Non-Linear Neurocontrol of Chemical Processes using Reinforcement Learning

by

Stephen Leon Hunter

Thesis presented in partial fulfilment
of the requirements for the Degree

of

MASTER OF SCIENCE IN ENGINEERING
(CHEMICAL ENGINEERING)

in the Faculty of Engineering
at Stellenbosch University

Supervisor
Prof. C. Aldrich

December 2011

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Signature: ....
S.L. Hunter

Date: 2011/09/23

Abstract

Non-linear Neurocontrol of Chemical Processes using Reinforcement Learning

S.L. Hunter

Department of Process Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MScEng (Chem)

December 2011

The difficulties of chemical process control using plain Proportional-Integral-Derivative (PID) methods include interaction of process manipulated- and control variables as well as difficulty in tuning. One way of eliminating these problems is to use a centralized non-linear control solution such as a feed-forward neural network.

While many ways exist to train such neurocontrollers, one of the promising active research areas is reinforcement learning. The biggest drawing card of the neurocontrol using reinforcement learning paradigm is that no expert knowledge of the system is neccesary - all control knowledge is gained by interaction with the plant model.

This work uses episodic reinforcement learning to train controllers using two types of process model - non-linear dynamic models and non-linear autoregressive models. The first was termed model-based training and the second data-based learning. By testing the controllers obtained during data-based learning on the original model, the effect of plant model mismatch and therefore real-world applicability could be seen. In addition, two reinforcement learning algorithms, Policy Gradients with Parameter-based Exploration (PGPE) and the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) were compared to one-another. Set point tracking was facilitated by the use of integral error feedback.

Two control case studies were conducted to test the effectiveness of each type of controller and algorithm, and allowed comparison to multi-loop feedback control. The first is a ball mill grinding circuit pilot plant model with 5 degrees of freedom, and the second a 41-stage binary distillation column with 7 degrees of freedom.

The ball mill case study showed that centralized non-linear feedback control using neural networks can improve on even highly optimized PI control methods, with the proposed integral error-feedback neural network architecture working very well at tracking the set point. CMA-ES produced better results than PGPE, being able to find up to 20% better solutions. When compared to PI control, the ball mill neurocontrol solution had a 6% higher productivity and showed more than 10% improvement of the product size set point tracking. In the case of some plant-model mismatch (88% fit), the data-based ball mill neurocontroller still achieved better set point tracking and disturbance handling than PI control, but productivity did not improve.

The distillation case study showed less positive results. While reinforcement learning was able to learn successful controllers in the case of no plant-model mismatch and outperform LV- and $(L/D)(V/B)$-based PI control, the best-performing neurocontroller still performed up to 20% worse than DB-based PI control. Once again, CMA-ES showed better performance than PGPE, with latter even failing to find feasible control solutions.

While on-line learning in the ball mill study was made impossible due to stability issues, on-line adaptation in the distillation case study succeeded with the use of a partial neurocontroller. The learner was able to achieve, with a success rate of just over 50%, greater than 95% purity in both distillate and bottoms within 2,000 minutes of interacting with the plant.

Overall, reinforcement learning showed that, when there is sufficient room for improvement over existing control implementations, it can make for a very good replacement control solution even when no model is available. Future work should focus on evaluating these techniques in lab-scale control studies.

Uittreksel

Nie-linieêre Neurobeheer van Chemiese Prosesse met behulp van Versterkingsleer

("Non-linear Neurocontrol of Chemical Processes using Reinforcement Learning")

S.L. Hunter

Departement Prosesingenieurswese,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MScIng (Chem)

Desember 2011

Die probleme van prosesbeheer met behulp van gewone Proporsioneel-Integraal-Afgeleide (PID) metodes sluit interaksie van proses gemanipuleerde- en beheerveranderlikes, sowel as probleme met in-stemming in. Een manier om hierdie probleme te elimineer, is deur 'n gesentraliseerde nie-lineêre oplossing te gebruik, soos 'n vorentoe-gevoerde neurale netwerk.

Daar is baie maniere is om sulke neurobeheerders op te lei, waarvan die meer innoverende maniere versterkingsleer is. Die grootste trekpleister van versterkingsleer is dat geen deskundige kennis van die stelsel nodig is nie - alle beheerkennis word opgedoen deur interaksie met die aanleg model.

Hierdie werk gebruik episodiese versterkingsleer om beheerders met behulp van twee tipes van prosesmodel op te lei - nie-lineêre dinamiese modelle en nie-lineêre outoregressiewe modelle. Die eerste was model-gebaseerde opleiding en die tweede data-gebaseerde opleiding genoem. Deur die beheerders wat verkry is tydens data-gebaseerde opleiding op die oorspronklike model te toets, kon die effek van die verskil tussen aanleg en model gesien word, en 'n aanduiding van werklike wêreld toepaslikheid gee. Twee versterkingsleer algoritmes was met mekaar vergelyk - *Policy Gradients with Parameter-based Exploration* (PGPE), en die *Covariance Matrix Adaptation Evolution Strategy.* Stelpunt volging was deur integraalfout-terugvoer gefasiliteer.

Twee gevallestudies is uitgevoer om die doeltreffendheid van elke tipe beheerder en algoritme te toets, deur vergelyking met PI terugvoerbeheer. Die eerste is 'n balmeul toetsaanleg met 'n vryheidsgraad van 5 en die tweede 'n binêre distillasie kolom met 'n vryheidsgraad van 7.

Die balmeul gevallestudie het getoon dat gesentraliseerde nie-lineêre terugvoerbeheer met behulp van neurale netwerke selfs op hoogs-geoptimeerde PI beheer metodes kan verbeter. In vergelyking met PI beheer, kon die balmeul neurobeheer oplossing 'n 6% hoër produktiwiteit handhaaf en het meer as 10% verbetering in die handhawing van die produkgrootte stel punt getoon. In die geval van 'n 12% aanleg-model verskil, het die data-gebaseerde balmeul neurobeheerder steeds beter stel punt handhawing en versteuring hantering as PI beheer gewys, alhoewel produktiwiteit nie verbeter het nie. In beide gevalle het die integraalfout oplossing sukses getoon, en CMA-ES het tot 20% beter as PGPE gevaar.

Die distillasie gevallestudie het getoon dat die sukses van die balmeul gevallestudie nie noodwendig na ander aanlegte uitbrei nie. Alhoewel versterkingsleer in staat was om suksesvolle beheerders in die geval van geen aanleg-model verskil te leer, het die beste presterende neurobeheerder steeds tot 20% swakker as DB-gebaseerde PI beheer gevaar. Weereens het CMA-ES beter as PGPE gevaar, met die laasgenoemde wat selfs nie daarin kon slaag om werkende oplossings te vind nie.

Alhoewel onstabiliteit aan-lyn aanpassing in die balmeul gevallestudie onmoontlik gemaak het, is an-lyn aanpassing in die distillasie gevallestudie moontlik gemaak deur die gebruik van 'n gedeeltelike neurobeheerder. Die leerder was in staat om, met 'n slaagsyfer van net meer as 50 %, meer as 95 % suiwerheid in beide uitlaatstrome te bereik in 2,000 minute van die interaksie met die aanleg.

Op die ou end het versterkingsleer getoon dat, wanneer daar voldoende ruimte is vir verbetering oor bestaande beheer implementasies, kan dit 'n baie goeie vervanging wees selfs wanneer daar geen model beskikbaar is nie. Toekomstige werk moet fokus op laboratoriumskaal toepassings van hierdie tegnieke.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations.

Firstly my Heavenly Father for giving me the talent to pursue my studies.

My family and friends for their support.

My supervisor Prof. Aldrich for his guidance and patience.

Sasol and the NRF for financial support.

Contents

Nomenclature

Variables

α	Relative volitility between light and heavy component	[−]
B	Bottoms flow rate	[kmol/min]
B_{ij}	Cumulative breakage function	[−]
b_{ij}	Discrete breakage function	[−]
D	Distillate flow rate	[kmol/min]
D	Mill diameter	[ft]
d_{50}	Hydrocyclone cut size	[μm]
d_i	Mesh opening for size interval i	[μm]
E_i	Fraction of solids in size interval i reporting to underflow	[−]
F	Feed flow rate	[kmol/min]
f_v	Volume fraction of solids in cyclone feed	[−]
H	Total solids holdup in mill	[kg]
K	Tuning constant	[−]
λ	Effect of vapor flow on liquid flow	[−]
L	Reflux rate	[kmol/min]
L_i	Liquid flow from stage i	[kmol/min]
M	Mass flow rate	[kg/min]
M_{balls}	Mass of balls in mill	[short tons]
M_i	Liquid holdup on stage i	[kmol]
m	Mass fraction	[−]
m_i	Mass fraction of material in size interval i	[−]
N	Fraction of mill critical speed	[−]
P	Mill power draw	[kW]
Q	Volume flow rate	[l/min]
q_F	Feed liquid fraction	[−]
R_f	Fraction of fines short-circuiting to underflow	[−]
S_i	Breakage selection function	[h^{-1}]
SP	Set point	[−]
τ_L	Time constant for liquid dynamics	[min]
T_I	Integral time	[min]

t	Time	[min]
V	Volume	[m^3]
V	Boilup rate	[kmol/min]
V_i	Vapour flow from stage i	[kmol/min]
x_B	Bottoms composition	[mol fraction]
x_{balls}	Ball loading fraction of mill	[−]
x_D	Distillate composition	[mol fraction]
x_i	Liquid light component fraction on stage i	[mol fraction]
y_i	Vapour light component fraction on stage i	[mol fraction]
z_F	Feed composition	[mol fraction]

Other Symbols

$\boldsymbol{C}$	Covariance matrix
$\boldsymbol{\epsilon}$	Parameter perturbation vector
$\boldsymbol{I}$	Identity matrix
l	Current offspring number
λ	Offspring population size
$\boldsymbol{\mu}$	Parameter means
n	Current iteration
N	Number of iterations before updating
NF	Feed stage number
NT	Total number of stages
$\mathcal{N}$	Normal Distribution
r, R	Episode Reward
$\boldsymbol{\sigma}$	Parameter variance vector

Subscripts

0	Nominal value
b	Below feed stage
ff	Feed flow
I	Integral
i	Stage number
MP	Mill product
MF	Mill feed
OF	Hydrocyclone overflow
p	Proportional
SP	Sump
t	Value at time t, or above feed stage
UF	Hydrocyclone underflow

W	Water

Acronyms

CMA-ES	Covariance Matrix Adaptation Evolution Strategies
CSTR	Continuous Stirred Tank Reactor
DNF	Did Not Finish
ERL	Evolutionary Reinforcement Learning
FQI	Fitted Q Iteration
IAE	Integral of the Absolute Error
IMC	Inverse Model Control
MIMO	Multiple-Input-Multipe-Output
MFLC	Model-Free Learning Control
MPC	Model Predictive Control
MRAC	Model Reference Adaptive Control
NARX	Non-linear AutoRegressive with eXogenous inputs
PGPE	Policy Gradients with Parameter-based Exploration
PID	Proportional-Integral-Derivative
RL	Reinforcement Learning
SMNE	Symbiotic Memetic Neuro-Evolution

Chapter 1

Introduction

The brain is a highly complex, non-linear, parallel, information processing system. It can do in a fraction of a second what a conventional computer would take much longer to do - recognize a face, for instance. It is from the brain (a biological neural network) and its internal structure that the inspiration for artificial neural networks and related work springs (Haykin, 1994). Artificial neural networks, commonly just called neural networks, are an attempt to capture the non-linear adaptive representation ability of the brain. This can be in the form of electronic circuitry or software. Since the introduction of back-propagation learning in the 1980's and the increasing availability of computing power, neural networks have seen widespread use in information-age technologies such as visual and audio recognition. Other applications include process and robotic control, financial modelling, health science applications, and much more (Balakrishnan *et al.*, 1996; Haykin, 1994).

The adaptive nature of neural networks and their ability to handle any number of inputs and outputs make them ideal for application in adaptive control. Control using neural networks (i.e. neurocontrol) has been an active research area for many years (Hunt *et al.*, 1992).

One of the ways of training neural networks as controllers is reinforcement learning, which allows a learning agent to find a best solution by interaction with the plant. This can be contrasted with the more popular neural network training technique of supervised learning, where an agent is taught behaviour by example. This distinction makes reinforcement learning a more appropriate controller training method, since the agent will not be limited by the non-optimality of the target solution.

One good example of the unique effectiveness of using reinforcement learning to train neural networks is the work of Tesauro, who created TD-gammon (Sutton *&* Barto, 1998). Created in 1992 by Gerry Tesauro, TD-gammon learned to play backgammon by playing against itself. It did this by adapting a multilayer neural network according to the TD(λ) reinforcement learning algorithm. It eventually learned to play at near grandmaster level, in the process using unique strategies which have subsequently been adopted by human players (Sutton *&* Barto, 1998).

While neural networks have seen many successes in process control (Hussain, 1999), only a few of these have involved reinforcement learning. This is due to their most common use as non-linear dynamic or steady-state models, instead of as feedback controllers. The relatively scarcity of work using neural networks directly as controllers is mostly due to their unproven nature. Previous work in this regard that has shown promise has made use of reinforcement learning (Conradie, 2004; Conradie *&* Aldrich, 2010).

This work aims to improve on previous work by making the neurocontrollers track the set point(s) using integral error feedback. This would effectively allow a measure of tuning by adjusting the integral gain, and gives a measure of the distance from the set point, which would be valuable in cases where the environment undergoes slight changes.

In addition to this, no previous work has deeply explored the effect of plant-model mismatch on neurocontroller training using reinforcement learning. This will be investigated by training neurocontrollers on data-based (system identified black box) plant models, which in turn will be tested on the original model.

For neurocontrollers to further justify their use in process control, they need to be able to adapt to changing process conditions. This adaptive control technique will attempt to adjust the controller parameters on-line using episodic reinforcement learning, with the goal of moving the controller closer to the economic optimum.

The goals stated above will be pursued on two very different control case studies. Both will be done by simulation on rigorous non-linear dynamic models based on real-world systems. The first is a ball mill grinding circuit pilot plant grinding limestone slurry, and the second a binary distillation column. In both cases, comparison to the real-world PI control solutions will be made, with the goal of at least surpassing their control performance with and without the presence of disturbances.

This work is divided as follows. Firstly, previous work involving neural networks and/or reinforcement learning in chemical process control is explored in the literature review, which ends with a statement of the key questions. This is followed by an explanation of the experimental methods that were used, including flow diagrams of the reinforcement learning algorithms, details of the neurocontroller design, and simulation environment. The opening two chapters are followed by the two case studies, each in its own dedicated chapter. The work ends with a conclusion of the results.

Chapter 2

Neurocontrol of Chemical Processes

2.1 Introduction

Applications of neural networks in chemical process control have become increaslingly popular over the past two decades (Dote & Ovaska, 2001; Govindhasamy *et al.*, 2005; Conradie & Aldrich, 2010). Their multiple-input, multiple-output (MIMO) nature, and universal approximation and generalization abilities have made them highly valuable as non-linear models. This has resulted in their successful use in various model-based control techniques (Hunt *et al.*, 1992; Hussain, 1999; Piche *et al.*, 2000).

Apart from their use as non-linear models, neural networks can also be made to act as MIMO controllers (Hunt *et al.*, 1992; Hussain, 1999; Bloch & Denoeux, 2003; Govindhasamy *et al.*, 2005). This literature study aims to investigate previous successful applications of neural networks in both modelling and control capacities, with the focus of new work involving direct neural control.

The chapter starts by exploring previous conventional neuroncontrol work that can be classified into one of the following categories (Hunt *et al.*, 1992; Hussain, 1999):

- Model Predictive Control
- Inverse Model-based Control
- Adaptive Control

Each of the above-mentioned techniques are briefly explained, and a selection of previous work that has been verified in real-world tests or adopted in industry is provided.

In the following section the focus is shifted to control using reinforcement learning, a learning technique that has been highly successful in various robot control applications (Peters & Schaal, 2008; Sehnke *et al.*, 2009; Deisenroth *et al.*, 2009), but has also been applied to process control (Ernst *et al.*, 2008; Syafiie *et al.*, 2009; Conradie & Aldrich, 2010).

The chapter continues with an exposition of previous successful work with neurocontrol and reinforcement learning, and some of the possible uses thereof in adaptive control are explored. In addition, the problem of tracking the set point under direct neurocontrol is investigated. The chapter ends with a summary of the key questions to be answered by this thesis.

Any reader that is unfamiliar with neural networks and exactly what they are should read the brief background in Appendix A on page 102. Reinforcement learning is also briefly explained in Appendix B on page 105.

2.2 Conventional Neurocontrol

2.2.1 Predictive Control

The most common application of neural networks in process control is in model predictive control (MPC) (Hussain, 1999). Predictive control chooses optimal future manipulated variable actions as predicted by a process model and a given performance measure over a finite time interval, starting at the present time and ending at some horizon. Optimization is typically done at each time step, with the control horizon moving forward each time in order that the prediction interval size remains constant.

The reason for the popularity of neural network-based models in model predictive control is their ability to map non-linearities and the relative ease of representing a complex system, as opposed to developing a first-principles model (Hussain, 1999).

Piche *et al.* (2000) introduce a highly successful and widely applicable non-linear model predictive control scheme, which combines a linear dynamic model and static neural network model. It has seen industrial applications in a number of refining, petrochemical, pulp and paper, power and food processes.

In a water treatment process of Ondeo IS (France), a neural network model was trained to predict optimal coagulant dosages (Bloch *&* Denoeux, 2003). One year of raw data was used to train and validate the model, which used bootstrap sampling to generate confidence intervals for the output data. The new control solution resulted in more consistent high quality treated water at an optimal cost point. Continuous updating of the training data after implementation allowed further improvement of the system.

In a plastic injection moulding process, Chen *et al.* (2008) use neural networks for quality prediction in a supervisory control manner to pro-actively enhance future product quality. Mevawalla *et al.* (2011) use neural networks in a similar manner to optimize integrated circuit production.

Lu *&* Tsai (2008) use neural networks for model predictive temperature control of a variable-frequency oil-cooling machine and show good performance under both set point and load disturbances.

Hosen *et al.* (2011) use neural network-based model predictive control to enhance the performance of polystyrene batch reactors. They conclude that performance compared to regular PID control is superior, especially at start-up and with smoother control actions.

Other investigations using neural networks in a model predictive control scheme are all limited to simulation case studies and have not been verified experimentally (da Cruz Meleiro *et al.*, 2009; Yuzgec *et al.*, 2008; Santos *et al.*, 2000; Hussain, 1999).

2.2.2 Inverse Model Control

Direct inverse control is the most basic form of inverse model control, making use of an inverse system model cascaded with the plant (Hunt *et al.*, 1992). The network receives the system outputs as inputs, uses the inverse mapping to get the corresponding controller actions, and outputs these to the system under control. Thus the network acts directly as controller. This control structure is common in robotics applications (Hunt *et al.*, 1992). The control performance is highly dependent on how well the inverse model was constructed, resulting in a definite lack of robustness in this approach (Hunt *et al.*, 1992).

Internal model control is a more popular and stable form of inverse model control, and implements both a forward and inverse system model in the control structure. The forward model is placed in parallel with the real system. The difference between the system and model outputs is fed back to the forward part of the control system, where it undergoes preprocessing (usually a linear filter) before being sent to the controller network. The properties of IMC dictate that the forward part of the controller structure be related to the system inverse (Hunt *et al.*, 1992). IMC is limited to open-loop stable systems; however, it has been adopted for process control in more than one instance (Hunt *et al.*, 1992).

As will be seen later, the training of a neurocontroller using reinforcement learning results in the same kind of control structure as seen in direct inverse control.

Govindhasamy *et al.* (2005) describe the development of a neural model-based control strategy applied to an industrial aluminium substrate disk grinding process. The work was done in collaboration with Seagate Technology Media (Ireland) Ltd for the optimization of a ring grinding process. Both a direct inverse controller and IMC controller were developed and tested, with the IMC controller proving to be most effective, and was able to reduce thickness defects by 50% or more.

Another application, which uses neural networks in both modelling and control capacities, is a hot dip galvanizing line in Florange, France (Bloch *&* Denoeux, 2003). This approach employed neural networks in two parts of the process: optimization of the alloying thermal cycle, and control of the induction furnace. For the alloying thermal cycle optimization, a radial-basis function network was trained to predict the optimal inductive temperature set points, which helped to estimate 98% of the set points with error lower than 1.2%. A steady-state inverse controller was used to provide the furnace power preset in order to control the strip temperature at the exit of the furnace close to the predicted optimum.

Many simulation studies of inverse model control strategies have been published, most of which can be found in the review by Hussain (1999). The vast majority of the published applications used IMC, and were mostly limited to CSTRs, but included distillation and bioprocesses as well.

2.2.3 Adaptive Control

Adaptive control with neural networks, specifically as described by Hussain (1999) for use in chemical plants, use reference models that provide desired plant outputs. This type of adaptive control is known as Model Reference Adaptive Control (MRAC). The controller parameters are adjusted (e.g., using backpropagation gradient descent) to make the actual plant output follow the reference plant output asymptotically.

Alaradi & Rohani (2002) use a first-principles dynamic model based on an industrial unit (located at the Consumer's Co-operative Refineries Ltd. Regina, Canada) to study identification and control using neural networks. They show that for a noise-free system, their model-reference adaptive control scheme is capable of maintaining the riser temperature, the pressure difference between the reactor vessel and regenerator, and the catalyst bed level in the reactor vessel in the presence of set-point and disturbance changes.

In other simulated work, Hussain (1999) lists the following neural network adaptive control applications in chemical processes: two CSTRs, two bioreactors, one tank level control, one neutralization process, one fermentation process, and a polymerization process. All applications used the multi-layer perceptron neural network, and showed good set point tracking robustness.

2.3 Reinforcement Learning Control

Reinforcement learning can be seen as a stepwise learning algorithm that needs exploration and subsequent feedback (i.e., reinforcement) in order to learn. A reinforcement leaning agent acts in discrete time, and receives a scalar reward signal at each time step, telling it how well it is doing based on its current state and chosen actions.

In traditional reinforcement learning, states and actions are taken as discrete variables. A value is associated with each state, and these values (called the value function) is updated as new experience becomes available. An agent can also possess a state-value function, which describes the combined value of a specific action in a specific state. When an agent acts, it can choose to explore or to act "greedily", meaning it chooses the optimal action as prescribed by its value or state-value function. The state or state-action value functions are most easily represented as tables. Model-free learning control, described in Section 2.3.1 is an example of an algorithm that uses the table approach.

Most modern reinforcement learning techniques use function approximation to represent the value function. Two such algorithms are discussed below, Fitted Q-Iteration in Section 2.3.2 and Gaussian Process Dynamic Programming in Section 2.3.3.

The necessary discretization of the state and action space has led to interest in other forms of reinforcement leaning that do not require the optimization of a value function to find optimal actions. These techniques can directly adjust a control policy (usually a neural network) as they learn. Two such forms are policy gradient reinforcement learning and evolutionary reinforcement leaning, which are disscussed separately in the next section (2.4).

2.3.1 Model-Free Learning Control

Model-free learning control (MFLC) is a type of Q-learning look-up table approach to process control, developed by researchers from the Universities of Putra Malaysia and Valladolid (Spain) (Syafiie *et al.*, 2009). This approach is more akin to traditional reinforcement learning in that it uses a value function, stored as a table, to learn control.

The authors attempt to solve two problems concerning the application of value function-based reinforcement learning to practical process control. The first problem is the invariably large state and action spaces encountered due to the use of discrete states and actions. Another problem is that control design needs to take into account input and output constraints.

The authors attempt to resolve this by bounding the incremental control signal, and including hierarchical input and output limitations. They focus the application of this control strategy on wastewater treatment processes, specifically the advanced oxidation process. Their motivation is a more cost-effective control solution than before due to its ability to learn.

Their control strategy is tested on a laboratory-scale pilot plant, where phenolic wastewater is oxidized to carboxylic acids and carbon dioxide. Their results showed that their strategy can achieve good performance, with guaranteed on-specification discharge at maximum degradation rate.

It is apparent that a major limitation in MFLC is the use of discrete states and actions, a necessary component in the use of a table-based value-function in reinforcement learning (Sutton *&* Barto, 1998).

This may not matter if the intervals are small, but it does introduce memory and computational limitations in cases of many states and actions discretized into small intervals (Sutton *&* Barto, 1998). While a more accurate value estimate is obtained than when value-function approximation is used (a good example is FQI, described next), this approach will still be hampered as the amount of states and actions increase. Therefore MFLC is not a good candidate for a universal control paradigm such as that of Conradie (2004).

2.3.2 Fitted Q-iteration

Fitted Q-iteration, first published in 2003, aims to resolve the problem of large state and action spaces by approximating the value function using an appropriate supervised learning algorithm on a suitable function approximator (e.g. neural networks, regression tree ensembles, etc.) (Ernst *et al.*, 2003). In a comparison with model predictive control (MPC), the good approximation ability of regression tree ensembles in representing a value-function for a power system oscillation problem is shown (Ernst *et al.*, 2008). The authors conclude that reinforcement learning (RL) may compete with MPC even when a good deterministic system model is available. In the case where no system model is available, the RL solution (fitted Q-iteration) would be preferable since MPC would require prior system identification, which RL does not need.

Fitted Q-iteration seems to provide reasonable promise for use in process control. However, the complexity of the value function still remains an issue, and in cases where there are many states and actions, a good approximation may be harder to come by. Another problem is that the value function needs to be evaluated as a grid, and the optimimum point (best value) on that grid needs to be found for each action in order to determine the optimal actions for the given state. This is computationally expensive, especially when a fine grid is used, and in larger problems may require more time than is available in a control interval.

2.3.3 Gaussian Process Dynamic Programming

Similarly to the work of Ernst *et al.* (2008), Deisenroth *et al.* (2009) use function approximation to model the value function. The authors use Gaussian processes to approximate the value functions and Bayesian active learning for state-space exploration. The learning algorithm is shown to be highly data-efficient and achieves on-line learning of transition dynamics and value functions on the fly and with almost no a priori knowledge. As a real-world test, authors show on-line learning of cart-pole inversion in less than a minute of real time. The speed of learning shown by this algorithm is unprecedented and shows a lot of promise for application in smaller-dimensional problems where discretization is a feasible option.

As with Fitted Q-iteration, the necessary discretiztion of the state and action space, as well as the necessary optimization of the approximated value function, makes for a computationally infeasible problem when many states and actions are found, and the discretization is fine enough for typical plant operation. Also, both algorithms have only been presented for the case of a single manipulated variable. Multiple manipulated variables necessitate the use of either multiple value functions or multidimensional value function optimization.

2.4 Reinforcement Learning and Neurocontrol

The problem of applying value-function based reinforcement learning control to high-dimensional MIMO chemical processes is a good example of the so-called "curse of dimensionality". The curse of dimensionality refers to a problem becoming intractable as the number of variables increase, which, for reinforcement learning, happens at a combinatorial rate (Sutton *&* Barto, 1998).

Value function approximation may negate this problem to some extent, but introduces other complexities arising from representation and optimization issues (Deisenroth *et al.*, 2009; Ernst *et al.*, 2008).

So if value-function based reinforcement learning is intractable in high-dimensional control problems, what is left to do? The answer, at least according to researchers in the fields of policy gradient and evolutionary reinforcement learning, is a direct search in policy space for optimal solutions (Sehnke *et al.*, 2009; Heidrich-Meisner *&* Igel, 2009).

An important distinction to make at this stage is that almost all policy search algorithms are *episodic*. This means that policy learning takes place in batches of episodes, which can have any constant length as prescribed by the end-user. During an episode, the control policy would typically remain constant (i.e. the neural network controller weights do not change). In many cases, the starting state assumed at the beginning of each episode would also be the same.

This can be contrasted with on-line reinforcement learning, where policy changes and value function updates take place continuously, with learning not broken up into episodes in the strict sense (Sutton *&* Barto, 1998; Deisenroth *et al.*, 2009). Of course, if the agent encounters an end state, the system will be reset, but, importantly, learning still takes place continuously.

There are some policy gradient algorithms that do not learn episodically, but in the author's experience these are generally data-inefficient and not robust to noise (Doya, 2000; Wawrzynski, 2009). An argument against such on-line learning techniques in chemical process control is dead time and delayed reinforcement - i.e. a good action may have positive consequences, but these consequences are delayed and reinforcement takes place too late for the agent to know what action caused it.

This section looks individually at policy gradient methods and evolution strategies, specifically the most successful algorithms in these respective fields. Previous experimental results are highlighted at the end.

2.4.1 Policy Gradient Methods

Policy gradient methods learn by performing gradient ascent in policy space using gradient estimates based on the expected average reward. They search directly in policy space instead of deriving a policy from a value function. Such methods are among the few feasible optimization strategies for complex, high-dimensional reinforcement learning problems with continuous states and actions (Sehnke *et al.*, 2009). This makes policy gradient methods well-suited to solving problems in chemical process control, where large numbers of continuous variables, both controlled and manipulated, are found.

A very successful and relatively simple policy gradient method is *Policy Gradients with Parameter-based Exploration*, or PGPE (Sehnke *et al.*, 2009). It improves on previous policy gradient techniques, which typically suffered from slow convergence in learning due to high variance in gradient estimates. This variance, the authors argue, is caused by repeated sampling from a probabilistic policy during learning.

The PGPE approach effectively tackles this problem by having the policy defined by a distribution over the controller parameters. The parameters are sampled from this distribution at the start of a learning episode, wherein the controller remains deterministic (constant controller weights during each episode). The resulting reward for each episode is therefore dependent on this deterministic controller and therefore more stable. Because the policy gradient is estimated from the reward received at each sequence or episode, the gradient estimate ends up being more stable as well.

The authors have developed two alternatives of the PGPE algorithm. One incorporates the basic idea, while the other improves on learning performance by making use of symmetric sampling. The complete mathematics of derivation is not shown here, but is available in their paper (Sehnke *et al.*, 2009). The algorithm is described further in the next chapter, where a flow diagram is also provided.

2.4.2 Evolutionary Reinforcement Learning

Reinforcement learning using evolutionary algorithms, specifically evolution strategies, is receiving growing interest (Heidrich-Meisner *&* Igel, 2009). An Evolution Strategy (ES) is a stochastic search algorithm for non-linear optimization. One such algorithm, Covariance Matrix Adaption Evolution Strategy (CMA-ES) has proven to be highly effective with non-separable, multi-modal test functions (Hansen & Kern, 2004). Recently, CMA-ES has been shown to be among the best currently available algorithms for performing policy search (Heidrich-Meisner & Igel, 2009).

Notable previous work in applying evolutionary reinforcement learning in chemical process control is that of Conradie, who did his PhD thesis at Stellenbosch University in South Africa on neurocontrol using evolutionary reinforcement learning (ERL) (Conradie & Aldrich, 2001*a*,*b*, 2005, 2010; Conradie *et al.*, 2002; Conradie, 2004). Conradie's work focused on the Symbiotic Memetic Neuro-Evolution (SMNE) algorithm.

Conradie's SMNE algorithm builds on previous work by using implicit fitness sharing for promoting genetic diversity in the neurocontroller population (Conradie, 2004). This entails a search for cooperative neurons that together encode the optimal solution within a single population of competing and cooperating neurons. He states that several parallel searches for partial solutions should prove more effective than a single search for the entire solution.

While the ERL algorithm maintains an explorative global search, local particle swarm optimization is done to refine existing solutions in the neurocontroller population, allowing an aggressive search for the global optimum (Conradie & Aldrich, 2010).

Two important features of Conradie's work can be identified. Firstly, SMNE allowed controller learning from a partial model, meaning that plant-model mismatch is mitigated to a certain extent. Secondly, his neuroevolution framework allowed for on-line adaptation to changing process conditions, which is one of the motivators for the use of neural networks as controllers.

The first item is important because real-time learning from the actual plant would require a prohibitive amount of time. Not only this, but an accurate first-principles model is expensive and time-consuming to develop, whereas a partial black-box model can be constructed directly from plant data (Conradie, 2004). The second item is important because without on-line adaptation, the trained neurocontroller may eventually become obsolete, or at least non-optimal.

2.4.3 Experimental Results

2.4.3.1 Policy Gradients with Parameter-based Exploration

Sehnke *et al.* (2009) evaluate the performance of their algorithm on a variety of tasks, extending from the inverse pendulum problem to robot standing, robot arm grasping, and ship steering. Without exception, the symmetric version of the algorithm learns faster than other policy gradient techniques such as Natural Actor-Critic, REINFORCE, SPSA (simultaneous perturbation stochastic adaptation), and ES (evolution strategies).

In their experiments, the authors mainly used a Jordan network (a type of recurrent neural network) with a single hidden layer. The controllers generally had many adjustable parameters (synaptic weights, bias values), with one controller having more than 1000 parameters. Even in this case, PGPE was able to learn a successful control policy.

The authors observed that PGPE performed better compared to other policy gradient techniques as episode length increased. This was attributed to the increase in gradient estimate variance with the number of actions under the other techniques. They also observed that symmetric sampling had a stronger impact when tasks had more complex, multimodal reward functions.

2.4.3.2 Symbiotic Memetic Neuro-Evolution

Conradie's control paradigm was applied to several simulated processes, including a ball mill pilot plant and a highly non-linear bioreactor (Conradie, 2004). In that work, the good control ability of neurocontrollers trained via SMNE was shown, as well as the improvement afforded by particle swarm optimization.

More recently, work showing the application of SMNE to a partially identified multi-effect batch distillation (MEBAD) column with experimental (real-world) validation was published (Conradie & Aldrich, 2010). This work shows the ability of SMNE to learn a good controller from a partial system model. The neurocontrol solution ends up improving power consumption by 45% compared to multi-loop PI control.

2.4.3.3 Covariance Matrix Adaptation Evolution Strategies

With regards to general non-linear optimization, Hansen & Kern (2004) show that CMA-ES outperforms many other black-box optimization algorithms in a variety highly non-linear test functions. In a comparison with other reinforcement learning algorithms on the ubiquitous cart-pole benchmark, Heidrich-Meisner & Igel (2009) show that CMA-ES outperforms algorithms such as Policy Gradient Reinforcement Learning (PGRL), Q-learning, SARSA(λ) with CMAC and even Symbiotic Adaptive Neuro-Evolution (SANE), a precursor to SMNE.

The algorithms are compared by looking at how many evaluations (or, equivalently, *episodes*) are required in order to reach a certain level of performance on various versions of the cart-pole benchmark. CMA-ES typically requires 2 to 10 times *fewer* episodes than competing algorithms. In one test, where incomplete state feedback makes up part of the problem, CMA-ES required 585 episodes. The next-best performing algorithm, CoSyNe, required 954 episodes, and the next one, ESP, 3800. The worst performing algorithm in this case was PGRL, which required over 400,000 episodes.

2.5 Set Point Tracking

How would one track the set point when using direct neurocontrol, such as when training is done with reinforcement learning? Unlike PI or PID control, simple feed-forward neural networks have no integral error feedback to force a system to achieve zero error for the set point. One approach could be to simply provide the set point(s) as scalar input(s) to a feed-forward network. During training, it would learn to recognize what behaviour corresponds to what input, and therefore perform a type of set point "tracking".

Conradie & Aldrich (2001*b*) use a recurrent (as opposed to feed-forward) network to achieve set point tracking. Instead of being fully recurrent, the network

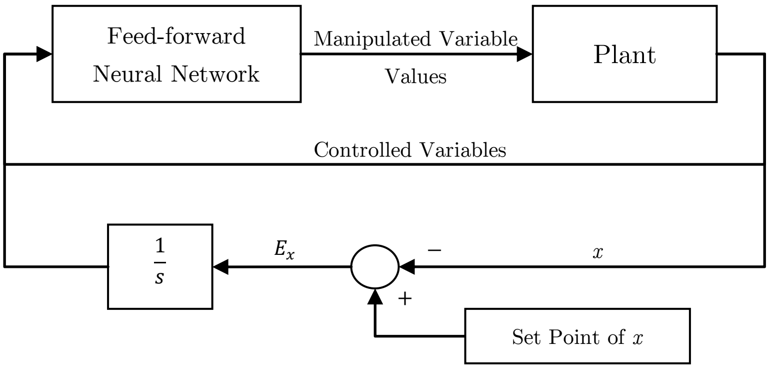

Figure 2.1: The proposed neurocontrol architecture, showing the integration of error for set point tracking for non-linear feedback control.

only feeds back previous outputs as additional inputs to the network at the next time step. As proposed above, the set point is simply provided as one of the network inputs. In tests involving a ball milling circuit control problem, this approach seems to work very well (Conradie & Aldrich, 2001*b*). Although this effect is never stated explicitly by the authors, the success of this approach would seem to be due to the dynamic nature of the network caused by the cyclic connection in the control structure.

Another approach would be to train a separate neural network for each desired set point. When it is desirable to change the set point, the existing neurocontroller is substituted with the appropriately trained one. In certain cases, where only a few set points are ever used, this will be a perfectly good solution. However, in an environment where customer specifications are variable, this approach would not work very well. In a study involving the development of a neurocontrol solution for the Tennessee Eastman control problem, Conradie & Aldrich (2001*a*) found that focusing on a single control objective (i.e. static set point) made learning easier than the approach given above. The end result was a separately trained controller for each set point, with boolean logic allowing for the neurocontrollers to perform the necessary set point changes (Conradie & Aldrich, 2001*a*).

Another approach, which this author will pursue in the following control studies, is to use a feed-forward network with the addition of integral error as an additional input. This idea is inspired by the integral action of PI controllers. It is hypothesised that by using this approach, configurable set point tracking will be achieved by the neurocontrol solutions. The idea is illustrated in the block diagram given in Figure 2.1. The configurability is afforded by the integrator itself - by adjusting its gain, the rate of integration is changed, allowing, possibly, for (partial) tuning of control actions. While the non-linearity between the integral error input and control output makes this approach different from the integral action of PI/PID controllers, it is believed that it is similar enough to be successful as a solution to the set point tracking problem. This author believes that this approach has enough potential to warrant further exploration.

2.6 Motivation

In the previous sections, several approaches to applying neurocontrol to chemical process control were highlighted. Different applications of reinforcement learning were also investigated. Why choose direct neurocontrol with policy search reinforcement learning?

The choice of using policy search, as opposed to value function optimization, is one that is largely motivated by the problem of dimensionality. As mentioned earlier, high-dimensional learning problems makes traditional reinforcement learning near-impossible to achieve (Sutton & Barto, 1998). This makes policy search the easiest choice in terms of the reinforcement learning approach.

In terms of control structure, the direct (inverse) neurocontrol structure is mostly a result of using policy search. Predictive or adaptive control with neural networks, at least in their typical NMPC or MRAC structures, are not well-solved with reinforcement learning. Not only is the direct neurocontrol approach a simpler solution, most of the benefits of using a neural network for control are still achieved: adaptability to changing conditions, a single (non-linear) multivariable controller for plant-wide control, and therefore implicit handling of controller interaction.

In terms of the specific choice of PGPE and CMA-ES for use as policy search algorithms, their use is motivated by the fact they are among the best algorithms in their respective fields - policy gradients for PGPE and evolution strategies for CMA-ES. Both algorithms, in their respective published results, show very good scaling with the size of the learning problem, and appear to find very good final solutions after a relatively small amount of learning episodes (Heidrich-Meisner & Igel, 2009; Sehnke *et al.*, 2009).

2.7 Key Questions

Judging from previous research, reinforcement learning-based neurocontrol of chemical processes has the potential to improve on traditional multi-loop PID control. The ability of neural networks to make complex non-linear mappings coupled with their generalization ability should eliminate the problem of controller interaction and provide better control of highly non-linear processes, all with the added benefit of being able to adapt online to changing process conditions. In light of previous research and the overall goal of this work, the following key questions can be identified.

- Does centralized neurocontrol, when trained via episodic reinforcement learning, improve on multi-loop PID control?
- If the above statement is true, can neurocontrollers trained on imperfect models (i.e. with significant plant-model mismatch) still outperform multi-loop PID control?
- How do the two chosen algorithms, PGPE and CMA-ES, compare to one-another on the same problems?
- Can either algorithm be used for on-line adaptation of the neurocontroller?

- Does neurocontrol with integral error feedback successfully track the set point?
- How does control performance scale with the number of neurons in the hidden layer of the neurocontroller?
- Is the proposed neurocontrol solution practical and economically viable?

The rest of this work will attempt to answer these questions as best as possible.

Chapter 3

Reinforcement Learning Methods

3.1 Introduction

The objectives of this study, taken from the key questions of the previous chapter, are as follows. (1) Evaluate the performance of neurocontrollers trained using reinforcement learning with rigorous first-principles models. (2) Evaluate the performance of neurocontrollers trained using reinforcement learning with system-identified models. (3) Evaluate on-line learning performance (i.e. adaptive control) using episodic reinforcement learning. Each of the above mentioned objectives can be termed a learning scenario, since each makes up a learning problem of its own.

In addition to these three objectives, other underlying questions also need to be answered. For instance, how does evolutionary reinforcement learning compare to policy gradient reinforcement learning? Also, what influence does neurocontroller size, specifically the number of hidden neurons, have on the learning and control performance of the reinforcement learning algorithms? What about the type of network output transformation? Questions such as these will be investigated in tandem with the main objectives.

The following section looks at the three learning scenarios mentioned above. The section after that describes the two reinforcement learning algorithms that will be used and compared with one another, with the help of some flow diagrams. The finer details surrounding the use of neural networks as controllers are then described, looking at the number of hidden units, neurons, and integral error feedback. The chapter ends with a description of the simulation environment and system identification that will be used in this study.

3.2 Learning Scenarios

The three learning scenarios are model-based learning, data-based learning, and on-line learning.

Model-based learning is where the assumption is made that the model on which the controller is trained is a perfect representation of the plant on which it will be applied. This case can be referred to as "no plant-model mismatch". Most simulation case studies make this assumption (Doya, 2000; Conradie *&* Aldrich, 2001*b*; Ernst *et al.*, 2008; Sehnke *et al.*, 2009).

In data-based learning, the attempt is made to introduce a measure of plant-model mismatch by performing system identification on plant data generated from standard PID control trajectories. The identified plant model is then used to train the controllers, which are subsequently tested on the original model. This can provide a measure of the suitability of the algorithms in real-world applications where plant models are only approximations of the real plant. While still removed from reality, these results should provide an indication of whether or not there is potential in using imperfect models to train controllers for real plants.

On-line learning or adaptive control is where a sub-optimal controller is improved by allowing it to "explore" the parameter space using the learning algorithm. The ability of neural networks to improve by learning is one of the prime motivators for their use in process control. It would therefore be useful if the reinforcement learning algorithm can continue to improve the controller after initial model- or data-based development.

3.3 Learning Algorithms

Two reinforcement learning algorithms will be compared with one-another in both of the upcoming case studies. The first is policy gradient reinforcement learning algorithm PGPE, or Policy Gradients with Parameter-based Exploration. The second is the evolutionary reinforcement learning algorithm CMA-ES, Covariance Matrix Adaptation Evolution Strategy. Each algorithm is briefly described in the following two subsections.

3.3.1 Policy Gradients with Parameter-based Exploration

The PGPE algorithm of Sehnke *et al.* (2009), as the name suggests, conducts parameter optimization by climbing a gradient. Policy gradient algorithms have been used in reinforcement learning almost since its inception (Busoniu *et al.*, 2010). As is usually the case in reinforcement learning, the gradient used in the algorithm is only an estimate of the true gradient. This algorithm differentiates itself from previous algorithms by estimating the gradient based on exploration in parameter space, rather than the usual action space. This effectively makes the gradient estimate independent of the policy representation, which would usually have to be differentiable for the algorithm to obtain the gradient. The biggest impact is in the variance of the gradient estimate, which is drastically reduced due to the policy remaining deterministic for the duration of an episode. It is one of the more promising gradient-based policy search algorithms to date.

The PGPE algorithm of Sehnke *et al.* (2009) is detailed in the flow diagram in Figure 3.1. The algorithm starts by initializing the policy parameters $\boldsymbol{\mu}$ and their respective variances $\boldsymbol{\sigma}$. It then iterates N times the following. First, a random parameter perturbation $\boldsymbol{\epsilon} = \mathcal{N}(0, \boldsymbol{\sigma})$ is drawn. This is first added to the parameter mean values μ and one episode is evaluated, giving the result r^+. The same $\boldsymbol{\epsilon}$ is then subtracted from the parameter mean values $\boldsymbol{\mu}$ and another episode is evaluated to get the result r^-. These r^+ and r^- values will then be stored as vectors $\boldsymbol{r}^+$ and

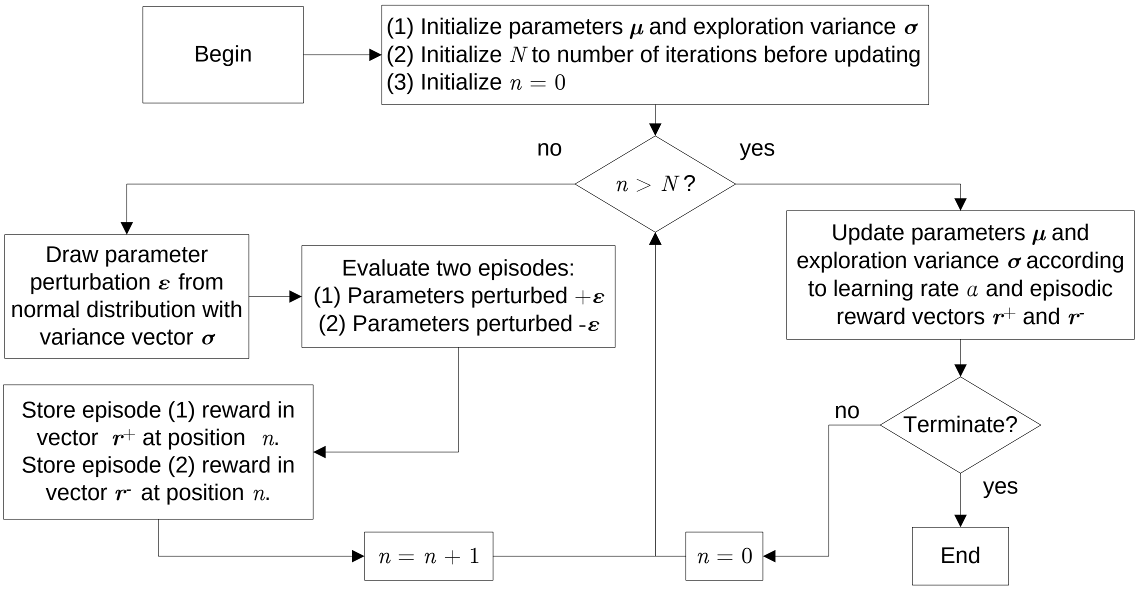
Begin
(1) Initialize parameters $\boldsymbol{\mu}$ and exploration variance $\boldsymbol{\sigma}$
(2) Initialize N to number of iterations before updating
(3) Initialize $n = 0$
no
yes
$n > N$?
Draw parameter perturbation $\boldsymbol{\varepsilon}$ from normal distribution with variance vector $\boldsymbol{\sigma}$
Evaluate two episodes: (1) Parameters perturbed $+\boldsymbol{\varepsilon}$ (2) Parameters perturbed $-\boldsymbol{\varepsilon}$
Update parameters $\boldsymbol{\mu}$ and exploration variance $\boldsymbol{\sigma}$ according to learning rate a and episodic reward vectors $\boldsymbol{r}^+$ and $\boldsymbol{r}^-$
Store episode (1) reward in vector $\boldsymbol{r}^+$ at position n. Store episode (2) reward in vector $\boldsymbol{r}^-$ at position n.
no
Terminate?
yes
$n = n + 1$
$n = 0$
End

Figure 3.1: PGPE flow diagram.

$\boldsymbol{r}^-$ of length N. Once the N iterations are complete, $\boldsymbol{r}^+$ and $\boldsymbol{r}^-$ are then used to update $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ according to learning rate α. This process is repeated until convergence or some stopping criteria is met, such as a certain number of episodes having completed. For all tests in this thesis, N was set equal to 1.

3.3.2 CMA Evolution Strategies

The Covariance Matrix Adaptation Evolution Strategies (CMA-ES) algorithm of Hansen & Ostermeier (2001) has been shown by Heidrich-Meisner & Igel (2009) to be ideally suited to the episodic reinforcement learning problem, in many cases performing better than most other algorithms. It is one of the most data-efficient episodic reinforcement learning algorithms for policy search.

An Evolution Strategy (ES) is a stochastic search algorithm for non-linear optimization. Search steps are taken based on mutation (stochastic variation) of points found so far. Mutation works by addition of a normally distributed random vector, meaning that the parameters of the normal distribution are very important for the performance of the search algorithm, i.e. the number of objective function evaluations before satisfactory convergence. The normal distribution parameters therefore need to be adapted in order for the algorithm to be effective.

CMA-ES uses rigorous mutative strategy parameter control to achieve effective search, and has proven to be highly effective with non-separable, multi-modal test functions (Hansen & Kern, 2004). CMA-ES essentially favours previously selected mutation steps in the future. When it pursues this objective rigorously, it results in a completely de-randomized self-adaptation scheme which adapts arbitrary normal mutation distributions.

A highly simplified flow diagram of the CMA-ES algorithm is provided in Fig-

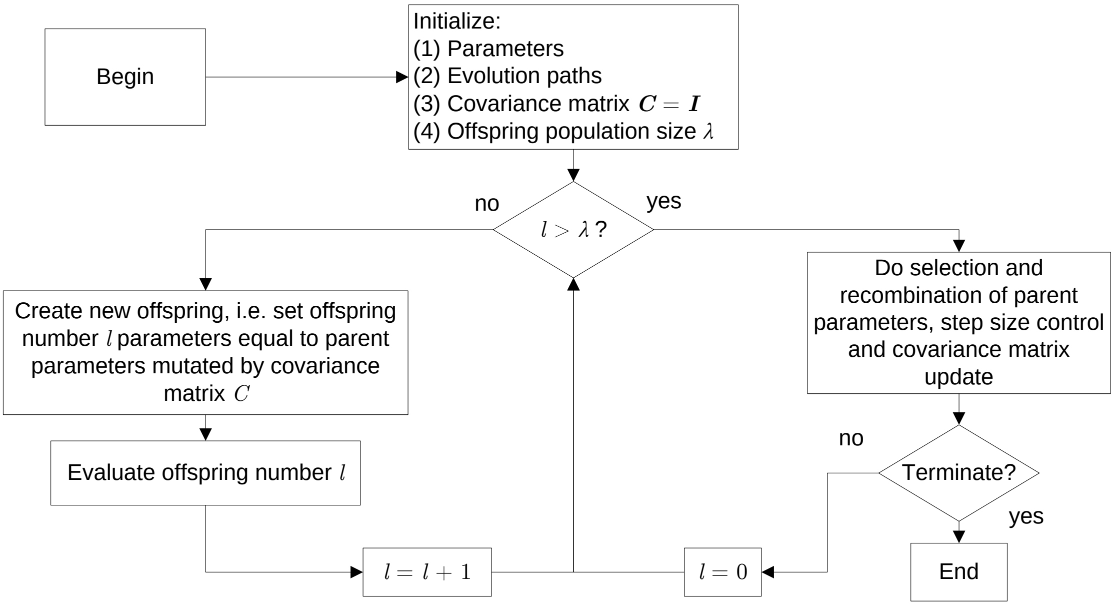
Begin
Initialize:
(1) Parameters
(2) Evolution paths
(3) Covariance matrix $C = I$
(4) Offspring population size λ
no
yes
$l > \lambda$?
Create new offspring, i.e. set offspring number l parameters equal to parent parameters mutated by covariance matrix C
Evaluate offspring number l
Do selection and recombination of parent parameters, step size control and covariance matrix update
no
Terminate?
yes
$l = l + 1$
$l = 0$
End

Figure 3.2: CMA-ES flow diagram.

ure 3.2. Any interested reader is referred to the articles of Hansen & Ostermeier (2001) and Heidrich-Meisner & Igel (2009).

3.4 Neurocontroller Design

Given enough hidden units, a neural network with only a single hidden layer can approximate any continuous function, linear or non-linear, to an arbitrary degree of accuracy (Haykin, 1994). Introducing an additional hidden layer is rarely needed, unless an arbitary decision boundary needs to be approximated (Heaton, 2008). In the case of neurocontrol, almost all applications only use a single hidden layer, since the control solution is practically always a continuous mapping from the state to the control space (Hussain, 1999).

The question of how many neurons to put in the hidden layer remains. How does the speed of learning, adaptability, and final control performance scale with the number of hidden neurons? Rules of thumb suggest that it should be less than twice the number of inputs, or somewhere between the number of inputs and the number of outputs (Heaton, 2008). In terms of reinforcement learning and neurocontrol, this question will be answered empirically by training different neurocontrollers on the exact same problem. Each neurocontroller will have the same inputs and outputs, but a different number of hidden neurons in the range suggested above. Measures such as integral of the absolute error during set point changes or input disturbances will help to rank the different neurocontrollers.

Another question is the type of output to use. In many cases, especially function approximation, the output layer is linear with no saturation (Haykin, 1994). In neurocontrol, however, these outputs need to be limited to feasible control values. For instance, one cannot specify a pumping rate of -10 l/min. Therefore the outputs need to be saturated. Doing so, however, increases the number of local minima, which can often be a hindrance in reinforcement learning, especially for policy gradient techniques.

One way to lessen to problem of local minima is to use smoother saturation to the minimum and maximum values. In this study, this is done by transforming the outputs using the hyperbolic tangent function. Linear saturated and hyperbolic tangent output networks will be compared alongside one-another.

As discussed in Section 2.5, a new avenue is explored in using a feed-forward (non-recurrent) neural network with integral error feedback, as illustrated in Figure 2.1. This is inspired by the integral action of PID controllers, which is arguably their most important feature. In addition to this proposed control configuration achieving the required level of set point tracking, it is hypothesised that a greater level of flexibility in terms of performance-tuning can be obtained. When the controller needs to track more than one set point, it is simply given the integral error in each case as a separate input.

In the practical application of this control approach, discrete-time integration with saturation to $[-10, 10]$ is used, with a sample time of 0.4 minutes. The relatively high sample time was to make sure that small-frequency oscillation did not affect the integral action of the controller.

3.5 Simulation Environment

The simulation environment for both case studies is Simulink ® 7.3. The dynamic models, neurocontrollers, and on-line optimizers in both case studies are programmed in C/C++ and compiled as MEX S-functions. Where appropriate, transport delays, rate transitions, etc. are used in order to approximate the delays in measuring process data. For example, in the distillation control case study, the composition measurements are delayed and updated every 30 seconds to more closely resemble an industrial gas chromatograph.

The PI controllers are also designed and implemented as Simulink blocks, as shown in Figure 3.3. The PI design makes use of the external feedback design as found in Marlin (2000), which achieves anti-reset windup, a necessary feature in both case studies.

A printout of the typical learning setup is shown in Figure 3.4. Here the ball mill model, `bmmex2`, is fed manipulated variable as well as disturbance inputs, with the outputs going to the controllers in a feedback loop (with measurement delays) and reward function for learning. The set point is made to follow a specified trajectory as seen in the bottom right corner of the figure. The `stop` block on the right ceases the simulation when "impossible" states are encountered, such as the sump running dry or overflowing. The neural network controller itself is the `MLPsim` block, with the inputs being the controlled variables and parameters, and output the network output. Custom blocks saturate the outputs.

The Simulink models are incorporated into MATLAB functions (i.e. using the `sim` function) that return an episode's reward for given network parameters. This allows MATLAB scripts for either of the reinforcement learning algorithms to search for the optimal neurcontroller parameters.

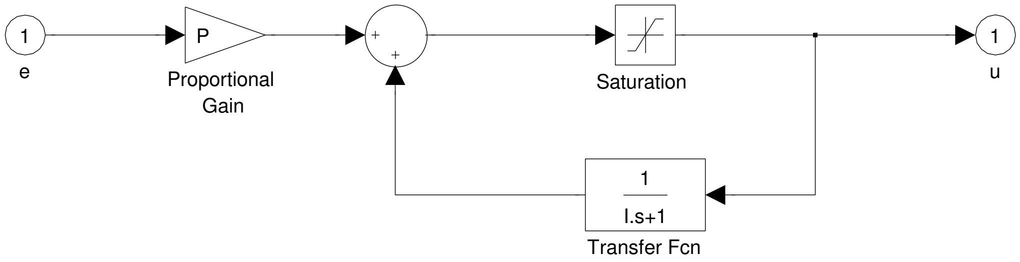

Figure 3.3: The PI control solution used in both case studies, using the external feedback design to achieve anti-reset windup.

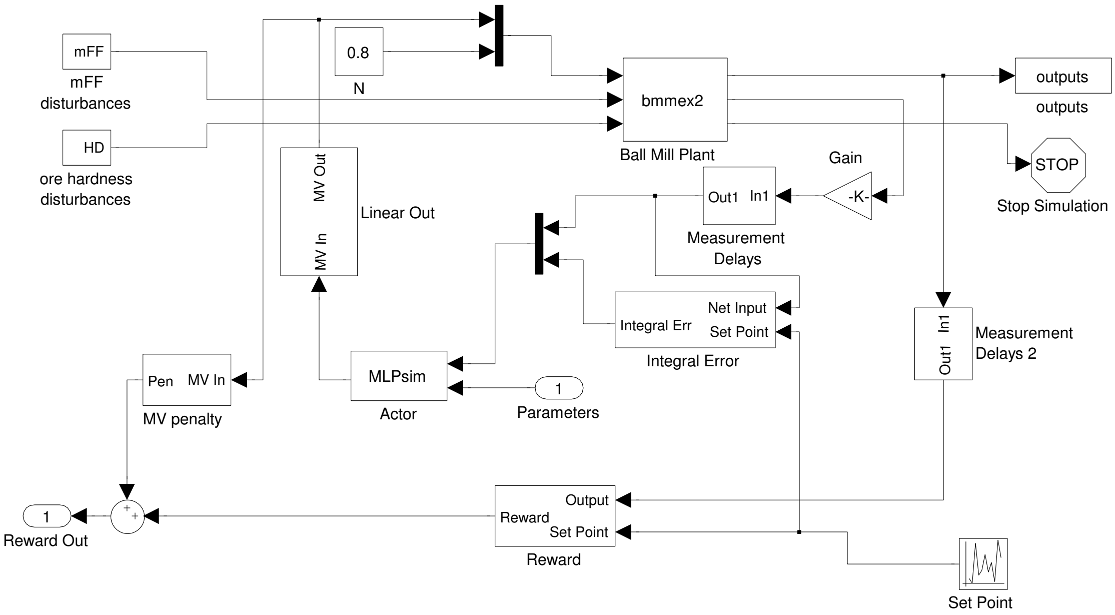

Figure 3.4: A screen capture of the Simulink model used for training neurocontrollers on the ball mill pilot plant model. The only external input is the neurocontroller parameters to use in the simulation, with the output being the reward obtained during simulation.

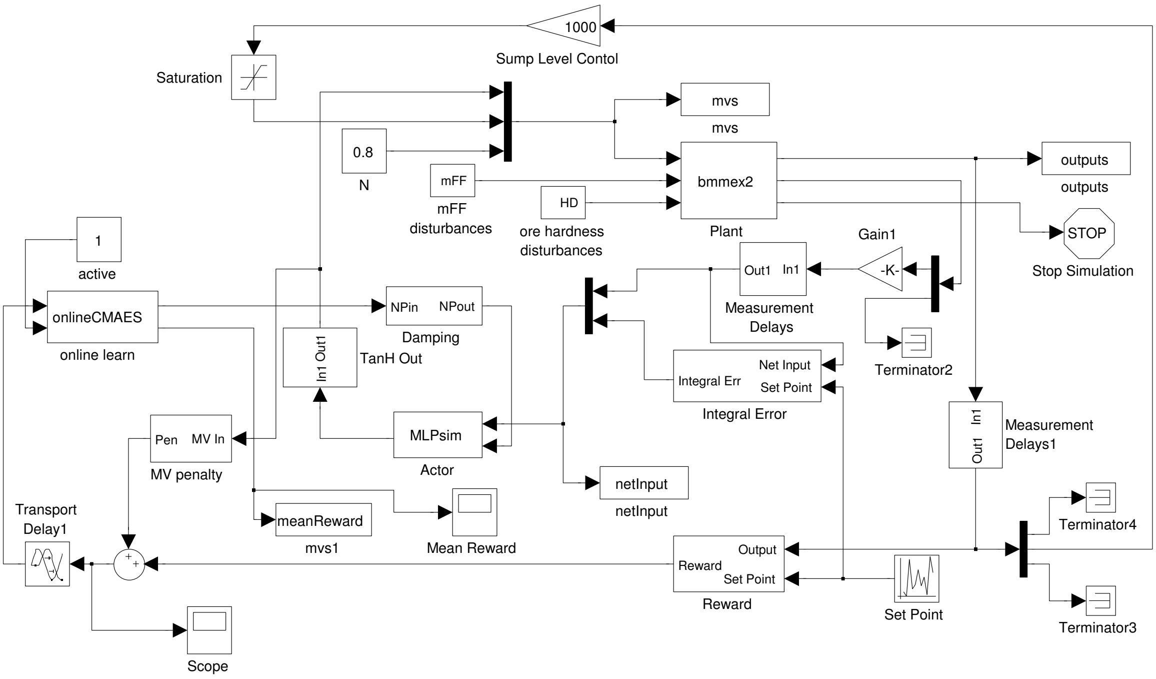

Figure 3.5: A screen capture of the Simulink model used for on-line training of neurocontrollers on the ball mill pilot plant model.

In order to do on-line learning with PGPE and CMA-ES, two additional MATLAB S-functions were written. The PGPE approach was written from custom code in C++, while the Shark Machine Learning Library was used in the CMA-ES approach (Igel *et al.*, 2008). These S-function blocks start from specified initial neural network parameters, and adapt on-line by specifying perturbed/mutated parameters and basing future adjustments on the subsequent episodic reward. Episode length and starting exploration intensity are also specified. Figure 3.5 shows a screen capture of the ball mill on-line learning set-up.

Figure 3.6 provides a comparison of the optimization of both algorithms on a 10-dimensional Rosenbrock test function, which can also serve to verify that both implementations work. Episodes were 10 time steps of 0.1 minutes long, with the simulations being 600 minutes. This equated to 600 iterations of either algorithm.

One can see in Figure 3.6 that CMA-ES turns out to be a better solver than PGPE, in finding a more optimal final solution, although initially it performs worse. It also shows how easily PGPE gets stuck in local minima, since from 200 iterations onwards it essentially fails to further optimize the function.

3.6 System Identification

A class of widely used non-linear time-series models based on input-output data is Non-linear Auto-Regressive with eXogenous input (NARX) models. These relate the current value of a time series to past values of the same series and current and past values of the driving series (Nelles, 2001). Therefore, an algebraic formulation for such a model would look like Equation 3.6.1, where u^n are inputs, y_t is an output

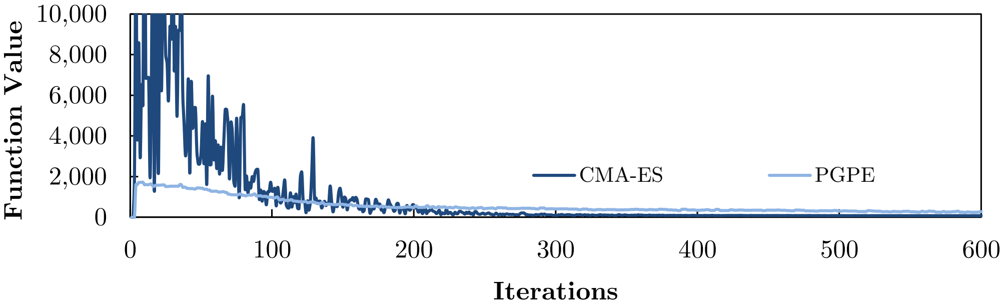

Figure 3.6: Example of optimization done using the online episodic solvers of CMA-ES and PGPE. This verifies that both implementations do work.

at time t, and F is some non-linear function such as a neural network, tree partition or wavelet network.

$$y_t = F(y_{t-1}, y_{t-2}, y_{t-3}, ..., u^1_t, u^1_{t-1}, u^1_{t-2}, ..., u^2_t, u^2_{t-1}, u^2_{t-1}, ...) + G \tag{3.6.1}$$

The System Identification Toolbox of MATLAB is used to construct the NARX models needed for the data-based learning scenario. The toolbox allows exporting of the models for use as Simulink blocks, so these can be directly used in the existing models (such as Figure 3.4) without any need for modification. The toolbox graphical user interface, invoked with the `ident` command, is used for training, testing, modifying, managing and exporting models.

The system identification done in both case studies relied on a mostly empirical trail-and-error approach, where various combinations of non-linearities and numbers of input and output terms for each was tried. The best resulting combinations are then combined for the final model, which is subsequently exported for use in the Simulink models used in training.

Chapter 4

Neurocontrol of a Ball Mill Grinding Circuit

4.1 Introduction

Grinding with tumbling mills is the last stage of comminution, where the ore particle sizes are reduced to the extent where the valuable minerals can be separated from the less valuable gangue (Wills & Napier-Munn, 2006). Ores have an economic optimum particle size dependent on many factors, such as subsequent separtion processes, dispersion of valuable mineral through the gangue, and so on.

The objective of the grinding process is therefore to control the product particle size closely. Good mineral processing is said to be largely dependent on correct grinding (Wills & Napier-Munn, 2006). Overgrinding wastes power for extra breakage, and may impede subsequent separation by lowering the valuable mineral's particle size below the expected value. Undergrinding will result in a lower degree of liberation, which lowers process economy.

Grinding circuits are notoriously energy-intensive, being the greatest single operating cost in minerals processing. In one survey of Canadian copper concentrators, average energy consumption in kWh.t^{-1} was 2.2 for crushing, 11.6 for grinding, and 2.6 for flotation (Wills & Napier-Munn, 2006).

Since grinding is so energy-intensive and affects subsequent separation processes, close control is extremely important. Maintaining a constant product size at the highest possible throughput can be taken as the overriding control objective. Due to interaction between plant variables, multi-loop proportional-integral (PI) control is difficult. In practice, one or more control loops are de-tuned in order to minimize the effect of interaction, resulting in a general slow-down in control response (Wills & Napier-Munn, 2006).

Due to the limitations of multiloop feedback control, alternative approaches such as model-predictive control and expert systems have been adopted at several sites (Wills & Napier-Munn, 2006).

The difficult non-linear nature of the grinding circuit control problem makes it an ideal testbed for new and unproven control techniques. In this chapter, reinforcement learning with neurocontrol is proposed as a feasible alternative to multi-loop PI control. One of the advantages of using neural networks for control is that their non-linear multivariable mapping ability makes them potentially immune to the controller interaction problem (Conradie & Aldrich, 2001*b*).

This chapter starts with an explanation of the control problem that will help to compare neurocontrol and PI control in the setting of a ball mill grinding circuit. Model details, the properties of the plant on which it is based, and the controlled and manipulated variables are presented. A process flow diagram is also provided.

The following section explains the details of the PI control solution that will be used in the control comparisons lying ahead, such as tuning and control interval times. Next, the specifics of the neurocontrol configurations that will be used are presented, with specific mention of integral set point tracking and the topologies that will be used.

The section on controller training ends off the discussion of experimental details, before the first section of results is presented. The first results section presents the model-based learning results, where no plant-model mismatch is assumed. The next section presents the data-based learning results, where training is done on system identified models, and the section after that presents the adaptive control results. The chapter ends with a conclusion of the results.

4.2 Circuit model

4.2.1 Circuit properties

The model used in this control study was developed by Rajamani & Herbst (1991*a*), which they used to compare feedback and optimal control of a ball mill pilot plant grinding limestone slurry (Rajamani & Herbst, 1991*b*).

A process flow diagram depicting the process is shown in Figure 4.1. The circuit consists of three unit operations: the ball mill, where grinding takes place, the sump, which collects the milled product, and the hydrocyclone, where separation takes place. The mill is fed by the mill feed stream, fresh water addition stream, and hydrocyclone underflow. The sump receives the milled ore incoming from the ball mill, as well as a water addition stream. The hydrocyclone receives the slurry incoming from the sump, sent by a variable-speed pump. The overflow of the hydrocyclone is the circuit product, while the underflow is sent back to the mill for further grinding.

The mill that was modelled by Rajamani & Herbst (1991*a*) is a 45 cm long standard Denver mill with an internal diameter of 76 cm. The ball load was 345 kg, corresponding to 40% mill filling. The classifier was a 7.5 cm Krebs hydrocyclone. The sump had a volume of 50 litres. The feed material was -1680 μm limestone.

The goal of the process is to produce a certain percentage (63% to 73%) of fines passing 44 μm as set by the controller, whilst still maintaining the maximum throughput possible.

The five manipulated variables and their operating range are summarised in Table 4.1.

For the purposes of process control, it assumed that the process controlled variables as given in Table 4.2 are measurable. These are the major controlled variables of a typical grinding circuit (Rajamani & Herbst, 1991*b*).

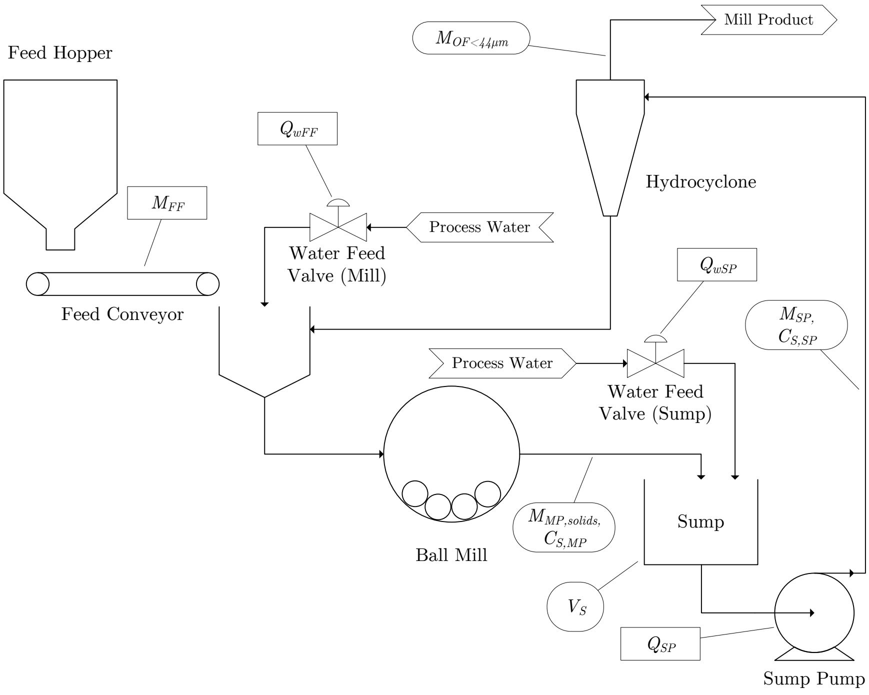

Figure 4.1: Process Flow Diagram of Ball Mill Grinding Circuit.

Table 4.1: Process manipulated variables and operating range.

Process Variable	Symbol	Minimum	Maximum
Fresh ore feed	M_{ff}	1.5 kg/min	2.3 kg/min
Sump pumping rate	Q_{SP}	15 l/min	33.3 l/min
Fresh water feed rate	Q_{Wff}	1.7 l/min	33.3 l/min
Sump water feed rate	Q_{WSP}	1.7 l/min	50 l/min
Mill rotation speed	N	30% of critical	80% of critical

Table 4.2: Process controlled variables and symbols.

Process Variable	Symbol
Product fraction passing 44 μm	$m_{OF<44\mu m}$
Mill throughput rate	$M_{MP,solids}$
Mill discharge density	$C_{S,MP}$
Cyclone feed density	$C_{S,SP}$
Cyclone mass feed rate	M_{SP}
Sump level	V_S

Both manipulated and controlled variables are indicated in rectangular boxes on the process flow diagram in Figure 4.1. Controlled variable boxes have round edges, while manipulated variable boxes have square edges. The symbols used are given in Tables 4.1 and 4.2.

4.2.2 Model properties

The model of Rajamani *&* Herbst (1991*a*) makes use of the population balance concept, where a series of differential equations are used to describe particle breakage in discrete size intervals. In the next subsections, the three different unit operations of the circuit will be described, starting with the ball mill itself, going on to the sump, and ending with the hydrocyclone.

4.2.2.1 Mill Model

In this study, 13 discrete size intervals were used, broken up into $\sqrt{2}$-proportional intervals based around the 1680μm particle size. The largest size interval was 2376-1680μm, and the smallest 37μm and less. The particle breakage in size interval i is described by Equation 4.2.1.

$$\begin{aligned}\frac{dH(t)\cdot m_{MP,i}(t)}{dt} = M_{MF,solids}\cdot m_{MF,i} - M_{MP,solids}\cdot m_{MP,i}(t) \\ - S_i\cdot H(t)\cdot m_{MP,i}(t) \\ + \sum_{j=1}^{i-1} b_{ij}\cdot S_j\cdot H(t)\cdot m_{MP,j}(t) \qquad (4.2.1)\end{aligned}$$

In Equation 4.2.1, $H(t)$ is the total mass holdup in the mill, while $m_{MP,i}$ and $m_{MF,i}$ refer to the mass fraction of material present in size interval i in the mill product and mill feed, respectively. $M_{MF,solids}$ and $M_{MP,solids}$ are the solids mass flow into and out of the mill, respectively. S_i (or S_j) is the size-discretized selection function for size interval i (or j), being defined in Eq. 4.2.3.

The size discretized breakage function b_{ij} represents the fraction of the primary breakage product material in the jth size interval. This breakage function has to be derived from its cumulative form given below in Eq. 4.2.2, which gives the fraction of material in size interval j that reports to all size intervals below and including interval i upon breakage.

$$B_{ij} = 0.31(d_i/d_{j+1})^{0.48} + 0.69(d_i/d_{j+1})^{2.8} \tag{4.2.2}$$

S_i (Eq. 4.2.3) is dependent upon the specific selection function given in Eq. 4.2.4, which is found by Rajamani *&* Herbst (1991*a*) to be sufficient in predicting all the data in the operating range of 1.5-2.3 kg/min feed rate. In addition, the mill power draw P (Eq. 4.2.5) and holdup H (Eq. 4.2.6) is required. The power draw is predicted as per Conradie *&* Aldrich (2001*b*) , who investigate the same circuit model for neurocontrol purposes. M_{balls} and x_{balls} refers to the mill filling of balls in short tons and fraction, respectively. D is the mill diameter in ft and N the fraction of critical speed that the mill is operating at.

$$S_i = S_i^E(P/H) \tag{4.2.3}$$

$$S_i^E(\text{tonne/kWh}) = 10.6\left(\sqrt{d_i d_{i+1}}/\sqrt{d_1 d_2}\right)^{1.427} \tag{4.2.4}$$

$$P = M_{\text{balls}}\left[3.1 \cdot D^{0.3} \cdot (3.2 - 3 \cdot x_{\text{balls}}) \cdot N \cdot \left(1 - \frac{0.1}{2^{9-10 \cdot N}}\right) - 1.13\right] \tag{4.2.5}$$

$$H(t) = V_M C_{S,MP} \tag{4.2.6}$$

Equation 4.2.7 describes the dynamics of solids concentration in the mill product ($C_{S,MP}$), with V_M being the volume of slurry in the mill and $C_{S,MF}$ the concentration of solids in the mill feed. Taken together with the 13 differential equations describing the particle breakage of the different size intervals, this gives 14 differential equations that describe the mill dynamic behaviour.

$$\frac{dC_{S,MP}}{dt} = \frac{Q_{MF}}{V_M}(C_{S,MF} - C_{S,MP}) \tag{4.2.7}$$

4.2.2.2 Sump Model

The slurry in the sump is assumed to be perfectly mixed due to the presence of an impeller. The following equations describe the sump behaviour if no size changes occur in the suspension (i.e. the impeller blades cause negligible attrition).

$$\frac{dm_{SP,i}}{dt} = M_{MP,solids} \cdot m_{MP,i} - M_{SP,solids} \cdot m_{SP,i} \tag{4.2.8}$$

$$\frac{dV_S}{dt} = Q_{MP} + Q_{W_{SP}} - Q_{SP} \tag{4.2.9}$$

$$\frac{d}{dt}(V_S \cdot C_{S,SP}) = Q_{MP} \cdot C_{S,MP} - Q_{SP} \cdot C_{S,SP} \tag{4.2.10}$$

In the equations above, $m_{SP,i}$ refers to the fraction of solids in the ith size interval present in the slurry, and Q_{MP} and Q_{SP} the volumetric addition and removal rate of slurry to and from the sump, respectively. $Q_{W_{SP}}$ is the volumetric rate of water addition to the sump and V_S the sump level (occupied volume). $C_{S,SP}$ is the concentration of solids in the sump product.

In total, given 13 size intervals, the sump behaviour is described by 15 differential equations.

4.2.2.3 Hydrocyclone Model

The hydrocyclone part of the model relies on the empirical modelling approach of Lynch *&* Rao (1975). The model equations are as follows, with the empirical constants as found by Rajamani *&* Herbst (1991*a*) for the appropriate operating range included.

$$WOF = \begin{cases} 1.363 \cdot WF - 10.75 & \text{for } WF < 21.4 \\ 0.837 \cdot WF + 0.35 & \text{for } WF \geq 21.4 \end{cases} \tag{4.2.11}$$

$$\log_e(d_{50}) = 3.616 + 15.006 \times 10^{-2} \cdot Q_{SP}\ [\mathrm{m}^3/\mathrm{h}]\ + 2.3 \cdot f_v \tag{4.2.12}$$

$$Y_i = 1 - \exp[-0.6931(d_i/d_{50})^{1.6}] \tag{4.2.13}$$

$$R_f = 0.818 - 0.7932 + \cdot WUF/WF \tag{4.2.14}$$

$$E_i = Y_i \cdot (1 - R_f) + R_f \tag{4.2.15}$$

In the equations above, WUF and WOF is the water flow rate in the under- and overflow, respectively, and WF is the water feed rate, all in kg/min. In addition, d_{50} is the size at which an equal amount of solids is split between the under- and overflow, f_v is the volume fraction of solids in the feed and R_f is the fraction of fines reporting to the underflow.

Equation 4.2.15 gives the fraction of solids split to the underflow (E_i), depending on the corrected efficiency curve Y_i given in Eq. 4.2.13, in which d_i refers to the size of the mesh opening for classifying size interval i.

As one can see, the hydrocyclone dynamics are not modelled, with only the mass balance given by the split equation (Eq. 4.2.15) being necessary.

4.2.2.4 Simulation details

In the simulations undertaken in this study, the second-order dynamic difference,

$$\frac{Q_{MP}(s)}{Q_{MF}(s)} = \frac{1}{(0.45s + 1)(0.41s + 1)} \tag{4.2.16}$$

between mill volumetric feed (Q_{MF}) and product (Q_{MP}) was not incorporated. This meant that the simulated dynamic response to an increase in volumetric feed rate was slightly faster than it should have been.

The time-delay of this second-order dynamic difference is typically only one to two minutes. Since there is a sump after the mill, the effect of ignoring this mill feed/product delay should be negligible.

Table 4.3: Size interval initial conditions.

Size Interval	Solids Mass Fraction	
(μm)	Fresh ore feed	Mill and Sump
2376-1680	0.15	1/13
1680-1188	0.15	1/13
1188-840	0.15	1/13
840-594	0.20	1/13
594-420	0.20	1/13
420-297	1/30	1/13
297-210	1/30	1/13
210-148	1/30	1/13
148-105	0.01	1/13
105-74	0.01	1/13
74-53	0.01	1/13
53-37	0.01	1/13
-37	0.01	1/13

Table 4.4: Initial process conditions.

Process Variable	Initial value
Mill holdup (H)	36 kg
Sump level (V_S)	0.0028 m^3
$C_{S,SP}$	350 kg/m^3

Mostly the same initial conditions as used by Conradie & Aldrich (2001*b*) were used in this study. The details are given in Tables 4.3 and 4.4.

4.3 Feedback Control

4.3.1 Control configuration

Feedback control for the milling circuit discussed here is optimized by Rajamani & Herbst (1991*b*) for the "Type II" case where mill throughput rate is controlled by fresh feed rate ($M_{MP,solids} \leftarrow M_{ff}$), and product size controlled by sump water rate ($m_{OF<44\mu m} \leftarrow Q_{W_{SP}}$).

Mill speed is set constant to 0.8 of the critical speed. The two remaining manipulated variables, being the sump pumping rate Q_{SP} and the mill water feed rate Q_{Wff}, are set proportional to sump level (V_S) and mill fresh ore feed (M_{FF}) respectively. The mill water feed rate is set so that the mill feed contains around 60% solids (Rajamani & Herbst, 1991*b*).

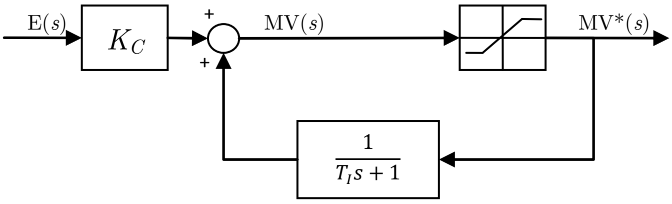

Figure 4.2: Block diagram of PI control with external feedback (Marlin, 2000). The right-hand block indicates saturation to the appropriate bounds.

4.3.2 Optimal tuning parameters

The optimization done by Rajamani *&* Herbst (1991*b*) made use of an objective function that was minimised. The optimal controller tuning parameters ended up being K_p = -0.11 and K_I = 1.2 for the particle size controller, and K_p = -0.75 and K_I = 0.12 for the mill throughput controller (Rajamani *&* Herbst, 1991*b*). The particle size controller was operated at 2 minute intervals, and the mill throughput controller at 0.4 minute intervals. These same constants and control intervals were used for the PI controllers implemented in the Simulink model. The controllers used were custom-made and made use of external feedback (see Figure 4.2) to achieve *anti-reset windup.*

4.4 Neurocontrol

4.4.1 Control configuration

The general neurocontroller solution is illustrated in Figure 4.3. It consists of a fully connected feed-forward neural network, which is fed representative process controlled variables and the set point integral error as inputs.

The inputs to neurocontroller are the variables in Table 4.2, with an additional input being the integral error as shown in Figure 4.3, giving a total of seven network inputs.

The network outputs are the manipulated variables of the plant as given in Table 4.1. As with the PI control solution, the fraction of mill critical speed is maintained constant at 0.8, so we effectively have four manipulated variables, and therefore four network outputs.

Only one hidden layer was used, and the number of units in the layer was varied in order to find the best configuration. In this case the hidden layer was given either 2, 3, 4, 6, 9 or 14 neurons, each using a hyperbolic tangent transfer function. The network inputs were all scaled to between $(0, 1)$. The network outputs were either linear with hard saturation to manipulated variable limits, or transformed with the hyperbolic tangent function to between these limits. This gave 12 different neural networks to be tested for control.

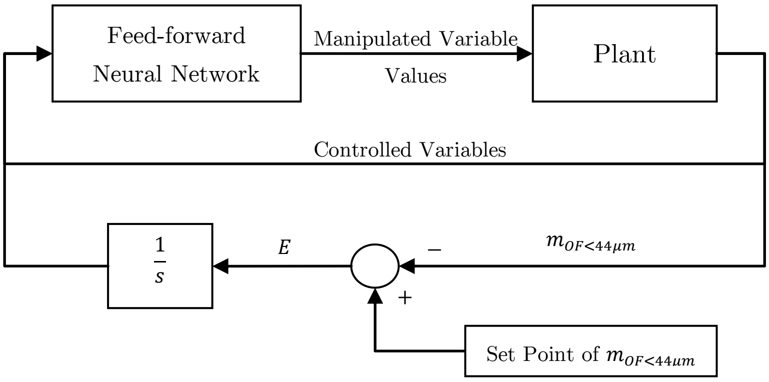
Feed-forward Neural Network
Manipulated Variable Values
Plant
Controlled Variables
$\frac{1}{s}$
E
−
+
$m_{OF<44\mu m}$
Set Point of $m_{OF<44um}$

Figure 4.3: Illustration of neurocontroller solution, showing the integration of error for set point tracking.

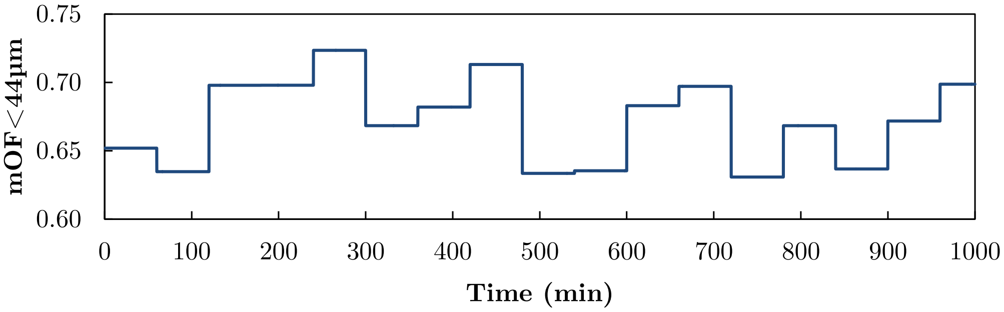

Figure 4.4: Example of the changing set point during a learning episode for the product size fraction passing 44 μm.

4.4.2 Training

The neural networks were trained with reinforcement learning, using either CMA-ES or PGPE, both of which were described in Section 3.3.

Controller learning takes place in episodes of 1000 simulated minutes in length, during which the set point is changed several times (see Figure 4.4).

For each episode, the learning algorithm needs a reward signal in order to know how well it did. The best controller would track the product fraction passing 44 μm set point closely, and do so at the highest possible production rate. Therefore, the reward needs to be based on the set point error and the product flow rate.

The reward equation that was used is given in Equation 4.4.1, where $M_{OF,solids}$ is the measured mass flow rate of solids in the product stream (i.e. hydrocyclone overflow) in kg/min. Since 2.3 kg/min is the maximum solids feed rate, it will also be the maximum solids output rate, hence its use in Equation 4.4.1.

Equation 4.4.1 was chosen based on the relative importance of the set point tracking versus the production rate. Initial work involving the PGPE algorithm

Table 4.5: Best fitness of five learning runs.

		Number of Hidden Units					
Algorithm	Output Type	2	3	4	6	9	14
CMA-ES	Linear	80,087	79,510	79,519	79,872	80,225	79,764
	TanH	78,194	78,517	78,172	78,377	79,440	80,855
PGPE	Linear	109,847	105,114	111,093	103,521	116,796	123,362
	TanH	100,659	97,158	85,050	94,247	84,699	93,603

helped to fine-tune the equation for best results, with the general structure remaining the same.

$$R = \sum_{t=0}(-200 \times (SP_{m_{OF<44\mu m}} - m_{OF<44\mu m}) - 10 \times |2.3 - M_{OF,solids}|) \qquad (4.4.1)$$

In the case where the system reaches a failure state, an appropriate negative reward (chosen as -100) is given and applied for each outstanding time step.

Note that the words "fitness" and "reward" are used interchangeably. Fitness is a commonly used term in evolutionary algorithms, while reward is commonly used in most reinforcement learning literature. For the CMA-ES implementation, fitness was a positive value that needed to be minimized (the absolute value of Equation 4.4.1). For the PGPE implementation, Equation 4.4.1 was used as-is. For comparison between the two, the absolute value (i.e. fitness) was used.

A maximum of 5000 episodes (i.e. function evaluations or iterations) was allowed for either algorithm to find a solution. The best solution obtained in those 5000 episodes was used for comparison between runs.

4.5 Model-Based Learning

The model-based learning results are presented in this section. The objective here is firstly to compare the quality of the different neurocontroller solutions obtained by CMA-ES and PGPE. Secondly, the best performing neurocontroller solution needs to be compared to the PI controller solution of Rajamani *&* Herbst (1991*b*).

4.5.1 Learning Results

Table 4.5 shows the best reward obtained in a set of five runs for each individual neurocontroller in both output configurations for both PGPE an CMA-ES. These results are visualized in Figure 4.5. A lower value (closer to zero) is better. The data show that both CMA-ES and PGPE can obtain better solutions when the smoother hyperbolic tan (TanH) output transformation is used, although the effect is much more pronounced with PGPE than with CMA-ES. It is also clear that CMA-ES can generally obtain better solutions than PGPE in the allowed number of iterations (which was 5000).

The reason for the increased performance of both learning algorithms when TanH output transformation is used has two interpretations.

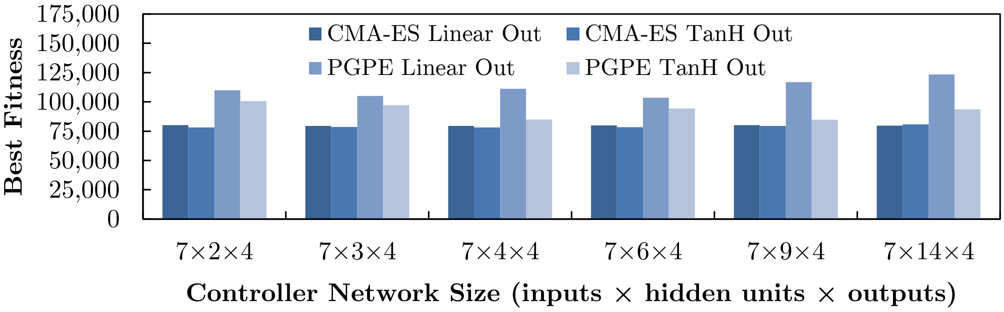

Figure 4.5: Comparison of the best neurocontroller solutions obtained under either algorithm for various hidden layer sizes. A lower fitness value is better.

Firstly, when TanH outputs are not used, hard saturation results in every control output saturating to the exact same maximum or minimum, no matter the distance of the output neuron sum from the origin (as long as it is far enough). Under TanH output saturation, there is always a difference, no matter how small, between different values of the output neuron sum. It is these small differences that facilitate better gradient estimation and more information being gained from exploration steps under PGPE and CMA-ES respectively.

Secondly, when outputs are transformed using the TanH function, the gradient near the origin is at its highest, and tends asymptotically towards zero as one approaches $\pm\infty$. Therefore the control response is larger for changes near the origin than for changes nearer the fringes of the control space. Overall this may result in a smoother control response, which increases performance as measured by the reward function.

What if a larger number of iterations were allowed? Figure 4.6 is a comparison of the learning trajectory of PGPE and CMA-ES during their best optimization runs of the $7 \times 6 \times 4$ with TanH output controller network. Both reach a value near their final optimum after around 1000 episodes, with CMA-ES showing earlier convergence. It is clear that even if more iterations were allowed, a much better solution would probably not have been obtained.

An important feature of that data that is not apparent from the previous figures is the variability of fitness of the solutions obtained. CMA-ES generally showed much lower fitness variance than PGPE, as shown in Table 4.6, which contains the standard deviation of five learning runs in each control configuration for both algorithms. Whilst CMA-ES almost invariably found a good solution, almost every second run under PGPE was a poor-performing on non-working controller. Both algorithms were more consistent under TanH output transformation.

It would seem that, for this problem at least, the problem dimensionality (which varied from 28 to 172 parameters depending on the network size) was not a big factor for CMA-ES optimization. The quality (Table 4.5) and variability (Table 4.6) of all CMA-ES solutions were relatively consistent. This is in line with literature, and shows how good CMA-ES is at neuroevolution even in high-dimensional problems (Hansen & Kern, 2004). It should be noted that, if fewer episodes were al-

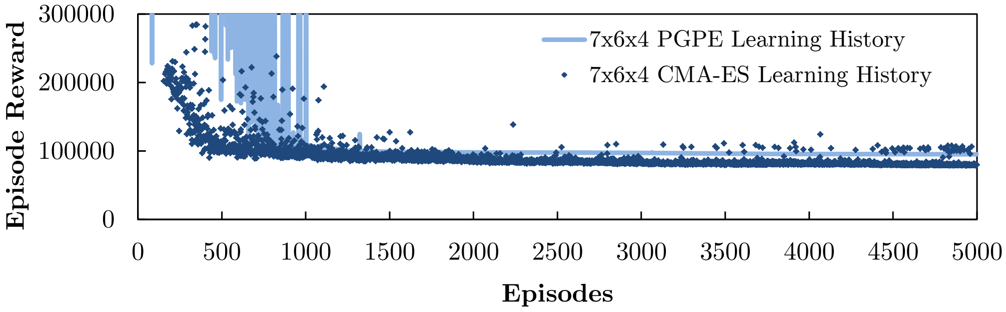

Figure 4.6: Learning trajectory for PGPE and CMA-ES during the best optimization runs of the $7 \times 6 \times 4$ with TanH output controller network.

Table 4.6: Standard deviation of five learning runs.

		Number of Hidden Units					
Algorithm	Output Type	2	3	4	6	9	14
CMA-ES	Linear	27,941	6,123	9,953	2,489	38,714	3,757
	TanH	4,595	469	605	793	672	378
PGPE	Linear	41,385	4,297	39,384	42,528	11,392	427,193
	TanH	6,953	6,647	15,106	3,419	13,214	416,620

lowed, a greater discrepancy between smaller and larger networks would have been obtained. This is because CMA-ES performance for locating the global optimum has been shown to scale between linearly and cubically with problem dimension (Hansen & Kern, 2004).

PGPE showed variability across the board, although Table 4.6 suggests that having a larger problem dimension had an impact when 14 hidden units were used. Previous results in literature suggest that PGPE should be able to scale well with the problem dimension (Sehnke *et al.*, 2009). However, these results show that, at least in comparison to CMA-ES, PGPE struggles to find consistent and good final solutions. This shows the relative difficulty of this learning problem.

4.5.2 Control Performance

For this control performance comparison, neurocontrol is compared with the PI control solution of Rajamani & Herbst (1991*b*).

Firstly, the best-performing neurocontroller is determined by a benchmark comparison where the Integral of the Absolute Error (IAE) of the product set point and mill throughput are the deciding factors. After the best neurocontroller is found, its set point tracking with and without the presence of disturbances is compared to that of the PI control solution of Rajamani & Herbst (1991*b*). This is followed by an investigation into performance improvement by adjusting the neural network's integral gain. The subsection closes with a discussion of the results.

4.5.2.1 Neurocontroller Comparison

According to Table 4.5, the best-performing neurocontroller, at least according to Equation 4.4.1, is the $7 \times 4 \times 4$ with TanH output controller optimized by CMA-ES. However, all the other CMA-ES solutions have very similar best results. Therefore they need to be tested on a benchmark test to determine their fitness for comparison to the PI control solution. PGPE controllers are not considered here since they clearly show inferior performance.

In order to do this, a simulation of 240 minutes is conducted and the set point is changed several times. The IAE from 60 minutes to 240 minutes is then calculated (the first 60 minutes allows the controller to stabilize from the starting state). The final amount of ground limestone produced is also calculated - an indication of the mill throughput throughout the test run.

The results are shown in Table 4.7 and visualized in Figures 4.7 and 4.8. The difference between each neurocontrol solution becomes much clearer when looking at the IAE results in Figure 4.7. All the neurocontrollers with linear output layers show more or less the same IAE and productivity, and also perform more poorly than their TanH output layer counterparts. As stated in the previous subsection, the better TanH-output performance could be explained by the smoother control response resulting from the high-gradient-to-low-gradient nature of the TanH function.

The neurocontrollers with TanH output show that, surprisingly perhaps, a neural network with only 2 hidden units can outperform larger networks on this relatively complex non-linear control problem. This trend is not as readily apparent from the reward scores in Table 4.5, and could even be entirely circumstantial. Nonetheless, the simplest neurocontroller that was trained gives the best result, providing some evidence for a 'simpler is better' approach to neurocontrol. The best explanation for this would seem to be the fact that the ball mill circuit considered here is a 2×2 control problem. There may be four manipulated variables, but two of these are typically controlled proportionally. This leaves the two more important manipulated variables, fresh ore feed M_{ff} and sump water feed $Q_{W_{SP}}$, to be controlled perfectly well by only two neurons in the hidden layer.

In order to verify that the above result is not simply due to fewer episodes being needed to properly train a simpler neurocontroller, a test run with 20,000 episodes on a $7 \times 4 \times 4$-TanH neurocontroller was done. It's final score was 78,218, with an IAE of 9.9 on the benchmark test. This showed an improvement over the previous best result for a $7 \times 4 \times 4$-TanH neurocontroller, but it is still not as good as the $7 \times 2 \times 4$-TanH best result.

The above result suggests that simpler indeed is better for neurocontrol design. However, as mentioned above, the fact that only two neurons in the hidden layer performs best is most probably explained by the fact that this is a 2×2 control problem. Other plants may require a larger hidden-to-input neurons ratio for best control, with the number of neurons in the hidden layer being at least as big as the control problem size. As the number of neurons given to the hidden layer increases, so does the problem dimensionality (and difficulty). Therefore the choice of hidden layer size should be given due attention, guided by the control problem size.

There also seems to be an IAE/productivity trade-off, however the productivity does not suffer too badly for the IAE performance gain. Going from the $7 \times 6 \times 4$-

Table 4.7: IAE and productivity for the different neurocontrollers optimized by CMA-ES

Performance	Output Type	Number of Hidden Units					
		2	3	4	6	9	14
IAE	Linear	17.7	17.0	16.8	15.7	16.8	19.3
	TanH	9.3	10.4	10.7	12.3	13.3	17.1
Productivity (kg)	Linear	408.4	406.5	406.0	410.1	406.2	409.3
	TanH	402.6	404.3	404.2	407.2	406.2	406.7

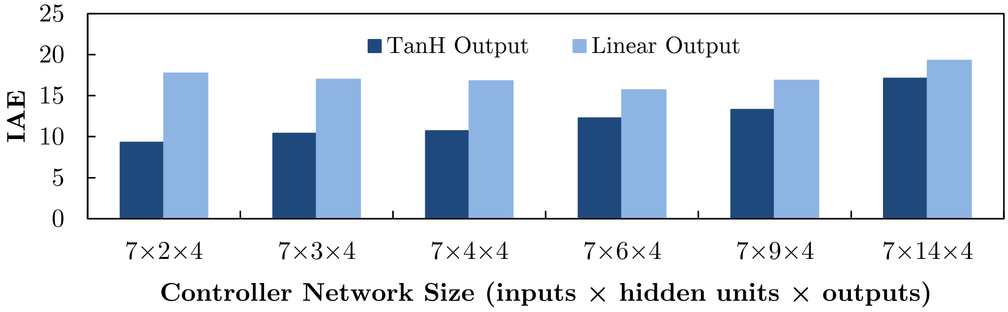

Figure 4.7: IAE for $t > 60$ of benchmark problem for the different neurocontrollers optimized by CMA-ES.

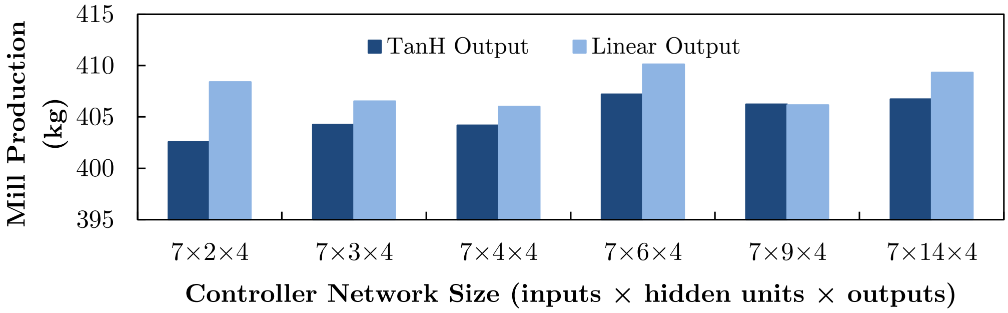

Figure 4.8: Productivity of benchmark problem for the different neurocontrollers optimized by CMA-ES.

TanH to 7×2×4-TanH neurocontroller, there is only a 1.1% decrease in productivity from 407.2 kg to 402.5 kg for a 24% improvement in IAE from 12.3 to 9.3.

Based on the results above, the $7 \times 2 \times 4$ with TanH output neurocontroller is chosen for comparison with the PI control solution in the following sections.

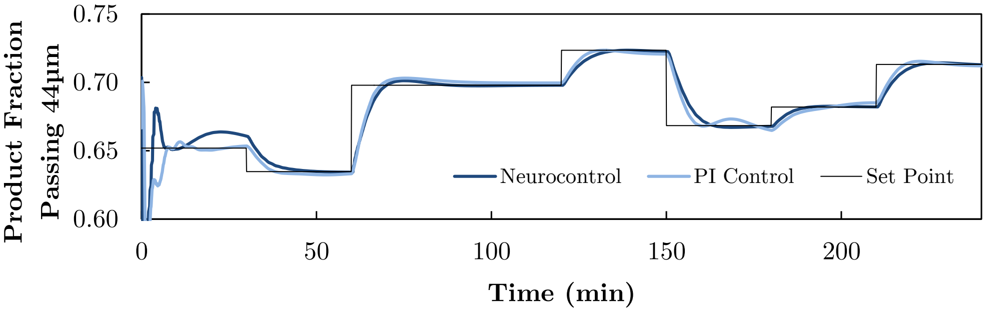

Figure 4.9: Set point tracking of the CMA-ES solution of the 7 × 2 × 4 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani & Herbst (1991*b*).

4.5.2.2 Set Point Tracking - No Disturbance

To compare the set point tracking ability of the neurocontrol solution with that of the PI control solution, the same benchmark test as used for comparison between the different neurocontrollers was used. The result is shown in Figure 4.9. Both control solutions give very similar results, with controller interaction not playing as large a role as expected in the case of PI control - which shows that the (de)tuning optimization done by Rajamani & Herbst (1991*b*) alleviated this problem.

The transient response at the beginning of the simulation is due to different starting states. PI control seems to suffer from a very slight steady-state offset.

The Integral of Absolute Error (IAE) from 60 to 240 minutes for the PI solution is 9.8, whilst that of the neurocontrol solution is 9.2, which is a 6% improvement.

An important area where neurocontrol provides superior performance over PI control is mill throughput. Figure 4.10 shows that for most of the 240 minutes, the neurocontrol solution has a higher product solids flow rate - meaning an overall higher circuit production rate. The total amount of ground limestone produced in the 240 minute simulation is 403 kg for the neurocontrol solution and 385 kg for the PI control solution. The neurocontrol solution therefore produced 5% more limestone than the PI control solution.

4.5.2.3 Set Point Tracking - Feed Size Distribution Disturbance

The ore feed size distribution disturbance is simulated by randomly perturbing the fraction of each size interval present in the feed every 12 minutes (similar to Conradie & Aldrich (2001*b*)). For the model of Rajamani & Herbst (1991*a*), there are a total of 13 size intervals. An example of the disturbance given to some of the size intervals is shown in Figure 4.11.

Figure 4.12 shows the set point tracking performance of the neurocontrol and PI control solutions under feed size distribution disturbances. This time, the PI control solution provides slightly superior performance over that of the neurocontrol solution - an IAE for $t > 60$ of 9.9 compared to 10.7, an 8% difference. Compared to the case where the plant experiences no disturbances, the PI control performance

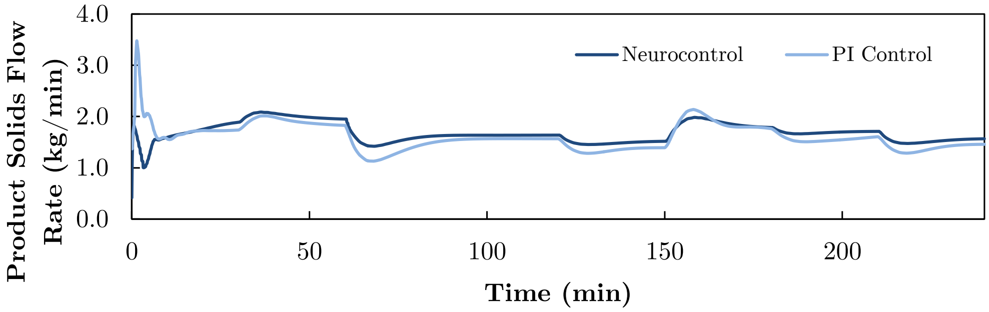

Figure 4.10: Product solids flow rate of the CMA-ES solution of the 7 × 2 × 4 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani *&* Herbst (1991*b*).

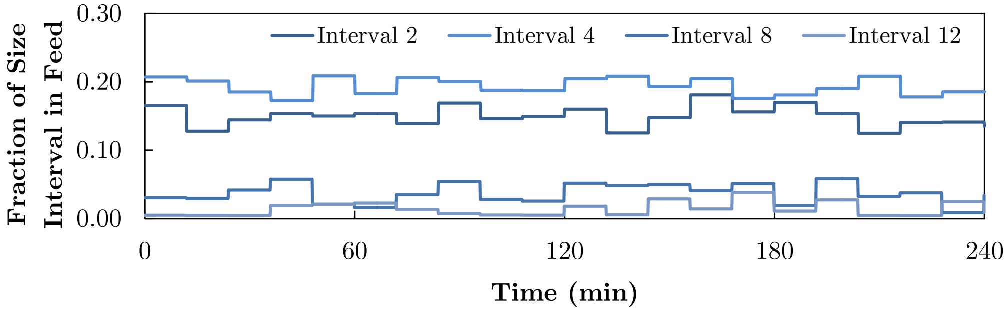

Figure 4.11: Example of feed size fraction distribution disturbances for some of the intervals. There are 13 intervals in total, 1 being the coarsest and 13 being the finest.

does not worsen very much at all, whilst the neurocontrol performance drops by 17% as measured by IAE.

Once again, neurocontrol maintains a higher mill throughput than PI control, as seen in Figure 4.13. In fact, the amount of ground limestone produced under neurocontrol is slightly higher than under no disturbances at 408 kg. The PI control solution also produced slightly more at 388 kg, meaning a 5% difference once again.

4.5.2.4 Set Point Tracking - Ore Hardness Disturbance

For the case of ore hardness disturbance simulation, the ore breakage selection function was perturbed every 7 minutes by a small random amount, as seen in Figure 4.14.

The set point tracking and production rate results under ore hardness disturbances are shown in Figures 4.15 and 4.16 respectively. The performance under both neurocontrol and PI control decrease by more than 40% each as measured by IAE. The IAE ($t > 60$) for PI control was 14.3, compared to 15.5 for neurocontrol (9% worse). Once again, production was 5% higher under neurocontrol.

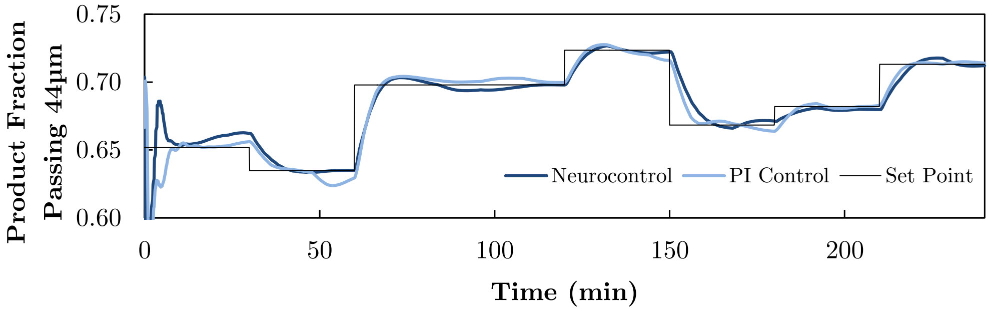

Figure 4.12: Set point tracking of the CMA-ES solution of the 7 × 2 × 4 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani & Herbst (1991*b*) under feed size fraction distribution disturbances.

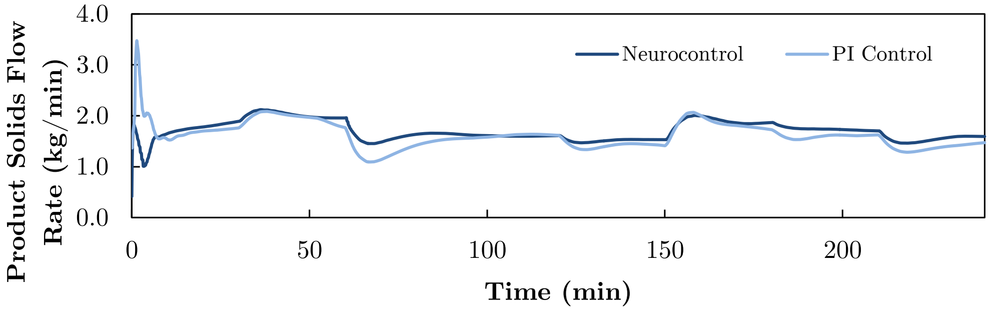

Figure 4.13: Product solids flow rate of the CMA-ES solution of the 7 × 2 × 4 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani & Herbst (1991*b*) under feed size fraction distribution disturbances.

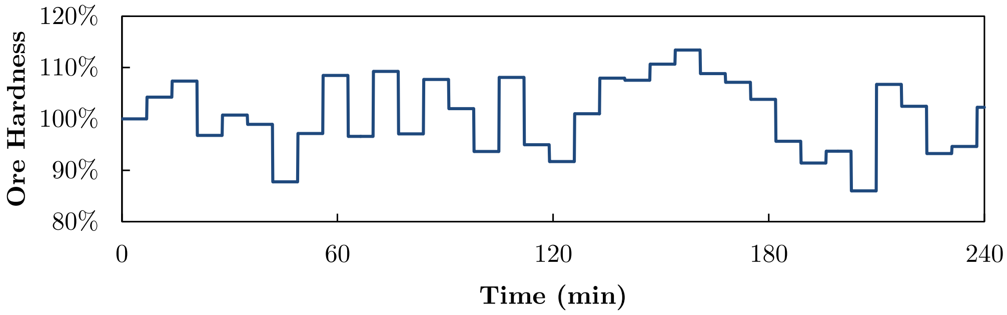

Figure 4.14: Example of the ore hardness disturbances.

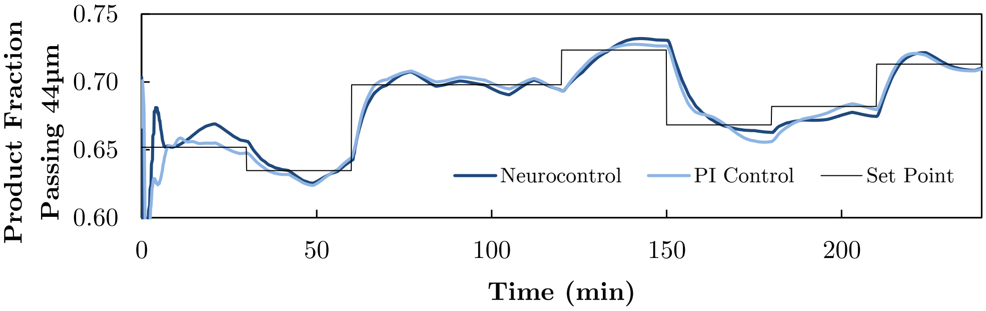

Figure 4.15: Set point tracking of the CMA-ES solution of the 7 × 2 × 4 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani & Herbst (1991*b*) under ore hardness disturbance.

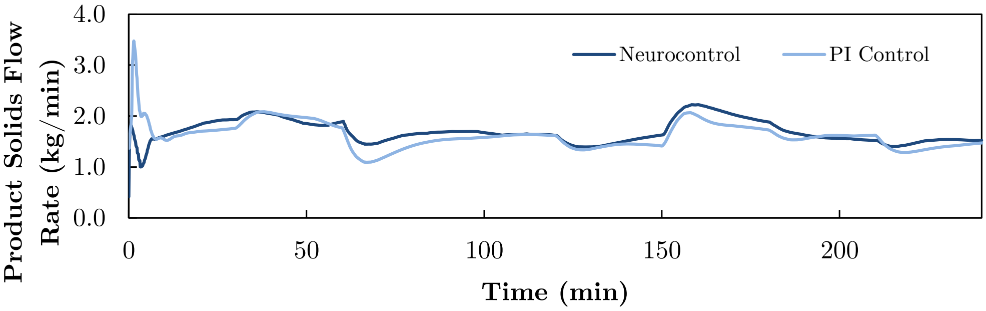

Figure 4.16: Product solids flow rate of the CMA-ES solution of the 7 × 2 × 4 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani & Herbst (1991*b*) under ore hardness disturbance.

4.5.2.5 Set Point Tracking - Combined Disturbance

When both types of disturbance are active, the performance takes a further hit, but this time the neurocontrol solution comes out on top. The results are shown in Figures 4.17 and 4.18. IAE ($t > 60$) was 16.8 for PI control and 16.1 for neurocontrol (4% better). The productivity was 6% higher under neurocontrol.

4.5.2.6 Constant Specifications - Combined Disturbance

The results when the product size fraction specification is set constant to 73% passing 44μm, and both disturbances are kept active are shown in Figures 4.19 and 4.20. PI outperforms the neurocontrol solution, with an IAE ($t > 60$) of 7.1 compared to 10.1.

Interestingly though, the productivity difference is even more pronounced under these conditions - 8% more ground limestone is produced by the neurocontrol solution, and consistently so, as Figure 4.20 would attest. The neurocontrol solution produced 359 kg ground limestone, compared to the 332 kg of the PI control

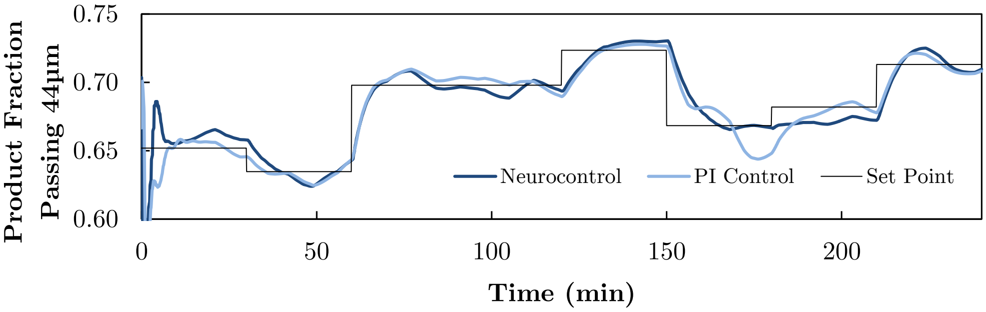

Figure 4.17: Set point tracking of the CMA-ES solution of the 7 × 2 × 4 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani & Herbst (1991*b*) under both ore hardness and feed size distribution disturbance.

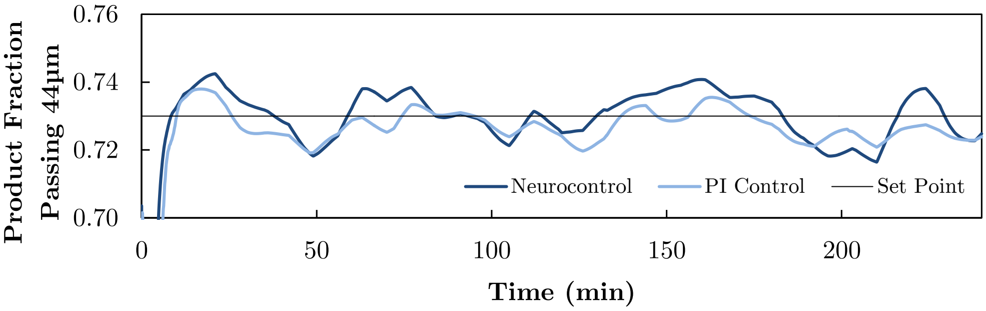

Figure 4.18: Product solids flow rate of the CMA-ES solution of the 7 × 2 × 4 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani & Herbst (1991*b*) under both ore hardness and feed size distribution disturbance.

solution.

4.5.2.7 Performance Improvement

How can one improve neurocontrol performance without retraining the controller? There is one variable that could be altered to potentially improve performance - the integral error gain of the neurocontroller. The variable is normally set constant to 1 - it is not regarded as one of the neurcontroller parameters. Increasing it could potentially increase the controller reaction speed, in the same way that decreasing the integral time of a PID controller would result in more 'aggressive' (but potentially unstable) control.

Figures 4.21 to 4.24 show the IAE for $t > 60$ under the different disturbance configurations for different integral gain values. The results are summarized in Table 4.8.

The data show a significant increase in performance in all cases. A 40% to 70% increase in integral gain would substantially improve performance, depending on the

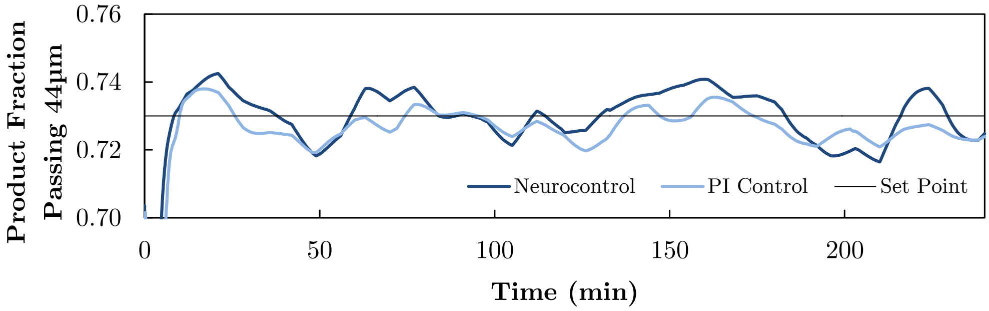

Figure 4.19: Constant specification performance of the CMA-ES solution of the 7 × 2 × 4 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani *&* Herbst (1991*b*) under both ore hardness and feed size distribution disturbance.

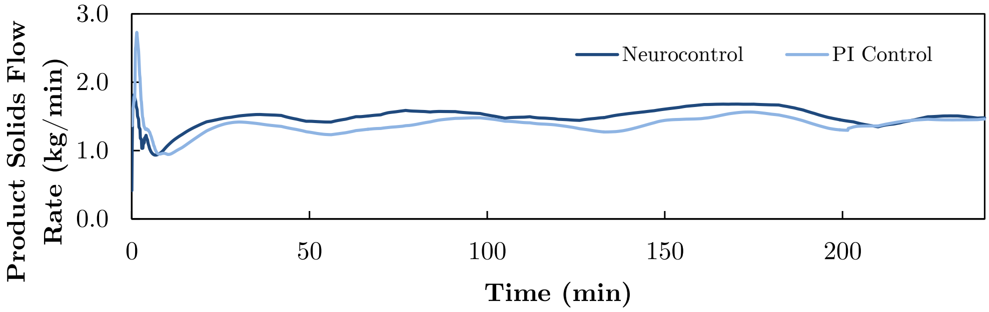

Figure 4.20: Constant specification productivity of the CMA-ES solution of the 7 × 2 × 4 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani *&* Herbst (1991*b*) under both ore hardness and feed size distribution disturbance.

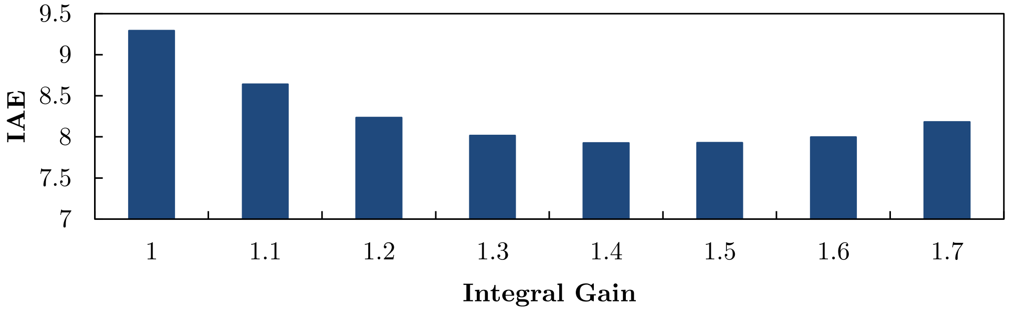

Figure 4.21: IAE ($t > 60$) of neurocontrol solution when no disturbances are present for different values of the integral gain. Default value is 1.0.

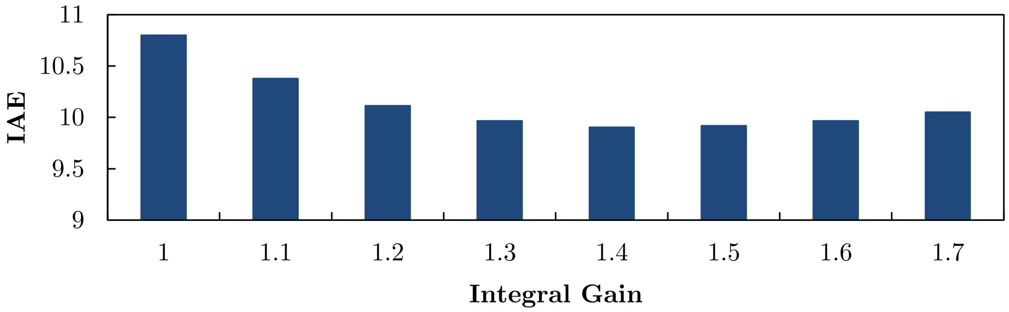

Figure 4.22: IAE ($t > 60$) of neurocontrol solution when feed size distribution disturbances are present for different values of the integral gain. Default value is 1.0.

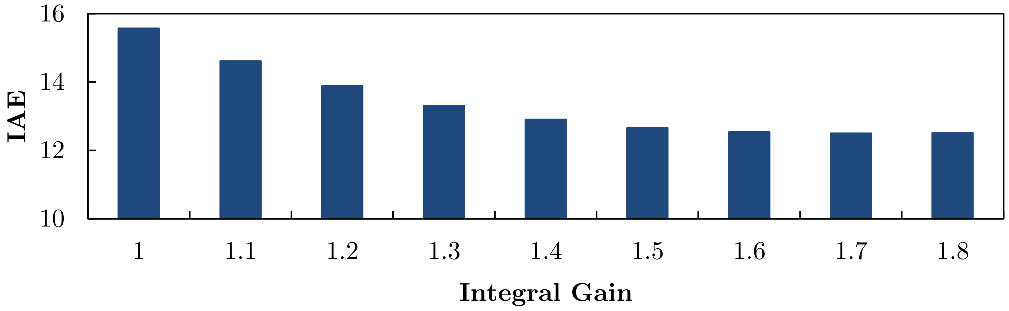

Figure 4.23: IAE ($t > 60$) of neurocontrol solution when ore hardness disturbances are present for different values of the integral gain. Default value is 1.0.

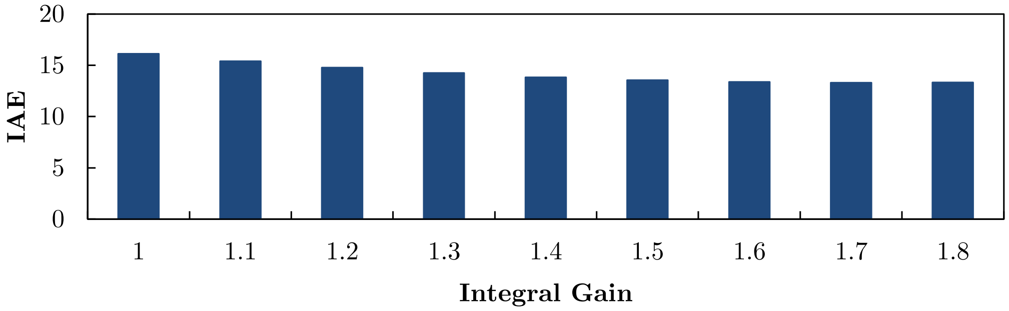

Figure 4.24: IAE ($t > 60$) of neurocontrol solution when both types of disturbance are present for different values of the integral gain. Default value is 1.0.

Table 4.8: Comparison of best performance improvement for different disturbance types.

Disturbance	PI IAE	Normal NN IAE	Best NN IAE (gain)	Improvement
None	9.8	9.2	7.9 (1.4)	14.7%
Feed size distribution	9.9	10.7	9.9 (1.4)	8.3%
Ore hardness	14.3	15.5	12.5 (1.7)	19.7%
Both	16.8	16.1	13.3 (1.7)	17.5%

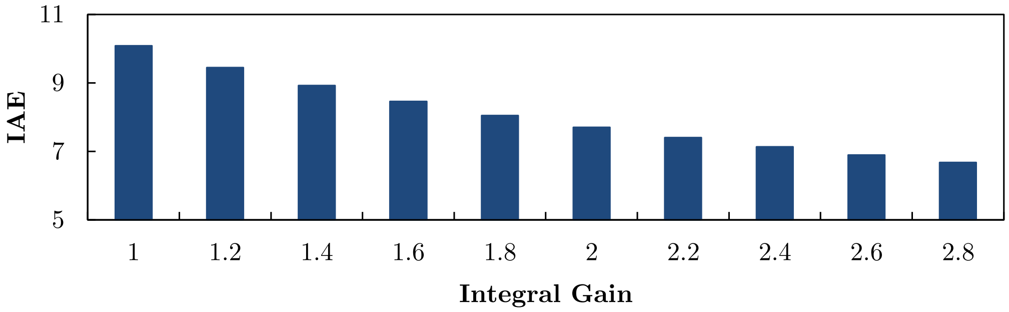

Figure 4.25: IAE ($t > 60$) of neurocontrol solution when both types of disturbance are present and no set point changes are made for different values of the integral gain. Default value is 1.0.

type of disturbance the plant is experiencing.

Figure 4.25 shows an interesting trend for the case where no set point changes are being made (product specification is maintained constant at 73% passing 44 μm) - performance continues to increase until the integral gain had been increased almost 3 times its normal value. The best IAE ($t > 60$) was 6.7 at a gain of 2.8 - any higher gain values caused the system to become unstable. The IAE is 34% better than under nominal circumstances, and 6% better than the PI control solution. In addition, productivity did not suffer too badly, reaching 351 kg, 6% higher than that of the PI control solution.

If integral gain is properly adjusted, neurocontrol of the ball mill grinding circuit will perform superior to PI control under all the circumstances that were tested.

It should be noted that, similarly to decreasing the integral time of a PID controller, increasing the integral gain in this way will eventually cause instability due to the increase in both amplitude ratio and phase lag. This is supported by most of the preceding figures, where IAE starts to increase with a high enough gain. In addition, all the simulations also eventually encounter instability around the same time the IAE starts to increase.

4.5.3 Discussion

The preceding results show that non-linear multi-variable feedback control using neural networks can outperform even highly optimized multi-loop PI control. However, the improvement in performance is only obvious when additional optimization is done by increasing the integral gain of the neural networks. Without such optimization, the PI control solution performed better than neurocontrol in two out of the five cases.

An advantage that the neurocontrol solution consistently showed was the higher production rate it offered. In all cases, at least 5% higher mill productivity was achieved, even when set point tracking performance was better than under PI control. This shows the non-linear optimization achieved by reinforcement learning on a plant-wide scale. Once the problem is well-defined with a representative reward function, it is left to the learning agent to achieve this goal. No tuning, apart from adjusting the integral gain, was necessary to find a 'sweet spot' where production rate and set point tracking are equally well achieved.

Why does neurocontrol only show the modest performance improvement under nominal circumstances? Two reasons come to mind.

Firstly, some of the blame could be placed on the fact that the optimization problem relied solely on a single performance measure - the reward function (Eq. 4.4.1). One can see from the figures in the previous subsection that when the product size specification is *increased*, the production rate *decreases*.

This means that the performance measure contains two (somewhat) mutually exclusive goals - the production rate and the product specification. Clearly, due to the inherent nature of the ball milling circuit, the production rate necessarily needs to decrease (meaning a lower value for Equation 4.4.1) if a higher specification is to be fulfilled (meaning a higher value for Equation 4.4.1). This could therefore be seen as a multi-objective optimization problem. Looking at it in this way, the performance showed by both reinforcement learning algorithms is satisfactory overall.

Secondly, just like the multiloop PI control solution of Rajamani *&* Herbst (1991*b*), the neurocontroller only operates in a *reactive* feedback control mode. It does not have any predictive mechanism in terms of anticipating future changes, and so its only advantages over PI control are its non-linearity and adaptive capabilities. It will therefore encounter the same problems as PI control, e.g. dead time, measurement delays, disturbances, etc.

The results show that in cases where a plant is well-modeled and a quick and easy performance improvement over multi-loop PID control is required, neurocontrol achieved by reinforcement learning can possibly provide the answer.

The next section looks at what could be done when no model of the plant exists, but some control data are available.

4.6 Data-Based Learning

In order for neurocontrol using reinforcement learning to be more applicable to real-world systems, it has to have the ability to learn from previous plant data. Since the two reinforcement learning algorithms used here require interactive experience to learn, this data would need to be represented as a dynamic model.

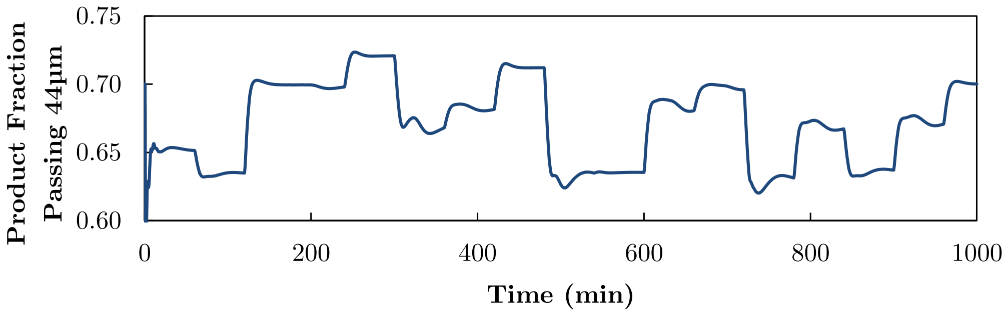

Figure 4.26: Example of the changing product specification of the training data used for building the NARX model.

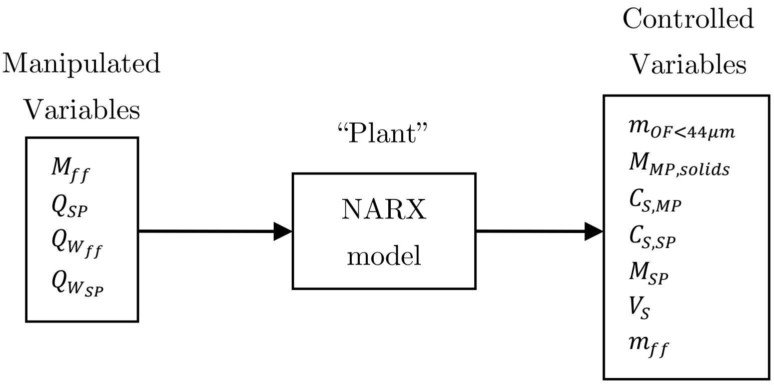

Figure 4.27: Illustration of the input-output mapping provided by the NARX model.

This section is split into three parts. The first part discusses system identification of the ball mill grinding circuit from PI control data and presents the relevant results. The second part presents the learning results of CMA-ES on the identified plant model. The last part presents the control performance results of the data-based controllers on the original plant model.

4.6.1 System Identification

Non-linear autoregressive models were used to model the ball mill grinding circuit (see Section 3.6). The model building and verification was done on time-series data from two 1000-minute PI control runs, where the set point was changed every 60 minutes. Figure 4.26 shows an example of the changing product fraction passing 44μm over 1000 minutes for the training data set. The time interval of the training and verification data was 0.1 minutes, giving 10,000 data points for each variable.

The models were built to predict the controlled variables (Table 4.2, page 26) as well as mill product flow rate m_{ff} based on the manipulated variables (Table 4.1, page 25) (minus mill rotation speed N which was constant at 0.8). Thus the total number of input variables was four, and the total number of output variables seven. The input-output format of the final model, showing the variables involved, is shown in Figure 4.27.

Two non-linearities showed the best results: the wavelet network and tree partition.

The wavelet network non-linearity is based on the function expansion given in Equation (4.6.1), where f is a scaling function and g is the wavelet function (The Mathworks, Inc., 2011). P and Q are m-by-p and m-by-q projection matrices, respectively, determined by principal component analysis of estimation data. The variable r is a 1-by-m vector representing the mean value of the regressor vector computed from estimation data, d, a_s, b_s and b_w are scalar scaling and wavelet parameters. L is a p-by-1 vector, and c_s and c_w are 1-by-q vectors. Both the scaling function f and wavelet functions are radial functions.

$$\begin{aligned} F(x) = (x-r)PL &+ a_{s1}f((b_{s1}(x-r))Q - c_{s1}) + ... \\ &+ a_{sn}f((b_{sn}s(x-r))Q - c_{sn}) \\ &+ a_{w1}g((b_{w1}(x-r))Q - c_{w1}) + ... \\ &+ a_{wn}wg((b_{wn}w(x-r))Q - c_{wn}) + d \end{aligned} \tag{4.6.1}$$

The tree partition defines a non-linear function $y = F(x)$ where F is a piecewise-linear function of x, y is a scalar and x is a 1-by-m vector. F is a local linear mapping where x-space partitioning is determined by a binary tree. The binary-tree network function is based on the function expansion given in Equation (4.6.2), where x belongs to the active partition D_a. D_k is a partition of x-space, L is a 1-by-m vector, C_k is a 1-by-$(m+1)$ vector and d is a scalar (The Mathworks, Inc., 2011). The active partition D_a is calculated as an intersection of half-spaces by a binary tree.

$$F(x) = xL + [1, x]C_a + d \tag{4.6.2}$$

The reader is referred to the MATLAB documentation for more information regarding the regression used in the system identification toolbox (The Mathworks, Inc., 2011).

When building the NARX models, several different parameters could be adjusted to improve model fit for each output variable. These are the number of previous input and output terms and the delay of each input term, each of which could be adjusted separately for each predicted (output) variable. In all cases the delays were set to zero. *All* manipulated variables were considered as inputs to each output variable, while previous output terms were only for the same predicted (output) variable.

The choice of the number of units to use in the non-linear block in the NARX estimation was set to auto for both tree partition and wavelet network non-linearities. No manual pre-processing of the input data was necessary for successful model building.

Since different variables are predicted better by different combinations of non-linearities and number of input and output terms, each combination was tested and compared. While the number of previous terms for each input variable could be adjusted separately, they were all set to the same value in the search for the best solution. To make the process feasible, the number of input or output terms were

Table 4.9: Comparison of best fitting nonlinearity and number of previous input and output terms used in prediction for each output.

Output	Non-linearity	No. of Prev. Input Terms	No. of Prev. Output Terms	Training Fit	Verification Fit
$m_{OF<44\mu m}$	Tree partition	1	4	96.2%	92.0%
$M_{MP,solids}$	Tree partition	4	1	95.1%	79.4%
$C_{S,MP}$	Wavelet network	4	4	96.5%	95.4%
$C_{S,SP}$	Wavelet network	3	3	92.3%	88.8%
M_{SP}	Tree partition	4	1	97.9%	93.8%
V_S	Tree partition	5	5	98.3%	85.6%
$M_{OF,solids}$	Tree partition	2	2	94.2%	87.0%

limited to between 1 and 5. The final best result as found by trial and error is shown in Table 4.9.

The model used to train the controllers is as seen in Table 4.9 - tree partition non-linearity for the first two and last three outputs, and a wavelet network for the two in between. This combination shows an average fit of 94.9% on the training data and 88.9% fit on the verification data.

4.6.2 NARX Model-Based Learning

The training was done in much the same way as with model-based learning, but one significant flaw was found when these trained models were applied again to the 'real' system (i.e. not the NARX model, but the model of Rajamani *&* Herbst (1991*b*) from which the NARX model was derived). The integrator in the system, being the sump level, was not being properly controlled by the data-based neurocontroller.

An example of the situation is shown in Figure 4.28. The figure shows three level trajectories. The light blue trajectory is for the desired behaviour of level control, termed "Model Based" since that is how the model based controllers of the previous subsection behave. As one can see, after about 45 minutes the level mostly stabilizes at around 0.05 m^3, with good level control having been achieved. The darkest blue trajectory shows the level behaviour of the data-based control solution when applied on the NARX model, termed "Data-Based (NARX)". This shows what the reinforcement learning agent would think it is achieving after training on the NARX model. As the figure shows, the NARX model predicts that it would maintain level almost constant at around 0.02 m^3. However, as shown by the medium blue trajectory (termed "Data-Based (REAL)"), the level behaviour of the same data-based control solution when applied on the 'real' system, the sump pumping rate Q_{SP} as prescribed by the data-based controller is too low in practice. This causes the level to skyrocket to over 500 litres within 20 minutes.

Since system identification failed to capture the sump level dynamics accurately enough (85.6% fit on the verification data), this result, while unfortunate, makes sense. What level of fit would remove this "integrator problem"? It is hard to say, but considering that an integrator keeps on adding or subtracting as time increases, any inaccuracy in the modelling of an integrator would end up being magnified to unacceptable levels.

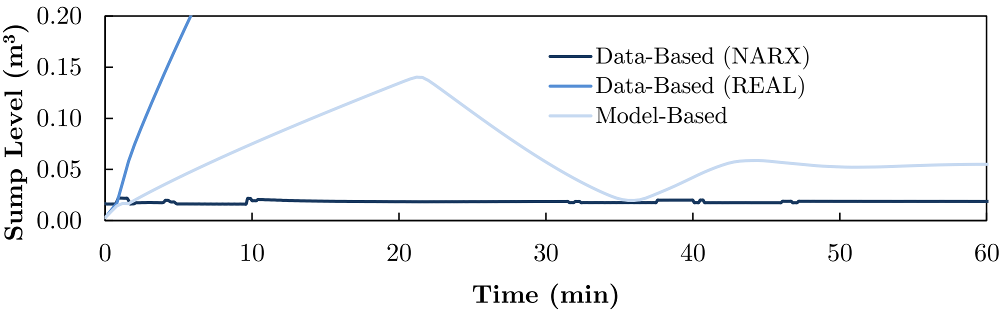

Figure 4.28: Sump level comparison for the data-based controller on both the NARX and ’real’ model compared to the sump level as controlled by the $7 \times 2 \times 4$-TanH model-based controller.

Table 4.10: Best and average fitness out of five learning runs of 5,000 episodes on the NARX model with sump level controlled externally.

	Number of Hidden Units		
Property	2	3	4
Best Fitness	90,204	84,758	85,736
Average Fitness	113,962	94,978	92,884

Table 4.11: Neurocontroller IAEs for $t > 60$ and productivities on the benchmark problem as used in the model-based learning section.

	Number of Hidden Units		
Property	2	3	4
IAE	9.3	12.4	10.9
Production	370 kg	379 kg	377 kg

In order to remove the integrators problem, it was decided to control the sump level in the same way as the PI approach - make the sump pumping rate (Q_{SP}) proportional to the sump level (V_S), thus removing it from the responsibilities of neurocontrol and putting it in its own control loop. Based on the model-based learning results, it was decided to train the controllers only with CMA-ES, and to give every controller the TanH output transformation. Also, since the smaller neurocontrollers performed perfectly well, this comparison was limited to three numbers of hidden units - 2, 3, and 4. Because sump level was not being controlled, it was removed from the inputs, and of course sump pumping was no longer an output, and thus the number of inputs and outputs were reduced to 6 and 3 respectively.

The learning results are presented in Table 4.10.

Once again, there is little difference in best fitness between the different neurocontrollers, but this time a lot more variance in fitness between runs was seen, although this is not evident from the slightly higher average fitness values. When tested on the dynamic model, most of the poorer-performing runs show no set point tracking ability, with the product fraction passing 44μm staying constant at an in-between value.

The neurocontroller that performed best out of the batch was the $6 \times 3 \times 3$-TanH controller with a best fitness of 84,758. The $6 \times 4 \times 3$-TanH neurocontroller is in a very close second place at 85,736. As with model-based learning, these controllers are first tested on a 240-minute simulation to see which gives the best IAE and productivity. Once again, the IAE was only calculated from time > 60, in order to allow the systems to stabilize.

The results (Table 4.11) show that all of the data-based controllers have lower productivity than the model-based ones (which all achieved between 400 and 410 kg), with values mostly on par with the PI controller results. This is most probably be due to plant-model mis-match.

The $6 \times 2 \times 3$-TanH controller has the best IAE, but a far worse productivity than the others. The $6 \times 3 \times 3$-TanH controller has a better productivity, but not much less than the $6 \times 4 \times 3$-TanH controller, which shows superior set point tracking.

As the most reasonable compromise between set point tracking performance and productivity, the $6 \times 4 \times 3$-TanH configuration is chosen for the control performance comparison in the following subsection.

4.6.3 Control Performance

In order to better visualize the performance in comparison to that of PI control, it will be tested in the following subsections in the same way as the model-based controller - set point tracking, disturbance response, and performance improvement by increasing the integral gain.

4.6.3.1 Set Point Tracking - No Disturbance

The set point tracking of the data-based neurocontrol solution is shown in comparison to PI control in Figure 4.29. Steady-state offset is clearly visible for the neurocontrol case from 120 to 180 minutes into the test. A very large transient response is also seen near the beginning of simulation, but disappears after 60 minutes. This is due to the initialization of the model, where the starting state is quite far from the eventual operating conditions as brought about by neurocontrol.

The production rate offered by the data-based neurocontrol solution is very similar to the PI control case, as seen in Figure 4.30. In fact, the neurocontrol solution has a production of 377 kg compared to the 384 kg of the PI control solution, meaning a 2% reduction in production going from PI to neurocontrol.

4.6.3.2 Set Point Tracking - Feed Size Distribution Disturbance

Figure 4.31 shows the set point tracking of the data-based neural network in the presence of feed size distribution disturbances. The controller seems to do relatively well, and integration of the error shows an IAE for $t > 60$ of 10.9 for the data-based

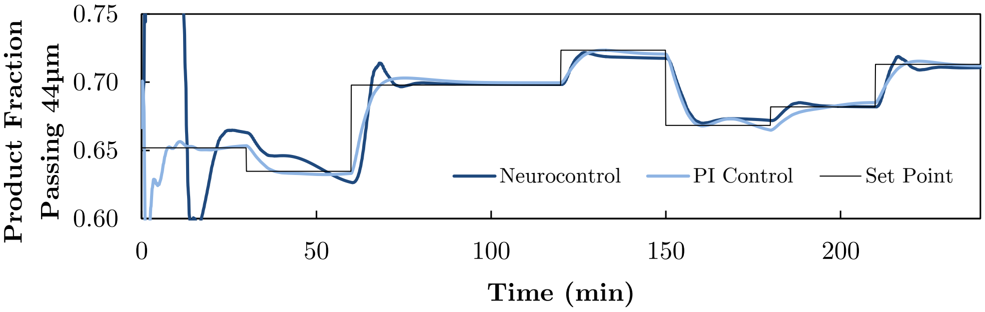

Figure 4.29: Set point tracking of the data-based 6 × 4 × 3 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani *&* Herbst (1991*b*).

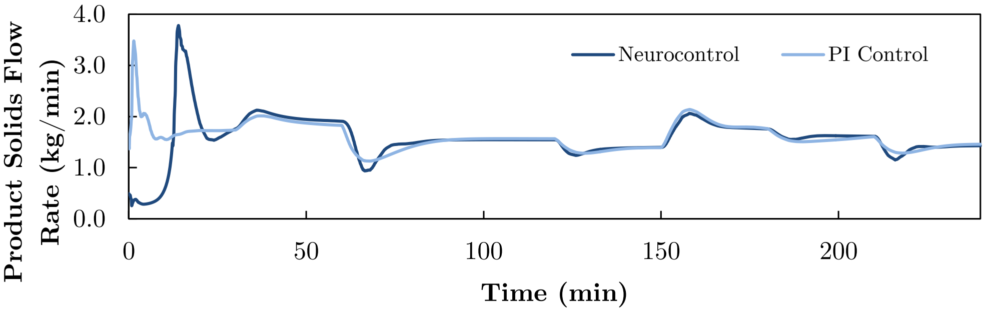

Figure 4.30: Product solids flow rate of the data-based 6 × 4 × 3 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani *&* Herbst (1991*b*).

controller, which compares with the previous value for the model-based controller of 10.7, and the PI controller, 9.9.

Figure 4.32 shows very little difference in production rate between the PI and data-based neural control method. Integrating the flow rate over the 240 minute simulation shows that PI control is more productive with 387 kg, compared to only 382 kg for the neurocontrol solution, a 2% difference.

4.6.3.3 Set Point Tracking - Ore Hardness Disturbance

Introducing only ore hardness disturbances (Figure 4.33) shows an interesting result - the data-based controller does even better than the model-based controller, with an IAE ($t > 60$) of only 12.1, compared to 14.3 for the PI controller and 15.5 for the model-based controller.

However, as evidenced by Figure 4.34, production rate is relatively low, reaching 379 kg, 2% lower than the PI controller at 386 kg.

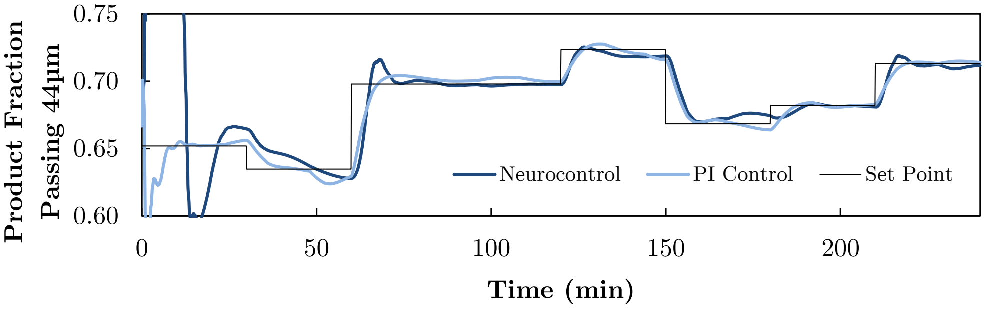

Figure 4.31: Set point tracking of the data-based 6 × 4 × 3 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani & Herbst (1991*b*) under feed size fraction distribution disturbances.

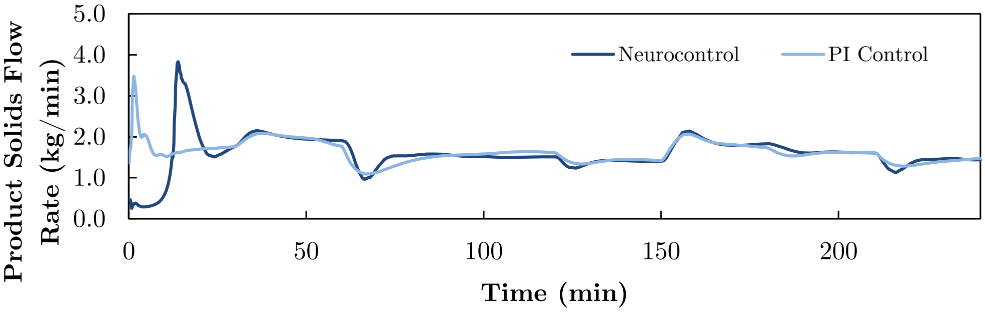

Figure 4.32: Product solids flow rate of the data-based 6 × 4 × 3 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani & Herbst (1991*b*) under feed size fraction distribution disturbances.

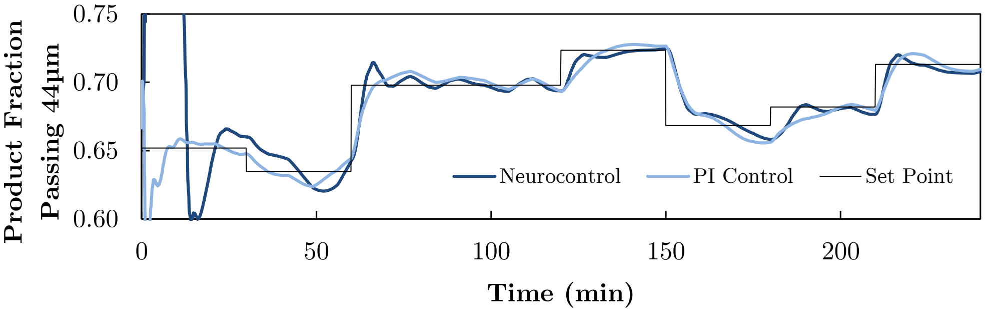

Figure 4.33: Set point tracking of the data-based 6 × 4 × 3 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani & Herbst (1991*b*) under ore hardness disturbance.

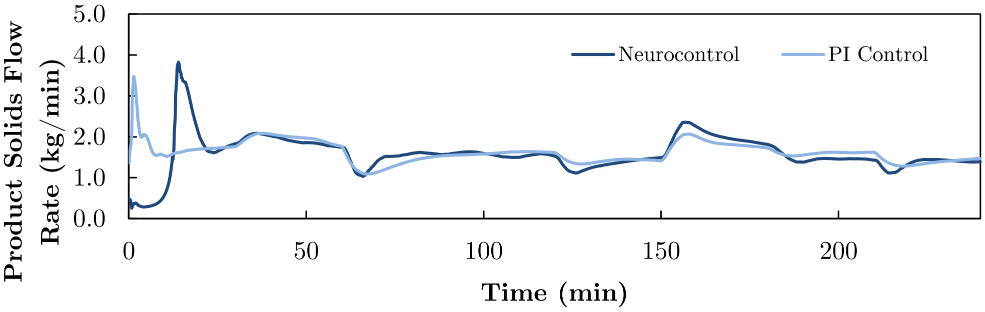

Figure 4.34: Product solids flow rate of the data-based 6 × 4 × 3 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani *&* Herbst (1991*b*) under ore hardness disturbance.

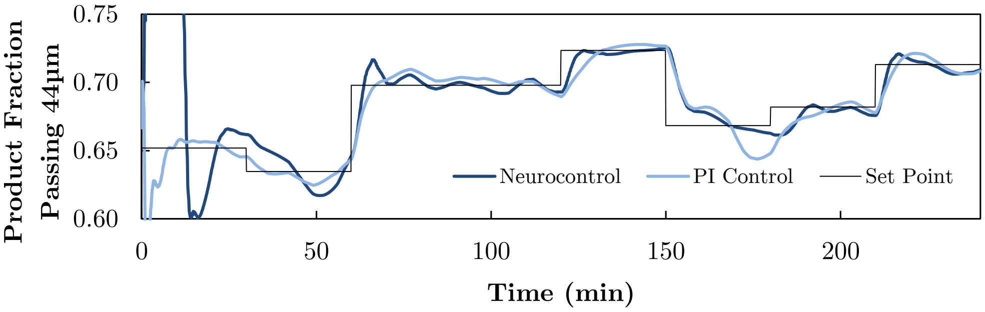

Figure 4.35: Set point tracking of the data-based 6 × 4 × 3 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani *&* Herbst (1991*b*) under both ore hardness and feed size distribution disturbance.

4.6.3.4 Set Point Tracking - Combined Disturbance

Introducing both disturbances at the same time (Figures 4.35 and 4.36) results in an even lower IAE than when either disturbance was encountered separately, with the two perhaps canceling out one-another in the case of the data-based neurocontroller. IAE ($t > 60$) for the data-based controller in this case is 12.0, compared to 16.8 for PI control and 16.1 for the model-based controller.

Once again, production rate is lower than both model-based neurocontrol and PI control, clocking in at 384 kg over the 240 minutes, compared to 388 kg for PI control and 411 kg for model-based neurocontrol.

4.6.3.5 Constant Specifications - Combined Disturbance

The results when the product size set point is kept constant at 73% passing 44μm are shown in Figures 4.37 and 4.38. The neurocontrol solution seems to be struggling slightly more than the PI control solution at keeping the product specification constant, and no superior productivity is shown. The IAE ($t > 60$) was 8.6 for the

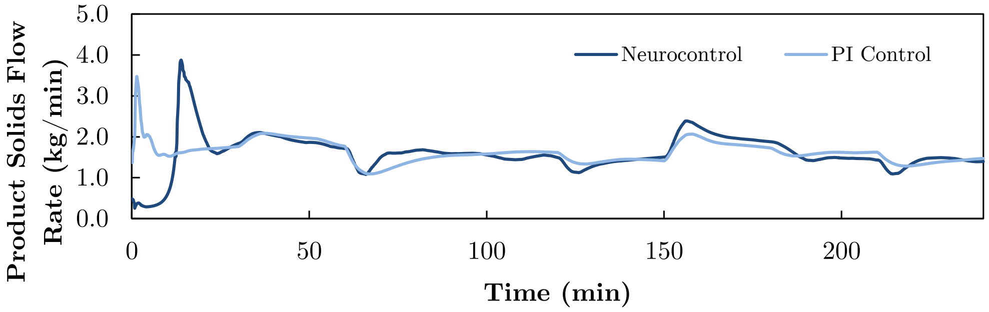

Figure 4.36: Product solids flow rate of the data-based 6 × 4 × 3 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani *&* Herbst (1991*b*) under both ore hardness and feed size distribution disturbance.

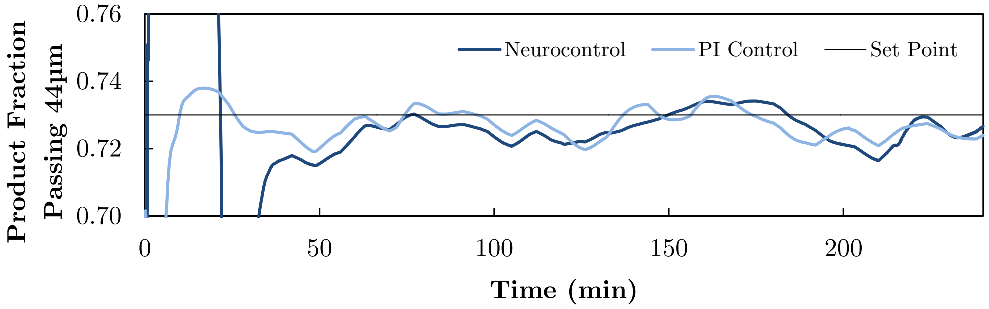

Figure 4.37: Constant specification performance of the data-based 6 × 4 × 3 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani *&* Herbst (1991*b*) under both ore hardness and feed size distribution disturbance.

data-based neurocontroller, which is better than what model-based neurocontroller could achieve (10.1), but is still not as good as the 7.1 for PI control. Productivity was slightly better than PI control this time, at 335 kg, compared to 332 kg for PI control.

4.6.3.6 Performance Improvement

Figures 4.39 to 4.42 show the improvement in IAE of the data-based 6 × 4 × 3 neurocontroller when the integral gain is adjusted. Table 4.12 summarizes the results. A quick look back to Table 4.8 (page 44) will show that the gain in performance is much more modest in this case when compared to the model-based controller. This could be because every trained neurocontroller is unique, and some have a higher built-in integral gain from the neural weights corresponding to the integral gain input. Another explanation could be that training on the NARX model makes the controller naturally more 'aggressive' due to some internal model characteristic.

Another difference between these results and the results of the model-based

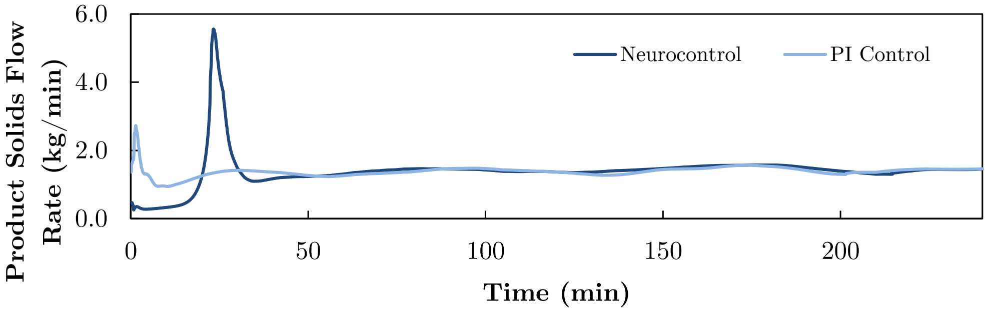

Figure 4.38: Constant specification productivity of the data-based 6 × 4 × 3 with TanH output neurocontroller compared to the optimized PI control solution of Rajamani & Herbst (1991*b*) under both ore hardness and feed size distribution disturbance.

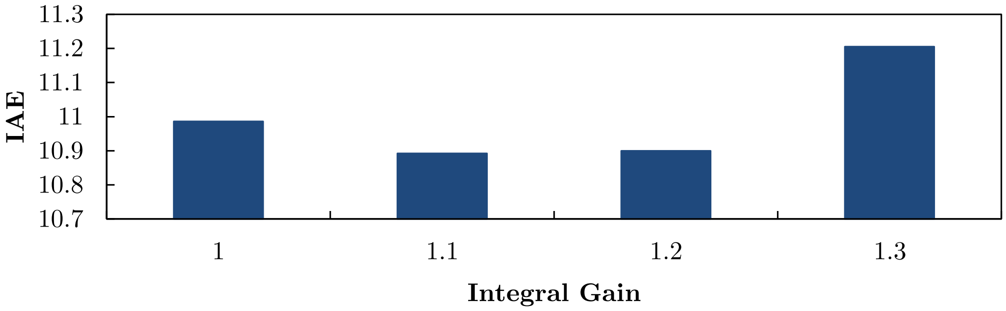

Figure 4.39: IAE ($t > 60$) of the data-based neurocontrol solution when no disturbances are present for different values of the integral gain. Default value is 1.0.

controller is the smaller increase in integral gain required to achieve the optimal level of control. As stated above, this must be because the controller already has a relatively high internal integral gain.

Figure 4.43 shows the improvement in IAE for the case where product specification is kept constant at 73% passing 44μm in the presence of both product size distribution and ore hardness disturbances. Performance increases moderately up to a gain of 1.6, where excessive oscillation of the product size passing 44μm starts to manifest. Just like for the case of model-based optimization, this can be explained by the increasing amplitude ratio and phase lag caused by the increasing integral action. An increase in gain to 1.4 could be taken as the optimal choice, where an IAE of 5.8 is better than both the model-based performance of 6.7 at a gain of 2.8 and the PI control performance of 7.1.

4.6.4 Discussion

The results show that a rigorous first-principles model is not necessary for training a neural network controller using reinforcement learning. Instead, as long as

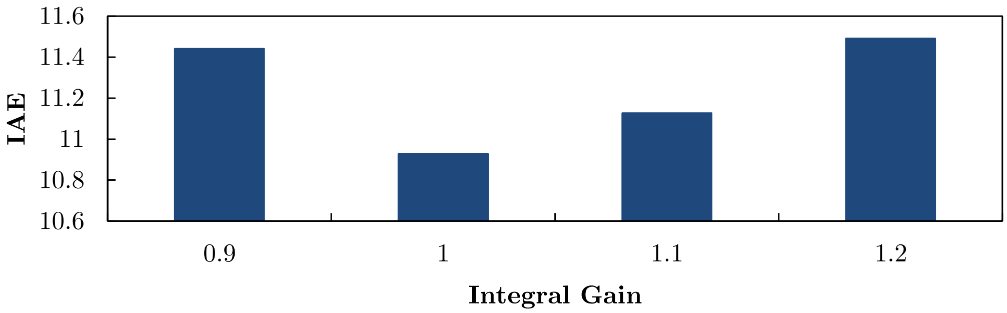

Figure 4.40: IAE ($t > 60$) of the data-based neurocontrol solution when feed size distribution disturbances are present for different values of the integral gain. Default value is 1.0.

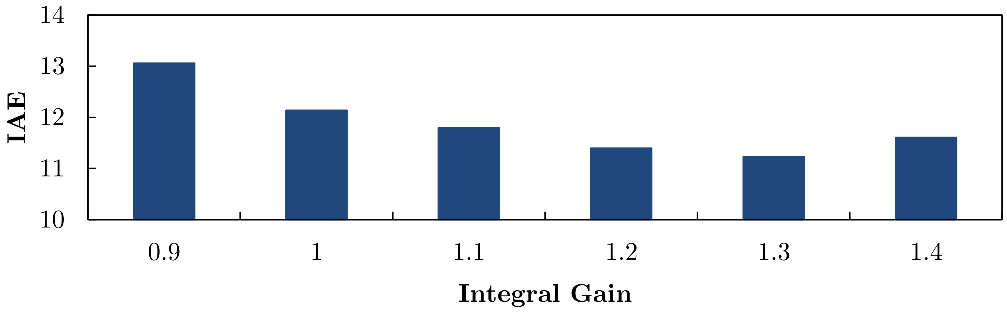

Figure 4.41: IAE ($t > 60$) of the data-based neurocontrol solution when ore hardness disturbances are present for different values of the integral gain. Default value is 1.0.

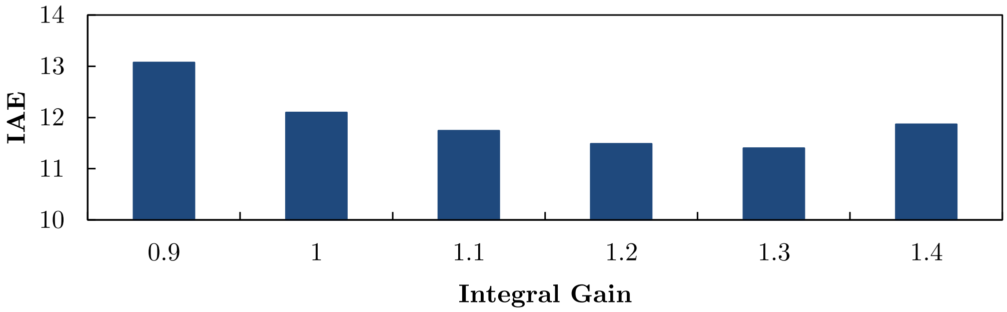

Figure 4.42: IAE ($t > 60$) of the data-based neurocontrol solution when both types of disturbance are present for different values of the integral gain. Default value is 1.0.

Table 4.12: Comparison of best performance improvement for different disturbance types. MB is for Model-Based and DB is for Data-Based. The model-based best results are included for easier comparison.

Disturbance	PI IAE	Best MB NN IAE (gain)	Normal DB NN IAE	Best DB NN IAE (gain)	DB Improvement
None	9.8	7.9 (1.4)	11.0	10.9 (1.1)	0.9%
Feed size	9.9	9.9 (1.4)	10.9	10.9 (1.0)	0.0%
Ore hardness	14.3	12.5 (1.7)	12.1	11.2 (1.3)	7.4%
Both	16.8	13.3 (1.7)	12.1	11.4 (1.3)	5.7%

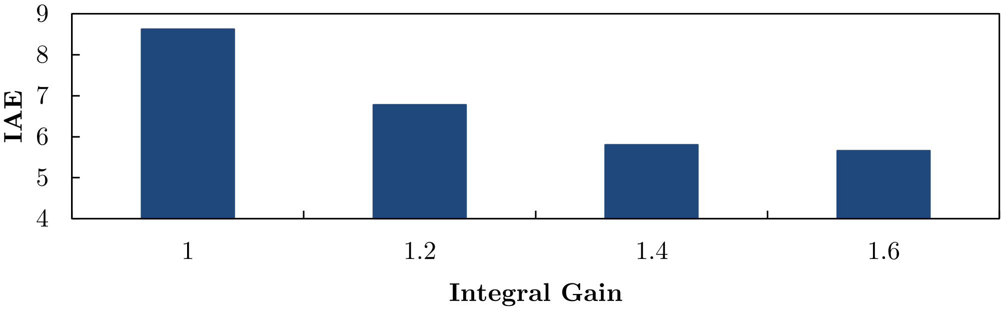

Figure 4.43: IAE ($t > 60$) of data-based neurocontrol solution when both types of disturbance are present and no set point changes are made for different values of the integral gain. Default value is 1.0.

representative plant input-output data are available, the potential for developing a neurocontrol solution exists.

The performance gains as measured by the IAE for $t > 60$ of the benchmark problem can be seen in Table 4.12. The results show that after adjusting the integral gain, the model-based controller can outperform the PI controller in all cases except when only ore feed size distribution disturbances are present, in which case the IAE is the same as for PI control. It also showed around 6% higher productivity in each case.

The data-based controller, which may be more representative of practical performance on a real plant, can outperform the PI controller in all cases except two: the ore feed size distribution disturbance case and when there are no disturbances present, in both cases having a 10% higher IAE. However, when either only ore hardness disturbances ore both disturbances are present, it performs even better than the model-based controller. A drawback of the data-based controller was its lower productivity as compared to the model-based and PI controllers, frequently producing around 1% less than under PI control. Further tests confirmed that this was a result of the model and not the loss of the sump level control loop. This was done by simply training the partial controller with no plant-model mismatch.

Given the performance of the data-based controller on the task, there is potential for real-world implementation of reinforcement learning-based neurocontrol to industrial processes, at least in the case of grinding circuits. However, one hurdle would make this a more difficult task - controller verification. The data-based controllers all perform well on the benchmark problem when the NARX model on which they are trained is used, but when it is switched for the 'real' model, some fail to replicate their performance. This represents what could happen when controllers trained on plant models identified from data are finally applied to the real plant.

The only remedy to the above problem is to train a large number of controllers on the data-based model, select the top 10%, and test them on the plant to see which perform best.

Another characteristic of the data-based solution that needs to be taken into consideration is its slightly lower stability, as can be seen by the higher peaks and overshoots for certain set point changes. While this was not an issue for the ball mill system, other plants may be less stable. This just underlines the importance of controller verification, which is a necessary hurdle in solving the data-based control problem.

The above problems notwithstanding, developing neurocontrollers using reinforcement learning from data-based models will remain a definite possibility and area of future research if the performance gains as seen in this study can justify it.

One possible way of further improving neurocontrol is to make it adaptive online, which is the topic of the next section.

4.7 Adaptive Control

On-line adaptive control is one of the major reasons for using neural networks for control in the first place. In the context of this work, adaptive control can be regarded as a controller that continually adapts to a changing or changed environment by exploring the space in which it finds itself.

This is much like a reinforcement learning agent, which is another reason why these algorithms are used here. As described in the literature survey in Section 2.4 on page 8, policy search algorithms, which both PGPE and CMA-ES are, are episodic in nature and therefore only update their control policy once an episode (or in the case of symmetric PGPE, two episodes) has been completed. An episode needs to have some constant length, which can be chosen at the discretion of the user.

Therefore, in order to effectively apply adaptive control with episodic reinforcement learning, the length of an episode needs to be chosen carefully, keeping the following in mind. Firstly, seeing as policy updates are based on (1) the way the policy was mutated or perturbed and (2) the subsequent return, episode length needs to be at least twice as long as the dead time and measurement delays in the system. Secondly, typically a few thousand episodes are needed to get to a proper problem solution, the episode length needs to be kept as short as possible without adversely affecting the quality of the reward feedback.

The question still remains whether the small amount of exploration that will be allowed in practice will be enough for a learning agent to improve itself, especially when noise and disturbances are present. The results presented in the coming subsections will try to shed some light on this matter.

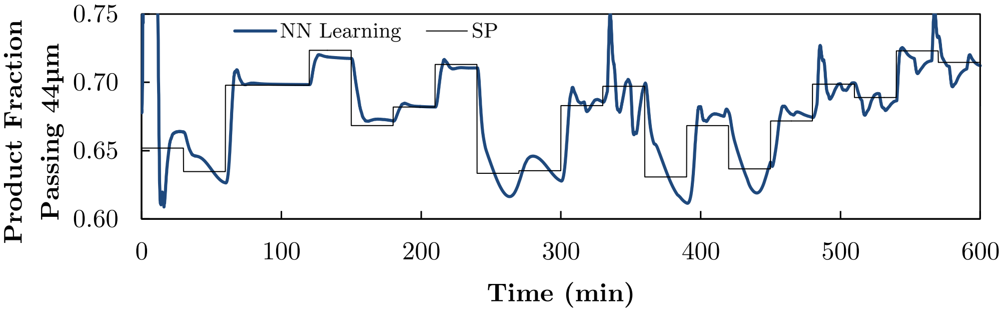

Figure 4.44: Example of the effect of on-line learning using PGPE when the episode length is set to 100 steps and exploration starting at $\sigma = 0.1$ for each parameter. Learning only starts at $t = 300$ minutes

4.7.1 On-line Learning - PGPE

The policy gradient reinforcement learning results are presented here. Learning was tested for four different episode lengths - 100, 500, 1000, and 2000 time steps of 0.1 minutes each.

The learning results presented here are based on improvement of the data-based $6 \times 4 \times 3$ with TanH output neurocontroller presented in the previous section. One of the areas where there is known room for improvement for this controller is in production rate, where it usually achieved around the same as the PI control solution and 6% less than the model-based neurocontrollers.

The effect of learning on control performance is neccesarily negative, since exploration needs to take place for the agent to learn anything. An example of the type of effect such learning has is presented in Figure 4.44 when episode length is set to 100 steps and exploration starting at $\sigma = 0.1$ for each parameter. For the first 300 minutes, learning is turned off, so that easier comparison can be made between non-learning and learning control.

The disturbances seen in the latter part of Figure 4.44 are due the change in neurocontroller parameters (being the exploration) when a new episode is started. These disturbances will be less frequent when episode length is increased.

Is the poor performance during learning made up for in neurocontroller improvement? This was answered by conducting simulations where learning was allowed for 20,000 minutes (14 days), no ore hardness or feed size distribution disturbances were present, and the set point was changed every 500 minutes. The results (Table 4.13) show that learning did not proceed as well as hoped. In fact, performance is poorer in each and every case by both IAE ($t > 60$) and production measures.

Why are the learning results negative? Are the starting positions stuck too deeply in local minima for the PGPE learning to get out? Could it be that the exploration being allowed was too small for proper learning to take place? Were more episodes needed?

To answer these questions, two additional simulations were done (with the episode length set to 1000 steps) - one where controller variance was increased (0.1 to 0.5), and another where learning was allowed for much longer (200,000 minutes instead

Table 4.13: Results for on-line adaptation using PGPE with exploration $\sigma = 0.1$ for each parameter for different episode lengths. Learning was allowed for 20,000 minutes (14 days) before testing on the usual benchmark problem used in the previous sections.

	Length of Episodes (steps)				
Property	No Learning	100	500	1000	2000
IAE	10.9	25.9	12.5	12.6	13.9
Production	377 kg	376 kg	375 kg	375 kg	374 kg

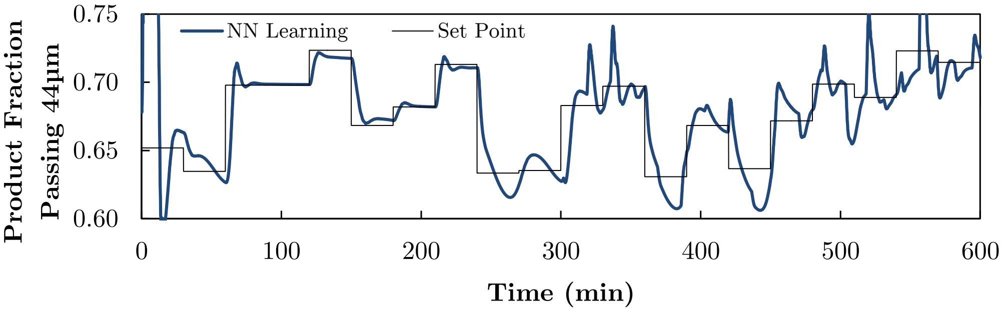

Figure 4.45: Example of the effect of on-line learning using CMA-ES when the episode length is set to 100 steps and a learning set size of 0.2. Offspring size was set to 4. Learning only starts at $t = 300$ minutes

of 20,000).

The first simulation proved that increased variance had no positive effect on learning in this case. An IAE of 19.8 was achieved, with a productivity of 373 kg, which means a further decrease in performance beyond the initial settings. The second simulation showed negative results for long-term learning, reaching an IAE of 15.1 and a productivity of 374 kg.

This results seems to show that, at least in the case of PGPE, on-line learning in an active control setting serves more to degrade controller performance than anything else.

In the following subsection the on-line learning performance of CMA-ES is investigated.

4.7.2 On-line Learning - CMA-ES

On-line learning using CMA-ES works much the same as with PGPE, except that the exploration variance is specified as the global learning step size. An example of the effect of CMA-ES learning on control peformance can be seen in Figure 4.45. The learning results under the same conditions as the PGPE learning is presented in Table 4.14, where the learning exploration was specified as a global step size of 0.2 and offspring count (λ) set to 4.

Table 4.14: Results for on-line adaptation using CMA-ES with a global step size of 0.2 and offspring size of 4 for different episode lengths. Learning was allowed for 20,000 minutes (14 days) before testing on the usual benchmark problem used in the previous sections.

	Length of Episodes (steps)				
Property	No Learning	100	500	1000	2000
IAE	10.9	10.9	10.9	11.9	12.3
Production	377 kg	377 kg	377 kg	375 kg	375 kg

While the results are looking better than for PGPE on-line learning, no meaningful improvement on the original controller can be seen. It seems that CMA-ES is less likely to degrade controller performance in the same way as PGPE. The same reasons for poor learning can be given here, especially the problem of already being too deep in a local minima, or perhaps being to far from the desired solution in the fitness landscape. It could be that on-line learning with the limited exploration that is being allowed (so that normal control can more-or-less continue), will never allow the learning agent to find a solution on the other side of the fitness landscape. In other words, the controller parameters that will allow higher production rate at the same level of set point tracking is too far out of reach for this particular neurocontroller.

It should be noted that neither PGPE or CMA-ES are specifically designed with adaptive control in mind. Adaptive control of open-loop stable processes is a highly specialized field, and therefore typically requires highly specialized solutions. The general nature of PGPE and CMA-ES, as seen here, will be no match for specialized solutions.

In order to see if CMA-ES can at least improve an artificially sub-optimal neurocontroller, the controller used above is perturbed with random noise, after which it achieves an IAE ($t > 60$) of 19.8 and productivity of 372 kg on the benchmark test. After several 20,000 minute learning runs with episode lengths of 100 and 500 steps, no improvement in IAE or productivity could be achieved. Both properties remained within less than 1% of their values before learning.

Another similar test was done on the model-based neurocontrollers. They were perturbed by random noise in order to decrease their performance, and CMA-ES was used to try and improve performance by on-line learning. In this case, however, the exploration seemed too severe at a global step size of 0.2 (all simulations bombed out with instability or divide by zero errors), and was decreased to 0.05. Nonetheless, no learning was achieved, and many of the simulations still failed to run their full 20,000 minutes.

4.7.3 Discussion

The non-performance of both reinforcement learning algorithms in an on-line adaptive setting is a major disappointment for their application in chemical processes. What this basically means is that the model-based and data-based controllers (with performance improvement as per integral gain adjustment) would need to perform very well as-is, and cannot rely on improvement on the plant once they are implemented.

As discussed in both preceding subsections, it could be that the starting neurocontrol solution is stuck to deep within a local minimum, or even too deep in a region of local minima. In other words, the controller is over-trained into a certain behavior, and the step-by-step learning done by episodic reinforcement learning with the small level of exploration available on a running plant is just not enough.

4.8 Conclusion

Neurocontrol of a ball mill grinding circuit using reinforcement learning can improve on existing multi-loop PI control methods. The level of improvement depends on whether on not an accurate process model of the system is available.

If an accurate plant model with little to no plant-model mismatch is available, the chances are good that overall increased performance will result. The results of the model-based tests showed that production increased by around 6% over that of PI control in all cases, which can be attributed to the plant-wide optimization achieved by evolutionary reinforcement learning on the neurocontroller. When additional optimization is done by adjusting the integral gain value, the neurocontroller was able to achieve more than 10% reduction in IAE of the product size set point, even in the presence of disturbances.

When only input-output data of an existing plant is available, such as when multi-loop PI control is being used, reinforcement learning can still train effective neurocontrollers. The success of this approach is dependent on the quality of the system identification, which in this case showed that an 88% fit on verification data for a NARX model was good enough. However, the data-based approach showed a weakness when integrators are present in the system. The ball mill grinding circuit has one integrator, being the sump level. It was found that the sump level control had to be removed to a separate proportional controller for neurocontrol based on the NARX model to succeed. If it was not removed, the sump level would rise to unacceptable levels once control was applied to the 'real' system, presumably since this dynamic is not modeled effectively enough.

Despite the integrator problem, the data-based neurocontroller achieved reasonable success on a benchmark problem, showing a reduction in IAE in the presence of disturbances to levels even lower than under the model-based neurocontroller. One drawback of this approach was the lower productivity it offered, typically achieving around 1% less than PI control and therefore also 6% less than the model-based neurocontrol solution.

In terms of the relative success of the two reinforcement learning algorithms, CMA-ES clearly outperformed PGPE by a large margin (see Table 4.5 on Page 32). This confirmed its high effectiveness compared to other episodic reinforcement learning algorithms, as found in previous literature (Heidrich-Meisner & Igel, 2009). In addition, it was shown that using smoother output transformation, e.g. by using the hyperbolic tangent function, both CMA-ES and PGPE were able to find better solutions more consistently. The effect of the smoother outputs can be seen as a kind of smoothing of the fitness landscape, which made a big difference during learning of good controllers. In addition, this output transformation showed much improved IAE values on a benchmark test problem, which is interpreted as the positive effect of a smoother control response resulting from the high-gradient-to-low-gradient nature of the TanH function.

Adaptive control using either reinforcement learning algorithms failed to achieve any measurable success. Both PGPE and CMA-ES was tested in various configurations, and it was found that learning could typically not proceed to levels better than the starting point. In other words, and especially in the case of PGPE, the learning effort typically resulted in a degradation in overall controller performance. An explanation could be that the starting controller positions were too deep in local minima to escape with the small level of exploration that was allowed. Any higher levels of exploration resulted in instability and were not suited for on-line control. The fact that neither CMA-ES nor PGPE are specifically designed for adaptive control further explains the poor performance. As mentioned before, open-loop adaptive control requires more specialized solutions.

The reason for the relatively modest performance gain under either model-based or data-based neurocontrol is the fact that they are still *reactive*, feed-back dependent approaches with no predictive capabilities. Their only advantage over traditional multi-loop PI control approaches are their non-linear multivariable and centralized control abilities. These abilities allow neurocontrollers to cope better with problems such as controller interaction, and allow global optimization as seen in the model-based control approach, where a 6% improvement in production of PI control was obtained. While it was hoped that their adaptive ability would also come to the fore in this study, this proved not to be the case when episodic PGPE or CMA-ES was used.

Chapter 5

Neurocontrol of a Distillation Column

5.1 Introduction

Distillation, where a feed mixture of two or more components is separated into two or more products, is the most common unit operation in the chemical industry (Skogestad, 1997; Seader & Henley, 2006). Distillation columns are systems, in that a column can be viewed as a set of integrated, cascaded flash tanks. This fact makes the distillation column a favourite subject area in the field of process systems engineering, including process control (Skogestad, 1997).

Distillation is also a highly energy-intensive operation, especially when the relative volatility (α) of the components being separated is low (< 1.5) (Seader & Henley, 2006). Distillation alone accounted for nearly 3% of the the entire United States energy consumption in 1976 (Seader & Henley, 2006), mostly by the petrochemical industry.

Due to their high energy consumption and widespread use, better control practice can go a long way in saving energy and bringing down production costs. Unfortunately, when both distillate and bottoms composition need to be closely controlled, severe interactions makes control difficult due to the highly non-linear nature of a typical distillation column. Therefore, alternate, centralized control approaches that implicitly deal with interaction become highly attractive. Neural networks, with their non-linear multivariable control abilities, are one such possibility.

This chapter aims to improve upon multi-loop PI control of a binary distillation column using the reinforcement learning-based neurocontrol approach. The work starts by describing the non-linear process model, "Column A", of Skogestad (1997), which will be used throughout this study. This is followed by a section on distillation feedback control, where the different PI control configurations are described. Centralized control is also touched upon. The specifics of the control solutions used in this study are described next, with a performance comparison of three different PI control solutions. This is followed by a section detailing the neurocontrol solution used here, looking at network topology and training details.

With the experimental specifics out of the way, the performance of the learning algorithms on the control problem is investigated, and the performance of learned

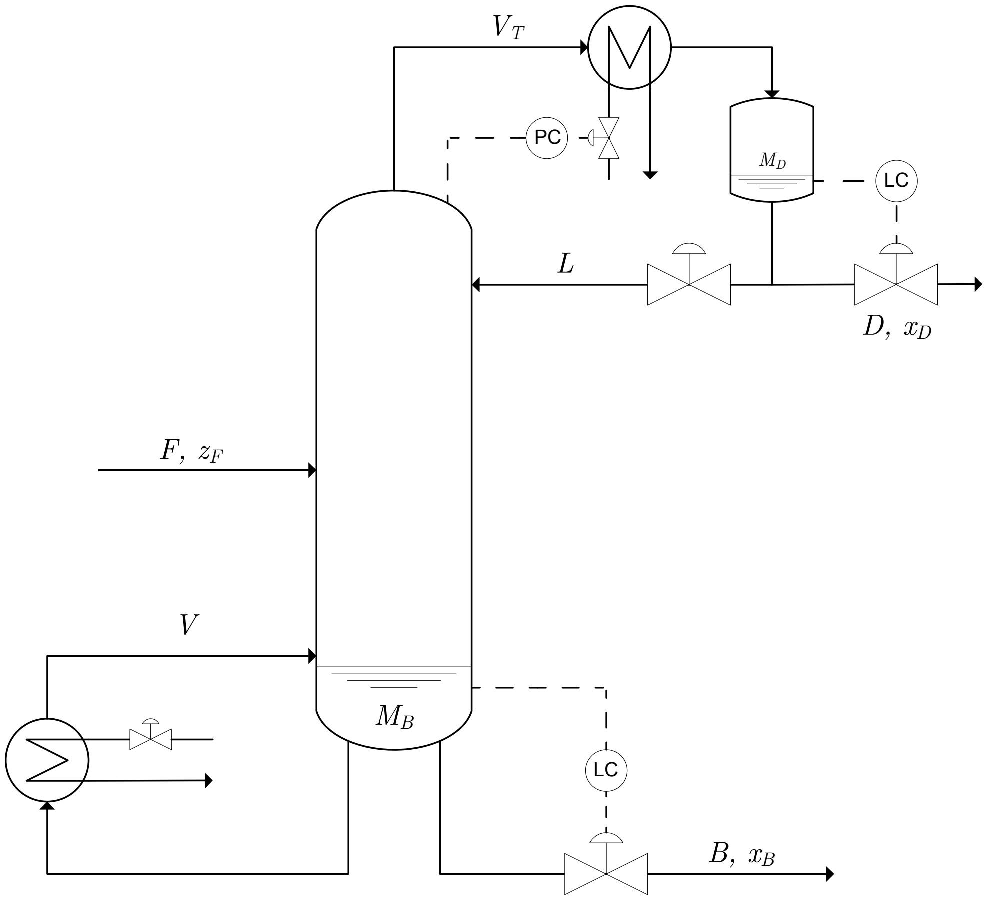

Figure 5.1: Distillation column with nomenclature as used throughout this study, with the LV control configuration. Reproduced from Skogestad (1997).

controllers is compared to the best PI control solution, both with and without the presence of disturbances. The results of system identification and learning is then presented, followed by an investigation into real-time on-line adaptation using reinforcement learning. The chapter ends with a conclusion of the results.

5.2 Process Model

The process model of Skogestad (1997) is used throughout this study. The model describes separation of a binary mixture, while assuming constant pressure, constant relative volatility, constant molar flows, no vapor holdup, linear liquid dynamics, equilibrium on all stages, and a total condenser. Despite these assumptions, the model maintains the important effects that will be required for a dynamics and control study.

A process flow diagram of this column is provided in Figure 5.1. The relevant process variables are explained in Table 5.1.

Holdup on the stages are not assumed to be constant (i.e. liquid flow dynamics are still included). Therefore when liquid flow is changed at the top of the column, the liquid flow into the reboiler changes over time according to the liquid flow

Table 5.1: Process variables and symbols including nominal (and initial) value.

Process Variable	Symbol	Nominal value	Unit
Feed rate	F	1.0	kmol/min
Feed composition	z_F	0.5	mol fraction
Feed liquid fraction	q_F	1.0	–
Distillate flow rate	D	0.5	kmol/min
Bottoms flow rate	B	0.5	kmol/min
Reflux flow	L	2.706	kmol/min
Boilup flow	V	3.206	kmol/min
Distillate composition	x_D	0.99	mol fraction
Bottom product composition	x_B	0.01	mol fraction
Vapour flow from stage i	V_i	–	kmol/min
Liquid flow from stage i	L_i	–	kmol/min
Liquid holdup on stage i	M_i	0.5	kmol
Relative volatility between light and heavy component	α	1.5	–
Effect of vapor flow on liquid flow	λ	0	–
Time constant for liquid dynamics	τ_L	0.063	min

dynamics. The model is used in various control studies, including multivariable control (Skogestad, 1997; Skogestad & Postlethwaite, 2005).

There are seven degrees of freedom, being the first seven variables listed in Table 5.1. Only four of these are considered manipulated variables, the other being disturbances. The manipulated variables are the four flows: reflux (L), boilup (V), distillate (D), and bottoms (B). The various control configurations using these variables are discussed in the following subsection.

The model allows specification of all important process properties, such as relative volatility, the number of stages, and feed position. The number of stages (including reboiler) was chosen as 41 and the feed position as stage 21. The relative volatility was 1.5.

The states predicted by the model are the mole fractions of light component and liquid holdup on each stage (x_i and M_i for $i = 1 \ldots N$). Thus, for a total of N stages, the model will predict a total of $2N$ states.

The vapor-liquid equilibria are predicted as per Eq. 5.2.1.

$$y_i = \frac{\alpha x_i}{1 + (\alpha - 1) \cdot x_i} \tag{5.2.1}$$

The vapor flows (assuming constant molar flows) are calculated as per Eq. 5.2.2, where NT denotes the number of stages and NF the location of the feed stage.

$$V_i = \begin{cases} V & \text{for } 1 \leq i < NF \\ V + (1 - q_F) \cdot F & \text{for } NF \leq i < NT \end{cases} \tag{5.2.2}$$

The liquid flows are predicted as per Equation 5.2.3, where L_{0i} and M_{0i} are the nominal values for the liquid flow and holdup on stage i, respectively. L_{0b} is the nominal liquid flow below the feed, being equal to $L_0 + q_{F0} \cdot F_0$, with q_{F0} and F_0 both being equal to 1.0. The nominal vapor flow V_0 is equal to 3.20629 kmol/min, while the nominal vapor flow above the feed, V_{0t}, is equal to $V_0 + (1 - q_{F0}) \cdot F_0$. The nominal reflux flow L_0 is equal to 2.70629 kmol/min.

$$L_i = \begin{cases} L_0 + (M_i - M_{0i})/\tau_L + \lambda \cdot (V_{i-1} - V_0) & \text{for } 2 \leq i \leq NF \\ L_{0b} + (M_i - M_{0i})/\tau_L + \lambda \cdot (V_{i-1} - V_{0t}) & \text{for } NF < i < NT \\ L & \text{for } i = NT \end{cases} \tag{5.2.3}$$

The following differential equations describe total (Eq. 5.2.4) and component (Eq. 5.2.5) holdup:

$$\frac{dM_i}{dt} = L_{i+1} - L_i + V_{i-1} - V_i \tag{5.2.4}$$

$$\frac{dM_i x_i}{dt} = L_{i+1} \cdot x_{i+1} - L_i \cdot x_i + V_{i-1} \cdot y_{i-1} - V_i \cdot y_i \tag{5.2.5}$$

The feed stage holdup needs special consideration, where the feed is assumed to be mixed in:

$$\frac{dM_{NF}}{dt} = L_{i+1} - L_i + V_{i-1} - V_i + F \tag{5.2.6}$$

$$\frac{dM_{NF} x_{NF}}{dt} = L_{i+1} \cdot x_{i+1} - L_i \cdot x_i + V_{i-1} \cdot y_{i-1} - V_i \cdot y_i + F \cdot z_F \tag{5.2.7}$$

The reboiler is assumed to be an equilibrium stage:

$$\frac{dM_1}{dt} = L_2 - V_1 - B; \tag{5.2.8}$$

$$\frac{dM_1 x_1}{dt} = L_2 \cdot x_2 - V_1 \cdot y_1 - B \cdot x_1 \tag{5.2.9}$$

The total condenser is not an equilibrium stage:

$$\frac{dM_{NT}}{dt} = V_{NT-1} - L - D \tag{5.2.10}$$

$$\frac{dM_{NT} x_{NT}}{dt} = V_{NT-1} \cdot y_{NT-1} - L \cdot x_{NT} - D \cdot x_{NT} \tag{5.2.11}$$

Since $d(Mx) = xdM + Mdx$, the following equation would describe dx/dt:

$$\frac{dx_i}{dt} = \left(\frac{dM_i x_i}{dt} - x_i \cdot \frac{dM_i}{dt} \right) / M_i \tag{5.2.12}$$

All of the above equations, when used together, describe the model for the so-called Column A of Skogestad (1997).

5.3 Distillation Control

5.3.1 Feedback Control

As with most other unit operations, multi-loop feedback control is the most popular way of controlling distillations columns. There are three main ways of composition control in distillation: open-loop, one-point, and two-point control (Skogestad, 1997). In open-loop control, no composition control is done automatically, meaning operators manually adjust flows to alter compositions. In one-point control, one composition loop is closed and with two-point control, both composition loops are closed.

Open-loop control is relatively common in industrial practice, which is a shame since the type of response under such control is slow and far from optimal. Skogestad (1997) shows that when a column is under open-loop control with a 'perfect' operator making step changes in the manipulated variables, the plant response is actually slower than when a more gradual response change is made such as with two-point control. This is illustrated by Figure 5.2.

One can see that the elapsed time before steady state is reached is much higher under manual control with a perfect operator. This is due to the large open-loop time constant of high-purity distillation columns (Skogestad, 1997). The reason that feedback control can bring about a much shorter response time is because feedback changes the dynamics by moving the poles (Skogestad, 1997).

Closing one or both of the composition loops will elucidate a faster response from the distillation column. One-point control is appropriate when one composition is more important than the other, e.g. an ethanol/water column, where the ethanol in the distillate needs to be maintained at set point closer than water in the bottoms product. It is also easier to use from an operational standpoint.

Two-point control is often best from an economic standpoint, since the operator usually overpurifies the uncontrolled composition in one-point control, leading to energy wastage and a reduction in capacity (Skogestad, 1997). Two-point control is also the obvious choice when control of both the distillate and bottoms composition are important. However, two-point control is also more difficult because of interaction between the two control loops.

Many different controlled/manipulated variable pairings can be made under the different control configurations. The LV pairing is shown partially in Figure 5.1 - distillate and bottoms flow rate are level controlled, meaning reflux (L) and boilup (V) are free to be controlled with PI composition controllers. Other configurations include DV, LB, DB, and $(L/D)(V/B)$ control.

Each configuration has its own advantages and disadvantages, depending on the situation and column properties. A detailed analysis of the various configurations and their frequency response can be found in Skogestad (1997).

For the purposes of this control study it is assumed that both distillate and bottoms need to be closely controlled. This means that the neurocontrol solution will be compared to the best-performing two-point control configuration.

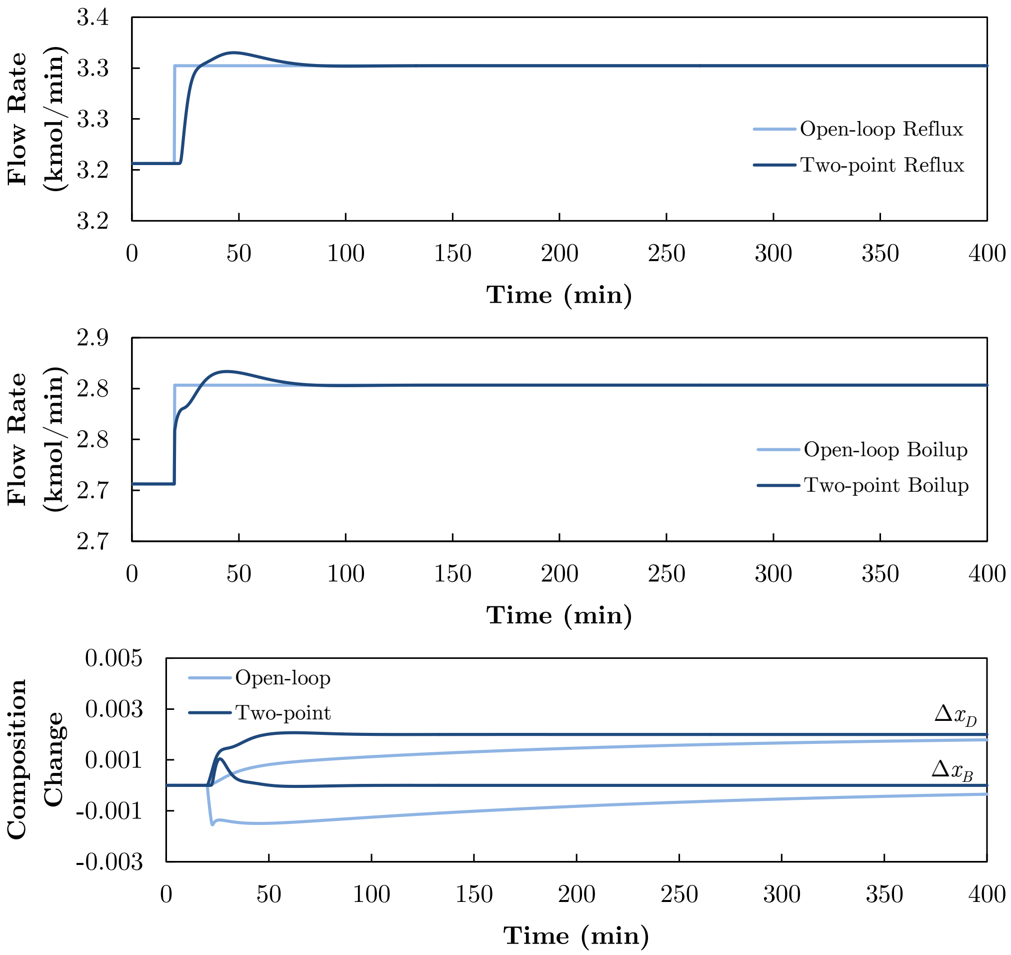

Figure 5.2: Example of the change in distillate and bottoms composition for the changes in reflux and boilup rate under open-loop control with a perfect operator and two-point *LV* control.

5.3.2 Centralized Control

Since distillation is a multivariable control problem, a strategy that can use all the manipulated variables to maintain all the controlled variables close to their desired value would be the ideal.

One such strategy is model predictive control (MPC), which is one of the most common, if not the most common, centralized control strategy in industry (Hussain, 1999; Camacho & Bordons, 2004). MPC can achieve very good control of distillation columns even when model uncertainty is taken into account (Skogestad & Postlethwaite, 2005). One of the biggest hindrances to the application of MPC in distillation is the cost and effort involved in finding an adequate model for the strategy to succeed. However, when such a model is available, very good control performance can result (Diehl *et al.*, 2002).

The author is only aware two published studies of neural networks applied as feedback controllers for distillation. One is the work of Conradie & Aldrich (2010),

Table 5.2: Process variable measurement delays and control interval sizes.

Controlled Variable	Symbol	Measurement Delay	Control Interval
Compositions	x_i	0.5 minutes	0.5 minutes
Holdup	M_i	-	0.1 minutes

who apply neural networks for both identification and control purposes of a multi-effect batch distillation pilot plant. Very good performance was achieved, with an overall 45% improvement in energy consumption due to a decrease in batch time. The other study is by Yu *et al.* (1999), who use neural networks in a model reference structure for identification and control of a multicomponent distillation column.

The work of Conradie & Aldrich (2010) is similar to this study, since the controller training will be done using reinforcement learning (but using different algorithms). In addition to the system studied by Conradie & Aldrich (2010) being a multi-effect batch distillation system, the major difference is that their work relied on only a partially identified system model for training of the neurocontroller. In this work, training and testing will be done on both the full system model and (if possible) a system identified model. The option of learning a partial neurocontroller on-the-fly (on-line) will also be explored.

The goal of the neurocontrol solution developed in this study will be to surpass the performance of any of the two-point feedback control methodologies by successfully mitigating the controller interaction which is inherent to the process.

5.4 Feedback Control

5.4.1 System Setup

The model of Skogestad (1997) was implemented in Simulink and the PI and proportional controllers were built from the available blocks. The PI controller subsystem was based on the *external feedback* design in order to achieve anti-reset windup. This design in shown in the previous chapter in Figure 4.2 on page 30.

Measurement delays and control discretization was the same for both feedback control and neurocontrol. A measurement delay of 30 seconds was introduced for all composition measurements, and these operated at a 30 second interval as well. This gave the composition controllers an interval time of 0.5 minutes. The level (holdup) measurements had negligible delay but operated at 0.1 minute intervals. This is summarized in Table 5.2

5.4.2 Control Configuration

Three of the most common distillation control configurations were considered in this study for use in comparison to neurocontrol: DB, $(L/D)(V/B)$, and LV control.

The control configuration that is partially shown in Figure 5.1 is LV control, where reboiler and condenser holdup are controlled by the distillate and bottoms flow rate, respectively. Composition is controlled by adjusting the reflux and boilup

Table 5.3: Performance comparison of the different controllers with corresponding tuning.

Configuration	x_B loop tuning		x_D loop tuning		x_B IAE	x_D IAE
DB	$k_P = 70$	$k_I = 15$	$k_P = -120$	$k_I = 20$	0.289	0.248
LV	$k_P = -37.5$	$k_I = 3.31$	$k_P = 26.1$	$k_I = 3.76$	0.452	0.341
$(L/D)(V/B)$	$k_P = -70$	$k_I = 6$	$k_P = 60$	$k_I = 6$	0.666	0.558

flow rates. This configuration is seen as best for one-point composition control where only one composition loop is closed, with the other being maintained stationary or controlled manually (Skogestad, 1997). It can also deliver relatively good performance in two-point control if fast composition measurements are available, such as by inference from tray temperature readings.

DB control is a material balance configuration because composition is controlled using the overall material balance for the column. It is the opposite of LV control in that the reboiler and condenser holdup are controlled using the boilup and reflux rate, respectively. Composition is controlled by adjusting the distillate and bottoms flow rate. This configuration can provide good quality two-point control, especially in high purity distillation or with large reflux (Skogestad, 1997).

$(L/D)(V/B)$ control works by adjusting the reflux ratio (L/D) and boilup ratio (V/B) to maintain control of the distillate and bottoms product, respectively. It is a good overall choice for all modes of operation, but needs tight level control for good performance.

5.4.3 Configuration Comparison

Only one of the controller configurations will be used to compare to the neurocontrol solution. In order to compare them against one-another, a test problem was devised where both set points were changed independently from one-another every 60 minutes. A simulation was done for 600 minutes, and the absolute set point error for both compositions were integrated to get their respective IAE (Integral of Absolute Error) values.

The best tuning for the LV controller configuration had already been determined by Skogestad (1997), but the others had to be determined by trial-and-error. The results are shown in Table 5.3. DB control is found to be superior by a large margin to both other configurations, probably because of the high purity (99%), not-so-tight level control (due to 0.1 minute measurement delays) and subsequent process interaction.

5.5 Neurocontrol

5.5.1 Control Configuration

The neurocontrollers used in this study were made to track two set points: the distillate and bottoms compositions. This was done in the same way as in the ball mill

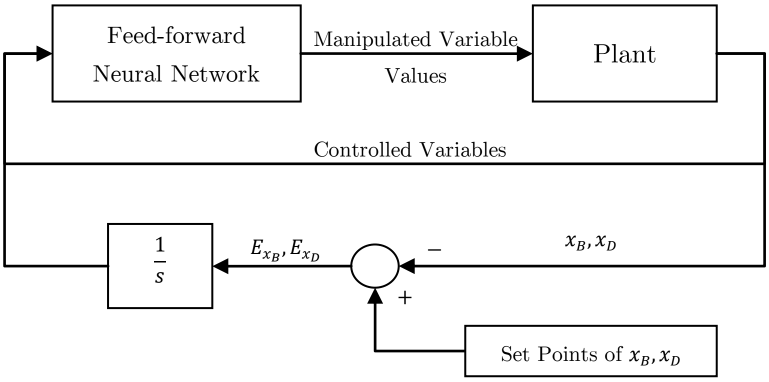

Figure 5.3: Illustration of neurocontroller solution, showing the integration of error for set point tracking.

Table 5.4: Neurcontroller output configurations.

Output Type	Interval			
	L	*V*	*D*	*B*
Linear	$[0.0, 5.0]$	$[0.0, 5.0]$	$[0.0, 5.0]$	$[0.0, 5.0]$
Hyperbolic Tan	$(1.0, 4.0)$	$(1.0, 4.0)$	$(0.0, 1.5)$	$(0.0, 1.5)$

study, by feeding the integral of the set point error as input to the neurocontroller. Since there were two set points, two integral errors were fed to the neurocontroller as additional inputs. This is illustrated in Figure 5.3.

The same composition and holdup measurement delays (Table 5.2) as with feedback control were experienced by the neurocontroller.

The output of the neurocontroller were the four process manipulated variables, being the reflux, boilup, distillate and bottoms flow rates. In addition to the set point integral errors, the network was given 10 other representative process measurements - five composition and five level measurements. These were the composition and holdup of the reboiler and condenser, as well as the composition and holdup of three stages in between, being stages 11, 21, and 31. This gave a total of 12 network inputs. The neural network inputs and outputs are summarized in Figure 5.4.

Given the number of inputs and outputs the neural network needed to handle, the number of hidden units to include in the (single only) hidden layer could be decided. The different numbers to be tested were chosen as 2, 4, 6, 8, 16, and 24.

As for the ball mill study, two different types of network output was studied - linear with hard output saturation and TanH output transformation to minimum and maximum values. This gave a total of 12 different neurocontrollers to be trained by the two reinforcement learning algorithms PGPE and CMA-ES. The network output saturation limits are given in Table 5.4.

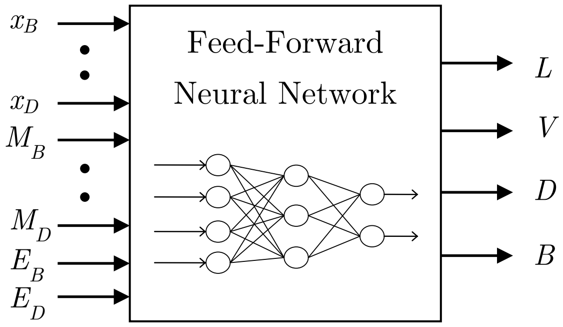

Figure 5.4: Illustration of neurocontroller inputs and outputs. Five composition measurements, five holdup measurements, and two integral errors serve as inputs to the network. The outputs are reflux, boilup, distillate and bottoms flow rate, respectively.

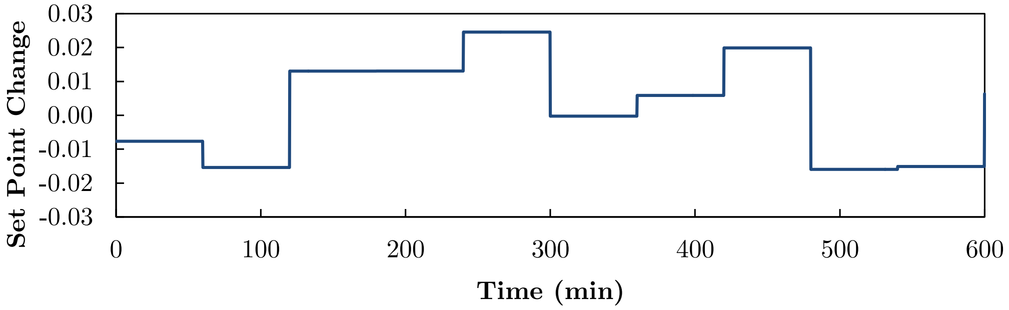

Figure 5.5: Change in set point over time during a training run for both distillate and bottoms composition.

5.5.2 Training

Training was done in episodes of 600 minutes, where the set point of both distillate and bottoms compositions were changed every 60 minutes to between $(0.95, 0.995)$ and $(0.005, 0.05)$ respectively. The set point change profile for both compositions is given in Figure 5.5.

The reward equation that guides the learning process is given in Equation 5.5.1. The goal of learning is made clear by equation: keep the compositions as close as possible to their set points and the reboiler and condenser liquid holdup at their nominal values. Because the change in composition under normal conditions is smaller than the change in holdup (in absolute terms), the corresponding reward is multiplied by 10 to increase its overall importance in the context of final reward.

$$R = \sum_{t=0} (-10 \times |SP_{x_B} - x_B| - 10 \times |SP_{x_D} - x_D| - |M_B - 0.5| - |M_D - 0.5|) \quad (5.5.1)$$

When learning ends prematurely (before the end of the episode, i.e. before 600

Table 5.5: Best fitness of ten learning runs.

		Number of Hidden Units					
Algorithm	Output Type	2	4	6	8	16	24
CMA-ES	Linear	10,514	6,724	7,510	9,595	149,741	152,935
	TanH	9,072	5,201	5,546	5,337	6,105	6,594
PGPE	Linear	167,467	219,553	160,881	272,970	299,288	299,472
	TanH	96,531	94,030	151,604	153,739	151,684	153,444

minutes have passed), a negative reward of 10 is added for each outstanding time step.

A training run was allowed to have 5000 episodes (i.e. function evaluations) before ending. The best fitness and corresponding parameters obtained during such a run is stored as the result.

5.6 Model-Based Learning

As explained previously, model-based learning refers to the case where a perfect plant model is available for training of a neurocontroller. Thus, no plant-model mismatch comes into play. This is done by doing both controller training and testing on the same plant model. This can be contrasted to the work of the following subsection, where system identification is used to construct a NARX model, which, if acccurate enough, will be used for training of the neurocontrollers. Testing is still done on the original model.

The learning results of PGPE and CMA-ES on the model of Skogestad (1997) are compared in this section, with the best-performing neurocontroller that was obtained used for comparison to the DB feedback control solution.

5.6.1 Learning Results

The model-based learning results are presented in Table 5.5. It is clear that CMA-ES was much more successful on this learning problem than PGPE. It is also clear that using the TanH output transformation greatly increases the measure of success of the learning algorithm. This becomes even clearer when the average fitness of the ten learning runs are investigated (Table 5.6), which shows that a learning run fails more often than not when linear output with hard saturation is used.

The fact that PGPE struggles to find a feasible solution and that TanH output transformation makes such a big difference shows how difficult and non-linear this learning problem is. Despite this, CMA-ES manages to find very good solutions in most situations, although when a large number of hidden units are combined with linear outputs it struggles just as much.

An example of the trajectory of a good learning run is given in Figure 5.6. One can see that after around 3000 episodes the learning is mostly stabilized around the region of best performance, but still experiences a poor run every now and again.

Table 5.6: Average fitness of ten learning runs.

		Number of Hidden Units					
Algorithm	Output Type	2	4	6	8	16	24
CMA-ES	Linear	144,478	142,193	122,328	110,235	199,752	253,471
	TanH	38,039	6,643	26,403	12,110	6,800	7,279
PGPE	Linear	262,174	285,722	279,330	296,369	299,454	299,519
	TanH	244,224	231,989	266,688	219,710	216,717	193,096

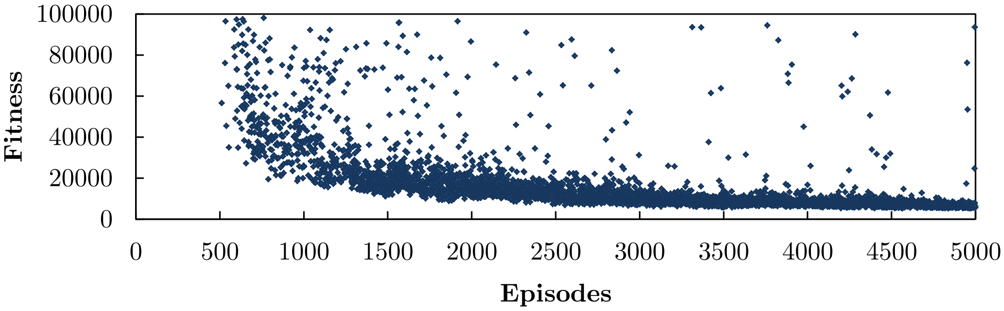

Figure 5.6: Learning progress for a CMA-ES optimization run for the 12 × 4 × 4-TanH neurocontroller. Each data point represents an episode with its corresponding fitness.

Table 5.7: Performance comparison for the various CMA-ES optimized neurocontrollers with TanH output on a 600 minute test run.

	Number of Hidden Units				
IAE	4	6	8	16	24
Bottoms	0.488	0.647	DNF	0.734	0.585
Distillate	1.022	1.158	DNF	1.108	1.237

According to the results in Table 5.5, the overall best solution is the CMA-ES solution of the 12 × 4 × 4 neurocontroller with TanH output. However, the other CMA-ES/TanH neural networks had relatively similar best results. In order to find the best-performing neurocontroller, the same testing scenario that was conducted for the PI controllers was used. A simulation of 600 minutes was conducted, where the distillate and bottoms composition set points were independently changed every 60 minutes. The results are reported in Table 5.7, and confirm that the 12 × 4 × 4-TanH neurocontroller performs best. The 12 × 8 × 4-TanH neurocontroller did not finish the test run due to the reboiler running dry because of inadequate level control.

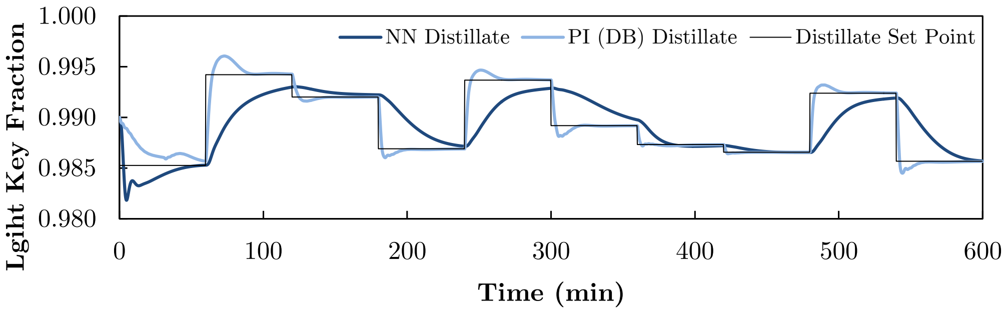

Figure 5.7: Distillate set point tracking comparison of PI (DB) control and the best-performing $12 \times 4 \times 4$-TanH neurocontroller when no disturbances are present.

5.6.2 Control Performance

In order to effectively compare neurocontrol and DB-based PI control of the distillation problem, several different scenarios will need to be tested. The first is to change the set points when no disturbances are present. Thereafter the different disturbances will be introduced in various combinations and the results will be compared.

5.6.2.1 Set Point Tracking - No Disturbance

Figures 5.7 and 5.8 compare the distillate and bottoms set point tracking of the best-performing $12 \times 4 \times 4$-TanH neurocontroller with PI (DB) control when no disturbances are present. It is quite clear that the default integral gain of the neurocontroller (1.0) is not high enough for it to respond as quickly as under PI (DB) control. The could be due to the way training was conducted, where the magnitude of the set point change during a learning run (as seen in Figure 5.5) is greater than for the test runs seen here. Fortunately this is easily remedied by increasing the integral gain.

The behaviour of reboiler and condenser holdup are shown in Figures 5.9 and 5.10 respectively. While significant changes occur in both reboiler and condenser holdup under DB control, neurocontrol does not seem to make much of an impact on these variables. This could be due to the way in which the reward equation (Equation 5.5.1) tells the learner to maintain the holdup at 0.5. A better option might have been to use a reward function that punishes the agent only when holdup moves a certain distance from the nominal value. However, further tests have shown that this results in very loose level control, which frequently results in trays overflowing or running completely dry.

To improve the neurocontroller performance, the integral gain of both distillate and bottoms composition set point error is increased to the point where IAE for both are minimized (see Figure 5.3). The improvement in IAE for an increase in both distillate and bottoms gain is shown in Figure 5.11. The IAE for both distillate and bottoms set point decrease drastically up to a gain of about 4 or 5. The minimum IAE for the bottoms set point (0.303) is found at a gain value of 6,

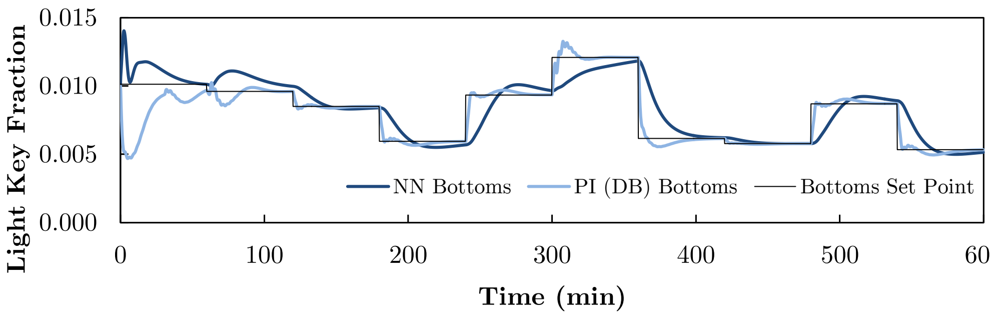

Figure 5.8: Bottoms set point tracking comparison of PI (DB) control and the best-performing $12 \times 4 \times 4$-TanH neurocontroller when no disturbances are present.

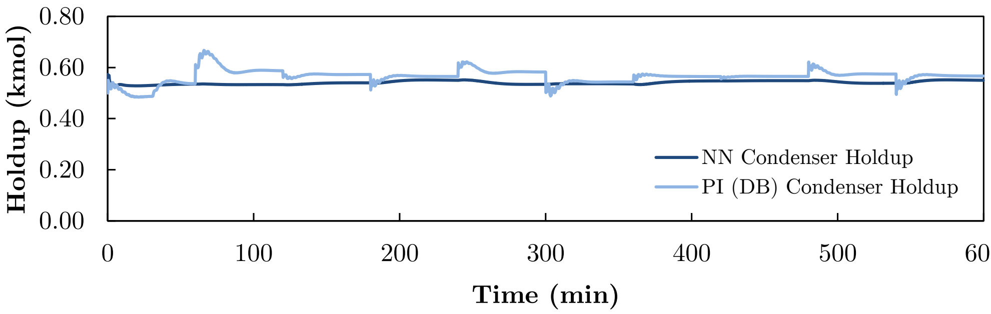

Figure 5.9: Condenser holdup comparison of PI (DB) control and the best-performing $12 \times 4 \times 4$-TanH neurocontroller when no disturbances are present.

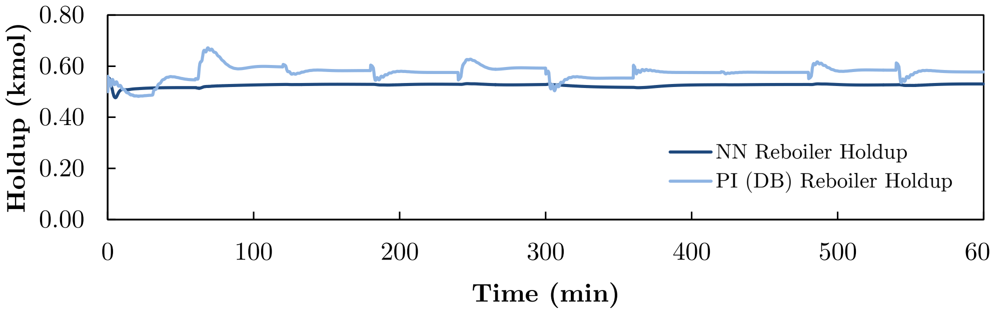

Figure 5.10: Reboiler holdup comparison of PI (DB) control and the best-performing $12 \times 4 \times 4$-TanH neurocontroller when no disturbances are present.

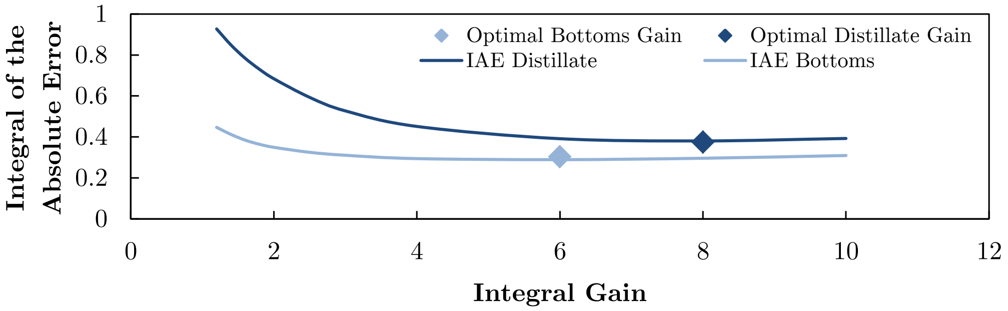

Figure 5.11: Reduction of IAE for increase of integral gain for both distillate and bottoms set points.

Table 5.8: Summary of integral gain optimization results.

Control Scheme	Integral Gain		IAE	
	Bottoms	Distillate	Bottoms	Distillate
PI (LV) Control	-	-	0.341	0.452
PI ($(L/D)(V/B)$) Control	-	-	0.558	0.666
PI (DB) Control	-	-	0.248	0.289
NN Default Gain	1.0	1.0	0.488	1.02
NN Optimal Gain	6.0	8.0	0.303	0.393

while the minimum IAE for the distillate set point (0.393) requires a higher gain value of 8.

The IAE optimization results are given in Table 5.11, with the PI control performance included for comparison. The overall improvement in neurocontrol performance is a 38% improvement for bottoms set point tracking and 63% improvement for distillate set point tracking as measured by the IAEs. This huge improvement further emphasizes the impact of the integral error gain in set point tracking of the neurocontroller.

The improvement is enough to bring neurocontrol performance to the point where it improves on both LV and $(L/D)(V/B)$ control, but not quite enough to reach the level of DB control, where it ends up performing 22% and 29% worse in terms of bottoms and distillate IAE, respectively.

The neurocontrol solution with optimal integral gains is compared to PI (DB) control in Figures 5.12 to 5.15. The neurocontroller shows a much sharper response to set point changes than before, although one can see some overshoot and oscillation in the case of the bottoms composition. Looking at Figures 5.14 and 5.15, the neurocontroller is once again seen to maintain the holdups at relatively constant values.

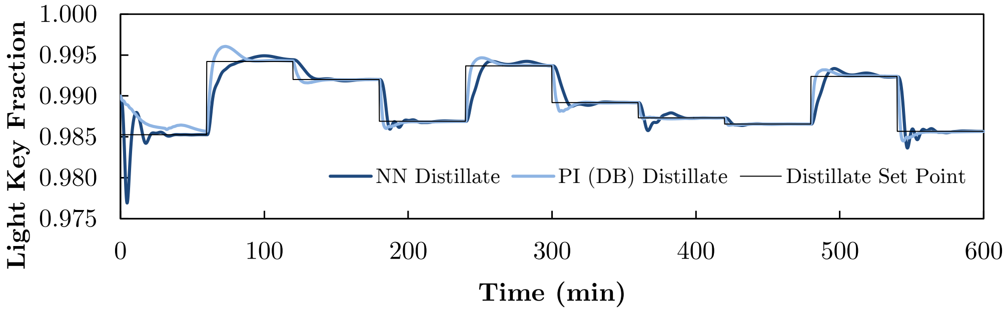

Figure 5.12: Distillate set point tracking comparison of PI (DB) control and the optimized best-performing 12 × 4 × 4-TanH neurocontroller when no disturbances are present.

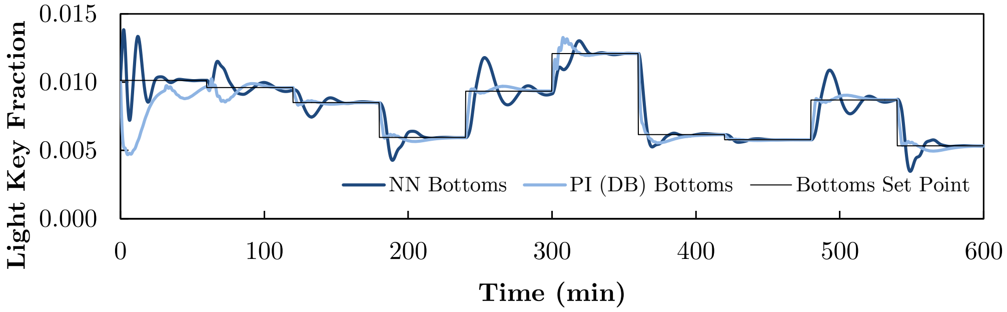

Figure 5.13: Bottoms set point tracking comparison of PI (DB) control and the optimized best-performing 12 × 4 × 4-TanH neurocontroller when no disturbances are present.

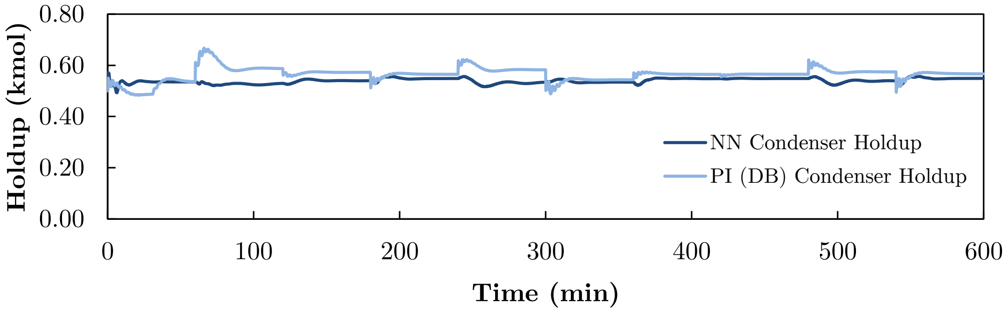

Figure 5.14: Condenser holdup comparison of PI (DB) control and the optimized best-performing 12 × 4 × 4-TanH neurocontroller when no disturbances are present.

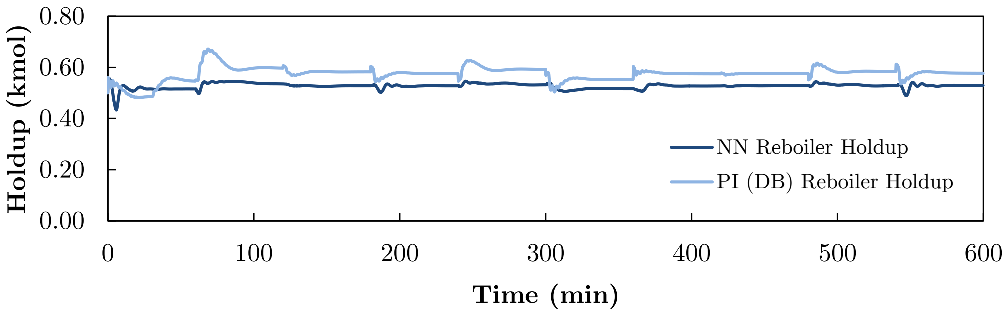

Figure 5.15: Reboiler holdup comparison of PI (DB) control and the optimized best-performing $12 \times 4 \times 4$-TanH neurocontroller when no disturbances are present.

Table 5.9: Feed rate and composition disturbance details

	Time Introduced			
Disturbance	Case 1	Case 2	Initial Value	Final Value
Feed rate	150 minutes	450 minutes	1.0 kmol/min	1.2 kmol/min
Feed composition	450 minutes	150 minutes	0.5	0.6
Distillate set point	300 minutes	300 minutes	0.985	0.994
Bottoms set point	300 minutes	300 minutes	0.0101	0.0096

5.6.3 Set Point Tracking - Feed rate and composition disturbances

In order to test performance under feed rate and composition disturbances, two simulations where these disturbances were introduced at different times was used. In order to make sure the neurocontroller is optimal, the integral gain values obtained in the previous subsection were used. The total length of both simulations was 600 minutes, with the feed rate and composition disturbances introduced at either $t = 150$ or $t = 450$ minutes. At $t = 300$ minutes, both the bottoms and distillate set point was changed. The details are provided in Table 5.9.

The results for distillate and bottoms composition as well as reboiler and condenser holdup, are shown in Figures 5.16 to 5.19 for case 1, and Figures 5.20 to 5.23 for case 2.

Looking at the distillate and bottoms composition time series in Figures 5.16 and 5.17, one could conclude that the neurocontroller cannot effectively handle either feed rate or composition disturbances. However, looking at Figures 5.20 and 5.21, where the feed composition disturbance takes place first, it can be seen that the neurocontroller does a relatively good job of maintaining the products at their set point. The big culprit, therefore, is the feed rate disturbance.

Why does the feed rate disturbance (which is so effectively handled under PI control) cause such a dramatic drop in performance? In the first place, one needs to understand that the variables that conduct primary level control, the distillate and bottoms flow rates, have both a fast and significant effect on composition (Skogestad,

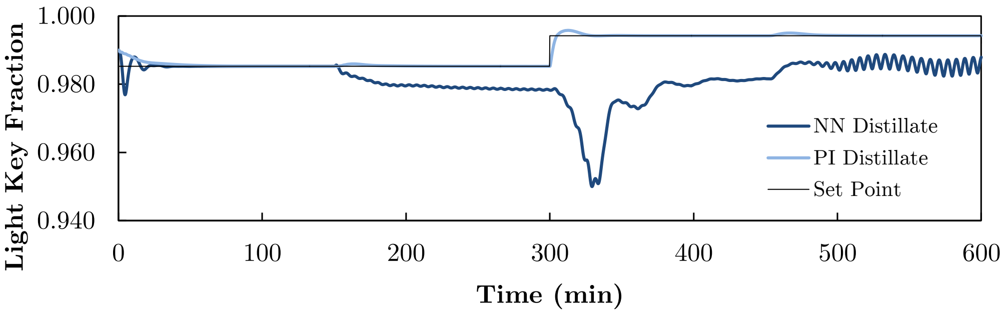

Figure 5.16: Distillate set point tracking comparison of PI (DB) control and the optimized best-performing 12 × 4 × 4-TanH neurocontroller when disturbances are introduced as per "Case 1" in Table 5.9.

1997). Since feed rate necessarily impacts on all stage levels, including that of the reboiler and condenser, the quality of level control will have a noticeable effect on compositions.

Looking at the reboiler and condenser levels in Figures 5.18 etc., it would appear that there is overall level control which starts oscillating (with low amplitude) as soon as the feed rate changes. These oscillations do not seem to increase in severity until other disturbances enter the system, indicating that the system is near the stability limit. Clearly, the reinforcement learning algorithm failed to capture the proper control response in the neurocontroller during learning.

The neurocontroller shown here has an inherent response to the integral error that results in the instability seen in these figures. This inherent response can effectively be interpreted as an "integral time" that is too low - i.e. it causes an amplitude ratio that is very close to 1.0, seen as the standing waves in the relevant figures. This is supported by the fact that *decreasing* the integral gain (corresponding to increasing the integral time of a PI controller) removes the oscillation from the level control response.

When the gain value is decreased low enough, the oscillations disappear, but the overall response of the controller remains the same - composition set points are not tracked successfully. This can only be explained by the overall neurocontroller response, where the reinforcement learning algorithm did not sufficiently capture the process dynamics for extrapolation to conditions such as these. This shows a lack of generalization on the part of the reinforcement learning algorithm on the chosen neurocontroller. One way to fix this would be to include more exploration of the state-space during learning episodes than was used in this case.

5.6.4 Partial Neurocontrol

Since the control performance of a full neurocontroller is unacceptable under feed rate disturbance conditions, additional work was done on developing a partial neurocontrol solution. In this control solution, the distillate and bottoms flow rate are used for column inventory control - being set to proportional control of the condenser and reboiler level, respectively. The two remaining degrees of freedom, reflux

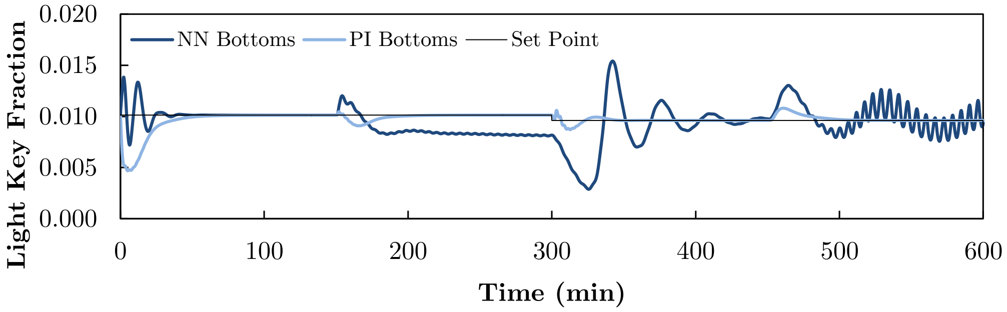

Figure 5.17: Bottoms set point tracking comparison of PI (DB) control and the optimized best-performing $12 \times 4 \times 4$-TanH neurocontroller when disturbances are introduced as per "Case 1" in Table 5.9.

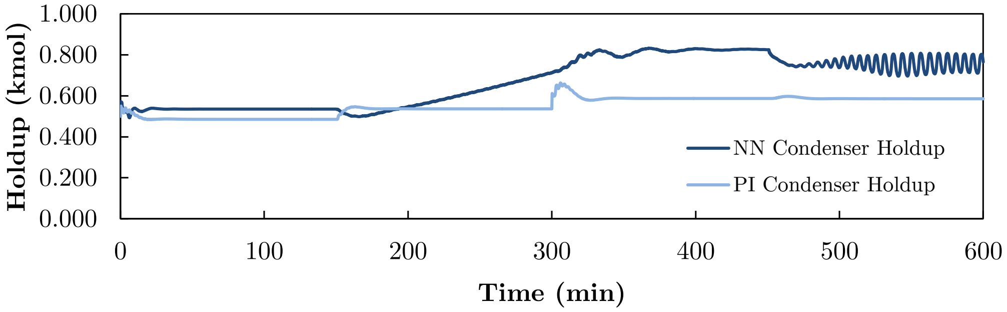

Figure 5.18: Condenser holdup comparison of PI (DB) control and the optimized best-performing $12 \times 4 \times 4$-TanH neurocontroller when disturbances are introduced as per "Case 1" in Table 5.9.

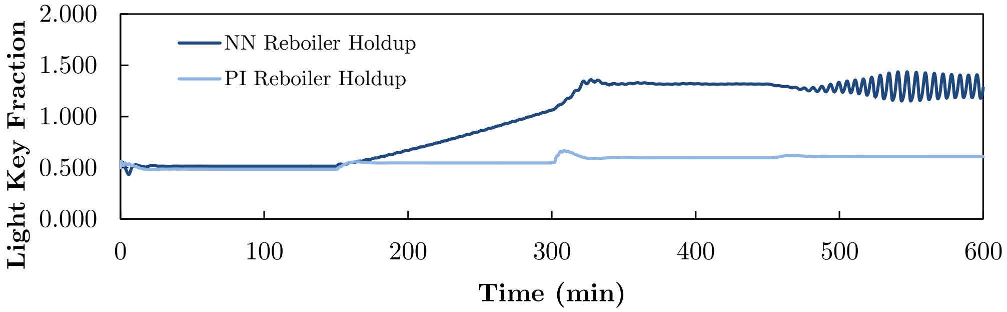

Figure 5.19: Reboiler holdup comparison of PI (DB) control and the optimized best-performing $12 \times 4 \times 4$-TanH neurocontroller when disturbances are introduced as per "Case 1" in Table 5.9.

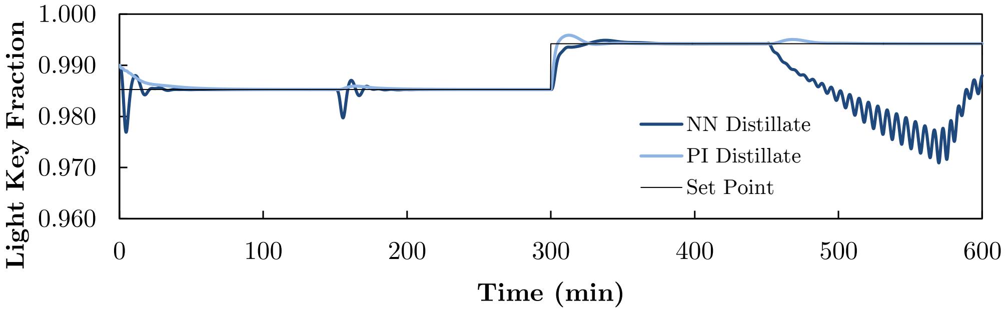

Figure 5.20: Distillate set point tracking comparison of PI (DB) control and the optimized best-performing 12 × 4 × 4-TanH neurocontroller when disturbances are introduced as per "Case 2" in Table 5.9.

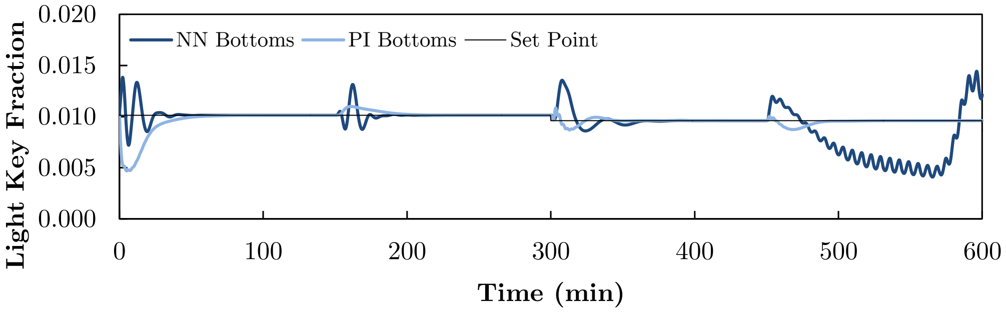

Figure 5.21: Bottoms set point tracking comparison of PI (DB) control and the optimized best-performing 12 × 4 × 4-TanH neurocontroller when disturbances are introduced as per "Case 2" in Table 5.9.

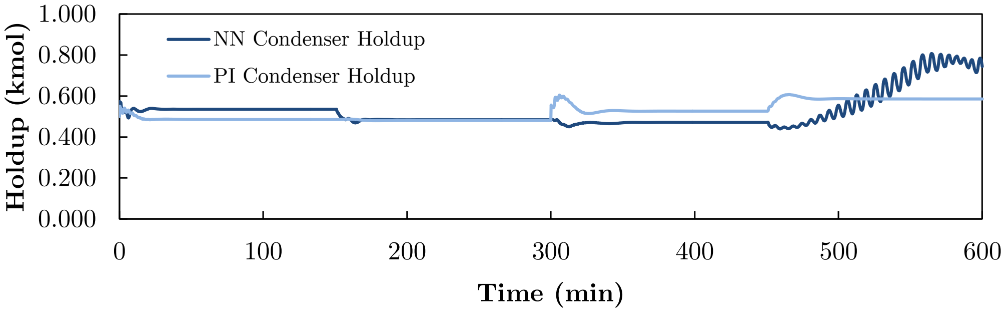

Figure 5.22: Condenser holdup comparison of PI (DB) control and the optimized best-performing 12 × 4 × 4-TanH neurocontroller when disturbances are introduced as per "Case 2" in Table 5.9.

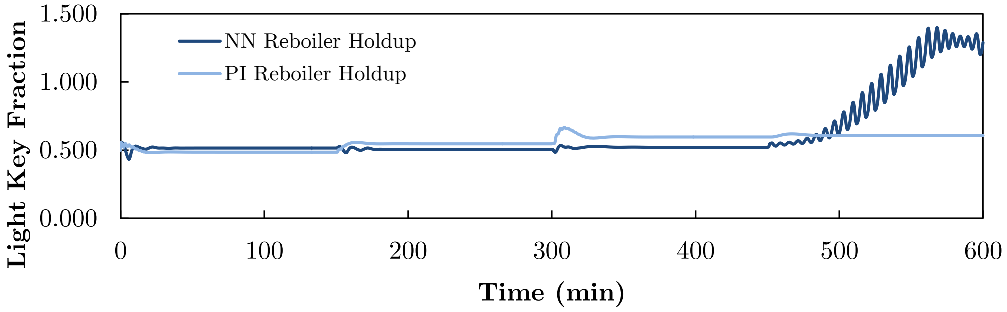

Figure 5.23: Reboiler holdup comparison of PI (DB) control and the optimized best-performing $12 \times 4 \times 4$-TanH neurocontroller when disturbances are introduced as per "Case 2" in Table 5.9.

Table 5.10: Best and average fitness of five learning runs for the reduced neurocontrol problem.

Algorithm	Output Type	Statistic	Neurocontroller	
			$2 \times 2 \times 2$	$6 \times 4 \times 2$
CMA-ES	TanH	Best Fitness	7,627	5,698
		Average Fitness	8,445	5,723

and boilup, are thus allotted to the neurocontroller.

Two different neurocontroller structures were trained. The first made use of a reduced number of six inputs, being the distillate and bottoms composition and their set point error integrals, as well as the reboiler and condenser holdup. The second had only two inputs, being the distillate and bottoms composition set point error integrals.

Training was the same as under the full model-based neurocontrol study. Episodes were 600 minutes long, the set points were changed according to Figure 5.5, and 5000 episodes were allowed before training was terminated. Because of their superior performance, CMA-ES was used and only TanH-output neurocontrollers were trained, with their output being limited to $(1.0, 5.0)$. Only five learning runs were done before the best-performing run was chosen for further study. The results are presented in Table 5.10.

Both neurocontrollers show relatively consistent performance for all five runs, with their average and best fitness values not lying too far apart. The results show that using only two inputs does limit the neurocontrol performance. The 6-input neurocontroller best fitness value is slightly higher than the values for the full neurocontroller (Table 5.5), probably because distillate and bottoms flow rate can no longer be directly controlled. Another reason, that would especially have affected the 2-input neurocontroller, is the fact that the controller receives an insufficient amount of inputs to form a proper state-space representation of the plant. Due to its better performance, the 6-input neurocontroller is used for the upcoming comparisons.

Table 5.11: Summary of integral gain optimization results.

Control Scheme	Integral Gain		IAE	
	Bottoms	Distillate	Bottoms	Distillate
PI (LV) Control	-	-	0.341	0.452
PI ($(L/D)(V/B)$) Control	-	-	0.558	0.666
PI (DB) Control	-	-	0.248	0.289
NN Default Gain	1.0	1.0	0.488	1.02
NN Optimal Gain	6.0	8.0	0.303	0.393
Reduced NN Default Gain	1.0	1.0	0.437	0.676
Reduced NN Optimal Gain	1.8	1.8	0.458	0.515

Table 5.11 is a copy of Table 5.8 with the partial (or reduced) neurocontroller integral error results included as obtained from the same benchmark test. At their default gain values, the reduced ($6 \times 4 \times 2$) neurocontroller outperforms the full neurocontroller. However, when integral gain optimization is done, no real performance increase is gained. In fact, bottoms composition integral error worsens because of the tighter distillate control.

Looking at the comparative performance under disturbance conditions (Figures 5.24-5.27, case 1 & 2 of Table 5.9), the higher stability does not seem to help control performance at all. Since inventory control was removed from the neurocontroller in this test, we know that other factors are at fault. The biggest reason would seem to be insufficient generalization on the part of the neurocontroller.

The extrapolation of unknown conditions (the combination of higher feed rate with higher feed composition) to the controller can be seen to have highly adverse effects on control performance - a large steady-state offset occurs for the bottoms composition (Figures 5.25 and 5.27). If the simulations are continued beyond 600 minutes, x_B is seen to remain at or climb to 0.4, with x_D still tracking the set point.

Is the blame to be placed on the non-linear controller itself, or the learning implementation? Since linear control is just a subset of non-linear control, it is obvious that there exists a non-linear control solution which can handle such disturbances and provide good control. It is therefore a problem with the learning system itself, being a combination of the algorithm used and insufficient exposure to disturbances during learning.

5.6.5 Discussion

The results in this section show that non-linear feedback control is not always the best, or in some cases even viable, option. Nonetheless, it was shown that, when given enough experience, reinforcement learning *can* learn good control of a binary distillation column.

The reason for the poor performance of the trained neurocontrol solutions rests on the fact that insufficient exploration of the state-space had taken place during learning. This in turn shows that there was insufficient generalization on the part of the neurocontrol solutions, being overtrained to perform only in the conditions

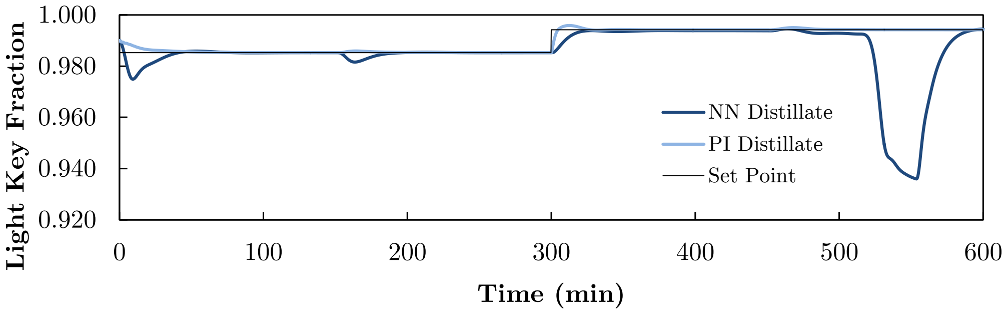

Figure 5.24: Distillate set point tracking comparison of PI (DB) control and the best-performing 6 × 4 × 2-TanH neurocontroller when disturbances are introduced as per "Case 1" in Table 5.9.

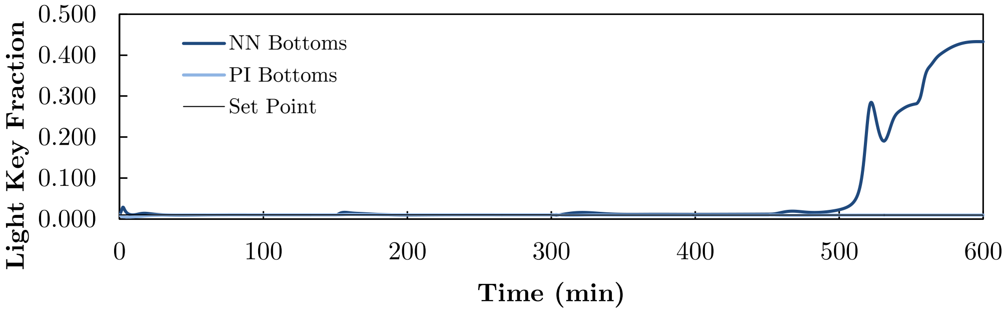

Figure 5.25: Bottoms set point tracking comparison of PI (DB) control and the best-performing 6 × 4 × 2-TanH neurocontroller when disturbances are introduced as per "Case 1" in Table 5.9.

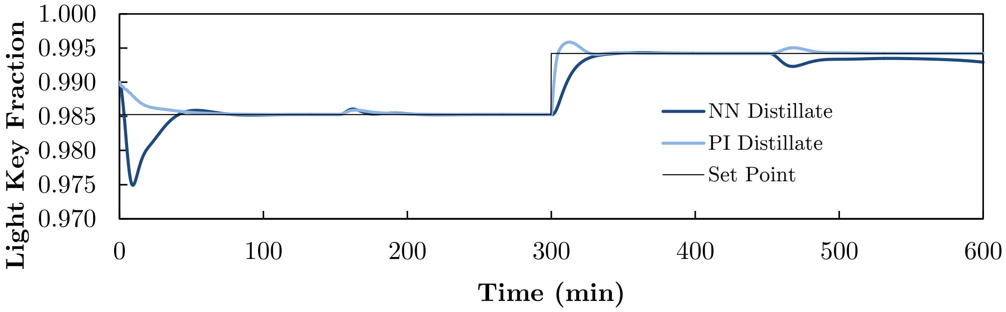

Figure 5.26: Distillate set point tracking comparison of PI (DB) control and the best-performing 6 × 4 × 2-TanH neurocontroller when disturbances are introduced as per "Case 2" in Table 5.9.

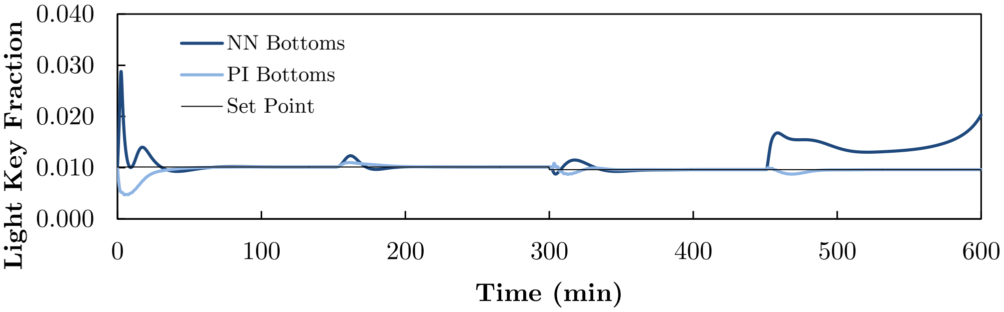

Figure 5.27: Bottoms set point tracking comparison of PI (DB) control and the best-performing 6 × 4 × 2-TanH neurocontroller when disturbances are introduced as per "Case 2" in Table 5.9.

to which they were exposed.

This had two impacts. Firstly, under feed rate disturbances, highly oscillatory control of the reboiler and condenser levels occurred, which also impacted the compositions. This was due to the inherently high integral gain of the neurocontrol solutions, which resulted in level control very close to the stability margin.

Secondly, even when level control was delegated to separate proportional control loops (with only reflux and boilup being neurocontrolled), results when both feed rate and composition disturbances were combined showed a large steady-state offset of bottoms composition.

What can be done to improve control performance? One could alter the learning process so that most possible disturbance combinations are encountered by the learner. This should alleviate the extrapolation problem observed above for most cases, although it would make learning even more difficult than it already is. In addition, one cannot include every possible eventuality in the learning process.

5.7 Data-Based Learning

Since accurate process models are not always available, the reinforcement learning agent needs to be able to learn from either historic data or on-line. This section explores the possibility of learning from historic data. Due to the learning-by-interaction nature of reinforcement learning, this can only be done in a two-step, roundabout fashion. First, one needs to perform system identification on historic plant data to generate a non-linear autoregressive (NARX) model. Second, one conducts reinforcement learning on the NARX model.

This approach was highly successful in the ball mill case study due to the relatively good fit of the NARX model (almost 90%). Applying the same approach to this distillation case study will be a good test of its applicability to other plants.

5.7.1 System Identification

This subsection looks at the possibility of training a NARX model on historic plant data. The training and verification data were both 1000 minutes in length, with a

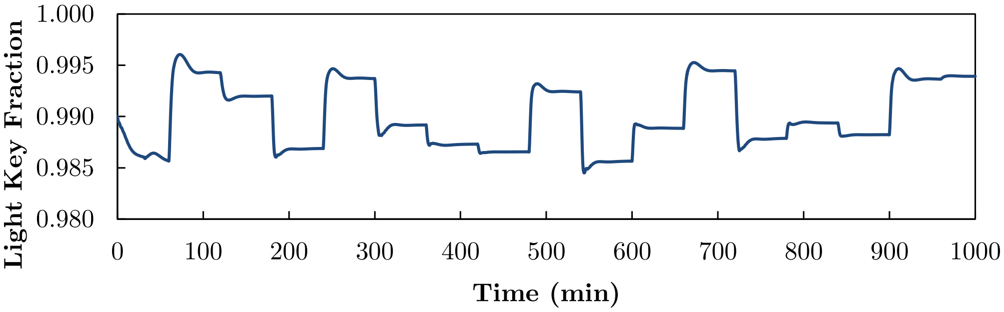

Figure 5.28: Training data for system identification. Distillate Composition.

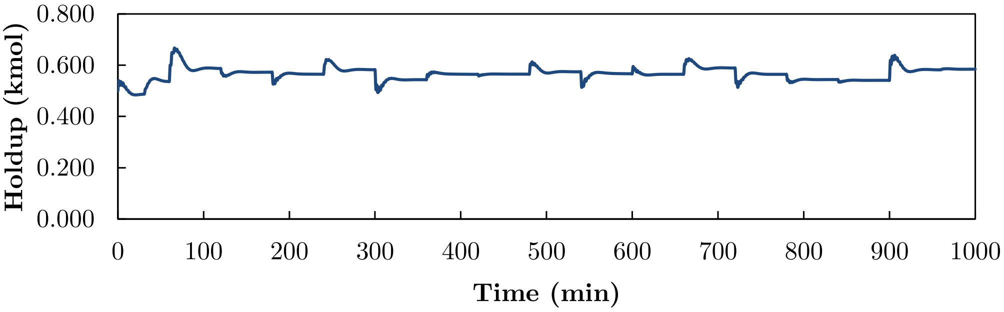

Figure 5.29: Training data for system identification. Condenser holdup.

time interval of 0.1 minutes. This gave 10,000 data points per variable for training and verification. The data was generated by conducting PI (DB) control on the plant, adjusting the set points for distillate and bottoms composition *independently* every 60 minutes. No disturbances were introduced for the training or verification data, with the feed rate kept constant at 1.0 kmol/min, and feed composition at 0.5 light key fraction.

The distillate composition and condenser holdup changes are shown in Figures 5.28 and 5.29 respectively. The verification data was similar, except that the set points were different.

The final (best) training and verification data fit is summarized in Table 5.12. The only non-linearity that was able to provide any fit at all was the Tree Partition. Using more input and output terms also improved fit. However, no matter what combination of non-linearity and number of input and output terms were used, verification fit of the composition data above 0% could not be obtained. Modeling the holdups, however, proved less problematic. Nonetheless, the poor fit of the data essentially prevents data-based learning for distillation column control. Fortunately, as we shall see in the next section, all is not lost for reinforcement learning control of distillation columns when no model is available.

Table 5.12: Comparison of best fitting nonlinearity and number of previous input and output terms used in prediction for each output.

Output	Non-linearity	Inputs	Outputs	Training Fit	Verification Fit
x_B	Tree Partition	2	7	83.8%	-27.6%
x_{11}	"	2	10	91.1%	-9.8%
x_{21}	"	2	4	92.0%	-32.3%
x_{31}	"	2	4	88.8%	-6.8%
x_D	"	2	7	86.0%	-41.1%
m_B	"	2	4	99.5%	97.2%
m_{11}	"	2	4	94.9%	68.7%
m_{21}	"	2	4	93.0%	66.0%
m_{31}	"	2	4	95.1%	73.4%
m_D	"	2	4	99.6%	98.5%

5.8 Adaptive Control

As reported in the previous section, training of a neurocontroller when no model and only historic plant data is available is made impossible due to unsuccessful system identification. However, one option still remains, and that is to learn on-line with a random starting controller. On many plants this will not be an option due to the chaos caused by the random fluctuation in control input. Training of a partial neurocontroller, where only some of the control inputs on the plant are controlled by the learning network, is more feasible. As we shall see, binary distillation is one plant where on-line learning of a partial neurocontroller from a random starting position is feasible.

This section starts by looking at the control configurations in which a partial neurocontroller could be used. The ability of CMA-ES to learn a successful partial neurocontroller is then investigated. Finally, the best controller that was obtained with on-line learning is compared to PI control.

5.8.1 Control Configurations

By necessity, the partial neurocontroller would have to leave out two of the four manipulated variables. This is in order that sufficient level control of the reboiler and condenser holdup can be achieved while learning takes place. If all manipulated variables were controlled by the learning neurocontroller, either the reboiler or condenser, if not both, would quickly run dry or overflow. The manipulated variables not assigned to the neurocontroller would then be put in their own proportional control loops to maintain the reboiler and condenser level.

Two control configurations can therefore be considered. The first is where the reflux (L) and boilup (V) rate are assigned to the neurocontroller. The second is where the distillate (D) and bottoms (B) flow rate are assigned to the neurocontroller. Both configurations would have very different behavior, since external flow changes (D and B) have a larger impact on composition while internal flow changes

Table 5.13: Different neurocontrol and learning configurations used for the on-line learning study.

Level Control	Neurocontrol	Inputs	Episode Length
D and B	L and V	6 or 2	1, 2, 4 or 10 minutes
L and V	D and B	6 or 2	1, 2, 4 or 10 minutes

(L and V) have faster dynamics. This can be seen in the difference between the LV and DB PI control configurations, where LV control is typically poorer at two-point control due to interaction caused by the faster dynamics.

In addition to the two neurocontrol configurations, different numbers of inputs could also be used. A large number of inputs, whilst possibly "informative" would make the fitness landscape more difficult to traverse. In other words, the optimization problem would be even more multi-modal. In order to see the effect that the number of inputs plays in on-line learning, two different input levels are considered, both of which are smaller than the 12 inputs used in the model-based learning study.

The first input level makes use of all the measurements available to a regular multi-loop PI control system - it combines the distillate and bottoms composition and reboiler and condenser holdup with the set point integral errors to give 6 inputs in total. The second input level is the minimum possible amount of inputs, where only the set point error integrals serve as inputs to give 2 inputs in total.

When doing on-line learning using reinforcement learning, one of the most important variables is the episode length. This needs to be long enough to gain a good enough idea of performance for the given controller parameters, but not so long as to make learning too slow. Episode lengths between 50 and 500 time steps will be considered. With a time interval of 0.02 minutes, this means episodes between 1 and 10 minutes in length, which is a reasonable range to consider.

In addition to episode length, further meta-parameters that needed to be set were λ (offspring population size) and the learning rate σ. Since the performance of CMA-ES is not highly dependent on fine-tuning of these parameters, they were set to the reasonable values of $\lambda = 4$ and $\sigma = 1.0$. The starting parameters provided to the learning agent at the start were generated as random numbers uniformly distributed between $(-0.1, 0.1)$.

Table 5.13 summarizes the different learning configurations. In order to provide adequate stability to the learning controller, only the TanH output type was considered. It was also found that the manipulated variable interval size needed to be narrowed. For the model-based learning study of the previous section these were relatively wide - L and V were limited to $(1, 4)$ and D and B to $(0, 1.5)$. The new intervals are provided in Table 5.14.

5.8.2 Results

In order to measure the effectiveness of a learning configuration, 10 runs were conducted per configuration. Three different success measures are provided:

1. Did the system remain stable during learning?

Table 5.14: Neurcontroller output configurations for online learning. Note that either L and V or D and B were controlled by a neural network. If one pair is controlled by the network, the other pair is used for level control (proportional).

Output Type	Interval			
	L	V	D	B
Hyperbolic Tan	$(2.0, 3.0)$	$(2.5, 3.5)$	$(0.45, 0.55)$	$(0.45, 0.55)$

Table 5.15: Learning time allowance for the different neural networks under the four tested episode lengths.

Episode Length	$2 \times 2 \times 2$ network	$6 \times 3 \times 2$ network
1 min	2,000 min	5,000 min
2 min	5,000 min	5,000 min
4 min	10,000 min	10,000 min
10 min	10,000 min	20,000 min

2. Did learning result in a successful control policy?

3. Was there any evidence of set point tracking?

A system was considered unstable upon reaching an impossible state, such as negative holdup, or when uncontrollable oscillations caused the solver to stop. A policy was considered successful if it could maintain the distillate and bottoms composition at least 95% pure, i.e. a light key fraction $> 95\%$ in the distillate and $< 5\%$ in the bottoms. A control policy was considered able to track the set point even if it took very long to change between different set points. Oscillation around the set point was not considered successful at tracking it.

Obviously, learning cannot continue indefinitely, and so there needs to be a cut off point. The points at which learning was stopped is shown in Table 5.15.

The two level control configurations were tested separately. First up is the DB-based neurocontrol solution.

5.8.2.1 DB-based on-line neurocontrol

This section describes the on-line learning results of neurocontrol where distillate (D) and bottoms (B) were controlled by the neural network. Level control was done by proportional control of the reflux (L) and boilup (V) rates. One issue which immediately came to light is that the system was very sensitive to changes in D and B, resulting in the relatively small intervals allowed for control - both were restricted to $(0.45, 0.55)$ as shown in Table 5.14.

This control configuration proved unable to show any learning, with especially holdup proving to be a problem. The typical trajectory of condenser and reboiler holdup is shown in Figure 5.30. The trend is that either reboiler or condenser holdup,

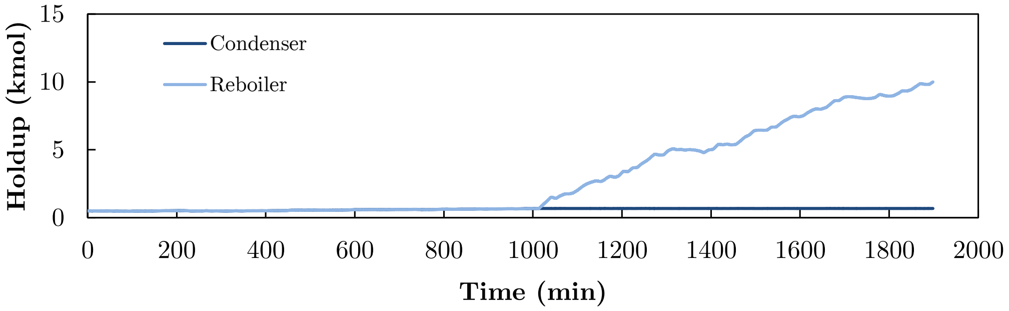

Figure 5.30: Holdup trajectory for a typical DB-based neurocontrol setup learning on-line.

Table 5.16: Stability rate of the different neural networks under the four tested episode lengths.

Episode Length	$2 \times 2 \times 2$ network	$6 \times 3 \times 2$ network
1 min	70%	50%
2 min	70%	90%
4 min	80%	30%
10 min	100%	70%

or both, will increase over time. This is caused by the fact non-linear control is being used to manipulate both of the available column output flows. This obviously has adverse consequences by causing buildup of inventory inside the column, since the inflow remains constant at 1.0 kmol/min.

Because of the holdup problem caused by non-linear control of the column output flows, the only option is to move non-linear control to the column reflux and boilup rates. This allows level control to immediately remove inventory from the system (by manipulating D and B), rather than just pumping the inventory back into the column.

5.8.2.2 LV-based on-line neurocontrol

This section describes the on-line learning results of neurocontrol where reflux and boilup were controlled by the neural network. Level control made use of the distillate and bottoms flow rates. In most cases, learning was able to continue with the column remaining stable. The only cause of instability that was observed was when a tray or the reboiler or condenser ran dry, i.e. had a holdup of zero or less. The learning results for the various learning configurations are shown in Tables 5.16 to 5.18.

The results show conclusively that the simpler 2-input neural network was able to learn more easily and even achieve some semblance of set point tracking by the end of a run. The learning trajectory of the most successful learning run of the $2 \times 2 \times 2$ network is shown in Figure 5.31. This run had an episode length of 4

Table 5.17: Success rate of the different neural networks under the four tested episode lengths.

Episode Length	$2 \times 2 \times 2$ network	$6 \times 3 \times 2$ network
1 min	40%	10%
2 min	30%	0%
4 min	40%	10%
10 min	50%	20%

Table 5.18: Set point tracking ability rate of the different neural networks under the four tested episode lengths.

Episode Length	$2 \times 2 \times 2$ network	$6 \times 3 \times 2$ network
1 min	10%	0%
2 min	10%	0%
4 min	20%	0%
10 min	0%	0%

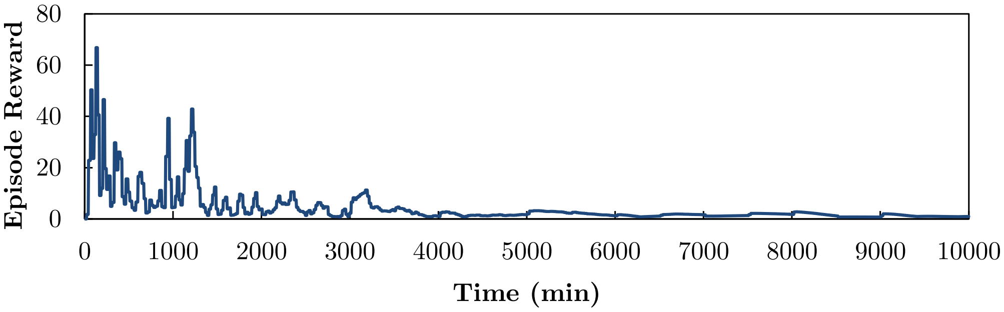

Figure 5.31: Learning trajectory for an LV-based $2 \times 2 \times 2$ neurocontrol setup learning on-line, showing the mean reward obtained during the evaluation of a generation (offspring population) of size $\lambda = 4$.

minutes and was able to track the set point at the end.

Looking at Figure 5.31, several things stand out. Firstly, it would look like around 3500 minutes (equal to almost 2.5 days) are sufficient for the controller to converge with learning. With an episode length of 4 minutes, this corresponds to almost 900 episodes. This is less than the 3000 episodes required for model-based learning, which is to be expected, since the model-based learning study had a much larger controller which also needed to learn to do level control.

The trajectory of distillate and bottoms during the learning run shown in Figure 5.31 are given in Figures 5.32 and 5.33. Neither bottoms nor distillate composition is perturbed too badly for more than 1000 minutes. However, the question

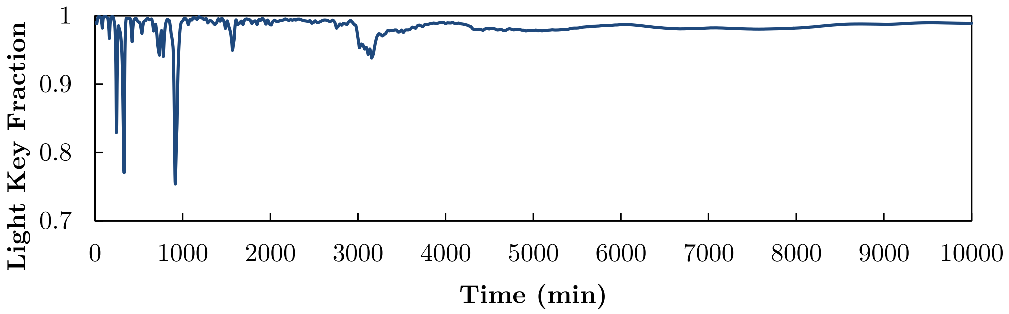

Figure 5.32: Distillate composition for an LV-based 2 × 2 × 2 neurocontrol setup learning on-line.

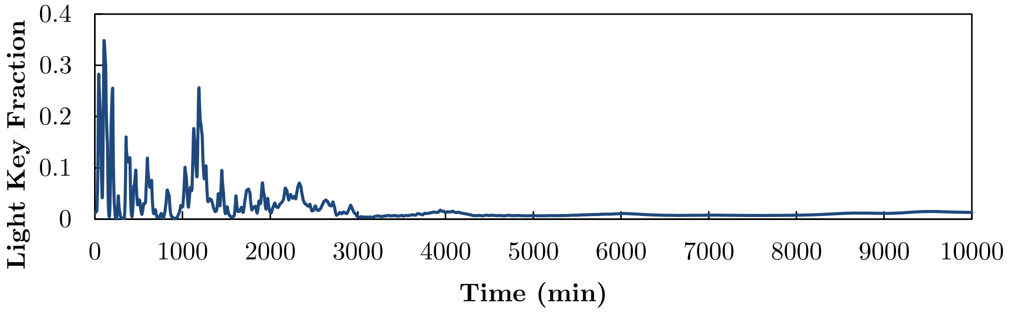

Figure 5.33: Bottoms composition for an LV-based 2 × 2 × 2 neurocontrol setup learning on-line.

of whether or not this learning resulted in any meaningful improvement over alternate methods remains to be seen, and will be answered in the next section when compared to PI and model-based neurocontrol.

5.8.3 Comparison to PI control

To show the effectiveness of the partial controller that learned control on-line, it is compared to PI (DB) control in a simple benchmark test. Due to the relatively poor set point tracking showed by the on-line controller, the test had to be 1200 minutes long, and, as shown in Figures 5.34 and 5.35, it does not compare very well to the top-performing PI control solution.

5.8.4 Discussion

The on-line learning results seen in this section are a step up from results of the ball mill study. One reason why learning could take place on-line is that greater exploration of the control space was allowed, because the system could handle large disturbances in both reflux and boilup rate. This allowed CMA-ES to develop (from scratch) a working neurocontroller with around 50% success rate. Some of

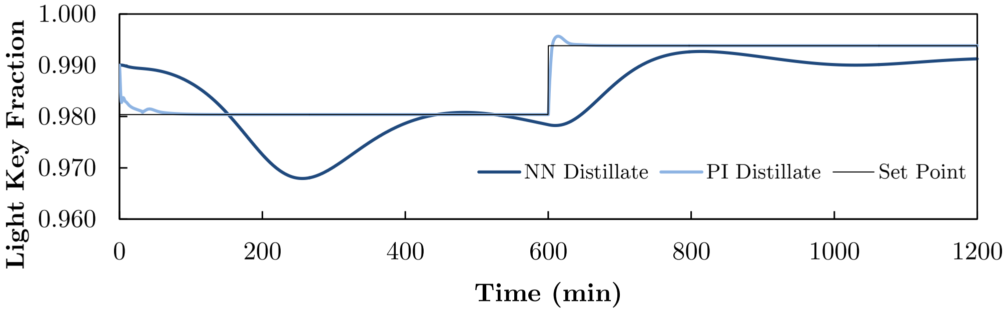

Figure 5.34: Distillate composition of the LV-based $2 \times 2 \times 2$ neurocontroller compared to PI (DB) control.

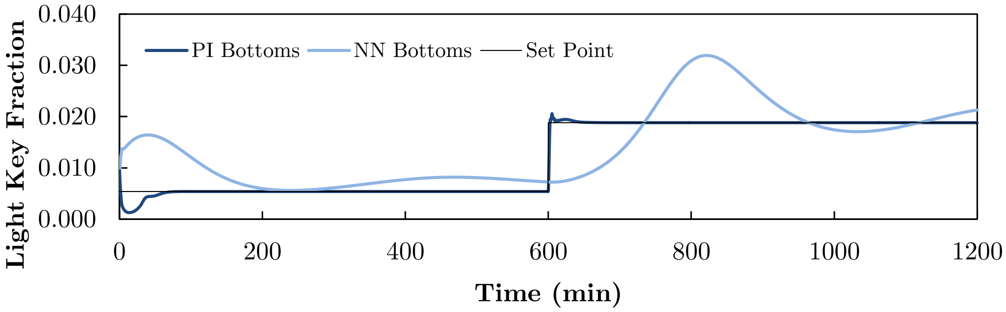

Figure 5.35: Bottoms composition of the LV-based $2 \times 2 \times 2$ neurocontroller compared to PI (DB) control.

the resulting controllers even showed some set point tracking ability, even though their performance was not satisfactory.

The success of CMA-ES at optimizing the neurocontroller parameters by reinforcement from exploring in episodic steps is promising. One of the regions where this algorithm may be better applied is in large-scale optimization of plant operations (areas where linear or non-linear programming are traditionally used). Whereas linear or non-linear programming requires a known objective function for optimization, reinforcement learning only requires the evaluation of the function - it does not need to be known. An example would be a plant where certain operating parameters have an unknown relationship to profitability. Since that relationship is unknown, their optimal values can only be obtained by trial-and-error, a process which can be automated by using CMA-ES.

5.9 Conclusion

Neurocontrol of a binary distillation column using reinforcement learning can improve on some existing multi-loop PI control methods - it can outperform both $(L/D)(V/B)$ and LV control, but lags 20-30% behind DB control. A disadvantage of the neurocontrol solution was its inability to generalize to unknown situations. When both feed rate and composition disturbances were combined, performance deteriorated to the point of not being feasible, with either large steady-state offset of bottoms composition or oscillation of levels and compositions around their set points depending on the control solution. Training controllers by including such disturbances should alleviate this problem, although this would make for an even more difficult learning problem.

In order to investigate the real-world applicability of reinforcement learning by training on plant data, system identification on PI control-generated data was attempted for subsequent use in training of controllers. While the model could fit the training data relatively well, testing on verification data showed very poor fit. This problem was encountered for the composition values, with the holdup values (simply being inventory) not being too much of a problem. Because of this, no controllers could be trained on the data. Further work could rather try fitting the data to grey-box (rather than black-box) models, although such model-building would involve much more work.

As with the ball mill case study, CMA-ES once again outperformed PGPE by a large margin. In fact, PGPE was not able to come up with any feasible control solutions under the exact same learning conditions as CMA-ES. Output transformation using the hyperbolic tangent function also did much to increase the accuracy and best fitness of the solutions obtained by both algorithms.

Moving on to adaptive control, the results were more promising, with CMA-ES showing learning converge from a random starting controller in over half of the test cases. It was found that several solutions even showed the ability to track the set point, although this ability was much slower than under PI control. The adaptive control results showed that there is potential for applying reinforcement learning algorithms such as CMA-ES in an on-line setting. The area where this could be best applied is in a more supervisory control region, where learning does not impact low-level control performance too negatively.

The results for neurocontrol of distillation were negative overall, mostly, as discussed in section 5.6.5 due to the lack of generalization on the part of the neurocontrollers. While non-linear control of the distillation column remains a possibility, and has the ability to improve on even PI (DB) control, the two reinforcement learning algorithms considered were not able to find such a solution within the given state-space exploration framework. While more work could be done to improve control performance, for example by more rigorous training, the very good performance showed by PI (DB) control would probably render the work moot.

The problem here is that the candidate systems for applying neurocontrol need to be carefully chosen. As was proven in the ball mill case study, reinforcement learning and non-linear feedback control has a lot to offer in terms of improving on existing control methods, the candidate system just needs to have sufficient room for improvement.

Chapter 6

Conclusions

The work done in the two preceding chapters show both the positives and negatives of centralized, non-linear feedback control using neural networks and reinforcement learning. The two case studies (ball mill grinding circuit and distillation) both explored three possible applications of reinforcement learning in process control. The first was learning from an exact plant model - in other words the same model was used for training and testing of the neurocontrollers. The second aimed to introduce some realism to the reinforcement learning task. This was done by generating data by applying PI control to the original plant model, and doing system identification on this data. Training of neurocontrollers using reinforcement learning was then conducted on this identified plant model and subsequently verified on the original model. The third was to apply reinforcement learning in an on-line fashion by trying to adapt a random or non-optimal controller to better control the plant in real time.

In the process, by comparing the gradient-based reinforcement learning algorithm PGPE with the evolutionary algorithm CMA-ES, it was found that the latter provided superior performance in every case, by converging more quickly and reaching better final solutions.

Most of the positives were gleaned from the ball mill case study. Here, it was shown in the model-based control study that the centralized nature of the controller effectively handled the interaction that was so problematic under multiloop PI control. Even in cases with combined ore hardness and feed size fraction disturbances, the neurocontroller was able to outperform multiloop control. Not only was it able to provide superior (>5%) mill productivity, the product size fraction set point was tracked more closely, with an IAE typically 10% lower than under multiloop PI control.

In addition to the success of model-based learning, where no plant-model mismatch was assumed, the ball mill case study showed that even when only plant data is available, reinforcement learning of neurocontrol was possible. With only 88% fit on verification data, the reinforcement learning algorithm CMA-ES was able to learn an effective controller from the data-based model. However, in order for the controller to work, the one integrator in the system, sump level, had to be controlled in a separate proportional control loop. Nonetheless, set point tracking of the new controller was either on par with or better than the model-based controller. The better performance, however, was paid for by a lower circuit productivity, being mostly the same as under PI control.

If the performance gains seen in the ball mill case study are justified, there is nothing stopping the introduction of this control methodology on real grinding circuits. This is especially true for cases where a good model is already available. However, even if no such model is available, the possibility still exists to train on an identified plant model based on previous control data, as long as said model provides a reasonably good fit (>88%). Since not all trained controllers are good controllers, the only hurdles are that (1) a large enough batch of controllers need to be trained and (2) the top performers need to be evaluated on the actual plant.

One area which did not see success in the ball mill case study was adaptive control using episodic reinforcement learning. In this study, the improvement of a non-optimal neurocontroller was attempted by applying reinforcement learning in episodic steps. In the case of PGPE, the control performance actually worsened, while in the case of CMA-ES, it remained about the same. The reason for this was the fact that an already-working controller was started out with, and that most solutions in the controller space are similar points of local minima. The low step size allowed in on-line implementation never allowed the learning agent to escape from these local minima, and explore to find more optimal solutions.

The distillation case study was less successful in terms of neurocontrol using reinforcement learning. While it was able to outperform both two-point LV and $(L/D)(V/B)$ PI control at best, it failed to reach the level of performance showed by DB control, where it performed around 20% worse according to distillate and bottoms set point IAE. When feed rate and composition disturbances were combined, the controller showed an inability to return to the set points, and even caused high-frequency oscillations in the compositions and holdup. Even when level control responsibilities were removed from the neurocontroller, it was unable to extrapolate control performance to such difficult (and unknown) conditions. The reason for this was the lack of generalization by the neurocontrollers, meaning that more exploration of the state-space during learning was needed before performance could be expected to improve (at least in the case of process disturbances).

Because of the inability to fit a black box (NARX) model on distillation control data, training on a system identified model was not possible. This showed that some systems, such as distillation, would require more effort in terms of model building and control before successful reinforcement neurocontrol can be achieved. And even then, as the model-based results showed, it may lag behind some form of PI control in terms of performance.

One area where success *was* achieved, is in adaptive control of distillation. In order for adaptive control to work, the inventory control of the distillation column had to be removed to separate control loops. This meant that reboiler and condenser level had to be controlled proportionally by the bottoms and distillate flow rate, respectively. Once this was done, it was shown that CMA-ES was able to start with a random neurocontroller with reflux and boilup rate as outputs (and only the set point integral errors as inputs), and learn to effectively control these to reach relatively high purities in terms of the distillate and bottoms. While almost half of the attempts failed to reach the requisite purities (>95%) to be considered successful, some even showed some set point tracking ability (although they performed really slowly compared to PI control).

So while CMA-ES was unable to learn controllers to outperform PI (DB) control, it did show that it was able to adapt on-line if given enough room to learn. This showed that it could be used at a higher level than direct control, perhaps to optimize operational parameters which had poorly-known effects.

The following questions were posed at the end of Chapter 2:

1. Does centralized neurocontrol, when trained via episodic reinforcement learning, improve on multi-loop PID control?
2. If the above statement is true, can neurocontrollers trained on imperfect models (i.e. with significant plant-model mismatch) still outperform multi-loop PID control?
3. How do the two chosen algorithms, PGPE and CMA-ES, compare to one-another on the same problems?
4. Can either algorithm be used for on-line adaptation of the neurocontroller?
5. Does neurocontrol with integral error feedback successfully track the set point?
6. How does control performance scale with the number of neurons in the hidden layer of the neurocontroller?
7. Is the proposed neurocontrol solution practical and economically viable?

To answer the first question in short: yes, centralized neurocontrol *can* improve on multiloop PID control. As the two case studies have shown, it only depends on whether the system itself can benefit from that type of control. The answer to the second question is also yes, as the ball mill case study has shown. In fact, a similar level of control to when no plant-model mismatch is present can be achieved.

The answer to the third question is most assuredly that CMA-ES is superior to PGPE in performing the type of reinforcement learning tasks seen in this study. Not only was it able to learn more quickly, it also reached better final solutions. In fact, PGPE was unable to learn successful control solutions for most of distillation case study.

The answer to the fourth question in more "no" than "yes", mostly because of the results of the ball mill case study, which showed that controllers that already work are hard to adapt on-line without inducing plant instability. However, as the results of the distillation case study has shown, on-line adaptation is still a possibility if the plant is not caused to become unstable. As stated before, a better use of reinforcement learning may be to optimize operational parameters which do not directly interfere with control.

The answer to the fifth question is a definite yes, with the work in both case studies relying very much on this new approach to integral error tracking of a non-linear feedback controller.

The answer to the sixth question is that it depends on the algorithm being used and the difficulty of the control problem. A good measure to obtain best performance was found to be the size of the control problem. With only two neurons in the hidden layer, the 2×2 ball mill control problem was very effectively controlled. For any algorithm, one would typically like to keep the problem dimensionality low, which makes this a good rule of thumb.

To answer the final question, a short answer would be a "yes", qualified by the positive results shown in the ball mill case study. The only prerequisite is a controllable plant with a good enough model and enough headroom for improvement over existing control methods.

Future work should attempt to evaluate the work done here in laboratory-scale tests, especially regarding system identification and subsequent application of data-based controllers to non-ideal real-world systems.

Appendices

Appendix A

Neural Networks

A.1 Basic Structure

A neural network has been described as "... a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use" (Haykin, 1994). The inspiration for the basic neural network topology comes from the study of biological neurons, the basic building block of the brain. It resembles the brain in two respects (Haykin, 1994):

- The network acquires knowledge through a learning process.
- The strength of the connections between neurons, known as synaptic weights, are used to store the knowledge.

An artificial neural network is a mathematical or computational model. It consists of layers of artificial neurons (see Figure A.1), each connected to the next. A neuron is basically a node which receives the outputs from all or some of the neurons in the preceding layer. For the purposes of this thesis, this output is always a real-valued scalar. Before arriving at the neuron, all these ouputs are first multiplied by

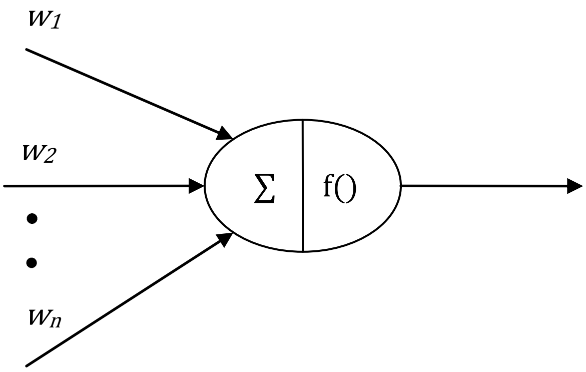

Figure A.1: Artificial Neuron Model.

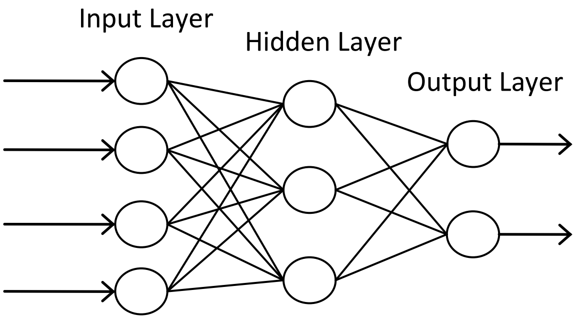

Figure A.2: Non-recurrent feedforward neural network example.

a value (also real-valued scalar) known as a synaptic weight, which is unique to each connection.

It sums these inputs, transforms the sum using some or other function, and sends this transformed sum to the neurons in the following layer. The neuron therefore consists of two computational units: the summing unit, and the transforming unit. The transforming function is usually an s-shaped squashing function such as the hyperbolic tangent (Balakrishnan *et al.*, 1996).

The basic neural network structure described above is known as a *feedforward* neural network. Another widely used neural network structure is a recurrent neural network, where some or all neurons are connected to neurons in a preceding network layer. Recurrent networks exhibit dynamic behaviour due to the cycles in the network topology. Non-recurrent neural networks, which have no cycles, having n inputs and m outputs, can be viewed as a transformation from n-dimensional Euclidean space to m-dimensional Euclidean space (Balakrishnan *et al.*, 1996).

What makes neural networks so popular and widely applicable is the fact they can approximate almost any continuous function very closely. For example, data that would be very difficult to describe using polynomials could be very well described by a neural network, provided enough samples are available. This ability is largely dependent upon the number of neurons in the hidden layer (Balakrishnan *et al.*, 1996).

Looking at a feed-forward network with a single hidden layer, such as the one shown in Figure A.2, the following "information-processing" happens from the input to the output. Firstly, the input values (in this case, four scalars) are sent to each neuron in the hidden layer. Before arriving at each neuron, these scalars are multiplied by another value, called the synaptic weight, which is unique to each connection. Each neuron in the hidden layer then proceeds to sum each of its inputs to give some total value. Before transforming this sum, the neuron will add an additional (scalar) value called the bias, which is unique to each neuron. After transforming the sum, by e.g. calculating tanh(summed value), it is sent to the output layer. Once again, before arriving at the output neurons it is multiplied by a synaptic weight unique to each connection. Similarly to the neurons in the hidden layer, the neurons in the output layer will sum these values, add the bias, transform them, and provide them as output.

The biases and synaptic weights make up the free parameters of the system, being the values which will change as a network is trained. As these values are changed, so will the output given by the network when activated on a specific input. The reason why such a seemingly simple addition and transformation of input values can be used so powerfully is beyond the scope of this study. The reader is referred to Haykin for a thorough investigation of neural networks and what they do (Haykin, 1994).

A.2 Network Learning

Before a network can be used, it needs to be trained. As mentioned previously, a network learns (e.g. is trained) by the systematic adjustment of the bias and weight values as determined by a learning algorithm (Haykin, 1994). While there are many types of learning suited to many types of neural networks, only two are of interest here. The one is supervised learning, by far the most popular type of learning, while the second is reinforcement learning, which is discussed in detail in Appendix B.

A.2.1 Supervised Learning

The main ingredient of supervised learning is an external teacher, which provides the network with examples to learn from (Haykin, 1994). Whenever an input is presented to the network, the teacher knows the target response. The difference between the output of the network and desired output (as provided by the teacher), called the error, is then used by the supervised learning algorithm to make appropriate adjustments to the network parameters. This is done incrementally, step-by-step, until some minimum error criterion is reached.

The most well-known supervised learning algorithm is the backpropagation algorithm, which is widely used to train networks for pattern recognition, system modelling, etc (Haykin, 1994). Learning can take place on-line or off-line. When learning takes place off-line, a separate computational facility is used to train the network until it meets the desired performance, and the design is then "frozen" for implementation (Haykin, 1994). For on-line learning, the network is adjusted while in use, and learning takes place in real time.

A drawback of supervised learning is that after learning, its behaviour is set to a certain pattern. Therefore, if new, unknown conditions are encountered, it cannot learn from these unless they are explicitly incorporated into the teacher and training is again undertaken. Reinforcement learning, discussed next, overcomes this limitation by not requiring an external teacher.

A.2.2 Reinforcement Learning

Reinforcement learning, as opposed to supervised learning, does not take place by example. Rather, it is learning by trail-and-error via interaction with the environment. Learning is guided by a scalar performance index, or *reinforcement signal*. Appendix B further describes reinforcement learning.

Appendix B

Reinforcement Learning

Reinforcement learning is a form of artificial intelligence or machine learning, perhaps best defined as learning by *interaction* (Sutton & Barto, 1998). In essence, it is learning what to do in order to maximise a numerical reward signal (Sutton & Barto, 1998). The learner finds out what are the best actions in whichever situation by trying different ones; over time, the learner finds the actions which maximise the reward it receives. This can be contrasted with supervised learning, the most commonly used form of machine learning, where the learner is presented with examples which it needs to copy, i.e. learning by *example* (Sutton & Barto, 1998). A typical example of such a situation is the training of neural networks for pattern recognition or data modelling. In reinforcement learning, the learner is not told what actions are best, it finds that out by trial and error.

B.1 Basic elements

The reinforcement learning framework consists of two main elements: the *learning agent* and the *environment* (Sutton & Barto, 1998). As defined above, the learning agent interacts with the environment, and in doing so learns more about it. In addition to the two main elements, four sub-elements can be identified: a policy, reward function, value function, and (optional) model of the environment (Sutton & Barto, 1998).

The policy defines the behaviour of the agent at any time. It can be described as a mapping from the perceived environmental state to the actions to be taken in that state (Sutton & Barto, 1998). The policy is the core of the reinforcement learning agent, since it determines the agent's behaviour. The policy can be a simple function, lookup table, or even a search process, and can be deterministic or stochastic (Sutton & Barto, 1998). A neural network can be used as policy, with the environmental state as input, and the agent's actions as output. For this thesis, the policy is defined by a neural network. Since this is the case, the agent behaviour is determined by the specific values of the synaptic weights and biases.

The reward function essentially defines the goal of the reinforcement learning problem (Sutton & Barto, 1998). It returns a single number, called the reward, which indicates how desirable the current state is. The learning agent tries to maximise the total reward it receives in a given period of time. The reward the agent receives can be compared to pleasure and pain in biological systems.

The value function specifies the expected accumulated reward an agent can receive over time for starting in a given state (Sutton & Barto, 1998). Thus the value function represents the *long-term* desirability of a state. This can be contrasted with reward, which is *immediate.* Actions are chosen based on value judgments. Not all reinforcement learning methods require a value function. Some methods search directly in policy space using function optimization methods such as genetic algorithms, genetic programming or simulated annealing (Sutton & Barto, 1998). One of these methods is known as policy gradients, which is particularly well-suited to learning problems with high dimensions and continuous states and actions, such as chemical processes (Sehnke *et al.*, 2009). Another highly successful genre of algorithms for performing policy search are Evolution Strategies (ES) algorithms (Heidrich-Meisner & Igel, 2009).

B.2 Policy search

As laid out in Chapter 2, the policy search branch of reinforcement learning is chosen for application to process control in this study. Specifically, a policy gradient algorithm and evolution strategies algorithm are compared with one-another.

Policy search typically takes place episodically. An episode is a set-length period of time in which the agent is allowed to interact with the environment. During the episode, the reward is calculated at each time step. An example of what the reward could be is the distance between the agent and its goal. At the end of the episode, the reward obtained for each time step is summed to give the total episode reward R. This episode reward R is used by the algorithm to update the policy. As mentioned above, the policy in the case of this thesis is a neural network, and the parameters being updated by the algorithm are the synaptic weights and biases of the network. It is important to note that for the entire duration of the episode, the network parameters remain constant. Therefore, in a single episode, when the agent encounters state S twice, it will choose the corresponding action A in both instances as determined by the policy (network).

Given enough episodes, a good reinforcement learning algorithm should find a good solution to the problem it is facing. The speed at which such a solution is found, and the final quality of the solution obtained, is dependent on the effectiveness of the algorithm, the amount of episodes allowed for learning, and the complexity of the learning problem.

When considering episodic reinforcement learning in the way described above, it can be seen to reduce to black-box optimization. Black-box optimization is a type of optimization where nothing is known about the nature of the function being optimized. In the case of episodic reinforcement learning, the 'function' being optimized is the summed reward as determined by the behaviour of the learning agent (itself determined by the free parameters of the neural network policy) over the course of the episode. A single episode is therefore equivalent to a single function evaluation.

Appendix C

Publications

Presentation, 2011, Southern African Institute of Mining & Metallurgy Mineral Processing Conference, 4-5 August 2011, Vineyard Hotel, Claremont, Cape Town. S. Hunter & C. Aldrich. *Non-linear Control of a Ball Mill Grinding Circuit using Reinforcement Learning.*

Working Paper, to be submitted to Minerals Engineering (Elsevier), S. Hunter & C. Aldrich. 2011. *Non-linear Control of a Ball Mill Grinding Circuit using Reinforcement Learning.*

List of References

Alaradi, A.A. & Rohani, S. 2002. Identification and control of a riser-type fcc unit using neural networks. *Computers & Chemical Engineering*, 26(3):401 – 421. ISSN 0098-1354.

Balakrishnan, R. *et al.*. 1996. Neurocontrol: A literature survey. *Mathematical and Computer Modelling*, 23(1-2):101–117.

Bloch, G. & Denoeux, T. 2003. Neural networks for process control and optimization: Two industrial applications. *ISA Transactions*, 42:39–51.

Busoniu, L., Babuska, R., Schutter, B. & Ernst, D. 2010. *Reinforcement Learning and Dynamic Programming Using Function Approximators.* Automation and Control Engineering. CRC Press. ISBN 9781439821084.
Available at: `http://books.google.co.za/books?id=UGUqcl8_T9QC`

Camacho, E. & Bordons, C. 2004. *Model predictive control.* Advanced textbooks in control and signal processing. Springer. ISBN 9781852336943.

Chen, W.-C., Tai, P.-H., Wang, M.-W., Deng, W.-J. & Chen, C.-T. 2008. A neural network-based approach for dynamic quality prediction in a plastic injection molding process. *Expert Systems with Applications*, 35(3):843 – 849. ISSN 0957-4174.

Conradie, A. 2004. A neurocontrol paradigm for intelligent process control using evolutionary reinforcement learning. Ph.D. thesis, Stellenbosch University.

Conradie, A. & Aldrich, C. 2001*a*. Plant-wide neurocontrol of the Tennessee Eastman challenge process using evolutionary reinforcement learning. *Proceedings of the Third International Conference on Intelligent Processing and Manufacturing of Materials.*

Conradie, A. & Aldrich, C. 2005. Development of neurocontrollers with evolutionary reinforcement learning. *Computers and Chemical Engineering*, 30(1):1–17.

Conradie, A. & Aldrich, C. 2010. Neurocontrol of a multi-effect batch distillation pilot plant based on evolutionary reinforcement learning. *Chemical Engineering Science*, 65(5):1627 – 1643. ISSN 0009-2509.

Conradie, A., Miikkulainen, R. & Aldrich, C. 2002. Intelligent process control utilising symbiotic memetic neuro-evolution. *E-Commerce Technology, IEEE International Conference on*, 1:623–628.

Conradie, A.V.E. & Aldrich, C. 2001*b*. Neurocontrol of a ball mill grinding circuit using evolutionary reinforcement learning. *Minerals Engineering*, 14(10):1277 – 1294. ISSN 0892-6875.

da Cruz Meleiro, L.A., Zuben, F.J.V. & Filho, R.M. 2009. Constructive learning neural network applied to identification and control of a fuel-ethanol fermentation process. *Engineering Applications of Artificial Intelligence*, 22(2):201 – 215. ISSN 0952-1976.

Deisenroth, M.P., Rasmussen, C.E. *&* Peters, J. 2009. Gaussian process dynamic programming. *Neurocomputing*, 72(7-9):1508 – 1524. ISSN 0925-2312. Advances in Machine Learning and Computational Intelligence - 16th European Symposium on Artificial Neural Networks 2008, 16th European Symposium on Artificial Neural Networks 2008.

Diehl, M., Bock, H., Schlöder, J.P., Findeisen, R., Nagy, Z. *&* Allgöwer, F. 2002. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577 – 585. ISSN 0959-1524.

Dote, Y. *&* Ovaska, S. 2001. Industrial applications of soft computing: a review. *Proceedings of the IEEE*, 89(9):1243 –1265. ISSN 0018-9219.

Doya, K. 2000. Reinforcement learning in continuous time and space. *Neural Computation*, 12(1):219–245.

Ernst, D., Geurts, P. *&* Wehenkel, L. 2003. Iteratively extending time horizon reinforcement learning. *Machine Learning: ECML 2003*, pages 96–107.

Ernst, D., Glavic, M., Capitanescu, F. *&* Wehenkel, L. 2008. Reinforcement learning versus model predictive control: a comparison on a power system problem. *IEEE Transactions on Systems, Man and Cybernetics-Part B (to appear, 2008)*.

Govindhasamy, J.J., McLoone, S.F., Irwin, G.W., French, J.J. *&* Doyle, R.P. 2005. Neural modelling, control and optimisation of an industrial grinding process. *Control Engineering Practice*, 13(10):1243 – 1258. ISSN 0967-0661.

Hansen, N. *&* Kern, S. 2004. Evaluating the cma evolution strategy on multimodal test functions. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 282–291. Springer.

Hansen, N. *&* Ostermeier, A. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195.

Haykin, S. 1994. *Neural networks: a comprehensive foundation.* Prentice Hall PTR Upper Saddle River, NJ, USA.

Heaton, J. 2008. *Introduction to Neural Networks for Java.* 2nd edition. Heaton Research, Inc.

Heidrich-Meisner, V. *&* Igel, C. 2009. Neuroevolution strategies for episodic reinforcement learning. *Journal of Algorithms*, 64(4):152 – 168. ISSN 0196-6774. Special Issue: Reinforcement Learning.

Hosen, M.A., Hussain, M.A. *&* Mjalli, F.S. 2011. Control of polystyrene batch reactors using neural network based model predictive control (nnmpc): An experimental investigation. *Control Engineering Practice*, 19(5):454 – 467. ISSN 0967-0661.

Hunt, K., Sbarbaro, D. *et al.*. 1992. Neural networks for control systems–A survey* 1. *Automatica*, 28(6):1083–1112.

Hussain, M.A. 1999. Review of the applications of neural networks in chemical process control - simulation and online implementation. *Artificial Intelligence in Engineering*, 13:55–68.

Igel, C., Glasmachers, T. *&* Heidrich-Meisner, V. 2008. Shark. *Journal of Machine Learning Research*, 9:993–996.

Lu, C.-H. *&* Tsai, C.-C. 2008. Adaptive predictive control with recurrent neural network for industrial processes: An application to temperature control of a variable-frequency oil-cooling machine. *Industrial Electronics, IEEE Transactions on*, 55(3):1366 –1375. ISSN 0278-0046.

Lynch, A. *&* Rao, T. 1975. Modeling and scale-up of hydrocyclone classifiers. In *Proceedings of the 11th International Mineral Processing congress*, pages 245–269.

Marlin, T.E. 2000. *Process control: designing processes and control systems for dynamic performance*. 2nd edition. McGraw-Hill Higher Education.

Mevawalla, Z., May, G. *&* Kiehlbauch, M. 2011. Neural network modeling for advanced process control using production data. *Semiconductor Manufacturing, IEEE Transactions on*, 24(2):182 –189. ISSN 0894-6507.

Nelles, O. 2001. *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Engineering Online Library. Springer. ISBN 9783540673699.

Peters, J. *&* Schaal, S. 2008. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190.

Piche, S., Sayyar-Rodsari, B., Johnson, D. *&* Gerules, M. 2000. Nonlinear model predictive control using neural networks. *Control Systems, IEEE*, 20(3):53 –62. ISSN 1066-033X.

Rajamani, R. *&* Herbst, J. 1991*a*. Optimal control of a ball mill grinding circuit–I. Grinding circuit modeling and dynamic simulation. *Chemical Engineering Science*, 46(3):861–870.

Rajamani, R. *&* Herbst, J. 1991*b*. Optimal control of a ball mill grinding circuit–II. Feedback and optimal control. *Chemical Engineering Science*, 46(3):871–879.

Santos, V., Carvalho, F. *&* Souza Jr, M. 2000. Predictive control based on neural networks: an application to a fluid catalytic cracking industrial unit. *Brazilian Journal of Chemical Engineering*, 17:897 – 906.

Seader, J. *&* Henley, E.J. 2006. *Separation Process Principles*. 2nd edition. John Wiley & Sons, Inc.

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J. *&* Schmidhuber, J. 2009. Parameter-exploring policy gradients. *Neural Networks*.

Skogestad, S. 1997. Dynamics and control of distillation columns: A tutorial introduction. *Chemical Engineering Research and Design*, 75(6):539 – 562. ISSN 0263-8762. Distillation.

Skogestad, S. *&* Postlethwaite, I. 2005. *Multivariable feedback control: analysis and design*. John Wiley. ISBN 9780470011676.

Sutton, R.S. *&* Barto, A.G. 1998. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press. ISBN 0262193981.

Syafiie, S., Tadeo, F., Martinez, E. *&* Alvarez, T. 2009. Model-free control based on reinforcement learning for a wastewater treatment problem. *Applied Soft Computing*.

The Mathworks, Inc. 2011. R2011b Documentation - System Identification Toolbox. Available at: `http://www.mathworks.com/help/toolbox/ident/`

Wawrzynski, P. 2009. Real-time reinforcement learning by sequential actor-critics and experience replay. *Neural Networks*, 22(10):1484 – 1497. ISSN 0893-6080.

Wills, B. & Napier-Munn, T. 2006. *Wills' Mineral Processing Technology.* Butterworth-Heinemann.

Yu, W., Poznyak, A.S. & Alvarez, J. 1999. Neuro control for multicomponent distillation column. IFAC.

Yuzgec, U., Becerikli, Y. & Turker, M. 2008. Dynamic neural-network-based model-predictive control of an industrial baker's yeast drying process. *Neural Networks, IEEE Transactions on*, 19(7):1231 –1242. ISSN 1045-9227.