

UNIVERSITY OF NATAL

**MODEL PREDICTIVE CONTROL
OF HYBRID SYSTEMS**

JASMEER RAMLAL

MODEL PREDICTIVE CONTROL OF HYBRID SYSTEMS

Jasmeer Ramlal

BSc Eng

Submitted in fulfillment of the academic requirements for the degree of

Master of Science in Engineering

in the

School of Chemical Engineering,

University of Natal, Durban

Durban

October 2002

Abstract

Hybrid systems combine the continuous behavior evolution specified by differential equations with discontinuous changes specified by discrete event logic. Usually these systems in the processing industry can be identified as having to depend on discrete decisions regarding their operation. In process control there therefore is a challenge to automate these decisions.

A model predictive control (MPC) strategy was proposed and verified for the control of hybrid systems. More specifically, the dynamic matrix control (DMC) framework commonly used in industry for the control of continuous variables was modified to deal with mixed integer variables, which are necessary for the modelling and control of hybrid systems.

The algorithm was designed and commissioned in a closed control loop comprising a SCADA system and an optimiser (GAMS). GAMS (General Algebraic Modelling System) is an optimisation package that is able to solve for integer/continuous variables given a model of the system and an appropriate objective function. Online and offline closed loop tests were undertaken on a benchmark interacting tank system and a heating/cooling circuit. The algorithm was also applied to an industrial problem requiring the optimal sequencing of coal locks in real time. To complete the research concerning controller design for hybrid behavior, an investigation was undertaken regarding systems that have different modes of operation due to physicochemical (inherent) discontinuities e.g. a tank with discontinuous cross sectional area, fitted with an overflow. The findings from the online tests and offline simulations reveal that the proposed algorithm, with some system specific modification, was able to control each of the four hybrid systems under investigation.

Based on which hybrid system was being controlled, by modifying the DMC algorithm to include integer variables, the mixed integer predictive controller (MIPC) was employed to initiate selections, switchings and determine sequences. Control of the interacting tank system was focused on an optimum selection in terms of operating positions for process inputs. The algorithm was shown to retain the usual features of DMC (i.e. tuning and dealing with multivariable interaction). For a system with multiple modes of operation i.e. the heating/cooling circuit, the algorithm was able to switch the mode of operation in order to meet operating objectives. The MPC strategy was used to good effect when getting the algorithm to sequence the operation of several coal locks. In this instance, the controller maintained system variables within certain operating constraints. Furthermore, soft constraints were proposed and used to promote operation close to operating constraints without the danger of computational failure due to constraint violations. For systems with inherent discontinuities, a MPC strategy was proposed that predicted trajectories which crossed discontinuities. Convolution models were found to be inappropriate in this instance and state space equations describing the dynamics of the system were used instead.

I dedicate this work to my source of inspiration, light and love Sri Sathya Sai Baba and to my mother, Rooplaka who has supported me in all my pursuits.

Message

“Every Being is a messenger to Mankind, entrusted with the task of spreading knowledge of the joy that is being missed.”

(Sri Sathya Sai Baba)

Preface

“Man has springs of joy and peace in his heart, even as a child. Cultivate them, give them fullest freedom to gush forth and fertilise all fields of activity. That is the real purpose of education.”

(Sri Sathya Sai Baba)

The above statement puts into perspective the true meaning of education. When one is educated in whatever field or sector, it is ultimately for one's own good. However education becomes sanctified when it is used for the benefit of others. This is done when learning opens up one's mind and heart to the world. This creates humility and breeds self confidence, both attributes necessary for any person seeking to make the world a better place.

The time spent during this masters degree has given me the opportunity to realise the true meaning of education. A masters degree allows one more time to reflect on one's learning experience and hence use it as a catalyst for other sectors in life. This has been an important phase in my development as an individual. I have learnt to understand the process of learning and also how to teach myself new things.

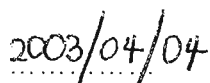
It helps when one has passion for one's field and in my case it is process control. This is such a dynamic area in chemical engineering and will continue to be so because of our continued pursuits for better automation and improved mathematical algorithms. Process control requires one to have a holistic view of the process and at the same time, understand its intricate details. This is very important, because one cannot control what one does not understand. I am grateful for the control engineering field because of the inspiration that I have drawn from studying and applying its principles.

Learning is an ongoing endeavour to discover who we really are. The destination is known, but the journey is not and needs to be undertaken. I conclude this preface with the words of Mahatma Gandhi: “Live today as if you were to die tomorrow, but learn today as if you were to live forever.”

I, Jasmeer Ramlal, declare that unless indicated, this dissertation is my own work and that it has not been submitted in whole or in part for a degree at another University or Institution.



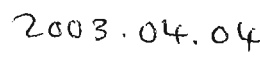
J. Ramlal



Date



Professor M. Mulholland



Date

Acknowledgements

I would like to express my gratitude to the following people/organisations for their contribution towards this research.

My supervisor Professor Michael Mulholland. Thank you for your guidance and calming presence. Your vision and approach to process control has certainly inspired me in this area.

Professor M. Starzak for assisting in the process modelling in chapter 8.

Mr. Malan Rossouw at the Sasolburg gasification plant. Thank you for your explanations regarding the gasification process, even if sometimes some of the queries were trivial.

Dr. Francois Gouws, my mentor at Sasolburg. Thank you for arranging the plant visit in January and providing the plant data for the gasifiers.

The University of Natal Research Fund for funding part of this research project.

Sasol, with whom I have been a bursar for the past four years. You have an excellent undergraduate and postgraduate program. Keep up the good work! I trust that I shall make a positive contribution to the company in the years to come.

My family: immediate; extended; and Sai, who have to put up with me. It is not by accident that we have been placed together. Your support and the interest that you have shown in my work is much appreciated.

Finally to my many friends (too many to be specific). Thank you for always being yourselves and for giving me the opportunity to know you all. Especially my fellow postgraduate students, from whom I have learnt that we should seek a balance in life, and also that life is something that we should not take too seriously. You have given meaning to the saying : "Life is a Game, Play it."

Table of Contents

	Page
<i>Abstract</i>	<i>iii</i>
<i>Preface</i>	<i>v</i>
<i>Acknowledgements</i>	<i>vi</i>
<i>Table of Contents</i>	<i>vii</i>
<i>List of Figures and Tables</i>	<i>xii</i>
<i>List of Symbols</i>	<i>xvi</i>
Chapter 1: Introduction	
1.1 Research Objective	1-2
1.2 Thesis Overview	1-3
Chapter 2: Hybrid Systems	
2.1 Illustrative example	2-1
2.2 Hybrid system control	2-2
2.2.1 Continuous feature	2-3
2.2.1 Discontinuous feature	2-3
a. Different modes of physical behaviour	2-3
b. Discontinuous outputs	2-3
c. Discontinuous inputs	2-3
d. Discontinuous control	2-4

2.3 Literature review	2-5
2.4 Hybrid systems in Industry	2-8
2.4.1 Pressure swing adsorption	2-8
2.4.2 Selection processes from Inventories	2-10
2.4.3 Heating/Cooling strategies for batch reactors	2-11
2.4.4 Tank with a weir	2-14

Chapter 3: Model Predictive control

3.1 Overview	3-1
3.2 Dynamic Matrix Control (DMC)	3-4
3.2.1 Introduction	3-4
3.2.2 DMC algorithm	3-5
3.2.3 DMC and integrating systems	3-10
3.2.4 DMC and Hybrid Systems	3-11
3.3 Control loop	3-12

Chapter 4: Mixed Integer Dynamic Optimisation (MIDO)

4.1 Introduction	4-1
4.2 Linear Integer Programming	4-2
4.3 Computational Aspects	4-4
4.3.1 Mixed integer non linear programming (MINLP)	4-4
4.3.2 Software	4-7
4.3.3 GAMS	4-9

Chapter 5: Laboratory Case Study 1 – Interacting Tanks

5.1 Process flow diagram	5-1
5.2 Modelling and Control	5-3
5.3 Offline tests	5-6
5.3.1 Controller tuning	5-7
5.3.2 Contrasting controller action	5-11
5.3.3 Multiple selection for V_2	5-12
5.4 Online tests	5-13
5.5 Conclusion	5-17

Chapter 6: Laboratory Case Study 2 – Thermal Circuit

6.1 Process Analysis	6-1
6.1.1 Process flow diagram	6-1
6.1.2 Modes	6-2
6.2 DMC for systems with mode switches	6-4
6.2.1 Switch responses	6-4
6.2.2 Adaptation for thermal circuit mode switching	6-6
6.3 Closed loop tests	6-8
6.3.1 Simulated symmetric responses	6-8
6.3.2 Measured online control	6-11
6.3.3 Simulated offline control	6-13
6.3.3.1 Number of optimised moves	6-13
6.3.3.2 Constrained control	6-15
6.4 Conclusion	6-17

Chapter 7: Industrial Application - Coal lock sequencing

7.1 Process overview	7-1
7.2 Current control strategy	7-4
7.3 Sequencing using MPC	7-8
7.4 Offline tests	7-10
7.4.1 Sequencing based on setpoint	7-11
7.4.2 Sequencing based on tuning	7-12
7.4.3 Introduction of soft constraint	7-13
7.4.4 Sequencing of six gasifiers	7-18
7.5 Conclusion	7-20

Chapter 8: Physicochemical Discontinuities

8.1 Modelling	8-1
8.2 Case Study – Non symmetrical tank with overflow	8-5
8.3 Simulations	8-11
8.3.1 Openloop tests	8-11
8.3.2 Closed loop tests	8-14
8.3.2.1 Forward Control	8-15

8.3.2.2 Multiple input multiple output	8-18
8.4 Conclusion	8-25

Chapter 9: Conclusions and Recommendations

9.1 Conclusions	9-1
9.2 Recommendations	9-2

References	R-1
-------------------	------------

Appendices

Appendix A: Control loop

A1. Extract from file CEXTObject for interfacing SCAD and GAMS	A-1
A2. Logic diagram showing spawning of control algorithm in GAMS	A-6

Appendix B: Interacting Tanks

B1. Step response data	B-1
B2. Operability region	B-2
B3. Equation block from GAMS for optimal continuous evaluation of V_1 and discrete selection of V_2	B-3

Appendix C: Thermal Circuit

C1. Non symmetrical switch step response data	C-1
C2. Symmetrical switch step response data	C-4
C3. Equation block from GAMS for optimal switching of modes	C-5

Appendix D: Coal lock sequencing

D.1 Step response data for 3 coal locks	D-1
D.2 Equation block from GAMS for optimal sequencing of coal locks using soft constraint	D-1

Appendix E: Physicochemical discontinuities

- | | |
|---|------------|
| E1. MATLAB code that initially spawns the control algorithm into
GAMS and then simulates the control loop with a single tank | E-1 |
| E2. Control algorithm executed in GAMS every time the controller
is called for optimal filling of two tanks | E-7 |

List of Figures and Tables

Figure	Page
2.1 Dynamics of an accelerating car	2-1
2.2 Interaction between finite state machines and continuous dynamical systems through analog-to-digital (A/D) and (D/A) interfaces	2-5
2.3 Pressure Swing Adsorption	2-8
2.4 Sequence of cycle steps in PSA	2-9
2.5 Blending system	2-10
2.6 Multifluid temperature control	2-11
2.7 Monofluid temperature control	2-12
2.8 Monofluid and multifluid temperature control	2-13
2.9 Production of a specified chemical product by crystallisation according to a specific temperature profile	2-14
2.10 Buffer tank with overflow	2-15
2.11 Discontinuous state behaviour in tank-weir system	2-16
3.1 Feedback control loop showing plant and model predictive controller	3-2
3.2 “Snap shot” of the moving horizon approach to MPC	3-3
3.3 Step response for SISO system	3-5
3.4 Partitioning step response for convolution model	3-6
3.5 Optimisation horizon for DMC	3-9
3.6 DMC in model predictive control configuration	3-13
3.7 Integration of software packages for closed loop testing of MIPC	3-14
3.8 Interfacing between GAMS and SCAD with file CEXTObject	3-15
4.1 Tree structure for branch and bound procedure	4-5
4.2 GBD and OA solution procedures	4-6
4.3 General structure of GAMS model	4-9
4.4 Reactor configuration selection	4-10
4.5 Variation in cost with feed to reactor2	4-12
5.1 Interacting Tanks	5-2
5.2 Offline simulated setpoint step test with 1 optimised move; $W_1=W_2=\Lambda_1=\Lambda_2=1$	5-8
5.3 Offline simulated setpoint step test with 2 optimised moves; $W_1=W_2=\Lambda_1=\Lambda_2=1$	5-8

5.4	Offline simulated setpoint step test with 2 optimised moves; $W_1=W_2=\Lambda_1=1; \Lambda_2=0.1$	5-9
5.5	Offline simulated setpoint step test with 2 optimised moves; $W_1=\Lambda_1=1; \Lambda_2=0.1; W_2=10$	5-10
5.6	Offline simulated oscillatory level Control with 2 optimised moves; $W_1=W_2=\Lambda_1=1; \Lambda_2=0.1$	5-11
5.7	Offline simulated setpoint step test with V_2 allowed intermediate positions while using 2 optimised moves; $W_1=W_2=\Lambda_1=1; \Lambda_2=0.1$	5-12
5.8	Online measured control with 2 step optimisation and V_2 allowed intermediate positions	5-15
5.9	Online measured level control with V_2 only open or shut with 3 step optimisation horizon	5-16
6.1	Thermal circuit	6-2
6.2	System with ternary modes	6-3
6.3	Conversion from primary inputs to mode transitions	6-3
6.4	Openloop switch response data	6-4
6.5	Symmetrical and asymmetrical state response to change in input	6-5
6.6	Switch responses used for convolution model	6-6
6.7	Mode selection for symmetrical responses with 1 optimised switch (control of model)	6-10
6.8	Mode selection for symmetrical responses with 2 optimised switches (control of model)	6-10
6.9	Online switching with the controller allowed one switch (control of plant)	6-12
6.10	Online switching with the controller allowed two switches (control of plant)	6-12
6.11	Simulated control switching with 1 optimised switch (control of asymmetric model)	6-14
6.12	Simulated control switching with 2 optimised switches (control of asymmetric model)	6-14
6.13	Constrained control switching with 2 optimised switches (control of asymmetric model)	6-15
6.14	Constrained control switching with 4 optimised switches (control of asymmetric model)	6-16
6.15	Constraint switching with soft constraint and 2 optimised moves	6-17
7.1	Single gasifier with coal / ash locks and schematic showing functioning of cones	7-3
7.2	Gas circuitry showing CLEG system	7-4
7.3	Sequence of events for a typical coal lock cycle	7-6

7.4	LE205 response for coal locks getting forced or setpoint starts	7-7
7.5	Short coming in convolution model to predict the actual state trajectory	7-10
7.6	Sequencing using MPC based on setpoint penalisation only	7-11
7.7	Coal lock sequencing using tuning	7-13
7.8	Violation of hard constraint due to plant model mismatch	7-14
7.9	Hard constraints on controlled variable x	7-15
7.10	Effect of soft constraint on sequencing	7-17
7.11	optimal sequencing of coal locks using soft constraint	7-17
7.12	Controller sequencing 6 coal locks	7-19
8.1	Model states for a single system model	8-2
8.2	Model states for different regimes	8-3
8.3	Approximated point of switching	8-4
8.4	Hypothetical tank with discontinuous cross sectional area and overflow	8-5
8.5	Flowchart for closed loop simulation in MATLAB for tank with overflow and discontinuous cross sectional area	8-10
8.6	Effect of non-uniform diameter when filling tank	8-12
8.7	Effect of overflow when filling tank	8-13
8.8	Combined effect of non-uniform diameter and overflow when filling	8-13
8.9	Combined effect of non-uniform diameter and overflow when emptying	8-14
8.10	Control sequence with 1 step optimisation horizon and single control move	8-16
8.11	Control sequence with 5 step optimisation horizon and single control move	8-17
8.12	Control sequence with 5 step horizon and 3 control moves	8-17
8.13	Two tank system	8-18
8.14	Two tank system filling with controller allowed 2 step optimisation horizon and single control move	8-21
8.15	Two tank system filling with controller allowed 5 step optimisation horizon and 2 control moves	8-22
B.1	Level response for step in V_1 (-30%)	B-1
B.2	Level response for step in V_2 (-30%)	B-1
B.3	Operability region in terms of levels	B-2
C.1	Temperature response for mode switch from heating to recycle	C-1
C.2	Temperature response for mode switch from recycle to heating	C-1
C.3	Temperature response for mode switch from heating to cooling	C-2
C.4	Temperature response for mode switch from cooling to heating	C-2

C.5	Temperature response for mode switch from cooling to recycle	C-3
C.6	Temperature response for mode switch from recycle to cooling	C-3

Tables

4.1	Conversion of logic into integer inequalities	4-3
5.1	Summary of controller parameters and performance for simulations	5-7
5.2	Summary of controller parameters and performance for online measurements	5-13
7.1	Simplified DMC representation for sequencing coal locks	7-8
8.1	Controller and system parameters used in simulations	8-11
B.1	Step data used for dynamic matrix extending over 10 step horizon	B-2
C.1	Non symmetric switch step data used for dynamic matrix extending over 15 step horizon	C-4
C.2	Symmetric switch step data used for dynamic matrix extending over 15 step horizon	C-4
D.1	Step response data for coal lock sequencing extending over 5 step horizon	D-1

List of Symbols

Acronyms

B&B – branch and bound
BC – bottom cone
CLEG – coal lock expansion gas system
DMC – dynamic matrix control
GAMS – general algebraic modelling system
GBD – generalised benders decomposition
MLD – mixed logical dynamical
MIDO – mixed integer dynamic optimisation
MIP – mixed integer program
MILP – mixed integer linear program
MINLP – mixed integer non linear program
MIPC – mixed integer predictive control
MIQP – mixed integer quadratic program
MPC – model predictive control
OA – outer approximation
PSA – pressure swing adsorption
QP – quadratic program
SCADA – sequential control and data acquisition
TC – top cone

Variables, Vectors and Matrices

Chapter 2 – Hybrid Systems

F_{out} – outflow from tank
 F_{in} – inflow into tank
 $h(t)$ – instantaneous level in tank
 h_{weir} – height of weir
 t – discrete time
 u – system input
 x – system output

Chapter 3 – Model Predictive Control

b_i – step response coefficient for offset between state at sampling instant i and steady state offset

Δb – difference between final two points in step response for integrating system
 \bar{B}_o – sub-matrix of dynamic matrix, used for predicting present state
 \bar{B}_{ol} – sub-matrix of dynamic matrix, used for predicting openloop trajectory
 \bar{B} – sub-matrix of dynamic matrix, used for predicting effect of future moves on the state
 \bar{e}_{cl} – vector of closed loop error
 m – system input
 $\Delta \bar{m}$ – vector of future input changes to system
 $\Delta \bar{m}_{past}$ – vector of past input changes to system
 J – value of objective function
 M – size of control horizon
 N – size of steady state horizon
 P – size of optimisation horizon
 t – discrete time
 \bar{W} – diagonal matrix for penalisation of state from setpoint
 x – system output
 x_i – system output state at sampling interval i
 \bar{x}_{cl} – vector of closed loop state response
 \bar{x}_{omeas} – vector of fed back states from plant/model at the present moment in time
 \bar{x}_{Pred} – vector of predicted states extending ahead in time
 \bar{x}_{oPred} – vector of duplicated predicted states at present moment in time
 \bar{x}_{ol} – vector of openloop state response
 $\bar{\Lambda}$ – diagonal matrix for move suppression

Chapter 4 Mixed Integer Dynamic Optimisation

m_1 – upper limit for feed to reactor1
 m_2 – upper limit for feed to reactor2
 p – subscript depicting point condition
 u – continuous controlled variables
 t_f – size of optimisation horizon
 x – continuous output state of the system
 x_o – combined feed to reactor1 and reactor2
 x_1 – feed to reactor1
 x_2 – feed to reactor2
 y – discrete time variant or invariant system input
 y_1 – binary variable for reactor1
 y_2 – binary variable for reactor2
 v – continuous time variant or invariant parameter

Chapter 5 – Interacting Tanks

$\overline{binvect}$ – vector of binary variables

$\overline{continput}$ – continuous sub vector of hybrid vector $\Delta\overline{m}$

$\overline{discinput}$ – discrete sub vector of hybrid vector $\Delta\overline{m}$

$\overline{I_1}$ – ordering matrix 1

$\overline{I_2}$ – ordering matrix 2

$\Delta\overline{m}$ – vector of future input changes to system

\overline{Sel} – tri-diagonal matrix containing possible integer selections for V_2

S_1 – tank 1

S_2 – tank 2

S_3 – tank 3

V_1 – input valve 1

V_2 – input valve 2

\overline{Y} – vector of continuous and discrete inputs (absolute values)

Chapter 6 – Thermal rig

\overline{A} – integrating matrix relating modes to mode switches

C – cooling mode

CV – control valve

H – heating mode

m – system input

$\Delta\overline{m}$ – vector of mode switches

Mo_i – vector of modes at time step i

R – recycle mode

$T14$ – thermocouple reading exit temperature to tank

$T30$ – thermocouple reading exit cooler temperature

$T31$ – thermocouple reading inlet temperature to tank

Chapter 7 – Coal lock sequencing

CL_i – coal lock i

T_i – temperature in coal lock i

$Depress1_17$ – sum of weights on gasifiers when cycle starts are initiated

TM_PV – setpoint indicator on cycle time

$LE205$ – inferred level in gasifier by measuring temperature in coal lock

$\Delta\overline{m}$ – vector of input changes to system

s_{hi} – state upper limit for soft constraint

s_{lo} – state lower limit for soft constraint

x_i – continuous output state of the system at time step i .

x_{hi} – output state upper limit for hard constraint
 x_{lo} – output state lower limit for hard constraint
 τ_{test} –estimated time for coal lock cycle

Chapter 8 – Physicochemical discontinuities

$A_{l,1}$ – cross-section area of tank1 below overflow [m^2]
 $A_{l,2}$ – cross-section area of tank1 above overflow [m^2]
 A_2 – cross-sectional area of tank2 [m^2]
 $d_{l,1}$ – diameter of tank1 below overflow [m]
 $d_{l,2}$ – diameter of tank1 above overflow [m]
 F_A – function A
 F_B – function B
 F_o –flow into tank1 [m^3/hr]
 F_l – flow from bottom of tank1 into tank2 [m^3/hr]
 F_2 – flow from tank1 over flow [m^3/hr]
 F_3 – flow from bottom of tank2 [m^3/hr]
 F_{max} – upper limit F_o [m^3/hr]
 h_{max} – maximum level in tank1 [m]
 h_{min} – minimum level in tank1 [m]
 h^o – steady state operating level from which Taylor expansion is done [m]
 $h_{l,1}$ – level in tank1 below overflow [m]
 $h_{l,2}$ – level in tank1 above overflow [m]
 h_l – level in tank1 [m]
 h_2 – level in tank2 [m]
 h_v – height of overflow in tank1 [m]
 k_l – line constant for flow out of tank1 [$m^3 \cdot hr^{-1}/m^{0.5}$]
 k_2 – line constant for flow out of tank2 [$m^3 \cdot hr^{-1}/m^{0.5}$]
 k_v – line constant for over flow line [$m^3 \cdot hr^{-1}/m^{0.5}$]
 t – discrete time
 Δt – integration time step
 u – controlled variables
 V – binary valve acting on F_l
 x –output state of the system
 $x_A^* / x_B^* / x_C^*$ – points of discontinuity for state x
 α – binary variable indicating operation in regime1
 β – binary variable indicating operation in regime2

CHAPTER 1

Introduction

Process control involves the operation of some level of intelligence in parallel with a process. This intelligent source directs the process to meet given objectives and at the same time stabilises its operation during disturbances.

In process control this intelligence is formally known as a controller. A process controller is typically a mathematical algorithm (model), designed to invert the process and thereby make logical, automatic decisions regarding its manipulation. Depending on the nature of the process variables, there exist two kinds of dynamical systems (mathematical algorithms representing a process) in simulation and control. These are continuous time systems and hybrid systems. Basic phenomena in processing systems are inherently time continuous. The dynamical properties can often be determined using principles of conservation of mass and energy together with equations describing process components. However, in some instances binary/integer variables are required to model a process. In this event where there is a combination of continuous and binary/integer variables, the system is said to be hybrid in nature.

Since the phenomena to be modeled are all concerned with fundamentally continuous flows of material and energy, the question arises as to when the use of binary/integer variables, in process modelling and control, will become necessary and thus warrant a shift to a hybrid modelling paradigm. In answering this question, there are a number of reasons for including discontinuous mechanisms into continuous process models:

- It may be of interest to explicitly model inherent process discontinuities, at which continuous behavior is drastically changed.
- Actuators and sensors (final and initial control elements) are often fundamentally discontinuous.
- Discrete events can be used to model various types of mode switching used in the control of continuous processes.

Apart from the processing industry, hybrid system behavior also arises in other application areas e.g. automotive and aeronautic industries, and the computer science community. Over the last decade, there has been great interest in studying hybrid systems, firstly from a theoretical point of view and secondly because of their wide variety of practical applications. Despite much research, there as yet does not exist a unified approach to deal with hybrid systems. Instead there are several theoretical frameworks that are motivated by specific objectives and tasks (section 2.2).

Control design for hybrid processes is challenging and complex due to the combination of continuous and discrete variables. Several approaches regarding the control of hybrid processes have been studied e.g. logic-based switching, supervisory control, game theoretic control, predictive control, switch control and hierarchical control. Despite research in these areas, the control of hybrid processes in the chemical industry has mainly been based on heuristic rules, designed from a “common sense” approach that accounts for specific plant scenarios (Torrissi et al., 2001). These rules are then used to compile “if-then-else” logic, which is then used as a basis for a control algorithm. Using this relatively more unstructured approach, makes the analysis of the overall controlled plant for all possible scenarios a hard task. This is typically solved in industry by performing a large number of runs with the controller.

In this research project a model predictive control (MPC) approach, is proposed and validated, for processes exhibiting hybrid characteristics. The aim is to provide a more “holistic” approach to optimally deal with constraints and multivariable interactions. The control strategy is aimed at providing stabilisation to an equilibrium state or tracking of a desired reference trajectory, via feedback control. MPC has already been adopted in industry over the past twenty years to solve control problems for continuous linear systems. The control strategy is based on a receding horizon philosophy, where a sequence of future control actions, is determined according to the future predicted output of the system. The first new control action is applied to the plant until new measurements become available. At the same time, based on the new measurements, another sequence is established, which replaces the previous one. Each sequence is determined by means of an optimisation procedure that takes into account two objectives: firstly optimise tracking performance and secondly protect the system from possible constraint violations. MPC can thus be viewed as an online procedure that systematically improves a systems dynamic performance, while keeping it within defined regions of operation.

Along with the effort and expense required to build a good process model for a MPC controller, the other limitation of MPC is its online computational complexity. It is for these reasons that this area of hybrid system control has not been extensively implemented. The continuous on-line optimisation of hybrid processes is a problem in mixed integer dynamic optimisation (MIDO). Numerical solutions of MIDO problems rely on mathematical techniques developed to solve mixed integer non-linear programming (MINLP) problems e.g. branch and bound techniques (B&B) and outer approximation (OA) methods. Over the past decade, there have been significant improvements on the methods for solving MINLP problems. Coupling these advancements to the advances in computing technology, the real time control of hybrid processes using MPC is now possible.

1.1 Research Objective

A MPC scheme is proposed which is able to control hybrid systems on desired reference trajectories, while fulfilling operating constraints. This project is aimed at evolving and using a standard formulation of MPC i.e. dynamic matrix control (DMC), to control and optimise the operation of

laboratory scale and industrial processes that exhibit hybrid system characteristics. When a convolution model, as required by DMC, is not viable, then state space equations describing the dynamics of the process are used.

The control algorithm is designed to initiate or anticipate (as is the case in physicochemical discontinuities i.e. discontinuities due to inherent fundamental physical behaviour): selection; switching; and sequencing procedures, all of which introduce hybrid behaviour into the processes considered. In extending the on-line model predictive control strategy to include integer variables, it will be possible to address a number of practical engineering problems that were previously not considered as candidates for MPC.

1.2 Thesis Overview

This thesis consists of nine chapters, including the introduction. Chapters 2, 3 and 4 focus on the theory concerning hybrid system predictive control. The next four chapters show the development and design of the mixed integer controller based on specific lab scale and industrial processes. Chapter 9 draws conclusions and makes recommendations.

Chapter 2 focuses on hybrid systems. It presents an illustrative example of a hybrid system and then goes on to identify typical features in the processing industry that warrant mixed integer modelling and control. Furthermore, it presents a literature review concerning hybrid system modelling and control, together with examples of industrial hybrid systems. Chapter 3 deals with model predictive control, more specifically dynamic matrix control, and how it can be modified to accommodate integer variables. It also presents the closed control loop structure used to validate the controller online, with the plant or offline against a model. Chapter 4, on mixed integer dynamic optimisation, is concerned with the computational aspects of the control algorithm. It also focuses on mixed integer programming (MIP) where process logic is converted to mathematical representations using equality and inequality constraints. This chapter also introduces GAMS, the optimisation software tool used to design the controller.

Chapter 5 presents the first of two laboratory scale case studies. These studies are aimed at showing the selection capability of the controller on an interacting tank system. Chapter 6 concerns a further study which deals with the switching capability of the controller on a thermal circuit with 3 modes of operation. An industrial application is considered in chapter 7, involving the sequencing of a series of gasifiers. Chapter 8 is focused on processing systems with physicochemical (inherent) discontinuities. The operation of a hypothetical tank structure is simulated and controlled. The thesis concludes with chapter 9. A summary of the findings from the closed loop tests with the controller is presented, and recommendations for future research are made.

CHAPTER 2

Hybrid Systems

This chapter deals with hybrid systems. It commences with an illustrative example of a hybrid system. Section 2.2 presents features that cause systems in the processing industry to have a hybrid nature and also how this relates to process control. Thereafter a literature review provides an overview of the previous work done regarding the modelling and control of hybrid systems, in terms of publications, research groups and conferences. The chapter concludes by presenting typical examples of industrial hybrid systems.

2.1 Illustrative Example

Presented below is the acceleration profile for the new Mercedes-Benz AMG SLK32 taken from Car Magazine April 2002. The plots show the speed, engine speed (revs) and gear selection for a thirty-second time interval as the car accelerates from standstill to a terminal speed of 190 km per hour.

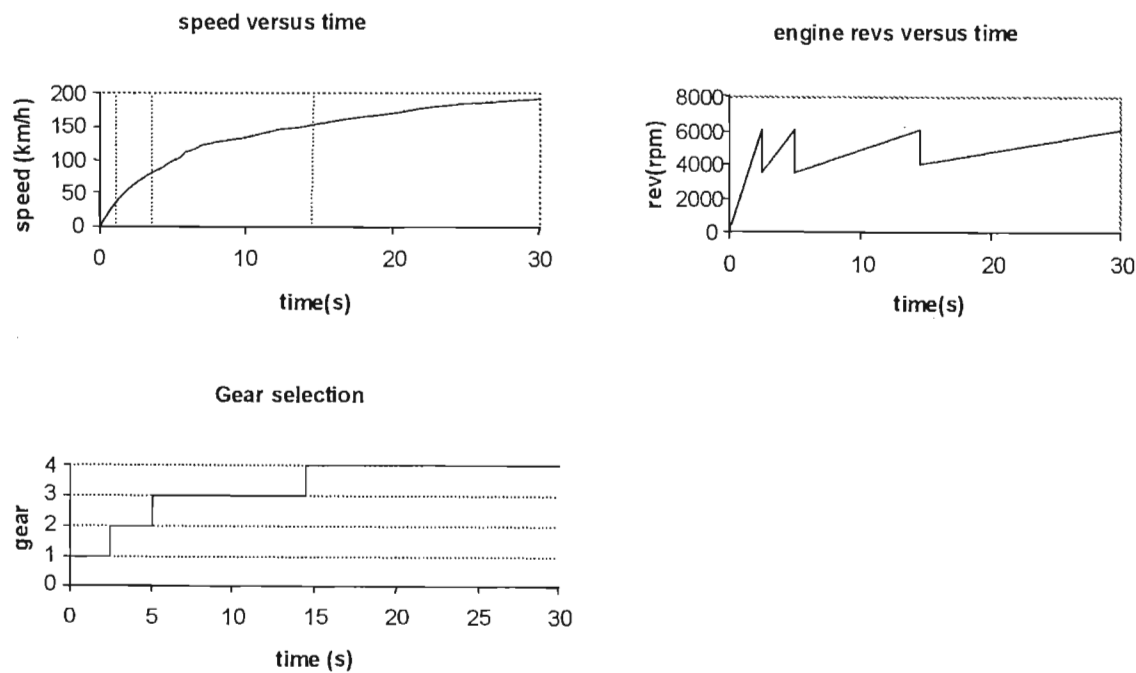


Figure 2.1: Dynamics of an accelerating car

A system is said to be hybrid if it has continuous components interacting with logical or discrete components. The different dynamic responses (car speed and revs) of the car accelerating through the gears is an illustration of a typical hybrid behaviour. The gear selection is either done manually, by the driver, or through an automatic transmission (as in this case), either way it is a discrete logical decision. Also the gears can only assume positions from an integer set $\{1,2,3,4\}$, determined by the number of gears that the car has. The speed of the car is a continuous variable, meaning that the car

can assume any speed from 0 km per hour to its terminal speed of 190km per hour. The engine speed exhibits discontinuous dynamic behaviour because it changes every time a new gear is selected. The interaction between the continuous and discrete dynamics is evident, as the car speed is determined by the engine speed differently for each gear. This interaction between the continuous and discrete/discontinuous dynamics is typical of hybrid system behaviour. Therefore in order to model this system, integer variables are required to simulate the gears, while continuous variables are necessary for the car and engine speed.

Another example in the motoring industry of a hybrid system is “hybrid vehicles.” Toyota introduced the Prius, the world’s first mass-produced hybrid car, to the Japanese market in 1997. Hybrid technology is said to introduce the best of two worlds to the operation of the motor car. In this hybrid vehicle, the power and self – contained mobility of an internal combustion engine are combined with the clean efficiency of an electric motor. The result is a car that is the equal of conventional vehicles in range and performance, with far fewer harmful emissions and much better fuel economy. This system is hybrid because it brings together two concepts into a single entity and operation requires that these concepts are switched between or function in tandem.

2.2 Hybrid system control

Discrete variables, used as part of a model to describe hybrid behaviour, represent decisions that have to be made regarding the operation of that system. These decisions are usually left up to some form of intelligence or control procedure. In process control, the control of hybrid systems then involves the evaluation or initiation of these decisions. One way to initiate these decisions is to trigger them based on the optimal dynamic performance of the system extending into the future (see chapter 3 model predictive control).

If the dynamics of the system are known, together with the interaction between the variables, the operation of the system can be optimised. The optimisation of systems with discrete/continuous dynamics, coupled with integer decisions and constraints, is a problem in Mixed Integer Dynamic Optimisation (MIDO) dealt with in chapter 4.

An example of a MIDO problem is when the accelerating car described in the previous section, has to maintain a certain speed, while using minimum fuel. The controller outputs would be the accelerator position (continuous) and gear selection (discrete). The control mechanism could then select the appropriate gear and optimum accelerator position in order to meet the objectives of speed and economy. Certain constraints would have to be accounted for e.g. engine speed limits i.e. minimum/maximum allowable engine speed in a given gear. These are used to protect the engine from stress caused by high engine speeds and strain caused by low engine revs. The optimisation could be done repeatedly in real time, to account for changes in the gradient of the road (up-hill or downhill) which will alter the load on the engine. Hence the car inputs viz. gear and accelerator position would have to be changed accordingly.

It is apparent that a model of a hybrid process has to account for the combined continuous and discontinuous features that characterise the process. The occurrences of these features are now discussed, together with control and modelling considerations.

2.2.1 Continuous feature

Basic phenomena in process control are inherently time continuous. Hence dynamics can often be described using physical properties. Modelling often leads to a system description in state space form,

$$\dot{x}(t) = f(x(t), u(t), t) \quad (2.1)$$

where $u(t)$ is a vector of external inputs.

2.2.2 Discontinuous feature

It is the discontinuous feature that brings about the hybrid nature into processes. In the processing industry, there are many kinds of discontinuous features. These are discussed below.

a. Different modes of physical behaviour

As pointed out in section 2.2.1, continuous dynamics can be described generally by equation 2.1, where x represents variables like level, temperature and concentration. Function f is generally smooth, but in certain instances may contain discontinuous changes that reflect a change in the process. Physicochemical discontinuities that are inherent in the operation of the process bring about these changes. In this instance, more than one function is required to completely define the operation of the process.

b. Discontinuous outputs

Discrete sensors are used often in the processing industry, even if there is no apparent process discontinuity. For example a discrete level sensor will signal that the level of a tank is above or below a critical value. Typically such sensors are used to indicate constraints of operation.

c. Discontinuous inputs

Discontinuous inputs are the most common discrete feature in the processing industry. An example of a discontinuous element in process control is the discrete actuator e.g. binary valve (open/closed), a pump (on/off) or mixing motor (on/off). These are examples of binary actuators, but other non-binary examples exist e.g. 3-valued motor driven actuator (forward/backward/stop). Such actuators form an important group of process components and together with the control functions associated with them

e.g. manual/automatic modes, supervision, sequence control etc, constitute a significant part of application specific software in control systems.

To be able to integrate the discrete actuators in the overall process modelling and system design, specific modelling concepts will have to be developed. There are important distinctions between these discontinuous features (sections b and c) and those discussed previously (section a). In this case, the events signifying a change of state will usually be a controllable event. In the previous case (section a), the system can be described by more than one set of equations that are continuous in nature, however the discontinuity exists in that only one set is active at any given moment. For discontinuous inputs, an explicit model of the actuator states and transitions are needed to be able to describe the system completely.

d. Discontinuous control

The final example of discrete features in process control is more related to the control strategy than the process itself. The most apparent example of this is found in the control of batch processes. Here it is natural to specify overall procedures or recipes as comprising a number of phases. During each phase, the control system acts in a specific mode, performing functions like filling, emptying, heating, mixing etc. In these situations, it is convenient to distinguish between different phases or modes via discrete variables (e.g. by multiplying changes occurring by 0 or 1) and then to couple these to different sectors in the control system.

Gain scheduling of controllers also gives rise to a coupled discrete – continuous system description. In advanced control systems like adaptive controllers, various switching functions are usually coupled to continuous control algorithms.

Furthermore with regards to discontinuous control, there exists a new and fascinating discipline bridging control engineering, mathematics and computer science, referred to as “hybrid control systems.” Hybrid control systems arise from the interaction of discrete planning algorithms and continuous processes and as such they provide the basic framework and methodology for the analysis and synthesis of autonomous and intelligent systems. These types of systems contain two distinct types of components i.e. subsystems with continuous dynamics and subsystems with discrete dynamics which interact with each other. Figure 2.2 is an illustration of digital computation, which is essentially discrete, but must interface to processes that are ultimately continuous. Such systems arise from computer aided control of continuous processes, communication networks, auto pilot design, computer synchronisation and industrial process control.

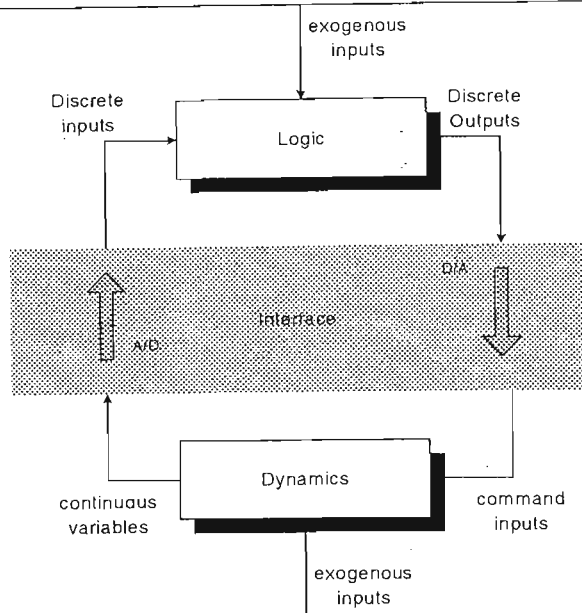


Figure 2.2: Interaction between finite state machines and continuous dynamical systems through analog-to-digital (A/D) and (D/A) interfaces

2.3 Literature Review

Over the past decade, there has been increasing interest shown in the study of hybrid systems, both from a theoretical and a practical point of view. There has been considerable effort to develop theoretical frameworks and models for such systems. Despite much research, there as yet does not exist a unified approach to deal with hybrid systems. Rather there are several theoretical frameworks that are motivated by specific tasks and applications.

The work done in the field of hybrid system modelling is extensive, because of the vast number of processes that can be considered as hybrid in nature. Most notably, much documentation comes from the hybrid control systems domain (Barton and Pantelides, 1994; Antsaklis, 1998; Mignone *et al.*, 1999), where computers and their algorithms, due to their digital nature, can be considered as hybrid systems.

There are many publications concerning the modelling, simulation and control of hybrid processing systems. Lennartson *et al.* (1994, 1996) were of the first to document the modelling and control of hybrid systems with particular emphasis on process control applications. A general model structure for hybrid systems was proposed. The model structure separated the openloop plant from the closed loop system and was said to be suitable for analysis and synthesis of hybrid control systems. The controller issued discrete/continuous commands to the plant based on feedback information. The control problem was treated as a time optimal one.

Barton and Pantelides (1994) introduced a new formal mathematical description of the combined discrete/continuous simulation problem. This enhanced the understanding of the fundamental discrete changes required to model processing systems. The model task was decomposed into two distinct

activities: modelling fundamental behaviour and modelling external actions imposed on the system. Both required significant discrete components.

Henzinger (1996) presented the Theory of Hybrid Automata. Here hybrid automata were classified according to what questions about their behaviour could be presented algorithmically. The classification revealed structure on mixed discrete – continuous state spaces that were previously studied as purely discrete state spaces only.

Slupphaud and Foss (1997) and Slupphaud *et al.* (1997) were one of the first cited researchers to use MPC for controlling hybrid systems. Together with the research group at the Swiss Federal institute of Technology (ETH) Zurich, they present the only attempts at using a model predictive approach in hybrid system control.

Lygeros *et al.* (1999) provided an overview of modelling, analysis and controller synthesis techniques for hybrid systems. A modelling framework for describing a wide class of hybrid phenomena was initially presented and then conditions for existence and uniqueness of solutions were provided. A method for designing controllers for hybrid systems with reachability specifications was also reviewed.

Mostermann (1999) dealt with an investigation into software packages regarding their support for hybrid system simulation. Based on hybrid system phenomena, numerical simulation required the implementation of specific features, similar to those discussed in section 2.2. An evaluation of several existing hybrid system simulation packages was also presented.

Riedinger and Krantz (1999) proposed the maximum principle of Pontryagin as an efficient way to solve optimisation problems in hybrid systems. After presenting a general structure for optimal control of hybrid systems, they make use of the maximum principle in order to get necessary conditions for optimal control.

Pepyne and Cassandras (2000) proposed a specific framework motivated by the common problem occurring in the manufacturing industry. In the manufacturing industry, the aim is to design a control strategy to maximise production within minimum time, but to maintain certain standards of quality. A hybrid system modelling framework was designed to address this problem. Ideas from queuing networks were adapted to form a non-linear optimisation structure.

Recently Sedghi *et al.* (2002) presented a controller design framework for hybrid systems based on approximate continuous time models and standard continuous control techniques. A model reduction procedure was used that eliminated all discrete states. Since the reduced system lost its hybrid nature, the procedure is termed dehybridisation.

There are also research groups that dedicate their efforts to hybrid system modelling. Most notably, Manfred Morari and his colleagues at the Automatic Control Laboratory¹ in the Swiss Federal institute of Technology (ETH) Zurich, have done the most extensive research in hybrid system modelling. They base all of their investigations concerning hybrid system modelling on their mixed logic dynamic (MLD) framework (Mignone, 1999). This generalised framework is used for control, fault detection, state estimation and verification of hybrid systems. Using the techniques of Raman and Grossmann (1991, 1992) propositional logic is transformed into linear equalities involving integer and continuous variables. In this transformation, a system of equations comprising linear dynamic equations and linear mixed integer inequalities is arrived at. Regarding the control of hybrid systems, the MLD framework is used as the foundation for a model predictive control strategy that is aimed at stabilising hybrid systems (Bemporad and Morari, 1998). Based on the MLD formulation, a number of industrial processes were modelled and controlled using a MPC strategy e.g. a gas supply system (Bemporad and Morari, 1998); furnace heating with 3 modes (Bemporad *et al.*, 2000); hydroelectric plant (Ferrari-Trecate *et al.*, 2000 and 2002); and even traction control for cars (Borrelli *et al.*, 2001). Further investigation by Torrisi *et al.* (2001) showed that the performance of a control strategy, using a MPC strategy and the MLD framework, compared favourably with a controller based on heuristic rules.

Other research groups in the field of hybrid system modelling are the Computer – Aided Systems Laboratory (CASL)² at Princeton University and another at Carnegie Melon University³. World leaders in the field of mixed integer optimisation i.e. C.A Floudas and I.E Grossmann head both groups respectively. These research groups are focused on developing rigorous and powerful theoretical algorithmic approaches based on advances in mixed integer non-linear optimisation. Most notably CASL has developed MINOPT, a procedure used in GAMS as a mixed integer solver.

Besides publications and the work done by various groups, several projects and workshops have been held to expand on expertise in the hybrid system arena. The VHS project⁴ was initiated in May 1998 with the principle aim to develop methods and tools for hybrid system verification. Another project, Hybrid-EC-US043⁵ was started in January 1995. The objectives of this project were to develop a theoretical basis for analysing hybrid systems and to build tools for improving the development of computer controlled systems.

The annual staging of conferences and workshops that attract leaders in this area, highlights the importance to the engineering and computer science industries of continuously evolving knowledge regarding hybrid systems. The hybrid system computation and control (HSCC) workshops attract researchers from industry and academia interested in modelling, analysis and implementation of

¹ <http://www.control.ethz.ch>

² <http://titan.princeton.edu/work.html>

³ <http://www.cheme.cmu.edu/who/faculty/grossmann.html>

⁴ <http://www-verimag.imag.fr/VHS/>

⁵ <http://albion.ncl.ac.uk/esp-syn/text/ec-us043.html>

dynamic systems involving both discrete and continuous behaviour. The fifth international workshop was held in March 2002 at Stanford California.

2.4 Hybrid Systems in Industry

Presented in this section are examples of industrial processes that have hybrid system characteristics. Common to these systems is the requirement that the controller is able to make informed discrete decisions regarding the operation of that particular system. In the case where the system has physicochemical discontinuities, the controller must be able to anticipate the change in dynamics at the point of discontinuity.

2.4.1 Pressure Swing Adsorption

Pressure swing adsorption (PSA) was invented in the 1960's for the purification of gases e.g. the removal of moisture from air. However, presently it is also used for bulk separations such as the partial separation of air to produce either nitrogen or oxygen and also the removal of impurities and pollutants from gas streams. Separation of the components is achieved by the adsorption under pressure, of a component from the mixture, onto a substrate e.g. zeolite packing in the case of air separation. The adsorbed component is then desorbed from the surface of the substrate when the pressure is lowered.

PSA is a typical illustration of a switching system. Such systems have a finite number of modes of operation and the controller design is based on when to initiate the switch from one mode to another. Often the switches are cyclic in nature, as is the case with PSA.

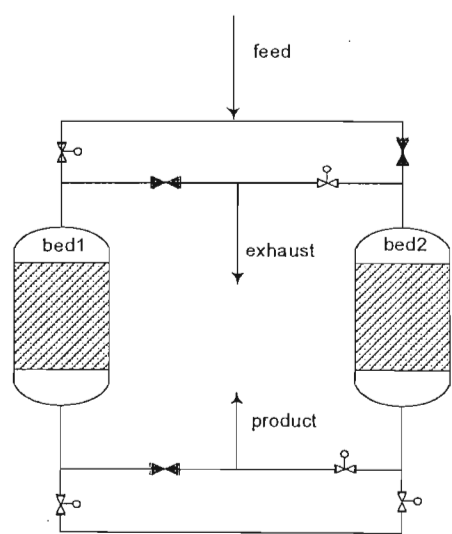


Figure 2.3: Pressure Swing Adsorption

In the PSA cycle, adsorption takes place at an elevated pressure, whereas desorption typically occurs at near ambient pressure. In its simplest configuration PSA is carried out with two fixed beds in parallel, operating in a cycle, as shown in figure 2.3 above. While one bed is adsorbing at one pressure, the other is desorbing at a lower pressure. A typical sequence of steps is shown below.

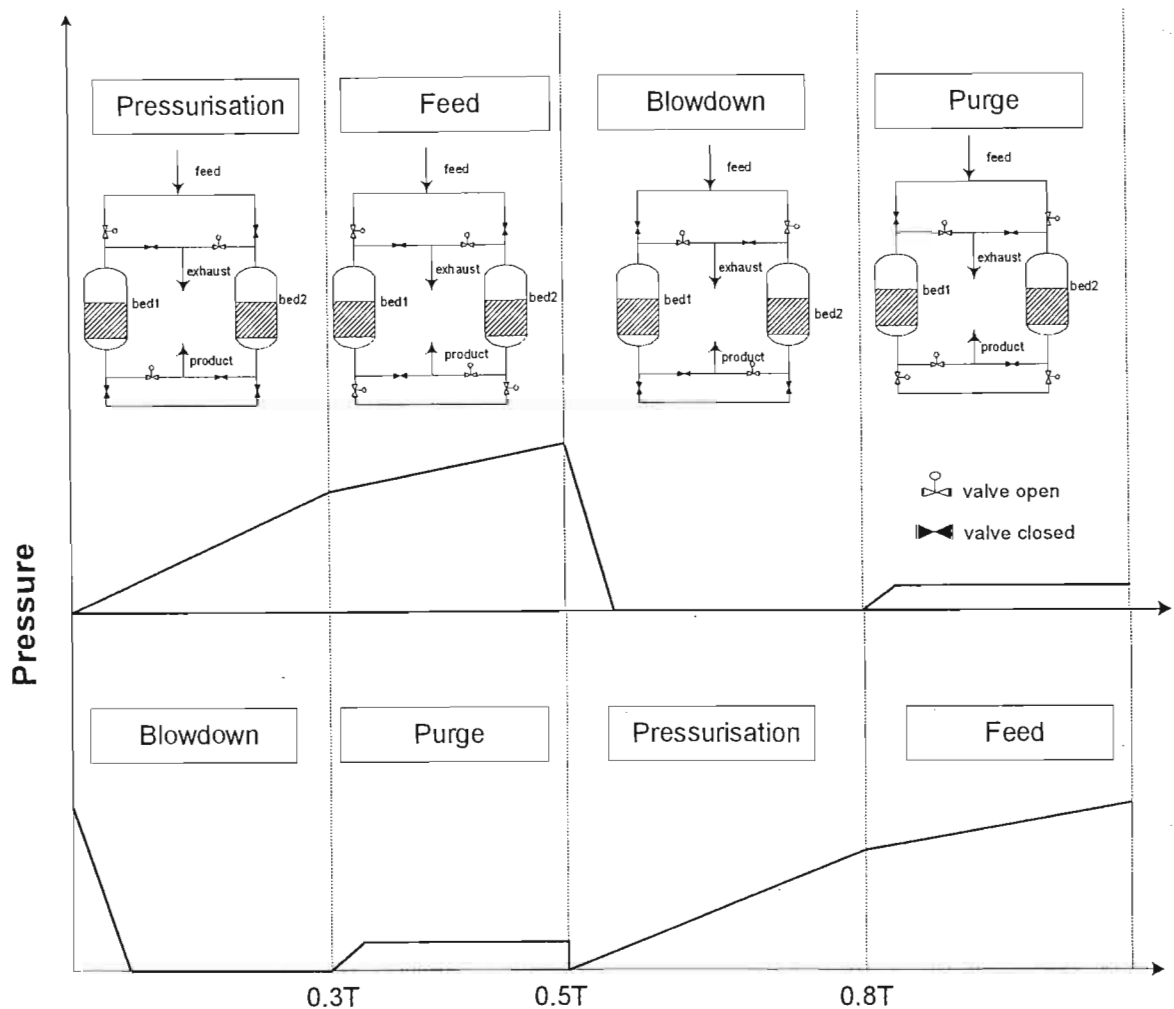


Figure 2.4: Sequence of cycle steps in PSA

Each bed operates alternately in two half-cycles ($0.5T$, where T is the full cycle time) of equal duration: (1) pressurisation followed by adsorption and (2) depressurisation (blowdown) followed by a purge. The feed gas is used for pressurisation, while a portion of the effluent product is used for the purge of the next bed. Thus, as in the above figure, while adsorption is taking place in bed1, part of the gas leaving bed1 is routed to bed2, to purge that bed in the direction countercurrent to the direction of flow of the feed gas during the adsorption step. During the second half of the cycle, the valve openings and the beds are switched.

There exist similar types of switching systems in the chemical industry e.g. temperature swing absorption, reverse flow reactors; swing reactors that switch when catalyst is being regenerated and simulated moving-bed (SMB) chromatography.

2.4.2 Selection processes from Inventories

Some chemical systems require the continuous selection of procedures, from a list of alternatives, that directly affects the operation and productivity of the system. The hybrid nature is introduced by the discrete nature of the decisions and the effect that they would have on the process. Considered here is a conversion process with various possible recipes and product grades.

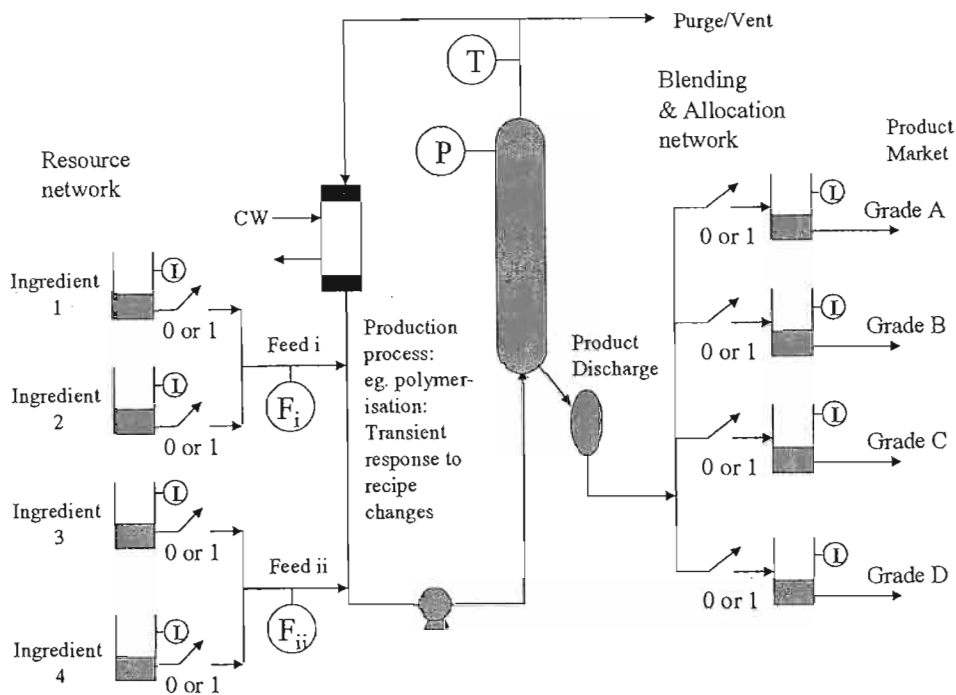


Figure 2.5 Blending system

The above system takes raw materials from a resource network, separates them into cuts, and then recombines them in the appropriate proportion, to form different grades of product. Choices have to be made about the appropriate ingredient to use from the resource network and how to combine the different splits to meet specifications.

The important feature is that there are dynamic responses that link the discrete choices (discontinuities). A simple problem would be the matter of correcting grades in the product storage, based on the present inventory. For example say that grade A was lighter than specification, then denser material might have to be fed into the A storage for a period to raise the fixed density. In oil refineries this is usually done with “blending controllers” which tend to be based on ‘in-line’ or static blending (using multivariable LP solutions). The actual situation may however be more complex, with inter-tank transfers being allowed and maximum/minimum inventories being set for any tank, as is the case in the distribution of drinking water between reservoirs.

It is easy to appreciate that this type of problem, i.e. searching for an optimal choice in order to meet objectives, requires an optimisation procedure. The selection is based on a number of aspects that

directly or indirectly influence the process, e.g. resources available, present inventories, market demand and also the time period where the process will have a transient “ off specification” response to recipe changes.

Other types of systems that rely on similar decisions are:

- the selection of a solvent or reagent, that is appropriate for a certain operation, from a list of candidates;
- selection of equipment items from an inventory available within a plant to perform a specific duty;
- initiation of reaction, separation or recycle structures at certain points in an operation;
- triggering the optimal number of units to perform a certain operation, e.g. determining the appropriate number of heat exchangers/reactors/separators in order to meet a certain demand while minimising cost.

2.4.3 Heating /Cooling strategies for batch reactors

The core of the drug manufacturing process is the batch or batch fed reactor, which is widely used in the pharmaceutical industry. Due to its flexibility, a single batch reactor can be used to carry out different reactions and operations under various operating conditions. During a batch reaction the reaction mixture has to adhere strictly to a certain temperature trajectory, to ensure firstly the synthesis of the appropriate product and secondly that the product is of the correct quality. It is for this reason that the temperature control of batch reactors is important.

To account for the wide range of temperature, over which the reaction mixture needs to extend, two different approaches exist in the pharmaceutical industry for reactor temperature control. These are the use of either a *multifluid* (figure 2.6) or a *monofluid* (figure 2.7) system.

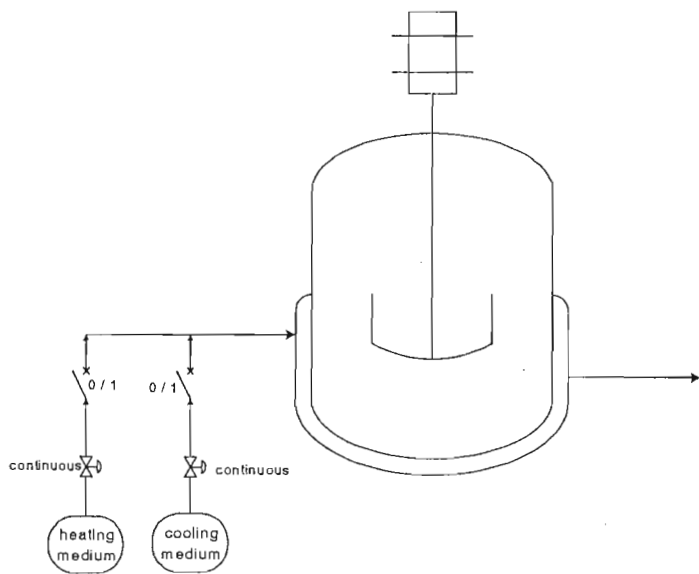


Figure 2.6: Multifluid temperature control

The multifluid system uses the direct injection of the required utility into the reactor jacket. The control task consists of firstly selecting the correct fluid and secondly the appropriate control action on the flowrate of the selected fluid, in order to track the desired temperature profile.

The monofluid system uses only one fluid, the temperature of which can be modified to achieve the desired reactor temperature via a thermal loop that includes heat exchangers. The main advantage of using a monofluid system, is that it offers smooth and continuous evolution of the temperature of the thermal fluid over a wide range. The drawback of using this approach is the slow dynamic responses, when switching from one exchanger to another, especially in the case of urgent need of rapid cooling or heating. Another disadvantage of the monofluid system is the high investment costs incurred due the need for heat exchangers.

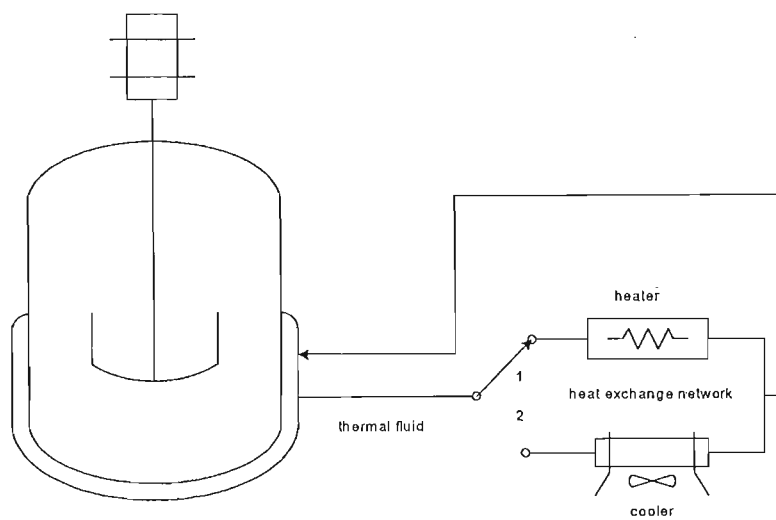


Figure 2.7: Monofluid temperature control

Both types of heating/cooling strategies exhibit hybrid behaviour, because both require selection processes. In the case of multifluid system, the selection procedure concerns the thermal fluid while for the monofluid system, the selection has to do with equipment. These selections result in discontinuities in the temperature control strategy. Once the selection has been made, then the appropriate amount of heating/cooling has to be determined for setpoint tracking.

Preu *et al.*(2002) propose a new heating/cooling strategy (figure 2.8) to incorporate advantages of both the multifluid and monofluid systems. They have combined the concepts of both systems and come up with a new system whose main characteristic is the use of an intermediate utility i.e. pressurised water, generated by mixing steam and water. By adjusting the ratio of steam to water, the desired temperature is obtained. For example, at a pressure of 3 bar, pressurised water covers a temperature range from 20-120 °C. Within this range, heating/cooling exhibits a monofluid type behaviour. If greater heating/cooling is required, then the system switches to multifluid behaviour. After the jacket is purged, steam or cold utility is directly injected into the reactor jacket. To allow for

setpoint tracking, the controller acts on the flowrate of the fluids as manipulated variables. In the case of the intermediate fluid (pressurised water) – the flowrate is kept constant and the ratio of steam to water is adjusted so that the desired temperature of the intermediate fluid is reached.

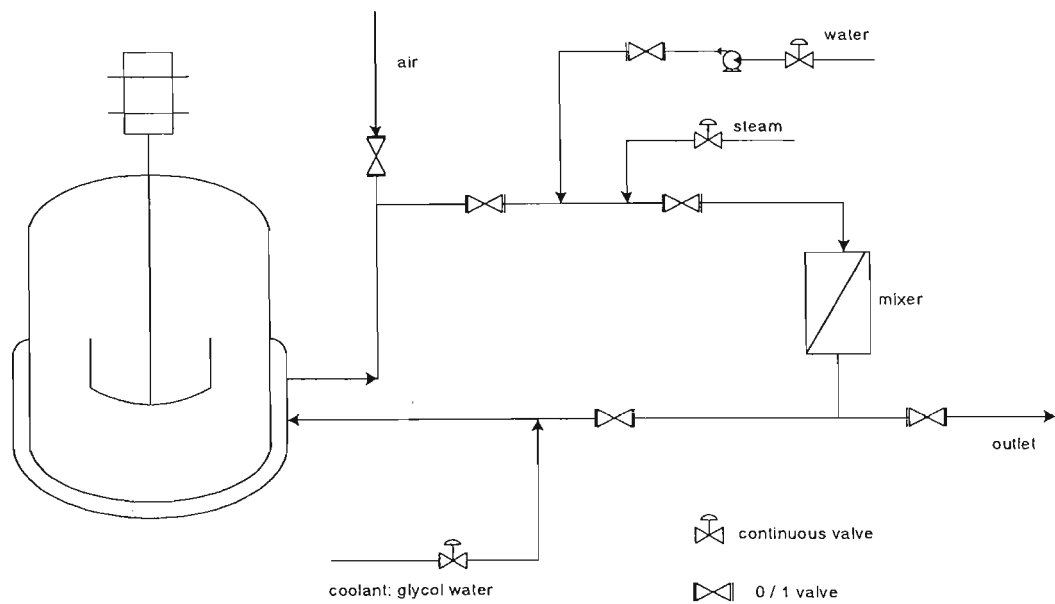


Figure 2.8 Monofluid and multifluid temperature control

The controller must therefore, for the sake of setpoint tracking, select the appropriate thermal fluid (discrete decision) and then set the appropriate continuous control valves. With the presence of discrete decisions and continuous variables, the hybrid nature of this temperature control strategy is thus apparent.

Figure 2.9 shows an example of a classical operation, under closed loop control, carried out in a batch reactor taken from Preu *et al.*(2002). This is a crystallisation performed over a 25-hour time interval, according to a very precise temperature setpoint profile. The precision in the temperature is of high importance since it has an effect on the quality of product obtained. Note the distinct areas of operation where different thermal fluids are used and the switching between the different fluids, based on a change in setpoint trajectory.

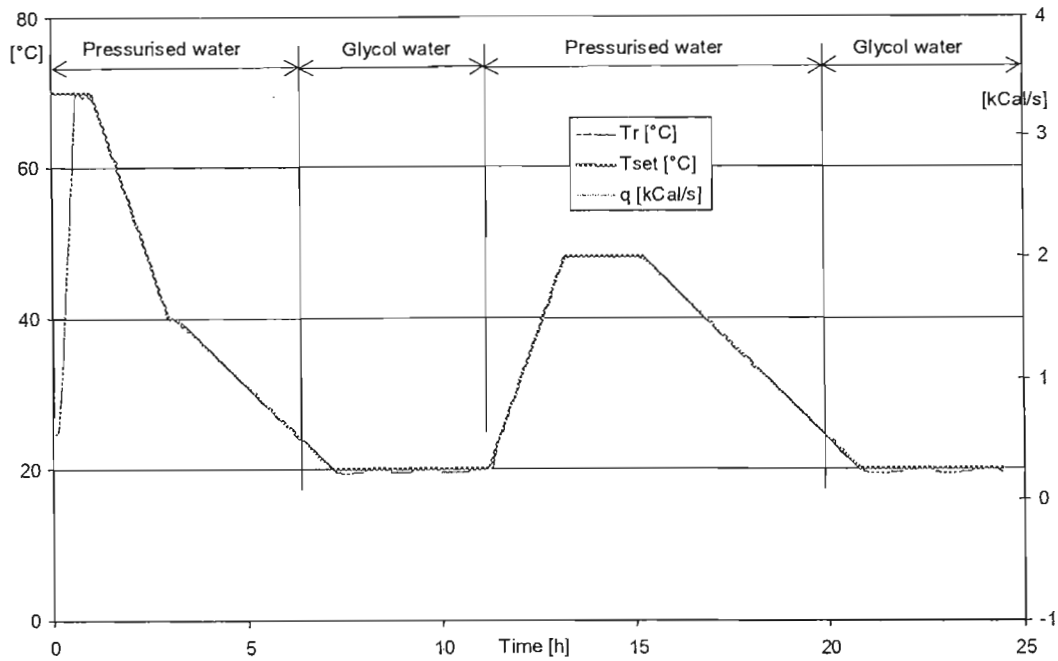


Figure 2. 9. Production of a specified chemical product by crystallisation according to a specific temperature profile

2.4.4 Tank with a weir

Some processing systems have discontinuities due to their fundamental physical behaviour. These physicochemical discontinuities usually arise from thermodynamic (e.g. phase), fluid mechanical (e.g. from laminar to turbulent regime) transitions, or from the geometry of the process vessels (e.g. non uniform cross-sections). Changes between the various regimes are state events, because the time of switching is not known in advance. It is rather a function of the system's state.

A simple chemical engineering illustration of a system showing a physicochemical discontinuity is a buffer tank fitted with an overflow at a fixed height (figure 2.10).

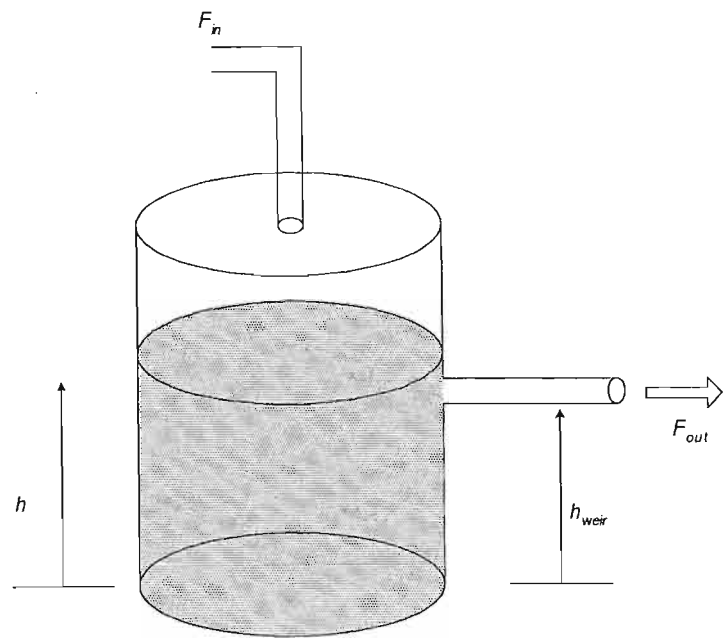


Figure 2.10: Buffer tank with overflow

The following equations could be used to model the influence of the weir on the dynamics of the buffer tank.

$$\left. \begin{aligned} F_{out} &= k_{weir} \sqrt{h - h_{weir}} \\ \frac{dh}{dt} &= F_{in} - F_{out} \end{aligned} \right\} h(t) \geq h_{weir}$$

$$\left. \begin{aligned} F_{out} &= 0 \\ \frac{dh}{dt} &= F_{in} \end{aligned} \right\} h(t) \leq h_{weir}$$

h is the level of liquid in the tank and h_{weir} is the height of the weir. This formulation models the fact that whilst the liquid level is above the weir, liquid flow out of the overflow (F_{out}) will be driven by the head above the weir, whereas while liquid level is below the weir, there is no flow. At the same time, the rate at which the tank fills, is also dependent on the level relative to the overflow. The discontinuity in states: tank level and weir flowrate, is presented graphically in figure 2.11 for a fixed inlet flowrate F_{in} . State variables are continuous within regimes of operation, but are discontinuous at a certain threshold point, defined by one of the states (in this case level).

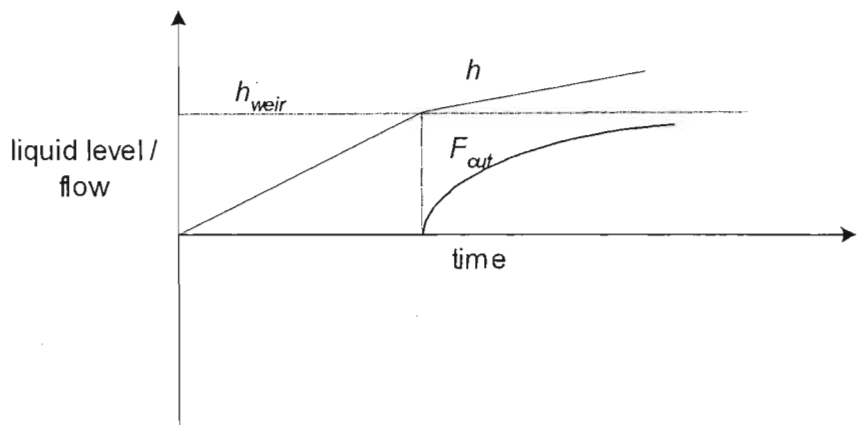


Figure 2.11: Discontinuous state behaviour in tank-weir system

Controller design for systems with physicochemical discontinuities has to account for the change in dynamics at the point of discontinuity. The state of the internal controller model will have to be switched at the point of discontinuity, so that the dynamics of the system can be correctly predicted. Other systems in the chemical industry that exhibit physicochemical behaviour are: pressure relief systems; vaporisation in vessels; crystallisation; and condensation followed by sub-cooling.

CHAPTER 3

Model Predictive Control

The term model predictive control (MPC) describes a class of computer control algorithms that control the future behavior of a plant through the use of an explicit process model. Using MPC strategies to control and stabilise hybrid systems is for the following reasons a natural and intuitive idea:

- Firstly, the MPC framework is not restricted in terms of model, objective function and/or constraint functionality. The underlying idea is easy to understand, thus making modifications easy. This versatility makes it possible to incorporate integer variables within the structure.
- Secondly MPC with integer variables has exactly the same structure as that used for MIDO covered in the following chapter. Both rely on dynamic models, take into account constraints and are dynamic optimisation procedures.
- Thirdly, MPC has brought with it great success regarding the control of constrained, multivariable systems in the processing industry. There have been numerous reports on the successful implementation of MPC strategies in the academic and industrial arenas over the past twenty years (Morari and Lee, 1999). The proposed algorithm deals directly with control i.e. the applications in this thesis focused on manipulated system inputs (valves), however industrially a hybrid multivariable algorithm is likely to work through a base layer of slave controllers, e.g. PID, to achieve outputs.

This chapter deals with model predictive control and how it can be adapted to control hybrid systems. It commences with section 3.1, which is an overview of MPC, explaining the strategy of MPC and why it has had such success in the control environment. Following this, section 3.2 is an explanation of the dynamic matrix control (DMC) algorithm, which is a form of MPC used in this thesis for the proposed control of hybrid systems. After the explanation of DMC, the following section illustrates how the DMC structure can be adapted to accommodate integer variables. Lastly section 3.3 shows the control loop used in the subsequent chapters, for validating the mixed integer predictive control algorithm (MIPC).

3.1 Overview

Interest of the processing industry in MPC can be traced back to the late 1970's (Garcia and Morshedi, 1984). In 1978 Richalet *et al.* described successful applications of "Model Predictive Heuristic Control." A year later engineers from Shell (Cutler and Ramaker, 1979; Prett and Gillette, 1979) outlined "Dynamic Matrix Control" (DMC) and reported applications to a fluid catalytic reactor. In both algorithms, an explicit dynamic model of the plant was used to predict the effect of future actions of manipulated variables on the output. Thus the name "Model Predictive Control." The future "moves" of the manipulated variables were determined by optimisation, with the objective being minimising the predicted error between the output states and a target, subject to

operating conditions. The optimisation was repeated at each sampling time, based on updated information (measurements) from the plant. Thus in the context of MPC, the control problem, including the relative importance of the different objectives and the constraints etc, is formulated as a dynamic optimisation problem. While this is hardly a new idea, Garcia *et al.* (1989) state that it constitutes one of the first examples of large-scale dynamic optimisation applied routinely in real time.

Although it is more than twenty years ago that MPC first appeared in industry as an effective means to deal with constrained control problems, it is far from being a saturated field. Over the past few years there have been several publications that track its history and forecast its future, together with possible areas of expansion. Most notably has been the publication by Qin and Badgwell (1999) that provides an overview of nonlinear model predictive control (NMPC) applications in industry, focussing primarily on recent applications reported by MPC vendors. Other researchers that have followed suit by providing their interpretation and perspective as to the evolution of the field are Garcia *et al.* (1989) and Morari and Lee (1999).

Figure 3.1 shows how MPC is used to control a process. It is coupled to the process in a feedback sense because it requires current state outputs for its computation of new process inputs. The controller provides the plant with optimum inputs at regular intervals in time. The period between controller intervention is commonly referred to as a time step or sampling instant. The main components of the controller are the internal model and optimiser. Figure 3.2 shows a “snap shot” in time of process variables while under supervision of a MPC. This snap shot is at a single moment in a moving horizon that extends ahead in time. This concept is characteristic of MPC strategies. The single snap shot can be at any given time step when the controller provides the plant with new, optimal inputs. When the controller reevaluates the optimum inputs at the next time step, its extended horizon over which the outputs are projected (i.e. optimisation horizon), although remaining fixed in size, advances one step into the future. Hence the term “moving horizon”. Figure 3.2 is thus a graphical illustration of what occurs within the controller at every time step.

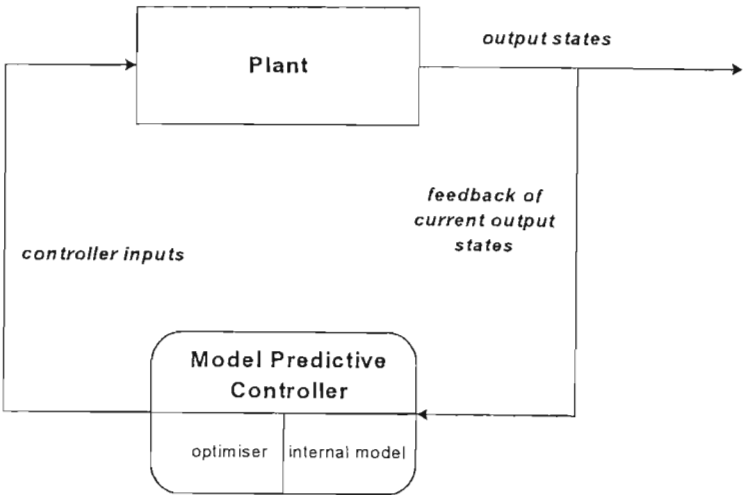


Figure 3.1: Feedback control loop showing plant and model predictive controller

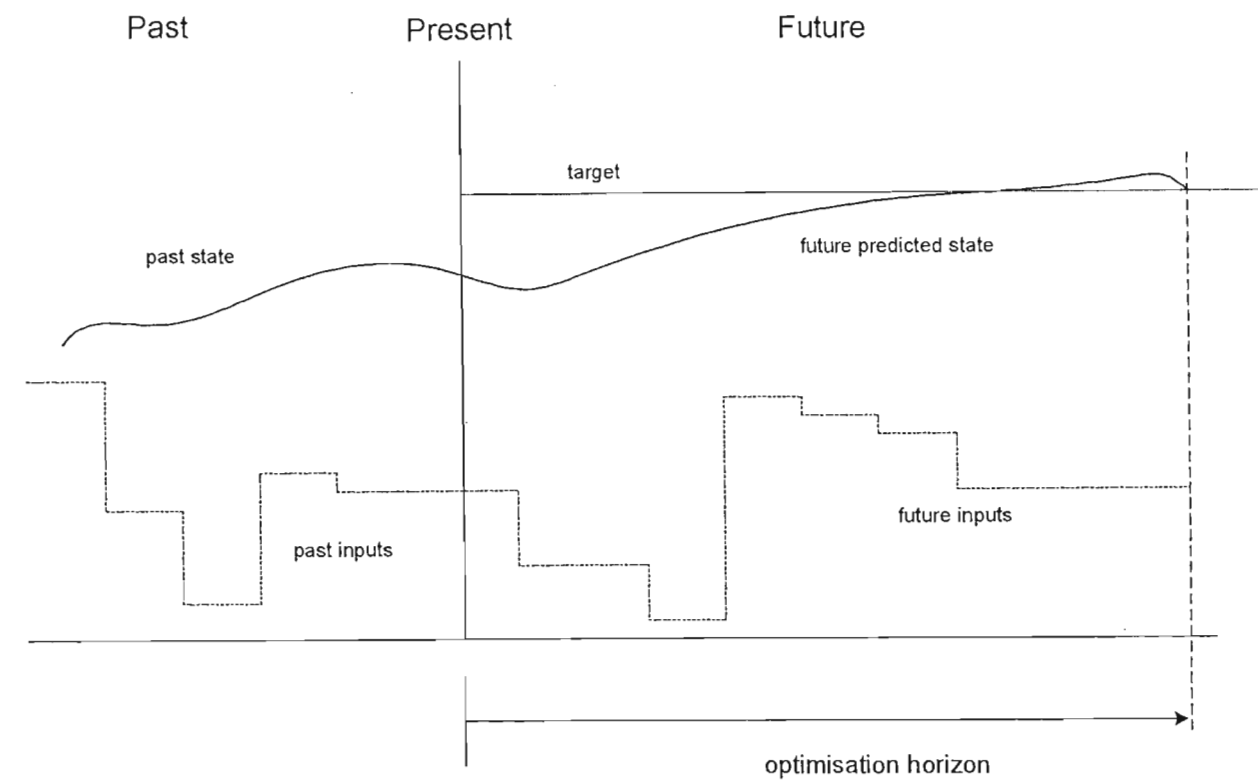


Figure 3.2: "Snap shot" of the moving horizon approach to MPC

The above figures typify the MPC strategy. The first procedure is to use a model of the system to predict the future evolution of the system over the prediction horizon (figure 3.2). Based on this prediction at each time step, the controller selects a sequence of future command inputs through an on-line optimisation procedure, aimed at maximising tracking performance and obeying constraints. Only the first sample of the optimal sequence is applied to the plant at the present time step. At the following time step, a new sequence is evaluated to replace the previous one. This online “replanning” provides the desired feedback control feature.

Most of the applications using MPC reported in the literature are multivariable and involve constraints. It is exactly these types of problems that initially motivated the development of MPC control techniques. The success of MPC technology, as a process control paradigm, can be attributed to three important factors. Firstly is the incorporation of an explicit process model into the control calculation. This allows the controller, in principle, to deal directly with all significant features of the process dynamics. Secondly, the MPC control algorithm considers plant behaviour over a future horizon in time. This means that the effects of feedforward and feedback disturbances can be anticipated and removed, thereby allowing the controller to drive the plant more closely along the desired trajectory. Finally the MPC controllers consider the process input, state and output constraints directly in the evaluation. This means that constraint violations are far less likely, resulting in tighter control at the optimal constrained steady state for the process, than for example, if the calculated valve

settings were just “clipped” at 100%. It is the inclusion of the constraints that most clearly distinguishes MPC from other process control paradigms.

Another aspect that has contributed to the success of MPC strategies is its flexibility and portability. Economics demand that control systems be designed with no “hard wiring” or ad hoc “fix ups.” This means that a controller designed for a single unit can be easily transported to another, by just replacing some key information. In the case of MPC, this is just the model and constraints which are characteristic of the process.

3.2 Dynamic Matrix Control (DMC)

3.2.1 Introduction

The different design techniques emanating from MPC are: Dynamic Matrix Control (DMC); Model Algorithmic Control (MAC); and Model Reference Adaptive Control (MRAC). The fundamental framework of MPC consists of certain distinct elements that are shared in common by these schemes. These distinct elements, as alluded to in the previous sub section, are:

- reference trajectory specification: representing desired target trajectories for process outputs;
- process output prediction: utilising a mathematical representation of the process to predict the process output over a predetermined extended horizon;
- control action sequence computation: where the same model is used to calculate future control moves of the manipulated variables that must satisfy objectives.

What differentiates one scheme (DMC,MAC,MRAC) from another, is the strategy and philosophy underlying how each scheme is implemented.

Of the three schemes, Morari and Lee (1999) declare that DMC is probably the most popular model predictive control algorithm currently used in the chemical process industry. Although one of the earliest formulations of MPC, DMC represents the industrial standard for MPC to date. The reason for this wide acceptance of DMC is due to its ability to handle process interactions and unusual dynamic responses, without requiring a rigorous process model derived from first principles. In the same publication, the authors state that most systems in the oil and chemical industries are multivariable constrained systems. DMC, with its ability to handle constraints, is therefore more suited to the control of such systems than other techniques that are unable to handle constraints.

Cutler and Ramaker (1980) of Shell Oil reported the fundamental work on the formulation of DMC. Four years later, Garcia and Morshedi (1984) reported on a quadratic programming solution. Since its inception, the DMC algorithm has been widely used in the processing industry, while simultaneously undergoing various advancements to make it applicable to a wider range of control problems. Linear dynamic matrix control (LDMC), based on a linear programming solution for optimal control policy, was developed by Mulholland and Prosser (1999) following the methods of Chang and Seborg (1983) and Morshedi *et al.* (1985). Mulholland and Prosser (1999) used this LDMC algorithm to control the top and bottom temperature of a semi-industrial distillation column, while Mulholland and Narotam

(1996) considered the control of a counter-current liquid-liquid extractor. Recently Lacave (2001) used the DMC framework for the modelling and control of a co-current sugar dryer while Pastre (2002) used it to control chlorine levels in reservoirs for potable drinking water. Regarding the extension of DMC to deal with other control problems, Guiamba (2001) showed how the algorithm can be expanded to deal with integrating systems i.e. systems that do not reach a steady state when disturbed e.g. reservoir levels. The work done by Guiamba is essential to the work presented in this thesis because some hybrid systems are integrating in nature. Further advancements include formulation of multi-rate DMC that deals with fast and slow output responses in the same system (Deghaye *et al.*, 2000); and adaptive DMC, that aims at real-time identification of step responses (Guiamba, 2001). Adaptive DMC is a step forward in using DMC, which is a linear algorithm, to control processes with non-linearities.

Any model predictive controller relies on a mathematical representation of the process. The model, typically in state space or convolution form, is used to predict the trajectories of the system's outputs up to a moving horizon. The following sub-section presents the development of the DMC algorithm that utilises a convolution model as a mathematical representation of the system, which is derived from step responses.

3.2.2 DMC algorithm

Consider a single input, single output system, initially at steady state. By introducing a unit step change in input m at time $t = 0$, the resultant output response x is recorded until it reaches a new steady state.

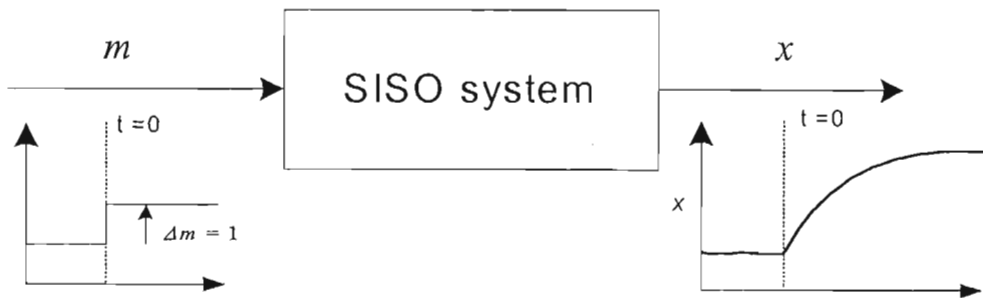


Figure 3.3: Step response for SISO system

Now take the response x and partition it into equally spaced, discrete-time instants (of total number N) that extend over the entire response.

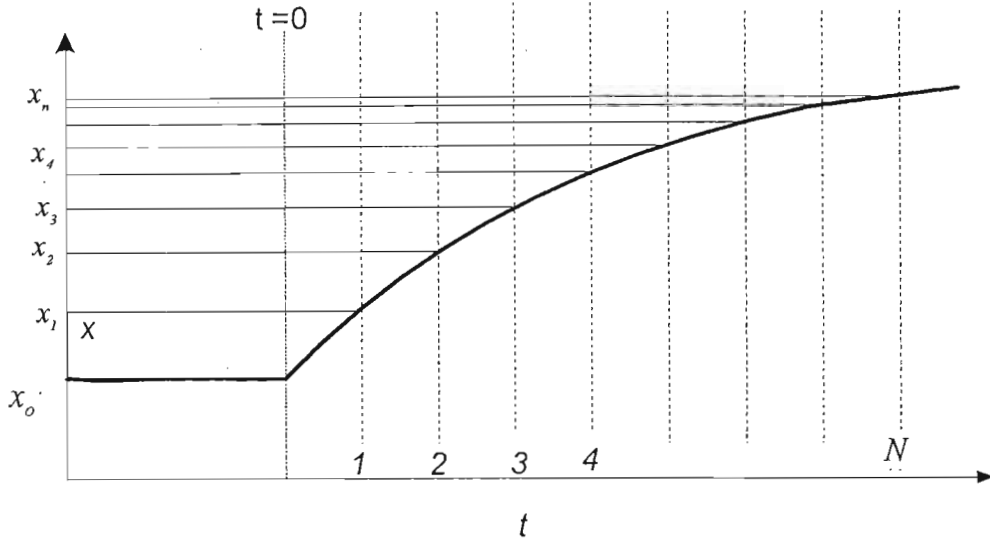


Figure 3.4: Partitioning step response for convolution model

For each sampling instant t , define the change from the initial steady state (x_o) prior to the step change as: $b_t = x_t - x_o$; for t extending from 1 to N (steady state horizon for step response). b_t is now a coefficient which represents the change from steady state, caused by a unit step change in input ($t=0$), at time step t .

Assuming linearity and that the system is initially at steady state x_o , the output states, for any change in input Δm (difference between absolute input at past time step and present absolute input), made at $t = 0$, can be evaluated using the step response coefficients:

$$\begin{aligned} x_1 &= x_o + b_1 \Delta m \\ x_2 &= x_o + b_2 \Delta m \\ &\vdots \\ x_N &= x_o + b_N \Delta m \end{aligned}$$

If a sequence of changes in moves are made in consecutive time intervals, starting at $t = 0$ and extending up to a control horizon of size N i.e. $\Delta m_0, \Delta m_1, \Delta m_2, \dots, \Delta m_N$, then using superposition the new state outputs can be evaluated:

$$\begin{aligned} x_1 &= x_o + b_1 \Delta m_0 \\ x_2 &= x_o + b_1 \Delta m_1 + b_2 \Delta m_0 \\ x_3 &= x_o + b_1 \Delta m_2 + b_2 \Delta m_1 + b_3 \Delta m_0 \\ &\vdots \\ x_N &= x_o + \sum_{t=1}^N b_t \Delta m_{N-t} \end{aligned}$$

The above set of linear equations can be expressed compactly in matrix form

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} x_o \\ x_o \\ x_o \\ \vdots \\ x_o \end{bmatrix} + \begin{bmatrix} b_1 & 0 & 0 & \cdots & 0 \\ b_2 & b_1 & 0 & \cdots & 0 \\ b_3 & b_2 & b_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ b_N & b_{N-1} & b_{N-2} & \cdots & b_1 \end{bmatrix} \begin{bmatrix} \Delta m_o \\ \Delta m_1 \\ \Delta m_2 \\ \vdots \\ \Delta m_N \end{bmatrix} \quad (3.1)$$

Equation 3.1 represents the dynamics of the system in the form of a convolution model, derived from step response data. Thus for any sequence of inputs $\Delta \bar{m}$, the resulting output states can be predicted. This piecewise convolution model forms the basis of the DMC algorithm.

Multivariable systems with multiple inputs and outputs (MIMO) can be modeled using equation 3.1, by expanding the step response coefficients b_i 's as two-dimensional matrices. The data for the matrices is generated by stepping each input in turn and recording the response for every output. Position (i,j) in each matrix b_i is the point on the trajectory, at time step t , for the i^{th} output as it responds to a step in the j^{th} input. Clearly also vectors \bar{X} and $\Delta \bar{m}$ will also be made up of sub-vectors, corresponding to the sequential steps in time, which compliment the dimensions of \bar{B} .

Mulholland *et.al* (2001) present explicitly how a step test identification approach (equation 3.1) can be used to develop the DMC algorithm which is here after presented in general. Consider the following “dynamic matrix” construction whose elements are multiplied by $\Delta \bar{m}_t$, the vector of input moves (changes) made at time t . The resulting product is $\Delta \bar{x}_t$, state vector of outputs at time t . A moving frame of reference for time is used in which $t=0$ represents present time. The “dynamic matrix” is constructed to predict P steps ahead ($P \geq N$), where N is the steady-state horizon for the step responses. (In the present work, P is taken equal to N).

$$\begin{pmatrix} x_{o \text{ Pred}} \\ x_{o \text{ Pred}} \\ x_{o \text{ Pred}} \\ x_{o \text{ Pred}} \\ \vdots \\ x_{o \text{ Pred}} \\ \hline x_1 \\ x_2 \\ \vdots \\ x_M \\ x_{M+1} \\ \vdots \\ x_P \end{pmatrix} = \begin{pmatrix} x_{o \text{ Pred}} \\ x_1 \\ x_2 \\ \vdots \\ x_M \\ x_{M+1} \\ \vdots \\ x_P \end{pmatrix} = \begin{bmatrix} b_N & b_{N-1} & b_{N-2} & b_{N-3} & \cdots & b_2 & b_1 & | & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ b_N & b_{N-1} & b_{N-2} & b_{N-3} & \cdots & b_2 & b_1 & | & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ b_N & b_{N-1} & b_{N-2} & b_{N-3} & \cdots & b_2 & b_1 & | & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ b_N & b_{N-1} & b_{N-2} & b_{N-3} & \cdots & b_2 & b_1 & | & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & | & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ b_N & b_{N-1} & b_{N-2} & b_{N-3} & \cdots & b_2 & b_1 & | & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \hline b_N & b_N & b_{N-1} & b_{N-2} & \cdots & b_3 & b_2 & | & b_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_N & b_N & b_N & b_{N-1} & \cdots & b_4 & b_3 & | & b_2 & b_1 & 0 & 0 & 0 & 0 & 0 \\ b_N & b_N & \vdots & \vdots & \ddots & \vdots & \vdots & | & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ b_N & b_N & b_N & b_N & \cdots & b_N & b_{N-1} & | & b_{N-2} & b_{N-3} & \cdots & b_1 & \cdots & 0 & 0 \\ b_N & b_N & b_N & b_N & \cdots & b_N & b_N & | & b_{N-1} & b_{N-2} & \cdots & \cdots & b_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & | & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ b_N & b_N & b_N & b_N & b_N & b_N & b_N & | & b_N & b_N & b_N & b_N & b_{N-1} & \cdots & b_1 \end{bmatrix} \begin{pmatrix} \Delta m_{-N+1} \\ \Delta m_{-N+2} \\ \Delta m_{-N+3} \\ \Delta m_{-N+4} \\ \vdots \\ \Delta m_0 \\ \hline \Delta m_1 \\ \Delta m_2 \\ \Delta m_3 \\ \vdots \\ \Delta m_M \\ \vdots \\ \Delta m_N \end{pmatrix} \quad (3.2a)$$

Simplifying for notational purposes

$$\begin{bmatrix} \bar{x}_{0Pred} \\ \bar{x}_{Pred} \end{bmatrix} = \begin{bmatrix} \bar{B}_o & \bar{0} \\ \bar{B}_{ol} & \bar{B} \end{bmatrix} \begin{bmatrix} \Delta \bar{m}_{past} \\ \Delta \bar{m} \end{bmatrix} \quad (3.2b)$$

In the dynamic matrix, matrices \bar{B}_o , \bar{B}_{ol} , and \bar{B} are clearly top-left, bottom-left and bottom-right respectively. In DMC on each time step a limited sequence of M moves ($M \leq N$ see equation 3.2), referred to as the control horizon, is generally solved for. Normally only one or two future moves are solved ($M=1$ or $M=2$), allowing a reduction in the size of \bar{B} . Although only the first move is ever used, a solution for a second move allows stronger action on the first move, because the solution plans a correction with the second move. Using an extended control horizon has the effect that the controller is able to plan a sequence of moves extending into the future. For a system that reaches steady state, b_N in matrix equation 3.2 can be repeated as in the final row of the dynamic matrix.

From the matrix multiplication 3.2b:

$\bar{x}_{0Pred} = \bar{B}_o \Delta \bar{m}_{past}$ - Generates N identical predicted copies of the output vector at the present time $t = 0$. This results in a reduction in computation as only $B_{ol}-B_o$ is stored (see equation 3.3).

$\bar{x}_{Pred} = \bar{B}_{ol} \Delta \bar{m}_{past} + \bar{B} \Delta \bar{m}$ - Generates a vector of predictions at N points on the future trajectory, as contributed by the past N control moves and future M control moves. The first part of the sum is the open loop response (response with no future control changes), while the second part accounts for future changes.

For step responses reaching a steady state at step N , if no further moves were to take place, moves up to N intervals ago will cause variations in the future outputs, while the moves prior to that will contribute steady state offsets. Variations in the future states caused by past moves are accounted for by using the openloop response ($\bar{B}_{ol} \Delta \bar{m}_{past}$). In the face of unmeasured disturbances and model system mismatch, some form of feed back is required to remove the steady state offset. The most common method of incorporating feedback involves comparing the measured \bar{x}_{0Meas} and predicted $\bar{B}_o \Delta \bar{m}_{past}$ process outputs states. The corrected openloop response is thus:

$$\begin{aligned} \bar{x}_{ol} &= \bar{x}_{olMeas} - \bar{B}_o \Delta \bar{m}_{past} + \bar{B}_{ol} \Delta \bar{m}_{past} \\ \Rightarrow \bar{x}_{ol} &= \bar{x}_{olMeas} + [\bar{B}_{ol} - \bar{B}_o] \Delta \bar{m}_{past} \end{aligned} \quad (3.3)$$

Now accounting for future moves results in the closed loop response being:

$$\begin{aligned} \bar{x}_{cl} &= \bar{x}_{ol} + \bar{B} \Delta \bar{m} \\ \Rightarrow \bar{x}_{cl} &= \bar{x}_{olMeas} + [\bar{B}_{ol} - \bar{B}_o] \Delta \bar{m}_{past} + \bar{B} \Delta \bar{m} \end{aligned} \quad (3.4)$$

This model (equation 3.4) can now be used to calculate a sequence of future control moves $\Delta \bar{m}$ that will satisfy some specified objective such as:

- Minimising the predicted deviation of the process output from setpoint \bar{x}_{sp} over the prediction horizon.
- Minimising the expenditure of control effort $\Delta\bar{m}$ in driving the process output to setpoint.

In completing the mathematical formulation, the above objectives can be expressed as a performance index:

$$J(\Delta\bar{m}) = (\bar{x}_{sp} - \bar{x}_{cl})^T \bar{W} (\bar{x}_{sp} - \bar{x}_{cl}) + \Delta\bar{m}^T \bar{\Lambda} \Delta\bar{m}$$

$$\Rightarrow J(\Delta\bar{m}) = (\bar{e}_{cl})^T \bar{W} (\bar{e}_{cl}) + \Delta\bar{m}^T \bar{\Lambda} \Delta\bar{m} \quad (3.5)$$

Where \bar{e}_{cl} is the closed loop error (difference between closed loop response and setpoint over the optimisation horizon). \bar{W} (positive definite weighting factor matrix) and $\bar{\Lambda}$ (positive definite move suppression matrix), both generally diagonal, are tuning parameters, that determine the extent to which deviations from setpoint and control moves are discouraged.

Graphically, as discussed in 3.1 under model predictive control, the DMC algorithm, with relevant equations, can be depicted by the following figure:

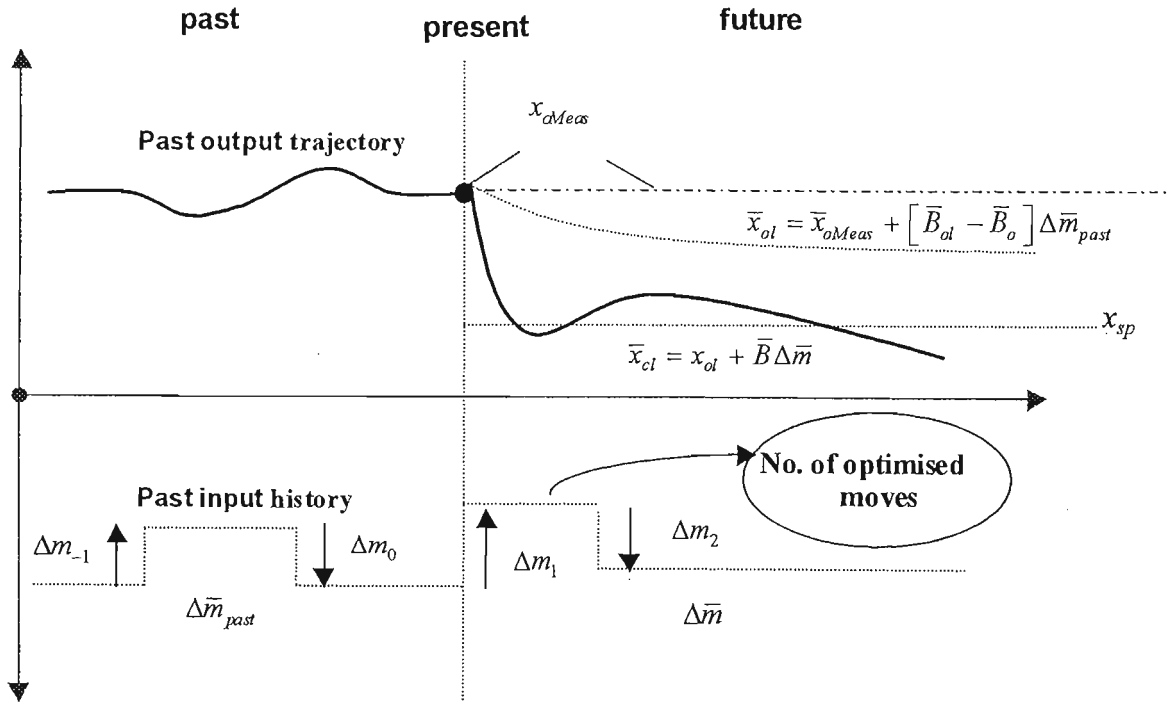


Figure 3.5 : Optimisation horizon for DMC

Thus by minimising J up to an optimisation horizon with respect to $\Delta\bar{m}$ in equation 3.5, an optimal sequence of control moves ($\Delta\bar{m}$) is found that results in minimal deviation of the state trajectory (x_{cl}) from setpoint (x_{sp}). Due to the linear model and a quadratic objective, the DMC algorithm takes the form of a structured convex Quadratic program (QP). This optimisation problem can be solved in several ways. Guiamba (2001) and Mulholland and Prosser (1999) used a linear programming

approach, whilst Cutler and Ramaker (1980) used a quadratic programming technique. Most of the software packages and solvers discussed in the following chapter are able to solve directly, the quadratic programming DMC formulation. Effective solvers and software packages are important, because the solution algorithm must converge reliably to the optimum in no more than a few seconds to be useful in real time control. In principle, the MPC method is therefore limited to those problems, for which a global optimal solution to the dynamic optimisation, can be found between one control execution and the next.

3.2.3 DMC and Integrating Systems

Traditional DMC, as presented in the previous sub-section, is based on the assumption that systems reach a steady state after being disturbed. However in many processing units, due to material or energy imbalance, this is not the case. Integrating behaviour is common in hybrid systems, where a ramp change in output is the result of a step change in input, or a decision being made.

Guiamba (2001) showed how the DMC framework could be expanded to include integrating systems. Using equation 3.2 as a basis, an integrating relationship can be represented by unequal corresponding elements in the final two matrices, b_N and b_{N-1} . It is assumed that this final gradient continues indefinitely from this point onwards as a result of the integration. Defining $\Delta b = b_N - b_{N-1}$ matrix equation 3.2 for integrating systems now becomes:

$$\begin{pmatrix} x_{o, pred} \\ x_{o, pred} \\ x_{o, pred} \\ x_{o, pred} \\ \vdots \\ x_{o, pred} \\ x_1 \\ x_2 \\ \vdots \\ x_M \\ x_{M+1} \\ \vdots \\ x_N \end{pmatrix} = \begin{bmatrix} 0 & b_N & b_{N-1} & b_{N-2} & b_{N-3} & \cdots & b_2 & b_1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & b_N & b_{N-1} & b_{N-2} & b_{N-3} & \cdots & b_2 & b_1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & b_N & b_{N-1} & b_{N-2} & b_{N-3} & \cdots & b_2 & b_1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & b_N & b_{N-1} & b_{N-2} & b_{N-3} & \cdots & b_2 & b_1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & 0 & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & b_N & b_{N-1} & b_{N-2} & b_{N-3} & \cdots & b_2 & b_1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \hline \Delta b & b_N + \Delta b & b_N & b_{N-1} & b_{N-2} & \cdots & b_3 & b_2 & b_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2\Delta b & b_N + 2\Delta b & b_N + \Delta b & b_N & b_{N-1} & \cdots & b_4 & b_3 & b_2 & b_1 & 0 & 0 & 0 & 0 & 0 \\ 3\Delta b & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & b_N & b_{N-1} & b_{N-2} & b_{N-3} & \cdots & b_1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & b_N + \Delta b & b_N & b_{N-1} & b_{N-2} & \cdots & b_1 & \cdots & \cdots & 0 \\ (P-1)\Delta b & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ P\Delta b & b_N + P\Delta b & \cdots & \cdots & \cdots & \cdots & b_N + 2\Delta b & b_N + \Delta b & b_N & b_N & b_N & b_N & b_{N-1} & \cdots & b_1 \end{bmatrix} \begin{pmatrix} \sum_{i=-\infty}^{-N} \Delta m_i \\ \Delta m_{-N+1} \\ \Delta m_{-N+2} \\ \Delta m_{-N+3} \\ \Delta m_{-N+4} \\ \vdots \\ \Delta m_0 \\ \Delta m_1 \\ \Delta m_2 \\ \Delta m_3 \\ \vdots \\ \vdots \\ \vdots \\ \Delta m_M \end{pmatrix} \quad (3.6)$$

There are two major changes in matrix equation 3.6 as compared to matrix equation 3.2. Firstly the introduction of a new column before the B_{ol} (lower left) matrix. Elements from this new column multiply, at every time step, with the summation term, $(\sum \Delta m_i)$ which accumulates all moves older than N steps back from the present time. Secondly is the modification of the \bar{B}_{ol} matrix by adding multiples of $\Delta \bar{b}$ further back than the \bar{b}_n term, as shown in equation 3.6. Each move from the past therefore contributes its own ramp effect in the future. Besides the above-mentioned changes, the

DMC algorithm (closed loop equation 3.4 and objective function 3.5) remains the same as for systems that attain steady states when disturbed.

3.2.4 DMC and Hybrid Systems

All accounts located in the literature that incorporate DMC are continuous in nature. This incorporates both input variables, i.e. those system variables that are stepped in order to generate the step response data, and outputs i.e. those variables that are controlled. DMC has been developed such that the optimisation procedure results in a solution of variables required by the system for optimal operation. In generating the data for the step responses, it is these input variables that are stepped. Equation 3.3, contains these input variables in the vector $\Delta \bar{m}$. Notice that it is the change in input Δm from its last position that is the result of the optimisation procedure. This is because the convolution model is based on a step response for a unit change in m .

Extending DMC to control hybrid systems, requires adapting the algorithm to accommodate binary/integer variables. In doing this, the DMC algorithm effectively becomes a MIDO problem, as their mathematical structures become identical. For the control of hybrid systems, the controller, rather than just evaluating continuous optimum inputs to a system, also has to make certain discrete decisions which affect the process e.g. selections or initiations of events. This effectively amounts to solving for binary/integer variables e.g. in the case of binary variables, the optimisation procedure will determine the value (0 or 1) that minimises the objective function.

In being consistent with the DMC structure, the binary/integer variables (indicating decisions, selections or switches) will have to be part of the vector $\Delta \bar{m}$ in equation 3.4. To generate the necessary step response data, each of these decisions or selections will have to be carried out in turn in openloop, as a switch or change procedure from a base operation, and the relevant responses captured. Using the step response as a basis, the controller will determine whether to implement a change or not by optimally evaluating the appropriate binary variable.

For example, a binary variable can be assigned to a valve that is either open or shut. “1” will represent the open position of the valve, while “0” will be the shut position. To generate a step response for the convolution model, the valve must be stepped from shut to open and the response of the system recorded. In the optimisation procedure, the algorithm will determine, using the convolution model derived from the step response, the optimum sequence of the binary variable (0 or 1) that minimises the objective function. Regarding the actual plant, the solution amounts to determining whether having the valve open or closed, best satisfies the dynamic performance of the system.

Binary variables can also be used to determine the optimum time to switch between certain modes of operation for a system consisting of multiple operating modes. Each mode can be assigned a binary variable and step response data is generated by switching from one mode to another. Constraint optimisation can be used to force the algorithm to set only one binary variable to “1”. With respect to

the system, this amounts to the controller selecting the appropriate mode, in real time, in order to meet process objectives.

Mixed integer predictive control can also be used on some processes that have a number of units, whose operation is cyclic, operating in parallel. It may be required that for some reason (economic or capacity considerations) to sequence or stagger the cycles of each unit relative to each other. In this case a binary variable can be assigned to each unit. During a constraint optimisation procedure, the DMC algorithm can be used to determine a real time sequencing strategy by optimally evaluating the binary variables. The necessary convolution model is generated by using responses that depict the actual cycle.

DMC is therefore capable of regulating hybrid systems that have discontinuities in their inputs whether they are direct plant inputs or just decisions. If it is possible to obtain a continuous step response for discrete inputs, then it will be possible to use DMC to stabilise and optimise the operation of that system. However there is another class of hybrid system that DMC cannot deal with. These systems have discontinuities in their outputs e.g. a tank with an overflow as presented in chapter two, section 2.4.4. This effectively means that the operation of the system is divided into distinct regimes, as defined by the state of the system. To model such a system, each regime is usually assigned its own process model. DMC is unable to deal with these types of system because the convolution model relies on past inputs ($\Delta \bar{m}_{past}$ in equation 3.3). When a system enters a new regime of operation, information from the past cannot be used, because it pertains to a different model. Inputs from a different regime cannot be used to determine the present state (x_o) and openloop trajectory \bar{x}_{ol} in a new regime. Chapter 8 considers this problem in greater detail and proposes the use of state space equations in this instance.

3.3 Control Loop

Figure 3.6 shows the integration of the DMC structure as a model predictive controller with a plant. The controller supplies the plant with optimum inputs, at fixed time intervals, in terms of its change from its previous value. This optimum input, for the next time step, is also used in an internal process model, together with previous inputs, to evaluate the current state and openloop trajectory of the system. An openloop error is calculated by comparing the openloop trajectory with the desired setpoint, and based on the convolution model, the optimum input Δm for the present time step, is evaluated through an optimisation procedure.

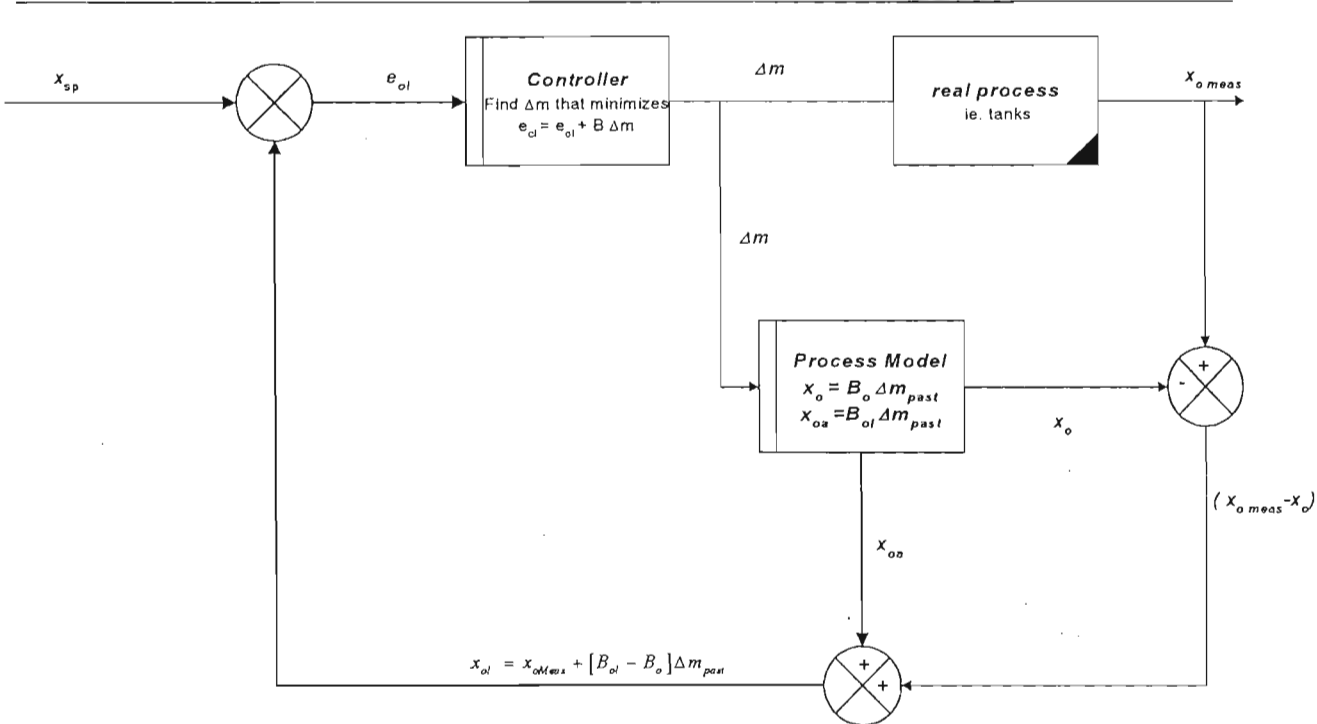


Figure 3.6: DMC in model predictive control configuration

To validate the proposed mixed integer model predictive control (MIPC) algorithm, based on the DMC framework, a control loop similar to the above diagram had to be commissioned. Since the DMC structure was written in GAMS, where the mixed integer optimisation was also done, this task essentially involved the integration of different software packages as shown in figure 3.7 and 3.8.

After obtaining the step response data, MATLAB is used to arrange it in convolution model form. The reason for using MATLAB is because the size of vectors and matrices can easily be varied based on problem size. This cannot be done in GAMS. Using the print function, MATLAB once compiled, writes out the entire control algorithm in GAMS. Refer to appendix A2, for a flowchart of this procedure in MATLAB, where the control algorithm is spawned into GAMS.

An in house developed SCADA (sequential control and data acquisition written in C++) package referred to as SCAD, interfaces with the plant or model. After receiving inputs necessary for the controller from the process, SCAD calls GAMS hence executing the control algorithm. After completing the optimisation, the optimum process inputs are written to a text file, from which SCAD collects them and passes them on to the process.

Figure 3.8 shows sequentially how program CEXTobject interfaces GAMS and SCAD to form a control loop in real time. Refer to Appendix A1 for the code written in C++(with acknowledgements to T.Brazier and M.Mulholland).

In the following chapters, using the integrated software structure shown in the figure 3.7, the validation of the MIPC is done on different types of hybrid systems. Tests are either done online with

an actual plant or offline, with the same SCADA framework, against a plant model also derived from step response data.

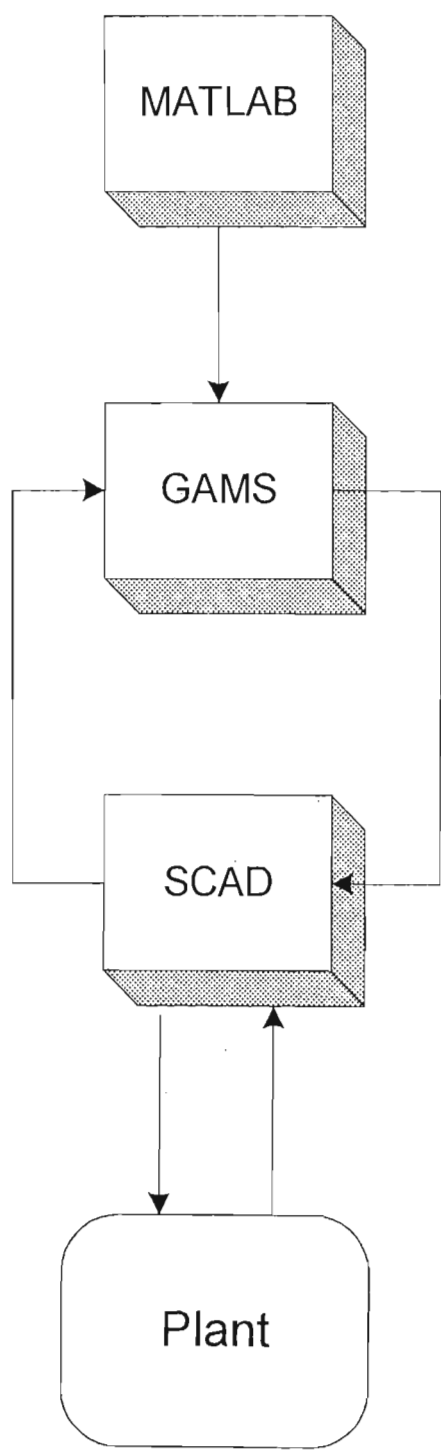


Figure 3.7: Integration of software packages for closed loop testing of MIPC

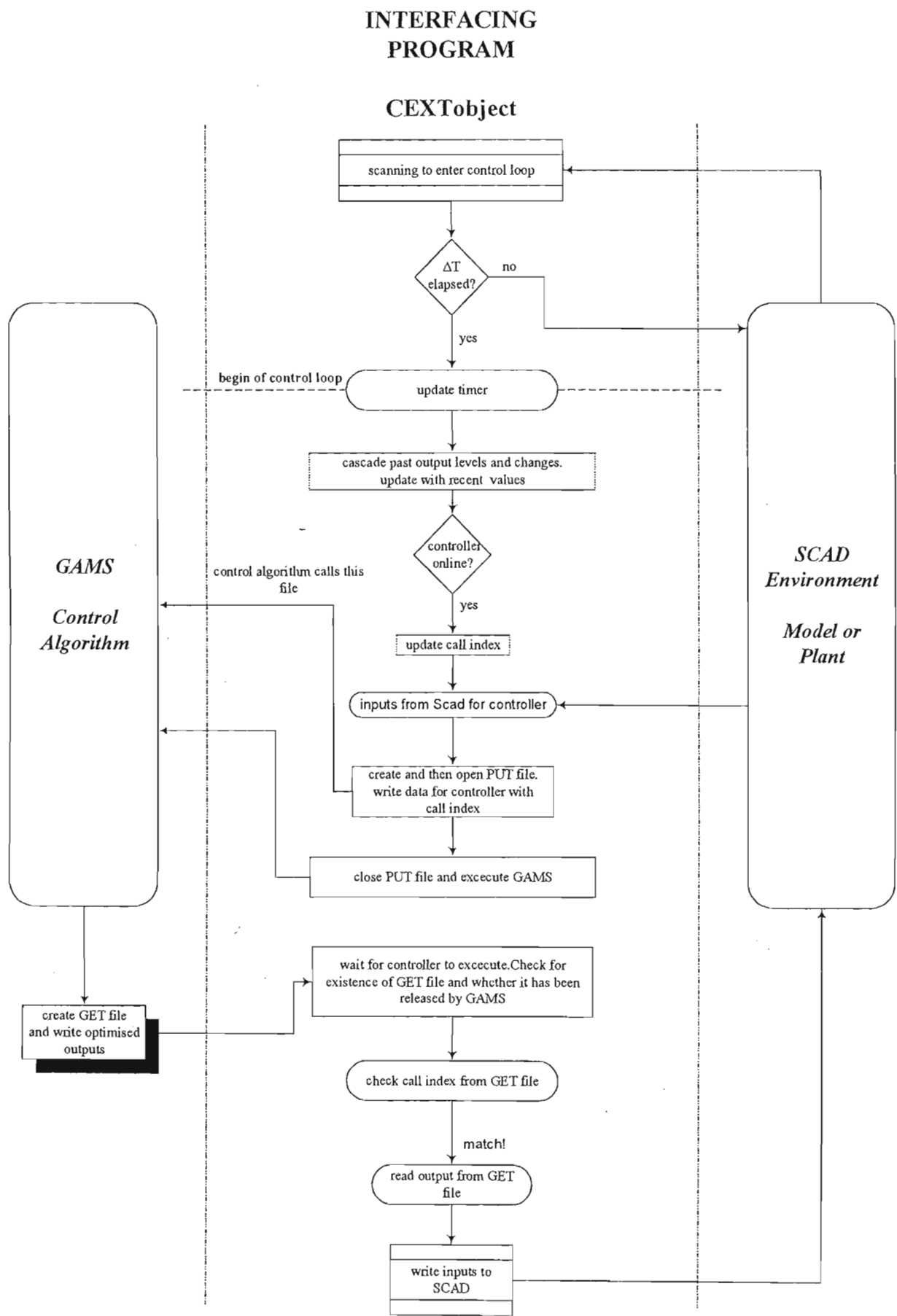


Figure 3.8 Interfacing between GAMS and SCAD with file CEXtObject

CHAPTER 4

Mixed Integer Dynamic Optimisation

The previous chapter proposed MPC as a strategy for the control of hybrid systems. MPC continuously optimises the closed loop dynamic performance of a system while taking into account disturbances, constraints and targets. This chapter deals with the computational aspect of the dynamic optimisation procedure. It starts with an introduction to mixed integer dynamic optimisation (MIDO). Thereafter section 4.2 shows how logic can be converted into general mathematical representations necessary for the optimisation procedure. Section 4.3 deals with the numerical solution of MIDO problems by proposing various solvers and software packages capable of dealing with the mixed integer nature of the problem. This section also introduces GAMS, the optimisation package used in this work.

4.1 Introduction

Many problems in process design and operation require the optimal solution of quantities that change in time. When a mathematical model of the process is available, then these quantities can be evaluated using dynamic optimisation. In process control, MPC is a problem in dynamic optimisation. The controller prompts an optimal change in the present based on a future prediction of the system.

When integer variables are part of the optimisation procedure, then the optimisation is known as a mixed integer dynamic optimisation (MIDO) problem. Consequently the predictive control of hybrid systems, where there is an interaction between continuous and integer/binary variables, is a problem in MIDO. Over the past 5 years, some preliminary research on MIDO formulations and solution algorithms have appeared in the literature. MIDO is one of three classes of dynamic optimisation problems with discontinuities discussed by Barton *et al.* (1998). The other two were path-constrained problems and hybrid discrete/continuous problems. Allgor and Barton (1997) considered the class of MIDO problems that conform to the following general formulation:

$$\min_{u(t), v, y, t_f} \left\{ \phi(x(t_f), u(t_f), y(t_f), v, t_f) + \int_0^{t_f} L(x(t), u(t), y(t), v, t) dt \right\} \quad (4.1)$$

subject to

$$f(x(t), \dot{x}(t), u(t), v, y, t) = 0 \quad t \in [0, t_f] \quad (4.2)$$

$$g(x(t), \dot{x}(t), u(t), v, y, t) \leq 0 \quad t \in [0, t_f] \quad (4.3)$$

$$k_p(x(t_p), \dot{x}(t_p), u(t_p), v, y, t_p) \leq 0 \quad p \in \{0, \dots, n_p\} \quad (4.4)$$

Here $x(t)$ are the continuous variables describing the state of the dynamic system, $u(t)$ are the continuous controls whose optimal time variations on the interval $[t_o, t_f]$ are required, v are continuous time invariant or variant parameters whose optimal values are required, y are a special set of time invariant or variant parameters that can take on only discrete values, and t_f is a special continuous time invariant parameter known as the final time. t_f defines the horizon over which the optimisation extends. It is the presence of the integer parameters y that distinguishes the above formulation from other general dynamic optimisation formulations.

Equation (4.1) is the objective function that, in being minimised, drives the optimisation procedure. It comprises in general two parts, the first depending on variables at the end time (end point conditions) and the second depending on the change in variables as end time is approached, hence the integral. Equation (4.2) represents a general set of differential algebraic equations describing the dynamics of the system. Typically they will include a lumped dynamic model of the system in question, coupled with any path equality constraints that the system must satisfy. Inequalities (4.3) represent a general set of path inequality constraints that must be satisfied by the solution of the optimisation. The point constraints equation (4.4) encompass a special case of constraints that involve the integer variables y , such as logical relationships, that must be satisfied e.g. an item of equipment may only be used for certain processing tasks.

Mohideen *et al.* (1996a, b) use the above framework for obtaining process designs and control systems (also considered as part of the optimisation problem), which are economically optimal and are able to cope with process variations. Process systems were modelled via dynamic mathematical models, while variations included both uncertain parameters and time varying disturbances.

With the MIDO structure comes the requirement for its numerical solution. All currently reported techniques for the numerical solution of MIDO problems rely on extensions of existing decomposition approaches for solving mixed-integer non-linear programming (MINLP) problems (see section 4.3). In the context of integration of design and control, Mohideen *et al.* (1996a, b and 1997) outline an efficient decomposition algorithm for the numerical evaluation of the MIDO problem. Allgor and Barton (1997) also presented another rigorous decomposition approach for MIDO problems that is based in the context of optimal batch process development.

4.2 Linear Integer Programming

Given the MIDO formulation, the challenge remains to convert the physical hybrid problem at hand, into mathematical representations. Because the focus is on systems that have logic and dynamics, there is a need to establish a link between the two. The first step is to obtain a model of the system that reasonably defines the dynamics of the system together with the interactions between inputs and outputs (equation 4.2). The model can for example be in state space form (finite difference equations) as used by Morari and his colleagues for MLD systems (Bemporad and Morari, 1998; Bemporad *et al.*, 2000; Borrelli *et al.*, 2001; Torrisi *et al.*, 2001) or convolution form (generated from step responses)

as in this thesis. Thereafter the need arises to build logical statements from operating events concerning physical relations. The key idea is to use mixed-integer linear inequalities i.e. linear inequalities involving both continuous and binary variables, to express the logic of the system mathematically (equation 4.4).

The following table (taken from Mignone *et al.*, 1999) shows the basic conversions of logic relations into mixed integer (in)equalities. Here logic relationships (AND, Or, Not, IMPLY, IFF) between the binary variables ($\delta_1, \delta_2, \delta_3$) are converted into equivalent mathematical relations which can then be included as part of an algorithm.

Proposition	Relation	Logic	Mixed integer (in)equalities
P1	And (\wedge)	$[\delta_1=1] \wedge [\delta_2=1]$	$\delta_1=1$ $\delta_2=1$
P2		$[\delta_3=1] \leftrightarrow [\delta_1=1] \wedge [\delta_2=1]$	$-\delta_1 + \delta_3 \leq 0$ $-\delta_2 + \delta_3 \leq 0$ $\delta_1 + \delta_2 - \delta_3 \leq 0$
P3	Or (\vee)	$[\delta_1=1] \vee [\delta_2=1]$	$\delta_1 + \delta_2 \geq 1$
P4		$[\delta_3=1] \leftrightarrow [\delta_1=1] \vee [\delta_2=1]$	$\delta_1 - \delta_3 \leq 0$ $\delta_2 - \delta_3 \leq 0$ $-\delta_1 - \delta_2 + \delta_3 \leq 0$
P5	Not(\sim)	$\sim[\delta_1=1]$	$\delta_1=0$
P6	XOR (\oplus)	$[\delta_1=1] \oplus [\delta_2=1]$	$\delta_1 + \delta_2 = 1$
P7		$[\delta_3=1] \leftrightarrow [\delta_1=1] \oplus [\delta_2=1]$	$-\delta_1 - \delta_2 + \delta_3 \leq 0$ $-\delta_1 + \delta_2 - \delta_3 \leq 0$ $\delta_1 - \delta_2 - \delta_3 \leq 0$ $\delta_1 + \delta_2 + \delta_3 \leq 0$
P8	IMPLY(\rightarrow)	$[\delta_1=1] \rightarrow [\delta_2=1]$	$\delta_1 - \delta_2 \leq 0$
P9		$[f(x) \leq 0] \rightarrow [\delta=1]$	$f(x) \geq \varepsilon + (m-\varepsilon)\delta$
P10		$[\delta=1] \rightarrow [f(x) \leq 0]$	$f(x) \leq M - M\delta$
P11	IFF(\leftrightarrow)	$[\delta_1=1] \leftrightarrow [\delta_2=1]$	$\delta_1 - \delta_2 = 0$
P12		$[f(x) \leq 0] \leftrightarrow [\delta=1]$	$f(x) \leq M - M\delta$ $f(x) \geq \varepsilon + (m-\varepsilon)\delta$

Table 4.1 : Conversion of logic into integer inequalities

M and m are the upper bound and lower bound of $f(x)$ respectively
 ε is a small tolerance beyond which the constraint is regarded as violated.

More complex logical operational procedures can be expressed by using combinations of the above rules. Raman and Grossmann (1991,1992) show that for process synthesis, logic and heuristics can also be integrated through proportional logic. This type of qualitative knowledge is useful for two reasons. Firstly, in many cases solutions that reflect the operator’s experience are preferred. Secondly, heuristic rules may aid in the search for feasible solutions.

An example showing the application of the above rules in converting process logic into linear equalities is in a multiple unit separation plant.

Logic: if an absorber or membrane separator is selected to recover a product, do not use cryogenic separation.

Introduce three binary variables (y_a , y_m , y_c) one for each unit. “1” for a specific variable implies selection of that particular unit.

y_a = binary variable for selecting absorber.

y_m = binary variable for selecting membrane.

y_c = binary variable for selecting cryogenic separation.

Linear inequalities

$$y_a + y_c \leq 1$$

$$y_m + y_c \leq 1$$

note that either $y_a = 1$ or $y_m = 1$ forces $y_c = 0$ or in Boolean algebra $y_c = \sim (y_a \vee y_m)$

Expressing the logic of the system as mixed integer linear inequalities is fundamental to the MIDO composition. It ensures that the solutions from the optimisation are in accordance with the actual system. The structuring of these linear inequalities is also important as they are usually integrated with the system model and the objective function in the MIDO formulation.

4.3 Computational Aspects

4.3.1 Mixed Integer Non-linear Programming

In an optimisation procedure, if the mathematical structure comprises integer variables coupled with continuous variables, then the problem is referred to as a *mixed integer program* (MIP). If the equations and objective function are linear, then the problem is a *mixed integer linear program* (MILP). On the other hand, if any of the equations and/or the objective function is non-linear, then the resulting structure is referred to as a *mixed integer non-linear program* (MINLP). There exists a special case of MINLP, known as *mixed integer quadratic program* (MIQP), where the equations and constraints are linear, but the objective function is quadratic. The MLD framework, developed by the researchers at the Automatic Control Laboratory, is a MIQP.

The numerical optimisation of discontinuous systems has motivated a need for procedures that solve algorithms comprising integer and continuous variables. All currently reported techniques for the numerical solution of MIDO problems rely on extensions of approaches used for solving MINLP's (Allgor and Barton, 1997). MIDO is after all only a MINLP problem with a dynamic aspect.

It is only recently (last 15 years) that breakthroughs have been made regarding algorithms capable of solving MIP's. The reason for the delay in progress is that MIP problems involving 0-1 variables are N-P complete, meaning that complexity and solution effort grows exponentially with the number of integer variables. However due to the work done by amongst others, Viswanathan, Grossmann and Floudas, efficient algorithms for solving mixed integer structures are now available. Major algorithms for solving the MINLP problems include: branch and bound (B&B), generalised Benders decomposition (GBD) and outer approximation (OA). In the engineering domain, much success has been reported in the last few years, where numerous problems were formulated as MINLP's and solved using these techniques.

B&B does not define a particular algorithm, but rather a whole class of methods that differ in their details of implementation. Common to all B&B methods for mixed integer non-linear programming problems is the generation of a set of easier sub-problems arranged in a tree.

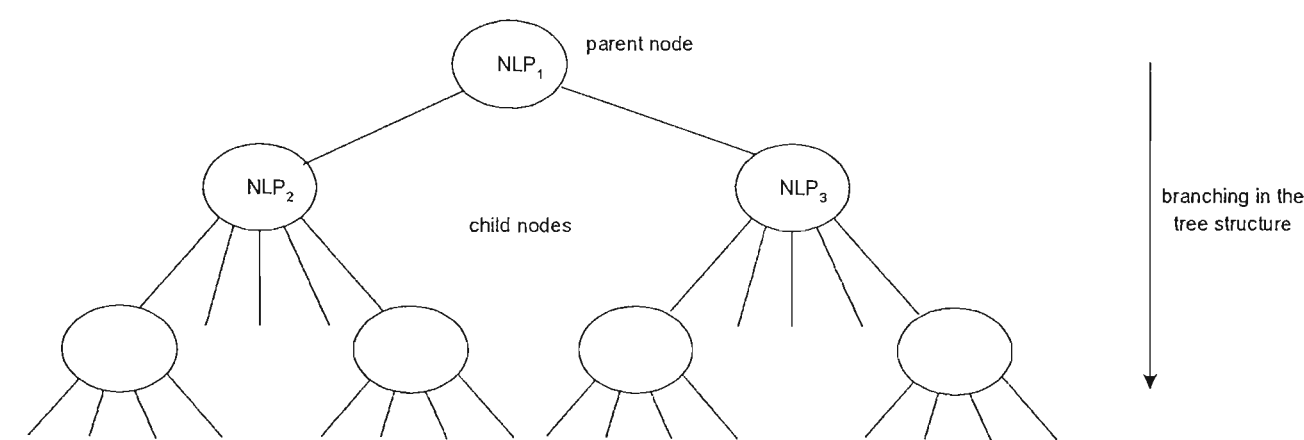


Figure 4.1: Tree structure for branch and bound procedure

The idea of solving MINLP's with B&B methods relies on the relaxation of the integer constraints i.e. binary variables are allowed to span over the entire continuous interval $[0,1]$. The relaxed problem is referred to as a sub-problem. The B&B algorithm for MINLP consists of solving and generating new NLP problems in accordance with a tree search (figure 4.1), where the NLP sub-problems correspond to nodes of the tree. Branching is obtained by generating child-nodes from parent nodes according to branching rules (depth first or breadth first). Optimal values of the NLP subproblem, if they exist, represent bounds on the optimal value of the original problem. The solution of the corresponding NLP at each node provides a lower bound for the optimal MINLP objective function. The lower bound is used to direct the search, either by expanding the nodes in a breadth first or depth first fashion. The depth first approach performs branching on the most recently created node, while the breadth first approach selects the node with the best value at each level and expands all its successor nodes. Nodes are labeled as either pending, if the corresponding NLP problem has not yet been solved, or fathomed, if the node has already been fully explored. The algorithm stops when all nodes have been fathomed.

The success of the B&B algorithm relies on the fact that whole subtrees can be excluded from further exploration by fathoming the corresponding parent nodes. This happens if the corresponding NLP subproblem is either infeasible or an integer solution is obtained. In the second case, the corresponding value of the cost function serves as an upper bound on the optimal solution of the MINLP problem, and is used to further fathom other nodes having a better optimal value or lower bound. The major disadvantage of the B&B method is that, depending upon the number of integer variables, it requires the solution of a large number of NLP sub-problems.

Generalised benders (GBD) decomposition and outer approximation (OA) solve the MINLP by an iterative process. The problem is decomposed into a NLP subproblem, which has integer values fixed, and an MILP master problem. The NLP subproblems optimise the continuous variables and provide an upper bound to the MINLP solution, while the MILP master problems have the purpose of predicting a new lower bound for the MINLP solution, together with new integer variables for each iteration. The search terminates when the predicted lower bound equals or exceeds the current upper bound. The difference between GBD and OA is in the definition of the MILP master problem.

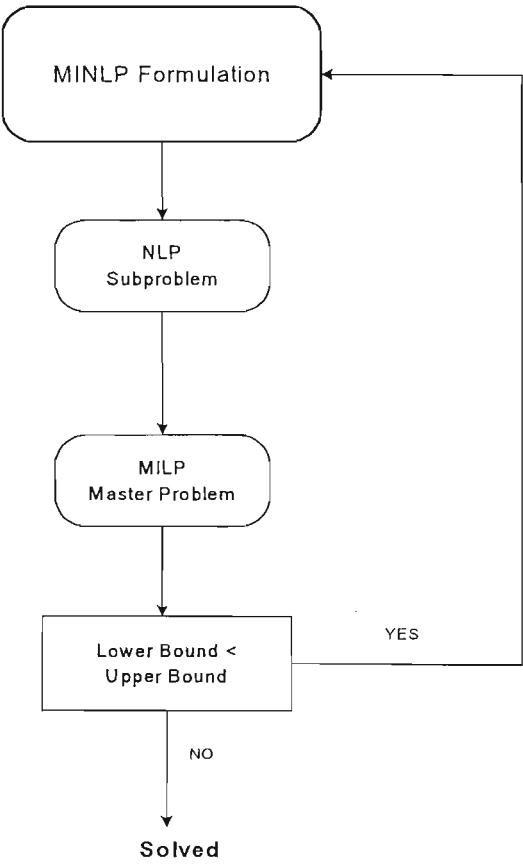


Figure 4.2 :GBD and OA solution procedures

GBD has the advantage that one can exploit more readily special structures in the NLP subproblems, but has the drawback that it may require a significant number of major iterations where the NLP and MILP master problems must be solved successively. The OA algorithm has the advantage that it

typically requires only a few major iterations, but the size of the MILP master problem is considerably larger than in GBD. For more detailed information regarding these methods used for the solution of MINLPs refer to the book written by Floudas (1995) *Nonlinear and Mixed-Integer Optimisation*.

4.3.2 Software

The previously mentioned algorithms for solving MINLP problems are available as portable code, in the form of software packages. These packages are either available as add-on solvers or stand-alone modelling systems. A solver is concerned primarily with solving the MINLP problem. It has to be interfaced with the problem at hand e.g. a model developed in an external general purpose programming language like C++. A translator is therefore required that reads the model and data expressed in the modelling language and builds a low-level representation (such as a list of coefficients) that the solver requires for its operation. On the other hand, stand-alone systems integrate the solver and modelling language software as a single unit. These units usually comprise several solvers, thus allowing the user to select from a list. By taking care of time consuming details like: interfacing between model and solver; machine specifications; system software implementation and compatibility, stand-alone modelling systems allow the user to concentrate on the modelling aspect of the problem.

Software is either available commercially i.e. it is licensed and supported for a fee, or noncommercially where it can be freely downloaded from the internet. Below are examples of solvers and modelling packages that have been associated with solving MINLP problems.

Solvers :

*DICOPT*¹: Viswanathan and Grossmann (1990) developed this DIcrete Continuous OPTimiser (DICOPT). Since then DICOPT++ has been released. It bases its solution of MINLP's on an extension of the OA algorithm.

*CPLEX*²: is arguably the most widely used solver in academic research throughout the world. CPLEX solves linear and mixed integer problems. Solution of the mixed integer programming option uses a variety of branch and bound techniques. CPLEX is also portable and interfaces easily with modelling languages e.g. GAMS and AMPL.

MIQP: is a m-file written by the researchers at the control laboratory at the Swiss Federal Institute of Technology (ETH) Zurich for solving MIQP problems (linear equations and constraints with quadratic objective function). MIQP is applicable as a function only in the MATLAB environment. This code is freely available for downloading at www.control.ee.ethz.ch/~hybrid/miqp/.

¹ <http://egon.cheme.cmu.edu/Group/ResearchAreas.html#Mixed-integer%20optimisation>

² <http://www.ilog.com/products/cplex/>

*OSL2*³: is a high performance integrated solver for LP and MIP problems. It is based on IBM's Optimisation Subroutine Library (OSL), to solve optimisation problems. The simplex method is used to solve LP subproblems, while branch and bound is used for MIP.

*MINOS*⁴: solver developed at the systems optimisation laboratory at Stanford university. MINOS is used for solving mainly LP and NLP problems.

*BARON*⁵: is a computational system for solving optimisation problems that are purely continuous, purely integer and mixed integer nonlinear programs. This software can also be run online via the internet. BARON integrates a number of specialised solvers with an easy to use interface.

Modelling Packages:

*MINOPT*⁶: is a portable optimisation package developed in the CASL laboratory at the department of Chemical Engineering at Princeton University. It features an advanced modelling language, for clear and concise representation of complex mathematical models, together with a robust algorithmic framework, for the efficient solution of a wide variety of mathematical programming problems. MINOPT solves the following problems: LP; NLP; MILP; MINLP; dynamic simulations, MINLP with dynamic models (MIDO), optimal control problems (OCP); and mixed integer optimal control problems (MIOCP). Its solvers CPLEX, MINOS, and LPSOLVE use the GBD, OA and GCD (generalised cross decomposition) algorithms. LPSOLVE is a freely available LP/MILP solver that can be obtained at ftp://ftp.ics.ele.tue.nl/pub/lp_solve.

*GAMS*⁷: the General Algebraic Modelling System is designed for modelling linear, nonlinear, and mixed integer optimisation problems. GAMS is similar to MINOPT in its operation and is available for use on personal computers, workstations, mainframes and supercomputers. The software provides a computer interface with OSL, CPLEX, MINOS, CONOPT and DICOPT++. GAMS features CONOPT and CONOPT2 as alternative solvers to MINOS for NLP problems. The algorithms used in these solvers are different to those used in MINOS. The introduction of additional nonlinear solvers, to the proven MINOS, is an attempt to increase the overall usefulness of nonlinear modelling in GAMS. GAMS is the tool used in this thesis to model and optimise the operation of hybrid systems. Version 19.0 (and later version 20.1) was loaded onto a PC that thereafter formed part of a closed control loop. The following section describes the structure of GAMS programs and concludes with an example.

³ <http://www.research.ibm.com/osl/>

⁴ <http://www-neos.mcs.anl.gov/neos/solvers/NCO:MINOS-AMPL/>

⁵ <http://archimedes.scs.uiuc.edu/baron.html>

⁶ <http://titan.princeton.edu/MINOPT/minopt.html>

⁷ <http://www.gams.com>

4.3.3 GAMS

Diagram 4.3 shows the structure of the GAMS language and its components. Since GAMS is a modelling language, the programs must be written in the language that it is familiar with.

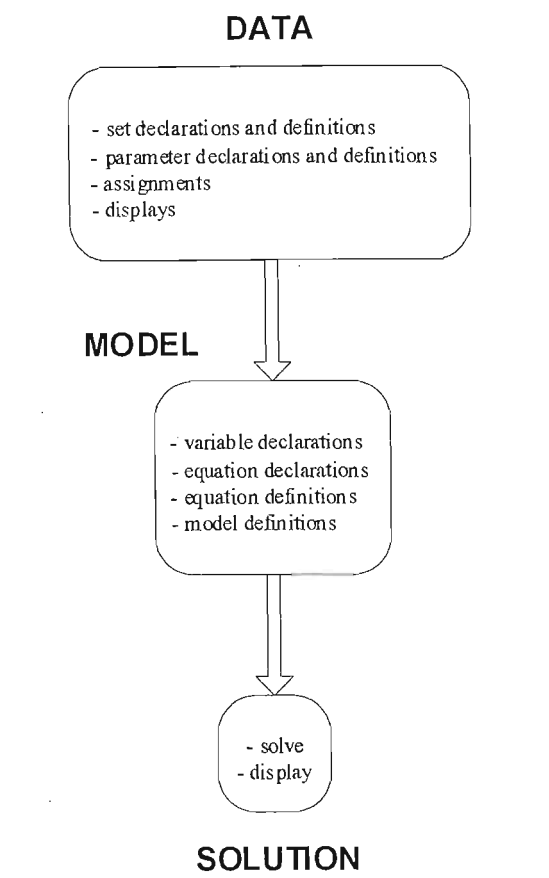


Figure 4.3: General structure of GAMS model

GAMS programs consist of statements that define data structures, initial values, data modifications and symbolic relationships (equations). While there is no fixed order in which statements are arranged, the order in which data modifications are carried out is important.

In figure 4.3 sets are placed first. Sets are the fundamental building blocks in any GAMS model. They allow for structuring and statement of the model. Then data is specified with parameter, scalar and table statements. This basic design paradigm of the GAMS language ensures the use of data in its most basic form. Next the model is defined with the variable, equation declaration, equation definition, and model statements.

The following features of the GAMS package make it a suitable basic building environment from which a MPC algorithm can be constructed. GAMS in its equation definitions, handles dynamic models involving time sequences, lags and leads and treatment of endpoint conditions. Whole sets of closely related constraints are entered in one statement. GAMS automatically generates each

constraint equation, and lets the user make exceptions in the cases where generality is not desired. The order of the equations in the model section is of no importance. The solution of the equations is not in any sequential or chronological order as is the case in other packages like MATLAB. GAMS solves the set of equations by satisfying all constraints (equality and inequality) simultaneously. This means that dependent variables need not be expressed explicitly i.e. on the left hand side of equations. In the solution of the problem, GAMS rearranges those equations with implicit variables.

Example

The following example, although not dynamic in nature, illustrates how GAMS can be used to optimise a system with mixed integer variables.

- The reaction $A \rightarrow B$ takes place in two reactors.
- Reactor 1: conversion 0.8 ; Cost of operation $5.5(\text{Feed})^{0.6}$ R/hr
- Reactor 2: conversion 0.667; Cost of operation $4(\text{Feed})^{0.6}$ R/hr
- x_o : feed A flowrate (kmol/hr); cost of feed 5 R/kmol
- Desired product, B, at 10 kmol/hr.

GAMS can be used to select the cheapest combination of the two reactors while fulfilling the product demand. Part (a) has both reactors operating separately. If reactor1 is in use, then reactor2 is not and visa versa. Configuration (b) has both reactors operating in parallel. The optimisation will determine if this is the optimum configuration in terms of cost, and if so, what are the splits to the respective reactors.

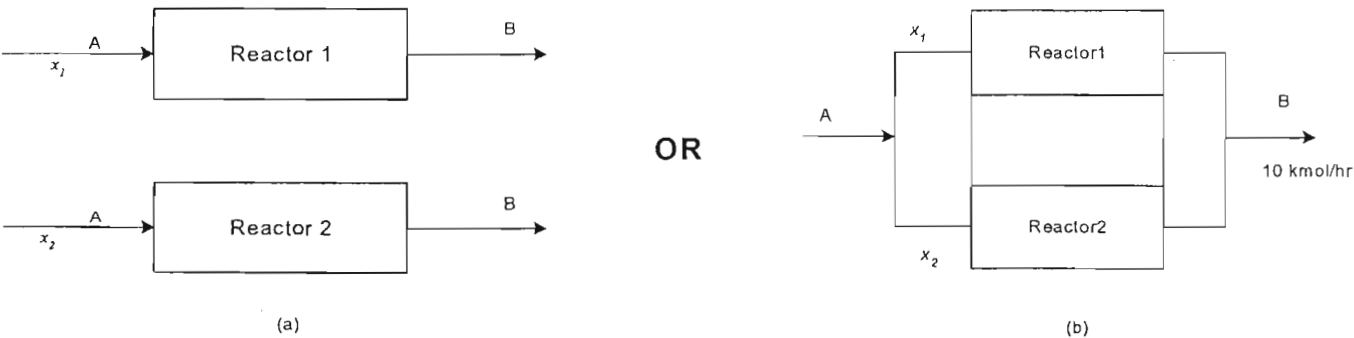


Figure 4.4: Reactor configuration selection

Using linear integer programming the problem can be converted into a MINLP, suitable for solution in GAMS.

Introduce two binary variables:

CASE 1

$$y_1 = \begin{cases} 1 & \text{if reactor1 is selected} \\ 0 & \text{otherwise} \end{cases}$$

CASE 2

$$y_2 = \begin{cases} 1 & \text{if reactor2 is selected} \\ 0 & \text{otherwise} \end{cases}$$

Now using the binary variables, define the following constraints to represent this problem mathematically

$$x_1 - m_1 y_1 \leq 0$$

$$x_2 - m_2 y_2 \leq 0$$

where m_1 and m_2 are the upper limits of x_1 (feed to reactor1) and x_2 (feed to reactor2) respectively and are evaluated by dividing the demand by the conversion of each individual reactor as follows:

$$m_1 = \frac{10}{0.8} = 12.5 \text{ kmol/hr}$$

$$m_2 = \frac{10}{0.667} = 15 \text{ kmol/hr}$$

thus the constraints now become:

$$x_1 - 12.5 y_1 \leq 0$$

$$x_2 - 15 y_2 \leq 0$$

$$x_1, x_2 \geq 0$$

The full MINLP formulation is thus:

Minimise $C = 5.5x_1^{0.6} + 4x_2^{0.6} + 5x_0$

Subject to $0.8x_1 + 0.667x_2 = 10$ (product demand)

$$x_1 - 12.5 y_1 \leq 0$$

$$x_2 - 15 y_2 \leq 0$$

$$x_o = x_1 + x_2 \quad (\text{mass balance for second configuration})$$

$$x_1, x_2 \geq 0$$

$$y_1, y_2 \in [0,1]$$

The cost can be expressed as a function of x_2 (feed to the second reactor), by eliminating x_1 using the product equality constraint in the objective function. Figure 4.5 shows the variation of cost (C) with feed to reactor2 (x_2).

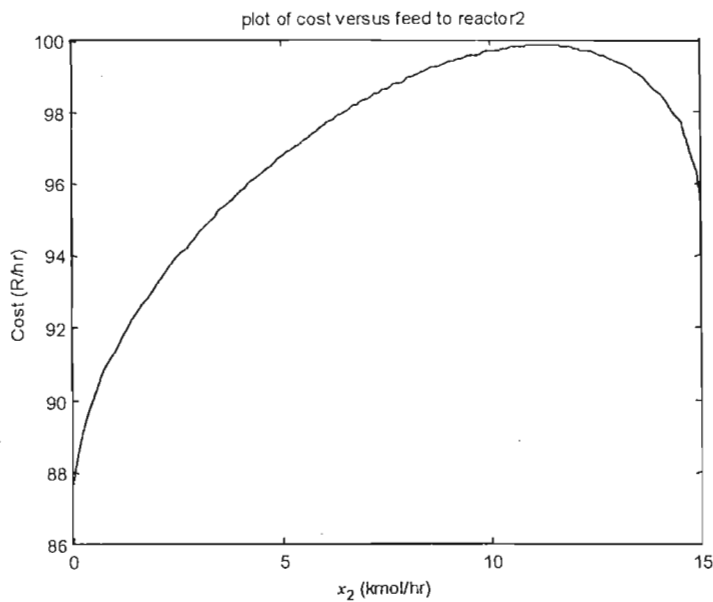


Figure 4.5 : Variation in cost with feed to reactor2

At $x_2=0$ a global optimum occurs corresponding to selecting reactor1 only. At $x_2=15$ there is a local optimum corresponding to selecting reactor2 only. The optimum selection should therefore be selecting only reactor1, which corresponds to a minimum cost of 87.5R/hr.

The option of selecting both reactors, while appropriately setting the feeds (x_1 and x_2) to meet the demands and constraints, will as shown from the plot, result in a higher cost (cost increases when feed to reactor2 is increased). The optimum configuration should thus be (a) where only reactor 1 is used.

The GAMS representation of this problem is shown by the code on the following page. GAMS solves the problem to the global minimum solution using DICOPT as the solver with MINOS solving the NLP subproblem. The solution is in agreement with the conclusions drawn from the plot. The optimum solution is with y_1 set to one (only reactor1 selected) and y_2 to zero. Correspondingly for optimality all the feed should be directed to reactor1 i.e. configuration (a). GAMS also calculates that the minimum cost of operation is 87.5R/hr.

```
binary variables
y1 selection of reactor1
y2 selection of reactor2;

positive variables
x1 feed to reactor1
x2 feed to reactor2;

variables
xo total feed
C cost;

equations
reactor1
reactor2
massbal
demand
objective;

reactor1..    x1-12.5*y1 =l= 0;
reactor2..    x2-15*y2  =l= 0;
massbal..     xo =e= x1 + x2;
demand..     0.8*x1 + 0.667*x2 =e= 10;
objective..   C =e= 5.5*(x1**.6) + 4*(x2**.6) + 5*xo;

model reactor_select /all/

option domlim = 100

solve reactor_select using minlp minimizing C;

display x1.l,x2.l,xo.l,y1.l,y2.l,C.l;
```

CHAPTER 5

Laboratory Case Study 1 - Interacting Tanks

The main laboratory at the School of Chemical Engineering at the University of Natal has pilot plants, equipped with up to date computer systems that are used to commission and validate control systems. These processes do not possess any hybrid behavior, but were easily manipulated to exhibit hybrid system characteristics. It was arranged so that their operation was dependent on discrete inputs and decisions. In the closed loop control of these processes, the control algorithm, based on DMC, had to optimally automate these decisions. Control of the lab processes in a sense directed the development and design of the control algorithm e.g. 3-mode input required for the thermal system in the following chapter.

This chapter presents the first case study undertaken which directed the design and commissioning of the mixed integer predictive controller (MIPC) using the DMC structure. It commences by presenting a description of the system and thereafter goes on to show briefly how the controller was designed. Section 5.3 and 5.4 presents offline (simulated) and online (measured) tests, performed with the controller in closed loop with a model and the actual plant respectively. The chapter ends with a conclusion, summarising the findings from these tests.

5.1 Process Flow Diagram

Figure 5.1 shows a pair of interacting tanks used by undergraduate and postgraduate students for process control practicals in the School of Chemical Engineering. This system, due to its interacting nature, is ideal for commissioning model predictive controllers. Control of a multivariable, interacting system like this one requires an optimisation procedure, which is the basis of MPC. The interacting nature of the system can also be used to show the tunability (ease of tuning) and robustness of MPC strategies.

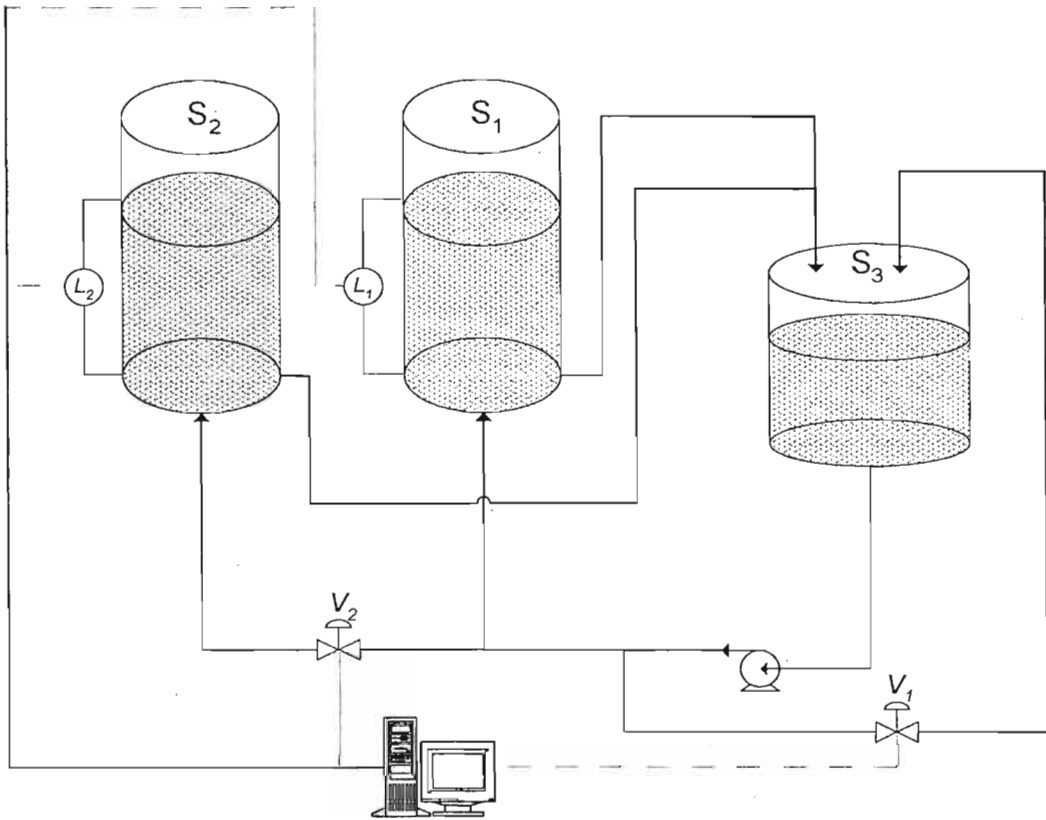


Figure 5.1 Interacting Tanks

The system consists of three tanks. S_3 is a storage tank, while S_1 and S_2 are the tanks that exhibit interacting behavior. A single pump transfers water from S_3 to S_1 and S_2 . The line from S_3 provides water for S_1 and S_2 separately, while V_2 , fitted on the line to S_2 , can adjust the flow to S_2 . Water returns to S_3 from S_1 and S_2 under gravity. The controlled variables are the levels of the two tanks, L_1 and L_2 . With this arrangement the interacting nature of the system is apparent. A variation in a single valve causes changes to both levels. If V_1 is opened, both levels go down, while if V_2 opens, the level in S_1 (L_1) goes down and that of S_2 (L_2) goes up. MPC is thus used to determine the optimum valve positions, in the light of constraints and interacting behavior, that best satisfies setpoint conditions.

Hybrid behaviour was introduced by getting one input to operate continuously (V_1) and the other discretely (V_2) i.e. allowing V_2 to take on only a finite number of positions, usually two or more. Effectively in terms of the controller this meant that it was able to treat V_1 as a continuous input variable, while for V_2 , it could only select its position from a finite set. This selection procedure introduced the necessary discontinuity into the system. The controller had to perform the selection while meeting system objectives i.e. setpoint tracking but using minimum control effort.

5.2 Modelling and Control

As explained in chapter four, using DMC for control requires step response data of the system, which in turn generates the convolution model. Appendix B.1 presents the step responses used to generate the dynamic matrix for this interacting tank system. Each input (V_1 and V_2) was stepped in turn and the response of the two controlled variables, for each step, was recorded. As explained in chapter three “DMC algorithm”, the responses were used to construct the \bar{B} , \bar{B}_{ol} , \bar{B}_o sub-matrices for the dynamic matrix in equation 3.2.

Because the tanks are continuously drained whilst they are filled, the levels reach a steady state after a step change in valve positions. For a steady state condition, the rate of inflow equals the rate of outflow. As is the case for most self-draining tanks, the rate of outflow is dependent on the level of liquid in the tank, which in turn is dependent on the flow into the tank. After a change in inlet flow (changing V_1 and/or V_2), there is transient level response before the rate of flow in and flow out is equal again and the system re-establishes a new equilibrium operating position.

Due to the interacting nature of the system, certain combinations of levels could not be attained with the manipulations available. As a result there was a need to define a certain operability region within which any combination of setpoints could be achieved with the current control effort. The restrictions were due to system characteristics i.e. the pumping capacity of the pump and the valve characteristics. The pump could only pump a certain volume of water for any given head and the valves could only allow a certain flowrate through when they were fully open. The operability region was especially important for the online tests, where absolute valve positions resulted in only certain tank levels.

By arranging the step response data to fill positions in the dynamic matrix (equation 3.2), a transfer function model of the process was created. Thus, using this as a base case, the resulting response of the system could be evaluated for any input changes, past and future. As it stands, this could now be used to control the interacting system with both valves operating continuously and the controller making continuous selections. If equation 3.4 were to be written into GAMS and optimised using equation 3.5 as the objective function, then GAMS, treating $\Delta\bar{m}$ as a continuous variable, would find the optimum changes to the valves in order to meet the objectives. $\Delta\bar{m}$ would be expressed implicitly in the program, as it appears in equation 3.4. As long as it is part of an expression, there is no need to express it explicitly. GAMS would solve this internally in its optimisation.

However to allow for the induced hybrid behaviour, further modifications had to be made, that enforced the discrete selection of V_2 . As explained in section 3.2.4 “DMC and Hybrid systems” modifications were aimed at the $\Delta\bar{m}$ vector in equation 3.4. For this particular case, it comprised both continuous changes for V_1 and integer changes for V_2 .

To model this in GAMS, two sets of vectors were required, one representing an absolute continuous input (*continput*) and the other an integer input (*discinput*). The idea was to generate them over

the optimisation horizon and then finally superimpose them to form the absolute input vector $\bar{M} \cdot \Delta \bar{m}$ in equation 3.4 could thereafter be evaluated by difference between the absolute input from the previous time step.

For discrete input vector $\overline{discinput}$, a vector of binary variables $\overline{binvect}$ (b_i are binary variables, taking on a value of only 0 or 1), was necessary for the selection of the discrete valve positions. Tri-diagonal matrix \overline{Sel} contained the integer valve positions for V_2 . If the following positions were allowed for V_2 , $[a \ b \ c]$, then $\overline{discinput}$ could be evaluated for a three step optimisation procedure as follows:

$$\overline{discinput} = \overline{Sel} \times \overline{binvect} = \begin{bmatrix} a & b & c & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a & b & c & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a & b & c \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \end{bmatrix} \quad (5.1)$$

b_i are binary variables, taking on a value of only 0 or 1. At any given time step, only one selection from, a , b , or c was possible, so the following equality constraint was necessary to enforce only one selection:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (5.2)$$

If the optimum positions for V_2 were $[a, a, c]$ over the control horizon, $\overline{Discinput} = \begin{bmatrix} a \\ a \\ c \end{bmatrix}$ and if the

optimum continuous inputs for V_1 were $[x_1, x_2, x_3]$, $\overline{Continput} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$,

then to comply with the DMC structure, inputs \bar{Y} (although in this case absolute) must have the form

$$\bar{Y} = \begin{bmatrix} x_1 \\ a \\ x_2 \\ a \\ x_3 \\ c \end{bmatrix}$$

So in addition to equations 5.1 and 5.2, the following matrix manipulation, with ordering matrices \bar{I}_1 and \bar{I}_2 , was necessary:

$$\bar{Y} = \bar{I}_1 \times \overline{continput} + \bar{I}_2 \times \overline{discinput} \quad (5.3)$$

which in this case was

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ a \\ c \end{bmatrix} = \begin{bmatrix} x_1 \\ 0 \\ x_2 \\ 0 \\ x_3 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ a \\ 0 \\ a \\ 0 \\ c \end{bmatrix} = \begin{bmatrix} x_1 \\ a \\ x_2 \\ a \\ x_3 \\ c \end{bmatrix} = \bar{Y}$$

Finally since DMC is based on changes in inputs ($\Delta \bar{m}$, see equation 3.2), the absolute valve positions now had to be converted into changes over consecutive time steps. If the present inputs for V_1 and V_2 are x_0 and b respectively then:

$$\Delta \bar{m} = \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ b \\ x_1 \\ a \\ x_2 \\ a \\ x_3 \\ c \end{bmatrix} \quad (5.4)$$

To control the interacting tank system with this “induced” discontinuity, equations 5.1 to 5.4 were used in tandem with the usual DMC equations (3.4 and 3.5), together with constraints on inputs and outputs. See appendix B.3 for the equation block, as compiled in GAMS to optimise the dynamic performance of the system by evaluating the continuous changes to V_1 and discrete changes to V_2 at regular time intervals. GAMS proved to be an appropriate tool for the design and development of the MIPC algorithm. Firstly, it had provisions for the continuous and binary vector variables (*continput* and *binvect*). As discussed, for a hybrid system with continuous and discrete inputs, these vectors proved to be fundamental to the control algorithm. Secondly GAMS simultaneously solved the system of equations as presented. Logical constraints involving the continuous and/or discrete variables could also be introduced as part of the optimisation procedure, i.e. equation 5.2.

Testing of this control algorithm took the form of online measurements and offline simulations. Offline tests were performed in closed loop with a convolution model of the process that replaced plant inputs in the SCADA system. The model was superior to the actual plant when performing investigations for tunability, because there was no noise and no disturbances in state responses as was the case with the plant. Testing against a model had the added advantage of running the model together with the controller faster in real time and also, tests could be performed away from the actual equipment.

5.3 Offline Tests

Simulated setpoint step tests were performed with the following objectives in mind:

- Firstly to show the tunability of the MIPC algorithm.
- Secondly to illustrate the different state behaviour (steady state offset and oscillations) induced into a system by a controller that is restricted in its selections of inputs, in this case V_2 .
- Thirdly, to evaluate the decision-making ability of the controller which is based on an optimisation procedure.

The convolution model against which the controller was tested for the offline tests was created using the step response data generated for the controller. (See appendix B2). Simulations for the offline tests were conducted with a control interval of 10 seconds and an optimisation horizon of 10 steps. In the following predictions, the control horizon is presented in terms of number of optimized moves. This can be converted into absolute time by multiplying the control horizon size by the control cycle which in this case is 10 seconds. The optimisation horizon for the simulated tests size, in this instance is thus 100 seconds ahead in time.

Table 5.1 presents the parameters set for the MIPC algorithm for the offline simulations. It also compares the different controller configurations in terms of performance which is gauged by the quality of level control attained.

Figure	horizon size(steps)	optimised moves	W_1	W_2	λ_1	λ_2	Allowed V_2 position	Level control
5.2	10	1	1	1	1	1	10,30	satisfactory
5.3	10	2	1	1	1	1	10,30	satisfactory
5.4	10	2	1	1	1	0.1	10,30	best
5.5	10	2	1	10	1	0.1	10,30	satisfactory
5.6	10	2	1	1	1	0.1	10,30	oscillatory
5.7	10	2	1	1	1	0.1	10,20,30,40,50	satisfactory

Table 5.1 Summary of controller parameters and performance for simulations

5.3.1 Controller Tuning

The tunability of a model predictive controller is important, especially for interacting systems and also where it is required that the controller does not take severe action in the light of plant model mismatch. As discussed in chapter three “Model Predictive Control,” the DMC algorithm can be tuned via three parameters. The first two involve the process inputs and outputs in equation 3.5. W , penalises the deviation of the closed loop error from setpoint for each system output state, while λ , referred to as move suppression, encourages or discourages severe changes on the inputs. The third tuning parameter is the size of the control horizon. Having a control horizon with more than one move, results in bolder action on the first move, with the subsequent moves thereafter accounting for correction. This results in larger gains on the first move and generally better control on the dynamic parts of the response. As compared to responses with more than one control move, those with a single control move tend to be sluggish because the computation is forced to target the steady state in one move.

Figure 5.2 presents the control effort for a setpoint step in both states with the controller allowed only one optimised move and the other tuning parameters (W and λ) set to one. An optimisation horizon of 10 steps was used throughout the course of the tests. V_1 was set continuously while V_2 could only be selected from two positions i.e. [30 10] % open, with the difference between them being of greater importance. A reasonably small Δm between the only two possible valve positions for V_2 ensured smoother dynamic responses, from which reliable conclusions could be drawn. Figure 5.3 presents the same setpoint sequence, but with the number of optimised moves set at two.

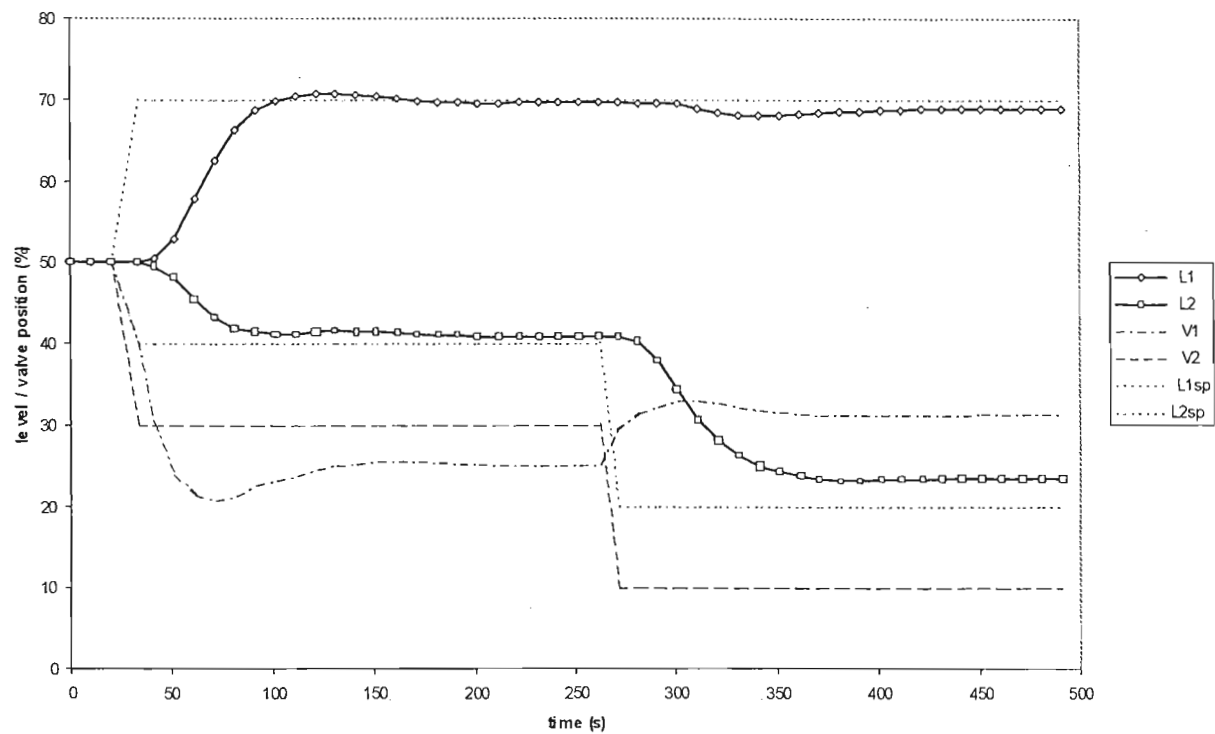


Figure 5.2: Offline simulated setpoint step test with 1 optimised move; $W_1=W_2=\Lambda_1=\Lambda_2=1$

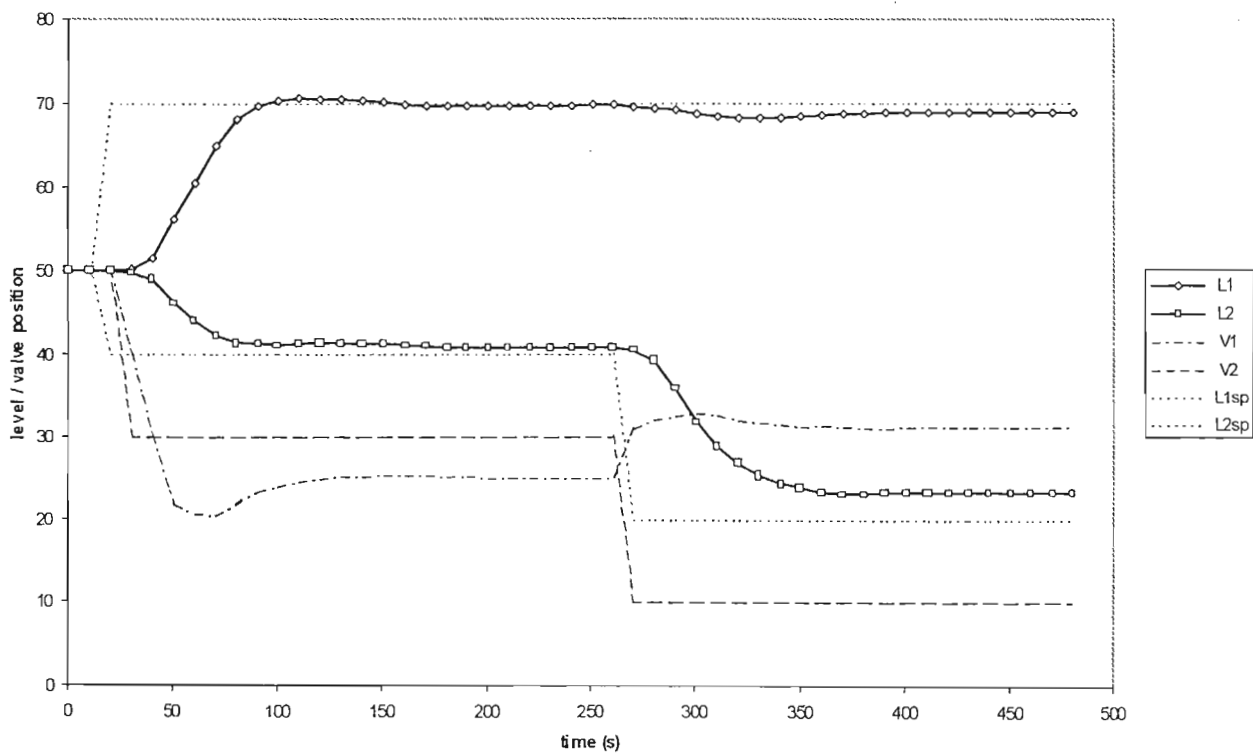


Figure 5.3: Offline simulated setpoint step test with 2 optimised moves; $W_1=W_2=\Lambda_1=\Lambda_2=1$

In response to changes in setpoint, the controller was seen to make logical decisions in order to get the tank levels to track setpoint. Contrary to expectation, the extra move in figure 5.3 made little

difference to the action of the controller as compared to one move. Notice in both cases, the offset in L_1 and L_2 when L_2 setpoint was stepped down to 20%. The controller set V_2 to its lower position (10% open) to accommodate for this drop in setpoint, but the change ($\Delta m = -20\%$) was not enough to get L_2 to setpoint. Effectively this left V_1 to get L_2 to setpoint, but V_1 could not be over-exerted as it also had to get L_1 back to setpoint, which was offset due to the change in V_2 . Eventually the optimisation determined that the optimum operation was with both levels experiencing a steady state offset. Whether the controller oscillates the valve between its operating positions (see section 5.3.2) depends entirely where it ends up at the time-interval. The algorithm has clearly calculated that the penalty will be less this way than if the state continuously crossed the setpoint by valve oscillation – where the output oscillation would be too offset or too big. A different controller action and state response was observed when the same procedure was repeated, but with $\lambda_2 = 0.1$ (figure 5.4).

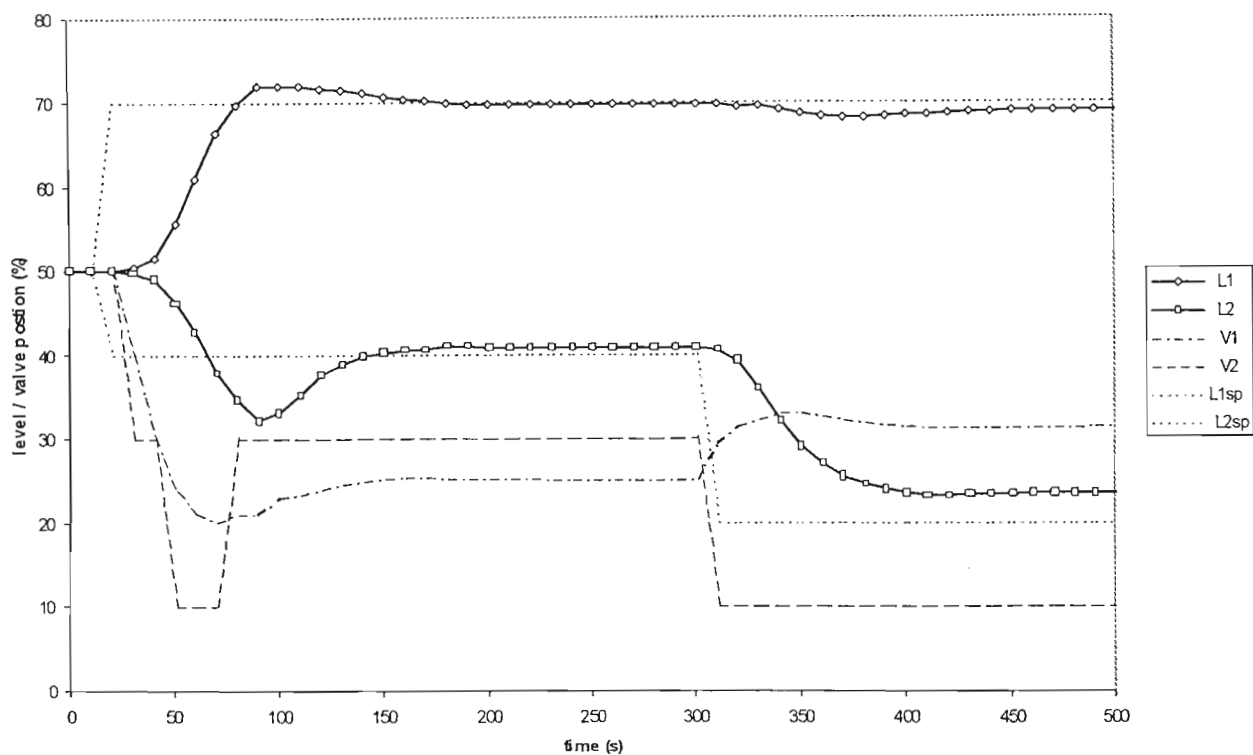


Figure 5.4: Offline simulated setpoint step test with 2 optimised moves; $W_1=W_2=\lambda_1=1$; $\lambda_2=0.1$

Lowering λ_2 made the controller more inclined to change the position of V_2 . Notice on the first part of the step test that the controller selected V_2 at 10% open, which resulted in an overshoot in both levels with respect to their setpoint. Thereafter the overshoot was corrected by returning V_2 to 30%, while V_1 was set accordingly. This kind of operation is typical of a control horizon with more than one move. The selection of V_2 to operate at 10% was initiated on the basis that the optimisation corrected for any overshoots on the second move. The net effect was that the controller had much more freedom in its variation of inputs, which was not the case in figures 5.2 and 5.3 where valve action, especially V_2 , was conservative. Reflecting on the control action taken in figure 5.3, with two control moves, but $\lambda_2=1$, it was evident from the similarity between the two plots 5.2 (one control move) and 5.3, that the

controller did not make use of the second move. It was only when the move suppression on V_2 was decreased that the second move started to influence the first. Thus in DMC as the number of moves increases in the control horizon, it obviously may be necessary to decrease the move suppression. Allowing more optimised moves in order to meet objectives does not guarantee that the optimisation will use all of them effectively. Lower move suppression will however encourage the optimisation to use the moves available to it, to plan a sequence of control actions extending ahead in time, especially in the event where the input can only take on positions from an integer set. Δm between integer positions is fixed and generally large, so a lower move suppression (λ) is necessary to encourage valve movement.

Increasing the number of optimised moves in the control horizon, while decreasing the move suppression, however did not alleviate the offset in the final part of the response. The controller still evaluated this, with the control action that it had at its disposal, as an optimum state of operation. To show the effect of weighting on setpoints, the weight (W_2) on L_2 was increased. Figure 5.5 shows the resulting control action.

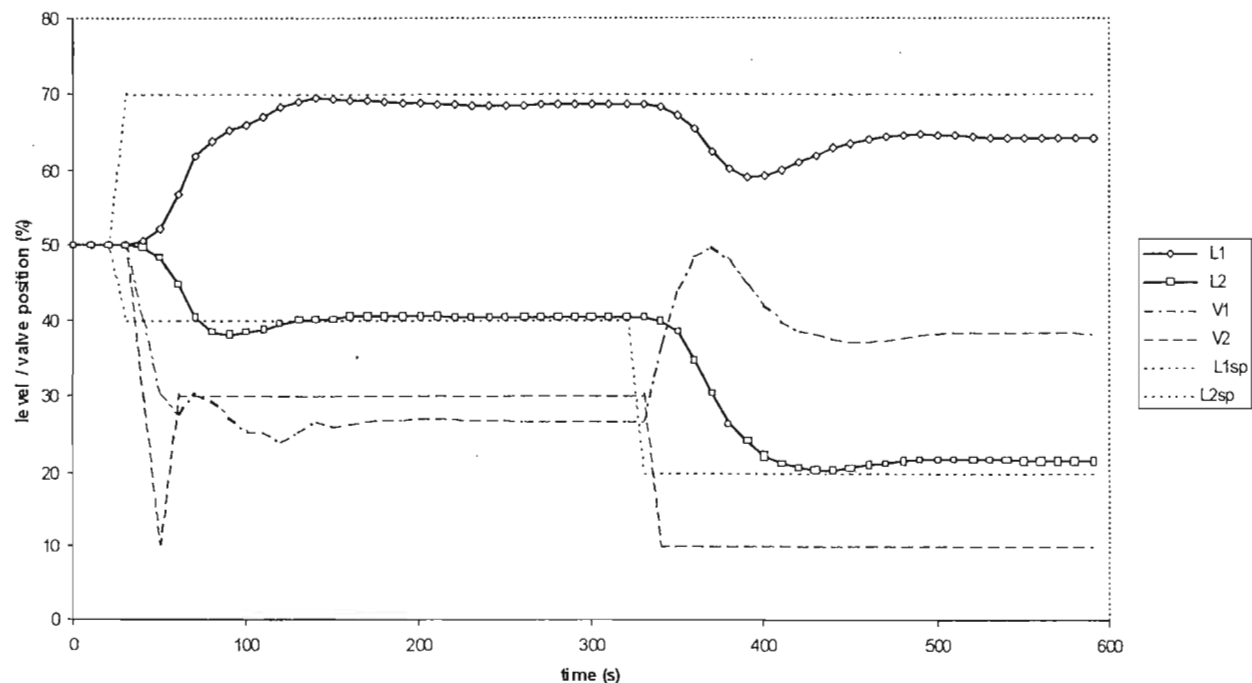


Figure 5.5: Offline simulated setpoint step test with 2 optimised moves; $W_1 = \lambda_1 = 1; \lambda_2 = 0.1; W_2 = 10$

From the responses, L_2 tracked its setpoint more closely, while L_1 endured the offset. Clearly both valve actions favoured L_2 . This was evident when L_2 setpoint was stepped down to 20%, V_2 , as expected, was selected to operate at 10% but V_1 increased to aid in decreasing L_2 . The action of V_1 occurred in spite of it causing L_1 to depart from its setpoint (see time period 350s-400s). Also at the final part of the response (after 500s), L_2 was closer to its setpoint than L_1 (relatively smaller offset as compared to L_2).

5.3.2 Contrasting controller action

Depending on the initial state positions relative to the setpoint, different types of control action and hence system responses were observed. If through its discrete selection, the controller was able to take states to their setpoints, or close enough with a reasonable offset, then steady state behaviour like that shown previously was noticed. If on the other hand, there was a selection available to the controller, that allowed it to alleviate the offset, then another type of response and control action was observed. As shown in figure 5.6, state responses in this case became oscillatory, with the controller periodically switching the position of V_2 .

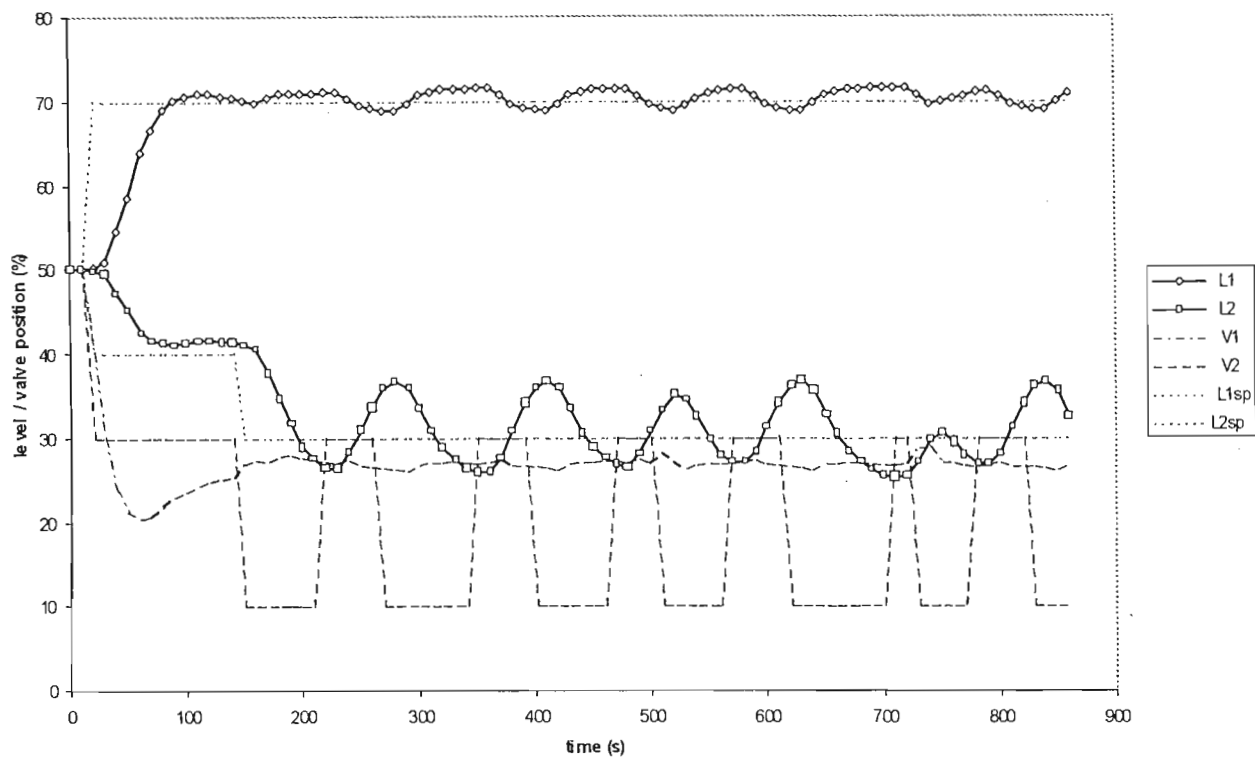


Figure 5.6: Offline simulated oscillatory level Control with 2 optimised moves; $W_1=W_2=\Lambda_1=1; \Lambda_2=0.1$

The operating point for L_2 on the second setpoint step was changed from 40% to 30%, instead of 20% as done before. In response to this shift in operating point, the controller immediately selected V_2 to operate at 10%. The impact of this change caused L_2 to decrease below setpoint. The controller thereafter determined that it was better to get the state above setpoint (optimal decision) by increasing V_2 to 30%, than just leaving L_2 to attain a steadystate offset. This sequence continued, with the controller making use of the integer positions for V_2 , to get an oscillation of L_2 about the setpoint. Evidently this type of operation was optimally better than that where the states previously had a steady state offset. Obviously this was dependent on: the initial state and its position relative to the setpoint; the input change (Δm) available to the controller; and relative phasing of the setpoint steps and controller interval. Clearly the constant switching between the discrete input state positions resulted in an unsteady operation. However on average, the output states are kept closer to setpoint than with the steady state offset.

5.3.3 Multiple selection for V_2

Finally in showing the versatility of the control algorithm, a control procedure was undertaken where the controller was allowed to select from more than two operating positions for V_2 i.e. [0 10 20 30 40 50]. Figure 5.7 shows the resulting control procedure with the same setpoint changes as before.

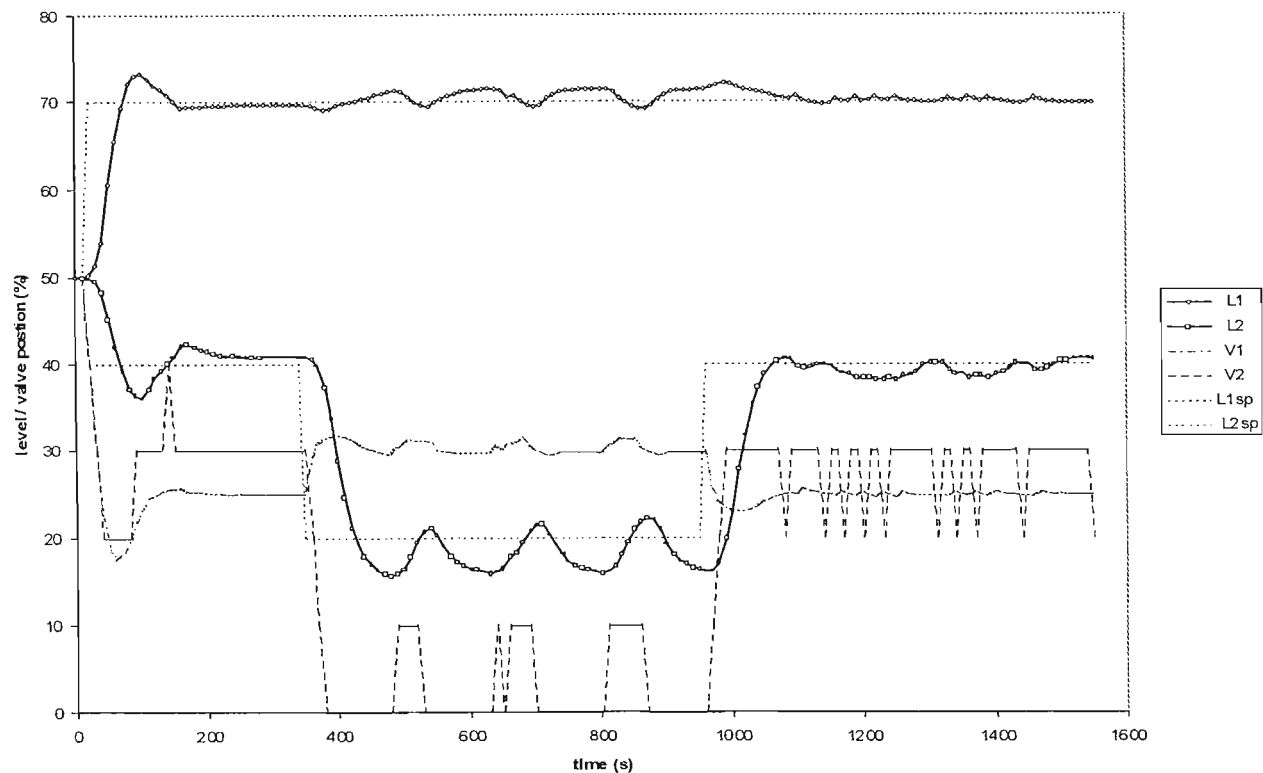


Figure 5.7: Offline simulated set point step test with V_2 allowed intermediate positions while using 2 optimised moves; $W_1=W_2=1$; $\Lambda_1=1$; $\Lambda_2=0.1$

Figure 5.7 shows that in getting the levels to track their respective setpoints, the controller made selections for V_2 from all integer positions made available to it. Steady state offsets occurred at only a single pair of setpoints ($L_1 = 70$ and $L_2 = 40$) during the initial part of the simulation. Thereafter when the next two changes in setpoint were made, the controller firstly made the appropriate changes to get the levels to their new operating positions, and subsequently selections were based on regulating the states at their setpoints. Regulatory control amounted to constantly changing the position of V_2 ($\Delta m = \pm 10$), while L_2 oscillated about its setpoint. In this case, giving the controller greater freedom of choice regarding V_2 , meant that it had a greater dependence on firstly the dynamic and then the regulatory aspect, once the operating point was reached. This action is typical of a servo control problem, where the operating points change frequently and span a wide range. In between the changes, regulatory control is also required. Controllers of this type are used to control systems requiring both servo and regulatory control through selections and initiations. More notably they find a place where gear and speed selections are needed to maintain variable loads and demands e.g. the car example discussed in chapter two.

5.4 Online Tests

Table 5.2 presents the parameters set for the MIPC algorithm for the online measurements. It also compares the different controller configurations in terms of performance which is gauged by the quality of level control attained.

Figure	horizon size(steps)	optimised moves	W_1	W_2	λ_1	λ_2	Allowed V_2 position	Level control
5.8	10	2	1	1	1	0.1	20,40,60,80,100	satisfactory
5.9	10	3	1	1	1	0.01	0,100	oscillatory

Table 5.2 Summary of controller parameters and performance for online measurements

Once commissioned on the model, it was interesting to see how the controller would perform online on the plant. The plant brought with it additional challenges for the controller which were not present in the offline tests. Firstly was the problem of noise, which had the possibility of introducing plant-model mismatch. This was however seldom of real concern as the mismatch, if anything would be slight and the feedback feature of MPC would be able to deal with it. Secondly was the possibility of non-linearity, where system responses could be different, for the same or reverse change in input, depending on where in the operability region the system was functioning. This would also result in plant model mismatch. Lastly was the problem of defining the operability region, within which the combination of setpoints (L_1 and L_2) could be achieved by the valve ranges on the plant. Any step test that was undertaken had to fall within this region (appendix B.2). Due to the restricted size of this region, the plant operation within it would be relatively linear, thereby reducing the concern of mismatch.

Many closed loop tests were done with the plant and controller in closed loop. The control interval for the online measured tests was set at 45 seconds. The optimisation horizon in this instance was thus 450 seconds, representing the time taken for the system to attain steady state after a change in inputs. Control quality was similar to that for the offline tests. Presented here are just two tests that show the online performance of the controller. Settings from the offline simulations could not be used for the online measurements mainly because in the real plant certain operating positions i.e. pairs of setpoints, could not be attained (see appendix B.2). However, similar responses should be achieved by using the same valve position gap, assuming that the system is not too non-linear.

Figure 5.8 shows the level control of the interacting system with V_2 allowed to take on only the positions, [0 20 40 60 80 100] % open. The plant response was similar to that with the model (figure 5.7). There was a region of offset in (1200s-1400s) and also a region of oscillatory behaviour in L_2 (final part of the step response). The controller made use of all possible operating positions of V_2 in order to get the states to track setpoint. An incremental limit of 20% was placed on V_2 , which ensured

that all possible positions of V_2 were utilised and none were skipped when a new operating position was introduced. While setpoint tracking was satisfactory, the controller was once again seen to make logical control decisions for the operating position of V_2 .

Figure 5.9 presents a more realistic scenario. V_1 is allowed to be set continuously, but V_2 can only be fully open or shut. This type of operation is typical of a reservoir distribution system (e.g. drinking water), where valves/pumps might change operation to maintain levels within limits. Tight level control in this instance is of secondary importance. Level control on both levels is not as tight as in the previous control sequences. Both states oscillate about their setpoints due to the opening and closing of V_2 . L_1 oscillates within a narrow band about its setpoint, while L_2 is not controlled as tightly. It is noticeable that at 12:35:00, L_2 becomes increasingly oscillatory. Slacker tuning e.g. higher move suppression or lower W_2 would result in less oscillation, but then at the expense of further distance from setpoint. Notice that in order to get satisfactory level control on both tanks, this test had to be done with a control horizon of three optimised steps. In addition, because Δm in this case was $\pm 100\%$, move suppression on V_2 had to be decreased to encourage changes.

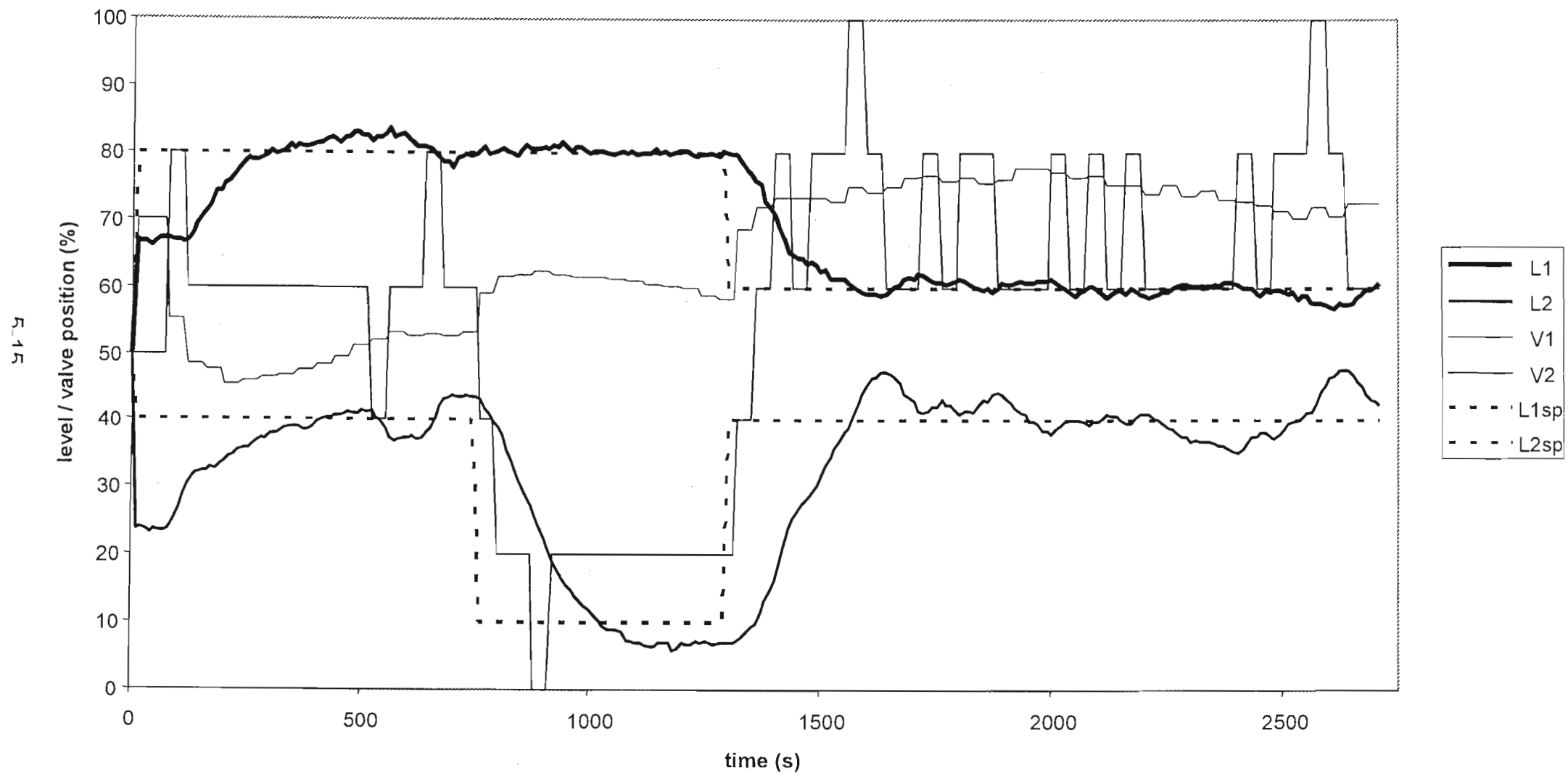


Figure 5.8: Online measured level control with 2 step optimisation and V_2 allowed intermediate positions

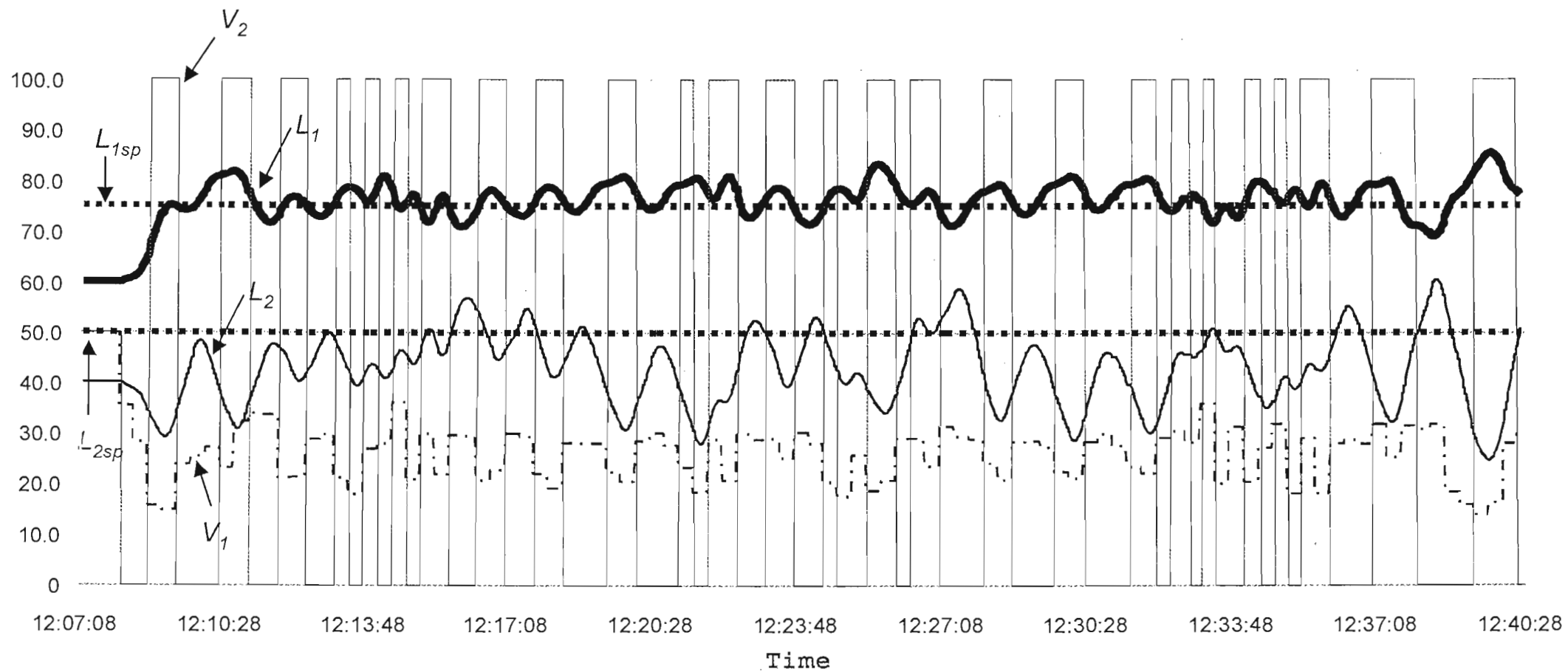


Figure 5.9: Online measured level control with V_2 only open or shut with 3 step optimisation horizon

5.5 Conclusion

To commission the MIPC algorithm on an interacting tank system (figure 5.1), hybrid system behavior was induced by getting the controller to evaluate V_1 continuously while discretely selecting operating positions for V_2 . This system proved itself to be a satisfactory environment for commissioning the MIPC algorithm. The controller, in basing its selections on an optimisation procedure, was observed to make logical control decisions for the operation of V_2 , while setting V_1 continuously.

The inherent dynamic behaviour of the interacting tank system, with the discontinuity being on the input, presented the opportunity to bring forth some interesting features concerning hybrid system control. Firstly, cross-coupling between input and output variables tested whether effective decoupled control could be achieved with normal tuning procedures.

Another feature of the system, used to reveal different types of control action and state response, was that the states of the system (levels) reached steady state if the position of the discrete input remained unchanged over a period of time. If through the real time optimisation, the controller found that it was not necessary to change the operating position of V_2 , then over a period of time, both levels would reach a steady state (figures 5.2/3). This effectively left V_1 to control L_1 and L_2 and in most cases, one or both of the states would experience a steady state offset with respect to their setpoints. On the other hand, if the state was close enough to setpoint and depending on the options available in selecting V_2 , the controller might decide that the optimal trajectory was to overshoot the setpoint, as compared to enduring a steady state offset. The overshoot was corrected for, by a reversal of the decision on the subsequent time steps (figure 5.6). The net effect was a type of “bang-bang” control action, with output states following an oscillatory behaviour.

For hybrid systems that have discrete inputs, with fixed Δm between positions of operation, move suppression (λ) was found to have a significant influence on the control action (figure 5.4). If it was too high then the controller was found to be reluctant in switching the position of the input. This resulted in the states enduring steady state offsets. When move suppression was lowered, with the control horizon increased, then the algorithm tended to switch the operating position of the input more frequently. Increasing the control horizon had to be followed by a decrease in move suppression, to ensure that the optimisation made effective use of all the moves in the horizon.

Offline tests show satisfactory results, where increasing the control horizon and decreasing the move suppression, was further shown to encourage changes in the discrete input.

CHAPTER 6

Laboratory Case Study 2- Thermal Circuit

After commissioning the mixed integer control algorithm on the interacting tank system, it was further extended to automate another laboratory scale process in the School of Chemical Engineering. This particular system, a thermal circuit, required a different approach due to its distinct modes of operation. The operation of a system can be divided into regimes or modes if there are discontinuities on the input or output of that particular system. For the thermal circuit, the controlled variable (output) was continuous, while the discontinuous manipulated variables (inputs) defined the modes of operation. When the position of the discontinuous inputs changed from one operating position to another, then a switch in mode had occurred. In automating hybrid systems having these characteristics, it is the purpose of the controller to determine the switching policy of that particular system.

This chapter presents the second case study undertaken, involving the control of a system with hybrid behaviour. It commences by presenting a description of the system and then goes on to define the different modes of operation. Section 6.2 shows how a model predictive controller can be designed to initiate the switching between modes in real time. As before, the DMC framework was used. Section 6.3 presents the results of offline and online tests that demonstrate, in closed loop, the performance of the mixed integer controller. Lastly, the chapter concludes with a recollection of the findings from this section of research.

6.1 Process Analysis

6.1.1 Process flow diagram

Figure 6.1 is a process diagram of the thermal circuit, as it exists in the laboratory, with gas heaters, cooling devices, valves and piping. The thermal fluid, water in this instant, is pumped out of tank S1 through a heating-cooling circuit and then returned to S1. Before the cooler is a return line, fitted with control valve *CV13* that allows flow to bypass the heating-cooling devices and return to the tank. In parallel with the gas heaters, is another line, fitted with control valve *CV04*, that allows the heaters to be bypassed. The heaters are introduced into the network by control valve *CV05*. All three flows: from the bypass; the cooler; and the heater, converge at a mixer before returning to S1.

Temperature measurements are available at three points in the circuit. Firstly, at the exit of S1, secondly after the cooler and lastly before the entrance to the tank. Thermocouples *T14*, *T30* and *T31*

measure these temperatures respectively. For the purpose of this investigation, the controlled variable was chosen to be T_{31} i.e. the temperature of the return flow to the tank.

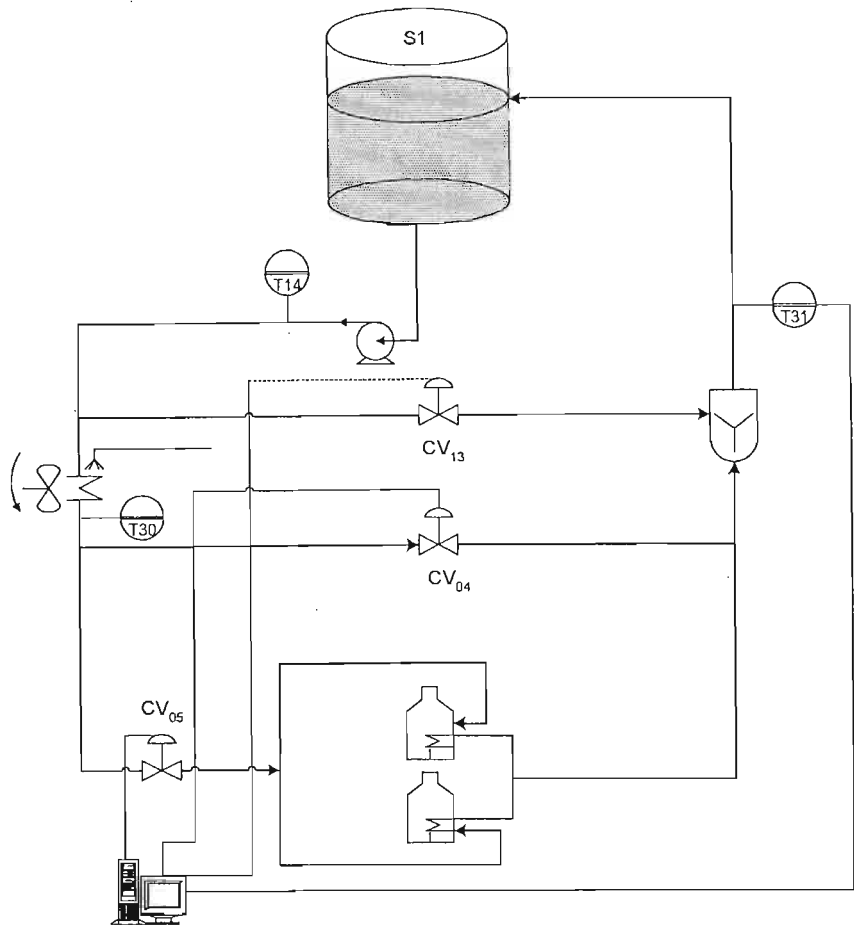


Figure 6.1 Thermal circuit

6.1.2 Modes

By strategically setting the combinations of the control valves relative to each other, the thermal circuit can be forced into having modes of operation. This system was configured into having three distinct, mutually exclusive modes of operation i.e. “recycle”, “cooling” and “heating”. The system was in recycle mode when of the three control valves, only CV_{13} was open, while for cooling only CV_{04} was open and for heating CV_{05} was open. The valves operated on a binary signal from the controller, i.e. they were either open or shut.

Figure 6.2 shows a graphical illustration of a system with ternary modes. Each side of the triangle represents a mode transition, while the apices represent the modes. This system therefore had three modes of operation, with 6 (3×2) possible switches between these modes.

It was the combinations of the different valve positions, from the primary inputs, that defined the modes. In this case for 3 inputs, each with 2 possible states (open or shut), the number of possible

modes was 8 (2^3). However certain modes were eliminated e.g. the case where all valves are simultaneously open. Thereafter the decision had to be made with regards to the switches between these modes during operation. As before, some transitions may not be allowed. Figure 6.3 shows this sequence, beginning from the primary inputs, and ending at the allowed mode transitions for the thermal circuit. This particular sequence, of logical permutations and combinations, was relevant for the design of the controller, as the plant operated on primary input level (valves), while the controller was designed on the basis of modes and switches between modes.

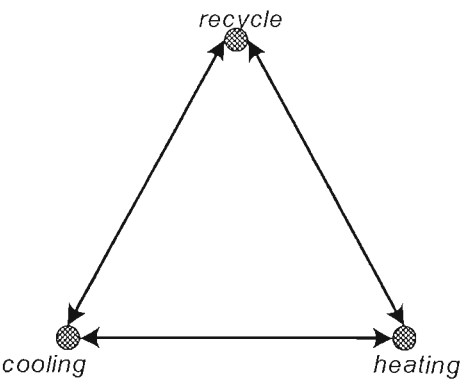


Figure 6.2 System with ternary modes

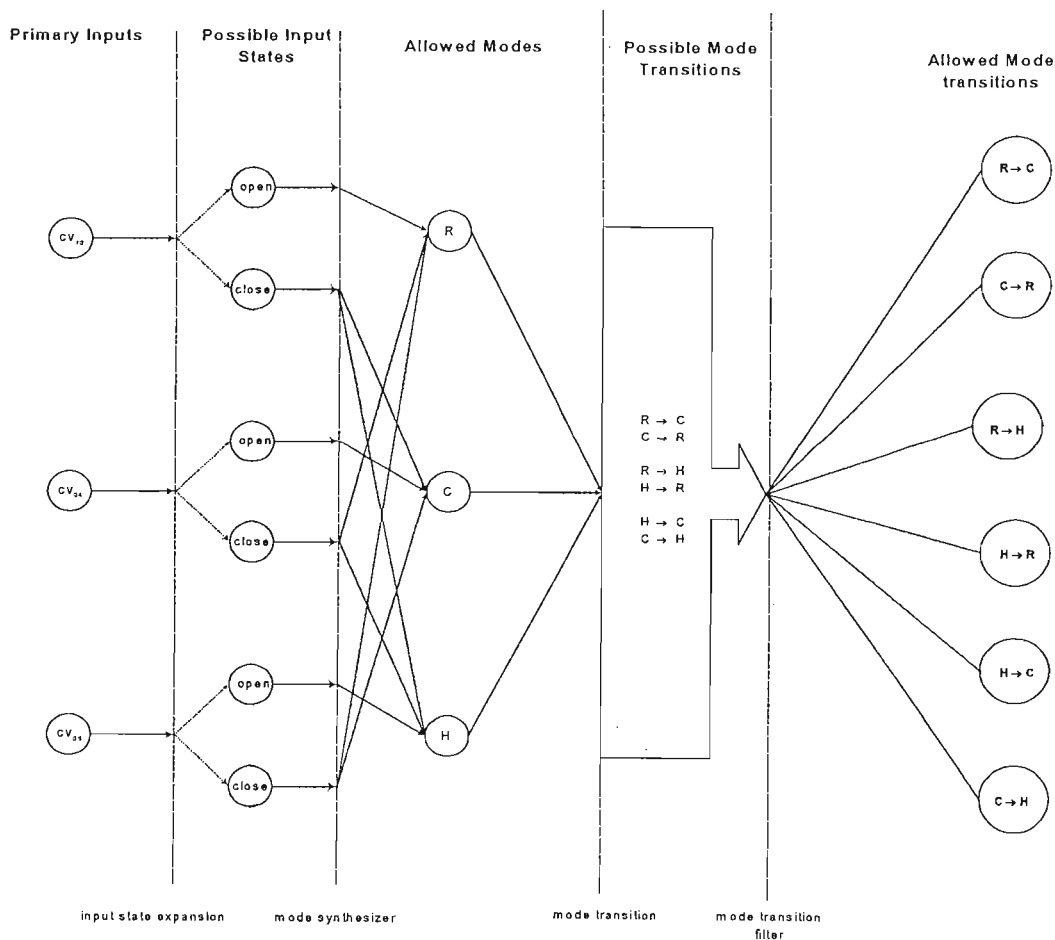


Figure 6.3 Conversion from primary inputs to mode transitions

6.2 DMC for systems with mode switches

6.2.1 Switch responses

Generating a convolution model for a system is usually done by stepping the input to the system and then using the system response to build the model. Actually this amounts to disturbing the system and then using its response to the measured disturbance, as a standard for other possible disturbances. For a system with modes, the disturbance amounts to switching between the modes of the system.

The DMC algorithm can therefore be applied to systems with modes, by switching between modes, recording the system response and then generating the convolution model in this way. In the optimisation procedure, the algorithm now solves for the optimum switch instead of the optimum primary plant input.

Figure 6.4 shows the openloop switch response for the actual thermal circuit. Recall that the controller was to be designed to control the inlet temperature ($T31$) to the tank. The responses for $T31$ from this test were used to generate the convolution model.

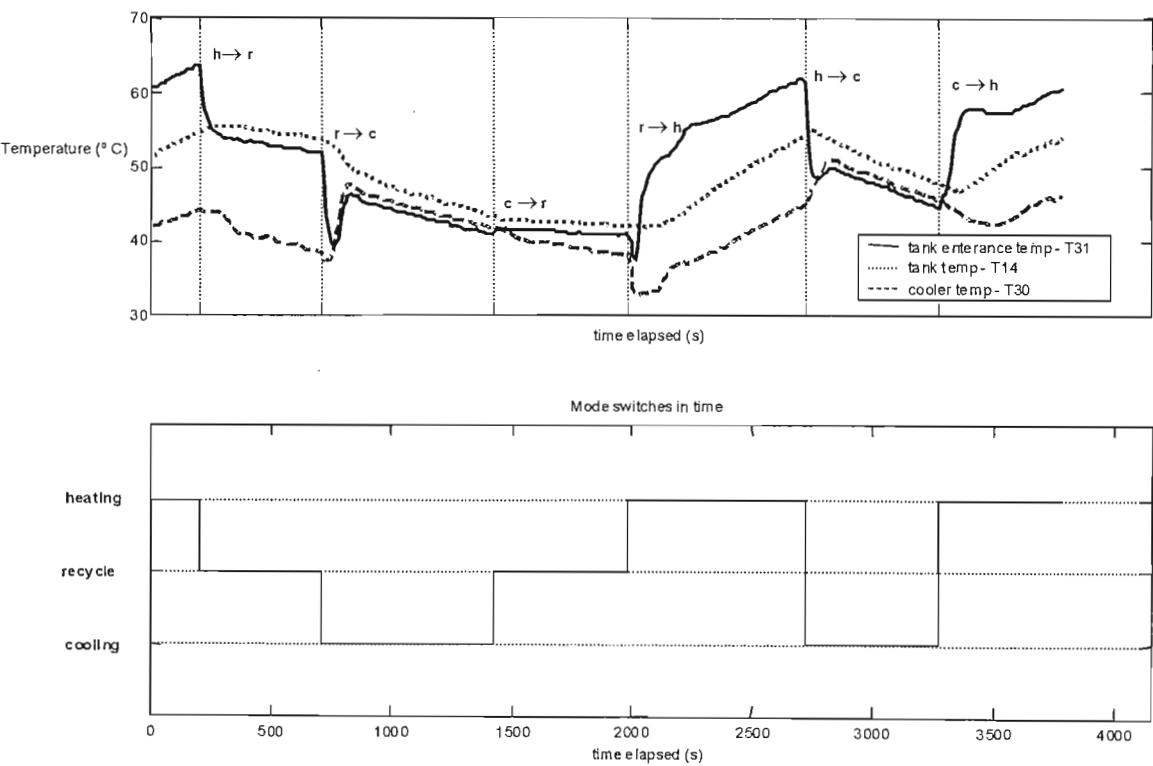


Figure 6.4 Openloop switch response data

The openloop switch responses, in figure 6.4, revealed two aspects regarding the dynamics of the system that had to be considered when designing the controller. Firstly, for heating and cooling modes of operation, the temperature of concern ($T31$) did not reach a steady state. So the integrating version of DMC as presented in section 3.2.3, “DMC and integrating systems,” had to be used. The other

aspect was the asymmetrical state responses when switching from one mode to another and then back again. For example, the response for the switch from recycle to cooling was not the mirror image of the response for the switch from cooling to recycle. Since DMC relies on a linear convolution model, symmetry is inherent in the algorithm. Figure 6.5 illustrates, in a general sense, this behaviour for a state response to a change in input. This effect of nonsymmetry can be overcome by using the response for all switches (6 in total) and then getting the optimisation to only utilise them in the direction for which each was generated. Since the switch responses are multiplied only by binary variables (i.e. 0 or 1) in the algorithm, responses are therefore only ever used in the direction that they were initially generated for the model. In the integrating case, where states do not reach a steady state, responses used in the internal model were configured to end with mirrored gradients, ensuring a cancellation when modes are switched. For example the slope of the switch from R→C is the negative of the slope C→R. In terms of the step response data, $\Delta B (R \rightarrow C)$ is the negative of $\Delta B (C \rightarrow R)$.

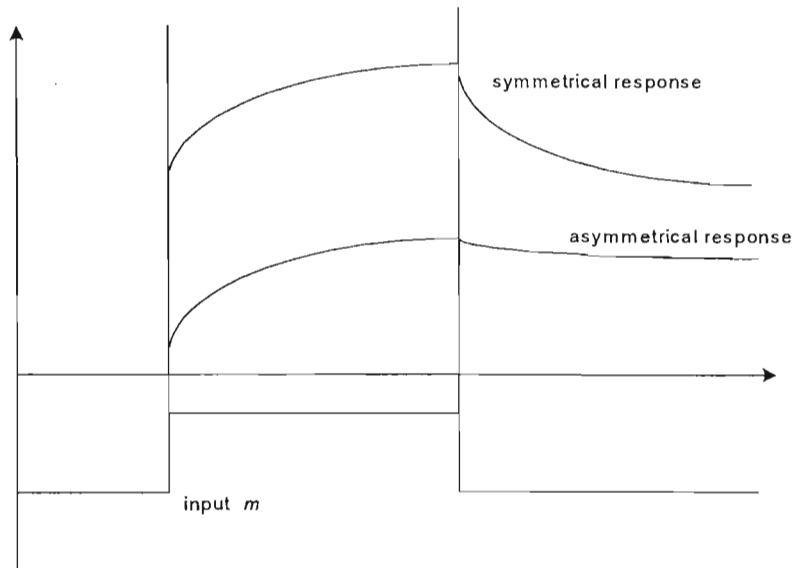


Figure 6.5 Symmetrical and asymmetrical state response to change in input

Notice a sudden, short increase in water temperature when the thermal circuit is changed from recycle to cooling mode and a decrease in temperature when changing from recycle to heating mode. The sudden increase in temperature when the mode is switched from recycle to cooling is due to the hold up effect (residence time) in the pipework. The sudden decrease in temperature when the mode is switched from recycle to heating is due to the hold up effect in the cooling device (radiator with fan and coils with water shower).

Figure 6.6 shows the switch step responses used to generate the convolution model. Note the asymmetry in the responses when switching from one mode to another and when the switch is reversed. Entries for the dynamic matrix, for a given switch, were evaluated by extrapolating the slope prior to the switch and subtracting that from the response, at every time step, after the switch. Refer to appendix C.1 for the non-symmetric switch response data used for generating the dynamic matrix.

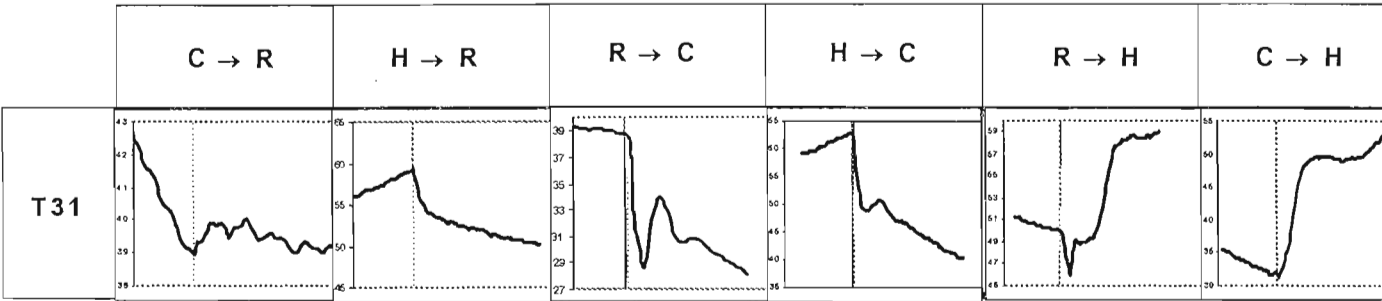


Figure 6.6 Switch responses used for convolution model

6.2.2 Adaptation for thermal circuit mode switching

In addition to the usual DMC equations (3.4 and 3.5) presented in chapter 3, “Model predictive control,” additional equality constraints were required for logical mode switching. Firstly the optimisation, in switching a mode, must select the switch based on the present mode. For example if the system is currently in heating mode, then it must only switch from heating to another mode. It is possible under the same circumstances that another mode switch e.g. R→ C, may result in a lower objective function. Secondly the conversion must be made within the control algorithm from mode switches to actual valve settings (open or closed). This conversion must be reversed (i.e. from valve position to mode switches) when the controller is called; and thereafter updated (mode switches to valve positions) when the new optimised inputs are passed to the plant. As stated previously, the controller functions on a switching basis, while the plant operates in terms of primary inputs i.e. valves.

In equation 3.4, $\bar{x}_{cl} = \bar{x}_{oMeas} + [\bar{B}_{ol} - \bar{B}_o] \Delta \bar{m}_{past} + \bar{B} \Delta \bar{m}$, the matrices making up the dynamic matrix $(\bar{B}_{ol}, \bar{B}_o, \bar{B})$ were generated as discussed previously. However the binary vector now becomes

$$\Delta \bar{m} = \begin{bmatrix} C \rightarrow R \\ H \rightarrow R \\ R \rightarrow C \\ R \rightarrow H \\ C \rightarrow H \\ H \rightarrow C \end{bmatrix} \tag{6.1}$$

Equation 6.2 was required to relate present modes $\begin{bmatrix} R \\ C \\ H \end{bmatrix}_i$ to the switches in modes (now $\Delta \bar{m}$) as evaluated by the optimisation.

$$\begin{bmatrix} R \\ C \\ H \end{bmatrix}_i = \begin{bmatrix} R \\ C \\ H \end{bmatrix}_{i-1} + \begin{bmatrix} 1 & 1 & -1 & 0 & -1 & 0 \\ -1 & 0 & 1 & 1 & 0 & -1 \\ 0 & -1 & 0 & -1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} C \rightarrow R \\ H \rightarrow R \\ R \rightarrow C \\ R \rightarrow H \\ C \rightarrow H \\ H \rightarrow C \end{bmatrix} \quad (6.2)$$

simplifying for notational purposes

$$\overline{Mo}_i = \overline{Mo}_{i-1} + \overline{A} \cdot \Delta \overline{m}$$

Where integrating matrix \overline{A} represents the two dimensional relation matrix with -1 for movement away from a state and +1 for movement towards a state. Since the transition vector ($\Delta \overline{m}$) is either 0 or 1, the elements in \overline{A} are set accordingly so that the correct mode is selected at every moment in time. \overline{Mo}_i was itself a vector of binary entries (0 or 1) with a “1” in the appropriate position representing the selection of that mode. The size of the matrices and vectors were duplicated for more than one optimised switch in time. \overline{A} was expanded out in lower triangular form, while \overline{Mo}_i got duplicated row wise to accommodate for the size of $\Delta \overline{m}$.

Another equality constraint (6.3) was required to ensure that the optimisation always made logical switches based on the current mode of the system at time step i .

$$\sum_{j=1}^3 Mo_{ij} = 1 \quad (6.3)$$

Equation 6.3 essentially ensured that the optimisation only selected 1 mode of operation at every time step i . Equations 6.2 and 6.3 were now sufficient to provide the algorithm with the necessary logic needed for optimal mode selection. Finally the optimal selection must be converted to primary inputs as required by the plant. In this case

$$\begin{bmatrix} CV_{13} \\ CV_{04} \\ CV_{05} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R \\ C \\ H \end{bmatrix} \quad (6.4)$$

If a record of past modes was not available, then before the optimisation procedure, 6.4 was inverted to convert current valve positions to modes.

6.3 Closed loop tests

Similar to the interacting tank system, the controller was commissioned and its performance evaluated in closed loop with the SCAD-GAMS control loop as presented in section 3.3. The aim of the control strategy was to maintain $T31$ at setpoint by selecting the correct mode of operation. The initial tests used simple simulated linear trajectories (section 6.3.1), where the responses to switches from one mode to another and back again were symmetric. Thereafter the controller, with different configurations, was tested online on the actual thermal circuit and then finally against an onboard model (6.3.2). The “onboard” model is a model (convolution in nature) that is loaded into the SCAD system software and simulates the actual process. Refer to appendix C.3 for the equation block in GAMS that optimised the switching of modes for the thermal circuit.

6.3.1 Simulated symmetric responses

To commission the new control algorithm and check that it was making logical selections, simple responses were used for the internal controller model and the external model. To ensure symmetric behaviour (because the controlled model was a linear convolution model), the switch step response data for the convolution model had the following interdependencies:

- $C \rightarrow R = - R \rightarrow C$
- $H \rightarrow R = - R \rightarrow H$
- $H \rightarrow C = H \rightarrow R + R \rightarrow C$
- $C \rightarrow H = C \rightarrow R + R \rightarrow H$
-

Refer to appendix C.2 for the symmetric switch response data used for the convolution model. For the external model, the selection of the heating mode resulted in an indefinite increase in temperature (similar response as switch $R \rightarrow H$). The cooling mode resulted in an indefinite decrease (similar response as switch $R \rightarrow C$). To match actual plant behaviour, the observed initial inverse parts of the $R \rightarrow C$ and $R \rightarrow H$ responses were included. Selection of the recycle mode results in an immediate steady state.

Figures 6.7 and 6.8 show controller action on the model with one and two optimised switches allowed in the optimisation horizon respectively. These tests show that the controller was able to make logical switches based on the state position relative to setpoint. When the state was below setpoint, the controller selected heating mode, or if the state was sufficiently above setpoint, then it selected cooling mode. When it approached setpoint, it was able to select the opportune moment in time to switch to recycle mode, so that the steady state response thereafter coincided with setpoint.

Figure 6.7 reveals that when the switch to recycle was made, the state had to endure an offset from setpoint thereafter. This offset was due to two reasons. Firstly, the switch in mode was based on an optimisation procedure that minimised state difference from setpoint over a closed loop horizon extending into the future. At the time of switching, a lower objective function was the result (even

with the offset), as compared to the value of the objective function if the switch had not been made then. This effect can however be influenced with the switch penalisation weight (λ) and penalisation from setpoint (W) that is part of the objective function. Secondly, the offset between the state and setpoint was also due to the timing between the model and the controller. In the control loop, the model had to wait for the optimum input from the controller. In effect, the precise optimal situation depends also on phasing – e.g. the point in the existing response at which a setpoint change might be made. A controlled variable might end steadily below or above setpoint, or even be made to oscillate near the setpoint depending on when the change in setpoint was made and also the magnitude of the change.

Comparing the two simulations, revealed that controller-switching behavior with two possible optimised switches was more effective than just a single switch. Switching based on one optimised switch was conservative, with the system having to endure offsets from setpoint. For two optimised switches, the optimisation used the extra degree of freedom to minimise the difference between state and setpoint in the closed loop horizon. This was observed when the system entered recycle mode for the final time and the response went to steady state thereafter. The temperature offset for 2 optimised switches in the control horizon was not as considerable as that for just 1. Essentially, due to the freedom it had for correction on subsequent moves, the controller with 2-optimised switches was more prone to switching the mode of operation.

Another comparison between the two control sequences shows that when the temperature is close to setpoint, the algorithm with a control horizon of 2 steps oscillates between cooling and heating mode. This is because it plans an appropriate switch (correction) on the subsequent moves, which was not the case for a single switch. The extended control horizon proved to alleviate the offset from setpoint.

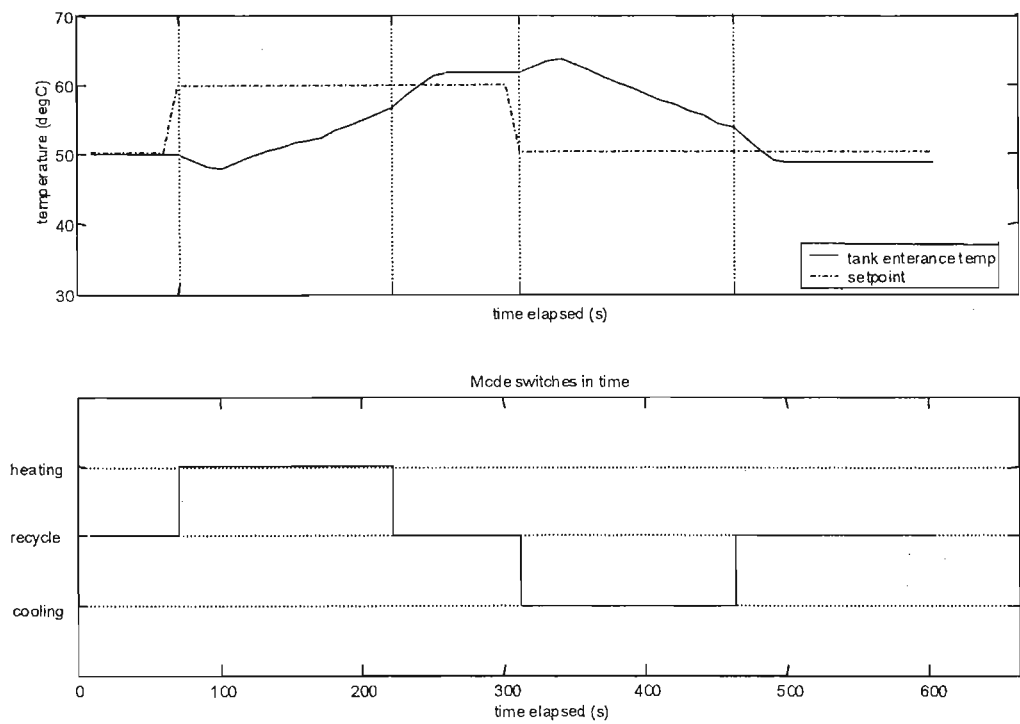


Figure 6.7 Mode selection for symmetrical responses with 1 optimised switch (control of model)

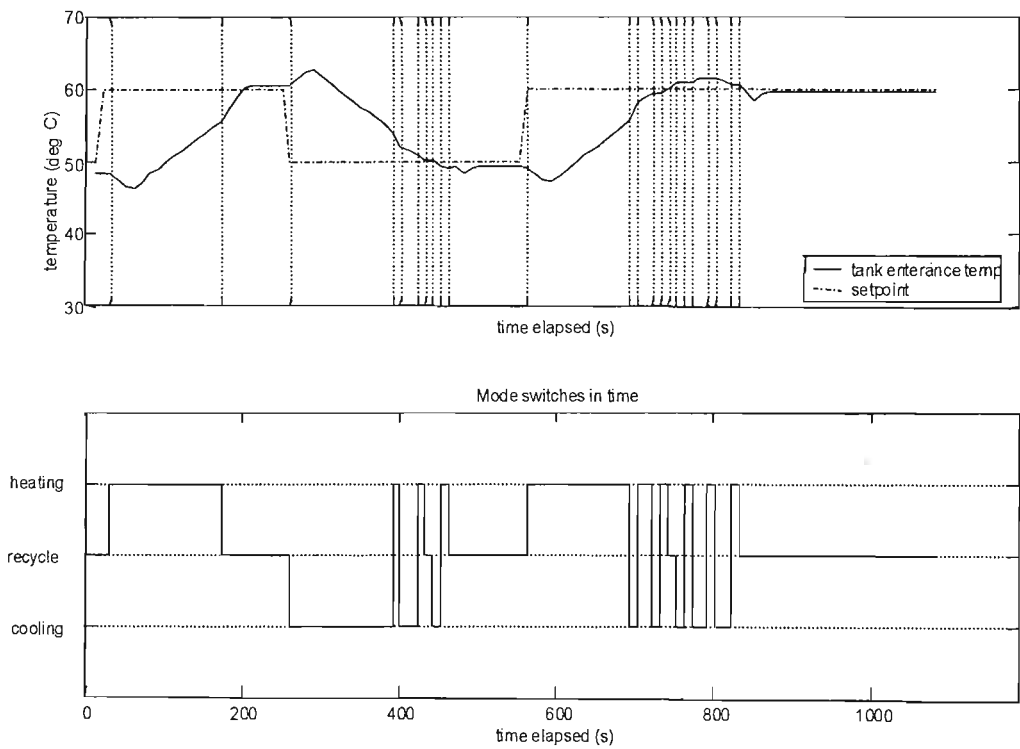


Figure 6.8 Mode selection for symmetrical responses with 2 optimised switches (control of model)

6.3.2 Measured online control

After commissioning the controller against an onboard model and establishing that it was making logical switching decisions, it was exercised online with the real plant. For the plant, responses to switches were asymmetric in nature. For its internal convolution model, the controller was thus loaded with the switch responses (see appendix C.1) evaluated from the openloop plant responses (figure 6.6). To account for the integrating effect in-between mode switches, the following modifications were made to the final pair of points on the switch responses such that:

- $\Delta B(C \rightarrow R) = -\Delta B(R \rightarrow C)$
- $\Delta B(H \rightarrow R) = -\Delta B(R \rightarrow H)$
- $\Delta B(H \rightarrow C) = \Delta B(H \rightarrow R + R \rightarrow C)$
- $\Delta B(C \rightarrow H) = \Delta B(C \rightarrow R + R \rightarrow H)$

This modification ensured the cancellation of the final temperature gradients during mode switches. The controller was thus able to keep track of the temperature change during mode switches.

Figures 6.9 and 6.10 show the online control of the thermal rig with the controller using 1 optimised switch and 2 optimised switches respectively. Unlike the simulated tests for the symmetric responses, the setpoint was never tracked for any length of time. This however was expected, because the switch to recycle mode in the actual plant had a dynamic response of its own. When the switch to recycle was made, *T31* would finally measure the initial water temperature in the tank before the switch. So the transient response after the switch to recycle, was based on the relative temperatures before the switch, of water entering the tank and that within the tank. This behaviour created a non-linear effect, as the trajectory of the response was determined by where in the temperature range the switch was made and also for how long the system was in a particular mode prior to the switch. Heat loss to the surroundings also meant that the temperature decreased gradually while in recycle mode. The model was unable to account for this due to the above relations involving ΔB .

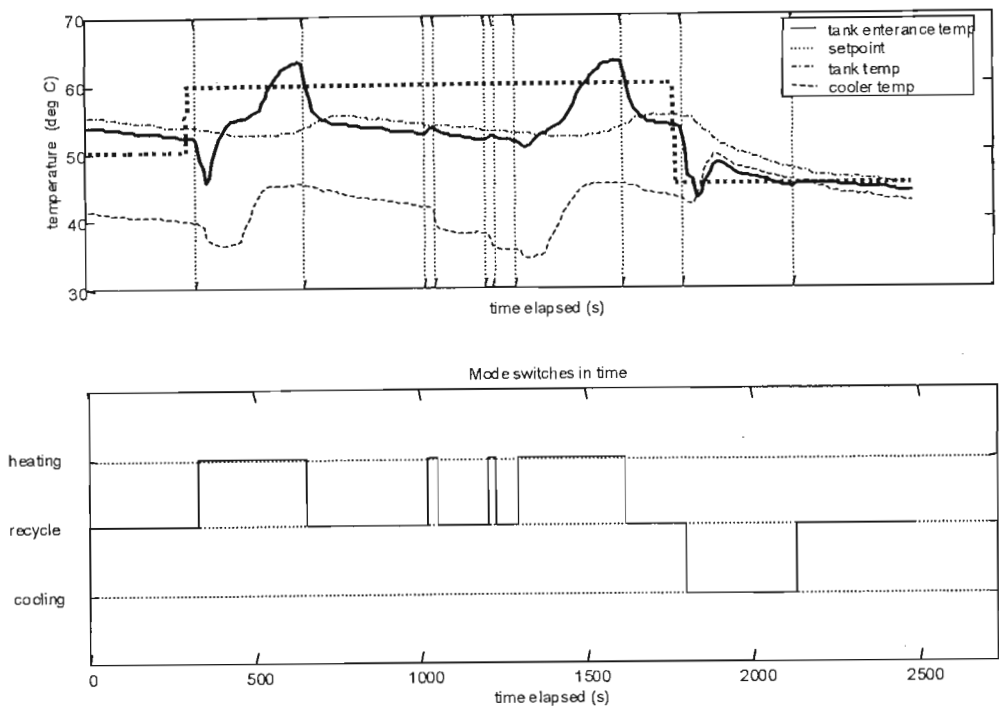


Figure 6.9 Online switching with the controller allowed one switch (control of plant)

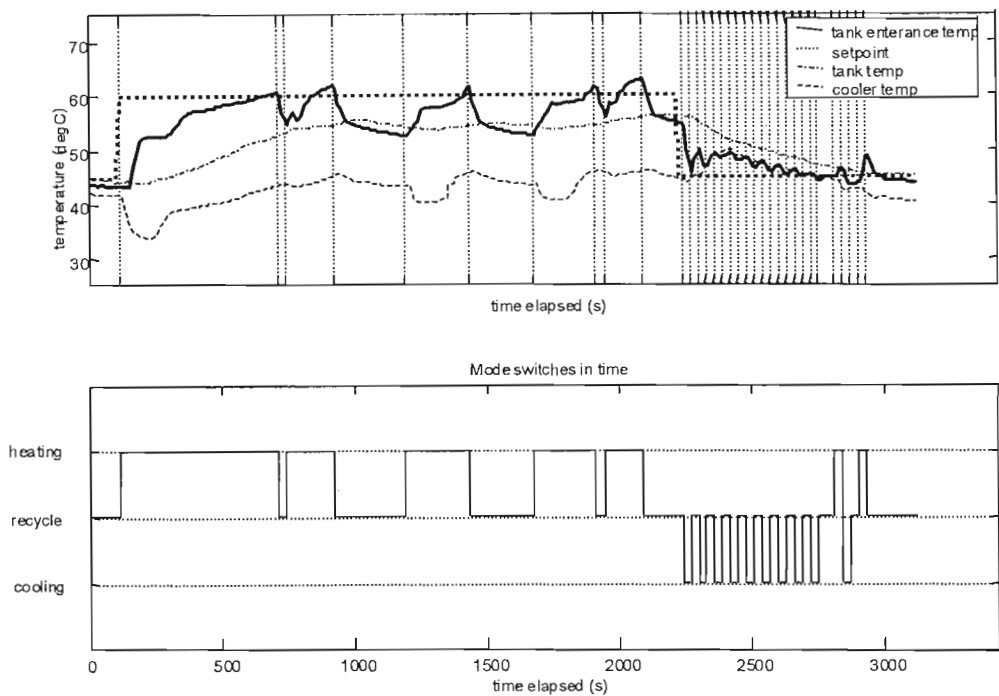


Figure 6.10 Online switching with the controller allowed two switches (control of plant)

For a single optimised move, after switching from heating to recycle, for the higher temperature setpoint, the state experienced an offset from setpoint. This control behaviour, to an extent was similar to that for the simulated case with the symmetric responses. For the plant control, the offset can be

traced back in time to when the switch had initially been made. At that time the algorithm evaluated that switching to recycle was the optimum decision. As usual, optimality was based on the state's closed loop trajectory relative to setpoint. In the recycle mode, when the temperature was sufficiently offset from setpoint due to heat losses, the controller returned the mode of operation to heating. Initially when the temperature was offset from setpoint (e.g. greater than 5 degrees below setpoint), the controller did not switch immediately to heating mode because it did not expect the temperature to keep on falling. This was because it did not have the incentive to act on the future. At the same time the switching penalty would be bigger than the penalty for the offset from setpoint.

When the setpoint was lowered, the controller switched the circuit to cooling mode and thereafter waited until the state was close enough to setpoint before switching back to recycle. Switching from cooling to recycle was initiated at the optimum moment in time, with setpoint thereafter being followed closely. Both graphs 6.9 and 6.10 show that the algorithm found it much easier to track a temperature decrease in setpoint rather than a temperature increase in setpoint. This is because there is a more elaborate dynamic response while in heating mode as compared to cooling mode (see figure 6.4). Also the rate of heating was faster than the rate of cooling, which directly affects ΔB of the respective mode in the integrating model.

With the two-switch optimisation, as with the simulated test, the controller took a greater initiative in regularly switching the mode of operation. For the higher setpoint temperature, the state response was still oscillatory, but the amplitude about the setpoint was not as pronounced as for a single switch. There was a greater effort by the controller to keep the state closer to setpoint. This effect was particularly noticeable when the setpoint was decreased.

6.3.3 Simulated offline control

The switch response data used for the online controller was used to commission a convolution model, to be run in closed loop with the controller. In this control loop, the controller initiated the switch, with the model directly accepting this switch and generating the response. For the previous simulations with the symmetric responses, the controller initiated the switch, but converted it internally to primary inputs (valve positions) before passing it to the model. In this case the model generated responses based on a change in primary input. For the offline tests presented in this section, the model simulated the actual plant nonsymmetrical behaviour.

6.3.3.1 Number of optimised moves

Figures 6.11 and 6.12 validate the findings of the previous tests where the performance of the controller was evaluated with 1 and 2 optimised switches. These responses were comparable to the actual plant responses in figures 6.9 and 6.10 thus showing a satisfactory match between plant and model. For these offline tests, the final response of the state in recycle mode however went to steady state. For the actual plant this was not the case due to heat losses.

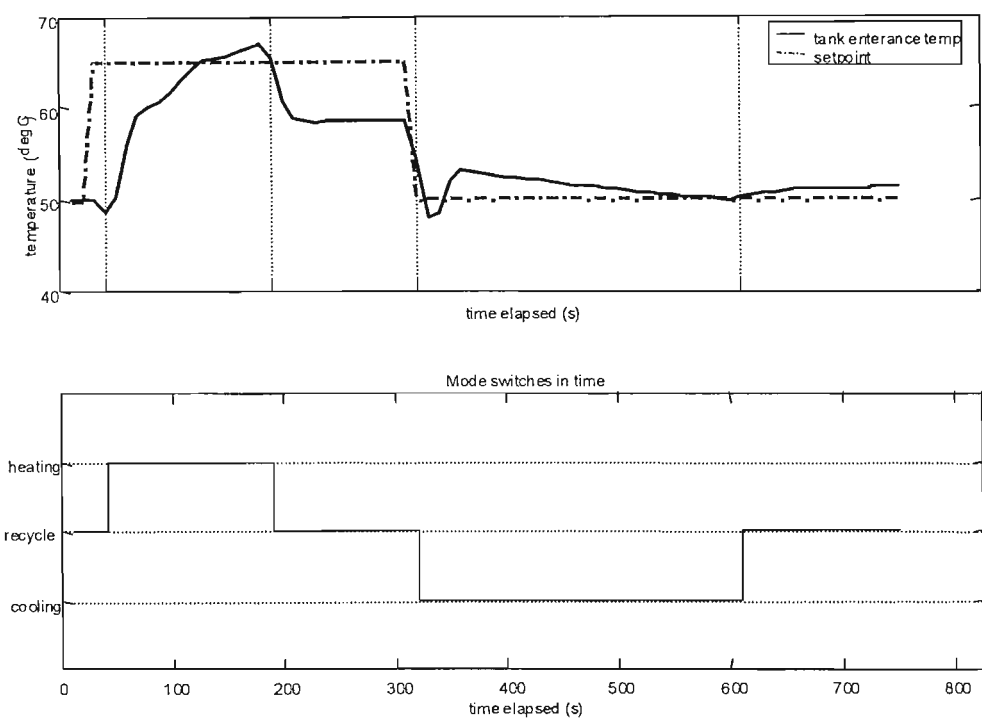


Figure 6.11 Simulated control switching with 1 optimised switch (control of asymmetric model)

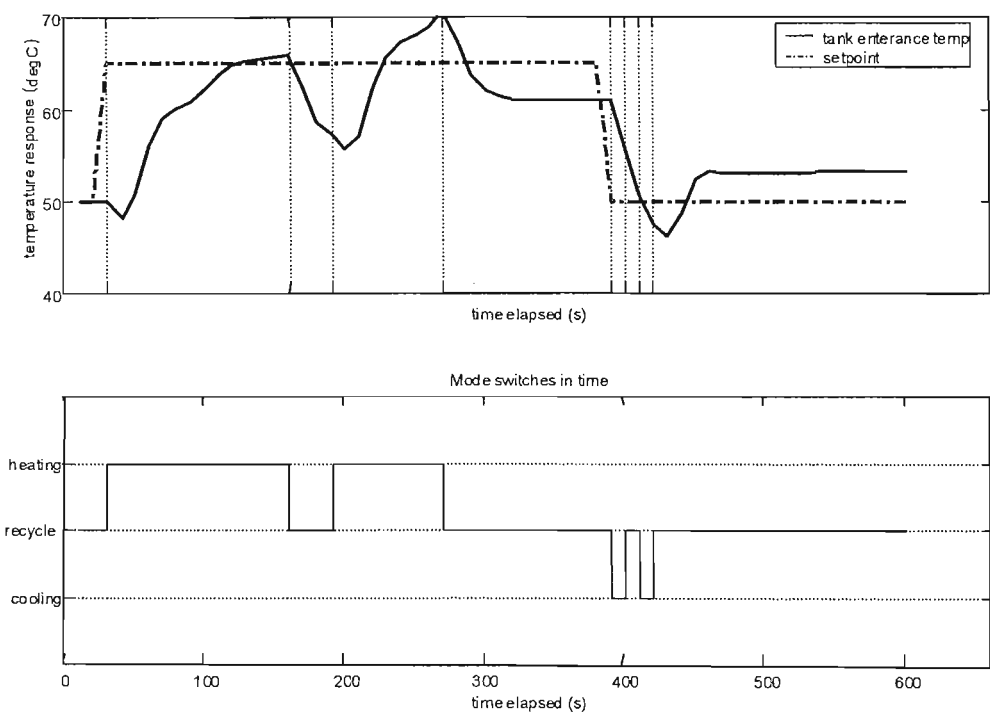


Figure 6.12 Simulated control switching with 2 optimised switches (control of asymmetric model)

6.3.3.2 Constrained Control

Very often, optimal operation for a particular system is close to an operating constraint. MPC can be used in this instance to drive the operation of a system as close as possible to the constraint, by setting the setpoint at the constraint. This section presents the controller, with various configurations, selecting the appropriate mode of operation in order to keep entrance tank temperature (as recorded by *T31*) as close as possible to 65°C, but not allowing it to go above. The simulation model with asymmetric responses was used in closed loop with the controller.

Figures 6.13 and 6.14 show attempts at driving the state as close as possible to the upper bound constraint using different control horizon sizes. Both controller configurations kept the temperature within the constraint, however the controller with 4 optimised switches allowed the system to operate much closer to the constraint. Allowing the algorithm more optimised switches made operation closer to the constraint possible because the extra switches could be used to prevent constraint violation in the closed loop horizon. In the case of two optimised switches, the controller could not allow operation close to the constraint, as it only had a single switch, after the first one, to prevent violation.

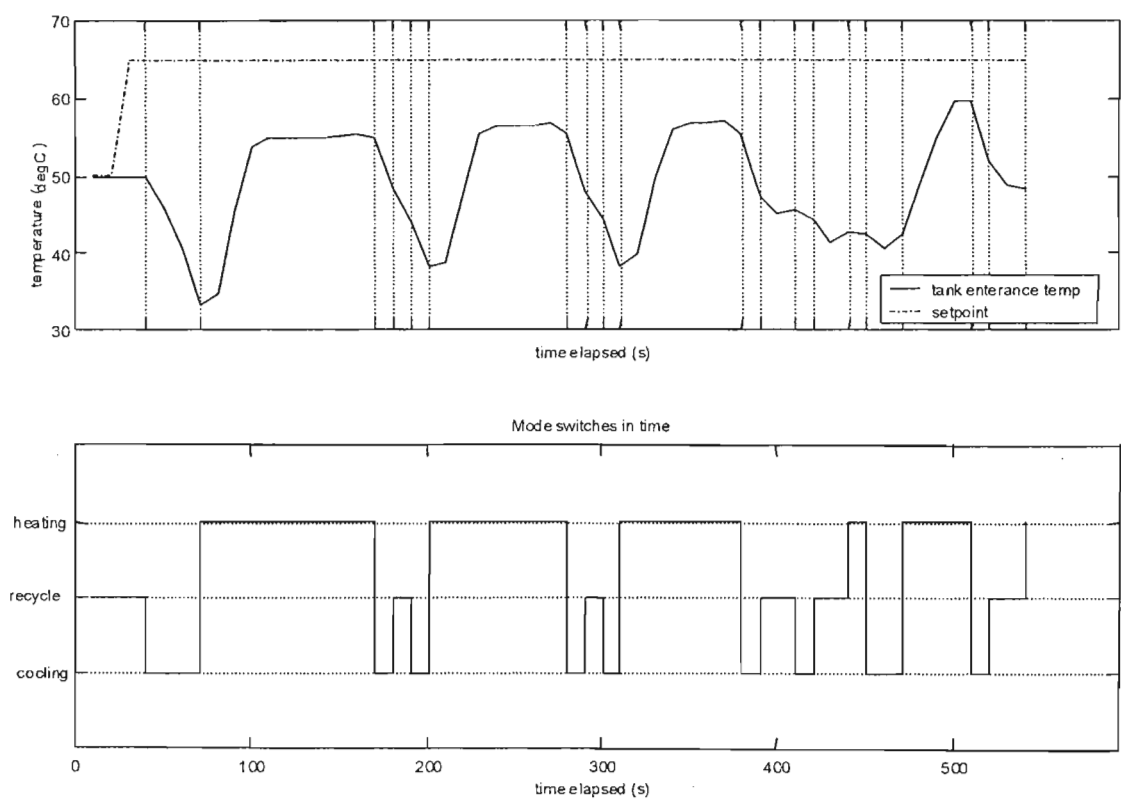


Figure 6.13 Constrained control switching with 2 optimised switches (control of asymmetric model)

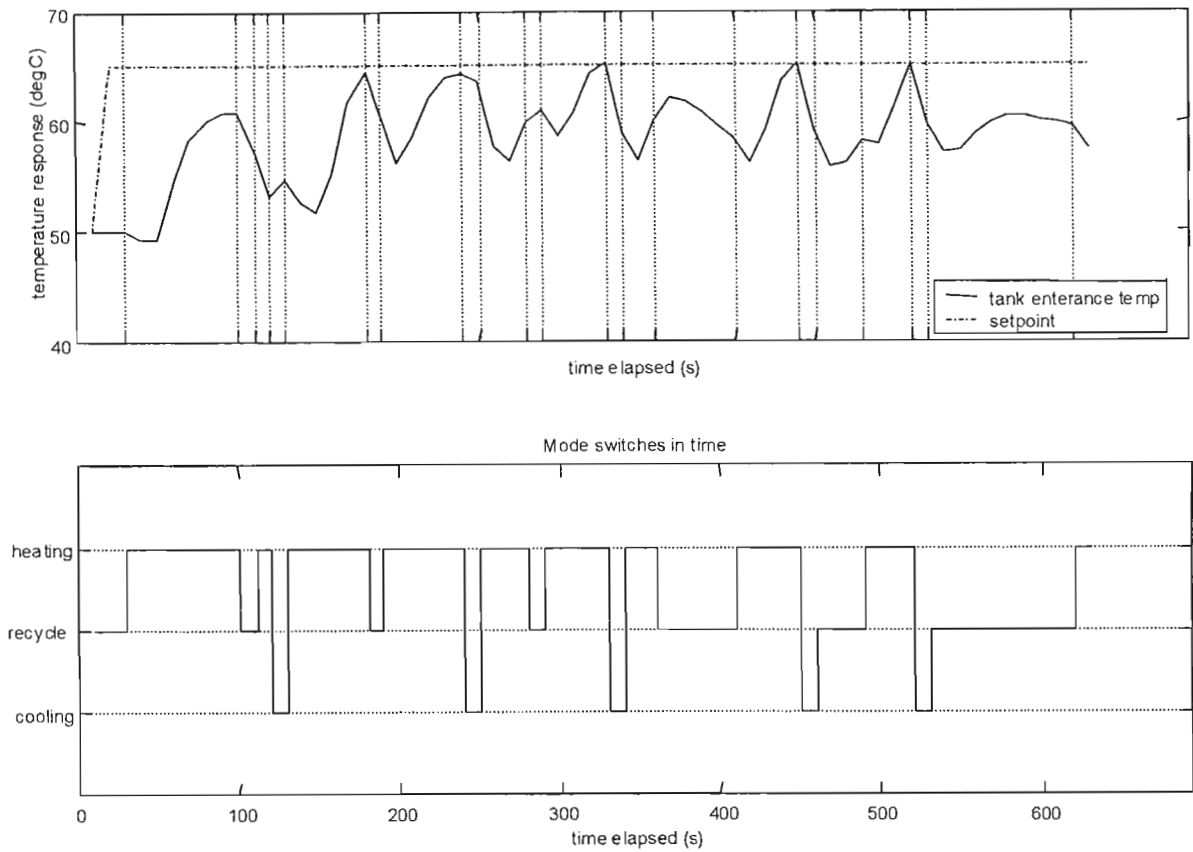


Figure 6.14 Constrained control switching with 4 optimised switches
(control of asymmetric model)

Figure 6.15 shows the controller attempting to do the same as in the above two tests, but in this instance, the constraint was soft in nature. (See section 7.4.3 for the reasoning behind and implementation of a soft constraint). A soft constraint is one that can be violated during the optimisation, but at some cost. The controller used in figure 6.15 was allowed 2 optimised switches. The operation was reasonably close to the constraint, which was clearly superior to that in figure 6.13, where the controller was also allowed 2 optimised switches and the constraint was hard in nature. In order to get the desired control effect, tuning between the setpoint penalty and soft constraint violation was important. For the present simulation, the weight penalisation from setpoint (W) was set at 20 and the soft constraint penalties (on square of positive excess) for the initial points were set at 10, while the final points in the horizon were heavily penalised. (Recall that the weights have a square effect in objective function). The algorithm was given the opportunity to drive the state close to the set-point constraint, by allowing it to violate the constraint to a certain degree, for the initial few steps, in the closed loop horizon. With only 2 optimised switches, not much correction could be done after the first move, so much of the closed loop trajectory, as predicted by the internal model, was due to the first switch. As a result, in figure 6.15, the state never violated the constraint.

In certain instances, the algorithm allowed the temperature to approach the setpoint/constraint more closely (180s and 230s), while on other occasions (280s and 370s), switches were made much earlier. This optimal time of switching depended on phasing, as explained earlier in section 6.3.1, and also the future prediction.

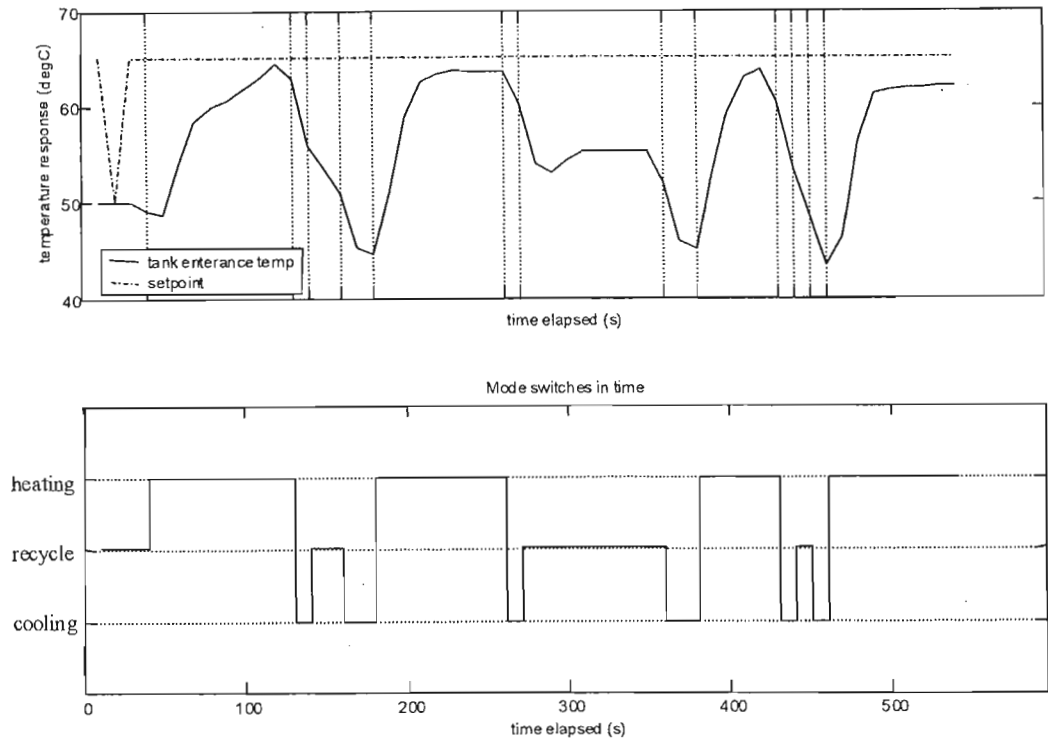


Figure 6.15 Constraint switching with soft constraint and 2 optimised switches

6.4 Conclusion

A model predictive controller was designed and commissioned to automate the switching of a thermal circuit with modes. System modes were defined by combining specific permutations of the discontinuities in inputs. The controller, based on the DMC framework, was modified to deal with real time mode switching of the thermal circuit. By using openloop switch response data of the system, the internal convolution model of the controller was generated. The controller was shown to make logical switching decisions regarding the mode of the system when meeting operating objectives.

Switches between successive modes resulted in asymmetric responses regarding the state. This phenomenon was dealt with by getting the algorithm to use the responses in direction for which they were generated. Although the model could not account for temperature changes due to heat losses, measured online and simulated offline closed loop tests showed good reconciliation between asymmetric model simulations and the actual plant. Finally, using the mixed integer predictive controller with variations in switching horizon and nature of constraints, it was shown how a system, for the purpose of optimality, could be directed to function close to an operating constraint.

CHAPTER 7

Industrial Application: Coal lock sequencing

During the course of the research, an industrial application that required the real time sequencing of events presented itself. It was thought at the time that this was an ideal opportunity to test the mixed integer predictive controller on a much larger scale. The industrial system exhibited typical hybrid system characteristics in that the sequencing dealt with both continuous aspects and discrete event logic. The controlled variable was periodic, with the controller having to initiate the start to the cycle.

This chapter presents an attempt at using model predictive control to sequence coal locks. Since the actual dynamics of the process were much more complex than expected, the attempt at using the MIPC to sequence the coal locks, was not pursued any further than the offline simulation stage. The complexity lay in the fact that responses were non-linear with the base of the response being fixed i.e. temperatures always started at the coal feed temperature, regardless of the final temperature in a cycle. Even then, simplified responses were used as step response data for DMC. For the real plant, DMC is in fact not a viable solution. DMC was used however as a preliminary investigation into the sequencing characteristics of model predictive controllers. The control problem required an optimisation procedure and the sequencing of events did require integer variables. With this in mind, the control algorithm used for the previous case studies was modified to deal with coal lock sequencing.

This chapter commences with an overview of the process. Thereafter a brief description of the control strategy presently used in industry is presented. Following this is an explanation of how MPC can be applied to the problem. Finally, presentations of offline tests are made using idealised responses, which illustrate the proposed control strategy. The chapter concludes with an overview of the findings.

7.1 Process Overview

SASOL is a petro-chemical company based in South Africa, whose major interest is the conversion of coal to higher valued synthetic fuels and chemicals. The fundamental part of this conversion process is gasification, where coal is gasified to form carbon monoxide and hydrogen, which in turn is converted to higher hydrocarbons through Fischer Tropsch technology. Sasolburg, one of the two major SASOL plants in South Africa, has 17 gasifiers operating in parallel, each producing on average 50 000 Nm³/hr of raw gas. The gasification process is essentially a chemical reaction, with the conversion of coal to raw gas occurring at a high temperature (400°C) and pressure (25 bar) in the

presence of an agent. The gasification agent is an oxygen/steam mix, set at a specified ratio as required by the raw gas composition.

Although the production of gas is a continuous process, coal is added batchwise to the gasifier, which is of the LURGI type (figure 7.1). Each gasifier unit is fitted with a coal lock that provides fresh coal to the gasifier. Gas is removed from the circuit downstream in order to pressurise the coal locks, so that the pressure within the lock is at the same pressure as the gasifier before coal enters the gasifier. In order to recharge the coal lock, the lock decompresses by releasing gas into a CLEG (coal lock expansion gas) network that processes the gas and returns it to the network (figure 7.2).

The single CLEG however does not have the capacity to deal with more than a few of the 17 coal locks decompressing at once. As a result, decompressing of the coal locks has to be staggered, so as not to overload the CLEG. If the CLEG holding capacity does get exceeded, then excess gas gets flared. Since this gas is taken from the raw gas line coming off the gasifiers, flaring gas registers as a loss to the process. A control strategy is thus required that sequences the decompression of the coal locks in time, so as not to overload the CLEG and thereby minimise flaring.

This system exhibits hybrid system behaviour. The decision to give a coal lock a “cycle start” is a discrete event. At the same time other process variables have to be monitored that directly affect the decision i.e. current status of the CLEG and the temperature within the coal lock, which is used to infer coal level in the gasifier.

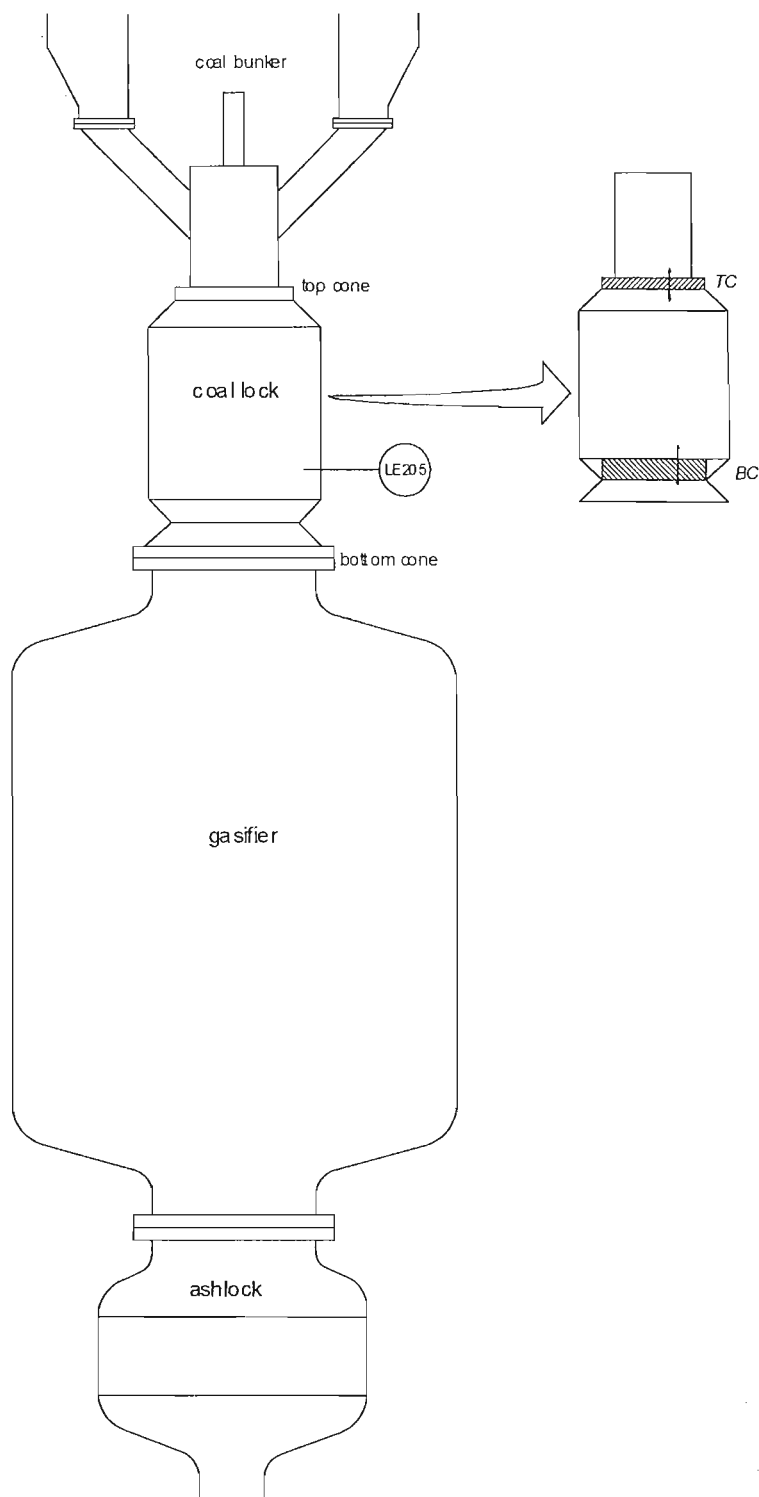


Figure 7.1 Single gasifer with coal / ash locks and schematic showing functioning of cones

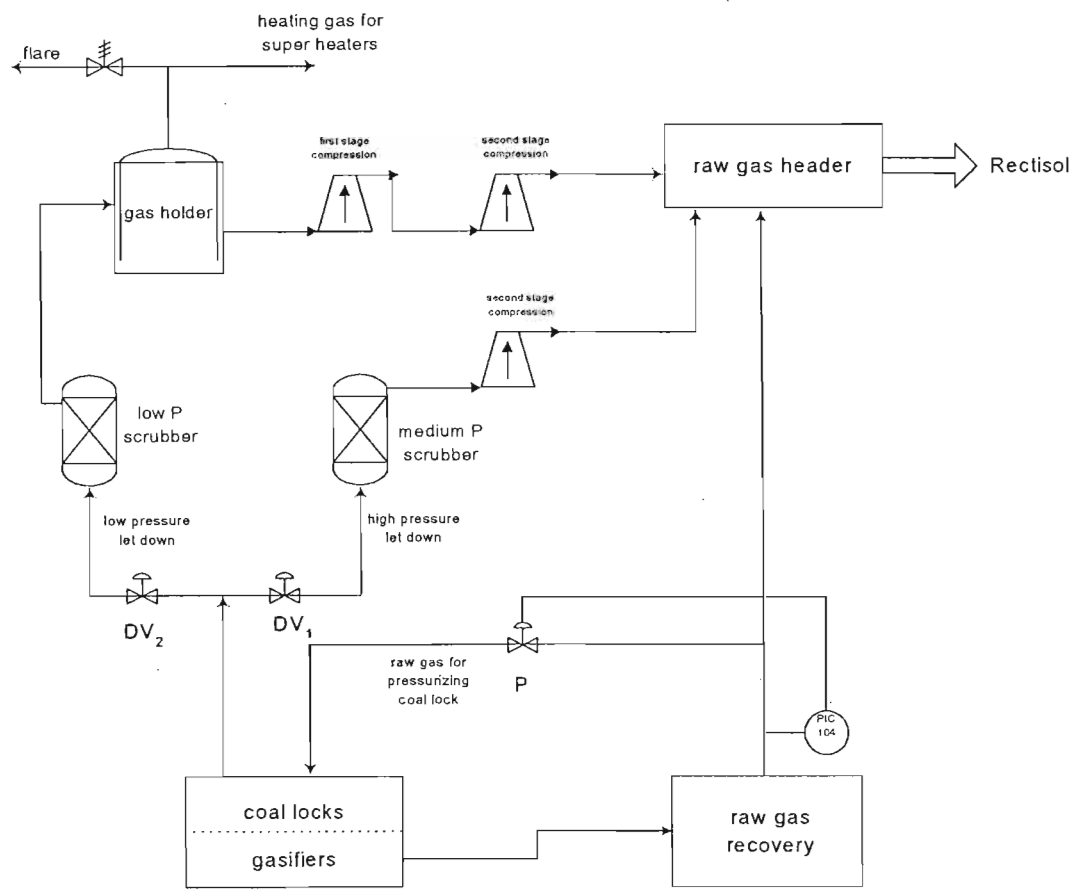


Figure 7.2 Gas circuitry showing CLEG system

7.2 Current control strategy

Coal locks are given a “cycle start” by the controller when the gasifier is empty and therefore needs to be refilled. Since there is no direct way of measuring the level in the gasifier, temperature sensor LE205 (measuring temperature in the coal lock) is used to infer the coal level in the gasifier. When the gasifier level is low (i.e. level nearing the bottom of the gasifier), the temperature within the coal lock rises because hot gases directly enter the coal lock from the gasifier. The current control strategy thus uses temperature LE205 as a process variable to initiate a coal lock cycle.

Figure 7.3 shows the temperature response as detected by LE205 during a typical cycle for a single gasifier. The second plot (PI204) is the pressure within the coal lock. Before a coal lock is given a cycle start, the bottom cone (BC) is opened and the pressure within the lock is that of the gasifier. When the level starts to decrease, temperature within the lock starts to increase. Each gasifier has a setpoint for LE205, at which the coal lock is given a cycle start. When LE205 for a particular gasifier reaches its setpoint, that gasifier is given a cycle start.

The cycle starts by isolating the coal lock by closing the bottom cone. Thereafter the coal lock is depressurised (through valves DV₁ and DV₂), with the gas passing to the CLEG. When the pressure of the coal lock is atmospheric, the top cone (TC) is opened and the coal lock is filled with fresh coal

from the bunker above. When the TC closes, the coal lock is repressurised (through valve PV_1) to the pressure in the gasifier and on a timed event, the BC is opened, thus refilling the gasifier. Notice that when the controller initiates the cycle, the temperature does not begin to decrease until the second stage of decompression, when the air in the lock expands. The temperature further decreases when coal is added through the TC. It only starts to increase when the level in the gasifier drops.

In an attempt to prevent the CLEG from being overloaded by many coal locks depressurizing at once, each gasifier could be given a “forced” cycle start if it meets certain conditions. Usually coal locks are given a cycle start at setpoint, but when they are given a forced start, the temperature within the lock lies within a certain margin below setpoint. This is an attempt to stagger the coal lock events, so that they do not all depressurise at once. Figure 7.4 shows the operation of 3 gasifiers in terms of their respective coal lock temperatures. Depending on certain real time criteria, at certain points they are given a forced start, while at others the cycle is started at setpoint.

For the present cycle, an average of the last three cycles (from BC closed until BC closed again for the next cycle) is taken (τ_{est}) and this is set as a setpoint on an indicator (TM_PV). The current cycle time is used as a process variable on this indicator. Simultaneously each gasifier is allocated a weight, which is a representation of its impact on the CLEG e.g. gasifiers 1-13 each have a weight of 1, while gasifiers 14-17 have a weight of 3 because of the size of their coal locks. When considering which gasifiers to give a cycle start to, the controller sums up these weights and displays it as a second indicator DEPRESS1_17.

A coal lock will be given a forced cycle start if the following conditions are met:

- The PV of TM_PV is less than 40 seconds from its SP.
- The PV of DEPRESS1_17 must be less than 3.
- LE205 for the gasifier in question must be less than 8 °C from its setpoint.

A cycle start for a gasifier is triggered if:

1. Start requested from operator.
2. LE205 = setpoint.
3. LE205 is high.
4. Given a forced cycle start.

Notice that the main event in the control procedure is the initiation of the cycle. Thereafter all other events are timed and predetermined i.e. depressurization; filling of the coal lock; repressurization and finally opening of the bottom cone.

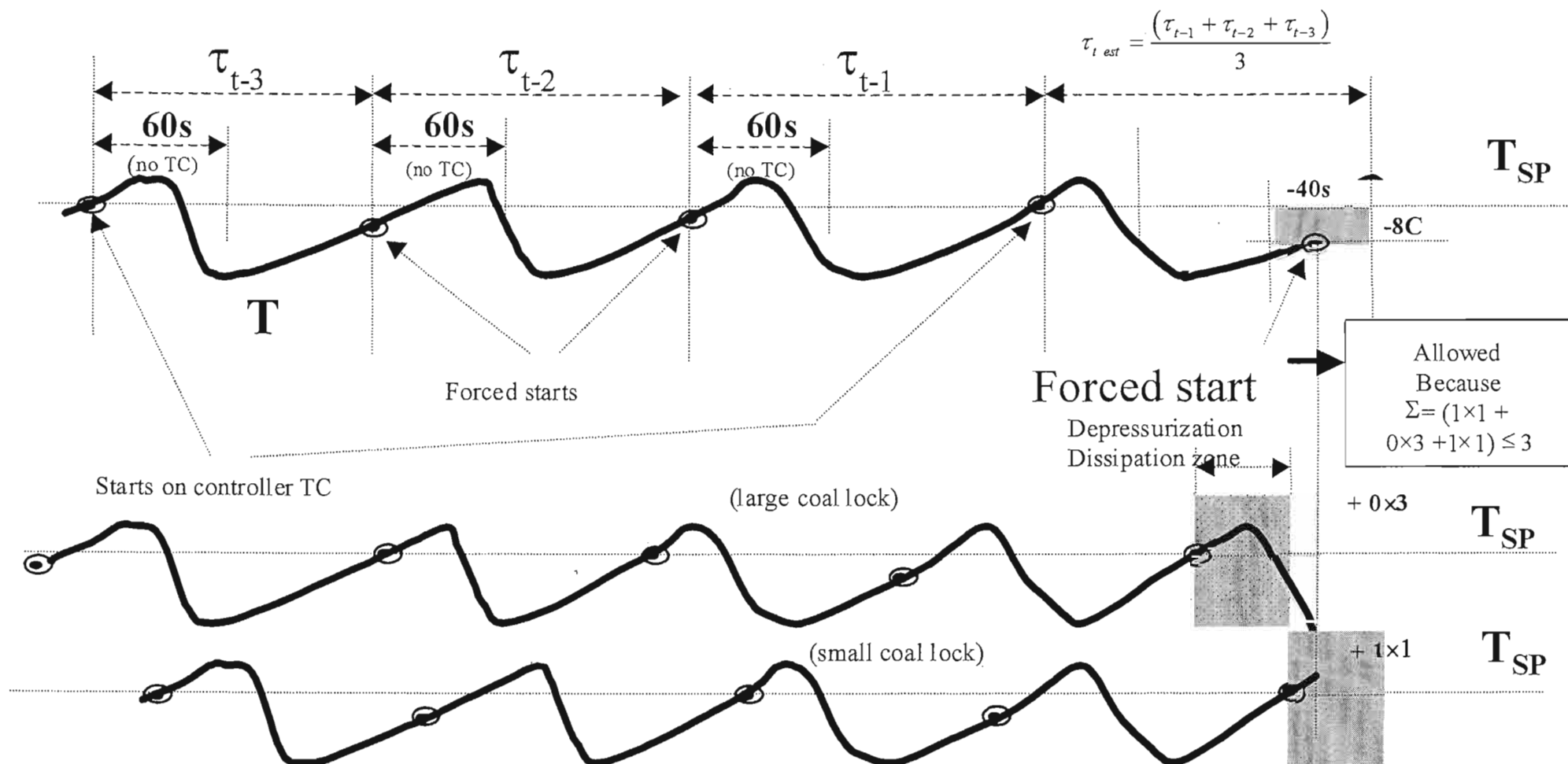


Figure 7.4 LE205 response for coal locks getting forced or setpoint starts

7.3 Sequencing using MPC

The previous section shows that the control personnel at SASOL have undertaken a heuristic approach in dealing with the sequencing of the coal locks. Sequencing is done using timing procedures, together with the relative position of the state with respect to setpoint. It seems to work well at the plant, but the operators agree that there is ample room for improvement.

Approaching this problem from a theoretical, rather than from a heuristic outlook, suggests that this is in fact an optimisation problem. The controller has to prioritise the sequence of events while making certain compromises. It has to make decisions based on which gasifier most urgently requires refilling, while at the same time protecting the CLEG. A model predictive approach is appropriate as it allows for planning into the future, which is useful when dealing with the CLEG. For example the controller can “see” ahead and based on knowing when the CLEG would have a low flow of gas through it, it could initiate events presently. Or if it “knows” that in the future the CLEG might become overloaded, then it could delay a cycle start for a specific coal lock. This planning-ahead approach effectively removes the need for forced cycle starts.

As with the case studies, variables had to be identified that drove the optimisation i.e. those, which needed to be controlled. The initiation of the event will essentially be based on these variables. In this case the logical choice was to use the temperature within the coal lock, as recorded by LE205. This would be a direct indication of when the gasifier was empty and it was also used in the heuristic approach to initiate a cycle start. Each coal lock was then assigned a binary variable, whose evaluation at every time step determined whether that particular lock got a cycle start or not.

Table 7.1 shows simply (for 3 gasifiers and a single CLEG) how the problem of optimally sequencing coal locks, was converted into a MPC problem using the DMC framework. The table contains the various responses, with the time at which the response starts, depicted by the circle at the start of the response.

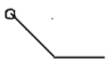

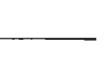




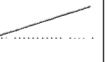




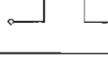
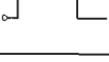
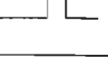
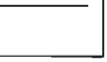
	CL1	CL2	CL3	rate
T1				
T2				
T3				
Fcleg				

Table 7.1 Simplified DMC representation for sequencing coal locks

CL1, CL2 and CL3 are three coal locks to which binary variables are attached which when activated (take on value “1”), trigger a cycle start by bringing about a decrease in their respective temperatures (T1,T2 and T3). One might occur gradually (T1), another instantaneously (T2), and yet another delayed (T3). These responses are selected simply for simulation purposes to ensure that there is diversity in the temperature responses for each coal lock initiation. The decisions that the controller makes must thus be based on an “awareness” of this diversity in these responses and at the same time fulfill constraints that may be violated by its decisions. Another output variable, F_{cleg} , is introduced which represents the extent of interaction between the gasifiers and the CLEG. Each gasifier will have its own impact on the CLEG. F_{cleg} represents the collective impact of all gasifiers on the CLEG. Very simply it can be viewed as the flowrate of gas through the CLEG as discharged from the locks during decompression. When a coal lock is given a cycle start, that particular binary variable will activate a delayed square extended pulse response for the variable F_{cleg} . The response is shaped the way it is; firstly because decompression occurs a while after a coal lock is given a start i.e. that particular coal lock will only impact on the CLEG later on. Secondly the pulse height depicts the extent of impact that a particular coal lock will have on the CLEG. It is maintained over a period of time and then it returns to the position before the cycle was initiated. F_{cleg} was introduced with the intention of placing an upper limit on it, which in turn is a representation in the model for protecting the CLEG. The “rate” input ensures that if no coal lock event was initiated, the respective temperature would increase indefinitely as set by the slope of the line.

These responses are idealised and are different from the actual plant responses (figure 7.3). In reality when a coal lock is given a cycle start, the temperature within the lock will always decrease to roughly the same temperature, before it starts to increase again. This will occur regardless of where the state was prior to initiation. DMC is however based on a floating point of reference. This means that the position of the state, after the initiation, is dependent on the position just prior to the initiation. The effect of the initiated event on the state is fixed. Figure 7.5 shows this shortcoming of using the standard convolution model, to predict the state after the event is initiated.

Therefore to correctly model the trajectory of the temperature, the DMC algorithm would have to be modified, so that the state returns to some base level regardless of where it was prior to switching. Due to the complications it would have introduced into the procedure, no attempt was made to modify the DMC algorithm in this respect. Firstly an approach would have to be devised that made the controller aware of the indefinite increase in temperature when the gasifier was empty. Secondly the temperature must be returned to a base value upon initiation of the cycle. A method of solution, in this case, might be to reset the model every time the event is triggered. This would consequently require removing the effect of all past initiations prior to the new event. Now the problem starts to shift away from being a DMC problem and tends towards a MINLP problem. An initiation for a single gasifier will trigger a certain temperature trajectory into the future, which gets interrupted and repeated on the next initiation. Thus to genuinely solve this problem, this method of approach is recommended i.e. set it out as a problem in MINLP and reset the state every time a cycle is initiated. However for the purpose of investigating the sequencing capabilities of a predictive controller, the problem was dealt

with using DMC. The following tests were based on a floating reference state with the decrease in temperature being fixed every time a cycle was initiated.

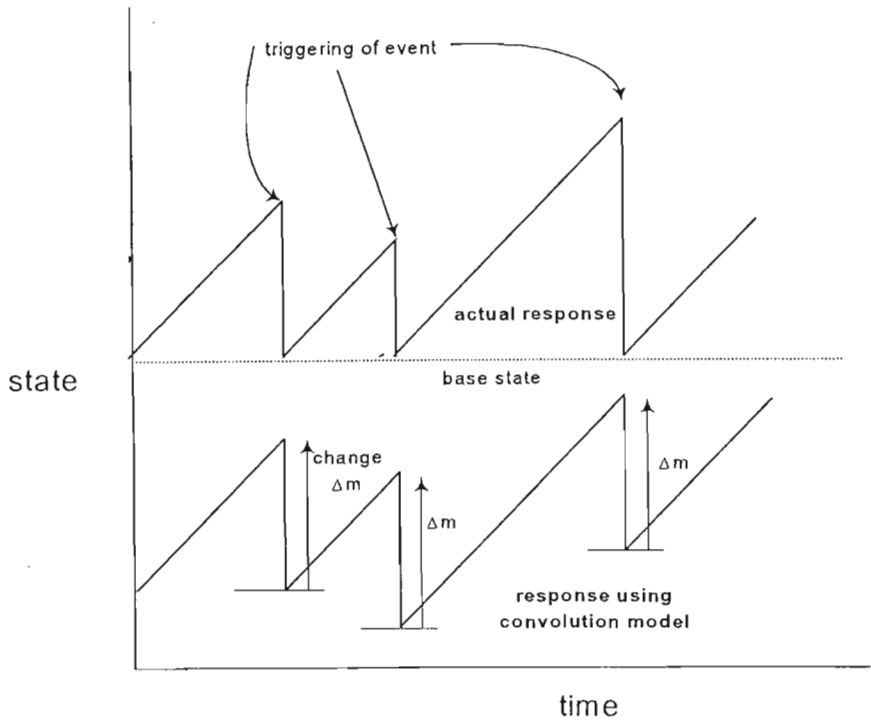


Figure 7.5 Shortcoming in convolution model to predict the actual state trajectory

7.4 Offline tests

To test the proposed control strategy, tests were undertaken with 3 gasifiers, functioning in parallel, using the SCAD-GAMS control loop presented in section 3.3. SCAD simulated the response of the temperatures within the coal locks using a convolution model. Temperatures are allowed to increase at a constant rate until a coal lock event is initiated and the temperature thereafter decreases gradually. After a while, the temperature starts to increase again, depicting a decrease in level within the gasifier. Temperature response is thus cyclic. A period is from when it first decreases, to when it starts to decrease again, as triggered by another cycle start. By using different responses for the coal locks, all three temperatures were simulated with different rates of increase and decrease. This also ensured that their cyclic periods were different. The internal model of the model predictive controller was commissioned with the same step responses. Refer to appendix D1 for the step response data used for the convolution model.

The response of F_{cleg} can be used as an indicator for the sequencing capability of the controller. The pulse height for each coal lock is: $[CL1 \ CL2 \ CL3] = [2 \ 3 \ 7]$. Therefore if there are spikes in the trajectory of F_{cleg} , then it is an indication that the coal locks are not being sequenced optimally and that the CLEG is overloaded by locks that depressurise at the same time. However a stable F_{cleg} response is an indication of optimal sequencing, illustrating that the controller has staggered the cycle

starts of the different coal locks. Cycle starts for certain locks are held back until such time that the CLEG is able to deal with the decompression.

7.4.1 Sequencing based on setpoint

As mentioned previously, temperature control was to be used to drive the sequencing procedure. Essentially, for each gasifier, the controller uses the rise in temperature to trigger a cycle start. Figure 7.6 shows a control sequence based on just setpoint penalisation. No constraints or penalties were placed on Fcleg.

There were sudden increases in the response of Fcleg (150s, 450s and 950s), indicating that there were times when the controller gave all three coal locks a cycle start at once. Hence control based on just setpoint penalty is not good enough for sequencing events. Such an outcome was however expected, as successful sequencing must be based on an “awareness” of the entire operation and not just one state. In this case the optimisation was based just on temperature only.

Temperature responses for the three coal locks were periodic in nature about their setpoint. This is another indication of an optimisation procedure based on just setpoint, where there is no interaction with other process variables. This test suggests that further penalties and even constraints were needed, in order to get the optimisation to sequence the coal locks successfully.

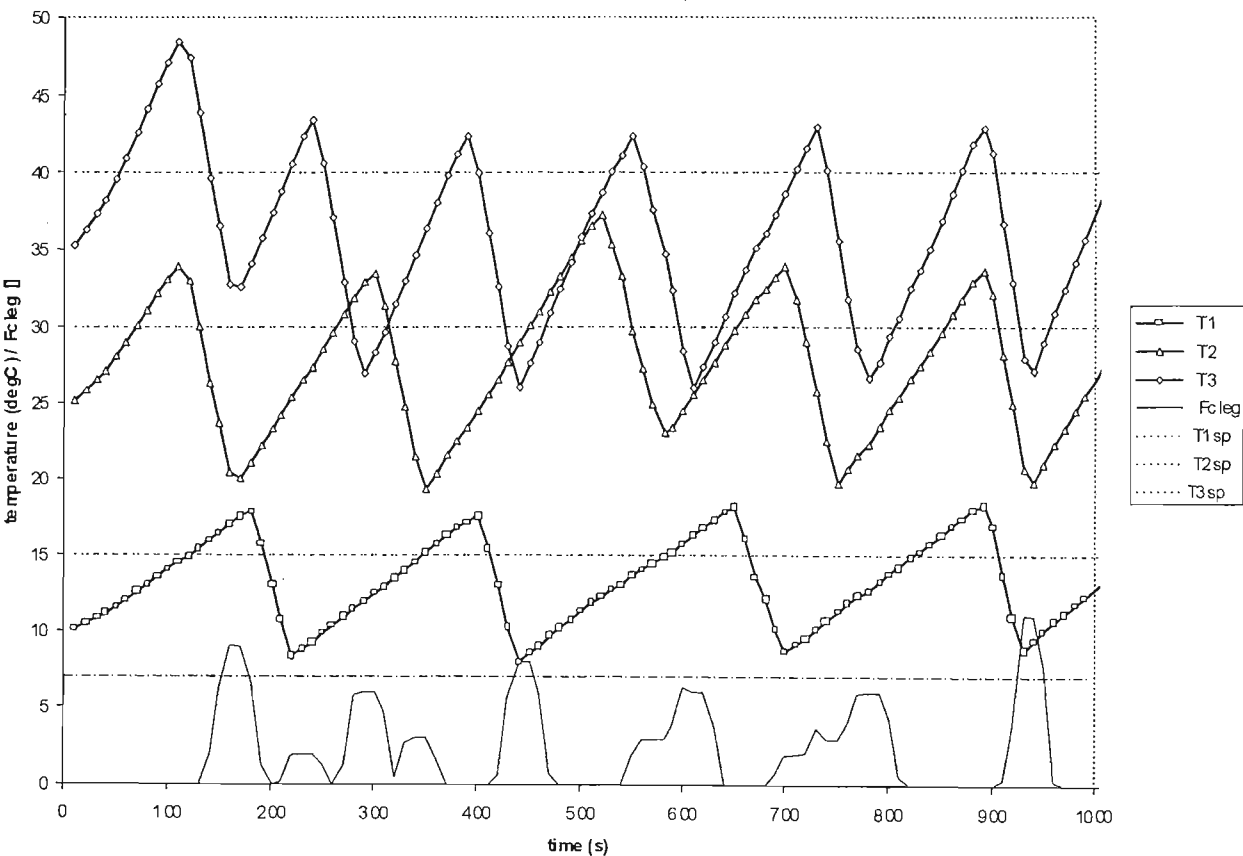


Figure 7.6 Sequencing using MPC based on setpoint penalisation only

7.4.2 Sequencing based on Tuning

Part of the optimisation procedure was to protect the CLEG. This was not done in the previous test, with the result that there was no sense of sequencing. For this test, by penalising Fcleg every time it deviated from zero, the optimisation was given a reason to protect the CLEG. The penalty on Fcleg was set higher than the weights on the temperatures ($W_{Fcleg}=20$; $W_{T1}=W_{T2}=W_{T3}=1$). Move suppression was kept at 1 for all lock cycle initiations. The optimisation was allowed only one optimised move, which was the case for all sequencing simulations undertaken. It was expected that the introduction of Fcleg into the optimisation, with the appropriate tuning, would aid in the sequencing of the coal locks.

Figure 7.7 shows the controller, tuned with the intent of sequencing the coal locks. Unlike figure 7.6 the temperature responses do not have fixed periods, indicating that there was another variable of concern in the optimisation, which in this case was Fcleg. Penalising Fcleg with the correct weight ensured that at no point in the control procedure did Fcleg exceed 7. The relative weights between Fcleg and the temperatures led to the controller sequencing the coal locks while protecting the CLEG. The sequencing procedure was based on a gasifier only being given a cycle start when its openloop temperature error exceeded that error which Fcleg would endure if that specific coal lock were given a cycle start.

Although appropriate tuning of the controller proved to be successful in sequencing the 3 coal locks, it did have one flaw. This was illustrated in the response of Fcleg in figure 7.7. After a coal lock was given a cycle start, the controller often waited for Fcleg to return to 0, before it gave another lock a start. It did not take advantage of the delay brought about when a lock was given a cycle start and when it first started to have its effect on Fcleg. This is understandable as the optimisation is based on minimising the deviation of Fcleg from setpoint (0). A minimum is obtained when one lock was given a cycle start after the previous one had completed having its effect on the Fcleg.

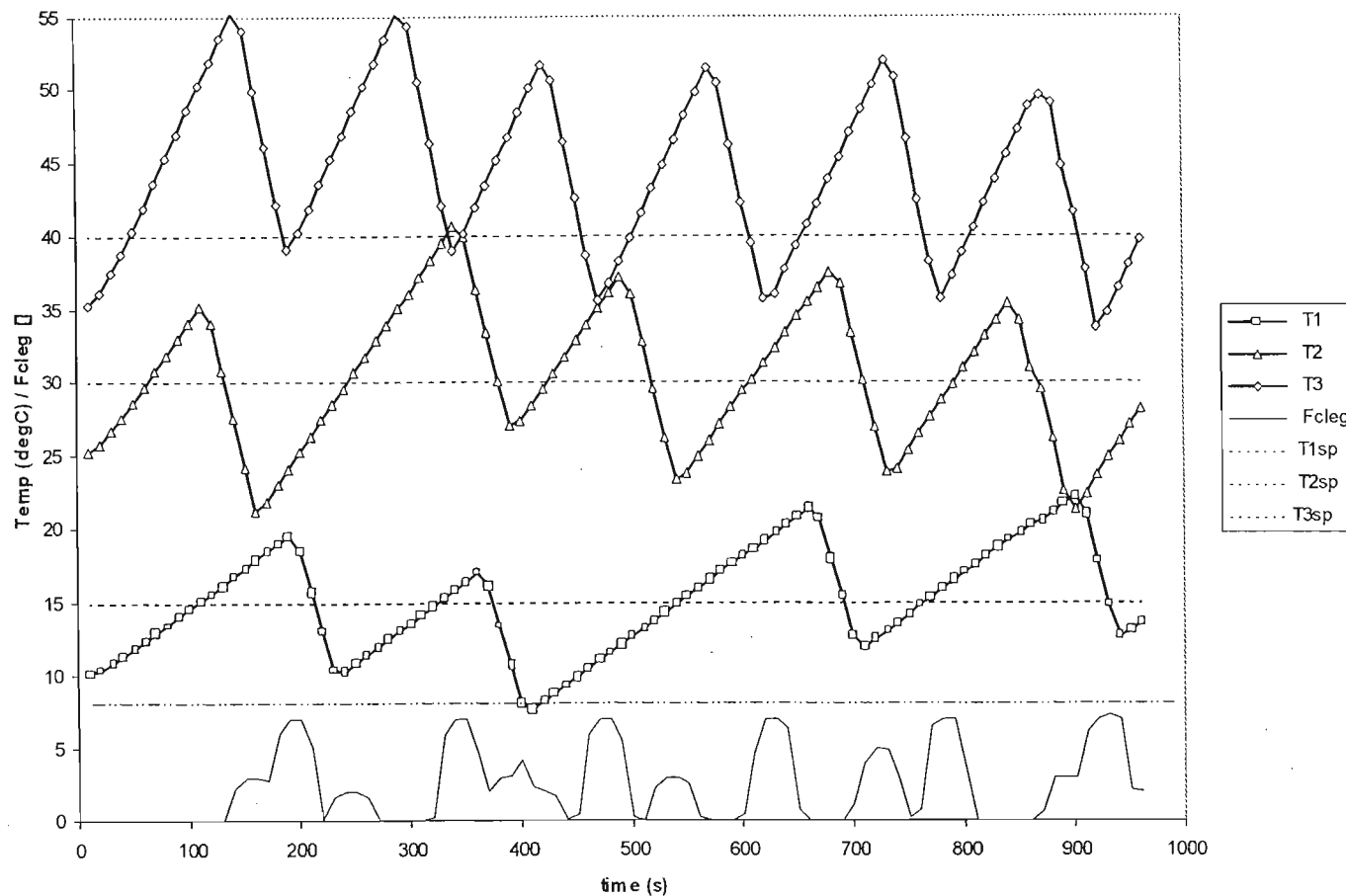


Figure 7.7 Coal lock sequencing using tuning

7.4.3 Introduction of soft constraint

Besides using tuning to get the controller to sequence coal locks, another feature of the MPC strategy that could be used is constraint handling. The main aim was protection of the CLEG, so it was expected that an upper limit constraint on Fcleg would suffice and thus cause the controller to hold back lock events so as not to overload the CLEG.

This strategy was theoretically correct, but was initially difficult to implement. The problem was that the constraint was a “less than or equal to” equation and therefore “hard” in nature. The optimiser, in searching for an optimum solution, will not violate this constraint at any cost. In real time, when the controller takes the decision to give coal locks a cycle start, it does so on the basis that Fcleg lies within the hard constraint. However when time moves on and the effect of the cycle start on Fcleg starts to come through, the constraint becomes violated in the future horizon. Usually this violation is far down in the optimisation horizon and an algorithm with only binary selections available to it, is unable to bring it back into the operability region. As a result it crashes with the report: “no integer solution.”

The question arises as to what would cause a constraint to be violated in time, when initially the move was made on the basis that it was not violated. The obvious answer is plant model mismatch. Here the controlled variable of the plant does not change according to the model prediction. So when the state is fed back at each time step in the form of x_{OMeas} (equation 3.3), the effect of the past moves cause it to violate the constraint if the present x_{OMeas} is different to that initially predicted by the model. Figure 7.8 illustrates graphically how a constraint can become violated in the future due to plant model mismatch.

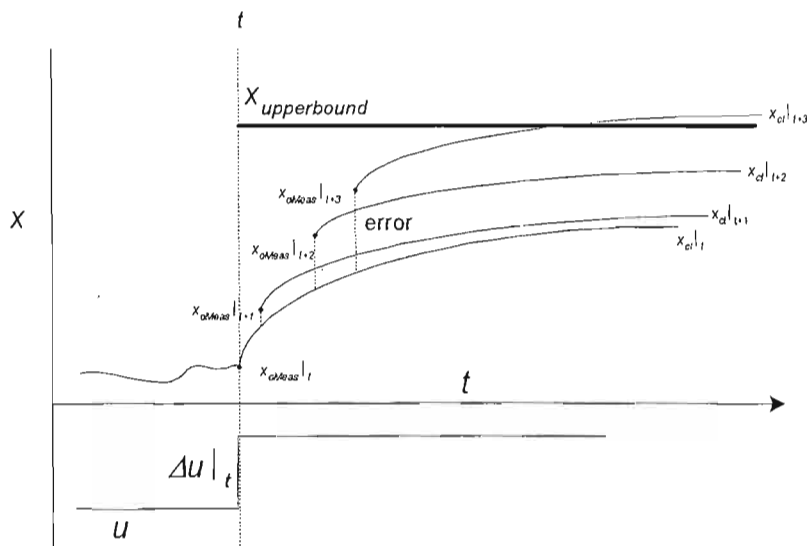


Figure 7.8 Violation of hard constraint due to plant model mismatch

This problem was first experienced in the SCAD-GAMS control loop, where identical convolution models were used for the internal controller model and the external model. Model and internal control model mismatch was therefore not the cause of the problem. What caused the violation of the constraint was the timing between controller and model. Because the model interpolated between the points as given by the step responses, it was possible for the controller and model to be offset when the controller intervened. Over a period of time this offset could cause a closed loop prediction too fall on the wrong side of a constraint.

In hybrid system control this is a problem of real concern. It means an important feature of MPC i.e. constraint handling cannot be used as easily for the control of hybrid systems. The core of the problem lies in the integer nature of inputs. For continuous inputs, this problem would not be apparent as the inputs could be freely evaluated to redirect the outputs back into the feasible region. However in the case when there are binary inputs and basically only two options available to the algorithm to rescue the computation, neither might prove to be sufficient. The result is that the computation fails and the continuity in the control loop is broken.

One way to work around this problem is to convert the hard constraint into a soft one. This allows the computation to be aware of the constraint and at the same time prevent the algorithm from crashing if for any of the above reasons the constraint is violated.

The following explanation, together with figure 7.9, demonstrates how a hard constraint can be converted to a soft constraint. If x_{hi} and x_{lo} are the hard upper and lower bound constraints for controlled variable x , then they can be converted to soft constraints, at every point on the trajectory i , by defining new variables $s_{hi} | _i$ and $s_{lo} | _i$. Where

$$s_{hi} | _i = \max[(x_i - x_{hi}), 0] \quad (7.1)$$

$$s_{lo} | _i = \max[(x_{lo} - x_i), 0] \quad (7.2)$$

Soft variables $s_{hi} | _i$ and $s_{lo} | _i$ can be made part of the optimisation procedure by including them in the objective function as squared sums. Notice that if x remains within bounds, s_{hi} and s_{lo} do not contribute to the objective function. They only do so when x is out of bounds; even then its contribution depends on the magnitude of violation.

This new addition to the optimisation procedure is sufficient to keep x within bounds and even when it does exceed, the computation does not fail. If x were to lie outside of its bounds in the closed loop response due to slight plant model mismatch or timing of when the controller intervenes, then s_{hi} or s_{lo} is usually relatively small. As a result they will contribute little to the objective function. In this case it is serving the purpose of protecting the algorithm from failing. On the other hand by multiplying $s_{hi} | _i$ or $s_{lo} | _i$ by a suitably high weighting factor, x can be kept within bounds.

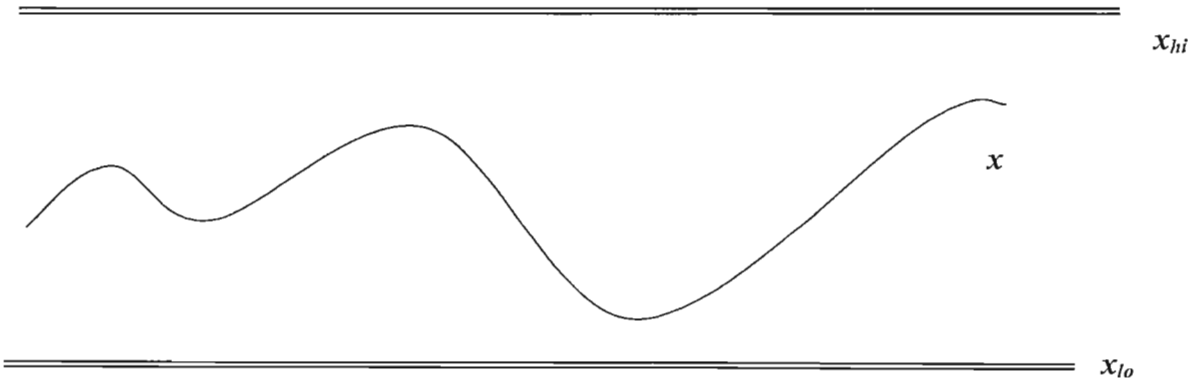


Figure 7.9 Hard constraints on controlled variable x

Fortunately the discontinuous \max function in equations 7.1 and 7.2 can be included in the equation block in GAMS (see appendix D2). Figure 7.10 shows the effect of the soft constraint on the sequencing problem when an upper bound of 5 was placed on Fcleg. Recall that CL3 had a relative contribution of 7 to Fcleg when it was given a cycle start.

Initially the controller delayed the cycle start for CL3 because of the soft constraint on Fcleg. However when the error due to the offset from setpoint, exceeded that placed on the soft constraint in

the objective function, then CL3 was given a cycle start. Also when CL3 decompressed, it always did so alone, therefore the maximum that F_{cleg} went up to was 7. CL2 and CL1 were free to be given a cycle start at any time because they caused F_{cleg} to lie within the soft constraint. Sometimes they were even given a cycle start together (see time period 150s, 700s, 900s) because their combined effect resulted in F_{cleg} being within the constraint. The addition of a soft constraint had thus resulted in the desired effect. The optimisation obeyed it at most times and only “violated” it in favour of a better solution (lower objective function).

Figure 7.11 shows an attempt at sequencing coal locks with the soft constraint for F_{cleg} set at 7. All tuning parameters are set at one with the penalty on the soft constraint set at 10 000.

Sequencing was similar to that as in 7.4.2 where the controller was tuned for the task. One major difference though was the greater continuity in F_{cleg} . F_{cleg} as before, never exceeded 7 due the soft constraint. As compared to the tuning case, the controller did not have to wait for F_{cleg} to return to zero before it initiated a cycle start for another coal lock. If F_{cleg} was within bounds in the closed loop horizon i.e. less than 7, then that coal lock was given a cycle start if its temperature was high. With the soft constraint, the optimisation took advantage of the lag, which F_{cleg} experienced after a cycle start. Hence there was continuity in the trajectory of F_{cleg} .

Another feature of the sequencing procedure, which was also present in 7.4.2, was the non-periodic nature of the temperature cycles. This shows that the controller was able to hold back on giving coal locks cycle starts, until such time the CLEG was ready. In the case for the tuned MPC, coal locks were held back based on the penalisation of F_{cleg} from setpoint. However in the present case, coal locks were held back due to the presence of the constraint. As mentioned, the constrained case proved to be superior because it allowed greater continuity in F_{cleg} .

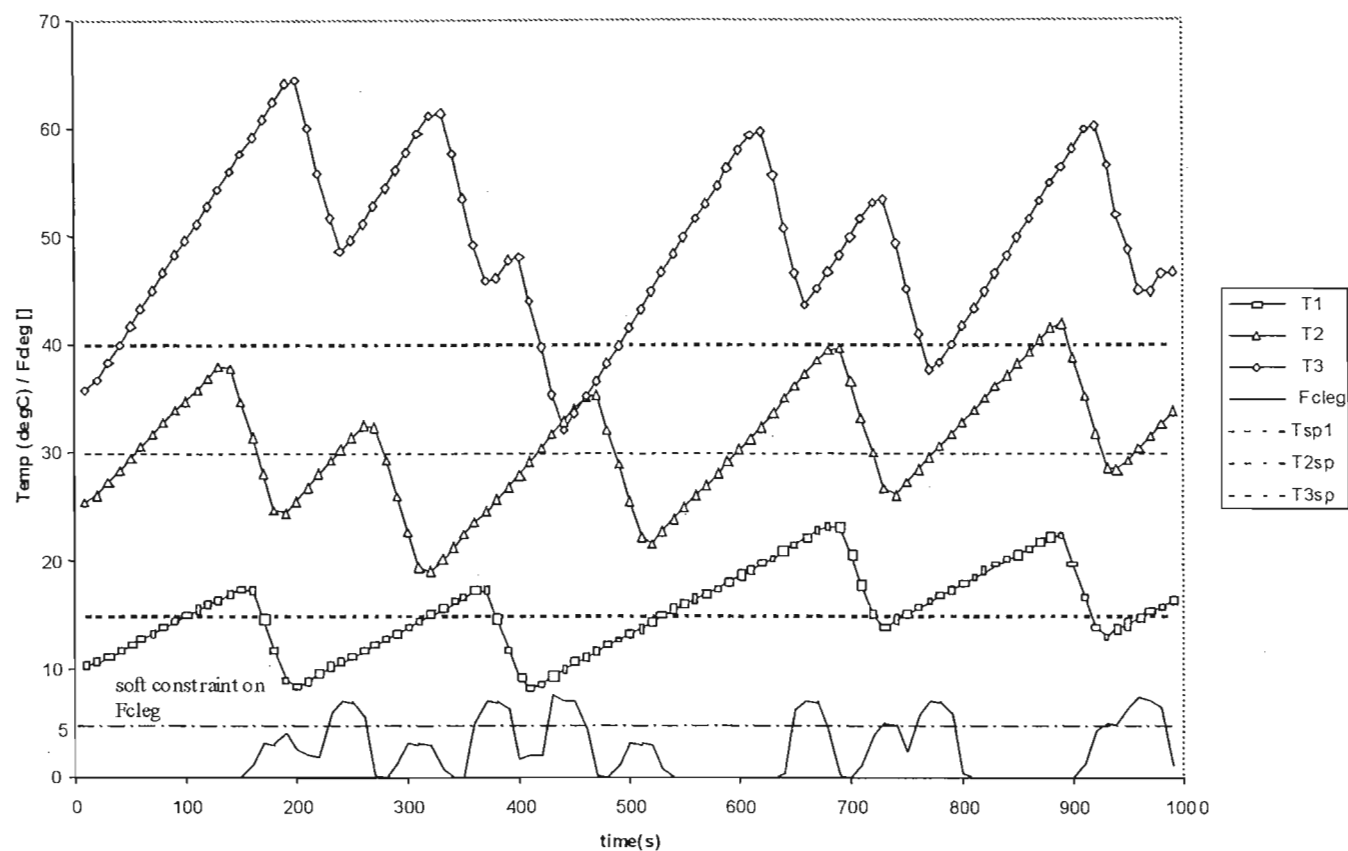


Figure 7.10 Effect of soft constraint on sequencing

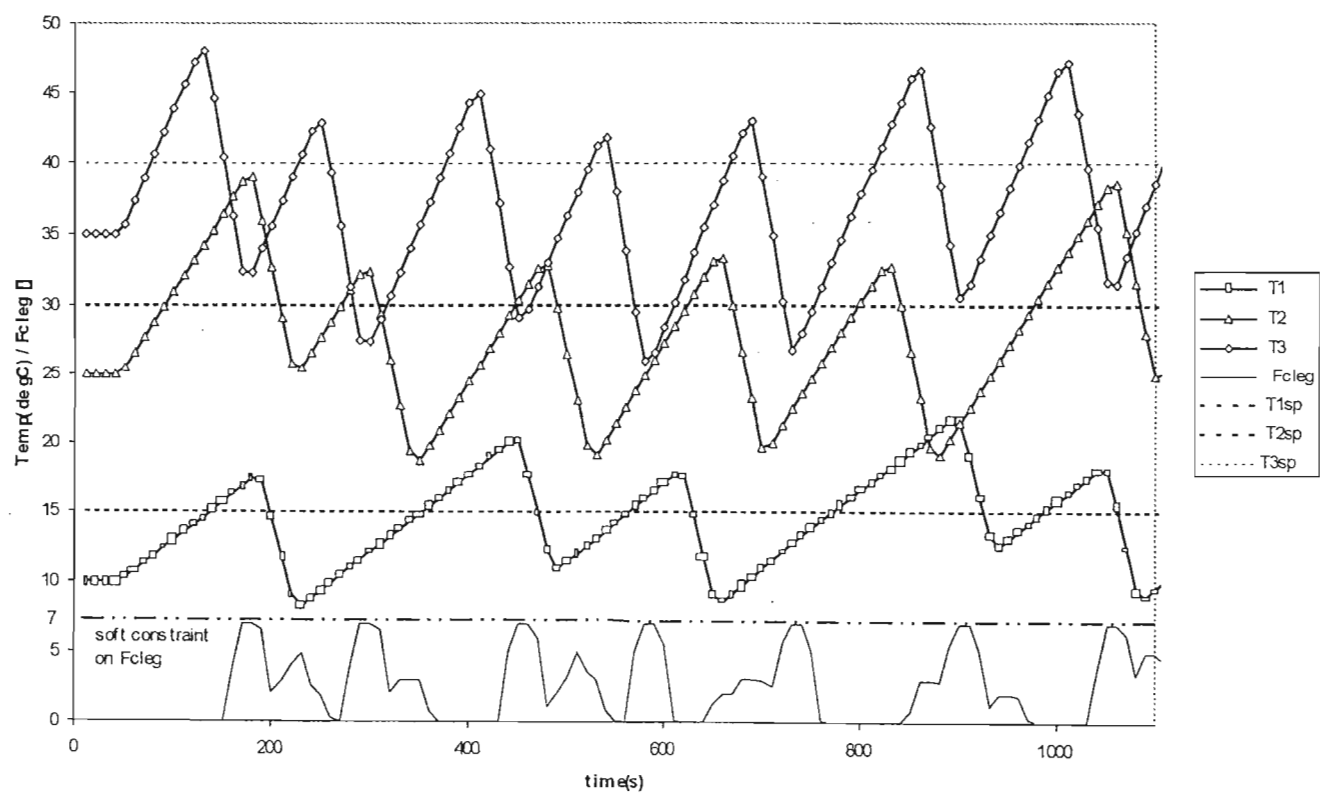


Figure 7.11 Optimal sequencing of coal locks using soft constraint

7.4.4 Sequencing of six gasifiers

It was interesting to see how the controller would perform if it had to deal with a greater number of gasifiers. In figure 7.12, the controller was commissioned to sequence six gasifiers with the soft constraint placed at 10. Out of the six gasifiers, every two had the same response (duplicated as in appendix D1), which were staggered at the start of the simulation.

Sequencing of the six coal lock starts was satisfactory with cycle starts generally being made in close vicinity to setpoints. Of concern though was the time at which the controller allowed F_{cleg} to “violate” the soft constraint. Although this occurred three times and only for short periods, based on the weightings, the controller should have delayed a start in favour of the CLEG. There exists no reasonable explanation for this behaviour with the exception that the increased computational load may have affected the GAMS solution close to the constraint. This phenomenon could perhaps be isolated and investigated by using fewer gasifiers with similar demands on the CLEG. It was however encouraging to see that in spite of this, F_{cleg} remained mostly within the soft constraint and was continuous in nature.

Notice that occasionally the controller gave a lock a cycle start, not long after it had given it its previous start (gasifier2, time 700s; gasifier3, time 400s). The double initiation of a cycle within close proximity of each other is due to the floating reference state property of the convolution model discussed earlier. The decrease in temperature is always fixed and if the state is far above setpoint, then a double initiation might be necessary to bring it to setpoint. Cycle starts, for the same gasifier, close to each is not possible during normal operation, as decompression occurs over a fixed period of time. One way to account for this operational constraint is to introduce a window constraint. Within a window, extending over several time steps, the controller can be forced not to initiate another cycle start after it had already initiated one. This should allow time required for decompression and when that lock emerges from the window, then it should be eligible for a start. The start should as before be based on temperature relative to its setpoint and the status of the CLEG.

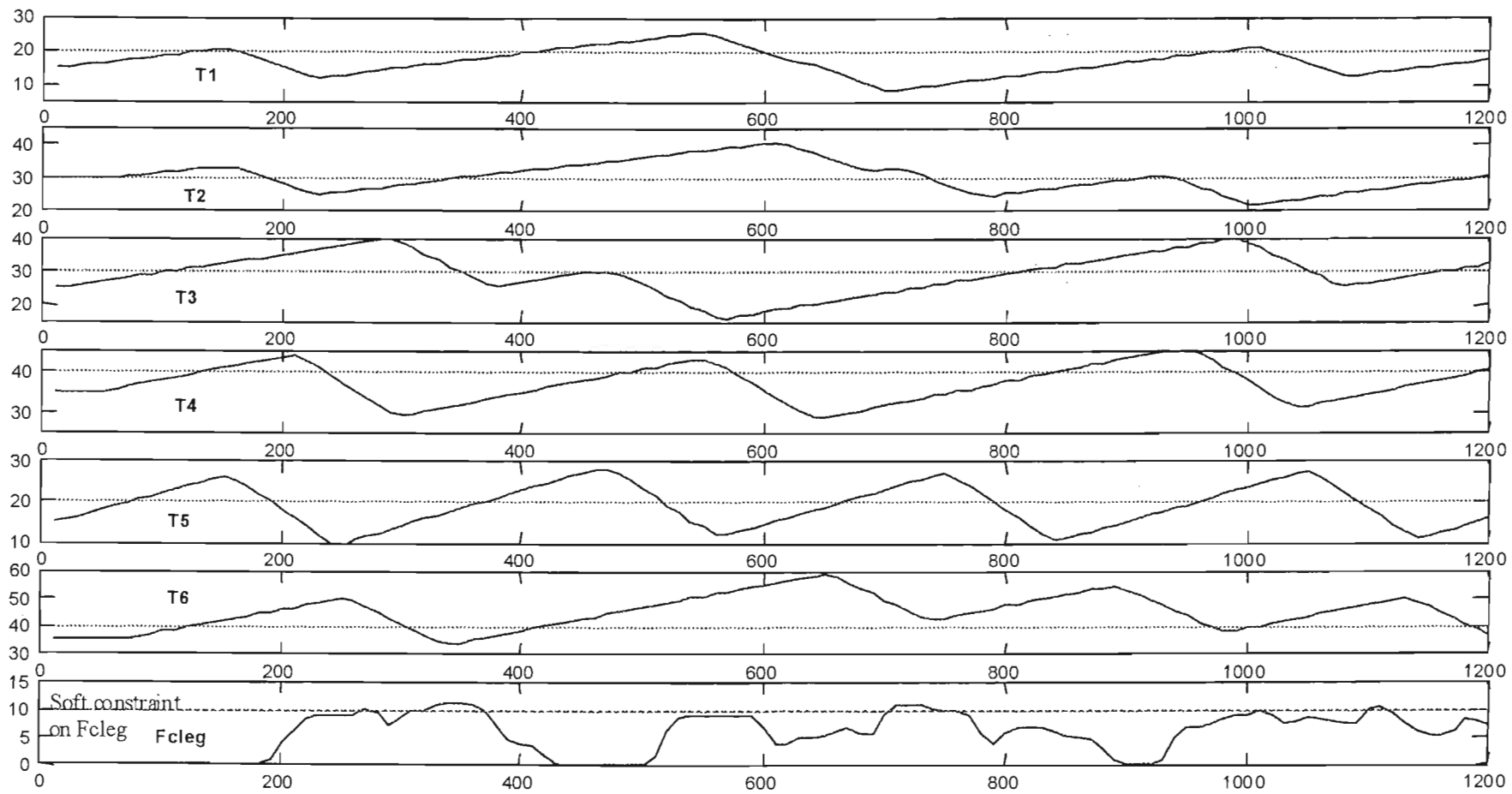


Figure 7.12 Controller sequencing 6 coal locks

7.5 Conclusion

This section shows that MPC and especially DMC can be used to optimally sequence a series of events that define a hybrid system structure. Interest in using DMC to automate a series of gasifiers was initiated with the intention of developing an applicable online controller for the gasification plant in Sasolburg. However, due to the complexity of the actual plant responses, this was not achieved. A true representation of the plant required that the temperature return to a fixed value after a cycle start, regardless of its position prior to initiation. DMC however, being linear, functions on a moving reference point, where changes are always made relative to this point. The actual dynamic response of temperature to an initiation was also complex. As a result, superimposing piecewise linear responses to generate the actual plant response was not possible. However for investigative purposes, where the sequencing capabilities of the MIPC were being tested, a general convolution model proved to be sufficient.

Therefore using simple responses (ramps and steps), a control loop was commissioned with a controller that was able to sequence events based on some strategy. For the current study, the coal locks had to be sequenced so as not to overload a CLEG. A MPC approach was proposed with different strategies. Of those considered, the optimisation based on satisfying a soft constraint proved to be the best. It was discovered that hybrid models based on future predictions had difficulty in fulfilling hard constraints due to plant model mismatch or timing between controller and plant. The problem was however alleviated by using a soft constraint as part of the optimisation.

CHAPTER 8

Physicochemical Discontinuities

Besides characterising systems that have discrete inputs, hybrid systems also encompass another class of systems that, due to their inherent fundamental physical behaviour, have discontinuities. Barton and Pantelides (1994) refer to such discontinuities as physicochemical in nature. Since the system states experience disjunctive behaviour at known threshold values, separate sets of dynamic equations are necessary to describe the complete evolution of the system. As the system's operation progresses, the active state of the model describing the system has to be switched.

Previously, all system outputs were continuous in nature. Controller design was focused on evaluating the optimum decision, at the present moment in time, based on a single fixed model. Physicochemical discontinuities introduce a new challenge regarding the design of model predictive controllers. Generally the internal algorithm has to monitor the system states over the closed loop horizon and then switch state models when a threshold limit is reached.

This chapter presents the design and evaluation of a model predictive controller for a tank system with physicochemical discontinuities. It commences by explaining how such systems can be modeled within the control algorithm for future closed loop projections. Next, a tank system is presented with the dynamic equations and simplifications (discretisation and linearisation) for discrete time modelling and optimisation. Simulations are performed in MATLAB by interfacing it with GAMS, which is used as a sophisticated optimisation toolbox for MATLAB. The chapter concludes by highlighting the findings from these tests.

8.1 Modelling

Barton and Pantelides (1994) report that physicochemical discontinuities can be modeled by dynamic manipulations that have the functional form of equation 4.2. In their explanation, they refer to several instances where systems having physicochemical discontinuities can be modeled using non-linear differential algebraic equations. For example the transition from laminar to turbulent flow in a pipe involves a discrete change in the relationship between friction factor and Reynolds number. Another example is that of fluid in a flash drum. At different times during the separation process, there are three distinct phase arrangements i.e. vapor and liquid, subcooled liquid only and superheated vapor only. In both examples, when the phase arrangement changes, not only does the functional form of the equations change, but also possibly the number of variables. At any given point in time, the current state of the model will thus be characterised by a set of variables and equations that relate these variables.

The model of a system with physicochemical discontinuities can be represented by digraphs with nodes denoting the different model states and arcs signifying instantaneous transitions between these states. For example a system model consisting of three states {A,B,C} can be represented as shown in figure 8.1.

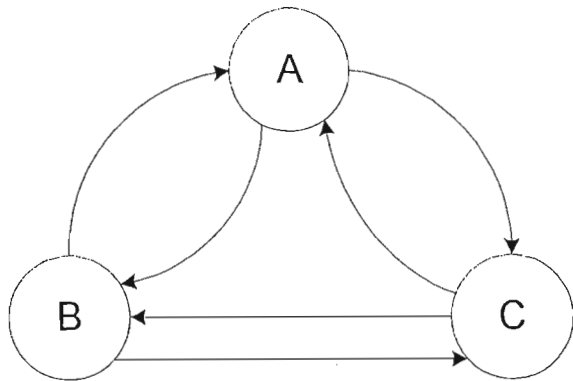


Figure 8.1 Model states for a single system model

During a simulation of the process, the active state of the model will determine the equations that describe the system at that point in time. The active state of the model is changed according to the evolution of the system. In this case, certain events have to be detected in order to trigger the switch. Typically these are state events, where system variables cross threshold values. The time of occurrence of these events is not known *a priori*, so mechanisms have to be devised to detect when they occur.

Figure 8.2 shows graphically the evolution of a system comprising two model states {A,B} and a single system output state (x). Each of the two model states corresponds to a different regime of operation as defined by the threshold values $[x_A^*, x_B^*, x_C^*]$ on the state x .

The differential equations describing the system dynamics in regime A and B respectively are:

$$\frac{dx}{dt} = F_A(x,u) \quad \text{for } x_A^* \leq x \leq x_B^* \tag{8.1}$$

$$\frac{dx}{dt} = F_B(x,u) \quad \text{for } x_B^* \leq x \leq x_C^* \tag{8.2}$$

where u is the input to the system. These equations make up the system model, which when active, individually define the state behaviour of the model.

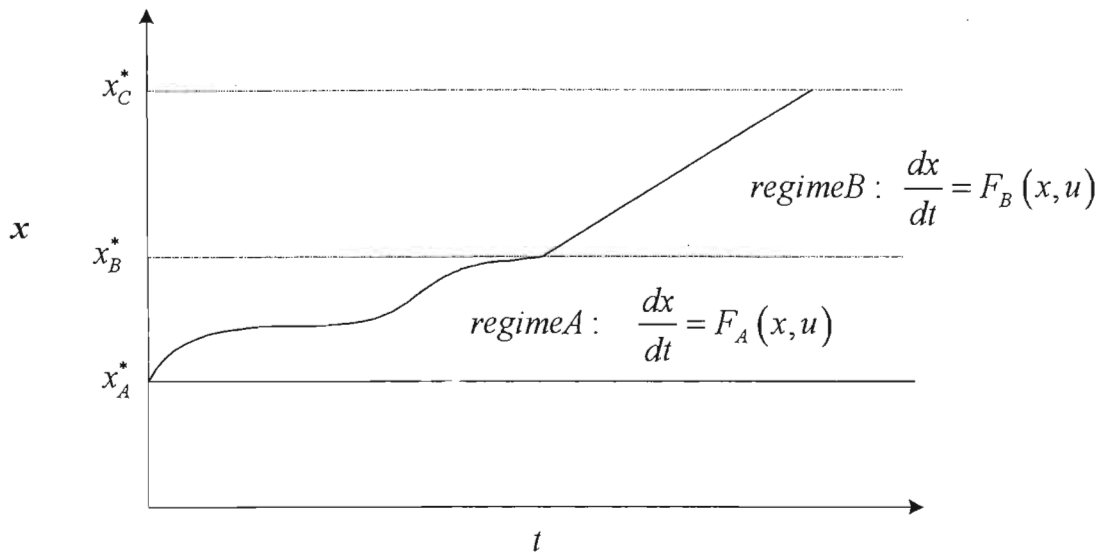


Figure 8.2 Model states for different regimes

When the system state x exceeds the threshold values, the model states are switched. For discrete time computations there however exists a dilemma in simulating the crossing of the boundaries. The selection of the model state depends on the system state. But initially, the system state gets predicted by the state of the model.

Mostermann (1999) briefly presents a general approach to crossing state boundaries. A variation in the integration step is used at the point of discontinuity. A fixed step can be used until within a small tolerance of the discontinuity. When the discontinuity is reached, then the time step is repeatedly reduced to within a set minimal value. This approach provides an accurate approximation to the point of crossing i.e. evaluating at which point in time, with the current model state, the system state crosses the boundary. Mostermann does however report that reducing time step at the point of discontinuity does result in longer simulation times.

The approach used in the present work is to use a small enough integration step, which is kept fixed over the prediction horizon. When the discontinuity is crossed, the state model for the adjacent regime, uses the system state in the present regime as an initial condition to evaluate the new state value that lies in the adjacent regime. In this case, as shown in figure 8.3, the point of switching is not exactly at the point of discontinuity. This is because when the switch into the new regime occurs, the Euler integration uses the state in the present regime as a starting condition for the model in the adjacent regime. GAMS solves equations 8.3, 8.4, 8.5 simultaneously, therefore the point of switching occurs before the actual point of discontinuity due to these logic constraints. This approximated approach relies on a small enough integration step, where the system state gets close enough to the discontinuity using the active model state.

In simulating systems with physicochemical discontinuities, one cannot guarantee that all switchings at the discontinuity surface will be correctly detected, and there may be “chattering” (state switching back and forth) at the surface. The proposed one-step state space predictor, because it has a finite time step, could result in “chattering” around the point where behaviour changes.

Equations 8.3, 8.4 and 8.5 are the conditional expressions devised for implementing this approximated approach. α and β are binary variables that become unity when the state is within a given regime.

$$x - \alpha x_A^* - \beta x_B^* \geq 0 \quad (8.3)$$

$$x - \alpha x_B^* - \beta x_C^* \leq 0 \quad (8.4)$$

$$\alpha + \beta = 1 \quad (8.5)$$

Consequently the complete differential equation governing the system over the entire range of the state is:

$$\frac{dx}{dt} = \alpha F_A(x, u) + \beta F_B(x, u) \quad (8.6)$$

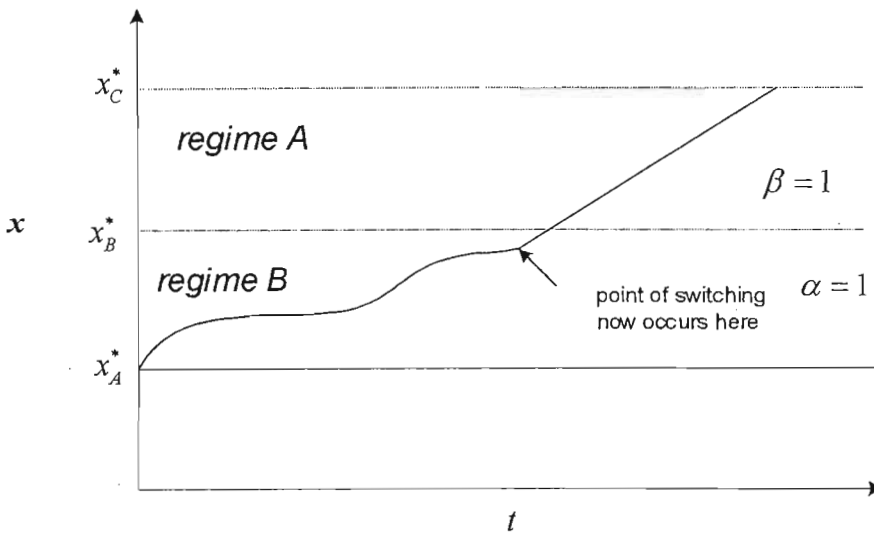


Figure 8.3 Approximated point of switching

In the previous case studies a convolution model was used to predict the future states of the system using past/present moves. Using a convolution model however was found to be not viable for systems with physicochemical discontinuities. The main drawback is the reliance of the model on the past input history. In the event that a new regime of operation is entered into, moves from the past regime cannot be used to predict the present state in the new regime. A convolution model is linear in nature and is generated from continuous step responses. When a new regime is entered into, their impact in the new regime will change. Even if a different convolution model is used in each regime, the past history of the system will still have to be known in order to determine the rate of change in the new regime. This effectively rules out using DMC for the control of systems with physicochemical discontinuities.

State space equations, the discretised version of the differential equations 8.1 and 8.2, are therefore suggested and used in the present work for the control of systems with state discontinuities. The numerical integration only relies on system variables one step back in time. This proves to be ideal, especially when a discontinuity is crossed, as minimal information from the previous regime is required.

8.2 Case Study – Non-uniform tank with overflow

The theory in section 8.1 was developed based on the simulation of a hypothetical tank system as shown in figure 8.4. The system, effectively a single tank, consisted of two tanks, one on top of another. The diameters of the two tanks were different, thereby giving the entire system a discontinuous cross sectional area. At the point where the two tanks met, was an overflow (F_2). Liquid continuously drained through restriction k_1 from the bottom tank according to the head of liquid in the system. Flow (F_o) was the continuous inlet to the system. The overflow height was set at height h_v , which was the height of the bottom tank. The aim of this case study was to simulate the closed loop control of level (h_l) by manipulating the inflow (F_o).

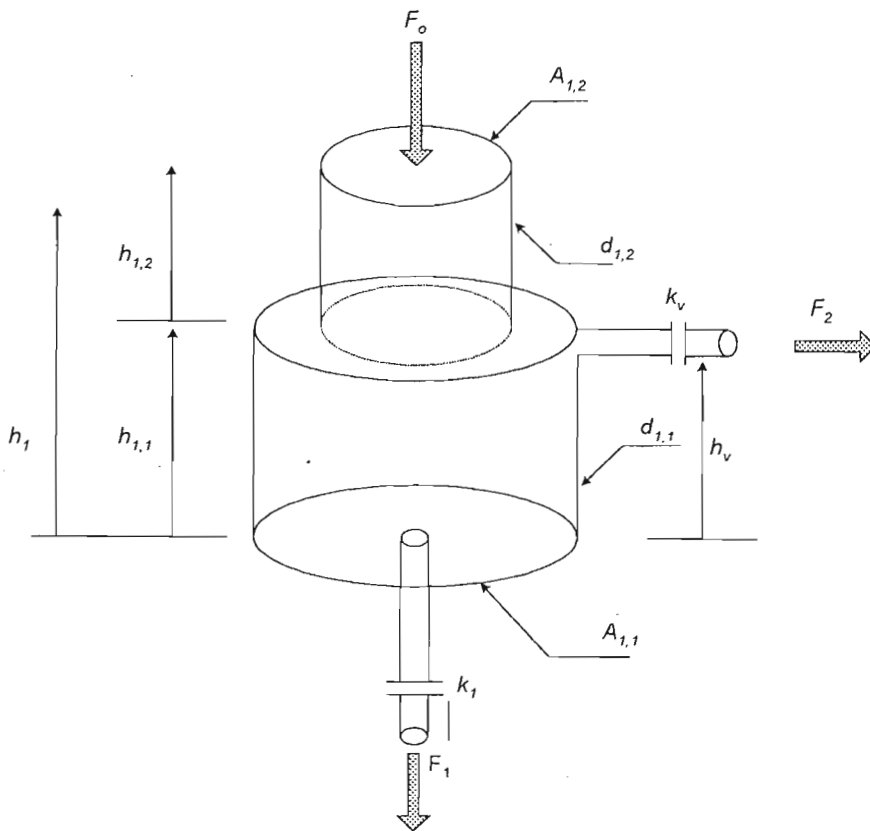


Figure 8.4 Hypothetical tank with discontinuous cross sectional area and overflow

Systems such as the above are commonly used for illustrating the control of hybrid systems that have discontinuities based on the state. In the literature, Torrisi *et al.* (2001) and Kowalewski *et al.* (1997) have followed similar approaches. The former citation was a time-optimal control problem that

compared a heuristic approach for controlling the levels in three tanks, to a systematic approach using model predictive control. The latter used it as a benchmark example for comparing different computer tools with respect to the validation of logic control programs.

The geometry of this system divided it into two regimes of operation i.e. when the level was below the overflow (in tank of area $A_{l,1}$) and when it was above the overflow (in tank area $A_{l,2}$).

Two separate differential equations were thus required to describe the level in the tank (equation 8.7). Equations 8.8 and 8.9 were derived from the mass balance of liquid in the tank. It was these equations, which resemble 8.1 and 8.2, that collectively made up the system model and were individually activated based on system state variable h_l .

$$\frac{dh_l}{dt} = \frac{dh_{l,1}}{dt} + \frac{dh_{l,2}}{dt} \quad (8.7)$$

$$\left. \begin{aligned} \frac{dh_{l,1}}{dt} &= \frac{1}{A_{l,1}} [F_o - F_1] \\ &= \frac{1}{A_{l,1}} [F_o - k_1 \sqrt{h_{l,1}}] \\ \frac{dh_{l,2}}{dt} &= 0 \end{aligned} \right\} \quad (8.8) \quad \text{for } h_l \leq h_v$$

$$\left. \begin{aligned} \frac{dh_{l,1}}{dt} &= 0 \\ \frac{dh_{l,2}}{dt} &= \frac{1}{A_{l,2}} [F_o - F_1 - F_2] \\ &= \frac{1}{A_{l,2}} [F_o - k_1 \sqrt{h_{l,2} + h_v} - k_v \sqrt{h_{l,2}}] \end{aligned} \right\} \quad (8.9) \quad \text{for } h_l \geq h_v$$

For discrete time modelling and simulation, equations 8.8 and 8.9 have to be discretised.

If $\frac{dx}{dt} = f(x, t)$ is an ordinary differential equation, then the state variable (x) at every time step Δt can be calculated by the Euler method as:

$$x_{n+1} \approx x_n + f(x_n, t_n) \Delta t$$

$$\text{where } t_{n+1} = t_n + \Delta t$$

The differentials in equations 8.8 and 8.9 can be discretised, using fixed integration step Δt , to yield

$$h_{1,1}(t) = h_{1,1}(t-1) + \frac{\Delta t}{A_{1,1}} \left\{ F_o - k_1 \sqrt{h_{1,1}(t-1)} \right\} \quad (8.10)$$

$$h_{1,2}(t) = h_{1,2}(t-1) + \frac{\Delta t}{A_{1,2}} \left\{ F_o - k_1 \sqrt{h_{1,2}(t-1) + h_v} - k_v \sqrt{h_{1,2}(t-1)} \right\} \quad (8.11)$$

Due to the presence of the square root function, the model had to be linearised before programming into GAMS. Nonlinear functions can be linearised by expanding them in a Taylor series, to one term, around a nominal operating point (\bar{x}). If $f(x)$ is a nonlinear function in x , then expansion with truncation up to the first term, about its steady state operating point yields the following approximation.

$$f(x) \approx f(\bar{x}) + \left. \frac{df}{dx} \right|_{\bar{x}} (x - \bar{x})$$

Consequently

$$k_1 \sqrt{h_{1,1}(t)} \approx k_1 \sqrt{h^o} + \frac{k_1}{2\sqrt{h^o}} (h_1(t-1) - h^o) \quad \text{for } h_1 \leq h_v$$

$$k_v \sqrt{h_{1,2}(t)} = k_v \sqrt{h_1 - h_v} \approx k_v \sqrt{h^o - h_v} + \frac{k_v}{2\sqrt{h^o - h_v}} [h_1(t-1) - h^o]$$

$$k_1 \sqrt{h_{1,2}(t) + h_v} = k_1 \sqrt{h_1(t)} \approx k_1 \sqrt{h^o} + \frac{k_1}{2\sqrt{h^o}} (h_1(t-1) - h^o) \quad \text{for } h_1 \geq h_v$$

where $h_1(t) = h_v(t) + h_{1,2}(t)$; if $h_1(t) \geq h_v$

$h_1(t) = h_{1,1}(t)$; if $h_1(t) \leq h_v$

h^o is the steady state operating point around which the Taylor expansion is done. For the current simulations, h^o for the present control cycle, was assigned the height (h_l) from the past cycle. All predictions for the future state of the system were thus evaluated relative to this assignment of h^o .

With α and β being binary variables for regime1 and regime2 respectively, the equation block for the simultaneous solution for process variables was therefore:

$$h_1(t) = h_1(t-1) + \alpha(t) \Delta h_{1,1}(t) + \beta(t) \Delta h_{1,2}(t) \quad (8.12)$$

$$\Delta h_{1,1}(t) = \frac{\Delta t}{A_{1,1}} \left\{ F_o(t) - \left[k_1 \sqrt{h^o} + \frac{k_1}{2\sqrt{h^o}} (h_1(t-1) - h^o) \right] \right\} \quad (8.13)$$

$$\Delta h_{1,2}(t) = \frac{\Delta t}{A_{1,2}} \left\{ F_o(t) - \left[k_1 \sqrt{h^o} + \frac{k_1}{2\sqrt{h^o}} (h_1(t-1) - h^o) \right] - \left[k_v \sqrt{h^o - h_v} + \frac{k_v}{2\sqrt{h^o - h_v}} (h_1(t-1) - h^o) \right] \right\} \quad (8.14)$$

$$\alpha(t) + \beta(t) = 1 \quad (8.15)$$

$$h_1(t) - \alpha(t)h_v - \beta(t)h_{\max} \leq 0 \quad (8.16)$$

$$h_1(t) - \alpha(t)h_{\min} - \beta(t)h_v \geq 0 \quad (8.17)$$

$$F_o(t) \leq F_{\max} \quad (8.18)$$

$$F_o(t) \geq 0 \quad (8.19)$$

Equations 8.13 and 8.14 reveal that, h^o cannot be allowed to be equal to zero, and in regime2, $h^o - h_v$ cannot be less than or equal to zero. In feedback h^o was prevented from being equal to zero. A nominal value for $h^o - h_v$ was predefined (e.g. equal to 1) while in regime1 and when the operation switched to regime2, this nominal value was used in equation 8.14, until h^o was updated at the next time step.

Figure 8.5 shows the control loop and sequence of events as executed in MATLAB for the simulation and control of this system. MATLAB was used for the plant block, where the nonlinearised version of the discretised differential equations (8.10,8.11) were used, while GAMS, using equations 8.12-8.19, was executed at every time step to evaluate the optimum plant input F_o . In this procedure it was the controller that switched the regime of operation of the system i.e. it evaluates α and β and then passes it to the plant model (equation 8.20). Using equation 8.20 as the plant model implies that there is a fair match (depending on the accuracy of the linearisation) between the plant and the controller. See Appendix E.1 for the executable MATLAB code.

$$h_1(t) = h_1(t-1) + \alpha \frac{\Delta t}{A_1} \left[F_o - k_1 \sqrt{h_1(t-1)} \right] + \beta \frac{\Delta t}{A_2} \left[F_o - k_1 \sqrt{h_1(t-1)} - k_v \sqrt{h_1(t-1) - h_v} \right] \quad (8.20)$$

The files for interfacing MATLAB and GAMS were freely downloaded from the internet.¹ With these come a document that explains the details for linking these two packages. By linking these two modelling software packages, a single environment was created that combined the best of both languages. MATLAB is superior in data manipulation and visualisation, while GAMS has the ability to perform large-scale non-linear optimisations, which is generally beyond the capacity of the optimisation tools in MATLAB. MATLAB optimisation tools are useful for small scale nonlinear models and to some extent for large linear models. It however lacks the ability to perform automatic derivatives, which makes it impractical for large scale nonlinear optimisation. GAMS with its wide variety of solvers (MINOS, DICOPT, CONOPT, OSL, CPLEX) has got this capability and has been used in many practical large scale nonlinear applications. Refer to the website² for the online GAMS model library that presents examples of models.

¹ <http://www.cs.wisc.edu/math-prog/matlab.html>

² www.gams.com/modlib/modlib.htm.

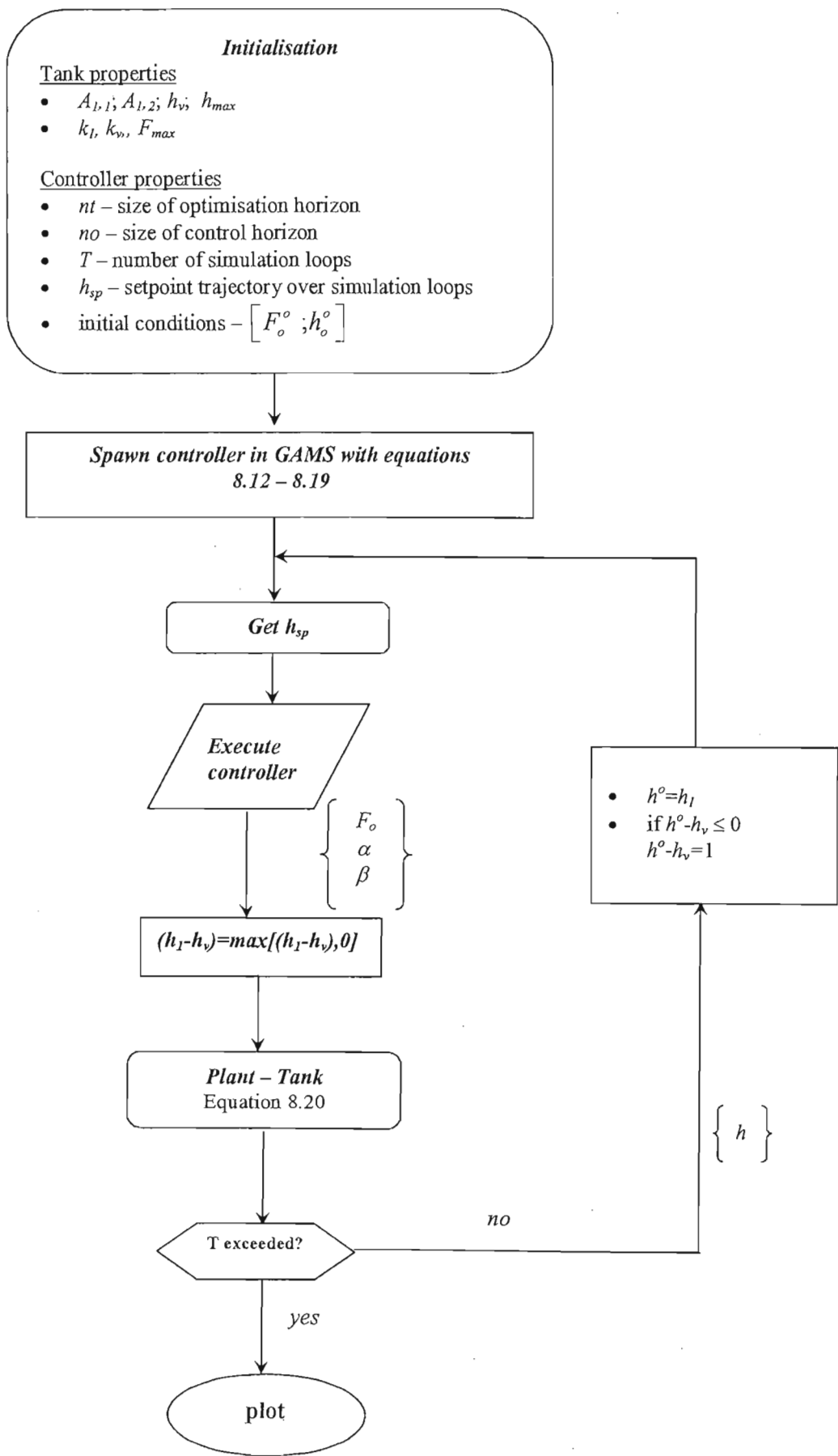


Figure 8.5 Flowchart for closed loop simulation in MATLAB for tank with overflow and discontinuous cross sectional area

8.3 Simulations

The procedure in figure 8.5 was utilised to simulate open/closed loop tests for the tank system. The aim of these tests was two-fold. Firstly it was to show the effect of the discontinuities (cross sectional area and overflow) on the operation of the system. Secondly it was also to illustrate the predictive capabilities of the control algorithm when the regime of operation was switched.

The following tank and controller properties were used in the simulations:

$d_{l,1}(m)$	2.5
$d_{l,2}(m)$	1.5
$A_{l,1}(m^2)$	4.9
$A_{l,2}(m^2)$	1.8
$k_l(m^3 \cdot h^{-1}/m^{0.5})$	0.009
$k_v(m^3 \cdot hr^{-1}/m^{0.5})$	0.004
$h_v(m)$	5
$h_{max}(m)$	15
$nt(-)$	variable
$no(-)$	variable
$T(s)$	variable
$F_{max}(m^3/hr)$	150
$\Delta t (s)$	60

Table 8.1 Controller and system parameters used in simulations

8.3.1 Openloop tests

Open loop tests were done in order to observe the effect of the discontinuities on the tank dynamics. The controller was used in open loop to fill or empty the tank. It initiated just one control action i.e. either setting F_o to zero or its maximum flow, depending on whether the tank was being filled or emptied. No further regulatory action was necessary thereafter.

Figure 8.6 shows the discontinuous dynamics caused by a change in diameter when the tank was being filled. No overflow was allowed, but the tank diameter above 5m was changed from 2.5m (cross sectional area 4.9 m²) to 1.5m (cross sectional area 1.8 m²). Tank level will obviously increase at a greater rate for a smaller cross sectional area. This was reflected in the simulation, where the dotted line was what the level would have been if no change in diameter had occurred. Notice that the switching, as depicted generally in figure 8.3 and explained on page 8-3, occurs before the point of discontinuity (time step 13).

Figure 8.7 shows the effect of the overflow when the tank is being filled. The tank diameter was kept constant (2.5m), but overflow was now allowed at the height of 5m. The simulation shows that after the point of discontinuity was initially crossed, there was little difference caused by the presence of the overflow. However in time, when the level in regime2 started to increase and the flow from the overflow increased as well (due to the greater head), the effect of the overflow became apparent.

Figure 8.8 illustrates the combined effect of both discontinuities. Both effects on the state essentially become superimposed. The effect of diameter change was prominent early in the second regime. However in time when the level started to increase, the effect of the overflow could be seen. Unlike figure 8.6 the level in the tank started tending towards a steady state. This was due to the overflow. The presence of the discontinuities still caused a major change in the filling dynamics, as shown by the offset between the curves in the second regime. As before, as the tank fills and the point of discontinuity is crossed, the model is switched prior to the actual point.

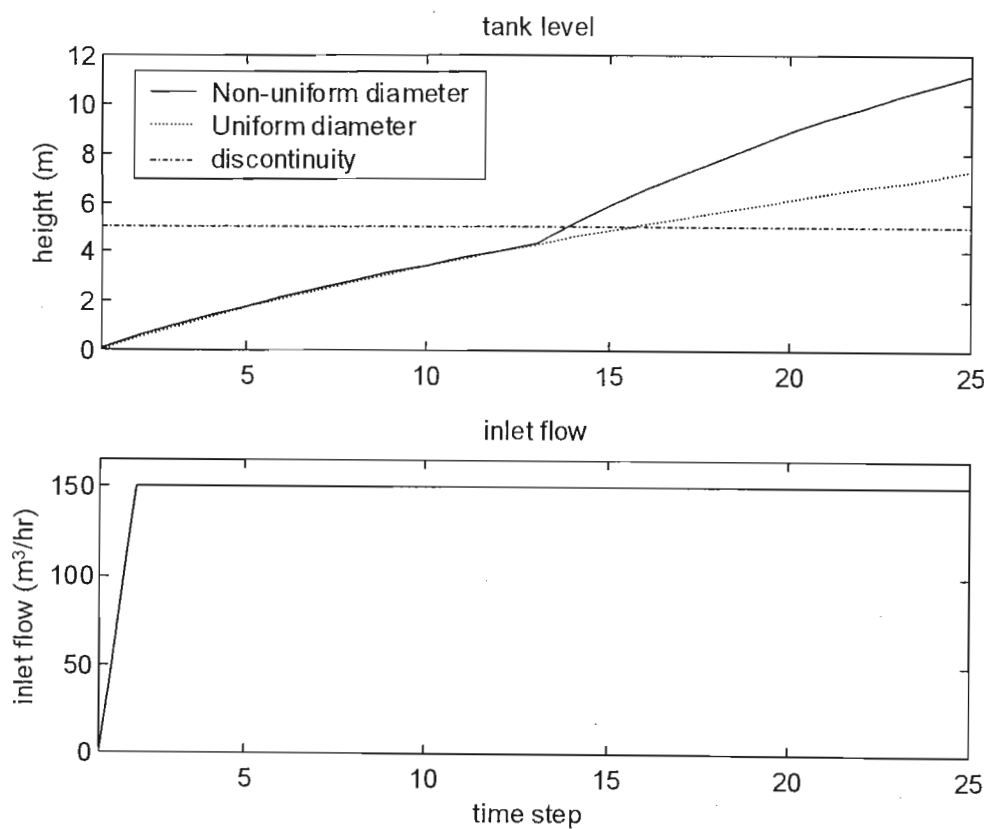


Figure 8.6 Effect of non-uniform diameter when filling tank

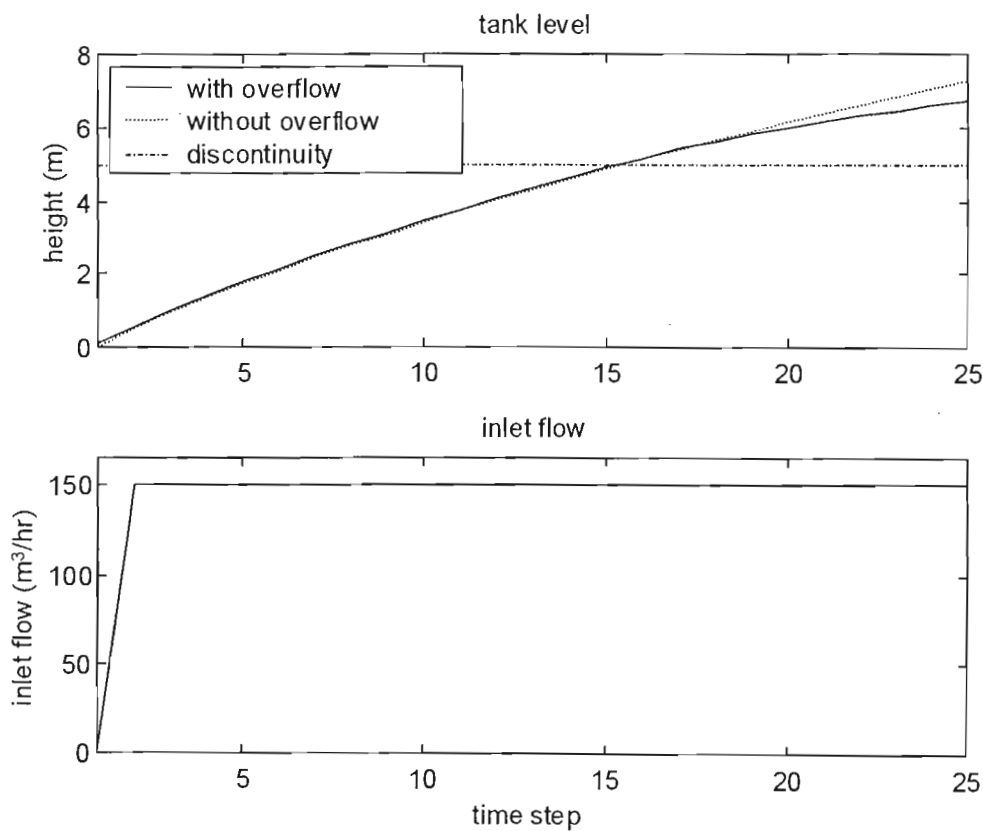


Figure 8.7 Effect of overflow when filling tank

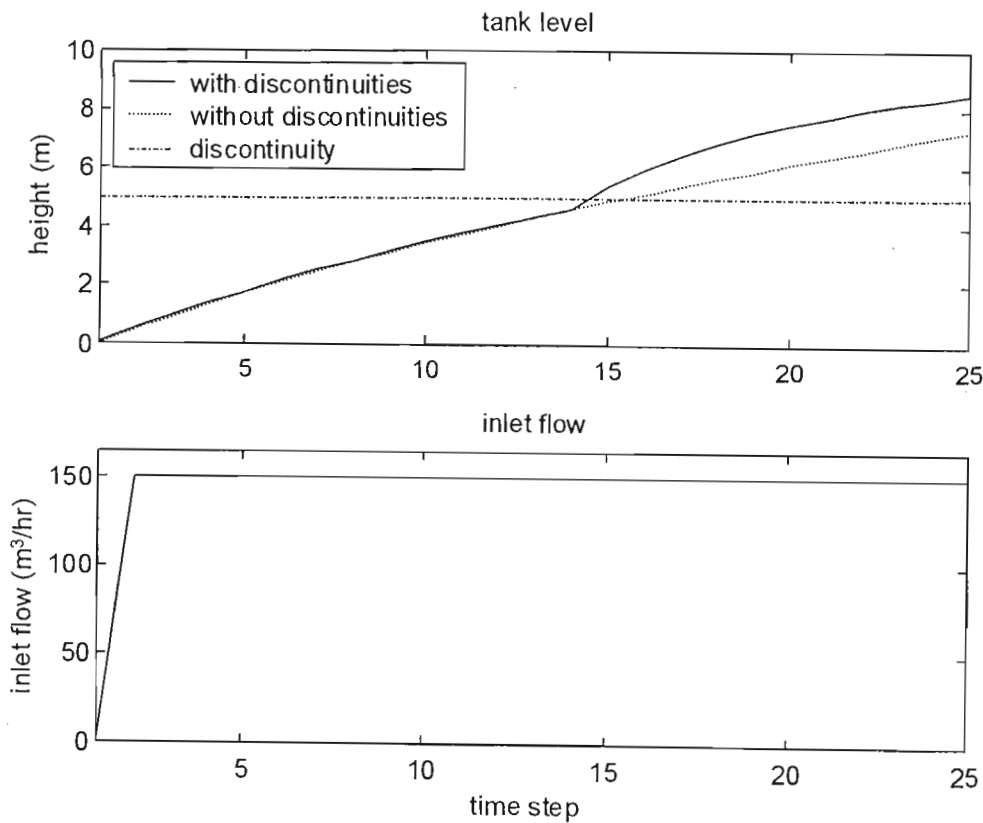


Figure 8.8 Combined effect of non-uniform diameter and overflow when filling

Figure 8.9 shows the combined effect of the discontinuities when emptying the tank from an initial level that was above the overflow. F_o was set to 0, and the tank drained through the bottom line (F_I) and overflow (F_2). For the comparison (dotted line), tank diameter was kept uniform at 1.5m and there was no overflow. The effect of the overflow was clearly apparent in regime2 ($h_I > 5\text{m}$). Although the effect was not significant when filling the tank (figure 8.7), it had a much greater influence on the rate of level change when the tank was emptied. Below the point of discontinuity, due to the increased diameter, the rate of decrease was more gradual.

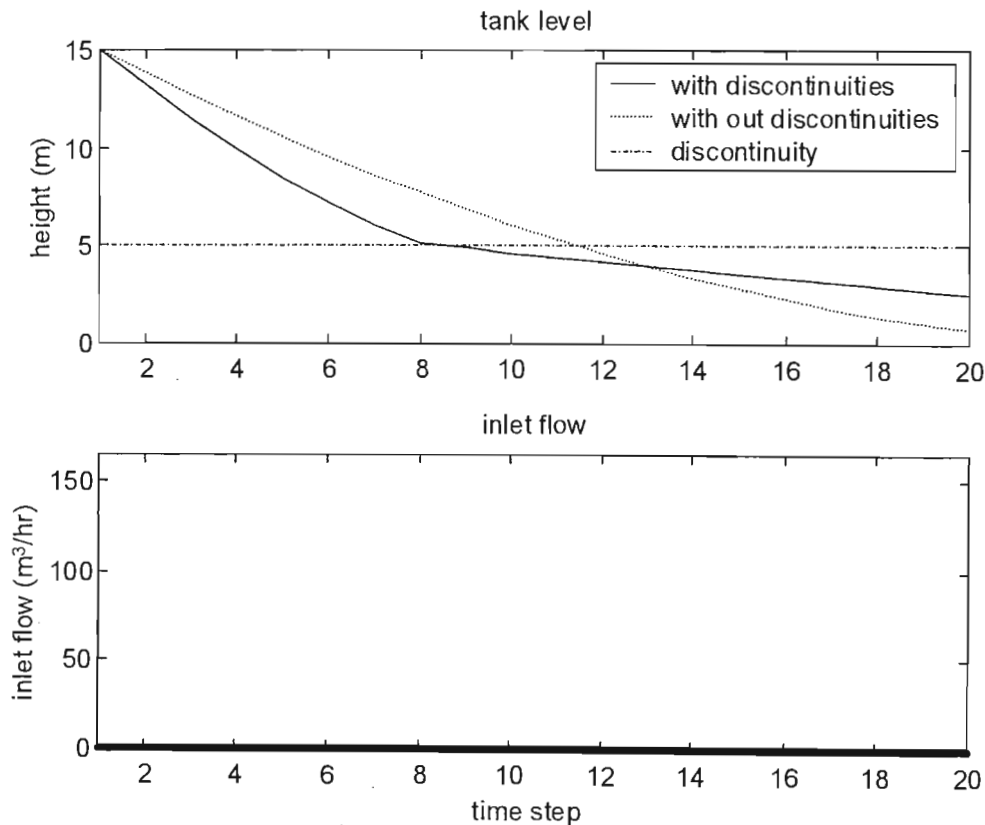


Figure 8.9 combined effect of non-uniform diameter and overflow when emptying

8.3.2 Closedloop Tests

Closed loop tests were undertaken to show that, by switching model states at the points of discontinuity, the internal controller model was able to evaluate closed loop trajectories that extended across regimes. Afterwards, the controller must use the appropriate model for the regulation of the system state. These tests involved variations in the size of the optimisation horizon and the number of optimised moves.

8.3.2.1 Forward Control

MPC is an attractive control strategy because it is able to anticipate the future demands on a system and hence plan accordingly. In the following tests the size of the optimisation horizon together with the control horizon was varied, to show the anticipation capabilities of the controller. This proved to be challenging as the predications had to be done across a system state discontinuity, which required switching the model states. At every control instant, the algorithm had a forecast view, extending over the optimisation horizon, regarding the system demands i.e. level setpoint tracking. With this forward control approach, it was expected that the algorithm would plan ahead and thus base changes made at the present moment in time, on the future demands of the system. Most notably it had to do this planning across regimes, each of which was defined by its own set of equations.

The following simulations show step changes in the tank level about the point of discontinuity. Figure 8.10 shows the sequence with the control algorithm allowed a single step optimisation and control horizon. (Recall that the optimisation horizon is the number of steps extending into the future where the model is used to project the evolution of the state. On the other hand, the control horizon is the number of initial steps of the optimisation horizon on which control moves are allowed). Figures 8.11 and 8.12 had the same setpoint step sequence, but the controller in this case had an optimisation horizon of 5 steps. The control horizon for the controller in figure 8.11 was 1 step, while that for figure 8.12 was 3 steps.

The controller in figure 8.10, due to its limited horizon, did not show any anticipation regarding the future. Despite this, it was still able to control the inlet flow to the tank (F_o), so that the level could initially approach setpoint and track it thereafter. In doing so, it showed the ability to switch models at the point of discontinuity.

Increasing the optimisation horizon immediately introduced the anticipation effect. In figure 8.11, the state never got to the setpoint when filling because the decrease in setpoint was anticipated over the subsequent time steps. The algorithm planned ahead for the decrease in setpoint, by reducing F_o before the change in setpoint was made. A comparison between the controller action on F_o , in figures 8.10 and 8.11 shows that the extended horizon resulted in far more conservative and smoother changes on system input. Consequently the state's approach to setpoint was extended. For the controller with a single optimisation horizon, changes in the plant input were more precise. This is because the single control step horizon did not involve any future predictions, so the single move was based entirely on the next step.

In figure 8.12, the controller was also able to anticipate the change in operating point (see time step 10). A larger control horizon however meant that system objectives were fulfilled on the first move, while the subsequent moves allowed for fulfilling demands further on in the optimisation horizon. This was not the case in figure 8.11, where the controller had a single control step. In that case a

single control action had to accommodate for the demands on the next time step and those extending into the future.

Together with validating the switching capacity of the internal model regarding the model state, these closed loop tests also allowed two further subtle observations. Firstly, a single step horizon was shown to be sufficient for control purposes, especially when the system is single-input-single-output like this one. In addition, a smaller optimisation horizon resulted in shorter solution times every time the control algorithm was executed. Secondly, due to the success the controller had in getting the level to track setpoint, there obviously was a good match between the internal model of the controller and the plant model. Recall that the internal model was a linearised version of the actual discretised differential equations 8.10 and 8.11. The linearisation, based on expansion about the state from the last time step, was thus a good approximation of the actual non-linear model.

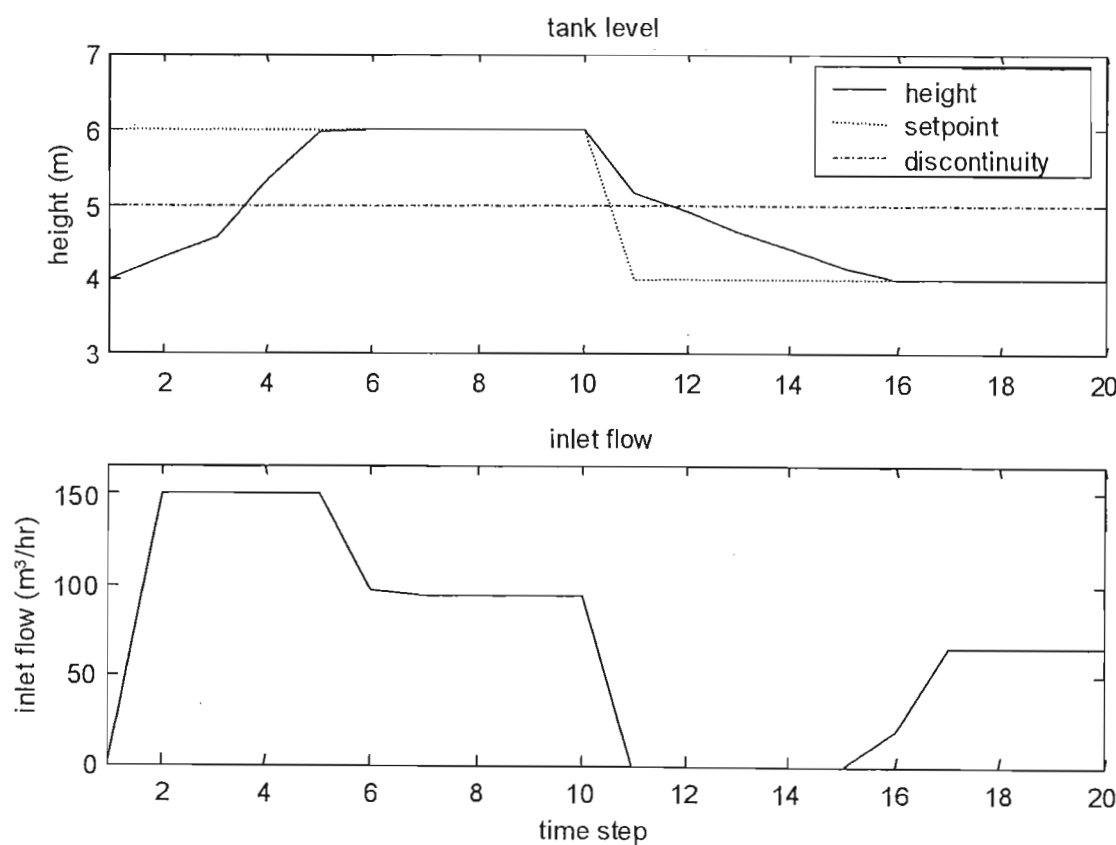


Figure 8.10 Control sequence with 1 step optimisation horizon and single control move

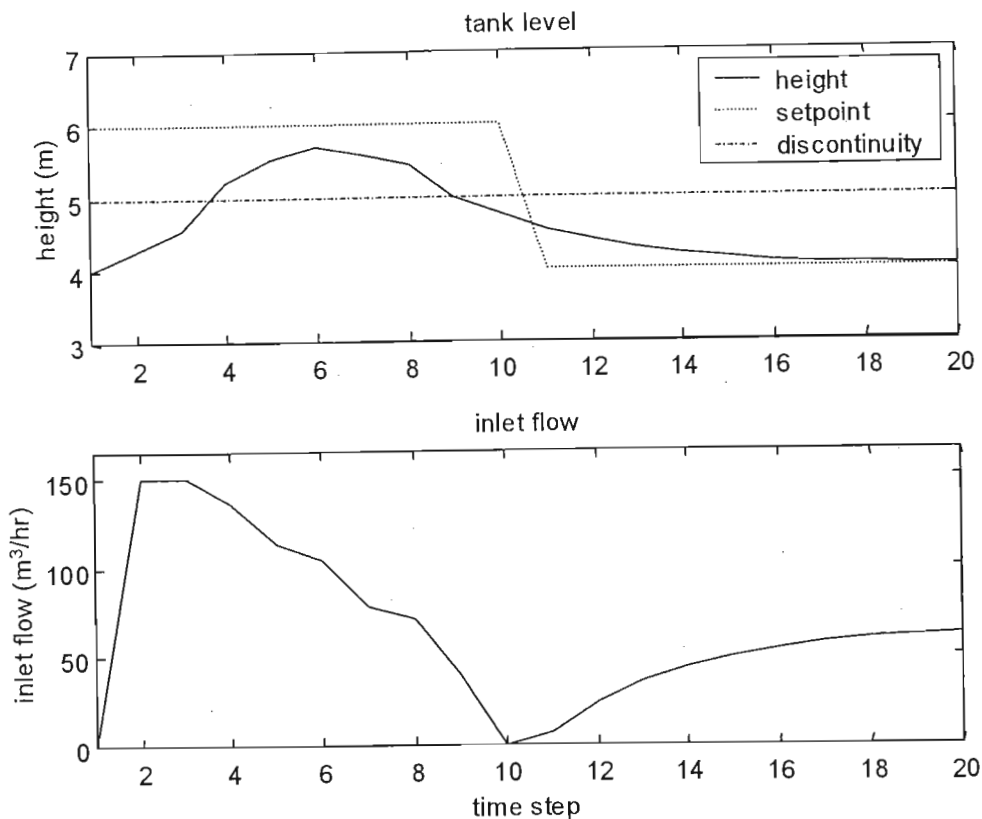


Figure 8.11 Control sequence with 5 step optimisation horizon and single control move

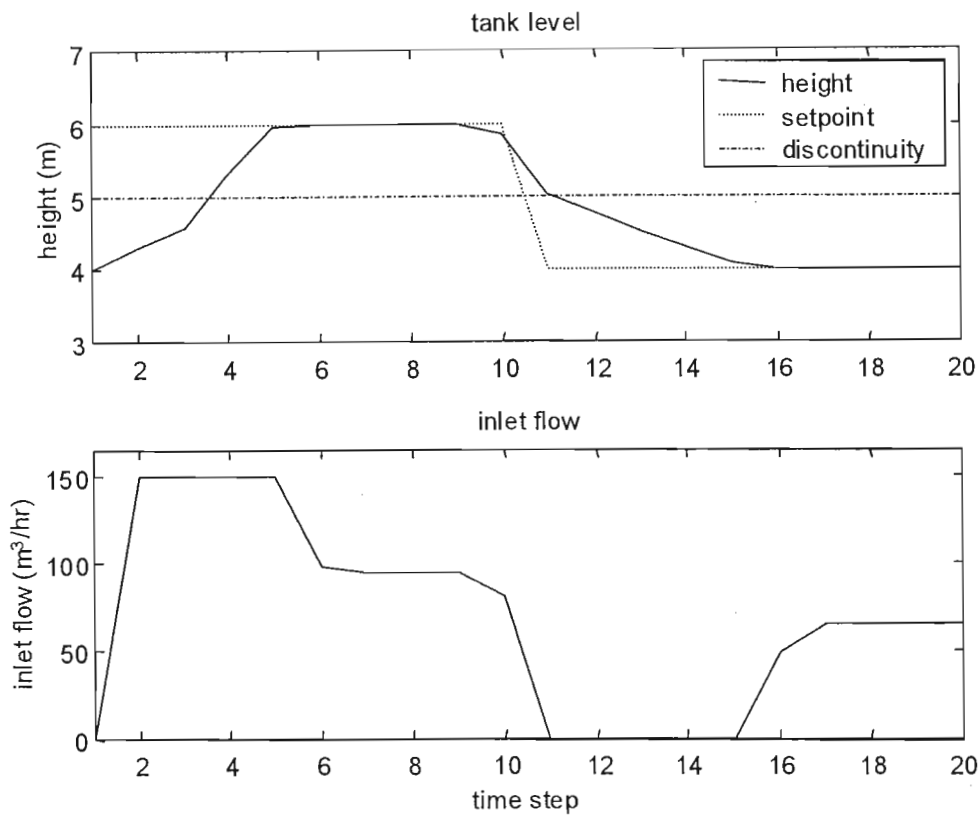


Figure 8.12 Control sequence with 5 step optimisation horizon and 3 control moves

8.3.2.2 Multiple-input-multiple-output

In order to test the predictive nature of the algorithm, another hypothetical tank system as shown in figure 8.13 was devised. The outlets (F_1 and F_2) from tank1 drain into tank2. Tank2 was similar to the lower half of tank1 i.e. in diameter and drained in the same way, but it did not have an overflow. The line coefficient (k_2) was set at $0.008m^3.hr^{-1}/m^{0.5}$. To give the system inputs a hybrid nature, the bottom line from tank1 had a binary valve that was either open or closed. Flow F_o into the first tank was still continuous and the controlled variables were now the two tank levels h_1 and h_2 .

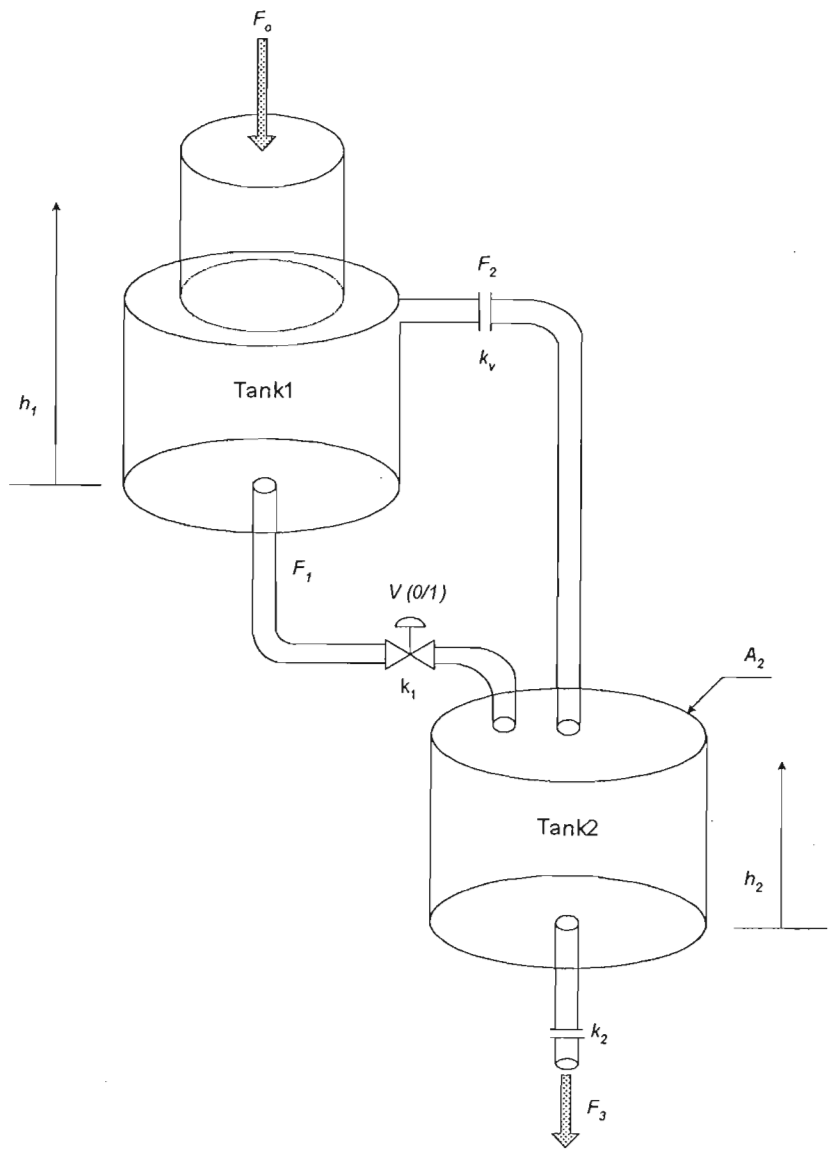


Figure 8.13 Two tank system

The dynamics of tank2 were simulated in the same way as that for tank1. Tank2 also had a physicochemical discontinuity depending on whether the liquid level in tank1 was above or below the overflow. The differential equations according to the annotations in figure 8.13, for tank2 were thus:

$$\left. \begin{aligned} \frac{dh_2}{dt} &= \frac{1}{A_2}(F_1 - F_3); \\ &= \frac{1}{A_2}(V(t).k_1\sqrt{h_1} - k_2\sqrt{h_2}) \end{aligned} \right\} \text{ if } h_1 \leq h_v \quad (8.21)$$

or

$$\left. \begin{aligned} \frac{dh_2}{dt} &= \frac{1}{A_2}(F_1 + F_2 - F_3); \\ &= \frac{1}{A_2}(V(t).k_1\sqrt{h_1} + k_v\sqrt{h_1 - h_v} - k_2\sqrt{h_2}) \end{aligned} \right\} \text{ if } h_1 \geq h_v \quad (8.22)$$

where $V(t)$ is a binary variable, whose state at every time step determined whether there was flow from the bottom of tank1 or not. A_2 is the cross sectional area of tank2.

Equations 8.21 and 8.22 were discretised and linearised as before. The same equations were used for tank1 as that for feed forward control in the previous section. See appendix E.2 for the equation block describing the controller in GAMS. In order to demonstrate a model predictive control strategy, the controller was used to direct the filling of both tanks, from an initially empty state, up to a desired level. However, a greater penalty was placed on the offset of h_2 from its setpoint. In this instance the control algorithm would therefore favour the level in tank2 ahead of that in tank1. The desired setpoint for tank1 was 4m i.e. below the point of discontinuity, while the setpoint for tank2 was 6m.

Figure 8.14 shows the control action and resulting tank level responses when the algorithm was allowed a 2-step optimisation horizon with 1 controlled move. Due to the greater penalty on the level in tank2 ($W_1=3; W_2=20$), the level in tank1 remained slightly above its setpoint ensuring that, without deviating too much from setpoint, the head for flow into tank2 (F_1) was as high as possible. When h_2 reached setpoint, h_1 was returned to setpoint.

On the other hand if the optimisation algorithm was allowed a future optimisation horizon of 5 steps and a 2-move control horizon, then a different control action was observed. The same tuning parameters as those used previously were used. Figure 8.15 shows that, by keeping F_o at its maximum flowrate for a longer period, tank1 was filled above the point of discontinuity, so that tank2 could get filled through the overflow as well. Simultaneously, a greater level in tank1 meant that there was greater head for the flow from the bottom of tank1.

Notice the slight inflection in the level response of tank2 at time step 14, indicating that tank2 was also being filled via the overflow (F_2) as well. With the level in tank2 approaching setpoint, the inflow to tank1, F_o , was stopped. As a result of this action, both levels got to setpoint thereafter. Tank1 drained and the excess liquid now passed to tank2, which in turn got the level in tank2 to setpoint. In the mean time, the position of the discrete valve was set accordingly. When both states reached

setpoint, F_o was activated periodically together with the discrete valve, to account for the loss from tank2 via F_3 .

Also notice that it took a smaller number of control cycles for both levels to reach their respective setpoints in figure 8.15 as compared to figure 8.14. If this were a minimal time problem, then this strategy would thus be the optimal one.

This test shows the predictive capability of the controller. It was because of the greater optimisation horizon, that it took the decision to fill tank1 above 5m. With a smaller horizon, as for the first simulation, the algorithm did not have a “far” enough view into the future to know that tank2 would fill faster due to the overflow, if the level in tank1 exceeded 5m. Consequently it had no reason to overfill tank1, except maybe for greater head, which would cause a larger flow out of the bottom of tank1. This would however depend on tuning. With the set tuning parameters, the algorithm however did not allow the level in tank1 to exceed the setpoint by too much in preference for greater head. The predictive nature, for the extended horizon, was also apparent when the controller made changes prior to both states approach to setpoint. The flow into tank1 (F_o), contrary to the sequence with a two step optimisation horizon, was set to zero before both states reached setpoint. (time step 26 in figure 8.15)

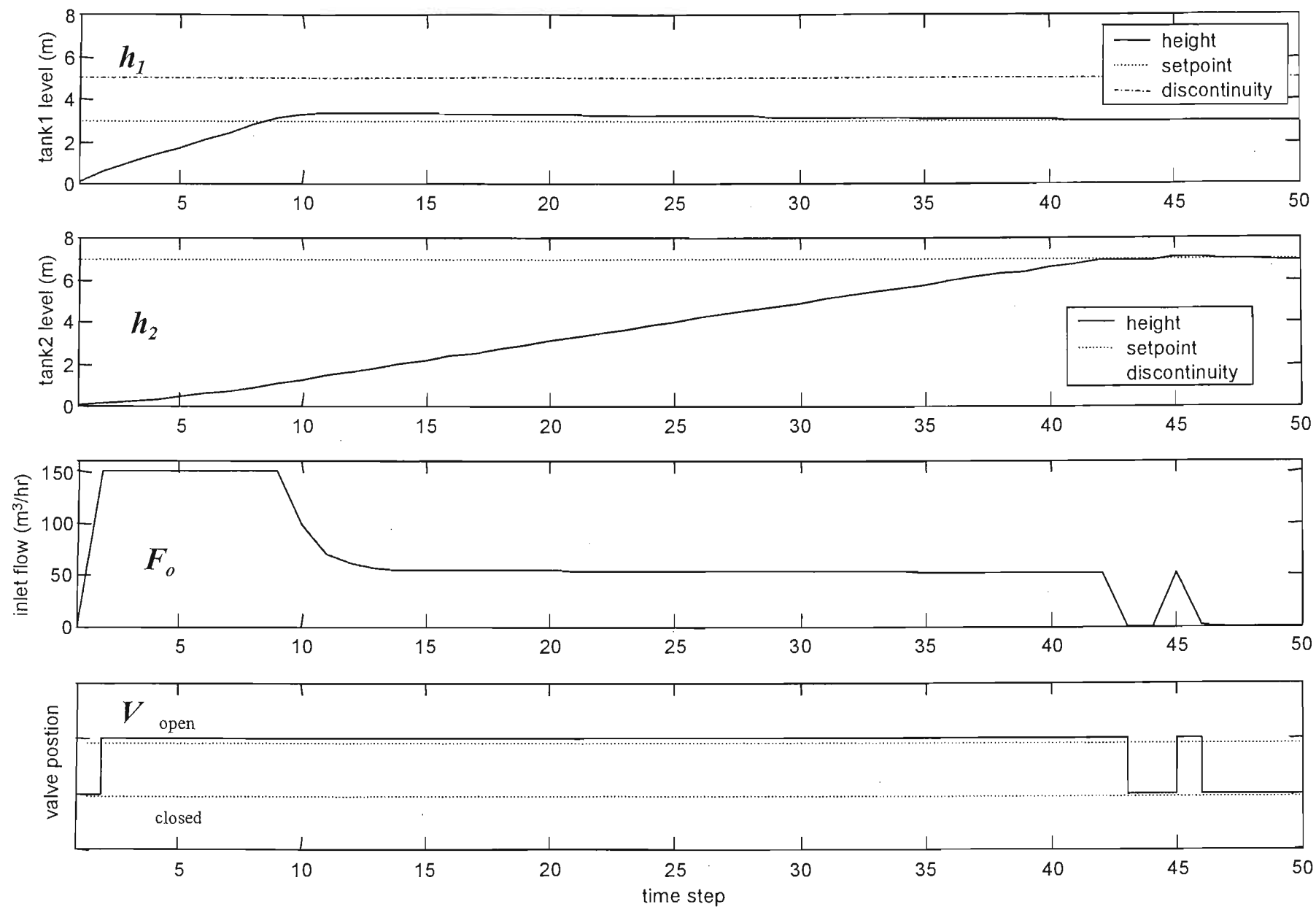


Figure 8.14 Two tank system filling with controller allowed 2 step optimisation horizon and single control move

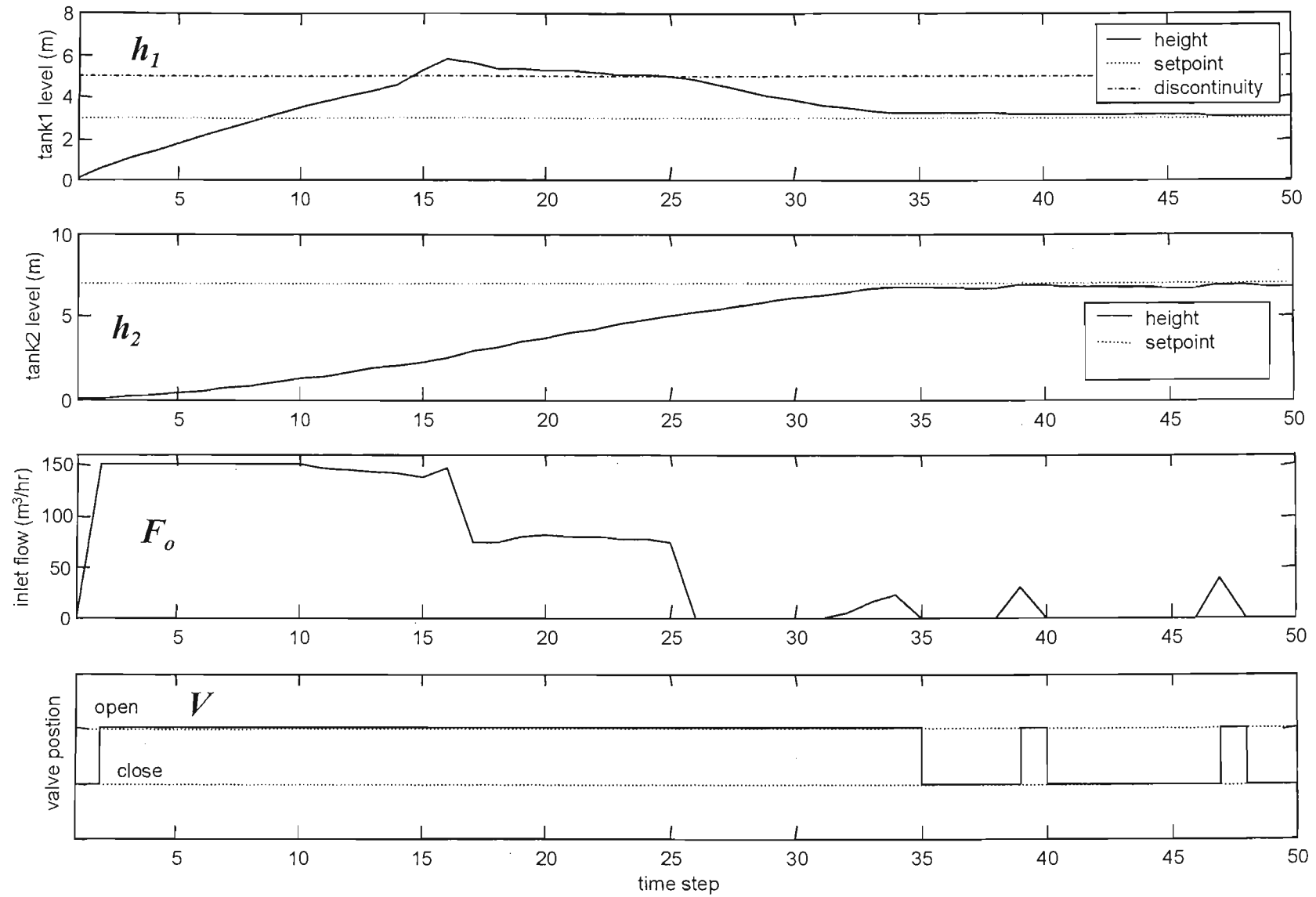


Figure 8.15 Two tank system filling with controller allowed 5 step optimisation horizon and 2 control moves

8.4 Conclusion

In order to correctly predict the output trajectories of systems with physicochemical discontinuities, model predictive controllers have to be able to switch model states. These switches are based on when output variables cross points of discontinuity. This chapter presents the design and validation of a MPC aimed at controlling the level in a tank that had discontinuities in the form of cross-sectional area and an overflow at a given height.

With the aid of binary variables, the control algorithm was able to switch the model state when the system state crossed certain threshold values. State space models of the system were used for future closed loop predictions instead of the usual convolution model. This is because state space models require minimal information concerning the history of the system. As a result each regime of operation is dealt with separately, with the only interaction occurring when the state “jumps” across the boundary. Convolution models on the other hand, require information from the past in order to predict the future and when it comes to simulation across boundaries, there exists no systematic approach to expressing the effect of moves made in one regime on the states in another.

Closed loop simulations were performed by interfacing MATLAB and GAMS. The tests show that the controller was able to successfully switch the model state over the optimisation horizon. This allows it to maintain its most important property which is control based on future predictions.

9. Conclusions and Recommendations

This chapter comprises two parts. Firstly the conclusion summarises in point form, the main findings from this research study. Thereafter the recommendation section raises questions and presents further ideas regarding the extension of this work.

9.1 Conclusions

From this investigation into MPC of hybrid systems, the following can be concluded:

1. The DMC algorithm can be modified to accommodate integer variables. This mixed integer control algorithm can be used for the optimal control of systems that have discontinuous inputs. However where there are discontinuities on the system output, DMC was found not to be applicable. In this case state space models are necessary.
2. Control of the interacting tank system, showed that the mixed integer predictive controller was able to optimally select the operating position of an input that could only take on positions from an integer set, while also accounting for a continuous input. The algorithm was shown to retain all the features from standard DMC i.e. constraint handling, multivariable interaction and tuning capability i.e. as tested with 1,2,3 optimised moves; setpoint error weights 1,10; and move suppression weights 0.1,1,10.
3. Through appropriate structuring of the $\Delta\bar{m}$ vector in equation (3.4), the control algorithm could be used to optimally switch the modes of operation of a thermal circuit comprising 3 modes. In this case the step response data for the convolution model had to be generated by switching between the modes of operation. In this format, the controller also possessed the ability to deal with asymmetric responses.
4. On a gasifier system, the controller was shown to be able to prioritise and sequence events in an optimal way. The DMC strategy proved to be useful, as its characteristics i.e. tuning parameters and ability to deal with constraints could be used to bring about the desired sequencing effect. It was found that constraints may become violated, due to mismatch or timing of the control cycle. In this case a soft constraint was employed. A soft constraint prevents the algorithm from failing, while at the same time ensuring that the constraint is still part of the optimisation.
5. A model predictive control approach was furthermore used to control systems with physicochemical discontinuities e.g. a hypothetical tank system. Control of such systems is challenging for predictive controllers, as the state of the internal model has to be switched at certain threshold values. Being able to correctly predict the future behaviour of a system was shown to improve its dynamic performance, as it allows for future planning and anticipation.

6. A convolution model, as warranted by the DMC framework, could not be adapted to control systems that have discrete outputs owing to physicochemical discontinuities. Discrete outputs partition the operation of a system into distinct regimes of operation. DMC is unable to deal with regimes of operation because of the manner in which the convolution model is generated. A convolution model is linear in nature and is generated from continuous step responses. When a new regime is entered into, their impact in the new regime will change. Thus information from the past cannot be used, because it pertains to a different model. In this instance, state space equations describing the system are proposed while the integration cycle is initiated from the past state of the system.

9.2 Recommendations

1. GAMS proved to be a satisfactory environment for the design and development of the mixed integer controller. Together with being easily interfaced with an external package, the program was able to deal with both dynamic and mixed integer aspects necessary for hybrid system modelling and control. GAMS also allows selection from a wide range of internal solvers that could be used for the numerical solution of the MIDO problem. There were however times when GAMS could not solve for an integer solution at a specific control cycle i.e. the problem became infeasible or the allowed number of iterations was exceeded. In the event of there not being a solution, there does not exist any form of error reporting nor any time out mechanism. This is of concern in real time control, because the plant awaits inputs from the controller and when a solution cannot be found, then there will be a break in the control loop. Suggested strategies to account for the event when an integer solution may not exist are: to reuse the inputs from the previous time step; or if more than 1 optimal move is being solved for, the second move from the previous time step could then be used.
2. This thesis presents a model predictive approach to prompt selections, initiate switches and direct sequences. There exists uncertainty as to whether this approach is superior to a heuristic approach and if so, by what measure. A heuristic approach involves “hard wiring” of the control algorithm based on specific rules derived from the process. In some instances this might provide more effective control. On the other hand, a MPC approach is more general and therefore easier to “port” between different applications.
3. The thermal circuit provides an opportunity to develop a control algorithm based on a hierarchy. In this case, separate controllers, each with its own model, could be used for controlling each regime. A separate controller can be assigned to control each individual mode and these can be switched when the system changes mode. There will thus be a need for a supervisory controller, which monitors the entire structure and is able to select the appropriate controller based on the specific mode. If implemented, the question of optimality when switching control strategies according to modes, needs to be investigated.

4. DMC was not able to solve the real problem of coal lock sequencing in industry. This is because standard DMC is linear, producing the same change at any operating point, while for the coal lock problem, temperature returns to a base value, regardless of the initial state. A model predictive approach to this sequencing problem is still thought to be a good solution. It is suggested that when a sequence is initiated, then the internal model is reset and the state returns to a base value. From this value, it should start to follow the usual trajectory until it is reset again.
5. Although MPC was shown to deal with systems having physicochemical discontinuities, there is still uncertainty about crossing the point of discontinuity. In this thesis, the model from the initial regime was used to transfer the state from the present regime into the adjacent one. Other researchers use a reduction in integration step in order to determine the exact point of crossing. How much more accurate this approach is, is uncertain. In this thesis, it was the controller that initiated the crossing of the boundary when model states were switched. A further condition to be considered is when between control updates, the system output state itself in feedback, causes the controller to switch model states.
6. This research considers using DMC as a direct means of controlling hybrid systems. However in industry MPC usually operates from a layer that resides above the base layer comprising PID controllers. Further research may include an investigation into the effect of an intermediate PID layer between the hybrid process and proposed algorithm.
7. This research showed that the available manipulated variables for a predictive controller could be generalised in a useful way for hybrid systems. An input could have discrete levels or be continuous. Furthermore selected discrete/continuous combinations and transitions could be disallowed by choice. It is felt that these concepts could be taken further in a generalised approach, even being influenced by the distinct physicochemical behavioural states that the controlled process might find itself in.

References

⊗ Reference cited, by not consulted.

ALLGOR RJ, BARTON PI (1997). *Mixed-Integer Dynamic optimisation*. Computers & Chem Engng **21** (Suppl) S451 – S456.

ANTSAKLIS PS (1998). *Guest Editorial Hybrid Control Systems: An introductory discussion to the special issue*. IEEE Transactions on Automatic Control **43**(4): p.457-459.

BANSAL V, PERKINS JD, PISTIKOPOULOS EN, ROSS R., VAN SCHIJNDEL JMG (2000). *Simultaneous design and control optimisation under uncertainty*. Computers & Chem Engng **24**: 261-266.

BARTON PI, ALLGOR RJ, FEEHERY WF, GALANS (1998). *Dynamic Optimization in a Discontinuous World*. Ind.Eng.Chem.Res **37**: p.966-981

BARTON PI, PANTELIDES CC (1994). *Modeling of Combined Discrete / Continuous Processes*. AIChE Journal Vol.**40**, No.6, p.966-979

BEMPORAD A, MORARI M, DUA V, PISTIKOPOULOS EN (2000). *The Explicit Solution of Model Predictive Control via Multiparametric Quadratic Programming*. (<http://control.ethz.ch/research/publications>).

BEMPORAD A, BORRELLI F, MORARI M (2000). *Piecewise Linear Optimal Controllers for Hybrid Systems*. (<http://control.ethz.ch/research/publications>).

BEMPORAD A, BORRELLI F, MORARI M (2000). *Optimal Controllers for Hybrid Systems: Stability and Piecewise Linear Explicit Form*. (<http://control.ethz.ch/research/publications>).

BEMPORAD A, MORARI M (1998). *Control of systems integrating logic, dynamics and constraints*. (<http://control.ethz.ch/research/publications>).

BORRELLI F, BEMPORAD A, FODOR M, HROVAT D (2001). *A hybrid approach traction control*. (<http://control.ethz.ch/research/publications>).

CHANG TS, SEBORG DS [⊗] (1983). *A linear programming approach for multivariable feedback control with inequality constraints*. Int.J.Control **37**: p.583-597.

CUTLER CR, RAMAKER BL[®] (1980). *Dynamic Matrix Control - A computer control algorithm*. AIChE National Meeting Houston Texas.

DEGHAYE SW, GUIAMBA I, MULHOLLAND M (2000). *Enhancements of Dynamic Matrix Control Applied to a Liquid-liquid Extractor*. mulholla@nu.ac.za.

FERRARI-TRECATE G, GALLETTI E, LETIZIA P, SPEDICATO M, MORARI M, ANTOINE M (2002). *Modeling and Control of Co-generation Power Plants: A Hybrid System Approach*. (<http://control.ethz.ch/research/publications>).

FERRARI-TRECATE G, MIGNONE D, CASTAGNOLI D, MORARI M (2000). *Mixed logic Dynamical Model of a Hydroelectric Power Plant*. (<http://control.ethz.ch/research/publications>).

FERRIS MC (1999). *MATLAB and GAMS: Interfacing Optimization and Visualization Software*. <http://www.cs.wisc.edu/math-prog/matlab.html>.

FLOUDAS CA[®] (1995). *Nonlinear and mixed integer optimisation Fundamentals and Applications*. Oxford University Press: Oxford U.K.

FERRARI-TRECATE G, MIGNONE D, MORARI M (2002). *Moving Horizon Estimation for Hybrid systems*. (<http://control.ethz.ch/research/publications>).

GARCIA CE, MORSHEDI AM[®]. (1984). *Quadratic Programming Solution of Dynamic Matrix Control (QDMC)*. Proc.Am.Control Conf. San Diego California.

GARCIA CE, PRETT DM, MORARI M (1989). *Model Predictive Control: Theory and Practice – a Survey*. *Automatica* 25(3): p.335-348.

GUIAMBA I (2001). *Adaptive Dynamic Matrix Control of a Multivariable training plant*. MSc thesis, School of Chemical Engineering, University of Natal, Durban.

HENZINGER TA (1996). *The Theory of Hybrid Automata*. (<http://www-er.df.op.dlr.de/cacsd/hds/publications.shtml>)

KERRIGAN EC, BEMPORAD A, MIGNONE D, MORARI M, MACIEJOWSKI JM (2000). *Multi-objective Prioritisation and Reconfiguration for the control of Constrained Hybrid Systems*. (<http://control.ethz.ch/research/publications>).

- KOWALEWSKI S, FRITZ M, GRAF H, HOFFMANN I, PREUBIG J, REMELHE J, SIMON S, TRESELER H (1997). *A case study in tool-aided analysis of discretely controlled continuous systems: The two tanks problem.* (<http://www-er.df.op.dlr.de/cacsd/hds/publications.shtml>).
- LACAVE B (2001). *Modelling and Control of a co-current sugar dryer.* MSc thesis, School of Chemical Engineering, University of Natal, Durban.
- LENNARTSON B, EGARDT B, TITTUS M (1994). *Hybrid Systems in Process Control.* (Proceedings of the 33rd IEEE Conference on Decision and Control IEEE vol.4, p.3587-3592).
- LENNARTSON B, EGARDT B, TITTUS M, PETTERSSON S (1996). *Hybrid Systems in Process Control.* IEEE Control Systems Magazine, vol.16, no.5, Oct. 1996. p. 45-56.
- LYGEROS J, PAPPAS G, SASTRY S (1999). *An Introduction to hybrid system modeling, analysis, and control.* <http://www-er.df.op.dlr.de/cacsd/hds/publications.shtml>
- MALER DO (1997). *Hybrid -EC-US043, Tools for the Analysis of Hybrid systems.* <http://albion.ncl.ac.uk/esp-syn/text/ec-us043.html>
- MIGNONE D, BEMPORAD A, MORARI M (1999). *A framework for Control, Fault Detection, State Estimation and Verification of Hybrid Systems.* (<http://control.ethz.ch/research/publications>).
- MOHINDEEN MJ, PERKINS JD, PISTOKOPOULOS EN 1996a. *Optimal design of dynamic systems under uncertainty.* AIChe J. 42 p 2251-2272.
- MOHINDEEN MJ, PERKINS JD, PISTOKOPOULOS EN 1996b. *Optimal synthesis and design of systems under uncertainty.* Computers and Chem Engng 20: S895-S900.
- MOHINDEEN MJ, PERKINS JD, PISTOKOPOULOS EN 1997. *Towards an efficient numerical procedure for mixed integer optimal control.* Computers and Chem Engng 21: S457-S462.
- MORARI M, LEE JH (1999). *Model predictive control: Past, Present and Future.* Computers and Chem Engng 23: 667-682.
- MORSHEDI AM, CUTLER CR, SKROVANEK TA [®] (1985). *Optimal Solution of Dynamic Matrix Control with linear programming techniques (LDMC).* Proc. Am. Control Conf., Boston, Massachusetts: 199-208.
- MOSTERMAN PJ (1999). *An overview of Hybrid Simulation Phenomena and their support by simulation packages.* (<http://www-er.df.op.dlr.de/cacsd/hds/publications.shtml>).

- MULHOLLAND M, NAROTAM NK (1996) [®]. *Constrained predictive Control of a Counter-current Extractor.*” mulholla@nu.ac.za
- MULHOLLAND M, PROSSER JA(1999).*Linear Dynamic Matrix Control of a Distillation Column.* mulholla@nu.ac.za
- NIEBERT P (2001). The VHS Project Homepage. (<http://www-verimag.imag.fr/VHS/>).
- OTTER M, SCHLEGAL C, ELMQVIST H (1997). *Modeling and realtime simulation of an automatic gearbox using modelice.* (<http://www-er.df.op.dlr.de/cacsd/hds/publications.shtml>).
- PEPYNE DL, CASSANDRAS CG (2000). *Optimal Control of Hybrid Systems in Manufacturing.* *Proceedings of the IEEE*, vol.88, no.7, July 2000. p. 1108-23.
- PRETT DM, GILLETTE [®] (1979). *Optimisation and constrained multivariable control of a catalytic cracking unit.* *AIChE National Mtg, Houston, Texas; also Proc. Joint Aut.Control Conf., San Francisco, California.*
- PREU K, Le Lann M,Cabassud M,Anne-Archard G (2002). *Implementation Procedure of an advanced supervisory and control strategy in the pharmaceutical industry.* mvlelann@laas.fr.
- QIN SJ, BADGWELL TA(1999). *An Overview of Nonlinear Model Predictive Control Applications.* (<http://www-er.df.op.dlr.de/cacsd/hds/publications.shtml>).
- RAMAN R, GROSSMANN IE (1991). *Relation between MINLP modeling and logical inference for chemical process synthesis.* *Computers & Chem Engng* **15**(2): p.73-84.
- RAMAN R, GROSSMANN IE(1992).*Integration of logic and heuristic knowledge in MINLP optimization for process synthesis.* *Computers & Chem Engng* **16**(3): p.155-171.
- RIEDINGER PZC, KRANTZ F (1999). *Time Optimal Control of Hybrid Systems.* *Proceedings of the 1999 American Control Conference* vol.4. Piscataway, NJ, USA: IEEE, 1999 vol.4. p. 2466-70.
- SEDGHI B, SRINIVASAN B, LONGCHAMP R (2002). *Control of Hybrid systems via Dehybridization.* *Proceedings of American Control Conference Anchorage, AK May 8-10 2002.*
- SLUPPHAUG O, FOSS BA [®] (1997a). *Model predictive Control for a class of Hybrid Systems.* *Proc.European Control Conf. Brussels,Belgium*
- SLUPPHAUG O, VADA J, FOSS BA [®] (1997b). *MPC in systems with continuous and discrete control inputs.* *Proc. American Control Conf.. Albuquerque,NM,USA.*

TORRISI FD, MIGNONE D, MORARI M (2001). *Time-Optimal Control of Hybrid Systems: Heuristic vs Systematic design*. (<http://control.ethz.ch/research/publications>).

TYLER ML, MORARI M (1999). *Propositional logic in control and monitoring problems*. *Automatica* **35**(4): p.565-582.

VISWANATHAN J, GROSSMANN IE (1990). A Combined Penalty Function and Outer-Approximation Method For MINLP Optimization. *Computers & Chem Engng* **14**(7): p.769-782.

A.1 Extract from file CEXTObject for interfacing SCAD and GAMS

(with acknowledgements to T.Brazier and M.Mulholland)

```
void CEXTObject::TimerUpdate (DWORD dwTime)
{
    iNumInputs = m_csInputStreamNames.GetSize (); // inputs to EXTOBJ (PV's)
    iNumOutputs = m_csOutputStreamNames.GetSize (); // outputs from EXTOBJ (MV's)

    // EXT update code
    double DeltaTime = (double)((int)dwTime - (int)m_dwPrevTime) / 1000.0;
    if (DeltaTime >= m_fTimeInterval)
    {
        // update time
        m_dwPrevTime += (int)m_fTimeInterval * 1000;

        // cascade m_pPastOutputChanges
        for (int i = 1; i <= iNumOutputs; i++)
            for (int j = 1; j < m_iStoredSteps; j++)
                (*m_pPastOutputChanges)[(j-1) * iNumOutputs + i] =
                    (*m_pPastOutputChanges)[j * iNumOutputs + i];

        // cascade m_pPastOutputLevels
        for (i = 1; i <= iNumOutputs; i++)
            for (int j = 1; j < m_iStoredSteps; j++)
                (*m_pPastOutputLevels)[(j-1) * iNumOutputs + i] =
                    (*m_pPastOutputLevels)[j * iNumOutputs + i];

        // move up most recent output
        for (i = 1; i <= iNumOutputs; i++)
        {
            (*m_pPastOutputChanges)[(m_iStoredSteps-1) * iNumOutputs + i] =
                ((CStream*)m_pOutputStreams[i-1])>Read ("User") - (*m_pPreviousOutput)[i]; //Was m_csName

            (*m_pPastOutputLevels)[(m_iStoredSteps-1) * iNumOutputs + i] = (*m_pPreviousOutput)[i];
            (*m_pPreviousOutput)[i] = ((CStream*)m_pOutputStreams[i-1])>Read ("User"); // Was m_csName #####MM010225
        }

        // cascade m_pPastInputLevels
        for (i = 1; i <= iNumInputs; i++)
            for (int j = 1; j < m_iStoredSteps; j++)
                (*m_pPastInputLevels)[(j-1) * iNumInputs + i] = (*m_pPastInputLevels)[j * iNumInputs + i];

        // store previous and find current state vector
        for (i = 1; i <= iNumInputs; i++)
        {
            (*m_pPastInputLevels)[(m_iStoredSteps-1) * iNumInputs + i] = (*m_pPreviousInput)[i];
            (*m_pPreviousInput)[i] = ((CStream*)m_pInputStreams[i-1])>Read ("User"); // Was m_csName #####MM010225
        }

        if (m_bEnabled) // actually on-line
        {
            // Update the external call index
            ExtCallIndex = ExtCallIndex + 1;

            // calculate InputMIN, InputMAX, MeasuredInput and InputSetpoint Vectors
            CVector MeasuredInputs (iNumInputs);
            CVector InputSetpoints (iNumInputs);
            CVector InputMIN (iNumInputs);
            CVector InputMAX (iNumInputs);
            CVector InputSPdevWt (iNumInputs);
            CVector PresentOutputs (iNumOutputs);
            CVector OutputMIN (iNumOutputs);
            CVector OutputMAX (iNumOutputs);
            CVector OutputChangeMAX (iNumOutputs);
            CVector OutputMoveWt (iNumOutputs);
            CVector NewOutputs (iNumOutputs);

            for (int i = 1; i <= iNumInputs; i++)
            {
                MeasuredInputs[i] = ((CStream*)m_pInputStreams[i-1])>Read (m_csName);
                InputSetpoints[i] = d(m_fSetpoints[i-1]);
                InputMIN[i] = d(m_fInputMins[i-1]);
                InputMAX[i] = d(m_fInputMaxs[i-1]);
                InputSPdevWt[i] = d(m_fInputSPdevWts[i-1]);
            }
        }
    }
}
```



```

for (i = 1; i <= iNumOutputs; i++)
{
    PresentOutputs[i] = ((Ciostream*)m_pOutputStreams[i-1])>>Read ("User"); // Was m_csName #####MM010225
    OutputMIN[i] = d(m_fOutputMins[i-1]);
    OutputMAX[i] = d(m_fOutputMaxs[i-1]);
    OutputChangeMAX[i] = d(m_fMaxChanges[i-1]);
    OutputMoveWt[i] = d(m_fOutputMoveWts[i-1]);
}

if (m_bAlgoDMC)
{
    //====START
    ofstream EXTput(m_cEXTputFile, ios::out, filebuf::sh_none);
    if (EXTput)
    {
        //-----construct GAMS file-----
        EXTput.close();
        FILE* EXTputFile = fopen (m_cEXTputFile, "w");

        // Write to EXTputFile
        fprintf(EXTputFile, ""GAMS Program for Real-time Closed-loop Optimisation\n\n");
        fprintf(EXTputFile, "$ONEMPTY\n\n"); // To allow declaration with empty sets

        // Index to check whether return file is response to this spec
        fprintf(EXTputFile, "Scalar ExtCallIndex / %d /\n\n", ExtCallIndex);

        // Present Inputs
        fprintf(EXTputFile, ""Real-time data input from SCAD\n\n");
        fprintf(EXTputFile, ""Present Inputs x(nx)\n");
        fprintf(EXTputFile, "Scalar rx / %d /\n", iNumInputs);
        if (iNumInputs < 10)
        {
            fprintf(EXTputFile, "Set ix / ix1*ix%d /\n", iNumInputs);
        }
        else
        {
            fprintf(EXTputFile, "Set ix / ix01*ix%d /\n", iNumInputs);
        }

        fprintf(EXTputFile, "Parameter x(ix) /");
        for (i=1; i<=iNumInputs; i++)
        {
            if (MeasuredInputs[i] != 0.0)
            {
                if ((iNumInputs < 10) || (i > 9))
                    fprintf(EXTputFile, "\nix%d %12.4e", i, MeasuredInputs[i]);
                else
                    fprintf(EXTputFile, "\nix0%d %12.4e", i, MeasuredInputs[i]);
            }
        }
        fprintf(EXTputFile, " /\n\n");

        // Present Outputs
        fprintf(EXTputFile, ""Present Outputs u(nu)\n");
        fprintf(EXTputFile, "Scalar nu / %d /\n", iNumOutputs);
        if (iNumOutputs < 10)
        {
            fprintf(EXTputFile, "Set iu / iu1*iu%d /\n", iNumOutputs);
        }
        else
        {
            fprintf(EXTputFile, "Set iu / iu01*iu%d /\n", iNumOutputs);
        }

        fprintf(EXTputFile, "Parameter u(iu) /");
        for (i=1; i<=iNumOutputs; i++)
        {
            if (PresentOutputs[i] != 0.0)
            {
                if ((iNumOutputs < 10) || (i > 9))
                    fprintf(EXTputFile, "\niu%d %12.4e", i, PresentOutputs[i]);
                else
                    fprintf(EXTputFile, "\niu0%d %12.4e", i, PresentOutputs[i]);
            }
        }
        fprintf(EXTputFile, " /\n\n");

        // Input Setpoints
        fprintf(EXTputFile, ""Input Setpoints xsp(nx)\n");
        fprintf(EXTputFile, "Parameter xsp(ix) /");
        for (i=1; i<=iNumInputs; i++)
        {
            if (InputSetpoints[i] != 0.0)
            {
                if ((iNumInputs < 10) || (i > 9))
                    fprintf(EXTputFile, "\nix%d %12.4e", i, InputSetpoints[i]);
                else
                    fprintf(EXTputFile, "\nix0%d %12.4e", i, InputSetpoints[i]);
            }
        }
        fprintf(EXTputFile, " /\n\n");

        // Input Maxima
        fprintf(EXTputFile, ""Input Maxima xmax(nx)\n");
    }
}

```

```

fprintf(EXTputFile,"Parameter xmax(ix) /");
for (i=1; i<=iNumInputs; i++)
{
    if (InputMAX[i] != 0.0)
    {
        if ((iNumInputs<10)&&(i>9))
            fprintf(EXTputFile,"nix%d %12.4e",i,InputMAX[i]);
        else
            fprintf(EXTputFile,"nix0%d %12.4e",i,InputMAX[i]);
    }
}
fprintf(EXTputFile," /\n\n");

// Input Minima
fprintf(EXTputFile,"Input Minima xmin(nx)\n");
fprintf(EXTputFile,"Parameter xmin(ix) /");
for (i=1; i<=iNumInputs; i++)
{
    if (InputMIN[i] != 0.0)
    {
        if ((iNumInputs<10)&&(i>9))
            fprintf(EXTputFile,"nix%d %12.4e",i,InputMIN[i]);
        else
            fprintf(EXTputFile,"nix0%d %12.4e",i,InputMIN[i]);
    }
}
fprintf(EXTputFile," /\n\n");

// Input SPdevWts
fprintf(EXTputFile,"Input Setpoint Deviation Weights xw(rx)\n");
fprintf(EXTputFile,"Parameter xw(ix) /");
for (i=1; i<=iNumInputs; i++)
{
    if (InputSPdevWt[i] != 0.0)
    {
        if ((iNumInputs<10)&&(i>9))
            fprintf(EXTputFile,"nix%d %12.4e",i,InputSPdevWt[i]);
        else
            fprintf(EXTputFile,"nix0%d %12.4e",i,InputSPdevWt[i]);
    }
}
fprintf(EXTputFile," /\n\n");

// Output Maxima
fprintf(EXTputFile,"Output Maxima umax(nu)\n");
fprintf(EXTputFile,"Parameter umax(iu) /");
for (i=1; i<=iNumOutputs; i++)
{
    if (OutputMAX[i] != 0.0)
    {
        if ((iNumOutputs<10)&&(i>9))
            fprintf(EXTputFile,"niu%d %12.4e",i,OutputMAX[i]);
        else
            fprintf(EXTputFile,"niu0%d %12.4e",i,OutputMAX[i]);
    }
}
fprintf(EXTputFile," /\n\n");

// Output Minima
fprintf(EXTputFile,"Output Minima umin(nu)\n");
fprintf(EXTputFile,"Parameter umin(iu) /");
for (i=1; i<=iNumOutputs; i++)
{
    if (OutputMIN[i] != 0.0)
    {
        if ((iNumOutputs<10)&&(i>9))
            fprintf(EXTputFile,"niu%d %12.4e",i,OutputMIN[i]);
        else
            fprintf(EXTputFile,"niu0%d %12.4e",i,OutputMIN[i]);
    }
}
fprintf(EXTputFile," /\n\n");

// Output Change Maximum
fprintf(EXTputFile,"Output Change Maxima dumax(nu)\n");
fprintf(EXTputFile,"Parameter dumax(iu) /");
for (i=1; i<=iNumOutputs; i++)
{
    if (OutputChangeMAX[i] != 0.0)
    {
        if ((iNumOutputs<10)&&(i>9))
            fprintf(EXTputFile,"niu%d %12.4e",i,OutputChangeMAX[i]);
        else
            fprintf(EXTputFile,"niu0%d %12.4e",i,OutputChangeMAX[i]);
    }
}
fprintf(EXTputFile," /\n\n");

// Output Move Weights
fprintf(EXTputFile,"Output Move Weights uw(nu)\n");
fprintf(EXTputFile,"Parameter uw(iu) /");
for (i=1; i<=iNumOutputs; i++)
{
    if (OutputMoveWt[i] != 0.0)
    {
        if ((iNumOutputs<10)&&(i>9))
            fprintf(EXTputFile,"niu%d %12.4e",i,OutputMoveWt[i]);
        else
            fprintf(EXTputFile,"niu0%d %12.4e",i,OutputMoveWt[i]);
    }
}
}

```

```

fprintf(EXTputFile, "\n\n");

//Number of Stored Steps
fprintf(EXTputFile, "Set for past limes stored\n");
fprintf(EXTputFile, "Scalar nnt / %d /\n\n", m_iStoredSteps);

// Set for past inputs stored
int m_inxnt = m_iStoredSteps*iNumInputs;

//Number of Stored Steps
fprintf(EXTputFile, "Set for past inputs stored\n");
fprintf(EXTputFile, "Scalar nrxnt / %d /\n\n", m_inxnt);
if (m_inxnt<10)
{
    fprintf(EXTputFile, "Set ixt / ixt1*ixt%d /\n", m_inxnt);
}
else
{
    //#### Beware if this goes over 99 !!!
    fprintf(EXTputFile, "Set ixt / ixt01*ixt%d /\n", m_inxnt);
}
fprintf(EXTputFile, "\n");

// Past Input Values
fprintf(EXTputFile, "Previous Input Values xt(ixt)\n");
fprintf(EXTputFile, "Parameter xt(ixt) /");
for (i=1; i<=m_inxnt; i++)
{
    if ((*m_pPastInputLevels)[i] != 0.0)
    {
        if ((m_inxnt<10))(>9))
            fprintf(EXTputFile, "\nixt%d %12.4e", i, (*m_pPastInputLevels)[i]);
        else
            fprintf(EXTputFile, "\nixt0%d %12.4e", i, (*m_pPastInputLevels)[i]);
    }
}
fprintf(EXTputFile, "\n\n");

// Set for past outputs stored
int m_inunt = m_iStoredSteps*iNumOutputs;
fprintf(EXTputFile, "Set for past outputs stored\n");
fprintf(EXTputFile, "Scalar nunt / %d /\n\n", m_inunt);
if (m_inunt<10)
{
    fprintf(EXTputFile, "Set iut / iut1*iut%d /\n", m_inunt);
}
else
{
    //#### Beware if this goes over 99 !!!
    fprintf(EXTputFile, "Set iut / iut01*iut%d /\n", m_inunt);
}
fprintf(EXTputFile, "\n");

// Past Output Values
fprintf(EXTputFile, "Previous Output Values ut(iut)\n");
fprintf(EXTputFile, "Parameter ut(iut) /");
for (i=1; i<=m_inunt; i++)
{
    if ((*m_pPastOutputLevels)[i] != 0.0)
    {
        if ((m_inunt<10))(>9))
            fprintf(EXTputFile, "\niut%d %12.4e", i, (*m_pPastOutputLevels)[i]);
        else
            fprintf(EXTputFile, "\niut0%d %12.4e", i, (*m_pPastOutputLevels)[i]);
    }
}
fprintf(EXTputFile, "\n\n");

// Past Output Changes
fprintf(EXTputFile, "Previous Output Changes dut(iut)\n");
fprintf(EXTputFile, "Parameter dut(iut) /");
for (i=1; i<=m_inunt; i++)
{
    if ((*m_pPastOutputChanges)[i] != 0.0)
    {
        if ((m_inunt<10))(>9))
            fprintf(EXTputFile, "\niut%d %12.4e", i, (*m_pPastOutputChanges)[i]);
        else
            fprintf(EXTputFile, "\niut0%d %12.4e", i, (*m_pPastOutputChanges)[i]);
    }
}
fprintf(EXTputFile, "\n\n");

fclose(EXTputFile);
//-----
}

// CALL the EXT program HERE

char *args[2];
args[0] = "c:\\gams\\gams19.0\\gams.exe";
args[1] = "c:\\research\\gams\\gams_scad\\Scad.gms";

//
//
//
//
//
//
char *environment[] =
{
    "curdir=c:\\research\\gams\\gams_scad",
    "inputdir=c:\\research\\gams\\gams_scad",
    "inputdir=c:\\research\\gams\\gams_scad",
    "libindir=c:\\research\\gams\\gams_scad",
    "putdir=c:\\research\\gams\\gams_scad",

```

```

//          "sysdir=c:\\gams\\gams19.0",
//          "relpath=0",
//          "suppress=0", //### 1
//          NULL
//      };
// ##### Original _spawnlp( _P_WAIT, args[0], args[0], args[1], NULL, environment );

int spawnflag = _spawnlp( _P_WAIT, args[0], args[0], args[1], NULL );

// char Buffer[80];
// sprintf(Buffer, "ermo = %d\n",ermo);
// ((CMainWnd*)AfxGetMainWnd ())->SetStatusBarText (Buffer);
// MessageBeep (MB_ICONASTERISK);

// Test if EXT get File exists and is not still open
ifstream EXTget(m_cEXTgetFile, ios::nocreate, filebuf::sh_none);
if (EXTget)
{
    //-----
    EXTget.close();
    int ExtCallIndex_returned;
    FILE* EXTgetFile = fopen (m_cEXTgetFile, "r");
    fscanf(EXTgetFile,"%d",&ExtCallIndex_returned);
    if (ExtCallIndex_returned == ExtCallIndex)
    {
        for (i = 1; i <= iNumOutputs; i++)
        {
            fscanf(EXTgetFile,"%lf",&NewOutputs[i]);
        }
    }
    else
    {
        char Buffer[80];
        sprintf(Buffer, "EXT Call Index MISMATCH ! : Returned %d != Sent %d\n",ExtCallIndex_returned,ExtCallIndex,ermo);
        ((CMainWnd*)AfxGetMainWnd ())->SetStatusBarText (Buffer);
        MessageBeep (MB_ICONASTERISK);
    }
    fclose(EXTgetFile);
}
else
{
    // Notify user that file is not ready
    char Buffer[80];
    sprintf(Buffer, "Return file is not ready for Index Call Sent ! : %d\n",ExtCallIndex);
    ((CMainWnd*)AfxGetMainWnd ())->SetStatusBarText (Buffer);
    MessageBeep (MB_ICONASTERISK);
    //-----
}

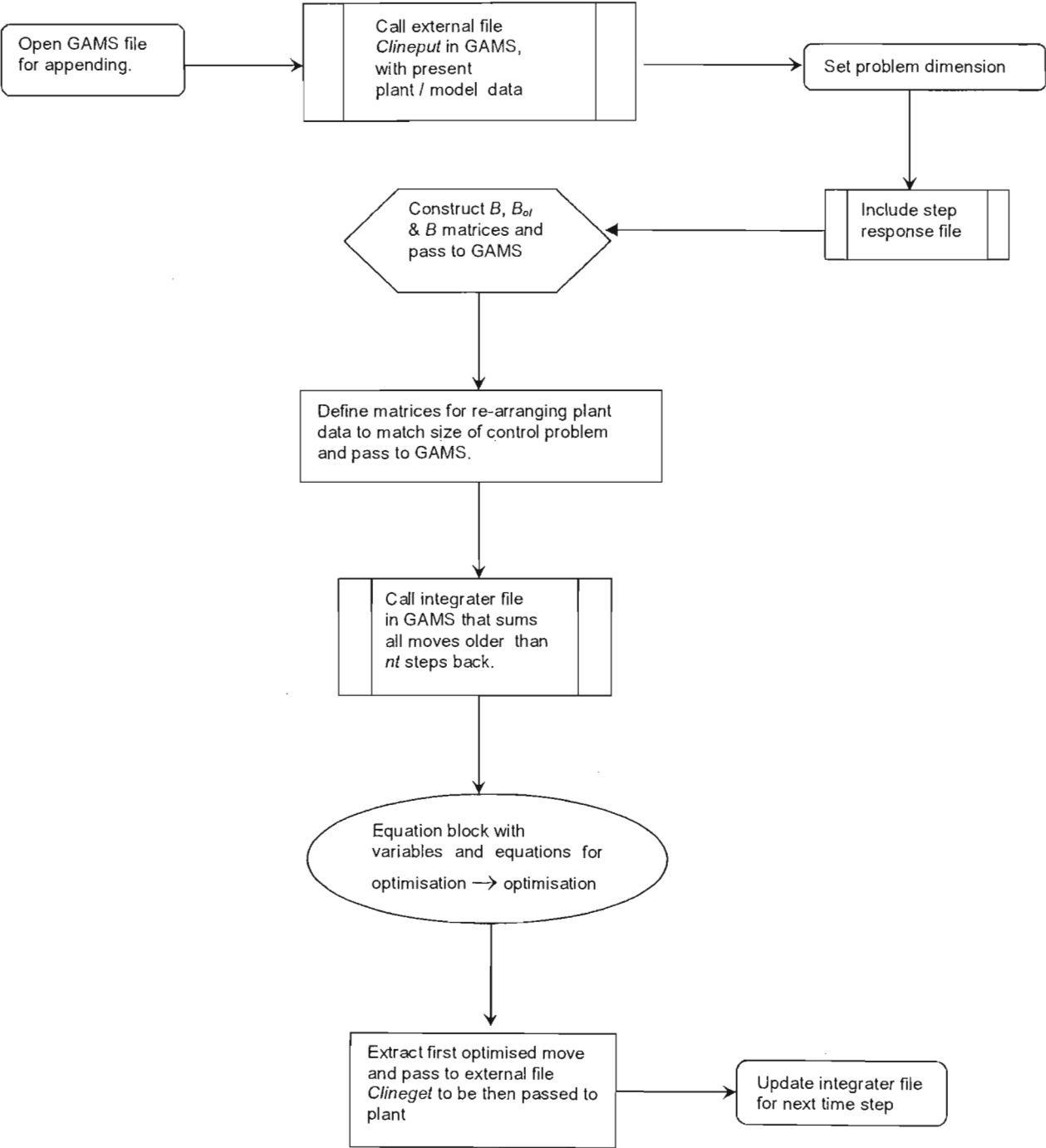
//====END of DMC Algorithm=====
}

if (m_bAlgoFuzzy)
{
    //====START of Fuzzy
    Algorithm=====
    //====END of Fuzzy Algorithm=====
}

// set outputs of EXTOject
for (i = 1; i <= iNumOutputs; i++)
{
    double m_fOutput = NewOutputs[i];
    m_fOutput = max (d(m_fOutputMins[i-1]), m_fOutput); // Clip in case went out
    m_fOutput = min (d(m_fOutputMaxs[i-1]), m_fOutput);
    ((CIStream*)m_pOutputStreams[i-1])>Write (m_csName, m_fOutput);
}
}
}

```

A.2 Logic diagram showing spawning of control algorithm in GAMS from MATLAB



Appendix B

Interacting Tanks

B.1 Step response Data

Figure B.1 Level response for step in V_1 (-30%)

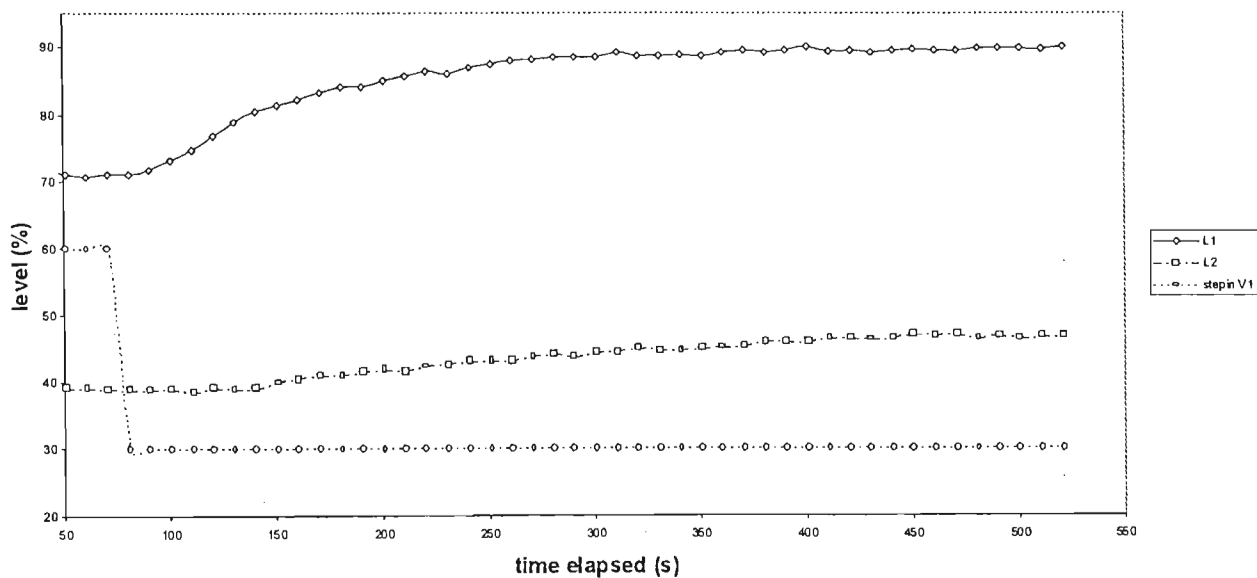


Figure B.2 Level response for step in V_2 (-30%)

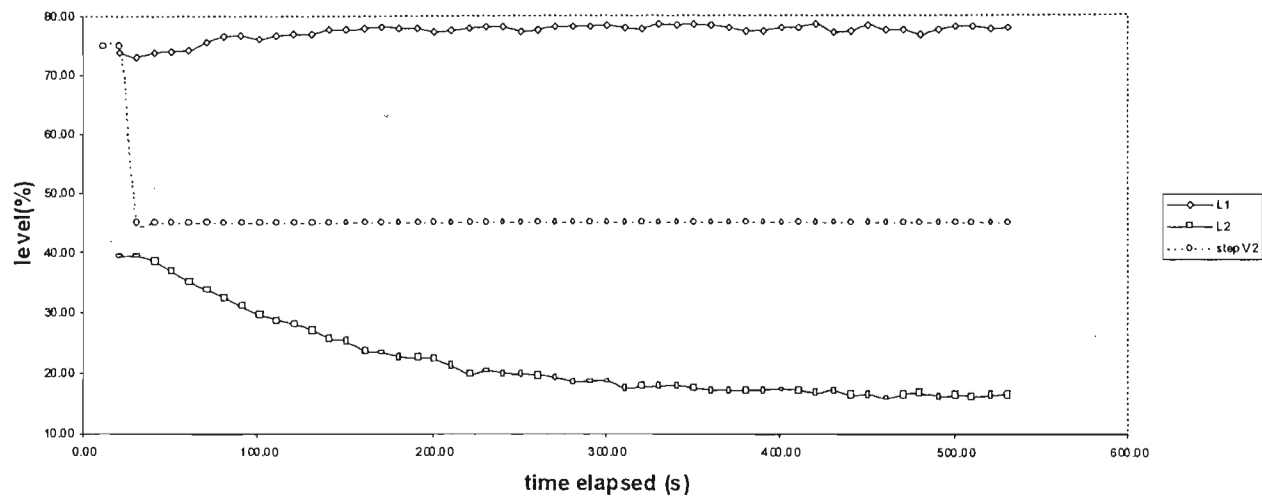
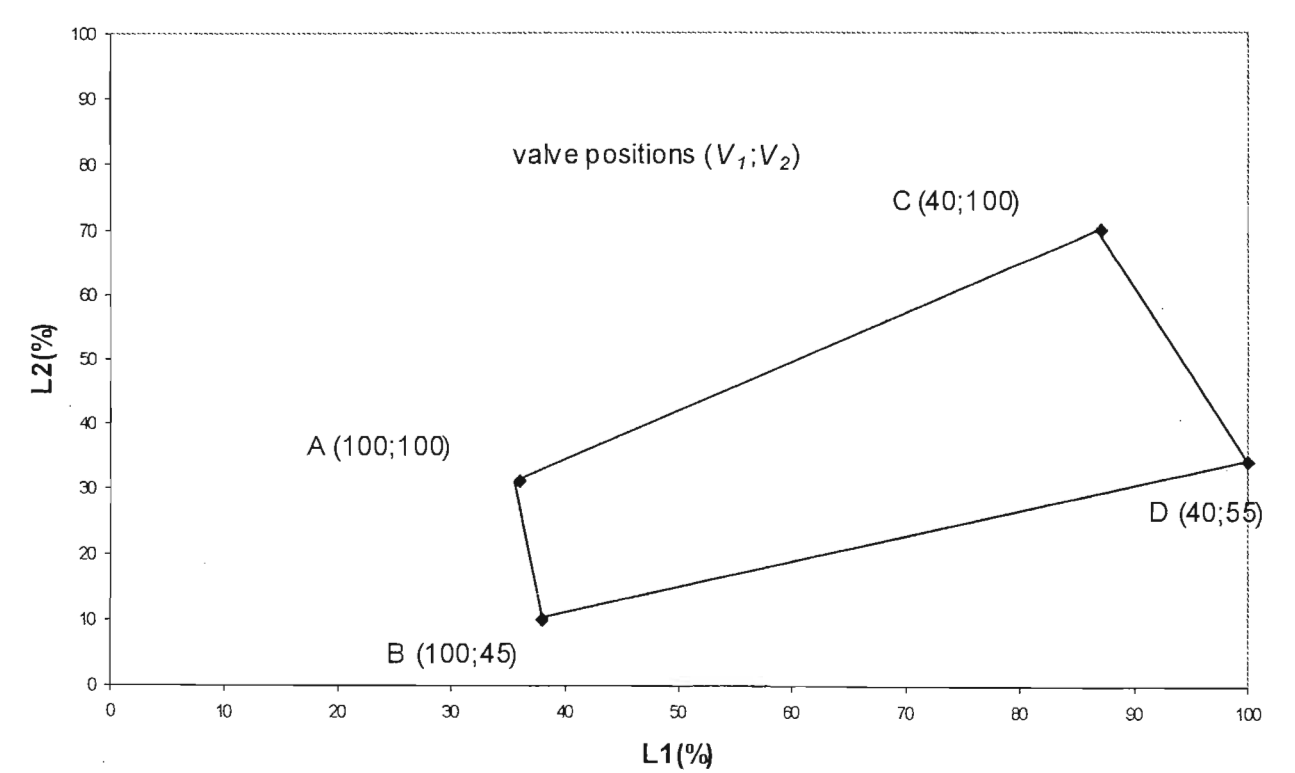


Table B.1 Step data used for dynamic matrix extending over 10 step horizon

	V_1	V_2
L_1	-0.06614	-0.0018
	-0.34063	-0.0916
	-0.46202	-0.1096
	-0.54345	-0.1335
	-0.57648	-0.144
	-0.58104	-0.1489
	-0.62612	-0.155
	-0.62017	-0.1581
	-0.6311	-0.16
	-0.6311	-0.16
L_2	-0.006	0.0356
	-0.0395	0.2291
	-0.1016	0.4089
	-0.1445	0.5604
	-0.1844	0.6334
	-0.2067	0.6996
	-0.2368	0.7265
	-0.2596	0.744
	-0.2616	0.773
	-0.2616	0.773

B.2 Operability region

Figure B.3 Operability region in terms of levels



B.3 Equation block from GAMS for optimal continuous evaluation of V_1 and discrete selection of V_2

```

Parameter xol openloop trajectory(ixt);
xol(ixt) = xo_meas(ixt) - sum(iut,(Bo(ixt,iut)*dut(iut))) + sum(iut,(Bol(ixt,iut)*dut(iut)));

binary variable binvectdup binary vector for discrete inputs(ido);

Variables
contvect continuous vector for continuous inputs(iuo)
contvectdup duplicates continuous vector for continuous inputs(ino)
binvect binary vector for discrete inputs(iuo)
absolvect sum of binary and continuous vectors – absolute input to the plant/ model(iuo)
dm change in inputs over optimization horizon(iuo)
ecl closed loop error(ixt)
absol absolute inputs(iuo)
J value of objective function
;

* .....

* Optimisation Equations
Equations
errorcl closed loop error(ixt)
discinput matrix of discrete positions acting on discrete vector(iuo)
selection ensures that only one selection per discrete input is made (ico)
continput matrix for arranging continuous inputs for summation with discrete inputs(iuo)
superimpose combines continuous and binary vector(iuo)
inputchange change between present and past input(iuo)
lower lower bound on input(iuo)
upper upper bound on input (iuo)
rampupper upper change in input per timestep(iuo)
ramplower lower change in input per timestep(iuo)
objective cost function
;

* .....

errorcl(ixt).. ecl(ixt) =e= xol(ixt) - setpoint(ixt) + sum(iuo,(B(ixt,iuo)*dm(iuo)));

discinput(iuo).. binvect(iuo) =e= sum(ido,(binmat(iuo,ido)*binvectdup(ido)));

selection(ico).. sum(ido,selmat(ico,ido)*binvectdup(ido)) =e= 1;

continput(iuo).. contvect(iuo) =e= sum(ino,(contmat(iuo,ino)*contvectdup(ino)));

superimpose(iuo).. absolvect(iuo) =e= binvect(iuo)+contvect(iuo);

inputchange(iuo).. dm(iuo) =e= absolvect(iuo)-absolvect(iuo-2)-outpres(iuo)$FirstMove(iuo);

lower(iuo).. absolvect(iuo) =g= outmin(iuo);

upper(iuo).. absolvect(iuo) =l= outmax(iuo);

```



```
rampupper(iuo)..dm(iuo) != dm_max(iuo);
ramplower(iuo)..dm(iuo) =g= -dm_max(iuo);

objective.. J =e= sum(iuo,(sqr(dm(iuo))*lamda(iuo))) + sum(ixt,(sqr(ecl(ixt))*weight(ixt)));

model dmc_scad /all/

option mip = osl

solve dmc_scad using minlp minimizing J;
```

where:

- ixt – set defining the optimisation horizon.
- iut – set defining size of optimisation horizon.
- iuo – set defining size of control horizon
- ido – set defining number of integer positions for V_2 .
- ico – set having size as defined by number of optimised moves.
- ino – duplicated set from iuo .
- $setpoint$ – vector with setpoints for input states.
- $binmat$ –matrix that contains all discrete positions for input V_2 over the optimisation horizon.
- $selmat$ –matrix with ones, enforcing only one position selection per discrete input.
- $contmat$ - matrix with ones, arranges continuous input vector for summation with integer input vector.
- $outpres$ – vector of duplicated present input states.
- $firstMove$ –binary vector for determining first time step in optimisation horizon.
- $outmin$ - vector for lower bound on outputs.
- $outmax$ – vector for upperbound on outputs.
- dm_max – vector for limit on the absolute change in outputs.
- $lamda$ – vector with weights on outputs.
- $weight$ – vector for weights on inputs.
- xo_meas – vector of present plant states.
- dut – vector of past input states.
- B, Bo, Bol – sub-matrices making up the dynamic matrix.

Appendix C

Thermal Circuit

C.1 Non - symmetrical switch step response data

Figure C.1 Temperature response for mode switch from heating to recycle

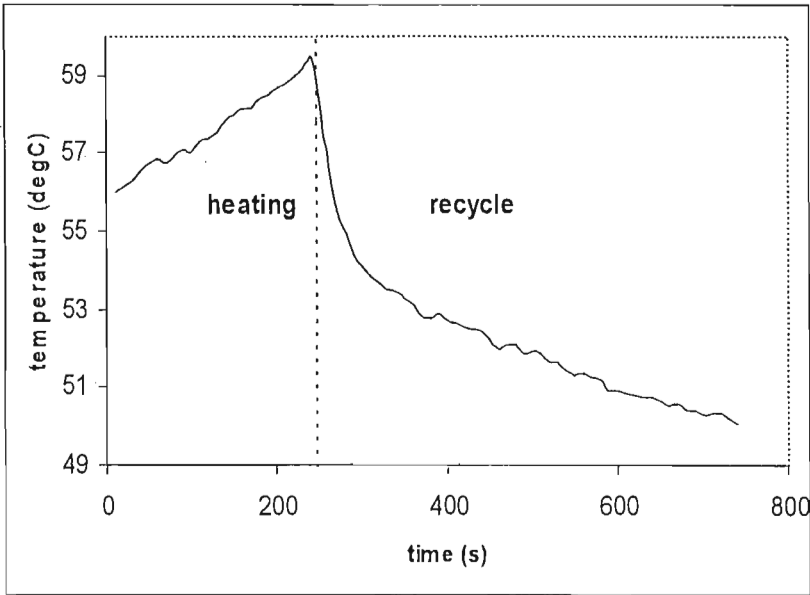


Figure C.2 Temperature response for mode switch from recycle to heating

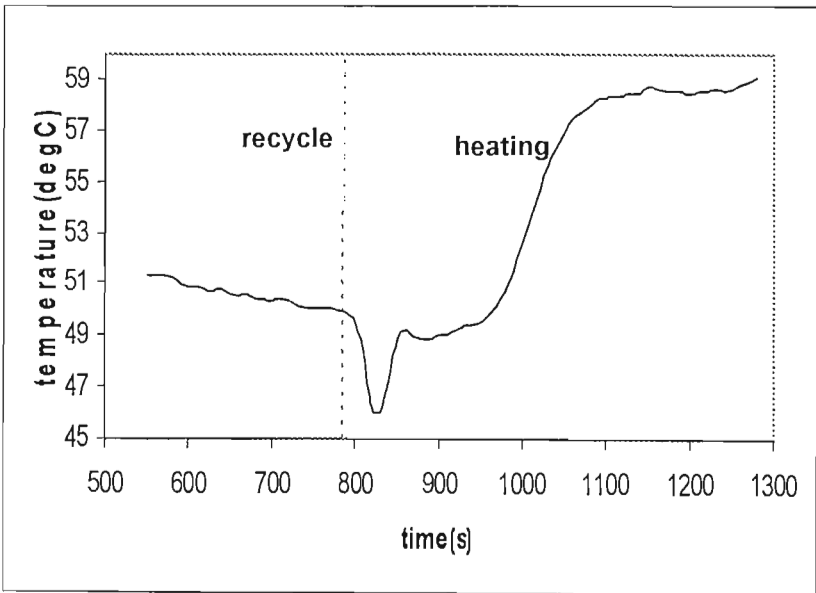


Figure C.3 Temperature response for mode switch from heating to cooling

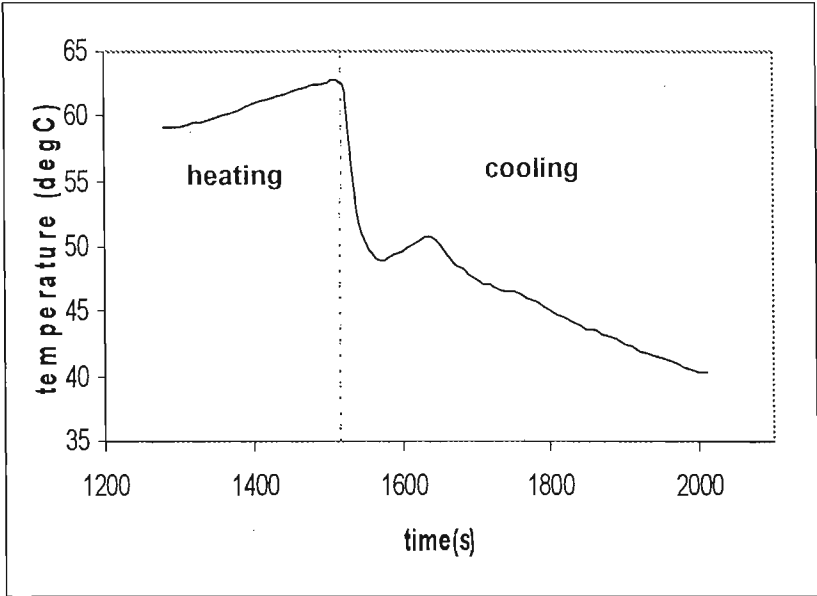


Figure C.4 Temperature response for mode switch from cooling to heating

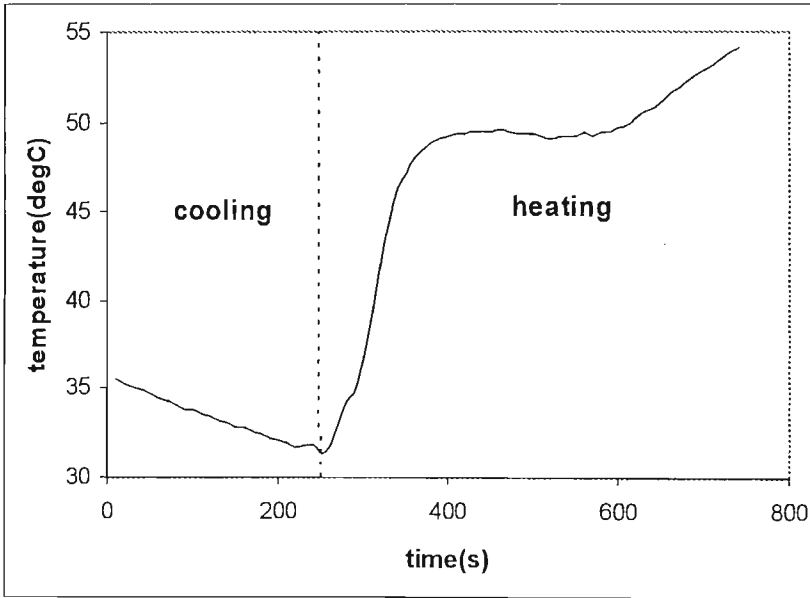


Figure C.5 Temperature response for mode switch from cooling to recycle

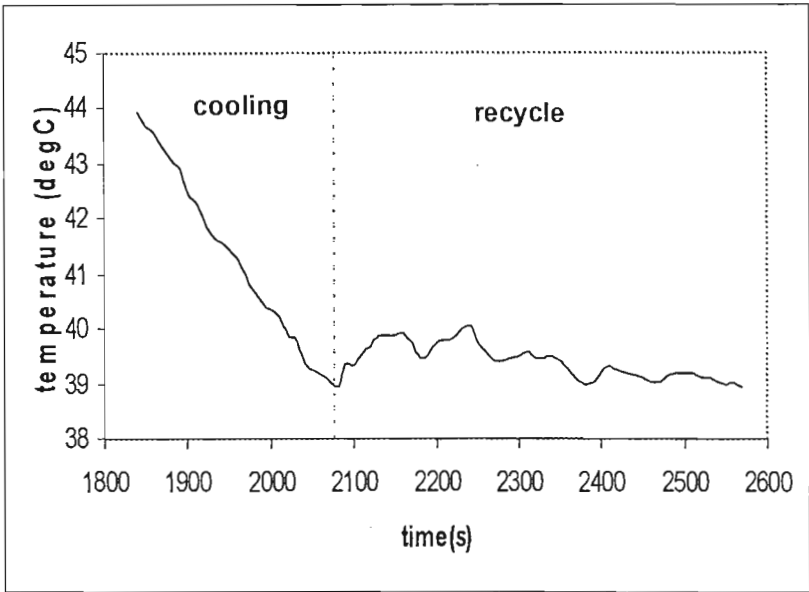


Figure C.6 Temperature response for mode switch from recycle to cooling

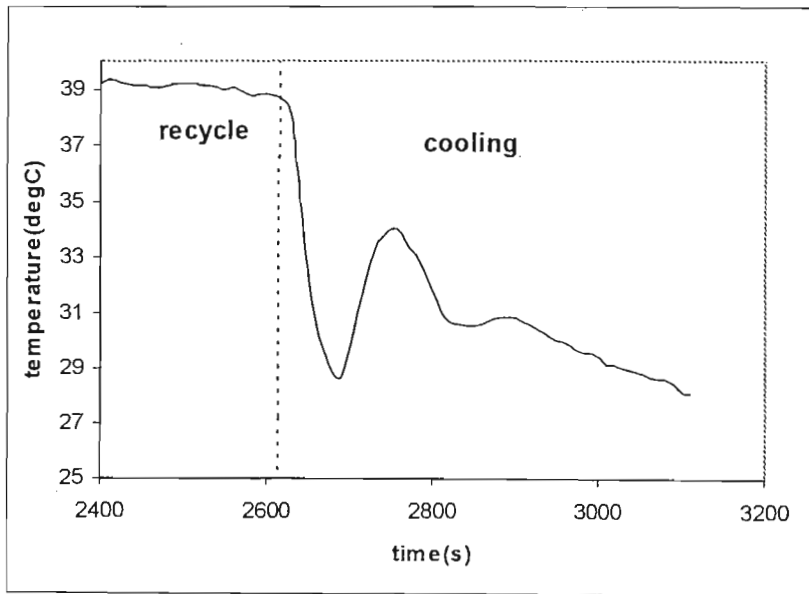


Table C.1 Non symmetric switch step data used for dynamic matrix extending over 15 step horizon

	C→R	H→R	R→C	H→C	R→H	C→H
1	0.867	-2.654	-1.688	-5.938	-0.526	0.931
2	2.020	-5.461	-8.144	-14.154	-3.145	4.710
3	2.861	-6.650	-9.600	-14.882	-0.424	11.801
4	3.216	-7.496	-6.479	-14.440	-0.336	17.484
5	4.089	-8.336	-4.729	-15.157	0.148	19.712
6	4.895	-8.972	-5.601	-17.520	0.645	20.691
7	5.079	-9.632	-7.343	-19.152	2.073	21.352
8	5.781	-10.371	-7.922	-20.219	5.088	21.826
9	6.392	-10.970	-7.687	-21.318	7.945	22.162
10	6.828	-11.702	-7.608	-22.798	9.581	22.538
11	7.325	-12.390	-7.986	-24.203	10.315	23.199
12	8.163	-13.106	-8.407	-25.424	10.606	23.921
13	8.681	-13.743	-8.714	-26.648	11.001	25.063
14	9.364	-14.323	-9.098	-27.976	11.081	26.656
15	10.059	-14.904	-9.339	-29.070	11.301	28.202

C.2 Symmetrical switch step response data

Table C.2 Symmetric switch step data used for dynamic matrix extending over 15 step horizon

	C→R	H→R	R→C	H→C	R→H	C→H
1	1.688	0.526	-1.688	-1.162	-0.526	1.162
2	8.143	3.145	-8.144	-4.999	-3.145	4.999
3	9.599	0.424	-9.600	-9.176	-0.424	9.176
4	6.478	0.336	-6.479	-6.143	-0.336	6.143
5	4.729	-0.148	-4.729	-4.877	0.148	4.877
6	5.601	-0.645	-5.601	-6.247	0.645	6.247
7	7.343	-2.073	-7.343	-9.416	2.073	9.416
8	7.921	-5.088	-7.922	-13.010	5.088	13.010
9	7.687	-7.945	-7.687	-15.632	7.945	15.632
10	7.607	-9.581	-7.608	-17.189	9.581	17.189
11	7.986	-10.315	-7.986	-18.301	10.315	18.301
12	8.407	-10.606	-8.407	-19.014	10.606	19.014
13	8.714	-11.001	-8.715	-19.715	11.001	19.715
14	9.097	-11.081	-9.098	-20.178	11.081	20.178
15	9.339	-11.301	-9.339	-20.640	11.301	20.640

C.3 Equation block from GAMS for optimal switching of modes

```

parameter sumtoBol adds on effect of integration onto openloop response(ixt);
sumtoBol(ixt) = sum(iim,(ramp_matrix(ixt,iim)*100*accum_moves(iim)));

Parameter xol open loop response(ixt);
xol(ixt)=xo_meas(ixt)-
sum(iut,(B0(ixt,iut)*100*dut(iut)))+sum(iut,(BOL(ixt,iut)*100*dut(iut)))+sumtoBol(ixt);

binary variable nmodeswitch vector for mode switches over control horizon(imo);

Variables
dm absolute plant inputs(iuo);
ecl closed loop error(ixt);
mode_var modes over control horizon(mno);
xcl closed loop response over optimisation horizon (ixt);
sofhi constraint(ixt);
soflo constraint (ixt);
J value of objective function
;

* Optimisation Equations
* .....
Equations
ecll evaluation of closed loop error(ixt)
update_mode updating future modes based on switches made in the future(mno)
closeloop(ixt)
sofhi upper bound soft constraint equation(ixt)
softlo lower bound soft constraint equation(ixt)
hardhi upper bound hard constraint equation(ixt)
hardlo lower bound hard constraint equation (ixt)
rule equality constraint on the modes that ensures logical selection(nno)
objective penalty function
;

ecll(ixt).. ecl(ixt) =e= xol(ixt) - setpoint(ixt) + sum(imo,(B(ixt,imo)*nmodeswitch(imo)*100));

update_mode(mno)..mode_var(mno)=e=mode_var(mno-modes)
+past_mode_stored(mno)$Firstmode(mno)+
sum(imo,(mode_to_switchmode(mno,imo)*nmodeswitch(imo)));

closeloop(ixt)..xcl(ixt) =e= xol(ixt)+ sum(imo,(B(ixt,imo)*nmodeswitch(imo)*100));

sofhi(ixt).. sofhi(ixt) =e= max((xcl(ixt)-softconshi(ixt)),0) ;

softlo(ixt).. soflo(ixt) =e= max((-xcl(ixt)+softconslo(ixt)),0) ;

hardhi(ixt).. xcl(ixt) =l= xmaxt(ixt) ;

hardlo(ixt).. xcl(ixt) =g= xmin(ixt) ;

```

```

rule(nno)..sum(mno,(rulemode(nno,mno)*mode_var(mno))) =e=1;

objective..J=e=sum(imo,((nmodeswitch(imo))*lamda(imo)))+
sum(ixt,(sqr(ecl(ixt))*weight(ixt)))+sum(ixt,(sqr(sofhi(ixt))*weighthi(ixt)))+sum(ixt,(sqr(soflo(ixt)
)*weightlo(ixt)));

model dmc_scad /all/

option mip = osl2

solve dmc_scad using minlp rminimizing J;

```

where:

- ixt – set defining the optimisation horizon.
- mno – set defining modes over control horizon.
- iuo – set defining primary inputs over optimisation horizon.
- iim – set defining number of possible mode switches in a single time step.
- mno – set defining modes over control horizon.
- imo – set defining mode switches over control horizon.
- nno – size of control horizon.
- accum_moves – vector accumulating changes per switch later than nt steps in time.
- ramp_matrix – matrix where rows are duplications of row vector ΔB and each row is multiplied by the row number.
- xo_meas – vector of present plant states.
- B, Bo, Bol – sub-matrices making up the dynamic matrix.
- dut – vector of past input states.
- setpoint – vector with setpoints for input states.
- modes – scalar defining number of modes.
- past_mode_stored – mode from past time step.
- softconshi – vector of upper limit soft constraint.
- softconslo – vector of lower limit soft constraint.
- firstMove – binary vector for determining first time step in optimization horizon.
- mode_to_switchmode – matrix relating modes and switches between modes.
- xmaxt - upperbound on state.
- xmint – lower bound on state.
- rulemode – matrix of 1's ensuring that only 1 mode is selected per time step.
- lamda – vector with weights on outputs.
- weight – vector for weights on inputs.
- weighthi – weighting factor on upper bound of soft constraint.
- weightlo – weighting on lower bound of soft constraint.

Appendix D

Coal Lock Sequencing

D.1 Step response data for 3 coal locks

Table D.1 Step data for coal lock sequencing extending over 5 step horizon

	CL1	CL2	CL3	rate
T1	-2	0	0	0.5
	-4	0	0	1
	-6	0	0	1.5
	-8	0	0	2
	-8	0	0	2.5
T2	0	-10	0	1
	0	-10	0	2
	0	-10	0	3
	0	-10	0	4
	0	-10	0	5
T3	0	0	-1	0.5
	0	0	-2	1
	0	0	-3	1.5
	0	0	-4	2
	0	0	-4	2.5
Fcleg	0	0	0	0
	2	3	7	0
	2	0	7	0
	0	0	0	0
	0	0	0	0

D.2 Equation block from GAMS for optimal sequencing of coal locks using soft constraint

```
parameter sumtoBol(ixt);
sumtoBol(ixt) = sum(iiu,(ramp_vector(ixt,iiu)*accum_moves(iiu)));

Parameter xol openloop response(ixt);
xol(ixt) = xo_meas(ixt) - sum(iut,(B0(ixt,iut)*dut(iut))) + sum(iut,(BOL(ixt,iut)*dut(iut))) +
sumtoBol(ixt) ;

binary variable dm(iuo);
Variables
ecl closed loop error(ixt);
```



```

absol absolute plant input(iuo);
e linearr(iuo);
bin duplication for binary variable dm (iuo);
xcl closed loop response(ixt);
sofhi upper limit of soft constraint(ixt);
soflo lower limit of soft constraint(ixt);
J value of objective function
* Optimisation Equations
* .....
Equations
eccl closed loop error(ixt)
absoll absolute controller output to plant(iuo)
softhi upper limit for soft constraint (ixt)
softlo lower limit for soft constraint(ixt)
closeloop closed loop evaluation(ixt)
linear direct assignment for linear purposes(iuo)
objective objective function;

* .....
* Equality and inequality constraints

eccl(ixt).. eccl(ixt) =e= xol(ixt) - setpoint(ixt) + sum(iuo,(B(ixt,iuo)*dm(iuo)));

absoll(iuo).. absol(iuo) =e= absol(iuo-nu) + outpres(iuo)$FirstMove(iuo) + dm(iuo);

closeloop(ixt).. xcl(ixt) =e= xol(ixt) + sum(iuo,(B(ixt,iuo)*dm(iuo)));

softhi(ixt).. sofhi(ixt) =e= max((xcl(ixt)-softconshi(ixt)),0) ;

linear(iuo).. bin(iuo) =e=dm(iuo);

softlo(ixt).. soflo(ixt) =e= max((-xcl(ixt)+softconslo(ixt)),0) ;

objective.. J =e= sum(iuo,(sqr(bin(iuo))*lamda(iuo))) +
sum(ixt,(sqr(eccl(ixt))*weight(ixt)))+sum(ixt,(sqr(sofhi(ixt))*weighthi(ixt)))+sum(ixt,(sqr(soflo(ixt)
)*weightlo(ixt)));

* .....
model dmc_scad /all/

option mip = osl2

solve dmc_scad using minlp minimizing J;

```

where:

- ixt – set defining the optimisation horizon.
- iuo – set defining number of controller outputs
- iuo – set defining primary inputs over optimisation horizon.
- $accum_moves$ – vector accumulating initiations later than nt steps in time.
- $ramp_matrix$ – matrix where rows are duplications of row vector ΔB and each row is multiplied by the row number.
- xo_meas – vector of present plant states.
- B, Bo, Bol – sub-matrices making up the dynamic matrix.
- dut – vector of past initiations extending nt steps back in time.

- firstMove –binary vector for determining first time step in optimization horizon.
- softconshi – vector of upper limit soft constraint.
- softconslo – vector of lower limit soft constraint.
- lamda – vector with weights on outputs.
- weight – vector for weights on inputs.
- weighthi – weighting factor on upper bound of soft constraint.
- weightlo – weighting on lower bound of soft constraint.

Appendix E

Physicochemical Discontinuities

E.1 Matlab code, initially spawns control algorithm into GAMS and then simulates control loop with a single tank.

```

%Matlab file simulating the closed loop operation of a tank and controller.
%Tank is fitted with an overflow at height hv and thus has 2 regimes of operation.
%Input to the tank is inflow Fo. The tank drains due to gravity
%Tank also has discontinuous cross sectional area at height hv

%This file does the following:
% a. contains information describing the system dynamics and controller.
% b. spawns the Gams file for control of the system.
% c. generates system state data (matlab) by using the controller output for the controller which %
%    operates in feedback .
% d. prompts the storage and recall of system states and inputs for the system and controller.
% e. controller has a forward horizon of nt on setpoint i.e. forward control

%-----
clear all

%open 2 files: one for the controller called at every timestep and the
% other for exchanging data between the model and the controller

%Gams file--->controller
filetank = fopen('C:\WINDOWS\Desktop\jasmeer\state_space\tank\filetank.gms','w');

fprintf(filetank,'*Gams file spawned from Matlab for control');
fprintf(filetank,'of a tank system having 2 regimes\n\n');

%-----Input data-----
%
%                                tank data governing dynamics

d1r1 = 2.5; %tank diameter below overflow [m]
A1r1 = pi*d1r1^2/4; %cross sectional area of the tank

d1r2 = 1.5; %tank diameter above overflow [m]
A1r2 = pi*d1r2^2/4;

k1 = 9e-3; % orifice constant for the bottom bottom line
kv = 4e-3; % valve constant for overflow line
hv = 5; %height of withdrawal[m]
hmax = 15; %height of tank [m]
Fmax = 150; % max flow allowed into tank [m3/hr]
g = 9.8; %gravity [m2/s]

%pass these to Gams
fprintf(filetank,'*Tank parameters.....\n');
fprintf(filetank,'Scalars \n');

```

```

fprintf(filetank,'d1r1 diameter of the tank below overflow/ %d \n',d1r1);
fprintf(filetank,'A1r1 diameter of the tank below overflow/ %d \n',A1r1);

fprintf(filetank,'d1r2 diameter of the tank above overflow/ %d \n',d1r2);
fprintf(filetank,'A1r2 diameter of the tank above overflow/ %d \n',A1r2);

fprintf(filetank,'k1 flow coefficient for bottom line/ %d \n',k1);
fprintf(filetank,'kv flow coefficient for overflow line/ %d \n',kv);

fprintf(filetank,'Fmax max flow into tank/ %d \n',Fmax);
fprintf(filetank,'hv height of overflow/ %d \n',hv);

fprintf(filetank,'hmax max height of tank/ %d \n',hmax);

fprintf(filetank,';\n\n');

fprintf(filetank,'* ..... \n');

%                               controller parameters and initial conditions

nt = 5; %size of controller horizon
no = 3; % no. of optimized moves
dt = 60; %time step [s]
T = 20; % number of control time steps

hsp = zeros(T,1); % level setpoint [m]
hsp(1:10) = 6;
hsp(11:20) = 4;

Fin_nought = 0; %initial inflow to the tank [m3/hr]
ho = 4; %initial height in the tank at start of simulation [m]
h=ho;
Fin=Fin_nought;
alpha=1; % regime at startup
beta=1-alpha;

%-----
%Pass this control data to filetank.....spawning of filetank for system control

%Pass set to Gams - size of optimization interval
if nt<=10
    fprintf(filetank,'Set t optimisation horizon / t1*t%d /;\n',nt);
else
    fprintf(filetank,'Set t optimisation horizon / t01*t%d /;\n',nt);
end
fprintf(filetank,' \n\n');

%Pass set to Gams - size of control horizon
if no<10
    fprintf(filetank,'Set n optimisation horizon / n1*n%d /;\n',no);
else
    fprintf(filetank,'Set n optimisation horizon / n01*n%d /;\n',no);
end
fprintf(filetank,' \n\n');

fprintf(filetank,'Scalar dt time step interval/ %d /;\n',dt);

```

```

fprintf(filetank, '\n\n');

fprintf(filetank, 'parameter firststep(t);\n');
fprintf(filetank, 'firststep(t)=yes$(ord(t) le 1 );\n');

fprintf(filetank, '\n\n');

%In filetank, call file cycle as included file with data from past timestep and setpoints

fprintf(filetank, '$INCLUDE C:\\WINDOWS\\Desktop\\jasmeer\\state_space\\tank\\cycle.gms\n\n');

% matrix for arranging optimised moves
moveopt=zeros(nt,no);
id=eye(no);
moveopt(1:no,1:no)=id;

if (nt-no)~=0
    for i=no+1:nt
        moveopt(i,no)=1;
    end
end

% protection for (hst-hv) from being less than 0
fprintf(filetank, 'scalar diffh2;\n');
fprintf(filetank, 'diffh2 = hst-hv;\n');
fprintf(filetank, 'if( ');
fprintf(filetank, '(diffh2 <= 0),\n');
fprintf(filetank, 'diffh2 = 1;\n');
fprintf(filetank, ');\n');

fprintf(filetank, '\n\n');

%Pass this to Gams-----

fprintf(filetank, '* matrix for arranging optimised moves\n');

what='Table';name='moveopt';
rowsno=nt;columnsno=no;rowset='t';columnset='n';matrix=moveopt;
write2gams(what,name,rowsno,columnsno,rowset,columnset,matrix); %function for writing table to
GAMS

fprintf(filetank, 'binary variables\n');

fprintf(filetank, 'alpha binary variable for the selection of regime1(t)\n');
fprintf(filetank, 'beta binary variable for the selection of regime2(t)\n');

fprintf(filetank, ';\n\n');

fprintf(filetank, 'variables\n\n');
fprintf(filetank, 'h tank level(t)\n');
fprintf(filetank, 'dh1 change in tank level in regime1(t)\n');
fprintf(filetank, 'dh2 change in tank level in regime2(t)\n');
fprintf(filetank, 'Finh continuous feed into the tank over optimisation horizon (t)\n');
fprintf(filetank, 'Fin continuous feed into the tank over control horizon(t)\n');
fprintf(filetank, 'er error in level(t)\n');
fprintf(filetank, 'alphal linear variable for alpha(t)\n');
fprintf(filetank, 'betal linear variable for beta(t)\n');
fprintf(filetank, 'reg1 continuous variable(t)\n');

```

```

fprintf(filetank,'reg2 continuous variable(t)\n');

fprintf(filetank,'J value of objective function\n');

fprintf(filetank,';\n\n');

fprintf(filetank,'equations\n\n');

fprintf(filetank,'level(t)\n');
fprintf(filetank,'regcont1(t)\n');
fprintf(filetank,'regcont2(t)\n');
fprintf(filetank,'flowopt(t)\n');
fprintf(filetank,'regime1(t)\n');
fprintf(filetank,'regime2(t)\n');
fprintf(filetank,'flowhi(n)\n');
fprintf(filetank,'flowlo(n)\n');
fprintf(filetank,'exclusive(t)\n');
fprintf(filetank,'lessthan(t)\n');
fprintf(filetank,'greaterthan(t)\n');
fprintf(filetank,'error(t)\n');
fprintf(filetank,'objective\n');
fprintf(filetank,';\n\n');

fprintf(filetank,'level(t)..h(t) =e= h(t-1)+ho(t)$firststep(t)+ reg1(t)*dh1(t)+ reg2(t)*dh2(t);\n');

fprintf(filetank,'regcont1(t)..alpha(t) =e= reg1(t);\n');

fprintf(filetank,'regcont2(t)..beta(t) =e= reg2(t);\n');

fprintf(filetank,'flowopt(t)..Fin(t) =e= sum(n,(Finh(n)*moveopt(t,n))); \n');

fprintf(filetank,'regime1(t)..dh1(t) =e= dt/A1r1* (Fin(t) - (k1*sqrt(hst) + k1/(2*sqrt(hst))*(h(t-1)+ho(t)$firststep(t)-hst)));\n');

fprintf(filetank,'regime2(t)..dh2(t) =e= dt/A1r2*(Fin(t)-(k1*sqrt(hst)+k1/(2*sqrt(hst))*(h(t-1)+ho(t)$firststep(t)-hst)) - (kv*sqrt(diffh2)+kv/(2*sqrt(diffh2))*(h(t-1)+ho(t)$firststep(t)-hst))) ;\n');

fprintf(filetank,'flowhi(n)..Finh(n) =l= Fmax/3600; \n');

fprintf(filetank,'flowlo(n)..Finh(n) =g= 0/3600; \n');

fprintf(filetank,'exclusive(t)..alpha(t)+beta(t) =e= 1; \n');

fprintf(filetank,'lessthan(t)..h(t)-hv*alpha(t)-hmax*beta(t) =l= 0; \n');

fprintf(filetank,'greaterthan(t)..h(t)-0*alpha(t)-hv*beta(t) =g= 0; \n');

fprintf(filetank,'error(t)..er(t)=e= sqrt(h(t)-hsp(t)); \n');

fprintf(filetank,'objective.. J =e= 1*sum(t,er(t)); \n');

fprintf(filetank,'model tank1 /all\n');

fprintf(filetank,'option solprint = off \n');

fprintf(filetank,'option minlp = dicopt \n');

fprintf(filetank,'option mip = osl2 \n');

```

Appendix E: Physicochemical Discontinuities

```
fprintf(filetank,'solve tank1 using minlp minimizing J ;\n');

% for Gams call when the controller intervenes
fprintf(filetank,'scalars \n');
fprintf(filetank,'Finpass\n');
fprintf(filetank,'alphapass\n');
fprintf(filetank,'betapass\n');

fprintf(filetank,';\n\n');

    if nt <= 10
        fprintf(filetank,'Finpass=Fin.l ("t1"); \n');
        fprintf(filetank,'alphapass=alpha.l("t1"); \n');
        fprintf(filetank,'betapass=beta.l("t1"); \n');
    else
        fprintf(filetank,'Finpass=Fin.l("t01"); \n');
        fprintf(filetank,'alphapass=alpha.l("t01"); \n');
        fprintf(filetank,'betapass=beta.l("t01"); \n');
    end

fprintf(filetank,'$libinclude matout Finpass\n');
fprintf(filetank,'$libinclude matout alphapass\n');
fprintf(filetank,'$libinclude matout betapass\n');

fprintf(filetank,'display alpha.l,beta.l,Fin.l,h.l;\n');

fclose(filetank);

%***** SIMULATION LOOP *****

for i=1:T % main loop determining number of time steps

    %Storage file-----CYCLE-----

    cycle = fopen('C:\WINDOWS\Desktop\jasmeer\state_space\tank\cycle.gms','w');

    % read setpoint trajectory extending nt steps ahead
    counter=1;
    for ii=i:i+nt-1
        if i<=T-nt
            setpoint(counter)=hsp(ii);
        else
            setpoint(counter)=hsp(i);
        end
        counter=counter+1;
    end
    fprintf(cycle,'parameter hsp(t)/ \n');

    %pass to GAMS
    for j=1:nt

        if nt==10 & j~=10
            fprintf(cycle,'\nt%d',j);

        elseif ((nt<10)|(j>9))
            fprintf(cycle,'\nt%d',j);
```

```

else
    fprintf(cycle,'\nt0%d',j);
end

fprintf(cycle,' %12.4e ',setpoint(j));
end

fprintf(cycle,' /;\n\n');

fprintf(cycle,' \n\n');

% now store the past inputs and states for use by the controller
fprintf(cycle,'parameters\n');
fprintf(cycle,'Fin_nought present inflow(t)\n');
fprintf(cycle,'ho present level(t)\n');
fprintf(cycle,',;\n\n');

fprintf(cycle,'Fin_nought(t)=%d;\n',Fin);

%protection against division by 0
h=max(h,0.1);

fprintf(cycle,'ho(t)=%d;\n',h);

fprintf(cycle,'scalar hst; \n');
fprintf(cycle,'hst=ho("t1"); \n');

fclose(cycle); %-----

if i>1 %....start loop only on second cycle
    %-----Controller-----

    gams_output='std';
    [Fin,alpha,beta]=gams('filetank');

    %-----Model-----
    diffh2=h-hv;
    diffh2=max(diffh2,0);
    h = h + alpha*dt/A1r1*(Fin-k1*h^0.5) + beta*dt/A1r2*(Fin-k1*(h)^0.5-kv*(diffh2)^0.5);

end

heightplot(i)=h;
flowplot(i)=3600*Fin;
alphaplot(i)=alpha;
betaplot(i)=beta;

%-----
step=i;
fprintf('timestep: %6.3f \n',i)

end

```



```

time=(1:T);
discont=hv*ones(T,1);

time=(1:T);
discont=hv*ones(T,1);

subplot(2,1,1), plot(time,heightplot,time,hsp,'.',time,discont,'-');
title('tank level');
ylabel('height (m)');
AXIS([1 T 3 7]);

d=legend('height','setpoint','discontinuity',1);

subplot(2,1,2), plot(time,flowplot);
title('inlet flow');
ylabel('inlet flow (m3/hr)');
xlabel('time step ');

AXIS([1 T 0 1.1*Fmax])

```

E.2 Control Algorithm executed in GAMS every time the controller is called for 2 tank filling.

*Gams file spawned from Matlab for control of 2 tank system having 2 regimes

```

*Tank parameters.....
Scalars
d1r1 diameter of the tank below overflow/ 2.500000e+000 /
A1r1 area of the tank below overflow/ 4.908739e+000 /
d1r2 diameter of the tank above overflow/ 1.500000e+000 /
A1r2 area of the tank above overflow/ 1.767146e+000 /
d2 diameter of the tank 2 / 2.500000e+000 /
A2 area of the tank 2 / 4.908739e+000 /
k1 flow coefficient for bottom line tank1/ 9.000000e-003 /
k2 flow coefficient for bottom line tank2/ 8.000000e-004 /
kv flow coefficient for overflow line/ 6.000000e-003 /
Fmax max flow into tank/ 150 /
hv height of overflow/ 5 /
hmax max height of tank/ 15 /
hmin max height of tank/ 0 /
;

* .....
Set t optimization horizon / t1*t5 /;

Set n optimization horizon / n1*n2 /;

```

Appendix E: Physicochemical Discontinuities

Scalar dt time step interval/ 60 /;

parameter firststep(t);
firststep(t)=yes\$(ord(t) le 1);

\$INCLUDE C:\WINDOWS\Desktop\jasmeer\state_space\tank\cycle.gms

scalar diffh2;
diffh2 = hst1-hv;
if((diffh2 <= 0),
diffh2 = 1;
);

* matrix for arranging optimised moves ahead in time

Table moveopt(t,n)
 n1 n2
t1 1.0000e+000 0.0000e+000
t2 0.0000e+000 1.0000e+000
t3 0.0000e+000 1.0000e+000
t4 0.0000e+000 1.0000e+000
t5 0.0000e+000 1.0000e+000 ;

binary variables

alpha binary variable for the selection of regime1(t)

beta binary variable for the selection of regime2(t)

Vh binary valve operation(n)

;

variables

h1 tank1 level(t)
h2 tank2 level(t)
dh1 change in tank1 level in regime1(t)
dh2 change in tank1 level in regime2(t)
dht2 change in level in tank2(t)
Finh continuous feed into the tank (t)
Fin continuous feed into the tank over control horizon(t)
V valve controlling out flow from tank1(t)
er1 error in level1(t)
er2 error in level2(t)
alpha linear variable for s1(t)
beta linear variable for s2(t)
reg1 continuous variable(t)
reg2 continuous variable(t)
J value of objective function
;

equations

level1 integration for level in tank1(t)
level2 integration for level in tank2(t)
level2min(t)
regcont1(t)
regcont2(t)
flowopt(t)
Vopt(t)
regime1 change in tank1 level in regime1(t)

Appendix E: Physicochemical Discontinuities

```

regime2 change in tank2 level in regime2(t)
tank2change change in tank2 level(t)
flowhi(n)
flowlo(n)
exclusive(t)
lessthan(t)
greaterthan(t)
Vmax(t)
Vmin(t)
error1 error between level and setpoint in tank1(t)
error2 error between level and setpoint in tank2(t)
objective
;

level1(t)..h1(t) =e= h1(t-1)+ho1(t)$firststep(t)+ reg1(t)*dh1(t)+ reg2(t)*dh2(t);

level2(t)..h2(t) =e= h2(t-1)+ho2(t)$firststep(t)+ dh2(t);

tank2change(t)..dht2(t) =e= dt/A2*(reg1(t)*(V(t)*(k1*sqrt(hst1) + k1/(2*sqrt(hst1)))*(h1(t-1)+ho1(t)$firststep(t)-hst1))) + reg2(t)*(V(t)*(k1*sqrt(hst1)+k1/(2*sqrt(hst1)))*(h1(t-1)+ho1(t)$firststep(t)-hst1)) + (kv*sqrt(diffh2)+kv/(2*sqrt(diffh2)))*(h1(t-1)+ho1(t)$firststep(t)-hst1))) - (k2*sqrt(hst2) + k2/(2*sqrt(hst2)))*(h2(t-1)+ho2(t)$firststep(t)-hst2)));

regcont1(t)..alpha(t) =e= reg1(t);

regcont2(t)..beta(t) =e= reg2(t);

flowopt(t)..Fin(t) =e= sum(n,(Finh(n)*moveopt(t,n)));

Vopt(t)..V(t) =e= sum(n,(Vh(n)*moveopt(t,n)));

regime1(t)..dh1(t) =e=dt/A1r1* (Fin(t) - V(t)*(k1*sqrt(hst1) + k1/(2*sqrt(hst1)))*(h1(t-1)+ho1(t)$firststep(t)-hst1)));

regime2(t)..dh2(t) =e= dt/A1r2*(Fin(t) - V(t)*(k1*sqrt(hst1)+k1/(2*sqrt(hst1)))*(h1(t-1)+ho1(t)$firststep(t)-hst1)) - (kv*sqrt(diffh2)+kv/(2*sqrt(diffh2)))*(h1(t-1)+ho1(t)$firststep(t)-hst1))) ;

flowhi(n)..Finh(n) =l= Fmax/3600;

flowlo(n)..Finh(n) =g= 0/3600;

exclusive(t)..alpha(t)+beta(t) =e= 1;

lessthan(t)..h1(t)-hv*alpha(t)-hmax*beta(t) =l= 0;

greaterthan(t)..h1(t)-hmin*alpha(t)-hv*beta(t) =g= 0;

level2min(t)..h2(t) =g= 0;

Vmin(t)..V(t) =g= 0;

Vmax(t)..V(t) =l= 1;

error1(t)..er1(t)=e= sqr(h1(t)-hsp1(t));

error2(t)..er2(t)=e= sqr(h2(t)-hsp2(t));

objective.. J =e= 3*sum(t,er1(t)) + 20*sum(t,er2(t));
* note weighting parameters '3' and '20' above

```

```
model tank1 /all/

option solprint = off
option minlp = dicopt
option mip = osl2
solve tank1 using minlp minimizing J ;
scalars
Finpass
Vpass
s1pass
s2pass
h1pass
;

Finpass=Fin.l('t1');
Vpass=V.l('t1');
s1pass=s1.l('t1');
s2pass=s2.l('t1');
h1pass=h1.l('t1');
$libinclude matout Finpass
$libinclude matout Vpass
$libinclude matout s1pass
$libinclude matout s2pass
$libinclude matout h1pass
display s1.l,s2.l,Fin.l,h1.l,h2.l,V.l;
```