ROBUST NONLINEAR MODEL PREDICTIVE CONTROL OF A CLOSED RUN-OF-MINE ORE MILLING CIRCUIT

by

Lodewicus Charl Coetzee

Submitted in partial fulfilment of the requirements for the degree

Philosophae Doctoral (Electronic Engineering)

in the

Faculty of Engineering, the Built Environment and

Information Technology

UNIVERSITY OF PRETORIA

June 16, 2009

SUMMARY

Title:	Robust Nonlinear Model Predictive Control of a Closed Run-Of-Mine Ore Milling Circuit
By:	Lodewicus Charl Coetzee
Promoter:	Professor I.K. Craig
Department:	Department of Electrical, Electronic and Computer Engineering
Name of degree:	Philosophae Doctoral (Electronic Engineering)

This thesis presents a robust nonlinear model predictive controller (RNMPC), nominal nonlinear model predictive controller (NMPC) and single-loop proportional-integral-derivative (PID) controllers that are applied to a nonlinear model of a run-of-mine (ROM) ore milling circuit. The model consists of nonlinear modules for the individual process units of the milling circuit (such as the mill, sump and cyclone), which allow arbitrary milling circuit configurations to be modelled easily.

This study aims to cast a complex problem of a ROM ore milling circuit into an RNMPC framework without losing the flexibility of the modularised nonlinear model and implement the RNMPC using open-source software modules.

The three controllers are compared in a simulations study to determine the performance of the controllers subject to severe disturbances and model parameter variations. The disturbances include changes to the feed ore hardness, changes in the feed ore size distributions and spillage water being added to the sump.

The simulations show that the RNMPC and NMPC perform better than the PID controllers with regard to the economic objectives, assuming full-state feedback is available, especially when actuator constraints become active. The execution time of the RNMPC, however, is much too long for real-time implementation and would require further research to improve the efficiency of the implementation.

Keywords: Run-of-mine ore milling circuit, Robust Nonlinear Model Predictive Control, ROM, RNMPC.

OPSOMMING

Titel:	Robuuste nie-lineêre-model voorspellende beheer van ’n geslote maalkring wat onbehandelde erts maal
Deur:	Lodewicus Charl Coetzee
Promoter:	Professor I.K. Craig
Departement:	Departement van Elektriese, Elektroniese and Rekenaaringenieurswese
Naam van graad:	Philosophae Doctoral (Elektroniese Ingenieurswese)

Die tesis beskryf ’n robuuste nie-lineêre-model voorspellende beheerder (RNMVB), ’n nominale nie-lineêre-model voorspellende beheerder (NMVB) en proporsioneel-integraal-afgeleide (PIA)-beheerders wat toegepas word op ’n nie-nielineêre-model van ’n geslote maalkring wat onbehandelde erts maal. Die model bestaan uit nie-lineêre modules vir die individuele proses-eenhede (soos die meule, opvangbak en sikloon) wat dit moontlik maak om arbitrêre proseskonfigurasies te modelleer.

Die studie beoog om ’n komplekse probleem van ’n geslote maalkring wat onbehandelde erts maal in ’n RNMVB raamwerk te plaas sonder om die voordeel van ’n modulêre nie-lineêre-model te verloor. Die RNMVB-implimentering gebruik oop-bronkode sagtewaremodules.

Die drie beheerders word met mekaar vergelyk deur van ’n simulasiestudie gebruik te maak om die werkverrigting van die beheerders te bepaal as beduidende steurings en model-parametervariasies teenwoordig is. Die steurings sluit in veranderinge in die hardheid en groottesamestelling van die erts wat na die meule gevoer word, asook water wat in die opvangbak gestort word.

Die simulasies wys dat die RNMVB en NMVB beter werkverrigting lewer as die PIA-beheerders as hulle vergelyk word wat betref die ekonomiese doelwitte, veral as die aktueerders hulle limiete bereik. Die simulasies neem aan dat alle proses toestandsveranderlikes terugvoerbaar is. Die berekeninge van die RNMVB neem te lank om dit te kan implementeer op ’n werklike aanleg en vereis verdere navorsing om die implimentering te bespoedig.

Sleutelwoorde: Maalkring wat onbehandelde ers maal, Robuuste Nie-lineêre-model Voorspellende Beheer.

ACKNOWLEDGEMENT

I thank God for blessing me with the opportunity to study.

To my wife, Nicole, I say thank you very much! Without your love, support and understanding during this time, this study would not have been possible.

I owe my parents a debt of gratitude for their support and understanding throughout all the years of my studies.

I sincerely thank my supervisor, Prof. Craig of the University of Pretoria, for his guidance over all these years, his insight into control, helping with this research and especially for keeping me on the right track.

I thank Dr. Eric Kerrigan of Imperial College, London, for his insights into robust nonlinear model predictive control theory and taking the time to explain it to me.

I thank Dr. Hulbert of Mintek for providing the nonlinear model of the ore milling circuit used in this thesis and for his practical insight into the process and the control challenges.

TABLE OF CONTENTS

1 INTRODUCTION **1**

1.1 MILL CIRCUIT DESCRIPTION 4

1.2 OBJECTIVES IN MILL CONTROL 5

1.3 AIMS AND OBJECTIVES 6

1.4 ORGANISATION 7

2 MILLING THEORY AND MODELLING **9**

2.1 INTRODUCTION 9

2.2 THEORY OF MILLING 9

2.2.1 Introduction 9

2.2.1.1 Mills 10

2.2.1.2 Hydrocyclones 14

2.2.1.3 Sump 15

2.2.2 Process of breakage 15

2.2.2.1 Size distribution 16

2.2.3 Motion of the load 16

2.2.4 Forces causing breakage 17

2.2.5 Breakage mechanisms 18

2.3 MILLING MODELLING 18

2.3.1 Discrete element method 19

2.3.2 Population balance models 20

2.3.2.1 Product discharge 21

2.3.2.2 Breakage rate r_i 23

2.3.2.3 Breakage distribution function (appearance function a_{ij}) . 24

2.3.2.4 Power draw model 25
2.3.2.5 Hydrocyclone models 28
2.3.3 Other models 28
2.3.3.1 Cumulative breakage rate model 28
2.3.3.2 Transfer function step-test model 28
2.3.3.3 Neural networks 29
2.3.4 Mintek mill circuit model modules 30
2.3.4.1 Feeder module 30
2.3.4.2 Mill module 31
2.3.4.3 Mixed-sump module 34
2.3.4.4 Hydrocyclone module 35
2.4 CONCLUSION 36

3 MODEL PREDICTIVE CONTROL 37
3.1 INTRODUCTION 37
3.2 HISTORICAL BACKGROUND 40
3.3 STABILITY OF MPC 42
3.3.1 Stability conditions for model predictive controllers 43
3.3.2 Terminal state MPC 45
3.3.3 Terminal cost MPC 45
3.3.4 Terminal constraint set MPC 46
3.3.5 Terminal cost and constraint set MPC 47
3.4 ROBUST MPC - STABILITY OF UNCERTAIN SYSTEMS 47
3.4.1 Stability conditions for robust MPC 48
3.4.2 Open-loop min-max MPC 49
3.4.3 Feedback robust MPC 50
3.5 ROBUST NONLINEAR MPC FORMULATIONS 52
3.5.1 Lyapunov-based robust model predictive control 52
3.5.2 Reachable set methods 53
3.5.3 Closed-loop min-max predictive control 53
3.5.4 Open-loop min-max predictive control 53

3.5.5 Linear embedding of nonlinear models 54
3.6 NONLINEAR MODEL PREDICTIVE CONTROL 55
3.7 ROBUST NONLINEAR MODEL PREDICTIVE CONTROL 57
3.7.1 Parameter uncertainty description 57
3.7.2 Direct approximate robust counterpart formulation 58
3.7.3 RNMPC implementation 61
3.8 STATE OBSERVERS 67
3.9 CONCLUSION 68

4 PID CONTROL 69
4.1 INTRODUCTION 69
4.2 PI CONTROL WITH ANTI-WINDUP 70
4.3 LINEARISED MODELS FOR SIMC TUNING METHOD 71
4.3.1 PSE – CFF model 72
4.3.2 LOAD – MFS model 72
4.3.3 SLEV – SFW model 73
4.4 SIMC TUNING METHOD 74
4.4.1 Simplifying first-order transfer function models 74
4.5 IMPLEMENTATION 75
4.5.1 PI controller for the PSE-CFF loop 76
4.5.2 PI controller for the LOAD-MFS loop 76
4.5.3 PI controller for the SLEV-SFW loop 77
4.6 SUMMARY 77

5 MILLING CIRCUIT CONTROL SIMULATION STUDY 78
5.1 INTRODUCTION 78
5.2 PERFORMANCE METRICS 79
5.3 SIMULATION RESULTS 84
5.3.1 Simulation parameters 84
5.3.2 Constant setpoint following and disturbance rejection 85
5.3.3 Reduced PSE setpoint to 75% and 70% < 75μm 89
5.3.4 Step change of -5% and -10% in PSE setpoint 90

5.3.5 Regulate PSE, LOAD and Throughput 90
5.4 DISCUSSION 116
5.5 SIMULATION SUMMARY 117

6 CONCLUSIONS AND FURTHER WORK 123
6.1 SUMMARY AND EVALUATION 123
6.1.1 Strong points 123
6.1.2 Drawbacks 125
6.2 FURTHER WORK 127

REFERENCES 129

A SOFTWARE IMPLEMENTATION 142
A.1 INTRODUCTION 142
A.2 IMPLEMENTATION 143
A.2.1 IPOPT TNLP class methods 144
A.2.1.1 Class constructor method `MyNLP::MyNLP` 144
A.2.1.2 Method `get_nlp_info` 145
A.2.1.3 Method `get_bounds_info` 146
A.2.1.4 Method `get_starting_point` 147
A.2.1.5 Method `eval_f` 148
A.2.1.6 Method `eval_grad_f` 148
A.2.1.7 Method `eval_g` 148
A.2.1.8 Method `eval_jac_g` 149
A.2.1.9 Method `finalize_solution` 151
A.2.2 RNMPC specific methods 151
A.2.2.1 Method `next_run` 151
A.2.2.2 Method `get_u` 152
A.2.2.3 Method `SundialsRun` 152
A.2.2.4 Method `CppADThreadRun` 157
A.2.2.5 Method `CppADNoThreadRun` 157
A.2.2.6 Method `CppADInit` 157
A.2.2.7 Method `CppADRun` 159
A.2.3 Main execution loop 165

B AUXILIARY RESULTS **170**

B.1 AUXILIARY SIMULATION RESULT 170

B.1.1 Constant setpoint following and disturbance rejection 170

B.1.2 Reduced PSE setpoint to 75% and 70% < 75μm 170

B.1.3 Regulate PSE, LOAD and Throughput 171

B.2 ADDITIONAL SIMULATION SCENARIOS 180

B.2.1 Setpoints on PSE, LOAD, POWER and RHEOLOGY 180

B.2.2 Increase the weighting on the MVs for objective function with only PSE and LOAD setpoints 180

B.2.3 Increase weightings on the MVs for objective functions with PSE, LOAD and THROUGHPUT setpoints 181

B.3 SIMULATION SUMMARY 194

C PID TUNING WITH INTERACTIONS **197**

C.1 INTRODUCTION 197

C.2 SUMP AND CYCLONE MODELS 198

C.2.1 PSE – SFW model 198

C.2.2 SLEV – CFF model 199

C.2.3 Interacting sump and cyclone model 199

C.3 FBS TUNING METHOD 200

C.4 PID CONTROLLER DESIGN 202

C.4.1 Design scenario 1 202

C.4.2 Design scenario 2 203

C.4.3 Design scenario 3 206

C.4.4 Design scenario 4 206

C.4.5 Design scenario 5 208

C.5 CONCLUSION 209

D SIMULATION SUMMARY **211**

LIST OF ABBREVIATIONS

ADMC	Adaptive Dynamic Matrix Control
AG	Autogenous
ARX	Auto-Regressive with eXogenous input
BC	The rate of steel Ball Consumption [$m^3/hour$]
CFD	Computational Fluid Dynamics
CFF	The volumetric flow-rate of slurry from the sump to the cyclone. [$m^3/hour$]
CPPAD	Software to perform automatic differentiation of software code through operator overloading.
DCS	Distributed Control System
DEM	Discrete Element Method
DMC	Dynamic Matrix Control
FBS	Frequency Based Specifications
FP	The rate of Fines Production [$m^3/hour$]
GPC	Generalized Predictive Control
IDCOM	IDentification and COMmand
INA	Inverse Nyquist Array
IPOPT	Software to solve large scale nonlinear programming problems.
ISpS	Input-to-State practical Stability
ISS	Input-To-State Stability
JKMRC	Julius Kruttschnitt Mineral Research Centre

LMI	Linear Matrix Inequality
LOAD	The total charge of the mill. [%]
LQ	Linear Quadratic
MFB	The feed-rate of steel balls to the circuit. [$^{\text{tons}}/_{\text{hour}}$]
MFS	The feed-rate of ore to the circuit (consists of rocks, coarse and fine ore). [$^{\text{tons}}/_{\text{hour}}$]
MHE	Moving Horizon Estimators
MHSO	Moving Horizon State Observers
MIMO	Multi-Input-Multi-Output
MIW	The volumetric flow-rate of water to the circuit. [$^{\text{m}^3}/_{\text{hour}}$]
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Controller
ODEs	Ordinary Differential Equations
ORC	Override Control
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
PSE	Product particle-size. [$\% < 75\mu$m]
QDMC	Quadratic Dynamic Matrix Control
QP	Quadratic Programming
RBF	Radial Basis Function
RBF-ARX	Radial Basis Function – Auto-Regressive with eXogenous input.
RC	The rate of Rock Consumption [$^{\text{m}^3}/_{\text{hour}}$]
RHC	Receding Horizon Control
RMHSO	Robust Moving Horizon State Observer
RNMPC	Robust Nonlinear Model Predictive Control
ROM	Run-of-Mine
RPM	Revolutions Per Minute

Runge-Kutta	Method to solve ordinary differential equations numerically.
SAG	Semi-Autogenous
SANE	Symbiotic Adaptive Neuro-Evolution
SDP	Semidefinite Programming
SFW	The volumetric flow-rate of extra water to the sump. [$^{m^3}/_{hour}$]
SID	System Identification
SIMC	Skogestad Internal Model Control
SISO	Single-Input-Single-Output
SLEV	The level of the sump. [m^3]
SMOC	Shell Multi-variable Optimizing Control
SPH	Smoothed Particle Hydrodynamics
THROUGHPUT	Product throughput consisting of coarse and fine solids. [$^{tons}/_{hour}$]
TITO	Two-Input-Two-Output

CHAPTER 1

INTRODUCTION

In recent years, the industrial use of model predictive control (MPC) has increased dramatically and the development of MPC resulted in increased capabilities (Qin and Badgwell, 2003). It is not surprising that recent milling control efforts focus on applying MPC to milling circuits. MPC has a number of desirable characteristics when applied to processes with (Chen *et al.*, 2007*a*)

- large time delays,
- time varying characteristics,
- nonlinearities, and
- constraints on the manipulated and controlled variables

compared to other control formulations. Better performing control is becoming more essential as surge capacity is reduced in modern plant design (Karageorgos *et al.*, 2006). The constraints handling of MPC makes it suitable in preventing mill under-/overload events and the improved performance has a positive effect on the economies of the grinding circuit (Bouche *et al.*, 2005, Van Drunick and Penny, 2006).

Chen *et al.* (2007*a*) applied hybrid control in the form of override control (ORC) and MPC. Their aim was to control only the product particle size by manipulating the feed rate of ore to the circuit. ORC is used to prevent overload of the mill and obtain an optimal feed rate setpoint. MPC is employed for handling the constraints and time delay of the process.

Chen *et al.* (2009) applied expert system-based adaptive dynamic matrix control (ADMC) to control a ball mill grinding circuit. The controller is a dynamic matrix controller with extra switching logic. The switching logic is based on an expert system to determine the ore hardness in the grinding circuit and choose the appropriate model for the dynamic matrix controller. The advantage of their formulation is that the switching logic can be made arbitrarily complex, without affecting the control computational burden.

Chen *et al.* (2007*b*) showed that linear MPC, using a complex four-input-four-output model and complex three-input-three-output model (Chen *et al.*, 2008) and applied to a real industrial plant (unnamed iron ore concentrator plant), provides better long-term stable performance than multi-loop decoupled proportional-integral-derivative (PID) control. Ramasamy *et al.* (2005) did comparative simulation studies between de-tuned multiloop proportional-integral (PI) controllers and unconstrained and constrained MPC on a two-input-two-output linear model of a ball mill grinding circuit. Their findings were that MPC performed well under different operating conditions compared to PI control, which produced oscillations and slow settling times.

Valenzuela *et al.* (1994) compared an early form of MPC called dynamic matrix control (DMC) to PI control and learning automata. The three control methods were simulated on a grinding circuit and the authors concluded that DMC provided the best performance of the three control schemes.

Neesse *et al.* (2004) presents a control strategy for hydrocyclones to increase the operating range of the milling circuit. It increases solids recovery from the hydrocyclone to minimise the recirculating load as well as overgrinding.

Galan *et al.* (2002) presented two H_∞ robust linear controllers for a single-input-single-output (SISO) system. The power draw of the mill was controlled by manipulating the feed rate of ore to the mill. The authors found that the design of the controllers were more systematic than for PI control with guarantees on performance and stability. The methodology can also easily be extended to multi-input-multi-output (MIMO). Craig and MacLeod (1995, 1996) implemented a MIMO robust controller on a three-input-three-output model of a run-of-mine (ROM) grinding circuit based on the μ-methodology. They found that regulation of the mill load and product particle size was satisfactory, but the sump level regulation was inadequate. Their conclusion was that the cost of the extra modelling required to quantify the uncertainties needed to implement a μ-controller was too high compared to inverse Nyquist array (INA) controllers that only needed a nominal model and some online tuning. The μ-controller would probably still require online tuning to provide the best performance.

Neural network-based control of grinding circuits is also investigated by Bhaumik *et al.* (1999), Conradie and Aldrich (2001), Duarte *et al.* (1999*a*, 2001). Duarte *et al.* (1999*a*, 2001) used three neural networks in the loop, the first one acting as the estimator for the states, the second one as the controller and the third providing predictions of the state trajectory for predictive control. Conradie and Aldrich (2001) use evolutionary reinforcement learning neural control that uses the symbiotic adaptive neuro-evolution (SANE) algorithm. The controller algorithm learns by applying control moves to the system and gets a reward based on how well the control move contributed to achieving the goal. The controller can therefore potentially apply unwanted control moves in order to learn. It is better to train the controller on simulation models to an acceptable level before applying the controller to a real plant.

Najim *et al.* (1995) presented an adaptive controller for a grinding circuit and implemented it for discrete (SISO) loops to reduce the number of parameters to estimate. The authors found that the controllers performed satisfactorily despite the variable interactions in the system. They concluded that adaptive controllers are easier to design and use than fixed structure multivariable controllers. Xingyan *et al.* (1992) presented an adaptive predictive controller that shows good robustness to delay mismatch in the process and the model, as well as robustness to load disturbances.

(Rajamani and Herbst, 1991*a*, *b*) developed a simplified nonlinear model for the grinding circuit and compared optimal control to PI control. They found the open-loop optimal controller to provide better performance than PI control.

Supervisory control is a second layer of control that optimises the process by calculating optimal setpoint values for the regulatory control layers. Radhakrishnan (1999) presented model-based supervisory control to optimise an economic objective function on-line. Borell *et al.* (1996) presented supervisory control based on expert systems with IF-THEN statements to optimise mill power that increased milling capacity.

Pomerleau *et al.* (2000) studied four control formulations applied to grinding circuits. The four controllers are decentralised PID, algebraic internal model controllers with an explicit decoupler, full multivariable predictive controllers and distributed adaptive predictive controllers. The authors have drawn the following conclusions:

- *"Fixed structure controllers e.g., PID perform as well as model-based controllers for processes where the delay is relatively small compared to the dominant time constant ($\theta < 5T$) since the models are rarely of order higher than two.*
- *Distributed controllers perform as well as multivariable controllers in regulation if the coupling of the process is taken into account in the design and the proper pairing is used for the dynamics required.*
- *Algebraic multivariable controllers perform as well as optimal multivariable controllers and they have exactly the same limitations e.g., perfect decoupling might be impossible.*
- *Algebraic tuning methods may require more know how than optimal controllers, but are easier to implement on an industrial distributed control system (DCS).*
- *Predictive controller and algebraic controllers perform equally in a stochastic environment if there is no noise model available.*
- *Adaptive controllers perform better than fixed controllers for parametric disturbances or soft nonlinearities. It must be noted though that identification is very difficult in regulation where external disturbances act on the process. They also have the advantage of facilitating the tuning principally for distributed controllers."* (Pomerleau *et al.*, 2000).

Duarte *et al.* (1999*b*) compared the performance of five multivariable controllers on a grinding circuit, viz. extended horizon adaptive control, pole-placement adaptive control, model reference adaptive control, direct Nyquist array control and sequential loop closing control. The authors studied the controllers theoretically as well as implemented them on a real plant. Their conclusion was that extended horizon adaptive control provided the best performance, but that multivariable control in general improves grinding performance. Duarte *et al.* (1999*b*) provides a thorough review of the literature on multivariable control applied to grinding circuits.

Earlier work took unmodelled dynamics, parameter variation and nonlinearities into account by using adaptive control (Duarte *et al.*, 1999*b*) to improve performance. Robust nonlinear MPC (RNMPC) can ensure stability and sustained performance while satisfying constraints on the inputs and the states of nonlinear uncertain systems. The uncertainties in the nonlinear model can be the result of parameter variations, unmodelled dynamics and external disturbances (Mhaskar and Kennedy, 2008).

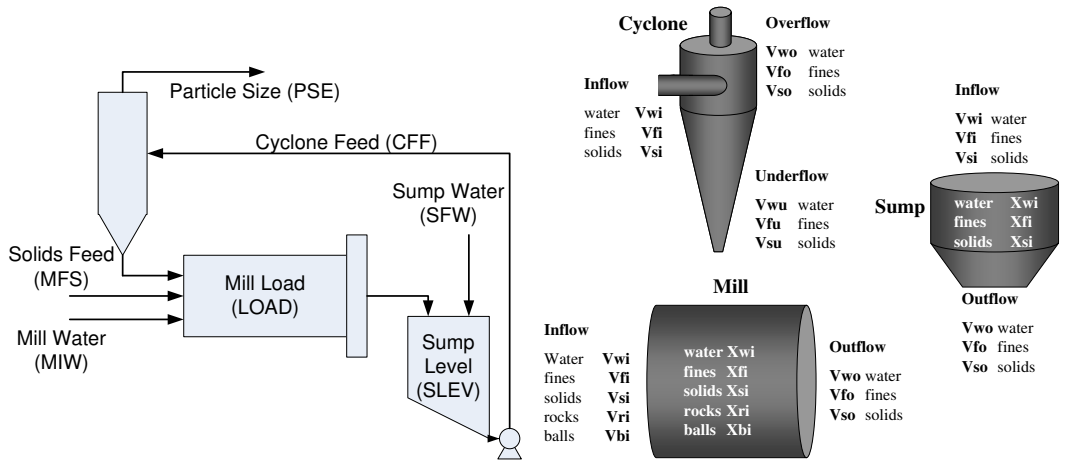
1.1 MILL CIRCUIT DESCRIPTION

The theoretical part of this study will assume a ROM milling circuit for gold-bearing ore. The circuit is closed and consists of a semi-autogenous (SAG) mill. A typical mill has dimensions of 5 m in diameter and a length of 9 m (Stanley, 1987).

The circuit is fed gold-bearing ore at about 100 $^{\text{tons}}/_{\text{hour}}$ and grinds it to give a product with 75% to 80% of the particles smaller than a mesh size of 75 μm. The mill discharges through a grate into a sump. Water is added to the sump to dilute the pulp from the mill. The diluted slurry is then pumped to a hydrocyclone that will separate the product from the out-of-specification material.

The hydrocyclone has an internal diameter of 1 m. The out-of-specification material is discharged from the cyclone underflow back to the mill for further grinding. Feed ore, the underflow of the hydrocyclone and water constitute the mill feed. The product is then sent downstream for further liberation of the gold from the ore.

The variables of the mill (Figure 1.1) that can be controlled are the level of the slurry in the sump (SLEV), the product particle-size (PSE) and the mass of material in the mill (LOAD). The inputs to the mill that can be manipulated are the feed-rate of water to the sump (SFW), the flow-rate of slurry to the cyclone (CFF), the feed-rate of solids to the mill (MFS) and the flow-rate of water to the mill inlet (MIW).



(a) Mill Circuit (b) Mill Modules

Figure 1.1: ROM closed ore milling circuit.

1.2 OBJECTIVES IN MILL CONTROL

The control of the milling circuit has multiple objectives, firstly to stabilise the system and secondly to optimise the economics of the process (Hulbert, 1989). The economic objective is divided into sub-objectives that each contributes to the overall economic objective of the milling process. A set of possible sub-objectives for the milling circuit is to (Craig and MacLeod, 1995):

- improve product quality
  - by increasing grind fineness, and
  - decreasing the fluctuations in product size,
- maximise throughput,
- minimise the amount of steel that is consumed for each ton of fines produced, and
- minimise the power consumed for each ton of fines produced, etc.

The objectives above are interrelated and require trade-offs to be made. There is a trade-off between the particle size of the product and the throughput of solids (objectives 1a and 2). More gold can be extracted at a finer product size (objective 1a), but the variation in particle size also influences recovery (objective 1b).

It is assumed that the throughput of the mill is maximised when it draws maximum power from the mill motor. The $\Delta LOAD/\Delta MFS$ input-output pair is therefore often under power peak seeking control (Craig *et al.*, 1992*b*). This is contrary to objective 4, which is to minimise electrical power, but given the value of the milling product versus the cost of electricity, objective 4 is usually considered less important.

Objectives 1 and 3 are interrelated. Steel is added to the mill to stabilise the conditions inside the mill and also to increase throughput. A controller that is capable of stabilising the particle size, will reduce the need for steel and thus objective 3 will be addressed when objective 1b is met.

A possible control strategy is to maximise throughput at a certain particle size setpoint. This strategy considers both objectives 1 and 2. The particle size setpoint may be determined by throughput targets or if throughput is not a consideration, the particle size can be optimised. There is a trade-off between throughput and grind, and grind and residue (product that is not recovered) (Craig *et al.*, 1992*b*). The main aim of control would usually be to increase throughput while keeping the grind constant.

1.3 AIMS AND OBJECTIVES

The main aim of this thesis is to determine the feasibility of applying robust nonlinear model predictive control to an industrial problem such as the run-of-mine ore milling circuit. To this aim:

- A robust nonlinear model predictive controller needs to be synthesised, which explicitly takes model uncertainty into consideration during controller synthesis. The objectives stated in Section 1.2 should be included in the objective function of the controller.
- The controller should be verified through a simulation study of the closed-loop system in order to evaluate the performance of the controller :
  - in the presence of uncertainty about the feed ore hardness and feed ore size distribution, and
  - disturbances such as spillage water being added to the sump.
- The performance of the RNMPC is compared to a nominal nonlinear model predictive controller and single-loop PI controllers to gauge the advantages of using robust nonlinear model predictive control.

This dissertation contributes the following:

- Mill model in the form required for control.
- Linearised models of the nonlinear model for tuning the PI controllers.
- Synthesis of PI controllers with anti-windup for the ROM ore milling circuit.

- Synthesis of a nonlinear model predictive controller (one that does not take model uncertainty into account) for the ROM ore milling circuit.
- Synthesis of an *open-loop* robust nonlinear model predictive controller for the ROM ore milling circuit. Open-loop model predictive control differs from closed-loop model predictive control in assuming that there is no feedback over the prediction horizon during the prediction and optimisation calculations. This therefore leads to more divergent state trajectories during prediction and optimisation when disturbances and uncertainties are present in open-loop formulations compared to closed-loop formulations, resulting in controllers with more conservative performance and with smaller feasibility regions.
- Simulation study to compare the stability and performance of the above-mentioned controllers under model mismatch situations
  - using severe parameter uncertainty with a uniform distribution and zero mean to establish a baseline,
  - adding a large step change that increases the feed ore hardness and drives some variables to their constraints,
  - adding a large step change in the feed ore size distribution that increases the number of large particles in the feed, and
  - adding a disturbance to the sump by adding a step increase to the sump feed water that simulates spillage water being added to the sump.

1.4 ORGANISATION

Chapter 2 provides a brief overview of the ROM ore milling circuit process and an overview of the modelling of the main process units in the ore milling circuit such as the mill and the cyclone.

Chapter 3 provides an overview of the theory of stability of MPC and the development of the RNMPC theory. The chapter continues by taking an in-depth look at the RNMPC formulation.

Chapter 4 outlines the theory of the single-loop PI controllers with anti-windup, Skogestad Internal Model Control (SIMC) tuning method and the application of the theory to the linearised process models of Section 4.3 to synthesise the PI controllers.

Chapter 5 provides an in-depth study of the robust and nominal nonlinear model predictive controllers as well as the single-loop PI controllers applied to the nonlinear model of the ROM ore milling circuit. Practical scenarios are investigated in an attempt to quantify the

effects of severe feed ore disturbances and parameter variations on the closed-loop performance.

Chapter 6 provides a short summary of the thesis, some conclusions drawn from the simulation studies and recommendations for further work regarding the development of an RNMPC for the ROM ore milling circuit.

Addendum A outlines the technical details related to the software implementation of the RNMPC control algorithm.

Addendum B shows some additional results relating to simulation scenarios described in Chapter 5 as well as simulation results for additional scenarios that were not covered in Chapter 5.

Addendum D provides a table that summarises the simulation results of both Chapter 5 and Addendum B.

CHAPTER 2

MILLING THEORY AND MODELLING

2.1 INTRODUCTION

Controlling a process usually requires a mathematical model of the process for design and simulation purposes. Model-based control further relies on an internal model of the process to calculate the control moves. A mathematical model of the ROM ore milling circuit is therefore essential in designing and simulating control strategies. There are two approaches to modelling grinding circuits, firstly a process design and optimisation approach and secondly a control and estimation approach.

Process engineers want as much information as possible about the operation of the mill and grinding circuit as a whole. Simulation models provide insights into the mechanisms of breakage and material flow inside the mill (Hinde, 2007). The simulation models consist of a large number of states and parameters in order to model the size distributions and breakage distributions.

Control engineers have only limited measurements available from the mill and circuit (Apelt *et al.*, 2002), that limit the number of states and parameters that can be estimated. Control engineers therefore prefer simple models with only a small number of states and parameters, while still capturing the essential dynamics for control purposes.

2.2 THEORY OF MILLING

2.2.1 Introduction

Milling or grinding reduces the size of ore by allowing the ore to tumble freely in a rotating or gyrating container, which causes a breaking action to be applied to the ore. The product

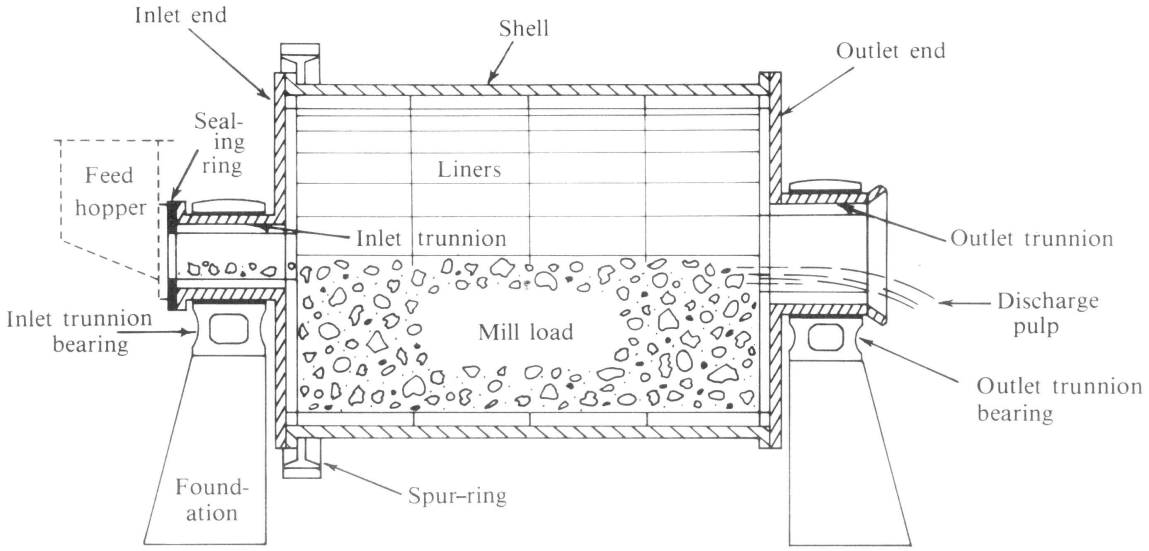


Figure 2.1: Mill Schematic (Stanley, 1987).

size is determined by the size of the feed to the mill and by relative probabilities of breakage of the individual size fractions inside the mill (Stanley, 1987).

In gold ores, the valuable gold usually constitutes a very small part of the volume of the ore. The rest of the ore is mostly worthless. In order to extract the gold from the ore, the ore has to be ground down to a fine powder. This action or process of reducing the size of ore to minute particles or fractions is called comminution (Stanley, 1987).

Comminuting ore therefore

- makes it more usable because of the reduced particle size, and
- liberates the different components in the ore from each other, which aids in the subsequent separation of the components by down-stream processes.

The separation of the valuable material from the gangue, or worthless material, is achieved through chemical or physical mineral recovery processes. It is reasonable to assume that the closer the maximum discharge particle size from the milling process is to the grain size of the valuable material, the more efficient the downstream recovery processes will be, because the valuable material will be better liberated from the ore (Stanley, 1987).

The desired fineness of the particle size obtained from the milling process is also determined by the economics of the process. The gain in recovery is weighed against the cost of finer comminution (Stanley, 1987).

2.2.1.1 Mills

Tumbling mills are usually cylindrically shaped machines made of steel (Figure 2.1). The cylindrical container is rotated about its horizontal axis by some form of drive, usually elec-

trical. Both ends of the cylinder are closed off with either cast iron or steel, with the centres of the ends extruding to from hollow trunnions that allow ore to enter the mill and pulp to exit. The trunnions are supported by trunnion bearings that allow the mill to rotate, while the bearings are mounted on some form of foundation (Stanley, 1987).

The ore enters the mill through the inlet trunnion by some form of feeder, which is a device that can continuously feed ore into the inlet trunnion from external sources. It can be a static device that allows the ore to flow by means of gravity (such as sprout and hopper feeders) or it may be attached to the mill that allows it to rotate and scoop the feed ore into the inlet trunnion (such as scoop and drum feeders) (Stanley, 1987).

The inside surfaces of the mill are protected against wear by mill liners. The mill liners are important for two reasons. Firstly, the mill liners are consumables that influence the operational cost of the mill, because they wear away over time. The operational costs associated with mill liners can be minimised by maximising the time before the liners need to be replaced, as well as minimising the cost of relining. Secondly, it is partly responsible for the effectiveness of the mill. The shell liners have different shapes that aid in lifting the ore and thus transferring energy to the ore for grinding. Ensuring that the lifters remain efficient over the lifetime of the liners is therefore also important (Chandramohan and Powell, 2006, McBride and Powell, 2006).

The pulp exits the mill through the outlet trunnion and there are several methods to facilitate this. There is a simple overflow mill (Figure 2.1) where the pulp overflows into the outlet trunnion aided by the rotation of the mill. Secondly, a screen-and-discharge mechanism can be employed, where the pulp passes through a screen that limits the particle size to the screen aperture size and lifter bars lift the pulp to the outlet trunnion (Figure 2.2). A peripheral discharge and open-ended discharge can also be employed to eliminate the need for lifting pulp through the outlet trunnion. The peripheral discharge (Figure 2.3) consists of grates on the side of the mill at the outlet end through which the pulp flows. With an open-ended discharge, the mill outlet end is not closed off by a solid side, but rather by a screen or grate, that allows pulp to exit through the whole outlet opening. The open-ended mill cannot be supported by an outlet trunnion, but rather by a roller system (Figure 2.4) or slipper bearing (Figure 2.5) (Stanley, 1987). The different discharge mechanisms influence the performance of the mill. If a slurry pool forms inside the mill, the grinding performance is degraded (Latchireddi and Morrell, 2003*a*, *b*).

There are different mill types:

Ball Mills use steel balls ranging from 100 mm down to 50 mm as grinding medium. Their primary role is for primary milling after crushing if the run-of-mine material has insufficiently sized pebbles to support autogenous milling.

Rod Mills use steel rods with diameters of about 100 mm and a length that is about 100 mm shorter than the length of the mill. Rod mills can handle coarser material than other

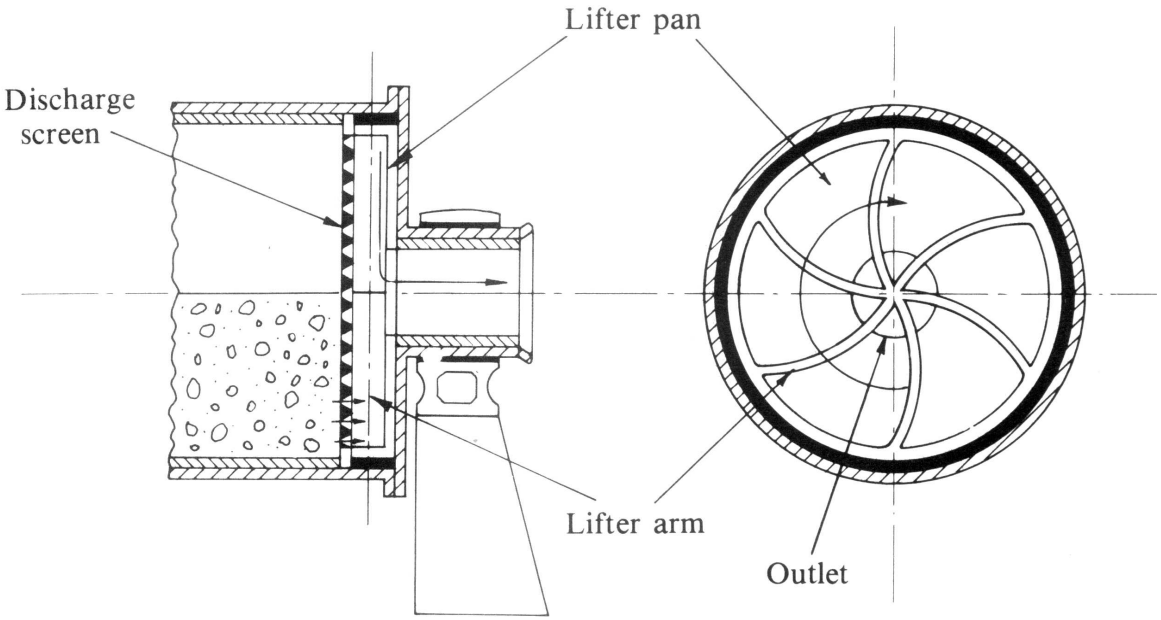


Figure 2.2: Mill Discharge - Pulp and lifter (Stanley, 1987).

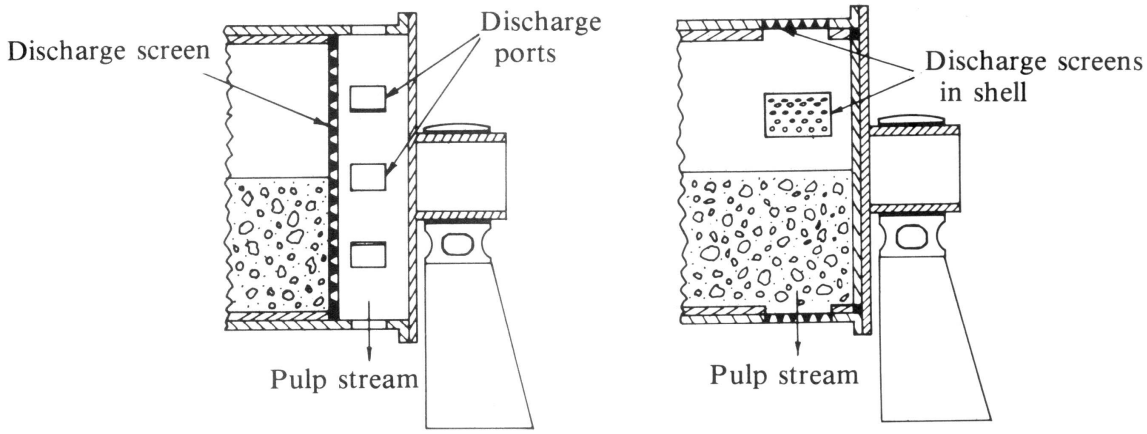


Figure 2.3: Mill Discharge - Peripheral discharge (Stanley, 1987).

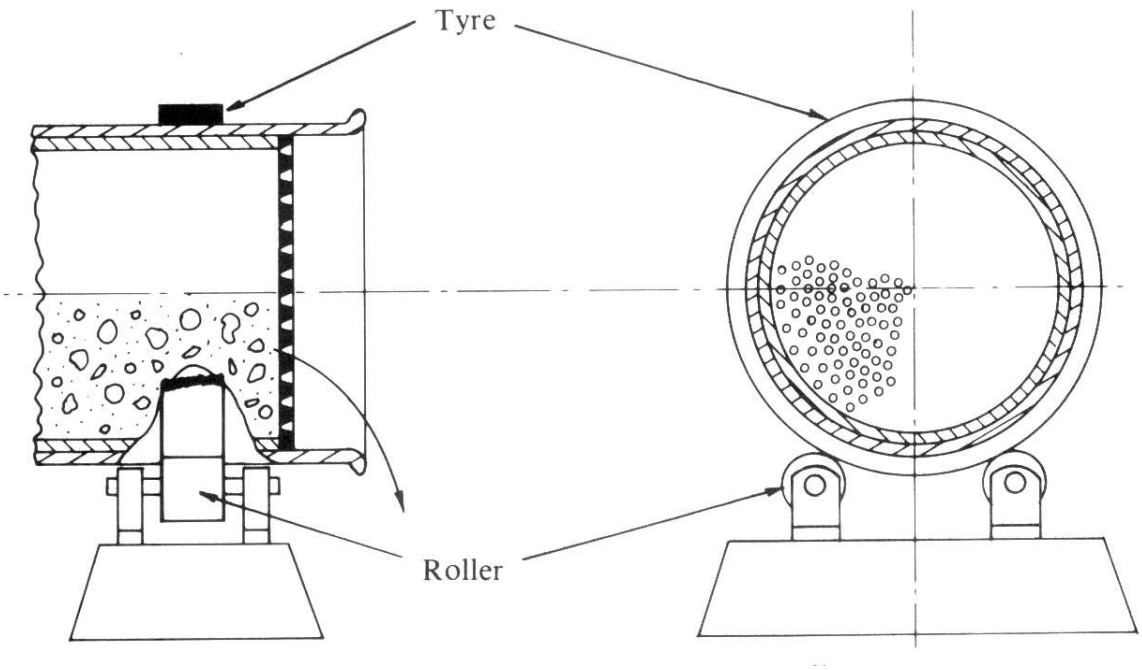


Figure 2.4: Mill Discharge - Open-ended discharge with roller and tyre (Stanley, 1987).

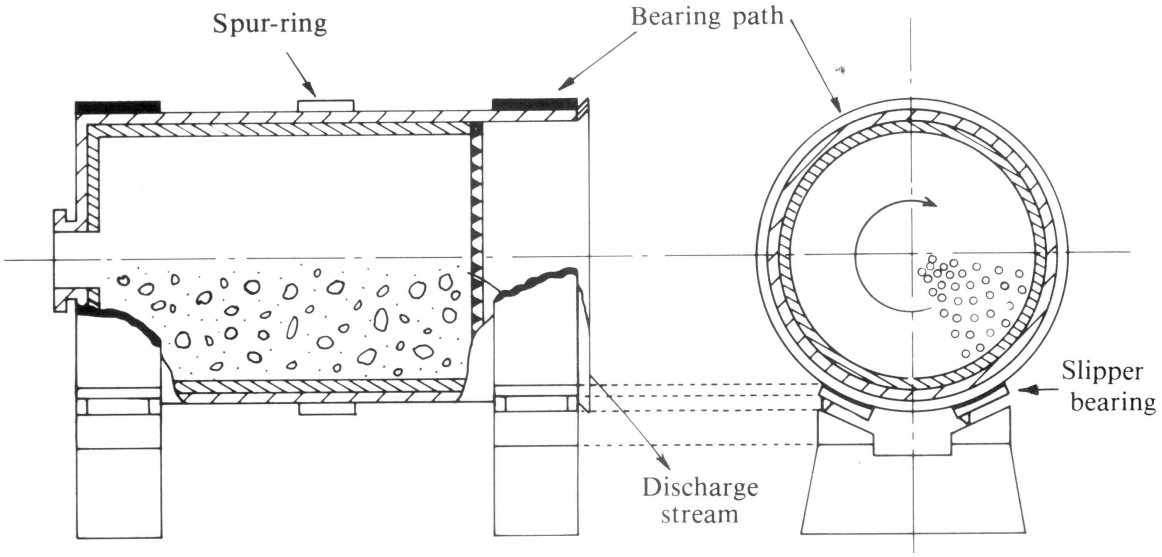


Figure 2.5: Mill Discharge - Open-ended discharge with slipper bearing (Stanley, 1987).

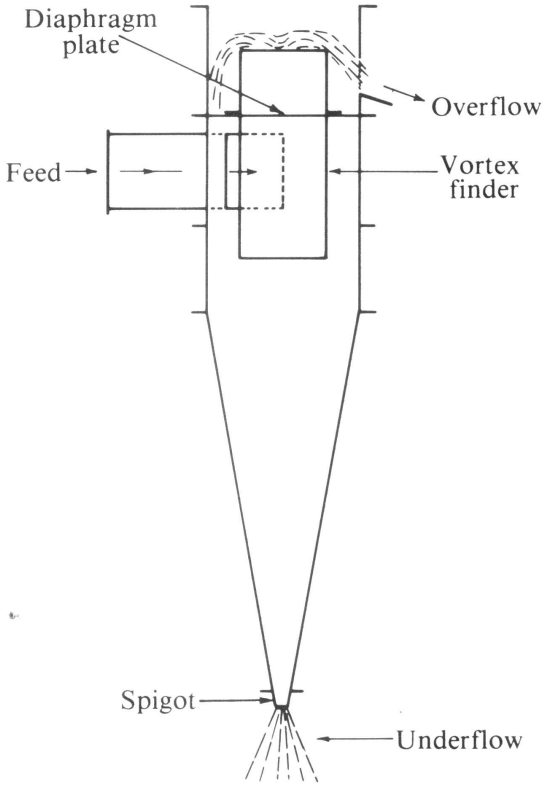


Figure 2.6: Hydrocyclone schematic (Stanley, 1987).

types of milling and produce finer particles than crushing, and are therefore ideal as intermediary between crushing and other types of milling.

Autogenous Mills use only the ore as grinding medium. There are two types of autogenous (AG) mills. The first type uses the ROM ore directly as grinding medium and is only suitable for primary milling. The advantage is that no crushing is required, except for very large pieces to facilitate handling. The second type uses pebbles rather than balls or rods as grinding medium and is therefore suitable for any stage of milling.

Semi-autogenous Mills use steel balls together with ore as griding medium. The power draw of the mill is a function of the bulk density of the load. The bulk density of AG mills is lower than that of same-sized ball and rod mills, which affects the power draw and the grinding capacity of the AG mills. The AG mills have to be larger than ball or rod mills to have the same grinding capacity. The SAG mills increase the bulk density of the mill compared to AG mills and therefore have increased grinding capacity for the same size as AG mills. SAG mills can still be used to grind run-of-mine ores.

2.2.1.2 Hydrocyclones

Hydrocyclones (Figure 2.6) are centrifugal classifiers. They work by injecting a feed slurry tangentially into the cylindrical section of the device. The centrifugal forces that result force

the solid particles through the suspended water to the cylinder wall. The particles experience drag forces as they move through the suspended water, causing heavier material to move preferentially to the cylinder walls, while lighter material does not move all the way to the cylinder wall. Most of the liquid is forced to the centre of the device owing to displacement taking place. The liquid together with the lighter material is ejected through the central vortex finder to the overflow. The heavier particles are displaced by newer particles entering the device that cause axial as well as tangential forces. The heavier particles cannot escape at the overflow, because they are stopped by the diaphragm and forced to the conical section where they are guided to the underflow opening or spigot. The cut size of the cyclone is highly dependent on the feed density and less on the cyclone dimensions. The cyclone cut can therefore be controlled by changing the feed density to the cyclone (Stanley, 1987).

2.2.1.3 Sump

The mill usually discharges into a sump from the top, where extra water is added to dilute the pulp. The sump acts as a buffer to disturbances in the feed to the cyclone. In smaller sumps, the pulp is assumed to be fully mixed, while in bigger sumps some settling may occur that could affect the discharge density at the bottom of the sump (Hulbert, 2005).

2.2.2 Process of breakage

Most comminution processes apply compression to ore particles. Elastic bodies distort when compressed by flattening in the direction of compression and bulging at right angles to the compression force. This bulging of the particle induces tensile stresses inside the particle. The tensile stresses are concentrated at the edges of flaws in the particles. The greater the flaw area is, the greater the concentration of force. If the tensile stress at the flaw edges reaches a critical value, the intermolecular bonds break. As the bonds break, the area of the flaw increases and the concentration of force at the edge of the flaw is increased, which causes more molecular bonds to break. The flaw is almost instantaneously converted to a crack. As the crack propagates, other flaws are activated and start to crack. This results in the particle being covered in a network of cracks that divide it into roughly equal fragments (Kelly and Spottiswood, 1990, Stanley, 1987).

The crack directs compressive strain energy in equal amounts to the two parts on either side of it. If the energy is sufficiently large, it can cause the pieces to break further. The two fragments are unlikely to be equally large in terms of mass, and the smaller fragment is therefore more likely to break owing to the greater energy per mass concentration of that fragment. The breakage process is concentrated on smaller and smaller particles. Every time a fragment splits, the energy is divided between the fragments, with the smaller fragment being more likely to break, until the energy levels fall below the threshold to support further breakage (Kelly and Spottiswood, 1990, Stanley, 1987).

The comminution process needs to apply enough energy to reach the critical level to cause the initial crack network to form. Once breakage starts to occur, most of the energy is then converted to heat in the form of vibrations within the particle fragments (Kelly and Spottiswood, 1990, Stanley, 1987).

2.2.2.1 Size distribution

Brittle fracture breakage produces a product with a spread of differently sized fragments. This final product consists of a particle distribution, where this distribution is an important factor in the efficiency of further processing. Determining this size distribution is done by sieving a sample of the final product. The aperture area of each successive sieve is half that of the previous sieve (Stanley, 1987). The sieve apertures are usually square and the size is expressed as the length of one of the aperture sides. The size distribution of the final product is expressed in discrete sizes by either stating

- the *differential distribution*, which is the percentage of the total sample mass in each size fraction in decreasing aperture sizes, or
- the *cumulative distribution*, which is defined as the combined mass of all the size classes starting from either the coarsest or the finest sieve to the sieve in question and expressing that mass as a percentage of the total sample mass.

The cumulative distribution is more commonly used to express the particle size distribution. The *size fraction* is defined as the amount of material between two successive sieves as a fraction of the total sample mass. For routine plant control the total size distribution is not required, but only the proportion passing or being retained by a certain sieve size, which is typically 75μm (Hulbert, 2005, Stanley, 1987).

2.2.3 Motion of the load

The contents of the mill or *load* consist of a mixture of grinding media and pulp that fills less than half the volume of the mill. The grinding media can either be balls or rods or pieces of rock called pebbles, depending on the type of mill. When the mill is at rest, the load lies at the bottom of the mill. When the mill starts to rotate, the load is lifted on the rising side of the mill to a height determined by the speed of the mill and the slip between the mill shell and the load (Stanley, 1987).

If the height is low when the materials reach the top of their travel, the load will curve over, slide and roll down the rising portion of the load until it reaches the bottom of the mill. This motion is called *cascading* (Stanley, 1987).

If the speed of the mill is sufficiently fast, the material in the load will be projected airborne after reaching the top of its travel on the mill shell. The uppermost point where the charge

leaves the mill shell is defined as the *shoulder* (Powell and McBride, 2004). After becoming airborne, the particles will rise some more before falling back, following parabolic paths. The highest vertical point that the airborne particles reach is defined as the *head* (Powell and McBride, 2004). The particles will be mostly out of contact while in the air. The area where the particles make contact with the mill after being airborne is called the the *impact toe* of the mill (Powell and McBride, 2004). This motion is called *cataracting (Powell and McBride, 2004, Stanley, 1987).*

There is high energy in the *impact toe* zone of the mill where most of the breakage occurs. There is relative motion between successive layers of the rising load. The slip occurring between layers causes additional breakage to occur through a process called attrition that is present from low rotational speeds up to high rotational speeds *(Powell and McBride, 2004, Stanley, 1987).*

If the rotational speed of the mill becomes sufficiently high, the centrifugal force acting on the outer layer of the load will overcome the centripetal component of the weight. At this point the grinding medium will stay connected to the mill shell for the whole rotation of the mill. This motion is described as *centrifuging.* In this state, very little grinding takes place in the mill. The speed at which this starts to occur is called the *critical speed* of the mill and can be calculated as

$$N_{\text{critical}} = \frac{42.23}{\sqrt{\text{Mill}_{\text{Diameter}}}} \tag{2.1}$$

where $\text{Mill}_{\text{Diameter}}$ is the inside diameter of the mill in metres, taking the mill liners into account and N_{critical} is the critical speed of the mill in revolutions per minute (RPM). (Stanley, 1987).

2.2.4 Forces causing breakage

Inside a mill, there are three types of forces that can cause breakage (Stanley, 1987):

- *Impact* forces occur when a particle is hit by another particle with high momentum. The particles can hit other particles or the mill shell. Impact energy is typically highest in the toe of the load, where most of the airborne particles make contact with the load.
- *Compression* forces are exerted where particles are trapped between other particles or against the mill shell and squeezed. This usually occurs at the bottom of the mill where the load is tightly packed.
- *Shear* forces occur where the layers of the load slide across each other with pressure. These forces occur mostly in the rising area of the load and in slowly rotating mills occur in the descending portion of the load as well.

2.2.5 Breakage mechanisms

The forces in the previous subsection do not necessarily result in breakage. If the forces are below the threshold required, no breakage will occur or only partial breakage will occur (Shi and Kojovic, 2007).

The threshold energy is also dependent on the size of the particle. The larger the particle, the more faults it contains and the weaker it becomes and therefore less energy is required to break it (Yashima *et al.*, 1987).

Partial breakage or *attrition* breakage can be subdivided into two types:

- Abrasion breakage (Loveday and Naidoo, 1997, Loveday *et al.*, 2006), which involves breaking away a part of the particle surface as a flake or even small particles.
- Chipping, which is the corner of an edge of the particle breaking away.

Both these breakage mechanisms occur when the forces are not large enough to cause the whole particle to shatter. Attrition breakage therefore causes bimodal size distribution to occur, because the main particle remains almost the same size with only some small fragments forming owing to the breakage (Stanley, 1987).

The third breakage mechanism is called true impact breakage and occurs when the particle is sufficiently stressed to the point where is shatters completely (Stanley, 1987).

2.3 MILLING MODELLING

Simulation of the mill operation is important to determine the mechanisms of breakage and the conditions affecting it. Better understanding of the charge behaviour will lead to optimising the level of grinding and maximising the capacity of the milling circuit. Measurement of the breakage mechanisms and charge behaviour inside the mill is impractical, because there are currently no online sensors that can withstand the harsh environment inside the mill (Mishra, 2003*a*).

Milling is not a very efficient process (Cleary, 2001, Kapakyulu and Moys, 2007) and better understanding of the mode and mechanisms of energy utilisation in the milling process through simulation can help increase the efficiency of milling (Mishra, 2003*a*).

Mill modelling research is divided into two main categories. Firstly the newer trend is to model the individual particles inside the mill using a discrete element method (DEM) (Mishra, 2003*a*, *b*) and secondly the mill model is based on mass balance models (Bazin *et al.*, 2005, Morrell, 2004*a*).

There are many papers in the field of mill modelling and this section gives a brief overview of the main results. This is by no means intended to be an exhaustive review of the available literature, as the focus of this thesis is the application of advanced control to a milling circuit.

2.3.1 Discrete element method

DEM modelling gives comminution researchers better insight into the breakage mechanisms taking place inside the mill. DEM allows the individual collisions between the constituents in the mill to be modelled and when this is applied to the entire mass of charge inside the mill, gives insight into the overall charge motion and behaviour (Mishra, 2003*a*). The discrete element method is making a significant contribution in analysing the elementary processes involved in grinding particles where impact geometries and other local environmental factors are very important (Mishra, 2003*b*).

The DEM could provide information on the interactions between particles, especially on how the collision energy is distributed between impact and abrasion, using energy distributions rather than average energy in the mill and using ore characteristics determined by impact and abrasion tests to predict particle breakage based on the energies applied as a result of particle interaction and size. There is still a need for research to couple the material motion and interactions to breakage tests in order to simulate breakage inside the mill reliably (Powell and McBride, 2006). The JKDrop Weight Tester characterises ores for predicting single event breakage caused by impact. Breakage in tumbling mills occurs as a result of several different modes of breakage and involves multiple events. It proves difficult to link the different modes of breakage to the motion of the particles in DEM (Morrison *et al.*, 2006).

Research in DEM focuses on understanding the contact between particles and the mechanisms of breakage (Chandramohan and Powell, 2005, Cleary, 2001, Djordjevic *et al.*, 2006, Morrison *et al.*, 2006, Morrison and Cleary, 2004, Morrison *et al.*, 2007). The development of a contact law for describing the collisions between particles and their environment is essential to properly describe the motion of particles inside the mill and any boundary objects with which they interact (Cleary, 2001).

DEM can better describe the charge motion inside the mill and the influence of different operating conditions on the charge motion and the effectiveness of grinding (Powell and McBride, 2004). By adding smoothed particle hydrodynamics (SPH), slurry effects such as lifters crashing into slurry pools, fluid draining from lifters, flow through grates and pulp lifter discharge can be modelled. The dynamics of slurry inside the mill can therefore be studied with respect to operating conditions, slurry viscosity and slurry volume (Cleary *et al.*, 2006).

DEM can be used to predict the power draw of the mill more accurately (Abd El-Rahman *et al.*, 2001, Djordjevic, 2005). Empirical models are reliable in predicting power draw, but are limited to mills and operating conditions that fall within the model database boundaries and cannot model the impact that the changing conditions inside the mill have on power draw, because of their static nature (Djordjevic, 2005).

Simulation of liner wear inside the mill can greatly assist in designing industrial mills. The effect of lifter wear on charge behaviour can aid in optimising grinding over the lifespan of

the lifters from an operational viewpoint as well as maximise lifter lifespan from a design viewpoint (McBride and Powell, 2006).

Similar to DEM for mills, hydrocyclones are modelled by capturing the fundamentals through computational fluid dynamics (Nageswararao *et al.*, 2004). In the short term this approach will result in optimisation of cyclone design and in the longer term it will provide more accurate simulation and control of hydrocyclones.

The DEM still faces a big hurdle for control purposes. The models are very computationally expensive and currently cannot be used for real-time simulation. It will take some time for the computational power to increase sufficiently to use DEM models for real-time simulation and control. For the time being, a hybrid approach might be employed where empirical models are developed with the aid of computationally intensive models such as DEM for mills and computational fluid dynamics for hydrocyclones (Djordjevic, 2005, Nageswararao *et al.*, 2004).

2.3.2 Population balance models

Modelling of mills, especially SAG/AG mills, can be described in terms of a population balance and perfectly mixed reactor (Whiten, 1974). The model a of size-by-size solids mass balance is based on the following equation (Apelt *et al.*, 2002, Morrell, 2004*a*) developed by the Julius Kruttschnitt Mineral Research Centre (JKMRC)

$$\text{Accumulation} = \text{In} - \text{Out} + \text{Generation} - \text{Consumption} \tag{2.2}$$

$$\frac{\partial s_i}{\partial t} = f_i - p_i + \sum_{j=1}^{i-1} r_j s_j a_{ij} - (1 - a_{ii}) r_i s_i \tag{2.3}$$

where

f_i	feedrate of particles of size class i [tph]
p_i	discharge rate of particles of size class i [tph]
r_i	breakage rate of particles of size i [hr^{-1}]
s_i	mass of particles in the charge of size i [tons]
a_{ij}	appearance function of particles in size i (describes the amount of material "selected" for breakage and the distribution of material after breakage occurred) [fraction]

The model describes the inflow of particles in each size class (f_i), the outflow of particles in each size class (p_i), the generation of material in the current size class i by breakage of material in the larger size classes j down to the current size class i, consumption of material in the current size class i by breakage down to smaller size classes and holdup of solids in size class i (s_i).

The model of water mass balance is based on the following equation (Apelt *et al.*, 2002)

$$\text{Accumulation} = \text{Inflow} - \text{Outflow} \tag{2.4}$$

$$\frac{\partial s_w}{\partial t} = f_w - p_w \tag{2.5}$$

where

f_w	feedrate of water [tph]
p_w	discharge rate of water [tph]
s_w	mass of water [tons]

The model describe the inflow of water (f_w), the outflow of water (p_w) and the holdup of water in the mill (s_w).

The advantage of this model is its simplicity, but this is the source of its greatest disadvantage as well. The model has no physical description of the sub-process of breakage and discharge. To make this model useful, models for the various sub-processes need to be defined.

2.3.2.1 Product discharge

The solids product p_i that is discharged from the mill is calculated as follows (Apelt *et al.*, 2002):

$$p_i = d_0 c_i s_i \tag{2.6}$$

where

p_i	discharge rate of product in size class i [tph]
d_0	maximum mill discharge rate constant [hr^{-1}]
c_i	grate classification function for size class i (probability of particle passing through discharge grate) [fraction]
s_i	mass of particles in the charge of size i [tons]

and water discharging from the mill is calculated as follows (Apelt *et al.*, 2002):

$$p_w = d_0 s_w \tag{2.7}$$

where

p_w	discharge rate of water [tph]
d_0	maximum mill discharge rate constant [hr^{-1}]
s_w	mass of water inside the mill [tons]

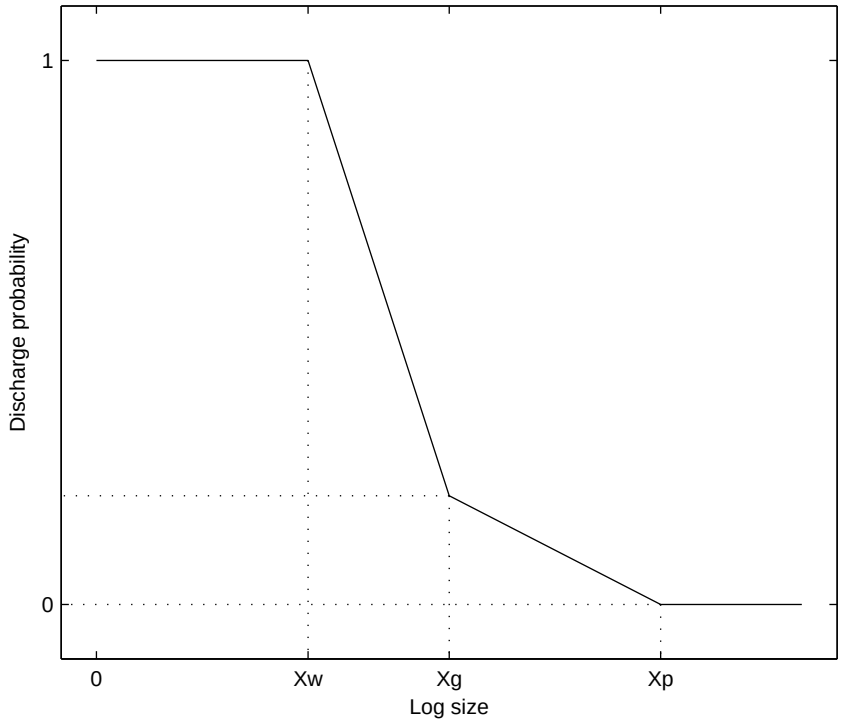


Figure 2.7: Grate classification function as in Morrell (2004*a*)

The grate classification function c_i describes the discharge behaviour of each size class through the discharge grate. It is defined as the probability of each size class passing through the grate. Particle sizes larger than the grate aperture size x_p have a discharge probability of zero. Small particle sizes that behave like water ($x < x_w$) have a discharge probability of one. The particle size classes x_i that lie between the water-like size and grate aperture size ($x_w < x_i < x_p$) show a linear decrease in discharge probability (Morrell, 2004*a*) or more complex probability curves (Amestica *et al.*, 1993, 1996).

A model is needed to describe the grate classification c_i and the maximum discharge value d_0 in equation (2.6). Development of a model to take advantage of the constant discharge seen for particles that are water-like ($x < x_w$) of the form seen in Figure 2.7 was done by Morrell and Stephenson (1996) by taking into consideration the effects of grate design, mill speed and charge volume. The work was later extended by doing an extensive laboratory study on the effects of pulp lifters and grate designs (Latchireddi and Morrell, 2003*a*, *b*).

Amestica *et al.* (1993) found that there is an increase in discharge in material just smaller than the effective grate size x_p that can be explained as the combination of two classification actions. The first classification action is larger dry material being thrown through the grate and the second action is the result of material being carried in the slurry percolating through the bed packed against the grate.

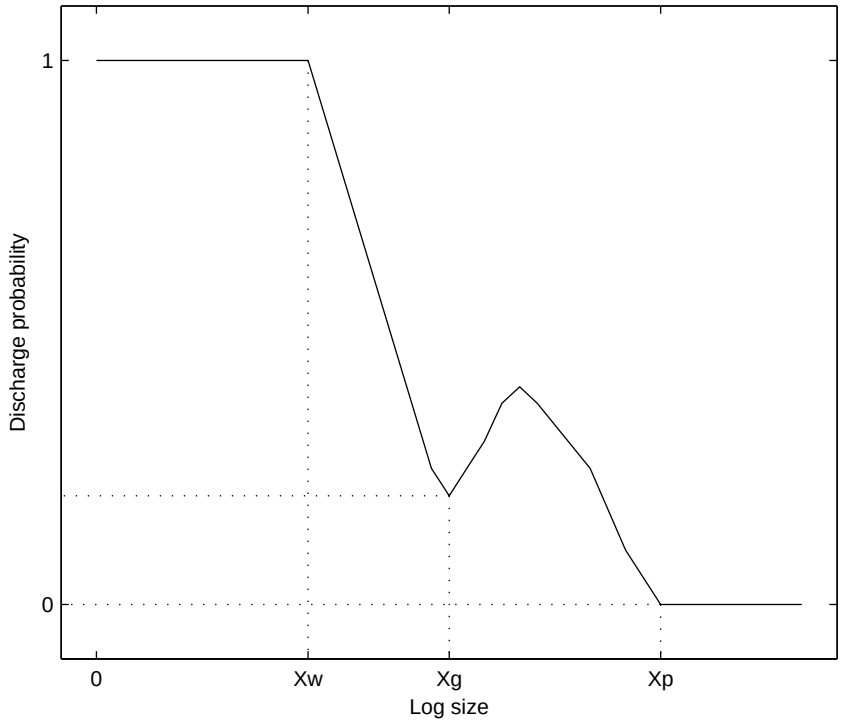


Figure 2.8: Grate classification function as in Amestica *et al.* (1996)

2.3.2.2 Breakage rate r_i

The breakage rate model or "variable rates model" (Morrell, 2004*a*, Morrell and Morrison, 1996) describes the fraction of material in each size class that is "*selected*" for breakage. The larger size classes usually have a larger percentage selected for breakage than the smaller size classes. The "variable rates model" extends the breakage rate model by varying the fraction selected for breakage for each size class according to the breakage conditions inside the mill. The breakage rate curve can be described in terms of cubic splines. The cubic splines are described by so-called "knots" with associated base breakage values (Morrell, 2004*a*, Morrell and Morrison, 1996). The values between the knots are calculated by interpolation. The equations describing the base breakage rates for the knot points are as follows:

$$\mathrm{Ln}\,(R_i) = k_{i1} + k_{i2} J_b D_b + k_{i3}\omega + k_{i4} J_t \tag{2.8}$$

where

R_i	breakage rate values for rates $i = 1-5$ [hr^{-1}].
J_b	volume of balls inside the mill [percentage].
J_t	volume of mill filled by grinding media (balls and rocks) [percentage].
D_b	make-up ball size.
ω	mill rotational rate.
$k_{i1} - k_{i4}$	constants for rates $i = 1-5$.

The effects of ball load, ball size, total load and speed on the breakage rate distribution were studied (Morrell, 2004*a*) as well as feed size distribution and recycle load (Morrell and Morrison, 1996). The difference between pilot plants and full-scale plant breakage rate distributions was also investigated (Morrell, 2004*a*) and the differences observed can be attributed to the higher rotational speed in pilot plants compared to full scale plants for the same fraction of critical speed. This is due to the differences in mill diameter (Morrell, 2004*a*) and should be included in the breakage rate models as a correction factor for scale-up from pilot plants to full-scale plants.

2.3.2.3 Breakage distribution function (appearance function a_{ij})

The appearance function a_{ij} of equation (2.3) is a matrix describing the *breakage* and the distribution that results after breakage, thus the breakage distribution (Apelt, 2002), for the material "selected" for breakage by the breakage rate function r_i (Section 2.3.2.2).

The breakage distribution function describes the distribution of smaller rock fragments that form when rock fragments in each size class breaks. This distribution is dependent on the hardness of the rock and the energy that is applied to break it. The function that describes the breakage distribution will therefore be ore-specific and related to the energy applied to break it (Morrell, 2004*a*). The breakage distribution function is the weighted average of two breakage processes; the first process is shatter at high energy intensities and the second process cleavage at low energy intensities. In mills, both these breakage processes are at work at the same time and the smaller rock fractions may break even further (Kelly and Spottiswood, 1990).

The breakage distribution function as used by the JKMRC (Morrell, 2004*a*) can be summarised as:

$$t_{10} = A\left(1 - e^{-b \cdot E_{cs}}\right) \tag{2.9}$$

where A and b are ore-specific and E_{cs} relates to the impact energy to give a size distribution index t_{10}. The characterisation of A and b is done through drop-weight tests for different ores (Napier-Munn *et al.*, 1996). The size distribution index t_{10} can then be related to a size distribution (Narayanan and Whiten, 1988).

Recent research shows that more energy is needed to break smaller particles than larger particles (Tavares and King, 1998). This was confirmed by Banini (2000) who conducted drop-weight tests on eight mineralised ore types from gold, copper and lead/zinc ore to quarry material. Fitting average characteristic A and b values for small to large particles leads to questionable performance in mills that have particles from 200 mm down to minus 1 mm (Shi and Kojovic, 2007). Shi and Kojovic (2007) fitted a modified version of a model developed by Vogel and Peukert (2003, 2005) to the drop-weight test data conducted by Banini (2000). The modified model explicitly incorporates particle size into the equation

together with material properties and cumulative impact energy as follows:

$$t_{10} = M\left\{1 - e^{-f_{mat}\cdot x\cdot k\cdot(E_{cs}-E_{\min})}\right\} \tag{2.10}$$

where

t_{10}	cumulative percentage passing 1/10 of the initial size [percentage].
M	maximum t_{10} for a material subject to breakage [percentage].
f_{mat}	material-specific breakage characteristic [dimensionless].
x	size of the particles [m].
E_{cs}	mass-specific impact energy [$\frac{\text{J}}{\text{kg}}$].
$E_{\min}$	threshold energy needed before breakage occurs or accumulates [$\frac{\text{J}}{\text{kg}}$].

The modified breakage distribution function in equation (2.10) can be related to the breakage distribution function in equation (2.9), because M is equal to A, $f_{mat}\cdot x$ to b, and $k\,(E_{cs}-E_{\min})$ to E_{cs}. The modified breakage distribution function (2.10) describes material with three property parameters (f_{mat}, x the particle size and threshold energy E_{cs}), which makes it more flexible than (2.9). Shi and Kojovic (2007) concluded that the modified breakage distribution function fitted the test data well, while requiring only one set of parameters for each ore type, and has a fundamentally better structure for describing the effect of particle size on the breakage distribution function.

Experimental procedures are necessary to obtain the breakage distribution so that it can be fitted to the models (2.9)-(2.10) given above. There are a number of methods to obtain the breakage distribution by single impact energy. There is the dual pendulum method (Narayanan, 1987) and the ultrafast load cell method (King and Bourgeois, 1993, Tavares and King, 1998). A batch grinding test can also be used to determine the breakage distribution function (Austin and Luckie, 1972, Austin and Bhatia, 1972).

2.3.2.4 Power draw model

The mill power model uses the holdup of rock and quantity of slurry in the mill to predict the volume, density and position of the charge in the mill (Morrell, 2004*a*). By conducting an energy balance around the mill and assuming that power is the tempo at which potential and kinetic energy is applied to the charge, the power draw of the cylindrical ($P_{cylinder}$) and conical (P_{cone}) sections of the mill can be expressed as (Morrell, 2004*a*)

$$P_{cylinder} = \int_{r_i}^{r_m} V_r L r g\left(\rho_c\left(\sin\theta_s - \sin\theta_t\right)\right) + \rho_p\left(\sin\theta_s - \sin\theta_{tp}\right)\mathrm{d}r \tag{2.11}$$

$$P_{cone} = \int_0^{L_i}\int_{r_i}^{r_c} V_r r g\left(\rho_c\left(\sin\theta_s - \sin\theta_t\right)\right) + \rho_p\left(\sin\theta_s - \sin\theta_{tp}\right)\mathrm{d}r\mathrm{d}L_c \tag{2.12}$$

where

L_c	The length of the cone-end, measured from the cylindrical section, at a radius of r_c [hr^{-1}].
L_i	The length of the charge surface within the cone ends [percentage].
L	The length of the cylindrical section of the mill inside the liners [percentage].
P	The power delivered to the charge (net power).
r	The radial position.
r_i	The radial position of the inner surface of the charge.
r_m	The radius of the mill inside the liners.
r_c	The radius of the cone-end of the mill at a distance of L_c from the cylindrical section of the mill [metres].
V_r	The tangential velocity of a particle at radial distance r from the centre of the mill.
θ_s	The angular displacement of the charge shoulder position at the mill shell.
θ_t	The angular displacement of the charge toe position at the mill shell.
θ_{tp}	The angular displacement of the slurry toe position at the mill shell.
ρ_c	The density of rock and ball charge (excluding the pulp).
ρ_p	The density of the pulp phase.

The power equations (2.11) and (2.12) describe the energy applied to the charge and exclude electrical and mechanical losses. A further equation is needed to describe the power draw that includes the losses to calculate the gross power draw of the mill. The no-load power equation developed by Morrell (1996) and summarised by Apelt *et al.* (2001) is given by:

$$P_{gross} = P_{No-Load} + kP_{Charge} \tag{2.13}$$

where

P_{gross}	The mill power [kW].
$P_{No-Load}$	The no-load or empty mill power draw [kW].
P_{charge}	The net power applied to the charge of the mill [kW].
k	The lumped parameter of all losses [dimensionless].

The no-load component of equation (2.13) is given by (Apelt *et al.*, 2001)

$$P_{No-Load} = 1.68\left(D_m^{2.5}\phi_{fcs}\left(0.677L_{cone}+L_m\right)\right)^{0.82} \tag{2.14}$$

where

$P_{No-Load}$	The no-load or empty mill power draw [kW].
D_m	The mill inside diameter [metres].
ϕ_{fcs}	The mill speed as a fraction of critical speed [fraction].
L_{cone}	The length of the conical section of the mill [metres].
L_m	The length of the cylindrical section of the mill [metres].

and the net power applied to the charge (P_{charge}) is given by (Apelt *et al.*, 2001)

$$P_{charge} = P_{cylinder} + P_{cone} \tag{2.15}$$

where the equation for the conical section power draw (P_{cone}) is given in equation (2.12) and the cylindrical power draw ($P_{cylinder}$) is given by equation (2.11). For more details on the JKMRC power model refer to Apelt *et al.* (2001) for a detailed summary.

The power draw of the mill can also be calculated by the torque-arm method that has the basic form (Dong and Moys, 2003)

$$P_{torque} = \frac{2\pi}{60}NT \tag{2.16}$$

$$T = F \times d = Mgx_{cog} \tag{2.17}$$

where

P_{torque}	the mill power draw [kW].
N	rotational speed of the mill [RPM].
T	torque [Newton $\times$ metres].
F	force [Newton].
d	torque-arm length [metres].
M	mass of charge in the mill [kg]
g	gravitational acceleration constant [$\frac{\text{meter}}{\text{second}^2}$].
x_{cog}	position of the centre of gravity of the charge [metres].

Dong and Moys (2003) note that the torque-arm power equations assume that the charge shape can be approximated with a cord between the shoulder and toe position, but that this assumption does not hold for a charge at high rotational speeds. Dong and Moys (2003) use position density plots of mills at steady state to obtain the parameters relating to load behaviour, such as the dynamic angle of repose, the shoulder and toe angles to determine

mill power draw from torque-arm power equations more accurately (2.16)-(2.17).

2.3.2.5 Hydrocyclone models

Nageswararao *et al.* (2004) describe two hydrocyclone models that are used in the more popular commercial simulators such as JKSimMet, *Limn* and MODSIM. These packages primarily use cyclone models developed by Nageswararao and Plitt in the 1970s. Newer developments focus on modelling cyclones from fundamental principles through computational fluid dynamics (CFD), but computer power will probably not be sufficient for process simulators for the next 25 years (Nageswararao *et al.*, 2004).

Nageswararao and Plitt both developed empirical models for the hydrocyclone, because the model structure is based solely on experimental data. Nageswararao and Plitt did follow different development methodologies to obtain their models. Nageswararao assumed a structure that explicitly decouples the machine and material characteristics, while Plitt chose an independent variable and equation structure that best fit the available database (Nageswararao *et al.*, 2004).

The models of both Nageswararao and Plitt and a comparison between the models are detailed by Nageswararao *et al.* (2004).

2.3.3 Other models

2.3.3.1 Cumulative breakage rate model

Amestica *et al.* (1993, 1996) developed a mechanistic dynamic model for semiautogenous mills called a *cumulative breakage rate* model. The advantage of this model is that there is only one function describing the breakage kinetics rather than two for selection and breakage function models (Morrell, 2004*a*). The parameters of the breakage function can be directly and uniquely derived from routine laboratory, pilot and plant data (Hinde, 2007) compared to the parameters of selection and breakage functions than need to be back-calculated (Morrell, 2004*a*). It still suffers from the same disadvantage as selection and breakage function models that need empirical relationships to relate the parameters of the cumulative breakage function to variations in the milling environment (Hinde, 2007).

2.3.3.2 Transfer function step-test model

The milling control literature favours transfer function models to describe the milling circuit (Chen *et al.*, 2007*a*, *b*, 2008, 2009, Craig and MacLeod, 1995, 1996, Pomerleau *et al.*, 2000, Radhakrishnan, 1999, Sbarbaro *et al.*, 2005) that is usually obtained by doing step-tests. Transfer function models obtained through step-tests form part of the more general approach

of linear system identification (Åström and Eykhoff, 1971). The model is obtained by applying step disturbances to the inputs of the plant and measuring the effects on the outputs. The MIMO system is described by a transfer function matrix (for example a three-input, three-output system)

$$G = \begin{bmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{bmatrix} \tag{2.18}$$

with each entry describing a specific input-output relationship. The most common form of the transfer function is first order with time delay

$$g_{ij} = \frac{k_{ij}}{\tau_{ij}s+1} e^{-\theta_{ij}s} \tag{2.19}$$

and integrator with time delay

$$g_{ij} = \frac{k_{ij}}{s} e^{-\theta_{ij}s} \tag{2.20}$$

where k_{ij} is the gain, τ_{ij} is the time constant and θ_{ij} is the time delay associated with the transfer function. The solids feed-rate to mill load and the sump feed water to sump level relationships are usually described by integrators, while the cyclone feed flow-rate to particle size relationship is described by a first order response with time delay.

2.3.3.3 Neural networks

Some authors use neural networks (Bhaumik *et al.*, 1999, Conradie and Aldrich, 2001, Duarte *et al.*, 1999*a*, 2001) to describe the input-output relationships for use in control systems, which form part of the more general approach of nonlinear system identification (Billings, 1980). Neural networks can learn the complex nonlinear behaviour of a process from its input-output data.

The advantage of using neural networks is that for model-based control, a black box linear or nonlinear model can be constructed very easily from only input-output data.

The disadvantage of using neural networks is that a large amount of data is required to train the neural network properly, but could possibly be offset by allowing online learning at a slow rate (Duarte *et al.*, 1999*a*). Neural networks that are structured to describe the current output $y(k)$ from the previous plant inputs $(u(k-1), u(k-2), \ldots, u(k-N))$ and previous plant outputs $(y(k-1), y(k-2), \ldots, y(k-N))$ can be used to study the dynamic of the plant, but it does not extend to neural networks in general, such as neural networks with hidden layers.

Table 2.11: Model internal flows and constants.

Symbol	Description
Volumetric flow-rates used for internal flows: in (i), out (o), underflow (u)	
V_{wi}, V_{wo}, V_{wu}	Water [$m^3/hour$]
V_{si},V_{so}, V_{su}	Solids [$m^3/hour$]
V_{ci},V_{co},V_{cu}	Coarse ($V_s - V_f$) [$m^3/hour$]
V_{fi},V_{fo}, V_{fu}	Fines [$m^3/hour$]
V_{ri},V_{ro}	Rocks [$m^3/hour$]
V_{bi},V_{bo}	Balls [$m^3/hour$]
D_s,D_b	Density of feed ore (D_s) and steel balls (D_b) [kg/m^3]

2.3.4 Mintek mill circuit model modules

Simulation models generate large amounts of data and models usually consist of many states and parameters. This is cumbersome for control and estimation purposes. Mintek developed a research mechanistic model (Coetzee *et al.*, 2009) based on the simulation models (Apelt *et al.*, 2001, Austin *et al.*, 1988, 1983, Morrell, 2004*b*, Whiten, 1974), but reduced the number of states and parameters to the bare minimum needed for control and estimation purposes.

Mintek gave permission that the following information concerning the model may be published (Coetzee *et al.*, 2009). The model consists of separate modules for the feeder, mill, sump and hydrocyclone. These modules can be connected in various configurations depending on the plant set-up. The model uses five states, namely water, rock, solids, fines and steel balls. Rocks are defined as the ore that is too big to be discharged from the mill. Solids are defined as the coarse ore as well as the fines. Coarse ore is defined as the ore that is discharged from the mill, but is out-of-specification, thus larger than 75 μm. Fines are defined as the ore that is in-specification and thus smaller than 75 μm.

The nomenclature for the model is shown in Table 2.11, Table 2.12 and Table 2.13.

2.3.4.1 Feeder module

The feeder module is a very simple one that takes the feed-rate of ore and divides it into the ore streams (fines, coarse and rock) that will be used throughout the model. This module can be replaced by models of real feeders, for example vibratory feeders.

Table 2.12: Model states, inputs and outputs.

Variable	Description
X_{mw}	The holdup of water in the mill. [m^3]
X_{ms}	The holdup of ore in the mill. [m^3]
X_{mf}	The holdup of fine ore in the mill. [m^3]
X_{mr}	The holdup of rock in the mill. [m^3]
X_{mb}	The holdup of balls in the mill. [m^3]
X_{sw}	The holdup of water in the sump. [m^3]
X_{ss}	The holdup of ore in the sump. [m^3]
X_{sf}	The holdup of fine ore in the sump. [m^3]
MIW	The flow-rate of water to the circuit. [m^3/hour]
MFS	The feed-rate of ore to the circuit (consists of rocks, coarse and fine ore). [tons/hour]
MFB	The feed-rate of steel balls to the circuit. [tons/hour]
α_{speed}	The fraction of critical mill speed.
CFF	The flow-rate of slurry from the sump to the cyclone. [m^3/hour]
SFW	The flow-rate of extra water to the sump. [m^3/hour]
PSE	Product particle-size. [$\% < 75\mu\text{m}$]
LOAD	The total charge of the mill. [%]
SLEV	The level of the sump. [m^3]
Γ	Rheology factor. [dimensionless]
THROUGHPUT	Product throughput consisting of coarse and fine solids. [tons/hour]
P_{mill}	Power draw of the mill motor. [kW]

The output functions for the feeder unit are given by

$$V_{\text{wo}} \triangleq \text{MIW} \tag{2.21}$$

$$V_{\text{so}} \triangleq \frac{\text{MFS}}{D_s}(1-\alpha_r) \tag{2.22}$$

$$V_{\text{fo}} \triangleq \alpha_f \cdot \frac{\text{MFS}}{D_s} \tag{2.23}$$

$$V_{\text{ro}} \triangleq \alpha_r \cdot \frac{\text{MFS}}{D_s} \tag{2.24}$$

$$V_{\text{bo}} \triangleq \frac{\text{MFB}}{D_b} \tag{2.25}$$

2.3.4.2 Mill module

The mill module is capable of modelling various mill types such as rod, ball, SAG and AG mills. It is similar to the models found in literature, but adds the effect of mill power and slurry rheology (Shi and Napier-Munn, 2002) to the breakage and power functions.

It receives the ore, balls and water and incorporates models that describe the production of fines, rock consumption, ball wear, outflow for various discharge mechanisms e.g. grate discharge, power consumption and rheology effects.

The rheology factor which relates to the fluidity of the slurry in the mill (Bazin and B-

Table 2.13: Model Parameters.

Parm	Description
α_f	Fraction of fines in the ore. [dimensionless]
α_r	Fraction of rock in the ore. [dimensionless]
ϕ_f	Power per fines produced. [$^{\text{kW·hr}}/_{\text{ton}}$]
ϕ_r	Rock abrasion factor. [$^{\text{kW·hr}}/_{\text{ton}}$]
ϕ_b	Steel abrasion factor. [$^{\text{kW·hr}}/_{\text{ton}}$]
P_{max}	Maximum mill motor power. [kW]
υ_{mill}	Mill volume. [m^3]
$\upsilon_{P_{\text{max}}}$	Mill filling volume for maximum power. [m^3]
$\Gamma_{P_{\text{max}}}$	Rheology factor for maximum mill power. [dimensionless]
ε_{ws}	Maximum water-to-solids volumetric ratio at zero pulp flow. [dimensionless]
V_V	Volumetric flow per "flowing volume" driving force. [hr^{-1}]
δ_{Pv}	Power-change parameter for volume. [dimensionless]
δ_{Ps}	Power-change parameter for fraction solids. [dimensionless]
α_P	Fractional power reduction per fractional reduction from maximum mill speed. [dimensionless]
α_{ϕ_f}	Fractional change in $^{\text{kW}}/_{\text{fines}}$ produced per change in fractional filling of mill. [dimensionless]
χ_P	Cross-term for maximum power. [dimensionless]
ε_c	Cyclone coarse split fraction. [dimensionless]
α_{su}	Fraction of solids in the underflow of the cyclone. [dimensionless]
C_1	Constant. [dimensionless]
C_2	Constant. [dimensionless]
C_3	Constant. [dimensionless]
C_4	Constant [dimensionless]

Chapleau, 2005, Shi and Napier-Munn, 1996, 2002) is calculated as

$$\Gamma \triangleq \left\{ \frac{\max\left[0, \left(X_{\text{mw}} - \left(\left(\frac{1}{\varepsilon_{ws}}\right) - 1\right) X_{\text{ms}}\right)\right]}{X_{\text{mw}}} \right\}^{0.5} \tag{2.26}$$

and the mill power as

$$P_{\text{mill}} \triangleq P_{\text{max}} \cdot \left\{1 - \delta_{Pv} Z_{\upsilon}^2 - 2 \cdot \chi_P \cdot \delta_{Pv} \cdot \delta_{Ps} \cdot Z_{\upsilon} \cdot Z_{\Gamma} - \delta_{Ps} \cdot Z_{\Gamma}^2\right\} \cdot \left(\alpha_{\text{speed}}\right)^{\alpha_P} \tag{2.27}$$

where load is represented by

$$Z_{\upsilon} \triangleq \frac{X_{\text{mb}} + X_{\text{mr}} + X_{\text{ms}} + X_{\text{mw}}}{\upsilon_{P_{\text{max}}} \cdot \upsilon_{\text{mill}} - 1} \tag{2.28}$$

and rheology is represented by

$$Z_{\Gamma} \triangleq \frac{\Gamma}{\Gamma_{P_{\text{max}}}} - 1. \tag{2.29}$$

The production of fines is a linear function of the total mill volume filling (LOAD $\triangleq (X_{\text{mw}} + X_{\text{ms}} + X_{\text{mr}} + X_{\text{mb}})$).

$$\text{FP} \triangleq \frac{P_{\text{mill}}}{D_s \cdot \left\{\phi_f \cdot \left[1 + \alpha_{\phi_f} \cdot \left(\frac{\text{LOAD}}{\upsilon_{\text{mill}}} - \upsilon_{P_{\text{max}}}\right)\right]\right\}} \tag{2.30}$$

The fines produced from rocks are not distinguished from fines produced from the coarse ore.

The rocks in the mill grind down to coarse and/or fines in indeterminable ratios. The total fines produced are modelled to come from both rocks and coarse material. The yield of coarse and fines from rocks will therefore not be modelled, only the consumption of rocks. The consumption is at it highest with high power, fluid slurry and relatively low amounts of fine and coarse ore. The model for rock consumption is given by

$$\text{RC} \triangleq \left(\frac{1}{D_s \cdot \phi_r}\right) \cdot P_{\text{mill}} \cdot \Gamma \left(\frac{X_{\text{mr}}}{X_{\text{mr}} + X_{\text{ms}}}\right) \tag{2.31}$$

The steel balls in the mill grind away over time. The consumption of the steel balls is at its highest with high power, fluid slurry and high ball loadings and the consumption can be described as

$$\text{BC} \triangleq \left(\frac{1}{D_b \cdot \phi_b}\right) \cdot P_{\text{mill}} \cdot \Gamma \cdot \left(\frac{X_{\text{mb}}}{X_{\text{mb}} + X_{\text{mr}} + X_{\text{ms}}}\right) \tag{2.32}$$

The volumetric flows from the mill are described by

$$V_{\text{to}} \triangleq V_V \cdot \Gamma \cdot X_{\text{mw}} \tag{2.33}$$

$$V_{\text{wo}} \triangleq V_{\text{to}} \cdot \left(\frac{X_{\text{mw}}}{X_{\text{ms}} + X_{\text{mw}}} \right) \tag{2.34}$$

$$V_{\text{so}} \triangleq V_{\text{to}} \cdot \left(\frac{X_{\text{ms}}}{X_{\text{ms}} + X_{\text{mw}}} \right) \tag{2.35}$$

$$V_{\text{fo}} \triangleq V_{\text{to}} \cdot \left(\frac{X_{\text{mf}}}{X_{\text{mr}} + X_{\text{mw}}} \right) \tag{2.36}$$

where $V_{\text{bo}} \triangleq 0$ and $V_{\text{ro}} \triangleq 0$ because it is assumed that the rocks and balls cannot escape through the discharge grate.

The changes in volumetric holdups in the mill are described as

$$\frac{dX_{\text{mw}}}{dt} \triangleq V_{\text{wi}} - V_{\text{wo}} \tag{2.37}$$

$$\frac{\partial X_{\text{ms}}}{\partial t} \triangleq V_{\text{si}} - V_{\text{so}} + \text{RC} \tag{2.38}$$

$$\frac{\partial X_{\text{mf}}}{\partial t} \triangleq V_{\text{fi}} - V_{\text{fo}} + \text{FP} \tag{2.39}$$

$$\frac{\partial X_{\text{mr}}}{\partial t} \triangleq V_{\text{ri}} - \text{RC} \tag{2.40}$$

$$\frac{\partial X_{\text{mb}}}{\partial t} \triangleq V_{\text{bi}} - \text{BC} \tag{2.41}$$

2.3.4.3 Mixed-sump module

The mixed-sump module assumes that the water, fines and coarse material are fully mixed. The sump only handles water, fine and coarse ore because it is assumed that the rock and the balls remain in the mill.

The volumetric flow-rates from the sump are described as

$$V_{\text{wo}} \triangleq \text{CFF} \cdot \left(\frac{X_{\text{sw}}}{X_{\text{ss}} + X_{\text{sw}}} \right) \tag{2.42}$$

$$V_{\text{so}} \triangleq \text{CFF} \cdot \left(\frac{X_{\text{ss}}}{X_{\text{ss}} + X_{\text{sw}}} \right) \tag{2.43}$$

$$V_{\text{fo}} \triangleq \text{CFF} \cdot \left(\frac{X_{\text{sf}}}{X_{\text{ss}} + X_{\text{sw}}} \right) \tag{2.44}$$

The changes in hold-ups in the sump are described by

$$\frac{\partial X_{\text{sw}}}{\partial t} \triangleq V_{\text{wi}} + \text{SFW} - V_{\text{wo}} \tag{2.45}$$

$$\frac{\partial X_{\text{ss}}}{\partial t} \triangleq V_{\text{si}} - V_{\text{so}} \tag{2.46}$$

$$\frac{\partial X_{\text{sf}}}{\partial t} \triangleq V_{\text{fi}} - V_{\text{fo}} \tag{2.47}$$

The sump level is defined as

$$\text{SLEV} \triangleq X_{\text{sw}} + X_{\text{ss}}. \tag{2.48}$$

The cyclone feed density is defined as

$$\text{CFD} \triangleq {X_{\text{sw}} + D_s \cdot X_{\text{ss}}}/{X_{\text{sw}} + X_{\text{ss}}} \tag{2.49}$$

.

2.3.4.4 Hydrocyclone module

The hydrocyclone is a classification device that splits a slurry feed with regard to weight, which usually relates to the size of the particles. Lighter particles are forced out at the overflow of the cyclone, while the heavier particles are forced out at the underflow. The cyclone model is designed to model the product size and density accurately by taking the effects of angular velocity of the particle inside the cyclone, slurry density and viscosity into account. The model is based on the empirical hydrocyclone models of Plitt and Nageswararao (Nageswararao *et al.*, 2004). The flow-rates of the various constituents of the slurry at the underflow of the hydrocyclone are defined as

$$V_{\text{cu}} \triangleq V_{\text{ci}} \cdot \left(1 - C_1 \cdot e^{\left(-\frac{V_{ti}}{\varepsilon_c}\right)}\right) \times \left(1 - \left(\frac{F_{\text{i}}}{C_2}\right)^{C_3}\right) \cdot \left(1 - P_i^{C_4}\right) \tag{2.50}$$

$$F_{\text{u}} \triangleq 0.6 - (0.6 - F_{\text{i}}) \cdot e^{\left(-\frac{V_{\text{cu}}}{\alpha_{su}\varepsilon_c}\right)} \tag{2.51}$$

$$V_{\text{wu}} \triangleq V_{\text{wi}} \cdot \frac{(V_{\text{cu}} - F_{\text{u}} \cdot V_{\text{cu}})}{(F_{\text{u}} \cdot V_{\text{wi}} + F_{\text{u}} \cdot V_{\text{fi}} - V_{\text{fi}})} \tag{2.52}$$

$$V_{\text{fu}} \triangleq V_{\text{fi}} \cdot \frac{(V_{\text{cu}} - F_{\text{u}} \cdot V_{\text{cu}})}{(F_{\text{u}} \cdot V_{\text{wi}} + F_{\text{u}} \cdot V_{\text{fi}} - V_{\text{fi}})} \tag{2.53}$$

$$F_i \triangleq \frac{V_{\text{si}}}{V_{\text{wi}} + V_{\text{si}}} \tag{2.54}$$

$$P_i \triangleq \frac{V_{\text{fi}}}{V_{\text{si}}} \tag{2.55}$$

The product particle size is defined as

$$\text{PSE} \triangleq \frac{V_{\text{fo}}}{(V_{\text{co}} + V_{\text{fo}})} \tag{2.56}$$

where $V_{fo} \triangleq V_{fi} - V_{fu}$ and $V_{co} \triangleq V_{ci} - V_{cu}$. The product throughput is defined as

$$\text{THROUGHPUT} \triangleq V_{co} + V_{fo}. \tag{2.57}$$

2.4 CONCLUSION

This chapter gives an overview of the milling process and the different simulation models available for modelling milling circuits. The more complex models are usually employed to simulate the milling circuit and generate vast amounts of data for circuit design and analysis. The simpler models are usually employed in controllers and controller design.

This chapter outlines the nonlinear model developed by Mintek (Section 2.3.4) that consists of models for the mill, cyclone and sump and will be used in later chapters as the prediction model for the milling circuit as well as the simulation model. The Mintek model is modular to simulate different milling circuit configurations easily.

CHAPTER 3

MODEL PREDICTIVE CONTROL

This chapter describes MPC and especially RNMPC that is applied to the plant outlined in Chapter 2. The chapter starts by explaining MPC and its history, followed by a description of robust MPC and the reason for its development, and finally focuses on the controller theory used for the simulation study in Chapter 5. The description of MPC and the development of stability theory, including robust stability, in Sections (3.1)-(3.4) are summaries from the survey done by Mayne *et al.* (2000) and provided here for background.

3.1 INTRODUCTION

MPC, also known as receding horizon control (RHC), takes a measurement of the plant, uses a mathematical model of the system to predict its future behaviour in order to calculate a sequence of control moves (N steps) into the future that will optimise (usually minimise) an objective or penalty function, which describes a measure of performance of the system. The first control move of the calculated sequence is applied to the system and a new measurement is taken. The process is then repeated for the next time step. MPC calculates the control sequence on-line at each time step, compared to conventional control theory where the control law is pre-calculated and valid for all possible states of the system. MPC has the distinct advantage of controlling multi-variable systems well and can explicitly take into consideration constraints on the inputs (such as actuators, valves, etc.) as well as states or outputs (Camacho and Bordons, 2003). MPC is especially useful in situations where an explicit controller cannot be calculated offline.

The basic ideas present in the MPC family, according to Camacho and Bordons (2003), are that

- outputs at future time instances are predicted by the explicit use of a *mathematical model,*

- an *objective function* is minimised by calculating the appropriate control sequence, and
- at each time instant, the horizon is displaced towards the future, which involves applying the first control signal calculated at each time instance to the system; called the *receding horizon* strategy.

The MPC theory described in this chapter is in discrete time and the system takes the following form (Mayne *et al.*, 2000):

$$x(k+1) = f(x(k),u(k)), \tag{3.1}$$
$$y(k) = h(x(k)). \tag{3.2}$$

To simplify the notation, the value of x at time k is given by $x_k = x(k)$. The value of x at time $k+1$, therefore, becomes $x_{k+1} = x(k+1)$, the value of y at time k becomes $y(k) = y_k$ and the value of u at time k becomes u_k. The equations (3.1)-(3.2), therefore, becomes

$$x_{k+1} = f(x_k,u_k), \tag{3.3}$$
$$y_k = h(x_k). \tag{3.4}$$

The control and state sequences must satisfy

$$x_k \in \mathbb{X}, \tag{3.5}$$
$$u_k \in \mathbb{U}, \tag{3.6}$$

where $\mathbb{X} \subset \mathbb{R}^{n_x}$ and $\mathbb{U} \subset \mathbb{R}^{n_u}$.

Given a control vector sequence of length N, $\mathbf{u}^N = \{u^N(k), u^N(k+1), \ldots, u^N(k+N-1)\}$, where $u^N(k+1)$ specifies the control vector at time $k+1$, the predicted state vector sequence based on the control sequence $\mathbf{u}^N$ is given by

$$\mathbf{x}^{\mathbf{u}}(x_k,k) = \{x^{\mathbf{u}}(k,x_k,k), x^{\mathbf{u}}(k+1,x_k,k), \ldots, x^{\mathbf{u}}(k+N-1,x_k,k), x^{\mathbf{u}}(k+N,x_k,k)\}, \tag{3.7}$$

where $x^{\mathbf{u}}(k+1,x_k,k)$ is the calculated state vector at time $k+1$ from the initial state x_k, initial time k and control vector $u^N(k)$ using $f(\cdot,\cdot)$. The notation for the state sequence is simplified by stating the current state x_k and time k as subscripts to give $\mathbf{x}^N_{x_k,k} = \mathbf{x}^N(x_k,k) = \left\{x^{\mathbf{u}}_{x_k,k}(k), x^{\mathbf{u}}_{x_k,k}(k+1), \ldots, x^{\mathbf{u}}_{x_k,k}(k+N-1), x^{\mathbf{u}}_{x_k,k}(k+N)\right\}$.

The objective function that is used in the optimisation process has the following form:

$$\phi(x_k,k,\mathbf{u}^N) = \sum_{i=k}^{k+N-1} L(x^{\mathbf{u}}_{x_k,k}(i), u^N(i)) + E(x^{\mathbf{u}}_{x_k,k}(k+N)), \tag{3.8}$$

where $L(x^{\mathbf{u}}_{x_k,k}(i), u^N(i))$ is the cost at each time step into the future with regard to the states and inputs, while $E(x^{\mathbf{u}}_{x_k,k}(k+N))$ is the cost at the final state, reached after the whole control sequence has been applied. At each time k, the final time is $k+N$, which increases as k increases and is called a *receding horizon*. In certain MPC formulations, a terminal constraint set is defined

$$x^{\mathbf{u}}_{x_k,k}(k+N) \quad \in \quad X_f \subset \mathbb{X}. \tag{3.9}$$

The optimal control problem $P(x_k,k)$ of minimising the objective function is performed subject to the constraints on the control and state sequences, and in certain cases the terminal constraint to yield the optimised control sequence

$$\mathbf{u}^{\text{opt}}(x_k,k) \quad = \quad \left\{u^{\text{opt}}_{x_k,k}(k), u^{\text{opt}}_{x_k,k}(k+1), ..., u^{\text{opt}}_{x_k,k}(k+N-1)\right\}, \tag{3.10}$$

and optimised value for the objective function

$$\phi^{\text{opt}}(x_k,k) \quad = \quad \phi(x_k,k,\mathbf{u}^{\text{opt}}_{x_k,k}). \tag{3.11}$$

The first control move at time k of the sequence $\mathbf{u}^{\text{opt}}(x_k,k)$ is implemented to form an implicit control law for time k

$$\kappa(x_k,k) \quad = \quad u^{\text{opt}}_{x_k,k}(k). \tag{3.12}$$

The objective function is time invariant, because neither $L(x^{\mathbf{u}}_{x_k,k}(i), u^N(i))$ nor $E(x^{\mathbf{u}}_{x_k,k}(k+N))$ has terms that depend on time. The optimal control problem $P(x_k,k)$ can be defined as starting at time 0 to give $P_N(x_k) = P(x_k,0)$. N represents the finite prediction horizon over which the optimisation takes place, and the optimisation problem can be redefined as

$$P_N(x_k) : \phi^{\text{opt}}_N(x_k) \quad = \quad \min_{\mathbf{u}^N} \left\{\phi_N(x_k,\mathbf{u}^N) | \mathbf{u}^N \in \mathbb{U}_N\right\}, \tag{3.13}$$

where the objective function is now

$$\phi_N(x_k,\mathbf{u}^N) \quad = \quad \sum_{i=0}^{N-1} L(x^{\mathbf{u}}_{x_k}(i), u^N(i)) + E(x^{\mathbf{u}}_{x_k}(N)), \tag{3.14}$$

with $\mathbb{U}_N$ the set of feasible control sequences that satisfy the control, state and terminal constraints. If problem $P_N(x_k)$ is solved, the optimal control sequence is obtained

$$\mathbf{u}^{\text{opt}}(x_k) \quad = \quad \left\{u^{\text{opt}}_{x_k}(0), u^{\text{opt}}_{x_k}(1), ..., u^{\text{opt}}_{x_k}(N-1)\right\}, \tag{3.15}$$

and the optimal state trajectory, if the control actions are implemented, is given by

$$\mathbf{x}^{\text{opt}}(x_k) \quad = \quad \left\{x^{\text{opt}}_{x_k}(0), x^{\text{opt}}_{x_k}(1), ..., x^{\text{opt}}_{x_k}(N-1), x^{\text{opt}}_{x_k}(N)\right\}. \tag{3.16}$$

The optimal objective value is

$$\phi_N^{\text{opt}}(x_k) \quad = \quad \phi_N(x_k, \mathbf{u}_{x_k}^{\text{opt}}). \tag{3.17}$$

The first control action is implemented, leading to the implicit time invariant control law

$$\kappa_N(x_k) \quad = \quad u_{x_k}^{\text{opt}}(0). \tag{3.18}$$

Dynamic programming can be used to determine a sequence of objective functions $\phi_j(\cdot)$ deterministically in order to calculate the sequence of control laws $\kappa_j(\cdot)$ offline, where j is the time-to-go until the prediction horizon. This is possible because of the deterministic nature of the open-loop optimisation. This would be preferable, but is usually not possible. The difference between MPC and dynamic programming is purely a matter of implementation. MPC differs from conventional optimal control theory in that MPC uses a receding horizon control law $\kappa_N(\cdot)$ rather than an infinite horizon control law.

3.2 HISTORICAL BACKGROUND

MPC builds on optimal control theory, the theory (necessary and sufficient conditions) of optimality, Lyapunov stability of the optimal controlled system, and algorithms for calculating the optimal feedback controller (if possible) (Mayne *et al.*, 2000). There are a few important ideas in optimal control that underlie MPC. The first links together two principles of the control theory developed in the 1960s: the Hamilton-Jacobi-Bellman theory (Dynamic Programming) and the maximum principle, which provides necessary conditions for optimality. Dynamic programming provides sufficient conditions for optimality, as well as a procedure to synthesise an optimal feedback controller $u = \kappa(x_k)$. The maximum principle provides necessary conditions of optimality as well as computational algorithms for determining the optimal open-loop control $u_{x_k}^{\text{opt}}(\cdot)$ for a given initial state x_k. These two principles are linked together as

$$\kappa(x_k) \quad = \quad u_{x_k}^{\text{opt}}(0), \tag{3.19}$$

in order for the optimal feedback controller to be obtained by calculating the open-loop control problem for each x (Mayne *et al.*, 2000). From the commencement of optimal control theory it is stated by Lee and Markus (1967, p. 423): "*One technique for obtaining a feedback controller synthesis from knowledge of open-loop controllers is to measure the current control process state and then compute very rapidly for the open-loop control function. The first portion of this function is then used during a short time interval, after which a new measurement of the process state is made and a new open-loop control function is computed for this new measurement. The procedure is then repeated.*"

Kalman, as discussed in Mayne *et al.* (2000), observed that *optimality* does not guarantee *stability*. There are conditions under which optimality results in stability: *infinite horizon* controllers are stabilising, if the system is stabilisable and detectable. Calculating infinite horizon optimal solutions is not always practical on-line and an alternate solution was needed to stabilise the receding horizon controller. The first results for stabilising receding horizon controllers were given by Kleinman (1970), who developed a minimum energy controller for linear systems. He showed that the feedback controller is linear, time invariant and stable if a Lyapunov function $\phi(x) = x^T Px$ is used as the objective function. Another approach is to define a *stability constraint* as part of the optimal control problem. The stability constraint is defined as an equality constraint $x(T) = 0$ that forces the solution to converge to the origin. Thomas, as discussed in Mayne *et al.* (2000), suggested this technique as part of a linear quadratic control problem and implemented it by using $M \triangleq P^{-1}$ in place of P as the Riccati variable and solving the Riccati-like differential equation with terminal condition $M(T) = 0$.

MPC was really driven by industry as part of process control theory. Richalet *et al.* (1978) was the first to propose MPC for process control applications, but MPC was proposed earlier by Propoi and Lee and Markus (as discussed in Mayne *et al.* (2000)). The MPC method, called identification and command (IDCOM), was proposed by Richalet *et al.* (1978). It uses a linear model in the form of a finite horizon impulse response, quadratic cost and constraints on the inputs and outputs. The method makes provision for linear estimation using least squares, and the algorithm for solving the open-loop optimal control problem is the "dual" of the identification algorithm.

DMC is a later method proposed by Cutler and Ramaker (1980) and Prett and Gillette (as discussed in Mayne *et al.* (2000)). DMC uses a step response model, but as in IDCOM, handles constraints in an ad hoc fashion. This limitation was addressed by García and Morshedi (as discussed in Mayne *et al.* (2000)) by using quadratic programming to solve the constrained open-loop optimisation problem. This method also allows certain violations of the constraints in order to enlarge the set of feasible states. This method is called Quadratic Dynamic Matrix Control (QDMC).

The third generation of MPC technology, introduced about a decade ago, *"distinguishes between several levels of constraints (hard, soft and ranked). This technology provides some mechanism to recover from an infeasible solution, and addresses the issues resulting from a control structure that changes in real time, and allows for a wider range of process dynamics and controller specifications*" (Qin and Badgwell, 2003). The Shell multi-variable optimising control (SMOC) uses state-space models, incorporates general disturbance models and allows for state estimation using Kalman filters (as discussed in Mayne *et al.* (2000)).

An independent but similar approach was developed from the adaptive control theory and is called generalised predictive control (GPC). The method uses models in the backward shift operator q^{-1} which is more general than the impulse and step response models of DMC. GPC started as minimum variance control (Mayne *et al.*, 2000) that only allowed for a horizon of

length 1. Minimum variance control was extended to allow for longer prediction horizons by Peterka (1984) as well as Clarke *et al.* (1987*a*, *b*). GPC, and early versions of DMC, did not explicitly incorporate stability in the method and had to rely on the tuning of the prediction horizon as well as the weights of the states and inputs to achieve stability.

3.3 STABILITY OF MPC

The inability of both GPC and DMC to guarantee stability caused researchers to focus more on modifying $P_N(x)$ to ensure stability, owing to increased criticism (Bitmead *et al.*, 1990) of the makeshift approach of using tuning to attain stability.

With *terminal equality constraints*, the system is forced to the origin by the controller that takes the form $E(x) = 0$, as there is no terminal cost and the terminal set is $X_f = \{0\}$. Keerthi and Gilbert, as discussed in Mayne *et al.* (2000), proposed this stabilising strategy for constrained, nonlinear, discrete time systems and showed a stability analysis of this version (terminal equality constraints) of discrete-time receding horizon control. MPC, with a terminal equality constraint, can be used to stabilise a system that cannot be stabilised by continuous feedback controllers, according to Meadows *et al.* (as discussed in Mayne *et al.* (2000)).

Using a *terminal cost function* is an alternative approach to ensure stability. Here the terminal cost is $E(\cdot)$, but there is no terminal constraint and the terminal set is thus $X_f = \mathbb{R}^{n_x}$. For unconstrained linear systems the terminal cost of $E(x) = \frac{1}{2}x^T P_f x$ is proposed by Bitmead *et al.* (1990).

Terminal constraint sets differ from terminal equality constraints, in that subsets of $\mathbb{R}^{n_x}$ that include a neighbourhood of the origin are used to stabilise the control, not just the origin. The terminal constraint set, like the terminal equality constraint, does not employ a terminal cost, thus $E(x) = 0$. The MPC controller should steer the system to X_f within a finite time, after which a local stabilising controller $\kappa_f(\cdot)$ is employed. This methodology is usually referred to as dual mode control and was proposed by Michalska and Mayne (1993) in the context of constrained, nonlinear, continuous systems using a variable horizon N.

A *terminal cost and constraint set* is employed in most modern model predictive controller *theory* (Mayne *et al.*, 2000). If an infinite horizon objective function can be used, on-line optimisation is not necessary and stability and robustness can be guaranteed. In practical systems, constraints and other nonlinearities make the use of infinite horizons impossible, but it is possible to approximate an infinite horizon objective function if the system is suitably close to the origin. By choosing the terminal set X_f as a suitable subset of $\mathbb{R}^{n_x}$, the terminal cost $E(\cdot)$ can be chosen to approximate an infinite horizon objective function. A terminal cost and constraint set controller therefore needs a terminal constraint set $\mathbb{X}_f$ in which the terminal cost $E(\cdot)$ and infinite horizon feedback controller K_f are employed. To synthesise these, Sznaier and Damborg (as discussed in Mayne *et al.* (2000)) proposed that the terminal cost

$E(\cdot)$ and feedback controller K_f of a standard linear-quadratic (LQ) problem be used, which is an unconstrained infinite horizon problem, when the system is linear ($f(x,u) = Ax + Bu$) and the state and input constraint sets, $\mathbb{X}$ and $\mathbb{U}$, are polytopes. The terminal constraint set X_f is chosen to be the *maximal output admissible set* (Gilbert and Tan, 1991) of the system $f(x,u) = (A + BK_f)x$.

Most industrial or commercial MPC controllers do not use terminal costs or constraints, because they do not even provide nominal stability that these terminal costs and constraints are designed to provide and can be attributed to their DMC and IDCOM heritage (Qin and Badgwell, 2003). Most industrial MPC controllers, therefore, require brute-force simulation to evaluate the effects of model mismatch on closed-loop stability (Qin and Badgwell, 2003). Time spent tuning and testing of industrial controllers can, however, be significantly reduced if the controllers implement nominal and potentially robust stability measures, even though closed-loop stability of industrial MPC itself is not perceived to be a serious problem by industry practitioners (Qin and Badgwell, 2003).

3.3.1 Stability conditions for model predictive controllers

From the above discussion, it is clear that the addition of a terminal constraint set X_f, terminal cost $E(\cdot)$ and local feedback controller κ_f in the terminal constraint set forms the basis of stabilising MPC. Some conditions, in the form of axioms, are formulated (Mayne *et al.*, 2000) for the terminal constraint set, terminal cost and local feedback controller, which ensures that the controller is stabilising.

Two related methods are available for establishing stability. Both methods use a Lyapunov function as the objective function. The first method ensures that the objective function $\phi_N^{\text{opt}}(x_k)$ evolves with the state from x_k to $x_{k+1} = f(x_k, \kappa_N(x_k))$ so that

$$\phi_N^{\text{opt}}(x_{k+1}) - \phi_N^{\text{opt}}(x_k) + L(x_k, \kappa_N(x_k)) \quad \leq \quad 0, \tag{3.20}$$

while the alternative method uses the fact that

$$\phi_N^{\text{opt}}(x_{k+1}) - \phi_N^{\text{opt}}(x_k) + L(x_k, \kappa_N(x)) \quad = \quad \phi_N^{\text{opt}}(x_{k+1}) - \phi_{N-1}(x_{k+1}), \tag{3.21}$$

and shows that the right-hand side is negative, either directly or by showing that $\phi_1^{\text{opt}}(\cdot) \leq \phi_0^{\text{opt}}(\cdot)$ and exploiting monotonicity, which implies that if $\phi_1^{\text{opt}}(\cdot) \leq \phi_0^{\text{opt}}(\cdot)$ then $\phi_{i+1}^{\text{opt}}(\cdot) \leq \phi_i^{\text{opt}}(\cdot)$ for all $i \geq 0$.

Assume a model predictive controller that can steer the system state x to the terminal constraint set X_f within the prediction horizon N or fewer steps. The control sequence that accomplishes this is called an admissible or feasible control sequence $\mathbf{u} = \{u(0), u(1), ..., u(N-1)\}$. This control sequence should satisfy the control constraints $u(i) \in \mathbb{U}$ for $i = 0, 1, ..., N-1$ and ensure that the controlled states satisfy the state constraints $x_{x_k}^{\mathbf{u}}(i) \in \mathbb{X}$ for $i = 0, 1, .., N$

and the final state satisfies the terminal constraint set $x_{x_k}^{\mathbf{u}}(N) \in \mathbb{X}_f$. If the control problem $P_N(x_k)$ is solved, the control sequence $\mathbf{u}^{\text{opt}}(x_k)$ is obtained that will steer the system within the set of states that is possible with an MPC with horizon N, $x \in \mathbb{X}_N$. The optimal control sequence $\mathbf{u}^{\text{opt}}(x_k) = \left\{u_{x_k}^{opt}(0), u_{x_k}^{opt}(1), ..., u_{x_k}^{opt}(N-1)\right\}$ will result in the optimal state sequence $\mathbf{x}^{\text{opt}}(x_k) = \left\{x_{x_k}^{\text{opt}}(0), x_{x_k}^{\text{opt}}(1), ..., x_{x_k}^{\text{opt}}(N-1), x_{x_k}^{\text{opt}}(N)\right\}$. The first control action of $\mathbf{u}^{\text{opt}}(x_k)$, that is $u_k = \kappa_N(x_k) = u_{x_k}^{\text{opt}}(0)$ is implemented to get to the next state $x_{k+1} = f(x_k, \kappa_N(x_k)) = x_{x_k}^{\text{opt}}(1)$. A feasible control sequence $\tilde{\mathbf{x}}(x_{k+1})$ for the state x_{k+1}, will result in an upper bound for the optimal objective function $\phi_N^{\text{opt}}(x_{k+1})$, because a feasible control sequence should give a larger value for the objective function than an optimal control sequence. The abbreviated control sequence $\left\{u_{x_k}^{\text{opt}}(1), u_{x_k}^{\text{opt}}(2), ..., u_{x_k}^{\text{opt}}(N-1)\right\}$ derived from $\mathbf{u}^{\text{opt}}(x_k)$ should be a feasible control sequence to steer state x_{k+1} to $x_{x_k}^{\text{opt}}(N) \in \mathbb{X}_f$. If an extra term is added to the control sequence $\{u_{x_k}^{\text{opt}}(1), u_{x_k}^{\text{opt}}(2), ..., u_{x_k}^{\text{opt}}(N-1), v\}$, the control sequence will be feasible for $P_N(x_{k+1})$ if $v \in \mathbb{U}$ and v steers $x_{x_k}^{\text{opt}}(N) \in \mathbb{X}_f$ to $f(x_{x_k}^{\text{opt}}(N), v) \in \mathbb{X}_f$. This will be true if $v = \kappa_f(x_{x_k}^{\text{opt}}(N))$, with the terminal state constraint X_f and local controller $\kappa_f(\cdot)$ having the properties:

$$\mathbb{X}_f \subset \mathbb{X},\ \kappa_f(x_k) \in \mathbb{U} \quad \text{and} \quad f(x_k, \kappa_f(x_k)) \in \mathbb{X}_f\ \forall x_k \in \mathbb{X}_f, \tag{3.22}$$

implying that the terminal set $\mathbb{X}_f$ is invariant when the controller is $\kappa_f(\cdot)$. The feasible control sequence for $P_N(x_{k+1})$ is

$$\tilde{\mathbf{u}}(x_k) \quad = \quad \left\{u_{x_k}^{\text{opt}}(1), u_{x_k}^{\text{opt}}(2), ..., u_{x_k}^{\text{opt}}(N-1), \kappa_f(x_{x_k}^{\text{opt}}(N))\right\}, \tag{3.23}$$

with the associated cost

$$\begin{aligned} \phi_N(x_{k+1}, \tilde{\mathbf{u}}(x_k)) \quad = \quad & \phi_N^{\text{opt}}(x_k) - L(x_k, \kappa_N(x_k)) - E(x_{x_k}^{\text{opt}}(N)) \\ & + L(x_{x_k}^{\text{opt}}(N), \kappa_f(x_{x_k}^{\text{opt}}(N))) \\ & + E(f(x_{x_k}^{\text{opt}}(N), \kappa_f(x_{x_k}^{\text{opt}}(N)))). \end{aligned} \tag{3.24}$$

This cost $\phi_N(x_{k+1}, \tilde{\mathbf{u}}(x_k))$ is the upper bound on $\phi_N^{\text{opt}}(x_{k+1})$ and satisfies

$$\phi_N(x_{k+1}, \tilde{\mathbf{u}}(x_k)) \quad \leq \quad \phi_N^{\text{opt}}(x_k) - L(x_k, \kappa_N(x_k)), \tag{3.25}$$

if

$$E(f(x_k, \kappa_f(x_k))) - E(x_k) + L(x_k, \kappa_f(x_k)) \leq 0,\ \forall x_k \in \mathbb{X}_f. \tag{3.26}$$

The condition (3.26) will hold if $E(\cdot)$ is a control Lyapunov function in the neighbourhood of the origin and the controller κ_f and the terminal constraint set $\mathbb{X}_f$ are chosen appropriately. If the condition (3.26) is satisfied, then (3.20) will hold for all $x_k \in \mathbb{X}_N$, which is sufficient for the closed-loop system $x_{k+1} = f(x_k, \kappa_N(x_k))$ to converge to zero as time k tends to infinity, provided that the initial state is within $\mathbb{X}_N$. The stability conditions can be summarised in the following axioms (Mayne *et al.*, 2000):

A1: $\mathbb{X}_f \subset \mathbb{X}$, $\mathbb{X}_f$ is a closed set and $0 \in \mathbb{X}_f$. This condition implies that the state constraints should be satisfied in the terminal constraint set.

A2: $\kappa_f(x) \in \mathbb{U}$, $\forall x \in \mathbb{X}_f$. This condition implies that the constraints on the controls should be satisfied by the local controller in the terminal constraint set $\mathbb{X}_f$.

A3: $f(x, \kappa_f(x)) \in \mathbb{X}_f$, $\forall x \in \mathbb{X}_f$. This implies that the terminal constraint set $\mathbb{X}_f$ is positively invariant under the local controller $\kappa_f(\cdot)$.

A4: $E(f(x, \kappa_f(x))) - E(x) + L(x, \kappa_f(x)) \leq 0 \; \forall x \in \mathbb{X}_f$. The terminal cost function $E(\cdot)$ is a local Lyapunov function in the terminal constraint set $\mathbb{X}_f$.

The conditions, as summarised in A1 to A4, are merely sufficient conditions to ensure stability in model predictive controllers. These conditions can be shown to hold for the monotonicity approach as well as the continuous case (Mayne *et al.*, 2000). The following paragraphs will show how the stabilising methods of Section 3.3 satisfy the stability conditions A1 to A4.

3.3.2 Terminal state MPC

The *terminal state* variant of model predictive controllers (Mayne *et al.*, 2000) uses the terminal state $\mathbb{X}_f = \{0\}$ with no terminal cost $E(\cdot) = 0$. The local controller in the terminal constraint set is $\kappa_f(x) = 0$ that will ensure that the state remains at the origin if this controller is applied. The functions $E(\cdot)$ and $\kappa_f(\cdot)$ are only valid in $\mathbb{X}_f$ which is at the origin. The satisfaction of the stability conditions A1 to A4 are as follows:

A1: $\mathbb{X}_f = \{0\} \in \mathbb{X}$ - Satisfied.

A2: $\kappa_f(0) = 0 \in \mathbb{U}$ - Satisfied.

A3: $f(0, \kappa_f(0)) = f(0,0) = 0 \in \mathbb{X}_f$ - Satisfied.

A4: $E(f(0, \kappa_f(0))) - E(0) + L(0, \kappa_f(0)) = 0$ - Satisfied.

The controller ensures that the closed-loop system is asymptotically (exponentially) stable with region of attraction $\mathbb{X}_N$.

3.3.3 Terminal cost MPC

Terminal cost model predictive controllers are only valid in linear unconstrained (Bitmead *et al.*, 1990) and linear, stable, constrained (Rawlings and Muske, 1993) cases. In order to ensure stability, a terminal constraint is necessary if the system is nonlinear or linear, constrained and unstable. *Linear, unconstrained systems* are defined as $f(x,u) = Ax + Bu$,

and $L(x,u) = \frac{1}{2}(|x|_Q^2 + |u|_R^2)$ where $Q > 0$ and $R > 0$. The first three conditions A1 to A3 are trivially satisfied in the unconstrained case, because $\mathbb{X} = \mathbb{R}^{n_x}$ and $\mathbb{U} = \mathbb{R}^{n_u}$. In the case where A and B are stabilisable, the local controller is defined as $\kappa_f \triangleq K_f x$, and $P_f > 0$ should satisfy the Lyapunov equation

$$A_f^T P_f A_f + Q_f = 0, \quad A_f \triangleq A + BK_f, \quad Q_f \triangleq Q + K_f R K_f, \tag{3.27}$$

then the terminal cost function $E(x) \triangleq \frac{1}{2}x^T P_f x$ satisfies A4 and the closed-loop system is asymptotically (exponentially) stable with a region of attraction $\mathbb{R}^{n_x}$. *Linear, constrained, stable* systems have control constraints $u \in \mathbb{U}$, but no constraints on the states, thus $\mathbb{X} = \mathbb{X}_f = \mathbb{R}^{n_x}$. In order to satisfy A2, the controller function, if linear, should be $\kappa_f(x) = 0$ (Rawlings and Muske, 1993), that leads to the first three conditions (A1 to A3) being satisfied. The final condition A4 is satisfied if the terminal cost function is $E(x) \triangleq \frac{1}{2}x^T P_f x$, where P_f satisfies the Lyapunov equation $A^T P_f A + Q = 0$, that results in a controller with asymptotic (exponential) stability with region of attraction $\mathbb{R}^{n_x}$.

3.3.4 Terminal constraint set MPC

Terminal constraint set model predictive controllers employ a terminal constraint set $x_{x_k}^{\mathbf{u}}(N) \in \mathbb{X}_f$ without a terminal cost $E(x) = 0$ for nonlinear, constrained systems. Michalska and Mayne (1993) introduced the idea of a variable prediction horizon N for continuous-time, constrained, nonlinear systems. Scokaert *et al.* (1999) proposed a fixed horizon version for nonlinear, constrained, discrete-time systems. The controller steers the state of the system x to within the terminal constraint set $\mathbb{X}_f$, after which a local stabilising controller $\kappa_f(x) = K_f x$ is employed. This type of MPC is sometimes referred to as *dual-mode MPC*. This method is similar to the terminal equality constraint method, except that the equality $\{0\}$ is replaced by a set $\mathbb{X}_f$. The local controller $\kappa_f(\cdot)$ and the terminal constraint set $\mathbb{X}_f$ are chosen to satisfy the first three conditions A1 to A3. The local controller $\kappa_f(\cdot)$ is chosen to steer the system exponentially fast to the origin for all states in the terminal constraint set ($\forall x \in \mathbb{X}_f$). The stage cost of the objective function $L(x, \kappa_f(x))$ should be 0 when the system state is within the terminal constraint set $\mathbb{X}_f$ in order to satisfy A4. A suitable choice for the stage cost is

$$L(x,u) \quad \triangleq \quad \alpha(x)\overline{L}(x,u), \tag{3.28}$$

where $\alpha(x) = 1$, $\forall x \notin \mathbb{X}_f$, else $\alpha(x) = 0$ and $\overline{L}(x,u) = \frac{1}{2}(x^T Q x + u^T R u)$, where $Q > 0$ and $R > 0$. The closed-loop system is exponentially stable with domain of attraction $\mathbb{X}_N$, because the MPC controller steers the system with initial state $x \in \mathbb{X}_N$ within finite time to $\mathbb{X}_f$ with the controller value $\kappa_N(\cdot)$.

3.3.5 Terminal cost and constraint set MPC

In *linear, constrained systems* the terminal cost function can be chosen as $E(x) = \phi_{uc}^0(x) = \frac{1}{2}x^T P_f x$, that is the same as the unconstrained infinite horizon optimal control problem. The local controller $\kappa_f(x_k) = K_f x_k$ is the optimal infinite horizon controller and the terminal constraint set $\mathbb{X}_f$ is the maximal admissible set for the system $x_{k+1} = A_f x_k$, $A_f \triangleq A + BK_f$, thus satisfying A1-A4. This results in an exponentially stable controller with domain of attraction $\mathbb{X}_f$. The ideal choice for the terminal cost would be to choose $E(x) = \phi_\infty^{\text{opt}}(x)$, the objective function of an infinite horizon optimal controller, that would result in the objective function for the model predictive controller being $\phi_N^{\text{opt}}(x) = \phi_\infty^{\text{opt}}(x)$, and on-line optimisation would not be necessary. The resulting model predictive controller will have all the advantages of infinite horizon control. This is usually not practical, and the use of the terminal constraint set $\mathbb{X}_f$ and $E(x) = \phi_{uc}^0(x) = \frac{1}{2}x^T P_f x$ approximates the advantages of using $E(x) = \phi_\infty^{\text{opt}}(x)$. The nonlinear case is also given in Mayne *et al.* (2000).

From this discussion, it is clear that the use of a terminal constraint set $\mathbb{X}_f$, terminal cost function $E(\cdot)$ and local stabilising controller $\kappa_f(\cdot)$ is necessary to ensure stability in MPC. The first two requirements, terminal constraint set $\mathbb{X}_f$ and terminal cost function $E(\cdot)$, are explicitly incorporated into the controller, while the feedback controller $\kappa_f(\cdot)$ is only implicitly needed to prove stability. If the cost function $E(\cdot)$ is as close to the objective function $\phi_\infty^{\text{opt}}(\cdot)$ as possible, the closed-loop trajectory is exactly the same as that predicted by the solution of the optimal control problem $P_N(x)$.

3.4 ROBUST MPC - STABILITY OF UNCERTAIN SYSTEMS

Robust MPC is concerned with the stability and performance of the closed-loop system in the presence of uncertainty in the plant model. Early studies in the robustness of model predictive controllers considered unconstrained systems and found that if the Lyapunov function retains its descent property in the presence of disturbances (uncertainty), it will remain stable. In the constrained case, the problem becomes more complex, because the uncertainty or disturbances should not cause the closed-loop system to violate its state or control constraints.

Richalet *et al.* (1978) performed one of the earliest studies in robustness on systems with impulse response models, by investigating the effect of gain mismatches on the closed-loop system. Later work on systems modelled by impulse responses approached the optimal control problem as a min-max problem, that caused the problem to grow exponentially with the size of the prediction horizon.

There are several approaches to robust MPC, the first being a study of the robustness of MPC designed with a nominal model (that does not take uncertainty into account). The second

approach considers all the possible realisations of the uncertain system when calculating the open-loop optimal controller (min-max open-loop MPC). The open-loop nature of MPC is a problem when model uncertainty is present and the third approach addresses this by introducing feedback in the optimal control problem that is solved on-line.

For the discussion of robust MPC, the uncertain system is described as

$$x_{k+1} = f(x_k, u_k, w_k), \tag{3.29}$$
$$y_k = h(x_k), \tag{3.30}$$

where the state x_k and control u_k satisfy the same constraints

$$x_k \in \mathbb{X}, \tag{3.31}$$
$$u_k \in \mathbb{U}, \tag{3.32}$$

and the disturbance or uncertainty w_k satisfies $w_k \in W(x_k, u_k)$ for all k where, for each (x_k, u_k), $W(x_k, u_k)$ is closed and contains the origin in its interior. The disturbance sequence $\mathbf{w}^N \triangleq \{w^N(0), w^N(1), ..., w^N(N-1)\}$, together with the control sequence $\mathbf{u}^N$ and initial state x_k, will produce the resulting state sequence

$$\mathbf{x}^{\mathbf{u},\mathbf{w}}(x_k) = \left\{x_{x_k}^{\mathbf{u},\mathbf{w}}(0), x_{x_k}^{\mathbf{u},\mathbf{w}}(1), \ldots, x_{x_k}^{\mathbf{u},\mathbf{w}}(N-1), x_{x_k}^{\mathbf{u},\mathbf{w}}(N)\right\}. \tag{3.33}$$

Let $\mathscr{F}(x_k, u_k) \triangleq f(x_k, u_k, W(x_k, u_k))$, which will map values in $\mathbb{X}$ and $\mathbb{U}$ to subsets of $\mathbb{R}^{n_x}$, resulting in $x_{k+1} \in \mathscr{F}(x_k, u_k)$.

De Nicolao *et al.* (1996) and Magni and Sepulchre (1997) studied the inherent robustness of model predictive controllers that were designed without taking uncertainty into account. A more recent study by Grimm *et al.* (2004) found that there are examples showing that when MPC is applied to a nonlinear system, it is asymptotically stable without any robustness to measurement error or additive disturbances. This only happens in a nonlinear system where both the MPC feedback control law and the objective function are discontinuous at some point(s) in the interior of the feasibility region.

3.4.1 Stability conditions for robust MPC

Most versions of robust MPC take all the realisations of the uncertainty or disturbance w into consideration that requires strengthened assumptions to be satisfied, which are summarised as *robust* versions of axioms A1-A4 (Mayne *et al.*, 2000):

A1: $\mathbb{X}_f \subset \mathbb{X}$, $\mathbb{X}_f$ closed, $0 \in \mathbb{X}_f$.

A2: $\kappa_f(x) \in \mathbb{U}$, $\forall x \in \mathbb{X}_f$.

A3a: $f(x, \kappa_f(x), w) \in \mathbb{X}_f$, $\forall x \in \mathbb{X}_f$, $\forall w \in W(x, \kappa_f(x))$.

A4a: $E(f(x,\kappa_f(x),w))-E(x)+L(x,\kappa_f(x),w)\leq 0,\ \forall x\in\mathbb{X}_f,\ \forall w\in W(x,\kappa_f(x))$.

If $E(\cdot)$ is a robust Lyapunov function in the neighbourhood of the origin, there exists a triple $(E(\cdot),\mathbb{X}_f,\kappa_f(\cdot))$, which ensures that A4a is satisfied and results in an asymptotically or exponentially stable controller.

3.4.2 Open-loop min-max MPC

Open-loop min-max MPC considers all the possible realisations of the uncertain system in order to ensure that the state, control and terminal constraints are met for all the possible realisations (Michalska and Mayne, 1993). The objective function value in this case is determined for each realisation

$$J(x_k,\mathbf{u}^N,\mathbf{w}^N) \triangleq \sum_{i=0}^{N-1} L(x_{x_k}^{\mathbf{u},\mathbf{w}}(i),u^N(i))+E(x_{x_k}^{\mathbf{u},\mathbf{w}}(N)), \tag{3.34}$$

and the final objective value is the worst case for all the realisations

$$\phi_N(x_k,\mathbf{u}^N) \triangleq \max_{\mathbf{w}^N}\left\{J(x_k,\mathbf{u}^N,\mathbf{w}^N)|\mathbf{w}^N\in W_N(x_k,\mathbf{u}^N)\right\}, \tag{3.35}$$

where $W_N(x_k,\mathbf{u}^N)$ is the set of admissible disturbance sequences. Other choices are to take the objective value as the nominal objective value by using $\mathbf{w}^N=\mathbf{0}$. Badgwell (as discussed in Mayne *et al.* (2000)) used an interesting approach, where the controller should reduce the objective function value for every realisation, which is assumed finite, for a linear system. This is stronger than only reducing the worst-case objective value.

The set of admissible control sequences $\mathscr{U}_N^{ol}(x_k)$ is the one that satisfies the control, state and terminal constraints for all possible realisation of the disturbance sequence $\mathbf{w}^N$ when the initial state is x_k. Suppose the set $\mathbb{X}_i^{ol}$, for all $i\geq 0$, is the set of states that can be robustly steered to the terminal state constraint $\mathbb{X}_f$ in i steps or fewer by an admissible control sequence $\mathbf{u}^N\in\mathscr{U}_N^{ol}(x_k)$. The open-loop optimal control problem is

$$P_N^{ol}(x_k):\ \phi_N^{\text{opt}}(x_k) = \min_{\mathbf{u}^N}\left\{\phi_N(x_k,\mathbf{u}^N)|\mathbf{u}^N\in\mathscr{U}_N^{ol}(x_k)\right\}. \tag{3.36}$$

The solution to $P_N^{ol}(x_k)$ yields the optimal control sequence $\mathbf{u}^{\text{opt}}(x_k)$, where the implicit min-max control law is

$$\kappa_N^{ol}(x_k) \triangleq u_{x_k}^{\text{opt}}(0), \tag{3.37}$$

as in the nominal case. The control sequence will result in multiple optimal state sequences $\{\mathbf{x}^{\text{opt}}(x_k,\mathbf{w}^N)\}$ as a result of the disturbance sequence $\mathbf{w}^N$, so that

$$\mathbf{x}^{\text{opt}}(x_k,\mathbf{u}^{\text{opt}}) = \left\{x_{x_k,\mathbf{w}}^{\text{opt}}(0),x_{x_k,\mathbf{w}}^{\text{opt}}(1),\ldots,x_{x_k,\mathbf{w}}^{\text{opt}}(N-1),x_{x_k,\mathbf{w}}^{\text{opt}}(N)\right\}. \tag{3.38}$$

The triple $(E(\cdot), \mathbb{X}_f, \kappa_f(\cdot))$ is assumed to satisfy the stability conditions A1-A4a. Assume the process is started with an initial state $x_k \in \mathscr{X}_N^{ol}$ and has an optimal (and by implication a feasible) control sequence $\{u_{x_k}^{\text{opt}}(0), u_{x_k}^{\text{opt}}(1), ..., u_{x_k}^{\text{opt}}(N-1)\}$ for the optimal control problem $P_N^{ol}(x_k)$ that steers the state to within the terminal constraint set $\mathbb{X}_f$ within N steps or fewer, so that $x_{x_k,\mathbf{w}}^{\text{opt}}(N) \in \mathbb{X}_f$, $\forall w \in \mathscr{W}(x_k, \mathbf{u}^{\text{opt}}(x_k))$. As a result the abbreviated control sequence $\{u_{x_k}^{\text{opt}}(1), u_{x_k}^{\text{opt}}(2), ..., u_{x_k}^{\text{opt}}(N-1)\}$ should steer the state $x_{k+1} \in \mathscr{F}(x_k, \kappa_N(x_k))$ to the terminal constraint set $\mathbb{X}_f$ within $N-1$ steps or fewer, where $x_{k+1} \in \mathscr{X}_{N-1}^{ol}$. A problem arises when a feasible control sequence needs to be generated by adding a term to the abbreviated control sequence

$$\tilde{\mathbf{u}}(x_k) \quad = \quad \left\{u_{x_k}^{\text{opt}}(1), u_{x_k}^{\text{opt}}(2), ..., u_{x_k}^{\text{opt}}(N-1), v\right\}, \tag{3.39}$$

for the optimal control problem $P_N^{ol}(x_{k+1})$, where the control action $v \in \mathbb{U}$ is required to satisfy $f(x_{x_k,\mathbf{w}}^{\text{opt}}(N), v, w_N) \in \mathbb{X}_N$ for all $\mathbf{w}^N \in \mathscr{W}(x_k, \mathbf{u}^{\text{opt}}(x_k))$. The stability condition A3a does not ensure that such a control action v can be obtained, which prevents the upper bound of the objective function $\phi_N^{\text{opt}}(x_{k+1})$ from being calculated. Michalska and Mayne (1993) circumvent this problem by using a variable horizon optimal control problem $P(x_k)$ with decision variables $(\mathbf{u}^N, N)$. The optimal solution $(\mathbf{u}^{\text{opt}}(x_k); N^{\text{opt}}(x_k))$ is obtained by solving the optimal control problem $P(x_k)$, where

$$\mathbf{u}^{\text{opt}}(x_k) \quad = \quad \left\{u_{x_k}^{\text{opt}}(0), u_{x_k}^{\text{opt}}(1;x), ..., u_{x_k}^{\text{opt}}(N(x_k)-1)\right\}.$$

For the optimal control problem $P(x_{k+1})$ the solution $(\overline{\mathbf{u}}(x_k), N^{\text{opt}}(x_k) - 1)$ is a feasible solution for any $x_{k+1} \in \mathscr{X}(x_k, \kappa_N(x_k))$. The variable horizon objective function $\phi^{\text{opt}}(\cdot)$ and implicit controller $\kappa^{ol}(\cdot)$ will ensure that stability condition A4a holds for all $x_k \in \mathbb{X}_N^{ol} \subset \mathbb{X}_f$, $\forall w^N \in W(x_k, \kappa^{ol}(x_k))$. Inside the terminal constraint set $\mathbb{X}_f$, a suitable local controller $\kappa_f(\cdot)$ is used subject to stability conditions A1-A4a. This will result in an asymptotic (exponential) stable controller with domain of attraction $\mathbb{X}_N^{ol}$, subject to further modest assumptions (Michalska and Mayne, 1993).

3.4.3 Feedback robust MPC

Feedback robust MPC is better suited for uncertain systems than open-loop min-max controllers, because open-loop controllers assume that the trajectories of the system may diverge, which may cause $\mathbb{X}_N^{ol}$ to be very small, or even empty for a modest-sized prediction horizon N, which is very conservative. This happens because the open-loop min-max controllers do not take the effect of feedback into consideration, which would prevent the trajectories from diverging too much. To address the shortcomings of open-loop min-max control, feedback MPC was proposed by Lee and Yu (1997), Scokaert and Mayne (1998), Magni *et al.* (2001) and Kothare *et al.* (1996). In feedback MPC, the control sequence $\mathbf{u}$ is replaced by a control

policy π which is a sequence of control laws:

$$\pi \triangleq \{u(o), \kappa_1(\cdot), ..., \kappa_{N-1}(\cdot)\}, \tag{3.40}$$

where $\kappa_i(\cdot) : \mathbb{X} \to \mathbb{U}$ is a control law for each i, while $u(0)$ is a control action, because there is only one initial state. The resulting state sequence when applying the sequence of control laws π and starting at the initial state x_k subject to the disturbance sequence $\mathbf{w}$ is given by $\mathbf{x}^{\pi,\mathbf{w}}(x_k) = \{x_{x_k}^{\pi,\mathbf{w}}(0), x_{x_k}^{\pi,\mathbf{w}}(1), \ldots, x_{x_k}^{\pi,\mathbf{w}}(N-1), x_{x_k}^{\pi,\mathbf{w}}(N)\}$. The objective function for the feedback model predictive controller is

$$\phi_N(x_k, \pi) \triangleq \max_{\mathbf{w}^N} \{J(x_k, \pi, \mathbf{w}^N) | \mathbf{w}^N \in \mathscr{W}_N(x_k, \pi)\} \tag{3.41}$$

and the objective function for each realisation

$$J(x_k, \pi, \mathbf{w}^N) \triangleq \sum_{i=0}^{N-1} L(x_{x_k}^{\pi,\mathbf{w}}(i), u^{\pi}(i)) + E(x_{x_k}^{\pi,\mathbf{w}}(N)), \tag{3.42}$$

where $u^{\pi}(i) \triangleq \kappa_i(x_{x_k}^{\pi,\mathbf{w}}(i))$, $i = 1, \ldots, N-1$ and $u^{\pi}(0) = u(0)$. The admissible set of disturbances, given the control policy π is implemented, is $\mathscr{W}_N(x_k, \pi)$. The set of admissible control policies that will satisfy the control, state and terminal constraints for all the admissible disturbances with initial state x_k, is $\Pi_N(x_k)$. The set of initial states that can be steered to the terminal constraint set $\mathbb{X}_f$ by an admissible control policy π in i steps or fewer, is $\mathbb{X}_i^{fb}$, $\forall i \geq 0$. The feedback optimal control problem becomes

$$P_N^{fb}(x_k) : \phi_N^{\text{opt}}(x_k) = \min_{\pi} \{\phi_N(x_k, \pi) | \pi \in \Pi_N(x_k)\}. \tag{3.43}$$

If a solution to $P_N^{fb}(x_k)$ exists, the optimal control policy is

$$\pi^{\text{opt}}(x_k) = \left\{u_{x_k}^{\text{opt}}(0), \kappa_{1,x_k}^{\text{opt}}(\cdot), \kappa_{2,x_k}^{\text{opt}}(\cdot), ..., \kappa_{N-1,x_k}^{\text{opt}}(\cdot)\right\}, \tag{3.44}$$

where the implicit feedback MPC law is

$$\kappa_N^{fb}(x_k) \triangleq u_{x_k}^{\text{opt}}(0). \tag{3.45}$$

If the stability conditions A1-A4a are satisfied for $P_N^{fb}(x_k)$, a feasible control policy for $P_N^{fb}(x_{k+1})$ for all $x_{k+1} \in \mathscr{F}(x_k, \kappa_N^{fb}(x_k))$ and $x_k \in \mathbb{X}_N^{fb}$ is

$$\tilde{\pi}(x_k, x_{k+1}) \triangleq \left\{\kappa_{1,x_k}^{\text{opt}}(x_{k+1}), \kappa_{2,x_k}^{\text{opt}}(\cdot), ..., \kappa_{N-1,x_k}(\cdot), \kappa_f(\cdot)\right\}. \tag{3.46}$$

With this feasible control policy, and with $\mathbb{X}_N^{fb}$ an invariant set for $x_{k+1} \in \mathscr{F}(x_k, \kappa_N^{fb}(x_k))$, assumption A4a will be satisfied for all $x_k \in \mathbb{X}_N^{fb}$ and $w \in W(x_k, \kappa_N^{fb}(x_k))$. The resulting robust model predictive controller is asymptotically (exponentially) stable with domain of

attraction $\mathbb{X}_N^{fb}$ under further modest assumptions. The results are very similar to open-loop min-max control, except that the domain of attraction $\mathbb{X}_N^{fb}$ includes $\mathbb{X}_N^{ol}$ and could possibly be much larger. Feedback MPC is encouraging, but suffers from much higher complexity than open-loop min-max control.

Other formulations for robust MPC are also provided in Mayne *et al.* (2000).

3.5 ROBUST NONLINEAR MPC FORMULATIONS

3.5.1 Lyapunov-based robust model predictive control

Mhaskar and Kennedy (2008) propose a robust model predictive controller for nonlinear systems with constraints on the magnitude of the inputs and uncertainties as well as *rate constraints on the inputs*. Rate constraints can easily be handled by MPC formulations as soft constraints (Zheng and Morari, 1995), but the effect of rate constraints on stability and guaranteed satisfaction of rate constraints as hard constraints have not been addressed. The proposed formulation can handle rate constraints as soft constraints or as hard constraints when possible with fall-back to soft constraints to maintain feasibility and robust stability. The controller synthesis is Lyapunov-based to allow for explicit characterisation of the initial conditions that guarantee stabilisation, as well as state and control constraint satisfaction (Mhaskar *et al.*, 2005, 2006). The explicit characterisation of the initial conditions is important to ensure that stability is guaranteed, because the stability guarantees rely on stability constraints embedded in the optimisation problem. Guaranteeing stability requires feasibility of the stability constraints, which is assumed by most formulations, not guaranteed (Mhaskar and Kennedy, 2008).

Mhaskar *et al.* (2005) proposes a robust hybrid predictive control structure that acts as a safety net for any other MPC formulation. The other MPC formulations can explicitly take uncertainties into account, but it is not required. The control structure includes a Lyapunov-based robust nonlinear controller developed by El-Farra and Christofides (2003) that has well-defined stability characteristics and constraint-handling properties, but cannot be designed to be optimal to an arbitrary performance function as with MPC. The structure in addition contains switching laws that try to use the performance of MPC as much as possible, but falls back to the Lyapunov robust controller when the MPC fails to deliver a control move, which may be due to computational difficulties or infeasibility, or to instability of the MPC, which may be caused by inappropriate penalties or horizon length in the performance function (Mhaskar *et al.*, 2005).

Mhaskar (2006) proposes a robust model predictive controller for nonlinear systems that are subject to constraints and uncertainty and that is robust to *faults* in the control actuator. Lyapunov-based techniques (El-Farra and Christofides, 2003) are employed to design the robust model predictive controller by formulating constraints that explicitly account for the

uncertainties in the predictive control law and allow the set of initial conditions to be explicitly characterised that will guarantee constraint satisfaction initially and successive feasibility necessary for robust stability.

3.5.2 Reachable set methods

Feedback robust MPC is concerned with bounding the system trajectories to increase the possible domain of attraction compared to open-loop min-max robust MPC. Bravo *et al.* (2006) present an RNMPC for constrained nonlinear systems with uncertainties. The evolution of the system under uncertainties is predicted by using reachable sets of the system. The reachable sets are approximated by outer bounds on the reachable set. Bravo *et al.* (2006) use a method based on zonotopes (Alamo *et al.*, 2005) to describe the approximate reachable sets, which improves on a previous formulation by Limon *et al.* (2005) that used a conservative approximation of the reachable set based on interval arithmetic. The controller drives the system to a robust invariant set and forces the closed-loop system to be bounded by using contractive constraints.

3.5.3 Closed-loop min-max predictive control

Closed-loop min-max MPC reduces the conservatism of open-loop min-max MPC by incorporating the effect of feedback at each time-step into the optimisation problem. This limits the spread of the trajectories into the future and increases the feasible region of the controller. This was first proposed by Lee and Yu (1997) and further developed by Lazar *et al.* (2008), Limon *et al.* (2006), Magni *et al.* (2006, 2003). The stability of the closed-loop min-max MPC was studied in the input-to-state stability (ISS) framework (Jiang and Wang, 2001, Sontag, 1989, 1990, 1996) by Limon *et al.* (2006), Magni *et al.* (2006), where Limon *et al.* (2006) showed that in general only input-to-state *practical* stability (ISpS) (Jiang *et al.*, 1996, Sontag, 1996) can be ensured for min-max nonlinear MPC. It is desirable to have ISS compared to ISpS, because ISpS does not imply *asymptotic stability* when there are no disturbances on the inputs, while ISS ensures that *asymptotic stability* will be recovered when the input disturbances vanish. Lazar *et al.* (2008) proposed a new approach that guarantees ISS for min-max MPC of nonlinear systems with bounded disturbances.

3.5.4 Open-loop min-max predictive control

Open-loop min-max MPC of a continuous-time nonlinear system with constraints and uncertainties uses robust nonlinear optimal control at its core. The open-loop optimal control problem is solved for the current state measurement and the first control move implemented. The next state measurement is taken and the nonlinear optimal control problem resolved in a receding horizon fashion that leads to RNMPC.

In order to solve the continuous-time nonlinear optimal control problem, it should be discretized by casting it into a nonlinear parameter optimisation problem. Continuous-time nonlinear optimal control can be cast into a nonlinear parameter optimisation problem using the direct-multiple shooting formulation (Bock and Plitt, 1984).

Robust control can be obtained by performing robust nonlinear parameter optimisation, which is quite difficult to solve in general. Diehl *et al.* (2006) describes a method of approximating robust nonlinear optimisation through an approximate worst-case formulation. The approximation linearises the uncertainty set, described by norm bounded parameter uncertainty, to obtain penalty functions for the objective function and constraint functions. The penalty functions are further approximated to maintain the sparsity of the large-scale problem as well as the smoothness of the objective and constraints functions, enabling efficient implementation of the resulting approximate robust nonlinear parameter optimisation problem. This formulation can be solved quite efficiently by using a real-time iteration scheme for nonlinear optimisation proposed by Diehl *et al.* (2005).

3.5.5 Linear embedding of nonlinear models

Embedding of the trajectories of the original nonlinear system into a family of linear plants is exploited by some authors to implement robust MPC of even highly nonlinear systems by using linear methods (Casavola *et al.*, 2003). Linear matrix inequality (LMI) based controllers produce feedback policies, which are implemented at each time interval. The problem with these controllers is that they use an ellipsoid invariant set for their domain of attraction, which makes them conservative. This is because the sets must be symmetric, and in systems where the constraints are non-symmetric, the ellipsoid sets will be a small subset of the maximum admissible set. The feedback robust MPC technique was introduced by Kothare *et al.* (1996). The technique was improved by Cuzzola *et al.* (2002) by describing the uncertain system as a polytope and applying different Lyapunov functions to each vertex of the uncertain polytope to reduce the conservatism of the method. The method uses semidefinite programming (SDP) to solve the minimisation problem on-line, which is computationally very expensive compared to quadratic programming (QP) used in nominal MPC. Further improvements made by Casavola *et al.* (2004), Ding *et al.* (2004), Wan and Kothare (2003) resulted in an attempt to move as much as possible of the calculation offline.

Nonlinear systems with uncertainties and constraints can be approximated by radial basis function – auto-regressive with exogenous input (RBF-ARX) models, which are hybrid pseudo-linear time-varying models that contain elements of Gaussian RBF neural network and linear ARX model structures. Peng *et al.* (2007) used this RBF-ARX model in conjunction with a min-max MPC to do RNMPC. The min-max MPC is based on the LMI-based method proposed in Cuzzola *et al.* (2002) that falls in the feedback robust MPC framework.

An approach to feedback robust MPC is proposed by Langson *et al.* (2004) who use tubes to

encapsulate all the possible states that can result from the controller. If the uncertainties can be sufficiently described, the optimisation problem only has to be calculated once, and the control policy will steer the system to the terminal constraint set $\mathbb{X}_f$, where a local stabilising controller will keep the uncertain system in the terminal constraint set.

The robust model predictive controllers do not always provide off-set free tracking and this problem is addressed by Wang and Rawlings (2004*a*, *b*), who use a robust predictor that updates itself each time measurements are available to ensure that the off-set is eliminated. Pannocchia and co-workers (Pannocchia, 2004, Pannocchia and Kerrigan, 2003, 2005) approached the problem by designing a robust linear feedback controller and an appropriate invariant set where the controller will satisfy the constraints. The controller uses the "pre-stabilisation" approach suggested by Rossiter *et al.* (1998) and later implemented by Kouvaritakis *et al.* (2000), Schuurmans and Rossiter (2000) and Lee and Kouvaritakis (2000), where the feedback law $u_i(\cdot)$ in the policy π is restricted to have the form $u_i(x) = v_i + Kx$, $i = 0,1,2,...,N-1$, that changes the optimisation problem to calculating the free control moves $\{v_0, v_1, v_2, ..., v_{N-1}\}$ rather than the policy. The "pre-stabilisation" of the controller in Pannocchia and Kerrigan (2005), however, uses a dynamic state feedback controller, rather than a static state feedback gain to obtain the offset free tracking property.

3.6 NONLINEAR MODEL PREDICTIVE CONTROL

NMPC takes a measurement of the current state of the plant and then uses a nonlinear model to predict the future behaviour of the plant in order to calculate the optimal control moves or control laws with regard to a specified objective function. NMPC is derived from nonlinear optimal control over a constant or varying time interval into the future $[t_k, t_k + T]$. Only the first control move or control law is implemented and a new state measurement is taken. The nonlinear optimal control problem is then recalculated for the new time interval $[t_{k+1}, t_{k+1} + T]$, which leads to receding horizon control (Mayne *et al.*, 2000).

The nonlinear optimal control problem is to find a control profile $u(\cdot)$ such that it minimises a particular scalar performance index

$$\min_{x,u} \quad \phi_c(x,u) \tag{3.47}$$

$$\text{s.t.} \quad \dot{x}(t) = f_c(x(t), u(t), \tilde{p}) \tag{3.48}$$

$$\theta_c(x,u) \leq 0 \tag{3.49}$$

$$t_1 \triangleq 0,\ x_1 \triangleq x(t_1) \tag{3.50}$$

$$t_f \triangleq T,\ x_f \triangleq x(t_f),\ \psi(x_f) = 0 \tag{3.51}$$

where $x : \mathbb{R} \to \mathbb{R}^{n_x}$ is the state trajectory, $u : \mathbb{R} \to \mathbb{R}^{n_u}$ is the control trajectory, $x(t) \in \mathbb{R}^{n_x}$ is the state vector, $\dot{x}(t) \in \mathbb{R}^{n_x}$ is the state sensitivities to time, $u(t) \in \mathbb{R}^{n_u}$ is the control vector, $x_f \in \mathbb{R}^{n_x}$ is the terminal state vector, $(x,u) \mapsto \phi_c(x,u)$ is the scalar performance function,

$(x,u) \mapsto \theta_c(x,u)$ is the inequality constraints function, $\psi : \mathbb{R}^{n_x} \to \mathbb{R}$ is the terminal constraint function, $\tilde{p} \in \mathbb{R}^{n_p}$ is the nominal parameter vector and $f_c : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ is the ordinary differential equation describing the dynamics of the plant. The plant dynamics are time invariant and each optimal control problem can therefore be solved from time $t_1 = 0$ without affecting the result. The initial state value $x_1 \in \mathbb{R}^{n_x}$ is the currently measured state of the system. The final state $x_f \in \mathbb{R}^{n_x}$ will be a fixed value based on the setpoint for the current iteration of the optimisation problem, because the terminal constraints are defined as equality constraints (3.55). The final state value x_f may vary from iteration to iteration if setpoint changes are made. The final time t_f of the optimisation problem is fixed for this implementation.

For the sequel the ordered pair $(a,b) \triangleq \begin{bmatrix} a^T & b^T \end{bmatrix}^T$ is defined as a column vector.

The nonlinear optimal control problem, consisting of a system with continuous dynamics, needs to be discretized in order to be cast in terms of a nonlinear parameter optimisation problem. This is accomplished by dividing the prediction horizon $[0,T]$ into N discrete time intervals called nodes (Hull, 1997) $t_0 \triangleq 0 < t_1 < t_2 < \cdots < t_k < \cdots < t_{N-1} < t_N \triangleq T$ where the sampling time is defined as $\tau_s \triangleq t_{k+1} - t_k$. The functions of time $x(\cdot)$ and $u(\cdot)$ are replaced by their values at the nodes $x_k \in \mathbb{R}^{n_x}$ and $u_k \in \mathbb{R}^{n_u}$ for $k = 0,\ldots,N$ and some form of interpolation between the nodes.

The resulting nonlinear controlled discrete-time system is $x_{k+1} \triangleq f_k(x_k,u_k,\tilde{p})$, $k = 0,1,\ldots,N-1$. The nonlinear optimal control problem can now be cast into the following nonlinear parameter optimisation problem

$$\min_{\mathbf{s},\mathbf{q}} \quad \phi(\mathbf{s},\mathbf{q}) \tag{3.52}$$

$$\text{s.t.} \quad g(\mathbf{s},\mathbf{q},\tilde{\mathbf{p}}) = 0 \tag{3.53}$$

$$\theta_{i,j}(s_j,q_j) \leq 0, \quad \begin{matrix} i = 1,\ldots,n_c, \\ j = 1,\ldots,N-1, \end{matrix} \tag{3.54}$$

$$\theta_N(s_N) \leq 0, \tag{3.55}$$

where $\phi : \mathbb{R}^{N\cdot n_x} \times \mathbb{R}^{(N-1)\cdot n_u} \to \mathbb{R}$ is the performance function to be optimised, $g : \mathbb{R}^{N\cdot n_x} \times \mathbb{R}^{(N-1)\cdot n_u} \times \mathbb{R}^{N\cdot n_p} \to \mathbb{R}^{N\cdot n_x}$ is the equality constraint function that describes the discrete time system dynamics, $\theta_{i,j} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$, $i = 1,\ldots,n_c$, $j = 1,\ldots,N-1$ are the inequality constraint functions, $\theta_N : \mathbb{R}^{n_x} \to \mathbb{R}$ is the terminal constraint function, $s_i \in \mathbb{R}^{n_x}$, $i = 1,\ldots,N$ are the estimated state parameters, $q_i \in \mathbb{R}^{n_u}$, $i = 1,\ldots,N-1$ are the control parameters, $\tilde{\mathbf{p}} \triangleq (\tilde{p},\ldots,\tilde{p}) \in \mathbb{R}^{(N-1)n_p}$ is the sequence of nominal model parameters, $\mathbf{s} \triangleq (s_1,\ldots,s_N)$ is the state sequence and $\mathbf{q} \triangleq (q_1,\ldots,q_{N-1})$ is the control sequence to be optimised in the nonlinear optimisation problem (Diehl *et al.*, 2005).

Using only the control moves in the parameter optimisation problem leads to a single integration of the state equations over the time interval $[0,T]$. If the time interval is long, the accuracy of the numerical integration is affected. The accuracy of the numerical derivatives

for a given perturbation size is affected even more. To compensate for this problem, an estimate of the state value x_k at each time node is made (represented by s_k) and the system dynamics are integrated between nodes. This method is called direct multiple shooting (Bock and Plitt, 1984, Hull, 1997). The nonlinear optimiser then removes any error between the estimate (s_k) and actual dynamics (x_k) through the equality constraints (3.53).

3.7 ROBUST NONLINEAR MODEL PREDICTIVE CONTROL

Plant models always differ from the real system owing to incomplete modelling, parameter uncertainty and unmodelled disturbances. In this section, the NMPC developed earlier in (Coetzee *et al.*, 2008) are robustified to parameter uncertainty, by approximating the worst-case realisations of the objective function and the constraint functions to the parameter variations (Coetzee *et al.*, 2009). The worst-case objective function is then minimised subject to the worst-case constraints. The approximated min-max optimisation problem is called an approximate robust counterpart formulation (Ben-Tal and Nemirovskii, 2001, 2002, Diehl *et al.*, 2006).

3.7.1 Parameter uncertainty description

Consider an uncertain parameter vector $p \in \mathbb{R}^{n_p}$ and nominal parameter vector $\tilde{p} \in \mathbb{R}^{n_p}$, which are assumed to be restricted to a generalised ball

$$\mathbb{P} = \{p \in \mathbb{R}^{n_p} \,|\, \|p - \tilde{p}\| \leq 1\} \tag{3.56}$$

defined by using a suitable norm $\|\cdot\|$ in $\mathbb{R}^{n_p}$ (Diehl *et al.*, 2006). A suitable norm may be the scaled Hölder q-norm ($1 \leq q \leq \infty$), $\|p\| = \left\|A^{-1}p\right\|_q$, with an invertible $A \in \mathbb{R}^{n_p \times n_p}$ matrix.

The Hölder q-norm is also suitable for describing box uncertainty where the upper p_u and lower p_l bounds on the parameters p are known:

$$\begin{aligned} \mathbb{P}_{\text{box}} &\triangleq \{p \in \mathbb{R}^{n_p} | p_l \leq p \leq p_u\}, \\ &= \left\{ p \in \mathbb{R}^{n_p} \left| \left\| \text{diag}\left(\frac{p_u - p_l}{2}\right)^{-1} \left(p - \frac{p_l + p_u}{2}\right) \right\|_\infty \leq 1 \right. \right\} \end{aligned} \tag{3.57}$$

where the centre of the box is defined as $\bar{p} \triangleq \frac{p_l + p_u}{2} \in \mathbb{R}^{n_p}$. In general the centre of the box $\bar{p}$ and the nominal parameter vector $\tilde{p}$ do not have to be the same point ($\tilde{p} \neq \bar{p}$).

The dual norm $\|\cdot\|_{\#}$ for the norm $\|\cdot\|$ is the mapping

$$\begin{aligned} \|\cdot\|_{\#} : \mathbb{R}^{(N-1)\cdot n_p} &\rightarrow \mathbb{R} \\ \mathbf{a} &\mapsto \|\mathbf{a}\|_{\#} \triangleq \max_{\mathbf{p}\in\mathbb{R}^{(N-1)\cdot n_p}} \mathbf{a}^T\mathbf{p} \text{ s.t. } \|\mathbf{p}\| \leq 1, \end{aligned} \tag{3.58}$$

and the dual norm $\|\cdot\|_{*}$ for the norm $\|\cdot\|$ is the mapping

$$\begin{aligned} \|\cdot\|_{*} : \mathbb{R}^{n_p} &\rightarrow \mathbb{R} \\ a &\mapsto \|a\|_{*} \triangleq \max_{p\in\mathbb{R}^{n_p}} a^T p \text{ s.t. } \|p\| \leq 1, \end{aligned} \tag{3.59}$$

It is well known that for any scaled Hölder q-norm $\|p\| = \|Ap\|_q$ (A an invertible matrix, $1 \leq q \leq \infty$), the dual norm is $\|a\|_{*} = \left\|A^{-1}a\right\|_{\frac{q}{q-1}}$ (for $q = 1$, define $\frac{q}{q-1} \triangleq \infty$ and for $q = \infty$, define $\frac{q}{q-1} \triangleq 1$) as observed in the context of the worst-case analysis by Ma and Braatz (2001) and independently by Bock and Kostina (2001).

The q-norm $\|\cdot\|_q : \mathbb{R}^{n_p} \longrightarrow \mathbb{R}$ for $1 \leq q \leq \infty$ is defined as

$$\|\mathbf{x}\|_q = \left\{ \sum_{i=1}^{n_p} |x_i|^q \right\}^{\frac{1}{q}} \tag{3.60}$$

and the case where $q = \infty$ becomes

$$\|\mathbf{x}\|_\infty = \max_{1\leq i\leq n_p} |x_i| . \tag{3.61}$$

3.7.2 Direct approximate robust counterpart formulation

To add uncertainty into an optimisation problem, a min-max optimisation can be done (Diehl *et al.*, 2006, Ma and Braatz, 2001). The worst-case value for the cost $\phi(\mathbf{s},\mathbf{q})$ is defined as

$$\psi(\mathbf{q}) \triangleq \max_{\mathbf{s},\mathbf{p}} \quad \phi(\mathbf{s},\mathbf{q}) \tag{3.62}$$

$$\text{s.t.} \quad g(\mathbf{s},\mathbf{q},\mathbf{p}) = 0 \tag{3.63}$$

and the worst-case values for the constraint functions $\theta_{i,j}(s_j,q_j)$ are defined as

$$\omega_{i,j}(q_j) \triangleq \max_{s_j,p_j} \quad \theta_{i,j}(s_j,q_j) \tag{3.64}$$

$$\text{s.t.} \quad g_{\tau_s}(s_j,q_j,p_j) = 0, \tag{3.65}$$

where $g_{\tau_s} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_x}$ is the system dynamic for one time step τ_s, $p_i \in \mathbb{P}_{box}$, $i = 1,\ldots N-1$ are defined as the unknown time varying model parameters and $\mathbf{p} \triangleq (p_1,\ldots,p_{N-1}) \in \mathbb{P}_{box}^{(N-1)}$ is defined as the sequence of time varying model parameters. The worst-case cost and constraint functions are calculated by maximising the cost function and constraint func-

tions with regard to the model parameter sequence $\mathbf{p} \in \mathbb{P}_{box}^{(N-1)}$ and state values $\mathbf{s} \in \mathbb{R}^{(N \cdot n_x)}$. The worst-case cost function $\psi(\mathbf{q})$ is then minimised by choosing the control moves $\mathbf{q} \triangleq (q_1, q_1, \ldots, q_{N-1}) \in \mathbb{R}^{((N-1)\cdot n_u)}$ subject to the worst-case constraints $\omega_{i,j}(q_j)$

$$\min_{\mathbf{q}} \quad \psi(\mathbf{q}) \tag{3.66}$$

$$\text{s.t.} \quad \omega_{i,j}(q_j) \leq 0, \quad \begin{matrix} i = 1, \ldots, n_c, \\ j = 1, \ldots, N-1. \end{matrix} \tag{3.67}$$

This min-max optimisation problem is difficult to solve for general nonlinear systems. The optimisation problem can, however, be simplified by approximating the worst-case calculations for the cost $\tilde{\psi}(\mathbf{q})$ and the constraints $\tilde{\omega}_{i,j}(q_j)$. The approximate robust counterpart formulation is given by

$$\min_{\mathbf{q}} \quad \tilde{\psi}(\mathbf{q}) \tag{3.68}$$

$$\text{s.t.} \quad \widetilde{\omega}_{i,j}(q_j) \leq 0, \quad \begin{matrix} i = 1, \ldots, n_c, \\ j = 1, \ldots, N-1, \end{matrix} \tag{3.69}$$

where the approximation of the worst-case cost $\tilde{\psi}(\mathbf{q})$ and constraints $\tilde{\omega}_{i,j}(\mathbf{q})$ can be done through linearisation of the system dynamics $g(\mathbf{s}, \mathbf{q}, \mathbf{p}) = 0$ and the cost $\phi(\mathbf{s}, \mathbf{q})$ and constraint $\theta_{i,j}(s_j, q_j)$ functions. The approximation of the worst-case cost, $\psi(\mathbf{q})$ by $\tilde{\psi}(\mathbf{s}, \mathbf{q})$, is defined by a convex optimisation problem

$$\max_{\Delta\mathbf{s}, \Delta\mathbf{p}} \quad \phi(\mathbf{s}, \mathbf{q}) + \frac{\partial \phi}{\partial \mathbf{s}}(\mathbf{s}, \mathbf{q}) \Delta\mathbf{s} \tag{3.70}$$

$$\text{s.t.} \quad \frac{\partial g}{\partial \mathbf{s}}(\mathbf{s}, \mathbf{q}, \bar{\mathbf{p}}) \Delta\mathbf{s} + \frac{\partial g}{\partial \mathbf{p}}(\mathbf{s}, \mathbf{q}, \bar{\mathbf{p}}) \Delta\mathbf{p} = 0, \tag{3.71}$$

$$\|\Delta p_i\| \leq 1, \, i = 1, \ldots, N-1, \tag{3.72}$$

and the approximation of the worst-case constraints $\omega_{i,j}(q_j)$ by $\tilde{\omega}_{i,j}(\mathbf{s}, \mathbf{q})$ are defined as

$$\max_{\Delta\mathbf{s}, \Delta\mathbf{p}} \quad \theta_{i,j}(s_j, q_j) + \frac{\partial \theta_{i,j}}{\partial \mathbf{s}}(s_j, q_j) \Delta\mathbf{s} \tag{3.73}$$

$$\text{s.t.} \quad \frac{\partial g}{\partial \mathbf{s}}(\mathbf{s}, \mathbf{q}, \bar{\mathbf{p}}) \Delta\mathbf{s} + \frac{\partial g}{\partial \mathbf{p}}(\mathbf{s}, \mathbf{q}, \bar{\mathbf{p}}) \Delta\mathbf{p} = 0, \tag{3.74}$$

$$\left\|\Delta p_j\right\| \leq 1, \; j = 1, \ldots, N-1 \tag{3.75}$$

where $\Delta\mathbf{s} \in \mathbb{R}^{(N \cdot n_x)}$, $\bar{\mathbf{p}} \triangleq (\bar{p}, \ldots, \bar{p}) \in \mathbb{R}^{(N-1)\cdot n_p}$ is a sequence of parameters at the centre of the box, $\Delta p_j \triangleq p_j - \bar{p} \in \mathbb{R}^{n_p}$ is the deviation of the model parameters from the centre of the box and $\Delta\mathbf{p} \triangleq (\Delta p_1, \ldots, \Delta p_{N-1}) \in \mathbb{R}^{(N-1)\cdot n_p}$ is defined as the sequence of model parameter deviations.

The optimisation problems (3.70)-(3.75) have analytical solutions. The approximation for the worst-case cost can be expressed as

$$\tilde{\psi}(\mathbf{s},\mathbf{q}) = \phi(\mathbf{s},\mathbf{q}) + \left\| -\left(\frac{\partial g}{\partial \mathbf{p}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^T \left(\frac{\partial g}{\partial \mathbf{s}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^{-T} \left(\frac{\partial \phi}{\partial \mathbf{s}}(\mathbf{s},\mathbf{q})\right)^T \right\|_{\#} \tag{3.76}$$

and the approximation of the worst-case constraints can be expressed as

$$\tilde{\omega}_{i,j}(\mathbf{s},\mathbf{q}) = \theta_{i,j}(s_j,q_j) + \left\| -\left(\frac{\partial g}{\partial \mathbf{p}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^T \left(\frac{\partial g}{\partial \mathbf{s}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^{-T} \left(\frac{\partial \theta_{i,j}}{\partial \mathbf{s}}(s_j,q_j)\right)^T \right\|_{\#} \tag{3.77}$$

where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$ as described in (3.59). For notational convenience the uncertainty term of (3.76) is defined as

$$\Delta\phi(\mathbf{s},\mathbf{q}) \triangleq \left\| -\left(\frac{\partial g}{\partial \mathbf{p}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^T \left(\frac{\partial g}{\partial \mathbf{s}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^{-T} \left(\frac{\partial \phi}{\partial \mathbf{s}}(\mathbf{s},\mathbf{q})\right)^T \right\|_{\#} \tag{3.78}$$

and the uncertainty term for (3.77) is defined as

$$\Delta\theta_{i,j}(\mathbf{s},\mathbf{q}) \triangleq \left\| -\left(\frac{\partial g}{\partial \mathbf{p}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^T \left(\frac{\partial g}{\partial \mathbf{s}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^{-T} \left(\frac{\partial \theta_{i,j}}{\partial \mathbf{s}}(s_j,q_j)\right)^T \right\|_{\#} \tag{3.79}$$

The approximate robust counterpart of (3.68)-(3.69) can now be expressed as

$$\min_{\mathbf{s},\mathbf{q}} \quad \phi(\mathbf{s},\mathbf{q}) + \Delta\phi(\mathbf{s},\mathbf{q}) \tag{3.80}$$

$$\text{s.t.} \quad \theta_{i,j}(s_j,q_j) + \Delta\theta_{i,j}(\mathbf{s},\mathbf{q}) \leq 0, \quad \begin{matrix} i = 1,\ldots,n_c, \\ j = 1,\ldots,N-1, \end{matrix} \tag{3.81}$$

$$g(\mathbf{s},\mathbf{q},\bar{\mathbf{p}}) = 0 \tag{3.82}$$

In order to solve this approximate min-max optimisation problem (3.68)-(3.69) efficiently, new optimisation variables $\mathbf{D} \in \mathbb{R}^{(N\cdot n_x)\times n_p}$ are defined (Diehl *et al.*, 2006) in the form of a sensitivity matrix $\mathbf{D} \triangleq (D_1, D_2, \ldots, D_N) = \left(-\left(\frac{\partial g}{\partial \mathbf{s}}(\bar{\mathbf{s}},\mathbf{q},\bar{\mathbf{p}})\right)^{-1} \frac{\partial g}{\partial \mathbf{p}}(\bar{\mathbf{s}},\mathbf{q},\bar{\mathbf{p}})\right)$ to prevent the explicit calculation of the matrix inverse in (3.78) and (3.79), which would reduce the sparsity of the problem. The direct approximate robust counterpart formulation becomes

$$\min_{\mathbf{s},\mathbf{q},\mathbf{D}} \quad \phi(\mathbf{s},\mathbf{q}) + \left\| \mathbf{D}^T \left(\frac{\partial \phi}{\partial \mathbf{s}}(\mathbf{s},\mathbf{q})\right)^T \right\|_{\#} \tag{3.83}$$

$$\text{s.t.} \quad g(\mathbf{s},\mathbf{q},\bar{\mathbf{p}}) = 0, \tag{3.84}$$

$$\frac{\partial g(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})}{\partial \mathbf{s}}\mathbf{D} + \frac{\partial g(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})}{\partial \mathbf{p}} = 0, \tag{3.85}$$

$$\theta_{i,j}(s_j,q_j) + \left\| D_j^T \left(\frac{\partial \theta_{i,j}}{\partial s_j}(s_j,q_j)\right)^T \right\|_* \leq 0, \quad \begin{matrix} i = 1,\ldots,n_c, \\ j = 1,\ldots,N-1, \end{matrix} \tag{3.86}$$

$$\theta_N(s_N) \leq 0. \tag{3.87}$$

In order to maintain smooth objective and constraint functions in the optimisation problem, slack variables $\delta \triangleq (\delta_{0,1}, \ldots \delta_{n_c,1}, \ldots, \delta_{n_c,N-1})$, $\delta_{i,j} \in \mathbb{R}$, $i = 0, \ldots, n_c$, $j = 1, \ldots, N-1$ are introduced to replace the dual norms in the optimisation problem (3.83)-(3.87) (Diehl *et al.*, 2006).

The nonlinear parameter optimisation problem that is solved at each time step is given by

$$\min_{\mathbf{s},\mathbf{q},\mathbf{D},\delta} \quad \phi(\mathbf{s},\mathbf{q}) + e^T\delta_{0,1} + \cdots + e^T\delta_{0,N-1} \tag{3.88}$$

$$\text{s.t.} \quad g(\mathbf{s},\mathbf{q},\bar{\mathbf{p}}) = 0, \tag{3.89}$$

$$\frac{\partial g(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})}{\partial \mathbf{s}}\mathbf{D} + \frac{\partial g(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})}{\partial \mathbf{p}} = 0, \tag{3.90}$$

$$-\delta_{0,j} \leq D_j^T \left(\frac{\partial \phi}{\partial s_j}(\mathbf{s},\mathbf{q})\right)^T \leq \delta_{0,j}, \quad j = 1, \ldots, N, \tag{3.91}$$

$$\theta_{i,j}(s_j,q_j) + e^T\delta_{i,j} \leq 0, \quad \begin{matrix} i = 1,\ldots,n_c, \\ j = 1,\ldots,N-1, \end{matrix} \tag{3.92}$$

$$-\delta_{i,j} \leq D_j^T \left(\frac{\partial \theta_{i,j}}{\partial s_j}(s_j,q_j)\right)^T \leq \delta_{i,j}, \quad \begin{matrix} i = 1,\ldots,n_c, \\ j = 1,\ldots,N-1, \end{matrix} \tag{3.93}$$

$$\theta_N(s_N) \leq 0. \tag{3.94}$$

where $e \triangleq (1, \ldots, 1) \in \mathbb{R}^{n_p}$ (Diehl *et al.*, 2006, 2005).

3.7.3 RNMPC implementation

For implementation of the RNMPC, the scalar performance function and the discrete time system dynamics function need to be defined. The scalar performance function is defined as

$$\phi(\mathbf{s},\mathbf{q}) \triangleq \sum_{i=1}^{N-1} L_i(s_i,q_i) + E(s_N) \tag{3.95}$$

where the scalar interval performance indexes and the terminal performance index are defined as quadratic functions

$$L_i(s_i,q_i) \triangleq h(s_i,q_i)^T Q h(s_i,q_i) + \Delta q_i^T R \Delta q_i \tag{3.96}$$

$$E(s_N) \triangleq s_N^T P s_N, \tag{3.97}$$

where Q and R represent the weighting matrices on the outputs and controls respectively, $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$ is the function that maps the current state and control vector to the output vector using nominal model parameters, $\Delta q_i \triangleq q_i - q_{i-1}$ and P is the terminal cost weighting matrix.

The equality constraint function describing the discrete time system dynamics is defined as

$$g(\mathbf{s},\mathbf{q},\mathbf{p}) \triangleq \begin{cases} x_k - s_1, \\ f_1(s_1,q_1,p_1) - s_2, \\ \vdots \\ f_{N-1}(s_{N-1},q_{N-1},p_{N-1}) - s_N. \end{cases} \tag{3.98}$$

where $f_i(s_i,q_i,p_i)$, $i = 1,\ldots,N-1$ are defined as the function values at the nodes for the continuous time dynamics $\dot{x} = f_c(x(t),u(t),p(t))$. The interpolation between nodes is defined as integrating the state equations for one sample time τ_s

$$\begin{aligned} x_{k+1} &\triangleq f_k(x_k,u_k,p_k) \\ &\triangleq \int_{t_k}^{t_k+\tau_s=t_{k+1}} f_c(x,u_k,p_k)\,dt \end{aligned} \tag{3.99}$$

with initial conditions $x(t_k) \triangleq x_k = s_k$ and the controls $u_k = q_k$ constant over the interval $[t_k,t_{k+1}]$. The integration of the state equations can be done with a software package called CPPAD (Lougee-Heimer, 2003). CPPAD is also capable of calculating sensitivities of the integral. This is useful for calculating the sensitivities of the system dynamics $f_k(x_k,u_k,p_k)$ with regard to states s_k, inputs q_k and parameters p_k. It is also capable of calculating second order derivatives of the integral. CPPAD does automatic differentiation of specially modified C++ code. It works by recording all the mathematical operations being done in the desired function and then uses the chain rule of differentiation to calculate the derivatives. The advantages of automatic differentiation are that it is fast to calculate and does not suffer from truncation errors present in other numerical methods. CPPAD also contains a module to solve ordinary differential equations (ODEs) with the Runge-Kutta method that is required to integrate the state equations. The derivatives of the integral are then calculated by automatic differentiation.

Nonlinear optimisation software is required to solve the nonlinear parameter optimisation problem stated in (3.88)-(3.94). The software used is a package called IPOPT (Kawajir *et al.*, 2006), which is useful when solving large-scale sparse nonlinear optimisation problems.

IPOPT requires the following functions to be provided:

- Scalar performance function value $\varphi(X)$
- Gradient of the performance $\nabla\varphi(X)$
- Constraint functions values $\vartheta(X)$
- Jacobian of the constraints $\nabla\vartheta(X)$

where X is the vector of all decision variables.

For the problem specified in (3.88)-(3.94), the vector of decision variables is defined as

$$X \triangleq (s_1, q_1, D_1, \delta_{0,1}, \ldots, \delta_{n_c,1}, \ldots, s_{N-1}, q_{N-1}, D_{N-1}, \delta_{0,N-1}, \ldots, \delta_{n_c,N-1}, s_N, D_N) \quad (3.100)$$

where $X \in \mathbb{R}^{((n_x+n_u+n_D+n_{slack})\cdot(N)+n_x+n_D)}$, $n_D \triangleq n_x \times N$ and $n_{slack} \triangleq (n_c+1) \times n_p$.

All matrix variables in X are unrolled into vectors to simplify implementation. The matrix is unrolled by placing the rows of the matrix sequentially to form a vector.

The scalar performance function is defined in terms of the decision variables X as

$$\varphi(X) \triangleq \sum_{i=0}^{N-1} \left(L_i(s_i, q_i) + e^T \delta_{0,i}\right) + E(s_N) \quad (3.101)$$

where the scalar interval performance indexes and the terminal performance index are the same as in (3.96) and (3.97).

The gradient of the performance function $\nabla\varphi(X) \in \mathbb{R}^{((n_x+n_u+n_D+n_{slack})\cdot N+n_x+n_D)}$ then becomes

$$\nabla\varphi(X) = \begin{pmatrix} 2Qs_1 \\ 2Rq_1 \\ 0_{n_D} \\ 1_{n_p} \\ 0_{n_c \cdot n_p} \\ \vdots \\ 2Qs_{N-1} \\ 2Rq_{N-1} \\ 0_{n_D} \\ 1_{n_p} \\ 0_{n_c \cdot n_p} \\ 2Ps_N \\ 0_{n_D} \end{pmatrix}. \quad (3.102)$$

where $0_n \triangleq (0, \ldots, 0) \in \mathbb{R}^{n_x}$ and $1_n \triangleq (1, \ldots, 1) \in \mathbb{R}^{n_x}$.

The values of the constraint functions $\vartheta(X)$ are interleaved equality and inequality constraint functions with the same decision variables that will result in a Jacobian of the constraints $\nabla\vartheta(X)$ that is sparse and banded. For RNMPC, the following structure for the constraints

function $\vartheta(X) \in \mathbb{R}^{((n_x+n_D+n_c+2\cdot n_{slack})\cdot N+n_x+n_D)}$ is defined

$$\vartheta(X) \triangleq \begin{pmatrix} \vartheta_1(X) \\ \vartheta_2(X) \\ \vdots \\ \vartheta_{N-1}(X) \\ \vartheta_N(X) \end{pmatrix} \tag{3.103}$$

where

$$\vartheta_1(X) \triangleq \begin{pmatrix} x_k - s_1 \\ -I_{n_x} D_2 + 0 \\ \theta_{1,1}(s_1,q_1) + e^T \delta_{1,1} \\ \vdots \\ \theta_{n_c,1}(s_1,q_1) + e^T \delta_{n_c,1} \\ D_2^T \left(\frac{\partial L_1}{\partial s_1}(s_1,q_1)\right)^T - \delta_{0,1} \\ D_2^T \left(\frac{\partial L_1}{\partial s_1}(s_1,q_1)\right)^T + \delta_{0,1} \\ D_2^T \left(\frac{\partial \theta_{1,1}}{\partial s_1}(s_1,q_1)\right)^T - \delta_{1,1} \\ D_2^T \left(\frac{\partial \theta_{1,1}}{\partial s_1}(s_1,q_1)\right)^T + \delta_{1,1} \\ \vdots \\ D_2^T \left(\frac{\partial \theta_{n_c,1}}{\partial s_1}(s_1,q_1)\right)^T - \delta_{n_c,1} \\ D_2^T \left(\frac{\partial \theta_{n_c,1}}{\partial s_1}(s_1,q_1)\right)^T + \delta_{n_c,1} \end{pmatrix}, \tag{3.104}$$

$$\vartheta_i(X) \triangleq \begin{pmatrix} f_{i-1}(s_{i-1},q_{i-1},\bar{p}) - s_i \\ \left(\begin{array}{c} \frac{\partial f_{i-1}(s_{i-1},q_{i-1},\bar{p})}{\partial s_{i-1}} \cdot D_{i-1} - \\ I_{n_x} \cdot D_i + \frac{\partial f_{i-1}(s_{i-1},q_{i-1},\bar{p})}{\partial p} \end{array}\right) \\ \theta_{1,i}(s_i,q_i) + e^T \delta_{1,i} \\ \vdots \\ \theta_{n_c,i}(s_i,q_i) + e^T \delta_{n_c,i} \\ D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T - \delta_{0,i} \\ D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T - \delta_{1,i} \\ \vdots \\ D_{i+1}^T \left(\frac{\partial \theta_{n_c,i}}{\partial s_i}(s_i,q_i)\right)^T - \delta_{n_c,i} \\ D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T + \delta_{0,i} \\ D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T + \delta_{1,i} \\ \vdots \\ D_{i+1}^T \left(\frac{\partial \theta_{n_c,i}}{\partial s_i}(s_i,q_i)\right)^T + \delta_{n_c,i} \end{pmatrix}, \forall i = 2,\ldots,N-1, \tag{3.105}$$

$$\vartheta_N(X) \triangleq \begin{pmatrix} f_{N-1}(s_{N-1},q_{N-1},\bar{p}) - s_N \\ \left(\begin{array}{c} \frac{\partial f_{N-1}(s_{N-1},q_{N-1},\bar{p})}{\partial s_{N-1}} \cdot D_{N-1} - \\ I_{n_x} \cdot D_N + \frac{\partial f_{N-1}(s_{N-1},q_{N-1},\bar{p})}{\partial p} \end{array}\right) \end{pmatrix}. \tag{3.106}$$

which results in a Jacobian of the constraints function $\nabla\vartheta(X) \in \mathbb{R}^{n_{\text{jac\_row}} \times n_{\text{jac\_col}}}$ where $n_{\text{jac\_row}} \triangleq ((n_x + n_D + n_c + 2 \cdot n_{slack}) \cdot N + n_x + n_D)$ and $n_{\text{jac\_col}} \triangleq ((n_x + n_u + n_D + n_{slack}) \cdot N + n_x + n_D)$, with a block structure defined in Figure 3.1. The blocks lie on the diagonal of the Jacobian matrix

$$\nabla\vartheta(X) = \begin{bmatrix} \nabla\vartheta_1(X) & 0 & 0 & 0 \\ 0 & \nabla\vartheta_2(X) & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \nabla\vartheta_{N-1}(X) \end{bmatrix} \tag{3.107}$$

where the first 2×3 entries of each block are the same entries as the last 2×3 entries of the previous block.

IPOPT only requires the non-zero entries to be populated for this sparse Jacobian matrix. The number of non-zero entries for this structure is defined as $n_{nz} \triangleq N(n_x + n_x(n_x + n_u)) + N(n_D(2n_x + n_u + 1)) + N(n_c(n_x + 2n_u + N)) + 2N(n_{slack}(2n_x + 2n_u + 1)) + n_x + n_D - n_c \cdot n_u - 2 \cdot n_{slack} \cdot n_u$

All the sensitivities and second-order derivatives as shown in the constraints (3.104)-(3.106) and Jacobian of the constraints (Figure 3.1) are also calculated by CPPAD (Lougee-Heimer, 2003).

$$\nabla \vartheta_i(X) = \begin{bmatrix}
-\mathbb{I}_{n_x} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
0 & -\mathbb{I}_{n_D} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
\frac{\partial \theta_{1,i}(s_i,q_i)}{\partial s_i} & 0 & \frac{\partial \theta_{1,i}(s_i,q_i)}{\partial q_i} & 0_{n_p}^T & 0_{n_p}^T & \cdots & 0_{n_p}^T & 0 & 0 & 0 & 0 \\
\frac{\partial \theta_{2,i}(s_i,q_i)}{\partial s_i} & 0 & \frac{\partial \theta_{2,i}(s_i,q_i)}{\partial q_i} & 0_{n_p}^T & 1_{n_p}^T & \cdots & 0_{n_p}^T & 0 & 0 & 0 & 0 \\
\vdots & 0 & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial \theta_{nc,i}(s_i,q_i)}{\partial s_i} & 0 & \frac{\partial \theta_{nc,0}(s_i,q_i)}{\partial q_i} & 0_{n_p}^T & 0_{n_p}^T & \cdots & 1_{n_p}^T & 0 & 0 & 0 & 0 \\
\frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} & -\mathbb{I}_{n_p} & 0 & 0 & 0 & 0 & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} & \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\
\frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} & 0 & -\mathbb{I}_{n_p} & 0 & \cdots & 0 & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} & 0 & 0 & 0 & -\mathbb{I}_{n_p} & 0 & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\
\frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} & \mathbb{I}_{n_p} & 0 & 0 & 0 & 0 & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} & \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\
\frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} & 0 & \mathbb{I}_{n_p} & 0 & 0 & 0 & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} & 0 & 0 & 0 & \mathbb{I}_{n_p} & 0 & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\
\frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} & 0 & \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial q_i} & 0 & 0 & 0 & 0 & -\mathbb{I}_{n_x} & 0 & 0 & 0 \\
\left(\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial s_i} + \frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial p}}{\partial s_i}\right) & 0 & \left(\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial q_i} + \frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial p}}{\partial q_i}\right) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mathbb{I}_{n_D} \\
 & & & & & & \frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial D_i} & & & &
\end{bmatrix} \tag{3.108}$$

Figure 3.1: Jacobian block.

3.8 STATE OBSERVERS

Some mathematical models of plants use state-space descriptions. State-space description of plant make practical control difficult, because it is not usually possible to measure all the states in the model and some sort of reconstruction of the states from the available measurements is needed. An *observer* can be constructed from a mathematical model of the plant to estimate the states from the available measurements (Marquez and Riaz, 2005). The milling circuit is described mathematically by nonlinear state-space models in Section 2.3.4. Observers were not used in this thesis and only a quick overview is provided here for completeness.

State observers for nonlinear system are more challenging than for linear systems, because the separation principle does not apply in general for nonlinear system (Freeman, 1995), where the separation principle states that a stable controller and a stable observer can be designed independently for a plant and the combined closed-loop system will be stable. Atassi and Khalil (1999, 2000) showed that high-gain observers can be constructed for certain classes of nonlinear systems that maintain the separation principle.

Some possible observers that can be used for nonlinear systems with robustness to model mismatches are input-output observers (Marquez and Riaz, 2005), robust H_∞ observers based on LMI for nonlinear systems with time-varying uncertainties (Abbaszadeh and Marquez, 2009), sliding mode observers (Bartolini *et al.*, 2003, Davila *et al.*, 2005, Spurgeon, 2008, Xiong and Saif, 2001), high gain observers (Atassi and Khalil, 2000) and moving horizon estimators or moving horizon state observers (Chu *et al.*, 2007, Michalska and Mayne, 1995, Rao *et al.*, 2003).

Sliding mode observers have received much attention recently, because: "*Sliding mode observers have unique properties, in that the ability to generate a sliding motion on the error between the measured plant output and the output of the observer ensures that a sliding mode observer produces a set of state estimates that are precisely commensurate with the actual output of the plant. It is also the case that analysis of the average value of the applied observer injection signal, the so-called equivalent injection signal, contains useful information about the mismatch between the model used to define the observer and the actual plant. These unique properties, coupled with the fact that the discontinuous injection signals which were perceived as problematic for many control applications have no disadvantages for software-based observer frameworks, have generated a ground swell of interest in sliding mode observer methods in recent years.*" (Spurgeon, 2008) and "*For both relatively general non-linear system representations and also application specific models with significant non-linearity sliding mode observers are seen to be at the forefront of robust techniques for state and parameter estimation.*" (Spurgeon, 2008)

Moving horizon estimators (MHE) or moving horizon state observers (MHSO) provide a very close parallel to MPC controllers (Chu *et al.*, 2007). It uses a moving window of

previous measurements to obtain an estimate of the current state of the system (Rao *et al.*, 2003). It can handle nonlinear systems and inequality constraints on the decision variables explicitly (Rao *et al.*, 2003). Recently, Chu *et al.* (2007) proposed a refinement to MHSOs that allow MHSOs to handle model uncertainty in addition to external disturbances in a robust manner to produce a robust moving horizon state observer (RMHSO). The RMHSO can possibly be combined with the RNMPC of Section 3.7 to obtain an output-feedback RNMPC (Michalska and Mayne, 1995).

3.9 CONCLUSION

This chapter outlines the development of MPC and stability theory for MPC. The stability theory focuses on the requirements for exponential stability, while some control formulations are based on other stability formulations such as Lyapunov, asymptotic, ISS and ISpS. It outlines the RNMPC theory (Section 3.7) that will be applied to the nonlinear model presented in Section 2.3.4. Simulation results of the RNMPC applied to the nonlinear milling circuit model are provided in Chapter 5 for different operational conditions.

CHAPTER 4

PID CONTROL

The PI controllers presented in this chapter serve only as an example implementation to provide a baseline for comparison with RNMPC and NMPC. Single-loop PI controllers without a MIMO compensator or a centralised design were used, because more than 60% of all respondents still use PI controllers, according to a recent survey by Wei and Craig (2009), usually single-loop PI controllers. A decentralised PID controller design that takes interaction into account was attempted in Addendum C.

The PI controllers presented here are not intended to serve as the best PI controller design for the presented milling circuit based on an exhaustive study, because that was not the main focus of this thesis. The comparison of the RNMPC and NMPC controllers to the PI controllers should, therefore, not be seen as a definitive, but rather serve as an example of possible benefits that RNMPC can provide over PI control typically employed in industry (Wei and Craig, 2009), when large feed disturbances are common in the milling circuit.

4.1 INTRODUCTION

PID control is a fundamental feedback control method employed broadly in process control. PID control usually forms the lowest level of control with multi-variable controllers such as MPC providing the setpoints for the PID controllers. PID control can be described in the time domain by the following algorithm

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(t)d\tau + T_d\frac{de(t)}{dt}\right) \tag{4.1}$$

where the error is defined as the difference between the plant output and the setpoint ($e(t) := y(t) - r(t)$). The transfer function form of the PID controller is given by

$$\frac{U(s)}{Y(s)} = C_{parallel}(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right) \tag{4.2}$$

where K is the gain, T_i the integral time and T_d the derivative time. The PID form in (4.2) is known as the parallel ("ideal", "non-interacting") form. The tuning rules based on the IMC principle developed by Skogestad (2003) uses the series (cascading, "interacting") form given by

$$\frac{U(s)}{Y(s)} = C_{series}(s) = K_c \left(\frac{\tau_I s + 1}{\tau_I s} \right) \cdot (\tau_D s + 1) \tag{4.3}$$

where K_c is the proportional gain, τ_I is the integral time and τ_D the derivative time for the series form of the controller. The parameters from the series form can be converted to the parallel form by

$$\begin{aligned} K &= K_c \left(1 + \frac{\tau_D}{\tau_I} \right) \\ T_i &= \tau_I \left(1 + \frac{\tau_D}{\tau_I} \right) \\ T_d &= \frac{\tau_D}{1 + \frac{\tau_D}{\tau_I}} \end{aligned} \tag{4.4}$$

The two transfer functions (equation (4.2) and (4.3)) describing the PID controller are improper. If the closed-loop transfer function is also improper, a suitable order filtering term should be added to the denominator to produce a proper closed-loop transfer function.

4.2 PI CONTROL WITH ANTI-WINDUP

The derivative term in PID control allows the controller to react quickly to sudden changes in the process. The derivative term of the PID controller acts on process noise as if the process is changing rapidly, which causes undesirable closed-loop behaviour. Industry therefore usually only employs PI control rather than full PID control, because the derivative term is so sensitive to noise. The rest of this chapter will therefore focus only on PI control.

Windup in PI/PID control is when there is saturation of the control action that prevents the control error from reaching zero. This will cause the integrator value to keep on increasing/decreasing in an effort to eliminate the control error. If the plant output passes the setpoint value, the sign of the error will change, but the integrator value has to wind down before normal operation can resume. Anti-windup therefore forces the integrator input to zero when saturation occurs to prevent it from winding up (Åström, 2002).

In Figure 4.4 the input to the integrator is the control error multiplied by the integrator gain

$$\frac{K}{T_i} e$$

where e is the control error, but if the error cannot vanish due to saturation on the control, the integrator value will keep increasing. To prevent windup, a second control loop is added

to the integrator input (Åström, 2002)

$$\frac{K}{T_i}e + \frac{1}{Tt}e_s$$

where e_s is the error between the desired control action v and the saturated control action u and T_t is the tracking time constant that controls how fast the controller resets after saturation. When the control action u saturates, the error e_S equals the control error e

$$e_s = -\frac{KT_t}{T_i}e \tag{4.5}$$

that results in the desired control action value that settles at

$$v = u_{\text{lim}} + \frac{KT_t}{T_i}e \tag{4.6}$$

which is slightly higher than the saturation value and prevents the integrator from winding up. The smaller T_t is, the faster the integrator resets and the quicker the controller can react to a change in error. The tracking time constraint should ideally be chosen to be larger than T_d and smaller than T_i and as a rule of thumb can be chosen to be $T_t = \sqrt{T_i T_d}$ (Åström, 2002).

Implementing anti-windup for PID is defined for the parallel form and the controller structure is shown in Figure 4.4.

4.3 LINEARISED MODELS FOR SIMC TUNING METHOD

Linearised models are necessary to design the PI controllers using the SIMC method. Linearised models can be created by performing step tests on the nonlinear model and performing system identification (SID) on the step responses.

The output-input pairings for single-loop controllers on multivariable systems are very important, because the output should be paired with the input that has the most influence on that output and the input should have the least interaction with other outputs. The traditional output-input pairings used on milling circuits are LOAD-MFS, PSE-SFW and SLEV-CFF (Chen *et al.*, 2007*b*, Conradie and Aldrich, 2001, Lynch, 1979, Napier-Munn and Wills, 2006). This pairing is not without its problems when used on industrial plants, as described by Chen *et al.* (2007*b*): *"Decoupled PID control had been frequently interrupted by changes in mineral ore hardness, feed rate, feed particle size, etc., ..."* This statement was supported by preliminary simulations using the above-mentioned pairing where the sump would either overflow or underflow as soon as ore hardness and composition disturbances were introduced. Craig and MacLeod (1996) also found SLEV control to be the most problematic aspect of controlling the milling circuit.

An alternative output-input pairing was then investigated that paired LOAD-MFS, PSE-CFF and SLEV-SFW (Smith *et al.*, 2001). The pairing of SLEV-SFW, rather than PSE-SFW and SLEV-CFF, was traditionally used only when a fixed speed sump discharge pump was available (Lynch, 1979). The pairing LOAD-MFS, PSE-CFF and SLEV-SFW, however, shows much better robustness to the feed disturbances subject to actuator limitations, as shown later in Section 5.3.2 and Addendum B.

The single-loop PI controllers are designed with the above-mentioned output-input pairings. The interactions between loops are ignored, because they cannot be included in the PI controller design using the SIMC tuning method, unlike other methods (Pomerleau *et al.*, 2000). The PI controllers are SISO controllers and the three controlled variables (PSE, LOAD and SLEV) will be independently controlled by three independent PI loops. Neither a multivariable compensator (Vázquez and Morilla, 2002) nor a centralised design (Morilla *et al.*, 2008) will be used for the PI controllers, because most plants that use PI controllers employ only single-loop PI controllers (Wei and Craig, 2009). An attempt at decentralised PID tuning that takes interactions into account was made in Addendum C.

4.3.1 PSE – CFF model

PSE exhibits a non-minimum phase first order response with time-delay to a change in CFF. A first order transfer function model was fitted to the step response data that has the following form:

$$G_{\text{PSE-CFF}}(s) = K_{PC}\frac{(1+Z_{PC}s)}{(1+P_{PC}s)}e^{(-\theta_{PC}s)} \tag{4.7}$$

$$G_{\text{PSE-CFF}}(s) = -0.00035\frac{(1-0.63s)}{(1+0.54s)}e^{(-0.011s)} \tag{4.8}$$

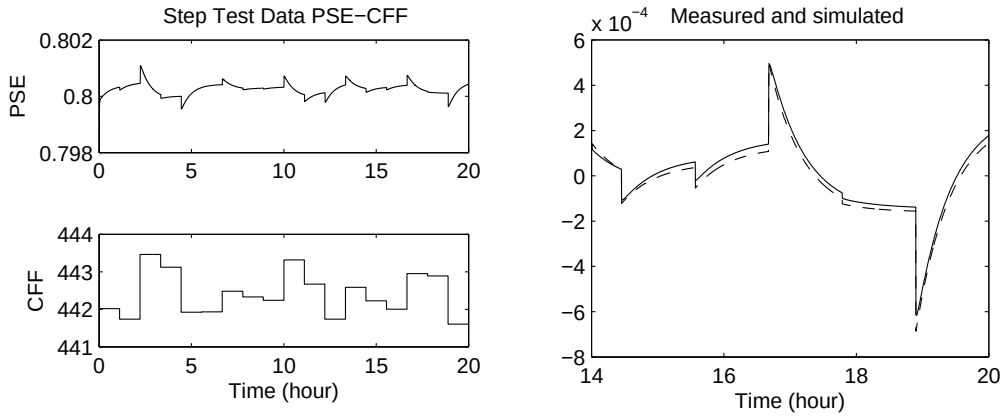
The model form was changed to include a zero in order to improve the fit of the linear model to the step response data of the nonlinear model. The step response data for the model fitting as well as the comparison between the linear and nonlinear models are shown in Figure 4.1. The linear model for PSE-CFF shows good agreement with the nonlinear model response.

4.3.2 LOAD – MFS model

The mill load volume exhibits an integrating response to the feed-rate of ore. An integrator transfer function model is fitted to the step test data of the nonlinear model with the following form:

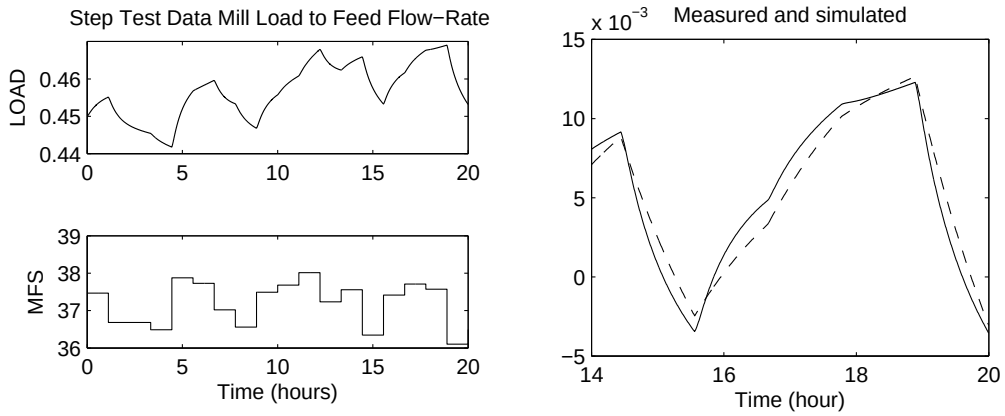
$$G_{\text{LOAD-MFS}}(s) = \frac{K_{LF}}{s} \tag{4.9}$$

$$G_{\text{LOAD-MFS}}(s) = \frac{0.01}{s} \tag{4.10}$$



(a) PSE vs CFF (b) Nonlinear vs Linear model

Figure 4.1: The change in PSE with a step change in CFF. The nonlinear (solid line) model is compared to the linear model (dashed line).



(a) LOAD vs MFS (b) Nonlinear vs Linear Model

Figure 4.2: The change in LOAD with a step change in MFS. The nonlinear (solid line) model is compared to the linear model (dashed line).

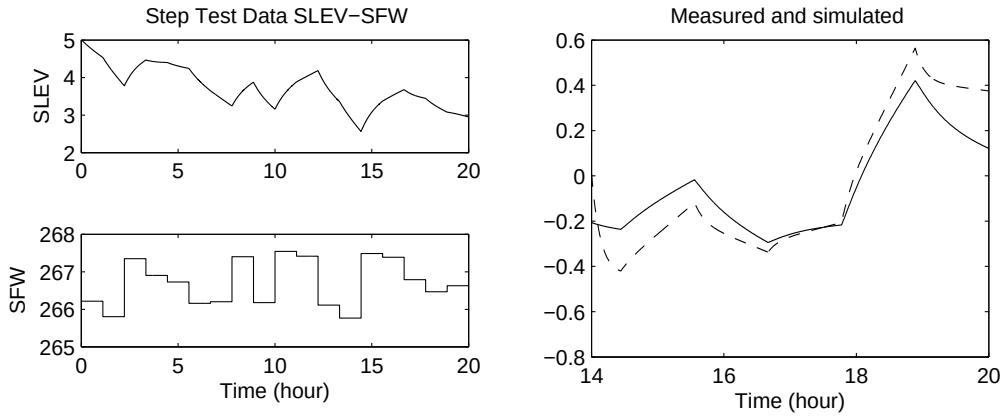
The step response data for the model fitting, as well as the comparison between the linear and nonlinear models, are shown in Figure 4.2.The linear model for PSE-CFF shows good agreement with the nonlinear model response.

4.3.3 SLEV – SFW model

The sump level exhibits an integrating response to the flow-rate of water added to the sump. An integrator transfer function model is fitted to the step test data of the nonlinear model with the following form:

$$G_{\text{SLEV-SFW}}(s) = \frac{K_{SW}}{s} \tag{4.11}$$

$$G_{\text{SLEV-SFW}}(s) = \frac{0.42}{s} \tag{4.12}$$



(a) SLEV vs SFW (b) Nonlinear vs Linear Model

Figure 4.3: The change in SLEV with a step change in SFW. The nonlinear (solid line) model is compared to the linear model (dashed line).

The step response data for the model fitting, as well as the comparison between the linear and nonlinear models, are shown in Figure 4.3. The linear model for PSE-CFF shows good agreement with the nonlinear model response.

4.4 SIMC TUNING METHOD

SIMC is a model-based tuning method with only a single tuning parameter for the closed-loop response. It is based on the IMC method. The PID parameter settings for the SIMC-PID method (Skogestad, 2003) are given in Table 4.1. The method tries to obtain a first-order closed loop response with time delay of the form

$$\left(\frac{y}{r}\right)_{\text{desired}} = \frac{1}{\tau_c s + 1} e^{-\theta s} \tag{4.13}$$

where τ_c is the desired closed-loop time constant, which is the only tuning parameter. If the process models are not in the forms given in Table 4.1, then they should be manipulated to conform to the basic forms given in Table 4.1 using the rules given below. Only the rules from Skogestad (2003) that apply to the linear models of Section 4.3 are given below.

4.4.1 Simplifying first-order transfer function models

Skogestad (2003) developed simplification rules to get almost any arbitrary transfer function model into either a first-order transfer function model with time delay or a second order model with time delay. Only the rules that apply to the linear models obtained in Section 4.3 will be given here.

The first-order linear model for the PSE-CFF loop contains a negative numerator time constant relating to a non-minimum phase zero. This is cast into the first-order response of

Table 4.1: SIMC-PID settings with τ_c as a tuning parameter for the serial form of the PID controller (from Skogestad (2003)).

Process	$g(s)$	K_c	τ_i	τ_D
First-order	$k\frac{e^{-\theta s}}{(\tau_1 s+1)}$	$\frac{1}{k}\frac{\tau_i}{\tau_c+\theta}$	$\min\{\tau_1, 4(\tau_c+\theta)\}$	—
Integrating	$k\frac{e^{-\theta s}}{s}$	$\frac{1}{k}\frac{1}{\tau_c+\theta}$	$4(\tau_c+\theta)$	—

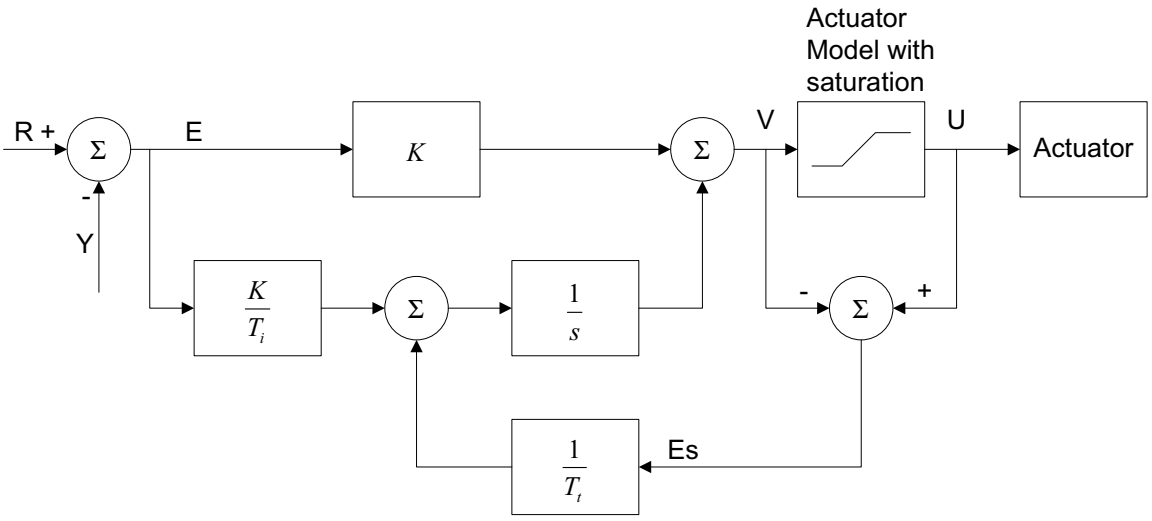


Figure 4.4: PI Controller with Anti-Windup (from Åström (2002)).

Table 4.1 by subtracting the value of the negative numerator time constant from the time delay to obtain the effective first-order time delay (Skogestad, 2003).

To illustrate the simplification, start with a first-order transfer function model with time delay of the following form

$$G_{\text{fo}} = K \cdot \frac{(1+Zs)}{(1+Ps)} e^{-\theta s}, \tag{4.14}$$

and define the effective time delay as

$$\theta_{\text{effective}} := \theta - Z. \tag{4.15}$$

Applying the effective time delay to the transfer function model in (4.14) gives the equivalent first-order transfer function model

$$G_{\text{fo-eq}} = K \cdot \frac{1}{(1+Ps)} e^{-\theta_{\text{effective}} s}. \tag{4.16}$$

4.5 IMPLEMENTATION

The PI parameter values are obtained from Table 4.1 using the simplified models obtained by following the rules outlined above. The PI parameter values are for the serial form of the

controller as well as the parallel form, because there is no differential term. The structure of the anti-windup PI controller is shown in Figure 4.4.

4.5.1 PI controller for the PSE-CFF loop

The PSE-CFF loop is characterised by a first-order transfer function model with time delay. The model is derived in Section 4.3.1 and given by

$$G_{\text{PSE-CFF}}(s) = K_{PC}\frac{(1+Z_{PC}s)}{(1+P_{PC}s)}e^{(-\theta_{PC}s)} \tag{4.17}$$

$$G_{\text{PSE-CFF}}(s) = -0.00035\frac{(1-0.63s)}{(1+0.54s)}e^{(-0.011s)} \tag{4.18}$$

with the effective time delay given by

$$\theta_{\text{PC-EFF}} = \theta_{PC} - Z_{PC} \tag{4.19}$$

$$= 0.01 + 0.63 \tag{4.20}$$

$$= 0.64 \tag{4.21}$$

to give the equivalent first-order model

$$G_{\text{PSE-CFF-EQ}}(s) = K_{PC}\frac{1}{(1+P_{PC}s)}e^{(-\theta_{\text{PC-EFF}}s)} \tag{4.22}$$

$$= -0.00035\frac{1}{(1+0.54s)}e^{(-0.64s)} \tag{4.23}$$

that gives the following PI parameter values by using the rules of Table 4.1

$$K_c = -1187,\ \tau_i = 2.6,\ \tau_d = 0. \tag{4.24}$$

4.5.2 PI controller for the LOAD-MFS loop

The LOAD-MFS loop is characterised by an integrating transfer function model. The model is derived in Section 4.3.2 and given by

$$G_{\text{LOAD-MFS}}(s) = \frac{K_{LF}}{s} \tag{4.25}$$

$$G_{\text{LOAD-MFS}}(s) = \frac{0.01}{s} \tag{4.26}$$

that gives the following PI parameter values by using the rules of Table 4.1

$$K_c = 10000,\ \tau_i = 0.04,\ \tau_d = 0. \tag{4.27}$$

Table 4.2: Parameters of the three PI Controllers

Process	g(s)	k	τ_1	θ	K_c	τ_i	τ_D
PSE-CFF	$k\frac{e^{-\theta s}}{(\tau_1 s+1)}$	-0.00035	0.54	0.64	-1187	2.6	—
SLEV-SFW	$\frac{k}{s}$	0.42	—	—	238	0.04	—
LOAD-MFS	$\frac{k}{s}$	0.01	—	—	10000	0.04	—

4.5.3 PI controller for the SLEV-SFW loop

The SLEV-SFW loop is characterised by an integrating transfer function model. The model is derived in Section 4.3.3 and given by

$$G_{\text{SLEV-SFW}}(s) = \frac{K_{SW}}{s} \tag{4.28}$$

$$G_{\text{SLEV-SFW}}(s) = \frac{0.42}{s} \tag{4.29}$$

that gives the following PI parameter values by using the rules of Table 4.1

$$K_c = 238,\ \tau_i = 0.04,\ \tau_d = 0. \tag{4.30}$$

4.6 SUMMARY

A tuning method for PI control with anti-windup is provided in this chapter.

Linearised models are derived from the nonlinear model of Mintek by conducting step tests on the nonlinear model and fitting it to models with relevant forms.

PI controllers are designed for the linear models derived in Section 4.3 and some models are further simplified in Section 4.5 before obtaining the PI controller parameters from Table 4.1.

The PI controllers are applied to the nonlinear model presented in Section 2.3.4 and the results of the simulations are given in Chapter 5.1 for comparison to the simulations of the robust nonlinear model predictive controller presented in Section 3.7.

The three loops with their model and controller parameters are summarised in Table 4.2.

CHAPTER 5

MILLING CIRCUIT CONTROL SIMULATION STUDY

This chapter details simulation results of applying the RNMPC of Section 3.7, NMPC and single-loop PI controllers of Section 4.5 to the milling circuit model of Section 2.3.4. The objectives for milling control are presented in Section 1.2. The performance metrics are described before the main results of the simulation study are presented. A summary of the different simulation scenarios together with the performance values are provided at the end of this chapter.

5.1 INTRODUCTION

In this simulation study, RNMPC is compared to NMPC and single-loop PI controllers for different operational conditions. In all the simulations, severe parameter uncertainty is employed.

The simulations all assume that the process is operating under normal operating conditions before the disturbances are introduced at time $t = 0$. The normal operating condition of the milling circuit can be obtained from the circuit mathematical model known as the nominal operating point.

The nominal operating point of the model is obtained by applying the nominal parameters ($\tilde{\mathbf{p}}$) for the milling circuit to the model and calculating the state ($\mathbf{x_0}$) and input ($\mathbf{u_0}$) values that result in the rate-of-change of the states being zero ($\frac{\partial x_i}{\partial t} = 0, \forall i = 1,2,\ldots,n$ where n is the number of states). The system will therefore remain at the current operating condition (state) if the inputs remain constant and there is no external disturbance acting on it.

A constrained nonlinear optimisation is performed with regard to the states x and inputs u that lead to zero state variation $\frac{\partial x}{\partial t} = 0$ to obtain the nominal operating point. Constraints are defined such that the operating point will be physically relevant, e.g. there are no negative

feed-rates, power-draw or holdup of material in the mill or sump. Table 5.1 details the constraints (Min and Max), the calculated operating point value (OP) and the objective function weighting (W) for all the states, manipulated variables and controlled variables of the milling circuit model. If a variable is not included in the objective function, it is indicated by "—" in the weightings column (W), otherwise, if the variable is present in the objective function, but not penalised, it is indicated by "0" in the weightings column (W).

The milling circuit model contains large parameter uncertainties; this is especially true of the parameters related to the composition of the feed-ore and the hardness of the ore, which has an impact on the energy needed to grind a ton of ore. The parameter vector changes every 200 seconds, to allow the parameter disturbances to sufficiently impact the simulation. The aim of these relatively fast changes is to simulate the natural variation of the feed. The parameters follow a uniform distribution to produce large changes in the parameter values in order to properly demonstrate the disturbance rejection capabilities of the controller.

Feed ore hardness and composition changes are major disturbances that milling circuit controllers have to contend with, especially when the feed ore is switched between feeds that originate from different stockpiles. A feed ore hardness increase is simulated by increasing the power needed to produce a ton of fines (ϕ_f) by 50% in some of the simulation scenarios. A feed ore composition change is simulated by increasing the fraction of the feed consisting of rock (α_r) by 50% in some of the simulation scenarios. These disturbances are very large but not uncommon in practice.

The nominal, minimum and maximum values as well as the percentage variation of all the model parameters are detailed in Table 5.2. Figure 5.1 shows an example of typical parameter variations graphically, as employed in the simulations.

5.2 PERFORMANCE METRICS

The metrics in this section help quantify the performance of the controllers in terms of the economic objectives stated in Section 1.2 and the tracking performance of the LOAD. The two main economic objectives considered in this thesis are PSE tracking (objective 1b) and average throughput (objective 2). PSE setpoint tracking performance is calculated as

$$\text{PSE}_{\text{performance}} \triangleq \sum_{k=0}^{T_{sim}/\tau_s} \left(\text{PSE}(k) - \bar{\text{PSE}}(k)\right)^2 \tag{5.1}$$

where T_{sim} is the simulation time, τ_s is the sampling time, $\text{PSE}(k)$ is the product particle size at sample k and $\bar{\text{PSE}}(k)$ is the setpoint for particle size at sample k. Average throughput is calculated by

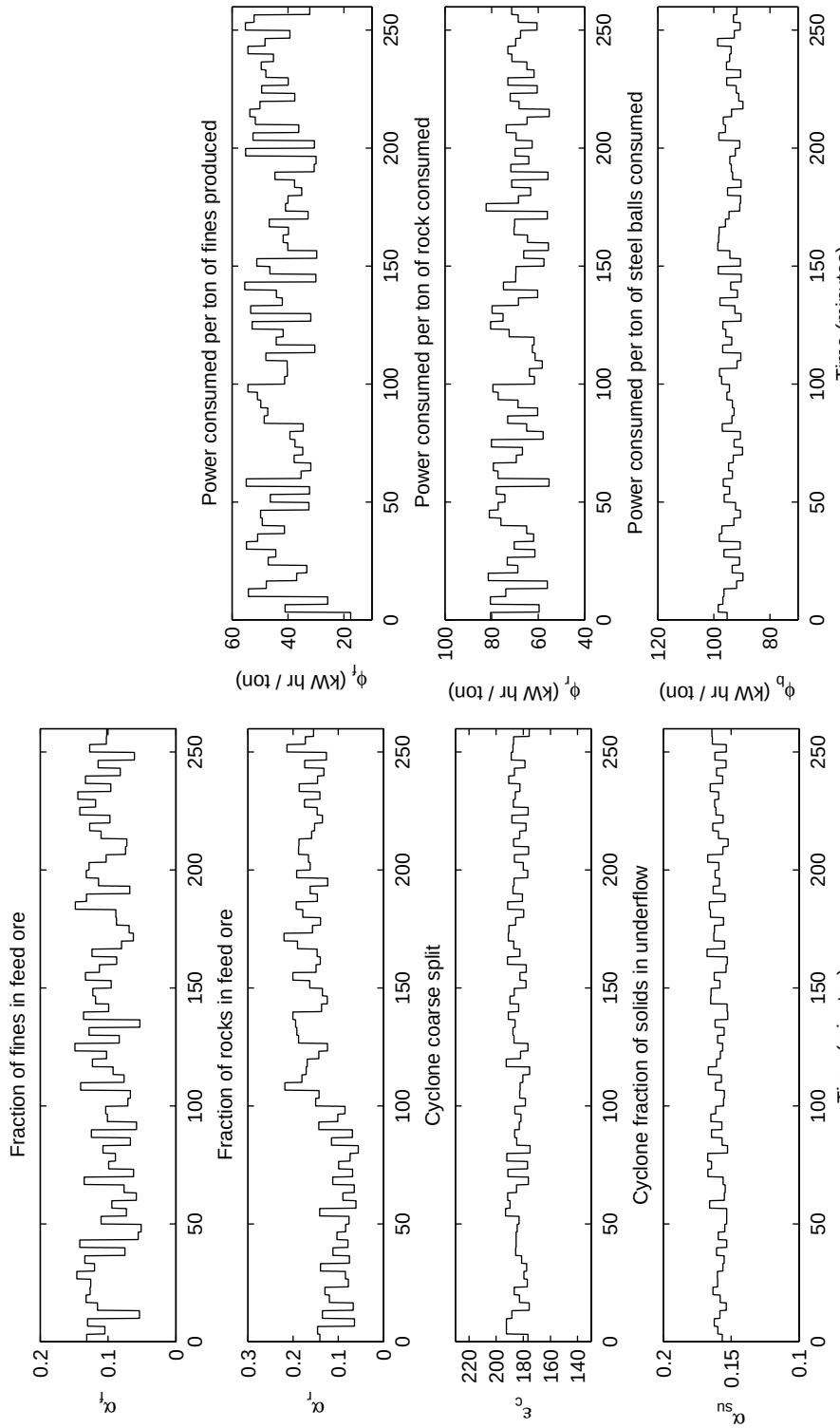
$$\text{THROUGHPUT}_{\text{average}} \triangleq \frac{1}{N} \sum_{k=0}^{N=T_{sim}/\tau_s} \text{THROUGHPUT}(k) \tag{5.2}$$

Table 5.1: Constraints (Min, Max, Δ), operating point (OP) and objective function weighting (W).

Variable	Min	Max	Δ	OP	W	Description
X_{mw}	0	50	—	8.53	—	The holdup of water in the mill. [m^3]
X_{ms}	0	50	—	9.47	—	The holdup of ore in the mill. [m^3]
X_{mf}	0	50	—	3.54	—	The holdup of fine ore in the mill. [m^3]
X_{mr}	0	50	—	20.25	—	The holdup of rocks in the mill. [m^3]
X_{mb}	0	20	—	6.75	—	The holdup of balls in the mill. [m^3]
X_{sw}	0	10	—	3.95	—	The holdup of water in the sump. [m^3]
X_{ss}	0	10	—	1.05	—	The holdup of ore in the sump. [m^3]
X_{sf}	0	10	—	0.14	—	The holdup of fine ore in the sump. [m^3]
MIW	0	100	10	33.33	0.01	The flow-rate of water to the circuit. [$m^3/hour$]
MFS	0	200	10	100	0.01	The flow-rate of ore to the circuit (consists of rocks, coarse and fine ore). [$tons/hour$]
MFB	0	4	1	2	0.01	The flow-rate of balls to the circuit. [$tons/hour$]
α_{speed}	0.7	1.0	—	0.82	—	The fraction of critical mill speed.
CFF	400	500	—	443	0.01	The flow-rate of water from the sump to the cyclone. [$m^3/hour$]
SFW	0	400	—	267	0.01	The flow-rate of extra water to the sump. [$m^3/hour$]
PSE	60	90	—	80	100	Product particle-size. [$\% < 75\mu m$]
LOAD	30	50	—	45	100	The total charge of the mill. [%]
SLEV	2	9.5	—	5.0	0	The level of the sump. [m^3]
Γ	0	1	—	0.51	0	Rheology Factor.
THROUGHPUT	100	0	—	200	1	Product throughput consisting of coarse and fine solids. [$tons/hour$]
P_{mill}	0	2000	—	2000	0	Power draw of the mill motor. [kW]

Table 5.2: Nominal, minimum and maximum parameter values for a closed-circuit ROM milling circuit.

Parm	Nom	Min	Max	%Δ	Description
α_f	0.1	0.05	0.15	50	Fraction of fines in the ore. [dimensionless]
α_r	0.1	0.05	0.15	50	Fraction of rocks in the ore. [dimensionless]
ϕ_f	28	14	42	50	Power per fines produced. [$^{\text{kW·hr}}/_{\text{ton}}$]
ϕ_r	69	55	83	20	Rock abrasion factor. [$^{\text{kW·hr}}/_{\text{ton}}$]
ϕ_b	94	89	99	5	Steel abrasion factor. [$^{\text{kW·hr}}/_{\text{ton}}$]
P_{max}	2000	—	—	—	Maximum mill motor power. [kW]
υ_{mill}	100	—	—	—	Mill volume. [m^3]
$\upsilon_{P_{\text{max}}}$	0.45	—	—	—	Fraction of mill volume filled for maximum power. [dimensionless]
$\Gamma_{P_{\text{max}}}$	0.51	—	—	—	Rheology factor for maximum mill power. [dimensionless]
ε_{ws}	0.6	—	—	—	Maximum water-to-solids volumetric ratio at zero pulp flow. [dimensionless]
V_V	40	—	—	—	Volumetric flow per "flowing volume" driving force. [hr^{-1}]
δ_{Pv}	1	—	—	—	Power-change parameter for volume. [dimensionless]
δ_{Ps}	1	—	—	—	Power-change parameter for fraction solids. [dimensionless]
α_P	0.82	—	—	—	Fractional power reduction per fractional reduction from maximum mill speed. [dimensionless]
α_{ϕ_f}	0.01	—	—	—	Fractional change in kW/fines produced per change in fractional filling of mill. [dimensionless]
χ_P	0	—	—	—	Cross-term for maximum power. [dimensionless]
ε_c	184	175	193	5	Cyclone coarse split. [dimensionless]
α_{su}	0.16	0.15	0.17	5	Fraction of solids in the underflow of the cyclone. [dimensionless]
C_1	0.6	—	—	—	Constant. [dimensionless]
C_2	0.7	—	—	—	Constant. [dimensionless]
C_3	3	—	—	—	Constant. [dimensionless]
C_4	3	—	—	—	Constant. [dimensionless]



(a) Feed ore composition and cyclone split characteristics.

(b) Ore and steel balls hardness.

Figure 5.1: Parameter variations for nonlinear model.

where N is the total number of samples in the simulation and THROUGHPUT(k) is the throughput of the milling circuit at sample k.

The LOAD setpoint tracking performance is calculated as

$$\text{LOAD}_{\text{tracking}} \triangleq \sum_{k=0}^{T_{sim}/\tau_s} \left(\text{LOAD}(k) - \bar{\text{LOAD}}(k)\right)^2 \tag{5.3}$$

where LOAD(k) is the volumetric filling of the mill at sample k and $\bar{\text{LOAD}}(k)$ is the setpoint for the volumetric filling of the mill at sample k, which is similar to the calculation for PSE tracking performance.

The stage cost of the objective function (3.95) used for the simulations in this chapter and Addendum B is given by

$$L_i(s_i, q_i) \triangleq \begin{bmatrix} \text{PSE} \\ \text{LOAD} \\ \text{SLEV} \\ \text{THROUGHPUT} \\ \text{Rheology} \\ \text{Mill Power} \end{bmatrix}^T Q \begin{bmatrix} \text{PSE} \\ \text{LOAD} \\ \text{SLEV} \\ \text{THROUGHPUT} \\ \text{Rheology} \\ \text{Mill Power} \end{bmatrix} + \begin{bmatrix} \Delta\text{CFF} \\ \Delta\text{MFS} \\ \Delta\text{SFW} \\ \Delta\text{MIW} \\ \Delta\text{Balls} \end{bmatrix}^T R \begin{bmatrix} \Delta\text{CFF} \\ \Delta\text{MFS} \\ \Delta\text{SFW} \\ \Delta\text{MIW} \\ \Delta\text{Balls} \end{bmatrix} \tag{5.4}$$

where Q and R are diagonal matrices and the diagonal entries are given in the "W" columns of Table 5.4. An example of a typical Q matrix is given by

$$Q = \begin{bmatrix} \mathbf{100} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{100} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{0} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{0} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{0} \end{bmatrix} \tag{5.5}$$

and a typical R matrix is given by

$$R = \begin{bmatrix} \mathbf{0.1} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{0.1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{0.1} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{0.1} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{0.1} \end{bmatrix} \tag{5.6}$$

where the bold entries are given in Table 5.4 for the different simulation scenarios.

The NMPC and RNMPC controllers allow any arbitrary form for the objective function and does not have to follow the form of equation (3.95). Alternative forms of the objective function can potentially express certain performance criteria more naturally. Alternative

forms of the objective function should, however, be verified not to affect the convergence and speed of the controller to the point where the controller becomes impractical.

5.3 SIMULATION RESULTS

This section details the simulation results of applying the RNMPC and NMPC of Section 3.7 and the single-loop PI controllers of Section 4.5 to the milling circuit model of Section 2.3.4.

5.3.1 Simulation parameters

This section outlines the sampling interval, prediction horizon and control horizon or nodes used by the both the RNMPC and NMPC controllers and the length of the simulations used by all the controllers, as shown in Table 5.3.

Process time constants for the dynamics that relate the MFS to LOAD and PSE are in the order of thirty minutes, whereas the time constants relating CFF and SFW to PSE are in the order of one or two minutes. Hence a sampling time of 10 seconds is recommended in Craig and MacLeod (1995). An additional motivation for this choice of sampling time is that during normal operation the sump volume is about 5 cubic metres and the flow rates of CFF and SFW range from 400 to 500 $^{\mathrm{m}^3}/_{\mathrm{hour}}$ and 0 to 400 $^{\mathrm{m}^3}/_{\mathrm{hour}}$ respectively. If, for example, the difference between CFF and SFW is 300 $^{\mathrm{m}^3}/_{\mathrm{hour}}$, the sump will run dry or overflow within about one minute.

The current implementation of the RNMPC and NMPC uses the same length of time for both the prediction and control horizons. It expresses the prediction and control horizons in multiples of the sampling time, which is 10 seconds for all the simulations. The prediction and control horizons are chosen to be 6 sampling intervals for all the simulations, thus 60 seconds. The number of nodes specifies the number of control vectors that is calculated over the control horizon. The number of nodes is also set to 6 for all the simulations, resulting in a control vector being calculated every 10 seconds over the control horizon. If, for example, the prediction and control horizons were set to 120 seconds and the number of nodes remained at 6, a control vector would have been calculated every 20 seconds over the length of the control horizon.

The weighting of the variables in Q and R of equation (3.96) is described by the "W" column in Table 5.1 and chosen based on the performance criteria of Section 1.2. Further, $P = Q$ with no terminal constraints ($\theta_N(s_N) \in \mathbb{R}^{N_x}$). The inputs are normalised according to their maximum range and outputs are normalised according to their setpoints in the objective function.

Figure 5.2 and Figure 5.3 show that the variables of interest, namely PSE, LOAD, SLEV, MFS and THROUGHPUT, reach steady-state within about 250 minutes. Other variables,

Table 5.3: Simulation Parameter Summary.

Variable	Value	Variable	Value
Prediction & Control Horizons (T)	60 seconds	Nodes (N)	6
Terminal Constraints ($\theta_N(s_N)$)	None ($\mathbb{R}^{N_x}$)	Simulation time	260 minutes
Terminal Cost Weighting (P)	Q	Sampling Time (τ_s)	10 seconds

such as MFB, take much longer to reach steady-state, but do not seem to have an impact on PSE and THROUGHPUT that form the basis of the economic performance criteria (Section 1.2). Therefore, the rest of the simulation will focus on the first 250 minutes to ensure that the disturbance rejection capabilities of the various controllers are clear with regard to PSE, LOAD, SLEV and THROUGHPUT.

5.3.2 Constant setpoint following and disturbance rejection

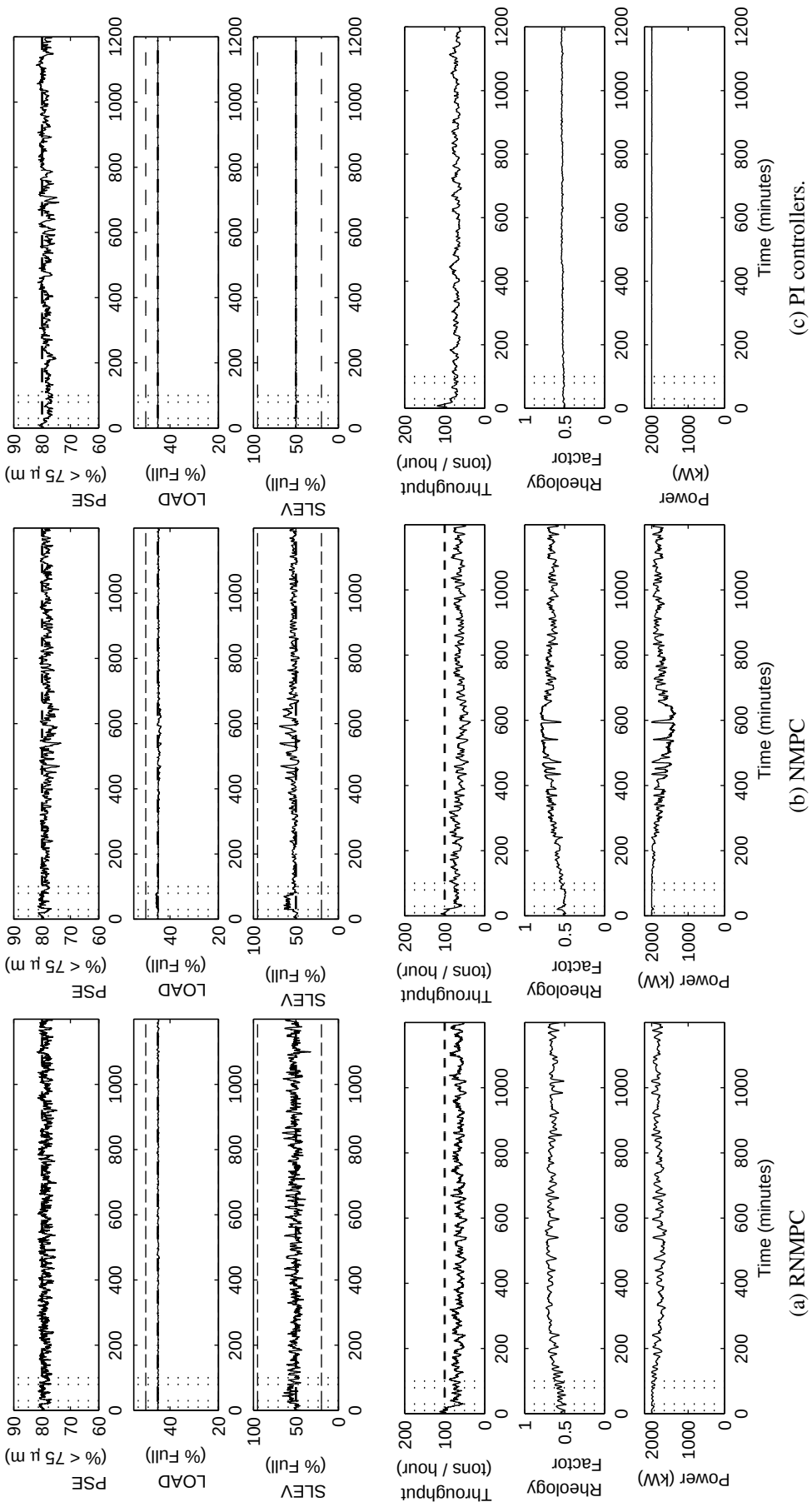
The most common operational mode for a milling circuit controller is to track a constant setpoint while rejecting external disturbances. The disturbances on the milling circuit can be quite severe, because the feed ore forms part of the grinding medium. Any changes to the size distribution or hardness of the ore will affect the throughput and grind of the milling circuit.

In this section the ability of the RNMPC, NMPC (shown in Section 3.7) and single-loop PI controllers (shown in Section 4.5) to follow constant setpoints of 80% < 75 μm for PSE and 45% for mill load volume (LOAD) in the face of disturbances is examined. LOAD is controlled strictly to prevent underload and overload conditions in the mill. The underload condition can result in steel balls and rocks hitting the liners of the mill directly, severely increasing the wear of the liners. Overload and underload conditions cause a drop in milling efficiency and affect the grind.

The simulation results illustrated in Figure 5.4a, Figure 5.5a, Figure 5.4b and Figure 5.5b show the RNMPC tracking the constant setpoints on PSE and LOAD without any step disturbances. Zero mean parameter variations are, however, present as summarised in Table 5.2. The simulation results for NMPC are discussed in Section B.1.1.

The simulation scenario shown in Figure 5.4a and Figure 5.5a allows SLEV to vary freely with only upper and lower constraints enforced. Steering SLEV to setpoint compared to allowing SLEV to vary freely does not significantly influence the closed-loop performance under RNMPC in terms of PSE setpoint tracking and the average circuit throughput (Table 5.4). Allowing SLEV to vary within bounds allows the RNMPC to change the density of the slurry inside the sump (assuming fully mixed conditions) and as a result allows the RNMPC to control the feed density to the cyclone. Control of the feed density to the cyclone can increase the control envelope of the RNMPC.

The results of the milling circuit under PI control are illustrated in Figure 5.4c and Fig-



(a) RNMPC (b) NMPC (c) PI controllers.

Figure 5.2: CVs of RNMPC (a), NMPC (b) and PI controllers (c). Feed ore hardness and composition step disturbances are present as well as a sump feed water disturbance simulating spillage water being added to the sump. This simulation is performed for 1200 minutes to determine the time to steady state. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

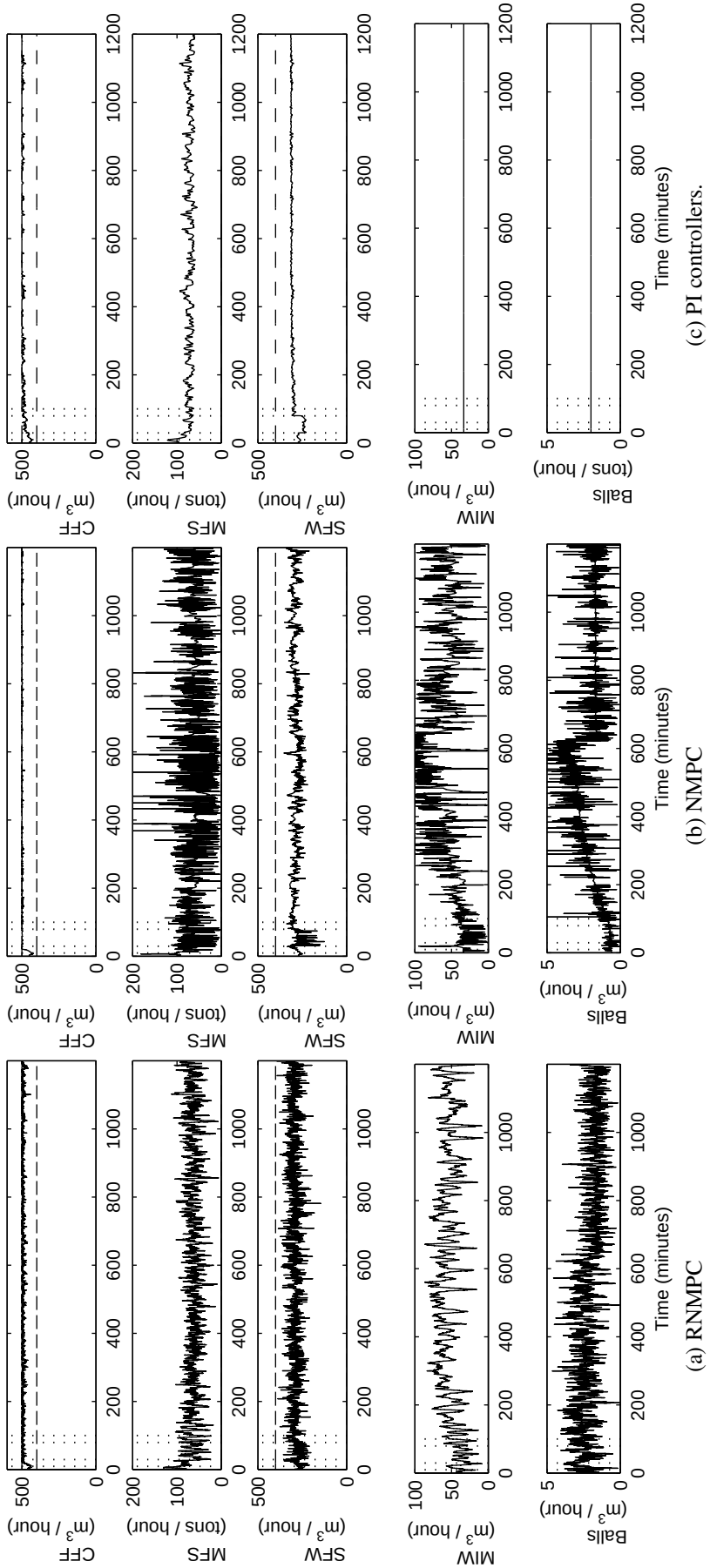


Figure 5.3: MVs of RNMPC (a), NMPC (b) and PI controllers (c).
Feed ore hardness and composition step disturbances are present as well as a sump feed water disturbance simulating spillage water being added to the sump. This simulation is performed for 1200 minutes to determine the time to steady state. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

ure 5.5c. Table 5.4 shows that the PSE tracking of the PI controllers is not as tight as the RNMPC at a comparable average throughput.

Other simulations investigating the effects of various disturbances will be compared against the baseline simulations presented in Figure 5.4 and Figure 5.5.

A 50% increase in feed ore hardness is introduced at time 180 minutes. The RNMPC (Figure 5.6a and Figure 5.7a) and NMPC (Figure 5.6b and Figure 5.7b) follow the PSE and LOAD setpoints very tightly despite the increase in feed ore hardness, as seen in Table 5.4. The PI controllers (Figure 5.6c and Figure 5.7c), however, show a drop in PSE when the ore hardness disturbance is introduced and this is reflected in a higher PSE error, as seen in Table 5.4.

The controller decreases MFS, causing the average throughput of the milling circuit to decrease as the ore hardness increases, because the harder ore needs more time inside the mill to grind down. The controller does not decrease MFS enough to maintain the grind of the mill, which causes the ratio of coarse to fine material in the sump to increase. The controller increases the CFF to compensate for the coarser grind in order to maintain PSE at setpoint, because higher pressure at the inlet of the cyclone results in finer material exiting at the overflow of the cyclone, called a finer cut, while lower pressure at the cyclone inlet results in a coarser cut. The cyclone feed contains more coarse material and a finer cut is therefore required to maintain PSE at setpoint. The increase in CFF also causes an increase in the recirculating load of the circuit due to the coarser grind of the mill. The controller allows the grind to become coarser and then compensates by increasing CFF in order to minimise the impact of the harder ore on the throughput while maintaining PSE at setpoint.

The amount of rocks in the feed ore is increased by 50% at time 180 minutes to simulate a feed disturbance. The RNMPC (Figure 5.8a and Figure 5.9a) and NMPC (Figure 5.8b and Figure 5.9b) follow the PSE and LOAD setpoints very tightly despite the increase of rocks in the feed ore as seen in Table 5.4. The PI controllers (Figure 5.8c and Figure 5.9c), however, show a slight increase in PSE as the increased amount of rocks inside the mill results in a finer grind. There is no immediate impact on the performance of the mill, but there is a build-up of rocks inside the mill that will eventually result in a decrease in throughput, because it takes longer to grind down the rocks inside the mill. Figure 5.10 and Figure 5.11 show the result of the milling circuit under RNMPC (a), NMPC (b) and PI control (c) when the feed disturbance is introduced at time 10 minutes. The build-up of rocks inside the mill produces more fines and forces the controller to decrease CFF in order to maintain PSE.

A 50 $m^3/hour$ increase in SFW is introduced at time 180 minutes to simulate spillage pumping. The RNMPC (Figures 5.12a and Figure 5.13a) follows the PSE and LOAD setpoints more tightly compared to the NMPC (Figures 5.12b and Figure 5.13b) and PI controller (Figures 5.12c and Figure 5.13c), as seen in Table 5.4, but SLEV increases slightly under both RNMPC and NMPC owing to the disturbance. The sump is assumed to be fully mixed and the increase in water lowers the slurry density inside the sump, resulting in a finer cut at

the cyclone overflow, which forces the controller to reduce CFF in order to maintain PSE at its setpoint.

In this scenario the feed ore hardness increases at time 10 minutes, SFW increases between 30 minutes and 80 minutes and the percentage of rocks in the feed increases at time 100 minutes. These disturbances are present in all subsequent simulations. The RNMPC (Figure 5.14a and Figure 5.15a) and NMPC (Figure 5.14b and Figure 5.15b) follow the PSE and LOAD setpoints very tightly despite all the disturbance working on the system. The PI controllers (Figure 5.14c and Figure 5.15c), however, show a larger PSE tracking error compared to the RNMPC and NMPC with all the disturbances present. Comparing Figure 5.15a and Figure 5.15b, it is clear that the NMPC is more aggressive in its control action compared to the RNMPC. The RNMPC follows the PSE setpoint more closely than the NMPC, as seen in Table 5.4. Figure 5.14 shows that all the controllers operate at the upper constraint of CFF, which suggests that the RNMPC and NMPC controllers use other variables in conjunction with CFF to maintain PSE at its setpoint.

5.3.3 Reduced PSE setpoint to 75% and 70% < 75μm

The increased hardness of the feed ore caused the average throughput of the circuit to reduce, because the ore needs more time to grind down. There is a well-established inverse relationship between PSE and throughput (Craig and MacLeod, 1995). The PSE setpoint is reduced in order to increase the milling circuit throughput. The simulations in this scenario investigate what effect a reduction of 5% and 10% in PSE setpoint has on the average throughput of the system.

Figure 5.16b and Figure 5.17b show that the RNMPC tracks the reduced PSE setpoint of 75% < 75μm and LOAD setpoint of 45% volumetric filling well. The throughput shows large variations due primarily to the ore hardness and composition variations. Decreasing the setpoint for PSE to 75% increased the average throughput of the milling circuit to 74.5 from 72.2 tons per hour. The PI controllers (Figure 5.18b and Figure 5.19b) show significantly better tracking of the PSE setpoint as well as increased throughput. The improved PSE tracking can primarily be attributed to the controller maintaining CFF within its constraints, because CFF is the only manipulated variable available to the PI controllers for controlling PSE.

Figure 5.16c and Figure 5.17c show that the RNMPC tracks the reduced PSE setpoint of 70% < 75μm and LOAD setpoint well. Decreasing the PSE setpoint to 70% resulted in an average throughput of 81.3 tons per hour, as seen in Table 5.4. The ability of the PI controllers (Figure 5.18c and Figure 5.19c) to track the PSE setpoint of 70% compared to 75% is significantly poorer. The poorer PSE tracking can primarily be attributed to CFF operating at its lower limit and reducing the ability of the PI controllers to control PSE.

Decreasing the PSE setpoint by 10% could not compensate for the 50% increase in ore

hardness. Maintaining the throughput at 100 tons per hour would probably require the PSE setpoint to be lowered to an unacceptably low value. The corresponding results for NMPC are discussed in Section B.1.2.

5.3.4 Step change of -5% and -10% in PSE setpoint

Changes in setpoints occur as the result of changing production targets and changing milling conditions. In the previous section, the increased ore hardness required that the PSE setpoint be reduced in order to maintain the circuit throughput at the desired level. In this section, the ability of the controller to track a modest and large setpoint change in PSE is investigated.

A -5% step change in PSE setpoint is introduced at time 100 minutes, while maintaining a constant setpoint on LOAD and SLEV. The RNMPC (Figure 5.20a and Figure 5.21a) and NMPC (Figure 5.20b and Figure 5.21b) follow the setpoint change well without affecting LOAD. The decrease in particle size setpoint increased throughput by a small margin from 72.2 to 74.0 tons per hour. The PI controllers (Figure 5.20c and Figure 5.21c) start to drift below the PSE setpoint but track the reduced setpoint after the step change very well, because the controller can maintain CFF within its limits.

A larger step change of -10% in the PSE setpoint is introduced at time 100 minutes, while maintaining a constant setpoint on LOAD and SLEV. The RNMPC (Figure 5.20a and Figure 5.21a) and NMPC (Figure 5.20b and Figure 5.21b) follow the larger setpoint change well without affecting LOAD. The decrease in PSE setpoint increased throughput by a bigger margin from 72.2 to 76.7 tons per hour. The PI controllers (Figure 5.20c and Figure 5.21c) start to drift below the PSE setpoint when the ore hardness disturbance is introduced. The PI controllers cannot follow the lower PSE after the setpoint step change, because CFF reaches its lower limit, reducing the ability of the PI controllers to control PSE. The RNMPC and NMPC increase the mill rheology by primarily increasing MIW, which will reduce the density of the slurry in the sump as well as the grinding efficiency of the mill, resulting in a decrease in PSE while CFF is constrained at its lower limit.

The RNMPC, NMPC and PI controllers showed a decoupled response by successfully changing PSE without any significant impact on LOAD and SLEV. The PI controllers were not able to handle the big PSE setpoint change as well as the RNMPC and NMPC controllers, because the PI controllers can only use CFF to control PSE, while the RNMPC and NMPC can use other variables, such as MIW and SFW, to control PSE.

5.3.5 Regulate PSE, LOAD and Throughput

The previous two sections show the dependence of throughput on PSE and ore hardness. Throughput is added to the objective function of the controller in order to determine if the

optimising capability of the controller can be exploited to increase the average throughput of the circuit while maintaining PSE at the desired setpoint when the ore hardness increases.

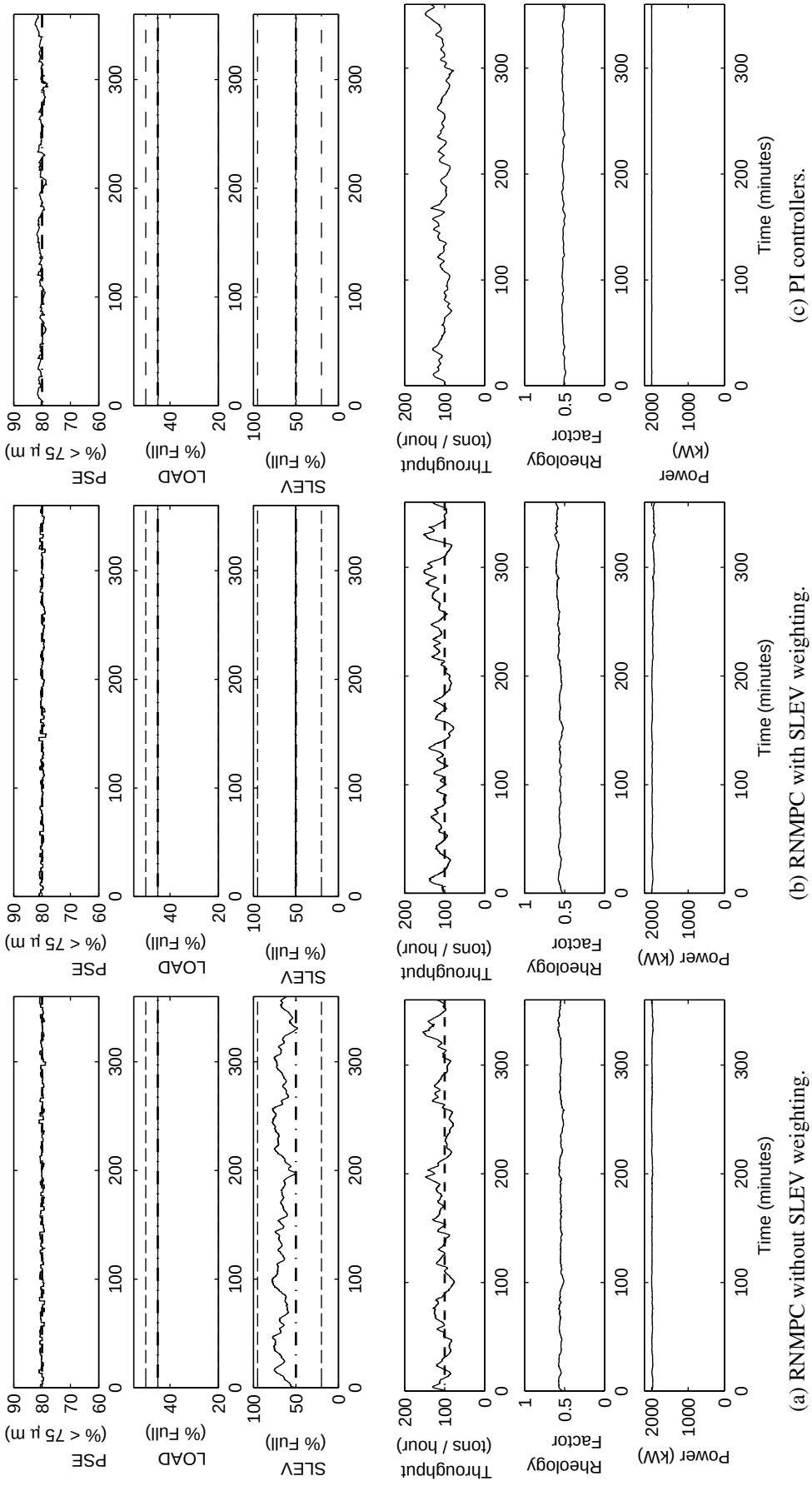
Figure 5.24b and Figure 5.25b show that adding the throughput to the objective function causes the RNMPC to make a trade-off between following the PSE setpoint and throughput setpoint according to their respective weightings. Table 5.4 shows that the PSE error increases significantly and that the average throughput decreases from 72.2 to 70.8 tons per hour, resulting in an overall worst result. Figure 5.25b shows large variation in the manipulated variables that can be the cause of the poor overall performance.

Figure 5.24c and Figure 5.25c show that increasing the throughput weighting causes a larger error in the PSE tracking performance while further reducing the average throughput as seen in Table 5.4.

The unexpected reduction in throughput can be attributed to the large variations in the manipulated variables as observed in Figure 5.25c. The large variations may be the result of the gain of the controller being too high. The gain of the controller is controlled by the weighting on the manipulated and controlled variables and more importantly the ratio of the weights between the controlled and manipulated variables. The contribution of the controlled variables to the objective function compared to the manipulated variables is increased by adding throughput to the objective function, effectively reducing the weighting of the manipulated variables and increasing the gain of the controller.

Figure 5.26c and Figure 5.27c show that oscillations are eliminated from the control action by increasing the weighting on the manipulated variables. The PSE tracking error is reduced and the average throughput increased compared to the scenario shown in Figure 5.24b and Figure 5.25b. The PSE tracking is worse but the average throughput is higher than the case where throughput is not included in the performance function (Figure 5.14a and Figure 5.15a), as seen in Table 5.4.

The results in this section show that the controller cannot overcome the inherent trade-off between PSE and throughput, because the controller lowers PSE in order to increase throughput. The corresponding results for NMPC are discussed in Section B.1.3.



(a) RNMPC without SLEV weighting. (b) RNMPC with SLEV weighting. (c) PI controllers.

Figure 5.4: CVs of RNMPC (a) & (b) and PI controllers (c). There is no step disturbance present in either the feed ore hardness or the fraction of rocks in the feed ore. This represents the nominal scenario with only zero mean parameter variations present. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

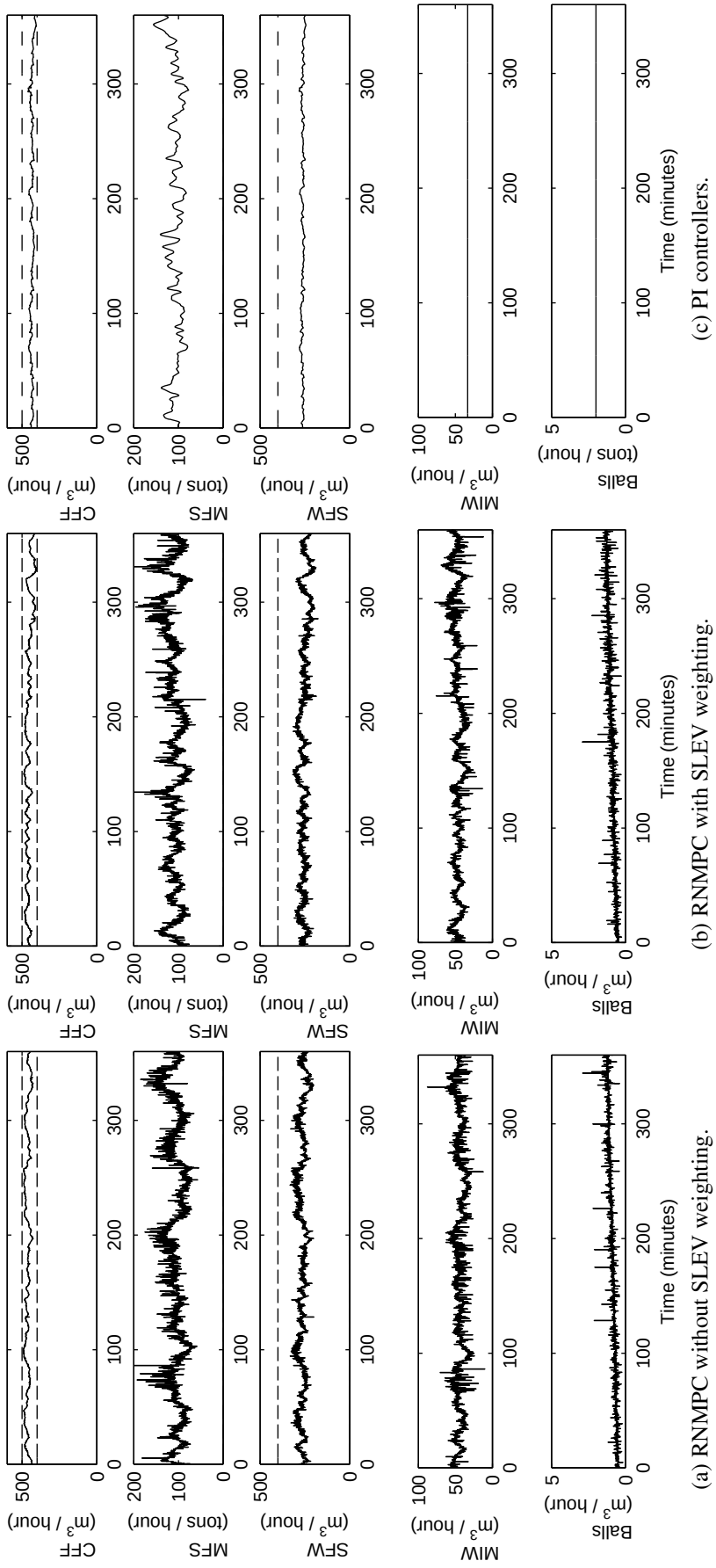


Figure 5.5: MVs of RNMPC (a) & (b) and PI controllers (c).
There is no step disturbance present in either the feed ore hardness or the fraction of rocks in the feed ore. This represents the nominal scenario with only zero mean parameter variations present. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

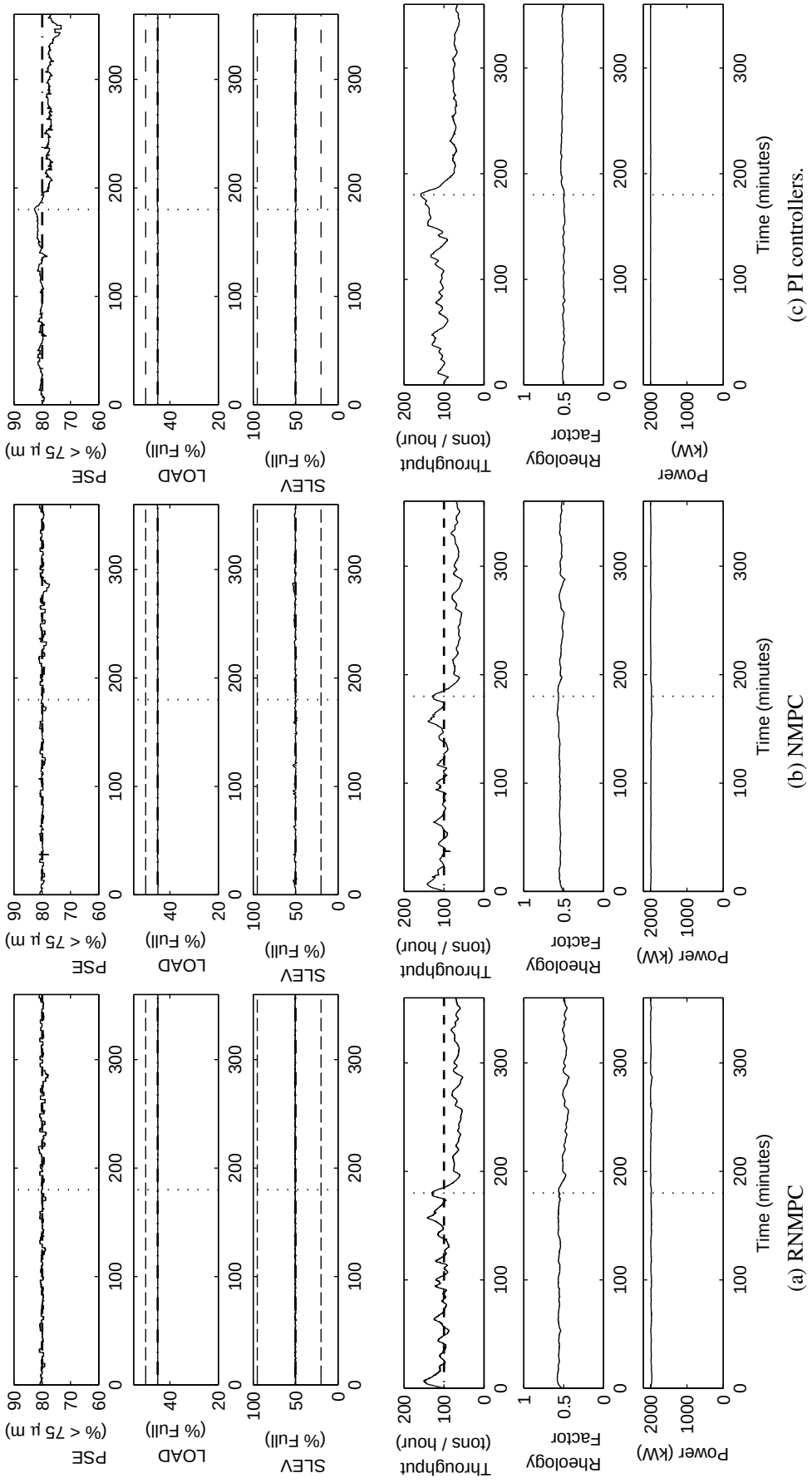


Figure 5.6: CVs RNMPC (a), NMPC (b) and PI (c) controllers.
A 50% increase in ore hardness is introduced at time 180 minutes. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

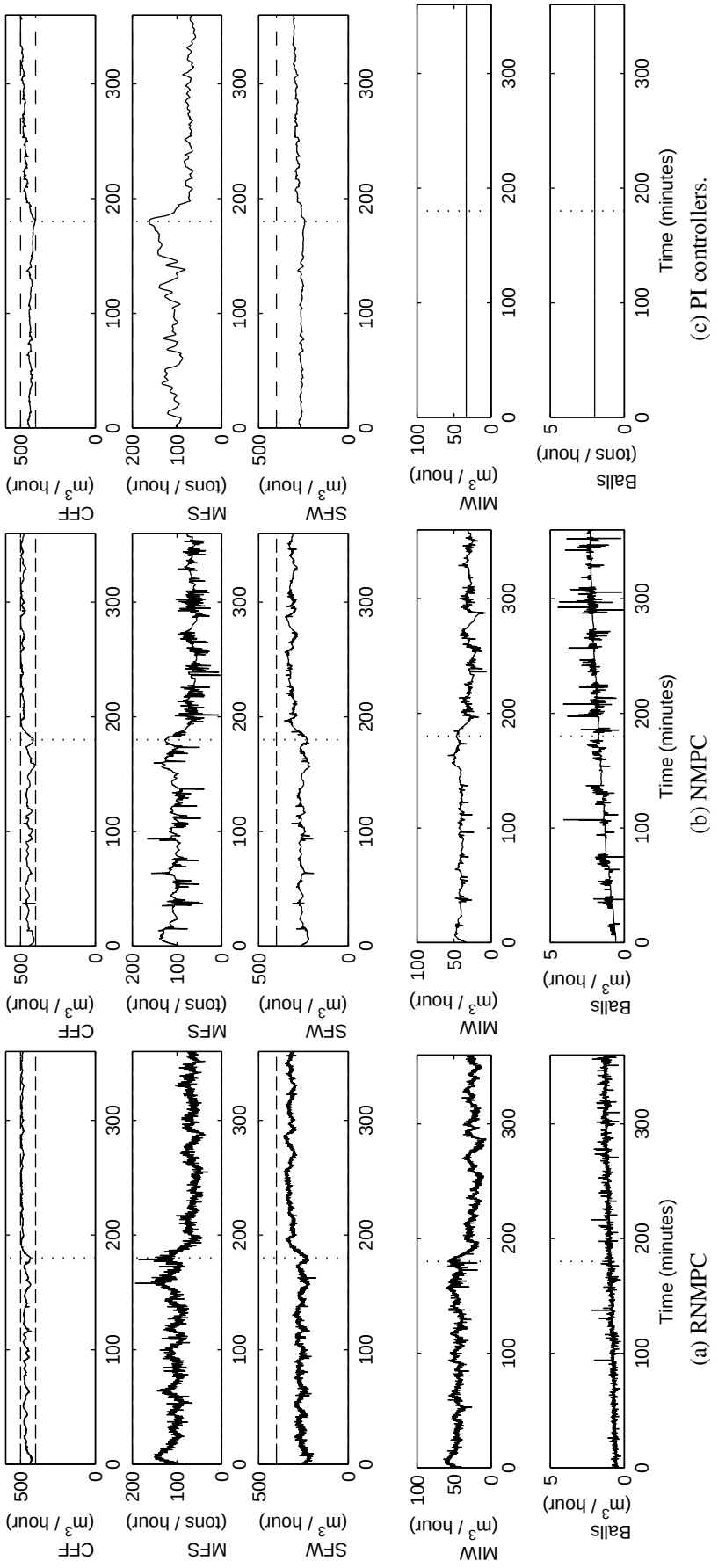
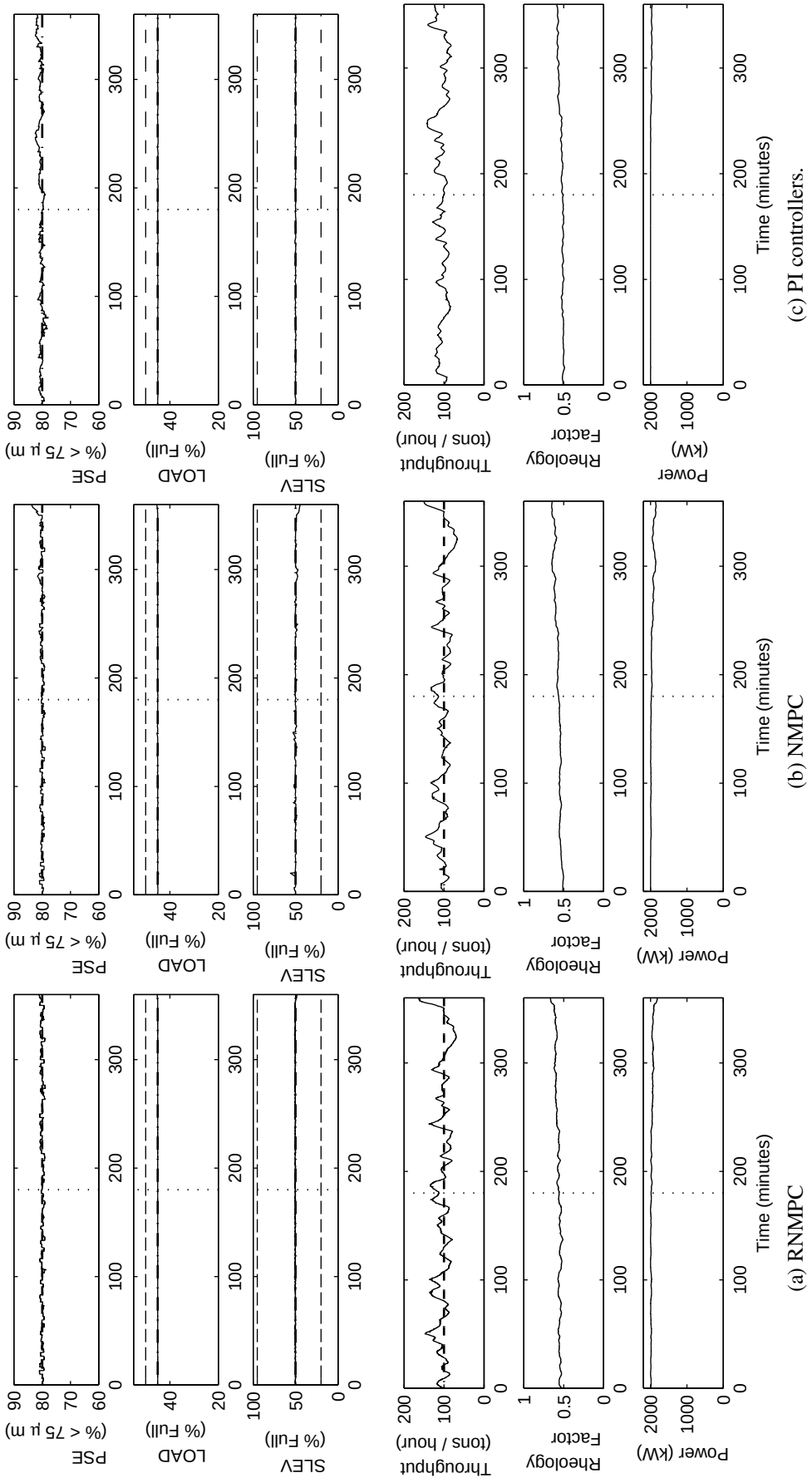


Figure 5.7: MVs RNMPC (a), NMPC (b) and PI (c) controllers.
A 50% increase in ore hardness is introduced at time 180 minutes. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) RNMPC (b) NMPC (c) PI controllers.

Figure 5.8: CVs RNMPC (a), NMPC (b) and PI (c) controllers.
A 50% increase in the fraction of rocks in the feed ore is introduced at time 180 minutes. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

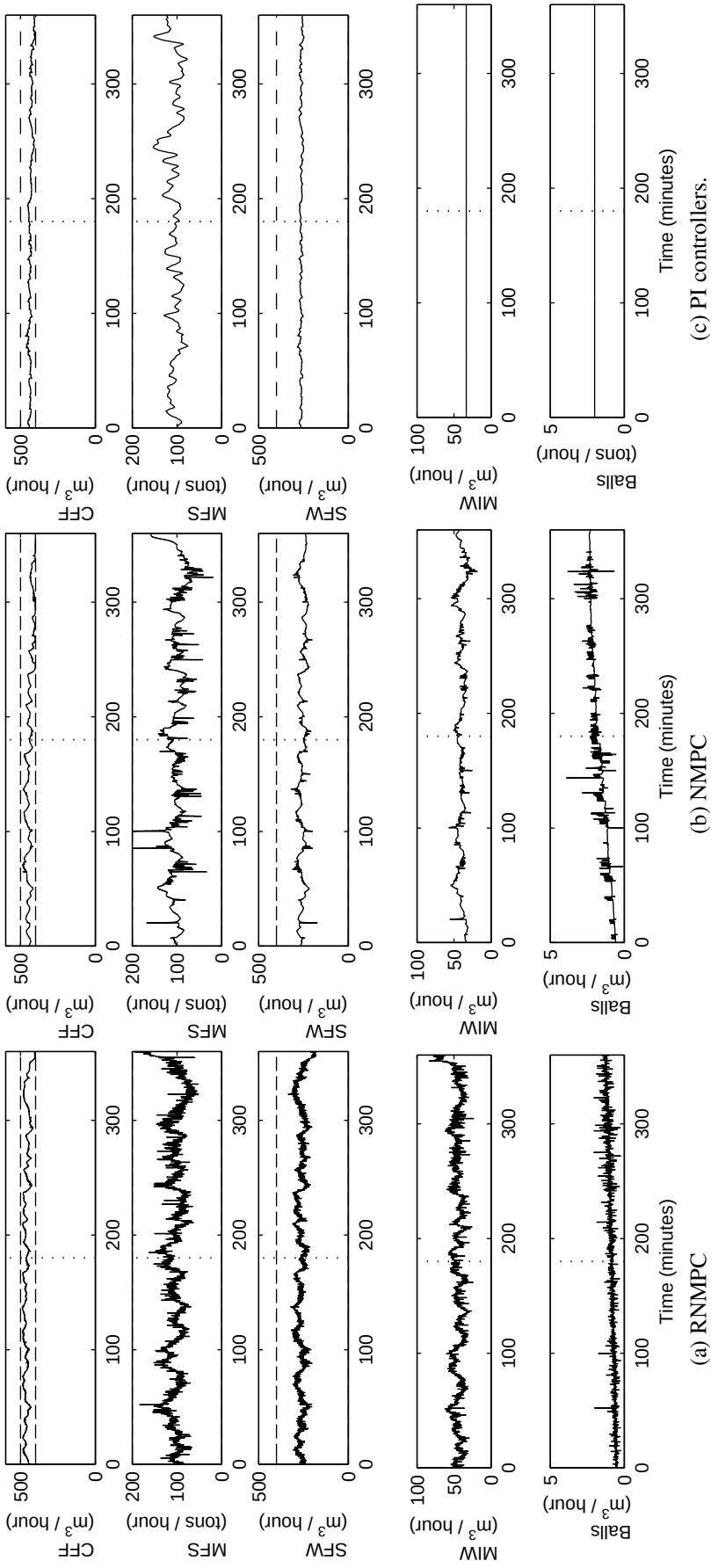


Figure 5.9: MVs RNMPC (a), NMPC (b) and PI (c) controllers.
A 50% increase in the fraction of rocks in the feed ore is introduced at time 180 minutes. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

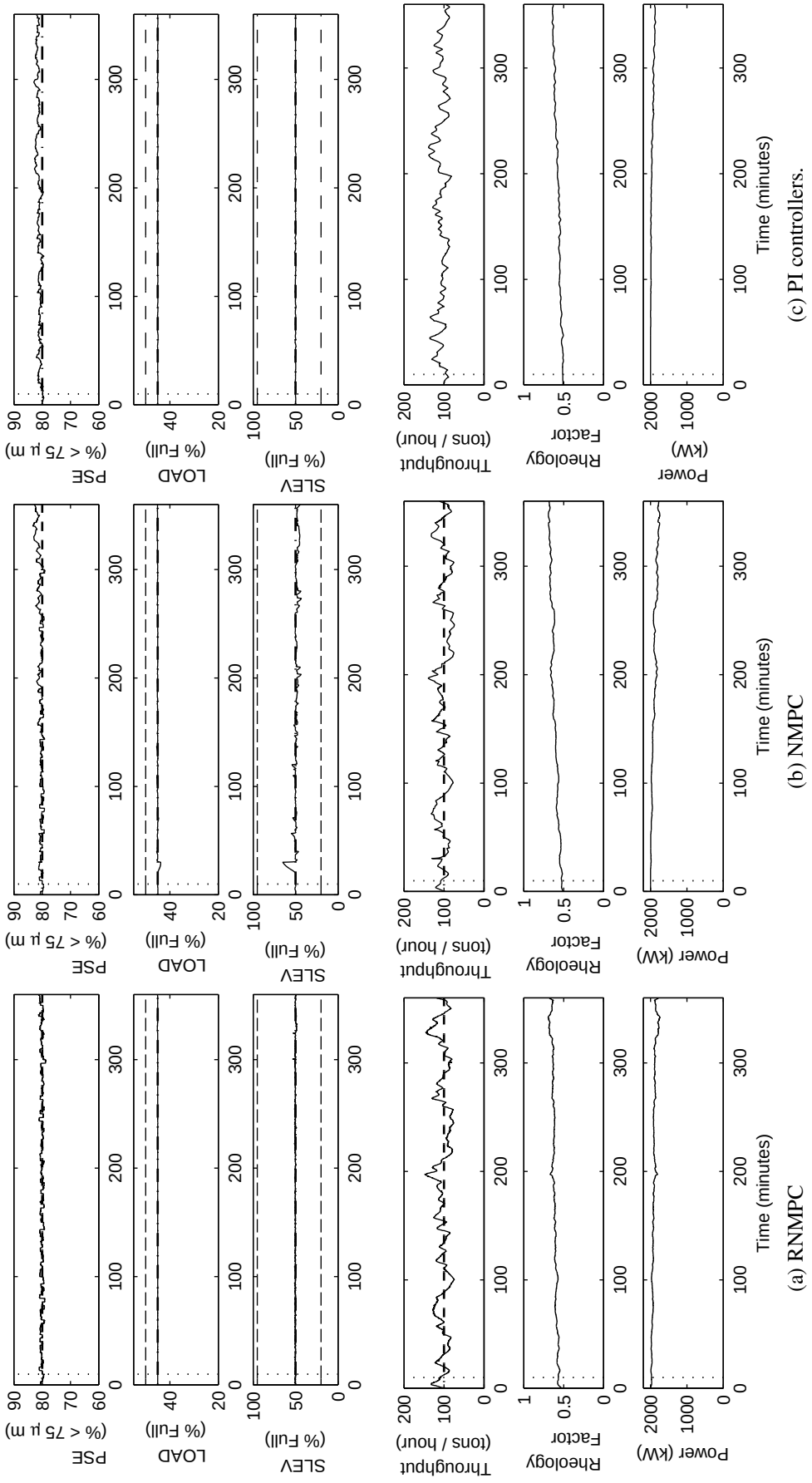


Figure 5.10: CVs RNMPC (a), NMPC (b) and PI (c) controllers.
A 50% increase in the fraction of rocks in the feed ore is introduced at time 10 minutes. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

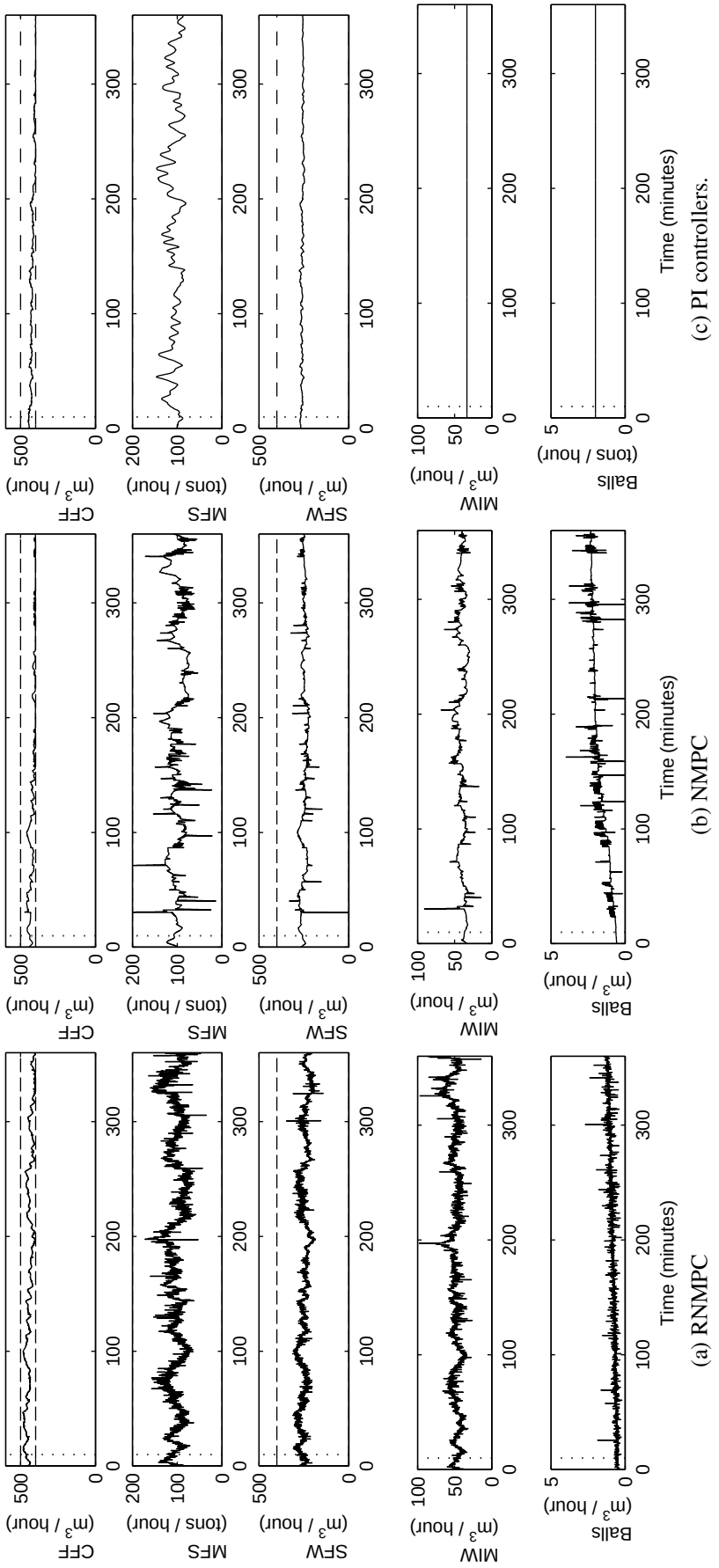
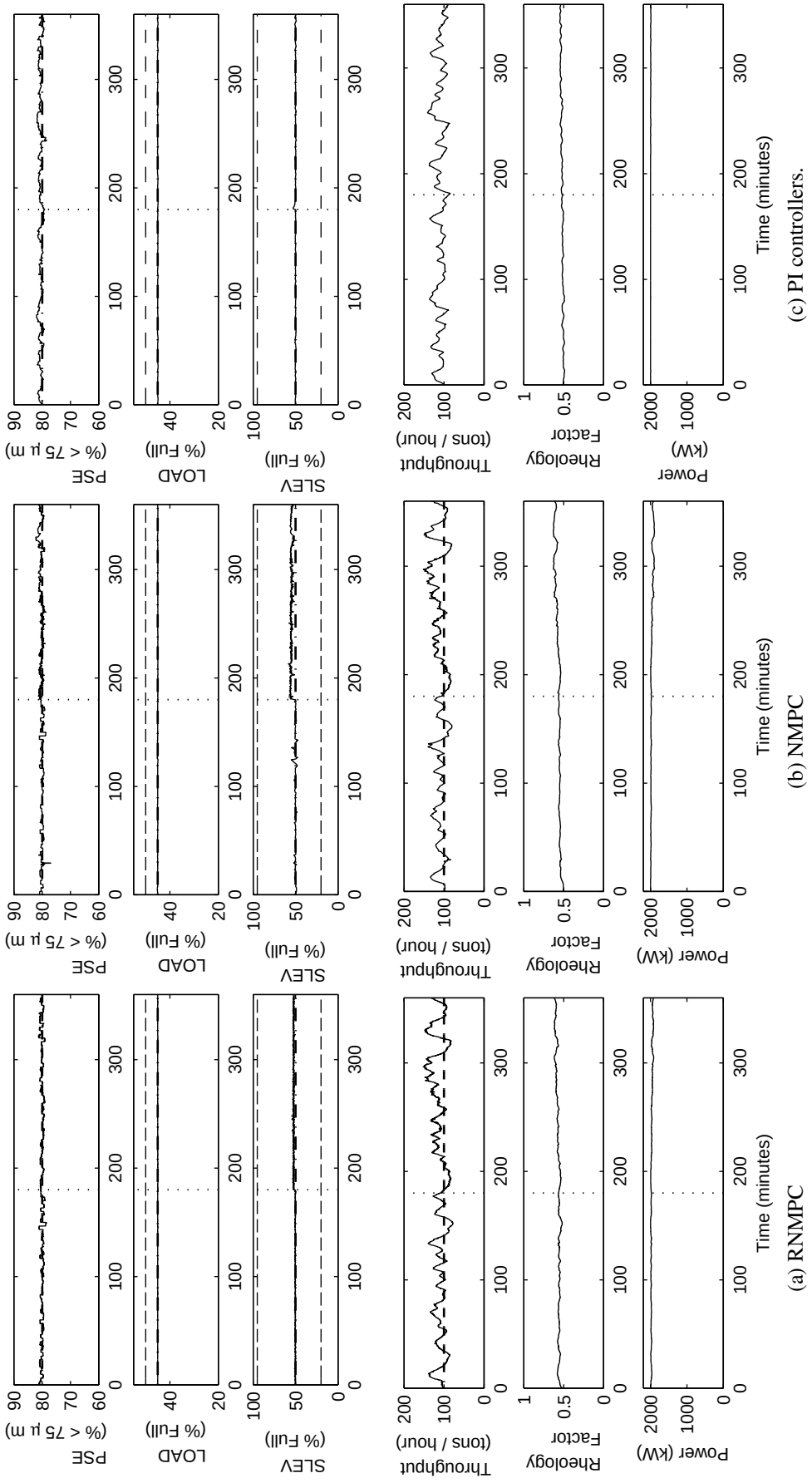


Figure 5.11: MVs RNMPC (a), NMPC (b) and PI (c) controllers.
A 50% increase in the fraction of rocks in the feed ore is introduced at time 10 minutes. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) RNMPC (b) NMPC (c) PI controllers.

Figure 5.12: CVs RNMPC (a), NMPC (b) and PI (c) controllers.
A 50 $m^3/hour$ increase in SFW is introduced at time 180 minutes. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

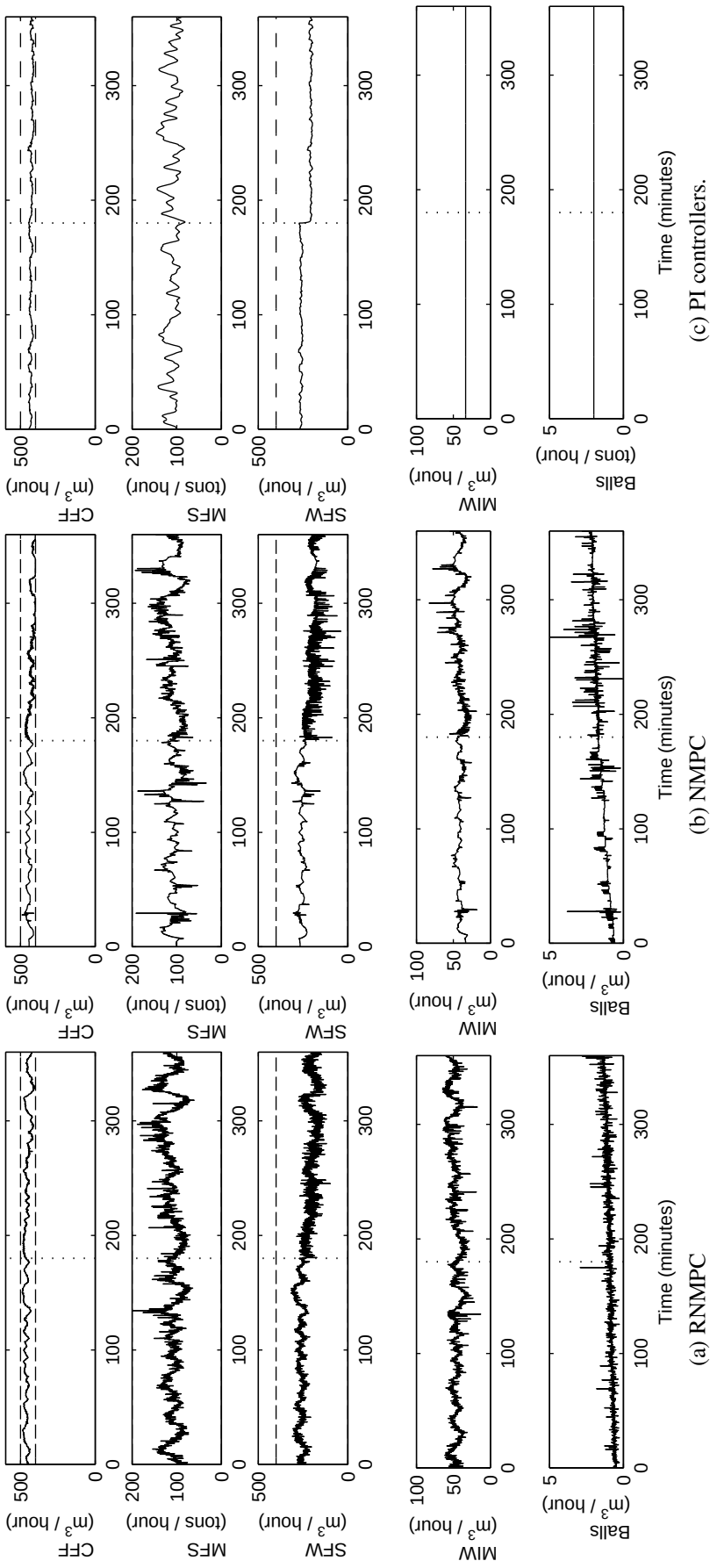
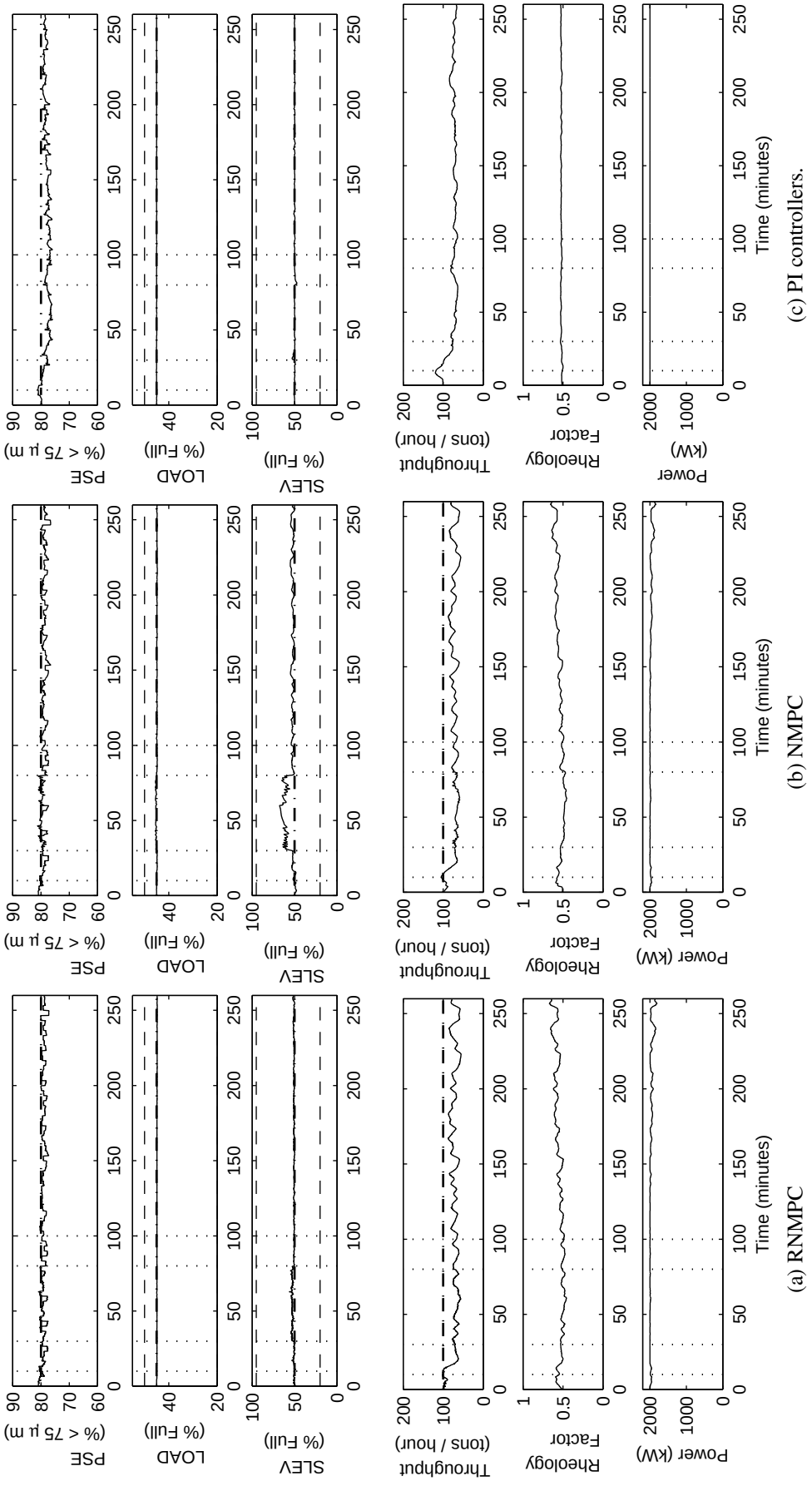


Figure 5.13: MVs RNMPC (a), NMPC (b) and PI (c) controllers.
A 50 $m^3/hour$ increase in SFW is introduced at time 180 minutes. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) RNMPC (b) NMPC (c) PI controllers.

Figure 5.14: CVs RNMPC (a), NMPC (b) and PI (c) controllers.
This represents the nominal disturbance scenario. The controller regulates PSE, LOAD and SLEV at constant setpoints with step disturbances: (1) 50% increase in rock hardness at time 10 minutes; (2) increase of 50 $m^3/hour$ in SFW from time 30 minutes until time 80 minutes; (3) 50% increase in the fraction of rocks in the feed ore at time 100 minutes. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

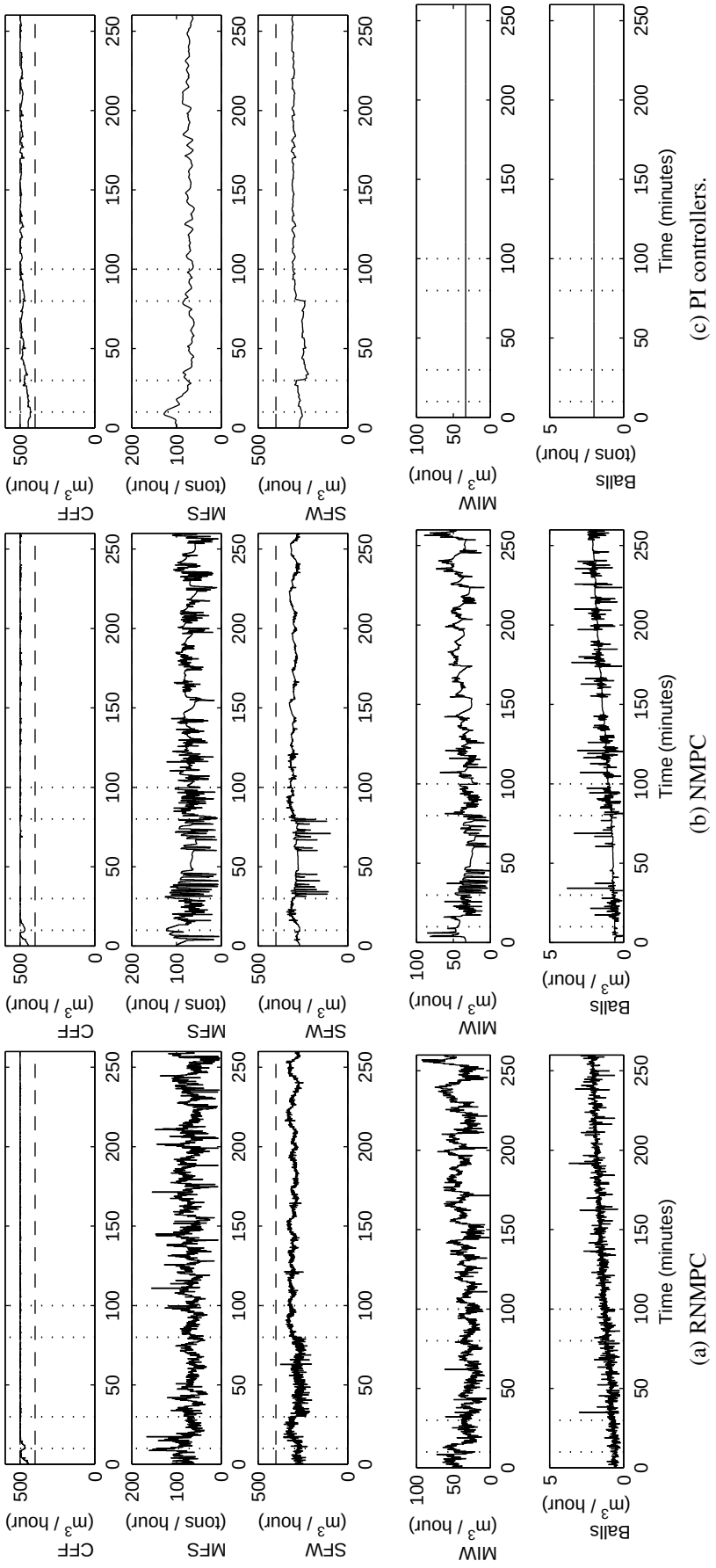
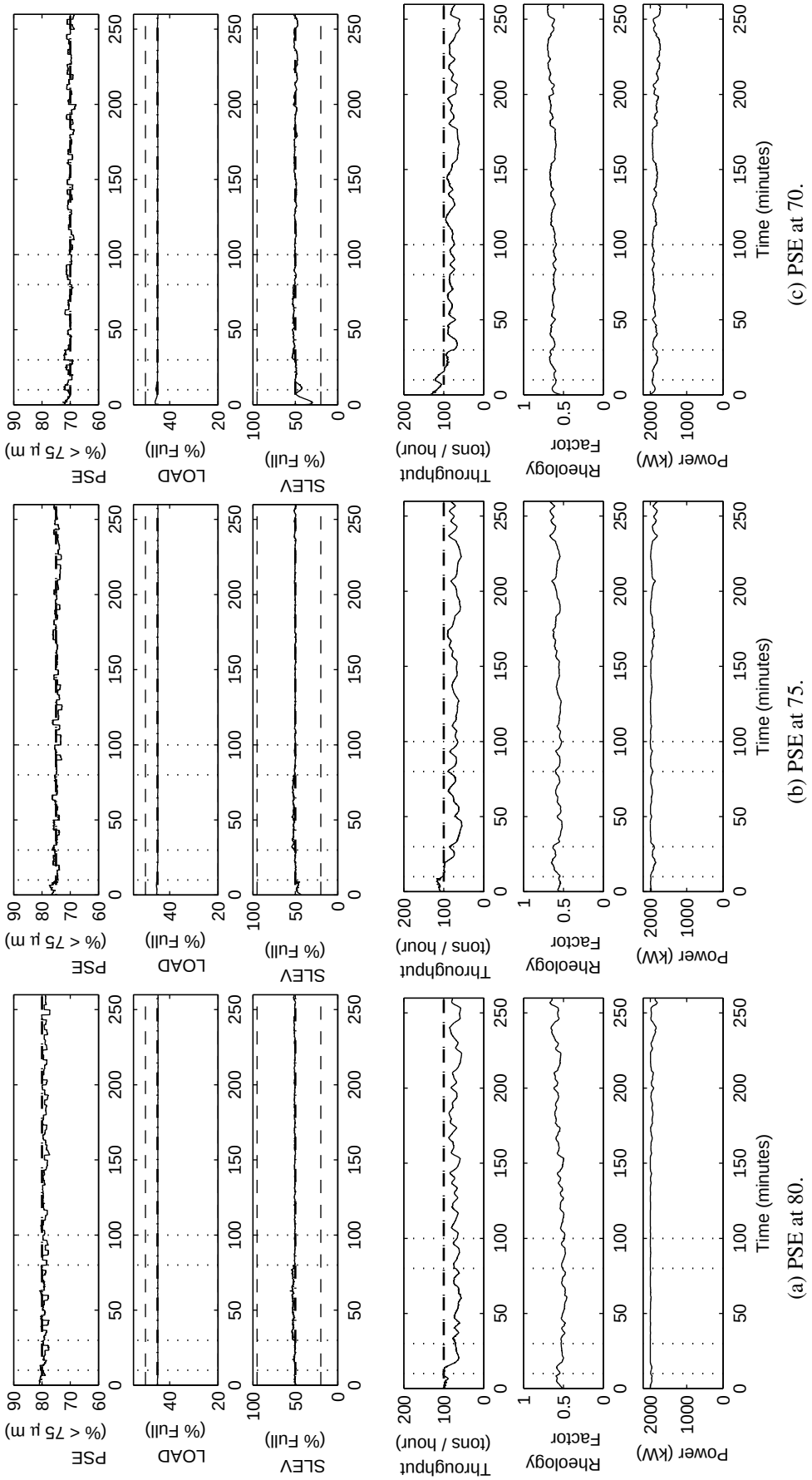
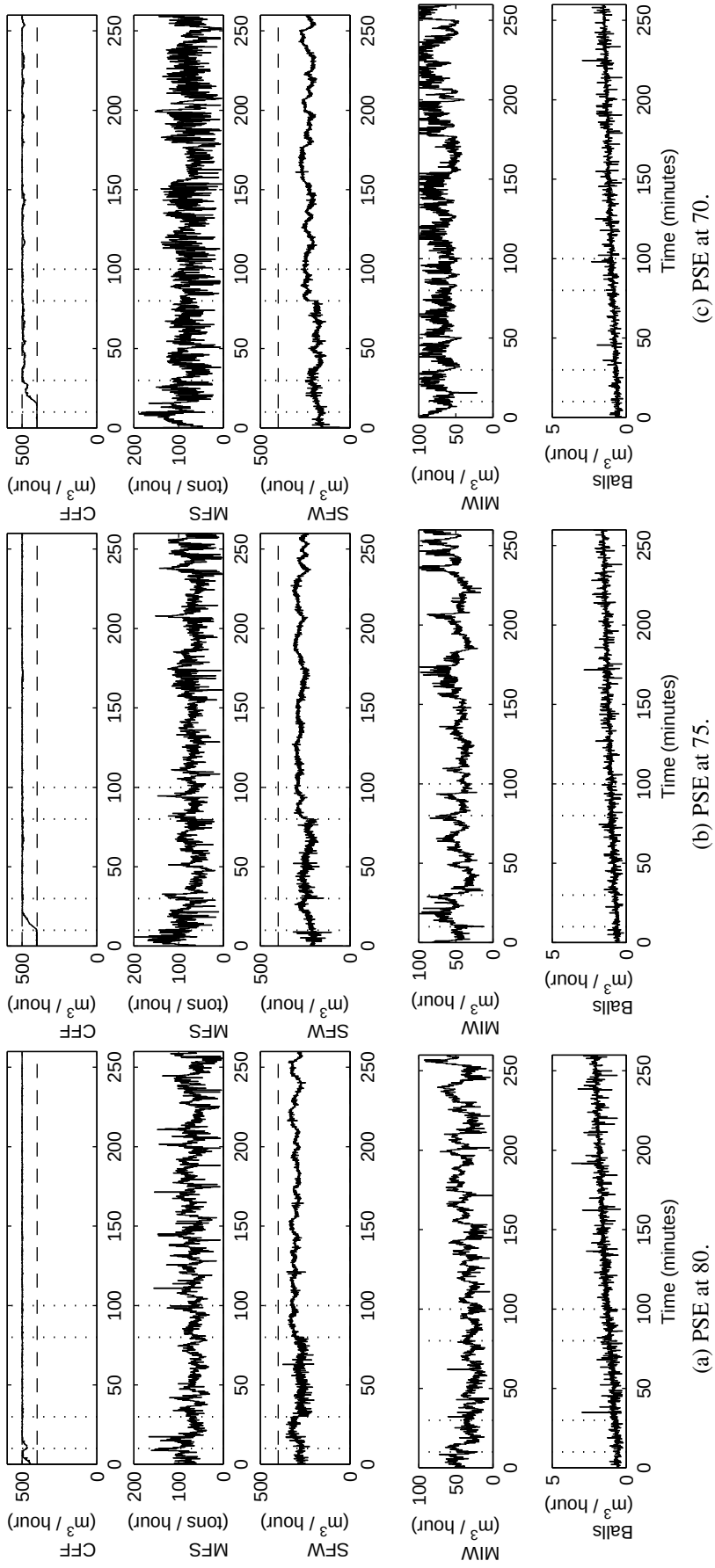


Figure 5.15: MVs RNMPC (a), NMPC (b) and PI (c) controllers.
This represents the disturbance scenario. The controller regulates PSE, LOAD and SLEV at constant setpoints with step disturbances: (1) 50% increase in rock hardness at time 10 minutes; (2) increase of 50 $m^3/hour$ in SFW from time 30 minutes until time 80 minutes; (3) 50% increase in the fraction of rocks in the feed ore at time 100 minutes. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



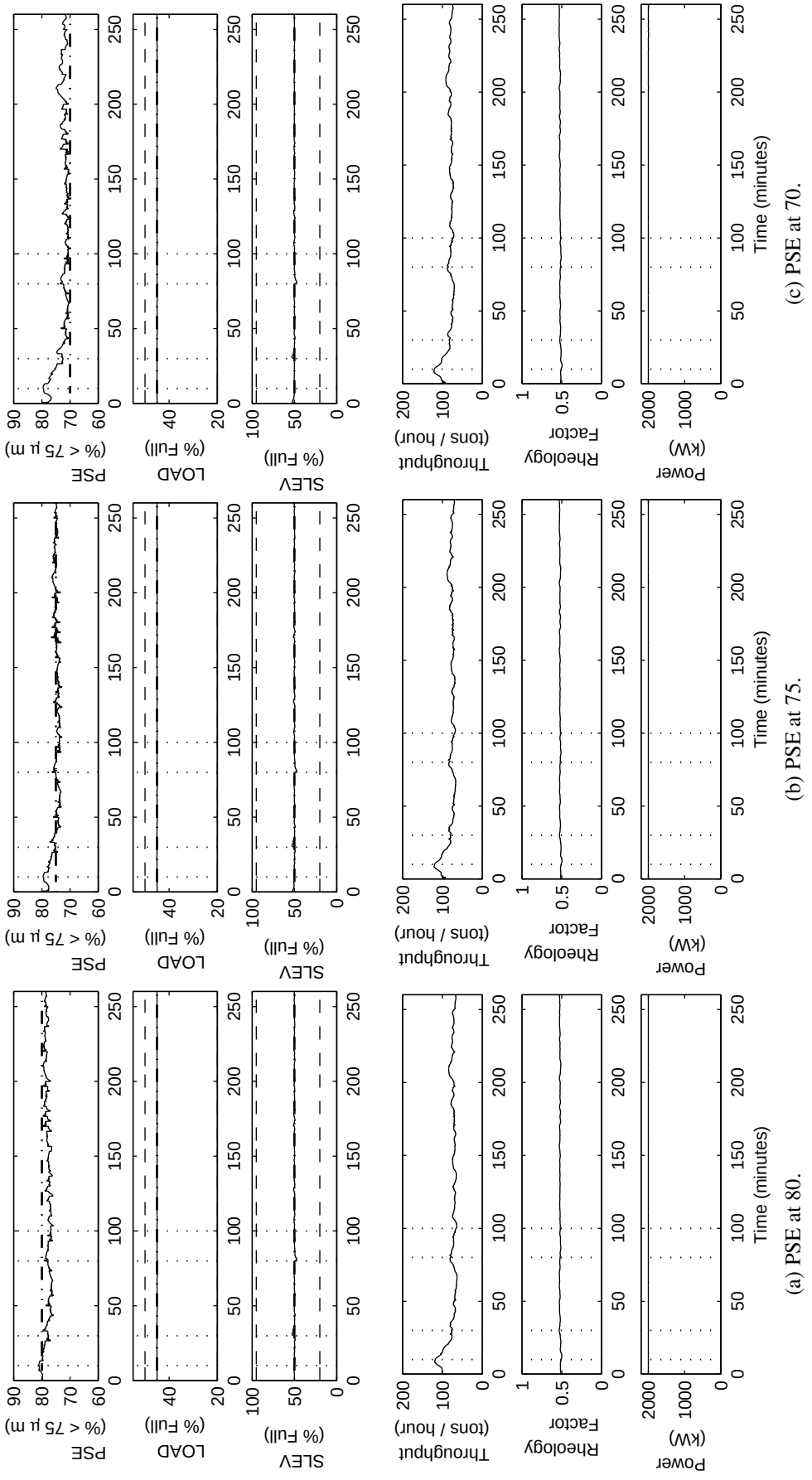
(a) PSE at 80. (b) PSE at 75. (c) PSE at 70.

Figure 5.16: CVs of the RNMPC.
The RNMPC regulates PSE at a constant setpoint of 80% (a), 75% (b) and 70% (c) with LOAD at a constant setpoint of 5 m^3. The PSE setpoint is reduced in order to increase the average throughput. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



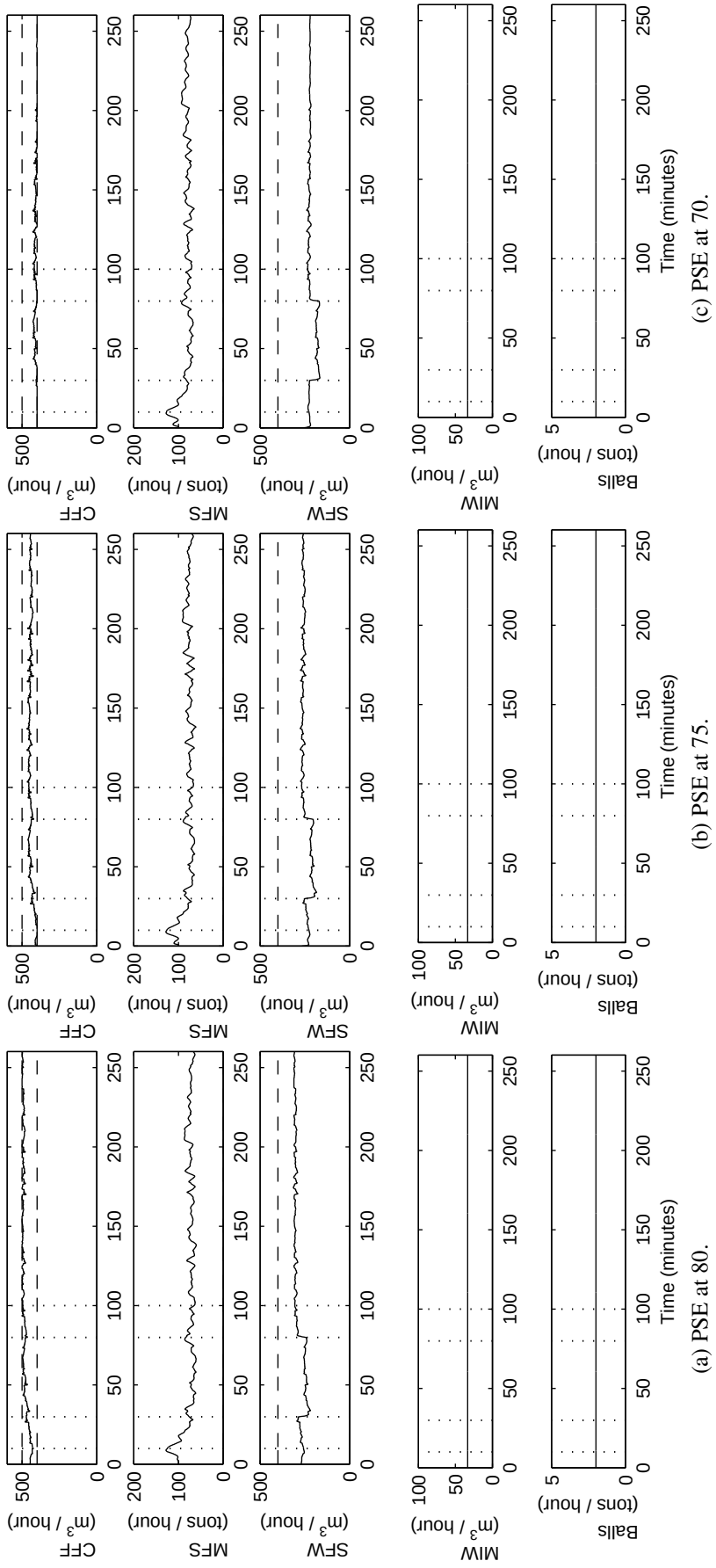
(a) PSE at 80. (b) PSE at 75. (c) PSE at 70.

Figure 5.17: MVs of the RNMPC.
The RNMPC regulates PSE at a constant setpoint of 80% (a), 75% (b) and 70% (c) with LOAD at a constant setpoint of 45% and SLEV at a constant setpoint of 5 m^3. The PSE setpoint is reduced in order to increase the average throughput. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) PSE at 80. (b) PSE at 75. (c) PSE at 70.

Figure 5.18: CVs of PI controllers.
The PI controllers regulate PSE at a constant setpoint of 80% (a), 75% (b) and 70% (c) with LOAD at a constant setpoint of 5 m^3. The PSE setpoint is reduced in order to increase the average throughput. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

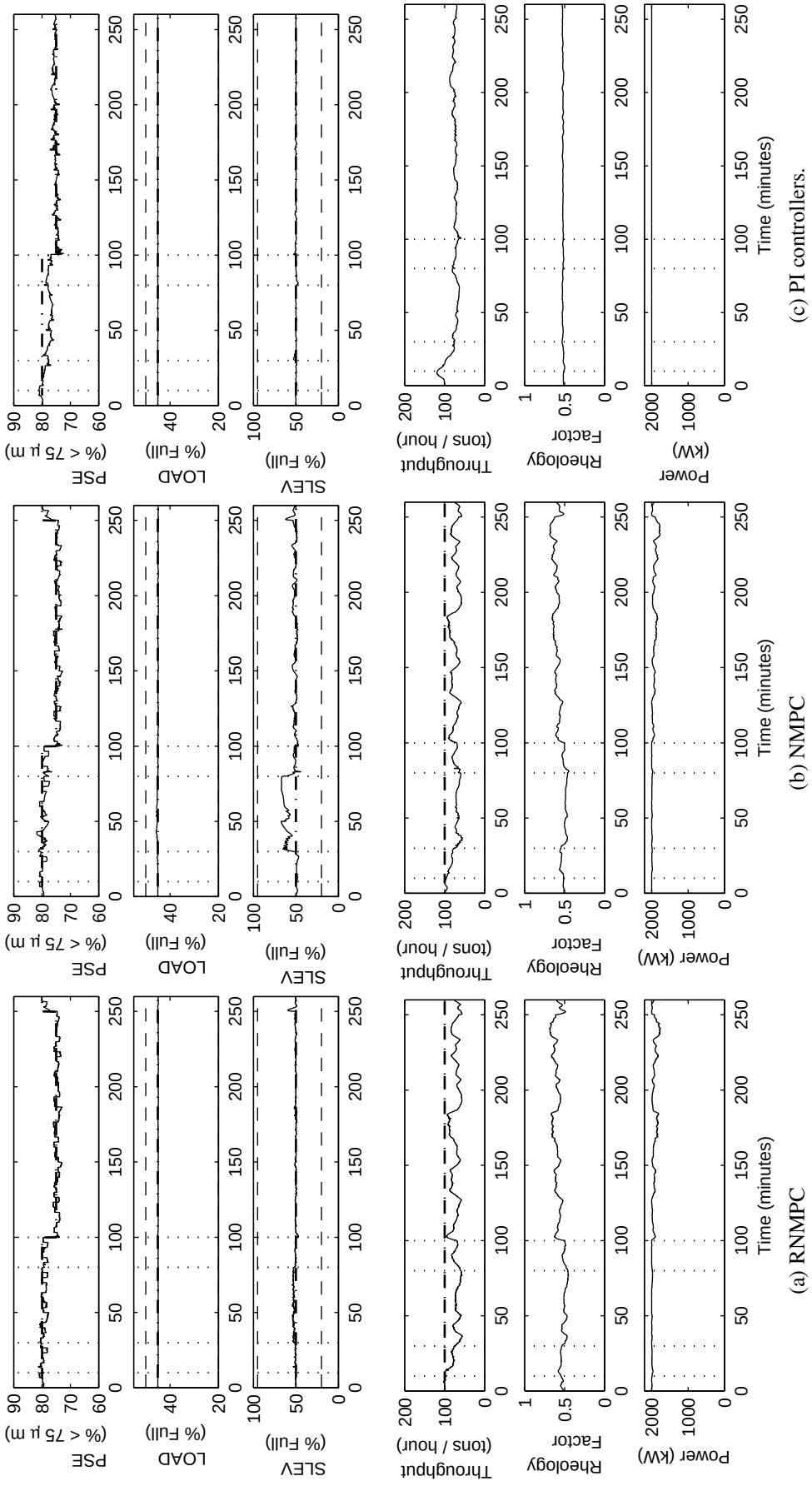


(a) PSE at 80.

(b) PSE at 75.

(c) PSE at 70.

Figure 5.19: MVs of PI controllers.
The PI controllers regulate PSE at a constant setpoint of 80% (a), 75% (b) and 70% (c) with LOAD at a constant setpoint of 45% and SLEV at a constant setpoint of 5 m^3. The PSE setpoint is reduced in order to increase the average throughput. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) RNMPC (b) NMPC (c) PI controllers.

Figure 5.20: CVs of RNMPC (a), NMPC (b) and PI controllers (c). The RNMPC and PI controllers follow a setpoint change from 80%-75% in PSE while maintaining LOAD and SLEV at a constant setpoint of 45% and 5 m^3 respectively. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

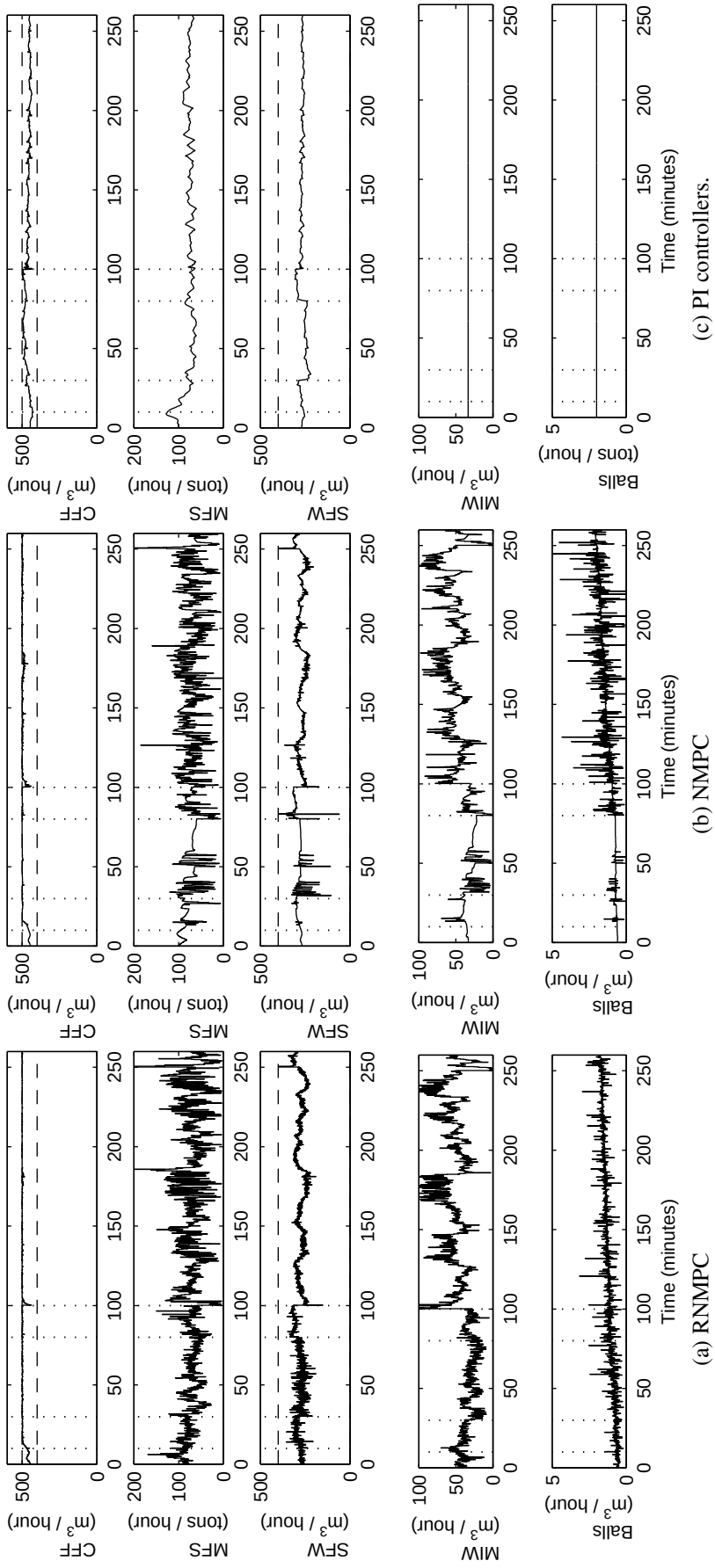


Figure 5.21: MVs of RNMPC (a), NMPC (b) and PI controllers (c). The RNMPC and PI controllers follow a setpoint change from 80%-75% in PSE while maintaining LOAD and SLEV at a constant setpoint of 45% and 5 m^3 respectively. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

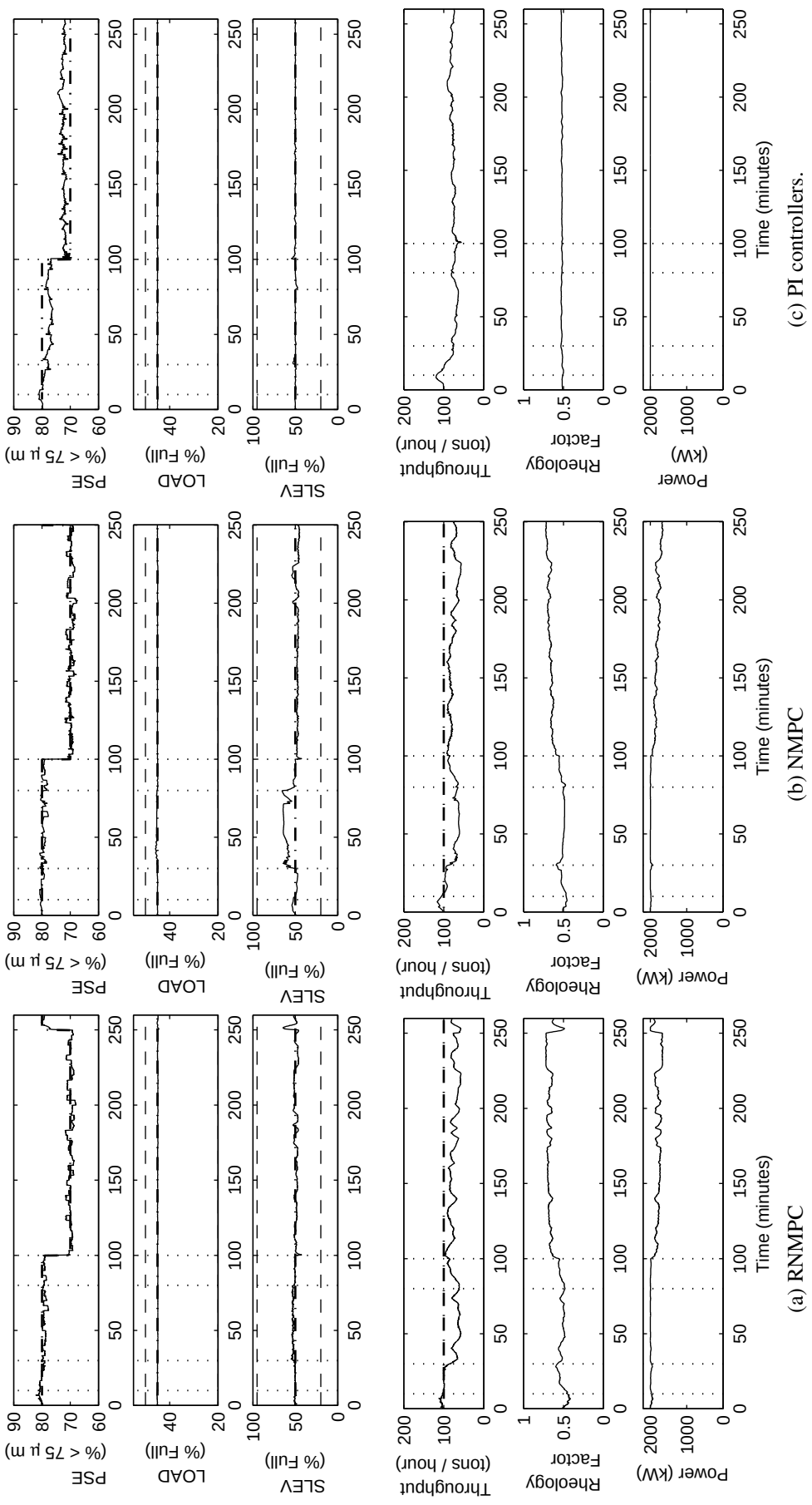


Figure 5.22: CVs of RNMPC (a), NMPC (b) and PI controllers (c). The RNMPC and PI controllers follow a setpoint change from 80%-70% in PSE while maintaining LOAD and SLEV at a constant setpoint of 45% and 5 m^3 respectively. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

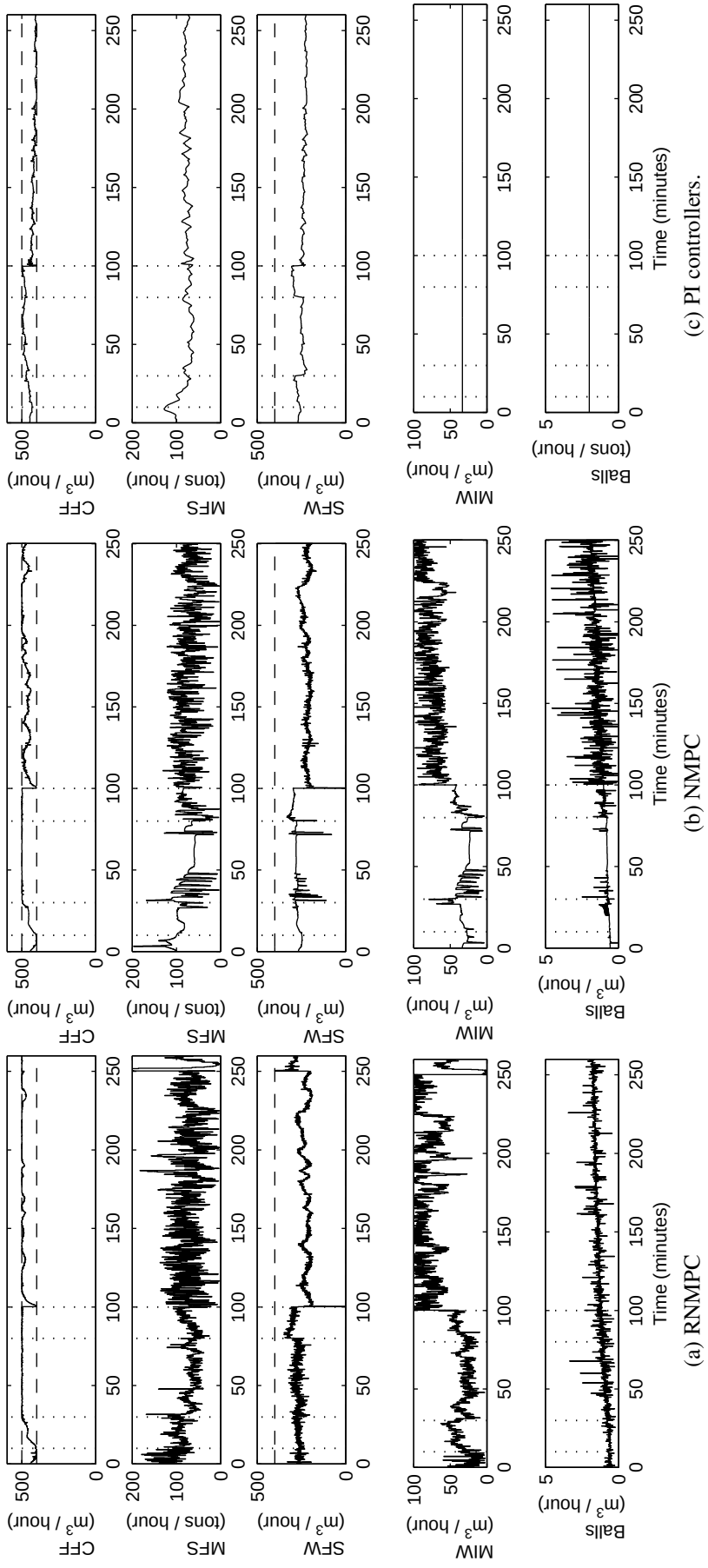
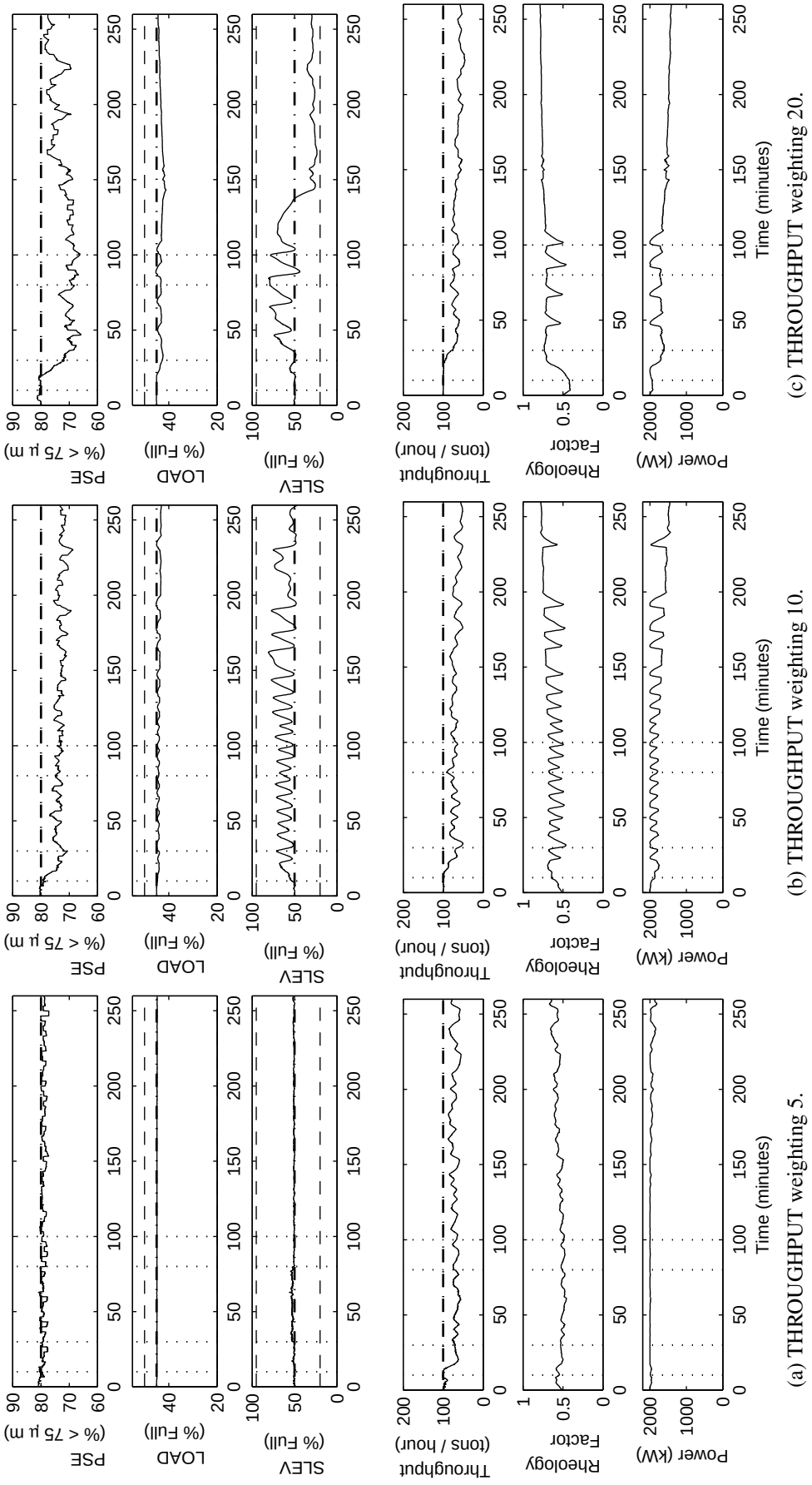


Figure 5.23: MVs of RNMPC (a), NMPC (b) and PI controllers (c). The RNMPC and PI controllers follow a setpoint change from 80%-70% in PSE while maintaining LOAD and SLEV at a constant setpoint of 45% and 5 m^3 respectively. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

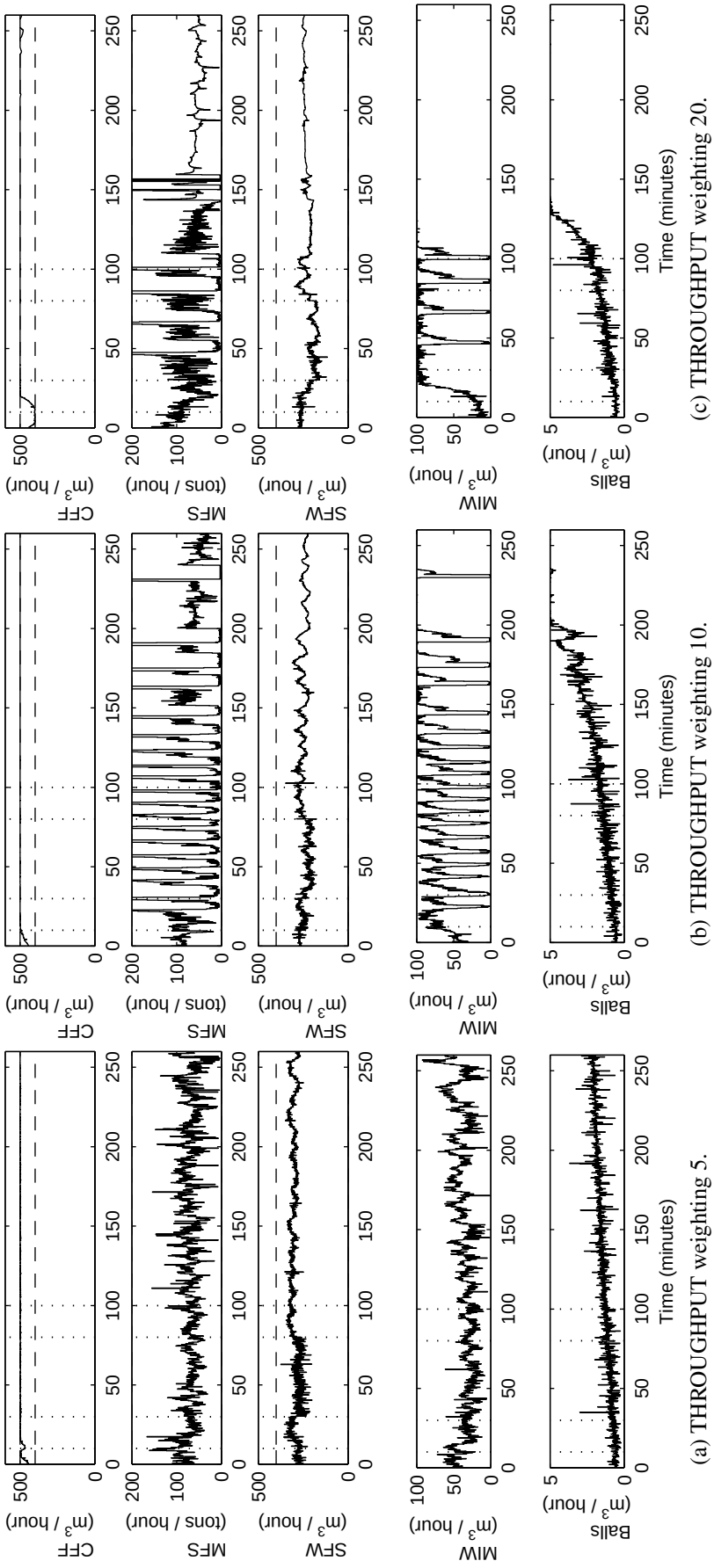


(a) THROUGHPUT weighting 5.

(b) THROUGHPUT weighting 10.

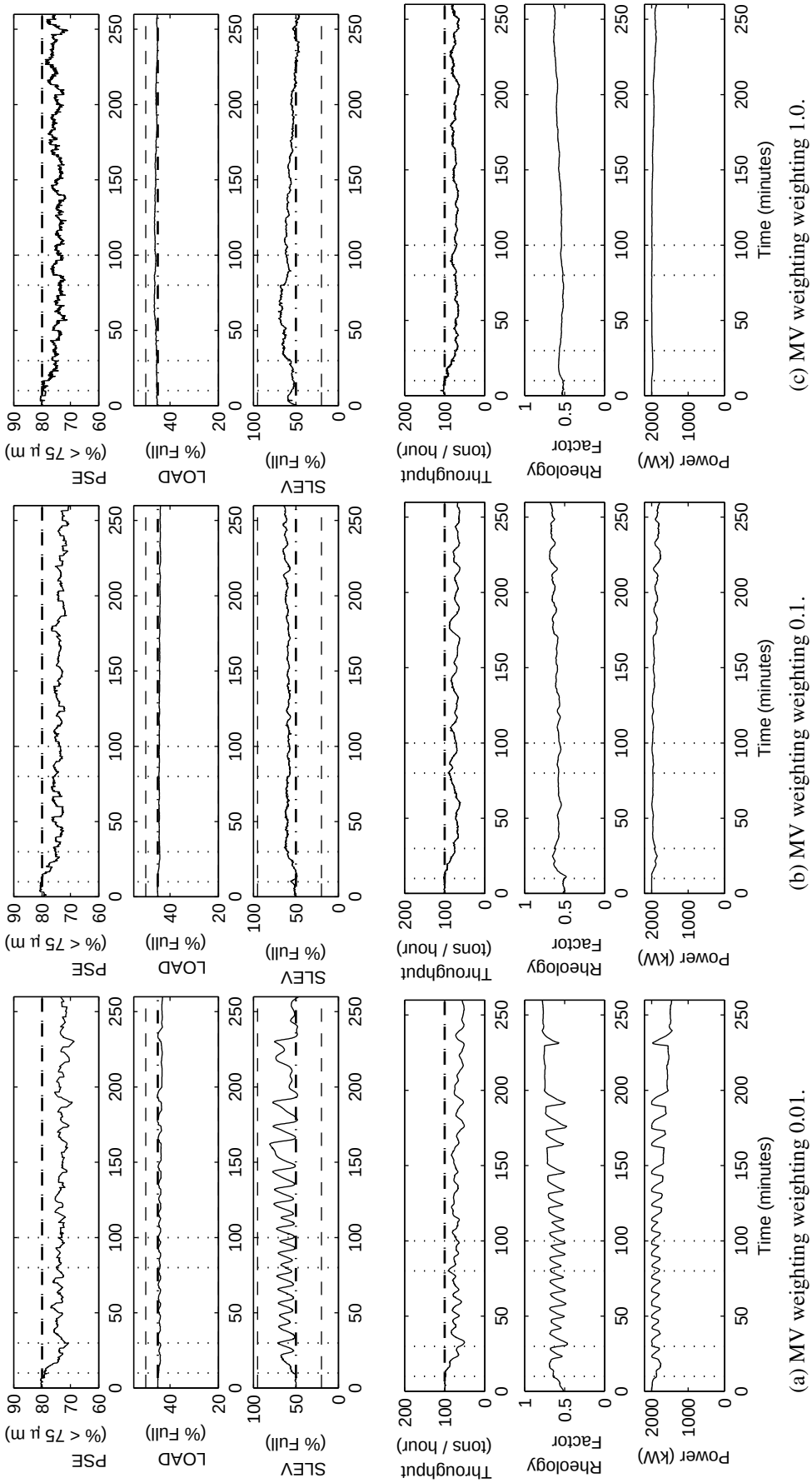
(c) THROUGHPUT weighting 20.

Figure 5.24: CVs of the RNMPC.
The RNMPC regulates PSE and LOAD at constant setpoints and throughout at a constant setpoint of 100 tons/hour. The weighting of THROUGHPUT in the objective function is 5 (a), 10 (b) and 20 (c), while the weighting on PSE and LOAD is 100 respectively. The optimising property of the RNMPC is employed to try and increase throughput without affecting PSE. An increase in THROUGHPUT, however, caused a decrease in PSE. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



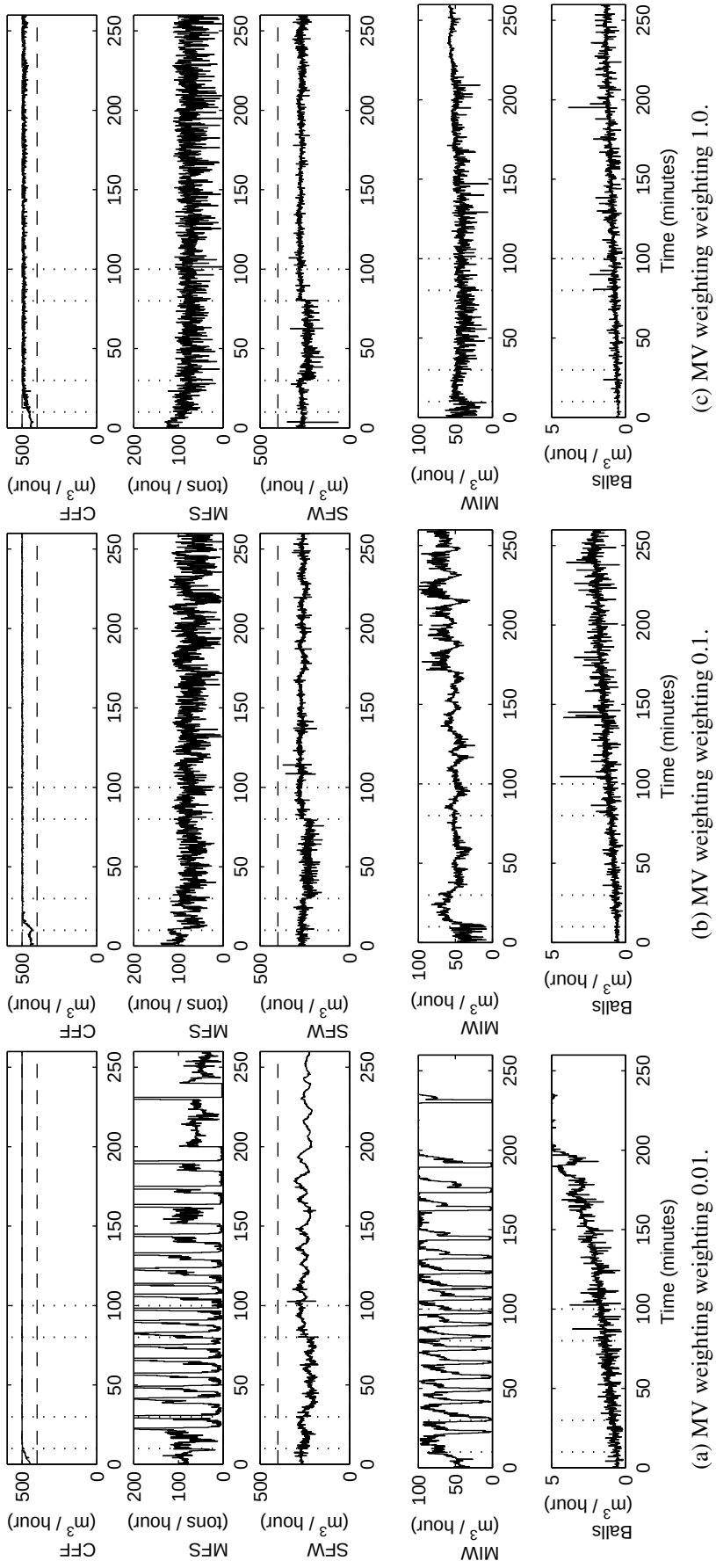
(a) THROUGHPUT weighting 5. (b) THROUGHPUT weighting 10. (c) THROUGHPUT weighting 20.

Figure 5.25: MVs of the RNMPC.
The RNMPC regulates PSE and LOAD at constant setpoints and throughout at a constant setpoint of 100 tons/hour. The weighting of THROUGHPUT in the objective function is 5 (a), 10 (b) and 20 (c), while the weighting on PSE and LOAD is 100 respectively. Increasing the weighting on THROUGHPUT caused increasingly oscillatory behaviour in the MVs. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) MV weighting weighting 0.01. (b) MV weighting weighting 0.1. (c) MV weighting weighting 1.0.

Figure 5.26: CVs of the RNMPC.
The weighting on the MVs are increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to reduce the oscillation in the MVs. The increased weighting on the MVs results in better tracking of PSE and higher average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) MV weighting weighting 0.01.

(b) MV weighting weighting 0.1.

(c) MV weighting weighting 1.0.

Figure 5.27: MVs of the RNMPC.
The weighting on the MVs is increased from 0.01 (b) to 0.1 (b) and 1.0 (c) in order to reduce the oscillation in the MVs. The increased weighting on the MVs results in better tracking of PSE and higher average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

5.4 DISCUSSION

The simulation results in this chapter may lead to some questions regarding the choices of the disturbances and controller design. This section aims to answer some of the questions that come to mind.

The step disturbance in ore hardness leads to CFF saturation in most of the simulations. Feed ore hardness and composition changes are major disturbances that milling circuit controllers have to contend with, as could happen when the feed ore is switched between feeds that originate from different stockpiles containing different ore types. The step disturbances are therefore chosen to simulate these disturbances.

The single-loop PI controller responsible for controlling the PSE-CFF loop exhibits a slow response. The PSE-CFF loop is characterised by a non-minimum phase response, as seen in Section 4.3.1. Tuning the PI controller more aggressively than recommended by the SIMC tuning method resulted in poorer performance. The limiting factor for tuning the PI controller is therefore the non-minimum phase response exhibited by the model.

It does not seem necessary to control SLEV, because the use of such a reservoir is to absorb disturbances, which is not done when its level is controlled. RNMPC and NMPC can be configured to allow SLEV to vary freely between upper and lower bounds. The scenario where SLEV is allowed to vary freely was investigated in Section 5.3.2 and found not to improve performance significantly. The weighting of SLEV compared to PSE and LOAD is significantly less, which allows the controller to vary SLEV in order to maintain PSE and LOAD at setpoint, as seen in Figure 5.26.

The performance of the PI controllers can be improved by using simple MIMO compensators. A recent survey by Wei and Craig (2009), however, reported that more than 60% of all respondents still only use PI control for grinding mill circuit control, usually single-loop PI controllers. This study aimed to determine the advantage that multivariable control, in the form of RNMPC and NMPC, may provide over SISO control, such as single-loop PI control, which is still employed in the majority of grinding circuits.

Figure 5.1 shows an example of typical parameter variations graphically, as employed in the simulations. The parameter variations were chosen to be uniformly distributed, because this results in a large change from one parameter vector to the next. The parameters are kept constant for a long enough period to allow the new parameter values to propagate through the process and affect its response.

MIW shows large variations that are not typically allowed in practice (Craig *et al.*, 1992*a*). The MIW is usually ratio-controlled to the MFS to maintain a relatively constant solids-to-water ratio inside the mill. Craig *et al.* (1992*a*) showed that MIW can be used to extend the control of PSE. For the simulation studies presented in this thesis, full authority of MIW was, therefore, given to the MPC controllers in order to determine if MIW can be used to increase control of the important variables, such as PSE and THROUGHPUT.

The manipulated variables of the RNMPC and NMPC show spikes. Figure 5.28 and Figure 5.29 enlarge a relatively spiky region of Figure 5.2 and Figure 5.3 between time 95 minutes and 105 minutes, showing that the spikes are not as severe as it seem from the full-length simulation results. The RNMPC show much less variation in the MVs than the NMPC, because RNMPC employ rate constraints as shown in the Δ column of Table 5.1. The rate constraints do not seem to have an impact on the performance of the RNMPC negatively compared to the NMPC. The spikes can be reduced further by filtering the measured variables used by the controllers. Mhaskar and Kennedy (2008) warn, however, that rate constraints can affect the closed-loop stability of the system if they are not incorporated correctly into the controller formulation.

The performance of the NMPC controller is very similar to the performance of the RNMPC controller. Some of the differences can be attributed to the ability of the RNMPC to handle rate constraints on the MVs. The RNMPC did not exhibit any significant stability or performance advantage in a majority of the simulations over the NMPC, which leads to the conclusion that the NMPC controller is more than adequate for controlling the ROM milling circuit presented here.

Nonlinear control can be justified by noting that the cyclone model (Section 2.3.4.4) and the mill power draw function (equation (2.27)) are static nonlinear models that will reduce to constant gains when linearised and will, therefore, only be accurate in a small region around the operating point. Static nonlinear models in the form of efficiency curves are used to model a number of classification units in minerals processing, such as cyclones (Nageswararao *et al.*, 2004) and screens. Nonlinear controllers, such as NMPC and RNMPC, have the advantage of being able to use these static nonlinear models directly as internal prediction models ensuring accurate results over a larger operating window.

5.5 SIMULATION SUMMARY

A summary of the simulation results are given in Table 5.4 that details the tracking performance of the controller with regard to the PSE, LOAD and throughput. The simulation scenarios are outlined in terms of the controlled variable weighting and setpoint values, as well as the step disturbances. The relevant changes in each simulation scenario are highlighted in bold. The dashes in the table represent values that are either zero or not applicable. The performance metrics are described in Section 5.2.

The headings of Table 5.4 are defined as

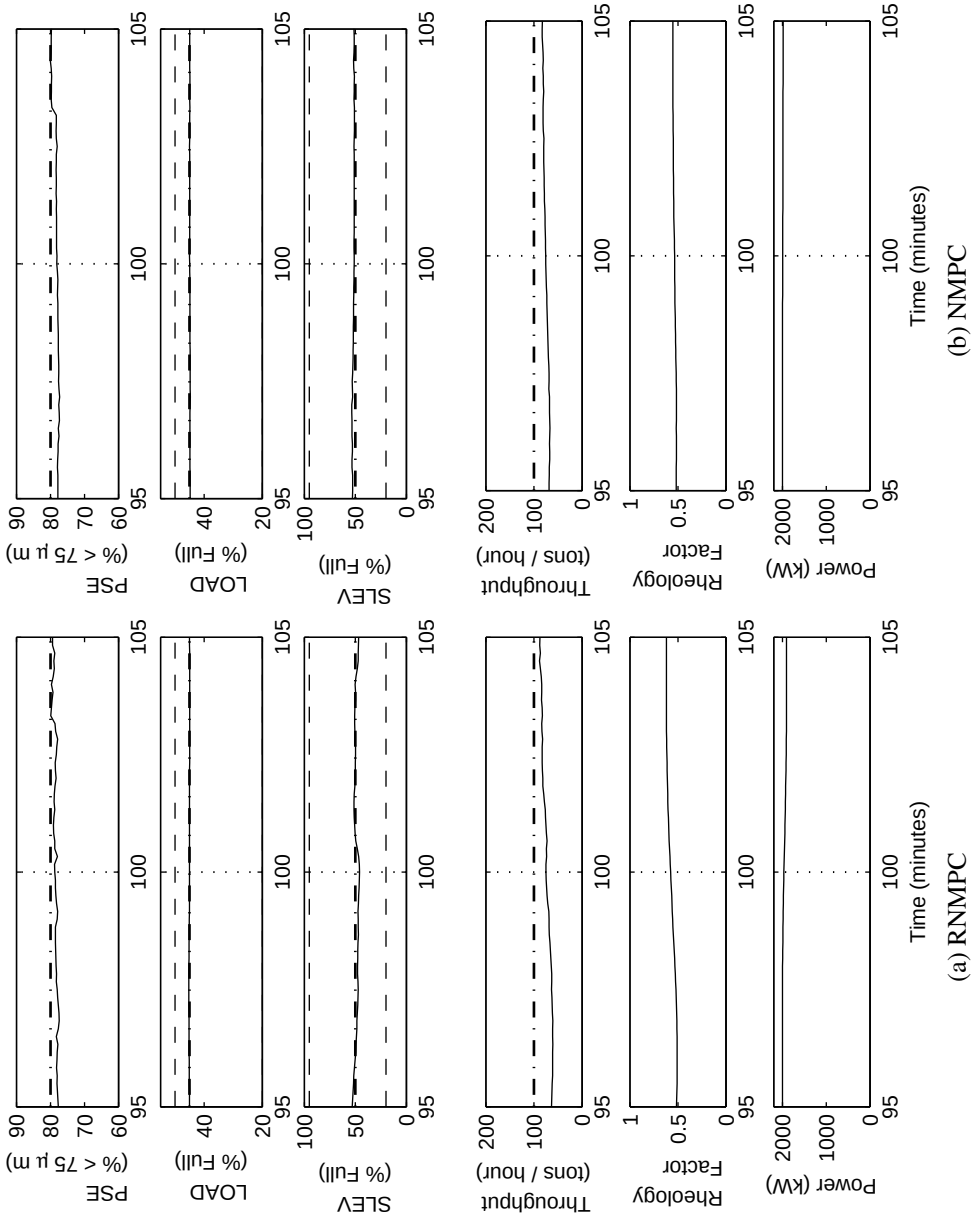


Figure 5.28: CVs of RNMPC (a) and NMPC (b) controllers.
Feed ore hardness and composition step disturbances are present as well as a sump feed water disturbance simulating spillage water being added to the sump. This simulation show the results from 95 minutes to 105 minutes of Figure 5.2. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

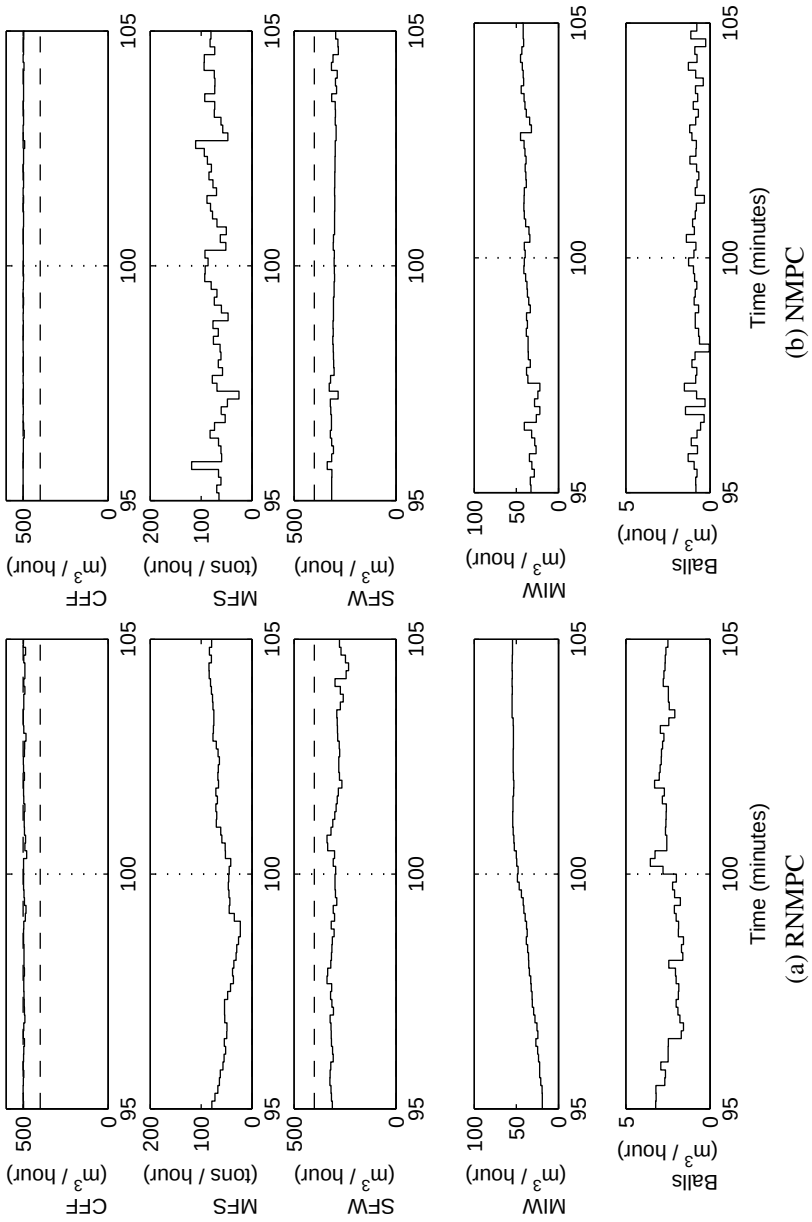


Figure 5.29: MVs of RNMPC (a) and NMPC (b) controllers.
Feed ore hardness and composition step disturbances are present as well as a sump feed water disturbance simulating spillage water being added to the sump. This simulation show the results from 95 minutes to 105 minutes of Figure 5.3. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

PSE	Particle Size Estimate. [$\% < 75\mu$m]
LOAD	The volumetric filling of the mill. [%]
Throughput	The amount of solids discharged at the cyclone overflow. [$^{\text{tons}}/_{\text{hour}}$]
SLEV	Sump level. [m^3]
Power	The electrical power draw of the mill motor. [kW]
Rheology	An indication of the fluidity of the slurry inside the mill. [fraction]
U	The manipulated variables.
Disturbances	Describes the step disturbances in ore hardness, fraction of rocks in the feed ore and SFW.
Time	Describes the average and maximum iteration time of the simulations [seconds].

The subheadings of Table 5.4 are defined as

Δ	The sum of the squares of the error from the setpoint.
S	The setpoint of the variable.
W	The weight of the variable in the objective function.
A	The average value of the variable over the simulation duration.
RH	The increase in the hardness of the feed ore. [%]
AR	The increase in the fraction of rocks in the feed ore. [%]
SFW	The increase in SFW. [$^{m^3}/_{\text{hour}}$]
T	The time when the disturbance is introduced.
M	The maximum value of the variable over the simulation duration.

The controllers are identified next to the figure numbers in Table 5.4 by

R	Robust Nonlinear Model Predictive Controller.
N	Nonlinear Model Predictive Controller.
P	Proportional-Integral-Differential Controller.

The simulations show that RNMPC and NMPC are capable of tighter control of PSE, especially when constraints are active, because the multivariable controllers can leverage the multivariable nature of the milling circuit to increase the control envelope. The RNMPC and NMPC controllers are, however, not capable of improving throughput while maintaining PSE at the desired setpoint. The PI controllers showed very good performance for SISO control, because they were tuned very aggressively. In certain milling circuits there are large

time delays that will only allow less aggressive tuning of the PI controllers, resulting in degraded performance. MPC controllers were found in practice to provide good performance over longer periods compared to PI controllers (Chen *et al.*, 2007*b*, Ramasamy *et al.*, 2005).

Some additional simulation scenarios are considered in Section B.2.

Table 5.4: Simulation summary.

Simulation			PSE			LOAD			Throughput			SLEV		Power		Rheology		U	Disturbances					Time	
			Δ	S	W	Δ	S	W	A	S	W	S	W	S	W	S	W	W	RH	T	AR	T	S	T	A	M
Figure 5.4	a	**R**	0.05	80	100	2.90E-05	45	100	106.72	99.9	0	5	**0**	2000	0	0.51	0	0.01	—	—	—	—	—	—	12.8	132
Figure 5.5	b	**R**	0.05	80	100	3.50E-05	45	100	111.00	99.9	0	5	**1**	2000	0	0.51	0	0.01	—	—	—	—	—	—	22.6	186
	c	**P**	0.15	80	—	5.35E-04	45	—	106.28	—	—	5	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Figure 5.6	a	**R**	0.07	80	100	3.30E-05	45	100	88.71	99.9	0	5	1	2000	0	0.51	0	0.01	**50**	**180**	—	—	—	—	21.7	318
Figure 5.7	b	N	0.08	80	100	2.40E-05	45	100	88.97	99.9	0	5	1	2000	0	0.51	0	0.01	**50**	**180**	—	—	—	—	0.1	1.52
	c	**P**	0.97	80	—	4.58E-04	45	—	95.67	—	—	5	—	—	—	—	—	—	**50**	**180**	—	—	—	—	—	—
Figure 5.8	a	**R**	0.05	80	100	4.00E-05	45	100	103.5	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**50**	**180**	—	—	24.5	204
Figure 5.9	b	N	0.09	80	100	7.30E-05	45	100	102.86	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**50**	**180**	—	—	0.1	1.56
	c	**P**	0.18	80	—	5.82E-04	45	—	105.53	—	—	5	—	—	—	—	—	—	—	—	**50**	**180**	—	—	—	—
Figure 5.10	a	**R**	0.06	80	100	4.40E-05	45	100	103.26	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**50**	**10**	—	—	27.1	291
Figure 5.11	b	N	0.23	80	100	3.06E-04	45	100	101.77	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**50**	**10**	—	—	0.1	1.47
	c	**P**	0.41	80	—	6.32E-04	45	—	106.96	—	—	5	—	—	—	—	—	—	—	—	**50**	**10**	—	—	—	—
Figure 5.12	a	**R**	0.05	80	100	4.40E-05	45	100	110.89	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**0**	—	**50**	**180**	26.7	340
Figure 5.13	b	N	0.08	80	100	6.60E-05	45	100	110.28	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**0**	—	**50**	**180**	0.1	1.59
	c	**P**	0.16	80	—	7.71E-04	45	—	110.30	—	—	5	—	—	—	—	—	—	—	—	**0**	—	**50**	**180**	—	—
Figure 5.14	a	**R**	0.23	80	100	2.03E-03	45	100	72.22	99.9	1	5	1	2000	0	0.51	0	0.01	**50**	**10**	**50**	**100**	**50**	**30-80**	34.9	1014
Figure 5.15	b	N	0.3	80	100	1.76E-03	45	100	72.58	99.9	1	5	1	2000	0	0.51	0	0.01	**50**	**10**	**50**	**100**	**50**	**30-80**	0.2	1.77
	c	**P**	0.77	80	—	1.52E-04	45	—	74.59	—	—	5	—	—	—	—	—	—	**50**	**10**	**50**	**100**	**50**	**30-80**	—	—
Figure 5.16	a	R	0.23	**80**	100	2.03E-03	45	100	72.22	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	34.9	1014
Figure 5.17	b	R	0.12	**75**	100	1.54E-03	45	100	74.51	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	35.9	1132
	c	R	0.12	**70**	100	9.33E-04	45	100	81.35	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	33.4	1136
Figure 5.18	a	P	0.77	**80**	—	1.52E-04	45	—	74.59	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
Figure 5.19	b	P	0.24	**75**	—	1.62E-04	45	—	77.98	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
	c	P	1.50	**70**	—	1.40E-04	45	—	81.20	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
Figure 5.20	a	**R**	0.15	**80→75**	100	1.65E-03	45	100	74.04	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	33.8	688
Figure 5.21	b	N	0.18	**80→75**	100	1.55E-03	45	100	74.22	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	9.6	58
	c	**P**	0.39	**80→75**	—	1.64E-04	45	—	76.43	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
Figure 5.22	a	**R**	0.15	**80→70**	100	1.42E-03	45	100	76.73	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	39.3	1127
Figure 5.23	b	N	0.14	**80→70**	100	1.21E-03	45	100	78.11	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	10.3	70
	c	**P**	0.92	**80→70**	—	1.74E-04	45	—	78.52	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
Figure 5.24	a	R	2.79	80	100	1.55E-02	45	100	71.14	99.9	**5**	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	21.0	813
Figure 5.25	b	R	7.09	80	100	3.44E-02	45	100	70.76	99.9	**10**	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	15.2	619
	c	R	10.01	80	100	6.85E-02	45	100	68.09	99.9	**20**	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	13.3	682
Figure 5.26	a	R	7.09	80	100	3.44E-02	45	100	70.76	99.9	**10**	5	1	2000	0	0.51	0	**0.01**	50	10	50	100	50	30-80	15.2	619
Figure 5.27	b	R	5.52	80	100	2.62E-02	45	100	74.85	99.9	**10**	5	1	2000	0	0.51	0	**0.1**	50	10	50	100	50	30-80	49.2	1215
	c	R	4.11	80	100	2.47E-02	45	100	75.6	99.9	**10**	5	1	2000	0	0.51	0	**1.0**	50	10	50	100	50	30-80	33.2	987

CHAPTER 6

CONCLUSIONS AND FURTHER WORK

6.1 SUMMARY AND EVALUATION

Advanced control such as MPC has not been as readily adopted by the mineral-processing industry as compared to, for example, the petrochemical industry (Wei and Craig, 2009). This thesis investigated the feasibility of applying RNMPC to a ROM ore milling circuit and the conditions under which such a controller might be worthwhile implementing.

A comprehensive modularised ROM ore milling circuit model was described in Section 2.3.4 and cast into an RNMPC framework. The results of practically motivated simulations presented in Chapter 5 show that an RNMPC controller can successfully control important milling circuit variables in the face of large disturbances that are not uncommon in practice.

The adoption of advanced control by the mineral processing industry will probably be determined by the trade-off between the added complexity of implementing and maintaining an advanced controller such as RNMPC and the benefits that can be derived from such an implementation. Results given in this thesis suggest that if a milling circuit regularly experiences large feed ore hardness and composition changes, when for example the feed ore is switched between feeds that originate from different stockpiles, RNMPC might well warrant a closer look.

6.1.1 Strong points

The RNMPC was successfully implemented by using an open-source package called IPOPT (Kawajir *et al.*, 2006) for large-scale nonlinear parameter optimisation and an open-source package called CPPAD (Lougee-Heimer, 2003) for calculating the derivatives of $g(\cdot)$ as well as solving the differential equation (6.1) by integration. The efficiency of the algorithm is

obtained by structuring the problem correctly (Section 3.7.3) and minimising unnecessary calculations by keeping track of the available results and only returning the results if these are already available, rather than recalculating them.

RNMPC can explicitly take model uncertainty into account as part of the prediction model. This allows RNMPC to be more robust against model uncertainty and process disturbances.

The results shown in Chapter 5 and Addendum B and summarised in Addendum D suggest that if a milling circuit regularly experiences large feed ore hardness and composition changes, when for example the feed ore is switched between feeds that originate from different stockpiles, RNMPC may provide better performance than PI controllers. RNMPC can extend the operation envelope of the process when constraints become active to maintain the performance of the process.

RNMPC can under certain conditions provide better performance compared to NMPC, as seen in the simulation results (Addendum D), because it optimises over the worst-case realisation of the system.

RNMPC can avoid conditions in the milling circuit that affect production negatively, such as mill overload conditions, because it can enforce state and output constraints. RNMPC can therefore use the sump to stabilise the slurry density before pumping it to the cyclone, effectively increasing the operational envelope of the milling circuit. This is accomplished by varying the sump level within its limits. PI control cannot achieve any of these two feats, because it cannot enforce constraints on the outputs.

RNMPC can incorporate almost any nonlinear model without major modifications (such as linearisation and conversion to a fixed structure). It even handles a nonlinear static model very well, such as the cyclone model, where most of the "dynamics" are lost when the model is linearised. Nonlinear static models are common in minerals processing, where cyclones and screens are usually modelled by efficiency equations (Wills and Napier-Munn, 2006). The RNMPC only requires the nonlinear model interface to be defined as

$$\dot{x}(t) = g(x(t), u(t), p(t)) \tag{6.1}$$

where $x(t)$ is the current state vector at time t, $\dot{x}(t)$ is the change in states at time t, $u(t)$ is the control vector at time t, and $p(t)$ the parameters at time t. The dimensions of the vectors $(x(t), u(t), p(t))$ may be of arbitrary size. The requirements on the nonlinear model are that the model should be stabilisable and twice continuous differentiable, but they do not impose any fixed structure on the model, which allows any arbitrary milling circuit to be simulated and controlled using this interface.

The RNMPC implementation presented in this thesis is parallel processor capable, which allows it to be sped up on multi-core/multi-processor systems. Complex systems can therefore be controlled with a sufficient investment in computer hardware.

Recent studies of applying MPC to ore milling circuits have shown that model predictive controllers provide better long-term stability than PI control (Chen *et al.*, 2007*a*, *b*, 2008).

Most industrial MPC controllers require brute-force simulation to evaluate the effects of model mismatch on closed-loop stability (Qin and Badgwell, 2003). Time spent tuning and testing of industrial controllers can, however, be significantly reduced if the controllers implement nominal and potentially robust stability measures, such as RNMPC, even though closed-loop stability of industrial MPC itself is not perceived to be a serious problem by industry practitioners (Qin and Badgwell, 2003).

6.1.2 Drawbacks

The simulation was executed at an average time of about 26 seconds and a maximum time of 123 seconds per iteration on a Dell PowerEdge 1955 blade with Intel Xeon 5140 (Dual-Core) 2.33GHz processor, 2GB RAM and 1333MHz FSB. This platform is typically faster than the implementation platforms available on most mineral-processing plants.

The current RNMPC implementation is not feasible for practical implementation, because the maximum and average calculation times are much longer than the recommended sampling time of 10 seconds (Craig and MacLeod, 1995). There are various factors that influence the calculation time. Increasing the prediction horizon T increases the calculation time because of an increase in total integration time. Increasing the number of nodes N significantly increases the computational time, because of an increase in the number of decision variables. The control horizon is determined by the number of nodes N, because the MVs are changed at each node. Tuning the controller will therefore need to include the selection of the prediction horizon T and number of nodes N for stability and performance, while maintaining a reasonable calculation time. With the continuous increase in computing power, this should become less of an issue in the foreseeable future.

The simulation further assumed full-state feedback, which is not available on real plants. Typically the controlled variables PSE, SLEV, LOAD and the cyclone feed or sump density would be measured online (Wei and Craig, 2009), from which an observer would need to infer the model states.

From a modelling perspective it is more difficult to obtain and fit a nonlinear model, especially if it contains an uncertainty description. More step tests would therefore be needed to obtain the uncertainty description. The nonlinear model presented in this study does contain model parameters that relate better to the physical process, which makes it easier to estimate bounds on the parameters from experience and other experiments than just step tests on the plant. The advantage of PI control is that it does not require an uncertainty description to design the controller. PI control therefore requires less engineering effort to design compared to RNMPC.

Online tuning of both the RNMPC and PI controllers would be required for best performance. Every time there is a change in the plant hardware, such as new instrumentation or actuators, recalibration of the model for the RNMPC as well as retuning of PI controllers would usually be required.

The cost of maintaining the nonlinear model may be reduced if the nonlinear model can be structured to have calibration parameters and process parameters. The calibration parameters can relate to gains of measurement equipment and gains of actuators. These calibration parameters can be assumed to be time-invariant and known. This will reduce the amount of work necessary for recalibration due to hardware changes.

However, the process parameters that relate to feed size distribution and ore hardness, for example, are specified as time-varying and uncertain. These process parameters would not need to be re-calibrated with changes in the process hardware, but would need to be re-calibrated for changes occurring in the process, such as a new type of ore being milled. The process parameters will require more effort to obtain, because uncertainty bounds would usually need to be established for these parameters.

The nonlinear model of the RNMPC is therefore more costly to commission and maintain compared to simple PI controllers. The cost of the hardware required to host the controller is also much higher. The feasibility analysis of using RNMPC compared to PI control will need to weigh the advantage that RNMPC can bring in terms of process performance against the added commissioning and maintenance costs.

The PI controllers presented in this thesis serve only as an example implementation to provide a baseline for comparison with RNMPC and NMPC. Single-loop PI controllers without a MIMO compensator were used, because more than 60% of all respondents still use PI controllers, according to a recent survey by Wei and Craig (2009), usually single-loop PI controllers. RNMPC and NMPC are, however, fundamentally different from single-loop PI controllers, because RNMPC and NMPC are multivariable, model-based controllers that can handle constraints explicitly, while single-loop PI controllers cannot handle multivariable interaction very well and only have constraint-handling capabilities through extensions, such as anti-windup. Single-loop PI controller design for multivariable systems with constraints is a complex field with a large number of issues and solutions. There are furthermore a number of different techniques to tune the controllers from Ziegler-Nichols, internal model control with SIMC (Skogestad, 2003) for example, to pole placement, etc. The PI controllers presented here are not intended to serve as the best PI controller design for the presented milling circuit based on an exhaustive study, because that was not the main focus of this thesis. The comparison of the RNMPC and NMPC controllers to the PI controllers should, therefore, not be seen as a definitive, but rather serve as an example of possible benefits that RNMPC can provide over PI control typically employed in industry (Wei and Craig, 2009), when large feed disturbances are common in the milling circuit.

The performance of the NMPC controller is very similar to the performance of the RNMPC

controller. Some of the differences can be attributed to the ability of the RNMPC to handle rate constraints on the MVs. The RNMPC did not exhibit any significant stability or performance advantage in a majority of the simulations over the NMPC, which leads to the conclusion that the NMPC controller is more than adequate for controlling the ROM milling circuit presented here.

6.2 FURTHER WORK

The first barrier to the practical implementation of RNMPC is reducing the computational time. The computational time of the optimisation problem can be improved by

- decreasing the optimisation problem size, because the control algorithm is hampered mainly by a large number of slack variables for implementing robustness, but this will require simplifying assumptions to be made that can increase conservatism,
- replacing the nonlinear optimiser by a method that is less computationally expensive, or
- employing a nonlinear optimiser with better support for multiprocessor systems.

The real time iteration scheme developed by Diehl *et al.* (2005) allows the whole optimisation problem to be prepared before the state measurement is available, and only requires a small amount of time to finalise the calculation when the state measurement becomes available. It also performs only one iteration per interval, which gives it a very consistent calculation time that is very beneficial for practical control purposes.

The second barrier is the lack of an observer to allow for output feedback, rather than full-state feedback. The simulations were conducted with the assumption of full-state feedback, which is not possible in real life scenarios. A form of state estimation should be added to the loop to obtain a more accurate simulation of the closed-loop response. A classical extended Kalman filter can be used for state estimation or a form of moving horizon estimator. Moving horizon estimators are similar to MPC in principle and can easily be incorporated in the current framework. The moving horizon estimator could also include the nonlinear model directly rather than a linearised model, as will be needed by the extended Kalman filter. The amount of work to maintain different models for simulation, estimation and control will be reduced by using the nonlinear simulation model directly in the estimator as well as in the controller. For nonlinear systems this is more difficult, because the combination of the controller and observer should be stable.

The RNMPC presented in this study is based on open-loop min-max optimisation. There is a spread of possible future state trajectories when predicting the evolution of an uncertain system, which is the result of accounting for all possible realisations of the system owing to

uncertainty in the model parameters, as well as the possible disturbances that might occur in future. The effects of feedback are not taken into consideration during predictions of future state trajectories in open-loop predictive control, which results in the state trajectories diverging more over the prediction horizon compared to closed-loop or feedback formulations. This increased spread in state trajectories will result in unnecessary conservatism and a reduced feasible region of the controller. The RNMPC can be modified to optimise over feedback laws that will result in a feedback robust nonlinear model predictive controller. The effects of feedback are then taken into consideration as part of the predictions, which will reduce the spread of state trajectory over the prediction horizon. This reduced spread of state trajectories compared to open-loop formulations has the result of an increased feasible region, reduced conservatism and increased performance of the controller.

Reduce the spikes in the manipulated variables by adding the appropriate rate constraints to the controller formulation that will not affect the stability properties of the controller (Mhaskar and Kennedy, 2008).

REFERENCES

Abbaszadeh, M. and H. J. Marquez (2009). LMI optimization approach to robust H_∞ observer design and static output feedback stabilization for discrete-time nonlinear uncertain systems. *International Journal of Robust and Nonlinear Control* **19**(3), 313–340.

Abd El-Rahman, M. K., B. K. Mishra and R. K. Rajamani (2001). Industrial tumbling mill power prediction using the discrete element method. *Minerals Engineering* **14**(10), 1321–1328.

Alamo, T., J. M. Bravo and E. F. Camacho (2005). Guaranteed state estimation by zonotopes. *Automatica* **41**(6), 1035–1043.

Amestica, R., G. D. Gonzalez, J. Barria, L. Magne, J. Menacho and O. Castro (1993). A SAG mill circuit dynamic simulator based on a simplified mechanistic model. In: *XVIII International Mineral Processing Congress*. Sydney. pp. 117–129.

Amestica, R., G. D. Gonzalez, J. Menacho and J. Barria (1996). A mechanistic state equation model for semiautogenous mills. *International Journal of Mineral Processing* **44–45**, 349–360.

Apelt, T. A. (2002). Inferential measurement models for semi-autogenous grinding mills. PhD thesis. Department of Chemical Engineering, University of Sydney. Australia.

Apelt, T. A., S. P. Asprey and N. F. Thornhill (2001). Inferential measurement of SAG mill parameters. *Minerals Engineering* **14**(6), 575–591.

Apelt, T. A., S. P. Asprey and N. F. Thornhill (2002). Inferential measurement of SAG mill parameters II: State estimation. *Minerals Engineering* **15**(12), 1043–1053.

Atassi, A. N. and H. K. Khalil (1999). A separation principle for the stabilization of a class of nonlinear systems. *IEEE Transactions on Automatic Control* **44**(9), 1672–1687.

Atassi, A. N. and H. K. Khalil (2000). Separation results for the stabilization of nonlinear systems using different high-gain observer designs. *Systems & Control Letters* **39**(3), 183–191.

Austin, L. G. and P. T. Luckie (1972). Methods for determination of breakage distribution parameters. *Powder Technology* **5**(4), 215–222.

Austin, L. G. and V. K. Bhatia (1972). Experimental methods for grinding studies in laboratory mills. *Powder Technology* **5**(5), 261–266.

Austin, L. G., R. S. C. Rogers, K. A. Brame and J. Stubican (1988). A rapid computational procedure for unsteady-state ball mill circuit simulation. *Powder Technology* **56**(1), 1–11.

Austin, L. G., Z. Rogovin, R. S. C. Rogers and T. Trimarchi (1983). The axial mixing model applied to ball mills. *Powder Technology* **36**(1), 119–126.

Banini, G. A. (2000). An integrated description of rock breakage in comminution machines. PhD thesis. University of Queensland (JKMRC). Australia.

Bartolini, G., A. Pisano, E. Punta and E. Usai (2003). A survey of applications of second-order sliding mode control to mechanical systems. *International Journal of Control* **76**(9), 875–892.

Bazin, C. and C. B-Chapleau (2005). The difficulty associated with measuring slurry rheological properties and linking them to grinding mill performance. *International Journal of Mineral Processing* **76**(1–2), 93–99.

Bazin, C., M. St-Pierre and D. Hodouin (2005). Calibration of the perfect mixing model to a dry grinding mill. *Powder Technology* **149**(2–3), 93–105.

Ben-Tal, A. and A. Nemirovskii (2001). *Lectures on Modern Convex Optimization: Analysis, Algorithms and Engineering Applications*. MPS-SIAM Series on Optimization. MPS-SIAM. Philadelphia.

Ben-Tal, A. and A. Nemirovskii (2002). Robust optimization – Methodology and applications. *Mathematical Programming, Series B* **92**, 453–480.

Bhaumik, A., J. Sil, S. Banerjee and S. K. Dutta (1999). Supervised learning algorithm for open loop NN based control of tumbling mill. In: *Proceedings of the IEEE Region 10 TENCON 99 Conference.*. Vol. 1. pp. 395–398 vol.1.

Billings, S. A. (1980). Identification of nonlinear systems – A survey. *IEE Proceedings D on Control Theory and Applications* **127**(6), 272–285.

Bitmead, R. R., M. Gevers and V. Wertz (1990). *Adaptive optimal control – The thinking man's GPC*. Prentice-Hall. Englewood Cliffs, NJ.

Bock, H. G. and E. Kostina (2001). Robust experimental design. In: H.G. Bock, Project A4, Optimization methods for reactive flows of Sonderforshungsbereich 359 Reactive Flows, Diffusion and Transport, Report 1999-2001. Technical report. University of Heidelberg.

Bock, H. G. and K. J. Plitt (1984). A multiple shooting algorithm for direct solution of optimal control problems. In: *Proceedings 9th IFAC World Congress Budapest*. Pergamon Press. pp. 243–247.

Borell, M., P. Backstrom and L. Soderberg (1996). Supervisory control of autogenous grinding circuits. *International Journal of Mineral Processing* **44–45**, 337–348.

Bouche, C., C. Brandt, A. Broussaud and W. Drunick (2005). Advanced control of gold ore grinding plants in South Africa. *Minerals Engineering* **18**(8), 866–876.

Bravo, J. M., T. Alamo and E. F. Camacho (2006). Robust MPC of constrained discrete-time nonlinear systems based on approximated reachable sets. *Automatica* **42**(10), 1745–1751.

Camacho, E. F. and C. Bordons (2003). *Model Predictive Control*. Vol. Second Edition. Springer-Verlag. London, UK.

Casavola, A., D. Famularo and G. Franze (2003). Linear embedding vs. direct nonlinear MPC schemes: A case study. In: *Proceedings of the American Control Conference, 4-6 June*. Vol. 5. pp. 4305–4310.

Casavola, A., D. Famularob and G. Franzéa (2004). Robust constrained predictive control of uncertain norm-bounded linear systems. *Automatica* **40**, 1865–1876.

Chandramohan, R. and M. S. Powell (2005). Measurement of particle interaction properties for incorporation in the discrete element method simulation. *Minerals Engineering* **18**(12), 1142–1151.

Chandramohan, R. and M. S. Powell (2006). A structured approach to modelling SAG mill liner wear – Monitoring wear. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology, 23 – 27 September*. Vol. 3. Vancouver, B. C., Canada. pp. 133–148.

Chen, X., J. Zhai, Q. Li and S. Fei (2007*a*). Override and model predictive control of particle size and feed rate in grinding process. In: *Proceedings of the 26th Chinese Control Conference, July 26-31*. Zhangjiajie, Hunan, China. pp. 704–708.

Chen, X., J. Zhai, S. Li and Q. Li (2007*b*). Application of model predictive control in ball mill grinding circuit. *Minerals Engineering* **20**(11), 1099–1108.

Chen, X., Q. Li and S. Fei (2008). Constrained model predictive control in ball mill grinding process. *Powder Technology* **186**(1), 31–39.

Chen, X., S. Li, J. Zhai and Q. Li (2009). Expert system based adaptive dynamic matrix control for ball mill grinding circuit. *Expert Systems with Applications* **36**(1), 716–723.

Chu, D., T. Chen and H. J. Marquez (2007). Robust moving horizon state observer. *International Journal of Control* **80**(10), 1636–1650.

Clarke, D. W., C. Mohtadi and P. S. Tuffs (1987*a*). Generalized predictive control: Part I: The basic algorithm. *Automatica* **23**(2), 137–148.

Clarke, D. W., C. Mohtadi and P. S. Tuffs (1987*b*). Generalized predictive control: Part II: Extensions and interpretations. *Automatica* **23**(2), 149–160.

Cleary, P. (2001). Modelling comminution devices using DEM. *International Journal for Numerical and Analytical Methods in Geomechanics* **25**(1), 83–105.

Cleary, P. W., M. Sinnott and R. Morrison (2006). Prediction of slurry transport in SAG mills using SPH fluid flow in a dynamic DEM based porous media. *Minerals Engineering* **19**(15), 1517–1527.

Coetzee, L. C., I. K. Craig and E. C. Kerrigan (2008). Nonlinear model predictive control of a run-of-mine ore milling circuit. In: *proceedings of the 17th IFAC World Congress, July 6-11*. Seoul, Korea.

Coetzee, L. C., I. K. Craig and E. C. Kerrigan (2009). Robust nonlinear model predictive control of a run-of-mine ore milling circuit. *IEEE Transactions on Control Systems Technology, accepted for publication*.

Conradie, A. V. E. and C. Aldrich (2001). Neurocontrol of a ball mill grinding circuit using evolutionary reinforcement learning. *Minerals Engineering* **14**(10), 1277–1294.

Craig, I. K. and I. M. MacLeod (1995). Specification framework for robust control of a run-of-mine ore milling circuit. *Control Engineering Practice* **3**(5), 621–630.

Craig, I. K. and I. M. MacLeod (1996). Robust controller design and implementation for a run-of-mine ore milling circuit. *Control Engineering Practice* **4**(1), 1–12.

Craig, I. K., D. G. Hulbert, G. Metzner and S. P. Moult (1992*a*). Extended particle-size control of an industrial run-of-mine milling circuit. *Powder Technology* **73**(3), 203–210.

Craig, I. K., D. G. Hulbert, G. Metzner and S. P. Moult (1992*b*). Optimized multivariable control of an industrial run-of-mine milling circuit. *Journal of the South African Institute of Mining and Metallurgy* **92**(6), 169–176.

Cutler, C. R. and B. L. Ramaker (1980). Dynamic matrix control – A computer control algorithm. In: *Proceedings of the Joint Automatic Control Conference*. Vol. 1. San Francisco, CA.

Cuzzola, F. A., J. C. Geromel and M. Morari (2002). An improved approach for constrained robust model predictive control. *Automatica* **38**, 1183–1189.

Davila, J., L. Fridman and A. Levant (2005). Second-order sliding-mode observer for mechanical systems. *IEEE Transactions on Automatic Control* **50**(11), 1785–1789.

De Nicolao, G., L. Magni and R. Scattolini (1996). On the robustness of receding horizon control with terminal constraints. *IEEE Transactions on Automatic Control* **41**(3), 451–453.

Desbiens, A., A. Pomerleau and D. Hodouin (1996). Frequency based tuning of SISO controllers for two-by-two processes. *IEE Proceedings of Control Theory and Applications* **143**(1), 49–56.

Diehl, M., H. G. Bock and E. Kostina (2006). An approximation technique for robust nonlinear optimization. *Mathematical Programming: Series A and B* **107**(1), 213–230.

Diehl, M., H. G. Bock and J. P. Schloder (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization* **43**(5), 1714–1736.

Ding, B., Y. Xi and S. Li (2004). A synthesis approach of on-line constrained robust model predictive control. *Automatica* **40**, 163–167.

Djordjevic, N. (2005). Influence of charge size distribution on net-power draw of tumbling mill based on DEM modelling. *Minerals Engineering* **18**(3), 375–378.

Djordjevic, N., R. Morrison, B. Loveday and P. Cleary (2006). Modelling comminution patterns within a pilot scale AG/SAG mill. *Minerals Engineering* **19**(15), 1505–1516.

Dong, H. and M. H. Moys (2003). Load behavior and mill power. *International Journal of Mineral Processing* **69**(1-4), 11–28.

Duarte, M., A. Suarez and D. Bassi (1999*a*). Multivariable predictive neuronal control applied to grinding plants. In: *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials, 1999. IPMM 1999*. Vol. 2. pp. 975–982 vol.2.

Duarte, M., A. Suarez and D. Bassi (2001). Control of grinding plants using predictive multivariable neural control. *Powder Technology* **115**(2), 193–206.

Duarte, M., F. Sepulveda, A. Castillo, A. Contreras, V. Lazcano, P. Gimenez and L. Castelli (1999*b*). A comparative experimental study of five multivariable control strategies applied to a grinding plant. *Powder Technology* **104**(1), 1–28.

El-Farra, N. H. and P. D. Christofides (2003). Bounded robust control of constrained multivariable nonlinear processes. *Chemical Engineering Science* **58**(13), 3025–3047.

Freeman, R. (1995). Global internal stabilizability does not imply global external stabilizability for small sensor disturbances. *IEEE Transactions on Automatic Control* **40**(12), 2119–2122.

Galan, O., G. W. Barton and J. A. Romagnoli (2002). Robust control of a SAG mill. *Powder Technology* **124**(3), 264–271.

Gilbert, E. G. and K. T. Tan (1991). Linear systems with state and control constraints: The theory and application of maximal output admissible sets. *IEEE Transactions on Automatic Control* **36**(9), 1008–1020.

Grimm, G., M. J. Messina, S. E. Tuna and A. R. Teel (2004). Examples when nonlinear model predictive control is nonrobust. *Automatica* **40**(10), 1729–1738.

Hinde, A. (2007). Cumulative rates models for the comminution of ores in rotary mills. Private Communication. Internal Report MINTEK.

Hulbert, D. G. (1989). The state of the art in the control of milling circuits. In: *6th IFAC Symposium on Automation in Mining, Mineral and Metal Processing (Buenos Aires)*.

Hulbert, D. G. (2005). Models for mill circuit simulation. Private Communication. MINTEK.

Hull, D. G. (1997). Conversion of optimal control problems into parameter optimization problems. *Journal of Guidance, Control, and Dynamics* **20**(11), 57–60.

Jiang, Z. and Y. Wang (2001). Input-to-state stability for discrete-time nonlinear systems. *Automatica* **37**(6), 857–869.

Jiang, Z., I. M. Y. Mareels and Y. Wang (1996). A Lyapunov formulation of the nonlinear small-gain theorem for interconnected ISS systems. *Automatica* **32**(8), 1211–1215.

Kapakyulu, E. and M. H. Moys (2007). Modeling of energy loss to the environment from a grinding mill. Part I: Motivation, literature survey and pilot plant measurements. *Minerals Engineering* **20**(7), 646–652.

Karageorgos, J., P. Genovese and D. Baas (2006). Current trends in SAG and AG mill operability and control. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology, 23 – 27 September*. Vol. 3. Vancouver, B. C., Canada. pp. 191–206.

Kawajir, Y., C. Laird and A. Wachter (2006). *Introduction to Ipopt: A tutorial for downloading, installing, and using Ipopt, Revision: 799*. Carnegie Mellon University. Pittsburgh, PA, USA.

Kelly, E. G. and D. J. Spottiswood (1990). The breakage function; what is it really?. *Minerals Engineering* **3**(5), 405–414.

King, R. P. and F. Bourgeois (1993). Measurement of fracture energy during single-particle fracture. *Minerals Engineering* **6**(4), 353–367.

Kleinman, B. L. (1970). An easy way to stabilize a linear constant system. *IEEE Transactions on Automatic Control* **15**(12), 693.

Kothare, M. V., V. Balakrishnan and M. Morari (1996). Robust constrained model predictive control using linear matrix inequalities. *Automatica* **32**, 1361–1379.

Kouvaritakis, B., J. A. Rossiter and J. Schuurmans (2000). Efficient robust predictive control. *IEEE Transactions on Automatic Control* **45**(8), 145–159.

Langson, W., I. Chryssochoos, S. V. Rakovic and D. Q. Mayne (2004). Robust model predictive control using tubes. *Automatica* **40**, 125–133.

Latchireddi, S. and S. Morrell (2003*a*). Slurry flow in mills: Grate-only discharge mechanism (Part 1). *Minerals Engineering* **16**(7), 625–633.

Latchireddi, S. and S. Morrell (2003*b*). Slurry flow in mills: Grate-pulp lifter discharge systems (Part 2). *Minerals Engineering* **16**(7), 635–642.

Lazar, M., D. Munoz de la Pena, W. P. M. H. Heemels and T. Alamo (2008). On input-to-state stability of min-max nonlinear model predictive control. *Systems & Control Letters* **57**(1), 39–48.

Lee, E. B. and L. Markus (1967). *Foundations of optimal control theory*. Wiley. New York.

Lee, J. H. and Z. Yu (1997). Worst-case formulations of model predictive control for systems with bounded parameters. *Automatica* **33**(5), 763–781.

Lee, Y. I. and B. Kouvaritakis (2000). Robust receding horizon control for systems with uncertain dynamics and input saturation. *Automatica* **36**(10), 1497–1504.

Limon, D., J. M. Bravo, T. Alamo and E. F. Camacho (2005). Robust MPC of constrained nonlinear systems based on interval arithmetic. *IEE Proceedings of Control Theory and Applications* **152**(3), 325–332.

Limon, D., T. Alamo, F. Salas and E. F. Camacho (2006). Input to state stability of min-max MPC controllers for nonlinear systems with bounded uncertainties. *Automatica* **42**(5), 797–803.

Lougee-Heimer, R. (2003). The Common Optimization INterface for Operations Research. *IBM Journal of Research and Development* **47**(1), 57–66.

Loveday, B. K. and D. Naidoo (1997). Rock abrasion in autogenous milling. *Minerals Engineering* **10**(6), 603–612.

Loveday, B., R. Morrison, G. Henry and U. Naidoo (2006). An investigation of rock abrasion and breakage in a pilot-scale AG/SAG mill. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology, 23 – 27 September*. Vol. 3. Vancouver, B. C., Canada. pp. 379–388.

Lynch, A. J. (1979). *Mineral Crushing and Grinding Circuits: Their Simulation, Design, and Control (Developments in Mineral Processing Series, Vol 1)*. Elsevier Science Ltd. 335 Jan van Galenstraat, P.O. Box 211, Amsterdam, The Netherlands.

Ma, D. L. and R. D. Braatz (2001). Worst-case analysis of finite-time control policies. *IEEE Transactions on Control Systems Technology* **9**(5), 766–774.

Magni, L. and R. Sepulchre (1997). Stability margins of nonlinear receding-horizon control via inverse optimality. *Systems & Control Letters* **32**(4), 241–245.

Magni, L., D. M. Raimondo and R. Scattolini (2006). Regional input-to-state stability for nonlinear model predictive control. *IEEE Transactions on Automatic Control* **51**(9), 1548–1553.

Magni, L., G. De Nicolao, R. Scattolini and F. Allgöwer (2003). Robust model predictive control of nonlinear discrete-time systems. *International Journal of Robust and Nonlinear Control* **13**, 229–246.

Magni, L., H. Nijmeijer and A. J. van der Schaft (2001). A receding-horizon approach to the nonlinear H_∞ control problem. *Automatica* **37**(3), 429–435.

Marquez, H. J. and M. Riaz (2005). Robust state observer design with application to an industrial boiler system. *Control Engineering Practice* **13**(6), 713–728.

Mayne, D. Q., J. B. Rawlings, C. V. Rao and P. O. M. Scokaert (2000). Constrained model predictive control: Stability and optimality. *Automatica* **36**, 789–814.

McBride, A. and M. Powell (2006). A structured approach to modelling SAG mill liner wear – Numerical modelling of liner evolution. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology, 23 – 27 September*. Vol. 3. Vancouver, B. C., Canada. pp. 120–132.

Mhaskar, P. (2006). Robust model predictive control design for fault-tolerant control of process systems. *Industrial & Engineering Chemistry Research* **45**(25), 8565–8574.

Mhaskar, P. and A. B. Kennedy (2008). Robust model predictive control of nonlinear process systems: Handling rate constraints. *Chemical Engineering Science* **63**(2), 366–375.

Mhaskar, P., N. H. El-Farra and P. D. Christofides (2005). Robust hybrid predictive control of nonlinear systems. *Automatica* **41**(2), 209–217.

Mhaskar, P., N. H. El-Farra and P. D. Christofides (2006). Stabilization of nonlinear systems with state and control constraints using Lyapunov-based predictive control. *Systems & Control Letters* **55**(8), 650–659.

Michalska, H. and D. Q. Mayne (1993). Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control* **38**(11), 1623–1633.

Michalska, H. and D. Q. Mayne (1995). Moving horizon observers and observer-based control. *IEEE Transactions on Automatic Control* **40**(6), 995–1006.

Mishra, B. K. (2003*a*). A review of computer simulation of tumbling mills by the discrete element method: Part I – Contact mechanics. *International Journal of Mineral Processing* **71**(1–4), 73–93.

Mishra, B. K. (2003*b*). A review of computer simulation of tumbling mills by the discrete element method Part II – Practical applications. *International Journal of Mineral Processing* **71**(1–4), 95–112.

Morilla, F., F. Vázquez and J. Garrido (2008). Centralized PID control by decoupling for TITO processes. In: *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation*. pp. 1318–1325.

Morrell, S. (1996). Power draw of wet tumbling mills and its relationship to charge dynamics – Part 1: A continuum approach to mathematical modelling of mill power draw. *Transactions of the Institution of Mining and Metallurgy Section C* **105**, C43—C53.

Morrell, S. (2004*a*). A new autogenous and semi-autogenous mill model for scale-up, design and optimisation. *Minerals Engineering* **17**(3), 437–445.

Morrell, S. (2004*b*). Predicting the specific energy of autogenous and semi-autogenous mills from small diameter drill core samples. *Minerals Engineering* **17**(3), 447–451.

Morrell, S. and I. Stephenson (1996). Slurry discharge capacity of autogenous and semi-autogenous mills and the effect of grate design. *International Journal of Mineral Processing* **46**(1–2), 53–72.

Morrell, S. and R. D. Morrison (1996). AG and SAG mill circuit selection and design by simulation. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology, 6-9 October*. Vol. 2. Vancouver, B. C., Canada. pp. 769–790.

Morrison, R., B. Loveday, M. Powell, N. Djordjevic and P. Cleary (2006). Applying Discrete Element Modelling to different modes of breakage in AG and SAG Mills. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology, 23 – 27 September*. Vol. 3. Vancouver, B. C., Canada. pp. 407–420.

Morrison, R. D. and P. W. Cleary (2004). Using DEM to model ore breakage within a pilot scale SAG mill. *Minerals Engineering* **17**(11–12), 1117–1124.

Morrison, R. D., F. Shi and R. Whyte (2007). Modelling of incremental rock breakage by impact – For use in DEM models. *Minerals Engineering* **20**(3), 303–309.

Nageswararao, K., D. M. Wiseman and T. J. Napier-Munn (2004). Two empirical hydrocyclone models revisited. *Minerals Engineering* **17**(5), 671–687.

Najim, K., D. Hodouin and A. Desbiens (1995). Adaptive control: State of the art and an application to a grinding process. *Powder Technology* **82**(1), 59–68.

Napier-Munn, T. J. and B. A. Wills (2006). *Wills' Mineral Processing Technology, Seventh Edition: An Introduction to the Practical Aspects of Ore Treatment and Mineral Recovery*. Butterworth-Heinemann. Linacre House, Jordan Hill, Oxford OX2 8DP, UK.

Napier-Munn, T. J., S. Morrell, R. D. Morrison and T. Kojovic (1996). Mineral comminution circuits their operation and optimisation. JKMRC Monograph Series.

Narayanan, S. S. (1987). Modelling the performance of industrial ball mills using single particle breakage data. *International Journal of Mineral Processing* **20**(3–4), 211–228.

Narayanan, S. S. and W. J. Whiten (1988). Determination of comminution characteristics from single particle breakage tests and its application to ball mill scale-up. *Transactions of the Institution of Mining and Metallurgy* **97**, C115–C124.

Neesse, T., V. Golyk, P. Kaniut and V. Reinsch (2004). Hydrocyclone control in grinding circuits. *Minerals Engineering* **17**(11–12), 1237–1240.

Pannocchia, G. (2004). Robust model predictive control with guaranteed setpoint tracking. *Journal of Process Control* **14**, 927–937.

Pannocchia, G. and E. C. Kerrigan (2003). Offset-free receding horizon control of constrained linear systems subject to time-varying setpoints and persistent unmeasured disturbances. Technical report. Department of Engineering, University of Cambridge. Cambridge, UK. CUED/F-INFENG/TR.468.

Pannocchia, G. and E. C. Kerrigan (2005). Offset-free receding horizon control of constrained linear systems. *AIChE Journal* **51**(12), 3134–3146.

Peng, H., Z. Yang, W. Gui, M. Wu, H. Shioya and K. Nakano (2007). Nonlinear system modeling and robust predictive control based on RBF-ARX model. *Engineering Applications of Artificial Intelligence* **20**(1), 1–9.

Peterka, V. (1984). Predictor-based self tuning control. *Automatica* **20**(1), 39–50.

Pomerleau, A., D. Hodouin, A. Desbiens and E. Gagnon (2000). A survey of grinding circuit control methods: From decentralized PID controllers to multivariable predictive controllers. *Powder Technology* **108**(2–3), 103–115.

Powell, M. S. and A. T. McBride (2004). A three-dimensional analysis of media motion and grinding regions in mills. *Minerals Engineering* **17**(11–12), 1099–1109.

Powell, M. S. and A. T. McBride (2006). What is required from DEM simulations to model breakage in mills?. *Minerals Engineering* **19**(10), 1013–1021.

Qin, S. J. and T. A. Badgwell (2003). A survey of industrial model predictive control technology. *Control Engineering Practice* **11**, 733–764.

Radhakrishnan, V. R. (1999). Model based supervisory control of a ball mill grinding circuit. *Journal of Process Control* **9**(3), 195–211.

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

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

Ramasamy, M., S. S. Narayanan and C. D. P. Rao (2005). Control of ball mill grinding circuit using model predictive control scheme. *Journal of Process Control* **15**(3), 273–283.

Rao, C. V., J. B. Rawlings and D. Q. Mayne (2003). Constrained state estimation for nonlinear discrete-time systems: stability and moving horizon approximations. *IEEE Transactions on Automatic Control* **48**(2), 246–258.

Åström, K. J. (2002). *Control System Design*. Department of Mechanical and Environmental Engineering, University of California. Santa Barbara.

Åström, K. J. and P. Eykhoff (1971). System identification – A survey. *Automatica* **7**(2), 123–162.

Rawlings, J. B. and K. R. Muske (1993). Stability of constrained receding horizon control. *Transactions on Automatic Control* **38**(10), 1512–1516.

Richalet, J., A. Rault, J. L. Testud and J. Papon (1978). Model predictive heuristic control: Applications to industrial processes. *Automatica* **14**(5), 413–428.

Rossiter, J. A., B. Kouvaritakis and M. J. Rice (1998). A numerically robust state-space approach to stable-predictive control strategies. *Automatica* **34**(1), 65–73.

Sbarbaro, D., J. Barriga, H. Valenzuela and G. Cortes (2005). A multi-input-single-output Smith predictor for feeders control in SAG grinding plants. *IEEE Transactions on Control Systems Technology* **13**(6), 1069–1075.

Schuurmans, J. and J. A. Rossiter (2000). Robust model predictive control using tight sets of predicted states. In: *IEE Proceedings for Control Theory and Applications*. Vol. 147. pp. 13–18.

Scokaert, P. O. M. and D. Q. Mayne (1998). Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic Control* **43**(8), 1136–1142.

Scokaert, P. O. M., D. Q. Mayne and J. B. Rawlings (1999). Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control* **44**(3), 648–654.

Shi, F. and T. Kojovic (2007). Validation of a model for impact breakage incorporating particle size effect. *International Journal of Mineral Processing* **82**(3), 156–163.

Shi, F. N. and T. J. Napier-Munn (1996). A model for slurry rheology. *International Journal of Mineral Processing* **47**(1–2), 103–123.

Shi, F. N. and T. J. Napier-Munn (2002). Effects of slurry rheology on industrial grinding performance. *International Journal of Mineral Processing* **65**(3–4), 125–140.

Skogestad, S. (2003). Simple analytic rules for model reduction and PID controller tuning. *Journal of Process Control* **13**(4), 291–309.

Smith, V. C., D. G. Hulbert and A. Singh (2001). AG/SAG control and optimization with PlantStar 2000. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology held September 30 – October 3*. Vol. II. Vancouver, B. C., Canada. pp. 282–293.

Sontag, E. D. (1989). Smooth stabilization implies coprime factorization. *IEEE Transactions on Automatic Control* **34**(4), 435–443.

Sontag, E. D. (1990). Further facts about Input to State Stabilization. *IEEE Transactions on Automatic Control* **35**(4), 473–476.

Sontag, E. D.; Yuan Wang (1996). New characterizations of Input-to-State Stability. *IEEE Transactions on Automatic Control* **41**(9), 1283–1294.

Spurgeon, S. K. (2008). Sliding mode observers: a survey. *International Journal of Systems Science* **39**(8), 751–764.

Stanley, G. G. (1987). The extractive metallurgy of gold in South Africa. Technical Report Vol 1. South African Institute of Mining and Metallurgy. Johannesburg.

Tavares, L. M. and R. P. King (1998). Single-particle fracture under impact loading. *International Journal of Mineral Processing* **54**(1), 1–28.

Valenzuela, J., M. Bourassa, K. Najim and R. Del Villar (1994). Dynamic matrix control of an autogenous grinding circuit. *Minerals Engineering* **7**(1), 105–114.

Van Drunick, W. I. and B. P. N. Penny (2006). Expert mill control at Anglogold Ashanti. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology, 23 – 27 September*. Vol. 3. Vancouver, B. C., Canada. pp. 266–281.

Vázquez, F. and F. Morilla (2002). Tuning decentralized PID controllers for MIMO systems with decouplers. In: *Proceedings of the 15th Triennial IFAC World Congress, Barcelona, Spain*. pp. 349–354.

Vogel, L. and W. Peukert (2003). Breakage behaviour of different materials – Construction of a mastercurve for the breakage probability. *Powder Technology* **129**(1–3), 101–110.

Vogel, L. and W. Peukert (2005). From single particle impact behaviour to modelling of impact mills. *Chemical Engineering Science* **60**(18), 5164–5176.

Wan, Z. and M. V. Kothare (2003). An efficient off-line formulation of robust model predictive control using linear matrix inequalities. *Automatica* **39**, 837–846.

Wang, Y. J. and J. B. Rawlings (2004*a*). A new robust model predictive control method I: Theory and computation. *Journal of Process Control* **14**, 231–247.

Wang, Y. J. and J. B. Rawlings (2004*b*). A new robust model predictive control method II: Examples. *Journal of Process Control* **14**(3), 249–262.

Wei, D. and I. K. Craig (2009). Grinding mill circuits – A survey of control and economic concerns. *International Journal of Mineral Processing* **90**(1-4), 56–66.

Whiten, W. J. (1974). A matrix theory of comminution machines. *Chemical Engineering Science* **29**(2), 589–599.

Wills, B. A. and T. J. Napier-Munn (2006). *Wills' Mineral Processing Technology: An introduction to the practical aspects of ore treatment and mineral recovery*. 7^{th} ed.. Elsevier, Butterworth-Heinemann. Linacre House, Jordan Hill, Oxford OX2 8DP, UK.

Xingyan, G., M. Zhizhong and W. Jian (1992). A predictive adaptive controller and its application to the control of the wet autogenous mill. In: *Proceedings of the IEEE International Symposium on Industrial Electronics, 1992.*. pp. 143–145 vol.1.

Xiong, Y. and M. Saif (2001). Sliding mode observer for nonlinear uncertain systems. *IEEE Transactions on Automatic Control* **46**(12), 2012–2017.

Yashima, S., Y. Kanda and S. Sano (1987). Relationships between particle size and fracture energy or impact velocity required to fracture as estimated from single particle crushing. *Powder Technology* **51**(3), 277–282.

Zheng, Z. Q. and M. Morari (1995). Stability of model predictive control with mixed constraints. *IEEE Transactions on Automatic Control* **40**(10), 1818–1823.

ADDENDUM A

SOFTWARE IMPLEMENTATION

A.1 INTRODUCTION

The robust nonlinear model predictive controller and nonlinear model predictive controller were implemented in C++ using open-source packages.

The first open-source package named IPOPT (Interior Point Optimiser, pronounced "I-P-Opt") (Kawajir *et al.*, 2006) does large-scale nonlinear parameter optimisation. It solves general nonlinear programming problems of the form

$$\min_{x\in\mathbb{R}^n} \quad f(x) \tag{A.1}$$

$$\text{s.t.} \quad g^L \leq g(x) \leq g^U \tag{A.2}$$

$$x^L \leq x \leq x^U \tag{A.3}$$

where $x \in \mathbb{R}^n$ are the optimisation variables (possibly with lower and upper bounds, $x^L \in (\mathbb{R}\cup\{-\infty\})^n$ and $x^U \in (\mathbb{R}\cup\{+\infty\})^n$), $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, and $g : \mathbb{R}^n \to \mathbb{R}^m$ are the general nonlinear constraints. The functions $f(x)$ and $g(x)$ can be linear or nonlinear and convex or non-convex, but should be twice continuously differentiable. The constraints have lower and upper bounds, $g^L \in (\mathbb{R}\cup\{-\infty\})^m$ and $g^U \in (\mathbb{R}\cup\{+\infty\})^m$. Equality constraints of the form $g_i(x) = \bar{g}_i$ can be specified by setting $g_i^L = g_i^U = \bar{g}_i$.

The optimisation problem (A.1)–(A.3) is rewritten in terms of the functions specified in Section (3.7.3). The nonlinear programming problem then becomes

$$\min_{X\in\mathbb{R}^{n_X}} \quad \varphi(X) \tag{A.4}$$

$$\text{s.t.} \quad g^L \leq \vartheta(x) \leq g^U \tag{A.5}$$

$$x^L \leq X \leq x^U \tag{A.6}$$

where $n_X \triangleq (n_x + n_u + n_D + n_{slack}) \cdot (N) + n_x + n_D$.

The second open-source package, named CPPAD (Lougee-Heimer, 2003), does automatic differentiation. Two classical ways of evaluating derivatives of functions on computers are to do symbolic differentiation or finite differences. Symbolic differentiation suffers from slow execution and the difficulty of converting a computer program to a single expression to be evaluated. Finite differences suffer from rounding errors in the discretisation process and cancellation. Automatic differentiation overcomes the mentioned problems by exploiting the fact that a computer program implementing the vector function $y = F(x)$ can generally be decomposed into a sequence of elementary assignments. Any one of the elementary assignments can be trivially differentiated by using a simple lookup table. The derivative can be constructed by applying the chain rule of differentiation to the elemental partial derivatives. This process yields exact (to numerical accuracy) derivatives.

A.2 IMPLEMENTATION

IPOPT requires some information about the nonlinear programming problem that it needs to solve. IPOPT provides a class *TNLP* that needs to be overridden to provide the nonlinear programming problem. The following information is required by IPOPT (Kawajir *et al.*, 2006) and member functions that need to be overridden and the return variables are provided in brackets:

1. Problem dimensions
    - Number of variables (get_nlp_info: Index &n)
    - Number of constraints (get_nlp_info: Index &m)
2. Problem bounds
    - Variable bounds (get_bounds_info: Number *x_l, Number *x_u)
    - Constraint bounds (get_bounds_info: Number *g_l, Number *g_u)
3. Initial starting point
    - Initial values for the primal X variables (get_starting_point: Number *x)
4. Problem structure
    - Number of non-zeros in the Jacobian of the constraints (get_nlp_info : Index &nnz_jac_g)
    - Sparsity structure of the Jacobian of the constraints (eval_jac_g : Index *iRow, Index *jCol)

5. Evaluation of problem functions

   - Objective function, $\varphi(X)$ (eval_f: Number &obj_value)
   - Gradient of the objective function, $\nabla\varphi(X)$ (eval_grad_f: Number *grad_f)
   - Constraint function values, $\vartheta(X)$ (eval_g: Number *g)
   - Jacobian of the constraints, $\nabla\vartheta(X)^T$ (eval_jac_g: Number *values)

The RNMPC is defined in Section 3.7. The basic definitions are:

- n_x — the number of states
- n_u — the number of manipulated variables
- n_c — the number of constraints
- n_p — the number of uncertain parameters
- T — the prediction horizon
- τ_s — the sampling time in seconds
- N — the number of steps over the prediction horizon, defined as $N = {}^{T}/_{\tau_s}$
- n_D — the number of entries in the $\mathbf{D}_i$ matrix, defined as $n_D \triangleq n_x \times n_p$
- n_{slack} — the number of slack variables, defined as $n_{slack} \triangleq (n_c + 1) \times n_p$

A.2.1 IPOPT TNLP class methods

A.2.1.1 Class constructor method `MyNLP::MyNLP`

This constructor method is called when a new instance of the class is created. This method does the following initialisations:

- Calculates the dimensions of X of equation (3.100), $\nabla\varphi(X)$ of equation (3.102), $\vartheta(X)$ of equation (3.103) and $\nabla\vartheta(X)$ of equation (3.107).
- Threading memory is allocated for the number of threads corresponding to the number of nodes. This memory is allocated whether threading is used or not. This allows all the nodes to be processed simultaneously on multi-processor and multi-core systems when threading is enabled.
- If threading is used, sets up the thread mutexes and signals.

- Allocates memory for the global arrays **`m_cost`**, **`m_costGrad`**, **`m_g`** and **`m_gJac`** and some arrays for storing the states, control moves, states setpoint, control moves setpoint, average control moves and optimisation variables needed for *warm startup*.
- Assigns the initial values for the states and control moves of X to **`m_xu`** and sets the entries corresponding to the **D**-matrix and slack variables (δ) to 0.0.
- Assigns some general values (such as the states setpoint, control moves setpoint, states and control moves weightings) to the thread memory of each node.
- If threading is not used, calls **`CppADInit`** for each node in order to initialise the CPPAD function objects and allocates memory for some miscellaneous variables required by that node in its associated thread memory.
- If threading is used, this method then creates the actual threads, with the same number of threads as the number of nodes.

A.2.1.2 Method `get_nlp_info`

The method **`get_nlp_info`** returns the following information on the nonlinear programming problem:

- **`n`**: (out), the number of variables in the problem (dimension of X) — For the RNMPC it is the dimension of X defined as $(n_x+n_u+n_D+n_{slack})\cdot N+n_x+n_D$.
- **`m`**: (out), the number of constraints in the problem (dimension of $\vartheta(X)$) — For the RNMPC it is defined as $(n_x+n_D+n_c+2\cdot n_{slack})\cdot N+n_x+n_D$.
- **`nnz_jac_g`**: (out), the number of nonzero entries in the Jacobian of the constraints (nonzero entries of $\nabla\vartheta(X)^T$) — For the RNMPC it is defined as $N(n_x+n_x(n_x+n_u))+N(n_D(2n_x+n_u+1))+N(n_c(n_x+2n_u+N))+2N(n_{slack}(2n_x+2n_u+1))+n_x+n_D-n_c\cdot n_u-2\cdot n_{slack}\cdot n_u$.
- **`nnz_h_lag`**: (out), the number of nonzero entries in the Hessian of the Lagrangian — This is not used by the RNMPC, because the quasi-Newton approximation of the Hessian is used.
- **`index_style`**: (out), the numbering style used in the row/column entries in the sparse matrix format (**`C_STYLE`**: indexes start at 0 is used in RNMPC, **`FORTRAN_STYLE`**: indexes start at 1).

All the values returned by this function are calculated in the class constructor. Only the calculated values are returned.

A.2.1.3 Method `get_bounds_info`

- **`n`**: (in), The number of variables in the problem (dimension of X).
- **`x_l`**: (out), The lower bounds x^L for X. For the RNMPC there is no lower bound on any of the variables and the lower bounds are therefore set to $-1.0e19$, which is similar to $-\infty$ in the program.
- **`x`**_u: (out), The upper bounds x^U for X. For the RNMPC there is no upper bound on any of the variables and the upper bounds are therefore set to $1.0e19$, which is similar to $+\infty$ in the program.
- **`m`**: (in), The number of constraints in the problem (dimension of $\vartheta(X)$).
- **`g_l`**: (out), The lower bounds g^L for $\vartheta(X)$. See details below for the RNMPC implementation.
- **`g`**_u: (out), The upper bounds g^U for $\vartheta(X)$. See details below for the RNMPC implementation.

The bounds on the constraints depend on whether it is an equality or an inequality constraint. Inequality constraints are all transformed to have only an upper bound of 0.0, except the second block of slack variables that only have a lower bound of 0.0. Equality constraints are all transformed to be equal to 0 by setting the lower and upper bound equal to 0. The constraints consist of N number of blocks with $n_x + n_D + n_c + 2 \cdot n_{slack}$ number of entries in each block. The last block only has $n_x + n_D$ number of entries. The nonlinear programming problem is described in (3.88)-(3.94). The upper and lower bounds for each block are as follows:

Constraint Function	Number of Entries	g^L	g^U
System Dynamics	n_x	0.0	0.0
Calculating **D**-matrix	n_D	0.0	0.0
User-defined state and input constraints on the system	n_c	$-1.0e19$	0.0
Slack variables for approximating the dual norms of the robust weighting terms	n_{slack}	$-1.0e19$	0.0
Slack variables for approximating the dual norms of the robust weighting terms	n_{slack}	0.0	$1.0e19$

A.2.1.4 Method `get_starting_point`

- **`n`**: (in), The number of variables in the problem (dimension of X).
- **`init_x`**: (in), if true, this method should provide an initial value for X.
- **`x`**: (out), The initial values for the primal variables. See below for details pertaining to the RNMPC.
- **`init_z`**: (in), if true, this method must provide an initial value for the bound multipliers z^L and z^U.
- **`z_L`**: (out), The initial values for the bound multipliers, z^L.
- **`z_U`**: (out), The initial values for the bound multipliers, z^U.
- **`m`**: (in), The number of constraints in the problem (dimension of $\vartheta(X)$).
- **`init_lambda`**: (in), if true, this method must provide an initial value for the constraint multipliers, λ.
- **`lambda`**: (out), The initial values for the constraint multipliers, λ.

The RNMPC initialises X with the initial state of the plant and the initial control moves and all the auxiliary variables to 0.0. The vector X consists of N blocks that each contains $n_x + n_u + n_D + n_{slack}$ variables. The last block contains only $n_x + n_D$ number of variables. The initial values for each block is given by:

Variables	Number of Entries	X
System state variables	n_x	Initial state
Control moves	n_u	Initial control moves
The **D**-matrix	n_D	0.0
Slack variables for approximating the dual norms of robust weighting terms	n_{slack}	0.0

These values should be returned when **`init_x`** is true. The initial values are calculated in the class constructor and the results returned when this method is called.

The values for **`z_l`**, **`z_u`** and **`lambda`** will only be requested if a *warm startup* is chosen. A *warm startup* can be used in RNMPC to initialise the system to the optimal solution of the previous time-step, which should be close to the optimal solution for the current time-step and therefore reduce the computational time needed to find the optimal solution for the current time-step. The values of **`z_l`**, **`z_u`** and **`lambda`** can be obtained when the optimisation of the previous time-step finishes and returns to the optimiser at the start of the current time-step using this method.

A.2.1.5 Method `eval_f`

This method returns the objective function value at the point X. For the RNMPC, this is the objective function of the controller.

- **`n`**: (in), The number of variables in the problem (dimension of X).
- **`x`**: (in), The values for the primal variables X, at which $\varphi(X)$ should be evaluated.
- **`new_x`**: (in), False if any of the other evaluation functions have been called with the same X values.
- **`obj_value`**: (out), The value of the objective function $\varphi(X)$.

This method calls **`sundials_run`** to calculate the objective function ($\varphi(X)$), the gradient of the objective function ($\nabla\varphi(X)$), the constraint function ($\vartheta(X)$) and the Jacobian of the constraints ($\nabla\vartheta(X)^T$). The method **`sundials_run`** only calculates a new solution for the functions if **`new_x`** is true, otherwise it only returns the previously calculated solution.

A.2.1.6 Method `eval_grad_f`

This method returns the gradient of the objective function at the point X, $\nabla\varphi(X)$.

- **`n`**: (in), The number of variables in the problem (dimension of X).
- **`x`**: (in), The values for the primal variables X, at which $\nabla\varphi(X)$ should be evaluated.
- **`new_x`**: (in), False if any of the other evaluation functions have been called with the same X values.
- **`grad_f`**: (out), The array containing the gradient of the objective function, $\nabla\varphi(X)$.

This method calls **`sundials_run`** to calculate the objective function ($\varphi(X)$), the gradient of the objective function ($\nabla\varphi(X)$), the constraint function ($\vartheta(X)$) and the Jacobian of the constraints ($\nabla\vartheta(X)^T$). The method **`sundials_run`** only calculates a new solution for the functions if **`new_x`** is true, otherwise it only returns the previously calculated solution.

A.2.1.7 Method `eval_g`

This method returns the value of the constraint function at the point X, $\vartheta(X)$.

- **`n`**: (in), The number of variables in the problem (dimension of X).
- **`x`**: (in), The values for the primal variables X, at which $\vartheta(X)$ should be evaluated.

- **new_x**: (in), False if any of the other evaluation functions have been called with the same X values.
- **m**: (in), The number of constraints in the problem, which is the dimension of $\vartheta(X)$.
- **g**: (out), The array containing the values of the constraint functions, $\vartheta(X)$.

This method calls **sundials_run** to calculate the objective function ($\varphi(X)$), the gradient of the objective function ($\nabla\varphi(X)$), the constraint function ($\vartheta(X)$) and the Jacobian of the constraints ($\nabla\vartheta(X)^T$). The method **sundials_run** only calculates a new solution for the functions if **new_x** is true, otherwise it only returns the previously calculated solution.

A.2.1.8 Method eval_jac_g

This method returns either the structure of the Jacobian of the constraints, or the values for the Jacobian of the constraints evaluated at point X.

- **n**: (in), The number of variables in the problem (dimension of X).
- **x**: (in), The values for the primal variables X, at which point $\nabla\vartheta(X)^T$ should be evaluated.
- **new_x**: (in), False if any of the other evaluation functions have been called with the same X values.
- **m**: (in), The number of constraints in the problem, which is the dimension of $\vartheta(X)$.
- **n_ele_jac**: (in), The number of nonzero elements in the Jacobian, which is the dimension of **iRow**, **jCol** and **values**.
- **iRow**: (out), The array containing the row indexes of the entries in the Jacobian of the constraints.
- **jCol**: (out), The array containing the column indexes of the entries in the Jacobian of the constraints.
- **values**: (out), The array containing the values of the entries in the Jacobian of the constraints.

If **values** is a NULL pointer, the structure of the Jacobian of the constraints is required. There are N blocks in the Jacobian of the constraints that look like Figure (3.1). The block in Figure (3.1) can be further subdivided into smaller blocks. The sub-blocks are populated using row major representation in the array, meaning that the column entries in the same row follow continuously in the array. The sparse structures of the sub-blocks are defined in the block structure in the order specified in Table A.1.

Table A.1: The definition of the sparse structure of the sub-blocks in the Jacobian of the constraints.

$\vartheta(X)$	X	Block Calculation	Block Size	Block Type	No Entries
States (s_i)	States (s_i)	$\frac{\partial s_i}{\partial s_i}$	$n_x \times n_x$	Diagonal	n_x
$\mathbf{D}_i$-matrix values	$\mathbf{D}_i$-matrix values	$\frac{\partial D_i}{\partial D_i}$	$n_D \times n_D$	Diagonal	n_D
States (s_{i+1})	States (s_i) and Inputs (q_i)	$\frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i}, \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial q_i}$	$n_x \times n_x \cdot n_u$	Rectangular	$n_x \times n_x \cdot n_u$
$\mathbf{D}_{i+1}$-matrix	States (s_i) and Inputs (q_i)	$\left(\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial s_i} + \frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial p}}{\partial s_i} \right), \left(\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial q_i} + \frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial p}}{\partial q_i} \right)$	$n_D \times n_x \cdot n_u$	Rectangular	$n_D \times n_x \cdot n_u$
$\mathbf{D}_{i+1}$-matrix	$\mathbf{D}_i$-matrix values	$\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial D_i}$	$n_D \times n_D$	Rectangular	$n_D \times n_x$
Inequality Constraints $(\theta_{1,i}, \ldots, \theta_{j,i}, \ldots \theta_{n_c,i})$	States (s_i) and Inputs (q_i)	$\frac{\partial \theta_{j,i}(s_i,q_i)}{\partial s_i}, \frac{\partial \theta_{j,i}(s_i,q_i)}{\partial q_i}$	$n_c \times n_x \cdot n_u$	Rectangular	$n_c \times n_x \cdot n_u$
Inequality Constraints $(\theta_{1,i}, \ldots, \theta_{j,i}, \ldots \theta_{n_c,i})$	Slack variables ($\delta_{j,i}$)	$\frac{\partial \theta_{j,i}(s_i,q_i) + e^T \delta_{j,i}}{\delta_{j,i}}$	$n_c \times n_{slack}$	Banded	$n_c \times n_p$
Slack variables $(\delta_{0,i}, \ldots, \delta_{j,i}, \ldots, \delta_{n_c,i})$ — Block 1 & 2	States (s_i) and Inputs (q_i)	$\frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,j,i}}{\partial s_i}(s_i,q_i) \right)^T}{\partial s_i}, \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{j,i}}{\partial s_i}(s_i,q_i) \right)^T}{\partial q_i}$	$n_{slack} \times n_x \cdot n_u$	Rectangular	$n_{slack} \times n_x \cdot n_u$
Slack variables $(\delta_{0,i}, \ldots, \delta_{j,i}, \ldots, \delta_{n_c,i})$ — Block 1 & 2	$\mathbf{D}_{i+1}$-matrix values	$\frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i) \right)^T}{\partial D_{i+1}}$	$n_{slack} \times n_x$	Rectangular	$n_{slack} \times n_x$
Slack variables $(\delta_{0,i}, \ldots, \delta_{j,i}, \ldots, \delta_{n_c,i})$ — Block 1 & 2	Slack variables ($\delta_{j,i}$)	$\frac{\partial \delta_{j,i}}{\partial \delta_{j,i}}$	$n_{slack} \times n_{slack}$	Diagonal	n_{slack}

If **values** is not a NULL pointer then the values of the Jacobian of the constraints are required and this method calls **sundials_run** to calculate the objective function ($\varphi(X)$), the gradient of the objective function ($\nabla\varphi(X)$), the constraint function ($\vartheta(X)$) and the Jacobian of the constraints ($\nabla\vartheta(X)^T$). The method **sundials_run** only calculates new a solution for the functions if **new_x** is true, otherwise only returns the previously calculated solution.

A.2.1.9 Method `finalize_solution`

This method returns the solution of the nonlinear programming problem.

- **status**: (in), Gives the final status of the solution such as SUCCESS or some failure.
- **n**: (in), The number of variables in the problem (dimension of X).
- **x**: (in), The final values for the primal variables X.
- **z_L**: (in), The final values for the bound multipliers, z^L.
- **z_U**: (in), The final values for the bound multipliers, z^U.
- **m**: (in), The number of constraints in the problem (dimension of $\vartheta(X)$).
- **lambda**: (in), The final values for the constraint multipliers, λ.
- **obj_value**: (in), The final value of the objective function $\varphi(X)$.

The values of **x** are saved for the next time-step. The values for **z_L**, **z_U** and **lambda** can be saved to be used in a *warm startup* of the next time-step.

A.2.2 RNMPC specific methods

These methods are not part of the base TNLP class, but are added to do the required calculations specifically geared towards RNMPC implementation.

A.2.2.1 Method `next_run`

This method sets up the optimisation problem for the next time-steps. Most of the initialisation is done in the class constructor and remains valid for all time-step. Only a few values need to change between iterations and this method makes it possible.

- **xSetpoint**: (in), Gives the setpoint that the controller should follow in this iteration. The setpoint can only be changed between iterations, not during an iteration.
- **xMeasured**: (in), The actual state of the process being controlled as measured or estimated online.

- **uPrevious**: (in), Provides the control moves of the previous time-step if $\Delta u(t)$ control is used rather than absolute control $u(t)$.
- **uAverage**: (out), The average control moves over a period of time.
- **objective**: (out), The objective value of the previous time-step.
- **Contraction**: (out), The amount the constraints are tightened to provide robustness.

A.2.2.2 Method get_u

This method returns the first control moves u_0^{opt}, which are part of the solution of the nonlinear programming problem.

- **u**: (out), The first control moves u_0^{opt} from the solution of the nonlinear programming problem.

A.2.2.3 Method SundialsRun

This method is called by **eval_f**, **eval_grad_f**, **eval_g** and **eval_jac_g**, to calculate the objective function ($\varphi(X)$), the gradient of the objective function ($\nabla\varphi(X)$), the constraint function ($\vartheta(X)$) and the Jacobian of the constraint function ($\nabla\vartheta(X)$). This method starts by calculating $x_0 - s_0$ and $-D_0$ of (3.104).

This method then sets up the threads to calculate $\varphi(X)$, $\nabla\varphi(X)$,$\vartheta(X)$ and $\nabla\vartheta(X)^T$ for each node by assigning the relevant pointers for the node and instructing the threads to execute. The threads can execute in parallel on multi-core or multi-processor systems, allowing the controller to perform faster on systems with more processors and/or processor cores. The method then waits for the threads to finish before continuing. An example of assigning pointers from the global arrays for the thread solving node 1 is shown in Figure (A.1) and Figure (A.2). The constraint function $\vartheta(X)$ can be subdivided into $N+1$ blocks (3.104)-

(3.106), with the calculations in the block (3.104) defined as:

$$G_{s_0} \triangleq x_k - s_0 \tag{A.7}$$

$$G_{D_0} \triangleq -I_{n_x} D_0 + 0 \tag{A.8}$$

$$G_{c_0} \triangleq \begin{cases} \theta_{1,0}(s_0, q_0) + e^T \delta_{1,0} \\ \vdots \\ \theta_{n_c,0}(s_0, q_0) + e^T \delta_{n_c,0} \end{cases} \tag{A.9}$$

$$G_{\delta 1_0} \triangleq \begin{cases} D_1^T \left(\frac{\partial L_0}{\partial s_0}(s_0, q_0) \right)^T - \delta_{0,0} \\ D_1^T \left(\frac{\partial \theta_{1,0}}{\partial s_0}(s_0, q_0) \right)^T - \delta_{1,0} \\ \vdots \\ D_1^T \left(\frac{\partial \theta_{n_c,0}}{\partial s_0}(s_0, q_0) \right)^T - \delta_{n_c,0} \end{cases} \tag{A.10}$$

$$G_{\delta 2_0} \triangleq \begin{cases} D_1^T \left(\frac{\partial L_0}{\partial s_0}(s_0, q_0) \right)^T + \delta_{0,0} \\ D_1^T \left(\frac{\partial \theta_{1,0}}{\partial s_0}(s_0, q_0) \right)^T + \delta_{1,0} \\ \vdots \\ D_1^T \left(\frac{\partial \theta_{n_c,0}}{\partial s_0}(s_0, q_0) \right)^T + \delta_{n_c,0} \end{cases} \tag{A.11}$$

the calculations in the blocks (3.105),$i = 1, \ldots, N-1$ defined as

$$G_{s_i} \triangleq f_{i-1}(s_{i-1}, q_{i-1}, \bar{p}) - s_i \tag{A.12}$$

$$G_{D_i} \triangleq \frac{\partial f_{i-1}(s_{i-1}, q_{i-1}, \bar{p})}{\partial s_{i-1}} \cdot D_{i-1} - I_{n_x} \cdot D_i + \frac{\partial f_{i-1}(s_{i-1}, q_{i-1}, \bar{p})}{\partial p} \tag{A.13}$$

$$G_{c_i} \triangleq \begin{cases} \theta_{1,i}(s_i, q_i) + e^T \delta_{1,i} \\ \vdots \\ \theta_{n_c,i}(s_i, q_i) + e^T \delta_{n_c,i} \end{cases} \tag{A.14}$$

$$G_{\delta 1_i} \triangleq \begin{cases} D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i, q_i) \right)^T - \delta_{0,i} \\ D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i, q_i) \right)^T - \delta_{1,i} \\ \vdots \\ D_{i+1}^T \left(\frac{\partial \theta_{n_c,i}}{\partial s_i}(s_i, q_i) \right)^T - \delta_{n_c,i} \end{cases} \tag{A.15}$$

$$G_{\delta 2_i} \triangleq \begin{cases} D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i, q_i) \right)^T + \delta_{0,i} \\ D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i, q_i) \right)^T + \delta_{1,i} \\ \vdots \\ D_{i+1}^T \left(\frac{\partial \theta_{n_c,i}}{\partial s_i}(s_i, q_i) \right)^T + \delta_{n_c,i} \end{cases} \tag{A.16}$$

and the calculations in the block (3.106) defined as

$$G_{s_N} \triangleq f_{N-1}(s_{N-1}, q_{N-1}, \bar{p}) - s_N \tag{A.17}$$

$$G_{D_N} \triangleq \frac{\partial f_{N-1}(s_{N-1}, q_{N-1}, \bar{p})}{\partial s_{N-1}} \cdot D_{N-1} - I_{n_x} \cdot D_N + \frac{\partial f_{N-1}(s_{N-1}, q_{N-1}, \bar{p})}{\partial p} \tag{A.18}$$

The sub-blocks in Figure (A.2) for $\nabla\vartheta(X)$ are stored sequentially in **`m_gJac`** in the following order

- **`m_eqJacEye`**,
- **`m_DJacEye`**,
- **`m_eqJacXU`**,
- **`m_DJacXU`**,
- **`m_DJacD`**,
- **`m_constraintsJacXU`**,
- **`m_constraintsJacSlack`**,
- **`m_slackJacXU`**,
- **`m_slackJacD`**,
- **`m_slackJacSlack`**,
- **`m_slackJacXU2`**,
- **`m_slackJacD2`**,
- **`m_slackJacSlack2`**.

After the threads have finished executing, the method assigns values to the final sub-blocks

$$\nabla f_N(x) \triangleq \frac{\partial \varphi(X)}{\partial s_N} \tag{A.19}$$

$$\nabla G_{s_N} \triangleq \frac{\partial G_{s_N}}{\partial s_N} \tag{A.20}$$

$$\nabla G_{D_N} \triangleq \frac{\partial G_{D_N}}{\partial D_N} \tag{A.21}$$

.

Figure A.1: Pointers to the global arrays representing X, $\varphi(X)$ and $\nabla\varphi(X)$ for the thread that processes node 1.

m_g $=\vartheta(X)$																							
Node 0					Node 1					Node 2						Node $N-1$					Terminal Node N	
G_{s_0}	G_{D_0}	G_{c_0}	$G_{\delta1_0}$	$G_{\delta2_0}$	G_{s_1}	G_{D_1}	G_{c_1}	$G_{\delta1_1}$	$G_{\delta2_1}$	G_{s_2}	G_{D_2}	G_{c_2}	$G_{\delta1_2}$	$G_{\delta2_2}$	$\cdots$	$G_{s_{N-1}}$	$G_{D_{N-1}}$	$G_{c_{N-1}}$	$G_{\delta1_{N-1}}$	$G_{\delta2_{N-1}}$	G_{s_N}	G_{D_N}
							$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$											
							m_constraints	m_slack	m_slack2	m_eq	m_D											
Pointers to the global array **m_g** for the thread processing block 1																						

$\vartheta(X)$	**m_gJac** $=\nabla\vartheta(X)$							
G_{s_1}	m_eqJacEye							
G_{D_1}			m_DJacEye					
G_{c_1}	m_constraintsJacXU			m_constraintsJacSlack				
$G_{\delta1_1}$	m_slackJacXU		m_slackJacD	m_slackJacSlack				
$G_{\delta2_1}$	m_slackJacXU2		m_slackJacD2	m_slackJacSlack2				
G_{s_2}	m_eqJacXU							
G_{D_2}	m_DJacXU		m_DJacD					
G_{c_2}								
$G_{\delta1_2}$								
$G_{\delta2_2}$								
X	s_1	q_1	D_1	δ_1	s_2	q_2	D_2	δ_2
Pointers to the global array **m_gJac** for the thread processing node 1								

Figure A.2: Pointers to the global arrays representing $\vartheta(X)$ and $\nabla\vartheta(X)$ for the thread that processes node 1.

A.2.2.4 Method `CppADThreadRun`

This method mainly encapsulates method **`CppADRun`** with the thread-handling code. When the thread is created in the class constructor, this method calls **`CppADInit`** to construct the CPPAD function objects that will be used by **`CppADRun`** to evaluate the objective function $\varphi(X)$, gradient of the objective function $\nabla\varphi(X)$, the constraint function $\vartheta(X)$ and the Jacobian of the constraint function $\nabla\vartheta(X)$.

This method locks onto **`g_thread_mutex_wait`** and waits for a signal from **`SundialsRun`** through **`g_thread_condition_wait`** to execute, after which this thread unlocks the **`g_thread_mutex_wait`** mutex, executes **`CppADRun`** and then attempts to lock the **`g_thread_mutex_wait`** again and then wait for a signal from **`SundialsRun`** to execute.

A.2.2.5 Method `CppADNoThreadRun`

If non-threaded execution is preferred, this method will just call **`CppADRun`**, without any thread-handling code. To use this method, **`CppADInit`** must be called from the class constructor before calling this method.

A.2.2.6 Method `CppADInit`

This method initialises the CPPAD object that will be used to evaluate $\varphi(X)$, $\nabla\varphi(X)$, $\vartheta(X)$ and $\nabla\vartheta(X)$.

It creates an instance **`Plant`** of the **`ProcessPlantODE`** class. The **`ProcessPlantODE`** class must have a member function called **`Ode`** that takes the following parameters:

- **`t`**: (in), The independent variable of the function to integrate.
- **`x`**: (in), The dependent variable of the function to integrate.
- **`f`**: (out), The change of the dependent variable with time ($\frac{\partial x}{\partial t}$).

The **`ProcessPlantODE`** class has some custom methods that allow additional values needed by the nonlinear model and cost function to be passed to the class:

- **`SetParams`**: Passes the parameter values needed by the nonlinear model.
- **`SetControls`**: Passes the value of the control moves to the nonlinear model.
- **`SetCost`**: Passes all the weighting and scaling matrices as well as the setpoint vectors for the states and the inputs needed by the cost function.

This method then calls

```
CppAD::Independent(xupuset);
```

that flags CPPAD to start "taping" the calculations. Once the taping is done, a CPPAD function object is created that will be used to evaluate the function and derivatives of the function such as the Jacobian and the Hessian. Only one variable can be declared the independent variable by CPPAD, and to capture all the independent variables of interest for the nonlinear model, the objective function and the constraint functions, a compound variable **xupuset** is defined that contains x, u, p, the weighting and scaling matrices and the setpoint vectors for the states and inputs for the objective function. The independent variable here is all the variables that one would like to differentiate against. Other variables can also be used during taping, but these variables will become constants in the CPPAD function object preventing them from changing in the future. The setpoints are therefore added to the independent variable, which allows the setpoints to be changed, without reconstructing the CPPAD function object through another "taping" with modified setpoint values.

The nonlinear model is then integrated to capture the calculations for the CPPAD function object by calling

```
xf = CppAD::Runge45(Plant, M, ti, tf, xi, e);
```

where **xf** is the final state of the nonlinear model and the objective function after the integration period (**tf** – **ti**), **M** is the number of steps over the integration period, **ti** is the starting time of the integration period, **tf** is the final time of the integration period, **xi** is the initial state of the nonlinear model and **e** is the absolute errors of the integration for each entry in **xf**.

The constraints pertaining to the limitations of the process variables are "taped" by evaluating the call to

```
Constraints(constraints, xf, u, deltaU);
```

where **constraints** is the solution of the constraint functions, **xf** is the final state of the nonlinear system for the current time-step, **u** is the constant control moves applied during the current time-step and **deltaU** is used to evaluate constraints on the change in control moves between the previous and current time-step.

The solutions of the nonlinear model, objective function and the constraint functions are combined into one variable **xf_constraints**, because CPPAD only allows one variable to be declared the dependent variable. The "taping" is stopped and the CPPAD function object is created by the call

```
thread->f.Dependent(xupuset, xf_constraints);
```

that assigns the CPPAD function object to the **f** object of the thread. Memory is preallocated

to allow the evaluation of third order derivatives by the call to

```
thread->f.capacity_taylor(4);
```

A similar CPPAD function object is created for the outputs of the nonlinear model (such as the PSE and SLEV), rather than the internal states by the following calls

```
CppAD::Independent(xup);
ProcessPlant(y, x_dot, x, u, params, paramsConst);
thread->fy.Dependent(xup, y);
thread->fy.capacity_taylor(3);
```

that assigns the CPPAD function object to the **fy** object of the thread and preallocates memory for second order derivatives of the CPPAD function object.

A.2.2.7 Method CppADRun

This method evaluates the objective function $\varphi(X)$, gradient of the objective function $\nabla\varphi(X)$, the constraint function $\vartheta(X)$ and the Jacobian of the constraint function $\nabla\vartheta(X)$ for one block. This method is usually called by $N+1$ threads simultaneously, each evaluating a different block, for parallel computing of the functions. Parallel execution of this function can lead to increased speed on multi-core and multi-processor systems.

This method starts by converting the percentage uncertainty of the variable parameters to upper and lower bounds on the parameter.

The method then evaluates $s_{i+1} = f_{\text{model}}(s_i, q_i, p)$, which is the integral of the nonlinear model over one time-step, the objective function and the constraint function by calling

```
thread->m_f = thread->f.Forward(0, thread->m_xupu);
```

where **m_xupu** is an array containing s_i, q_i and p. The 0 indicates that this is the normal function evaluation and not a derivative. The first n_x entries in **thread->m_f** is $s_{i+1} = f_{\text{model}}(s_i, q_i, p)$, the next entry is the objective function value and the next n_c entries is the solution of the constraint functions.

The method then evaluates $\nabla f_{\text{model}}(s_i, q_i, p)$, the gradient of the objective function and the Jacobian of the constraints by calling

```
thread->m_fJacTemp = thread->f.ForOne(thread->m_xupu, i);
for i = 1,...,n_x+n_u+n_p
```

The **ForOne** indicates that this is a first-order derivative of the integral of the nonlinear model over one time-step, the objective function and the constraint functions with regard to state (s_i), inputs (q_i) and parameters (p).

The method then evaluates $\nabla^2 f_{\text{model}}(x,u,p)$, the Hessian of the objective function and the Hessian of the constraint functions by calling

```
thread->m_fHessian = thread->f.ForTwo(thread->m_xupu, x1, x2);
```

The **ForTwo** indicates that this is a second-order derivative of the integral of the nonlinear model functions over one time-step, the objective function and the constraint functions with regard to state (s_i), inputs (q_i) and parameters (p).

The rest of the method extracts the results of the function and derivative evaluations above and assigns it to the correct variables. To understand how the values are extracted from the results, the packing of the results should be understood. Given a function $J : \mathbb{R}^n \rightarrow \mathbb{R}^p$ such that $y = J(x)$, where $y \in \mathbb{R}^p$ and $x \in \mathbb{R}^n$, and the Jacobian is $\frac{\partial J(x)}{\partial x}$. Given a function $H : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^p$ such that $y = H(x,u)$, where $u \in \mathbb{R}^m$ then the Hessian is given by $\frac{\partial^2 H(x,u)}{\partial u \partial x}$. The packing of the Jacobian and the Hessian is show in Figure (A.3).

This method calls **RobustWeight** to form the robust weighting term from the current slack variable values for the objective function and the inequality constraints on the states and inputs that describe the limitations of the process variables.

This method calls the macro **JAC_MATRIX** to return a pointer to the starting point of the relevant Jacobian block and calls the macro **HESSIAN_MATRIX** to return a pointer to the relevant starting point of the required Hessian block. The **JAC_MATRIX** macro takes the variable pointer to the array, the required row and column entry and the number of column entries in a row. The **HESSIAN_MATRIX** macro takes the variable pointer to the array, the required output entry, the entry of the first derivative vector, the entry of the second derivative vector, the number of entries in the first derivative vector and the number of entries in the second derivative vector.

This method assigns the solutions in the following order:

- **m_eq**: Calls

  ```
  thread->m_eq[i] = f[i] - thread->curXU[VAR_BLOCK + i];
  ```

  that returns $f_{i-1}(s_{i-1}, q_{i-1}, \bar{p}) - s_i$.

- **m_eqJacEye**: The non-zeros entries of $-I_{n_x}$.

- **m_eqJacXU**: Copy the result of $(\frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i}, \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial q_i})$ by calling

  ```
  memcpy(&thread->m_eqJacXU[nIndex], &fJac[NXUPU*i], sizeof(double)*NXU);
  ```

- The solutions of the inequality constraint functions describing the limitations of the system process variables.

	Jacobian Packing in Memory											
	y_1			y_2			$\cdots$	y_p				
	x_1	$\cdots$	x_n	x_1	$\cdots$	x_n		x_1	$\cdots$	x_n		
Array Entries	$\frac{\partial J_1}{\partial x_1}$	$\cdots$	$\frac{\partial J_1}{\partial x_n}$	$\frac{\partial J_2}{\partial x_1}$	$\cdots$	$\frac{\partial J_2}{\partial x_n}$	$\cdots$	$\frac{\partial J_p}{\partial x_1}$	$\cdots$	$\frac{\partial J_p}{\partial x_n}$		

	Hessian Packing in Memory																							
	y_1							y_2							$\cdots$	y_p								
	u_1			u_m				u_1			u_m					u_1			u_m				
	x_1	$\cdots$	x_n	x_1	$\cdots$	x_n		x_1	$\cdots$	x_n	x_1	$\cdots$	x_n			x_1	$\cdots$	x_n	x_1	$\cdots$	x_n			
Array Entries	$\frac{\partial^2 H_1}{\partial u_1 \partial x_1}$	$\cdots$	$\frac{\partial^2 H_1}{\partial u_1 \partial x_n}$	$\frac{\partial^2 H_1}{\partial u_m \partial x_1}$	$\cdots$	$\frac{\partial^2 H_1}{\partial u_m \partial x_n}$	$\cdots$	$\frac{\partial^2 H_2}{\partial u_1 \partial x_1}$	$\cdots$	$\frac{\partial^2 H_2}{\partial u_1 \partial x_n}$	$\frac{\partial^2 H_2}{\partial u_m \partial x_1}$	$\cdots$	$\frac{\partial^2 H_2}{\partial u_m \partial x_n}$	$\cdots$	$\cdots$	$\frac{\partial^2 H_p}{\partial u_1 \partial x_1}$	$\cdots$	$\frac{\partial^2 H_p}{\partial u_1 \partial x_n}$	$\frac{\partial^2 H_p}{\partial u_m \partial x_1}$	$\cdots$	$\frac{\partial^2 H_p}{\partial u_m \partial x_n}$		

Figure A.3: Jacobian and Hessian packing in memory.

– **m_constraints**: Copies the result from of the evaluation of the constraint functions ($\theta_{j,i}(s_i,q_i)$) and then adds the robust weighting terms ($e^T\delta_{j,i}$) by calling **RobustWeight**. Calling

```
thread->m_constraints[i] = constraints[i] +
RobustWeight(thread->curSlack + NP*(i+1));
```

returns $\theta_{j,i}(s_i,q_i)+e^T\delta_{j,i},\ j=1,\ldots,n_c$.

– **m_constraintsJacXU**: Stores the Jacobian of the inequality constraint functions relative to the states (s_i) and the inputs (q_i) by calling

```
memcpy(&thread->m_constraintsJacXU[iRow*NXU],
&constraintsJac[NXUPU*iRow], sizeof(double)*NXU);
```

that returns $\frac{\partial\theta_{j,i}(s_i,q_i)}{\partial s_i}$ and $\frac{\partial\theta_{j,i}(s_i,q_i)}{\partial q_i}$, $j=1,\ldots,n_c$.

– **m_constraintsJacSlack**: The Jacobian of the inequality constraints relative to the slack variables (δ_{ji}) that basically returns $1_{n_p}^T \triangleq (1,\ldots,1)^T \in \mathbb{R}^{n_p}$.

- The solutions of the constraint functions pertaining to the $\mathbf{D}_i$-matrix.

– **m_D**: It first calculates ($\frac{\partial f_{i-1}(s_{i-1},q_{i-1},\bar{p})}{\partial p} - I_{n_x}\cdot D_i$) by calling

```
JAC_MATRIX(thread->m_D, iRow, jRow, NP) =
JAC_MATRIX(fJac, iRow, NXU+jRow, NXUPU) -
JAC_MATRIX(thread->nextD, iRow, jRow, NP);
```

and then adds $\frac{\partial f_{i-1}(s_{i-1},q_{i-1},\bar{p})}{\partial s_{i-1}}\cdot D_{i-1}$ to the answer by calling

```
JAC_MATRIX(thread->m_D, iRow, jRow, NP) +=
JAC_MATRIX(fJac, iRow, nX, NXUPU) *
JAC_MATRIX(thread->curD, nX, jRow, NP);
```

that returns $\frac{\partial f_{i-1}(s_{i-1},q_{i-1},\bar{p})}{\partial s_{i-1}}\cdot D_{i-1} - I_{n_x}\cdot D_i + \frac{\partial f_{i-1}(s_{i-1},q_{i-1},\bar{p})}{\partial p}$.

– **m_DJacXU**: This is the Jacobian of the constraints relating to the $\mathbf{D}_i$-matrix relative to the states (s_i) and the inputs (q_i). It first assigns $\frac{\partial\frac{\partial f_i(s_i,q_i,\bar{p})}{\partial p}}{\partial s_i}$ by calling

```
JAC_MATRIX(thread->m_DJacXU, NP*iRow+jRow, iCol, NXU) =
HESSIAN_MATRIX(fHessian, iRow, NXU+jRow, iCol, NXUP, (NXU+NU));
```

and then adds $\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial s_i}$ by calling

```
JAC_MATRIX(thread->m_DJacXU, NP*iRow+jRow, iCol, NXU) +=
HESSIAN_MATRIX(fHessian, iRow, nX, iCol, NXUP, (NXU+NU)) *
JAC_MATRIX(thread->curD, nX, jRow, NP);
```

that returns $\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial s_i} + \frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial p}}{\partial s_i}$, which also includes $\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial q_i} + \frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial p}}{\partial q_i}$.

- **m_DJacD**: This is the Jacobian of the constraints relating to the $\mathbf{D}_{i+1}$-matrix relative to the the $\mathbf{D}_i$-matrix. It retrieves the result by calling

```
JAC_MATRIX(thread->m_DJacD, NP*iRow+jRow, nX, NX) =
JAC_MATRIX(fJac, iRow, nX, NXUPU);
```

that returns $\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial D_i}$.

- **m_DJacEye**: This is the non-zero entries in $-\mathbb{I}_{n_D}$.

- The solutions of the constraint functions pertaining to the slack variables.

  - **m_slack**, **m_slack2**: The solutions of the constraints functions pertaining to the slack variables. Calls

```
for (nX = 0; nX < NX; nX++)
 JAC_MATRIX(thread->m_slack, iRow, jRow, NP) +=
JAC_MATRIX(thread->nextD, nX, jRow, NP) *
JAC_MATRIX(costGrad, iRow, nX, NXUP) * box_weighting[jRow];

JAC_MATRIX(thread->m_slack2, iRow, jRow, NP) =
JAC_MATRIX(thread->m_slack, iRow, jRow, NP);
```

    and subtracts the current slack variable values from **m_slack** and adds the current slack variable values to **m_slack2** by calling

```
JAC_MATRIX(thread->m_slack, iRow, jRow, NP) -=
JAC_MATRIX(thread->curSlack, iRow, jRow, NP);

JAC_MATRIX(thread->m_slack2, iRow, jRow, NP) +=
JAC_MATRIX(thread->curSlack, iRow, jRow, NP);
```

    that returns $G_{\delta 1_i}$ (A.15) and $G_{\delta 2_i}$ (A.16).

  - **m_slackJacXU**, **m_slackJacXU2**: This is the Jacobian of the constraint functions pertaining to the slack variables relative to the states (s_i) and the inputs (q_i). It calls

```
JAC_MATRIX(thread->m_slackJacXU, NP*iRow+jRow, iCol, NXU) = 0.0;

for (nX = 0; nX < NX; nX++)
JAC_MATRIX(thread->m_slackJacXU, NP*iRow+jRow, iCol, NXU) +=
JAC_MATRIX(thread->nextD, nX, jRow, NP) *
HESSIAN_MATRIX(fHessian, NX+iRow, nX, iCol, NXUP, (NXU+NU)) *
box_weighting[jRow];
```

that return

$$\begin{matrix} \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_0} & \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} \\ \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} \\ \vdots & \vdots \\ \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{n_C,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{n_C,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} \end{matrix}$$

to **m_slackJacXU** and because **m_slackJacXU2** = **m_slackJacXU** calls

```
JAC_MATRIX(thread->m_slackJacXU2, NP*iRow+jRow, iCol, NXU) =
JAC_MATRIX(thread->m_slackJacXU, NP*iRow+jRow, iCol, NXU);
```

– **m_slackJacD**, **m_slackJacD2**: This is the Jacobian of the constraint functions pertaining to the slack variables relative to the $\mathbf{D}_{i+1}$-matrix. It calls

```
JAC_MATRIX(thread->m_slackJacD, NP*iRow+jRow, nX, NX) =
JAC_MATRIX(costGrad, iRow, nX, NXUP)*box_weighting[jRow];
```

that returns

$$\begin{matrix} \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\ \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\ \vdots \\ \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{n_C,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \end{matrix}$$

to **m_slackJacD** and because **m_slackJacD2** = **m_slackJacD** calls

```
JAC_MATRIX(thread->m_slackJacD2, NP*iRow+jRow, nX, NX) =
JAC_MATRIX(thread->m_slackJacD, NP*iRow+jRow, nX, NX);
```

– **m_slackJacSlack**, **m_slackJacSlack2**: This is the Jacobian of the constraint functions pertaining to the slack variables (δ_{ji}) relative to the slack variables (δ_{ji}). The non-zero entries of $-\mathbb{I}_{n_{slack}}$ are returned to **m_slackJacSlack** and the non-zero entries of $\mathbb{I}_{n_{slack}}$ are returned to **m_slackJacSlack2**.

A.2.3 Main execution loop

The main execution loop is implemented in **Robust_NMPC.cpp**. Figure A.4 shows the important calls in the main execution loop and Figure A.5 shows the program flow during optimisation.

- The program starts by reading in the simulation scenario file (**FileInName**) and opening the simulation solution file (**FileOutName**) from the file names passed in as arguments to the program.
- Memory is allocated for the
  - initial state (**x0 (NX)**),
  - state setpoint (**x_setpoint (NX)**),
  - steady state values of the control moves (**u_setpoint (NU)**),
  - scaling of the states for the objective function (**x_scale (NX)**),
  - scaling of the control moves for the objective function (**u_scale (NU)**),
  - control move returned by the nonlinear optimisation problem for the current time-step (**u (NU)**),
  - output of the system such as the PSE and SLEV for the milling circuit (**y (NY)**),
  - final state of the current time-step after simulation (**xf (NX)**),
  - disturbances on the control moves (**u_dist**),
  - time-varying parameter vector (**p_dist**), and
  - objective function solution (**obj**).
- The program reads in the values of the above-mentioned variables from the simulation scenario file.
- An instance of the **MyNLP** class is created that contains the definition of the nonlinear programming problem to solve with a call to

  ```
  SmartPtr<TNLP> mynlp = new MyNLP(x_setpoint, u_setpoint, x_scale,
  u_scale, x0);
  ```

  that also initialises the class with the state setpoint, steady state values of the control moves, the scaling factors for the states, control moves used in the objective function and the initial values for the states.

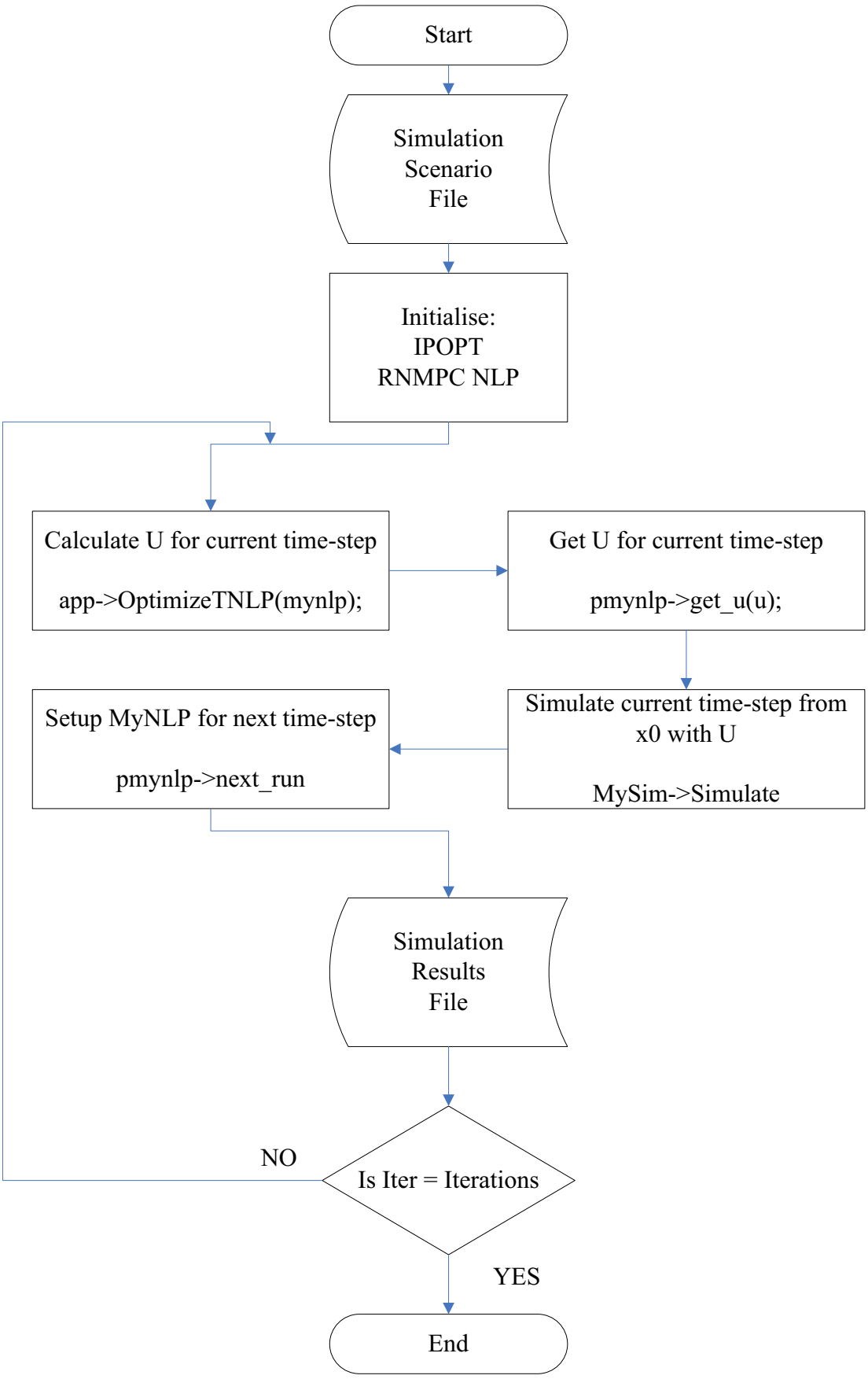


Figure A.4: Main execution loop.

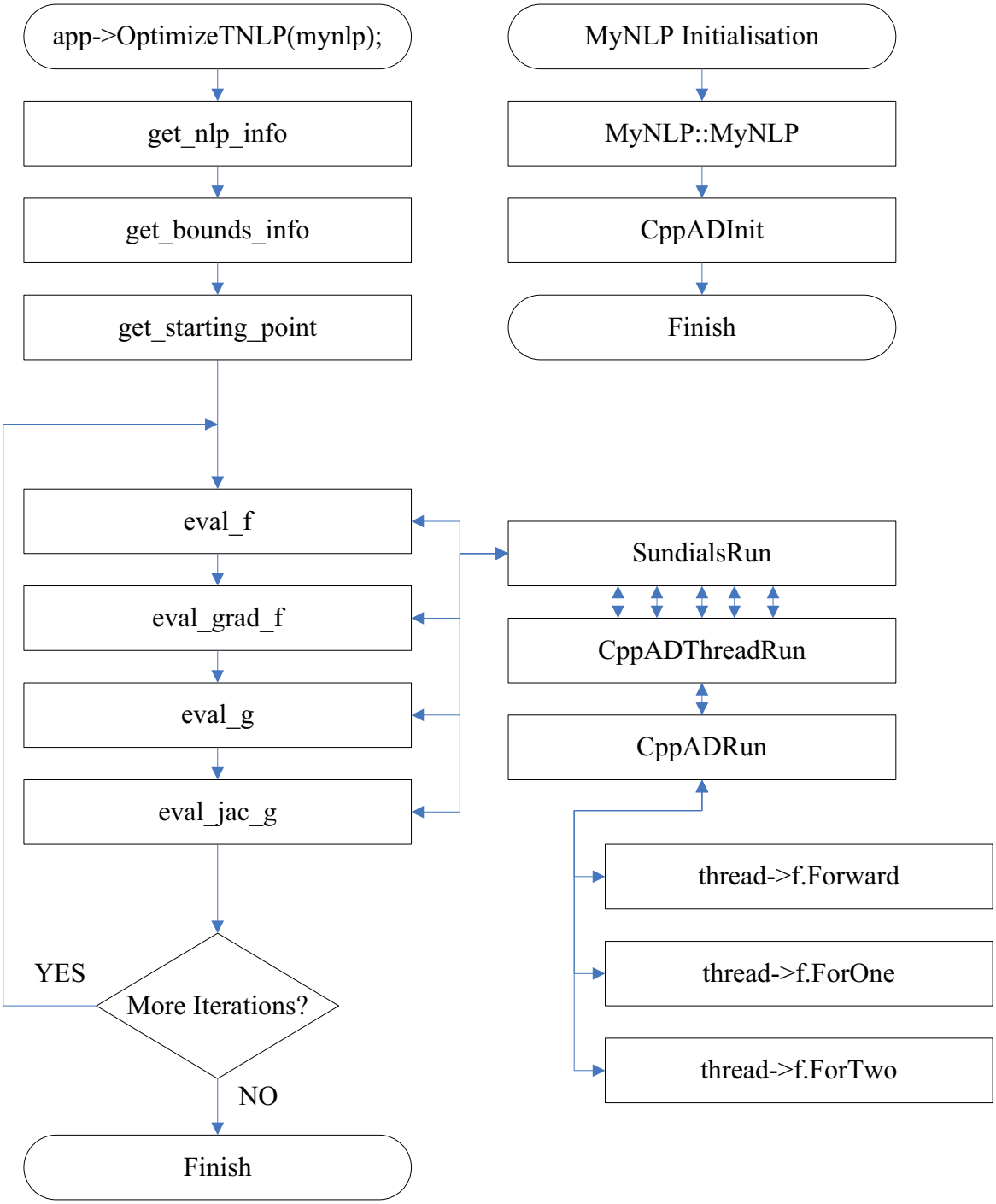


Figure A.5: Optimisation program flow.

- An instance of the IPOPT solver is created with a call to

```
SmartPtr<IpoptApplication> app = new IpoptApplication();
```

- IPOPT is instructed to use the quasi-Newton approximation of the Hessian and the IPOPT solver initiated with calls to

```
app->Options()->SetStringValue("hessian_approximation",
"limited-memory");
app->Initialize();
```

- The main simulation loop performs the following actions for the number of iterations specified in the simulation scenario file.
  - Reads the parameter vector, disturbances on the control moves and state setpoint for the current iteration from the simulation scenario file.
  - Calls

    ```
    status = app->OptimizeTNLP(mynlp);
    ```

    to perform a *cold-startup* optimisation or

    ```
    status = app->ReOptimizeTNLP(mynlp);
    ```

    to perform a *warm-startup* optimisation for the current time-step.
  - The control moves for the current time-step is extracted with a call to

    ```
    pmynlp->get_u(u);
    ```

The current time-step is simulated with a call to

```
mysim->simulate(x0, u, u_dist, p_dist, samples, xf, y);
```

with the initial state (`x0`), the calculated control moves (`u`), the disturbances on the control moves (`u_dist`), the disturbed parameter vector (`p_dist`), the number of sub-samples to calculate (`samples`) for more accurate simulation. The method returns the "real" final state (`xf`) of the system as well as the output of the system (`y`). The simulation basically just integrates the system dynamics with the provided parameters for one sampling interval.

  - The nonlinear programming problem is initialised for the next iteration with a call to

    ```
    pmynlp->next_run(x_setpoint, xf, u, u_setpoint, &obj, contraction);
    ```

with the state setpoint for the next time-step (`x_setpoint`), the initial state of the next time-step (`xf`) being the same as the final state of the current time-step, the previous control moves (`u`) for the next time-step and the steady-state values for the control moves (`u_setpoint`) for the next time-step. The method returns the objective function value (`obj`) of the current time-step as well as the margins applied to the constraints for robustness (`contraction`).

- The simulation results are saved at every iteration to the simulation results file. It is done to ensure that data are available, should the simulation not complete successfully. This will allow the simulation to be debugged, should it fail prematurely. The simulation data that is returned for each iteration is
  - the system state,
  - the control moves,
  - the system output,
  - the steady-state values of the control moves,
  - the setpoint of the system,
  - the objective function value,
  - the margins applied to the constraints for robustness,
  - the execution time of the iteration, and
  - the total execution time until the current iteration.
- The "real" final state of the system (`xf`) is assigned to the initial state (`x0`) of the system and the loop repeats until all the iterations are done.

The application returns statistics on each iteration to the terminal to help track the progress of the simulation as it executes. The current iteration, the total number of iterations, the execution time of the last iteration and the total execution time until the last iteration are displayed.

ADDENDUM B

AUXILIARY RESULTS

B.1 AUXILIARY SIMULATION RESULT

This section contains the graphs of mostly the NMPC controller that corresponds to the simulation scenarios of Section 5.3.

B.1.1 Constant setpoint following and disturbance rejection

The simulation scenario shown in Figure B.1a and Figure B.2a allows SLEV to vary freely with only upper and lower constraints enforced. Steering SLEV to setpoint compared to allowing SLEV to vary freely does not have a significant impact on the closed-loop performance under NMPC in terms of PSE setpoint tracking and the average circuit throughput (Table B.1). Allowing SLEV to vary within bounds allows the NMPC to change the density of the slurry inside the sump (assuming fully mixed conditions) and as a result allows the NMPC to control the feed density to the cyclone. Control of the feed density to the cyclone can increase the control envelope of the NMPC.

B.1.2 Reduced PSE setpoint to 75% and 70% < 75μm

Figure B.3b and Figure B.4b show that the NMPC tracks the reduced PSE setpoint of 75% < 75μm and LOAD setpoint of 45% volumetric filling well. The NMPC does not track PSE as closely as the RNMPC shown in Figure 5.16b and Figure 5.17b. The throughput shows large variations due primarily to the ore hardness and composition variations. Decreasing the setpoint for PSE to 75% increased the average throughput of the milling circuit to 74.3 from 72.6 tons per hour.

Figure B.3c and Figure B.4c show that the NMPC tracks the reduced PSE setpoint of 70% < 75μm and LOAD setpoint well. Decreasing the PSE setpoint to 70% resulted in an average throughput of 81.9 tons per hour, as seen in Table B.1.

The RNMPC of Section 5.3.3 and the NMPC results presented here show almost identical results, with the RNMPC tracking the PSE setpoint slightly better.

B.1.3 Regulate PSE, LOAD and Throughput

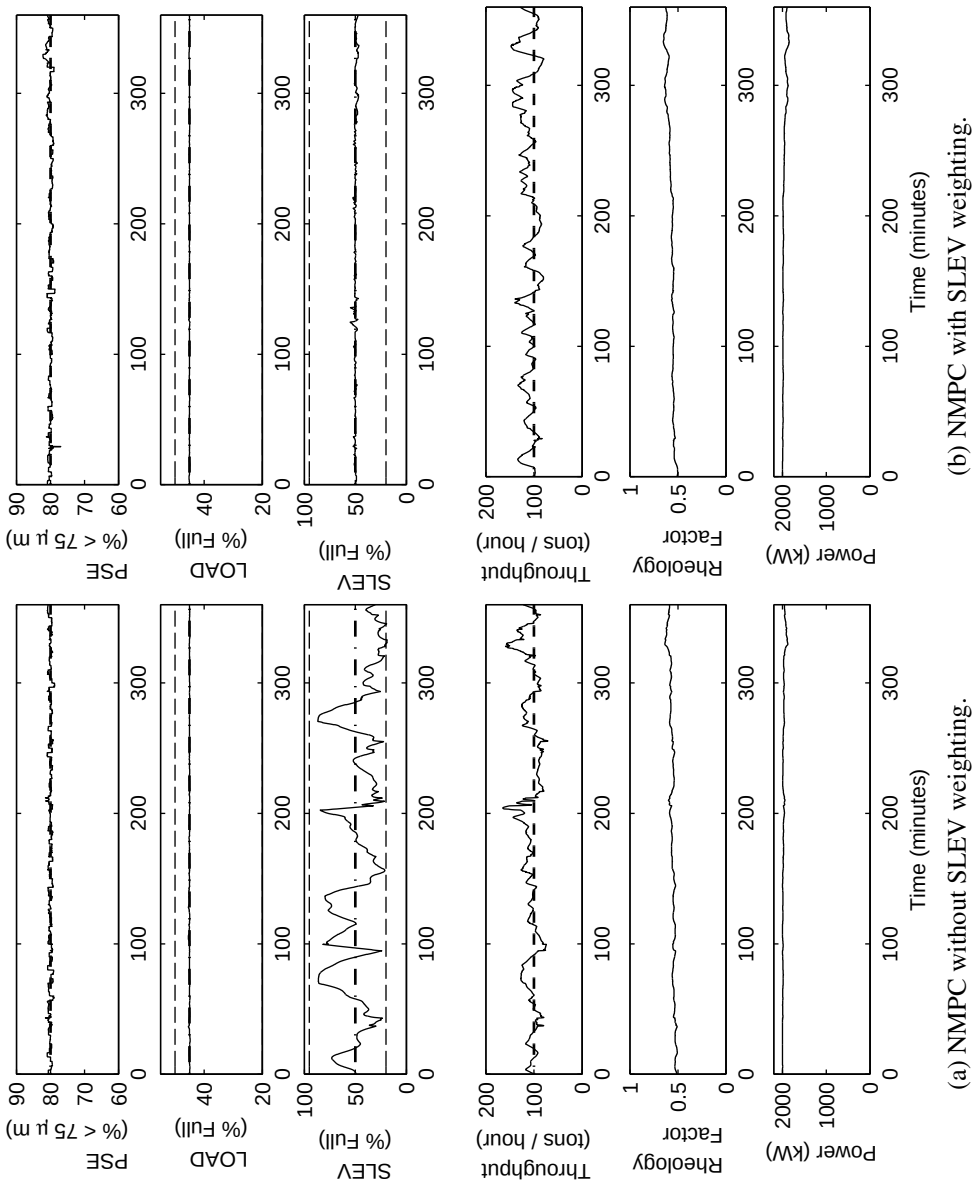
Figure B.5a and Figure B.6a show that adding the throughput to the objective function causes the NMPC to make a trade-off between following the PSE setpoint and throughput setpoint according to their respective weightings. Table B.1 shows that the PSE error increases significantly (from 0.30 to 3.31) while the average throughput decreases slightly (from 72.5 $^{\text{tons}}/_{\text{hour}}$ to 71.5 $^{\text{tons}}/_{\text{hour}}$), compared to the scenario where throughput is not included in the objective function (Figure 5.14b and Figure 5.15b). It was expected that adding THROUGHPUT to the objective function would increase the average throughput, but it resulted in a slight reduction in the average throughput, which was unexpected. Figure B.6 shows large variation in the manipulated variables that can be the cause of the poor overall performance.

Figure B.5b and Figure B.6b show that increasing the weighting on THROUGHPUT from 5 to 10 increases the PSE setpoint tracking error from 3.31 to 7.89 while decreasing the average throughput from 71.5 $^{\text{tons}}/_{\text{hour}}$ to 69.3 $^{\text{tons}}/_{\text{hour}}$. Figure B.6b shows that the large variation in the manipulated variables persists and this may explain the unexpected drop in average throughput.

Figure B.5c and Figure B.6c show that increasing the weighting on THROUGHPUT further from 10 to 20 increases the error in the PSE setpoint tracking from 7.89 to 13.01 (as expected) while increasing the average throughput from 69.3 $^{\text{tons}}/_{\text{hour}}$ to 70.8 $^{\text{tons}}/_{\text{hour}}$, as seen in Table B.1. Figure B.6c shows large variation in the manipulated variables, because the gain of the controller is too high. The gain of the controller is controlled by the weighting on the manipulated and controlled variables.

Figure B.7 and Figure B.8 show that the large variations in the MVs are reduced each time the weightings on the MVs are increased. The PSE setpoint tracking error is reduced while the average throughput is increased each time the weightings on the MVs are increased. The PSE setpoint tracking error is worse but the average throughput is higher compared to the case where THROUGHPUT is not included in the objective function (Figure 5.14b and Figure 5.15b), as seen in Table B.1.

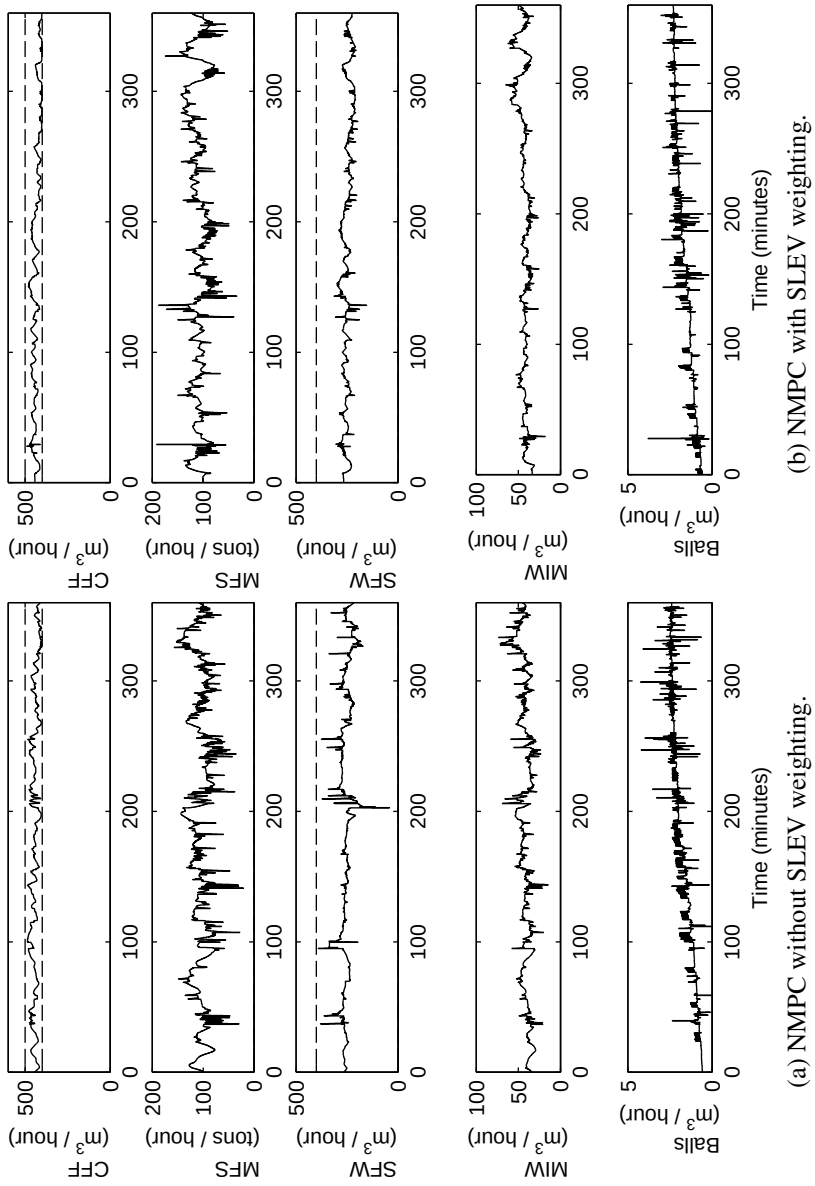
The results in this section show that the NMPC, like the RNMPC, cannot overcome the inherent trade-off between PSE and THROUGHPUT. The objective function allows the trade-off between PSE setpoint tracking and average throughput to be manipulated by changing the weighting of PSE and THROUGHPUT in the objective function.



(a) NMPC without SLEV weighting.

(b) NMPC with SLEV weighting.

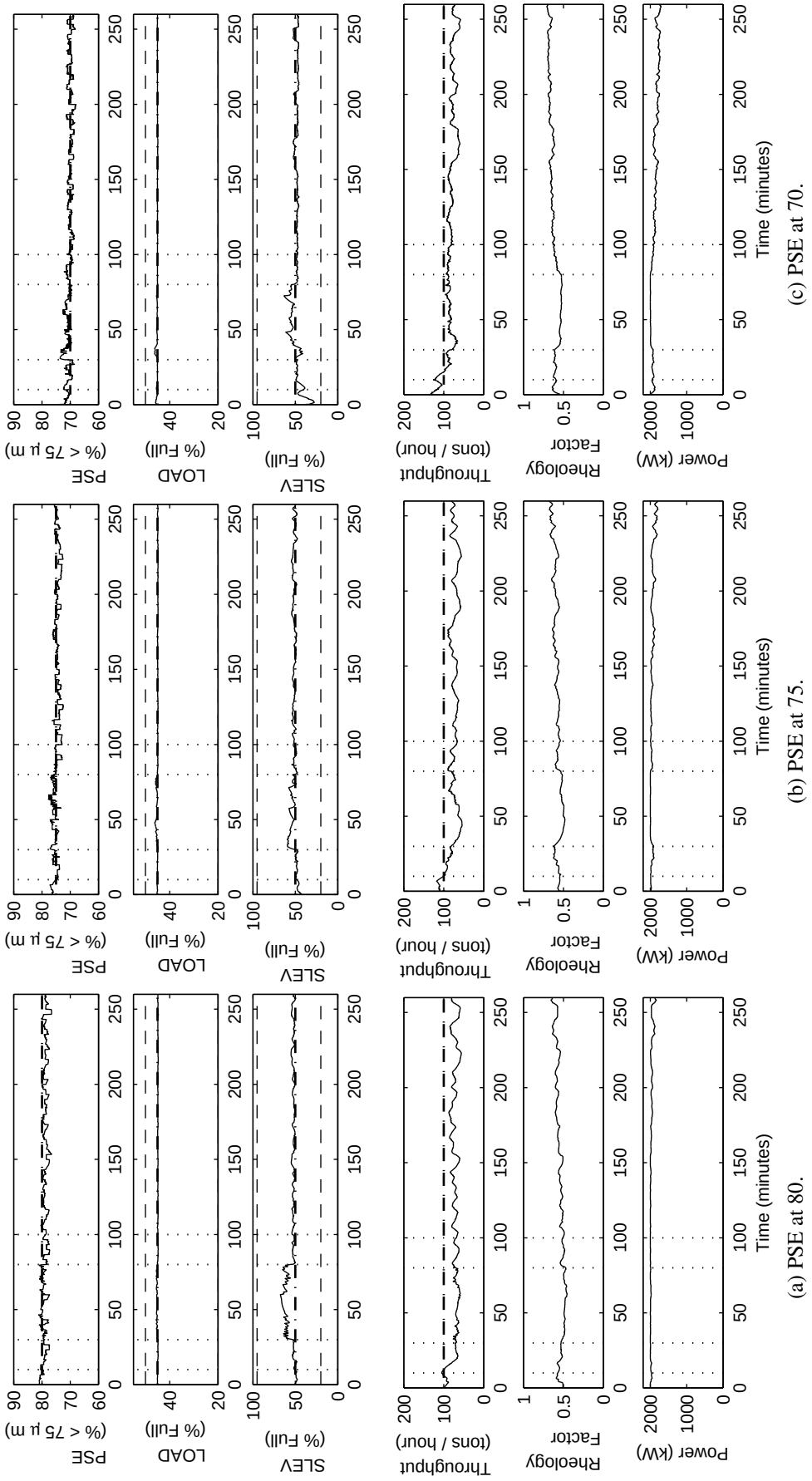
Figure B.1: CVs of NMPC
There is no step disturbance present in either the feed ore hardness or the fraction of rock in the feed ore. This represents the nominal scenario with only zero mean parameter variations present. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) NMPC without SLEV weighting.

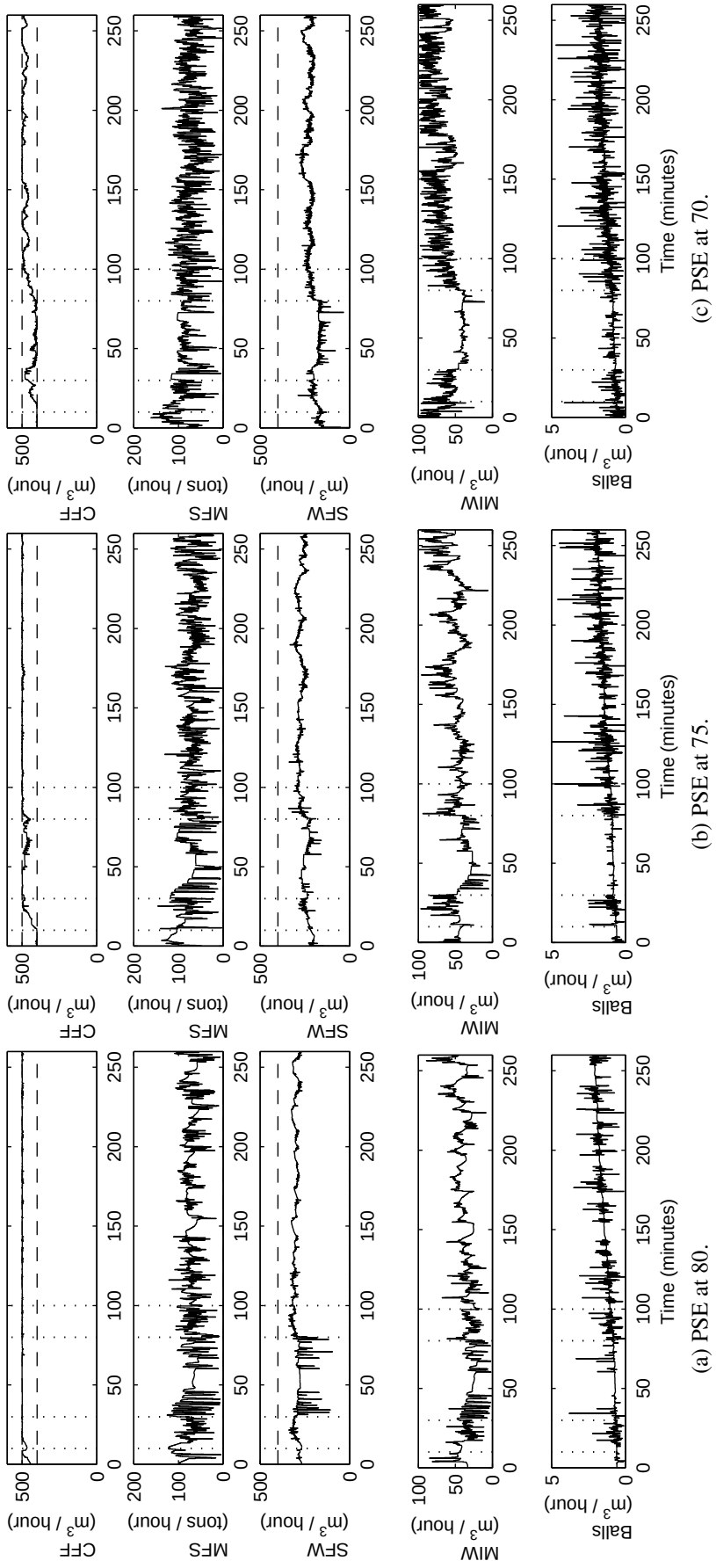
(b) NMPC with SLEV weighting.

Figure B.2: MVs of NMPC.
There is no step disturbance present in either the feed ore hardness or the fraction of rock in the feed ore. This represents the nominal scenario with only zero mean parameter variations present. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) PSE at 80. (b) PSE at 75. (c) PSE at 70.

Figure B.3: CVs of the NMPC.
The NMPC regulates PSE at a constant setpoint of 80% (a), 75% (b) and 70% (c) with LOAD at a constant setpoint of 45% and SLEV at a constant setpoint of 5 m^3. The PSE setpoint is reduced in order to increase the average throughput. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

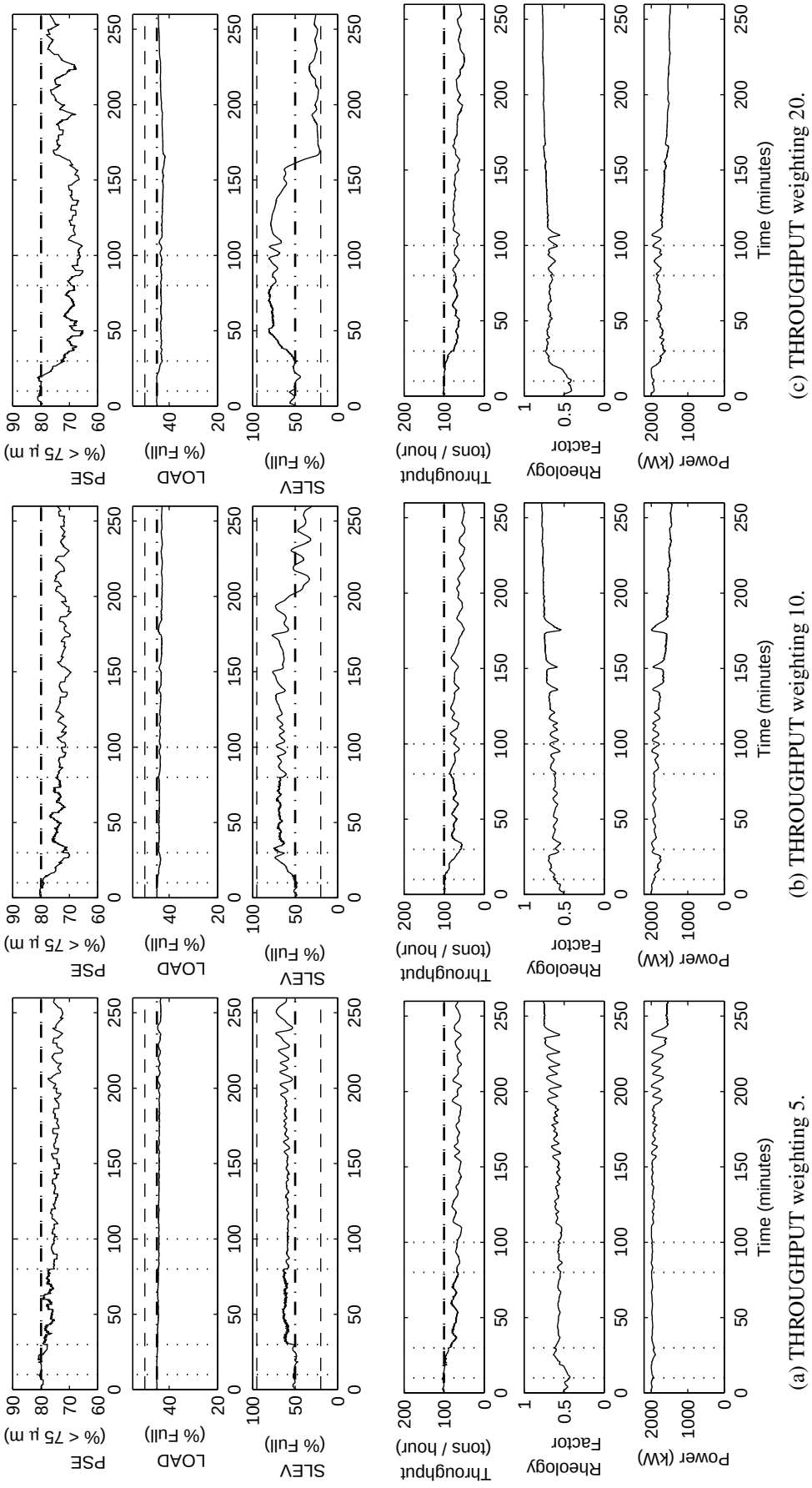


(a) PSE at 80.

(b) PSE at 75.

(c) PSE at 70.

Figure B.4: MVs of the NMPC.
The NMPC regulates PSE at a constant setpoint of 80% (a), 75% (b) and 70% (c) with LOAD at a constant setpoint of 45% and SLEV at a constant setpoint of 5 m^3. The PSE setpoint is reduced in order to increase the average throughput. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

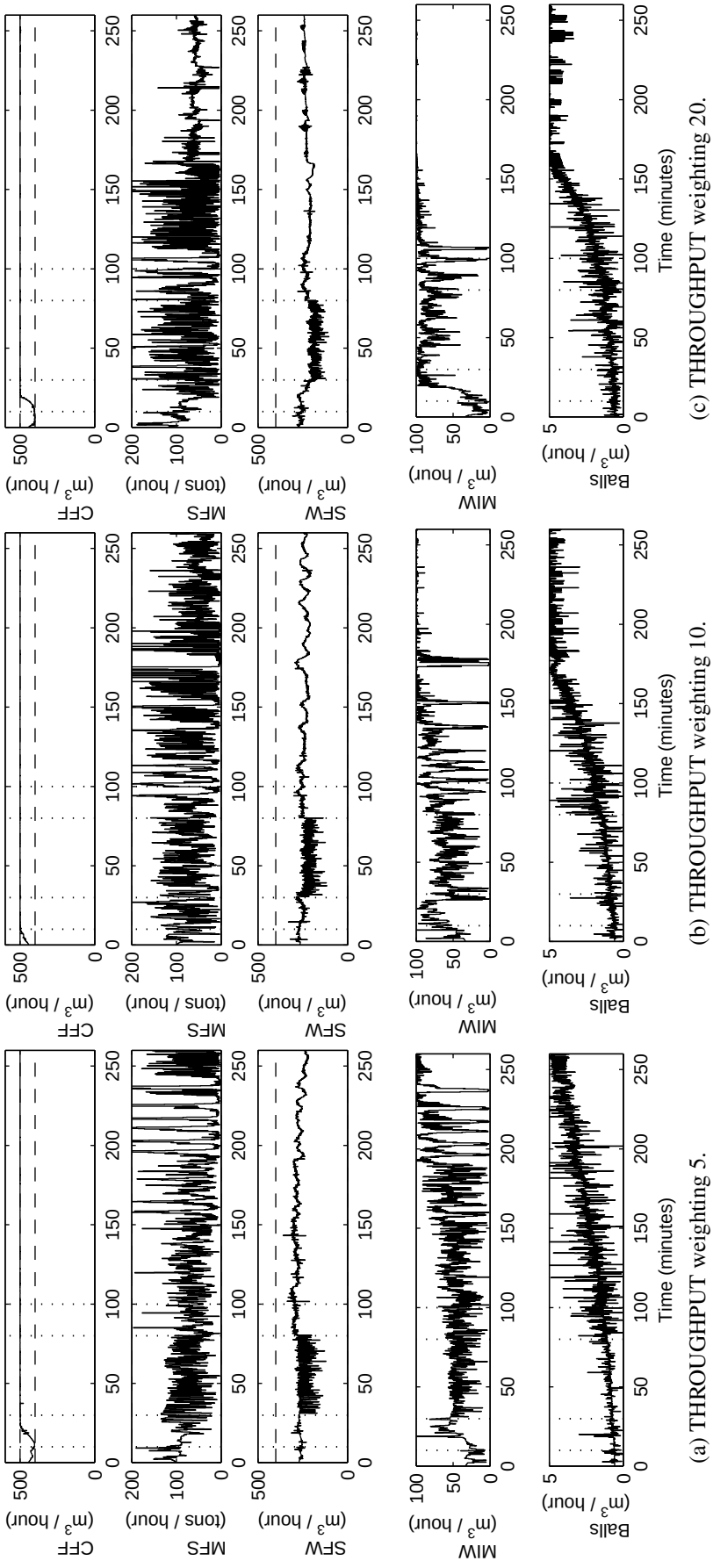


(a) THROUGHPUT weighting 5.

(b) THROUGHPUT weighting 10.

(c) THROUGHPUT weighting 20.

Figure B.5: CVs of the NMPC.
The NMPC regulates PSE and LOAD at constant setpoints and throughout at a constant setpoint of 100 tons/hour. The weighting of THROUGHPUT in the objective function is 5 (a), 10 (b) and 20 (c), while the weighting on PSE and LOAD is 100 respectively. The optimising property of the RNMPC is employed to try increasing throughput without affecting PSE. An increase in THROUGHPUT, however, causes a decrease in PSE. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

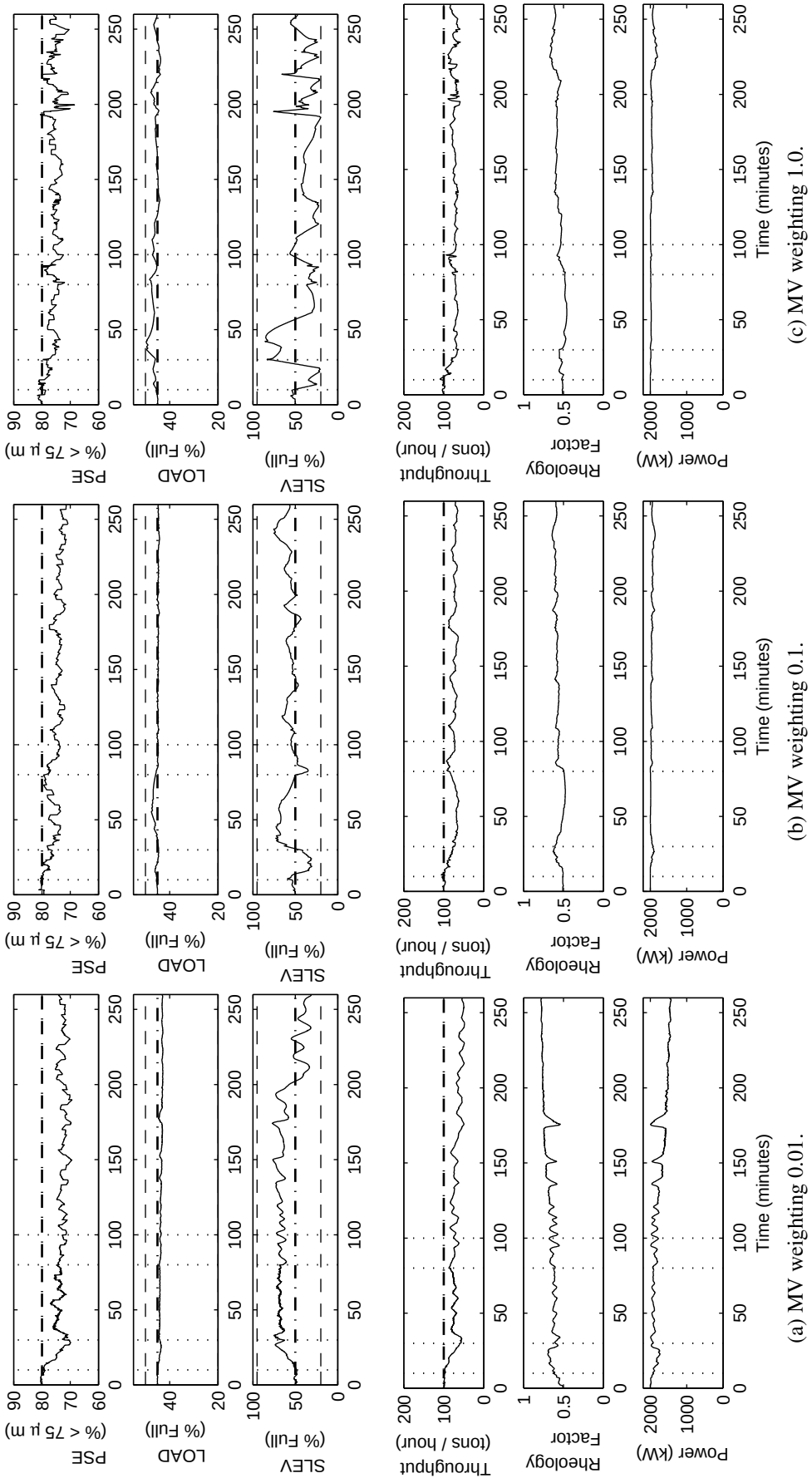


(a) THROUGHPUT weighting 5.

(b) THROUGHPUT weighting 10.

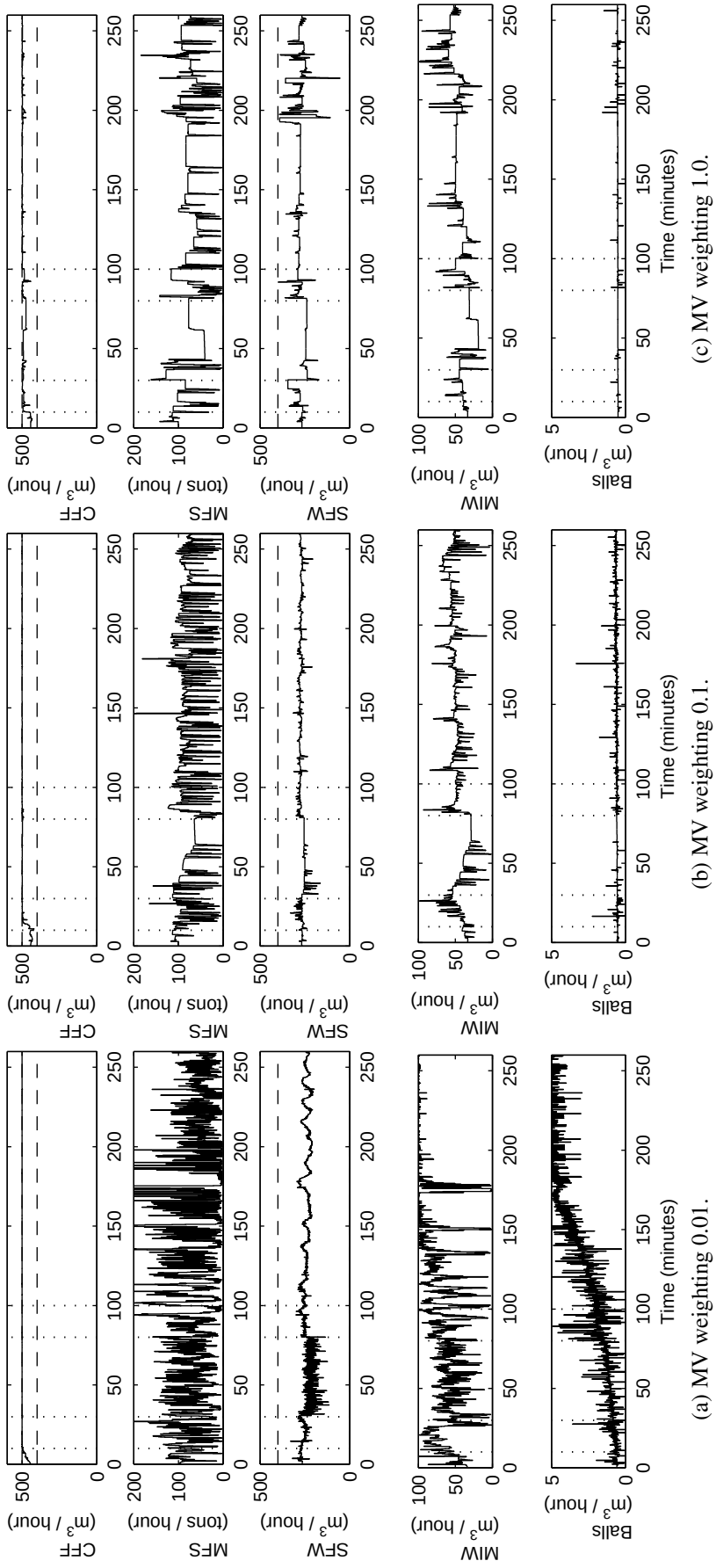
(c) THROUGHPUT weighting 20.

Figure B.6: MVs of the NMPC.
The NMPC regulates PSE and LOAD at constant setpoints and throughout at a constant setpoint of 100 tons/hour. The weighting of THROUGHPUT in the objective function is 5 (a), 10 (b) and 20 (c), while the weighting on PSE and LOAD is 100 respectively. Increasing the weighting on THROUGHPUT caused increasingly oscillatory behaviour in the MVs. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) MV weighting 0.01. (b) MV weighting 0.1. (c) MV weighting 1.0.

Figure B.7: CVs of the NMPC.
The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to reduce the oscillation in the MVs. The increased weighting on the MVs results in better tracking of PSE and higher average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.8: MVs of the NMPC.
The weighting on the MVs is increased from 0.01 (b) to 0.1 (b) and 1.0 (c) in order to reduce the oscillation in the MVs. The increased weighting on the MVs results in better tracking of PSE and higher average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

B.2 ADDITIONAL SIMULATION SCENARIOS

B.2.1 Setpoints on PSE, LOAD, POWER and RHEOLOGY

In this scenario, setpoints for POWER and RHEOLOGY are added to the objective function of the RNMPC and NMPC. It is believed that operating the mill at maximum power causes most breakage to occur and therefore the highest throughput to be obtained. The breakage is also most effective at the optimum slurry rheology. This simulation scenario is more of academic interest, because the maximum power value is not known for varying process conditions and the optimal rheology is also not known online. For the simulation model, however, the optimum values are known and used in these simulations to determine if it yields better performance compared to the previous simulation scenarios.

The closed-loop response with setpoints for POWER and RHEOLOGY is shown for the RNMPC in Figure B.9 and Figure B.10 and for the NMPC in Figure B.11 and Figure B.12. The weightings on the MVs are increased to gauge the effect of a slower response on the closed-loop performance.

The RNMPC (Figure B.9a and Figure B.10a) and the NMPC (Figure B.11a and Figure B.12a) show very good closed-loop performance with PSE tracking being tighter and the average throughput being higher compared to the scenario where POWER and RHEOLOGY are not included in the objective function (RNMPC: Figure 5.14a & Figure 5.15a; NMPC: Figure 5.14b & Figure 5.15b).

Table B.1 shows that increasing the weighting on the MVs results in larger PSE and LOAD setpoint tracking errors for the RNMPC (Figure B.9b and Figure B.10c) and the NMPC (Figure B.11b and Figure B.12c), as would be expected, because the system responds more slowly to disturbances.

This scenario does improve on the scenarios where POWER and RHEOLOGY are not included in the objective function, because it reduces the variability of these two variables from their optimum values. Practically, the optimum values of POWER and RHEOLOGY would somehow need to be estimated, as well as the actual value for RHEOLOGY. Fixing these values also reduces the freedom of the controller to follow the more important variables, such as PSE.

B.2.2 Increase the weighting on the MVs for objective function with only PSE and LOAD setpoints

In this scenario, the effect of increasing the weighting on the MVs is investigated where only PSE and LOAD are included in the objective function. In Section 5.3.5 and Section B.1.3, increasing the weighting on the MVs improved the closed-loop performance. At first this seems counter-intuitive, but there were large variations and even oscillations occurring on

the MVs. Increasing the weighting reduced the variations and consequently improved the closed-loop performance of the process.

The closed-loop performance of the process under RNMPC (Figure B.13 and Figure B.14) and NMPC (Figure B.15 and Figure B.16) shows increased PSE and LOAD setpoint tracking errors as the weightings on the MVs are increased and the MVs show smaller variations. This is expected, because the controllers are slower to respond to disturbances and therefore increase the variation of the process variables around their setpoints.

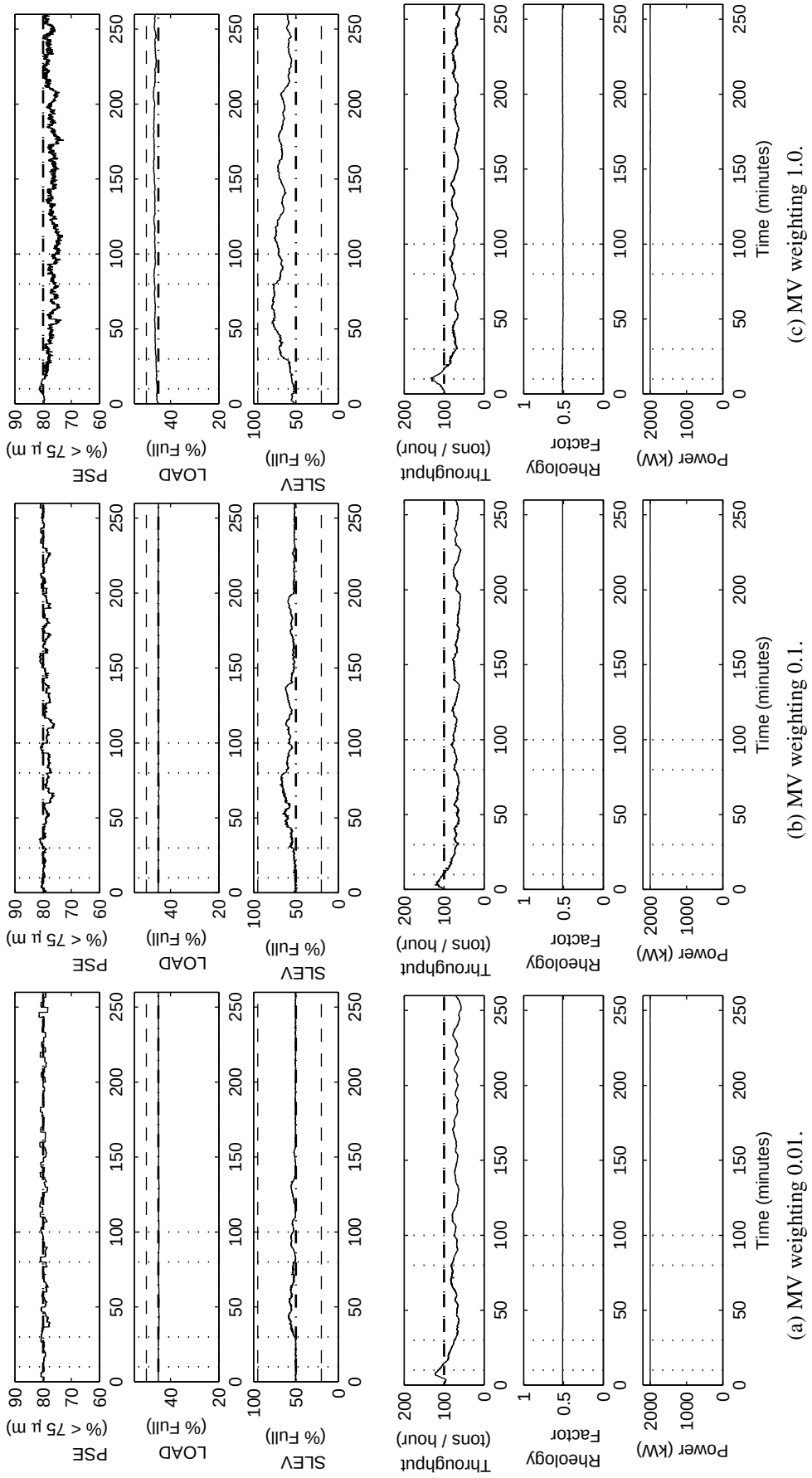
In this scenario where THROUGHPUT was not included in the objective function, increasing the weightings on the MVs reduced the closed-loop performance of the process with regard to PSE and LOAD setpoint tracking.

B.2.3 Increase weightings on the MVs for objective functions with PSE, LOAD and THROUGHPUT setpoints

This is similar to Section 5.3.5 and Section B.1.3 to see if the closed-loop performance for the RNMPC (Figure 5.24c and Figure 5.25c) and NMPC (Figure B.5c and Figure B.6c) with a weighting of 20 for THROUGHPUT in the objective function improves if the weightings on the MVs are increased.

The RNMPC (Figure B.17 and Figure B.18) and the NMPC (Figure B.19 and Figure B.20) show improved PSE setpoint tracking and an increase in the average throughput when the weightings on the MVs are increased from 0.01 to 1.0.

Including THROUGHPUT in the objective function changes the relative weighting between the CVs and the MVs, causing the overall gain of the controller to increase. The increased gain causes the controllers to make large control moves, which ultimately leads to sub-optimal closed-loop performance. Therefore, increasing the weightings on the MVs when THROUGHPUT is included in the objective function improved the overall closed-loop performance of the process.

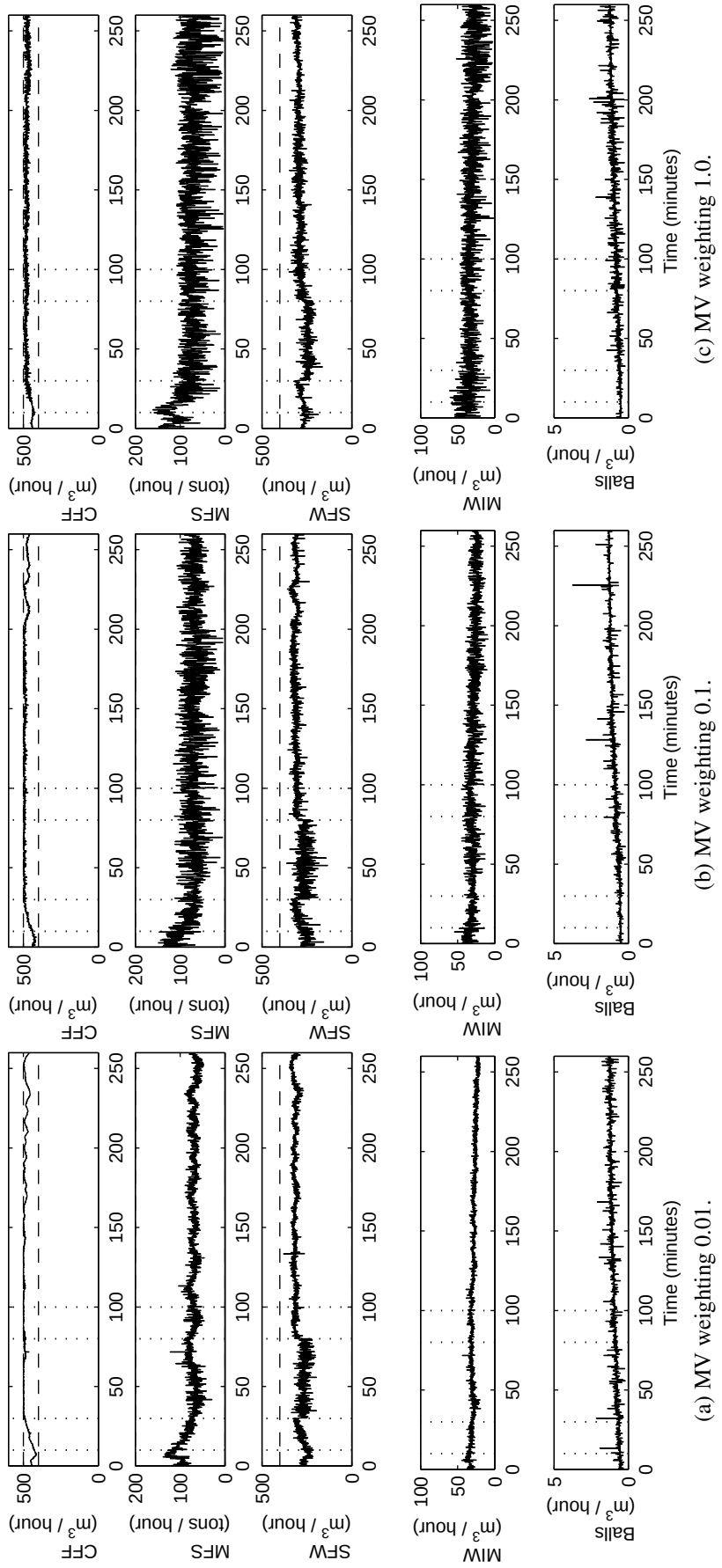


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.9: CVs of the RNMPC.
This scenario includes setpoints for the POWER and RHEOLOGY at their optimum values in the objective function. The added setpoints improve PSE setpoint tracking and the average THROUGHPUT. The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movements of the MVs on the setpoint tracking performance of the CVs. The increased weighting on the MVs results in poorer tracking of the PSE and LOAD setpoints while increasing the average THROUGHPUT slightly. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

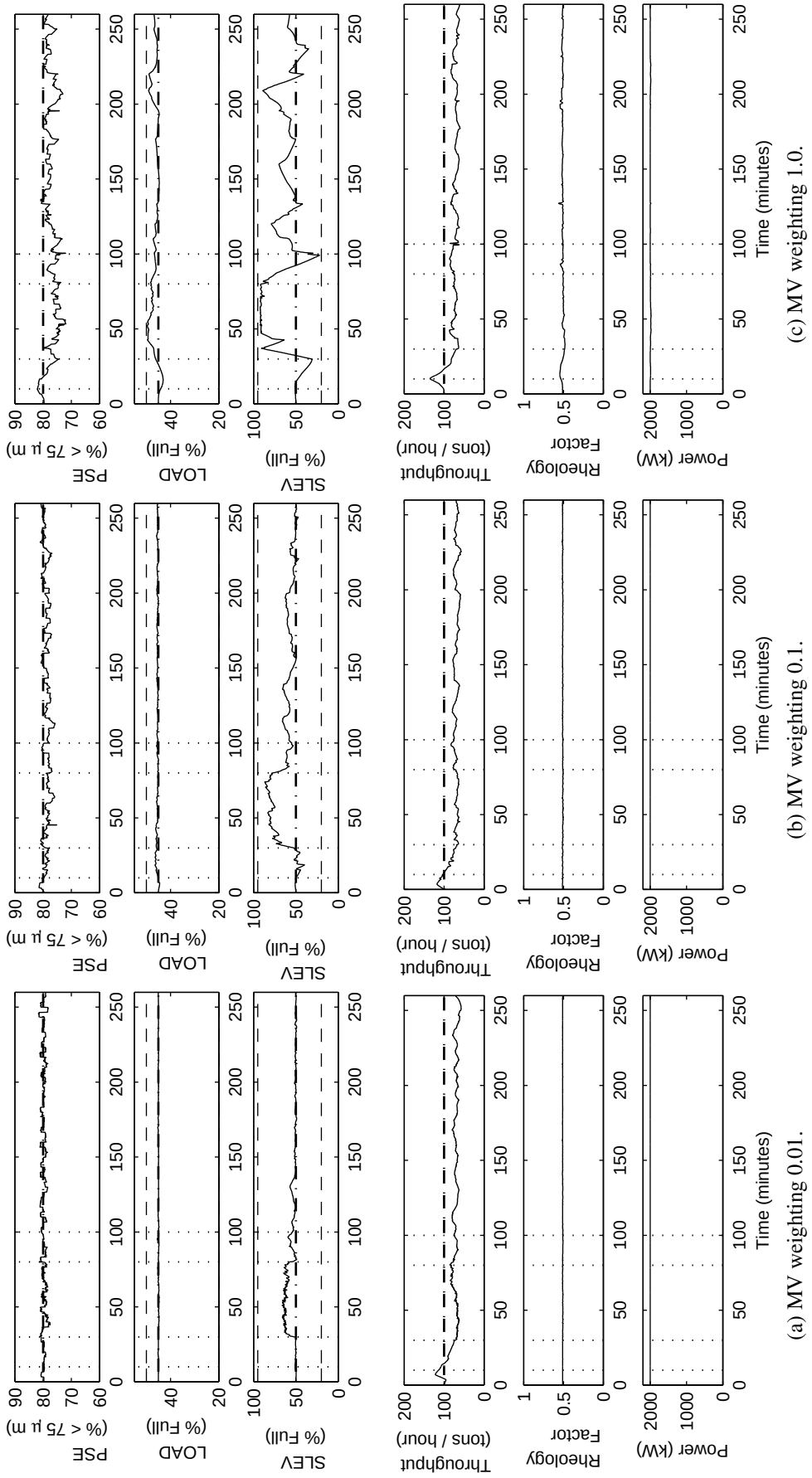


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.10: MVs of the RNMPC.
This scenario includes setpoints for the POWER and RHEOLOGY at their optimum values in the objective function. The added setpoints improve PSE setpoint tracking and the average THROUGHPUT. The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movements of the MVs on the setpoint tracking performance of the CVs. The increased weighting on the MVs results in poorer tracking of the PSE and LOAD setpoints while increasing the average THROUGHPUT slightly. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

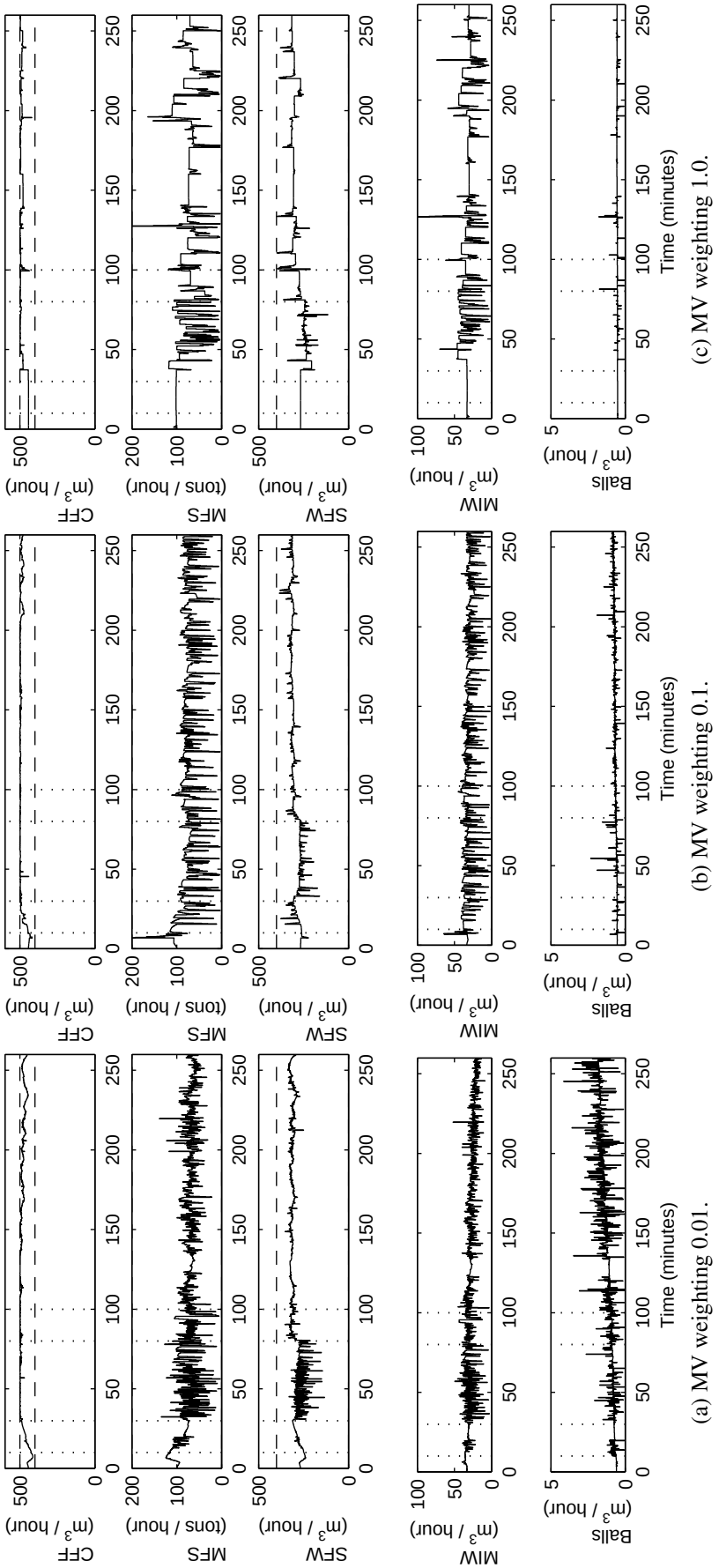


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.11: CVs of the NMPC.
This scenario includes setpoints for the POWER and RHEOLOGY at their optimum values in the objective function. The added setpoints improve PSE setpoint tracking and the average THROUGHPUT. The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movements of the MVs on the setpoint tracking performance of the CVs. The increased weighting on the MVs results in poorer tracking of the PSE and LOAD setpoints while increasing the average THROUGHPUT slightly. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

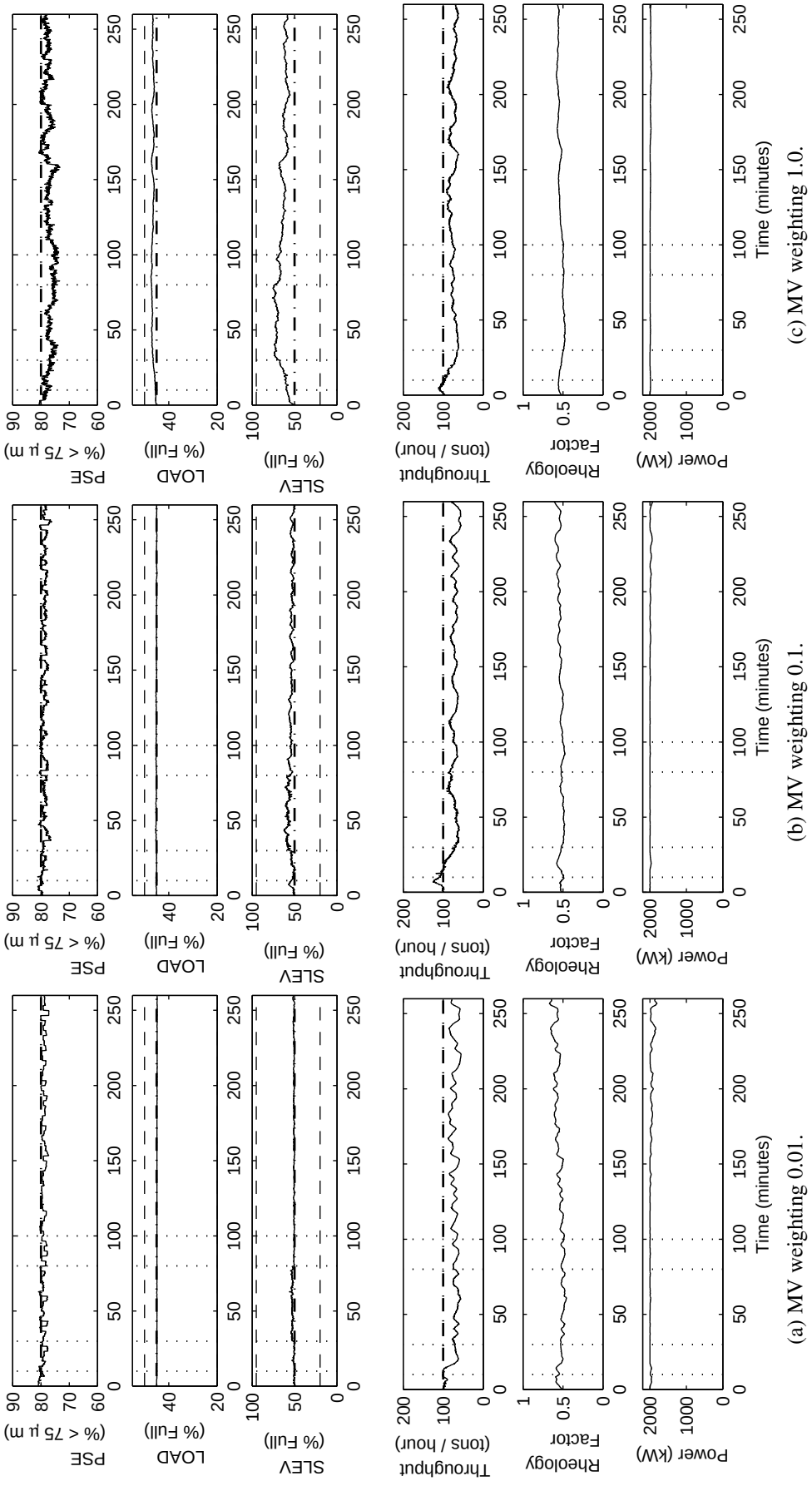


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.12: MVs of the NMPC.
This scenario includes setpoints for the POWER and RHEOLOGY at their optimum values in the objective function. The added setpoints improve PSE setpoint tracking and the average THROUGHPUT. The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movements of the MVs on the setpoint tracking performance of the CVs. The increased weighting on the MVs results in poorer tracking of the PSE and LOAD setpoints while increasing the average THROUGHPUT slightly. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

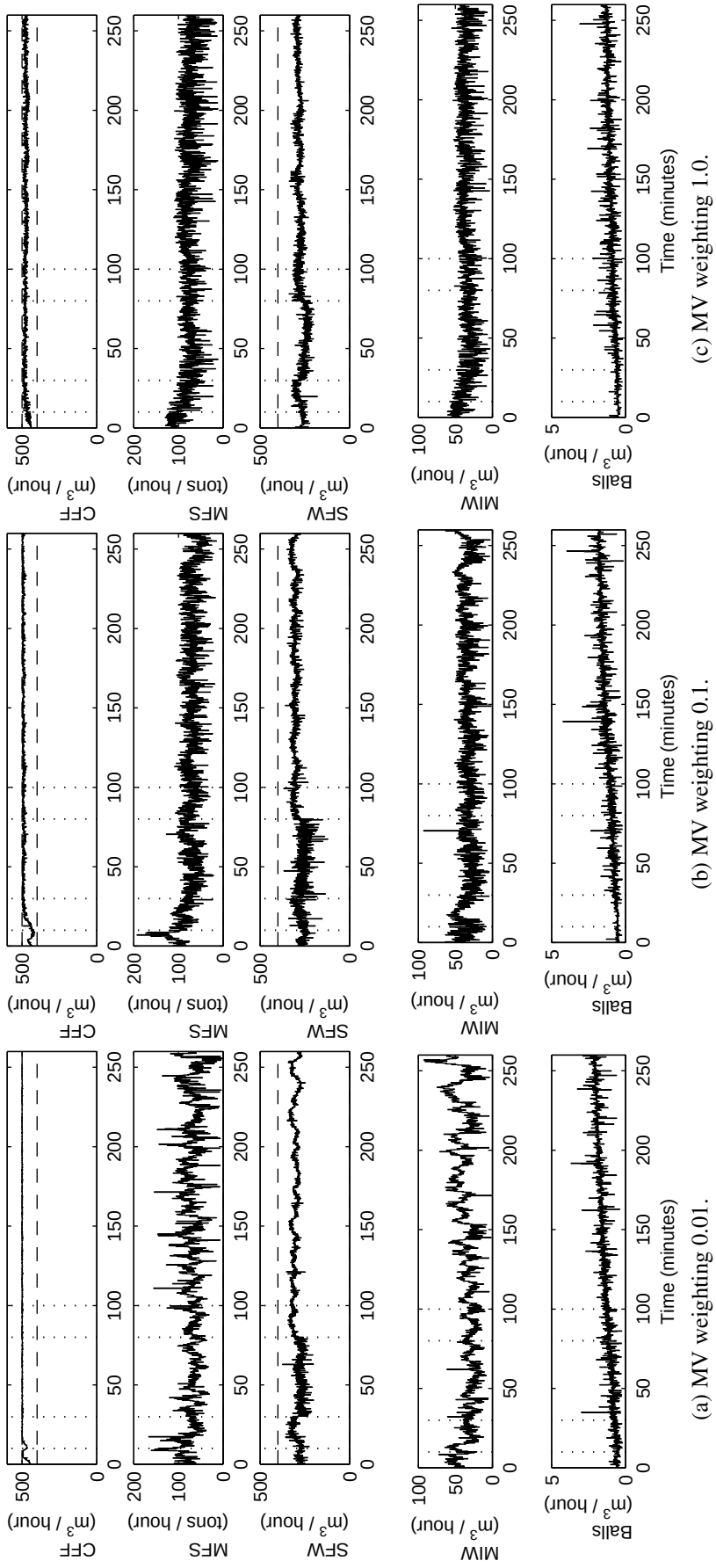


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.13: CVs of the RNMPC.
The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movement of the MVs on the setpoint tracking of the CVs. The increased weighting on the MVs results in poorer tracking of the PSE and the LOAD setpoints while increasing the average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

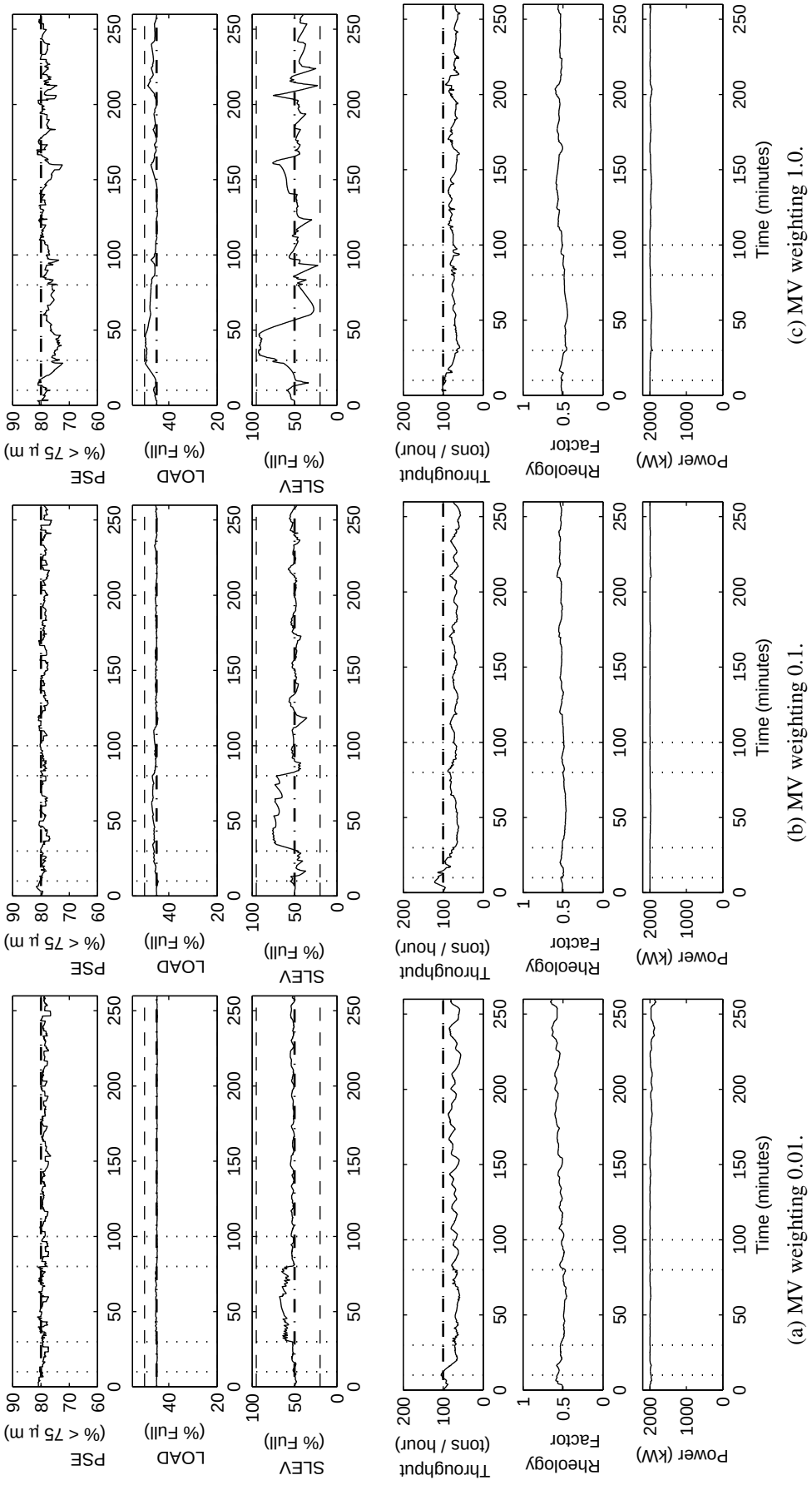


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.14: MVs of the RNMPC.
The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movement of the MVs on the setpoint tracking of the CVs. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

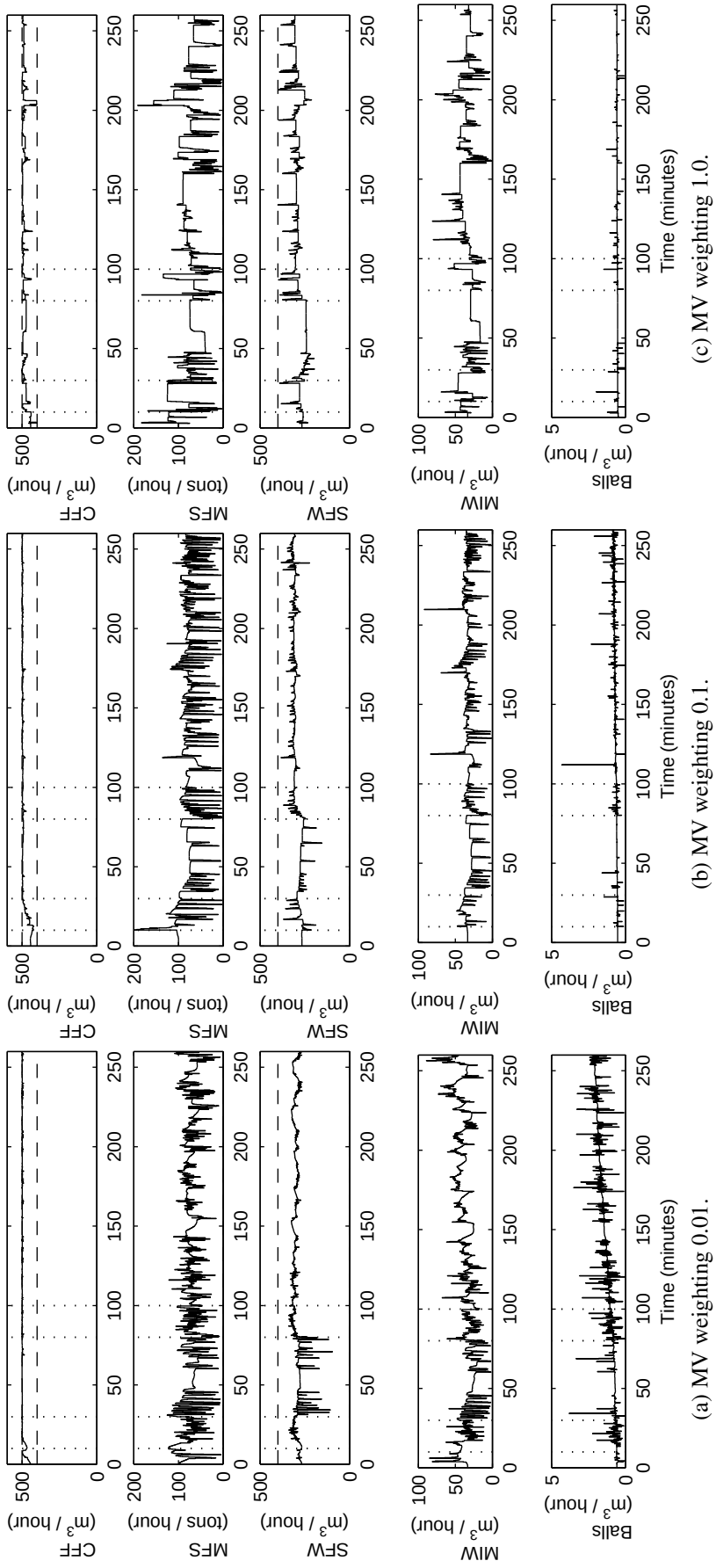


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.15: CVs of the NMPC.
The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movement of the MVs on the setpoint tracking of the CVs. The increased weighting on the MVs results in poorer tracking of the PSE and the LOAD setpoints while increasing the average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

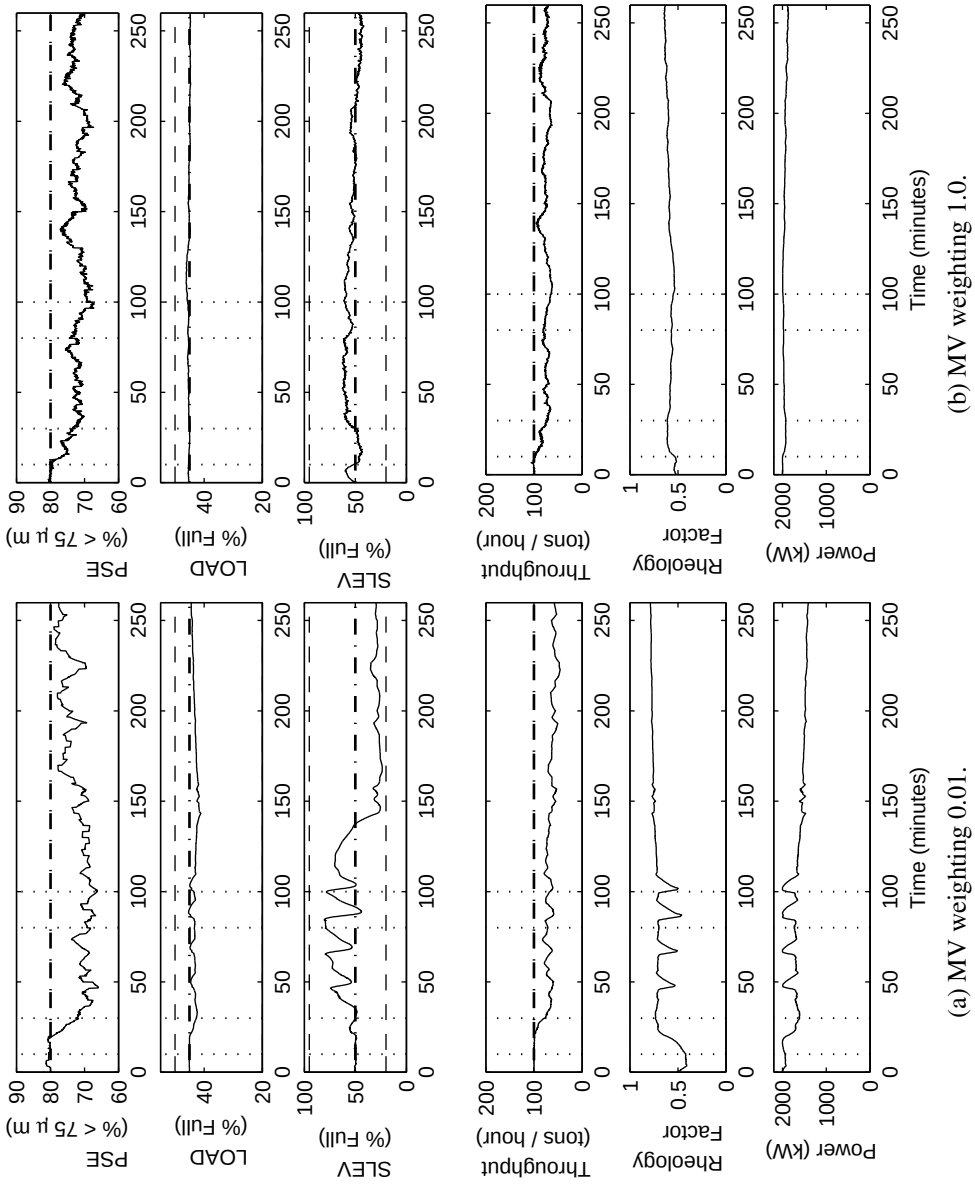


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

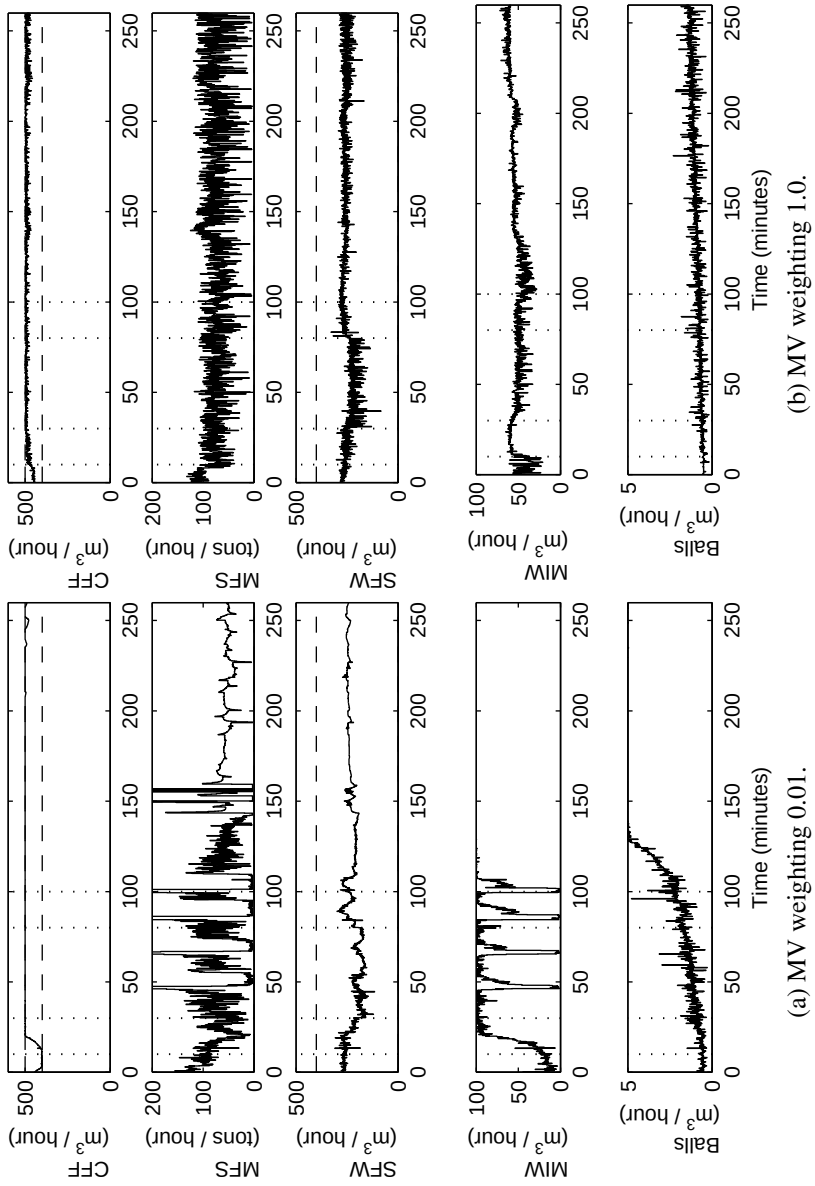
Figure B.16: MVs of the RNMPC.
The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movement of the MVs on the setpoint tracking of the CVs. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) MV weighting 0.01.

(b) MV weighting 1.0.

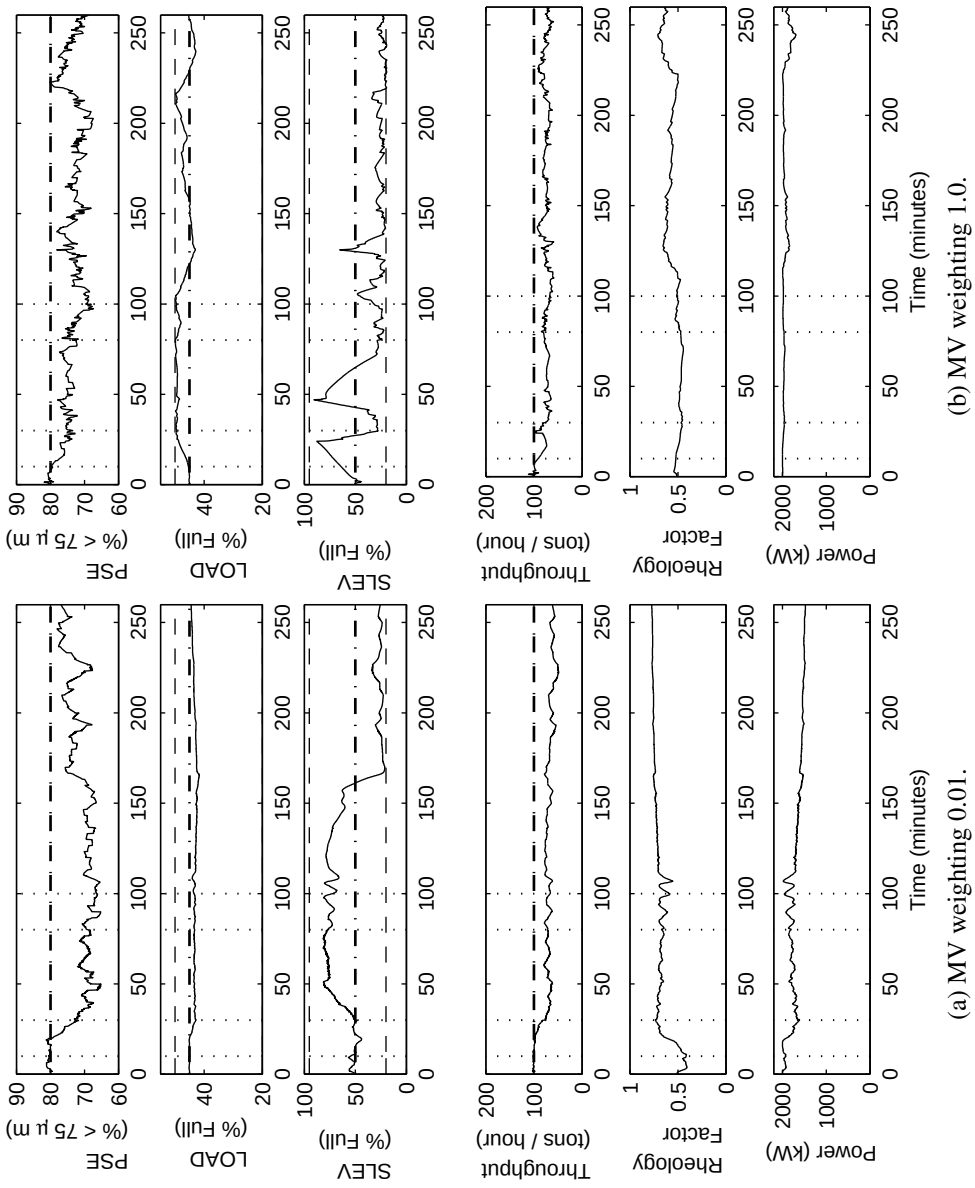
Figure B.17: CVs of the RNMPC.
The weighting on the MVs is increased from 0.01 (a) and 1.0 (b) in order to reduce the oscillations of the MVs. The increased weighting on the MVs results in better tracking of the PSE setpoint while increasing the average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) MV weighting 0.01.

(b) MV weighting 1.0.

Figure B.18: MVs of the RNMPC.
The weighting on the MVs is increased from 0.01 (a) and 1.0 (b) in order to reduce the oscillations of the MVs. The increased weighting on the MVs results in better tracking of the PSE setpoint while increasing the average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) MV weighting 0.01.

(b) MV weighting 1.0.

Figure B.19: CVs of the NMPC.
The weighting on the MVs is increased from 0.01 (a) and 1.0 (b) in order to reduce the oscillations of the MVs. The increased weighting on the MVs results in better tracking of the PSE setpoint while increasing the average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

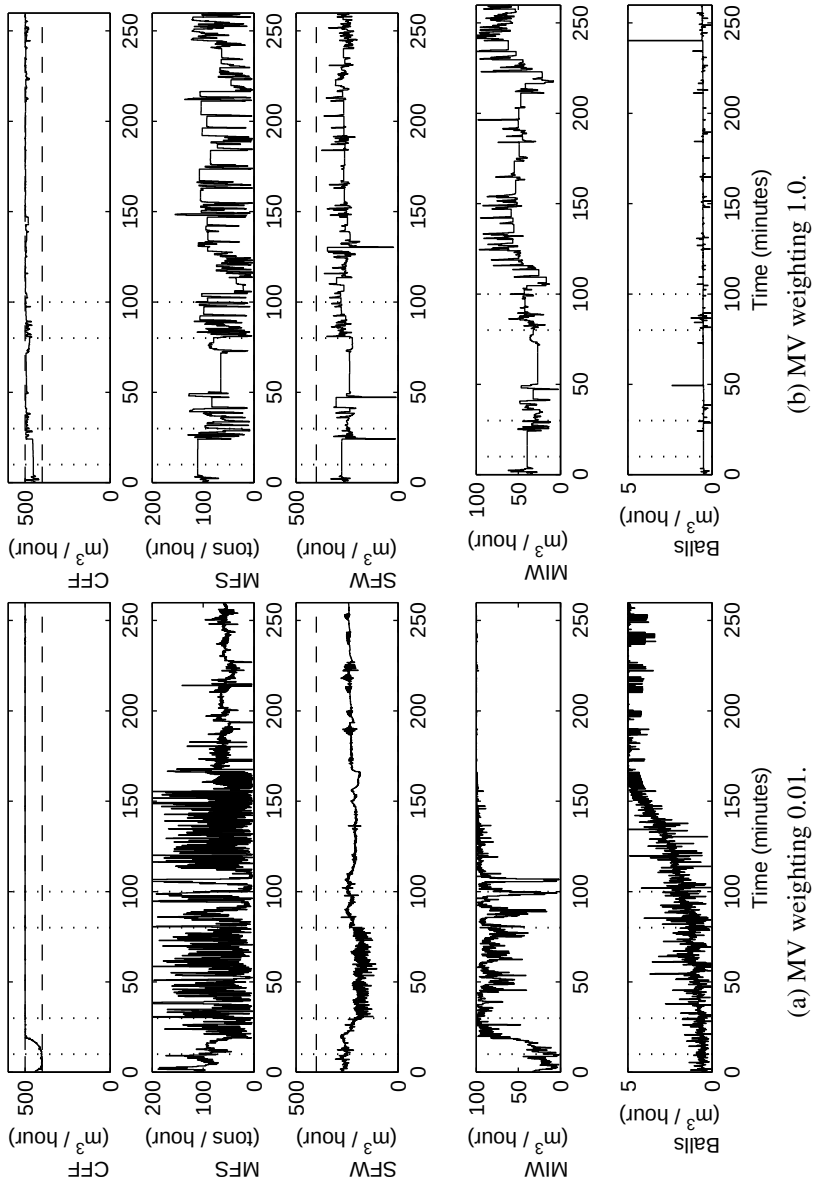


Figure B.20: MVs of the NMPC.
The weighting on the MVs is increased from 0.01 (a) and 1.0 (b) in order to reduce the oscillations of the MVs. The increased weighting on the MVs results in better tracking of the PSE setpoint while increasing the average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

B.3 SIMULATION SUMMARY

A summary of the simulation results are given in Table B.1 that details the tracking performance of the controller with regard to the particle size (PSE), mill load level (LOAD) and throughput. The simulation scenario is outlined in terms of the controlled variable weighting and setpoint values, as well as the step disturbances. The relevant changes in each simulation scenario are highlighted in bold. The dashes in the table represent values that are either zero or not applicable. The performance metrics are described in Section 5.2.

The headings of Table B.1 are defined as

PSE	Particle Size Estimate. [$\% < 75\mu$m]
LOAD	The volumetric filling of the mill. [%]
Throughput	The amount of solids discharged at the cyclone overflow. [$^{\text{tons}}/_{\text{hour}}$]
SLEV	Sump level. [m^3]
Power	The electrical power draw of the mill motor. [kW]
Rheology	An indication of the fluidity of the slurry inside the mill. [fraction]
U	The manipulated variables.
Disturbances	Describes the step disturbances in ore hardness, fraction of rock in the feed ore and SFW.
Time	Describes the average and maximum iteration time of the simulations [seconds].

The subheadings of Table B.1 are defined as

Δ	The sum of the squares of the error from the setpoint.
S	The setpoint of the variable.
W	The weight of the variable in the objective function.
A	The average value of the variable over the simulation duration.
RH	The increase in the hardness of the feed ore. [%]
AR	The increase in the fraction of rock in the feed ore. [%]
SFW	The increase in SFW. [$^{m^3}/_{\text{hour}}$]
T	The time when the disturbance is introduced.
M	The maximum value of the variable over the simulation duration.

The controllers are identified next to the figure numbers in Table B.1 by

R	Robust Nonlinear Model Predictive Controller.
N	Nonlinear Model Predictive Controller.
P	Proportional-Integral-Differential Controller.

The simulations show that RNMPC and NMPC are capable of tighter control of PSE, especially when constraints are active, because the multivariable controllers can leverage the multivariable nature of the milling circuit to increase the control envelope. The RNMPC and NMPC controllers are, however, not capable of improving throughput while maintaining PSE at the desired setpoint. The PI controllers performed very well, because they were tuned very aggressively. In certain milling circuits there are large time delays that result in less aggressive tuning of the PI controllers and degraded performance. MPC controllers were found in practice to perform well over longer periods compared to PI controllers (Chen *et al.*, 2007*b*, Ramasamy *et al.*, 2005).

Table B.1: Auxiliary simulation summary.

			PSE			LOAD			Throughput			SLEV		Power		Rheology		U	Disturbances					Time	
Simulation			Δ	S	W	Δ	S	W	A	S	W	S	W	S	W	S	W	W	RH	T	AR	T	S	A	M
Figure B.1	a	N	0.05	80	100	1.80E-05	45	100	106.31	99.9	0	5	**0**	2000	0	0.51	0	0.01	—	—	—	—	—	0.1	2.25
Figure B.2	b	N	0.07	80	100	4.80E-05	45	100	110.27	99.9	0	5	**1**	2000	0	0.51	0	0.01	—	—	—	—	—	0.1	1.6
Figure B.3	a	N	0.30	**80**	100	5.45E-03	45	100	72.54	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.2	1.77
Figure B.4	b	N	0.19	**75**	100	8.36E-03	45	100	74.33	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	2.23
	c	N	0.17	**70**	100	6.73E-03	45	100	81.87	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	2.08
Figure B.5	a	N	3.31	80	100	1.06E-01	45	100	71.45	99.9	**5**	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	2.04
Figure B.6	b	N	7.89	80	100	3.43E-01	45	100	69.25	99.9	**10**	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	1.77
	c	N	13.01	80	100	5.00E-01	45	100	70.83	99.9	**20**	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	1.38
Figure B.7	a	N	7.89	80	100	3.43E-01	45	100	69.25	99.9	**10**	5	1	2000	0	0.51	0	**0.01**	50	10	50	100	50	30-80	0.3	1.77
Figure B.8	b	N	4.71	80	100	5.86E-02	45	100	75.56	99.9	**10**	5	1	2000	0	0.51	0	**0.1**	50	10	50	100	50	30-80	0.2	2.3
	c	N	4.79	80	100	7.87E-01	45	100	75	99.9	**10**	5	1	2000	0	0.51	0	**1**	50	10	50	100	50	30-80	0.2	6.43
Figure B.9	a	R	0.1	80	100	1.62E-04	45	100	73.04	99.9	**0**	5	1	2000	**50**	0.51	**50**	**0.01**	50	10	50	100	50	30-80	36.2	665
Figure B.10	b	R	0.27	80	100	1.05E-03	45	100	71.43	99.9	**0**	5	1	2000	**50**	0.51	**50**	**0.1**	50	10	50	100	50	30-80	61.3	1168
	c	R	1.63	80	100	9.98E-03	45	100	74.81	99.9	**0**	5	1	2000	**50**	0.51	**50**	**1.0**	50	10	50	100	50	30-80	34.4	204
Figure B.11	a	N	0.1	80	100	2.49E-04	45	100	73.03	99.9	**0**	5	1	2000	**50**	0.51	**50**	**0.01**	50	10	50	100	50	30-80	0.2	2.07
Figure B.12	b	N	0.42	80	100	2.04E-03	45	100	71.93	99.9	**0**	5	1	2000	**50**	0.51	**50**	**0.1**	50	10	50	100	50	30-80	0.2	2.13
	c	N	1.81	80	100	9.88E-03	45	100	74.7	99.9	**0**	5	1	2000	**50**	0.51	**50**	**1.0**	50	10	50	100	50	30-80	0.2	2.99
Figure B.13	a	R	0.23	80	100	2.03E-03	45	100	72.22	99.9	**1**	5	1	2000	0	0.51	0	**0.01**	50	10	50	100	50	30-80	34.9	1014
Figure B.14	b	R	0.29	80	100	2.75E-03	45	100	73.82	99.9	**1**	5	1	2000	0	0.51	0	**0.1**	50	10	50	100	50	30-80	50.0	1295
	c	R	1.53	80	100	1.12E-02	45	100	76.07	99.9	**1**	5	1	2000	0	0.51	0	**1.0**	50	10	50	100	50	30-80	25.8	348
Figure B.15	a	N	0.30	80	100	5.45E-03	45	100	72.54	99.9	**1**	5	1	2000	0	0.51	0	**0.01**	50	10	50	100	50	30-80	0.2	1.77
Figure B.16	b	N	0.28	80	100	9.11E-02	45	100	73.79	99.9	**1**	5	1	2000	0	0.51	0	**0.1**	50	10	50	100	50	30-80	0.2	2.41
	c	N	1.31	80	100	6.07E-01	45	100	75.1	99.9	**1**	5	1	2000	0	0.51	0	**1**	50	10	50	100	50	30-80	0.2	3.75
Figure B.17	a	R	10.01	80	100	6.85E-02	45	100	68.09	99.9	**20**	5	1	2000	0	0.51	0	**0.01**	50	10	50	100	50	30-80	13.3	682
Figure B.18	b	R	9.19	80	100	4.38E-02	45	100	76.83	99.9	**20**	5	1	2000	0	0.51	0	**1.0**	50	10	50	100	50	30-80	44.7	1171
Figure B.19	a	N	13.01	80	100	5.00E-01	45	100	70.83	99.9	**20**	5	1	2000	0	0.51	0	**0.01**	50	10	50	100	50	30-80	0.3	1.38
Figure B.20	b	N	7.52	80	100	1.32E+00	45	100	75.73	99.9	**20**	5	1	2000	0	0.51	0	**1**	50	10	50	100	50	30-80	0.2	3.77

ADDENDUM C

PID TUNING WITH INTERACTIONS

In this addendum, the problem of tuning decentralised PID controllers while taking interaction of the MIMO system into account, is studied. There are two main approaches to handling interactions, the first is to design the decentralised controllers by taking interaction into account. Three main approaches exist to handle interaction in decentralised PID design: (1) by detuning the controllers to account for interaction, (2) using the critical gains of the system to tune the controllers and (3) using the whole transfer function to explicitly take interaction into account (Vázquez and Morilla, 2002). The second approach is to do a fully centralised controller design (Morilla *et al.*, 2008). A hybrid approach is to first design a decoupling network to minimise interaction between the loops and then apply decentralised methods to the decoupled plant (Vázquez and Morilla, 2002).

C.1 INTRODUCTION

Desbiens *et al.* (1996) presents a frequency-domain method to design decentralised PID controllers for a two-input-two-output (TITO) system while explicitly taking interaction into account. Pomerleau *et al.* (2000) applied the method by Desbiens *et al.* (1996) to a milling circuit to design the decentralised PID controllers and will be the method used in this addendum. The method by Desbiens *et al.* (1996) falls under the third approach of decentralised PID controller design methods, as described by Vázquez and Morilla (2002).

The method by Desbiens *et al.* (1996) aims to design decentralised controllers $G_{C1}(s)$ and $G_{C2}(s)$ for the 2×2 system

$$G(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \tag{C.1}$$

where the decentralised controllers have the following structure

$$G_C(s) = \begin{bmatrix} G_{C1}(s) & 0 \\ 0 & G_{C2}(s) \end{bmatrix} \tag{C.2}$$

and the closed-loop system that forms by combining the decentralised controllers $G_C(s)$ in a feedback loop with the 2×2 system $G(s)$ is given by

$$T(s) = \frac{G(s)G_C(s)}{I_2 + G(s)G_C(s)} \tag{C.3}$$

C.2 SUMP AND CYCLONE MODELS

The process transfer function $G_{11}(s)$ is given by (4.8) in Section 4.3.1 as

$$G_{11}(s) = -0.00035\frac{(1-0.63s)}{(1+0.54s)}e^{(-0.011s)} \tag{C.4}$$

and the process transfer function $G_{22}(s)$ is given by (4.12) in Section 4.3.3 as

$$G_{22}(s) = \frac{0.42}{s}. \tag{C.5}$$

The models that describe the interactions are given in the next subsections.

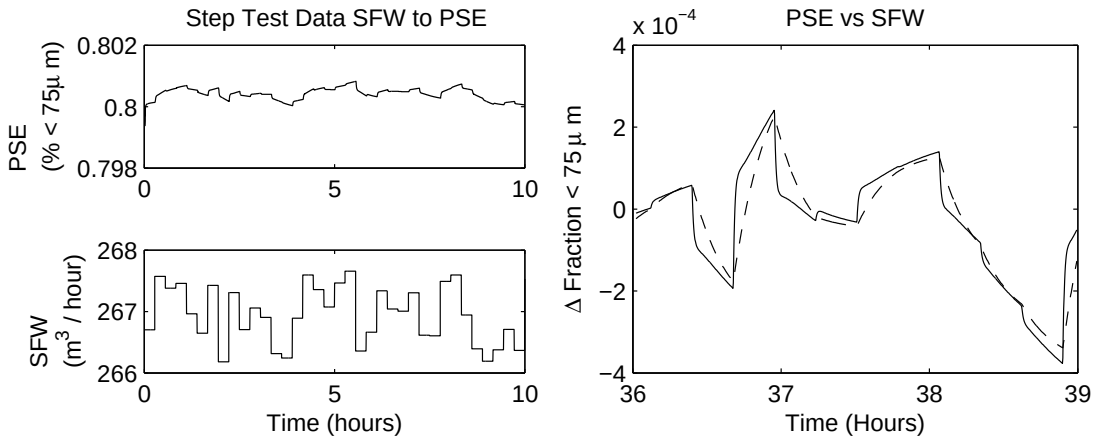
C.2.1 PSE – SFW model

The first interaction model describes the behaviour of PSE with a change in SFW. This model is represented by $G_{12}(s)$ in equation (C.1). PSE exhibits a first-order response to SFW and a first-order order transfer function model is fitted to the step test data of the nonlinear model with the following form:

$$G_{\text{PSE-SFW}}(s) = \frac{K_{\text{PS}}}{(1+P_{\text{PS}}s)}e^{(-\theta_{\text{PS}}s)} \tag{C.6}$$

$$= \frac{0.00055}{(1+0.24s)}e^{(-0.011s)} \tag{C.7}$$

The step response data for the model fitting as well as the comparison between the linear and nonlinear models are shown in Figure C.1. The linear model for PSE-CFF shows good agreement with the nonlinear model response.



(a) PSE vs SFW (b) Nonlinear vs Linear model

Figure C.1: The change in PSE with a step change in SFW. The nonlinear (solid line) model is compared to the linear model (dashed line).

C.2.2 SLEV – CFF model

The second interaction model describes the behaviour of SLEV with a change in CFF. This model is represented by $G_{21}(s)$ in equation (C.1). SLEV exhibits an integrating response to CFF and an integrator transfer function model is fitted to the step test data of the nonlinear model with the following form:

$$G_{\text{SLEV-CFF}}(s) = \frac{K_{\text{PS}}}{s} \tag{C.8}$$

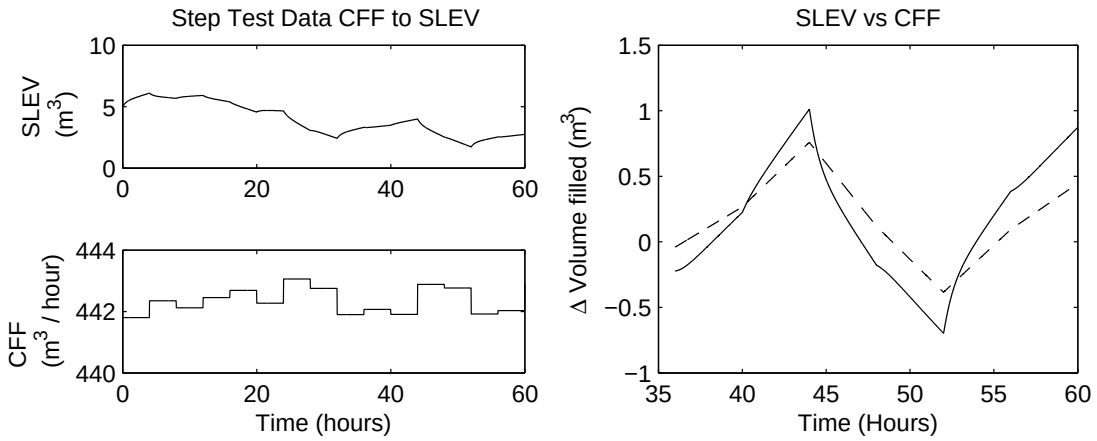
$$= \frac{-0.29}{s} \tag{C.9}$$

The step response data for the model fitting as well as the comparison between the linear and nonlinear models are shown in Figure C.2. The linear model for PSE-CFF shows good agreement with the nonlinear model response.

C.2.3 Interacting sump and cyclone model

The final model with interaction for equation (C.1) is given by

$$G(s) = \begin{bmatrix} -0.00035\frac{(1-0.63s)}{(1+0.54s)}e^{(-0.011s)} & \frac{0.00055}{(1+0.24s)}e^{(-0.011s)} \\ \frac{-0.29}{s} & \frac{0.42}{s} \end{bmatrix} \tag{C.10}$$



(a) SLEV vs CFF (b) Nonlinear vs Linear model

Figure C.2: The change in SLEV with a step change in CFF. The nonlinear (solid line) model is compared to the linear model (dashed line).

C.3 FREQUENCY BASED SPECIFICATIONS (FBS) TUNING METHOD

This section will give a quick overview of the FBS tuning method of Desbiens *et al.* (1996) before applying it to the interacting sump and cyclone model in the next section.

The FBS tuning method starts by describing the transfer functions that are seen by the two controllers for a 2×2 system $G(s)$ by

$$G_1(s) = G_{11}(s) - \frac{G_{12}(s)G_{21}(s)G_{C2}(s)}{1 + G_{C2}(s)G_{22}(s)} \tag{C.11}$$

$$G_2(s) = G_{22}(s) - \frac{G_{12}(s)G_{21}(s)G_{C1}(s)}{1 + G_{C1}(s)G_{11}(s)}. \tag{C.12}$$

The transfer functions (equation (C.11) and (C.12)) include the interacting controller that leads to the tuning of the one controller affecting the tuning of the other controller. There are two requirements for the closed-loop response:

1. No steady-state tracking errors, and
2. a closed-loop second-order tracking response of the setpoints.

The specifications for the tracking closed-loop responses can be specified using the following transfer function

$$\frac{Y_1(s)}{R_1(s)} = \frac{1 - \tau_{01}s}{(1 + \tau_{11}s)(1 + \tau_{21}s)} \tag{C.13}$$

$$\frac{Y_2(s)}{R_2(s)} = \frac{1 - \tau_{02}s}{(1 + \tau_{12}s)(1 + \tau_{22}s)} \tag{C.14}$$

that makes provision for non-minimum phase zeros in order to produce realisable controller designs for certain plants (Desbiens *et al.*, 1996). The closed-loop tracking specifications of equation (C.13) and (C.14) are translated into open-loop characteristics given by

$$\begin{aligned} G_{OL1}(s) &= G_{C1}(s)G_1(s) \\ &= \frac{1-\tau_{01}s}{s(\tau_{11}+\tau_{21}+\tau_{01}+\tau_{11}\tau_{21}s)} \end{aligned} \tag{C.15}$$

$$\begin{aligned} G_{OL2}(s) &= G_{C2}(s)G_2(s) \\ &= \frac{1-\tau_{02}s}{s(\tau_{12}+\tau_{22}+\tau_{02}+\tau_{12}\tau_{22}s)} \end{aligned} \tag{C.16}$$

The open-loop characteristics (equation (C.15) and (C.16)) together with the tracking closed-loop specifications (equation (C.13) and (C.14)) are needed to provide enough information to solve the unknowns for $G_{C1}(s)$ and $G_{C2}(s)$, because the tuning of the one controller is dependent on the tuning of the other controller. The two controllers $G_{C1}(s)$ and $G_{C2}(s)$ can be obtained from equation (C.15) and (C.16) by rewriting the equations as

$$A_1(s)G_{C1}^2(s)+A_2(s)G_{C1}(s)+A_3(s) = 0 \tag{C.17}$$

$$B_1(s)G_{C2}^2(s)+B_2(s)G_{C2}(s)+B_3(s) = 0 \tag{C.18}$$

where

$$A_1(s) = G_{11}(s)\left(G_{11}(s)G_{22}(s)-G_{12}(s)G_{21}(s)\right)\left(1+G_{OL2}(s)\right) \tag{C.19}$$

$$\begin{aligned} A_2(s) &= G_{11}(s)G_{22}(s)\left(1-G_{OL1}(s)G_{OL2}(s)\right) \\ &\quad +\left(G_{OL2}(s)-G_{OL1}(s)\right)\left(G_{11}(s)G_{22}(s)-G_{12}(s)G_{21}(s)\right) \end{aligned} \tag{C.20}$$

$$A_3(s) = -G_{OL1}(s)G_{22}(s)\left(1+G_{OL2}(s)\right) \tag{C.21}$$

$$B_1(s) = G_{22}(s)\left(G_{11}(s)G_{22}(s)-G_{12}(s)G_{21}(s)\right)\left(1+G_{OL1}(s)\right) \tag{C.22}$$

$$\begin{aligned} B_2(s) &= G_{11}(s)G_{22}(s)\left(1-G_{OL1}(s)G_{OL2}(s)\right) \\ &\quad +\left(G_{OL1}(s)-G_{OL2}(s)\right)\left(G_{11}(s)G_{22}(s)-G_{12}(s)G_{21}(s)\right) \end{aligned} \tag{C.23}$$

$$B_3(s) = -G_{OL2}(s)G_{11}(s)\left(1+G_{OL1}(s)\right) \tag{C.24}$$

The frequency response of $G_{C1}(s)$ and $G_{C2}(s)$ can be obtained from equation (C.17) and (C.18) by calculating the frequency responses of $A_1(s), A_2(s), A_3(s), B_1(s), B_2(s)$ and $B_3(s)$ on a frequency-by-frequency basis over a spread of frequencies: $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$. The spread of frequencies should be large enough to

- capture the integrating response at low frequencies,
- capture important behaviour at high frequencies, and
- include the closed-loop cross-over frequency.

Both controllers will have two possible frequency responses, because of the quadratic nature

of equation (C.17) and (C.18). The appropriate frequency response for each controller is chosen based on the facts that

- the controller should have an integrating response, and
- the required sign of the controller is known, based on $G_{11}(s)$ and $G_{22}(s)$.

The final step is to approximate the chosen frequency response for each controller ($G_{C1}(j\omega)$ and $G_{C2}(j\omega)$) by $G_{Cp1}(s)$ and $G_{Cp2}(s)$. The form of $G_{Cp1}(s)$ and $G_{Cp2}(s)$ proposed by Desbiens *et al.* (1996) is given by

$$G_{Cp1}(s) = \frac{K_{C1}\left(1+T_1 s\right)\left(1+T_{d11}s+T_{d21}s^2\right)}{T_1 s\left(1+T_{f11}s+T_{f21}s^2\right)} \tag{C.25}$$

$$G_{Cp2}(s) = \frac{K_{C2}\left(1+T_2 s\right)\left(1+T_{d12}s+T_{d22}s^2\right)}{T_2 s\left(1+T_{f12}s+T_{f22}s^2\right)} \tag{C.26}$$

with a second-order differential term and a second-order filtering term that is needed to accurately approximate the complex frequency responses $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$ produced by solving equation (C.17) and (C.18). The designs of $G_{Cp1}(s)$ and $G_{Cp2}(s)$ are first performed by hand and then refined using an optimisation technique, which is described in Desbiens *et al.* (1996).

C.4 PID CONTROLLER DESIGN

This section details the design scenarios investigated for controlling the interacting sump and cyclone model with decentralised PID for a number of closed-loop tracking specifications. The tracking specifications for the PSE-CFF loop ($Y_1(s)/R_1(s)$) must have a settling time of 5 minutes or $5/60 = 0.083$ hours. In order to reach a settling time of 0.083 hours, a dominant time constant of $0.083/4 \approx 0.021$ hours is required. The SLEV-SFW loop ($Y_2(s)/R_2(s)$) can be slower to act as a buffer to disturbances. The sign of the controller for the PSE-CFF loop should be negative or +90 degrees at low frequencies from $G_{11}(s)$ in equation (C.4). The sign of the controller for the SLEV-SFW loop should be positive or -90 degrees at low frequencies from $G_{22}(s)$ in equation (C.5). A few of the design scenarios that were tested are given in the following subsections.

C.4.1 Design scenario 1

This design is for first-order tracking specifications on both controllers, while using the full interacting model with time-delay. The time-delay is approximated using a first-order Padé

approximation. The model used for this design scenario is given by

$$G_{D1}(s) = \begin{bmatrix} -0.00035\frac{(1-0.63s)}{(1+0.54s)} \cdot \frac{(-s+181.8)}{(s+181.8)} & \frac{0.00055}{(1+0.24s)} \cdot \frac{(-s+181.8)}{(s+181.8)} \\ \frac{-0.29}{s} & \frac{0.42}{s} \end{bmatrix} \tag{C.27}$$

The first-order tracking specifications are given by

$$\frac{Y_1(s)}{R_1(s)} = \frac{1}{(1+0.021s)} \tag{C.28}$$

$$\frac{Y_2(s)}{R_2(s)} = \frac{1}{(1+0.05s)} \tag{C.29}$$

The frequency responses for the two controllers $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$ are given in Figure C.3, which shows that design 1 for $G_{C1}(j\omega)$ has an integrating response, but it does not have the correct sign and neither of the two designs of $G_{C2}(j\omega)$ has an integrating response. The method, therefore, does not provide a feasible design for the model with the design specifications given in equation (C.31) and (C.32).

C.4.2 Design scenario 2

This design is for first-order tracking specifications on both controllers, while using the interacting model without time-delay. The time-delay can be brought back through a Smith predictor structure. The model used for this design scenario is given by

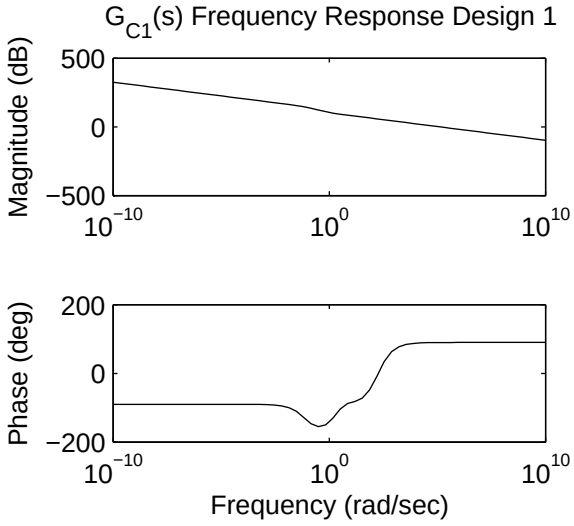
$$G_{D2}(s) = \begin{bmatrix} -0.00035\frac{(1-0.63s)}{(1+0.54s)} & \frac{0.00055}{(1+0.24s)} \\ \frac{-0.29}{s} & \frac{0.42}{s} \end{bmatrix} \tag{C.30}$$

The first-order tracking specifications are given by

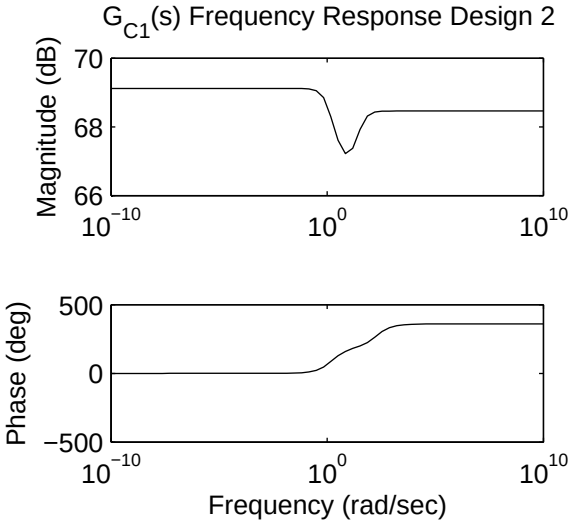
$$\frac{Y_1(s)}{R_1(s)} = \frac{1}{(1+0.021s)} \tag{C.31}$$

$$\frac{Y_2(s)}{R_2(s)} = \frac{1}{(1+0.05s)} \tag{C.32}$$

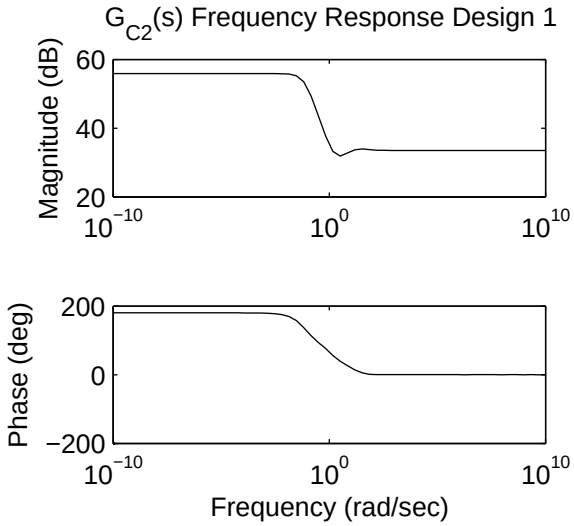
The frequency responses for the two controllers $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$ are given in Figure C.4, which shows that design 1 for $G_{C1}(j\omega)$ has an integrating response, but it does not have the correct sign and neither of the two designs of $G_{C2}(j\omega)$ has an integrating response. The method, therefore, does not provide a feasible design for the model of equation (C.30) with the design specifications given in equation (C.31) and (C.32).



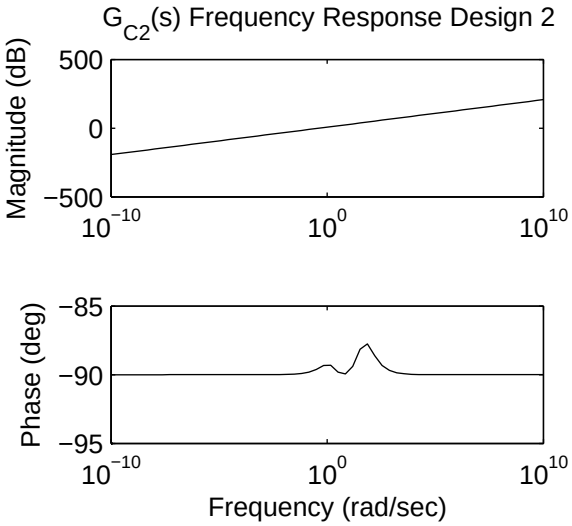
(a) $G_{C1}(j\omega)$ design 1.



(b) $G_{C1}(j\omega)$ design 2.

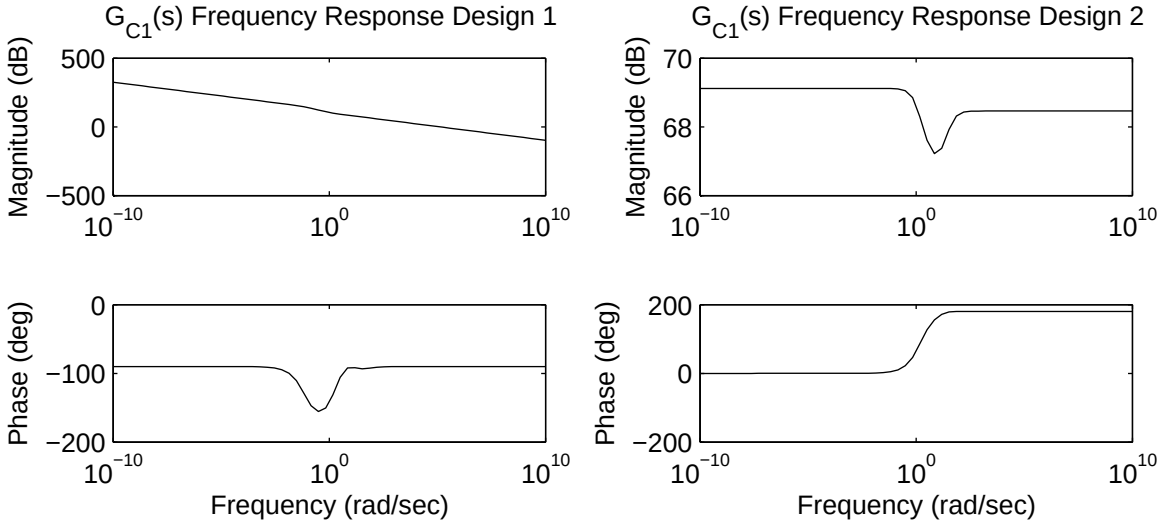


(c) $G_{C2}(j\omega)$ design 1.



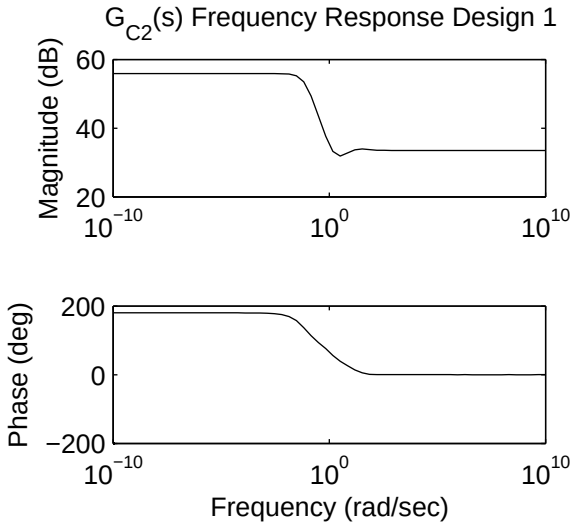
(d) $G_{C2}(j\omega)$ design 2.

Figure C.3: Design Scenario 1: Frequency response designs for $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$.

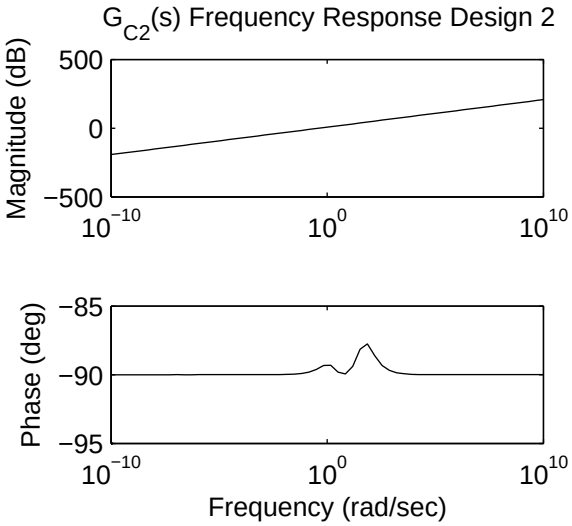


(a) $G_{C1}(j\omega)$ design 1.

(b) $G_{C1}(j\omega)$ design 2.



(c) $G_{C2}(j\omega)$ design 1.



(d) $G_{C2}(j\omega)$ design 2.

Figure C.4: Design Scenario 2: Frequency response designs for $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$.

C.4.3 Design scenario 3

This design is for first-order tracking specifications on both controllers, while using the interacting model without time-delay and the integrators approximated by fast first-order responses. The time-delay can be brought back through a Smith predictor structure. The model used for this design scenario is given by

$$G_{D3}(s) = \begin{bmatrix} -0.00035\frac{(1-0.63s)}{(1+0.54s)} & \frac{0.00055}{(1+0.24s)} \\ \frac{-0.29}{(s+0.001)} & \frac{0.42}{(s+0.001)} \end{bmatrix} \tag{C.33}$$

The first-order tracking specifications are given by

$$\frac{Y_1(s)}{R_1(s)} = \frac{1}{(1+0.021s)} \tag{C.34}$$

$$\frac{Y_2(s)}{R_2(s)} = \frac{1}{(1+0.05s)} \tag{C.35}$$

The frequency responses for the two controllers $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$ are given in Figure C.5, which shows that design 2 for $G_{C1}(j\omega)$ has an integrating response, but it does not have the correct sign and design 2 of $G_{C2}(j\omega)$ has an integrating response, but also with the incorrect sign. The method, therefore, does not provide a feasible design for the model in equation (C.33) with the design specifications given in equation (C.34) and (C.35).

C.4.4 Design scenario 4

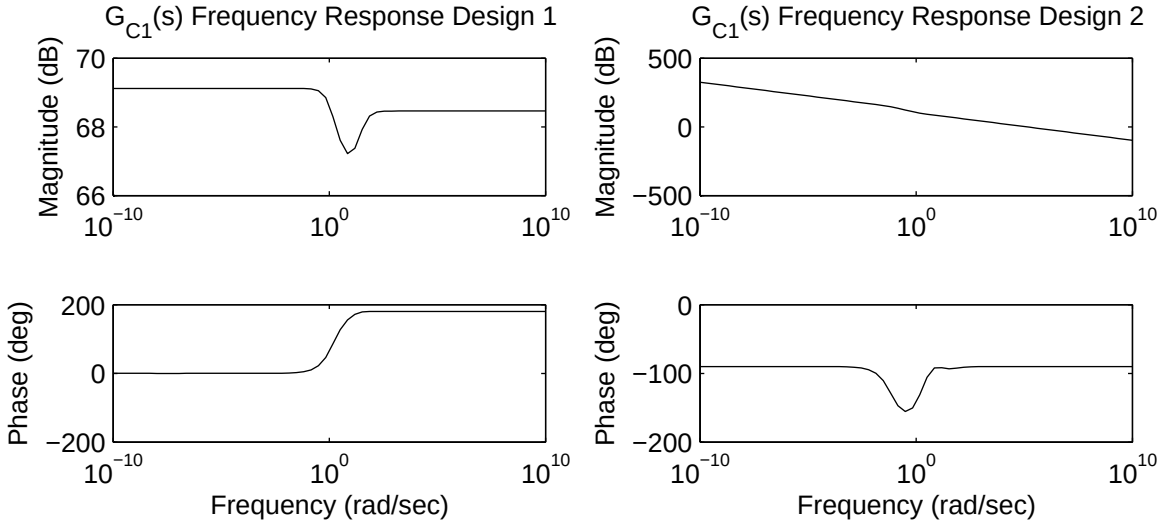
This design defines a second-order non-minimum phase closed-loop tracking specification for the PSE-CFF loop and a first-order closed-loop tracking specification for the SLEV-SFW loop, while using the interacting model without time-delay and the integrators approximated by fast first-order responses. The time-delay can be brought back through a Smith predictor structure. The model used for this design scenario is given by

$$G_{D4}(s) = \begin{bmatrix} -0.00035\frac{(1-0.63s)}{(1+0.54s)} & \frac{0.00055}{(1+0.24s)} \\ \frac{-0.29}{(s+0.001)} & \frac{0.42}{(s+0.001)} \end{bmatrix} \tag{C.36}$$

The closed-loop tracking specifications are given by

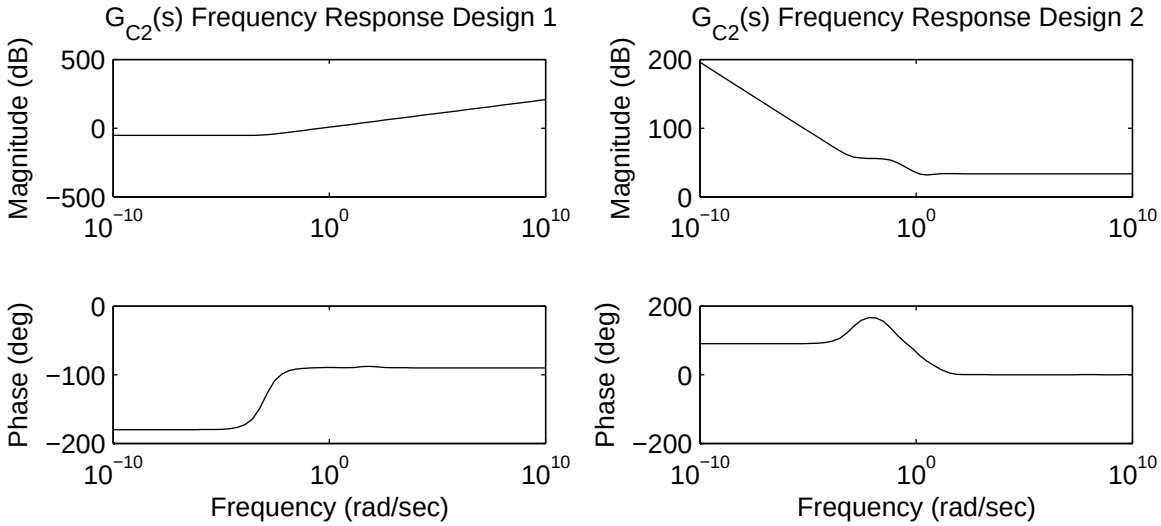
$$\frac{Y_1(s)}{R_1(s)} = \frac{(1-0.021s)}{(1+0.021s)(1+0.021s)} \tag{C.37}$$

$$\frac{Y_2(s)}{R_2(s)} = \frac{1}{(1+0.05s)} \tag{C.38}$$



(a) $G_{C1}(j\omega)$ design 1.

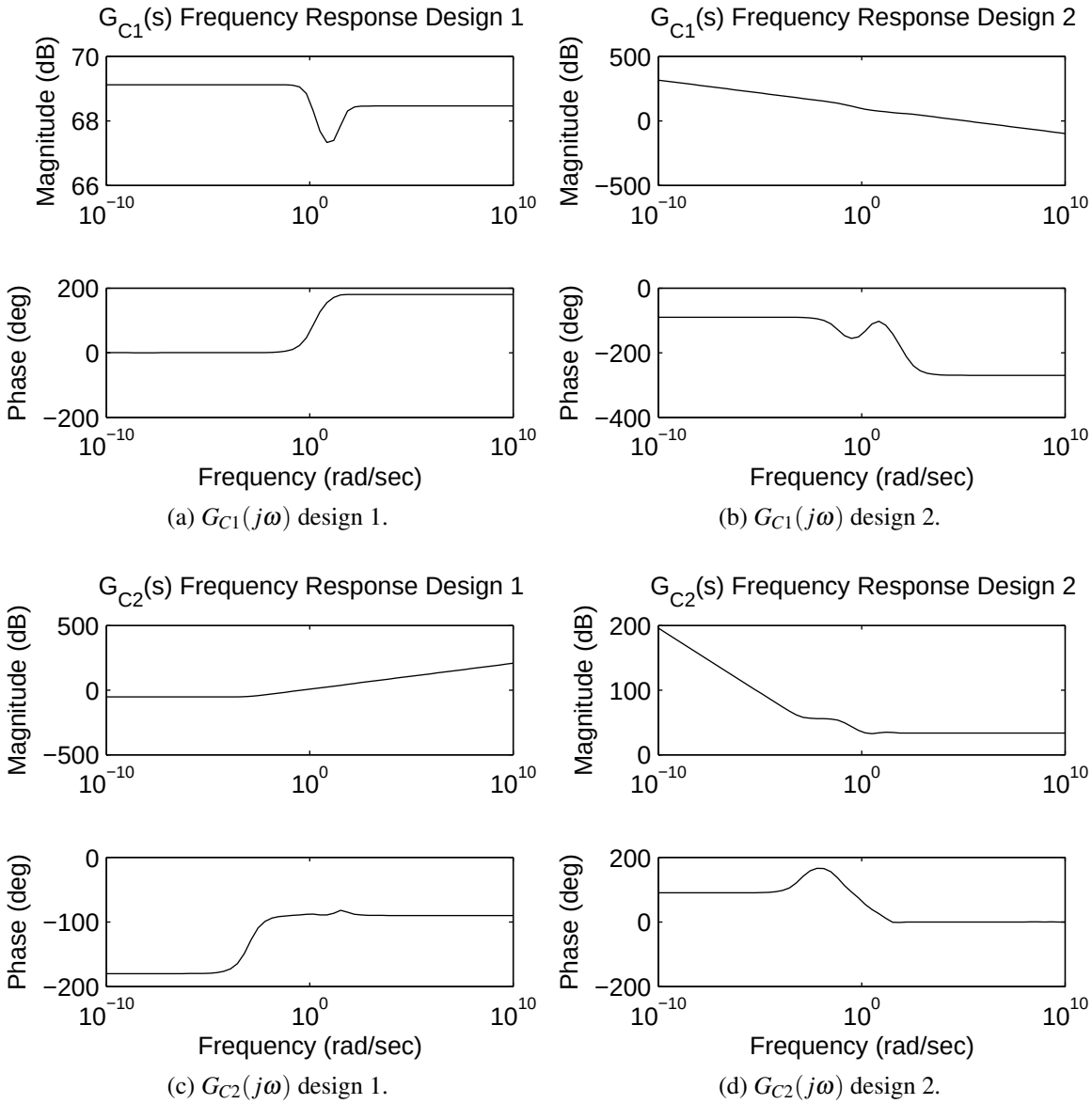
(b) $G_{C1}(j\omega)$ design 2.



(c) $G_{C2}(j\omega)$ design 1.

(d) $G_{C2}(j\omega)$ design 2.

Figure C.5: Design Scenario 3: Frequency response designs for $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$.



(a) $G_{C1}(j\omega)$ design 1.

(b) $G_{C1}(j\omega)$ design 2.

(c) $G_{C2}(j\omega)$ design 1.

(d) $G_{C2}(j\omega)$ design 2.

Figure C.6: Design Scenario 4: Frequency response designs for $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$.

The frequency responses for the two controllers $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$ are given in Figure C.6. Design 2 for $G_{C1}(j\omega)$ has an integrating response, but it does not have the correct sign and the change to a second-order non-minimum phase closed-loop tracking specification only influences the high frequency phase response. Design 2 of $G_{C2}(j\omega)$ has an integrating response, but does not have the correct sign. The method, therefore, does not provide a feasible design for the model in equation (C.36) with the design specifications given in equation (C.37) and (C.38).

C.4.5 Design scenario 5

This design defines a second-order non-minimum phase closed-loop tracking specification for the PSE-CFF loop and a first-order closed-loop tracking specification for the SLEV-SFW

loop that is faster than the PSE-CFF loop. The interacting model without time-delay and the integrators approximated by fast first-order responses is used in the design. The time-delay can be brought back through a Smith predictor structure. The model used for this design scenario is given by

$$G_{D5}(s) = \begin{bmatrix} -0.00035\frac{(1-0.63s)}{(1+0.54s)} & \frac{0.00055}{(1+0.24s)} \\ \frac{-0.29}{(s+0.001)} & \frac{0.42}{(s+0.001)} \end{bmatrix} \tag{C.39}$$

The closed-loop tracking specifications are given by

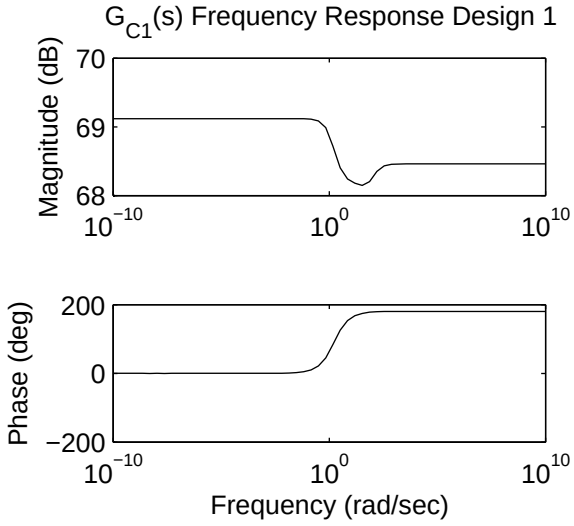
$$\frac{Y_1(s)}{R_1(s)} = \frac{(1-0.021s)}{(1+0.021s)(1+0.021s)} \tag{C.40}$$

$$\frac{Y_2(s)}{R_2(s)} = \frac{1}{(1+0.01s)} \tag{C.41}$$

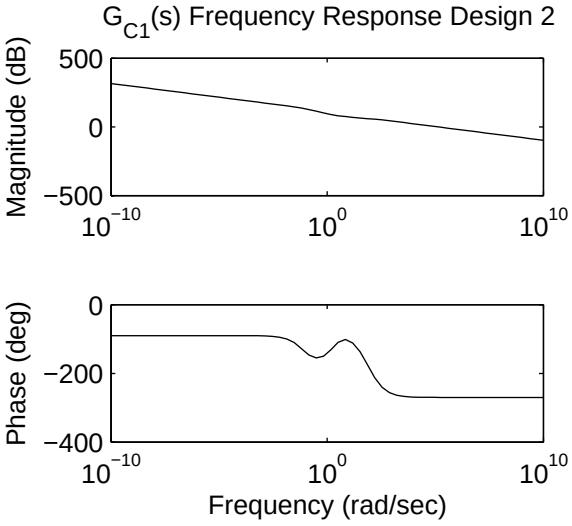
The frequency responses for the two controllers $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$ are given in Figure C.7. This specification change made no material change compared to design scenario 4, shown in Section C.4.4. The method, therefore, does not provide a feasible design for the model in equation (C.39) with the design specifications given in equation (C.40) and (C.41).

C.5 CONCLUSION

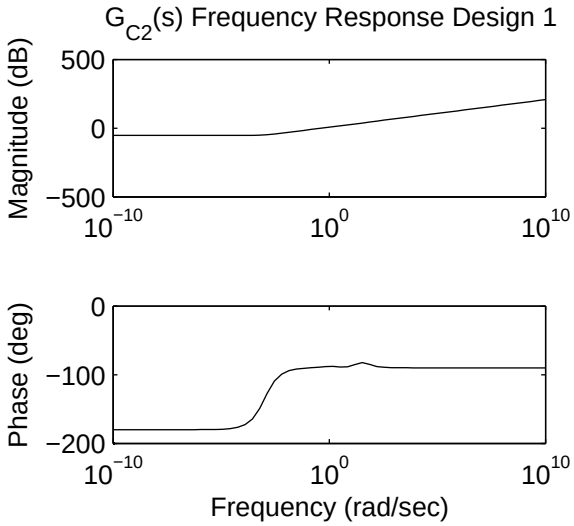
This addendum aimed to tune the decentralised PID controllers while explicitly taking interactions into consideration for the PSE-CFF and SLEV-SFW loops. A number of design scenarios were investigated in Section C.4 as well as other scenarios that were not documented here that investigated second-order closed-loop tracking specifications with and without non-minimum phase zeros for both controllers as well as slowing down the tracking specifications. Controller designs were found for both controllers with integrating responses when the integrators were approximated by fast first-order responses, but none of the model or specification changes produced controllers with integrating responses and the correct signs. This method, therefore, does not produce feasible controllers for the required closed-loop tracking specifications and valid interacting models.



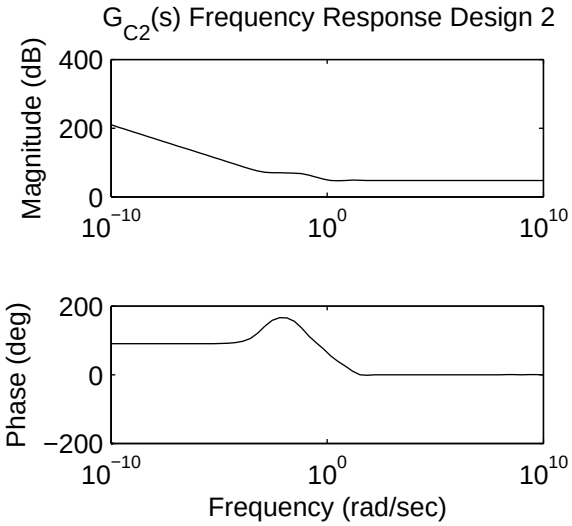
(a) $G_{C1}(j\omega)$ design 1.



(b) $G_{C1}(j\omega)$ design 2.



(c) $G_{C2}(j\omega)$ design 1.



(d) $G_{C2}(j\omega)$ design 2.

Figure C.7: Design Scenario 5: Frequency response designs for $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$.

ADDENDUM D

SIMULATION SUMMARY

A summary of the simulation results are given in Table B.1 that details the tracking performance of the controller with regard to the particle size (PSE), mill load level (LOAD) and throughput. The simulation scenario is outlined in terms of the controlled variable weighting and setpoint values, as well as the step disturbances. The relevant changes in each simulation scenario are highlighted in bold. The dashes in the table represent values that are either zero or not applicable. The performance metrics are described in Section 5.2.

The headings of Table B.1 are defined as

PSE	Particle Size Estimate. [$\% < 75\mu$m]
LOAD	The volumetric filling of the mill. [%]
Throughput	The amount of solids discharged at the cyclone overflow. [$^{\text{tons}}/_{\text{hour}}$]
SLEV	Sump level. [m^3]
Power	The electrical power draw of the mill motor. [kW]
Rheology	An indication of the fluidity of the slurry inside the mill. [fraction]
U	The manipulated variables.
Disturbances	Describes the step disturbances in ore hardness, fraction of rock in the feed ore and SFW.
Time	Describes the average and maximum iteration time of the simulations [seconds].

The subheadings of Table B.1 are defined as

Δ	The sum of the squares of the error from the setpoint.
S	The setpoint of the variable.
W	The weight of the variable in the objective function.
A	The average value of the variable over the simulation duration.
RH	The increase in the hardness of the feed ore. [%]
AR	The increase in the fraction of rock in the feed ore. [%]
SFW	The increase in SFW. [$m^3/hour$]
T	The time when the disturbance is introduced.
M	The maximum value of the variable over the simulation duration.

The controllers are identified next to the figure numbers in Table B.1 by

R	Robust Nonlinear Model Predictive Controller.
N	Nonlinear Model Predictive Controller.
P	Proportional-Integral-Differential Controller.

The simulations show that RNMPC and NMPC are capable of tighter control of PSE, especially when constraints are active, because the multivariable controllers can leverage the multivariable nature of the milling circuit to increase the control envelope. The RNMPC and NMPC controllers are, however, not capable of improving throughput while maintaining PSE at the desired setpoint. The PI controllers performed well, because they were tuned very aggressively. In certain milling circuits there are large time delays that result in less aggressive tuning of the PI controllers and degraded performance. MPC controllers were found in practice to perform well over longer periods compared to PI controllers (Chen *et al.*, 2007*b*, Ramasamy *et al.*, 2005).

Table D.1: All simulations summary.

			PSE			LOAD			Throughput			SLEV		Power		Rheology		U	Disturbances						Time	
Simulation			Δ	S	W	Δ	S	W	A	S	W	S	W	S	W	S	W	W	RH	T	AR	T	S	T	A	M
Figures 5.4 & 5.5	a	R	0.05	80	100	3.20E-05	45	100	106.72	99.9	**0**	5	0	2000	0	0.51	0	**0.01**	—	—	—	—	—	—	12.8	132
Figures B.1 & B.2	a	N	0.05	80	100	1.36E-03	45	100	106.31	99.9	0	5	0	2000	0	0.51	0	0.01	—	—	—	—	—	—	0.1	2.25
Figures 5.4 & 5.5	b	R	0.05	80	100	3.60E-05	45	100	111	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	—	—	—	—	22.6	186
Figures B.1 & B.2	b	N	0.07	80	100	6.34E-04	45	100	110.27	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	—	—	—	—	0.1	1.6
Figures 5.4 & 5.5	c	P	0.15	80	—	5.35E-04	45	—	106.28	—	—	5	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Figures 5.6 & 5.7	a	R	0.07	80	100	4.70E-05	45	100	88.71	99.9	0	5	1	2000	0	0.51	0	0.01	**50**	**180**	—	—	—	—	21.7	318
Figures 5.6 & 5.7	b	N	0.08	80	100	6.36E-04	45	100	88.97	99.9	0	5	1	2000	0	0.51	0	0.01	**50**	**180**	—	—	—	—	0.1	1.52
Figures 5.6 & 5.7	c	P	0.97	80	—	4.58E-04	45	—	95.67	—	—	5	—	—	—	—	—	—	**50**	**180**	—	—	—	—	—	—
Figures 5.8 & 5.9	a	R	0.05	80	100	1.14E-04	45	100	103.5	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**50**	**180**	—	—	24.5	204
Figures 5.8 & 5.9	b	N	0.09	80	100	8.72E-04	45	100	102.86	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**50**	**180**	—	—	0.1	1.56
Figures 5.8 & 5.9	c	P	0.18	80	—	5.82E-04	45	—	105.53	—	—	5	—	—	—	—	—	—	—	—	**50**	**180**	—	—	—	—
Figures 5.10 & 5.11	a	R	0.06	80	100	2.28E-04	45	100	103.26	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**50**	**10**	—	—	27.1	291
Figures 5.10 & 5.11	b	N	0.23	80	100	7.08E-03	45	100	101.77	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**50**	**10**	—	—	0.1	1.47
Figures 5.10 & 5.11	c	P	0.41	80	—	6.32E-04	45	—	106.96	—	—	5	—	—	—	—	—	—	—	—	**50**	**10**	—	—	—	—
Figures 5.12 & 5.13	a	R	0.05	80	100	2.70E-05	45	100	110.89	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**0**	—	**50**	**180**	26.7	340
Figures 5.12 & 5.13	b	N	0.08	80	100	4.71E-04	45	100	110.28	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**0**	—	**50**	**180**	0.1	1.59
Figures 5.12 & 5.13	c	P	0.16	80	—	7.71E-04	45	—	110.30	—	—	5	—	—	—	—	—	—	—	—	**0**	—	**50**	**180**	—	—
Figures B.9 & B.10	a	R	0.10	80	100	1.59E-03	45	100	73.04	99.9	0	5	1	2000	50	0.51	50	0.01	50	10	50	100	50	30-80	36.2	665
Figures B.11 & B.12	a	N	0.10	80	100	1.42E-03	45	100	73.03	99.9	0	5	1	2000	50	0.51	50	0.01	50	10	50	100	50	30-80	0.2	2.07
Figures B.9 & B.10	b	R	0.27	80	100	9.50E-04	45	100	71.43	99.9	0	5	1	2000	50	0.51	50	0.1	50	10	50	100	50	30-80	61.3	1168
Figures B.11 & B.12	b	N	0.42	80	100	3.84E-02	45	100	71.93	99.9	0	5	1	2000	50	0.51	50	0.1	50	10	50	100	50	30-80	0.2	2.13
Figures B.9 & B.10	c	R	1.63	80	100	3.49E-01	45	100	74.81	99.9	0	5	1	2000	50	0.51	50	1	50	10	50	100	50	30-80	34.4	204
Figures B.11 & B.12	c	N	1.81	80	100	5.75E-01	45	100	74.7	99.9	0	5	1	2000	50	0.51	50	1	50	10	50	100	50	30-80	0.2	2.99
Figures 5.14 & 5.15	a	R	0.23	80	100	3.10E-03	45	100	72.22	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	34.9	1014
Figures 5.14 & 5.15	b	N	0.30	80	100	5.45E-03	45	100	72.54	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.2	1.77
Figures 5.14 & 5.15	c	P	0.77	80	—	1.52E-04	45	—	74.59	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
Figures B.13 & B.14	b	R	0.29	80	100	3.50E-03	45	100	73.82	99.9	1	5	1	2000	0	0.51	0	0.1	50	10	50	100	50	30-80	50.0	1295
Figures B.15 & B.16	b	N	0.28	80	100	9.11E-02	45	100	73.79	99.9	1	5	1	2000	0	0.51	0	0.1	50	10	50	100	50	30-80	0.2	2.41
Figures B.13 & B.14	c	R	1.53	80	100	3.97E-01	45	100	76.07	99.9	1	5	1	2000	0	0.51	0	1	50	10	50	100	50	30-80	25.8	348
Figures B.15 & B.16	c	N	1.31	80	100	6.07E-01	45	100	75.1	99.9	1	5	1	2000	0	0.51	0	1	50	10	50	100	50	30-80	0.2	3.75

Table D.3: All simulations summary (continued).

			PSE			LOAD			Throughput			SLEV		Power		Rheology		U	Disturbances					Time	
Simulation			Δ	S	W	Δ	S	W	A	S	W	S	W	S	W	S	W	W	RH	T	AR	T	S	T	A	M
Figures 5.16 & 5.17	b	R	0.12	75	100	3.35E-03	45	100	74.51	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	35.9	1132
Figures B.3 & B.4	b	N	0.19	75	100	8.36E-03	45	100	74.33	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	2.23
Figures 5.18 & 5.19	b	P	0.24	75	—	1.62E-04	45	—	77.98	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
Figures 5.16 & 5.17	c	R	0.12	70	100	3.88E-03	45	100	81.35	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	33.4	1136
Figures B.3 & B.4	c	N	0.17	70	100	6.73E-03	45	100	81.87	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	2.08
Figures 5.18 & 5.19	c	P	1.50	70	—	1.40E-04	45	—	81.20	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
Figures 5.20 & 5.21	a	R	0.15	80	100	2.65E-03	45	100	74.04	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	33.8	688
Figures 5.20 & 5.21	b	N	0.18	80	100	5.22E-03	45	100	74.22	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	9.6	58
Figures 5.20 & 5.21	c	P	0.39	80	—	1.64E-04	45	—	76.43	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
Figures 5.22 & 5.23	a	R	0.15	80	100	1.80E-03	45	100	76.73	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	39.3	1127
Figures 5.22 & 5.23	b	N	0.14	80	100	4.74E-03	45	100	78.11	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	10.3	70
Figures 5.22 & 5.23	c	P	0.92	80	—	1.74E-04	45	—	78.52	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
Figures 5.24 & 5.25	a	R	2.79	80	100	5.87E-02	45	100	71.14	99.9	5	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	21.0	813
Figures B.5 & B.6	a	N	3.31	80	100	1.06E-01	45	100	71.45	99.9	5	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	2.04
Figures 5.24 & 5.25	b	R	7.09	80	100	1.70E-01	45	100	70.76	99.9	10	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	15.2	619
Figures B.5 & B.6	b	N	7.89	80	100	3.43E-01	45	100	69.25	99.9	10	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	1.77
Figures 5.26 & 5.27	b	R	5.52	80	100	8.20E-02	45	100	74.85	99.9	10	5	1	2000	0	0.51	0	0.1	50	10	50	100	50	30-80	49.2	1215
Figures B.7 & B.8	b	N	4.71	80	100	5.86E-02	45	100	75.56	99.9	10	5	1	2000	0	0.51	0	0.1	50	10	50	100	50	30-80	0.2	2.3
Figures 5.26 & 5.27	c	R	4.11	80	100	1.21E-01	45	100	75.6	99.9	10	5	1	2000	0	0.51	0	1	50	10	50	100	50	30-80	33.2	987
Figures B.7 & B.8	c	N	4.79	80	100	7.87E-01	45	100	75	99.9	10	5	1	2000	0	0.51	0	1	50	10	50	100	50	30-80	0.2	6.43
Figures 5.24 & 5.25	c	R	10.01	80	100	5.15E-01	45	100	68.09	99.9	20	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	13.3	682
Figures B.5 & B.6	c	N	13.01	80	100	5.00E-01	45	100	70.83	99.9	20	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	1.38
Figures B.17 & B.18	b	R	9.19	80	100	2.69E-02	45	100	76.83	99.9	20	5	1	2000	0	0.51	0	1	50	10	50	100	50	30-80	44.7	1171
Figures B.19 & B.20	b	N	7.52	80	100	1.32E+00	45	100	75.73	99.9	20	5	1	2000	0	0.51	0	1	50	10	50	100	50	30-80	0.2	3.77