, the supremacy of the human brain over the silicon chip is believed to lay in its dense interconnection of simple computational units connected in a parallel way.

The human brain contains about  $10^{11}$  nerve cells, named *neurons*, interconnected via the so-called *synapses*. About 1000 synapses of various types are connected to a single neuron. To take advantage of this high interconnection and parallelism, artificial neural networks algorithms have been thought. A schematic drawing of a typical brain neuron is shown in Figure 3.1. The body cell called *soma* is surrounded by *dendrites*. These structures are tree-like networks of nerve fiber. Extending from the cell body is a single long fiber called the *axon*, which eventually *arborizes* into strands and substrands. At the ends of these are the transmitting ends of the *synaptic junctions* or *synapses* to other neurons. The

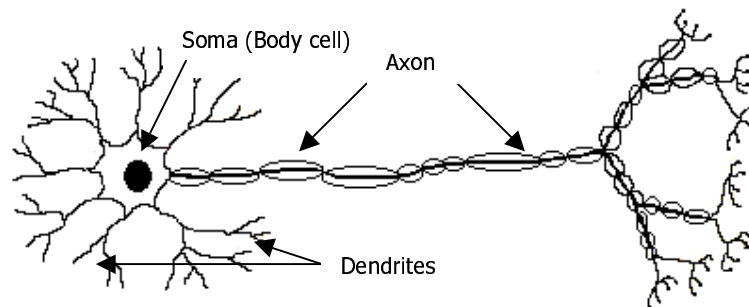


Figure 3.1 Schematic drawing of a biological neuron

receiving ends of these junctions on other cells can be found both on the dendrites and on the cell bodies themselves. The axon of a typical neuron makes a few thousand synapses with other neurons. The neurons transmit signals by propagating an electrical impulse along the entire length of the cell in the form of a change in potential of cell's membrane. The effect is to raise or lower the electrical potential inside the body of the receiving cell. If this potential reaches a threshold, a pulse of fixed strength and duration is sent to the axon. It is said that the cell has fired and the pulse branches out through the axonal arborization to the synaptic junctions to other cells. An impulse triggers a release of neurotransmitters, specific signaling substances, which refuse to bind to the receptors on the adjacent neuron. The synaptic strength of the junctions is supposed to be modified when knowledge is stored in the brain. The first studies about the neural system were carried out by the Spanish scientist Ramon y Cajal during the last decades of nineteenth century and the beginning of twentieth. One of his drawings is shown in Figure 3.2. It is amazing the similarity of his illustration with the modern microscopy pictures, shown in Figure 3.3.

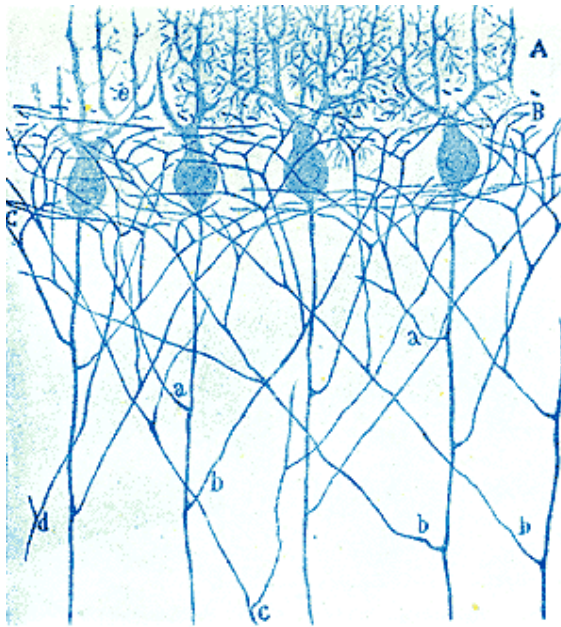


Figure 3.2 Ramon y Cajal drawing of the cerebellar cortex and neural tissue for a cat. From *Histologie du systeme nerveux de l'homme et des vertebres* 1909

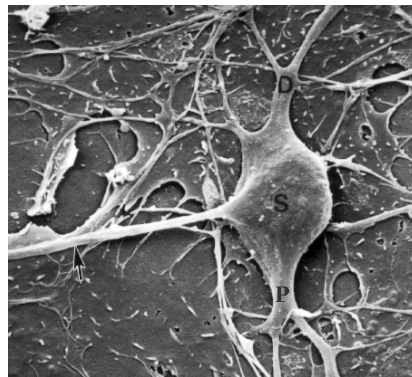
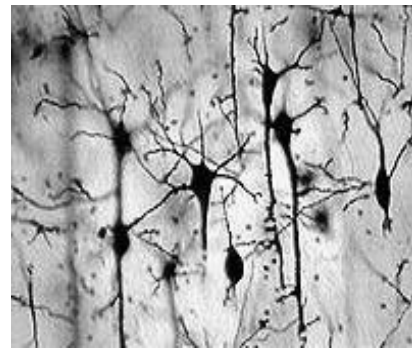


Figure 3.3 **Upper picture:** Hippocampus neurons with permission of *Slice of Life* project. **Lower picture:** Pyramid neuron grown in culture. From Vision Concepts (© information in [www.4colorvision.com/citation.htm](http://www.4colorvision.com/citation.htm))



### 3.1.2 Mathematical modeling of biological neural systems: Artificial neural networks

Similarly to the brain, an artificial neural network consists of a number of interconnected artificial neurons as processing units. These neurons also process information by their dynamic response to external inputs, but in comparison to the brain an artificial neural network model is, however, relatively simpler. The neural network model is composed of many non-linear parallel computational units connected via connection weights that are typically adapted during operation to improve performance. One computational unit, called neuron, performs a part of the overall network computation. Several of these neurons constitute a neural network (Hertz and Krogh, 1993; Bishop, 1995)

A schematic drawing of a typical artificial neuron is shown in Figure 3.4. The model neuron computes a weighted sum of its  $j$  inputs from another units, and outputs according to a threshold or activation function  $f$ . Mathematically, this can be written as:

$$output = f\left(\sum_j w_j input_j - \theta\right) \quad (3.1)$$

where  $\theta$  is an adjustable constant term called bias or threshold. The activation function  $f$  can be a step function (neuron is active or inactive) but commonly it is an s-shaped or sigmoidal function. It is normally wanted that the activation function saturate at both extremes of the input range. Any function with this feature will fit but it is desirable to be a differentiable function because the gradient of  $f$  is used in common neural network learning algorithms. Also, this function is chosen in a way that does not saturate in the range of the weighted sum of inputs. The output of the activation function is usually within the interval  $(-1,1)$  or  $(0,1)$ .

There has been a recent resurgence in the field of artificial neural nets caused by new topologies and algorithms, analog VLSI implementation techniques, and the assumption that massive parallelism is essential for high computational performance. Besides parallelism, another distinctive characteristic of neural networks is that the output of the network is the state of the entire network and not the result of only a single processing unit. This characteristic, and the fact that memory allocation in a neural network is not physically addressable, makes an artificial neural network model a very robust model in the sense that a loose of few neurons does not disrupt the entire network performance. Another similarity of artificial neural networks with the human brain is in its way of learning. Children learn to recognize shapes by trial and error. A common toy given to toddlers consists of different solid shapes that can be inserted in a box only through correspondingly shaped holes. As the child learns about shapes by repeatedly trying to fit the solid shapes into the holes, a neural

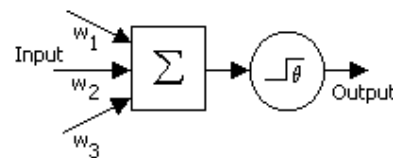


Figure 3.4 Schematic drawing of a typical artificial neuron. The output is a function of the inner product of the input vector and the weight vector of the connections  $\mathbf{w}$

network uses a set of examples into a trial and error algorithm to learn its task. These examples are called patterns and the set of patterns is called training or learning set. The learning process involved is called training. To train a neural network means to adaptatively adjust the values of the connection weights in order to improve performance. Training stops when the specific task is learnt or a given performance criterion is achieved. Usually to accomplish this, the network has to be feed several times with the entire training set. Each time the entire training set is feed to the neural network is called an *epoch*.

### **3.1.3 Neural network classification: types of artificial neural networks**

Mathematically, neural networks can be seen as universal approximators of computable functions (Cybenko, 1989; Hornik et a., 1989). It means that in principle, neural networks can estimate any computable function, i.e., they can do everything a normal digital computer can do. Artificial neural networks commonly used in the modeling and identification fields are deterministic algorithms. However neural networks can also be stochastic. Stochastic networks, such as the Boltzmann Machine, are used mostly in pattern recognition (Widrow and Lehr, 1990). There are several types of neural networks, but they can be divided by its learning method into supervised learning or unsupervised learning (Sarle, 1997). In supervised learning, the correct network results (target values or desired outputs) are known and are presented to the neural network during training so that the neural network can adjust its weights while matching its outputs to the target values. After training, the neural network performance is tested by presenting to it only input values and seeing how close it comes to predict the correct target values. In unsupervised learning, the neural network is not provided with the correct results during training. Supervised neural networks usually perform regression. Unsupervised neural network in contrast usually perform some kind of data compression, such as dimensionality reduction or clustering.

On the other hand, neural networks can also be divided by their network topology. Two major kinds of network topology are feedforward and recurrent neural networks. In a feedforward neural network, the connections between units do not form cycles; this is called straight-through connection. They usually produce a response to an input quickly. Most of them can be trained using a wide variety of efficient conventional optimization methods. In a feedback or recurrent neural network, there are cycles in the connections. Usually each time an input is presented, the neural network must iterate for a potentially long time before it produces a response. This kind of networks is usually more difficult to train than feedforward neural network but they can be successfully implemented to model dynamic systems that feedforward neural networks cannot model.

Neural networks also differ in the type of data they accept. Two major kinds of data are categorical and quantitative. By categorical variables is understood those variables that take only a countable number of possible values. There are usually several or more cases falling

into each category. Categorical variables have symbolic values such as "white" or "gray" that must be encoded into numbers before feeding them to the network. Both supervised and unsupervised learning with categorical outputs are called classification. Quantitative variables are numerical measurements of some attribute, such as temperature. The measurements must be made in such a way that at least some arithmetic relations among the measurements reflect analogous relations among the attributes of the objects that are measured. Supervised learning with quantitative target values is called regression. Some variables can be treated as either categorical or quantitative, such as number of children or any binary variable. Most regression algorithms can also be used for supervised classification by encoding categorical target values as 0/1 binary variables and using those binary variables as target values for the regression algorithm. The outputs of the network are probabilities and, in this case, any of the most common training methods can be used.

Finally a useful scheme for neural network classification was presented by Sarle (1997) and it is reproduced in Table 3.1. Also, this scheme presents a resume of the relevant works in the neural networks field. The following sections contain detailed references to some of those works. For further information refer to Sarle (1997).

**Table 3.1** Neural network classification scheme adapted from Sarle (1997)

1. Supervised learning algorithms
  - A. Feedforward neural networks
    - Linear
      - Hebbian - Hebb (1949), Fausett (1994)
      - Perceptron - Rosenblatt (1958), Minsky and Papert (1969/1988), Fausett (1994)
      - Adaline - Widrow and Hoff (1960), Fausett (1994)
      - Higher Order - Bishop (1995)
      - Functional Link - Pao (1989)
    - MLP: Multilayer perceptron - Bishop (1995), Reed and Marks (1999), Fausett (1994)
      - Backprop - Rumelhart, Hinton, and Williams (1986)
      - Cascade Correlation - Fahlman and Lebiere (1990), Fausett (1994)
      - Quickprop - Fahlman (1989)
      - RPROP - Riedmiller and Braun (1993)
    - RBF networks - Bishop (1995), Moody and Darken (1989), Orr (1996)
      - OLS: Orthogonal Least Squares - Chen, Cowan and Grant (1991)
    - CMAC: Cerebellar Model Articulation Controller - Albus (1975), Brown and Harris (1994)
    - Classification only
      - LVQ: Learning Vector Quantization - Kohonen (1988), Fausett (1994)
      - PNN: Probabilistic Neural Network - Specht (1990), Masters (1993), Hand (1982), Fausett (1994)
    - Regression only
      - GNN: General Regression Neural Network - Specht (1991), Nadaraya (1964), Watson (1964)

- B. Recurrent neural networks - Hertz, Krogh, and Palmer (1991), Medsker and Jain (2000)
    - BAM: Bidirectional Associative Memory - Kosko (1992), Fausett (1994)
    - Boltzmann Machine - Ackley et al. (1985), Fausett (1994)
    - Recurrent time series
      - Backpropagation through time - Werbos (1990)
      - Elman - Elman (1990)
      - FIR: Finite Impulse Response - Wan (1990)
      - Jordan - Jordan (1986)
      - Real-time recurrent network - Williams and Zipser (1989)
      - Recurrent backpropagation - Pineda (1989), Fausett (1994)
      - TDNN: Time Delay NN - Lang, Waibel and Hinton (1990)
  - C. Competitive
    - ARTMAP - Carpenter, Grossberg and Reynolds (1991)
    - Fuzzy ARTMAP - Carpenter, Grossberg, Markuzon, Reynolds and Rosen (1992), Kasuba (1993)
    - Gaussian ARTMAP - Williamson (1995)
    - Counterpropagation - Hecht-Nielsen (1987; 1988; 1990), Fausett (1994)
    - Neocognitron - Fukushima, Miyake, and Ito (1983), Fukushima, (1988), Fausett (1994)
2. Unsupervised learning algorithms - Hertz, Krogh, and Palmer (1991)
- A. Competitive
    - Vector Quantization
      - Grossberg - Grossberg (1976)
      - Kohonen - Kohonen (1984)
      - Conscience - Desieno (1988)
    - Self-Organizing Map
      - Kohonen - Kohonen (1995), Fausett (1994)
      - GTM: - Bishop, Svensén and Williams (1997)
      - Local Linear - Mulier and Cherkassky (1995)
    - Adaptive resonance theory
      - ART 1 - Carpenter and Grossberg (1987a), Moore (1988), Fausett (1994)
      - ART 2 - Carpenter and Grossberg (1987b), Fausett (1994)
      - ART 2-A - Carpenter, Grossberg and Rosen (1991a)
      - ART 3 - Carpenter and Grossberg (1990)
      - Fuzzy ART - Carpenter, Grossberg and Rosen (1991b)
    - DCL: Differential Competitive Learning - Kosko (1992)
  - B. Dimension Reduction - Diamantaras and Kung (1996)
    - Hebbian - Hebb (1949), Fausett (1994)
    - Oja - Oja (1989)
    - Sanger - Sanger (1989)
    - Differential Hebbian - Kosko (1992)
  - C. Auto-association
    - Linear auto-associator - Anderson et al. (1977), Fausett (1994)
    - BSB: Brain State in a Box - Anderson et al. (1977), Fausett (1994)
    - Hopfield - Hopfield (1982), Fausett (1994)
3. Nonlearning
- A. Hopfield - Hertz, Krogh, and Palmer (1991)
  - B. Various networks for optimization - Cichocki and Unbehauen (1993)

### 3.2 Neural networks history : Major milestones in the development of neural computation

The first formal model of a neural network was proposed by McCulloch and Pitts in 1943. In this model the neuron computes a weighted sum of its inputs from other units, and outputs a one or a zero according whether this sum is above or below a certain threshold. Though simple, a McCulloch-Pitts neuron is computationally a powerful device. They demonstrated that a synchronous assembly of such neurons is capable in principle of universal computation for suitably chosen weights. This means that it can perform any computation that an ordinary digital computer can, though not necessarily so rapid or conveniently. Few years later, in 1949, Hebb postulated in his *Organization of Behavior* the learning scheme. An assembly of neurons can learn by strengthening the connections between two neurons whenever they were both simultaneously excited. Almost a decade later, in 1958, Rosenblatt developed his Perceptron. i.e., a simple formalized model of a biological neuron based on the McCulloch-Pitts neurons and Hebb's postulate. The perceptron consist of sensory units connected to a single layer of McCulloch-Pitts neurons. Perceptrons are networks in which the units are organized into layers with feed-forward connections between one layer and the next. Rosenblatt focused his work on the problem of how to find appropriate weights values for particular computational tasks. Rosenblatt perceptron is very similar to a network called *adaline*, proposed by Hoff (1960) and Widrow (1962).

For the simplest case of perceptrons without any intermediate layers, Rosenblatt was able to prove a theorem that states that the training of a perceptron to classify patterns into linearly separable classes converges in a limited number of steps. He probed the convergence of a learning algorithm, a way to change the weights iteratively so that a desired computation can be performed. Many people expressed a great deal of enthusiasm and hope that such machines could be a basis for artificial intelligence.

However in 1969, Minsky and Papert thoroughly analyzed perceptrons and proved mathematically that a perceptron could not implement the XOR function or any non-linearly separable problem. They recognized that the extension of perceptrons with hidden units would probably overcome this limitation. Although, Rosenblatt had also studied network structures with more layers of units, there was no learning algorithm known which could determine the weights necessary to implement the given calculation. Minsky and Papert (1969) doubted that one of such algorithm could be found. They stated that training of intermediate units was probably unsolvable. They thought that it was more profitable to explore other approaches to artificial intelligence. The impact of Minsky and Papert research was destructive to the just born neural network field. The interest in neural networks disappeared in the following decades. Most of the computer science community left the neural network paradigm for almost 20 years.

Still, there were a number of people who continued to develop neural networks theory in the 1970's. A major theme was associative content-addressable memory, in which different input patterns become associated with another if sufficiently similar. During these years, Grossberg (Carpenter and Grossberg, 1991) developed his Adaptive Resonance Theory (ART), a number of novel hypotheses about the underlying principles governing biological neural systems. These ideas served as the basis for later work by Carpenter and Grossberg involving three classes of ART architectures. These structures were the first self-organizing neural implementations of pattern clustering algorithms. Other important theory on self-organizing systems was pioneered by Kohonen in 1982 with his work on feature maps. In the same year, Hopfield demonstrated formal analogy between a net of neuron-like elements with symmetric connections and a spin glass. Hopfield was able to add some physical insight by introducing an energy function, and by emphasizing the notion of memories as dynamically stable attractors. He demonstrated that a net could be trained as an associative memory using an early rule for synaptic weight modification proposed by Hebb. Hopfield introduced outer product rules as well as equivalent approaches to train a class of recurrent networks, now called Hopfield models. Other significant models of the 1980's include probabilistic ones such as Hinton, Sejnowski and Ackley's Boltzmann Machine (Hinton et al., 1984; Hinton and Sejnowski, 1986), which, to over simplify, is a Hopfield model that settles into solutions by a simulated annealing process governed by Boltzmann statistics. The Boltzmann Machine is trained by a clever two-phase Hebbian-based technique.

But perhaps the most influential development of this decade, takes up the old thread of Rosenblatt's perceptrons where it was cut by the work of Minsky and Papert. Almost simultaneously various people developed a learning algorithm that adjusts well the weights of the connections of the successive layers of a multiplayer perceptron. Known as backpropagation it appears to be first discovered by Werbos in 1974. He first published this algorithm in his doctoral dissertation. Unfortunately, Werbo's work remained almost unknown in the scientific community. The backpropagation algorithm was largely ignored for years after its development, probably because its usefulness was not fully appreciated. In the early 1980's Rumelhart at the University of California and Parker at Stanford University, independently rediscovered this algorithm. Rumelhart, Hinton and Williams (1986) popularized the algorithm by demonstrating that it could teach the hidden units to produce interesting representations of complex input patterns (Rumelhart et al. 1986). They finally succeeded in making backpropagation algorithm widely known, largely as a result of the clear framework within which they presented their ideas. The backpropagation algorithm is a learning algorithm for perceptron networks with hidden units based on Widrow and Hoff learning. It is the most widely used learning algorithm. It is most useful in situations in which the relation between input and output is non-linear and training data are abundant. It should be noted however, that in the field of variational calculus the idea of backpropagation error

through non-linear systems existed centuries before Werbos first thought to apply this concept to neural networks. In the past four decades, these methods have been used widely in the field of optimal control, as discussed by le Cun (1988).

In the last decade, neural networks have been widely applied in the field of pattern recognition (Bishop, 1995) and its relation with statistical models has been studied (Sarle, 1994). The business and management community have been also interested in neural networks, mostly for market price movement and lending risk predictions (Business Week, 1992). The scientific community entered a new era of neural networks applications (Hinton, 1992). In the chemical engineering field neural networks began to be interesting for industries (Bhagat, 1990, from Exxon Research and Engineering Co., Chitra, 1993 from Hercules Inc., and Hellstrom and Brinsley, 1993 from Westinghouse Electric Corporation, for example). A useful summary of the fundamental developments in feedforward artificial neural networks can be found in Widrow and Lehr (1990).

### **3.2 Advantages and disadvantages of neural networks : when they can be used**

The neural network field has developed so quickly that neural networks were seen as the major step towards artificial intelligence. It was thought that neural network algorithms were able to do anything. In theory, they are universal approximators of any continuous function (Cybenko, 1989; Hornik,1989). It means that a parameterization of any given function could be found through a suitable neural network. The main objection to this statement is that there is no clear methodology to follow. It is not clear how to find this suitable neural network. It depends on the type of problem to be solved. Sometimes it is almost unfeasible to fit a neural network. Some times the network is required to be too big and training takes a lot of time, sometimes the appropriate connection weights for the specific problem are not found. These are the main reasons why it is especially important to be clear about what neural networks can and cannot do.

The main advantages of neural network algorithms are that they collective do problem solving, they auto learn input/output characteristics of the specific problem, they are robust to noise and easy to implement in hardware and they are available in a wide range of software packages. Practical applications of neural networks most often employ supervised learning. For supervised learning, training data that include both the input and the desired result (the target value) have to be provided. After successful training, the network computes an output value that approximates the desired result when presented with input data. However, for training to be successful, sufficient training data and relatively long computational time are needed for training. Each forward and backward pass is computationally complex. Usually error derivatives have to be calculated for each training data input and each connection weight has to be updated several times. Besides, it is not guaranteed to actually find a global optimum and the training algorithm may not converge.

Moreover, in many applications, such as image and text processing, considerable effort is needed to select an appropriate input-output data set and to code the data as numerical values.

In practice, neural networks are especially useful for classification and function approximation/mapping problems which are tolerant of some imprecision, and almost as important as this, have lots of training data available but to which hard and fast rules (such as those that might be used in an expert system) cannot easily be applied. Almost any finite-dimensional vector function on a compact set can be approximated with an arbitrary precision by feedforward neural network if enough data and enough computing resources are available, although, it is still open the question about the network generalization/extrapolation capabilities.

To be somewhat more precise, Sarle (1997) found that feedforward networks with a single hidden layer and trained by least-squares are statistically consistent estimators of arbitrary square-integrable regression functions under certain practically-satisfiable assumptions regarding sampling, target noise, number of hidden units, size of weights, and form of hidden-unit activation function (see also White, 1990). Such networks can also be trained as statistically consistent estimators of derivatives of regression functions (White and Gallant, 1992) and quantiles of the conditional noise distribution (White, 1992). Feedforward networks with a single hidden layer using threshold or sigmoid activation functions are universally consistent estimators of binary classifications (Faragó and Lugosi, 1993; Lugosi and Zeger 1995; Devroye et al., 1996) under similar assumptions. Note that these results are stronger than the universal approximation theorems (Cybenko, 1989 and Hornik, 1989) that merely show the existence of weights for arbitrarily accurate approximations, without demonstrating that such weights can be obtained by learning.

Unfortunately, the above consistency results depend on one impractical assumption: that the networks are trained by an error minimization technique that comes arbitrarily close to the global minimum. Such minimization is computationally intractable except in small or simple problems. In practice, however, good results can be obtained without attempting a full-blown global optimization, e.g., using multiple random weight initializations is usually sufficient.

For example, a function that a typical neural net cannot learn is  $Y=1/X$  on the open interval  $(0,1)$ . An open interval is not a compact set. With any bounded output activation function, the error will get arbitrarily large as the input approaches zero. Of course, you could make the output activation function a reciprocal function and easily get a perfect fit, but neural networks are most often used in situations where you do not have enough prior knowledge to set the activation function in such a clever way. There are also many other important problems that are so difficult that a neural network will be unable to learn without memorizing the entire training set. This is the case when predicting random or pseudo-



random numbers, factoring large integers, determining whether a large integer is prime or composite, decrypting anything encrypted by a good algorithm and so on.

Feedforward neural networks are restricted to finite-dimensional input and output spaces. Recurrent neural networks can in theory process arbitrarily long strings of numbers or symbols, but they are more difficult to train. Neural networks are, at least today, difficult to apply successfully to problems that concern manipulation of symbols and rules, but research on this topic continues. Bibliography about this topic can be found in Sarle (1997).

Finally, it is important to understand that there are no methods for training neural networks that can magically create information that is not contained in the training data. Neural networks can extract only the information contained into the data set. They are black box representations of input/output data, although there are some techniques to interpret the network connection weights, as exposed by Chitra (1993).

As for simulating human consciousness and emotion, that's still in the realm of science fiction. Consciousness is still one of the world's great mysteries. Neural networks may be useful for modeling some aspects of or prerequisites for consciousness, such as perception and cognition, but they provide no insight into the question: "Why is all this brain processing accompanied by an experienced inner life? "

The advantages and disadvantages of neural networks can be summarized as suggested by Venkatasubramanian 2000.

Advantages:

- Collective problem solving
- Auto learning of Input/Output characteristics
- Robust to noise
- Pattern recognition
- Easy to implement
- Availability of software

Disadvantages:

- Often require thousands, ten of thousands of iterations to learn the data
- Not guaranteed to actually find the global optimum
- May not converge
- Each forward and backward pass is computationally complex
- Questionable generalization/extrapolation
- Black-box character

### **3.3 Some neural networks types**

As it was said in the previous section, for solving a given problem, a neural network architecture should be chosen on the basis of the characteristics of the problem. The

architecture of an artificial neural network is defined by the network structure; its components or neurons and the relations or connections between them. There are many alternative architectures of neural networks, but the feedforward layered backpropagation network of interconnected neurons is the most widely one applied in modeling a wide range of non-linear relationships and has been claimed to be especially suitable for chemical processes (Linko and Zhu, 1992). Feedforward layered networks are also called Multilayer perceptrons (MLP). A matter of current research is to find MLP optimal network architecture for a specific problem. To overcome this optimization problem, genetic algorithms have been used (Glasse et al., 1994) but in many cases, the improvement over an experienced worker carrying out trial and error procedures was small. Besides, soft computing, including decision trees and genetic programming are methodologies of current research in the development of optimal network architectures for control problems (Kawaji, 2002; Hinchliffe and Willis, 2002; Mirea and Marcu, 2002).

Another common network architecture is the Radial Basis Function (RBF) network. This network structure is similar to a feedforward neural network, but the response of the neuron to a given input is based on the distance between the input vector and a vector of parameters that describes the position of the neuron activity in the input space. Instead of using an inner product, usually a Euclidian distance is employed. Also, feedforward neural networks can be seen as a "static" network architecture because, for the neural networks used in this work, a fixed number of neurons and then connections between them are needed a priori, before training. On the other hand the RBF network architecture is built in a more dynamic manner. Both architectures are described in more detail below.

One more network architecture of interest is the Kohonen's Self Organized Maps. These maps are useful clustering algorithms and, by the characteristics of the bioreactor control problem treated in the present work, they are not used here and in consequence they will be explained briefly. Nevertheless, this algorithm has great potential in the development of further work. A self-organized map is a non-linear projection of the probability density function of the high dimensional input space into a 2 or 3-dimensional map.

As it was said before, another content-addressable neural algorithm is based in Grossberg's Adaptive Resonance Theory (ART). Fuzzy ART map systems learn to classify inputs by a fuzzy set of features. This characteristic makes this algorithm adequate for the current optimization and control problem. Fuzzy logic has been also included into several neural network algorithms to empower their capabilities. One example is Fuzzy ARTMAP but there are several networks that include fuzzy logic characteristics. The fuzzy ARTAMP system, and another systems based on fuzzy logic are described below. It is worth mentioning that a good and condensed summary of the characteristics of some of the mentioned architectures is found in Lippmann (1987). He explains Hopfield networks, Grossberg classifier, and perceptrons. Hertz and Krogh (1993) in his book on neural computation explores the relation

between statistical mechanics and neural networks. Another, now classical, book on neural networks is Bishop (1995), with a good introduction to pattern recognition through both statistical and neural networks methodologies.

### 3.3.1 Multilayer Perceptron

The basic features of a MLP network architecture are the input, hidden and output layers, each with a pre-specified number of neurons interconnected with a number of adjustable parameters called weights. A general scheme of a MLP is shown in Figure 3.5. To build a MLP network the number of neurons and layers, and the neuron processing activation function should be specified a priori. The number of neurons in the input and output layers is determined by the number of input and output variables involved in the problem to be solved, while the number of hidden layers and the corresponding number of neurons are related to the converging performance of the output error function during the training process, and to the network complexity, that increases the training time. The objective of training the network is to adjust the interconnection weights so that the application of a set of inputs produces the desired set of outputs. The optimal number of neurons in the hidden layers is usually determined by trial and error. Generally speaking, too few hidden neurons would limit the ability of the network to model the process, while too many could cause the solutions (weights) of the training process to yield local minima instead of a global one during the optimization problem, over fitting the neural network model.

Each neuron or processing unit (i) calculates the projection (inner product) between its input vector and the vector of the weights associated with each input connection (initially random) and (ii) uses the result of this product to compute the neuron output by means of a transfer function, also called threshold or activation function. The typical shape of this transfer function is a sigmoid, with a range fixed to the interval  $(-1,1)$  or  $(0,1)$ . Training is necessary to determine the weight vectors that will reproduce the relationships between input and output. As it was said before, only until 1986, when the backpropagation algorithm was rediscovered, a method to determine the

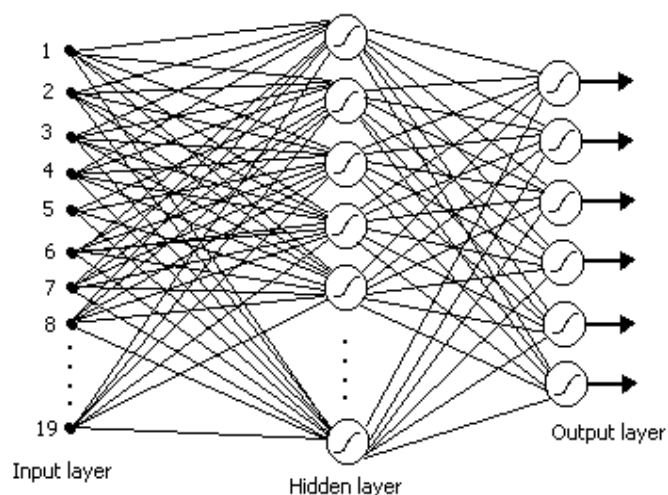


Figure 3.5 Scheme of a multilayer perceptron architecture. The activation function of each neuron is  $f(y) = 1/(1 + e^{-y})$  where  $y$  is the inner product of the input vector and the weight vector of the neuron input connections

weight vectors of each neuron was developed. This algorithm minimizes the error of the network predictions. Usually the error is defined as the quadratic difference between the network output and its target value. A measure of the entropy or a Minkowsky error function could be used as well. Any statistical way to measure the error will suffice. Anyway, it is necessary target data to calculate the network prediction error and to train the network. This is the reason why the backpropagation algorithm is called a supervised learning algorithm. There is a need of a set of pairs of input-output data do teach the network. Backpropagation updates each weight connection of each neuron based on the gradient of the error surface  $E$ ,

$$w_{ij}(t+1) = w_{ij}(t) - \eta \frac{\partial E}{\partial w_{ij}} \quad (3.2)$$

in the space defined by the connection weights  $w_{ij}$  from the  $j$ th neuron to the neuron  $i$ . The learning time step is denoted by  $t$  and the learning rate by  $\eta$ .

The gradient descent algorithm can be very slow if  $\eta$  is small, and can oscillate widely if  $\eta$  is too large. The problem essentially arises in the error surface valleys with sharp sides but a shallow slope along the valley floor. An adaptive learning parameter has been used to solve this problem in which  $\eta$  increment is defined by the increment or decrement of error. The backpropagation algorithm computes the error derivatives using the chain rule for the derivatives of the parameters on the hidden layers.

Let us assume the error measure or cost function as

$$E = \frac{1}{2} \sum_{j=1}^p (Y_j^t - Y_j)^2 \quad (3.3)$$

where  $p$  is the number of patterns of the training set,  $Y$  is the network output and  $Y^t$  is the desired output or target. The update rule for the weights of the last layer is

$$w_{ij}(t+1) = w_{ij}(t) - \eta \sum_{\mu=1}^p (Y_{\mu}^t - Y_{\mu}) f'(h_{i\mu}) V_{j\mu} \quad (3.4)$$

where  $V_j$  is the output of the hidden neuron  $j$  per pattern, and  $f'$  is the first derivate of the transfer or activation function of the  $j$  neuron. The overall input to neuron  $i$  for each pattern is

$$h_i = \sum_j w_{ij} V_j \quad (3.5)$$

See Figure 3.6 for index notation. Equation (3.4) can be written as

$$w_{ij}(t+1) = w_{ij}(t) - \eta \sum_{\mu=1}^p \delta_{i\mu} V_{j\mu} \quad (3.6)$$

with the error signal  $\delta_i$  given by

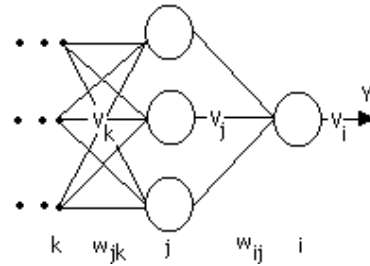


Figure 3.6 Scheme of a multilayer perceptron architecture showing the notation for neurons and weights

$$\delta_i = (Y^t - Y) f'(h_i) \quad (3.7)$$

For the connection weights  $w_{jk}$  of the preceding neuron the update rule is obtained using the chain rule

$$\begin{aligned} w_{jk}(t+1) &= w_{jk}(t) - \eta \frac{\partial E}{\partial w_{jk}} = w_{jk}(t) - \eta \sum_{\mu=1}^p \frac{\partial E}{\partial V_j} \frac{\partial V_j}{\partial w_{jk}} \\ &= w_{jk}(t) - \eta \sum_{\mu=1}^p (Y_\mu^t - Y_\mu) f'(h_{i\mu}) w_{ij} f'(h_{j\mu}) V_{k\mu} \\ &= w_{jk}(t) - \eta \sum_{\mu=1}^p \delta_{i\mu} w_{ij} f'(h_{j\mu}) V_{k\mu} \\ &= w_{jk}(t) - \eta \sum_{\mu=1}^p \delta_{j\mu} V_{k\mu} \end{aligned} \quad (3.8)$$

In the above equations (3.8)  $\delta_j$  is defined by

$$\delta_j = f'(h_j) \sum_i w_{ij} \delta_i \quad (3.9)$$

Note that equation (3.8) has the same form as equation (3.6) but with a different definition of  $\delta$ . In general, the backpropagation update rule has always the above form. Also equation (3.9) is the definition of  $\delta$  for all hidden layers.  $\delta$  is determined in terms of the  $\delta$ 's of the units that it feeds. The coefficients are just the forward connection weights, but here they are errors propagating backwards instead of signals propagating forward; this justifies the name error back-propagation algorithm. Although equations (3.6) and (3.8) are written as sums over all patterns in the training set, they are usually used incrementally. One pattern is presented at the input and then all weights are updated before the next pattern is considered. This decreases the cost function at each step. If the patterns are chosen in a random order, it also makes the path through weight space stochastic, allowing wider exploration of the cost surface. The alternative batch mode (taking 3.6 and 3.8 literally and only updating the weights after all patterns have been presented) requires additional local storage for each connection. Besides, the incremental approach seems superior in most cases, especially for very regular or redundant training sets.

The training algorithm can be described as follows:

1. Initialise the connection weights of each neuron to small random values. Usually in the interval (-1,1).
2. Choose an input-output pair so  $V_0 =$  input pattern. For a network of 3 layers,  $V_0=V_k$  (See Figure 3.6). Apply the input pattern to the input layer of the network and propagate the signal forwards though the network using

$$h_j = \sum_k w_{jk} V_k \quad (3.10)$$

and

$$V_i = f(h_j) \quad (3.11)$$

Proceed, until  $Y=V_i$  of the last layer is calculated

3. Compute each delta value for the output layer  $i$  using equation (3.7)
4. Compute each delta value for the preceding layers by propagating the error backwards, using equation (3.9), until a delta has been calculated for every neuron.
5. Update the connection weights, using:

$$w_{ij}(t+1) = w_{ij}(t) - \eta \delta_i V_j \quad (3.12)$$

6. Go back to step 2 and repeat for next pattern.

This training algorithm stops when the overall prediction error (usually a quadratic error) stabilizes or reaches a defined threshold. The above algorithm is extremely slow and many variations have been suggested to accelerate it. Other goals have included avoidance of local minima and improvement of generalization ability. For further details refer to Hertz and Krogh (1993). One modification of Backpropagation is called Backpropagation with variable step size. It was proposed by Magoulas et al., in 1997. Another training method was proposed by Elsimary et al. in 1993. His training method describes how to build a MLP network to be fault tolerant against weight perturbation. His results are useful especially in hardware implementation of neural networks.

One more option is to include a momentum term to avoid oscillation. The idea is to give each connection some inertia or momentum, so that it tends to change in the direction of the average downhill "force" that it feels, instead of oscillating wildly every little kick. This scheme is implemented by giving a contribution from the previous time step to each weight change

$$w(t+1) = w(t) - \eta \frac{\partial E}{\partial w} + \alpha(w(t) - w(t-1)) \quad (3.13)$$

The momentum term  $\alpha$  must be between 0 and 1; a value of 0.9 is often chosen. A small value of  $\alpha$  makes equation (3.13) become equation (3.4).

Other and faster algorithms have been proposed for MLP training, instead of steepest descent algorithm. For example, quasi-Newton and conjugate gradient optimization methods. Those algorithms and others used to train neural networks are implemented in MATLAB's Neural Network Toolbox. One faster learning algorithm is the Levenberg-Marquardt method. It is used when the cost function to be minimized is the sum of squared errors. For further details on all mentioned algorithms refer to Demuth and Beale (1998).

Another issue to be addressed is how many hidden neurons are needed in a MLP. Huang and Huang (1991) answer this question. His results are useful for pattern recognition and classification problems. They mathematically proved that a MLP with  $m$ -hidden neurons is capable of realizing arbitrary functions defined on an  $(m+1)$  element set.

### 3.3.2 Radial Basis Function

RBF networks share the basic architecture of MLPs, as shown in Figure 3.7, but use radial basis functions (such as equation (3.14)) in the hidden layer (RBF layer) and a linear transfer function in the output layer. The RBF units have a receptive field around their centre,  $\mu_j$ , for which the output of the neuron is maximal. Otherwise, the output tails off in a Gaussian form as the distance between the input and this centre increases. The width of the Gaussian bell is given by  $\sigma_j$ . The output of the RBF units is usually defined as

$$output = \exp\left(-\frac{\|input - \mu_j\|^2}{2\sigma_j}\right) \quad (3.14)$$

A RBF network is built when the input space is mapped to the output space through these Gaussian functions. This mapping produces a clustering of the input space into the different RBFs that form the network-hidden layer. The centre of each Gaussian function is not necessary a datum of the pattern set. The response of the RBF layer propagates throughout the output layer via some connections (weights) that are determined during the training process. Training consists both in the adaptation of centres and functions of the RBF layer to the input space and the determination of weights to the output layer. The number of neurons in the hidden layer, RBF units, depends on the complexity of the function to be mapped. Also, instead of  $\sigma_j$ , a covariance matrix  $\Sigma_j$  could be used for the RBF units

$$output = \exp\left\{-\frac{1}{2}(input - \mu_j)^T \Sigma_j (input - \mu_j)\right\} \quad (3.15)$$

In this way, a large number of adjustable parameters are available in a RBF network. If equation (3.15) is used,  $d(d+3)/2$  adjustable parameters are in a single neuron, where  $d$  is the dimension of the input vector. In RBF units following equation (3.14),  $d+1$  parameters can be adjusted.

While MLP neurons define hyper planes in the output space, RBF neurons define hyper spheres or hyper ellipsoids in that space. Another difference is that a MLP network is a distributed

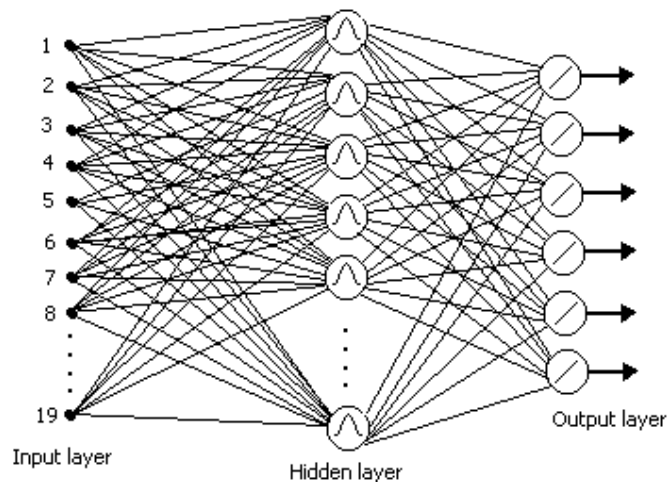


Figure 3.7 Scheme of Radial Basis Function network architecture. The activation function of each RBF neuron is  $f(y) = \exp\left(-\frac{\|y - \mu_j\|^2}{\sigma_j}\right)$  where  $y$  is the input vector,  $\mu_j$  is the center of the RBF neuron and  $\sigma_j$  is a measure of its width.

representation of the input space. For a given input, several neurons contribute to the output calculation. In RBF networks, for a given input, only a couple of neurons will transmit an active signal. It is said that RBF networks are a local representation of the input space.

Supervised learning can be used to train a RBF network. This network has the advantage over a MLP network, that a small quantity of target data is needed. The centres of the bell function can be chosen randomly from the training set through a backward selection or through a forward selection. Commonly, the width of the bell is twice the average distance between centres of adjacent neurons. The linear neurons connection weights are linear solution of the network-desired performance. A backward selection starts with a number of hidden units equal to the number of patterns in the training set. Then, neurons are removed one-by-one and each time the prediction error in a validation set is checked. The process continues until a certain threshold error is reached.

In forward selection, neurons are created one at a time. At each iteration, the input vector is used to create a RBF neuron. The error of the new network is checked, and if low enough, the training stops. Otherwise a new neuron is added. This procedure is repeated until the error goal is met, or the maximum number of neurons is reached. The described procedure is also coded in MATLAB's Neural Network toolbox. It can also be found in other software packages.

Clustering methods can also be used to find the centres of RBF neurons. For instance, k-means is a clustering algorithm where the squared distance between the chosen centres  $\mu_j$  and the input patterns  $x$  of the training set  $S$  is minimized,

$$\min \left( \sum_{j=1}^k \sum_{n \in S} \|x^n - \mu_j\|^2 \right) \quad (3.16)$$

In this equation (3.16)  $k$  is the number of clusters (RBF hidden neurons), and

$$\mu_j = \frac{1}{N_j} \sum_{n \in S} x^n \quad (3.17)$$

for  $N_j$  patterns  $x$ , with centre in  $\mu_j$ .

This algorithm can be described as follows:

1. The input patterns set is randomly partitioned
2.  $\mu$  is calculated through equation (3.17) and the closest centre to each pattern is chosen to evaluate distance

3. Regroup input  $x$ , to minimize  $\sum_{j=1}^k \sum_{n \in S} \|x^n - \mu_j\|^2$

4. Go to step 2 and repeat the sequence until a desired minimum is obtained

Finally, the widths of each RBF neuron, or its covariance matrix, are calculated from the variance of each cluster.



Learning Vector Quantization is another useful clustering algorithm. It is an especial case of Self-Organized Maps and it will be explained in detail in next section.

### 3.3.3 Self Organized Maps

A self-organized map (SOM) is an artificial neural network where the nodes become specifically tuned to various input signal patterns or classes of patterns by means of an unsupervised learning process. In the basic version, only a cell or local group of cells at a time response to the current input. The location of the response tends to become ordered as if some meaningful coordinate system for different input features was being created over the network. The spatial location or coordinates of a cell in the network correspond to a particular domain of input signal patterns. The learning results achieved by this kind of map seem very natural, at least indicating that the adaptive processes themselves at work in the map may be similar to those encountered in the brain. There may be therefore sufficient justification for calling these maps "neural networks" in the same sense of the multilayer perceptron and RBF networks.

A SOM defines a mapping from the input data space  $R^n$  onto a regular  $m$ -dimensional array of nodes. With every node  $i$ , a parametric reference vector  $\mathbf{m}_i \in R^n$  is associated. An input vector  $\mathbf{x} \in R^n$  is compared with each  $\mathbf{m}_i$  in any metric and the best match is defined as the winner and the input is thus mapped onto this location. In practical applications, the smallest of the inner products or of the Euclidean distances is used to define the best-matching node, identified by the subscript  $c$

$$c = \arg \min \|\mathbf{x} - \mathbf{m}_i\| \quad (3.18)$$

SOM is a non-linear projection of the probability density function of the high-dimensional input data onto a  $n$ -dimensional display. Usually  $n=2$ , so the map is a two dimensional lattice. It is necessary to find the  $\mathbf{m}_i$  vectors such as the responses of the map for a set of inputs  $\mathbf{x}$  are ordered spatially. In this case, to find appropriate values for  $\mathbf{m}_i$  is called learning. This learning principle assumes that the internal representations of information in the brain are generally organized spatially. Different regions in the brain are dedicated to specific tasks.

It is crucial to the formation of ordered maps that the cells or nodes being trained are not affected independently of each other, but as topologically related subsets. In the learning process, defining a neighborhood  $N_c$  around the node  $c$  enforces lateral interaction between nodes. At each learning step, all the nodes within  $N_c$  are updated, whereas nodes outside this region

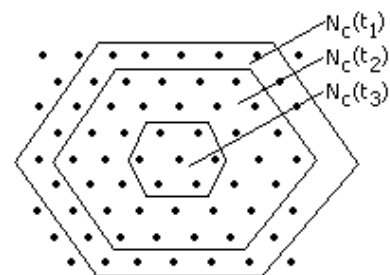


Figure 3.8 Example of a topological neighborhood  $N_c(t)$  in an hexagonal lattice of a SOM, where  $t_1 < t_2 < t_3$

are left intact. The width of  $N_c$  can be time-variable: wide at the beginning and shrinking monotonically with time. An example of this is shown in Figure 3.8. The learning process requires to:

1. Present an input  $\mathbf{x}$
2. Find the winner node  $c$  using equation (3.18), and
3. Update  $\mathbf{m}_i$  at each learning time-step  $t$  according to:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t)[\mathbf{x} - \mathbf{m}_i(t)] \quad \text{if } i \in N_c \quad (3.19)$$

If the node  $j$  is not inside the neighborhood then its parametric reference vector  $\mathbf{m}_j$  is not updated. The function  $\alpha$  is a scalar-valued adaptation gain which is initially equal to 1 but decreases monotonically with  $t$  afterwards. It is related to a similar gain used in the stochastic approximation processes. At the end of the learning process,  $\alpha$  should attain small values of the order of or less than 0.01.

An alternative to defining the neighborhood is to define a kernel function  $h_{ci}(t)$

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{ci}(t)[\mathbf{x} - \mathbf{m}_i(t)] \quad (3.20)$$

This equation becomes equation (3.19) when  $h_{ci} = \alpha$  if  $i \in N_c$  and  $h_{ci} = 0$  elsewhere. This kind of kernel function is nicknamed bubble, because it is related to certain activity bubbles in laterally connected neural networks. Another widely applied neighborhood kernel can be written in terms of the Gaussian function, because a biological lateral interaction often has the shape of a bell curve. Denoting the coordinates of the nodes  $c$  and  $i$  by the vectors  $\mathbf{r}_c$  and  $\mathbf{r}_i$ , respectively, a proper form for  $h_{ci}$  might be

$$h_{ci} = \alpha(t) \exp\left(\frac{-\|\mathbf{r}_i - \mathbf{r}_c\|^2}{2\sigma^2(t)}\right) \quad (3.21)$$

where  $\sigma$  is a suitable decreasing function of time. Its exact form is not crucial and it can be linear. Some practical hints for the construction of this kind of map are (Kohonen, 1990):

- a) Since learning is a stochastic process, the final statistical accuracy of the mapping depends of the number of learning steps, which must be reasonably large. A rule of thumb is that the number of steps must be at least 500 times the number of network units.
- b) If the neighborhood  $N_c$  is too small, the map will not be ordered globally. A wide starting  $N_c = N_c(0)$ , shrinking with time afterwards is advisable. The initial radius of  $N_c$  can even be more than half the diameter of the network. During the first 1000 steps the radius of  $N_c$  can shrink linearly. During the fine adjustment phase,  $N_c$  can still contain the nearest neighbors of the node  $c$ .
- c) Normalization of the inputs  $\mathbf{x}$  is not necessary, but it may improve numerical accuracy because the resulting reference vectors tend to have the same dynamic

range. A process that normalizes the reference vectors at each step can be found in Kohonen (1990)

If a SOM is to be used as a pattern classifier in which the nodes or their responses are grouped into subsets, then the problem becomes a decision process and a fine tuning of the map by learning Vector Quantization (LQV) is required. This fine-tuning clustering algorithm is used to determine the labels of the codebook vectors. The idea of LQV is to pull codebook vectors away from the decision surfaces to demarcate the class borders more accurately, like a Bayes classifier. The update rule for  $\mathbf{m}_i$  can be found in Kohonen (1990). To comply better LVQ classification algorithm with Bayes classifier, LVQ2 is proposed. In LQV2, a symmetric window around the discrimination surface between two classes is defined. A correction to the reference vector of a given class is made only if the input  $\mathbf{x}$  falls into the window on the wrong side of the discrimination surface. The optimal width of the window must be determined experimentally. This algorithm ought to be applied for a relatively short time only because, first, the distance between the reference vectors of two neighbors classes decreases monotonically. Second, if the tuning of the map is continued, it may lead to another asymptotic equilibrium of the reference vectors that is no longer the optimal. Further details are given elsewhere (Kohonen, 1990). To overcome the above limitations an improved algorithm LVQ3 was proposed. The main difference with LQV2 is the inclusion on the updating rule, of a parameter  $\epsilon$  when the input  $\mathbf{x}$  and the reference vectors of two consecutive nodes belongs to the same class.

It should be noted that a software package with all the above algorithms is freely available for any anonymous ftp user at the Internet site cochlea.hut.fi (130.233.168.48). It was developed by Kohonen and his coworkers at the Helsinki University of Technology. It can be used in MATLAB and it is supposed to compile in various machines without any specific modifications needed on the code, because all programs have been written in ANSI C.

Current research in the field consists of modifications to the Kohonen self-organizing maps. For example, one self-organizing neural network for non-linear mapping of data sets is called Curvilinear Component Analysis (Demartines and Herault, 1997). In this algorithm the output is not a fixed lattice but a continuous space able to take the shape of the submanifold. As it was said before, SOMs are most useful for pattern recognition and classification, and because the bioreactor control problem stated deals with a regression problem, SOM algorithm is not employed in the present work. This useful tool can be employed in the control-affine model of the fermentation (Aoyama et al., 1996) to map the possible steady states of the bioreactor process.

### **3.3.4 Fuzzy Neural Networks**

There are many successful applications of neural networks to non-linear model-based process control. However a potential disadvantage of the input-output modeling scheme is that

resulting models are complete black-box models. Hence, fuzzy logic is introduced to clarify the inner structure of such a model, i.e. to facilitate the inclusion of a priori knowledge of chemical process modeling, both analytical and qualitative (e.g. a linguistic statement). Fuzzy logic has been used before in control (up to 100 papers were presented at the 15th IFAC world congress in 2002), but there are very few applications of fuzzy logic control to chemical engineering, because of its highly complex nature. Therefore, a more promising application of fuzzy logic in chemical engineering seems to be a fuzzy logic model-based control scheme.

An introduction to fuzzy logic theory can be found in Tsoukalas and Uhrig (1997). The main contributor to this theory was Zadeh in 1965. In fuzzy logic, a fuzzy set is defined by a membership function that describes the degree of membership of a given element to the set. The membership function has values in the interval  $[0,1]$ , where 0 means that the element does not belong to the set and 1 means that it totally does so. As opposite to a fuzzy set, in a crisp set, the membership function only has two values, 1 or 0, since the element belongs or not to the set. Also, in fuzzy logic several operations between fuzzy sets, such as "and", "or", "min" and "max", are defined. Similarly to a fuzzy set, a fuzzy variable is represented by a convex and normal membership function. Linguistic variables, such as high, medium and low, can be translated into fuzzy variables. Linguistic variables are essentially aggregations or categories of crisp variables. They can be represented by fuzzy variables, through an adequate membership function, usually bell shaped. Another fuzzy logic component relevant to process control are fuzzy logic rules. Fuzzy logic rules are if/then rules, based on expert knowledge, used to manipulate fuzzy variables. Fuzzy logic rules, as any other logic conjunction, have two parts, antecedent and consequent.

There are two types of control schemes based on fuzzy logic: the fuzzy logic control and the fuzzy-model based control. In the fuzzy logic control, fuzzy logic is used directly as a controller. Fuzzy logic provides an algorithm that can convert the linguistic control strategy based on expert knowledge into an automatic control strategy. In the fuzzy-model based control, a model-based controller is constructed based on a fuzzy logic model of the process dynamics. A significant problem in the design of a fuzzy logic model is the determination of the proper membership function and the fuzzy logic rules. The fuzzy logic model can be structured with a fuzzy neural network scheme that implements the traditional fuzzy logic system with learning ability. As it was said before, the more promising application of fuzzy logic in chemical engineering seems to be a fuzzy logic model-based control scheme.

#### *3.3.4.1 Description*

The general structure of any fuzzy logic model is composed by a fuzzifier, a fuzzy inference engine with rule bases and a defuzzifier. A scheme is shown at the bottom of Figure 3.9. There are a variety of fuzzy neural network schemes, distinguished by the type of inference, membership function and defuzzification function. A fuzzifier performs the function of

fuzzification, which converts input data from an observed input space into proper linguistic values of fuzzy sets through predefined input functions. The fuzzy inference matches the output of the fuzzifier with the fuzzy logic rules and performs the appropriate reasoning. The decision-making operator simulates human decision-making based on a set of linguistic description rules, which in turn are based on expert knowledge and the

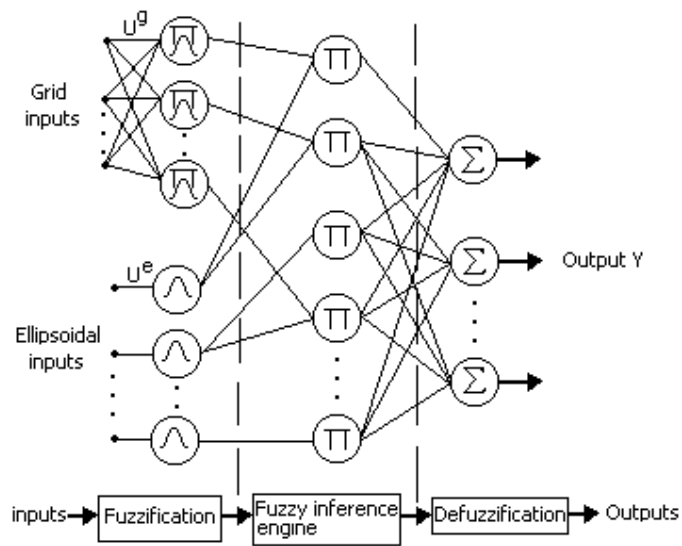


Figure 3.9 Fuzzy neural network architecture composed by a fuzzification layer, a fuzzy inference layer and a defuzzification layer

inference rule in fuzzy logic. Finally, the defuzzifier performs the function of defuzzification to yield a crisp output through predefined output membership functions. The output of the inference process on the decision-making operator is a fuzzy set, specifying a possible distribution of the control action. In online control a crisp signal is required. That signal is usually decided to be the center of gravity of the fuzzy resulting membership function.

The fuzzy network scheme presented here is a simple version of a fuzzy logic system in which the consequent parts of fuzzy rules are constants. However, it is distinguished by a unique partition of the input space, which is suitable for the kind of modeling in which only a partial qualitative knowledge of a modeled process is available (Aoyama et al. 1995a). The input space is partitioned by both fuzzy grids and hyper-ellipsoidal regions that represent the inference rules, as illustrated in Figure 3.10. The number of hyper-ellipsoidal regions is determined independently of the input space dimension. Fuzzy grids are used only to include a priori knowledge. In the grid, shaded strips express the condition part of rules, and the cross areas of strips express the rules. The combination of a fuzzy grid and ellipsoid for a

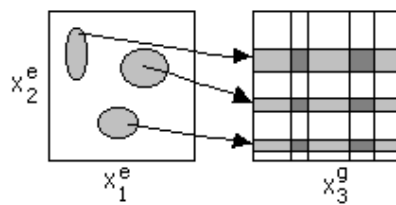


Figure 3.10 Combination of fuzzy grid and fuzzy ellipsoid

three variable ( $X_1$ ,  $X_2$ , and  $X_3$ ) input space is shown in this figure.

The discussed fuzzy neural network has three layers. As it was said before, the input dimensions are divided in two categories: grid and ellipsoidal. The network structure is shown in Figure 3.9. The function of the nodes of each layer is defined as follows:

1. First layer or fuzzification layer. There are two kinds of nodes. One kind takes the grid inputs and the other kind takes the ellipsoidal input regions. Each node in this layer represents the condition part of rules and the node output is equal to the degree of match for each condition. The node output of each grid node  $j$  is

$$U_j^g = \exp\left(\frac{(\mu - U^g)^2}{\sigma^2}\right) \quad (3.22)$$

where  $U_j^g$  is the node output,  $U^g$  is one grid input and  $\mu$  and  $\sigma$  are, respectively, the center and the width (or variance) of the bell shaped function of the node.

Ellipsoidal nodes take all ellipsoidal inputs as input. Thus, the output of ellipsoidal node  $j$  is

$$U_j^e = \prod_{i=1}^n \exp\left(\frac{(\mu_i - U_i^e)^2}{\sigma_i^2}\right) \quad (3.23)$$

where  $U_j^e$  is the node output,  $U_i^e$  is the  $i$ th input of the ellipsoidal dimension and  $\mu_i$  and  $\sigma_i$  are, respectively, the center and the width (or variance) of the bell shaped function of the  $i$ th input. The total number of ellipsoidal dimensional inputs is denoted by  $n$ .

2. Second layer of fuzzy inference engine. The fuzzy inference is carried out by arithmetic multiplication. Each node in this layer corresponds to a rule and the node output is equal to the firing strength of each rule. The number of nodes in this layer is equal to the number of rules. The inputs to all nodes can be from both dimensions: grid inputs and ellipsoidal inputs. Thus the output of the  $j$  node of this layer is defined as

$$U_j = \prod_{i=1}^n U_i^g \times U_i^e \quad (3.24)$$

where  $U_j$  is the node output,  $U_i^g$  is the  $i$ th grid dimensional input and  $U_i^e$  is the  $i$ th ellipsoidal dimensional input. The parameter  $n$  depends on the fuzzy rule. Commonly if/then rules have only two variables in the antecedent part, so  $n$  is equal to 1. Arithmetic multiplication can be replaced by "and" or "or" operations, but the training method employed here requires that  $U_j(U^g, U^e)$  be a derivable function. For example an if/then rule of the type "IF  $U^g$  and  $U^e$  THEN  $U_j$ " can be written mathematically, as the output of the  $j$ th node:

$$U_j = \min(U^g, U^e) \quad (3.25)$$

3. Layer 3 or defuzzification layer. The nodes in this layer transmit the decision signal  $Y$  out of the network. The number of nodes is equal to the output dimension. The center of gravity of the defuzzification method is simulated through

$$Y = \frac{\sum_{j=1}^l \mu_j U_j}{\sum_{j=1}^l U_j} \quad (3.26)$$

where  $\mu_j$  are the  $l$  centers of the membership functions.

A priori knowledge can be included as constraints imposed on the network parameters  $\mu$  and  $\sigma$ , in the fuzzy inference rules and in the way network connections are made. The following section describes how non-set parameters are found through a learning process.

#### 3.3.4.2 Learning Algorithm

The learning rules for the fuzzy neural network are based on the backpropagation-type gradient descent method, which calculates the error ( $E$ ) rates recursively from the output backwards to the input nodes. The goal is to minimize the squared difference of the network output  $Y$  and the desired output  $Y_t$ .

Assuming that  $w$  is the adjustable parameter ( $\mu$  or  $\sigma$ ), the update rule is:

$$w(t+1) = w(t) - \eta \frac{\partial E}{\partial w} \quad (3.27)$$

Where  $\eta$  is the learning rate. As in MLP the learning rate value depends on the problem characteristics. Any method used to adaptively update  $\eta$  in a MLP can be used also for this kind of network.

Therefore it is necessary to find the error derivative for each parameter of each layer.

- Third layer. In this layer the update rule for each parameter is equal to

$$\mu_j(t+1) = \mu_j(t) - \eta(Y - Y_t) \frac{U_j}{\sum_{j=1}^l U_j} \quad (3.28)$$

The error to be backpropagated from this layer to the preceding one is

$$\frac{\partial E}{\partial U_j} = (Y - Y_t) \frac{\mu_j \sum_{j=1}^l U_j - \sum_{j=1}^l \mu_j U_j}{(\sum_{j=1}^l U_j)^2} \quad (3.29)$$

- Second layer. There is no parameter to adjust, so only the error signals must be calculated. In the case of an arithmetic multiplication rule of one grid dimensional input and one ellipsoidal dimensional input, the backpropagated signal to a grid node of first layer is

$$\frac{\partial E}{\partial U_j^g} = \sum_{k=1}^m \left( \frac{\partial E}{\partial U_j} U^e \right)_k \quad (3.30)$$

where  $m$  is the number of connections of the grid node of the first layer to the second layer. The error derivative is stated above, in equation (3.29). Similarly, the backpropagated signal to an ellipsoidal node of first layer is

$$\frac{\partial E}{\partial U_j^e} = \sum_{k=1}^m \left( \frac{\partial E}{\partial U_j} U^g \right)_k \quad (3.31)$$

- First layer. As before, the error derivatives are obtained from the backpropagated signal from second layer, stated above in equations (3.30) and (3.31). The adaptive rule of the center of a grid node can be derived using

$$\frac{\partial E}{\partial \mu} = - \frac{\partial E}{\partial U_j^g} \frac{2(\mu - U^g)}{\sigma^2} \exp\left(-\frac{(\mu - U^g)^2}{\sigma^2}\right) \quad (3.32)$$

The adaptive rule for the width of a grid node is

$$\frac{\partial E}{\partial \sigma} = \frac{\partial E}{\partial U_j^g} \frac{2(\mu - U^g)^2}{\sigma^3} \exp\left(-\frac{(\mu - U^g)^2}{\sigma^2}\right) \quad (3.33)$$

The adaptive rule of the center of the ellipsoidal nodes can be derived using

$$\frac{\partial E}{\partial \mu_i} = - \frac{\partial E}{\partial U_j^e} \frac{2(\mu_i - U_i^e)}{\sigma_i^2} \prod_{i=1}^n \exp\left(-\frac{(\mu_i - U_i^e)^2}{\sigma_i^2}\right) \quad (3.34)$$

and for the width of the ellipsoidal node

$$\frac{\partial E}{\partial \sigma_i} = \frac{\partial E}{\partial U_j^e} \frac{2(\mu_i - U_i^e)^2}{\sigma_i^3} \prod_{i=1}^n \exp\left(-\frac{(\mu_i - U_i^e)^2}{\sigma_i^2}\right) \quad (3.35)$$

The above scheme was used by Aoyama et al. (1995b) in a control-affine approach for non-linear process control. In 1999 he presented several control schemes for fuzzy neural networks for a CST reactor and a pH neutralization process.

The main drawbacks of the fuzzy neural networks made it very difficult to use fuzzy neural network modeling for high dimensional cases. The problem with the dimensionality of these networks is mainly because the number of hidden nodes (fuzzy rules) increases exponentially with the input dimension. This problem arises from the fact that the input space is partitioned by a fuzzy grid of if/then rules. Usually the number of fuzzy rules is equal to the number of hidden nodes for most fuzzy neural network schemes. Therefore, the exponential growth in the number of rules leads to an exponential growth in the number of hidden neurons and tuning parameters. It slows down the training of fuzzy neural networks to the point that process modeling is unrealistic. This problem could be solved if the input space is partitioned with ellipsoidal regions instead of a fuzzy grid.



Another drawback of fuzzy neural networks is related to training data. In fuzzy networks the input space is divided into cells, the number of which is equal to the number of basis functions. To obtain an accurate model, each cell has to contain at least one training datum. Thus, in high dimensional processes, a lot of data are needed for proper training. The number of necessary training data increases with input dimension. If a limited number of data are available, then increasing the dimensionality of the input space rapidly leads to the point where the data become very sparse leading to poor performance. High dimensional process are also problematic for the fuzzy "and" operation. The multivariable fuzzy neural network basis functions are constructed by the tensor product of the fuzzification operators connected by this operation. This way of construction of a multivariable function is computationally expensive; the cost is roughly proportional to the dimension. These drawbacks are especially problematic for chemical process modeling because they often involve models of high dimensions. This is a motivation for preprocessing the data to reduce the dimensionality. One useful method is principal component analysis. Another possible remedy is to abandon the fuzzy grid altogether as proposed by Sugeno and Yasukawa (1993).

Fuzzy logic has been included into several neural networks algorithms. Another kind of fuzzy neural networks was studied by Feuring (1996a and b). The fuzzy neural networks he studied used fuzzy variables as inputs and outputs, and, also, the connection weights were fuzzy numbers. Feuring (1996a) describes several aspects of these networks, while Feuring (1996b) proposes several training algorithms. Also, stability analysis of controllers that used a special kind of fuzzy neural network was studied by the same author in 1999. He developed conditions on the training set in order to find a stable controller. Another more industrial oriented work was developed by Moreno et al. (2001). He compared neural and fuzzy logic techniques for a classification/decision engine included in an automatic coin recognizer. Another example of a neural network algorithm that employs in some degree fuzzy logic is the fuzzy ARTMAP system, explained in the next section.

### ***3.3.5 Fuzzy ARTMAP***

ARTMAP is a class of neural network architecture that performs incremental supervised learning of recognition categories and multidimensional maps in response to input vectors presented in arbitrary order. It was proposed by Carpenter in 1992. This system uses two adaptive resonance theory (ART) modules linked by an associative learning network and an internal controller that ensures autonomous system operation in real time. A fuzzy ARTMAP utilizes fuzzy operations instead of crisp operations to classify inputs by a fuzzy set of features. It could also classify analog patterns that are not necessarily interpreted as a fuzzy set. The main difference with other neural networks architectures is that it learns each input

as it is received on-line, rather than performing an off-line optimization of a performance criterion function. Another important fact is that a fuzzy ARTMAP neural network architecture does not require of the previous definition of number of neurons or connections between them. The main drawback of this function approximation is that, as any other neural network system (Nahas et al., 1992), it has bad extrapolation capacities.

Both fuzzy ART modules of the fuzzy ARTMAP system work in the same way. Each one of the ART modules has a weight or parameter vector  $\mathbf{w}_j$  associated with each category  $j$ , where  $j=1,2,\dots,N$  and  $N$  is the total number of categories. This weight vector has as many components as the input vector and initially all of them are set to 1. The ART module dynamics are determined by three parameters: a choice parameter  $\alpha > 0$ , a learning rate parameter  $\beta \in [0,1]$  and a vigilance parameter  $\rho \in [0,1]$ . For a given input vector  $\mathbf{I}$  the category choice is made based on the choice function  $T_j$  defined as

$$T_j(\mathbf{I}) = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} \quad (3.36)$$

The fuzzy AND operator  $\wedge$  is defined by

$$(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i) \quad (3.37)$$

and the norm  $|\cdot|$  is defined by

$$|\mathbf{p}| \equiv \sum_{i=1}^M |p_i| \quad (3.38)$$

for any  $M$ - dimensional vectors  $\mathbf{p}$  and  $\mathbf{q}$ .

The chosen category  $J$  is the smallest index  $j$  where  $T_j$  is maximal. It is said that resonance occurs if

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} \geq \rho \quad (3.39)$$

and then the weights are updated according to

$$\mathbf{w}_J^{new} = \beta(\mathbf{I} \wedge \mathbf{w}_J^{old}) + (1 - \beta)\mathbf{w}_J^{old} \quad (3.40)$$

If resonance does not occur a new category is chosen: the next smaller index  $j$ , where  $T_j$  is maximum, is selected.

The search for a category continues until a resonant category is found. If there are not resonating categories among all  $T_j$  maximum values then a new category node  $\mathbf{w}_j$  is created.

This new category node is located at the point described by the input vector  $\mathbf{I}$ :

$$\mathbf{w}_j = \mathbf{I} \quad (3.41)$$

To avoid proliferation of categories, the inputs should be normalized, i.e., for some  $\gamma > 0$

$$|\mathbf{I}| = \sum_{i=1}^M |I_i| \equiv \gamma \quad (3.42)$$

Complement coding is one way to ensure that the above equation holds, as it will be explained below. In complement coding a complement vector of the input vector is concatenated to it. The complement vector of  $\mathbf{a}$ ,  $\mathbf{a}^c$  represents the off-response. Each one of its components  $a_i^c$  is defined as:

$$a_i^c \equiv 1 - a_i \quad (3.43)$$

Thus the complement coded input  $\mathbf{I}$  is

$$\mathbf{I} = (\mathbf{a}, \mathbf{a}^c) = (a_1, \dots, a_M, a_1^c, \dots, a_M^c) \quad (3.44)$$

With this new input, it can be proved that

$$|\mathbf{I}| = M \quad (3.45)$$

Therefore, complement coding is a way to avoid proliferation of categories.

A few relevant things about ART dynamics are:

- If more than one category is a fuzzy subset choice, the small but positive parameter  $\alpha$  breaks the tie by choosing the category index that maximizes  $|\mathbf{w}_j|$  among the fuzzy subset choices.
- It is said that fast learning occurs when  $\beta=1$ . When this happens  $\mathbf{w}_j^{(new)} = \mathbf{I}$ .
- If  $0 \leq \rho < 1$  and complement coding, fast learning and constant vigilance are used, the number of categories is bounded, even if the number of exemplars in the training set is unbounded.

The proof of the above items and more information about ART dynamics can be found in the paper written by Carpenter et al. (1992). This paper also provides a geometric interpretation of a fuzzy ART module.

Fuzzy ARTMAP uses two fuzzy ART modules linked by an associative learning module and an internal controller that ensures autonomous operation of the network. One of the fuzzy ART modules is used to classify the input patterns into categories. This module will be used to classify the state space of the process into regions. It is called fuzzy ART<sub>A</sub> module. The second fuzzy ART module, called fuzzy ART<sub>B</sub>, is used to classify the desired network outputs into categories. In the present work, fuzzy ART<sub>B</sub> module will be used to classify the cost space of a dynamic optimization problem into regions. The network input-output map is done through an inter-ART module, called map field, which links fuzzy ART<sub>A</sub> with fuzzy ART<sub>B</sub>. There is a set of weight vectors  $\mathbf{w}_j^{ab}$  or parameters associated to each node category of fuzzy ART<sub>A</sub> module. Each weight vector links a node of fuzzy ART<sub>A</sub> output to the map field. Initially all components of the weight vector are set to 1. The map field is used to form predictive associations between categories of both fuzzy modules and to realize the match-tracking rule, whereby the vigilance parameter of fuzzy ART<sub>A</sub> increases in response to a predictive mismatch at fuzzy ART<sub>B</sub>. Match tracking reorganizes category structure so the predictive error is not repeated on subsequent presentations of the input. This is done in the following way. If

node J of the fuzzy ART<sub>A</sub> is chosen, then its weight vector  $\mathbf{w}_j^{ab}$  activates the map field. If node K is active in fuzzy ART<sub>B</sub> then the node K in the map field is activated in the 1-to-1 pathways between the fuzzy ART<sub>B</sub> module and the map field. If both fuzzy ART modules are active then the map field becomes active only if fuzzy ART<sub>A</sub> predicts the same category as fuzzy ART<sub>B</sub> via the weight vector  $\mathbf{w}_j^{ab}$ . The output of the map field  $\mathbf{x}^{ab}$  depends on which one of the fuzzy ART modules is active, as illustrate in Table 3.2.

**Table 3.2** Fuzzy ARTMAP map field output based on which one of the fuzzy ART modules is active

Fuzzy ART <sub>A</sub>	Fuzzy ART <sub>B</sub>	Map field output $\mathbf{x}^{ab}$
Active	Active	$\mathbf{w}_j^{ab} \wedge \text{Fuzzy ART}_B \text{ output}$
Active	Inactive	$\mathbf{w}_j^{ab}$
Inactive	Active	Fuzzy ART <sub>B</sub> output
Inactive	Inactive	0

The output  $\mathbf{x}^{ab}$  is used by the internal controller, when both fuzzy ART modules are active, to check the match between them in a similar manner than the one used for a single fuzzy ART module. At each input presentation the ART<sub>A</sub> vigilance parameter equals a base line vigilance value. The map field has also a vigilance parameter  $\rho_{ab}$ . It is said that resonance occurs in the map field if

$$\frac{|\mathbf{x}^{ab}|}{|\text{Fuzzy ART}_B \text{ output}|} = \frac{|\mathbf{w}_j^{ab} \wedge \text{Fuzzy ART}_B \text{ output}|}{|\text{Fuzzy ART}_B \text{ output}|} \geq \rho_{ab} \quad (3.46)$$

If resonance occurs in the map field then the connection between both fuzzy modules is learned

$$\mathbf{w}_j^{ab} = \mathbf{x}^{ab} \quad (3.47)$$

If resonance does not occur then the vigilance parameter of the fuzzy ART<sub>A</sub> module is increased until it is slightly larger than  $|\mathbf{A} \wedge \mathbf{w}_j^a| |\mathbf{A}|^{-1}$ , where  $\mathbf{A}$  is the input vector to fuzzy ART<sub>A</sub> in complement coding form and  $\mathbf{w}_j^a$  is the weight vector of fuzzy ART<sub>A</sub> of the winner category J. If the resonance parameter of fuzzy ART<sub>A</sub> is increased, fuzzy ART<sub>A</sub> search leads either to activation of another category node that satisfies both resonance criteria (fuzzy ART<sub>A</sub> and map field), or, if such node does not exist, to the shutdown of the fuzzy ART<sub>A</sub> module for the remainder of the input presentation.

Once all the input patterns are classified in the ART<sub>A</sub> module and all the network desired outputs are classified in the ART<sub>B</sub> module, and both modules are linked through the map field, the neural network can be used as a predictor as proposed by Giralt et al. (2000). For a given input vector the output category from module fuzzy ART<sub>A</sub> is calculated using the choice function. Next, through the map field, the corresponding fuzzy ART<sub>B</sub> category is found. Then,

the predicted output value is found in the weight vector of fuzzy ART<sub>B</sub> associated with that category.

Carpenter implemented in 1995 a fuzzy ARTMAP system in a probability estimation procedure, suitable for pattern recognition problems. In her work fuzzy ARTMAP was allowed to operate either as a classifier or as probability estimator. Fuzzy ARTMAP systems have been employed successfully in the prediction of physical properties of organic compounds (Espinosa et al. 2001, 2002) and in the recognition of turbulence structures (Giralt et al., 2000). Those systems will be used in the present work to develop a cost map to solve the optimal control problem for a fed-batch bioreactor.

### **3.4 Building a NN control model: multivariable fed-batch bioreactor case study**

The conventional approach to process control model development based on a fundamental knowledge of the system is very often inefficient and unfeasible from an industrial viewpoint. Structured models usually demand the specification of a significant number of parameters thus requiring extensive, time-consuming research to be performed. Moreover, these parameters have to be updated relatively often depending on the process characteristics. An alternative approach to model specification are neural networks algorithms.

After the selection of the NN architecture for the specific problem it is necessary to follow a certain methodology to build the final model. The current methodology includes four main steps: (i) Obtain raw data from the process variables, (ii) pre-process the raw data so that the relevant variables are considered, i.e., ensuring that no process information is loosed, and normalize process data to make it suitable for neural-processing, (iii) tune the network architecture, and (iv) tune the training algorithm and train the network. Finally, the designed NN model should be tested. In the current study the tests consider performance and suitability for control purposes.

A general scheme of the methodology proposed to build the current NN control models is presented in Figure 3.11. Each box represents a coherent phase to be completed entirely and the arrows establish the sequence in the methodology. Note that some alternatives in this scheme for MLP have been previously proposed to obtain a NN fermentation model (Tsaptinos and Leigh, 1993). In the following four subsections the main steps that conform the current methodology are presented and discussed.

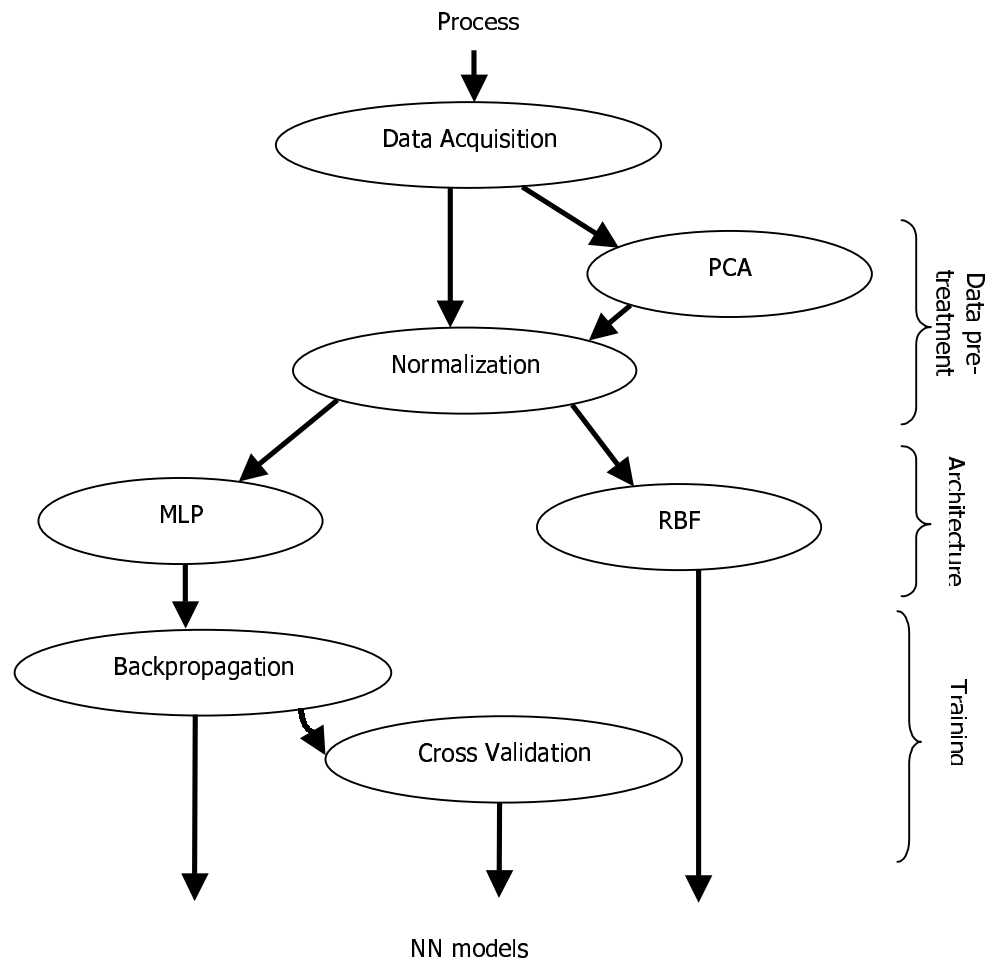


Figure 3.11 Scheme followed to obtain the NN control models

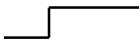
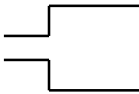
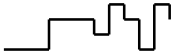
### 3.4.1 Data acquisition and control models

A sufficiently large and diverse set of inputs and desired outputs is needed to train a NN by a supervised training algorithm. The acquisition of this experimental data is the first challenge that should be confronted, since it is crucial to determine how to perturb the process operating conditions to generate a wide diversity of behaviours (therefore, patterns for training) for all relevant variables. There are a large number of previous studies related to linear model input sequence perturbation. There is also a MATLAB toolbox created by Ljung (1987) for linear system identification. However, there is not a systematic approach for non-linear models (Ljung, 1987). Ljung (1989) discusses the system identification problem in a noise free environment. Pearson et al. (1997) consider the selection of input sequences for non-linear model identification. Lin and Jang (1998) have a similar objective. They presented a systematic approach for data set design in order to train the neural network or any other black box model. Their approach is based on information theory. Information entropy was

derived to identify the mutual positions among data points in all feasible regions. Information enthalpy was derived to obtain a system's dynamic non-linearity. The placements of the new data were designed on the basis of a compromise between the information entropy and information enthalpy- the information free energy.

In the present study, a control model for the multivariable fed-batch bioreactor is developed. The substrate flow rate  $F$  has been chosen for this problem as a perturbation variable because of its direct influence on five of the six process output variables. This variable is perturbed by a positive step (increment of the flow rate), a positive and a negative (decrement of the flow rate) step, and finally by a random multiple sequence of positive and negative steps, around the five different steady states given in Table 3.3.

**Table 3.3** Disturbances applied to the substrate flow rate (input variable) in a multivariable fed-batch bioreactor for process simulation

Name	Disturbance	F value needed to reach the perturbed steady state ( $10^9 \text{ m}^3/\text{s}$ )
Positive Step		5, 6.6, 10, 11.6
Positive and Negative Step		5, 6.6, 8.3, 10, 11.6
Multiple Step		5, 6.6, 8.3, 10, 11.6

The patterns produced by multiple random steps were selected as raw material for the oncoming steps of the control model development, because of the wide range of training data obtained. The training data set consisted of 3445 training patterns that contained information about the dynamics of 5 different process steady states near the process operation state point. The time span of the training patterns was only 5 process time simulation units, because of the relative slowness of the fermentation process studied.

The different input-output variables selected for the direct model of the multivariable bioreactor are represented in Figure 3.12(a). The configuration proposed includes input information of nineteen variables: Six variables in  $\mathbf{V}$  at three time instants (historical data) and the inlet flow rate  $F(t)$ . The six outputs are  $\mathbf{V}(t+1)$ . Different alternatives exist for the inverse model.

Figure 3.12(b) shows the inverse MLP-based models proposed by Aoyama and Venkatasubramanian (1995c) and by Hussain and Kershenbaum (2000), which perform well for step changes between steady state operations. The original MLP inverse model of Aoyama

and Venkatasubramanian (1995c) had an input layer with four neurons and the output is the dilution rate  $F(t-1)/V_R$ , while that of Hussain and Kershenbaum (2000) used six inputs and the output was the thermal fluid flow rate  $u(t)$  into the jacket of the bioreactor. The dimension of the input variables in these two inverse models has to be increased to eighteen and nineteen, respectively, to adequately respond to the random step changes that occur in many practical situations where the model has an actuator role inside the controller. Figure 3.12(c) shows the modifications of these two literature schemes considered here and a current new structure with an input dimension of nineteen, which replaces the first component of  $\mathbf{V}(t+1)$  by its following steady state value, using known information about the process steady states. This new state vector is denoted  $\mathbf{V}_{ss}$ .

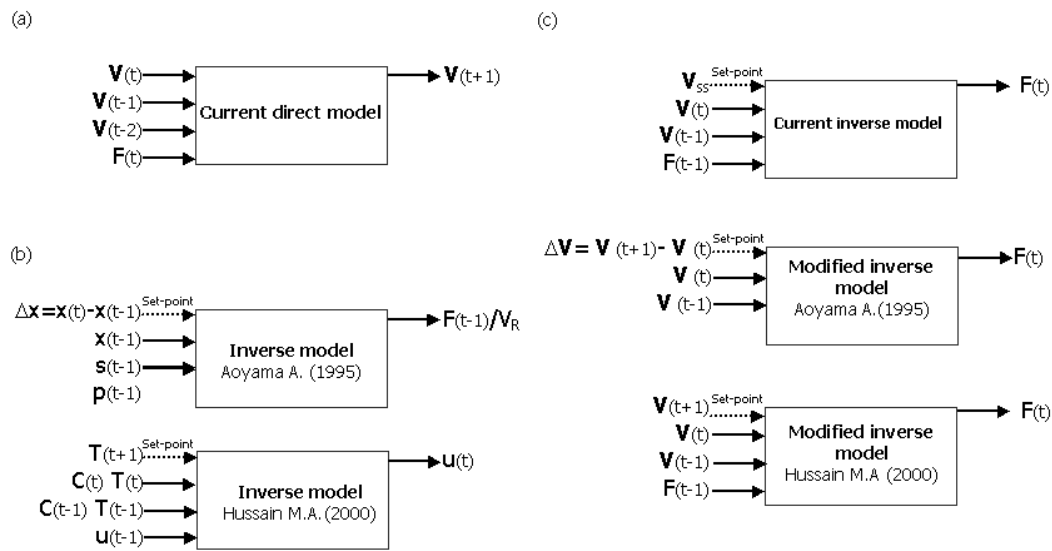


Figure 3.12 Input-Output scheme for the neural network models of the multivariate bioreactor: (a) Current direct process model with  $\mathbf{V}=\{x, s, p, T, \text{pH}, [\text{O}_2]\}$ ; (b) literature inverse process models; (c) modified literature inverse models and current proposal

### 3.4.2 Data pre-processing

Although neural networks have been used in the rectification of inaccurate and inconsistent plant data (Himmelblau and Karjala 1996), input data has to be adjusted to the requirements of the neural network. Data is generally normalized between their maximum and its minimum values. This is especially useful when the data is noisy because the input space of one variable can be shrunk to minimize its error contribution to the total Sum of Squared Errors (SSE) during the learning phase. To illustrate this effect, Figure 3.13 summarizes the performance of a MLP network for a fed-batch bioreactor process control model built with three different normalization limits. The training process was stopped when the SSE stabilized. Note that the SSE increases as the normalization limits are closer to the maximum



and the minimum values of the noisy pattern. In particular, the pH data input space was contracted and a lower total SSE was reached. The normalization limits should be carefully chosen because if the normalization limits are smaller (in the case of the minimum) and/or larger (in the case of the maximum) than the data limits of the test set, the predictive capabilities of the NN are substantially reduced. Neural network models tend to give good interpolation results between learned patterns but have poor and unreliable extrapolation capabilities (Nahas et al. 1992).

For the multivariable fed-batch bioreactor the global bounds of the process given in Table 3.4 are used as the normalization limits since they are the physical limits imposed by the microorganism on the process variables. This ensures that at any time, even for large process disturbances, the NN input noise will not be magnified by the normalization process because the NN input data will always be within the normalization limits.

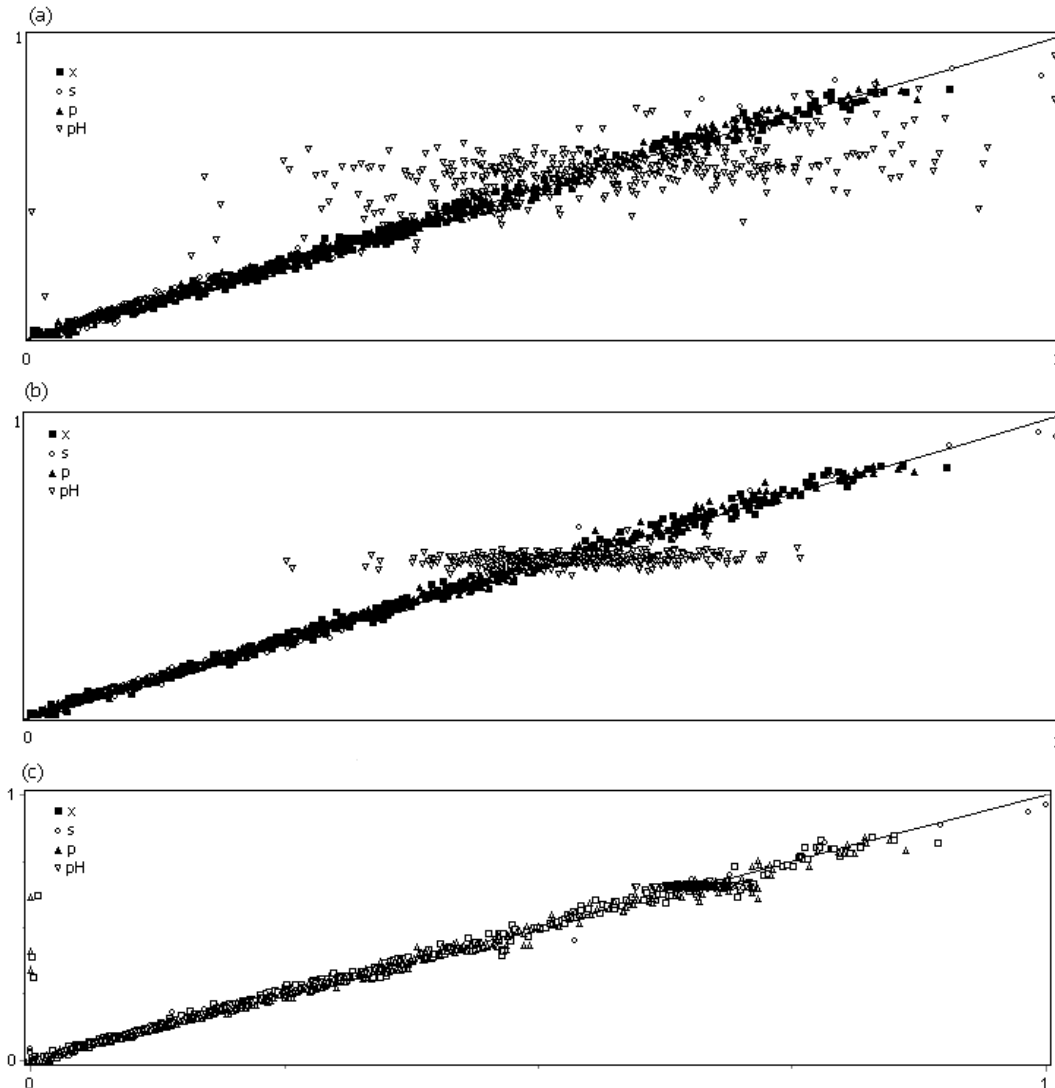


Figure 3.13 Influence of the normalization procedure when the 404 patterns of data from the multivariable bioreactor used to train the neural model contain noise. Comparison between predicted and expected outputs for three different normalization limits: (a) SSE=6.88; (b) SSE=2.50; (c) SSE=1.65.

**Table 3.4** Normalization limits for the NN input data in the multivariable bioreactor model

Pattern	Minimum						Maximum					
	$\times 10^2$	$s \cdot 10^4$	p	T	pH	$O_2$	$\times 10^1$	$s \cdot 10^4$	p	T	pH	$O_2$
Positive Step	1.51	0.24	0.08	303.14	0	0.03	3.70	7.66	0.54	303.15	14	0.298
Positive and Negative Step	1.39	0	0.13	303.14	9.16	0	3.65	6.52	0.246	303.15	9.20	0.298
Multiple Steps	5.15	0	0	273.15	0	0	4.72	6.5	1.75	323.15	14	0.298

Besides normalization, Principal Component Analysis (PCA) is another useful method for data treatment (Aoyama A., 1999). Finding the principal components of the matrix given by the NN inputs values provides a way for reducing the dimension of these inputs, which implies a reduction in training time and in network complexity. PCA is also used to reduce noise in the data. However, the principal drawback of this method for control purposes is that a principal components analyzer has to be installed inside the controller, just before the input to the NN model, increasing the controller complexity and decreasing the controller time response. In the present study this issue is minimal because fermentation is a slow process and does not require a fast controller response.

The PCA results for the raw data (with and without noise) and for the normalized data of the multivariable fed-batch bioreactor are plotted in Figure 3.14. Three principal components are needed to describe the 90% of the raw data information from six variables without noise. In the presence of noise, there is no dimensional reduction since no smaller descriptive subsystem than the original system of six variables system can be found. A greater variable reduction was achieved for the normalized dataset. From 19 variables, only 10 principal components are enough to describe 99% of the process behaviour captured by this training data.

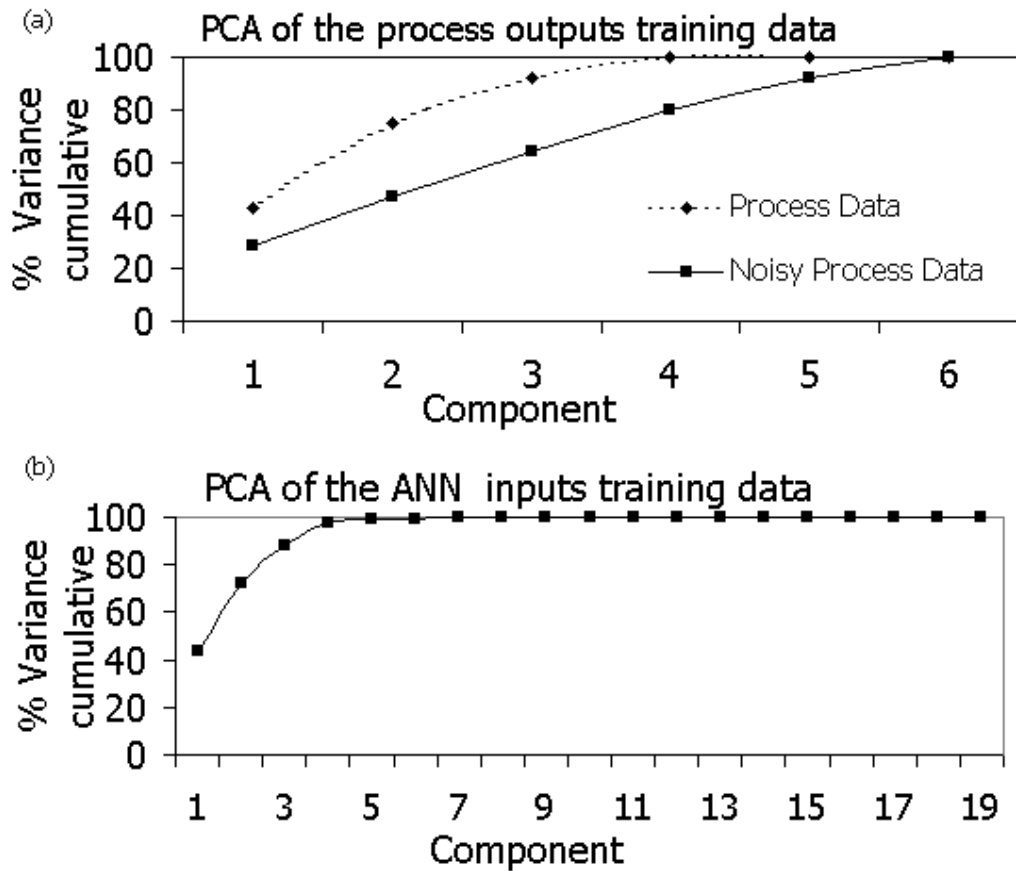


Figure 3.14 PCA of the multivariable bioreactor data (a) Process outputs; (b) neural network input training data

### 3.4.3 Tuning the network architecture

The number of hidden layers and neurons in each of these layers has to be specified prior to training the MLP network. Theoretically, two hidden layers are enough to represent any function, with arbitrary accuracy being obtainable (Cybenko, 1988). It has also been proved that only one hidden layer is enough to approximate any continuous function (Cybenko, 1989; Hornik, 1989). Although the utility of these results depends in how many hidden neurons are necessary, and this is not known in general. A MLP neural network can be seen as a linear combination of localized bumps that are each non-zero in the domain of a function to be represented (Hertz and Krogh, 1993). The bump approach may not be the best for any particular problem, but it is only intended as an existence proof. To represent any function in  $N$  dimensions,  $2N$ -hidden neurons are needed in the first hidden layer. Those two neurons per dimension, together output a peak anywhere that is desired, but also some secondary peaks and valleys. All but the highest peak can be suppressed by another neuron in the second hidden layer with an appropriate threshold. Thus one hidden neuron in the second hidden layer is needed for each bump. A linear output layer then sums the bumps to produce

the desired function, in a manner similar to Fourier analysis or Green's function representation.

There are also several approaches to construct or modify the architecture to suit a particular task, proceeding incrementally, which modify mostly the training algorithm (Hertz and Krogh, 1993). Two ways to reach as few units as possible can be found: start with too many and take some away or start with too few and add some more. The first procedure may cause each connection to decay to zero, so that connections disappear unless reinforced during the learning process. However, rather than starting with too large a network and perform some pruning it is more appealing to start with a small network and gradually grow one of the appropriate size. Through this way, cascade-correlation algorithm builds a hierarchy of hidden units. The tiling algorithm creates multilayer architectures, starting from the input layer and going to the output layer, each successive layer has fewer neurons than the previous one, so the output layer has only one neuron. However, this kind of approach seems unlikely to be practical for applications requiring large networks, where training requires massive CPU and memory allocation power.

In practice, different network configurations should be chosen and tested by a trial and error procedure. Some of the best ones are chosen in this work for comparison purposes. Note that the computational time required to extend the search in the space of possible architectures is prohibitive. For instance to train a 19-18-12-6 backpropagation architecture with 3445 training patterns, in a Sun Enterprise 450 computer with a UltraSPARC-II 400Mhz processor, requires a CPU time of 1'19" minutes for each epoch. Since 3000 epochs are needed to stabilize the SSE, a total of 2.7 days would be needed to train this single network.

RBF networks can be seen also through "the bump approach" as networks of units that themselves have a localized bump-like response, each becoming activated only for inputs in some small region. Thus, only one hidden layer of such units is needed to represent any reasonable function. In practice, for the RBF architecture, different RBF networks should be built by adding neurons one by one until several goal SSE values are reached. The criteria used to choose the best RBF network architecture is the SSE of the test, e.g., the architecture that yields the minimum average test SSE is selected.

#### ***3.4.4 Tuning the training algorithm***

The training algorithm is closely related to the NN architecture. Usually the NN architecture defines the major characteristics of its training method. A training method is the solution of an optimization problem. To train a MLP or RBF means to find the set of NN connection weights that give a desired NN behavior. As an optimization problem, it can be solved by different optimization methods, resulting on different ways of training. In a fuzzy ARTMAP means to find the appropriate center of each class, which is done automatically by means of the vigilance parameter as it was previously presented in section 3.3.5.

The backpropagation learning algorithm with a momentum term (Hertz J., 1993) is used to train the MLP models of the multivariable bioreactor. An online training is performed, i.e. the NN connection weights are actualized each time an input pattern is presented to the network. Some of the MLPs obtained have been trained using Cross Validation. This technique divides the training set used in the backpropagation algorithm in two parts: a validation set (composed by 689 patterns picked up from the process operation point in our case) and a training set (with the remaining process data of 2756 patterns). Each time all the training set had been presented to the NN the SSE of the validation set is calculated. The MLP is trained until the SSE of the validation set increases over five consecutive time steps. This allows choosing the best architecture given the performance over the validation set.

The drawback of the cross validation technique is that it fits very well the process dynamics surface around the operation point given by the validation set, but it does not model the surroundings very well. Thus, the model obtained with such a technique, while useful for steady processes degenerates when the operation point changes. Similar problems are observed when a linear control technique is used on a non-linear process.

The search of the optimal RBF architecture is implicit in the training steps. The control of this optimization process is fixed in the training process fixing the SSE goal to be achieved. However, a minimum SSE goal in the training set is not always optimal in the generalization stage because of overtraining (Bishop, 1995). Then a trial and error process is necessary to fit well the test data sets.



## Chapter 4 : Process Control

*Process control: the hidden technology that you use every day*

IFAC B02

*Controls will be the physics of the 21<sup>st</sup> century*

Larry Ho

(From J. Doyle workshop in Complexity)

### 4.1 Introduction

When in 1781 James Watt invented the centrifuge force regulator for its steam machine, feedback automatic control and the entire cybernetic field was born. Since that time, with the second industrial revolution, control systems are an integral part of modern society. Numerous applications ranging from simple home appliances to sophisticated aerospace systems use feedback control one way or another. Control systems are not limited to "man-made" technologies. Biological species cannot function and survive without feedback control. e.g. regulating temperature, hormones, heart rate, motor control, etc. Biological systems are the typical example of highly optimized tolerance (HOT) systems. HOT systems are robust to uncertainties that are common, or that the system was designed for, or has evolved to handle; yet fragile otherwise. Another HOT system is a turbulent fluid. Control of non-linear distributed process systems like fluid flows, size-distribution of particles and material microstructure is one major research challenge in process control (Chistofides, 2001). Based on some of the obtained results on complex systems, new control methodologies have slowly emerged and been implemented in industry. The most common control scheme in industry is the model predictive control, but model free adaptive control, based on neural networks has been of recent insurgence (Control Engineering Europe, 2001). It is believed that advance process control delivers lasting benefits when built on a solid -mathematical- foundation (Control Engineering, 2001a).

The objective of control systems is to regulate or track a system output by adjusting its input subject to physical limitations. A dynamic system is a causal system in which its outputs depend on its previous inputs and its initial states values. From this definition three types of variables can be differentiated: (a) Input variables or those influences that originate outside the system and are not affected by what happens in the system; (b) output variables or a subset or a functional combination of state variables, which one is interested to monitor or regulate. (c) states or the minimum set of system variables necessary to describe completely the dynamical system states at any given time. Kalman filter can be used to determine the states of a dynamical system. Dynamical systems can be autonomous (time invariant) or not

autonomous, linear or non-linear or a combination of both. In time invariant systems model parameters do not change as time passes. While in time variant dynamical systems model parameters are usually a function of time. Linear systems are dynamical systems that can be expressed as

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu} \\ \mathbf{y} &= \mathbf{Cx} + \mathbf{Du}\end{aligned}\tag{4.1}$$

where  $\mathbf{x}$  are the states of the system,  $\mathbf{u}$  are the system inputs,  $\mathbf{y}$  are the system outputs and  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{C}$  and  $\mathbf{D}$  are matrices that describe the dynamical behavior of the process. In linear autonomous systems  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{C}$  and  $\mathbf{D}$  are constant, while in linear time variant systems, those matrices are functions of time.

The complexity of a controller system increases if the process to be controlled presents non-linear time variant dynamics or if a multivariable system is to be controlled. Zafiriou (1987a) presented a digital controller design for multivariable systems with structural closed-loop performance specifications. Quantitative criteria are provided for comparing different designs and evaluating trade-offs. The robust control of a multivariable experimental four-tank system was explored by Vadigepalli et al. (2001). Traditional control methods like inverse model control,  $H_\infty$  (a robust control method for controller synthesis), and Proportional Integral (PI); are compared.

#### **4.2 Objectives of process control**

The objective of any control system is to influence the behavior of a dynamic system. The control problem is to determine the control inputs to the process using all available data. The quality of a controller should be judged by the following criteria (Garcia and Morari, 1982):

1. Regulatory behavior. The output variable is to be kept at its set point despite unmeasured disturbances affecting the process.
2. Servo behavior. Changes in the set point should be tracked fast and smoothly.
4. Robustness. Stability and acceptable control performance should be maintained in the face of structural and parametric changes in the underlying process model. A controller should be designable with a minimum of information about the process. Closed-loop stability of the entire system in the absence of plant variations is desirable, but this almost never is an issue because the majority of chemical processes are open-loop stable. Sometimes a trade-off situation is imagined between stability and control quality, but with the proper dynamic compensator any gain is possible for any system.
5. Ability to deal with constraint on the inputs and states. Almost always the major economic return from process control arises from the optimization of the operating conditions. Optimal operating points often lie at the intersection of constraints.



Therefore the ability of the regulatory controller to deal with constraints on both the inputs and the states is very important.

6. Controller complexity is an important issue. The control structure and the effects of the tuning parameters should be transparent to the operator.

The response of the controller to specific disturbances is of secondary importance. A filter inserted in the completed control system can correct any undesirable features to the maximum possible extent. The set point tracking vs. disturbance rejection problem for stable and unstable processes have been discussed by Zafiriou E (1987b).

### **4.3 Non-linear vs. linear process control**

All physical systems are non-linear. Often the linear models employed to design control systems design are only very poor approximation of the real behavior. While it is generally feasible to deal with mild non-linearities just by using detuned linear controllers, in the presence of strong non-linearities non-linear controllers can offer distinct advantages. Until recently, chemical processes have traditionally been controlled by using linear systems analysis and design tools based on linear models. A major reason for the widespread use of linear control systems is the availability of analytical solutions and rigorous stability and performance proofs. If a linear control system is used, linear input-output models are sufficient for the control (Hussain et al., 1995a and b). However, this leads to poor control if the chemical process is highly non-linear.

To improve the control of linear processes through non-linear controllers, a partition of the state space is usually employed and linear controllers are designed for each region of the partition. Recent progress in non-linear control theory and advances in computer technology now allow non-linear control strategy based on non-linear models to be successfully implemented for chemical processes. The design of open-loop non-linear controllers is a well-established practice. When variational methods have been used, virtually every conceivable problem has been tackled. On the other hand, internal model structure is particularly well suited for the design of feedback controllers. Most of the non-linear feedback control literature concentrates in stability analysis. In the next section some feedback control schemes suitable for non-linear models will be presented.

### **4.4 Some structures for model based process control**

Many high-level control schemes successfully applied as process controls in chemical plants, are model based. A process model is required to build the controller and find an adequate control action at each time step. There are two kinds of modeling schemes in chemical process systems engineering. A traditional approach to modeling is to develop a model from first principles and estimate the values of model parameters from process data. However, this

procedure is often difficult and costly because the dynamics of chemical process system may not be well understood or is too complex to model. An alternative approach is to identify the model nonparametrically from input-output data. Neural networks are especially suited for that purpose. In the following sections, some of the most common model based controllers will be explained in detail. The involved models can be of any kind, including first principles and neural network models, but neural networks were successfully applied in the those control schemes to solve many process control problems.

#### 4.4.1 Inverse model control

Direct inverse control utilizes an inverse system model. The inverse model is simply cascaded with the controlled system in order that the composed system results in an identity mapping between derived response and the plant outputs.

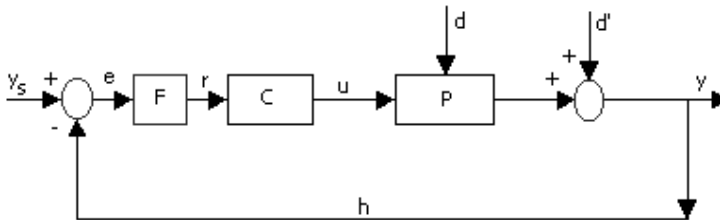


Figure 4.1 Inverse model control scheme

In Figure 4.1 a scheme of an inverse feedback controller is shown. Direct inverse control is common in robotics applications. Clearly this approach relies heavily on the fidelity of the inverse model used as controller. For general purposes, serious questions arise regarding the robustness of direct inverse control. This lack of robustness can be attributed primarily to the absence of feedback. This problem can be overcome to some extent by using an inverse model into a feedback control loop. In this case, the difference between the reference signal and the plant output constitutes the inverse controller input. The inverse controller can be obtained from an inverse model developed with neural networks, as discussed later in section 4.5.1.2. The complete inverse model control scheme has a filter  $F$  before the inverse model of the plant. The primary reason for including this filter is to introduce robustness in the overall control system structure in the face of modeling errors, by appropriately reducing the loop gain. Usually a simple exponential filter gives satisfactory results.

#### 4.4.2 Internal model control

In this scheme of control the controller is implemented simply as the inverse of the process model but a direct process model is also implemented within the feedback loop. Internal model control has been thoroughly examined and shown to yield transparency to robustness

and stability analysis. An unifying review of the internal model control (IMC) scheme was presented by Garcia and Morari (1982). Moreover, IMC extends readily to non-linear systems control (Economou et al., 1986). In IMC, a system model is placed in parallel with the real system and model output is used for feedback purposes. This feedback signal is then processed by a control block as shown in Figure 4.2. In the figure, P is the plant, the system to be controlled. M is a direct model of the plant, its inputs are plant inputs and its outputs are plant outputs, and C is the inverse of M. The properties of IMC dictate that this part of the controller should be related to the system inverse. These properties are:

1. **Stability.** The closed-loop system is input-output stable if the direct model used mimics exactly the plant response  $M=P$ . It should be noted that the implementation of IMC is limited to open-loop stable systems and systems which do not exhibit multiple output steady states.
2. **Perfect control.** If the closed-loop system is input-output stable, then the control will be perfect (perfect tracking) if the inverse model used in the control scheme mimics exactly the dynamics of the inverse of the direct model  $C=M^{-1}$ .
3. **Zero offset.** If stability and perfect control are obtainable in steady state operation, then offset free control is attained for asymptotically constant inputs.

IMC transforms a control problem of a non-linear system in a feedforward control problem, which can be solved even for non-linear systems. But, on the other hand, IMC preserves all the important characteristics of feedback control, in particular the suppression of unmeasured plant disturbances. Also, the complete IMC structure has a filter F before the inverse model of the plant. Again, the primary reason for including this filter is to introduce robustness in the IMC structure in the face of modeling errors, by appropriately reducing the loop gain. Zafiriou (1987c) used the structured singular value approach to quantify the concept of robust performance and to design the IMC filter. In the same year Marino-Galarraga et al. (1987) used a relative disturbance gain to evaluate the dynamic performance of multiloop control systems, among them IMC. These authors propose a IMC design which makes use of the interaction effects between the variables for a given disturbance.

The theoretical aspects of IMC are summarized by Zafiriou (1986). IMC was used by the same author to synthesise multivariable discrete controllers. An extension to open-loop unstable plants is added in his 1990's paper (Zafiriou and Morari, 1990). Gawthrop et al. (1995), compared and linked IMC with the

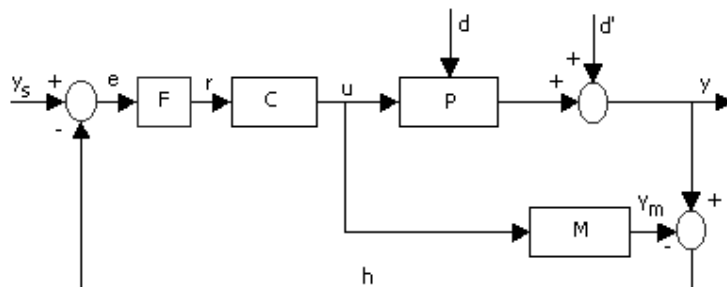


Figure 4.2 Complete scheme of non-linear internal model control structure

self tuning Emulator-based Control. Hu and Rangaiah (1999a) proposed an adaptive IMC for non-linear coupled multivariable processes. They stated that adaptive control of non-linear systems not decoupleable by static-state feedback is not easy or obvious. They developed a non-linear IMC controller for multivariable processes based on input-output linearization. These authors (Hu and Rangaiah 1999b) used this methodology for the control of pH.

Internal model control has been implemented also using neural network models. Nahas et al. (1992) proposed a non-linear IMC controller for singular input singular output (SISO) systems. Their approach was restricted to processes with stable inverses. They used neural networks to identify the process from input-output data. The control action was based on the inversion of the neural network model. The conjugate gradient algorithm was used to train the neural network and a first order filter was used into the IMC scheme. He successfully tested his control scheme on a CSTR to control the effluent concentration and pH. One year later, Hunt and Sbarbaro (1993) showed that adaptive inverse control is a further member of the class of controllers with IMC structure. They described the relation between both control methods, and using neural networks as non-linear adaptive filters, they implemented a non-linear analogue of inverse control. They used neural networks to realise both the direct process model and the inverse model of the process for the IMC control scheme. Aoyama et al. (1995c) used an IMC framework to control a bioreactor process. They proposed a hybrid model of the process to find an inverse model, instead of training a neural network with inverse process data. The hybrid model used the MLP neural network for both the process model and the inverse process model, so the controller exactly inverts the steady-state gain of the process model and offset is eliminated when the IMC structure is used to control. Hybrid modeling of a process through neural networks is discussed later in section 4.5.1.3. Aoyama et al. demonstrated that his approach is superior to the PI controller, to an IMC based on neural networks models and comparable to an exact IMC based on first principles. A year later Hussain et al. (1996) performed a discrete time analysis of IMC strategies based on neural networks. Guidelines for Lyapunov stability analysis for neural network based controllers were provided in his work. Brown et al. (1997) proposed an IMC structure using local model networks. These networks represent a non-linear dynamical system by a set of locally valid submodels across the operating range. Local model networks are similar to RBF but the basis functions are commonly chosen to be normalised gaussian functions. The main property of such local model networks is that the inverse model of the process can be derived analytically, thus the IMC offset is eliminated. They used a pH neutralization process to test this control approach. They concluded that significantly improved performance in terms of setpoint tracking and disturbance rejection compared with linear model was observed. A practical application of IMC was developed by Hussain et al. (2000). They implemented an IMC using MLP neural networks for a partially simulated exothermic reactor. They showed the

capability of neural-network based controllers and pointed out the differences between simulation studies and on-line experimental tests.

#### 4.4.3 Model predictive control

Model predictive control algorithms have been recognized as effective tools for handling some of the difficult control problems in the chemical industry. It appeared in industry almost 20 years ago as an effective way to deal with multivariable constrained control problems. The model predictive control scheme derives some of its industrial appeal from its ability to handle input-output constraints and time delay non-

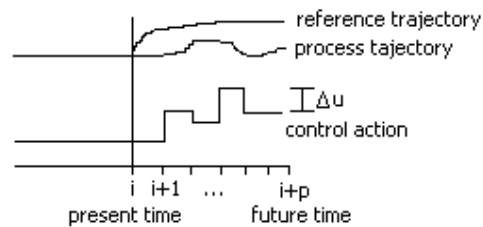


Figure 4.3 Scheme for model based predictive control. The  $i$ th control action is based on the prediction of future process outputs over a prediction horizon  $p$ . An optimal control action trajectory is found and only the first control move is implemented.

minimum behavior. Model predictive control (MPC) is an open loop control design procedure based on obtaining plant measurements and predicting future outputs by means of a model of the process. This is done at each sampling time. The predictions are used to compute  $m$  control moves by minimizing an objective function defined over a prediction horizon. The objective function is based on a sum of the squares of the differences between model predicted outputs,  $y$ , and a desired output variable,  $r$ , trajectory over a prediction horizon  $p$

$$\phi = \sum_{i=1}^p \Gamma_i (r_i - y_i)^2 + \sum_{i=0}^{c-1} \Lambda_i (\Delta u_i)^2 \quad (4.2)$$

The traditional MPC optimization stated above penalizes deviation of future model predictions  $y_i$  from set-points  $r_i$  while minimizing future control moves  $\Delta u_i$ . Variables  $p$  and  $c$  represent the prediction and control horizons, respectively, and  $\Gamma_i$  and  $\Lambda_i$  denote the error penalty and move suppression factors at  $i$ th instant. Then the MPC control law is stated as the first component of

$$\mathbf{u} = \arg(\min_u \phi) \quad (4.3)$$

where the minimization is done over the set of all possible control actions. Although more than one optimal control input is computed, only the first computed control move is implemented. At the next sampling time, new measurements are obtained from the plant and the optimization problem is solved again. Both the control horizon and the prediction horizon move or recede ahead by one step as time moves ahead one-step. This is the reason why MPC is also sometimes referred to as Receding Horizon Control or Moving Horizon Control. The purpose of taking new measurements at each time step is to compensate for

unmeasured disturbances and for model inaccuracy, both of which cause the system output to be different from the one predicted by the model.

The use of model predictive control in the process industries was pioneered in petroleum refining. Up to 2001, more than 1000 applications covering all major processes were published in this industry. In late 1980's MPC was incorporated in the controller system of chemical plants. Today MPC is applied for the control of complex processes, like cryogenic air separation, condensation polymers, terephthalic acids manufacture and so on. Since 1979, the refining plants in the USA are running at full capacity, as a consequence of environmental restrictions placed severe cost restraints to build new refineries. As a result, MPC benefits are generally determined based on increasing production capacity of higher value materials. However energy savings can be much more critical in determining the total benefit and thus the feasibility of applying MPC to chemical plants (Segura and Meziou, 2001). MPC has been used also to control a steam turbine of a thermal plant (Pedret et al., 2000) and to regulate the cell biomass exit concentration of a continuous-flow bioreactor (Parker and Doyle III 2001). Morari and Lee (1999) presented a unifying review of MPC over last 15 years. Multivariable system identification, performance monitoring and diagnostics, non-linear state estimation and batch system control are the research areas concerning the improvement of MPC performance. Morari and Lee (1999) stated that what limits the performance and applicability of MPC are not the deficiencies of the control algorithm, but difficulties in modeling, sensing, state estimation, fault diagnosis/detection, etc.

The advent of high-speed computers has made the application of non-linear model-based predictive control for online control a reality. As a result, a significant number of new control algorithms have been proposed based on non-linear programming techniques. There is a MATLAB toolbox created by Morari and Ricker useful to solve model predictive control optimization/simulation problems. The potential disadvantage of MPC is that it involves online optimization. It is especially problematic for the non-linear model predictive control because there are no guarantees of global optima. Some of these aspects are discussed in detail in Morari's paper. Another drawback of this control method is MPC's inability to take into account the evolution of uncertainty in the optimal control calculation. To address these issues Lee and Lee (2001) proposed a neuro dynamic programming method for MPC, in which the cost-to-go function of dynamic programming can be used to reduce a long horizon problem to an equivalent short horizon problem, thereby lowering the on-line computational load. This method will be explained in detail in section 4.6.3. Another computationally efficient approach for non-linear MPC is presented by Bhartiya and Whiteley (2001). He uses a RBF model to made non-linear predictions across a m-step horizon without using future unknown process measurements. The resulting MPC formulation using the RBF model provides analytical expressions for the gradient and the Hessian of the controller's objective function in terms of RBF network parameters. Loquasto III and Seborg (2001) also used

neural networks into a MPC system to detect significant degradation in performance and cope with changes in process behavior.

#### **4.5 Neural networks in process control**

The objective of control is to influence the behavior of dynamical systems. A large part of control theory deals with linear systems and powerful methods for designing controllers for them. In fact, most of the controllers used in modern industry belong to this class. However, as applications become more complex, the difficulties encountered in designing controls arise. Those difficulties, such as poor modeling, multiple subsystems, high noise levels and complex information patterns, can be broadly classified under three headings: complexity, non-linearity and uncertainty. Neural networks are capable of coping with all three categories (Narendra, 1996). Numerous dynamical systems have been identified and controlled through neural networks (Narendra and Parthasarathy, 1990; Narendra and Mukhopadhyay, 1992; Narendra 1995; Hussain et al., 1995b; Jagannathan and Lewis, 1996a, Mukhopadhyay and Narendra, 1999; Magni and Kershenbaum, 2000). For example, MLP neural networks have been used for moisture content control in fluidized bed granulation (Watano et al., 1997), for the long-term predictive control in thermal power plants (Prasad et al., 1998) and for the control of pH in a neutralization process (Yeo and Kwon, 1999).

From a systems theoretical point of view, a neural network can be considered as a conveniently parameterized class of non-linear map. Since they can approximate non-linear maps to any desired degree of accuracy, they can also be used to identify and control non-linear dynamical systems (Mukhopadhyay and Narendra, 1992; Lightbody and Irwin, 1997). They can approximate arbitrary functions from one finite dimensional space to another with any desired degree of accuracy (if given enough training data). Polynomials, trigonometric series, splines and orthogonal functions share the same properties, but neural networks, in view of their architecture, are more fault tolerant and less sensitive to noise and they are more easily implementable in hardware because of the parameterization used. Besides, as the dimensionality of the input space increases, multilayer perceptrons are preferable to approximation schemes in which the adjustable parameters arise linearly. Both, multilayer perceptron and RBF neural networks require substantially fewer parameters for a desired degree of accuracy (Narendra, 1996). If a task can be done equally well using conventional control methods or neural networks, then there are several advantages to using the latter. Intel, among others, has produced a neural net chip, which has more effective throughput than all of the Crays of the world put together (Werbos, 1991). Implicit in the use of neural networks to approximate a non-linear plant is the thought that an input-output model can exactly represent the plant. Underneath the apparent complexity of neural networks based controllers (required by specific applications) there are really only five generic designs now used to build neural networks to directly control actuators or effectors of some kind.

### 1. Supervised control

In supervised control a neural network learns the mapping from sensor inputs to desired actions, by adapting a training set of examples of what it should have done. Thus one can "clone" a human expert. Commonly, fuzzy rules are used to accomplish this task. This method can also be used to copy a slow but accurate computer control. The main challenge of this way of control is to build up an accurate and sufficient database of sensor inputs and desired actions.

### 2. Direct inverse control

In direct inverse control, a neural net learns the inverse dynamics of the system, so it can make the system to follow a desired trajectory. This controller structure is used in the present work to solve the bioreactor control problem. This approach cannot be used in applications where the mapping input-output of the plant is not invertible.

### 3. Neural adaptive control

In neural adaptive control, linear mappings used in standard adaptive control designs are replaced by neural networks resulting in greater robustness and ability to handle non-linearity. In general, adaptive control deals with the problem of controlling the output of a plant in the presence of parametric or structural uncertainty. In conventional adaptive control theory, to make the problem analytically tractable, the plant is assumed to be linear with unknown parameters. A suitable controller structure is chosen and the parameters of the controller are adjusted using an adaptive law, so that the output of the plant follows the output of the reference model asymptotically. Two design approaches have been proposed for the adaptive controller. The indirect approach estimates the parameters of the plant to be controlled and the control parameters are directly adjusted based on these estimates. In the direct approach, the control parameters are directly adjusted based on the observed output error. In neural adaptive control, the indirect approach is preferable (Narendra, 1992) but an input-output structure of the plant has to be assumed. There are two approaches also for parameter adaptation: gradient methods and stability methods. Gradient methods are effective from the viewpoint of performance; most of the neural adaptive controllers are built using gradient methods, but the stability of such algorithms cannot be demonstrated theoretically. In the other hand, stability algorithms are designed to assure the overall stability of the system from the outset. At the present time, such stable adaptive laws can be generated only for a restrictive class of systems. RBF networks, in which the outputs are linearly dependent on their parameters, are ideally suited for generating stable identification algorithms and stable controllers if the difference equations that represent the plant behavior are linear in the



control inputs. It could be said that internal model control implemented with neural networks follows the adaptive control design approach.

#### 4. Backpropagation of utility

This method adapts an optimal controller essentially by solving a calculus of variations problem. It maximizes profit. As a calculus of variations, this method requires a model of the system to be controlled, which may be itself a neural network. Backpropagation of utility (or cost) involves information flow backwards in time. It can be used to adapt parameters or weights of a controller or action network or to adapt a schedule of control actions over future time. Usually backpropagation through time algorithm is employed. It maximizes some measure of utility or performance over time, but cannot efficiently account for noise and cannot provide real time learning for very large problems. Model predictive control with neural networks follows this approach.

#### 5. Adaptive critic methods

These methods approximate optimal control over time in noisy, non-linear environments. The underlying idea is to approximate the Bellman equation of dynamic programming. They are the only design approach that shows serious promise of duplicating critical aspects of human intelligence: the ability to cope with large numbers of variables in parallel, in real-time, in a noisy non-linear environment (Werbos, 1992). Dynamic programming is the only exact and efficient method for finding an optimal strategy of action over time in a noisy, non-linear environment, but the cost of running true dynamic programming is proportional to the number of possible states in the plant or environment and that number in turn grows exponentially with the numbers of variables in the environment. Therefore approximate methods are needed even with many small-scale problems. One of these methods is used in the present work to solve the optimal control problem of the invertase production in a fed-batch bioreactor. Thus adaptive critic or neuro dynamic programming will be explained in detail in the section 4.6.2. The key theorem states that the strategy of action that maximizes utility in the short term will also maximize the sum over all future times. Adaptive critic designs can be defined more precisely as designs that include a critic network, i.e., a network whose output is an approximation of the utility or cost function, or to its derivatives. The inputs to the network are the values of the state variables (the complete plant description at time  $t$ ) only, or the states and the vector of actions. In the latter case, is called action dependent method.

Hunt (1992) presented a survey of neural network theory for control. Different controller schemes are summarized in his paper. In the following sections a more recent review of neural networks applications in process control will be presented. A review of neural networks

applications in chemical process control is presented by Hussain (1999). His review shows the MPL neural network as the most popular network for such process control applications and also shows the lack of current successful online applications.

#### ***4.5.1 Neural networks in identification***

Traditional methods for system identification of complex systems from finite data have been not completely effective addressing the question of identification in the context of uncertainty in the model class/parameterization (Venkatesh and Dahleh, 2001). This issue can be addressed through neural networks. As it was said before in the introduction chapter, biotechnology has used neural networks to estimate difficult measure process variables. Besides, companies like Owens Corning Glass and General Mill had implemented soft sensors in their control systems. Those soft sensors have been developed applying neural networks techniques in the inferential calculation of process variables. For example General Mills applying soft-sensor software from Aspen Technology, reduced run cycles, lowered energy usage, reduced product waste and improved product consistency and quality (Control Engineering 2001b). Another example from industry is the use of a neural network into an Early Warning System to predict the reliability of the products. In this system, the MLP neural network interprets the results of failure rates. It was implemented in United Technologies Carrier. An increase in productivity was obtained by at least eight times in terms of process time (Moon et al., 1998). Moreover, recurrent neural networks have been used for phase and quadrature detection in an electrical system (Kamwa et al., 1996). On the other hand, from the theoretical point of view, identification of dynamical systems using neural networks has been also explored by Jagannathan and Lewis (1996b). They presented a rigorous proof of identification error convergence. They showed how to train a neural network to obtain bounded error in the identification of four different non-linear dynamical models. Besides, fuzzy logic has been included in neural network models to make them easier to interpret. Zhang and Morris (1999) presented a recurrent neuro-fuzzy network to build long-term prediction models for non-linear processes. In their work, fuzzy logic was used to define process-operating regions. Also, neural nets have been used in conjunction with partial least square method for non-linear dynamical modeling (Qin and McAvoy, 1996). Partial least squares is used as a dimension reduction tool to remove colinearity and the MLP neural networks are trained to capture the non-linearity in the projected latent space. Another method for non-linear model reduction, inspired by the concept of subspace identification was proposed by Lee et al. (1999). Also, Li and Wang (2001) through a principal component analysis and clustered fuzzy diagraphs identified the process temporal behavior.

Identification of a dynamical process can be done through neural networks in two ways: a direct or forward modeling and an inverse modeling. Both identification schemes are explained in the next two sections.

#### 4.5.1.1 Direct process identification

In direct modeling the neural network is placed in parallel with the system and the error between the system and network outputs (the prediction error) is used as the network-training signal. See Figure 4.4. This learning structure is the classical supervised learning problem where the teacher (the system) provides target values (system outputs) directly in the output coordinate system of the learner (the NN). The dynamic nature of the

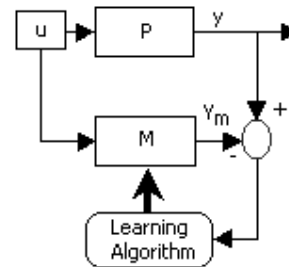


Figure 4.4 Direct process identification scheme. M learns the direct model of the plant P

systems under study can be introduced into the network itself. This can be done either using recurrent networks or by introducing dynamic behavior into the neurons. Those approaches increment the complexity of the modeling problem. A straightforward approach is to augment the network input with signals corresponding to past inputs and outputs. This approach is used in the present work to find a control model for the multivariable fed-batch bioreactor.

#### 4.5.1.2 Indirect process identification

Inverse models of dynamical systems play a crucial role in a range of control structures. However, obtaining these models raises several important issues. For inverse model development a synthetic signal is introduced to the system. See Figure 4.5. The system output is then used as

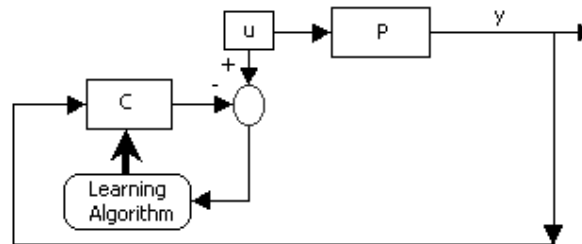


Figure 4.5 Indirect process identification scheme. C learns the inverse model of the plant P

input to the network. The network output is compared with the synthetic signal  $u$  (the system input) and this error is used to train the network. This method attempts to produce the inverse of the plant over the entire state space, but it can be very difficult to use it alone to provide adequate performance. A second approach to inverse modeling, which aims to overcome these problems, is known as specialized inverse learning (Psaltis et al., 1988). This learning scheme trains the neural network controller to operate properly in the regions of specialization only. The neural network may be trained online. In this approach, the network inverse model precedes the system and receives as input a reference signal, which spans the

desired operational output space of the controlled system (command signal of the system). In this case the error signal for the training algorithm is the difference between the training signal and the system output. The tracking error is propagated backwards to the plant using the partial derivatives of the plant at its operating point. Then, the network weights can be updated. The specialized learning approach is computationally complex, and the system derivatives should be easily known. In the present work, the first approach is used to build an inverse control model for the multivariable fed-batch bioreactor. A theoretical framework to inverse modeling of non-linear dynamical systems can be found in Cabrera and Narendra (1999). They state conditions for the existence of non-linear inverse controllers for the regulation and tracking of such systems.

#### 4.5.1.3 Hybrid neural network modeling

One major disadvantage of the previous input-output neural network modeling scheme is that the resulting models are complete black-box models. In chemical process modeling, an accurate first-principles model is rarely available, however, one often has some knowledge of the process behavior, which can be expressed as a mathematical model, or qualitative knowledge. Although this model is not complete, it does explain some behavior. However, in an approach that uses neural networks exclusively for modeling such a priori knowledge is not included. If the known knowledge is combined with neural network for parameter identification, such as the input-output mapping of those aspects of the process that cannot be modeled mathematically are modeled using the neural networks, then the resulting models become gray-box models. One way to integrate neural networks with a first-principle model is to put the NN parallel to a first principle model and train the NN to model discrepancies between the real process dynamics and the first-principle model. Sometimes this way of modeling results in no directly measurable neural network inputs or outputs, so the training complexity increases. Another approach that integrates a mathematical model with neural networks is the control-affine neural network model (Aoyama et al. 1996). Many chemical processes can be described by this procedure with a non-linear model in which the manipulated input  $u$  appears linearly in the output dynamic equations. A control-affine NN model scheme uses two neural networks. With this modeling scheme the process is approximated by

$$\Delta y(k) = f_{nn}(\mathbf{y}, \mathbf{u}, \mathbf{x})(u(k) - g_{nn}(\mathbf{y}, \mathbf{u}, \mathbf{x})) \quad (4.4)$$

where  $\Delta y(k)$  is an increment of the controlled output at time  $k$ ,  $u(k)$  is a scalar manipulated input at time  $k$ , the vector  $\mathbf{y}$  contains present and past values of the controlled output, the vector  $\mathbf{u}$  contains past values of the manipulated input, and  $\mathbf{x}$  is a vector of present and past

values of the state variables. The non-linear functions  $f_{nn}$  and  $g_{nn}$  implemented by the neural networks. This approach has several advantages. Because each neural network model only a portion of the non-linear process, the requisite size of the NN is reduced and the training becomes easier. A specific model structure is imposed leading to the possibility for greater model insight and facilitating an analytical model inversion procedure. The control law is

$$u(k) = \frac{\Delta y(k)}{f_{nn}(\mathbf{y}, \mathbf{u}, \mathbf{x})} + g_{nn}(\mathbf{y}, \mathbf{u}, \mathbf{x}) \quad (4.5)$$

where  $\Delta y(k)$  becomes  $y_{\text{setpoint}} - y(k)$  when perfect tracking is accomplished. Note that to train  $f_{nn}$  and  $g_{nn}$  two kinds of training sets are required: steady state data and transient data. Steady state data is used to train  $g_{nn}$  since it captures the steady state gain of the process. These data consist in process data at various steady-state values of the process inputs. Transient data is used to train  $f_{nn}$ . These data are obtained by forcing the control input  $u$  with a perturbation signal superimposed upon its nominal operation value. The magnitude of the perturbations is determined to make the response of the controlled output cover the expected range of operation. Once  $g_{nn}$  is properly trained, using transient data at each measured data point, the following quantity can be calculated

$$f_{nn}(\mathbf{y}, \mathbf{u}, \mathbf{x}) = \frac{\Delta y(k)}{u(k) - g_{nn}(\mathbf{y}, \mathbf{u}, \mathbf{x})} \quad (4.6)$$

and then  $f_{nn}$  can be trained.

#### **4.5.2 Fault diagnosis and neural networks**

An overview of the challenges of the industrial applications of fault diagnosis systems was carried out by Dash and Venkatasubramanian (2000). Multiscale analysis and dynamic PCA have been used for fault diagnosis (Luo, 1999). Also, neural networks have been employed to solve fault detection problems. Fault diagnosis is an essential ingredient property of an intelligent control system. Vaidyanathan and Venkatasubramanian, in 1992a used a MLP network to represent and diagnose a CSTR dynamic process. Trend data is the network input and the malfunction of six process variables is the network output. They tried several schemes of input the trend data and two ways of defining a fault function for the network output. They found that a moving average input scheme performs better. In the same year Vaidyanathan and Venkatasubramanian (1992b) studied the impact of hidden units and input units on fault space structure, specifically the effect of hidden units on the discrimination of decision regions of a non-linearly separable problem. They explained neural network structure using analytical geometry. Following the same thought, Kavuri and Venkatasubramanian (1993) proposed a neural network that, instead of computing the inner product between the input vector and the weight vector, computes an ellipsoidal function of the neuron inputs. This type of neurons is especially suitable for classification problems. This ellipsoidal function incorporates the notion of distance into networks, overcomes unnecessary extrapolation of

the network to regions in measurement space where no training patterns are available and it avoids arbitrary determination of the number of hidden nodes in classification problems. They stated that the inner product defines a hyper plane in the classification space, thus the space is divided in two unbounded regions, while the ellipsoidal function defines a hyper ellipsoid region that encloses a decision region. They used the mentioned scheme for fault diagnosis of a reactor-distillation column system, where fault and normal operation regions were classified. They compared their classifier with distance-based methods such as RBF networks and k-means clustering, that worked well when the patterns were available in large numbers and the classes were presented in nearly equal proportions. A more extend comparison of neural network classes for fault diagnosis was performed by Keyvan and Durg (1996). They compare all ART paradigms, MLP, Cascade Correlation and RCE networks for fault diagnosis in nuclear reactor systems. Another industrial work was done by Bissessur et al. (2000) where a wavelet neural network was used for fault detection in hot steel rolling. Through this method the manufacture performance of a hot rolling mill was enhanced.

#### **4.6 Optimal control**

Optimal control formulates the control problem as an optimization problem. In optimal control, the objective is to find an optimal control action that minimizes a cost or maximizes a profit function. This profit function can be written in terms of a desired or reference trajectory. Optimal control can be achieved by using any optimization method. MPC is an example of an optimal controller.

A common problem in optimal control is that the objective function and/or constraints of the optimization problem are usually non-linear. Extensive comparisons between optimal control and conventional control can be summarized as follows (Garcia and Morari, 1982). Optimal control yields improved servo and regulator behavior but the crucial robustness issue cannot be addressed directly. Weighting matrices and/or noise models have to be varied in a roundabout obscure fashion in the hope of achieving some robustness, which then has to be checked through simulation. It should be added that all model based methods and conventional controllers are equally incapable of handling constraints. Only in MPC and neuro dynamic programming, output constraints can be addressed. A review of the methods for optimal control can be summarized mostly as a review of optimization methods (e.g., Bertsekas and Tsitsiklis, 1989). Thus, the optimal control of a fermentation processes will be emphasized in the following sections.

##### ***4.6.1 Methods for optimal control used for fermentation processes : invertase production case study***

Optimization of fed-batch fermenters can substantially increase the profitability of these processes. Cuthrell and Biegler (1989) based the solution of the optimal control problem of a

fed-batch penicillin reactor in a successive quadratic programming and orthogonal allocation on finite elements. Luus (1993a) used iterative dynamic programming to find the optimal feed rate profile for the same bioreactor process and Banga et al. (1997) presented a fast stochastic dynamic optimization method to solve the same optimal control problem. They called their method integrated controlled random search for dynamic systems.

Ethanol production also has been studied. Fed-batch fermentation of *Saccharomyces cerevisiae* has been optimized by several authors. Modak and Lim (1987) provided the restrictions on the initial conditions, the fermentation kinetics and the objective function, needed for realization of the feedback optimization of ethanol and lysine fermentations. Luus (1993b) with a penalty function approach, used iterative dynamic programming to solve this ethanol production optimization problem. Wang and Cheng (1999) solved the simultaneous optimization of feeding rate and operation parameters for ethanol production using an evolutionary algorithm called hybrid differential evolution. Pushpavanam et al. (1999) solved both ethanol and lysine production optimal control problems stated by Modak and Lim (1987). They used sequential dynamic programming. Jayaraman et al. (2001) using the ant algorithm maximized the productivity of ethanol in a fed-batch process. He also used the same algorithm to maximize the profitability of the fed-batch production of an induced foreign protein by recombinant bacteria. The profit was stated in terms of the sales value of the protein product and the cost of the inducer.

The fed-batch ethanol fermentation through *Zymomous mobilis* also has been studied and optimized. Chiou and Wang (1999) used a kind of evolutionary algorithm, called hybrid differential evolution, to solve this fed-batch optimization problem. Previously, iterative dynamic programming was used by Wang and Chiou (1997) to solve the same problem and Chiou and Wang (1999) also used this optimal solution obtained by iterative dynamic programming, as starting point of a sequential dynamic programming method. They found their evolutionary algorithm to be superior.

The in vitro growth of hybridoma cells and the production of monoclonal antibodies by these cell lines were studied by Iyer et al. (1999). The fed-batch fermentation was optimized offline using the heuristic random optimizer procedure, and optimized online using a one step application of Newton's method, per control interval, in a technique called IMPOL. The same hybridoma reactor was optimized by Roubos et al. (1999) using an evolutionary program, based on real-code genetic algorithm to calculate the optimal control action. He used also this method to find optimal control policies for the Tholodur and Ramirez (1993) bioreactor. Dhir et al. (2000) maximized the cell mass and monoclonal antibody production of the same fed-batch hybridoma cell culture. They used fuzzy logic techniques for the adjustment of the dynamic parameters of the model and a heuristic random optimizer (HRO) to optimize the feed rates of glucose and glutamine.

The degradation of phenol has been studied by Cruickshank et al. (2000) in a two-phase partitioning fed-batch bioreactor. The maximal phenol consumption in a fixed time interval was achieved using iterative dynamic programming. Further bibliography on the optimization of fed-batch reactors can be found in Harting et al. (1995). They present a comparison of optimization methods for fed-batch reactors. Optimal operation of batch processes was studied by Srinivasan et al. (2001). Direct and variational optimization approaches are discussed specifically for batch processes. Optimal control a fed-batch fermenter is usually based on a nominal process model. Parameters uncertainties are not taken into account. Usually the results obtained with nominal model parameters can be quite sensitive to the uncertainty in the parameter values. Zafiriou and Zhu (1989) presented a methodology to find the optimal feed rate profile for a fed-batch fermentation process in the presence of plant-model mismatch. They used batch to batch information to gradually reach the optimum. Kuhlmann et al. (1998) presented a method for obtaining robust optimal control profiles in the presence of uncertainty in the model parameters.

Other biotechnological systems that have been optimized are the metabolic pathways of biochemical elements inside the cell. For example, the rates of ethanol and glycerol and the carbohydrate production in *Saccharomyces cerevisiae* yeast were optimized by Torres et al. (1997). His indirect linear optimization lead to profiles of enzyme activities that are compatible with the physiology of the cells. Later, Rodriguez-Acosta et al. (1999) optimized the same biotechnological process using a non-linear optimization technique based on a stochastic multi-start search algorithm. They found qualitative agreement between the profiles they obtained and the profiles reported by Torres et al. (1997). Another example is the work done by Cameron et al. (1998). He studied different pathways for the conversion of sugars to propadeniol. Linear optimization studies indicated that, under aerobic conditions, propadeniol yields approach the theoretical maximum values.

In this work we are going to focus on the optimization of fed-batch bioreactors where the substrate(s) is added continuously in an otherwise batch operation mode. It is necessary to determine the optimal substrate feed rate profile in order to achieve the maximum fermentation profit. Neuro dynamic Programming (NDP) is an optimization method that can be used to overcome the limitations of previous optimization methods presented and discussed before (Patkar et al., 1993; Chaudhuri and Modak, 1998) for cloned invertase production in *Saccharomyces cerevisiae* yeast in a fed-batch fermentation process. The goal is to find the optimal control action at any time during the fermentation process. An optimal control action in this case should be the one that maximizes the productivity and, at the same time, minimizes the total fermentation time. To accomplish this goal a neuro dynamic programming methodology coupled with MLP or with fuzzy ARTMAP is used. This NDP method is employed in conjunction with fuzzy ARTMAP neural networks to improve its



performance. The aim is to find an optimal feeding profile  $\pi$  that could adapt itself when a disturbance arises. Mathematically this objective could be written as

$$\pi = \arg \max_u \{ \text{productivity} - \lambda \cdot \text{final time} \} \quad (4.7)$$

where  $u$  belongs to the set of all possible values of the manipulated variable, in this case the substrate feed rate, and,  $\lambda$  is a positive constant that penalizes the invested time in the fermentation process. The main constraint of this optimization is the total bioreactor volume, which is assumed to have a maximum value. This optimal control problem is solved using NDP. Dynamic programming and NDP are introduced in the following sections. Next, it is explained how NDP methodology is applied to solve the optimal control problem of the invertase fermentation process stated in equation (4.7).

#### **4.6.2 Dynamic programming and reinforcement learning**

Dynamic programming is an approach to model dynamic decision problems, to analyze the structural properties of these problems and to solve them. Dynamic programming is a sequential decision making procedure under uncertainty. It could be used for control and optimization of stochastic problems. The advantage of using dynamic programming over other gradient-based methods is that penalty functions do not have to be continuously differentiable. Using dynamic programming, constraints on both state and manipulated variables are handled easily without increasing the computational burden. It can handle heavily constrained optimization problems as well as singular optimal control problems. In this method, the process is modeled as a chain of consecutive transitions from one state to another.

We have a dynamical system whose evolution is influenced or controlled by our decisions or control actions. The way each transition is made depends on the control or decision variable. The decision made at any given time can, in general, depend on the state of the system. Each action has an associated cost or reward. The objective is to maximize or minimize the total incurred cost, obtained from the transitions needed to reach the final desired process state from the initial state. The set of all the decisions made is called a policy. So, an optimal cost has an associated optimal policy. The dynamic programming goal is to select a decision making rule, also called feedback policy, which optimizes a certain performance criterion. Using this approach control and optimization problems can be solved, in principle, with the classical methods of dynamic programming. In practice, however, the applicability of this method to many important problems is limited by the enormous size of the underlying state spaces, and the complexity of the iterative algorithm involved. This is the so-called Bellman's "curse of dimensionality". Neuro dynamic programming, or "Reinforcement Learning", which is the term used in the Artificial Intelligence literature, uses neural network and other

approximation architectures to overcome the bottlenecks to the applicability of dynamic programming. This NDP methodology allows systems to learn about their behavior through simulation, and to improve their performance through iterative reinforcement. There are two ways to attain reinforcement. In one approach, called value function approximation, simulation is used to tune the parameters of a "value function" that quantifies the relative desirability of different states in the state space. In mathematical terms, the objective is to compute an approximate solution to Bellman's equation, which is then used to construct near-optimal policies. This approach was studied by Tsitsiklis (Bertsekas and Tsitsiklis, 1996). Another approach, called optimization in policy space, involves the tuning of policy parameters in a direction of improvement.

Some of the research in NDP is theoretical in nature, aiming at understanding the convergence and degree of suboptimality of different algorithms, while some involves the application of this methodology to specific problem domains. A survey of NDP methodologies is given in the following section.

#### *4.6.2.1 NDP Methodologies*

- **Policy space and actor-critic algorithms**  
Instead of tuning the parameters of a value function, or tune directly the parameters of a policy, a parametrically described class of policies can be assumed. A class of methods that can be interpreted in terms of estimated Q-factors was studied by Marbach and Tsitsiklis 2001. Such methods may suffer from large variance and slow convergence. This can be partially mitigated by certain variants, e.g., by introducing a discount factor (Marbach and Tsitsiklis, 2002). Even better, learning in policy space and value function approximation can be combined. This is what actor-critic methods do. It turns out that once a policy parameterization is fixed, it prescribes a natural set of "features" to be used in value function approximation, and one obtains algorithms with provable convergence properties (Konda and Tsitsiklis, 2001). Policy learning in actor-critic algorithms takes place at a slower rate than value function approximation. Thus, the convergence analysis of actor-critic algorithms relies on the convergence of certain two-time scale stochastic approximation algorithms (Konda and Tsitsiklis, 2002).
- **Average cost temporal difference learning**  
Temporal difference methods can be applied to average cost problems. The convergence and approximation error guarantees are essentially the same as for discounted problems. Thus, there is no need to use discounted formulations as a proxy for undiscounted ones (Tsitsiklis and van Roy, 1999a). The properties of average and discounted criterion temporal difference methods are compared in more detail by Tsitsiklis and van Roy (2002a)

- Convergence of methods based on value function learning  
The convergence of a method that uses a lookup table representation of the value function, simulation using a greedy policy, and plain "Monte-Carlo" (averaging) for learning the value function can be found in Tsitsiklis (2002b)
- Optimal stopping problems is the only known class of problems for which convergence is guaranteed for methods like Q-learning, with arbitrary linearly parameterized value function approximators, and without a restriction to a fixed policy (Tsitsiklis, 1999b)
- Temporal difference methods, for the single policy case, and with linearly parameterized function approximators, are guaranteed to converge. The approximation error obtained in the limit is not too far from the best possible approximation error under the particular approximation architecture (Tsitsiklis and van Roy, 1997). Convergence results and approximation error bounds for Q-learning type methods for certain special types of function approximation, e.g., state aggregation was found by Tsitsiklis, 1996. Q-learning and the temporal difference methods (with a lookup table representation) are viewed as stochastic approximation methods for solving Bellman's equation. Their convergence is established by first developing a stochastic approximation theory for the case where the iteration mapping is a contraction with respect to a weighted maximum norm. (Tsitsiklis, 1994)
- Rollout algorithms  
Starting with a good heuristic and carrying out what is essentially single policy iteration, in the dynamic programming sense, provides a systematic method for improving the performance of a heuristic, and has great promise in practical settings (Bertsekas and Tsitsiklis 1996).

#### *4.6.2.2 Applications*

Neuro dynamic programming has been used to solve several optimization problems of different fields. Since 1999 neuro dynamic programming has been proposed in finances to price complex American options (van Roy and Tsitsiklis 2001 and Tsitsiklis and van Roy 1999c). It has been used also for inventory management (Van Roy et al., 1996). In communication networks NDP has been used for control and routing (Marbach et al. 2000). Card et al., 1997 used a cascade neural network and the policy-iteration algorithm to provide suggested set points for a plasma etch process.

Since NDP is able to deal with complex optimization problems, it is believed in this work that it could be successfully used to optimize a non-linear fed-batch bioreactor. The next section explains in detail the NDP procedure. Then, a description of how NDP is used to solve the invertase fed-batch optimization problem is stated.

#### 4.6.3 Neuro dynamic programming approach

In dynamic programming the objective function of the cost optimization has two parts. One of them is the cost incurred in the transition from the actual state to the next state. The other part is a term called cost-to-go. The cost-to-go is the optimal cost incurred from the next state to the final state. The cost-to-go is a measure of the desirability of the next state. Mathematically, the optimal cost for a given state  $x_k$  of a deterministic problem can be written through Bellman's equation as

$$\begin{aligned} J^*(x_k) &= \min_u [g(x_k, x_{k+1}, u) + J^*(x_{k+1})] \\ &= \min_u [g(x_k, x_{k+1}, u) + J^*(f(x_k, u))] \end{aligned} \quad (4.8)$$

where  $x_{k+1}$  is the next state,  $g(x_k, x_{k+1}, u)$  is the cost associated to the transition from the actual process state  $x_k$  to the next process state  $x_{k+1}$ , and  $f(x_k, u)$  is a function of the process dynamics.

To solve this minimization problem, different costs incurred in the transitions associated to all possible control actions should be explored. Also in order to find the optimal policy for a given initial state, one has to calculate an associated cost-to-go for each of all the states of the state space of the process. As it was said before, in the case of problems with very large number of states, this task is computationally extremely demanding. In addition, the number of states to be explored increases as the dimension of the state vector increases. The solution to this curse of dimensionality is to use suboptimal methods that approximate the cost-to-go of each state to a parametric function, like neural networks. Thus, the above equation can be written as

$$\begin{aligned} J^*(x_k) &= \min_u \left[ g(x_k, x_{k+1}, u) + \tilde{J}(x_{k+1}, \mathbf{r}) \right] \\ &= \min_u \left[ g(x_k, x_{k+1}, u) + \tilde{J}(f(x_k, u), \mathbf{r}) \right] \end{aligned} \quad (4.9)$$

where  $\mathbf{r}$  is a vector of parameters and  $\tilde{J}$  is a map of the state space to the optimal cost-to-go of each state. One can use any parametric function for this mapping but artificial neural networks have been seen as universal function approximation (Cybenko, 1989), and could be

applied to feature extraction. This suboptimal dynamic programming method, which uses a neural network, is called neuro dynamic programming.

In order to find  $\tilde{J}$  through a NN, data of the states and its associated optimal cost-to-go are needed to train the network. There are no readily training pairs, so one has to evaluate  $u$  policies through simulation and calculate an approximate value for the cost-to-go and then improve it. This improvement is called value iteration and it is done through Bellman equation. The value iteration algorithm can developed with the following procedure (Lee and Lee, 2001):

1. Use a suboptimal policy into a process simulation of the closed-loop system to generate cost-to-go vs. state data.
2. Fit a NN to approximate a parametric function of the states to the cost-to-go. This gives an approximate function of the cost-to-go for the relevant region of the state space. This first map is denoted  $J^0$  in this work.
3. To improve the approximation (due to the use of a suboptimal control law), use the Bellman iteration to find the best cost-to-go map,

- 3.1. With the current estimate  $\tilde{J}^i(f(x_k, u), \mathbf{r}_i)$  calculate  $J^{i+1}$  for the given sample of states  $x$  by solving

$$J^{i+1}(x_k) = \min_u \left[ g(x_k, x_{k+1}, u) + \tilde{J}^i(f(x_k, u), \mathbf{r}) \right] \quad (4.10)$$

- 3.2. Fit an improved cost-to-go approximation to the new  $x$  vs.  $J^{i+1}$  data so that

$$\tilde{J}^{i+1}(x_k) \equiv \tilde{J}^{i+1}(f(x_k, u), \mathbf{r}_{i+1}) \cong J^{i+1} \quad (4.11)$$

- 3.3 Repeat until  $\sum_{k=1}^N \left| J^{i+1}(x_k) - \tilde{J}^i(x_k) \right| / N < \varepsilon$ , where  $\varepsilon$  is a small number

and where the order of  $\varepsilon$  depends on the order of the cost-to-go.

The main premise here is that not all points in the state space are relevant for control and one can obtain the most relevant sample points by performing closed-loop simulations. This method is less computationally demanding, even for very large dimensional systems, because the operating region of the state space that the closed-loop system visits may be relatively low in dimension. Also, the first iteration starts with a very good approximation of the cost-to-go due to the use of a suboptimal (but good) policy. The on-line implementation of this controller involves the converged cost-to-go function. The value of the control variable is found solving at each sample time

$$\begin{aligned}
J^*(x_k) &= \min_u \left[ g(x_k, x_{k+1}, u) + \tilde{J}(x_{k+1}, \mathbf{r}) \right] \\
&= \min_u \left[ g(x_k, x_{k+1}, u) + \tilde{J}(f(x_k, u), \mathbf{r}) \right]
\end{aligned} \tag{4.12}$$

The described procedure can also be improved with the so called "policy iteration" method. This method constitutes an upper loop iteration where the closed-loop simulation of the process with the new control law defined in the above equation yields a new more relevant set of states and more accurate cost-to-go data for them. This procedure is used to solve the optimal control of invertase fermentation in a fed-batch bioreactor. How it is done is described in the following section.

#### **4.6.4 Optimal control trough NDP: invertase production case study**

The fed-batch fermentation process considered here consists of the production of invertase by recombinant yeast cell using glucose as substrate. Patkar and Seo (1992) reported the fermentation kinetics of invertase production in fed-batch cultures. They reported also experimental data for cell density (expressed as optical density OD), glucose (G) concentration and specific invertase activity (I) obtained with six different glucose-feeding strategies in a 1.2 liters bioreactor. The number of state variables of the process is 4: OD, G, I and the bioreactor volume (V). The state space of the system is considered to be of fourth order. The productivity of this fermentation process at a certain point in time t is given by

$$\text{Productivity}(t) = I(t) \cdot OD(t) \cdot V(t) \tag{4.13}$$

Thus, the optimization problem consists in maximizing the profitability given by

$$\max_u \{ I(t_f) \cdot OD(t_f) \cdot V(t_f) - \lambda \cdot t_f \} \tag{4.14}$$

where  $t_f$  is the fermentation ending time. The optimization of invertase production can be seen also as the minimization of the overall cost of production

$$\min_u \{ \lambda \cdot t_f - I(t_f) \cdot OD(t_f) \cdot V(t_f) \} \tag{4.15}$$

The main constraint of the above optimization problem, is the maximum bioreactor volume

$$V(t_f) < 1.2 \text{ liters} \tag{4.16}$$

Thus, the objective of the optimal control of this process is to find an optimal feeding profile  $\pi$  such as

$$\pi = \arg \min_u \{ \lambda \cdot t_f - I(t_f) \cdot OD(t_f) \cdot V(t_f) \} \tag{4.17}$$

To solve this problem, a neuro dynamic approach is used. For the invertase production optimization the cost-to-go function can be expressed as

$$J^*(\mathbf{x}) = \min_u \left[ \lambda \Delta t + \tilde{J}(\mathbf{f}(\mathbf{x}, u), \mathbf{r}) \right] \tag{4.18}$$

where  $J^*$  is the optimal cost-to-go,  $\mathbf{x}=(OD, G, I, V)$  is the state of the fermentation process,  $u$  is the substrate feed rate,  $\lambda$  is stated as above and  $\Delta t$  the time step or the expended time between two consecutive states. Note that the transition cost given by the term  $\lambda \Delta t$  is only a function of time and that  $\tilde{J}$  is the state space-cost-to-go map to be performed by a MLP or a fuzzy ARTMAP neural system. The weight vectors associated to MLP or to the fuzzy ART modules is given by the set of parameters  $\mathbf{r}$ . Finally  $\mathbf{f}(\mathbf{x}, u)$  is a multidimensional function that describes the dynamics of fermentation process. This function is the model of cloned invertase fermentation as described in the fed-batch fermentation chapter. With the NDP approach, the optimal feeding profile can be written as

$$\pi = \arg \min_u \left[ \lambda \Delta t + \tilde{J}(\mathbf{f}(\mathbf{x}, u), \mathbf{r}) \right] \quad (4.19)$$

To solve this equation, the state space- cost-to-go map  $\tilde{J}$  is needed. Once it is obtained through a NN, the above equation can be implemented on-line into a controller. In this way, the optimal policy found  $\pi$  could adapt itself in case of disturbances.

The invertase production optimization problem modeled this way is a deterministic, finite horizon NDP problem, where the overall sum of the transition costs is minimized. The described methodology is used to find an optimal controller for invertase production in a fed-batch bioreactor. The results of this procedure using two kinds of neural network systems are presented in next chapter.





## Chapter 5 : Results

The results presented at this chapter can be divided into two parts. First the results obtained with a model-based control of fed-batch multivariable fermentation process are presented. Secondly, are presented the results obtained when an optimal controller of an invertase fermentation on a fed-batch bioreactor is implemented. The corresponding manuscripts of two publications are respectively attached in appendix A and B.

### 5.1 Model based control

To build a suitable control model, the multivariable fed-batch bioreactor was simulated and process data was obtained as described in section 3.4. Using these data several MLP and RBF neural network models were obtained and tested following the steps shown in Figure 3.11, for both the direct and the inverse control models schemes depicted in Figure 3.12. The results obtained with the testing schemes illustrated in Figure 5.1 for the direct (a) and inverse (b) models are presented below.

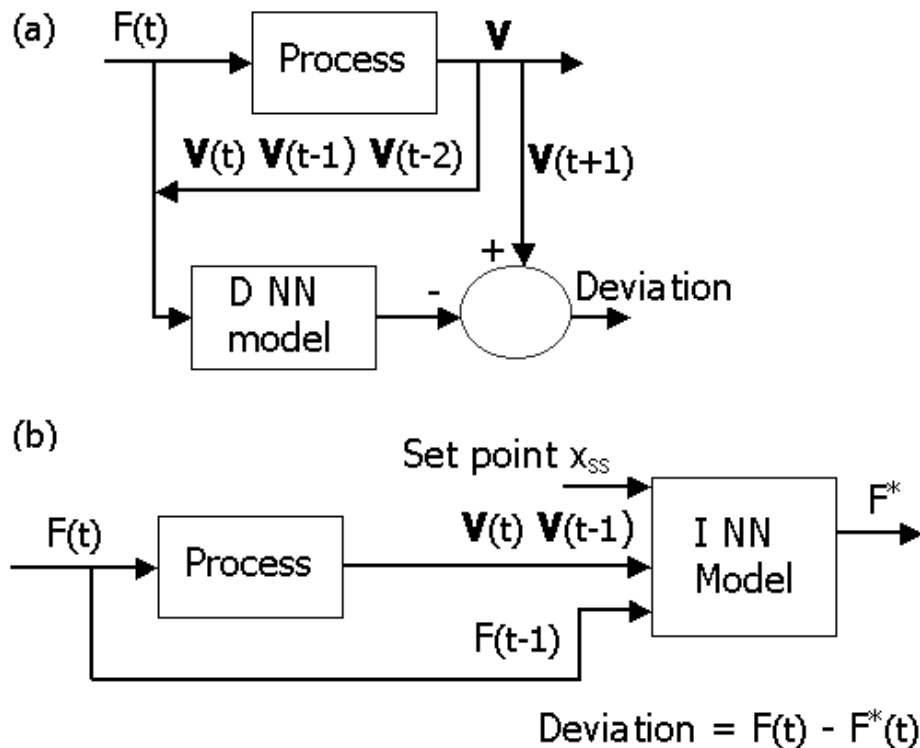


Figure 5.1 Test schemes for the obtained direct and inverse process models of the multivariable fed-batch bioreactor. The models prediction error (deviation) is computed as shown: (a) Test for direct process models (b) Test for the inverse process models

### ***5.1.1 Direct model***

The direct model of the process must be highly reliable because its predictive task inside the controller and its repercussions in the controller performance are critical. It should track the process behaviour, following exactly the same routes as those for the simulated process, and yield exactly the same process outputs from the same inputs. Thus, the direct model must accurately represent the process operation at the steady state point and the transient behaviour of the process until a new steady state is reached when the set point changes.

The performance of the direct model of the process was checked for (a) the steady state process operation, (b) the transient process operation and (c) a plant/model mismatch caused by a process, as indicated in Table 3.3. The performance of the control model was measured in terms of the average deviation between the NN model outputs and the values expected in a simulated "infinite" (10,000 time steps) process.

#### *(a) Steady State Performance*

The steady state process operation studied was  $x=0.2475 \text{ kg/m}^3$ ,  $p=1.7499 \text{ kg/m}^3$  and a constant  $F = 8.3 \cdot 10^{-9} \text{ m}^3/\text{s}$ . The direct model test results in terms of the individual deviations for each variable and the average ones for the steady state operation are presented in Table 5.1. The average deviations shown in this table indicate that the RBF models are superior in performance than the MLP models. The best model is a RBF neural network with goal 3.0, followed closely by another with goal 3.1. The pH values in the bioreactor are almost kept constant and equal to the initial value of 9.15. The largest deviations observed in this and following tables are caused by variations in the substrate flow since the acid and base flow rates  $F_a$  and  $F_b$  were kept equal to zero in all experiments.

**Table 5.1** Average relative errors of the direct model outputs with respect to the final process values for steady state operation of the multivariable fed-batch bioreactor

NN	x	s	p	T	pH	O <sub>2</sub>	Average deviation
RBF goal 3.0	0.06	0.25	0.00	0.00	0.57	0.00	0.15
RBF goal 3.1	0.13	0.29	0.00	0.00	0.60	0.00	0.17
RBF goal 4.0	0.60	0.20	0.00	0.00	0.63	0.00	0.24
RBF goal 2.75	0.08	1.01	0.00	0.00	0.51	0.00	0.27
RBF goal 3.5	0.65	0.36	0.00	0.00	0.63	0.00	0.27
RBF goal 3.4	0.65	0.36	0.00	0.00	0.63	0.00	0.27
RBF goal 3.2	0.06	1.63	0.00	0.00	0.62	0.00	0.39
RBF goal 3.3	0.06	1.63	0.00	0.00	0.62	0.00	0.39
RBF goal 2.0	0.03	2.26	0.00	0.00	0.03	0.00	0.39
MLP 19-18-12-6	1.75	0.69	0.04	0.01	0.56	0.00	0.51
RBF goal 0.8	0.00	3.51	0.00	0.00	0.00	0.00	0.59
RBF goal 2.5	0.04	4.84	0.00	0.00	0.19	0.00	0.85
MLP 19-10-5-6	5.84	0.78	0.01	0.01	0.61	0.02	1.21
MLP 19-5-6	3.77	3.61	0.00	0.00	0.63	0.00	1.34
RBF goal 1.0	0.01	8.40	0.00	0.00	0.01	0.00	1.40
MLP 19-18-12-6+PCA	5.30	6.71	0.01	0.00	0.44	0.02	2.08
MLP 19-18-12-6+CV	0.75	18.71	0.02	0.00	1.12	0.06	3.44

*(b) Transient process operation*

*(b.1.)* The first transient case analysed was caused by a positive step change of 10% in the substrate flow rate ( $F$ ), which is the control variable. The test results are shown in Table 5.2. To simulate correctly the time response of the process, a simultaneous increment in the stirrer rpm was also implemented to increase the amount of dissolved oxygen. This combined action was absolutely necessary to reach a new steady state operation. In this case the best NN architectures for the direct process model are also based on RBFs and again the best model in terms of average deviations is the architecture with goal 3.0. As in the steady state operation, the pH is usually the variable with the highest deviations.

**Table 5.2** Average relative errors of the direct model outputs with respect to the process expected values for a positive step in the manipulated variable of the multivariable fed-batch bioreactor

NN	x	s	p	T	pH	O <sub>2</sub>	Average deviation
RBF goal 3.0	0.03	0.35	0.00	0.00	0.59	0.00	0.16
RBF goal 3.1	0.10	0.26	0.00	0.00	0.65	0.00	0.17
RBF goal 2.0	0.03	1.10	0.00	0.00	0.03	0.00	0.19
RBF goal 4.0	0.42	0.44	0.00	0.00	0.67	0.00	0.26
RBF goal 3.5	0.41	0.83	0.00	0.00	0.67	0.00	0.32
RBF goal 3.4	0.41	0.83	0.00	0.00	0.67	0.00	0.32
RBF goal 0.8	0.01	2.02	0.00	0.00	0.00	0.00	0.34
RBF goal 2.75	0.09	1.41	0.00	0.00	0.55	0.00	0.34
RBF goal 3.2	0.10	1.59	0.00	0.00	0.66	0.00	0.39
RBF goal 3.3	0.10	1.59	0.00	0.00	0.66	0.00	0.39
MLP 19-18-12-6	0.96	1.51	0.04	0.03	0.68	0.00	0.54
MLP 19-10-5-6	4.56	0.57	0.01	0.01	0.74	0.02	0.98
RBF goal 2.5	0.05	6.13	0.00	0.00	0.18	0.00	1.06
RBF goal 1.0	0.01	6.44	0.00	0.00	0.01	0.00	1.08
MLP 19-5-6	1.98	4.66	0.01	0.00	0.63	0.00	1.21
MLP 19-18-12-6+PCA	6.58	6.47	0.01	0.00	0.49	0.02	2.26
MLP 19-18-12-6+CV	3.37	12.32	0.05	0.19	0.45	0.05	2.74

(b.2.) The transient of the direct model was also studied for the case of multiple random steps in  $F$ . These steps were between  $\pm 10\%$  of  $F$  at the operation point, and occurred at randomly distributed instants within 50 to 200 process time steps. The results obtained are shown in Table 5.3. In this case the best NN architecture for the direct model are the RBF models with goals 2.75 and 3.0, followed closely by the MLP 19-18-12-6.

**Table 5.3** Average relative errors of the direct model outputs with respect to the process expected values for multiple random steps in the manipulated variable for the multivariable fed-batch bioreactor

NN	x	s	p	T	pH	O <sub>2</sub>	Average deviation
RBF goal 2.75	0.11	1.90	0.00	0.00	0.57	0.00	0.43
RBF goal 3.0	0.07	2.31	0.00	0.00	0.62	0.00	0.50
MLP 19-18-12-6	1.10	1.35	0.04	0.02	0.66	0.00	0.53
RBF goal 3.2	0.27	2.79	0.00	0.00	0.67	0.00	0.62
RBF goal 3.3	0.27	2.79	0.00	0.00	0.67	0.00	0.62
RBF goal 3.1	0.19	2.98	0.00	0.00	0.66	0.00	0.64
RBF goal 4.0	0.90	2.37	0.00	0.00	0.67	0.00	0.66
RBF goal 3.5	0.88	2.82	0.00	0.00	0.66	0.00	0.73
RBF goal 3.4	0.88	2.82	0.00	0.00	0.66	0.00	0.73
RBF goal 2.0	0.08	5.01	0.00	0.00	0.04	0.00	0.85
RBF goal 2.5	0.08	5.15	0.00	0.00	0.18	0.00	0.90
MLP 19-10-5-6	3.97	3.20	0.01	0.02	0.76	0.02	1.33
MLP 19-5-6	2.59	5.17	0.01	0.01	0.61	0.00	1.40
RBF goal 0.8	0.04	11.59	0.00	0.00	0.01	0.00	1.94
MLP 19-18-12-6+PCA	6.82	7.63	0.01	0.00	0.50	0.02	2.50
MLP 19-18-12-6+CV	3.63	11.92	0.05	0.18	0.75	0.06	2.76
RBF goal 1.0	0.05	22.70	0.00	0.00	0.01	0.00	3.79

*(c) Unexpected disturbance*

The direct model of the process should be robust against unknown disturbances. It was tested for unknown disturbances in the pH. The disturbances, hypothetically caused by an uncertainty in the value of the constant of dissociation of the acid, were such that measured pH was lower than the pH used for training the NN models. The process operation steady state point and the transient process response to a positive step in F during the pH disturbance tests are shown in Tables 5.4 and 5.5, respectively. In the steady state process operation test (Table 5.4), the MLP 19-5-6 architecture is the best model. This NN model is also the best to perform in the transient process test results summarized in Table 5.5.

**Table 5.4** Average relative errors of the direct model outputs with respect to the expected process values for a steady state process operation with a pH perturbation of the multivariable fed-batch bioreactor

NN	X	s	p	T	pH	O <sub>2</sub>	Average deviation
MLP 19-5-6	17.59	15.20	0.02	0.05	29.64	0.01	10.42
MLP 19-18-2-6	29.26	36.32	0.02	0.92	2.70	0.04	11.54
MLP 19-18-12-6+CV	10.64	71.92	0.01	1.01	1.87	0.08	14.26
RBF goal 3.0	13.54	57.12	0.01	0.01	24.51	0.01	15.87
RBF goal 3.1	0.21	65.48	0.01	0.01	33.02	0.01	16.46
MLP 19-18-12-6+PCA	79.26	98.03	31.48	1.28	1.16	66.11	46.22

**Table 5.5** Average relative errors of the direct model outputs with respect to the expected process values for a positive step in the manipulated variable with a pH perturbation of the multivariable fed-batch bioreactor

NN	x	s	p	T	pH	O <sub>2</sub>	Average deviation
MLP 19-5-6	12.95	32.12	0.01	0.05	29.48	0.00	12.43
MLP 19-18-12-6	53.59	11.41	0.01	9.76	29.90	0.00	17.44
MLP 19-18-12-6+CV	11.27	91.80	0.00	0.43	2.21	0.07	17.63
RBF goal 3.0	22.07	100.12	0.01	0.00	21.43	0.00	23.94
MLP 19-18-12-6+PCA	82.26	97.06	31.47	1.27	1.21	66.12	46.56

Once the best NN models have been found it would be convenient to establish a consensus about which one of the obtained and tested direct models could be considered the most suitable one for the control system. The idea is to rank the NN models based on different aspects of the control system performance. Three requirements related to performance are considered here.

The direct model should perform well (i.e. have a low average of deviation) in regulatory control. It should also perform well in a servo control and it should be accurate enough to ensure the robustness of the controller against plant/model mismatches. Weights of 60%, 30% and 10% have been heuristically assigned to these 3 controller characteristics, respectively. When this criterion is used to rank the test results of the direct process model the best NN direct model is the RBF with goal 3, as shown in Table 5.6, followed very closely by MLP 19-5-6 and the rest of MLP models. Obviously, different criteria and weights could have been selected, as is usually the case in the fields of synthesis and design of chemical engineering processes (Biegler et al., 1997).

**Table 5.6** Ranking of controller performance for direct process models of the multivariable fed-batch bioreactor

NN / Importance in controller performance	Servo control		Regulatory control		Robustness	Rank
	60%		30%		10%	
	SS	Multiple Step	Positive Step	pH disturb. Positive Step	pH disturb. SS	
RBF goal 3.0	0.15	0.50	0.16	23.94	15.87	2.50
MLP 19-5-6	1.34	1.40	1.21	12.43	10.42	2.56
MLP 19-18-12-6	0.51	0.53	0.54	17.44	17.35	2.71
MLP 19-18-12-6+CV	3.47	2.76	2.74	17.63	15.72	4.79
MLP 19-18-12-6+PCA	2.08	2.50	2.26	46.56	46.22	7.95

### 5.1.2 Inverse model

A similar analysis has been carried out for the inverse process model. The performance of this inverse model has to be very robust in front of changes in the set point since it will have an actuator role inside the controller in the vast majority of control schemes considered in practice. For this reason, the current and the modifications of the previous MLP-based inverse process models proposed by Hussain and Kershenbaum (2000) and Aoyama and Venkatasubramanian (1995) given in Figure 3.12(c) were tested with random step variations around the process operational state at the set point input to the NN model.

The tests results for the current and modified previous inverse models are shown in Table 5.7. The best inverse model of the three in Figure 3.12(c) is the current scheme with a MLP 19-11-7-1 architecture, followed by the MLP model of Aoyama and Venkatasubramanian (1995) and by the current scheme with a RBF goal 1.0 architecture in terms of average deviations. Current models are faster when tracking the set point and are more stable than both modified previous models.

**Table 5.7** Average relative errors of the inverse model output with respect to the expected process value for random values in the set point for the multivariable fed-batch bioreactor

NN	Random set point
Current MLP 19-11-7-1	1.16
MLP Aoyama & Venkatasubramanian (1995)	1.31
Current RBF goal 1.0	1.46
MLP Hussain & Kershenbaum (2000)	1.66
Current MLP 19-11-7-1+CV	2.06

## 5.2 Optimal control

The optimal control problem of invertase production in a fed-batch bioreactor is solved using neuro dynamic programming approach, presented in section 4.5. The objective of the optimization procedure of the fed-batch bioreactor, stated in equation (4.17), is to find the optimal feeding profile  $\pi$  defined in equation (4.17).

The first step on any the NDP optimization is to obtain a sub-optimal cost-to-go value for each possible state of the problem. To obtain a sub-optimal cost-to-go value for each possible state of the fermentation, the invertase fermentation process was simulated using Patkar's model (1993) with 36 different suboptimal feeding policies and three different initial fermentation volumes. The simulation initial volumes were 0.4, 0.6, 0.8 liters. The suboptimal policies were chosen in a way that they have the same shape of the optimal policies found by both Patkar (1993) and Chaudhuri (1998). Those policies can be described in the following manner. When the fermentation starts, there is no feeding rate until time  $t_i$ . Then, the feeding flow rate increases potentially until the bioreactor volume is reached. Then fermentation continues until the system arrives to the maximum profitability value, i.e., the optimum final time ( $t_f^*$ ) for a given process trajectory. The behavior of the feed rate flow can be expressed mathematically by

$$u(t, t_i, b) = \begin{cases} 0 & \text{if } t < t_i \\ 0.02 * (1 + b * (t - t_i)^{2.2}) & \text{otherwise} \end{cases} \quad (5.1)$$

where  $b$  is a parameter used to control the growth rate of the feed rate flow. The values of  $b$  and  $t_i$  used to generate different suboptimal policies were:  $b=[0.05,0.07,0.10,0.13]$  and  $t_i=[1,2,3,4,5,6,7,8,9]$ . Figure 5.2 shows the different policies for an initial volume of 0.6 liters, obtained with initial times of 3, 5 and 7 h:  $u(t,3,b)$ ,  $u(t,5,b)$ ,  $u(t,7,b)$ . Also shown are the optimal policies obtained by Patkar (1993) and Chaudhuri (1998). For a given initial time, the differences on the policies drawn in this figure are the values of  $b$ .

The response of the key states of the fermentation process ( $x$ ,  $s$ ,  $p$ ,  $V$ ) and its associated profit curve when the suboptimal policy  $u(t,5,0.13)$  is used, is shown in Figure 5.3. The profit is given by:

$$profit = I(t_f) \bullet OD(t_f) \bullet V(t_f) - \lambda \cdot t_f \quad (5.2)$$



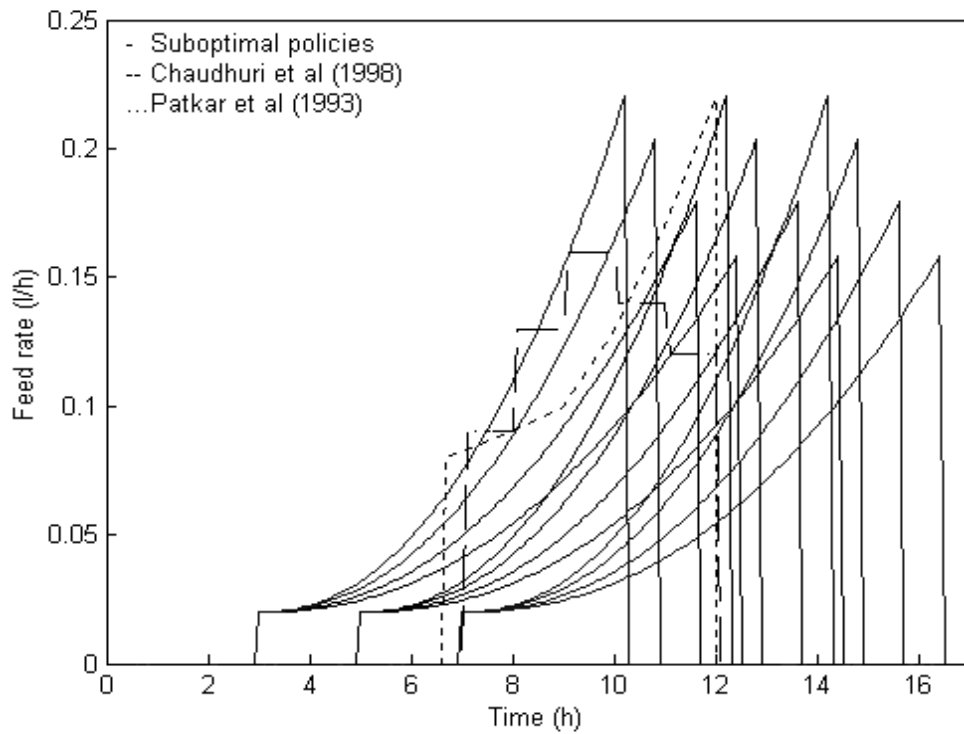


Figure 5.2 Different policies used for an invertase bioreactor with initial volume of 0.6 liters. (—) Suboptimal policies given by eq. 5.1 with initial times of 3, 5 and 7 (---) Optimal policy by Chaudhuri and Modak (1998) (···) Optimal policy by Patkar et al. (1993)

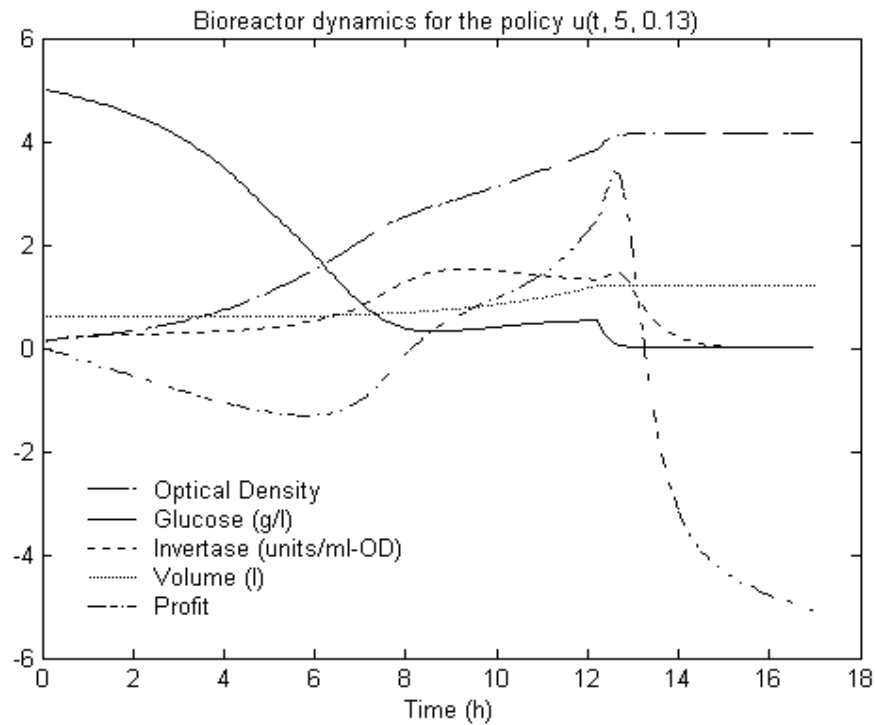


Figure 5.3 Fermentation dynamics for a policy  $u(t, 5, 0.13)$  given by eq. 5.1. Cells, glucose and invertase concentrations and fermentation volume are shown. Also shown is the associated profit when  $\lambda = 0.3$

The main constraint of this process is the total bioreactor volume, which has a maximum value of 1.2 liters. So the flow rate of substrate follows the given policy until the maximum volume is reached. At that time, the feed flow stops and the fermentation continue until the higher productivity is found. A larger fermentation process is shown in Figure 5.3. In that fermentation, the microorganisms consume all the amount of glucose available in the fermentor and the invertase production decreases. Because time goes on, the profit gets lower and lower.

The effect of different fermentation initial conditions on the productivity is shown in Figure 5.4. For a given initial volume, at time 7.4 h the feed rate starts at 0.18 l/h. It remains constant until the maximum value is reached. The fermentation continues until the maximum profit value is attained. The initial state for each line is different, but the operation conditions are the same for all processes. It could be seen in this figure that the maximum values for the profit are obtained when different initial fermentation volumes are used. This is the reason why three different initial volumes were chosen as different initial conditions.

A total of 9328 state points were generated through 108 simulations of the invertase fermentation using the 36 suboptimal policies generated with equation (5.1) and the three different initial fermentation volumes. The explored state space is shown in Figure 5.5. In this figure each drawing represents a projection on a 2-dimensional space of the state points of the 4-dimensional state space. For a 4 dimensional space, a total of six 2D representations are needed.

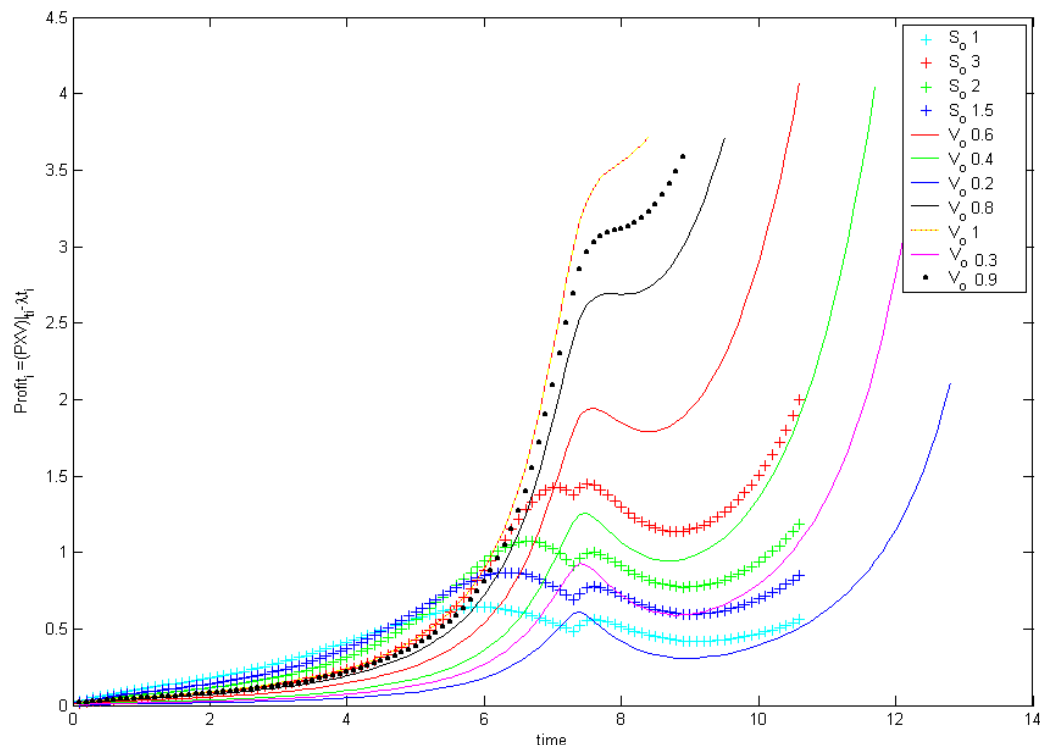


Figure 5.4 Effect of initial conditions on the productivity of a fermentation process. Solid lines: initial state  $(0.15, 5, 0.1, V_0)$  where  $V_0 = \{0.2, 0.4, 0.6, 0.8, 1\}$ . + lines: initial state  $(0.15, S_0, 0.1, 0.6)$  where  $S_0 = \{1, 1.5, 2, 3\}$

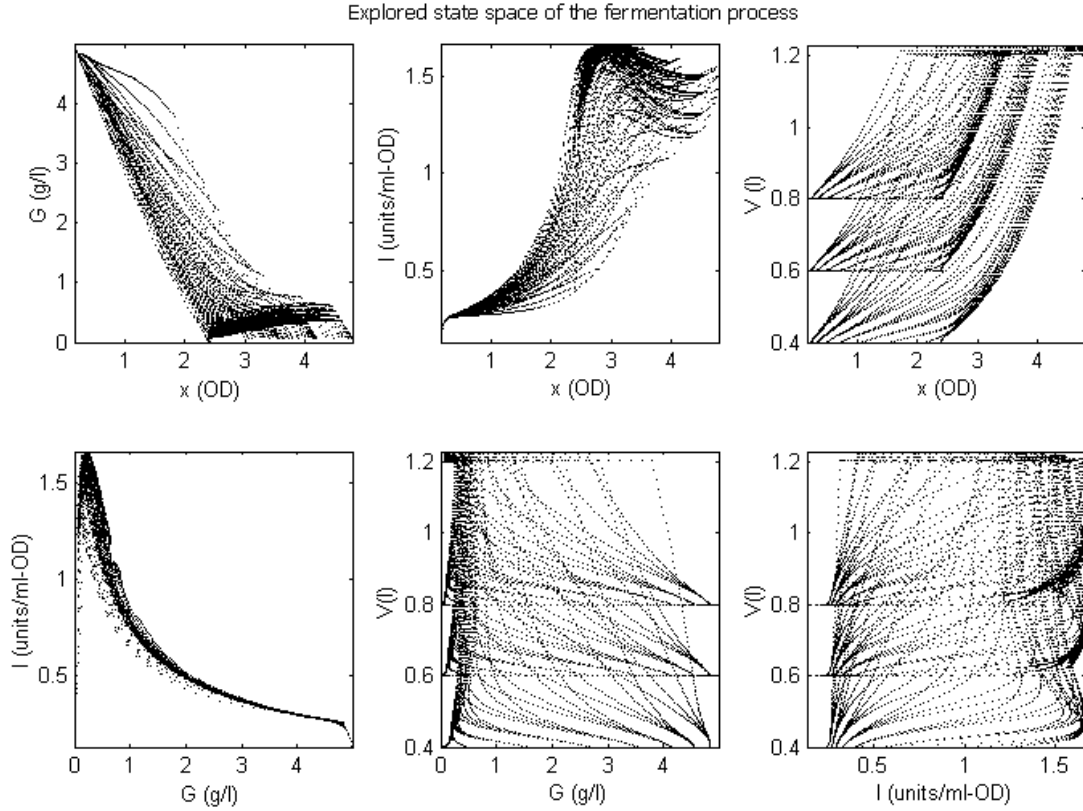


Figure 5.5 Explored state space 9328 points of the invertase fermentation process by using the suboptimal policies described by eq. 5.1

Once the state data are obtained with several suboptimal policies, a suboptimal cost-to-go should be calculated for each state point. Each policy has final an optimal final time  $t_f^*$  as is shown in Figure 5.6. With those optimum final times, for each of the visited states in a given process trajectory, a cost-to-go value can be computed according to

$$J(x_k) = \lambda \cdot (t_f^* - t_{x_k}) - I(t_f^*) \cdot OD(t_f^*) \cdot V(t_f^*) \quad (5.2)$$

where  $t_{x_k}$  is the time of the process associated to the given state  $x_k$ .

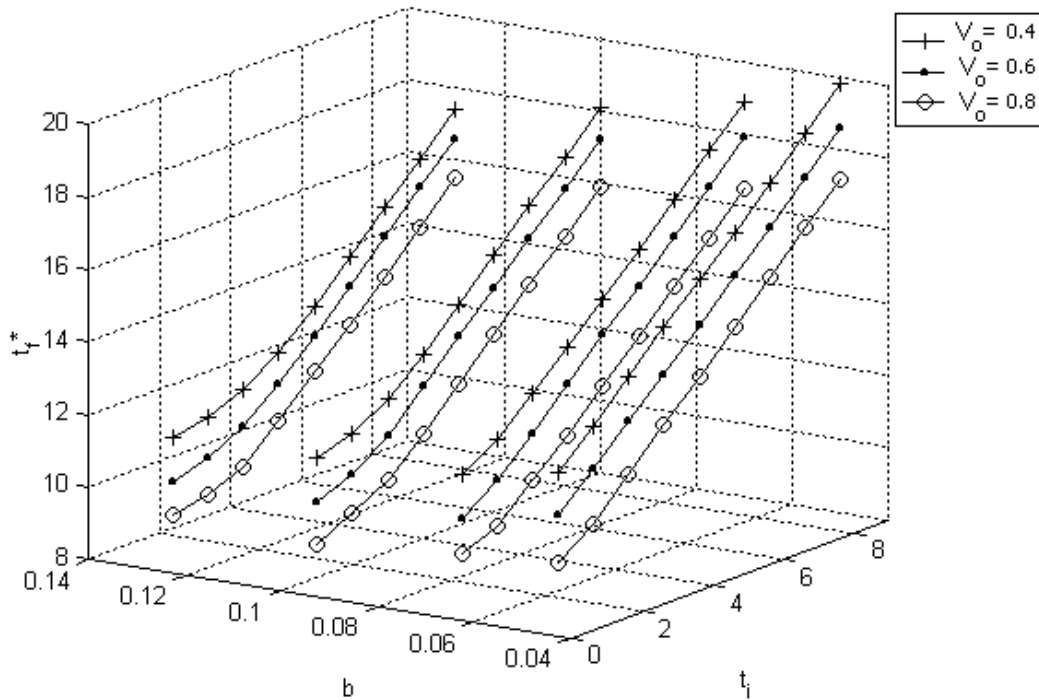


Figure 5.6 Optimum final time  $t_f^*$  of the invertase fermentation for each suboptimal policy  $u(t, t_i, b)$  and different initial volumes of the bioreactor. The final optimum fermentation time depends on the value of the initial fermentation volume  $V_0$  and on the policy parameters  $t_i$  and  $b$

The next step in NDP is to fit a function approximator to the state-cost-to-go obtained. Two neural network architectures have been chosen for this purpose. Also, a hyper cubic region of the state space of the problem is chosen. This region has its limits in the hyper planes given by the maximum and minimum values of each of the state variables. Those minimum and maximum are found in the set of all the states obtained through the use of all the suboptimal policies into the simulation of the fermentation process.

MLP have been widely used to fit non-linear functions of dynamic process (Hussein, 1999). This kind of NN is used here to fit the cost-to-go surface in the state space hyper cubic region. The MLP architecture has 4 inputs (4 process state variables) and 1 output (cost-to-go value). The training fitting criteria is the sum of squared errors (SSE). The best structure

found has 2 hidden layers, each with 17 and 5 nodes. This NN is  $\tilde{J}^0$ . A fuzzy ARTMAP neural system is also used to fit the cost-to-go surface in the state space hyper cubic region. The fuzzy ARTMAP architecture also has 4 inputs (4 process state variables) and 1 output (cost-

to-go value). This NN is called  $\tilde{J}^0$  in the application of fuzzy ARTMAP to the procedure of fitting improvement. Fuzzy ARTMAP is a powerful classifier that should perform better as a function approximator than MLP, since it will provide better input-output association over the considered state space. To build a fuzzy ARTMAP, the obtained state- cost-to-go data are preprocessed (normalization and complement coding) prior to their presentation to the fuzzy

ART<sub>A</sub> and fuzzy ART<sub>B</sub> modules during the training of the network. Fast learning is used in both fuzzy modules. The baseline of the vigilance parameter for fuzzy ART<sub>A</sub> is  $\rho_A=0.9$ . The vigilance parameter for fuzzy ART<sub>B</sub> and the map field are both set to 0.999999. The set of input-output data is presented randomly to both fuzzy ART modules. The training process evolves according to the set of fuzzy rules applied in the classification of input and output patterns in each fuzzy ART module until stability of classes is reached. In this case fuzzy ART<sub>A</sub> found 746 categories among 9328 state points and fuzzy ART<sub>B</sub> classified the corresponding cost-to-go values in 1042 categories. The fuzzy ARTMAP was tested in the same states-cost-to go data set and an average absolute prediction error of 0.0437 was found.

Given this first approximation of the cost-to-go an improvement of the surface captured by the NN is made through Bellman iteration,

$$J^{i+1}(\mathbf{x}) = \min_{u \in [0, u_{\max}]} \left[ \lambda \bullet \Delta t + \tilde{J}^i(\mathbf{f}(\mathbf{x}, u), \mathbf{r}) \right] \quad (5.3)$$

where  $u_{\max}$  is the maximum value between  $(1.2-V_k)/\Delta t$  and  $0.2722$  l/h,  $V_k$  is the actual fermentation volume, and  $\Delta t$  is the time step between states. This time step is constant and equal to 0.1 h. The transition cost from one state to the next is only a function of time. That cost is the main difference in each cost-to-go value associated to two consecutive states of one process trajectory. After each Bellman's iteration is completed a new NN should be fitted using the new value of the cost-to-go for each state. Again two neural network schemes are used in the present work to do this mapping: MLP and fuzzy ARTMAP. The following sections describe how this is done and what are the results obtained for each case. When cost-to-go approximator has been obtained and the state space- cost-to-go map is completed a NN a controller can be implemented online with equation (4.19).

In this way, the optimal policy  $\pi$  found could adapt itself in case of disturbances. Let's proceed with the results obtained when MLP or fuzzy ARTMAP are used to approximate

$$\tilde{J}(\mathbf{f}(\mathbf{x}, u), \mathbf{r}).$$

### 5.2.1 MLP dynamic programming

The fitting of cost-to-go with MLP is carried out in three iterations, as described in section 4.6.3. The termination condition of the iteration procedure described in that section is  $\varepsilon=0.202$ . The best NN architecture found for each iteration step are 4-17-5-1, 4-17-5-1 and 4-13-5-1. The converging criteria  $(\sum_{k=1}^N |J^{i+1}(x_k) - J^i(x_k)| / N)$  at each step are: 0.5927, 0.2171 and 0.2020.

The last cost-to-go approximator (the third NN) is implemented online into a controller resulting in a new feeding policy for the fermentation process described in section 2.2.2, and in new visited states. The trajectories followed by the fermentation process under control and the new visited states are plotted in Figure 5.7. Those new states and their associated cost-to-go value are included in the set of data to be fitted by a new NN (in total 9600 states). The complete set of states is shown in Figure 5.8. This Figure is similar to Figure 5.5. For these data the best NN structure found is 4-17-5-1. Only one Bellman iteration step is needed to find the new state-cost-to-go approximator. For this iteration the converging criteria is 0.194.

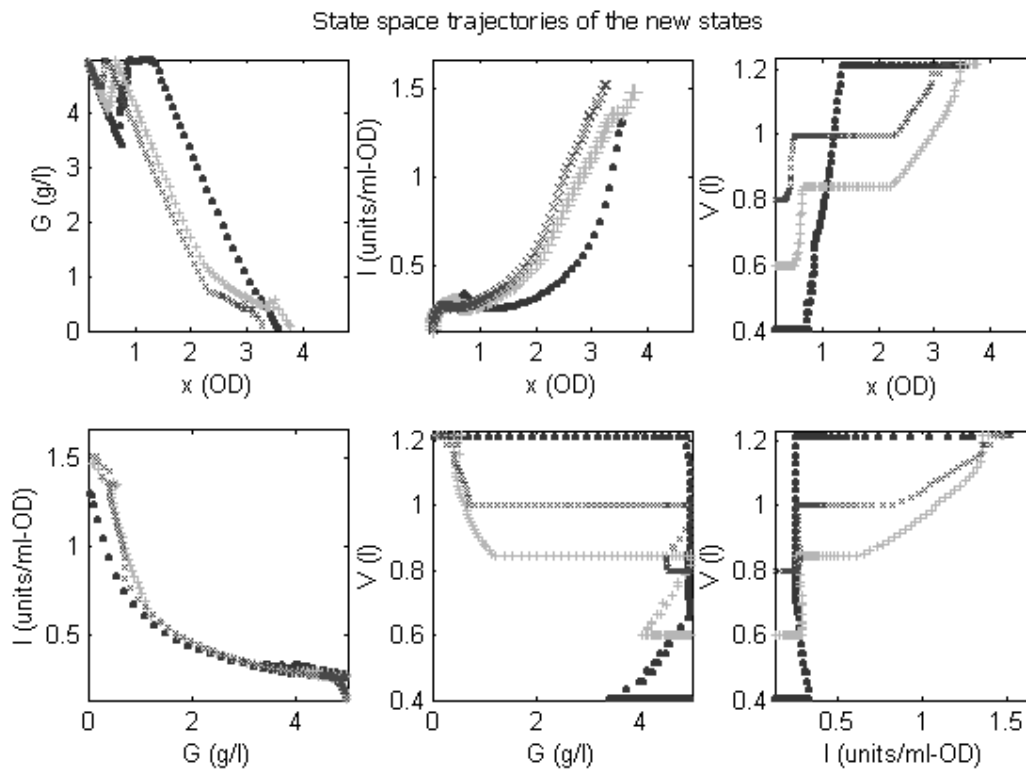


Figure 5.7. State space representation of the trajectories followed by the controlled fermentation process for initial fermentation volumes of 0.4 ( $\cdot$ ), 0.6 ( $+$ ), 0.8 ( $\times$ ). The controller is implemented with the cost-to-go approximator obtained from the second Bellman iteration. The new visited states and its associated cost-to-go are used into a new approximator.

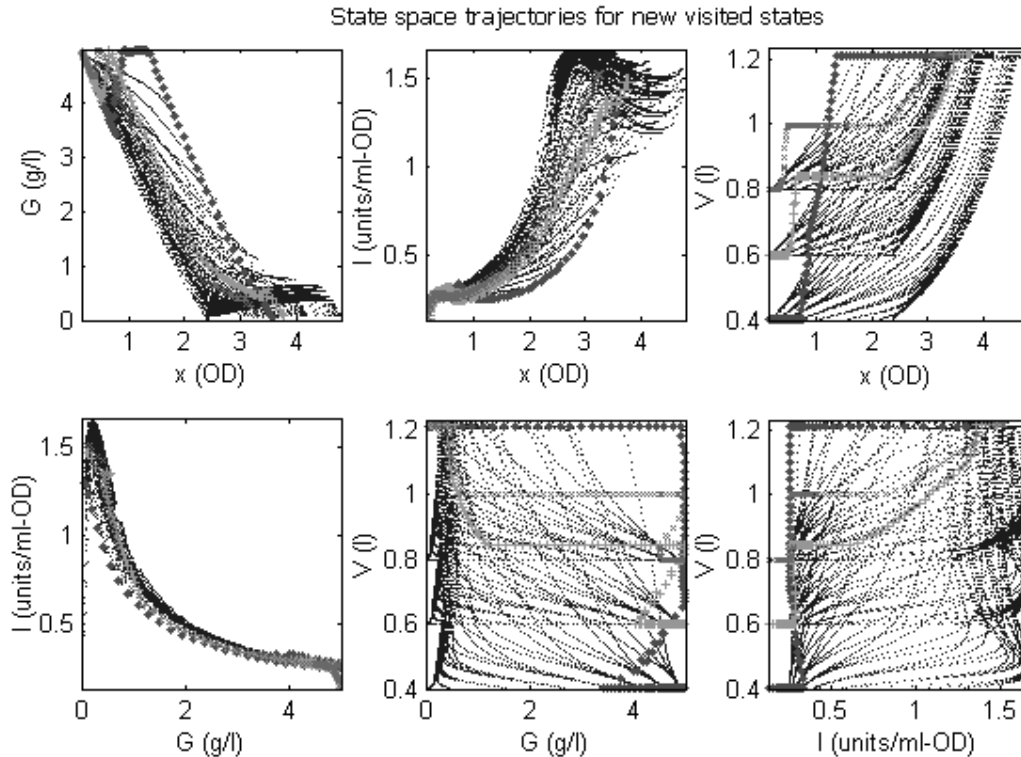


Figure 5.8 Explored state space of the invertase fermentation process after policy iteration procedure. The controlled process followed a different trajectory and visited new states.

### Optimal controller performance

Once the value iteration procedure is over, the final cost-to-go approximator (fourth NN) can be implemented online into a controller resulting in a new feeding policy for the fermentation process. This NN is implemented into a controller system with the on-line implementation of the Bellman equation. The performance of the obtained optimal controller is assessed in one known and in unknown fermentation processes (a) For known process, the fermentation starts with an initial volume of 0.6 liters which is the initial state of one of the process trajectories used in the training of the cost-to-go approximator (b) Unknown fermentation processes correspond to different initial fermentation volumes not seen before by the cost-to-go approximator.

#### (a) Known process performance

The fermentation is again simulated, starting with an initial volume of 0.6 liters. The results of the simulation are shown in Table 5.8. Those results are compared with the prediction of different optimization methods. Although invertase fermentation productivity is higher when

the Patkar et al. (1993) optimization method is used, the best profit is found when the current MLP- NDP method is employed to obtain the best feeding strategy. Fermentation time is lower in this case, making the MLP-NDP obtained policy more profitable.

**Table 5.8** Invertase production optimization results for an initial fermentation volume of 0.6 liters. The profit depends on the productivity and the total fermentation time

POLICY GIVEN BY	PROFIT	PRODUCTIVITY	FINAL TIME (H)
Patkar et al. 1993	3.70	7.30	12
Chaudhuri and Modak 1998	3.50	7.10	12
Best suboptimal policy	3.72	7.23	11.7
MLP-NDP	3.80	7.25	11.5

The optimal policy determined for an initial volume of 0.6 l is shown in Figure 5.9. The figure also shows the optimal policies attained with previous optimization procedures. It is clear that although NDP policy yields the best fermentation results, the trajectory of the manipulated variable (feed rate) is not smooth. This could be a problem for the design of actuators like valves. To overcome this problem the simplest solution would be to filter the manipulated variable signal from the controller. The control action signal is studied through a Fourier analysis in order to find its characteristic frequency. Once knowing that frequency a suitable

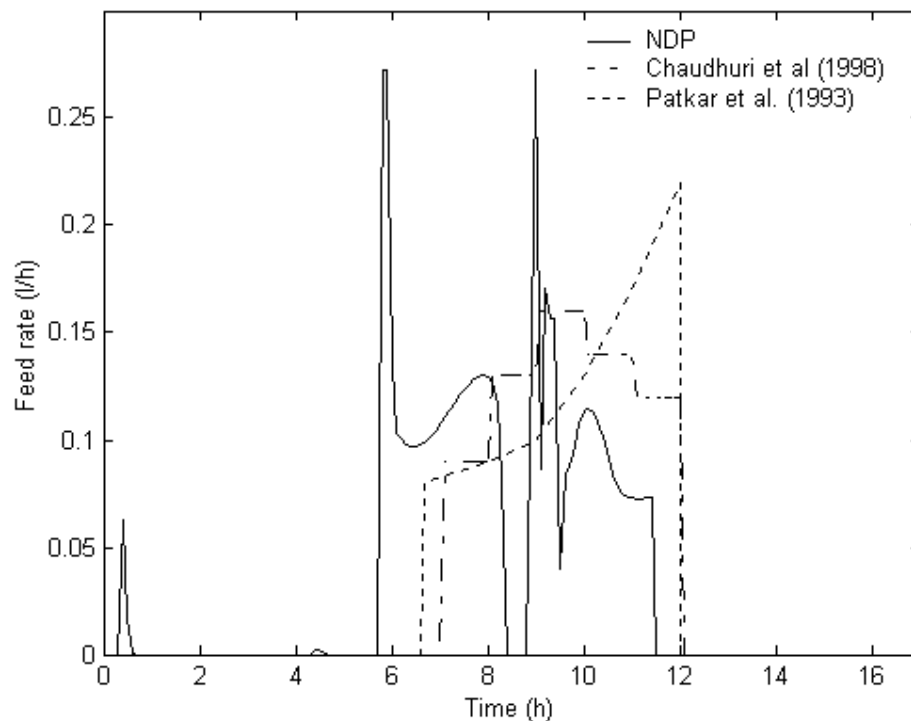


Figure 5.9 Optimal policy for a fermentation process with an initial volume of 0.6. The obtained NDP policy is compared against other optimization procedures results. (-) NDP methodology (- -) Chaudhuri et al. (1998), (- · -) Patkar et al. (1993)



filter can be designed. The results of Fourier analysis are shown in Figure 5.10. It can be seen that there is no characteristic frequency to work with. A filter must be designed to obtain a smoother control action signal. An average of the optimizer output is proposed. Thus the control action is equal to the average of five or six consecutive optimizer results. The control policy obtained is shown in Figure 5.11. The control action is smooth and suitable for practical controller implementation but, as expected, a lower profit is obtained. For an initial volume of 0.6 l the profit obtained when the average is over five consecutive times is 3.1224 and when the average is over six consecutive times is 3.1222. So, this approach should be improved. Below in this work a fuzzy ARTMAP architecture as clustering algorithm is used to perform the filtering.

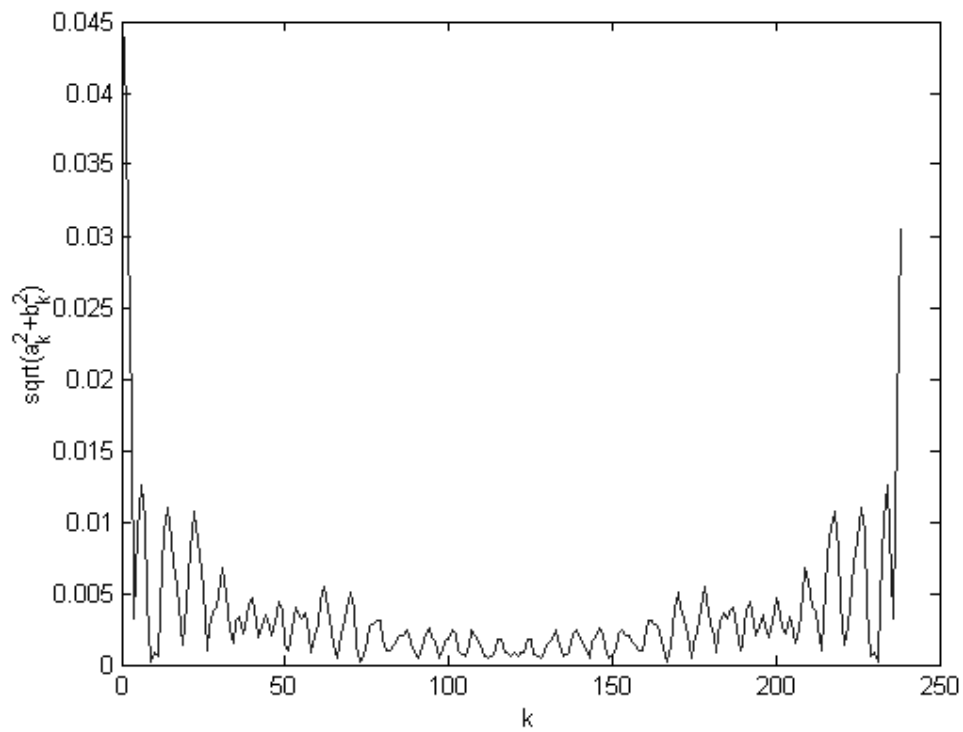


Figure 5.10 Fourier analysis of the control signal from the optimizer

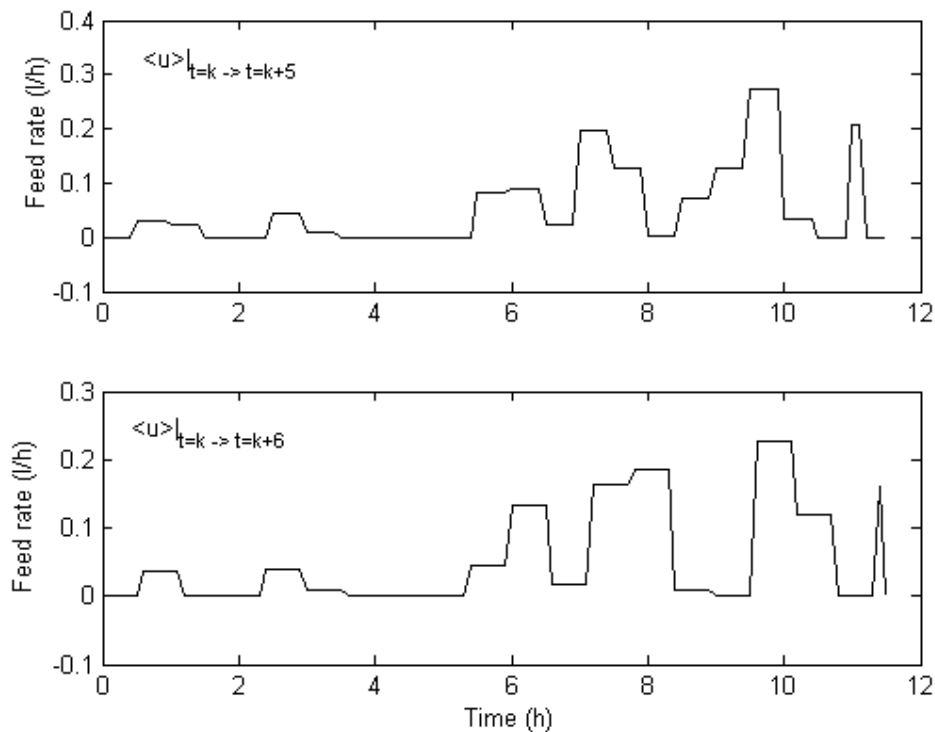


Figure 5.11 Control policy obtained when the optimizer output is filtered during a fermentation process with initial volume of 0.6 l. Upper figure. Control action average is over five consecutive times. Lower picture, when the average is over six consecutive times

#### (b) Unknown process performance

Another advantage of NDP optimization procedure over the Patkar et al. (1993), Chaudhuri and Modak (1998) methodologies and other optimization methods, is that a NDP controller can be built and used in different fermentation processes. To test the MLP-NDP controller developed here, its performance is evaluated again for different initial volumes of fermentation process. Note that for the optimization procedures reported in literature, this means solving additional optimal control problems. The resulting dynamic optimization is a non-linear programming problem, which is very demanding computationally. For the MLP-NDP controller, changing volumes only means to change the fermentation initial state since future costs are a function of the system state.

The results of these unknown fermentation simulations are summarized in Table 5.9. Note that the initial fermentation volumes are different from those used to obtain cost-to-go vs. state data, i.e., the system has to visit new state space points. Since Patkar et al. (1993) and Chaudhuri and Modak (1998) do not report data for these cases, the MLP-NDP profits are compared with those profits obtained when the best of the 36 suboptimal policies for the given fermentation process is applied. It should be noted that there is a different best policy

for each initial volume. This is the reason why the "suboptimal" policy profit for the fermentation with initial volume of 0.7 liters is higher than the profit obtained with MLP-NDP. Also the obtained profits are compared with the results obtained when Patkar et al. (1993) optimal policy is employed in each different fermentation process.

**Table 5.9** Profits obtained for unknown initial fermentation volumes  $V_0$  when the MLP-NDP controller is used. Profit results are compared against results obtained by applying the best policy for the given volume

$V_0$ (l)	Profit obtained through NDP method	Profit obtained with the best suboptimal policy for the given volume	Profit obtained when Patkar et al. (1993) policy is used
0.4	4.36	4.28	2.89
0.5	4.06	3.95	3.74
0.7	3.52	3.58	3.58
0.8	3.10	3.33	3.42

Figure 5.12 shows the optimal policies obtained for both initial volumes. Note that all policies have similar shape. The state space representation of the optimal trajectories for the fermentation process are presented in Figure 5.13 for the initial volumes 0.5, 0.6 and 0.7 l.

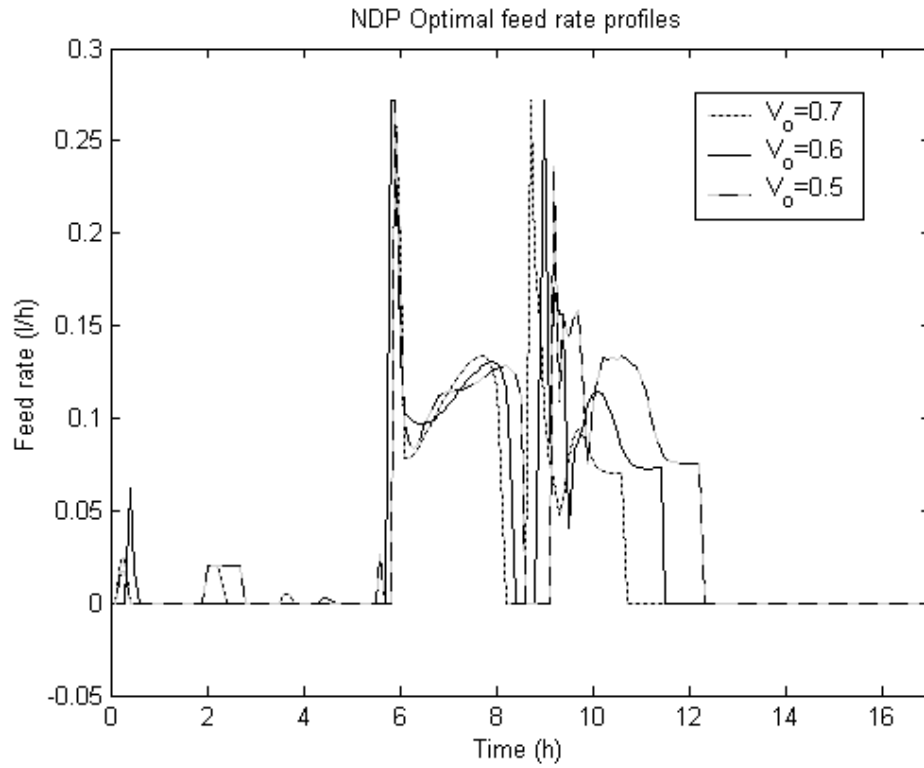


Figure 5.12. Optimal NDP policies for different fermentation processes obtained with the MLP-NDP controller

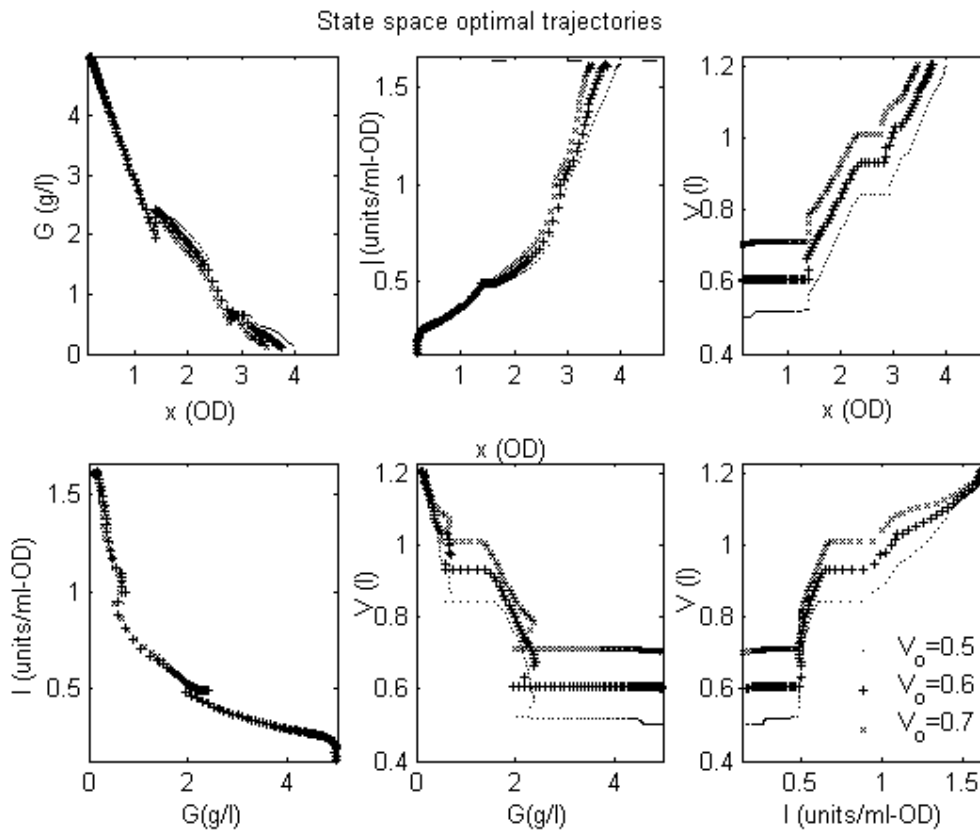


Figure 5.13 State space representation of the optimal trajectories followed by the fermentation process when MLP-NDP controller is used for different initial volumes: 0.5 ( $\cdot$ ), 0.6 ( $+$ ), 0.7 ( $\times$ ).

### 5.2.2 Fuzzy ARTMAP dynamic programming

Fuzzy ARTMAP systems are powerful classifiers and predictors (Espinosa et al. 2001, 2002; Giralt et al. 2000) and thus should be better function approximators than the MLP. Fuzzy ARTMAP fits the cost-to-go of the invertase fermentation in four iterations for a termination condition  $\varepsilon=0.2172$  of the Bellman iteration procedure. The best NN architectures found in each iteration step are summarized in Table 5.10. Fast learning is used for both fuzzy modules. The input-output data are always randomly presented.

**Table 5.10** Best Fuzzy ARTMAP structures used into Bellman's iteration for the invertase production optimization

Name	Fuzzy ART		Map field	Categories found		Average
	ART <sub>A</sub>	ART <sub>B</sub>		f-ART <sub>A</sub>	f-ART <sub>B</sub>	
	baseline vigilance parameter	vigilance parameter	vigilance parameter			absolute training error
$\tilde{J}^1$	0.9	0.999999	0.999999	536	142	0.0340
$\tilde{J}^2$	0.999	0.999999999	0.999999999	625	115	0.0292
$\tilde{J}^3$	0.99999	0.999999999	0.999999999	3396	115	0.0143

The converging criteria ( $\sum_{k=1}^{9328} |J^{i+1}(\mathbf{x}) - \tilde{J}^i(\mathbf{x})| / N$ ) of Bellman's iteration at each of the four steps are 4.1141, 0.2291, 0.2272 and 0.2172.

#### Optimal controller performance

The improved cost-to-go approximator (fourth trained fuzzy ARTMAP network) is implemented into a controller system formed by an on-line implementation of Bellman equation (equation 4.19). The performance of the obtained optimal controller is checked in 3 different cases: (a) For known process, the fermentation process starts with an initial volume of 0.6 liters, i.e., the initial state of one process trajectory used in the training of the cost-to-go approximator; (b) for unknown fermentation process different initial fermentation volumes are used as initial states not seen before by the approximator; (c) for unexpected disturbances, an abrupt death of microorganisms is imposed, causing a change in cell concentration.

## (a) Known process performance

The fermentation process described in section 2.2.2 is simulated, starting with an initial volume of 0.6 liters. The profit results are shown in Table 5.11. Also there is a comparison with the results of previous optimization methods. The best profit for this nominal initial volume is obtained with the MLP-NDP method. With Fuzzy ARTMAP-NDP the profit coincides with the one obtained with the best suboptimal policy. In both cases the NDP methodology improves previous optimization methods. As it will be seen below, although MLP-NDP controller gives better profit than fuzzy ARTMAP controller, in a real implementation of a NDP controller on a real fermentation process, it is more desirable to use fuzzy ARTMAP methodology due the characteristics of the policy obtained.

**Table 5.11** Invertase production optimization results for an initial fermentation volume of 0.6 liters.

The profit depends on the productivity and the total fermentation time

POLICY GIVEN BY	PROFIT	PRODUCTIVITY	FINAL TIME (H)
Patkar et al., 1993	3.70	7.30	12
Chaudhuri and Modak, 1998	3.50	7.10	12
Best suboptimal policy	3.72	7.23	11.7
MLP-NDP	3.80	7.25	11.5
Fuzzy ARTMAP-NDP	3.72	7.23	11.7

The optimal policy for this fermentation obtained using the fuzzy ARTMAP-NDP controller is plotted in Figure 5.14. Note that the fuzzy ARTMAP-NDP policy is smooth with a progressive increase of the feed rate of glucose. This feed rate profile is better than the one obtained using the MLP-NDP controller (See Figure 5.9), although the profit of the former is a little lower as indicated in Table 5.11. The feed rate profile obtained with fuzzy ARTMAP-NDP controller is more convenient for a real control implementation than the one obtained using the MLP-NDP controller, even though the profit of the latter is a little higher. The fuzzy ARTMAP-NDP policy in Figure 5.14 shows a trend similar to the Patkar et al. (1993) policy shown in Figure 5.9. The fuzzy ARTMAP-NDP controller has several advantages over MLP-NDP. The smooth policy of the fuzzy ARTMAP-NDP controller is desirable because:

- The actuators energy consumption and their wearing out are lower;
- In a real implementation abrupt changes in feed rate are not feasible. There is always a lag time between valve states.
- If an abrupt change in feed rate of glucose could be effected in a real fermentation process, it would result in local gradients of concentration over the entire bioreactor, which in turn would result in regions of the bioreactor with a glucose concentration

higher than the concentration needed only for cell growth. This would increase then ethanol production, and, thus, the enzyme production would be inhibited. This would cause a lower enzyme production and a lower profit than the values expected from the simulation of the same process.

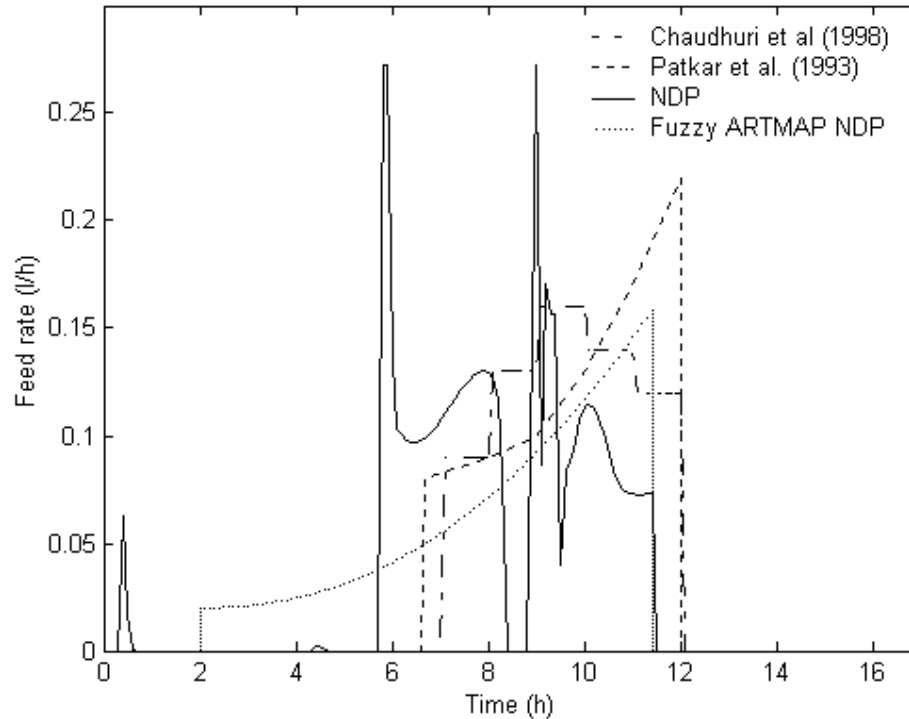


Figure 5.14 Optimal policy for invertase fermentation process with an initial volume of 0.6. The obtained fuzzy ARTMAP-NDP policy is compared against other optimization results. (...) Fuzzy ARTMAP methodology, (-) MLP-NDP methodology, (--) Chaudhuri and Modak (1998), (---) Patkar et al. (1993)

#### (b) Unknown process performance

The fuzzy ARTMAP-NDP controller was also tested also for different fermentation process with different initial volumes. The results of those simulations are shown in Table 5.12. Note again that the initial fermentation volumes 0.5 and 0.7 l are different from those used to obtain cost-to-go vs. state data. Comparing the profits obtained with the results obtained using the MLP-NDP controller (Table 5.9) it can be said that the fuzzy ARTMAP-NDP controller performs slightly worse in the case of the fermentation with initial volume of 0.5 liters and slightly better in the case of the fermentation with initial volume of 0.7 liters.

**Table 5.12** Profits obtained for different initial fermentation volumes for invertase production

Vo	Profit obtained through Fuzzy ARTMAP-NDP method	Profit obtained with the best suboptimal policy for a given volume
0.4	4.28	4.28
0.5	4.01	3.95
0.7	3.58	3.58
0.8	3.33	3.33

The small difference between both the profit obtained with fuzzy ARTMAP and the best profit obtained with the best suboptimal policy for a given initial volume, resides in the classification characteristics of the fuzzy ARTMAP. The control system searches for the best trajectory based on the built in map of the cost-to-go. The prediction of a cost-to-go value for a given state is done in a way that the state is classified into a category that is linked with a given value of the cost-to-go. That link and the resulting cost-to-go value is built and obtained based on previous information of the states vs. cost-to-go regions. The fuzzy ARTMAP only performs an interpolation of the previously seen states cost-to-go data.

The trajectories followed by the fermentation process under control are plotted in Figure 5.15. The obtained optimal policies for all initial volumes are shown in Figure 5.16. It should be noted that in all cases the policy attained with the fuzzy ARTMAP controller follows a smooth increase in the feed rate of glucose. This behavior is superior to that of the MLP-NDP controller, and to any of the previous optimization procedures. As a result, and for practical purposes, the fuzzy ARTMAP-NDP optimal controller is preferred over the MLP-NDP controller, and to any of the previous optimization procedures, which would require additional optimization calculations for each different initial condition.



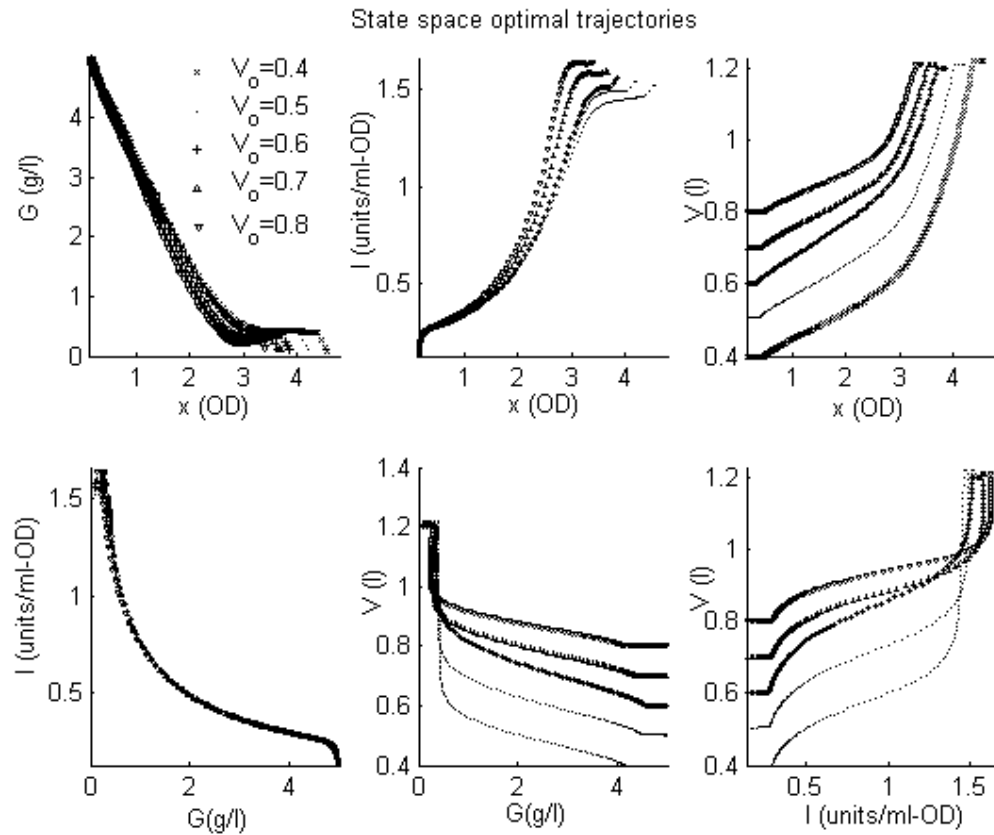


Figure 5.15 State space representation of the optimal trajectories followed by the invertase fermentation process under control with fuzzy ARTMAP-NDP, for different initial volumes: 0.4(x), 0.5 (.), 0.6 (+), 0.7 (^), 0.8 (v)

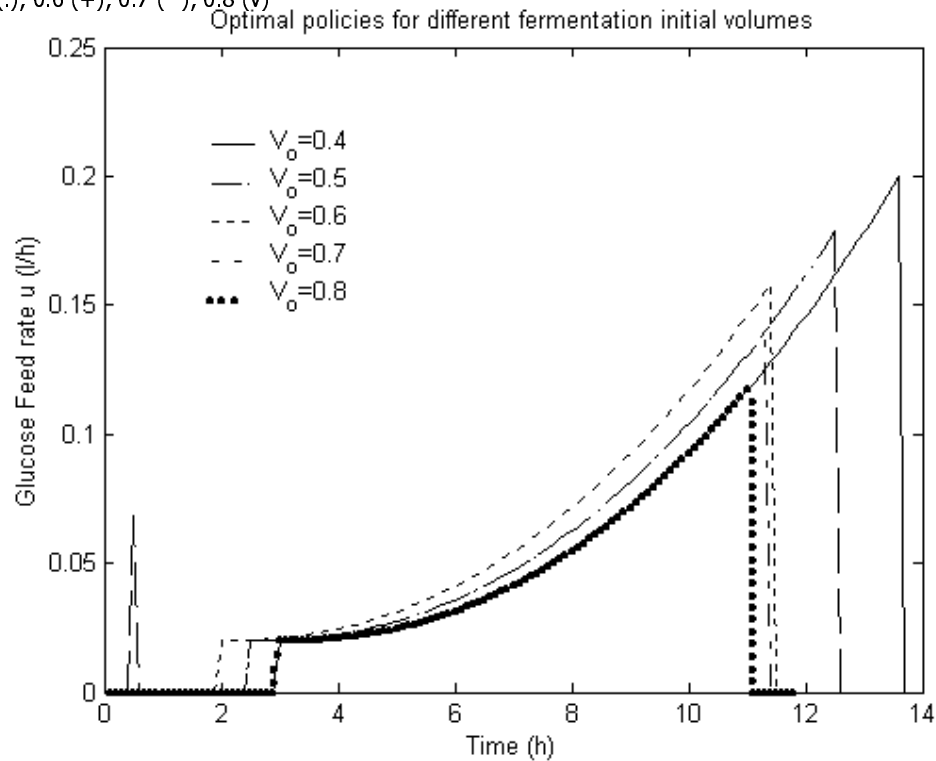


Figure 5.16 Optimal policies for different fermentation initial volumes obtained through the control of the process by the fuzzy ARTMAP-NDP based controller

## (c) Unexpected disturbance performance

To better illustrate the effectiveness of the proposed methodology, the fuzzy ARTMAP-NDP controller performance is tested against an unknown disturbance. An abrupt change in the cell concentration is imposed in the middle of the fermentation, as shown in Figure 5.17(a). The fermentation process starts with an initial volume of 0.4 liters. After nine hours of fermentation have passed, the value of cells concentration is decreased by 50%. The controller senses this change of state the optimal control action is found, and a new optimal and smooth process trajectory is followed as shown in Figure 5.17(b). Figure 5.17 (b) shows the optimal policy found by the controller in this case. The best suboptimal policy for a fermentation with initial volume of 0.4 l it is also depicted in this figure. This best suboptimal policy was found previously in a normal process operation. The profit value obtained using the best suboptimal policy for that given initial volume when this disturbance arises is 1.88, while, the profit value obtained with the fuzzy ARTMAP-NDP controller process in the case of this disturbance is 2.18. Clearly fuzzy ARTMAP-NDP controller deals successfully with unexpected disturbances. This property can be attributed to the feedback action implicitly implemented in equation (4.19); the actual process state is needed to decide the optimal control action to follow. The controller optimal trajectory in the state space is plotted in Figure 5.18. The trajectory followed by the process in this fermentation is completely different to the process trajectories used as starting guess for the NDP procedure (See Figure 5.5)

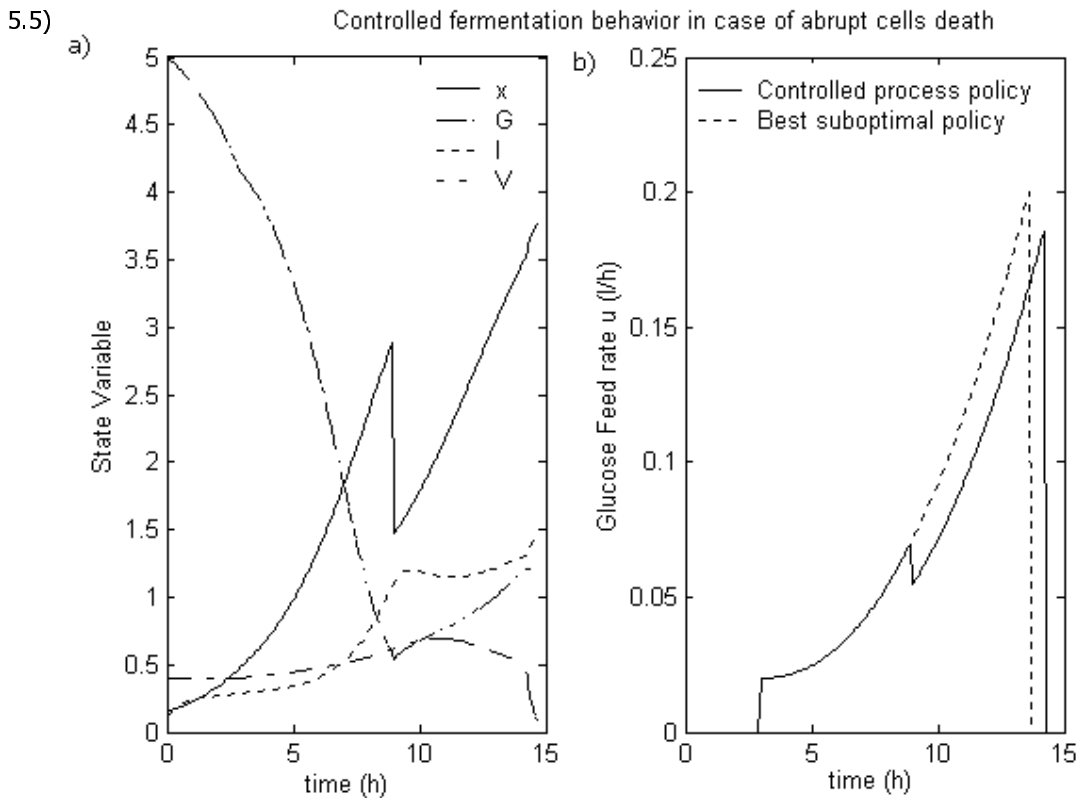


Figure 5.17 Invertase fermentation process behavior when an abrupt death of microorganisms occurs at  $t = 9$ h. The evolution of the controlled process is shown in figure 5.14(a). The control variable, glucose feed rate is shown in figure 5.14(b). The optimal policy is compared against the best suboptimal policy for that given initial volume (0.4 l)

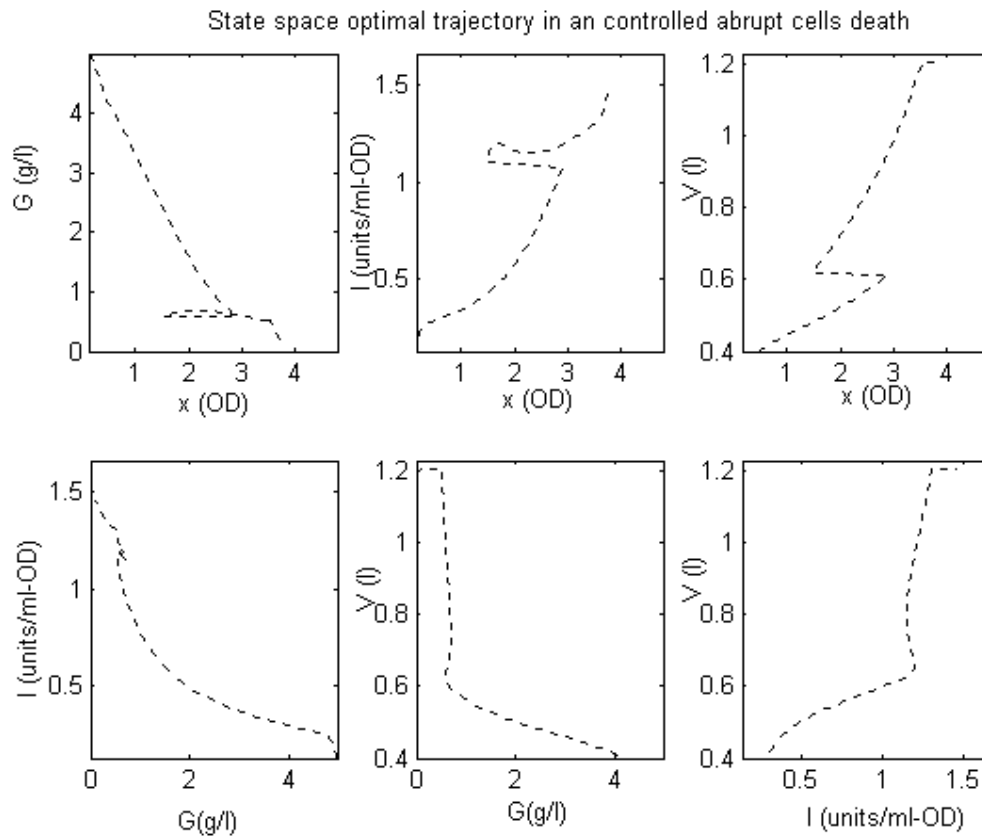


Figure 5.18 Optimal trajectory of a controlled abrupt microorganisms death. The invertase fermentation process has an initial volume of 0.4 l. When fermentation time equals 9 hours, cells concentration decreases 50%. The controller senses this new bioreactor state and finds both the optimal control action and the new optimal control policy



## Chapter 6: Conclusions and work perspectives

A non-linear multivariable (MIMO) process control model based on multi layer perceptrons (MLP) and radial basis functions (RBF) neural networks has been developed for a fed-batch bioreactor. It has been found that previous knowledge of the operation process, an adequate implementation of neural network architectures to establish input-output relationships, and the processing of information and how it is presented to the network are critical issues to develop reliable direct and inverse neural models for control purposes.

Two fermentation processes were studied in detail. A non-linear multivariable bioreactor control problem was used as a case study for model based control techniques. The control model of this process was considered as MIMO system with 6 inputs and 6 outputs. The inputs to the control system were the five flow rates of substrate, air, thermal fluid, acid and base concentrations, and the stirring RPM of the mixer. The six outputs were the vessel bulk concentrations of cells, substrate and products, the temperature, pH and dissolved oxygen concentration.

The best performance of the direct model for steady state process operations as well as for positive step perturbations in the manipulated variable was obtained with the RBF architecture with goal 3.0. If the steps in the manipulated variable are random, the RBF architecture with goal 2.75 performed slightly better. When an unexpected pH disturbance occurred, the best neural model was the MLP 19-5-6 both for the steady state process operation and for a positive step in the manipulated variable. The RBF architecture with goal 3.0 was the best ranked for any operation mode in the current control problem based on a global engineering criterion for performance. The best inverse model was a MLP 19-11-7-1 with training including past information of the steady states of the process. This inverse model improved the performance of previous schemes proposed in the literature when tested for random step changes in the set point.

The other fermentation process studied was the cloned invertase production in *Saccharomyces cerevisiae* yeast. Optimal control techniques that employ neural networks were used to optimize the production of invertase in a fed-batch mode of operation. This process has four state variables, namely cells, glucose and invertase concentrations and the fermentation volume. In this second fermentation process, the glucose feed rate is used as a control variable. The main constraint of the optimization process involved is the fixed bioreactor volume. NDP was used to obtain different optimal feeding policies for different initial conditions of invertase fermentation. The proposed optimization procedure includes prior knowledge of a suboptimal feeding strategy, which was used as a starting guess to the

simulation. On the visited state space of the fermentation a MLP neural network is used as state-cost-to-go approximator. Bellman equation is used to improve the cost surface. This procedure was compared against other optimization methods used for the same fermentation process. Also, it was implemented inside a feedback controller. The improved cost-to-go function approximator was used on-line to choose the optimal control action. This procedure has the advantage of been able to find different optimal policies for different fermentation processes, compared to literature methodologies that have to solve the optimization problem for each initial condition. The results attained with MLP-NDP controller improve previous literature optimization methodologies. A fuzzy ARTMAP-NDP system was also examined and tested for the optimization and control of cloned invertase expression in *Saccharomyces cerevisiae* yeast. The control system was built using a dynamic programming approach. Cost-to-go for a given state was found using fuzzy ARTMAP neural network modified to effect prediction capabilities. This control system also can perform an on-line optimization of the process and take into account possible process disturbances. It was found that the cost-to-go approximator is crucial for the controller performance. The ability of a fuzzy ARTMAP system to perform incremental classification is a key issue to attain a consistently smooth feed rate manipulation. The output variable, feed rate profile, has to be manipulated in a smooth way to minimize energy loses in the actuator and to implement a stable control action. It was shown that fuzzy ARTMAP-NDP controller is robust to an unknown disturbance, when 50% of cells died suddenly. The proposed fuzzy ARTMAP-NDP control methodology outperforms MLP-NDP and other optimization methods reported previously in the literature for the same fermentation process.

Some of the most relevant neural network algorithms used in process control have been discussed. Also several controller schemes have been reviewed. The flexibility of the neural networks-based controllers and their success suggest that they should be applicable to a wide range of process control problems. Further work could be focused on the implementation of control schemes to the multivariable fed-batch bioreactor process. IMC and MPC can be easily implemented with the neural network-based control models obtained in the present work. The hybrid controller could also be implemented with the use of a RBF or SOM models for process operating at steady state. Fuzzy neural networks could be used to build an inverse model applicable to a wider range of fermentation operation conditions and Kohonen maps could be applied to switch between different intelligent controllers and to fault diagnosis of the bioreactor process.

On the other hand, NDP has proved to be an appropriate optimization method. For processes such as the production of the penicillin, NDP methodology similar to the one applied for the invertase production could be implemented. Note that the optimization algorithm employed to solve Bellman's equation, was sensitive to initial values of the glucose feed rate due to the

---

high non-linear characteristics of the fermentation problem. To overcome this difficulty a cost-to-go function that depends on both system states and control variable could be considered.

Further work should include a better representation of the cost-to-go approximator that is not limited to the explored state space region, in the present case a hypercube.

An additional aspect to study is the robustness of the obtained controller due to uncertainties in the bioreactor model parameters. This is a key issue in the actual implementation of the described control strategy.

Finally the experimental implementation of all controller schemes and procedures proposed here could provide the necessary information to improve current control systems or to propose new and better algorithms.





## 7. Bibliography

Agrawal P., Lee C., Lim H.C., Ramkrishna D., Theoretical Investigations of Dynamic Behavior of Isothermal Continuous Stirred Tank Biological Reactors, Chemical Engineering Science, Vol 37 Num 3 p. 453-462 1982

Aoyama A. Doyle III F.J., Venkatasubramanian V., Control-affine neural network approach for non-minimum phase non-linear process control, J. Process Control Vol 6 Num 1 p 17-26 1996

Aoyama A., Doyle III F.J., Venkatasubramanian V., A fuzzy neural network approach for non-linear process control, Engineering Applications of Artificial Intelligence, Vol 8 Num 5 p.483-498 1995a

Aoyama A., Doyle III F.J., Venkatasubramanian V., Control-affine fuzzy neural network approach for non-linear process control, Journal of Process Control Vol 5 Num 6 p. 375-386 1995b

Aoyama A., Doyle III F.J., Venkatasubramanian V., Fuzzy neural network systems techniques and their applications to non-linear chemical process control systems, chapter 18 of Fuzzy theory systems: techniques and applications Vol 2 Edited by Leonides C.T., Academic Press ISBN 0-12-443872-5 1999

Aoyama, A., & Venkatasubramanian, V. Internal Model Control Framework Using Neural Networks for the Modeling and Control of a Bioreactor. Eng. Appl. Artif. Intell, 8, 689-701 1995c

Atkinson B., Marituaña F., Biochemical Engineering and Biotechnology Handbook, 2<sup>nd</sup> edition, Mexico 1991

Banga J.R., Alonso A.A., Singh R.P., Stochastic dynamic optimization of batch and semicontinuous bioprocesses, Biotechnology Progress Vol 13 p. 326-335 1997

Bertsekas D.P., Tsitsiklis J.N., Parallel and distributed computation. Numerical methods. Prentice Hall Englewood cliffs, NJ ISBN 0-13-648700-9 1989

Bertsekas D.P., Tsitsiklis J.N., Neuro Dynamic Programming. Athena Scientific. Belmont MA 1996

Bertsekas D. P., J. N. Tsitsiklis, C. Wu, Rollout Algorithms for Combinatorial Optimization, Journal of Heuristics, Vol 3, p. 245-262, 1997 \*

Bhagat P. An introduction to neural nets, Chemical Engineering Progress p.55-60 August 1990

Bhartiya S., Whiteley J.R., Factorized approach to non-linear MPC using a Radial Basis Function model, AIChE journal Vol 47 Num 2 p. 358-368 February 2001

Biegler, L., Grossmann, I.E., Sirolo, J., & Westerberg, A. Systematic Methods of Chemical Process Design, Prentice Hall. 1997

Bishop C.M., Neural networks for pattern recognition, Clarendon press, Oxford 1995.

Bissessur Y., Martin E.B., Morris A.J., Kitson P., Fault detection in hot steel rolling using neural networks and multivariable statistics, IEE Proceedings-Control Theory Applications Vol 147 Num 6 November 2000

Boskovic J.D., Narendra K. S., Comparison of Linear, Non-linear and Neural-network Based Adaptive Controllers for a Class of Fed-batch Fermentation Processes, *Automatica*, Vol 31 Num 6 p. 817-840. 1995

Brown M.D., Lightbody G., Irwin G.W., Non-linear internal model control using local model networks, *IEE Proc-Control theory Appl.* Vol 144 Num 6 November 1997

Business Week, The new rocket science: welcome to the future of finance, Special Report Business Week p. 131-140 November2, 1992

Cabrera J.B.D., Narendra K.S., Issues in the application of neural networks for tracking based on inverse control, *IEEE Transactions on automatic control* Vol 44 Num 11 p. 2007- 2027 November 1999

Cameron D.C., Altaras N.E., Hoffman M.L., Shaw A.J., Metabolic engineering of propadeniol pathways, *Biotechnology progress* 14 p. 116-125 1998

Card J.P., Sniderman D.L., Klimasauskas C., Dynamic neural control for a plasma etch process, *IEEE Transactions on neural networks* Vol 8 Num 4 p. 883-901 July 1997

Carpenter G.A., Grossberg S., Pattern recognition by self-organizing neural networks, Cambridge MA:MIT Press, 1991

Carpenter G.A., Grossberg S. Markuzon N., Reynolds J.H., Rosen D., Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps, *IEEE Transactions on neural networks* Vol 3 Num 5 p. 698-713 September 1992

Carpenter G.A., Grossberg S., Reynolds J.H., A fuzzy ARTMAP nonparametric probability estimator for no stationary pattern recognition problems, *IEEE Transactions on neural networks* Vol 6 Num 6 p. 1330-1336 1995

Chiou J., Wang F., Hybrid method of evolutionary algorithms for static and dynamic optimization problems with application to a fed-batch fermentation process, *Computers and chemical engineering* 23 p. 1277-1291 1999

Chistofides P.D., Control of non-linear distributed process systems: Recent developments and challenges, *AIChE Journal*, Vol 47 Num 3 p. 514-518 March 2001

Chitra S.P., Use neural networks for problem solving, *Chemical Engineering Progress* p. 44-52 April 1993

Chaudhuri B., Modak J.M., Optimization of fed-batch bioreactor using neural network model, *Bioprocess Engineering* 19 p. 71-79 1998

Control Engineering, Merging Mom's Perceptive power with technology creates startling results, *Control Engineering* April 2001

Control Engineering, Model free adaptive control, *Control Engineering Europe* February/March 2001

Control Engineering, Push the limits, *Control Engineering* February 2001

Cruickshank S.M., Daugulis A.J., McLellan P.J., Dynamic modeling and optimal fed-batch feeding strategies for a two-phase partitioning bioreactor, *Biotechnology and bioengineering* Vol 67 Num 2 p.224-233 January 2000

Cuthrell J.E., Biegler L.T., Simultaneous optimization and solution methods for batch reactor control profiles, *Computers and Chemical Engineering* Vol 12 Num 1/2 p. 49-62.1989

Cybenko G. Continuous valued neural networks with two hidden layers are sufficient. Technical Report, Department of Computer Science, Tufts University, Medford, MA, 1988 \*

Cybenko, G. Approximation by Superposition of Sigmoidal Function. *Mathematics of Control Signals & Systems*, 2, 303-314. 1989 \*

Dash S., Venkatasubramanian V., Challenges in the industrial applications of fault diagnosis systems, Laboratory for intelligent process systems, school of chemical engineering, Purdue University. [venkat@ecn.purdue.edu](mailto:venkat@ecn.purdue.edu) 2000

Demartines P., Herault J., Curvilinear Component Analysis: A self-organizing neural network for non-linear mapping of data sets, *IEEE Transactions on neural networks* Vol 8 Num 1 p.148-154 January 1997

Demuth H., Beale M., Neural network toolbox for use with MATLAB©. User's guide Version 3. The MathWorks Inc. July 1998

Dhir S., Morrow K.J., Russell Rhinehart R., Dynamic optimization of hybridoma growth in a fed-batch bioreactor, *Biotechnology and bioengineering* Vol67 Num 2 p.197-205 January 2000

Economou C.G., Morari M., Palsson B.O., Internal model control 5. Extension to non-linear systems, *Ind. Eng. Chem. Process Des. Dev.* 25, p. 403-411 1986

Efe M.O., Abadoglu E., Analysis and Design of a Neural Network Assisted Non-linear Controller for a Bioreactor, a [F. Giralt!!! Bogazini University Turkey!!](#)

Efe M.O., Kaynak O., Abadoglu E., Neural Network Assisted Non-linear Controller for a Bioreactor, b [F. Giralt!!! Bogazini University Turkey!!](#)

Efe M.O., Kaynak O., Identification and Control of a Non-linear Bioreactor Plant Using Classical and Dynamical Neural Networks c [F. Giralt!!! Bogazini University Turkey!!](#)

Elsimary H. Mashali S., Shahhen S., A method for training feed forward neural network to be fault tolerant, *IEEE* 0-7803-1363-1/93 1993

Espinosa, G., Yaffe, D., Cohen, Y., Arenas, A., & Giralt, F., A Fuzzy ARTMAP Based Quantitative Structure-Property Relationships (QSPRs) for Predicting Physical Properties of Organic Compounds, *Industrial and Chemical Engineering Research* Num 40 p.2757 – 2766 2001\*

Espinosa, G., Arenas, A., and F. Giralt, An integrated SOM-fuzzy ARTMAP neural system for the evaluation of toxicity, *Journal of Chemical Information and Computer Sciences*, Vol 42 Num 2 p. 343-359 2002

Feuring T. Fuzzy neural networks are overlapping, *IEEE International conference on Fuzzy Systems* p. 1154-1150 1996a

Feuring T. Learning in fuzzy neural networks, *IEEE International conference on neural networks* p. 1061-1066 1996b

Feuring T., Buckley J.J., Lippe W., Tenhagen A., Stability analysis of neural net controllers using fuzzy neural networks, *Fuzzy sets and systems* Num 101 p. 303-313 1999

Garcia C.E., Morari M., Internal model control 1. A unifying review and some new results, Ind. Eng. Chem. Process Des. Dev., 21 p.308-323 1982

\*Gawthrop P.J., Jones R.W., Sbarbaro D.G., Emulator-based control and internal model control: complementary approaches to robust control design, Submitted to Automatica, 15 May 1995 [p.gawthrop@eng.gla.ac.uk](mailto:p.gawthrop@eng.gla.ac.uk)

Giralt, F., Arenas, A., Ferre-Gine, J., & Rallo, R., The Simulation and Interpretation of Turbulence with a Cognitive Neural System. Physics of Fluids, Vol 12 Num 7 p. 1826 - 1835. 2000\*

Glasse J., Montague G.A., Ward A.C., Kara B.V., Enhanced Supervision of Recombinant E. Coli Fermentations via Artificial Neural Networks, Vol 29 p.387-398. 1994

Hartig F., Keil F.J., Luus R., Comparison of optimization methods for fed-batch reactor. Hung. J. Ind. Chem. 23 p. 141-148 1995hh

Hellstrom B., Brinsley J., Characterization of network responses to known, unknown, and ambiguous inputs, 0-7803-0928-6/93 IEEE 1993

Hertz, J.; Krogh, A., Palmer, R.G. Introduction to the theory of neural computation. Vol. 1. Santa Fe Institute. 7th edition Addison & Wesley. Redwood City CA. 1993

Himmelblau D.M., Karjala T.W., Rectification of data in a dynamic process using artificial neural networks, Computers and Chemical Engineering Vol 20 Num 6/7 p 805-812 1996

Hinchliffe M., Willis M., Dynamic modeling using genetic programming, Proceedings of the 15<sup>th</sup> IFAC World Congress, Barcelona Spain. July 21-26, 2002

Hinton G.E., Sejnowski T.J., Ackley D.H., Boltzmann machines: Constraint satisfaction networks that learn, Tech Rep. CMU-CS-84-119 Carnegie-Mellon University Dept. Computer Science, 1984

Hinton G.E., Sejnowski T.J., Learning and relearning in Boltzmann machines, Chapter 7 of Parallel and distributed processing Vol 1 Edited by Rumelhart D.E., McClelland J.L. Cambridge MA MIT Press 1986

Hinton G.E., How neural networks learn from experience, Scientific American p. 145-151 September 1992

Hornik, K., Stinchcombe, M., & White, H. Universal Approximation of an Unknown Mapping and its Derivatives Using Multilayer Feedforward Networks. Neural Networks, 3, 551-560 1989 \*

Hu Q. Rangaiah G.P., Adaptive internal model control of non-linear processes, Chemical Engineering Science 54 p. 1205-1220 1999a

Hu Q. Rangaiah G.P., Strategies for enhancing non-linear internal model control of pH processes, Journal of chemical engineering of Japan, Vol 32 Num 1 p. 59-68 1999b

Huang S., Huang Y., Bounds on the number of hidden neurons in multilayer perceptrons, IEEE Transactions on neural networks Vol 2 Num 1 January 1991

Hunt K.J., Sbarbaro D., Zbikowski R., Gawthrop P.J., Neural networks for control systems-a survey, Automatica Vol 28 Num 6 p.1083-1112 1992

Hunt K.J., Sbarbaro D., Adaptive filtering and neural networks for realization of internal model control, Intelligent Systems Engineering, Summer 1993

Hussain M. A., Allwright J.C., Kershenbaum L.S., Adaptive Feedback Linearising Control with Linearised Models and Neural Networks, Instrn. Chem. Eng. ICHEME - Advances in Process Control 4, p 195 - 202 1995 a

Hussain M.A., Kershenbaum L.S., Allwright J.C., Nonlinear control with linearised models and neural networks, Artificial neural networks, 26-28 June 1995 Conference Publication N 409 IEE 1995b

Hussain M.A., Allwright J.C., Kershenbaum L.S., Discrete time analysis of internal model control strategies based on neural networks, The 1996 IchemE Research event/ Second European conference for young researchers. 1996

Hussain M.A., Review of the applications of neural networks in chemical process control-simulation and online implementation, Artificial intelligence in Engineering Vol 13 p. 55-68 1999

Hussain, M. A., Kershenbaum, L.S. Implementation of an Inverse-Model-Based Control Strategy Using Neural Networks on Partially Simulated Exothermic Reactor. Chemical Engineering Research & Design Trans. of IChemE, 78, 299-311 2000

Iyer M.S., Wiesner T.F., Russell Rhinehart R., Dynamic reoptimization of a fed-batch fermentor, Biotechnology and bioengineering Vol 63 Num 1 p. 9-21 April 1999

Jagannathan S., Lewis F.L., Discrete-time net controller for a class of nonlinear dynamical system, IEEE Transactions on automatic control Vol 41 Num 11 p.1693-1700 November 1996a

Jagannathan S., Lewis F.L., Identification of non-linear dynamical systems using multilayered neural networks, Automatica Vol 32 Num 12 p. 1707-1712 1996b

Jayaraman V.K., Kulkarni B.D., Gupta K., Rajesh J., Kusumaker H.S., Dynamic optimization of fed-batch bioreactors using the ant algorithm, Biotechnology progress 17 p. 81-88 2001

Kamwa I., Grondin R., Sood V. K., Gagnon C., Nguyen V.T., Mereb J., Recurrent neural networks for phasor detection and adaptive identification in power system control and protection, IEEE Transactions on instrumentation and measurement Vol 45 Num 2 p. 657-663 April 1996

Karim M.N., Riviera S.L., Artificial Neural Networks in Bioprocess State Estimation, Advances in Biochemical Engineering, Vol 46 p. 1-33. 1992

Kavury S.N., Venkatasubramanian V., Representing bounded fault classes using neural networks with ellipsoidal activation functions, Computers and Chemical Engineering Vol 17 Num 2 p. 139-163 February 1993

Kawaji S., Hybrid soft computing approaches to identification of non-linear systems, Proceedings of the 15<sup>th</sup> IFAC World Congress, Barcelona Spain. July 21-26, 2002

Keyvan S., Durg A., Evaluation of the performance of various artificial neural networks to the signal fault diagnosis in nuclear reactor systems, 0-7803-2566-4/96 IEEE 1996

Kohonen T. The self-organizing map, Proceedings of the IEEE Vol 78 Num 9 September 1990

- Konda V. R., Tsitsiklis J. N., Actor Critic Algorithms, submitted to the SIAM Journal on Control and Optimization, February 2001 \*
- Konda V. R., Tsitsiklis J. N., Linear stochastic approximation driven by slowly varying Markov chains, submitted to Systems and Control Letters, June 2002 \*
- Kuhlmann Ch., Bogle I.D.L., Chalabi Z.S., Robust operation of fed-batch fermenters, Bioprocess Engineering 19 p. 53-59 1998
- le Cun Y., A theoretical frame-work for back-propagation, Proc. 1988 Connectionist models summer school, Touretzky D., Hinton G., Sejnowski T. Eds. June 17-26 p. 21-28 San Mateo CA, Morgan Kaufmann 1988
- Lee J., Lee S.Y., Park S., Middelberg A. P.J., Control of Fed-batch Fermentations, Biotechnology Advances 17 p. 29-48. 1999
- Lee J.H., Pan Y., Sung S., A numerical projection based approach to non-linear model reduction and identification, Proceedings of the 1999 American Control Conference, 3, p. 1568-1572 1999
- Lee J.M, Lee J.H., Neuro-dynamic programming method for MPC, School of Chemical Engineering, Georgia Institute of Technology, Atlanta GA 30332 USA, 2001
- Li R.F., Wang X.Z., Qualitative/quantitative simulation of process temporal behavior using clustered fuzzy diagraphs, AIChE journal, Vol 47 Num 4 p. 906-920 April 2001
- Lightbody G., Irwin G.W., Nonlinear control structures based on embedded neural system models, IEEE Transactions on neural networks Vol 8 Num 3 p. 553-567 May 1997
- Lim H.C, Tayeb Y.L., Modak J.M., Bonte P., Computational algorithms for optimal feed rates for a class of fed-batch fermentation: Numerical results for penicillin and cell mass production, Biotechnology and bioengineering Vol 28 p. 1408-1420.1986
- Lin J., Jang S., Nonlinear dynamic artificial neural network modeling using an information theory based experimental design approach, Ind. Eng. Chem. Res. 37 p. 3640-3651 1998
- Linko P., Zhu Y., Neural Network Modeling for Real-time Variable Estimation and Prediction in the Control of Glucoamylase Fermentation, Process Biochemistry, Vol 27 p. 275-283. 1992
- Lippmann R.P., An introduction to computing with neural nets, IEEE ASSP magazine p.4-22 April 1987
- Ljung L., System identification in a noise free environment, IFAC Adaptive systems in control and signal processing, Identification, Glasgow UK 1989
- Ljung, L. System Identification: Theory for the User, Prentice-Hall, Englewood Cliffs, NJ. 1987
- Loquasto III F., Seborg D.E., Monitoring model predictive control systems: a novel neural network approach, AIChE 2001 Annual meeting Reno NV, 2001
- Luo R., Misra M., Himmelblau D.M., Sensor fault detection via multiscale analysis and dynamic PCA, Ind.Eng. Chem. Res., 38 p.1489-1495 1999
- Luus R., Optimization of fed-batch fermentors by iterative dynamic programming, Biotechnology and Bioengineering Vol 41 p. 599-602 1993a

- Luus R. Application of dynamic programming to differential-algebraic process systems, Computers and chemical engineering Vol 17 Num 4 p. 373-377 1993b
- Magni A., Kershenbaum L.S., The use of committees of neural networks in process identification and control, Technical report Center for process systems engineering, Imperial College London, England. [L.kershenbaum@ic.ac.uk](mailto:L.kershenbaum@ic.ac.uk) 2000
- Magoulas G.D., Vrahatis M. N., Androulakis G., Effective Backpropagation training with variable stepsize, Neural Networks Vol 16 Num 1 p. 69-82 1997
- Marbach P., Mihatsch O., Tsitsiklis, J. N. Call Admission Control and Routing in Integrated Service Networks Using Neuro dynamic Programming, IEEE Journal on Selected Areas in Communications, Vol 18, Num 2, p. 197-208, February 2000
- Marbach P., Tsitsiklis J.N., Simulation-based optimization of Markov reward processes, IEEE Transactions on Automatic Control, Vol 46, Num 2, p. 191-209, February 2001 \*
- Marbach P., Tsitsiklis J.N., Approximate gradient methods in policy-space optimization of Markov reward process, Journal of Discrete Event Dynamical Systems, April 2002. (preliminary version: "Simulation-based optimization of Markov reward processes: implementation issues," in Proceedings of the 38th IEEE Conference on Decision and Control, December 1999, pp. 1769-1774.), 2002 \*
- Marino-Galarraga Maria, McAvoy T.J., Marlin T.E., Short-cut operability analysis. Part III-short-cut methodology for the assessment of process control designs, Research report chemical process systems engineering laboratory, University of Maryland, SRC-TR-87-66. 1987
- Mirea L., Marcu T., System identification using functional link neural networks with dynamic structure, Proceedings of the 15<sup>th</sup> IFAC World Congress, Barcelona Spain. July 21-26, 2002
- Modak J.M., Lim H.C., Feedback optimization of fed-batch fermentation, Biotechnology and bioengineering Vol XXX p. 528-540 1987
- Moon Y.B., Divers C.K., Kim H. AEWS: an integrated knowledge-based system with neural networks for reliability prediction, Computers in industry Vol 35 p.101-108 1998
- Morari M., Lee J.H., Model predictive control: past, present and future, Computers and chemical engineering, 23 p. 667-682 1999
- Moreno J.M., Madrenas J., Cabestany J., Commercial Coin Recognizers using neural and fuzzy techniques, chapter 3 of Practical Applications of computational intelligence techniques Edited by Jain L. and De Wilde P., Kluwer Academic Publishers 2001
- Mukhopadhyay S., Narendra K.S., Disturbance rejection in nonlinear systems using neural networks, Proceedings of the 31<sup>st</sup> conference on decision and control Tucson Arizona p. 2696-2701 December 1992
- Mukhopadhyay S., Narendra K.S., Two problems in decentralized adaptive control using neural networks, Proceedings of the 1999 IEEE International symposium on intelligent control/intelligent systems and semiotics, Cambridge MA p. 167-172 September 15-17 1999
- Nahas, E.P, Henson, M.A., & Seborg, D.E. Non-linear Internal Model Control Strategy for Neural Network Models. Computers Chem. Eng., 16, 1039-1057 1992
- Narendra K.S., Parthasaraty K., Identification and control of dynamical systems using neural networks, IEE Transactions on neural networks Vol 1 Num 1 p. 4-27 March 1990

Narendra K.S., Mukhopadhyay S., Neural networks in control systems, Proceedings of the 31<sup>st</sup> Conference on decision and control Tucson Arizona p. 1-6 December 1992

Narendra K.S., Adaptive Control Using Neural Networks, chapter 5 of Neural Networks for Control, Edited by Werbos P.J., Miller III T., Sutton R.S., MIT press, Cambridge MA 1995

Narendra K.S., Neural networks for control: Theory and practice, Proceedings of the IEEE Vol 84 Num 10 p. 1385-1406 October 1996

Narendra K.S., Adaptive control of dynamical systems using neural networks. Chapter 5 of Handbook of Intelligent Control. Neural, fuzzy and adaptive approaches. Edited by White D.A., Sofge D.A., Multiscience Press Inc. ISBN 0-442-30857-4 1992

Parker R.S., Doyle III F.J., Optimal control of a continuous bioreactor using an empirical non-linear model, Ind.Eng. Chem. Res. Vol 40 p. 1939-1951 2001

Patkar, A.; Seo, J., Fermentation kinetics of recombinant yeast in batch and fed-batch cultures, Biotechnology and Bioengineering. Vol 40 p. 103-109. 1992

Patkar, A.; Seo, J., Lim H. C., Modeling and optimization of cloned invertase expression in *Saccharomyces cerevisiae*, Biotechnology and Bioengineering Vol 41 p. 1066-1074. 1993

Pearson R. K., Categories of Non-linear Dynamic Models 1, chapter 29 in System Theory: Modeling Analysis and Control pp. 391-403. T.E. Djaferis and I.C. Schick editors 2000

\*Pearson R.K., Menold P.H., Allgower F., Practically-motivated input sequences for non-linear model identification, technical research report Institut fur Automatik ETH, Zurich, 1997

Pedret C., Poncet A., Stadler K., Toller A., Glattfelder A.H., Bemporad A., Morari M., Model-varying predictive control of a non-linear system, Technical report (AUT 00-07) Automatic Control Laboratory ETH, Zurich Switzerland 2000

Philippidis G.P., Hatzis C., Biochemical Engineering Analysis of Critical Process Factors in the Biomass-to-Ethanol Technology, Biotechnology Progress 13 p. 222-231. 1997

Philippidis G.P., Smith T.K., Wyman C.E., Study of the Enzymatic Hydrolysis of Cellulose for the Production of Fuel Ethanol by the Simultaneous Saccharification and Fermentation Process, Biotechnology and Bioengineering, Vol 41 p. 846-853. 1993

Prasad G., Swidenbank E., Hogg B.W., A neural net model-based multivariable long-range predictive control strategy applied in thermal power plant control, IEEE Transactions on Energy Conversion, Vol 13 Num 2 p. 176-182 June 1998

Psaltis D., Sideris A., Yamamura A.A., A multilayered neural network controller, IEE Control systems magazine p. 17-21 April 1988

Pushpavanam S., Rao S., Khan I., Optimization of a biochemical fed-batch reactor using sequential quadratic programming, Ind. Eng. Chem. Res. 38 p. 1198-2004 1999

Qin S.J., McAvoy T.J., Non-linear FIR modeling via neural net PLS approach, Computers and chemical engineering, Vol 20 Num 2 p. 147-159 1996

Quantrille T.E., Liu Y., Artificial Intelligence in Chemical Engineering, Academic press, San Diego USA (1991).



- Randall M.J., Winfield A.F.T., Pipe A.G., Stable on-line neural control of systems with closed kinematic chains, IEE Proc. Control Theory Applications Vol 46 Num 6 p.619-632 November 2000
- Ritza A., Sosnitza P., Ulber R., Scheper T., Fermentation Monitoring and Process Control, Current Opinion in Biotechnology, Vol 8, Num 2 p. 160-164. 1997
- Rodriguez-Acosta F., Regalado C.M., Torres N., Non-linear optimization of biotechnological processes by stochastic algorithms: Application to the maximization of the production rate of ethanol, glycerol and carbohydrates by *Saccharomyces cerevisiae*, Journal of Biotechnology 68 p. 15-28 1999
- Roubos J.A., van Straten G., van Boxtel A.J.B., An evolutionary strategy for fed-batch bioreactor optimization; concepts and performance, Journal of biotechnology 67 p. 173-187 1999
- Rumelhart D.E., Hinton G.E., Williams R.J., Learning Internal representations by error propagation in Parallel distributed processing: Explorations in the microstructures of cognition, Vol1 Edited by Rumelhart D.E., McClelland J.L., MIT Press Cambridge MA, p.318-362 1986
- Sargantanis I., Karim M.N., Adaptive Pole Placement Control Algorithm for DO-control in  $\beta$ -Lactamase Production. Biotechnology and Bioengineering, Vol 60 Num 1 p. 1-9. October 1998
- Sarle W.A., Neural networks and statistical models, Proceedings of the 9<sup>th</sup> annual SAS users group International Conference. April, 1994
- Sarle, W.S., ed, Neural Network FAQ, part 1 of 7: Introduction, periodic posting to the Usenet newsgroup comp.ai.neural-nets, URL: <ftp://ftp.sas.com/pub/neural/FAQ.html> 1997
- Segura C.J., Meziou A.Z., Quantifying the benefits of model predictive control in the petrochemical process industries, Technical Session 277. AIChE 2001 Annual Meeting Reno November 4-9 2001
- Srinivasan B., Palanki S., Bonvin D., Optimal operation of batch processes with multiple inputs and constraints, Session: Quantifying the benefits of advanced control, AIChE Annual meeting, Nov 7, 2001
- Sugeno M. Yasukawa T., A fuzzy-logic based approach to qualitative modeling, IEEE Transactions on Fuzzy Systems 1 p. 7-31 1993 \*
- Syu M. Hou C., Neural Network Dynamic Identification of 2,3-butanediol Fermentation by *Klebsiella oxytoca*, Process Control and Quality, Vol 10 p. 299-311. 1997
- Tholodur A., Ramirez W.F., Optimization of Fed-batch Bioreactors Using Neural Network Parameter Function Models, Biotechnology Progress, Vol 12, Num 3 p. 302-309. 1996
- Torres N.V., Voit E.O., Glez-Alcon C., Rodriguez F., An indirect optimization method for biochemical systems: Description of method and application to the maximization of the rate of ethanol, glycerol, and carbohydrate production on *Saccharomyces cerevisiae*, Biotechnology and bioengineering Vol 55 Num. 5 p.758-772 September 1997
- Tsaptinos D., Leigh J.R., A step-by-step approach for the construction of a fermentation process estimator, Proceedings of the world congress on neural networks, Vol 1, p. 216-219. 1993

Tsitsiklis J. N., Asynchronous Stochastic Approximation and Q-learning, Machine Learning, 16, p. 185-202 1994 \*

Tsitsiklis J. N., B. Van Roy, An analysis of temporal-difference learning with function approximation, IEEE Transactions on Automatic Control, Vol 42, Num 5, p. 674-690 May 1997 \*

Tsitsiklis J. N., B. Van Roy, Average cost temporal-difference learning, Automatica, Vol 35, Num 11, p. 1799-1808, November 1999a\*

Tsitsiklis J. N., B. Van Roy, Feature-Based Methods for Large Scale Dynamic Programming, Machine Learning, Vol 22, p. 59-94 1996 \*

Tsitsiklis J. N., B. Van Roy, On average versus discounted reward temporal-difference learning, Machine Learning, Vol. 49, No. 2, pp. 179-191, November 2002a\*

Tsitsiklis J. N., B. Van Roy, Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms and an application to pricing financial derivatives, IEEE Transactions on Automatic Control, Vol 44, Num. 10, p. 1840-1851 October 1999b \*

Tsitsiklis J. N., On the convergence of optimistic policy iteration, Journal of Machine Learning Research, Vol 3, p. 59-72 July 2002b\*

Tsitsiklis J.N., Van Roy B., Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and application to pricing financial derivatives, IEEE Transactions on Automatic Control; Vol 44, Num 10, p. 1840-1851 October 1999c\*

Tsoukalas L., Uhrig R., Fuzzy and Neural Approaches in Engineering. John Wiley & Sons. ISBN 0-471-16003-2. 1997

Vadigepalli R., Gatzke E.P., Doyle III F.J., Robust control of a multivariable experimental four-tank system, Ind. Eng. Chem. Res. 40 p.1916-1927 2001

Vaidyanathan R., Venkatasubramanian V., Representing and diagnosing dynamic process data using neural networks, Engineering Applications of Artificial Intelligence Vol 5 Num 1 p.11-21 1992a

Vaidyanathan R., Venkatasubramanian V., On the nature of fault space classification structure developed by neural networks, Engineering Applications of Artificial Intelligence Vol 5 Num 4 p.289-297 1992b

van Can H.J.L., te Braake H.A.B., Bijman A., Hellinga C., Luyben K.Ch.A.M, Heijnen J.J., An Efficient Model Development Strategy for Bioprocess Based on Neural Networks in Macroscopic Balances: Part II, Biotechnology and Bioengineering, Vol 62, Num 6, p.666-680. March 20, 1999

van Roy B., Bertsekas D. P., Lee Y., Tsitsiklis J. N., A Neuro dynamic Programming Approach to Retailer Inventory Management, November 1996

van Roy B., Tsitsiklis J. N., Regression methods for pricing complex American-style options, IEEE Trans. on Neural Networks, Vol 12, Num 4, p. 694-703 July 2001 \*

Venkatasubramanian V. Artificial neural networks : an introduction. Class notes. Laboratory of Intelligent Process Systems, School of Chemical Engineering, Purdue University, West Lafayette IN. 2000

- Venkatesh S., Dahleh M., On system identification of complex systems from finite data, *IEEE Transactions on automatic control*, Vol 46 Num 2 p. 235-257 February 2001
- Wang F.S., Chiou J.P. Optimal control and optimal time location problems of differential-algebraic systems by differential evolution, *Industrial & Engineering Chemistry Research*, 36 p. 5348-5357 1997
- Wang F.S. Cheng W., Simultaneous optimization of feeding rate and operation parameters for fed-batch fermentation processes, *Biotechnology progress* 15 p. 949-952 1999
- Watano S., Takashima H., Miyanami K., Control of moisture content in fluidized bed granulation by neural network, *Journal of chemical engineering of Japan* Vol 30 Num 2 p. 223-229 1997
- Werbos P.J., Overview of neural networks for control, *IEEE Control Systems* p. 40-41 January 1991
- Werbos P.J., Neurocontrol and supervised learning: an overview and evaluation. Chapter 3 of *Handbook of intelligent control. Neural, fuzzy and adaptive approaches*. Edited by White D.A., Sofge D.A., Multiscience Press Inc. ISBN 0-442-308574 1992
- Widrow B., Lehr M., 30 years of adaptive neural networks: Perceptron, Madalilne, and Backpropagation, *Proceedings of the IEEE* Vol 78 Num 9 p. 1415-1442 September 1990
- Williams, J.A., Keys to Bioreactor Selection, *Chemical Engineering Progress*, March 2002
- Yeo Y., Kwon T., A Neural PID Controller for the pH Neutralization Process, *Ind. Eng. Chem. Res.* 38 p. 978-987. 1999
- Zadeh L.A. Fuzzy sets, *Information and Control* Vol 8 p. 338-353 1965\*
- Zafiriou E., Morari M., Digital controller design for multivariable systems with structural closed-loop performance specifications, *Technical Research report University of Maryland, SCR TR 87-145* 1987a
- Zafiriou E., Morari M., Set point tracking vs. disturbance rejection for stable and unstable processes, *Technical Research report University of Maryland, SCR TR 87-142* 1987b
- Zafiriou E., Morari M., Design of the IMC filter by using the structured singular value approach, *Technical research report, Systems research center, University of Maryland, SRC TR 87-141* 1987c
- Zafiriou E., Recent advances in the use of the internal model control structure for the synthesis of robust multivariable controllers, Presented at the Shell workshop on process control, Dec 15-18 1986. *Technical research report, University of Maryland, TR-87-147*.1987
- Zafiriou E., Zhu J.M., Optimal feed rate profile determination for fed-batch fermentations in the presence of model-plant mismatch, *Technical research report, University of Maryland, TR-89-48* 1989
- Zafiriou E., Morari M., Internal model control: robust digital controller synthesis for multivariable open-loop stable or unstable processes, *Technical research report, University of Maryland, TR-90-48* Submitted to *International Journal of Control* 1990
- Zhang J., Morris A.J., Recurrent neuro-fuzzy networks for non-linear process modeling, *IEEE Transactions on neural networks* Vol 10 Num 2 March 1999

## **Appendix A**

*Non-linear multivariable (MIMO) process control model of a fed-batch bioreactor with neural networks*



## **Appendix B**

*Optimization of invertase production in a fed-batch bioreactor using dynamic programming coupled with fuzzy ARTMAP*