

University of KwaZulu-Natal

**DEVELOPMENT AND IMPLEMENTATION OF A
REAL-TIME OBSERVER MODEL FOR MINERAL
PROCESSING CIRCUITS**

John-Roy Ivy Vosloo Bsc. (Chem. Eng.)

**DEVELOPMENT AND IMPLEMENTATION OF A
REAL-TIME OBSERVER MODEL FOR MINERAL
PROCESSING CIRCUITS**

John-Roy Ivy Vosloo BSc. (Chem. Eng.)

Submitted in partial fulfilment of the academic requirements for the degree of Master of Science in Engineering in the School of Chemical Engineering at the University of KwaZulu-Natal, Durban.

Date: January 2004

Supervisor: M. Mulholland

PREFACE

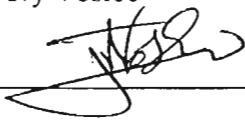
In addition to the research work presented in this dissertation, one postgraduate course was completed in the School of Chemical Engineering at the University of Natal:

- Real Time Process Data Analysis [DNC5RT1]

This work has also led to one conference poster presentation.

I certify that all of the work in this dissertation is my own work, except where otherwise indicated and referenced, and that it has not been submitted for a degree to any other university or institution.

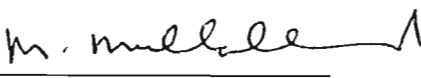
John-Roy Ivy Vosloo

Signed:  _____

Date: 28/01/2004

As the candidate's supervisor I have approved this dissertation for submission.

Professor Michael Mulholland

Signed:  _____

Date: 2004.01.28

ACKNOWLEDGEMENTS

I would like to express my gratitude towards a number of people who have assisted me during my research:

- Professor Michael Mulholland for his input, guidance and assistance throughout the project.
- The staff of the Measurement and Control Division at MINTEK for their patience and assistance.
- The Department of Arts, Culture, Science and Technology's Innovations project for funding.
- My friends and colleagues in the Department of Chemical Engineering at the University of Natal.

Finally I would like to thank my family and friends for their support and encouragement. Special thanks go to my parents François and Wendy, to my sister Chiquita, my aunt Noleen, Claire Hean, Nicholas du Preez, Sebastien Lacour, Johan Steyn and Allen Hemphill.

ABSTRACT

Mineral processing plants, such as LONMIN's Eastern Platinum B-stream, typically have few on-line measurements, and key measures of performance such as grade only become available after samples have been analysed in the laboratory. More immediate feedback from a dynamic observer model promises enhanced understanding of the process, and facilitates prompt corrective actions, whether in open or closed loop. Such plants easily enter sub-optimal modes such as large, uselessly re-circulating loads as the feed conditions change. Interpretation of such modes from key combinations of the variables deduced by an observer model, using a type of expert system, would add another level of intelligence to benefit operation.

The aim of this thesis was to develop and implement a dynamic observer model of the LONMIN Eastern Platinum B-Stream into one of the existing control platforms available at the plant, known as PlantStar®, developed by MINTEK.

The solution of the system of differential and algebraic equations resulting from this type of flowsheet modelling is based on an extended Kalman filter, which is able to dynamically reconcile any measurements which are presented to it, in real time. These measurement selections may also vary in real time, which provides flexibility of the model solution and the model's uses.

PlantStar passes the measurements that are available at the plant, to the dynamic observer model through a "plugin" module, which has been developed to incorporate the observer model and utilise the PlantStar control platform. In an on-line situation, the model will track the plant's behaviour and continuously update its position in real-time to ensure it follows the plant closely. This model would then be able to run simulations of the plant in parallel and could be used as a training facility for new operators, while in a real-time situation it could provide estimates of unmeasurable variables throughout the plant. An example of some of these variables are the flotation rate constants of minerals throughout the plant, which can be estimated in real time by the extended Kalman filter. The model could also be used to predict future plant conditions based on the current plant state, allowing for case scenarios to be performed without affecting the actual plant's performance.

Once the dynamic observer model and "plugin" module were completed, case scenario simulations were performed using a measured data set from the plant as a starting point because real-time data were unavailable as the model was developed off-site.

TABLE OF CONTENTS

PREFACE.....	ii
ACKNOWLEDGEMENTS.....	iii
ABSTRACT.....	iv
TABLE OF CONTENTS.....	v
LIST OF FIGURES.....	x
LIST OF TABLES.....	xv
NOMENCLATURE.....	xvi
1 INTRODUCTION.....	1-1
1.1 ORGANISATION OF THIS THESIS.....	1-1
1.2 BACKGROUND TO MINERAL PROCESSING.....	1-3
1.3 MINERAL PROCESSING OVERVIEW.....	1-4
1.4 METHODS OF SEPARATION.....	1-6
1.5 FROTH FLOTATION.....	1-9
1.6 PROBLEMS FACED BY INDUSTRIAL FLOTATION CIRCUITS.....	1-11
1.7 OBJECTIVES OF THIS THESIS.....	1-12
2 FLOTATION MODELLING REVIEW.....	2-1
2.1 GENERAL TYPES OF FLOTATION MODELS.....	2-1
2.1.1 <i>Fundamental Models</i>	2-1
2.1.2 <i>Empirical Models</i>	2-3
2.1.3 <i>Semi-empirical Models</i>	2-3
2.1.3.1 <i>Probability Models</i>	2-3
2.1.3.2 <i>Kinetic Rate Models</i>	2-4
2.1.3.3 <i>Collision Models</i>	2-5
2.2 KINETIC RATE MODELLING OF FLOTATION SUB-PROCESSES.....	2-7
2.2.1 <i>Flotation Rate Order</i>	2-7
2.2.2 <i>Sub-systems considered in Froth</i>	2-8
2.2.3 <i>Residence Time in Froth</i>	2-10
2.2.4 <i>Gangue Entrainment</i>	2-12
2.2.5 <i>Bubble-particle Collision, Attachment and Detachment</i>	2-14
2.2.6 <i>Chemical Factors</i>	2-18

2.2.6.1	Collectors	2-18
2.2.6.2	Frothers and Regulators	2-19
2.2.6.3	Importance of pH	2-19
2.2.7	<i>Froth Depth</i>	2-20
2.2.8	<i>Particle Size</i>	2-23
2.2.9	<i>Air Addition Rate</i>	2-24
3	FLOTATION CIRCUIT DESIGN, CONTROL AND OPTIMISATION.....	3-1
3.1	FLOTATION CIRCUIT DESIGN	3-1
3.2	OPTIMISATION OF CIRCUIT DESIGNS	3-6
3.3	CONTROL OF FLOTATION CIRCUITS	3-7
3.3.1	<i>Basic Control Strategies</i>	3-8
3.3.2	<i>Advanced Control Strategies</i>	3-9
3.3.2.1	Feed-Forward Control	3-9
3.3.2.2	Adaptive Control	3-10
3.3.2.3	Multivariable Control	3-10
3.3.2.4	State Estimators	3-10
3.3.2.5	Expert Systems	3-11
4	DYNAMIC OBSERVER MODEL (DOM) DEVELOPMENT	4-1
4.1	OBJECTIVES OF THE DYNAMIC OBSERVER MODEL	4-1
4.1.1	<i>Explain Plant Behaviour</i>	4-1
4.1.2	<i>Used for Simulation and Playback</i>	4-1
4.1.3	<i>Provide a better insight into the process</i>	4-2
4.2	REQUIREMENTS TO MEET OBJECTIVES	4-3
4.3	MODEL STRUCTURE	4-4
4.4	DIFFERENTIAL AND ALGEBRAIC EQUATIONS (DAE) SOLUTION METHOD	4-13
4.4.1	<i>The Specific Dynamic Observer Model Solution</i>	4-13
4.5	THE DYNAMIC OBSERVER MODEL FEATURES	4-19
4.5.1	<i>Inference of Unmeasurable Parameters</i>	4-19
4.5.2	<i>Model Operational Modes</i>	4-19
4.5.2.1	Run-Forward Mode	4-20
4.5.2.2	Real-Time Mode	4-20
4.5.2.3	Historical Mode	4-20
5	INTRODUCTION TO THE PLANTSTAR® CONTROL PLATFORM.....	5-1

5.1	PLANTSTAR'S LAYERED APPROACH.....	5-1
5.1.1	<i>Process Infrastructure</i>	5-2
5.1.2	<i>Process Stabilisation</i>	5-2
5.1.3	<i>Process Optimisation</i>	5-2
5.1.4	<i>Process Management</i>	5-3
5.2	PLANTSTAR ARCHITECTURE.....	5-4
5.3	PLANTSTAR USER INTERFACE.....	5-6
5.3.1	<i>Configuration Module</i>	5-6
5.3.2	<i>The Graph Tool</i>	5-7
5.3.3	<i>The Mimic Tool</i>	5-9
6	DYNAMIC OBSERVER MODEL IMPLEMENTATION IN PLANTSTAR....	6-1
6.1	THE MODULAR APPROACH.....	6-1
6.2	THE DYNAMIC LINK LIBRARY DEVELOPMENT.....	6-2
6.2.1	<i>Requirements Of the DLL for Use with PlantStar</i>	6-2
6.2.1.1	Software Requirements and Updates.....	6-2
6.2.1.2	Member Functions Required by PlantStar	6-3
6.2.1.3	Data Types and Handling.....	6-4
6.2.1.4	PlantStar Libraries and Include Files.....	6-4
6.2.2	<i>Model Class</i>	6-5
6.2.2.1	Model Header File	6-5
6.2.2.2	Model File.....	6-5
6.2.3	<i>Solution Methods Available</i>	6-5
6.2.3.1	MATLAB Engine	6-5
6.2.3.2	MATLAB Standalone File.....	6-6
6.2.3.3	Sparse Matrix Library Solution	6-6
6.2.4	<i>External Dependancies of the DLL</i>	6-6
6.2.4.1	MATLAB Engine	6-6
6.2.4.2	Matrix Manipulation Libraries.....	6-7
6.2.5	<i>Variable Handling between PlantStar and Model Class</i>	6-7
6.2.6	<i>Testing The DLL with Visual Basic</i>	6-9
6.3	THE PLANTSTAR PLANT DEFINITION DEVELOPMENT.....	6-11
6.3.1	<i>The Process Tree</i>	6-11
6.3.2	<i>The Process Strategy Tree</i>	6-12
6.3.2.1	Connecting to the DLL	6-13
6.3.2.2	Static Parameters.....	6-14

6.3.2.3	Inputs to the DLL.....	6-15
6.3.2.4	Outputs from the DLL.....	6-17
6.3.2.5	Parameters.....	6-17
6.3.3	<i>Graphing the Outputs</i>	6-18
6.3.4	<i>Debugging the System</i>	6-19
6.3.5	<i>The Model Execution Flowsheet</i>	6-21
6.4	USER INTERFACES AND MIMICS.....	6-22
7	DISCUSSION AND RESULTS	7-1
8	CONCLUSION	8-1
	REFERENCES	I
	APPENDIX A GLOSSARY OF TERMS	I
	APPENDIX B CLASSIFICATION OF FLOTATION CIRCUIT PROCESS VARIABLES	III
	APPENDIX C OTHER METHODS OF DAE SOLUTION	VIII
C.1.1	<i>A Simple DAE Case</i>	<i>VIII</i>
C.1.2	<i>Explicit Numerical Integration Methods</i>	<i>XI</i>
C.1.2.1	Explicit Euler.....	XI
C.1.2.2	Runge-Kutta (Fourth Order).....	XII
C.1.3	<i>Implicit Euler</i>	<i>XII</i>
	APPENDIX D MICROSOFT VISUAL BASIC 6 TESTING PROGRAM CODE .	XV
D.1.1	<i>Program Declarations</i>	<i>XV</i>
D.1.2	<i>The Form Load Code</i>	<i>XVII</i>
D.1.3	<i>The Run Code</i>	<i>XIX</i>
D.1.4	<i>Shutdown Code</i>	<i>XIX</i>
D.1.5	<i>Example of the Display Code for Primary Rougher 1</i>	<i>XIX</i>
	APPENDIX E MATLAB COMPILER INFORMATION	XXI
	APPENDIX F DETAILED DESCRIPTION OF VARIABLE HANDLING BETWEEN PLANTSTAR AND MODEL CLASS	XXIV
	APPENDIX G DART VALVE CALIBRATION	XXIX
	APPENDIX H MODEL TUNING AND SOLUTION PARAMETERS	XXXI
H.1	MODEL SOLUTION PARAMETER SET.....	XXXI

H.2	AUTOMATIC COVARIANCE MATRIX TECHNIQUE	XLII
APPENDIX I	COMPLETE MIMICS AND GRAPHICAL USER INTERFACES (GUI).....	XLV
APPENDIX J	THE KALMAN FILTER.....	LIV
J.1	INTRODUCTION TO THE KALMAN FILTER	LIV
J.2	THE DISCRETE KALMAN FILTER.....	LV
J.3	THE EXTENDED KALMAN FILTER.....	LVIII

LIST OF FIGURES

Figure 1-1:	Simple block flowsheet (Wills, 1997)	1-5
Figure 1-2:	Magnetic counter rotation drum separator (Wills, 1997).....	1-6
Figure 1-3:	Cross section of a spiral (Wills, 1997).....	1-7
Figure 1-4:	Cross-sectional diagram of a flotation cell	1-9
Figure 2-1	The Two Phase Model of Arbiter and Harris (1962).....	2-8
Figure 2-2	The Three Phase Model of Hanumanth and Williams (1992)	2-10
Figure 2-3:	The effect of mean residence time on overall recovery (Mathe et al. 1998).	2-11
Figure 2-4:	A figure representing the definition of the angle of tangency θ_t (after Dai et al 2000)	2-16
Figure 2-5:	Classification of collectors (Wills, 1997)	2-19
Figure 2-6:	Flotation zone rate constant (k) as a function of froth depth (FD) (Vera et al., 2002).....	2-21
Figure 3-1:	A typical flotation circuit layout (Schroder,1999).....	3-2
Figure 3-2:	Counter current circuit design (Schroder, 1999).....	3-3
Figure 3-3:	The B-Stream Circuit Layout at LONMIN Eastern Platinum	3-4
Figure 3-4:	Flotation control objective (Wills, 1997).....	3-8
Figure 4-1:	LONMIN's Eastern platinum B-Stream circuit configuration.....	4-4
Figure 4-2:	The comparison between a single cell's unit step response and that of a bank of 5 cells.	4-6
Figure 4-3:	The flotation cell modelling balances and variables	4-7
Figure 4-4:	The mills and surge tanks variables and balances.....	4-9

Figure 4-5:	The three different types of elements found in the flowsheet modelling of the LONMIN Eastern platinum B-stream(C = Flotation Cells, MT = Mills and Tanks, PS = Product Streams)	4-9
Figure 4-6:	Use of receiving node selection matrix representing stream splitters.....	4-10
Figure 5-1:	PlantStar's layered approach to control (PlantStar user manual, 2002).....	5-1
Figure 5-2:	PlantStar Architecture (PlantStar user manual, 2002)	5-4
Figure 5-3:	An example of a plant definition file showing the hierarchical rule tree structure (DemoPlant definition file, 2002)	5-7
Figure 5-4:	Creating a new trend on a graph with the graph tool (DemoPlant definition file, 2002)	5-8
Figure 5-5:	A graph of the demonstration plant from PlantStar (DemoPlant definition file, 2002).....	5-9
Figure 5-6:	Mimic of the demonstration plant from PlantStar (DemoPlant definition file, 2002)	5-10
Figure 6-1:	The member functions of the PSTest81B.TestClass.....	6-3
Figure 6-2:	A screenshot of the Microsoft Visual Basic 6 testing program used to test the DLL in the early stages.	6-10
Figure 6-3:	The Process tree structure of the DOM DLL plant definition file for Primary Rougher 1.....	6-12
Figure 6-4:	The Connection of PlantStar to the DLL class through the algorithm.....	6-13
Figure 6-5:	The Static parameter node from the plant definition file.	6-14
Figure 6-6:	The inputs node of the plant definition file and properties of Inflow_X1	6-16
Figure 6-7:	The mass flowrate of material leaving the cells in the concentrate (Frothflow_fm) after a step in mass pull of 15% for all banks (The trends number corresponds to the unit numbers which is consistent for all graphs unless otherwise indicated.).....	6-19

Figure 6-8:	The responses window from PlantStar used for debugging.....	6-20
Figure 6-9:	The execution loop of the Dynamic observer model through PlantStar.	6-21
Figure 6-10:	The user interface for the primary mill	6-23
Figure 6-11:	The feed condition to the simulated plant.....	6-23
Figure 6-12:	The primary flotation circuit user interface	6-24
Figure 6-13:	The detailed user interface of the primary Rougher 1	6-25
Figure 6-14:	Cell bank properties other than flow rate conditions	6-25
Figure 6-15:	Model Parameters user interface.....	6-26
Figure 6-16:	The graph links user interface.....	6-28
Figure 7-1:	A step in the feed flow rate of 30% to the plant (left axis is the mass fractions and the right axis, the mass flowrates).....	7-6
Figure 7-2:	Effect of the 30% feed flowrate step on the tailings flows (Left axis is the cleaners and right axis, the roughers, mills and tanks)	7-7
Figure 7-3:	The platinum mass fraction leaving in the tailings flows as a result of the reduced residence time in the cells after the increase in feed flowrate.....	7-8
Figure 7-4:	The slight increase in the platinum mass fraction in the concentrate after the feed flowrate step.....	7-8
Figure 7-5:	The concentrate mass flowrates after the step in feed flow showing the slight increase in flow of concentrate from the secondary roughers.....	7-9
Figure 7-6:	The final combined concentrate flows and mass fractions after the step.....	7-10
Figure 7-7:	The final combined tailings conditions after the feed flow step.	7-10
Figure 7-8:	The step in cell factors of 15% for all of the cells	7-11
Figure 7-9:	Mineral mass flowrate of the concentrates exiting the 10 flotation cells after the step in cell factors.	7-11

Figure 7-10:	The liquid concentrate mass flowrate leaving the 10 flotation cells after the step test.	7-12
Figure 7-11:	The minerals mass flowrates and the liquid mass flowrates combined for the tailings streams of unit operation 1 to 4 and 11 to14.....	7-12
Figure 7-12:	The minerals mass flowrates and the liquid mass flowrates combined for the tailings streams of unit operation 5 to 10.....	7-13
Figure 7-13:	This graph shows the reduction in the platinum mass fraction of the tailings flows as a result if the increased concentrate mass flows.....	7-14
Figure 7-14:	Platinum mass fraction in the concentrate flows leaving cells after the cell factor step	7-14
Figure 7-15:	The step in the tailings flowrate of 50%	7-15
Figure 7-16:	The cell factors converged positions after the step in tailings flow for unit 5.	7-16
Figure 7-17:	Mass flowrates of concentrates resulting from the new cell factors positions.	7-16
Figure 7-18:	The step in feed grade from 4.6 g/ton to 9.2 g/ton.....	7-17
Figure 7-19:	The increase in platinum mass fraction in the concentrate due to the step in feed grade.....	7-18
Figure 7-20:	Trends showing the increased loss to the tailings of platinum due to the higher feed grade.	7-19
Figure 7-21:	Indicators of the performance of the model solution and its convergence towards steady state.	7-20
Figure C-1:	Explicit Euler solution as compared to exact solution.....	XI
Figure F-1:	The Observation Tree structure for Unit 1.....	XXV
Figure F-2:	The outputs tree structure in the plant definition file for Unit 1.....	XXVII

Figure G-1:	Dart valve calibration to determine mass flowrates from valve positions on LONMIN's pilot plant.....	XXIX
Figure I-1:	The Primary Mill GUI.....	XLV
Figure I-2:	The Primary Flotation Circuit GUI.....	XLVI
Figure I-3:	The Secondary Mill GUI.....	XLVI
Figure I-4:	The Secondary Flotation Circuit GUI.....	XLVII
Figure I-5:	The Model Parameters GUI.....	XLVII
Figure I-6:	The Graph Links GUI.....	XLVIII
Figure I-7:	Detailed GUI of Unit 1.....	XLVIII
Figure I-8:	Detailed GUI of Unit 2.....	XLIX
Figure I-9:	Detailed GUI of Unit 3.....	XLIX
Figure I-10:	Detailed GUI of Unit 4.....	L
Figure I-11:	Detailed GUI of Unit 5.....	L
Figure I-12:	Detailed GUI of Unit 6.....	LI
Figure I-13:	Detailed GUI of Unit 7.....	LI
Figure I-14:	Detailed GUI of Unit 8.....	LII
Figure I-15:	Detailed GUI of Unit 9.....	LII
Figure I-16:	Detailed GUI of Unit 10.....	LIII

LIST OF TABLES

Table 2-1:	Values of A and n for different collision probability models.....	2-17
Table 4-1:	Table of variables used in model solution, of the flotation cells, mills and tanks (i = element, j = species).....	4-11
Table 4-2:	Model initialisation set of variables.....	4-11
Table 4-3:	Model input on each time step. (Minimum set of observations required for convergence, however any variables may be observed to utilise the Kalman filter)	4-12
Table 4-4:	Model Outputs on each time step.....	4-12
Table 6-1:	Table of variables comparison between those used by PlantStar and those used by the DLL and model class (The 'i' in the model names indicates inflow and is replaced by the word 'inflow' in the names of the PlantStar variables).	6-9
Table 6-2:	Table of the graphs and variables plotted by PlantStar.....	6-18
Table 7-1:	A comparison between the converged set of flotation rate constants obtained using the extended Kalman filter model solution compared to the same flotation rate constants obtained from the cell inventories and the plant data set	7-5
Table B-1:	Typical Controlled Variables in Flotation Circuits (Schroder, 1999).....	III
Table B-2:	Typical Disturbance Variables in Flotation Circuits (Schroder, 1999).....	IV
Table B-3:	Typical Measurements Available in Flotation Circuits (Schroder, 1999)	V
Table B-4:	Typical Manipulated Variables in Flotation Circuits (Schroder, 1999).....	VI

NOMENCLATURE

φ	Air hold-up (volume/volume)
f	Factor of 2 for free non-retarded bubble surface
ϕ_λ	Shape factor (dimensionless)
ϕ_n	Area factor (= 1 for a sphere) (dimensionless)
θ_t	Angle of tangency
ϕ_v	Volume factor (= 1 for a sphere) (dimensionless)
a	Rate order of the process for mineral i (dimensionless)
A_{cell}	Area of the base of the cell (area)
A_p	Surface area of particle (area)
C_i	Rate of removal of mineral i from the cell (mass/time)
Col_i	Number of collisions per second per bubble (number/time.bubble)
CR_i	Degree of entrainment on the basis of tailings
D_l	Second longest dimension of the particle (length)
d_b	Bubble diameter
d_{bubble}	Sauter mean bubble diameter (length)
D_e	Equivalent diameter of particle (length)
d_i	Particle size (length)
d_p	Particle diameter
E_c	Collision efficiency (1/time)
F	Froth stability factor
$f(k_i)$	Continuous distribution function for flotation rate constants of species i
g	Acceleration due to gravity (length/time ²)
k	Specific flotation rate (1/time)
K	Stokes number
k_i	Flotation rate constant of mineral i (mass ^{1-n-m} /time)
$k_{kinetic\ i}$	Kinetic rate constant for mineral i (mass ^{1-a} /time)
$k_{phase\ i}$	Phase rate constant for mineral i (1/time.mass)
L	Length of path travelled by bubble through the suspension (length)
M_b	Mass of air bubbles available for flotation (mass)

$M_{e,i}(h)$	Entrained mass flow rate at froth height h (mass/time)
M_{fpulp}	Mass flow rate of species from pulp into froth (mass/time)
$M_{f,i}(h)$	Mass flow rate of species i attached to bubbles at froth height h (mass/time)
M_i	Mass of component i in each phase (mass)
M_i	Mass of mineral i in the pulp (mass)
$M_i(h)$	The mass of mineral i at froth height h (mass)
M_p	Mass of mineral i in the pulp (mass)
M_{pulp}	Mass of species per unit volume in the pulp (mass/volume)
$M_{r,i}(h)$	Mass flow rate of particles diffusing from the upward-flowing stream to the downward-flowing stream (mass/time)
N	Number of particles collected (dimensionless)
n	Number of cells in series
n, m	Respective orders for particles and bubbles (dimensionless)
N'	Number of bubbles per unit pulp volume (bubbles/volume)
N_0	Number of particles per unit feed volume (particles/volume)
N_i	Rate of collision between particles and bubbles (collision/time)
N_p	Number of particles per unit volume of pulp (number/volume)
P	Probability of flotation of any mineral constituent in any one cell
P_a	Probability of adhesion
P_{at}	Probability of attachment
P_c	Probability of collision
P_f	Probability that particle survives the froth
P_r	Probability of the elementary step "rupture of the film"
P_t	Probability of the thinning of the film to critical thickness
P_{TPC}	Probability of the formation of the three-phase contact line
Q_G	Volumetric flow rate of air into the cell (volume/time)
R	Radius of bubble (length)
r	Radius of particle (length)
Rec_i	Overall recovery of mineral i after time t
$Rec_{i,\infty}$	Ultimate recovery after a long time
R_i	Rate of transfer of mineral i between phases (mass/time)
S_B	Bubble surface area flux (area/area.time)
t	Time after which R_i is measured
$u(h)$	Velocity of the downward-flowing stream
V	Velocity of bubble relative to the particle (length/time)
$v(h)$	Velocity of the bubble films

V_{bubble}	Average vertical velocity of the bubbles (length/time)
V_G	Superficial gas velocity (length/time)
V_p	Pulp volume(volume)
v_p	Volume of particle (volume)
W	Weight of mineral constituent remaining in the pulp after passing through n cells
W_0	Weight of mineral constituent present in the flotation feed
α, β	Empirical parameters
λ	Induction time (time for bubble and particle to collide and attach) (time)
μ	Fluid Viscosity (power/area)
ρ_f	Fluid density (mass/volume)
ρ_p	Particle density (mass/volume)
ρ_{pulp}	Specific gravity of the pulp
ρ_s	Specific gravity of entrained material
τ_{fg}	Froth residence time (time)
τ_{fs}	Specific froth residence time (time/length)

1 INTRODUCTION

1.1 ORGANISATION OF THIS THESIS

Chapter One: Introduction

This chapter gives an introduction and background to mineral processing and some of the methods of minerals separation and concentration before going into more detail on the specific separation method of interest to this project, that of froth flotation. Some of the common problems faced by industrial flotation circuits are discussed followed by the objectives of this thesis.

Chapter Two: Flotation Modelling Review

This chapter reviews the literature on current flotation modelling with regards to fundamental models, empirical models and semi-empirical models. Kinetic rate modelling of flotation sub-processes are reviewed in more detail.

Chapter Three: Flotation Circuit Design, Control and Optimisation

Flotation circuit design is analysed and discussed from a plant performance and valuable minerals recovery point of view. Optimisation of flotation circuit designs are reviewed and discussed. The control systems most commonly found on flotation circuits are reviewed and each of their pros, cons and specific areas of implementation are discussed.

Chapter Four: Dynamic Observer Model (DOM) Development

The objectives of the dynamic observer model are laid out, before the steps required to meet these objectives are considered. A detailed description of the LONMIN B-Stream circuit model development and its topology, structure and mathematical solution are discussed. Features of the model and its usefulness are investigated at the end of the chapter.

Chapter Five: Introduction to the PlantStar® Control Platform

An introduction and overview of the control platform used to implement the dynamic observer model is given. Some of the features and functions available on this control platform

are discussed and demonstrated to give some enlightenment as to the work carried out in the next chapter.

Chapter Six: Dynamic Observer Model Implementation in PlantStar

The modular approach of PlantStar is further discussed after which a detailed description of the development of the dynamic link library (DLL) is given. The requirements of the DLL for use with PlantStar are discussed along with the model class implementation within the DLL and solution methods available for the model. External dependencies of the DLL and variables handling between the model class and the control platform are investigated. A short description of the testing and debugging of the DLL is described before a detailed examination of the plant definition file development. The functioning and features of the user interfaces and graphing of variables is discussed once a firm understanding of the process trees within the plant definition file has been established.

Chapter Seven: Discussion and Results

The project is briefly summarised and discussed before some results obtained from test simulations are analysed.

Chapter Eight: Conclusion

Conclusions about the modelling, and applicability of the dynamic observer model “plugin” module created for the PlantStar control platform are considered.

1.2 BACKGROUND TO MINERAL PROCESSING

The increase in the demand for metals in the last century has been dramatic and has led to many large-scale mining operations being undertaken worldwide. Economically, mining is costly due to its high-energy consumption and it is therefore affected by changes in the global price of oil. In contrast, the price of metals is generally governed by supply and demand, but some common metals such as copper have dropped in some cases and are no longer economical to mine (Wills 1997).

Metals reside in the earth's crust in different forms depending upon their reactivity and the environment in which they are found. Gold, and the platinum group metals, for instance are found primarily in their elemental or metallic form, whereas other metals such as silver and copper can sometimes be found in the form of sulphides, carbonates and chlorides. Compounds such as these are known as minerals and are, by definition, natural inorganic substances possessing definitive chemical compositions and atomic structures. However, this definition is flexible, as some minerals exhibit iso- and polymorphism, which results in differences in their structure and composition.

If all minerals were distributed equally throughout the earth's crust there would be no economic way of extracting the valuable metals, but fortunately due to natural geological conditions, mineral deposits are frequently found in sufficient concentration as to enable profitable recovery of the metals, at which point the deposit is known as an ore body. Ores are mostly mixtures of extractable minerals and worthless rocky material known as *gangue*. Generally, classification of ores is based on the nature of the valuable minerals they contain as well as the nature of their gangues.

Mining the ores is one of the major costs involved and can vary greatly, hence some ores of very low contained value can still be economically mined given the correct mining techniques, such as high-tonnage open-pit and underground block caving methods. Only ores of very high-contained value can be mined from underground vein-type deposits, as these methods are very expensive. Platinum and gold are found in such deposits and due to their high value warrant being recovered in this manner.

1.3 MINERAL PROCESSING OVERVIEW

Once the ore has been removed from the earth, further processing is required to obtain the refined metal. Metals are recovered primarily through a smelting process whereby ore is added to a furnace within the smelter and reduced to its elemental form. When mining an ore, much of the material recovered from the earth is worthless gangue. Due to the high cost of smelting and the smelters' often remote location with respect to the mine, it is economically unfeasible for large amounts of this worthless material to be transported to the smelter and to be passed through the very expensive process of smelting in order to recover the small amount of valuable metal within the ore. Thus most mining operations have some form of mineral processing on or very near the mine, so that the bulk of the transportation of ore is over a short distance. Mineral processing consists of the physical separation of valuable minerals from the bulk of the material to produce a concentrate of valuables and a tailings of discarded material. It follows mining and preparing the ore for the extraction of the valuable metal. Ore is processed to increase the concentration of valuable metals so that the amount of material sent through the smelter is kept to a minimum and transportation costs are reduced. The economic advantage gained through this on-site mineral processing must be weighed against the increased cost of the milling operations and the losses incurred during these concentration processes. One of the most significant factors to be considered is the loss to tailings, which is affected by the ore mineralogy and depends on the technology available to achieve efficient separation. Through the development of froth flotation many previously low-grade deposits have become economical to process. Mineral processing operations are often a compromise between mineral processing costs and improvements in metallurgical efficiency. With high value ores such as platinum bearing ores, high metallurgical efficiency is of more importance and the cost of high-efficiency mineral processing can be justified (Wills, 1997).

In some cases there is more than one valuable mineral in the ore, resulting in the need for further separation of these valuable minerals, from both the gangue and each other. Similarly with undesirable minerals that may interfere with the refining of the valuable mineral, such as chrome found in platinum bearing ore. It is therefore imperative to remove these undesirable minerals at the separation stage. The two fundamental operations in mineral processing are liberation and concentration. Liberation is the release of the valuable minerals from their entrapping gangue minerals, with the separation of these valuable minerals from the gangue being known as concentration.

Liberation of the valuable mineral is accomplished by comminution, which involves crushing and grinding so as to reduce the ore particle size and release any mineral that could be completely entrapped inside the gangue. At small particle sizes any of the valuable material that exists in the particle should be exposed to the surface, which is required for some separation techniques such as froth flotation. As particle size drops, the grinding energy requirement to reduce size further increases greatly and can account for a very large percentage of the concentrator's total energy consumption.

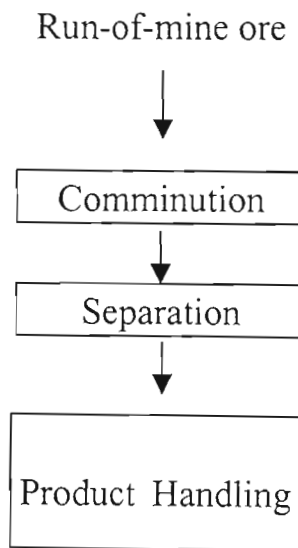


Figure 1-1: Simple block flowsheet (Wills, 1997)

To achieve clean concentrates with little contamination from gangue it is necessary to grind the ore finely enough to liberate the associated metals. However this leads to increased energy costs and can sometimes produce very fine particles, which result in untreatable slimes often lost to the tailings. Grinding therefore becomes a compromise between clean high-grade concentrates, operating costs and the loss of fine minerals. A good knowledge of the aggregation and dissemination of the minerals within the ore is required for efficient processing to be carried out.

1.4 METHODS OF SEPARATION

Based upon the characteristics of the valuable mineral in the ores and the nature of the gangue, different separation techniques can be used to produce a clean concentrate. Some of the most important properties with regards to separation are optical and radio active properties, specific gravity differences, surface properties, magnetic properties and electrical conductivity properties.

Magnetic separators achieve separation through the difference in the magnetic properties of the minerals found in ores. All materials are affected by magnetic fields in some way, however with most substances the effect is too slight to be detected, therefore magnetic separation can only be used in special cases where the minerals in the ore exhibit specific magnetic properties, which can be exploited to an appreciable extent through magnetic fields. Ferromagnetic minerals, such as magnetite, exhibit strong magnetic properties, which can be used to aid in their separation through low intensity magnetic separators, whilst paramagnetic minerals require much higher intensity magnetic fields to achieve separation. Magnetic separation has been used mainly in the iron ore beneficiation process, but has found use in the removal of small amounts of unwanted minerals in otherwise desirable ores.

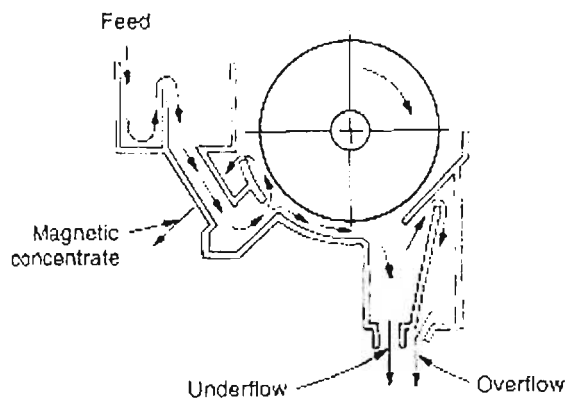


Figure 1-2: Magnetic counter rotation drum separator (Wills, 1997)

High-tension separation can be used to separate conducting minerals from non-conducting minerals. Because all minerals show some difference in their conductivity it should be possible to separate almost anything, however high-tension separation is expensive and requires completely dry minerals, as water could change the nature of the electron movement.

Gravity concentration methods separate minerals of different specific gravity by their movements in response to gravity and some other force, usually their resistance to motion in a viscous fluid such as water. In order for gravity separation to be effective, an appreciable difference in the specific gravities of the mineral and gangue must exist. An idea of the possible separation achievable through gravity separation techniques can be gained from the concentration criterion. Gravity methods are relatively simple and cheap to implement and operate, as there are few requirements of expensive equipment, no reagents and little energy expenditure. As such, it can play an important role in terms of pre-separation in some industries such as platinum mining, where large amounts of fine material can be removed in spiral separators.

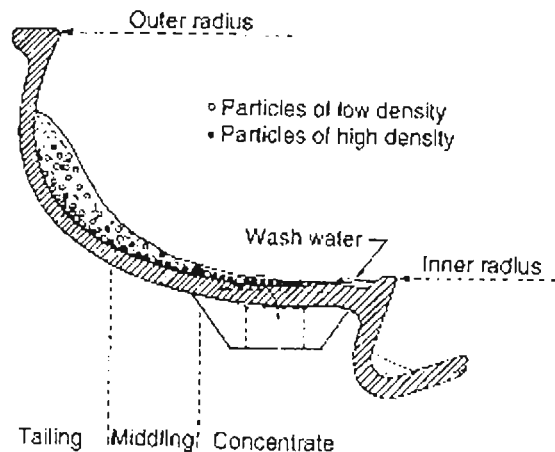


Figure 1-3: Cross section of a spiral (Wills, 1997)

Already fine particles, such as ilmenite found in beach sand, are separated very successfully through the use of gravity separation. Some examples of gravity separators are spirals, jigs, pinched sluices, cones, shaking tables and centrifugal concentrators.

One of the most common separation methods in South Africa, and most pertinent to this thesis, is froth flotation, which utilises the difference in surface properties of minerals. Froth flotation is effected by the affinity of minerals for rising up attached to air bubbles within an agitated pulp, and by adjusting the reagents within the pulp, it is possible to make the valuable minerals aerophilic and the gangue minerals aerophobic. As a result, the valuable minerals are transported to the surface froth layer via the bubbles and the gangue minerals remain in the pulp, thereby achieving separation.

Optical separation is often simply hand sorting by suitably experienced people. High-grade ores can be recognised and then set aside for further processing. An obvious example of hand sorting would be in the mining of diamonds, where the diamond can easily be seen among the gangue mineral and hence recovered. This method is however increasingly being replaced by machine sorting approaches such as x-ray and γ -ray detectors.

1.5 FROTH FLOTATION

As mentioned above, froth flotation is the most important method of concentrating valuable minerals, in terms of present mineral processing techniques. The principle of flotation is quite simple, yet its theory is complex and not entirely understood. Under controlled conditions and through the use of chemical reagents known as collectors, some valuable minerals can be made hydrophobic, whilst their associated gangue minerals are selectively made hydrophilic. The resulting difference in surface properties of the minerals can then be exploited in a well-stirred and aerated tank known as a *cell*. Through vigorous mixing, the suspension of fine ore particles will collide with air bubbles. The hydrophobic mineral will attach itself to a bubble surface to escape contact with the surrounding pulp phase, which consists largely of water. As a result the bubble will have a higher concentration of valuable minerals attached to it. It then floats to the surface of the pulp to form a froth foam floating on the surface of the cell. The gangue minerals do not attach themselves to the bubbles and therefore remain in the pulp phase. The froth is collected in overflows resulting in a degree of separation. This process is very inefficient and results in many cells being necessary to achieve a sufficiently high concentration of valuable minerals. Below is a schematic diagram of a cell and how it functions.

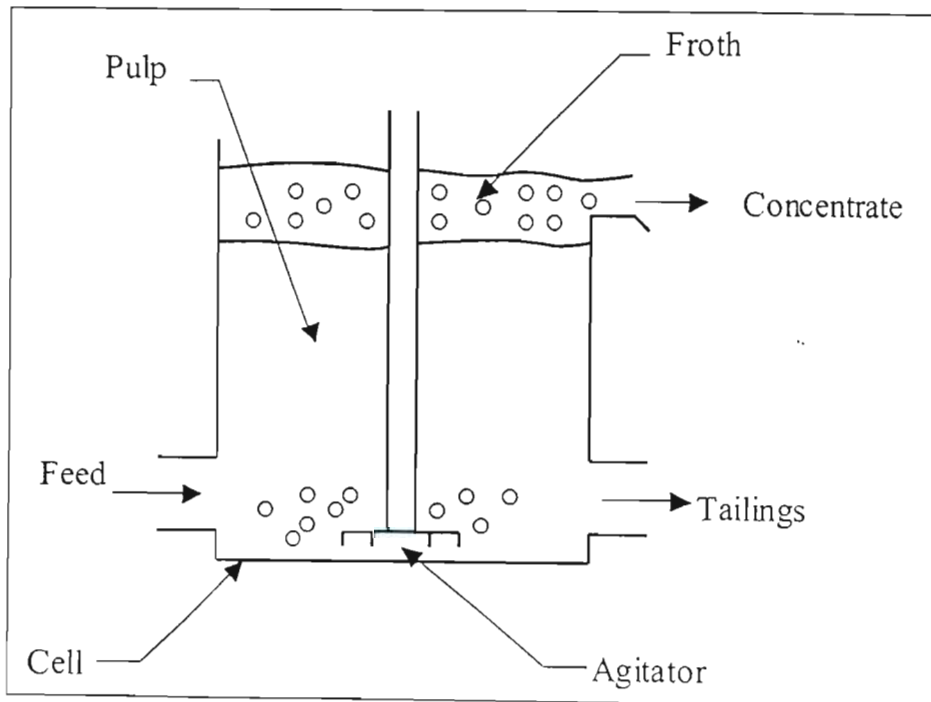


Figure 1-4: Cross-sectional diagram of a flotation cell

It can be seen in figure 1-4 that there are two phases present in the cell, the froth phase consisting of bubbles floating on the surface of the cell and the *pulp* phase containing the fine suspension of ore. The froth *concentrate* is removed from the cell by an overflow or with the use of mechanical paddles, depending upon the equipment used. The *tailings*, rich in gangue minerals, leave via an underflow. Many cells in series are known as a *bank*, where the tailings flow from one cell becomes the feed to the next. There are many factors that influence the mechanics of flotation and these will be covered in detail in the following chapter.

1.6 PROBLEMS FACED BY INDUSTRIAL FLOTATION CIRCUITS

In industrial flotation circuits there are a number of common problems that plague the flotation process, resulting in increased losses and reduced returns. Many researchers have tried to address these issues in the past and have posed numerous methods of improvement. Some of these problems will be put forward in this section.

- Recycles in flotation are necessary due to the poor efficiency of the flotation process and these ultimately increase the recovery of valuable minerals, however they also result in large mass hold-ups in the circuit. Due to these large internal recycles, the circuit operation and disturbances are difficult to control.
- The feed conditions and characteristics of the circuit can vary greatly. As the ore body is not consistent throughout, the chemical and physical properties of the minerals change. The flotation characteristics of the feed will therefore fluctuate with time and this creates difficulties in achieving constant concentrate grades while there are disturbances passing through the circuit.
- Flotation circuits have few on-line measurements. The nature of the flotation process and its circuit design limit the number of access points where meaningful variables can be measured. Often operators' visual inspection and experience are the judge of performance and conditions throughout the circuit. This limits the control options available to the circuit.
- Certain disturbances in the circuit cannot be easily detected. The overall performance of the circuit is governed by the flotation of the valuable minerals. These flotation properties cannot be measured directly and therefore a loss in controllability and circuit performance results.

From the above it is noted that a better understanding of the processes behind flotation is required in order to improve control strategies used by flotation circuits. A useful tool in maintaining control could be a well-defined and operational plant model running alongside the plant control platform in real-time, predicting the effect of changes and disturbances along with unmeasurable variables.

1.7 OBJECTIVES OF THIS THESIS

The work in this thesis focuses primarily on one flotation circuit, namely, LONMIN Eastern Platinum, B-Stream. Complex circuits such as these can enter sub-optimal modes easily as feed conditions change. More immediate feedback from a dynamic observer model could enhance process understanding and facilitate prompt corrective actions, whether in open or closed loop. Interpretation of key combinations of variables deduced by the observer model, using an expert system, would possibly add another level of intelligence, benefiting the operation. As such, the process of creating a dynamic observer model and implementing it was the main focus of this research. In order to achieve this, certain requirements had to be met:

- Create an accurate real-time flotation model of the B-Stream flotation circuit.
- Interface the plant model to the existing plant control platform, PlantStar®.
- Test the model's accuracy using the control platform running "what if" scenarios.

2 FLOTATION MODELLING REVIEW

Much research has recently been carried in flotation modelling and many varying ideas have been put forward to fully identify the flotation process. However no one model has been accepted as the norm, but some types of models have been more widely accepted than others.

2.1 GENERAL TYPES OF FLOTATION MODELS

There are three basic flotation model categories:

- Fundamental models: These models attempt to characterise flotation performance using the basic laws of physics and chemistry.
- Empirical models: Process inputs are used to correlate flotation performance by regression.
- Semi-Empirical: Empirical relationships describing the process are used in conjunction with fundamental principles of flotation.

2.1.1 FUNDAMENTAL MODELS

Sutherland (1948) proposed that the rate of collection of mineral to the froth was proportional to the product of the rate of minerals colliding with bubbles and the probability that the particle would adhere to the surface of the bubble. Then, by considering the flow patterns around bubbles, he described the rate of removal of mineral to concentrate as:

$$C_i = N_i P_a P_f \quad \text{Eq 2-1}$$

$$N_i = 3\pi R r V N_0 N' \quad \text{Eq 2-2}$$

$$\gamma_i = \sec h^2 \left(\frac{3V\lambda}{4R} \right) \quad \text{Eq 2-3}$$

where:

C_i Rate of removal of mineral i to the concentrate (mass/time)

N_i	Rate of collision between particles and bubbles (collision/time)
P_a	Probability of adhesion of the particle to the bubble (dimensionless)
P_f	Probability of attached particle not detaching (dimensionless)
R	Radius of bubble (length)
r	Radius of particle (length)
V	Velocity of bubble relative to the particle (length/time)
N_0	Number of particles per unit feed volume (particles/volume)
N'	Number of bubbles per unit pulp volume (bubbles/volume)
λ	Induction time (time for bubble and particle to collide and attach) (time)

This model was the first to describe the rate of flotation based on physical principles, but was limited, as it did not consider:

- The chemical conditions of the pulp (Leal Filho et al., 2000)
- Sub-processes in the froth (Harris and Rimmer, 1966; Harris, 1978; Hanumanth and Williams, 1992)
- The effects of entrainment (Savassi et al, 1998; Neethling and Cilliers, 2002)
- The effect of weir angle on bubble motion (Neethling and Cilliers, 1998)
- The effect of particle-bubble attachment angle (Stechemesser and Nguyen, 1999; Dai et al., 1998)
- Inertial effects of the particle (Woodburn, 1970)

and many others that have come to the attention of recent researchers.

The fundamental models have the most insight into the process of flotation and therefore would be the most valuable when successful. The variables that are required by these models are, however, difficult to acquire in an industrial setting, which creates problems when implementing them into control strategies.

2.1.2 EMPIRICAL MODELS

Empirical models are developed through the gathering of input and output data from experimental or plant units and regressing the data to find models describing the processes. As a result, empirical models only show the correlation of inputs to process outputs as shown by the data. However the advantage of these models is that they require no physical insight or prior knowledge about process characteristics.

Gonzalez et al. (2003) made use of identification, or training data sets, with a number of models for use with soft sensors. The models used were the ARMAX, Takagi and Sugeno, fuzzy combinational, projection on latent states (PLS) and wavelet based models. These models were identified using actual rougher plant data from the Andina copper flotation plant. The results showed that the best dynamic model was the PLS, while the ARMAX, fuzzy combination and Takagi and Sugeno dynamic models gave large errors.

Empirical models are easy to develop and maintain but have little insight into process mechanisms. As well as their poor predictive ability, they are circuit specific and cannot be used to predict performance on alternative circuit configurations (Schroder, 1999).

2.1.3 SEMI-EMPIRICAL MODELS

2.1.3.1 Probability Models

Schuhmann (1942) analysed the mechanism required in order to float a particle and concluded that the process was a result of firstly, the probability of collision of the particle with a bubble, secondly, whether the particle would attach to the bubble, and thirdly, a factor of stability in which he considered that an attached particle might not float due to loss to the pulp through bubble coalescence. Schuhmann then proposed that the net rate of transportation of a particle to the concentrate was:

$$k = P_c P_a F \quad \text{Eq 2-4}$$

where:

k	Specific flotation rate (1/time)
P_c	Probability of collision
P_a	Probability of adhesion
F	Froth stability factor

Kelsall (1961) considered a flotation bank of n identical cells in series and applied a probability approach to obtain the fractions of mineral that would remain in the tailings,

$$W = W_0 (1 - P)^n \quad \text{Eq 2-5}$$

where:

W	Weight of mineral constituent remaining in the pulp after passing through n cells
W_0	Weight of mineral constituent present in the flotation feed
P	Probability of flotation of any mineral constituent in any one cell
n	Number of cells in series

2.1.3.2 Kinetic Rate Models

Schuhmann (1942) also considered that the flotation rate should be treated mathematically in a similar way as to the kinetics of first-order homogeneous reactions. This modelling procedure is based upon the particle-bubble aggregate being a collision process and therefore the recovery rate is dependant on the rate at which particles collide with air bubbles and are carried to the concentrate. The rate of flotation is then described by the usual rate law expression:

$$C_i = k_i M_p^n M_b^m \quad \text{Eq 2-6}$$

where:

C_i	Rate of removal of mineral i from the cell (mass/time)
M_p	Mass of mineral i in the pulp (mass)
M_b	Mass of air bubbles available for flotation (mass)
k_i	Flotation rate constant of mineral i (mass ^{1-n-m} /time)
n, m	Respective orders for particles and bubbles (dimensionless)

The process is usually considered to be carried out in excess air and is assumed to be first order with respect to mass. The flotation rate constant, k , is considered to be a complex function involving, amongst other things, reagent concentration, particle size, induction time, rate of froth removal, flotation cell design and previous treatment (Greaves and Allan, 1974).

2.1.3.3 Collision Models

Collision models stem from the early work carried out by Sutherland (1948) in which flotation is described in terms of bubbles rising through pulp and colliding with particles. The mineral recovery rate (Flint and Howarth, 1971) is then given by:

$$Col_i = \frac{\pi N_p r^2 R^2 V}{(K_1 u + r^2)} \quad \text{Eq 2-7}$$

where:

Col_i Number of collisions per second per bubble (number/time.bubble)

N_p Number of particles per unit volume of pulp (number/volume)

R Air bubble radius (length)

r Particle radius (length)

V Velocity of bubble relative to particles (length/time)

$$K_1 = \frac{9\mu}{2g(\rho_p - \rho_f)}$$

μ Fluid Viscosity (power/area)

g Acceleration due to gravity (length/time²)

ρ_p Particle density (mass/volume)

ρ_f Fluid density (mass/volume)

Collision efficiency is governed by the viscous, inertial and gravitational forces acting upon the particle, as well as the streamlines around the bubble, and is described by (Anfruns and Kitchener, 1977):

$$E_c = \frac{C}{\pi R^2 L N_p} \quad \text{Eq 2-8}$$

where:

E_c Collision efficiency (1/time)

N Number of particles collected (dimensionless)

R Radius of the bubble

L Length of path travelled by bubble through the suspension (length)

N_p Number of particles per unit volume in the suspension (number/volume)

Semi-empirical models can be tuned to particular circumstances and have the advantage that they contain some of the physical nature of the process. However, they still suffer from the limitations of all empirical models when used for extrapolation, although less severe because of the higher understanding of the processes that are incorporated into the models. Kinetic rate models in particular, have been used effectively in control and optimisation of flotation circuits (Schroder, 1999).

2.2 KINETIC RATE MODELLING OF FLOTATION SUB-PROCESSES

Flotation performance is affected by many factors, some of which are reviewed in the following section with special attention paid to those that could affect kinetic rate constants, and hence the control and optimisation of flotation circuits using models based on the kinetic approach.

2.2.1 FLOTATION RATE ORDER

The general form of the kinetic rate equation, when air is not rate limiting, is given by:

$$C_i = k_{kinetic_i} M_i^a \quad \text{Eq 2-9}$$

where:

C_i	Rate of transport of mineral i to concentrate (mass/time)
$k_{kinetic_i}$	Kinetic rate constant for mineral i (mass ^{1-a} /time)
M_i	Mass of mineral i in the pulp (mass)
a	Rate order of the process for mineral i (dimensionless)

There has been much discussion as to the true value of the rate order, a , in the above equation. Sutherland (1948) first suggested that flotation was a first order process, whilst other researchers have disputed this, finding that non-unity orders fitted their experimental data best (Greaves and Allen, 1974). It has been suggested that the rate order of flotation is affected by certain factors in the process, such as high pulp densities and hindered flotation.

Arbiter and Harris (1962) proposed a new model, where flotation was modelled as two phases, viz. pulp phase and froth phase (section 2.2.2). Greaves and Allen (1974) fitted their data to equations derived from this model in which the kinetic orders could be varied. They concluded that the order for transport of material from the pulp to the froth was 1.06 while the reverse order (material transported from the froth to the pulp) was about 2.6-3.1. They noted, however, that the inaccuracy of their experiment made it difficult to determine the true rate orders and in subsequent work both researchers chose to assume first order in both directions.

Harris (1978) in his review of flotation modelling, chose to assume first order kinetics for both directions after testing alternative values.

2.2.2 SUB-SYSTEMS CONSIDERED IN FROTH

Harris (1978) observed that flotation is the sum of a number of separate sub-processes and that these are often competitive. He concluded that models that separate these sub-processes are more likely to identify effects on flotation successfully.

Arbiter and Harris (1962) submitted the first multiphase kinetic model for froth flotation, which recognised that the cell contents are partitioned into froth and pulp phases and that two-way interchange can occur between these two phases. Schematically the model has the following structure:

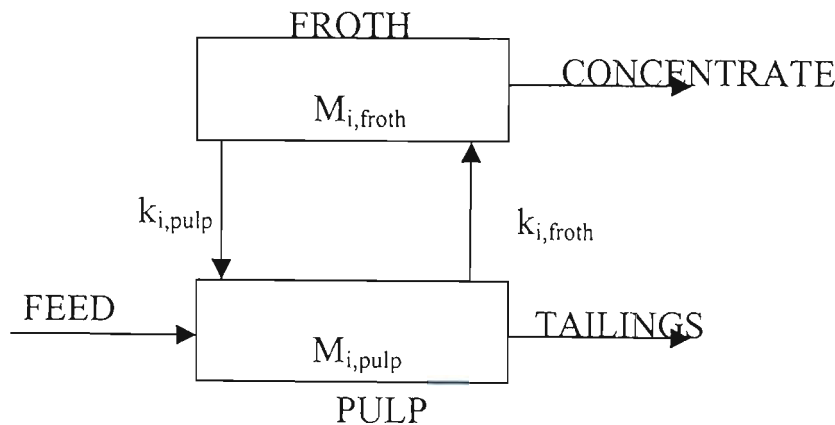


Figure 2-1 The Two Phase Model of Arbiter and Harris (1962)

$$R_{i,pulp_to_froth} = k_{phase_i,pulp} M_{phase_i,pulp} \quad \text{Eq 2-10}$$

$$R_{i,froth_to_pulp} = k_{phase_i,froth} M_{phase_i,froth} \quad \text{Eq 2-11}$$

where:

R_i Rate of transfer of mineral i between phases (mass/time)

$k_{phase\ i}$ Phase rate constant for mineral i (1/time.mass)

$M_{phase\ i}$ Mass of component i in each phase (mass)

Harris and Rimmer (1966) made use of an analog computer to validate the two-phase model and found that it captured flotation features that were not shown by the single-phase model. They found that it gave a better fit to their experimental data than the single phase models and that it could also manage a wider range of circumstances which might be expected to arise in real systems. It was noted that the model demonstrated that an increase in the feed rate led to an increase in the flotation rate and that it was a consequence of a non-zero froth return rate constant.

The assumptions made with the two-phase model as put forward by Harris (1978) are:

- The phases are each ideally mixed and as a consequence the tailings provides a pulp phase sample and the concentrate is representative of the froth phase.
- Transport of material takes place in one or both directions and the mass flowrate is dependent upon the mass of material in the phase from which it comes.
- Any order of kinetics can be accommodated by the model (Harris and Rimmer, 1966), but first order is usually assumed in both directions. Other cases have been tested (Greaves and Allan, 1974).

A three-phase model for froth flotation consisting of a pulp and two distinct froth phases in series, was developed by Hanumanth and Williams (1992). A simplified illustration of their model follows in figure 2-2.

The assumptions made in this model are:

- The pulp phase is well mixed and the rate of entry of solids from the pulp phase into the froth phase is proportional to the instantaneous mass of solid present in the pulp.
- The froth phase is divided into two well mixed layers, the first of which adjoins the pulp phase, referred to as the primary froth, and the remaining froth constitutes the secondary froth.

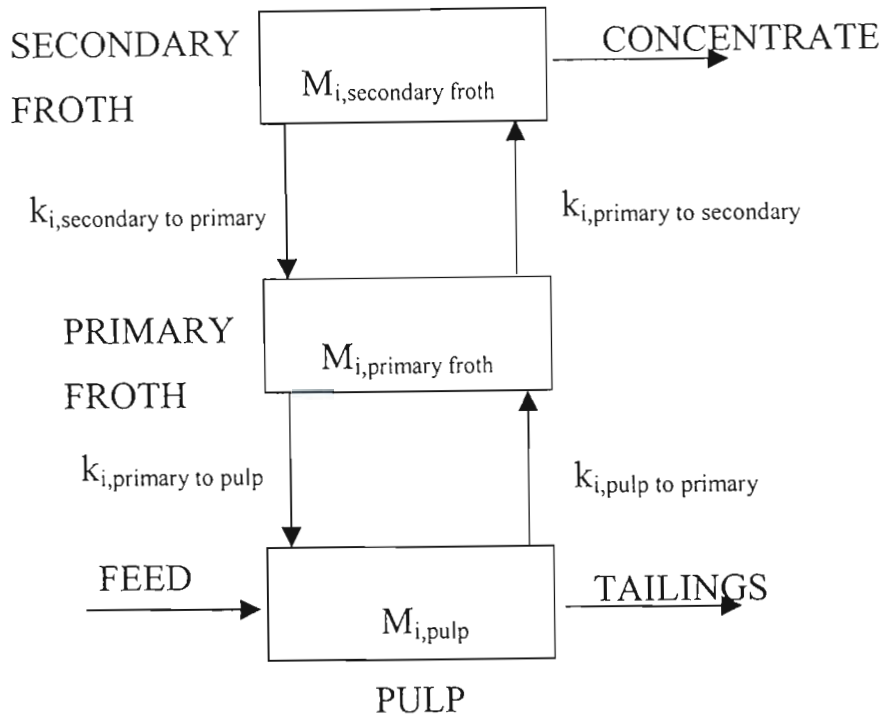


Figure 2-2 The Three Phase Model of Hanumanth and Williams (1992)

They concluded that this model is useful when estimating the solid drainage rate, its variation with froth depth, and its influence on recovery kinetics over a large range of froth depths.

Other researchers have put forward multi-phase models. All have a similar modelling strategy, but increase the number of parameters that can be fitted using linear regression techniques and thus obtain better fits to experimental data.

2.2.3 RESIDENCE TIME IN FROTH

The general form of flotation rate modelling is given by equation 2-9 and from this the overall recovery of a mineral can be obtained from the integrated form of that equation, namely

$$Rec_i(t) = Rec_{i,\infty} \left[1 - \int_0^{\infty} f(k_i) \exp(-k_i t) dk \right] \quad \text{Eq 2-12}$$

where:

Rec_i	Overall recovery of mineral i after time t
$Rec_{i,\infty}$	Ultimate recovery after a long time
k_i	Specific flotation rate constant for species i
t	Time after which R_i is measured
$f(k_i)$	Continuous distribution function for flotation rate constants of species i

A typical example of the effect of residence time in the froth can be seen in the following figure, where the overall recovery as a result of first order kinetics can be seen as a function of time for three different froth depths (see section 2.2.7 for effects of froth depth).

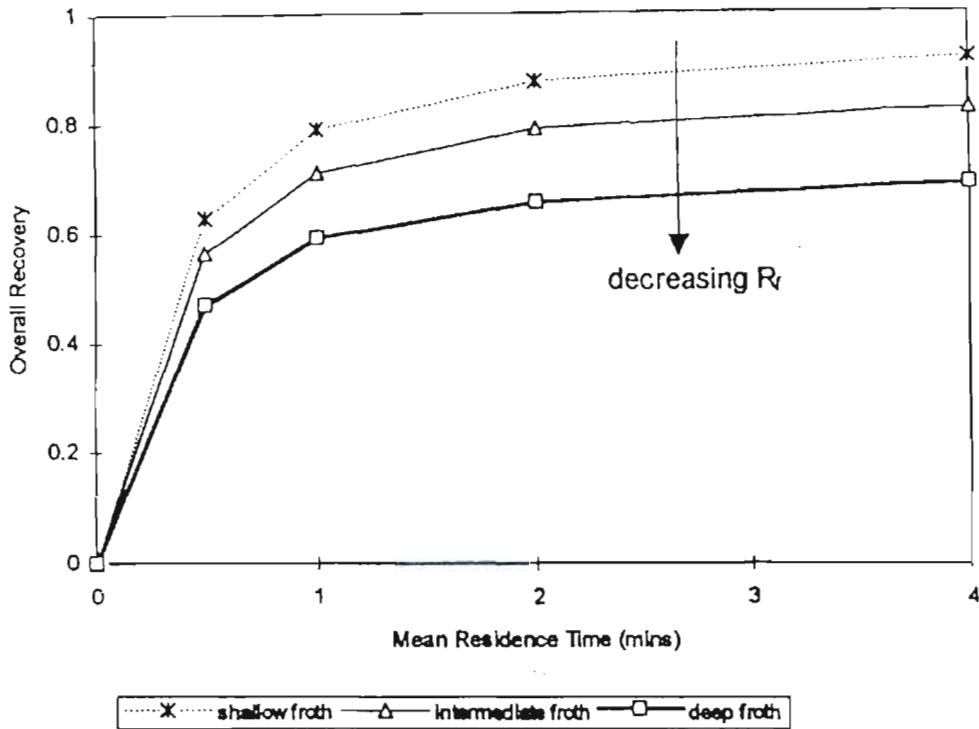


Figure 2-3: The effect of mean residence time on overall recovery (Mathe et al. 1998)

Gorain et al. (1998) examined the effect of froth residence time on the kinetics of flotation. The froth residence time τ_{fg} , as described by the researchers, was the ratio of froth height to superficial gas rate. It was found that the flotation rate constant decreased exponentially with τ_{fg} , but that the relationship is dependent on cell size and can therefore not be used directly for

scale-up. To accommodate this they proposed a specific froth residence time, τ_{fs} , which is independent of cell size:

$$\tau_{fs} = \frac{\tau_{fg}}{L} \quad \text{Eq 2-13}$$

where:

τ_{fs}	Specific froth residence time (time/length)
τ_{fg}	Froth residence time (time)
L	Perpendicular distance from impeller to launder (length)

Researchers were then able to plot the exponential relationship between the flotation rate constant and the specific froth residence time. They then proposed a methodology for predicting the flotation rate constants in a large cell using the k - τ_{fs} relationship together with other relationships.

2.2.4 GANGUE ENTRAINMENT

In the flotation process, particle-bubble collision and attachment accounts for the majority of valuable minerals reporting to the concentrate, however it is not the only mechanism by which material can be transported to the froth. A portion of the hydrophilic material may also be found in the concentrate as a result of entrainment. Entrainment is the transportation of gangue particles in the wake of rising bubbles or as part of composite particles.

Savassi et al. (1998), in their review of gangue entrainment mentioned, the commonly proposed mechanisms used to explain it:

- Entrained material is carried up in the thin layer of water surrounding an air bubble.
- Entrained material is transported in the wake of an ascending air bubble.

They commented that these mechanisms did not account for the water content found in the froths and that the concentration of solids in the bubble wake is approximately the same as that in the pulp. Other researchers suggested that the water is mechanically pushed up in the froth region by ascending swarms of bubbles. Methods of drainage in the froth region, as reviewed by Savassi et al., are:

- Film water drainage: slow drainage of entrained material around the air bubble contours.
- Column drainage or weight induced drainage: fast vertical drainage of entrained material as a result of local froth collapse.
- Mobility induced drainage: drainage as a result of shear forces induced from relative bubble motion.

The principal factors affecting entrainment found in literature reviewed by Savassi et al., 1998 included:

- Recovery of water.
- Particle size.
- Froth structure.
- Mineral specific gravity.
- Froth residence time.
- Pulp dilution or % solids in pulp.

The degree of entrainment can be related to the residence time of the entrained material in the froth (Bishop and White, 1976):

$$CR_i = \frac{1 + \alpha(\rho_s - \rho_{pulp})\lambda_{ent}}{1 + \alpha(\rho_s - \rho_{pulp})\lambda_{ent} \exp(\beta d_i)} \quad \text{Eq 2-14}$$

where:

- | | |
|-----------------|---|
| CR_i | Degree of entrainment on the basis of tailings |
| α, β | Empirical parameters |
| d_i | Particle size (length) |
| ρ_s | Specific gravity of entrained material |
| ρ_{pulp} | Specific gravity of the pulp |
| τ_{fg} | Froth residence time of entrained material (time) |

Savassi et al. (1998) proposed an empirical partition curve to prescribe the relationship between the degree of entrainment and the particle size. Their model has two empirical parameters: the entrainment parameter, which is the particle size at which the degree of entrainment is 20%, and a drainage parameter, which is related to the preferential drainage of coarse particles.

Neethling and Cilliers (2002) noted that predominantly empirical models had been developed to describe entrainment, and attempted to improve the understanding of the entrainment mechanism through the development of a fundamentally based model. Their model considered gas and liquid motion along with solids motion based on the physics of foam structure, and was therefore not specific to flotation froths. The fundamentally based model accurately predicted gangue-water recovery behaviour found experimentally.

2.2.5 BUBBLE-PARTICLE COLLISION, ATTACHMENT AND DETACHMENT

One of the key elements in the effectiveness of the recovery of valuable minerals in the flotation process is the interaction between the particles and the air bubbles in water. Bubble-particle interaction involves a number of sub-processes, which can be divided into collision, attachment and detachment. These three processes are, however, not well understood though much research has recently been carried out in this area (Bloom and Heindel, 2002; Dai et al., 2000; Deglon et al., 1999; Honaker and Ozsever, 2003; Nguyen et al., 1998; Nguyen et al., 2001; Yoon, 2000)

Dai et al. (2000) performed a review of the various literature collision models. They found the collision efficiency of quartz particles with single bubbles using experimental flotation experiments where the attachment and stability were near unity. These collision efficiencies were then compared to the various collision models. They found the best agreement between experimental and calculated efficiencies was obtained from the Generalised Sutherland Equation.

According to Sutherland (1948), all particles lying within a collision radius will collide with the bubble. Sutherland then proposed that the collision efficiency, E_{c-SU} , is given by

$$E_{c-SU} = \frac{3d_p}{d_b} \quad \text{Eq 2-15}$$

where:

d_p Particle diameter

d_b Bubble diameter

Dai et al. (1998) studied the bubble-particle sub-processes and paid special attention to the bubble surface mobility, the fluid flow regime at the bubble surface and the influence of inertial forces on the collision efficiency. The Dukhin or generalised Sutherland equation (GSE model) then follows on from the following equations:

$$\theta_t = \arcsin \left\{ 2\beta \left[(1 + \beta^2)^{1/2} - \beta \right] \right\}^{1/2} \quad \text{Eq 2-16}$$

$$\beta = \frac{2fE_{c-SU}}{9K_3} \quad \text{Eq 2-17}$$

$$K_3 = K \frac{\rho_p - \rho_f}{\rho_p} = \frac{2v_b(\rho_p - \rho_f)r_p^2}{9\eta r_b} \quad \text{Eq 2-18}$$

$$E_{GSE} = E_{c-SU} \cdot \sin^2 \theta_t \cdot \exp \left\{ 3K_3 \left[\cos \theta_t \left(\ln \frac{3}{E_{c-SU}} - 1.8 \right) - \frac{2 + \cos^3 \theta_t - 3 \cos \theta_t}{2E_{c-SU} \sin^2 \theta_t} \right] \right\} \quad \text{Eq 2-19}$$

where:

θ_t Angle of tangency

f Factor of 2 for free non-retarded bubble surface

K Stokes number

Some clarification is required for the θ_t term. The angle of tangency is defined as the angle, which is formed between the intersection of the tangential component of the particle's grazing trajectory and its initial trajectory with respect to the centre line through the bubble parallel to the initial particle trajectory.

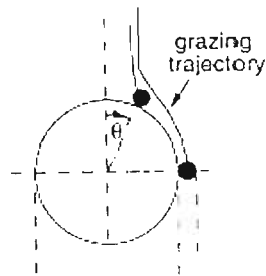


Figure 2-4: A figure representing the definition of the angle of tangency θ_t (after Dai et al 2000)

There are many other models that take into account the gravitational effect, inertial effect, negative effect of centrifugal forces and the interceptional effect, such as the Schulze model (Schulze, 1989).

Bloom and Heindel (2001) found new analytical expressions for both the collision and bubble attachment frequency for use with flotation separation models. Their expression for collision frequency takes into account the particle settling velocity and the bubble rise velocity. They showed that the inclusion of the particle settling velocity increases the collision frequency by a factor of 1.5 and that the most significant factor affecting the collision frequency is that of bubble radius.

Nguyen et al. (1998) developed a basis for experimental determination of bubble-particle attachment probability in single bubble flotation experiments. They showed that attachment probability models based on sliding time and induction time cannot adequately describe the elementary steps in the bubble particle attachment microprocess during flotation. They demonstrated the necessity for the elementary step of, 'rupture of liquid film and formation of three-phase contact of a critical radius (TPC nuclei)'. It was then proposed that the probability of recovery due to particle attachment can be described by the product of three elementary steps, the thinning of intervening film to a critical film thickness, rupture of the intervening liquid film and the formation of TPC nuclei with the expansion of the three-phase contact line to form a stable wetting perimeter.

$$P_{at} = P_t P_r P_{TPC} \quad \text{Eq 2-20}$$

where:

P_{at}	Probability of attachment
P_t	Probability of the thinning of the film to critical thickness
P_r	Probability of the elementary step “rupture of the film”
P_{TPC}	Probability of the formation of the three-phase contact line

Stechemesser and Nguyen (1999) investigated the gas-solid-liquid phase contact (TPC) in theoretical bubble-particle attachment. A theoretical model for the time of the TPC expansion was proposed and compared to experimental data. They showed that TPC kinetics are strongly controlled by line tension and particle hydrophobicity.

Nguyen et al. (2001) made use of a high-speed CCD video camera to measure the particle trajectories on the surface of an air bubble. The rupturing of the intervening water film between the hydrophobic particle and the air bubble surfaces with the formation of the three-phase contact angle was observed. They concluded that the available theory does not adequately describe the bubble-particle attachment interaction and discussed possible corrections to the available attachment models.

Yoon (2000) reviewed collision models put forward over the last decade and found that the probability of collision for bubbles and particles is dependant on the bubble size and therefore on the Reynolds number. A generalised form of the probability of collision model is then:

$$P_c = A \left(\frac{D_p}{D_b} \right)^n \quad \text{Eq 2-21}$$

where A and n are the parameters which change with the Reynolds number. The values of A and n are given in the table below for some published models.

Table 2-1: Values of A and n for different collision probability models

Flow Conditions	A	n
Gaudin, 1957	$\frac{2}{3}$	1
Sutherland, 1948	3	1

Yoon and Luttrell, 1989	$\left[\frac{3}{2} + \frac{4 \text{Re}^{0.72}}{15} \right]$	2
Weber and Paddock, 1983	$\frac{3}{2} \left[1 + \frac{\frac{3}{16} \text{Re}}{1 + 0.249 \text{Re}^{0.56}} \right]$	1

2.2.6 CHEMICAL FACTORS

The mechanism of flotation is dependent upon particles being hydrophobic, and chemical reagents are required to obtain these conditions. The chemical environment of the flotation cell is therefore paramount to achieve separation. Every ore has a specific chemical requirement and therefore the effects of reagents are very specific, and hence difficult to model. A general overview, from Wills (1997), of the chemical environment in flotation is to follow.

2.2.6.1 Collectors

According to a mineral's surface characteristics, it can be classified as either a non-polar or polar type. This polarity affects the behaviour of the mineral due to molecular effects experienced in the presence of collector chemicals. Collectors are organic compounds which render selected materials water-repellent by adsorption of molecules or ions on to the surface of minerals, reducing the hydrated layer stability and allowing the attachment of the particle to a bubble on contact.

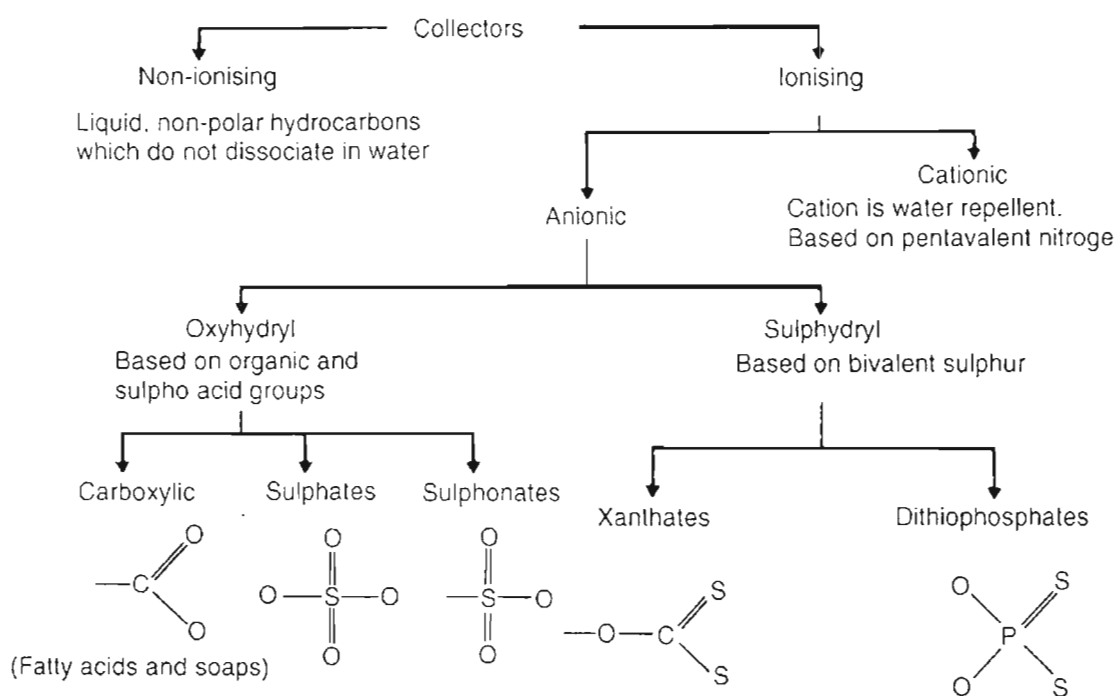


Figure 2-5: Classification of collectors (Wills, 1997)

2.2.6.2 Frothers and Regulators

Once particles have been rendered hydrophobic and attach to a bubble, the stability of the bubble attachment must be considered if the particle is to report to the concentrate. Frothers are responsible for this bubble stability requirement. Good frothers must produce a froth that is stable enough to facilitate transfer of floated material from the cell surface to the collecting launder. Selection of suitable frothers for a particular ore can only be done after extensive laboratory test work as interactions between frother, mineral and other reagents does occur.

Regulators are used in flotation to modify the action of the collector's water-repellent effect on the mineral surface. Regulators are classed as activators, depressants, or pH modifiers.

Activators alter the chemical nature of the mineral surface so that they become hydrophobic due to the action of the collector. Depression is used to increase the selectivity of flotation by rendering certain minerals hydrophilic and thus preventing their flotation.

2.2.6.3 Importance of pH

pH plays an important role in flotation. Selectivity in complex separations is dependant on a balance between reagents and pH. Flotation, where possible, is carried out in an alkaline medium to keep corrosion of cells and piping to a minimum and many collectors are stable under these conditions.

Molecular modelling of reagents for flotation were carried out by Leal Filho et al. (2000) with the use of a Silicon Graphics 5.1 IRIX operational system. Systems such as these are required to perform the computationally intensive modelling procedure. They modelled mineral surfaces and molecules of two polysaccharides, which were candidates as depressing agents for the minerals. The researchers used a “total fitting parameter” to assess the steric compatibility between the minerals and the polysaccharide species.

This type of modelling is, however, very specific to the species of mineral and reagent considered and does not create a basis on which other reagents and mineral interactions can be modelled.

Ekmekci et al. (2003) conducted research into the effect of frother type and froth height on the flotation behaviour of chromite in UG2 ore which is more similar to the ore processed in the LONMIN Eastern Platinum B-Stream.

2.2.7 FROTH DEPTH

Froth depth has been recognised as an important aspect of flotation in the determination of flotation performance. With the advent of computers and their use in the automatic control of flotation circuits, pulp level and froth depth have been used as a computer control variable. Many of these control strategies are based on empirical schemes.

Feteris et al. (1987) combined the two-phase kinetic model and the probability model to derive a new relationship between the overall flotation rate coefficient and the probability that a particle-bubble aggregate would survive the cleaning action of the froth. The model was parameterised through batch tests using galena, and the relationship used to investigate the effects of varying froth depth.

$$k_f = \frac{M_{fpulp}}{(V_p M_{pulp})} \quad \text{Eq 2-22}$$

$$k = k_f P_f \quad \text{Eq 2-23}$$

where:

M_{fpulp} Mass flow rate of species from pulp into froth (mass/time)

V_p Pulp volume(volume)

- M_{pulp} Mass of species per unit volume in the pulp(mass/volume)
- P_f Probability that particle survives the froth

They found a linear relationship between the probability that a particle will survive the cleaning action of the froth and the depth of the froth phase.

Vera et al. (2002) showed that when the froth depth is zero, the collection rate, k_c , of the pulp zone may be determined by extrapolation.

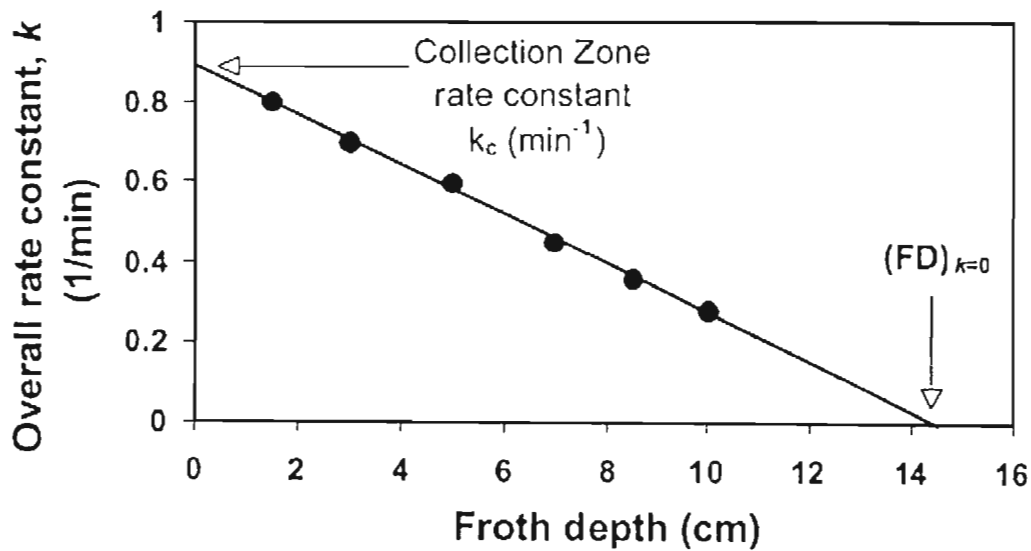


Figure 2-6: Flotation zone rate constant (k) as a function of froth depth (FD) (Vera et al., 2002)

The froth depth (FD) at which no collection is possible (when the froth is very deep) is then the intercept of the straight line with the X-axis. This leads to

$$k = k_c \left\{ 1 - \frac{(FD)}{(FD)_{k=0}} \right\} \tag{Eq 2-24}$$

From this relationship they extracted a froth zone recovery (R_f) and in turn related it to the froth retention time, defined as the ratio of froth volume to the volumetric flow rate of concentrate from the cell.

Moys (1978) developed a counter-current plug-flow model to describe equilibrium froth behaviour, given by the equation:

$$\frac{dM_i(h)}{dh} = \left(\frac{M_{f,i}(h) + M_{e,i}(h)}{v(h)} + \frac{M_{r,i}(h)}{u(h)} \right) \quad \text{Eq 2-25}$$

where:

$M_i(h)$	The mass of mineral i at froth height h (mass/time)
$M_{e,i}(h)$	Entrained mass flow rate at froth height h (mass/time)
$M_{f,i}(h)$	Mass flow rate of species i attached to bubbles at froth height h (mass/time)
$M_{r,i}(h)$	Mass flow rate of particles diffusing from the upward-flowing stream to the downward-flowing stream (mass/time)
$u(h)$	Velocity of the downward-flowing stream (length/time)
$v(h)$	Velocity of the bubble films (length/time)

The mass fraction of mineral can then be obtained from the following expression:

$$x_i(h) = \frac{M_i(h)}{\sum_j M_j(h)} \quad \text{Eq 2-26}$$

These equations are based on the following postulates as described by Mathe et al. (1998):

- The rate of detachment of particles of component i is proportional to their concentration within the froth.
- The net upward flow rate of water is constant and equal to the concentrate water flow rate.
- The solid particles diffusing from the upward-flowing stream enter the downward-flowing stream at velocity u and rate $M_{r,i}(h)$.
- Both floatable and entrained particles rise at the velocity of the bubble films, $v(h)$, towards the top of the froth.

From the above models it can be seen that froth characteristics change with froth depth and have been modelled as such to obtain the mass fraction of mineral remaining in the froth at a particular height.

2.2.8 PARTICLE SIZE

Particle size has the most significance to the flotation process when considering composite minerals. Valuable minerals exist as fine grains within the rock structure and must therefore be liberated. Liberation is achieved through the grinding of the ore to such an extent that the majority of associated particles are comprised of the valuable grains of mineral. As a result, particle size plays a major role in determining the amount of gangue material that is retained with the valuable mineral after flotation. Particle size also affects the probability of particle-bubble collision as discussed in section 2.2.5, as well as the entrainment of gangue particles (see section 2.2.4). The suspension characteristics of particles also change with particle size and can therefore affect the ability of the pulp to keep particles in suspension. Large particles may also be difficult to lift to the froth due to their increased mass and size. From this it can be concluded that flotation properties are affected by particle size.

The effects of particle size on kinetic rate models developed by many researchers can be seen in the earlier sections (as mentioned above). Trahar and Warren (1976) concluded the following regarding the effects of particle size on flotation:

- Almost all minerals exhibit an optimum particle size for flotation. The size range is generally between 10-100 microns and is an intermediate range of most particles entering flotation circuits.
- Fine particles generally float slower than intermediate sizes.
- Fines recovery is poor and is attributed to the lower rate of collision and attachment.
- Recovery drops rapidly when particle size is above the intermediate range, which suggests there is a maximum particle size for flotation.

Flint and Howarth (1971) reduced the equations of motion for particles to a dimensionless form by introducing an inertial parameter, K , and a particle gravity parameter, G . They found that there exists a critical bubble-particle parameter, K_c , below which no collisions can occur. This showed that there exists a smallest size particle that could collide with a particular sized bubble. The smaller particles were thus unable to be floated as they essentially follow the flow streamlines in the pulp and were therefore washed aside by bubbles without a collision occurring.

van Deventer et al. (2002) worked on the dynamic behaviour of coarse particles in flotation froths. They studied the effect of various physical parameters in the rupture of bubble films in two-phase foams. Their work considered mineral particles not to be spheres and accounted for this by using an equivalent diameter, determined using the following expressions:

$$v_p = \phi_v \frac{\pi}{6} D_1^3 \quad \text{Eq 2-27}$$

$$A_p = \phi_a \pi D_1^2 \quad \text{Eq 2-28}$$

$$\lambda = \frac{\phi_a}{\phi_v} \quad \text{Eq 2-29}$$

$$D_e = \frac{D_1}{\phi_\lambda} \quad \text{Eq 2-30}$$

where:

D_e	Equivalent diameter of particle (length)
D_1	Second longest dimension of the particle (length)
v_p	Volume of particle (volume)
A_p	Surface area of particle (area)
ϕ_v	Volume factor (= 1 for a sphere) (dimensionless)
ϕ_a	Area factor (= 1 for a sphere) (dimensionless)
ϕ_λ	Shape factor (dimensionless)

Their work involved modelling the trajectory of particles over discrete time events. They accomplished this by calculating bubble flow streamlines and modelling bubble size, thickness of bubble films, air hold-up and bubble velocity at any point on the streamline.

2.2.9 AIR ADDITION RATE

The flotation process is directly affected by the air supplied to it, as it is dependent upon particle-bubble attachment to achieve separation. It is therefore imperative that flotation cells have an ample air supply. As was discussed in section 2.2.1, the general form of the kinetic rate equation for flotation (equation 2-9) is applicable when air is not a rate-limiting factor.

However, air dispersion in the pulp will have an effect on flotation characteristics and is itself affected by the rate of air addition, impeller design and the surface tension in the pulp. Air dispersion has the following important properties that affect flotation:

- Bubble surface area flux, S_B , which affects the flotation rate constant.
- Bubble diameter which has an effect on the collision efficiency of particles and bubbles.
- Air hold-up in the pulp. Air hold-up is the volume of space in the cell that is occupied by air bubbles and consequently reduces the available volume for pulp.

The characterisation of gas dispersion is achieved through the following expressions:

$$S_B = \frac{6V_G}{d_{bubble}} \quad \text{Eq 2-31}$$

$$V_{bubble} = \frac{V_G}{\varphi} \quad \text{Eq 2-32}$$

$$V_G = \frac{Q_G}{A_{cell}} \quad \text{Eq 2-33}$$

where:

S_B	Bubble surface area flux (area/area.time)
V_G	Superficial gas velocity (length/time)
d_{bubble}	Sauter mean bubble diameter (length)
φ	Air hold-up (volume/volume)
V_{bubble}	Average vertical velocity of the bubbles (length/time)
Q_G	Volumetric flow rate of air into the cell (volume/time)
A_{cell}	Area of the base of the cell (area)

Gorain et al. (1997) worked on the effect of impeller type, impeller speed and air flow rate on flotation performance. They found that the rate constant and bubble size showed no relationship for the operating conditions that were investigated. However, it has been shown in earlier sections (2.2.5) that the bubble size does have an effect on collision efficiency.

It was also found that there was little or no correlation between the rate constant and gas hold-up for their experimental data, but that the rate constant relates better to gas hold-up than to bubble size. The superficial gas velocity showed a much better correlation to the rate constant than both of the above and they found a trend of increasing flotation rate with increasing superficial gas velocity at low air flow rates.

Finally their work showed that there is a definite linear relationship between the flotation rate constant and bubble surface area flux. From their graphs it was concluded that the confidence with which flotation rate can be related to bubble surface area flux is much greater than compared to bubble size, air hold-up or superficial gas velocity individually. Other studies have also found that there is a near-linear relationship between the bubble surface area flux and the flotation rate constant under certain conditions (Deglon et al., 1999).

3 FLOTATION CIRCUIT DESIGN, CONTROL AND OPTIMISATION

In the operation of flotation circuits there are a number of difficulties faced when designing the circuits, controlling them and trying to find the optimum operating conditions for the plant. Poorly designed flotation circuits, under poor control strategies, invariably lead to low plant performance and hence a loss in revenue. As a result, it is required at the outset of the circuit design to optimise the plant performance and this is considered in the following chapter.

3.1 FLOTATION CIRCUIT DESIGN

In earlier sections it was mentioned that the efficiency of the flotation process is very poor and therefore more than one flotation stage is required to achieve any appreciable separation. This is where the design of flotation circuits plays a major role, in that the dynamic behaviour of the floatable species must be understood in such a way as to facilitate the successful design of the circuit to achieve maximum recovery of the valuable material.

Flotation is a rate process and as such is time dependant. The specific separation of valuables at a given plant throughput is therefore an important consideration when designing a flotation circuit. Certain factors are critical when considering the design of a flotation circuit:

- The volume of the flotation cells. The volume determines the residence time and therefore the separation achieved per cell at a specific throughput.
- The configuration of recycles and flows between the cells in order to recover slow floating valuable minerals.
- The number of cells required to achieve a specific concentrate grade.

Flotation circuits generally have three specific stages, which each perform specific duties in the overall recovery process, known as roughers, scavengers and cleaner stages respectively. Below is a basic diagram showing the general function of each of the three stages:

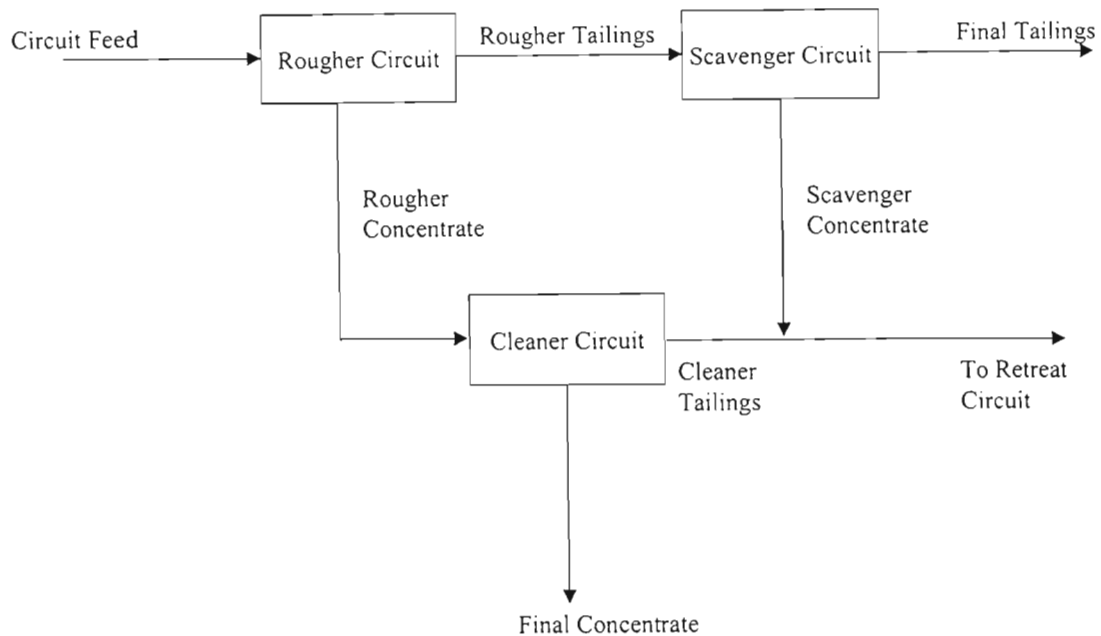


Figure 3-1: A typical flotation circuit layout (Schroder,1999).

The roughers provide the initial separation of valuables from the bulk of the material and as such must be of a sufficient size to accommodate the maximum throughput of the plant. Due to the chemical reagent additions, the roughers will ultimately remove most of the valuable material as it will be the fastest floating and hence have the highest flotation rate constant. As such, the roughers perform a first pass at recovering the valuables and provide the cleaners with an already floated material high in valuable minerals. The cleaners provide a much more selective separation as their material throughput is much lower, and are therefore responsible for maintaining a high concentrate grade. The scavengers, as their name implies, are responsible for recovering any valuables that escaped the roughers. Each of these stages has a specific role to play and as such requires different operating philosophies.

Cells are usually grouped together to form flotation banks, consisting of a number of cells. The flotation banks are tiered in such a way that the tailings from one bank flows to the next under gravity, therefore interaction between the pulp levels in adjacent cells is common and disturbances can carry over to other banks downstream. Concentrates collected from the launders usually flow under gravity down to floor sumps at ground level, before they are pumped up to the head of the next bank. These sumps can act as buffers to small fluctuations in the concentrate composition and flow rate.

Another common circuit layout is the counter-current circuit. Shown below is a schematic diagram of a counter-current flotation circuit.

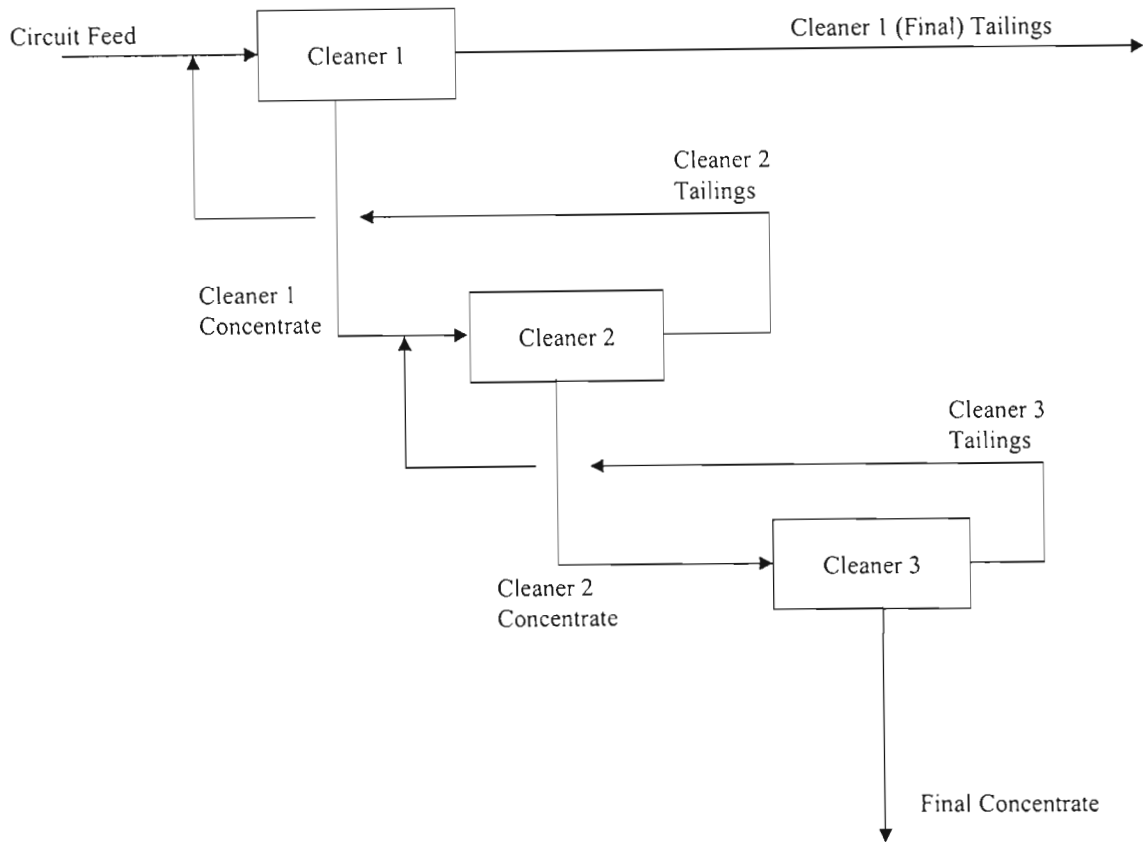


Figure 3-2: Counter current circuit design (Schroder, 1999)

As can be seen from the diagram, this layout encourages greater separation as the flow of material passes through a number of cells repeatedly, with the opportunity of flotation greatly increased. Unfortunately, these circuit layouts promote large re-circulating loads as can be intuitively seen from the diagram.

The circuit considered in this thesis is shown below.

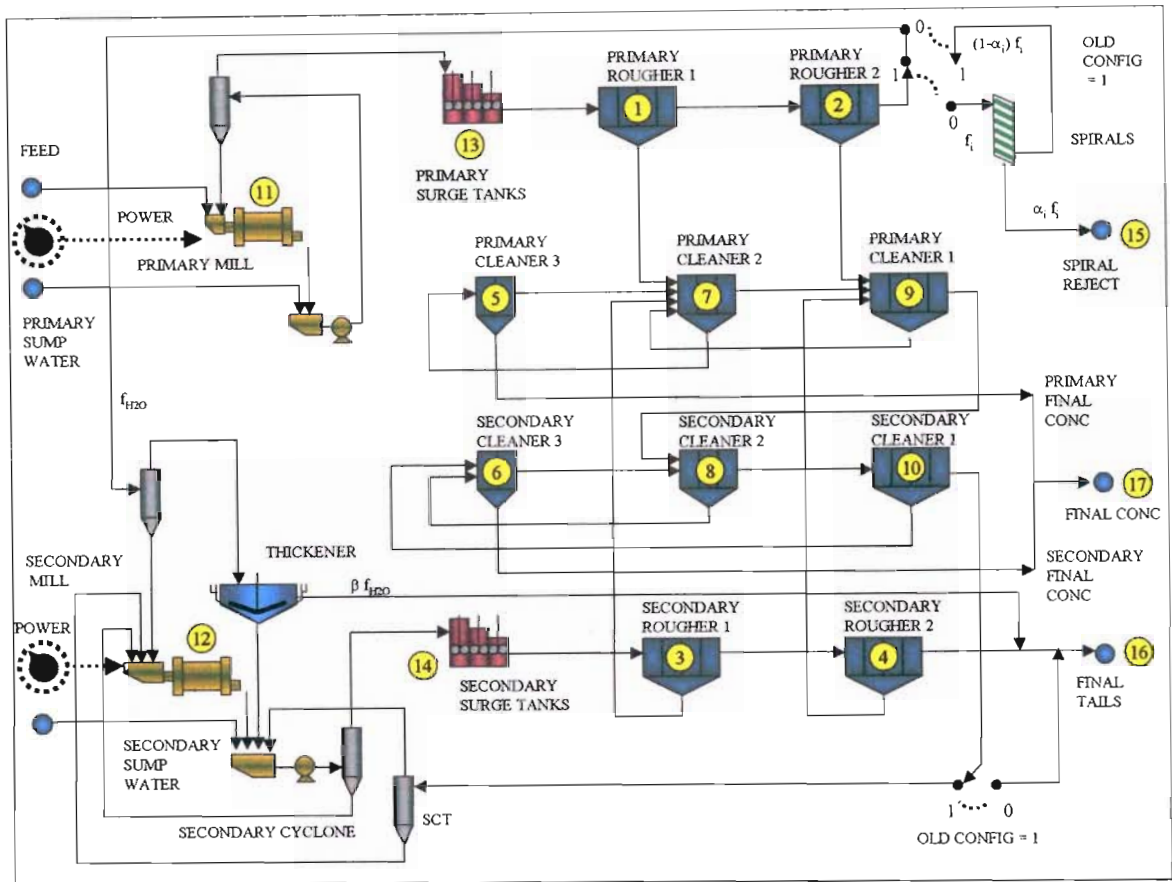


Figure 3-3: The B-Stream Circuit Layout at LONMIN Eastern Platinum

As can be seen above, the primary rougher tailings are sent to a secondary mill where they are re-ground so as to ensure that no valuable minerals are lost through complex mineral structures. Counter-current design is used, along with a complete secondary circuit for the processing of re-ground material. The final tailings leave only through the last secondary rougher and as a result there is a large mass hold-up in the circuit. Good separation is expected due to the large number of cells through which the material must pass.

From the diagram above it is clear that there are a number of switches, allowing for different configurations to be available to the plant engineers if they feel it necessary to change the flows around the plant circuit.

Circuit flexibility is necessary so as to provide for small fluctuations in the flowrate of ore to the plant and the grade of the incoming ore. A simple way to provide a smooth flow to the

plant is to use the storage agitator tanks, as seen in figure 3-3. Any minor variations are smoothed out in these tanks and the conditioned pulp is then pumped to the flotation cells at a controlled rate.

Loveday and Brouchaert (1995) carried out a study of flotation design principles comparing the recovery for two and three stage circuits. They concluded that:

- The separation achieved in a single well-mixed cell is poor and the conventional practice of using several cells in series for each stage of flotation is justified.
- Increasing circulation can increase the sharpness of separation.
- Rougher and scavenger stages are significantly larger than cleaner stages for ore with a small mass of floatable material. Also, scavengers may be inappropriate for these ores.
- Separation is significantly improved by increasing the number of stages.

In their study the assumption was made that the floatability of all particles remains constant in all stages, whereas in practice the dynamic changes in the chemical environment and circulating loads, may influence effectiveness.

3.2 OPTIMISATION OF CIRCUIT DESIGNS

The design of flotation circuits plays an important role in the recovery the plant is able to achieve and the economic factors within the plant. With an understanding of the flotation process and the characteristics of the material to be floated, a circuit optimisation can be carried out in order to determine the circuit layout that would give the best results. There are two approaches to optimisation of circuits, parameter optimisation and structural optimisation.

Yingling (1993), in his review of parameter and configuration optimisation of flotation circuits, described a process by which a flowsheet superstructure is formulated which embeds many alternative configurations for the plant. In the superstructure, a mixture of discrete and continuous decision variables represents the configuration alternatives for the circuit, for example the existence or non-existence of a bank of cells, bank interconnections, stream splits and number of cells per bank. Continuous decision variables are used to represent alternative steady state specifications for the individual flotation cells and the formulation might include certain requirements that the plant must meet, such as grade. An objective function is thus devised based on the rate performance of the alternative systems embedded in the superstructure, and search procedures are used to identify the values of the decision variables, such as flowsheet design.

Specific examples of flowsheet design optimisation found in the literature include the work by Abu-Ali and Sabour (2003) in which they developed a model to determine the optimum number of cells in a flotation bank as well as their respective volume in order to achieve a required target recovery. Their model is based on comparing the marginal cell effects on both the capital and present operating costs.

3.3 CONTROL OF FLOTATION CIRCUITS

Ideally the control of flotation circuits would be fully automated, but due to the complexity of the flotation process, as described earlier, it is an almost insurmountable task. Therefore, in the short term, the best plant performance and efficiency is achieved by a combination of plant control and conscientious operators who are able to adjust set-points and their limits. The advantage of computer-aided control is that it is ever watchful and present, unaffected by human activities, such as shift changes and breaks etc.

In the analysis of the control of flotation circuits it was found that there are certain process variables that are more commonly used in control and can be divided into the following general groups (Schroder, 1999):

- **Controlled variables:** These variables obey constraints so as to maintain plant performance.
- **Measured variables:** Very few variables are available on-line for control purposes.
- **Disturbance variables:** These are inputs to the flotation system that vary and affect the system but which the operators have no control over.
- **Manipulated variables:** These are the inputs to the system, which the plant operators or controllers have access to and can manipulate.

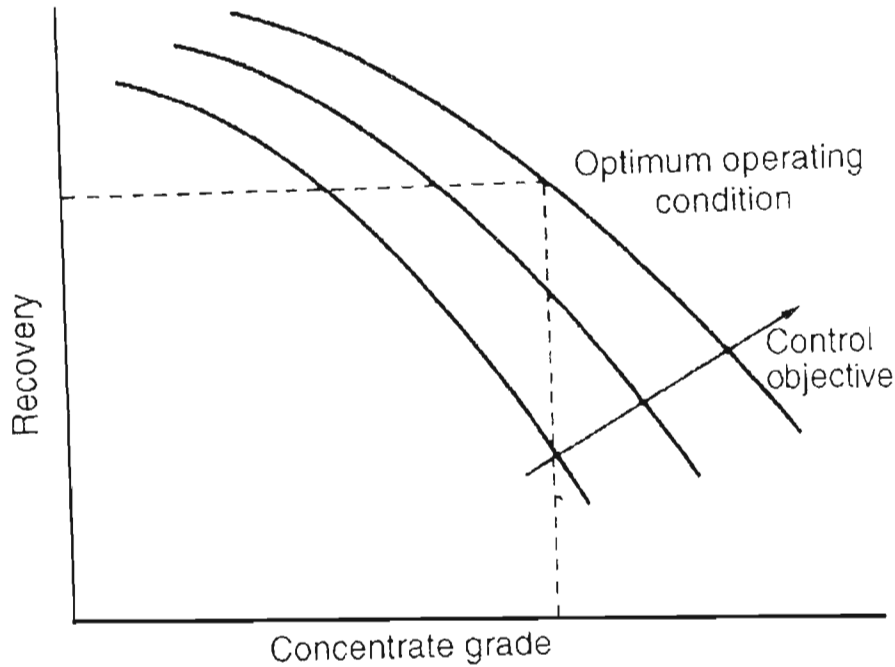


Figure 3-4: Flotation control objective (Wills, 1997)

Tables of the typical variables mentioned above and comments on their uses can be found in Appendix B, as reviewed by Schroder (1999).

The variables controlled are pulp levels, grades, reagent addition rate, bank tailings valves, water and air additions and pump speeds.

Few measurements are available on-line but sump and pulp levels along with pH are readily available. Depending upon the type of material being floated and the technology levels in use at the plants, stream compositions, flowrates and density can be made available, through the use of very expensive equipment.

Due to this lack of online data the internal flowrates and compositions of flotation circuits as well as flotation performance are unknown and hence create difficulties when attempting to control the circuits.

3.3.1 BASIC CONTROL STRATEGIES

Operators on the plant have experience as to which conditions in their plant are conducive to good recovery and performance, and as such know when the plant has moved away from

these conditions and must be brought back to the optimum operating point. To aid in achieving this, there are basic control strategies in circuits for the control of variables such as air addition rate, sump and pulp levels and reagent addition rates. This control is usually in the form of proportional, integral and derivative (PID) controllers. PID controllers facilitate easy set-point achievement without continuous input from operators. The operators simply notice that a change is required, and from experience gauge what the new operating point should be and set the set-point to this value and can then leave the unit operation manipulation to the controller. Due to the inherent instabilities such as interactions between cells and disturbance propagation as well as the non-linearity of flotation, PID control has limited use in controlling the flotation process, leading to the development of more sophisticated types of control for flotation, known as advanced control strategies.

3.3.2 ADVANCED CONTROL STRATEGIES

There are a number of advanced control strategies and those with the most relevance and applicability to flotation will be reviewed in the following section.

3.3.2.1 Feed-Forward Control

The basic idea behind feed-forward control is to detect disturbances as they enter the process, then to make adjustments to manipulated variables so that the output variables are held constant. As a result, there is no delay while waiting for an error signal after the disturbance has passed through the process as immediate action is taken to compensate for the disturbance as it enters the process. Feed-forward control is widely applicable to chemical engineering processes and can accommodate linear, non-linear and multivariable systems along with lag and dead-time. The disadvantages of feedforward control are:

- The disturbances must be detected. In order to apply feedforward control there must be an available measurement of the variables.
- Knowledge is required of how the disturbance and manipulated variables affects the process.

This provides a limitation as far as the flotation process is concerned as there are few on-line measurements, but through the use of estimators this type of control may be possible. The use of feedforward control is commonplace with ratio control of the reagent additions based on flowrate and the composition of the feed (Luyben, 1990).

3.3.2.2 Adaptive Control

Adaptive controllers continuously identify the parameters of the process as they change and retune the controller appropriately. This on-line adaptation is fairly complex and can sometimes lead to poor performance. There can be delays whilst the on-line identification is being achieved, which could result in the plant moving to another condition before the tuning has been applied, in which case it has to be repeated. On-line identification can sometimes be avoided by running off-line tests for all possible conditions of the plant and determining controller tunings at all these possible conditions, then as the process moves from one set of conditions to another, the controller settings are automatically changed. This type of control is known as openloop-adaptive control or gain scheduling (Luyben, 1990).

3.3.2.3 Multivariable Control

Multivariable controllers generally tend to minimise some quadratic performance index, however many chemical engineering systems are too high-order or non-linear for successful linear quadratic methods. Advances in this area have resulted in two popular multivariable controllers, namely, Dynamic Matrix Control (DMC) and Internal Model Control (IMC) methods. Applications of multivariable control to the flotation process should result in improvements in performance, but examples were not found (Luyben, 1990).

3.3.2.4 State Estimators

As mentioned earlier, there are few available on-line measurements in flotation circuits. These unmeasurable variables, if obtained, could provide an improvement in the control of the process. In order to determine these unmeasurable variables, an estimator might be used, such as a Kalman filter. Kalman filter estimators utilise on-line fitted models using available measurements with the model then providing information about the process that is not measurable. Kalman filter estimates of measurements contain less random error and therefore calculations made with these measurements are more accurate and can even replace existing measurements should the instrumentation fail. By building such “well-informed” models, responses to disturbances can be obtained that allow for “optimal” control performances (Herbst et al., 1992).

Herbst et al. (1992) discussed four examples of model-based control strategies, two of which used Kalman filter estimates to provide variables that were not measurable on-line.

3.3.2.5 Expert Systems

An expert system makes use of the knowledge and experience that operators obtain while working on flotation plants. The operator's knowledge is "captured" in the form of a hierarchal knowledge structure. Given a certain set of conditions in the plant that the operator has prior knowledge about and experience in dealing with, the knowledge structure will check the states of the system against a set of rules in its structure, and if the conditions are within those set by the knowledge structure, the expert system performs the necessary steps to change the process in a similar fashion to the steps taken by an operator given the same situation.

Edwards and Mular (1992) used an expert system for real-time control of the copper flotation circuit at Brenda Mines Ltd. The supervisory control was able to achieve operating levels superior to those attained by the manual control of operators.

4 DYNAMIC OBSERVER MODEL (DOM) DEVELOPMENT

The dynamic observer model was developed by Professor Mike Mulholland (School of Chemical Engineering, University of Natal, Durban, mulholland@ukzn.ac.za) for the Department of Arts, Culture, Science and Technology (DACST) Innovation project in order to fulfil the requirements as set forward by MINTEK (200 Hans Strijdom Drive, Randburg, 2194, South Africa). What follows is an overview of the modelling of the plant, the solution technique used, and some of the features of the model. The author's contribution was to translate this model into a real-time format under PlantStar.

4.1 OBJECTIVES OF THE DYNAMIC OBSERVER MODEL

MINTEK and the DACST innovations project set out a guideline for the requirements of the complex model. The model developed attempted to meet the following requirements:

4.1.1 EXPLAIN PLANT BEHAVIOUR

The primary objective as proposed by MINTEK to DACST was to explain plant behaviour. The complex plant model should be adaptive, inferring certain parameters that cannot be measured and hence providing MINTEK with the information they require to explain both the dynamics of the control system and the plant online through an intelligent expert system (IES). It was envisaged that linear models derived from the complex model would be able to do most of the explanation work, but the complex model should explain certain things that these simplified linear models could not.

4.1.2 USED FOR SIMULATION AND PLAYBACK

The complex model needed to provide an accurate representation of the plant, which could be used for:

- Playback on PlantStar (PlantStar is the control software to be used to implement the dynamic observer model on-line, while playback refers to passing historical data to the model instead of real plant data)

- Training purposes
- Test work, research and development
- Derivation of linear models

4.1.3 PROVIDE A BETTER INSIGHT INTO THE PROCESS

In order to optimise processes, a good understanding of the underlying dynamics is required.

In addition, a good model of the process is necessary to provide good control.

4.2 REQUIREMENTS TO MEET OBJECTIVES

- The model should be able to run alongside the process and continuously adapt to any changes being made in the process i.e. An “IDENTIFICATION MODE”. During this mode, new k -values are estimated continuously as new measurements become available.
- The model should be able to simulate the process, predicting the outputs based on any given inputs. This should be done independently of the real process i.e. “PREDICTION MODE”. No adaptation of the model is required during this stage.
- The model must predict output responses to input step changes. These responses could then be used to fit the simplified linear models.
- The model should support a playback feature, allowing the operator to go back to any stage in the past, make changes to the conditions at that stage and then simulate ahead from that point to any point in the future. The support for playback will, however, be provided within the PlantStar environment.
- The model should be configurable for any circuit configuration.

4.3 MODEL STRUCTURE

The complex model was developed specifically for LONMIN's Eastern platinum B-stream as depicted in figure 3.3, but will be repeated here for completeness.

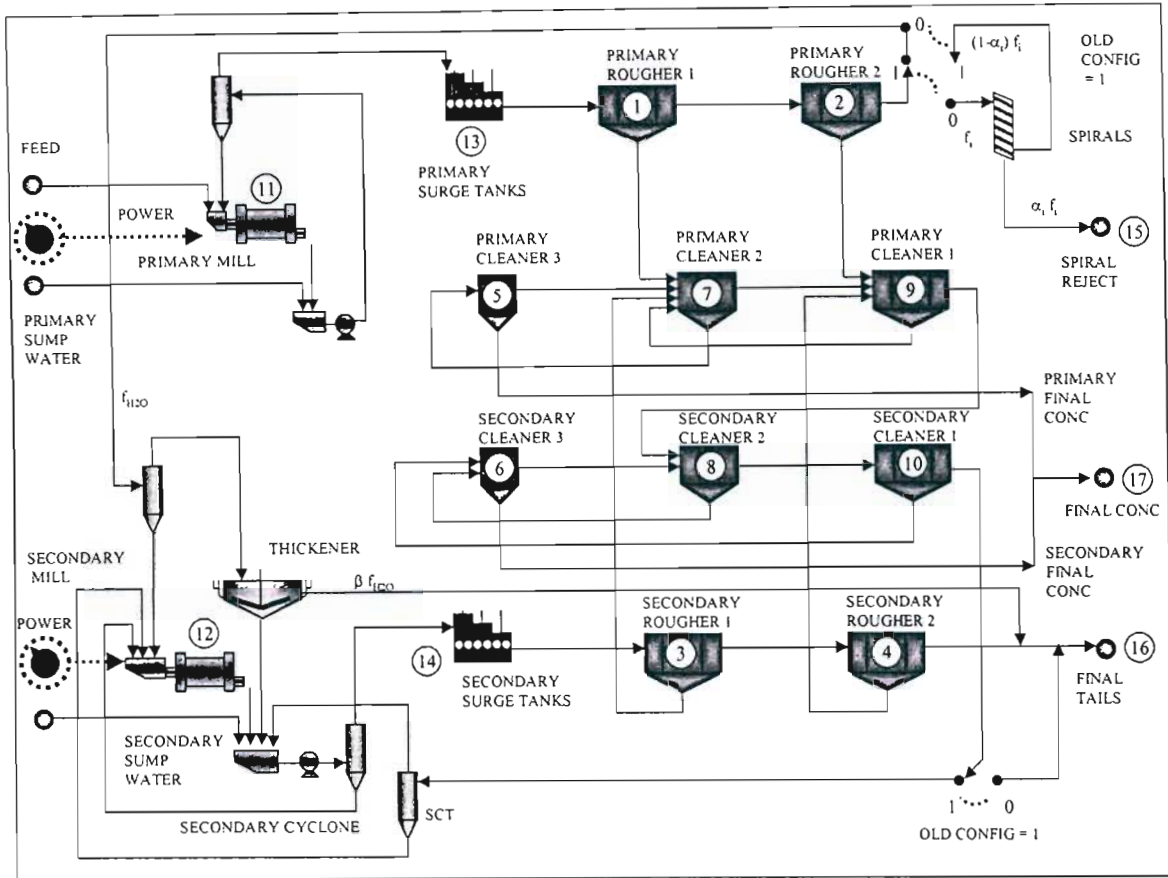


Figure 4-1: LONMIN's Eastern platinum B-Stream circuit configuration

The creation of the dynamic model had to ensure that a general structure was used, to ensure that easy conversion to other plant flowsheets could be accomplished. This particular type of flowsheet modelling of processing plants inevitably leads to a set of Differential and Algebraic Equations (DAE), which must be solved simultaneously. The algebraic equation set usually describes the connections of streams at nodes in the flowsheet, whilst the differential equation set describes the cell accumulations based on the equations (Equation 2-9) as discussed in section 2.2.1. Although a solution of this set of equations by simply substituting

the algebraic equations into the differential equations might be feasible, it would reduce readability and lead to problems when reconfiguring the circuit.

To facilitate the complex modelling of the process, three types of flowsheet elements were identified:

- Flotation Banks: Combinations of cells in series with one inflow and two outflows.
- Mills and Tanks: One inflow and one outflow.
- Product Streams: Results from the combinations of several streams.

These elements become the nodes of the network. Each element has a single input made up of the combination of outflows from its respective up-stream elements. The feed to the entire process is obviously set as the feed conditions and pass on to the first element.

The flotation banks have been treated as a single large flotation cell, which is a small approximation to avoid a large increase in the number of states of the system. What follows is a comparison between the unit step responses for a single cell and a bank of 5 cells.

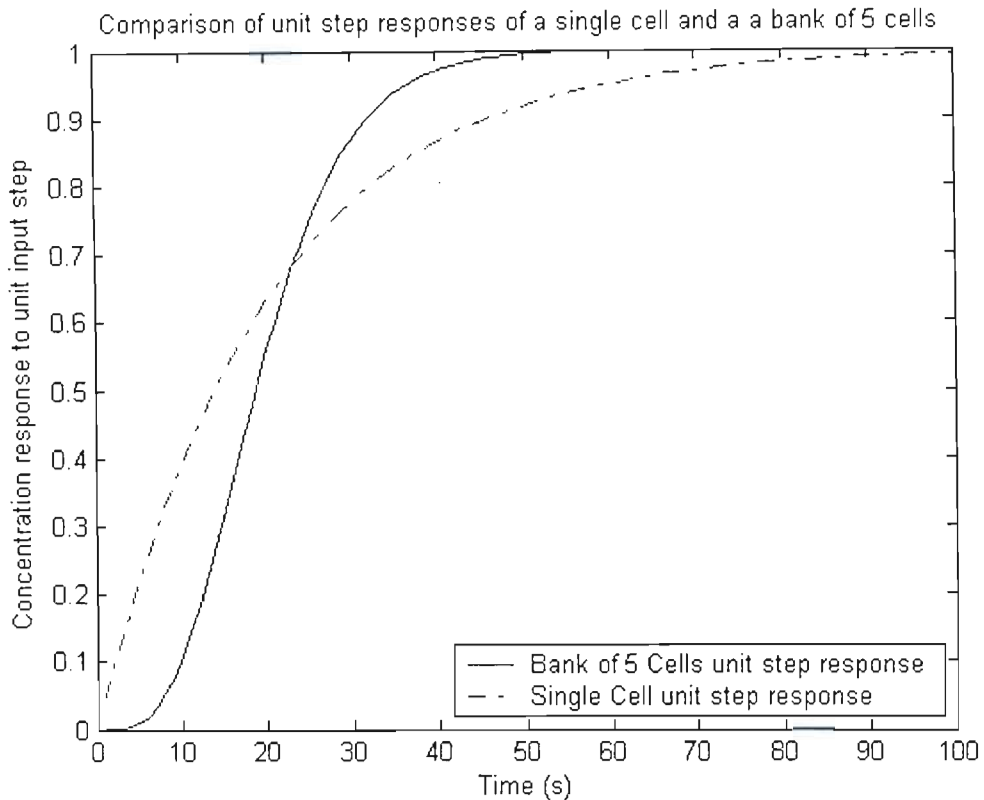


Figure 4-2: The comparison between a single cell's unit step response and that of a bank of 5 cells.

From figure 4-2 it can be noted that there is a difference in the step responses of the two scenarios as would be expected. The first order response of the single cell shows that initially it responds quicker than the bank of cells but the bank of cells approaches the final step position sooner than the single cell. The residence time used for the illustration is a fifth of the total time to ensure that both trends could be accurately seen.

The modelling considered four components or species with unique floatability characteristics in the pulp:

- j=1: Platinum and Gold rich ore fraction
- j=2: Chromite rich ore fraction
- j=3: Gangue fraction
- j=4: Water

The number of species considered is not limited to these four, and can be expanded to include any number of components to facilitate more accurate modelling or to reflect differing

degrees of floatability or different basic flotation models. The first order flotation rate constant, k_{ij} , found in equation 2-9, is distinct for each mineral fraction or species in each cell.

It can be seen from figure 4-1, that there are 10 flotation banks each consisting of a varying number of cells and having different sizes. These flotation banks are represented by elements 1 through 10, or for the modelling code purposes $C=10$, the number of cell banks. There are two mills and two tanks, which are elements 11 through 14 (represented in the code by $MT=4$). The product streams shown on the diagram are represented by elements 15 through 17, but in the model itself there are, in fact, a further six product streams, elements 18 through 23 ($PS=9$). The product streams are the means by which combinations of streams are modelled dynamically. For modelling purposes these 23 elements make up the process units of the LONMIN eastern platinum B-stream flotation circuit.

The modelling of the flotation cells was carried out as follows:

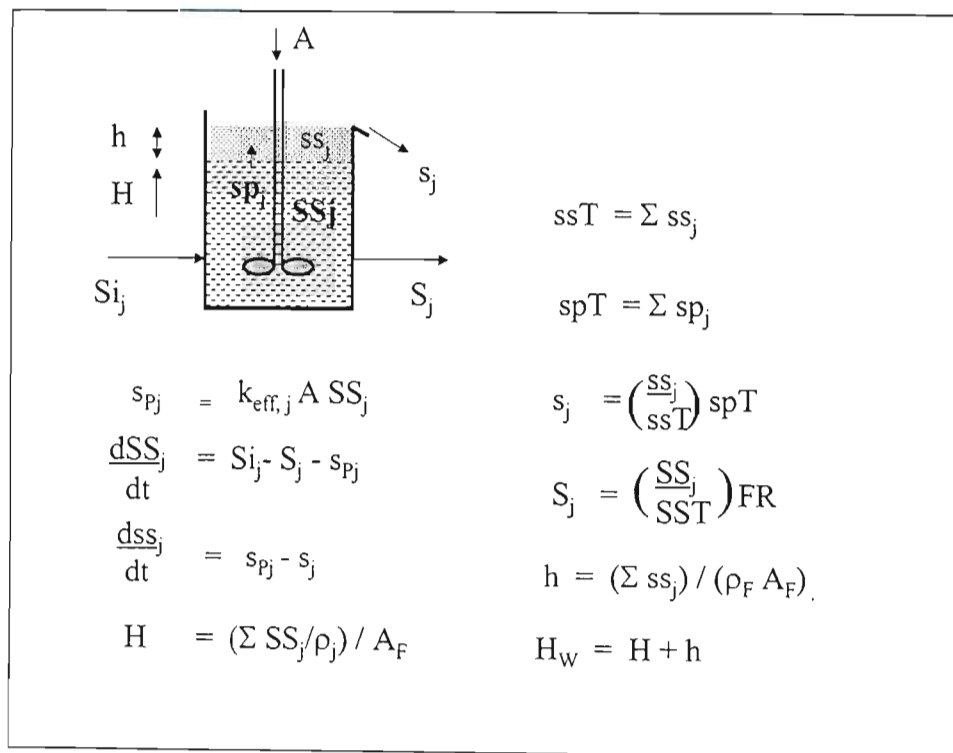


Figure 4-3: The flotation cell modelling balances and variables

The first order flotation kinetic rate equation (equation 2-9) is used to determine the rate of transport of material to the froth. The flotation rate constant, k_{eff} , used above takes into account the mill power, the airflow rate, the froth depth and a cell factor. The cell factor is used as an indication of the mass pull on the cells. The effective flotation rate constant k_{eff} is calculated in the following way:

$$k_{eff} = k \cdot (\text{cellfactor}) \left(1 + kgradA \left(\frac{A}{A_b} - 1 \right) \right) \left(1 + kgradh \left(\frac{h}{h_b} - 1 \right) \right) \left(1 + kgradPm (Pm - 1) \right)$$

Eq 4-1

where:

k_{eff}	Effective flotation rate constant
k	Flotation rate constant at the base condition $A = A_b$, $h = h_b$, and $Pm = 1$
A	Air flowrate
h	Froth height
Pm	Mill power effect, affects only those cells receiving material from each specific mill
cellfactor	A factor used to indicate mass pull on cells
$kgradA$	Fractional change of flotation rate constants k with fractional air flow
$kgradh$	Fractional change of flotation rate constants k with fractional froth height
$kgradPm$	Fractional change of flotation rate constants k with mill power factor
A_b	Standard or base airflow rate
h_b	Standard or base froth height

As can be seen from the above equation, if the airflow rate, froth height and mill powers are all at their base values, or standard values, they will have no effect on the flotation rate constant. Therefore once a base condition is identified, the $kgrad$ terms can be established by observing changes about the base condition on the plant. The extended form of the Kalman filter to be used will accommodate the non-linearity of this equation in terms of the system states.

The modelling of the Mills and surge tanks was carried out in the following form:

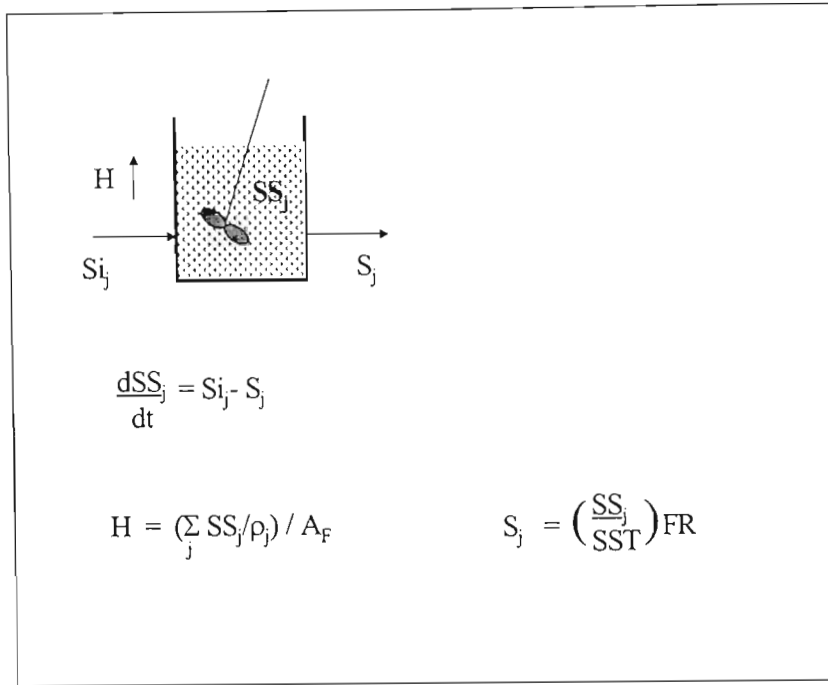


Figure 4-4: The mills and surge tanks variables and balances

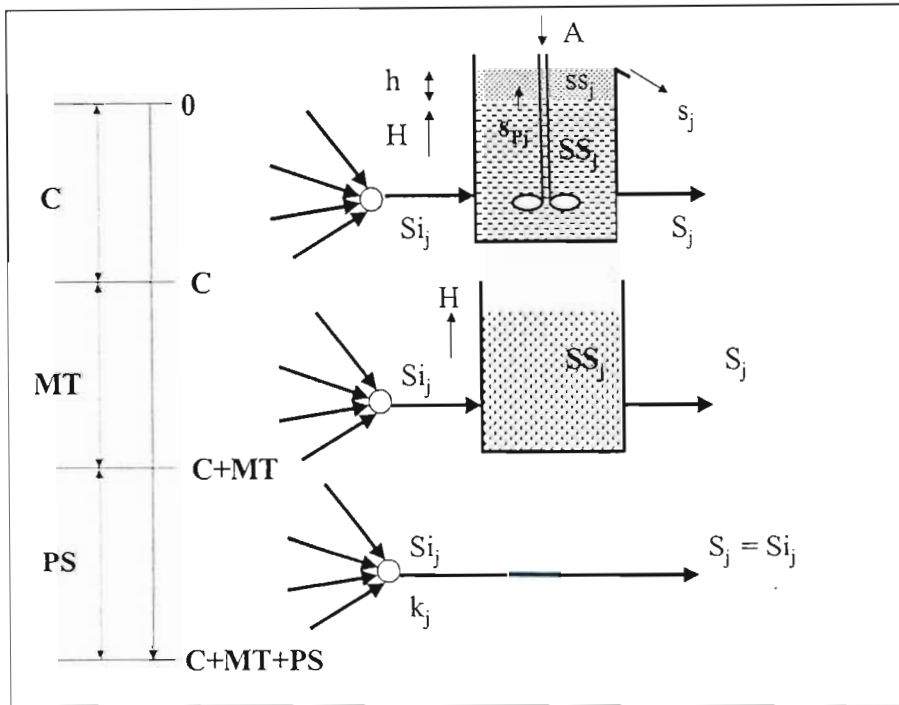


Figure 4-5: The three different types of elements found in the flowsheet modelling of the LONMIN Eastern platinum B-stream (C = Flotation Cells, MT = Mills and Tanks, PS = Product Streams)

Consider how a stream split such as a flow splitter in a manifold or a splitting device, such as a thickener or spiral separators are handled in the modelling. These cases are dealt with using a selection matrix operating at the summation node of each element. This allows for the definition of the circuit to be element based and connecting streams do not exist as separate entities. In the circuit considered the maximum number of streams arriving at any one receiving node is four, therefore the selection matrix is made up of C rows and four columns. This matrix then indicates which flows go to which elements. If an entry in the selection matrix is negative it indicates that the flow is from the indicated element but is a concentrate flow and not a tailings flow. A separate vector stores the splits of species for example after a spiral separator. Below is a diagram explaining how the selection matrix is used along with the species splits.

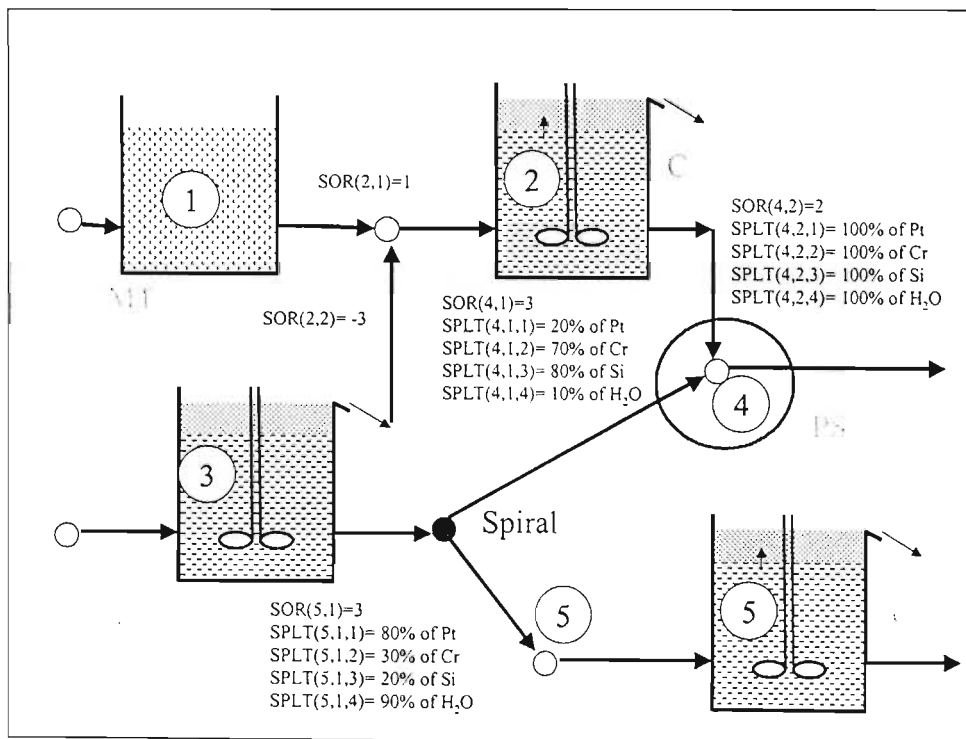


Figure 4-6: Use of receiving node selection matrix representing stream splitters.

Thus the final set of algebraic equations required to define the system topology has the following form:

$$S_{i,j} = \sum_{k=1}^4 SPLT(i,k,j) \cdot S_{SOR(i,k)_{IF POSITIVE} \cdot j} + \sum_{k=1}^4 SPLT(i,k,j) \cdot S_{-SOR(i,k)_{IF NEGATIVE} \cdot j} \quad \text{Eq 4-2}$$

Further information is required to provide for a realistic starting point of the solution if it is not started off a previously run simulation's end point. This information, such as initial values for compositions and flows of exit streams of the elements, is based on plant assays where available, or otherwise roughly estimated by plant operators.

Table 4-1 is a table of most of the variables used in the modelling of the circuit along with a description of each. Table 4-2 is a table of the initial value variables required for a start-up of the simulation. Table 4-3 and 4-4 are the models inputs and outputs variables respectively.

Table 4-1: Table of variables used in model solution, of the flotation cells, mills and tanks (i = element, j = species)

Variables	Description
h_i	Froth height in the cell
H_i	Height of pulp in the cell
A_{Fi}	Airflow rate into the cell
ρ_j	Density of species
$k_{eff\ i,j}$	Effective flotation rate constant
$sp_{i,j}$	Mass flowrate of material into the froth phase
$s_{i,j}$	Mass flowrate of material into concentrate launders
$ss_{i,j}$	Mass of each species in the froth phase
spT_i	Total mass flowrate of material entering the froth phase
ssT_i	Mass inventory of material in the froth phase
$Si_{i,j}$	Mass flowrate of pulp entering the flotation cell
$S_{i,j}$	Mass flowrate of each species leaving in the tailings
$SS_{i,j}$	Mass of each species in the pulp phase in the cell
FR_i	Total mass flowrate of material leaving the cell i.e. tailings
SST_i	Mass inventory of material in the pulp phase in the cell

Table 4-2: Model initialisation set of variables

Variables	Description
-----------	-------------

$X_{i,j}$	Mass fraction of mineral j in tailings flow ($j = 1, \dots, 3$)
SMF_i	Solids mass fraction of tailings flow from cell i .
$x_{i,j}$	Mass fraction of mineral j in concentrate
smf_i	Solids mass fraction of concentrate flow from cell i
H_i	Pulp height in cell i

Table 4-3: Model input on each time step. (Minimum set of observations required for convergence, however any variables may be observed to utilise the Kalman filter)

Variables	Description
$X_{i,j}$	Mass fraction of mineral j in plant feed. ($i=11$)
SMF_i	Solids mass fraction in plant feed ($i=11$)
FM_i	Minerals mass flowrate of the plant feed ($i=11$) (t/h)
A_i	Airflow rates to each bank ($i=1,2,3, \dots, 10$)
Pm_i	Mill power ($i = 11, 12$)
FL_i	Water make-up on sump ($i=23$)
OPTIONAL	FOR CELLS $i=1,2,3, \dots, 10$
FR_i	Total mass flowrate of tailings from cell i
$X_{i,j}$	Mass fractions of minerals in the tailings of cell i
SMF_i	Solids mass fractions in the tailings of cell i
$x_{i,j}$	Mass fractions of minerals leaving in the concentrate of cell i
smf_i	Solids mass fraction of concentrate flow of cell i
fm	Concentrate mass flowrate of cell i

Table 4-4: Model Outputs on each time step

Variables	Description
$X_{i,j}$	Mass fraction of mineral j entering cell i .
SMF_i	Solids mass fraction entering cell i .
FM_i	Minerals mass flowrate entering cell i (t/h)
$x_{i,j}$	Mass fractions of minerals leaving in the concentrate
smf_i	Solids mass fractions of concentrate flows

f_{m_i}	Concentrate mass flowrates (t/h)
$X_{i,j}$	Mass fractions of minerals leaving in the tailings
SMF_i	Solids mass fractions in the tailings
FM_i	Mass flowrate of material leaving the cell i.e. tailings (t/h)
H_i	Pulp Height in cell i . (m)
h_i	Froth depth in cell i . (m)
ADDITIONAL OUTPUTS FOR OPTIONAL INPUTS AND SIMULATION MODES	
$k_{i,j}$	Flotation rate constant k for each cell i and species j .

4.4 DIFFERENTIAL AND ALGEBRAIC EQUATIONS (DAE) SOLUTION METHOD

Modelling the LONMIN Eastern platinum B-stream results in a set of Differential and Algebraic (DAE) equations which must be solved simultaneously. A number of methods are available for achieving a solution which would accurately integrate the states and at the same time balance the algebraic equations, some of which have been reviewed and may be found in Appendix C. However, in the present case the solution is achieved by re-constituting the entire problem as an extended Kalman filter (EKF). This method allows additional incidental variables, such as stream flows, to be represented as state variables with fast time constants. "Observed" incidental variables decay towards their observed value, whereas unobserved incidental variables decay towards their last value. The forcing of equivalences is handled inside the "observation" part of the extended Kalman filter. The solution technique for the extended Kalman filter in this work makes use of numerically calculating gradients by perturbation. For a more detailed description and summary of the most common forms of the Kalman filter, namely the discrete and extended Kalman filters refer to Appendix J. The specific Kalman filter solution method used in this work will be described in the following section.

4.4.1 THE SPECIFIC DYNAMIC OBSERVER MODEL SOLUTION

Consider the system of first order differential and algebraic equations, with y states, for example composition and z incidental variables, for example flows.

$$\frac{dy}{dt} = f(y, z) \tag{Eq 4-3}$$

$$0 = g(y, z) \tag{Eq 4-4}$$

Defining the Jacobians:

$$\mathbf{A}_{[i,j]} = \frac{\partial \mathbf{f}_{[i]}}{\partial \mathbf{y}_{[j]}}(\mathbf{y}, \mathbf{z}) \quad \text{Eq 4-5}$$

$$\mathbf{B}_{[i,j]} = \frac{\partial \mathbf{f}_{[i]}}{\partial \mathbf{z}_{[j]}}(\mathbf{y}, \mathbf{z}) \quad \text{Eq 4-6}$$

$$\mathbf{C}_{[i,j]} = \frac{\partial \mathbf{g}_{[i]}}{\partial \mathbf{y}_{[j]}}(\mathbf{y}, \mathbf{z}) \quad \text{Eq 4-7}$$

$$\mathbf{D}_{[i,j]} = \frac{\partial \mathbf{g}_{[i]}}{\partial \mathbf{z}_{[j]}}(\mathbf{y}, \mathbf{z}) \quad \text{Eq 4-8}$$

Linearizing the right hand sides of equations 4-3 and 4-4 about $(\mathbf{y}_0, \mathbf{z}_0)$ and substituting the Jacobians, gives the augmented system:

$$\begin{aligned} \begin{pmatrix} \dot{\mathbf{y}} \\ \mathbf{0} \end{pmatrix} &= \begin{pmatrix} \mathbf{f}(\mathbf{y}_0, \mathbf{z}_0) \\ \mathbf{g}(\mathbf{y}_0, \mathbf{z}_0) \end{pmatrix} + \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{pmatrix} (\mathbf{y} - \mathbf{y}_0) \\ (\mathbf{z} - \mathbf{z}_0) \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{F}_0 \\ \mathbf{G}_0 \end{pmatrix} + \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix} \end{aligned} \quad \text{Eq 4-9}$$

where:

$$\begin{pmatrix} \mathbf{F}_0 \\ \mathbf{G}_0 \end{pmatrix} = \begin{pmatrix} \mathbf{f}(\mathbf{y}_0, \mathbf{z}_0) - \mathbf{A}\mathbf{y}_0 - \mathbf{B}\mathbf{z}_0 \\ \mathbf{g}(\mathbf{y}_0, \mathbf{z}_0) - \mathbf{C}\mathbf{y}_0 - \mathbf{D}\mathbf{z}_0 \end{pmatrix} \quad \text{Eq 4-10}$$

To allow for the possibility that some of the \mathbf{z} elements might be free, the behaviour is over specified, by suggesting that \mathbf{z} will move towards some observed value \mathbf{z}_0 the rate of which is controlled by τ .

$$\dot{\mathbf{z}} = \frac{1}{\tau}(\mathbf{z}_0 - \mathbf{z}) \quad \text{Eq 4-11}$$

so that

$$\begin{pmatrix} \dot{\mathbf{y}} \\ \dot{\mathbf{z}} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_0 \\ \mathbf{H}_0 \end{pmatrix} + \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{E} \end{bmatrix} \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix} \quad \text{Eq 4-12}$$

with

$$\mathbf{H}_0 = \frac{1}{\tau} \mathbf{z}_0 \quad \text{Eq 4-13}$$

$$\mathbf{E} = -\frac{1}{\tau} \mathbf{I} \quad \text{Eq 4-14}$$

Additionally, it is required that

$$[\mathbf{C} \quad \mathbf{D}] \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix} = -\mathbf{G}_0 \quad \text{Eq 4-15}$$

To handle the possibility that the states \mathbf{y} may also be observed as \mathbf{y}_0 , the above equation is augmented as follows,

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{y}_0 \\ -\mathbf{G}_0 \end{pmatrix} \quad \text{Eq 4-16}$$

letting $\mathbf{x} = \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix}$, $\phi = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{E} \end{bmatrix}$ and $\mathbf{u}_0 = \begin{pmatrix} \mathbf{F}_0 \\ \mathbf{H}_0 \end{pmatrix}$, integrate equation 4-12 from $\mathbf{0}$ to \mathbf{t} keeping \mathbf{u}_0 fixed:

$$\mathbf{x}_t = \mathbf{e}^{\phi t} \mathbf{x}_0 + [\mathbf{e}^{\phi t} - \mathbf{I}] \phi^{-1} \mathbf{u}_0 \quad \text{Eq 4-17}$$

The same integration can be applied to the interval $\mathbf{t} \rightarrow \mathbf{t} + \Delta \mathbf{t}$, giving

$$\mathbf{x}_{t+\Delta t} = \mathbf{A}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t \quad \text{Eq 4-18}$$

with

$$\mathbf{A}_t = \mathbf{e}^{\phi \Delta t} \quad \text{Eq 4-19}$$

$$\mathbf{B}_t = [\mathbf{e}^{\phi \Delta t} - \mathbf{I}] \phi^{-1} \quad \text{Eq 4-20}$$

The fact that Φ is usually singular does not mean that we cannot evaluate these coefficient matrices, as can be seen from the following:

$$e^{\phi t} \approx e^{[0]} + \frac{(\phi \Delta t)}{1!} e^{[0]} + \frac{(\phi \Delta t)^2}{2!} e^{[0]} + \frac{(\phi \Delta t)^3}{3!} e^{[0]} + \frac{(\phi \Delta t)^4}{4!} e^{[0]} + \dots \text{ with } e^{[0]} = \mathbf{I} \quad \text{Eq 4-21}$$

So

$$\mathbf{B}_t = [e^{\phi \Delta t} - \mathbf{I}] \phi^{-1} = \Delta t \left\{ \frac{\mathbf{I}}{1!} + \frac{(\phi \Delta t)}{2!} + \frac{(\phi \Delta t)^2}{3!} + \frac{(\phi \Delta t)^3}{4!} + \dots \right\} \quad \text{Eq 4-22}$$

Setting $\mathbf{C}_t = \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}$, $\mathbf{w}_t = \begin{pmatrix} y_0 \\ -\mathbf{G}_t \end{pmatrix}$ with \mathbf{L} selecting observed states, it is also required that

$$\mathbf{C}_t \mathbf{x}_t = \mathbf{w}_t$$

By representing the equivalent set of measurements by $\hat{\mathbf{w}}_t$ the Kalman filter is configured as follows,

$$\mathbf{K}_t = \mathbf{M}_t \mathbf{C}_t^T [\mathbf{C}_t \mathbf{M}_t \mathbf{C}_t^T + \mathbf{R}]^{-1} \quad \text{Eq 4-23}$$

$$\mathbf{x}_{t+\Delta t} = \mathbf{A}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t + \mathbf{K}_t [\hat{\mathbf{w}}_t - \mathbf{C}_t \mathbf{x}_t] \quad \text{Eq 4-24}$$

$$\mathbf{M}_{t+\Delta t} = \mathbf{A}_t [\mathbf{I} - \mathbf{K}_t \mathbf{C}_t] \mathbf{M}_t \mathbf{A}_t^T + \mathbf{Q} \quad \text{Eq 4-25}$$

where the covariance matrix is initialised with \mathbf{M}_0 small to give fast changes at the start, while \mathbf{Q} and \mathbf{R} are the expected error covariance matrices for the process and the measurements respectively.

We shall use \mathbf{R} as a diagonal matrix with the expected observation error variances on the axis. A small term indicates a high confidence in the measurement, which results in the \mathbf{K}_t tracking the measurement closely as opposed to following the model equations. The confidence in these model equations is expressed in a similar way by the diagonal weighting matrix \mathbf{Q} . It was found that the ranges of sizes of terms in the inversion $[\mathbf{C}_t \mathbf{M}_t \mathbf{C}_t^T + \mathbf{R}]^{-1}$ could be reduced by removing \mathbf{R} , and applying a pre-multiplication of the inverse square-root of \mathbf{R} to \mathbf{C}_t .

$$\mathbf{K}_t = \mathbf{M}_t \mathbf{W}_t^T [\mathbf{W}_t \mathbf{M}_t \mathbf{W}_t^T + \mathbf{I}]^{-1} \mathbf{R}^{-1/2} \quad \text{Eq 4-26}$$

where:

$$\mathbf{W}_t = \mathbf{R}^{-1/2} \mathbf{C}_t \quad \text{Eq 4-27}$$

It can be noted that the inverse $[\mathbf{W}_t \mathbf{M}_t \mathbf{W}_t^T + \mathbf{I}]^{-1}$ is not required for any other purpose, so the set of equations for the elements of the matrix \mathbf{K}_t in equation 4-26 can be solved as it stands. In practise it is more efficient to solve in terms of the transpose, i.e.

$$\mathbf{K}_t^T \mathbf{R}^{1/2} = [\mathbf{W}_t \mathbf{M}_t \mathbf{W}_t^T + \mathbf{I}]^{-1} \mathbf{W}_t \mathbf{M}_t^T \quad \text{Eq 4-28}$$

which has the form

$$\mathbf{x} = \mathbf{A}^{-1} \mathbf{y} \quad (\text{or } \mathbf{x} = \mathbf{A} \setminus \mathbf{y} \text{ in MATLAB}) \quad \text{Eq 4-29}$$

Four different solutions have been provided in the model code for this computation, the most time-consuming entity in the program execution, which are further explained in section 6.2.3. It should be noted that \mathbf{K}_t is updated less frequently if the operating point does not change while simultaneously if any variable moves more than a specified fraction (5%) of its range, \mathbf{K}_t is recalculated immediately.

Equation 4-24 is the key determining the dynamics of the filtered system, with equations 4-23 and 4-25 recursively updating the Kalman gain \mathbf{K} used in equation 4-24. Descriptions of how the \mathbf{Q} and \mathbf{R} matrices are operated have already been provided, but it should be noted that the elements of these matrices are related to squares of the variables. Bearing in mind the ranges of variables in engineering units, from mass fractions of 10^{-6} to flows of 300t/h, the squares span a ratio of about 10^{17} , so numerical errors can be expected if no attempt to normalise key equations is made. These errors were removed by rearranging the relationships, with the smallest values concerned, the cell inventories of froth, represented in kilograms rather than tons ($FrScale = 1000$). Thus variables could be worked with, in their original engineering units, which aided debugging and the equations retained their readability (Mulholland et al., 2003).

It is difficult for a user to decide upon appropriate settings for the \mathbf{Q} and \mathbf{R} matrices, so an automatic technique was developed for this. The technique is based upon the user specifying a nominal range for each variable and then to specify the percentage error in this range that is to be expected in the tracking of that variable. The method used to accomplish this may be found in Appendix H.

4.5 THE DYNAMIC OBSERVER MODEL FEATURES

The objectives of the dynamic observer model were clear at the beginning of the project. Although most were achieved, not all were easily met and some could not be fulfilled. What follows is an overview of the features of the dynamic observer model and its usefulness in terms of simulations and plant performance description.

4.5.1 INFERENCE OF UNMEASURABLE PARAMETERS

As mentioned earlier, the amount of information that is available from a flotation circuit is limited. Sensors do not provide all of the information necessary and in many cases, either a device cannot be devised to generate measurements of the desired variable, or the cost of including such a measurement is prohibitive. In some situations a number of different devices yield functionally related signals and a best estimate of the variables of interest must be generated based on partially redundant data. The solution to such a problem is the Kalman filter, discussed earlier, as it uses a process model along with available plant measurements to infer unmeasurable variables. The Kalman filter operating inside the observer model generates all of the mass species concentrations and mass flowrates throughout the entire plant. If all of the stream properties throughout the plant are known and are provided to the Kalman filter, the model can be used as an identification tool for the effective flotation rate constants. These new variables and information about the plant give a different, more coherent view of the plant and its performance. This can be a useful tool when considering what control actions should be taken and what effect control actions will have on the plant. The model can also be used as a complete plant simulator as all of the plant variables are considered and can be manipulated, and analysis can be undertaken as to the effects of the changes.

4.5.2 MODEL OPERATIONAL MODES

The dynamic observer model can be configured in a number of ways to facilitate different operating scenarios and to achieve different goals. Since the solution of the model is based on a Kalman filter, it is really a mixed mode solution in which any available measurements can be presented continuously to it and it will reconcile the remaining variables with this information for the output. Some of the available modes of operation with specific objectives will be described here.

An interesting feature of the model is the “speed-up factor”, which is used during most simulation runs and modes of operation. This can be set to increase the speed at which the

simulated plant operates. For example, with the speed-up factor set to 2, the simulated plant runs at twice the normal plant's operating speed. This is essential when running "what if scenarios", as the lengthy amount of time necessary for changes to take effect on the real plant are averted during the simulation. The speed-up factor is limited to 5 at present due to computation limitations, as any further increase would result in too large a computational demand.

4.5.2.1 Run-Forward Mode

In this mode of operation the simulated plant can run forward in time past the present and predict where the plant should be at some set time in the future. This proves useful in seeing what effect a change presently made will have on the plant and its performance in a few hours time. In this mode, the simulated plant can be sped up and run through to the set end time, and the results viewed. Manual inputs by operators can also be applied to the simulator instead of the plant to see what effects these changes will have on the plant in the future.

4.5.2.2 Real-Time Mode

When the simulator runs in this mode, it will simply attempt to mimic the plant's behaviour, following any changes made in the plant and predicting all unmeasurable variables in real-time. This mode of operation is the most useful, as the unmeasurable variables are made available, and possible control strategies can be based upon them and executed to improve plant performance. In this mode the speed up feature is turned off to allow the simulator to follow the plant at the correct time. This mode of operation should also prove useful when training new plant operators as it allows changes to be made to a simulated plant and the effects of these changes observed next to the real plant outputs, without necessarily affecting the plant in a negative way.

4.5.2.3 Historical Mode

In this mode the simulator receives plant measurements off a database of previously saved plant measurements. This feature would allow users to make different changes to a standard set of control inputs and see what effect each would have, with the advantage of reproducibility. This feature must still be integrated into the real-time PlantStar environment by MINTEK.

5 INTRODUCTION TO THE PLANTSTAR® CONTROL PLATFORM

PlantStar is an open plant-wide control system developed by MINTEK, which is widely applicable, but aimed mainly at metallurgical processes. It utilises advances in software technology and advanced process control to provide a stable optimised control platform. This control platform forms the basis of the on-line implementation of the DOM (Dynamic Observer Model). An overview of the PlantStar control platform and its features is to follow so as to give a better understanding of the control system, before undertaking an in-depth look at the implementation of the DOM into this type of on-line environment in chapter 6.

5.1 PLANTSTAR'S LAYERED APPROACH

PlantStar uses a layered approach to process control and optimisation, as depicted below.

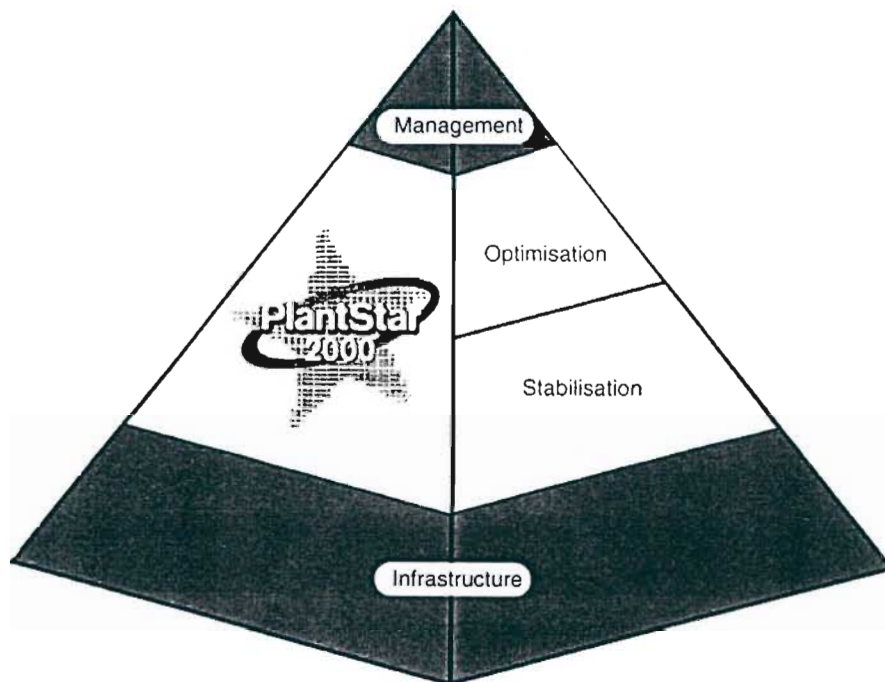


Figure 5-1: PlantStar's layered approach to control (PlantStar user manual, 2002)

5.1.1 PROCESS INFRASTRUCTURE

PlantStar communicates with the plant infrastructure layer, made up of plant instruments such as PLC's and DCS systems, through OLE when undertaking process control (OPC), which is a standard protocol for the process industry. Communications to the outside world are then facilitated through an OPC server allowing remote access.

5.1.2 PROCESS STABILISATION

PlantStar uses a rules based tree structure for defining the control strategy. Since PlantStar is open, any control modules can be integrated or "plugged" into it, from simple PI control to very advanced multivariable control strategies and sophisticated optimisation algorithms such as the DOM.

5.1.3 PROCESS OPTIMISATION

PlantStar's rule tree structure can vary from simple safety loops through to integrated process optimisation routines. These rules and strategies can be adapted on-line, which enables plant management to implement any process improvement strategy in a robust and user friendly environment. Though PlantStar can be used for low-level plant control it is intended especially for more complex control strategies that cannot be carried out by a standard PLC or DCS, such as,

- General stabilisation of a plant by manipulating low-level setpoints, such as flowrate setpoints.
- Manipulating setpoints that would normally be adjusted manually so as to achieve long-term target values such as production targets.
- Control of Process Variables that are normally difficult to control, e.g. highly non-linear processes.
- Control of processes with long time delays by using time delay compensation algorithms.
- "Safety-jackets": temporarily relinquishing the normal control strategies to rectify abnormal situations on the plant.

- Complex multivariable algorithms with differing degrees of interaction between the variables.
- Sophisticated optimisation strategies.
- Rules-based expert systems.
- Artificial intelligence modules.
- Modelling tools.

Process specific model-based controllers can be tailor made for specific installations, as is the case with the dynamic observer model developed for this thesis, and implemented through PlantStar's open "plugin" module system.

5.1.4 PROCESS MANAGEMENT

PlantStar strongly separates the execution of control strategies from the user interface, which is operated by the User Manager. This approach makes the platform very robust, not allowing the user interface to destabilise the critical execution of the core control algorithms. PlantStar is available remotely through the User Manager, which allows users to view plant mimics, process trends, or even to edit control variables and rules from a remote location. Data logging of historical trends is also made available for use with a range of database programs, such as SQL.

5.2 PLANTSTAR ARCHITECTURE

The schematic diagram below shows a typical installation of PlantStar on a plant:

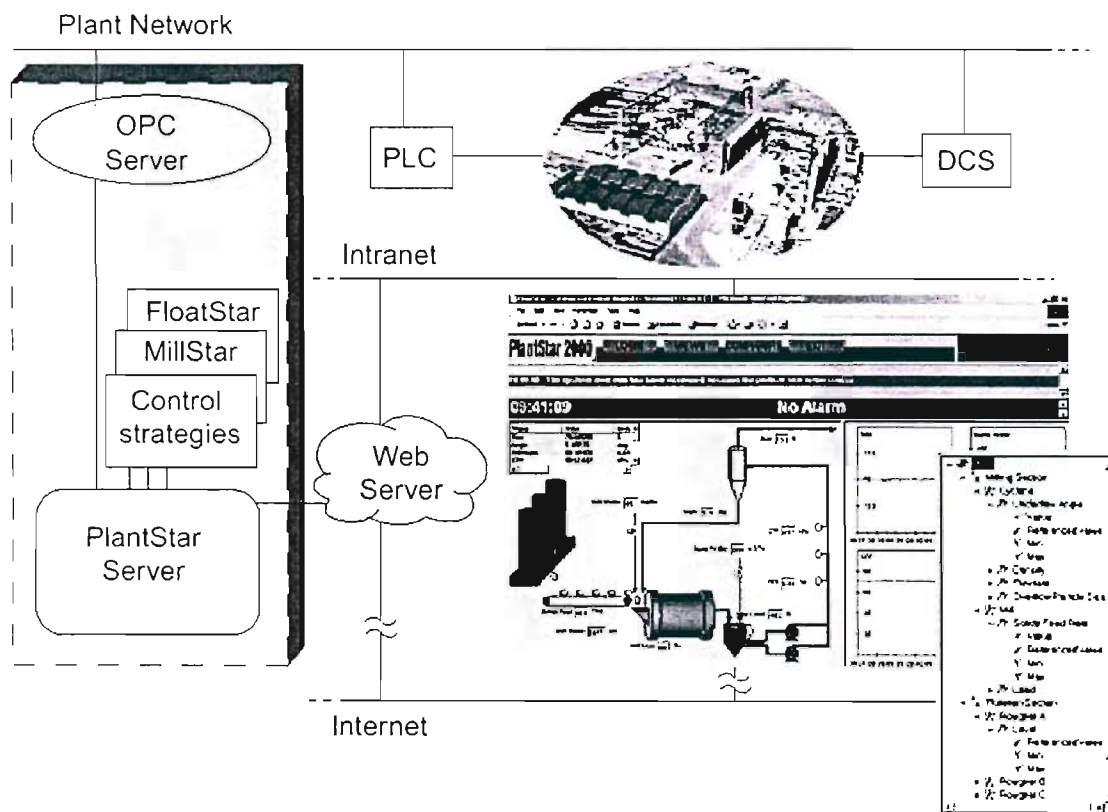


Figure 5-2: PlantStar Architecture (PlantStar user manual, 2002)

PlantStar obtains the values from the PLC and DCS through an OPC server. The user interface is accessed through the web server.

The PlantStar system contains the following components:

- PlantStar Executive (Core functionality inside the PlantStar Server).
- Control Strategies (Control algorithms that plug into PlantStar in the form of dynamic link libraries).
- OPC Client (For connecting to the OPC server).
- OPC Server (For exposing PlantStar values to the outside world).

- PlantStar Service (NT service in Windows running the PlantStar Server).
- User Manager.
- Windows Client.

The PlantStar Executive is responsible for most of the background processes, including control of the plant, opening of the OPC Connections, creation of a snapshot of the plant and handling security. (Taken directly from the PlantStar user manual, 2002)

5.3 PLANTSTAR USER INTERFACE

The PlantStar user interface is available through the PlantStar Windows Client, which connects to the User Manager. The windows client makes three distinct modules available to the user, namely: the configuration tool, the graph tool and the mimic tool.

5.3.1 CONFIGURATION MODULE

The rule tree structure used by PlantStar and mentioned above, is written and designed using the configuration module. This module allows for a complete *plant definition file* to be created and tested. The plant definition file has the complete plant layout and interconnections along with control strategies, which are required to control the plant. Section 6.3 has a more detailed description of the plant definition file and its creation and uses. The standard user interface would not have access to this module as it is used mainly during the design and optimisation of the plant's control strategies. Figure 5-3 is an example of a plant definition file and its tree structure.

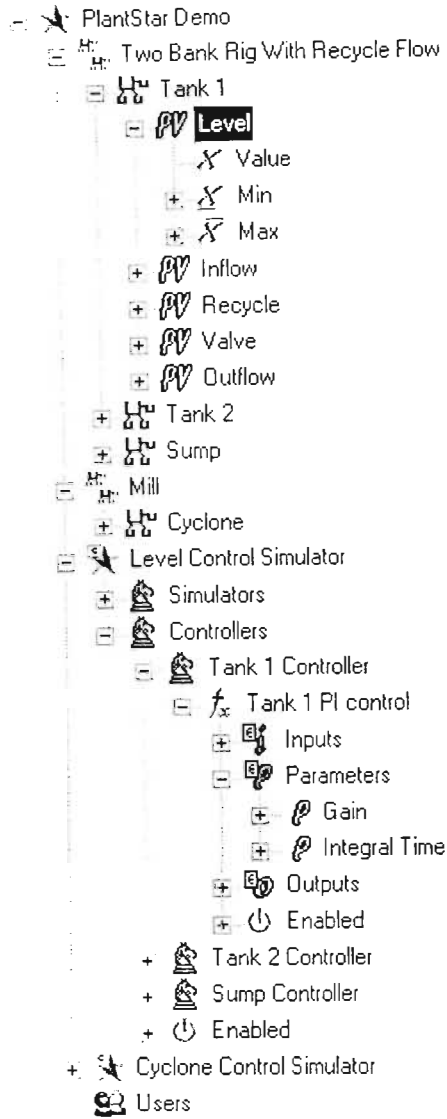


Figure 5-3: An example of a plant definition file showing the hierarchical rule tree structure (DemoPlant definition file, 2002)

5.3.2 THE GRAPH TOOL

The graphing tool is very easy to use and makes tracking of plant variables simple. Graphs can be made during run time but are usually created at design time to track pertinent variables. The graph tool's simplicity contributes to PlantStar's ease of use, as the creation of new graphs in real-time is simply a matter of referencing a trend to a plant variable in the plant definition file, as follows:

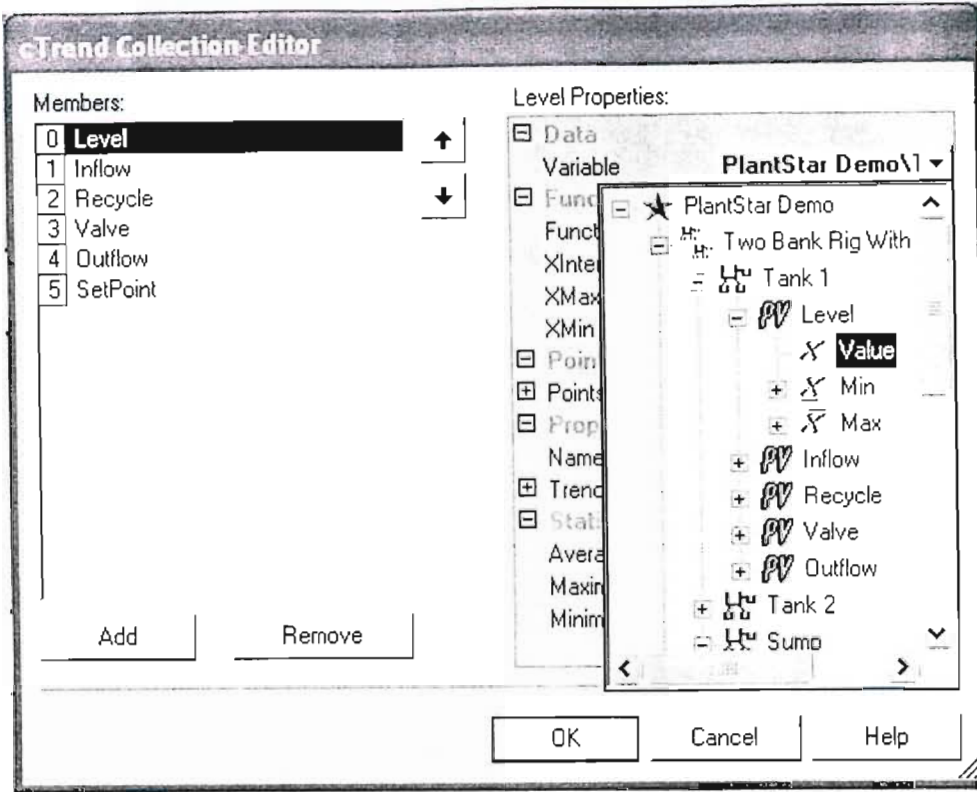


Figure 5-4: Creating a new trend on a graph with the graph tool (DemoPlant definition file, 2002)

The graphs are easy to configure and alter as far as their properties are concerned. Tracked points are stored in a data set, the size of which is variable along with the interval sampling time. These data sets allow the graphs to be scrolled back if the current position of the graph has moved beyond the viewing area of the graph and changes cannot be fully seen. The graphs can also be exported as an image or printed straight from within the windows client workspace. This is a useful tool when wanting to quickly save the state of a specific unit operation to a snapshot that can be reviewed later by other plant operators without having to find the logged data and plot it for a specific time interval. Below is a captured graph from PlantStar's demonstration plant showing the scrolling and image exporting options.

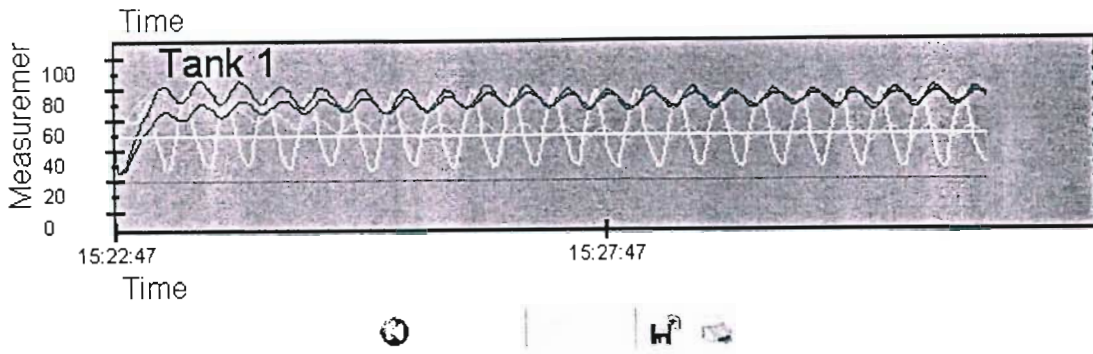


Figure 5-5: A graph of the demonstration plant from PlantStar (DemoPlant definition file, 2002).

5.3.3 THE MIMIC TOOL

The mimic tool is used to create a mimic of the plant's unit operations, so that a feel for the plant can be obtained. The mimics in PlantStar are easily created with standard format images and have many useful features for displaying the changes in the plant, for example, the changing levels in a tank can be seen on the tank diagram. This is achieved by simply linking the add-in diagram to the specific plant variable, in this case the tank level, in a similar manner to the graph trends referencing. Figure 5-6 displays one of the mimics from the PlantStar demonstration plant, showing the levels and setpoints of tanks and a sump.

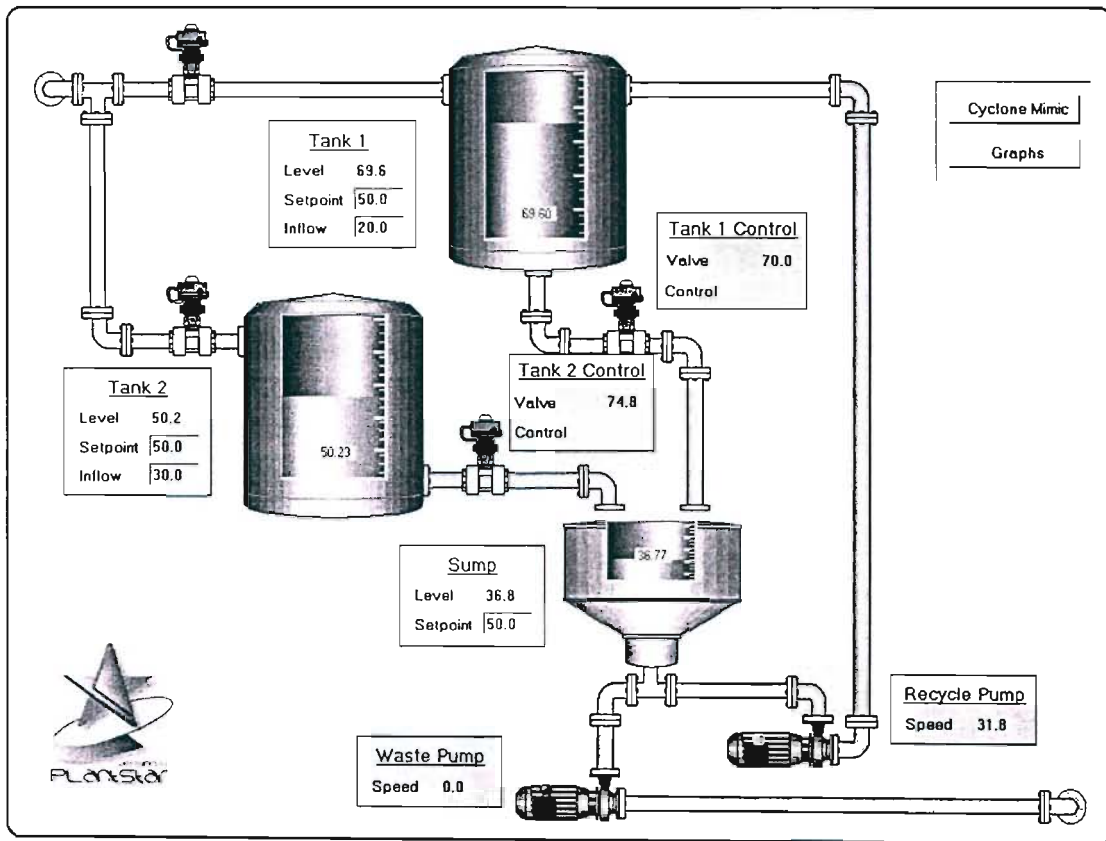


Figure 5-6: Mimic of the demonstration plant from PlantStar (DemoPlant definition file, 2002)

Mimics similar in nature to the one above are useful for operators as they can easily see the setpoints and the corresponding plant state.

6 DYNAMIC OBSERVER MODEL IMPLEMENTATION IN PLANTSTAR

The complex plant model is made up of a complicated solution algorithm as described in chapter 4, however a batch simulator that runs through a certain number of steps and sends the final state of the solution to a file to be reviewed was not the aim of the project. An on-line simulator as described in section 4.1 was required, and in order to implement it, the existing control platform, PlantStar, was viewed as the best approach, as it already has all available process data on-line. An overview of how the model module was developed and implemented is to follow.

6.1 THE MODULAR APPROACH

The control platform that the complex model runs on, PlantStar, was briefly described in Chapter 5. It has a modular structure that enables more complicated control algorithms, or simulators, to make use of its process variables that form part of the overall control strategy of the plant, be it low-level control or more sophisticated algorithms. The complex plant model operates using an extended Kalman filter and as such requires measurement inputs for some of the process variables, so that it may infer the rest. It is this which makes PlantStar useful, as these process variables are made available through the windows client. As a result the complex model module was created in such a way, that it could be “plugged” into PlantStar, to make use of these variables. The “plugin” modules are created as dynamic link libraries (DLL’s), which can connect to PlantStar through the plant definition file and still have the model library code, with all the relevant algorithms and solution methods, included into the DLL. These DLL’s need to be created in a specific way as to function properly with PlantStar. A brief overview of the development of the complex model’s DLL is presented in the following section.

6.2 THE DYNAMIC LINK LIBRARY DEVELOPMENT

PlantStar can be extended by means of add-on modules, referred to as algorithms, in the plant definition files. These algorithms are implemented as DLL's, to which PlantStar connects after interpreting the plant definition file. The DLL created for this project is called 'PSTest81B.dll' and the PlantStar class created in the DLL is 'TestClass'.

6.2.1 REQUIREMENTS OF THE DLL FOR USE WITH PLANTSTAR

6.2.1.1 Software Requirements and Updates

PlantStar, as a software program, has a minimum system requirement of at least:

- Pentium III 750MHz with 256MB RAM or equivalent.
- Microsoft Windows 2000/XP (Recommended XP).
- Optional Microsoft Excel/SQL for data exporting purposes.
- An OPC server for interfacing with the plant (required only at the site and not for off-line simulation).
- Internet/Intranet connection for client machines.

These are basic guidelines as set out in the PlantStar user manual, 2002, but as machines become more and more powerful, the latest and most powerful machine will most definitely provide the best performance and stability. The machine that the DLL was created and tested on, was a Pentium III 933MHz machine, with 384MB RAM and a page file size of approximately 1.2 GB. However, this machine struggled while running the simulations and could not perform data exportation as a result of the intense solution method used to solve the DAE system. More powerful machines should be able to fully realise the potential of the model and its ability to save all output data on every step, to a database.

For the DLL development, MINTEK developers recommended that the model be created as a compiled class in a Static Library in Microsoft Visual C++. This facilitates the creation of a PlantStar DLL, which has an instance of the main class in the Static Library. Microsoft Visual Studio 6, however, required an update to service pack 4 before it could be used to create PlantStar DLL's.

6.2.1.2 Member Functions Required by PlantStar

The algorithm, in the plant definition file, connects to the DLL through classes ('TestClass' in this project's DLL). Each of these classes needs to implement several *member functions*. These member functions must be available to PlantStar to facilitate connection, and are the following:

- INITIALISE: The Initialise function is called or executed only once by PlantStar, on startup. Any parameters, if defined, are static parameters, and are passed only once, at startup.
- EXECUTE: The Execute function is called at each interval provided some criterion set by PlantStar are met in such a way that it may proceed to the next step. One of the criterion is the interval step time of the algorithm, which must have elapsed since the previous algorithm execution. The Execute function accepts three groups of parameters, which are, Outputs, Inputs and Parameters, on every execution. The Inputs and Parameters are read-only, whereas the Outputs can be read and written by the DLL.
- SHUTDOWN: This is called only on shutting down PlantStar and has been used to save the end state of the simulator to data files, which can be used on startup to initialise the model.
- SAVE: Data can be saved as a string in XML format.
- LOAD: Data can be loaded as a string in XML format.

The last two functions are not necessary for operation of a DLL but may be useful for specific cases. They were not utilised for the dynamic observer model DLL.

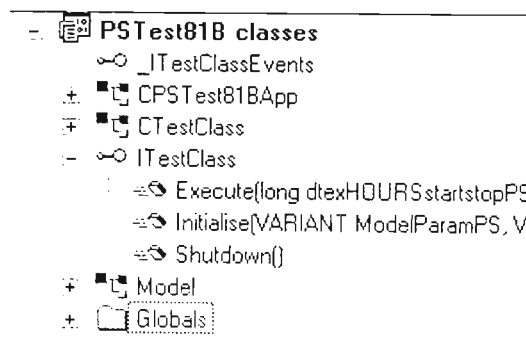


Figure 6-1: The member functions of the PSTest81B.TestClass

6.2.1.3 Data Types and Handling

PlantStar supports a variety of different data-types, such as doubles, longs, strings and booleans. If the DLL was created in Microsoft Visual Basic 6, the inputs would be passed by value and the outputs by reference, but in C++, (the programming language used for this DLL) the inputs were passed as value inputs, and the outputs as pointers. It is also possible to pass strings to and from PlantStar, done by means of BSTR strings, which must be converted to CStrings for use in C++. In order to achieve this conversion some files were supplied by MINTEK which carried out the conversions directly. These will be discussed in the following section.

When an array or matrix is passed from PlantStar to the DLL, it is passed as a Variant. Variants are constructs of Visual Basic and are not directly supported by C++, and therefore the Microsoft Automation libraries must be used to manipulate and use them. Fortunately MINTEK provided conversion libraries that were created specifically for use with PlantStar, and will be described in the following section. These libraries allowed the Variant arrays, that were passed from PlantStar, to be converted into the correct format in which all the matrix manipulations were carried out. These libraries are constructs of MINTEK and are propriety software. The matrix and sparse matrix manipulation libraries make use of these formats to facilitate the model solution.

6.2.1.4 PlantStar Libraries and Include Files

The DLL created for use with PlantStar required specific data types and handling routines as described earlier. The libraries and files required by the DLL to perform this were:

- PlantStarSDK_d.lib. This library contains code for conversions of Variants to other data types, amongst others.
- BSTRFixes.c: This facilitates the conversion of BSTR strings that are passed from PlantStar into the CStrings format that C++ uses and visa versa.
- ConvertVariant.h: The header file containing information for conversion of Variants to other data types.

6.2.2 MODEL CLASS

6.2.2.1 Model Header File

The model header file is the interface for the model class and contains all of the variable declarations of variables that the model uses to describe the plant and to find the solution to the extended Kalman filter problem, along with all solving functions and member functions used by PlantStar, such as “initialise” and “execute”.

6.2.2.2 Model File

The model class is where the entire process of identification, initialisation, execution and shutting down of the DOM is carried out. All starting conditions, stream connectivity's, DAE's, DAE solutions and parameters of the model are contained in this class. The DLL facilitates the connection of PlantStar, to this model class, and through it, the simulation of the complex plant. The model class receives all PlantStar variables through the DLL and then manipulates them to create all outputs, which are then passed back through the DLL, to PlantStar and the user interface.

6.2.3 SOLUTION METHODS AVAILABLE

The model class has the ability to make use of three different methods of determining the Kalman Filter Gain matrix, **K**, and the Covariance Matrix, **Mt**. More than one method was required, as the solutions involve large matrix inversions and manipulations, which must be performed rapidly for a real-time solution. Each of the methods were tested to find the most stable and fastest solution.

6.2.3.1 MATLAB Engine

The MATLAB Engine was found to be the fastest and most stable method, however as MATLAB is a licensed software program, it poses the problem of portability for the DLL, as any user of the DLL with this solution method would require a licensed copy of MATLAB to be installed on the machine in question, in order to run the MATLAB engine. This method involves passing the matrices in question directly to an opened MATLAB command window, where the appropriate calculations are undertaken. The results are passed back to the DLL and into the model class, where they are used in further calculations.

6.2.3.2 MATLAB Standalone File

The MATLAB standalone file method uses standalone MATLAB programs to calculate the solutions for the **K** and **Mt** matrices. The DOM writes intermediate files, which are read by the MATLAB standalone programs, which find the solutions. As the standalone programs are not integrated with the DLL, these solutions must be written to files before the DOM reads them and places them within its working matrices before carrying on with its execution. This method solves the problem of portability of the DLL and yet still makes use of MATLAB's enhanced matrix calculation routines. The drawback to this method is the disk read and write speed of the system, which is required by this solution method. In real-time this could be a problem, but if the model is only executed every few seconds, then this method of solution becomes more favourable.

6.2.3.3 Sparse Matrix Library Solution

The third method makes use of the matrix and sparse matrix libraries provided by MINTEK and professor M. Mulholland, including a conjugate-gradient linear solution algorithm. This method, however, is very slow and definitely cannot be used in a real-time scenario. The advantage of this method is that all of the manipulations are carried out by the model class itself and no external file dependencies are created. The sparse matrices pose the problem of clipping off values outside specified ranges, and information is therefore lost and hence the system could lose stability.

See Appendix H-1 for LinSolnMethod and FilterMethod, which give a more detailed description of these methods and their uses in the model solution.

6.2.4 **EXTERNAL DEPENDANCIES OF THE DLL**

Besides the essential PlantStar external dependencies above, the following library files were required in order to operate the DLL and its different solution methods.

6.2.4.1 MATLAB Engine

As was described above in section 6.2.3.1, one of the methods available for the solution of the matrix calculations required by the extended Kalman filter, is through the direct use of the MATLAB Engine. In order to have the engine operate with the DLL, some MATLAB files had to be included into the project, namely

- Engine.h: This allowed for the calling of the engine and its functions.

- Libeng.lib: The MATLAB engine library file.
- Matrix.h: The MATLAB matrix manipulation libraries header file.

This method turned out to be the fastest solution method and could handle the largest matrix sizes comfortably.

6.2.4.2 Matrix Manipulation Libraries

The matrix manipulations libraries made available by MINTEK and Professor M. Mulholland, are available as one of the solution methods, yet they are the slowest to converge and result in large lag times when using PlantStar. The libraries included are the following:

- MatrixMM: The matrix manipulation library based on the original MINTEK matrix library.
- SMatrix: The sparse matrix manipulation library adapted by Professor M. Mulholland.

6.2.5 **VARIABLE HANDLING BETWEEN PLANTSTAR AND MODEL CLASS**

As discussed earlier, PlantStar passes arrays and matrices of variables as data type Variant. These Variants are not directly supported in the C++ environment and therefore some data manipulations must be carried out in order to use the information. These data manipulations are carried out in the DLL in such a way that the data from PlantStar is made available to the model class. To accomplish this, the data is converted from type Variant into the appropriate data type for each variable utilised by the model class. For example, the model uses a class called 'cMatrix' which handles matrices. The data coming from PlantStar is therefore converted from type Variant into type 'cMatrix<double>' if the data array in PlantStar was of type double. This conversion is carried out using the ConvertVariant libraries discussed in section 6.2.1.4 and provided by MINTEK's development team.

As will be seen in later sections, the plant definition file created for the complex plant model has each of the process units as a whole and complete unit with all variables defining it as part of its tree structure. This layout is then copied for the Observation tree structure and for the Output tree structure.

The Observations tree is used to set whether or not the model is observing a variable. It is simply a flag inside the model code that tells the model to have a higher confidence in this plant variable's value rather than the value obtained from the last execution of the model. In other words the measurement equations (C_t) change along with the corresponding K_t matrix value. The observations are added or removed from the C_t matrix and observation error increased or decreased accordingly, while the M_t matrix size does not change. These flags must be set in order for the model to recognise an available plant measurement or a manual input of one of the plant variables. The Output tree receives and displays the predicted plant variable values, from the previous time step.

Each of the unit's input data for the observations and the outputs from the model have a single array with all of the process variables making up its elements. However the model has each of the process variables, as arrays, and each of the units as its elements. Therefore a complicated process of indexing and rearranging the incoming observations and exiting unit output arrays was required to match the model class's variables and the plant definition variables. The plant definition file could have been made with the same structure as the model class, but would however rapidly lose readability, creating problems when debugging the program. The plant information or states of the plant received from its measuring devices, does not pose the same problem, though they are also passed into the DLL on each time step. The reason being, is that these are passed from an element known as a referenced array which allows for a specific search through the tree structure for a key word. Starting at the top of the tree, it places each identified variable into the array in the order in which it is found. Therefore all model inputs from the plant are automatically "in" variable arrays and can simply be converted to `cMatrix<double>`'s and passed to the model class.

A detailed example of the conversions and manipulations described above can be found in Appendix F.

A table of the variables used in the solution of the dynamic observer model was provided in chapter 4. The variables that are used in PlantStar and by the DLL are the same process variables, but have different naming conventions to make them more readable to the user. What follows is a table of the process variables that are displayed by PlantStar and which might be encountered in the figures that follow.

Table 6-1: Table of variables comparison between those used by PlantStar and those used by the DLL and model class (The ‘i’ in the model names indicates inflow and is replaced by the word ‘inflow’ in the names of the PlantStar variables).

Model	PlantStar	Description
$X_{i,j}$	Inflow_X(i,j)	Mass fraction of mineral j entering cell i .
SMF_i	Inflow_SMF(i)	Solids mass fraction entering cell i .
FM_i	Inflow_FM(i)	Minerals mass flowrate entering cell i (t/h)
FL_i	Inflow_FL(i)	Liquid mass flowrate entering cell i (t/h)
$x_{i,j}$	Frothflow_x(i,j)	Mass fractions of minerals leaving in the concentrate
smf_i	Frothflow_smf(i)	Solids mass fractions of concentrate flows
fm_i	Frothflow_fm(i)	Concentrate mass flowrates (t/h)
fl_i	Frothflow_fl(i)	Liquid mass flowrate in concentrate from cell i (t/h)
$X_{i,j}$	Outflow_X(i,j)	Mass fractions of minerals leaving in the tailings
SMF_i	Outflow_SMF(i)	Solids mass fractions in the tailings
FM_i	Outflow_FM(i)	Mass flowrate of material leaving the cell i.e. tailings (t/h)
FL_i	Outflow_FL(i)	Liquid mass flowrate in tailings from cell i (t/h)
H_i	H_o(i)	Pulp Height in cell i . (m)
h_i	ho(i)	Froth depth in cell i . (m)
A_i	Ao(i)	Airflow rate to cell i
Pm_i ($i=11-12$)	Pmo(i)	Mill power factor (default =1)
Pm_i ($i=1-10$)	Pmo Cell factor	Mass Pull factor for cells
FR_i	FR(i)	Combined tailings flowrate from cell i
$k_{i,j}$	ko(i,j)	Flotation rate constant k for each cell i and species j .

6.2.6 TESTING THE DLL WITH VISUAL BASIC

Once the DLL’s basic structure had been defined, a small Microsoft Visual Basic 6 program was written in order to test all of the connections that PlantStar would use. It simply imitates PlantStar and applies all the member function calls to the DLL that PlantStar would while operating. Any problems that the DLL might have could then be easily detected and changed before re-testing. In this way the DLL could be written and debugged and then tested swiftly without using PlantStar as the test platform. The reason being, is that PlantStar initialises and starts up slowly because it has more functions to perform other than running the algorithm. The code used in the Microsoft Visual Basic 6 testing program can be found in Appendix D. A small setback was discovered while using this software to test the DLL, and that is the limitation of Microsoft Visual Basic 6’s (VB) variable passing into and out of a DLL. The program can only handle a specific number of variables passed into or out of it. This number

with regards to this project was in the region of 65 different variables. If the number of variables connecting to the DLL exceeds this value, the program will give an error and won't be able to continue. A reference to this limitation can be found at the following web address <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon98/html/vbconprojectlimitations.asp> in the section "Procedure, types and variables". The limitation is defined as the table size limit for storing variable names and cannot exceed 64K. Another reference to this limitation can be found in the MSDN help under "Too many Arguments". The model's DLL did exceed this size limit with its variable names and therefore certain variables that were unlikely to change were omitted from the DLL for testing in this way, and then re-introduced for the PlantStar DLL. Figure 6-2 is an image captured from the VB testing program after a run, displaying the variables associated with the primary Rougher 1.

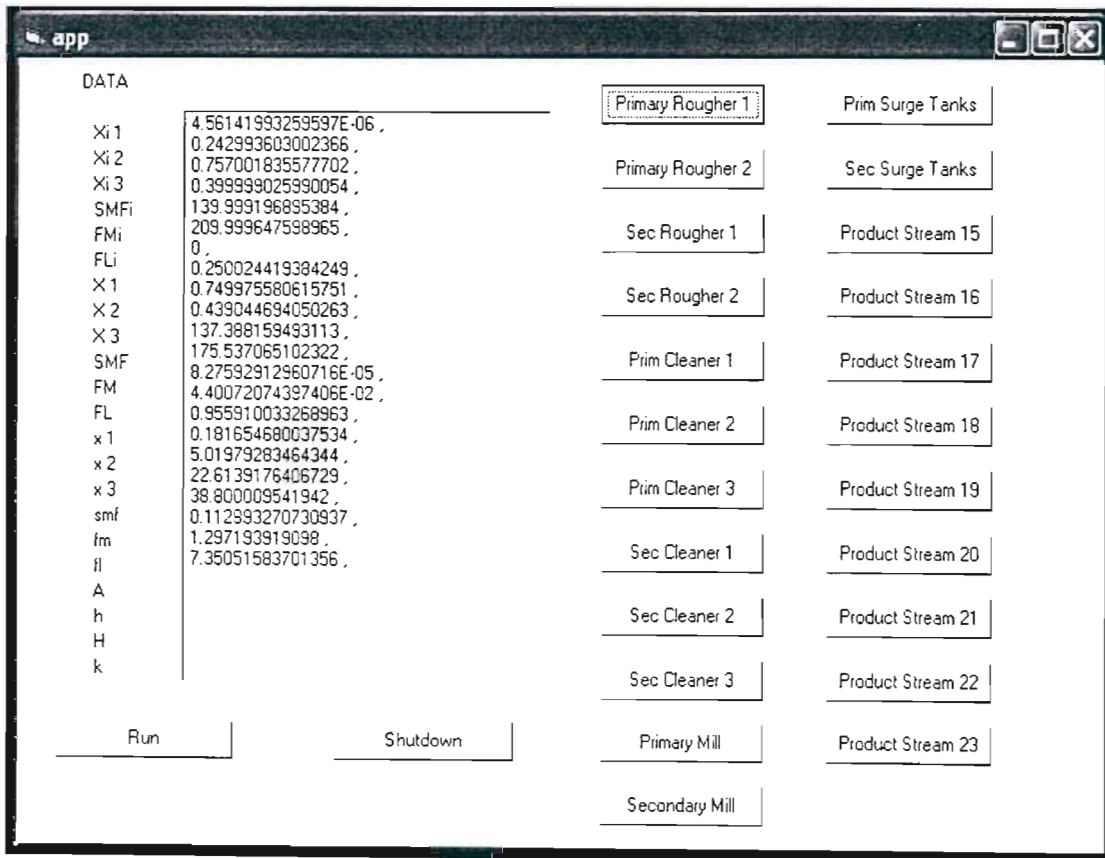


Figure 6-2: A screenshot of the Microsoft Visual Basic 6 testing program used to test the DLL in the early stages.

6.3 THE PLANTSTAR PLANT DEFINITION DEVELOPMENT

Once completed, the DLL would be useless without the accompanying plant definition file that PlantStar utilises. This definition file contains all of the relevant information that the DLL uses to find a solution for the complex plant. All the inputs, outputs and parameters that the DLL makes use of are passed in and out of the definition file and then displayed appropriately inside of the PlantStar platform or workspace. What follows is a description of how the plant definition file was created and its features that make it possible to operate with the DLL.

6.3.1 THE PROCESS TREE

The Process tree is the part of the plant definition file that receives values from the plant. It has all of the process units of the plant controlled by PlantStar in its structure, with their accompanying values which are received at intervals from the plant's low level monitoring systems, such as tank levels, valve positions, flowrates etc. These values can then be referenced and used by control strategies or algorithms such as the dynamic observer model DLL. The dynamic observer model process tree is made up of 23 separate process units, which can be seen on figure 4-1. There are the 10 flotation banks, 2 Mills and 2 surge tanks. The remaining 9 units are process stream summations that are used by the model to combine 2 or more streams together and result in one combined outflow stream.

Figure 6-3 shows the process tree structure taken from the plant definition file used by the DOM DLL. All of the variables that are used by the DOM are in the tree structure and will not necessarily be available from plant measuring devices, however the values can also be entered manually in this location and the model would treat them in the same way as a normal plant variable such as a level that is available from the plant. In this way if an operator has decided that the mass pull on a unit has increased by for example 15% then in order for the model to accurately reflect the change, the cell factor would be changed to 1.15 and this variable's observation flag would be set to true so that the model would recognise it and move towards the set value. The operator would not necessarily change the values on this tree structure, though this is in fact where the values would be changed for the DLL's functionality. The operator would use a user-friendly graphical user interface (GUI), which makes the changes to this tree easy and simple. These user interfaces will be described in a later section.

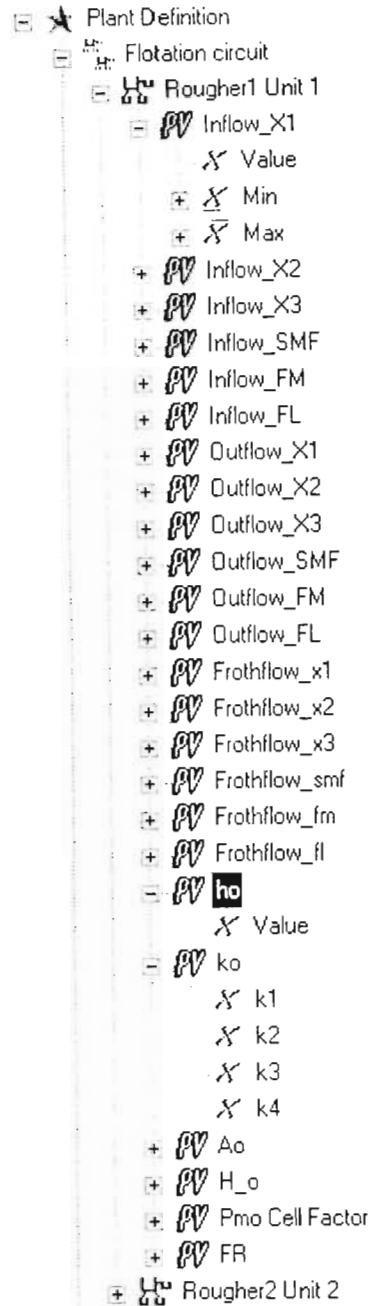


Figure 6-3: The Process tree structure of the DOM DLL plant definition file for Primary Rougher 1.

6.3.2 THE PROCESS STRATEGY TREE

Considering that the process tree is a replica of the plant with all plant data, the process strategy tree is used for all of the control and manipulation algorithm implementations. This tree is separate from the process tree and does not change any of the values there, but uses

those available variables from the plant to control and optimise the process. This control and optimisation is programmed into the process strategy tree. To follow is a description of how the process strategy tree was created and structured for use with the DOM DLL to provide an easy-to-use platform for simulation.

6.3.2.1 Connecting to the DLL

The process strategy tree for the DLL consists of a strategy node, and an algorithm node containing the controlling information. Nodes can be thought of as branches in the tree structure of the xml files. The algorithm node connects to the DLL's class as can be seen below in the diagram (PSTest81B.TestClass).

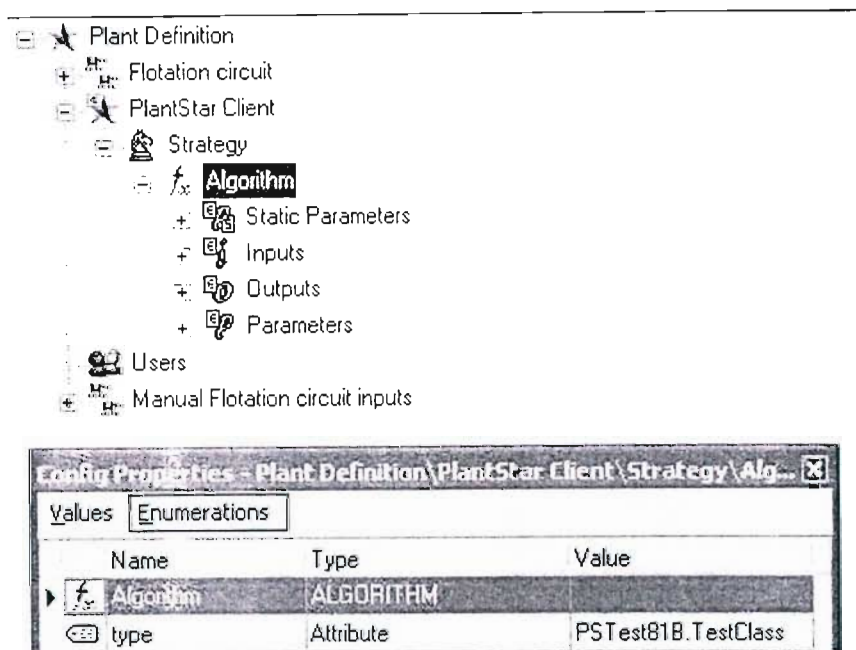


Figure 6-4: The Connection of PlantStar to the DLL class through the algorithm

The algorithm node contains all of the variables and unit's information in its tree structure that is passed backwards and forwards between PlantStar and the model class through the DLL. The algorithm contains four sections or independent nodes that carry certain information to the DLL, namely the static parameters node, the inputs node, the outputs node and the parameters node. The static parameters node contains information that is passed only once to the DLL, on initialisation. This information cannot be changed during a simulation as the initialise member function in the DLL is called only once, at start-up. The Inputs node contains information relating to the specific process units, such as the cells and mills and

receives this information off the plants measuring instrumentation. The outputs node receives the model outputs and predictions from the extended Kalman Filter. The parameters node passes the set of observed variables to the model along with solution and tuning parameters amongst others. The Inputs, Outputs and Parameters variables, which will be discussed shortly, are passed to the DLL on every step or execution of PlantStar, and can be modified during the simulation. These nodes are passed through to the DLL in the execute member function. The execute member function is the function inside the DLL which is called on every execution by PlantStar in order to update the simulation. All variables passed to and from the plant definition file, must be declared and have the correct array sizes in the DLL's member functions, for the simulation to execute.

6.3.2.2 Static Parameters

The set of static parameters is passed to the DLL only once, on start-up. These parameters include switches for the model modes and start-up options for the model.

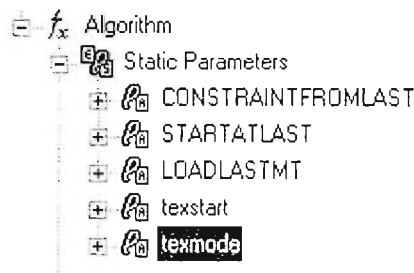


Figure 6-5: The Static parameter node from the plant definition file.

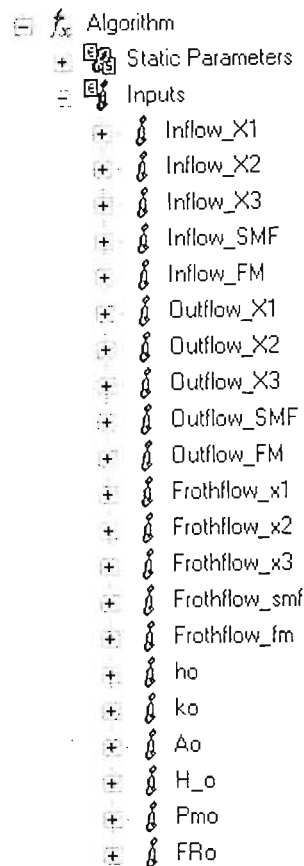
Above is an image of the static parameters node from the plant definition file. The static parameters have the following functions:

- CONSTRAINTFROMLAST: When this variable or flag is “true”, the model uses a set of constraints on variables (allowed ranges) from a previous simulation, that were saved to a file named “LASTCONSTRAINTS.bin”, as the initial set of constraints.
- STARTATLAST: If this flag is “true” the model uses the set of plant variables from the final time step of a previously saved simulation. In this way if the model was well converged when the data set was saved from the previous run, the model will start at this location and will reach steady state sooner. The data set used is saved to the file named “LASTVAR.bin”.

- LOADLASTMT: When set “true” the model uses a saved covariance matrix for the extended Kalman filter solution. If the model is well converged for a specific set of observations on a previous simulation, and the data saved to the file “LASTMT.bin”, the model will use the saved covariance matrix resulting in a faster convergence when changes are made.
- Texstart: This is the time input to the model when running off historical data, giving the model the position that it should start at, in the historical data.
- Texmode: This flag sets the operational modes that the model can use, which are REAL-TIME, HISTORICAL and FORWARD RUN. The historical mode allows for the simulation to run off historical data saved from the plant, which then allows changes to be made by operators in order to check “what if scenarios”. Real-time mode simulates the plant on every time step and tries to keep all the variables as close to actual plant data, inferring unknown process data around the plant. Forward run gives the time-variation of the plant up to a specified time in the future, using the last available plant measurements on each time step.

6.3.2.3 Inputs to the DLL

The inputs to the DLL are actually made up of two sections, the inputs section and the parameters section. Both of these sets of variables are passed to the DLL on every execution step. The parameters set will, however, be discussed in a later section. The inputs node of the plant definition file contains all the available plant variables, such as airflow rates, pulp levels, mass flowrates, mill power etc. Some other variables that are inferred by the extended Kalman filter such as species mass fractions and flotation rate constants are also made available in this data set, in the event that the operators acquire plant information through other means such as assays, or lab analyses, and can then enter them into the model’s solution. If a measurement that is normally inferred by the Kalman filter does become available it could be used to update the model and these variables are therefore made available in the inputs section in order to accommodate this.



Config Properties - Plant Definition\PlantStar Client\Strategy\Algorit...

Values Enumerations

Type	Value
INPUT	
Attribute	Inflow_X1
Attribute	1.1
REFERENCED_ARRAY	Plant Definition\Flotation circuit*\Inflow_X1\Value

Figure 6-6: The inputs node of the plant definition file and properties of Inflow_X1

Figure 6-6 shows that the inputs node is made up of referenced arrays for each of the process variables used to define each of the process units. The referenced arrays allow for easy addition or removal of process units, because the process variables for that process unit are automatically included into the data array sent to the model. The referenced arrays search for some defined criteria in the plant definition file and place each value found into the referenced array in the order in which it is found. In this way the array is built up automatically and process units can be added and removed easily. A more detailed description of the referenced arrays and their use in terms of the model can be found in Appendix F.

6.3.2.4 Outputs from the DLL

The models outputs and predictions for all the plant variables are returned from the model to the plant definition files outputs node. Each of the process unit's extended Kalman filter estimates of the process variables are displayed in this section of the process strategy tree. The outputs node holds process variables in a different way to the inputs section. To make the tree structure more readable the DLL passes out separate arrays for each of the process units, containing all of the process variables relevant to that unit. In this way the user can expand the tree structure for a specific process unit and view all relevant data for that unit. Using this structure for output arrays required some rearrangement of data arrays inside the DLL as the model class uses a different set of data arrays. A more detailed look at this process can be found in Appendix F.

6.3.2.5 Parameters

An extended Kalman filter requires a set of process measurements from the plant so that it can minimise the error in the predictions and bring the predicted values closer to the actual plant values. With such a large and complex system, the variables that the extended Kalman filter uses for this purpose may be changed. The model uses an observation set, that tells the filter which variables are currently being observed and must be driven towards their set value received from the inputs section. This observation set is made up from the unit's observation arrays found in the parameters node. Each of the variables in these arrays are simply flags that are set to true (1) or false (0). If a process variable is observed then the filter will have a high confidence in the value obtained from the corresponding input variable as opposed to the value for that variable from the previous model steps output, and will therefore drive the model towards this value while the rest of the process variables find a new steady state position based upon these observed variables. The model tuning parameters are also found in this node in the form of an array containing pertinent tuning parameters for adapting the model's behaviour to differing plant conditions, thus allowing for changes in the solution criteria of the process. The "model speed up factor" is one such model tuning parameter and allows the model to be run forward faster than real-time. This function only has use when the model is not in REAL-TIME mode. When using this function, changes made to the plant can be followed and their affects observed quicker than on the plant. This could facilitate prompt corrections to the plant before the disturbances resulting from the changes have an effect.

6.3.3 GRAPHING THE OUTPUTS

The DLL returns all of the outputs generated by the extended Kalman filter to PlantStar. There is a lot of data available from the model, therefore to make it more readable and to observe the effects of the changes made on the simulated plant, all the variables handled by the model are plotted. In this way, trends can be viewed and responses to step tests and other scenarios can be analysed. Below is a table of all the graphs that are plotted by PlantStar.

Table 6-2: Table of the graphs and variables plotted by PlantStar

Variable Plotted	Description
Inflow_X1	Platinum mass fraction entering cell. Units 1-14
Inflow_X2	Chrome mass fraction entering cell. Units 1-14
Inflow_X3	Gangue mass fraction entering cell. Units 1-14
Inflow_SMF	Solids mass fraction entering unit. Units 1-14
Inflow_FM	Minerals mass flowrate entering unit (t/h) Units 1-14
Inflow_FL	Liquid mass flowrate entering unit (t/h) Units
Outflow_X1	Platinum mass fraction leaving unit. Units 1-23
Outflow_X2	Chrome mass fraction leaving unit. Units 1-23
Outflow_X3	Gangue mass fraction leaving unit. Units 1-23
Outflow_SMF	Solids mass fractions in the tailings. Units 1-23
Outflow_FM	Mass flowrate of material leaving the unit (t/h) Units 1-23
Outflow_FL	Liquid Mass flowrate leaving the unit (t/h) Units 1-23
Frothflow_x1	Platinum mass fraction leaving in the concentrate. Units 1-10
Frothflow_x2	Chrome mass fraction leaving in the concentrate. Units 1-10
Frothflow_x3	Gangue mass fraction leaving in the concentrate. Units 1-10
Frothflow_smf	Solids mass fraction of concentrate flows. Units 1-10
Frothflow_fm	Concentrate mass flowrates (t/h) Units 1-10
Frothflow_fl	Liquid mass flowrate in concentrate (t/h) Units 1-10
H_o	Pulp Height (m) Units 1-14
ho	Froth depth (m) Units 1-10
Ao	Airflow rate. Units 1-10
Pmo	Mill power. Units 11-12
Cell factor	Mass Pull factor for cells. Units 1-10
FR	Combined tailings flowrate from units. Units 1-23
k1	Platinum flotation rate constants. Units 1-10
k2	Chrome flotation rate constants. Units 1-10
k3	Gangue flotation rate constants. Units 1-10

k4	Water flotation rate constants. Units 1-10
Errors	Indication of the performance of the model solution

Figure 6-7 shows an example of one of the output graphs from PlantStar.

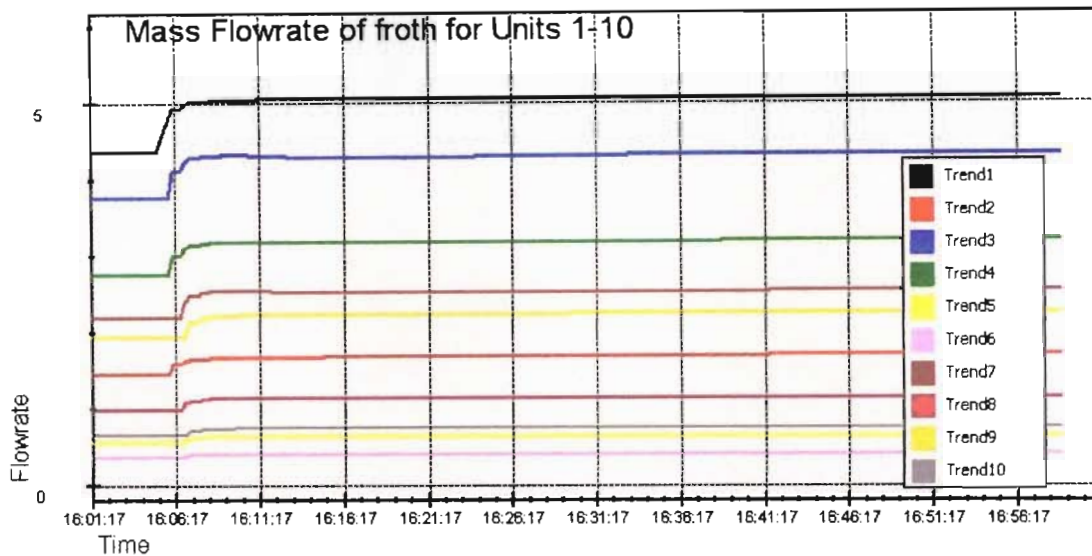


Figure 6-7: The mass flowrate of material leaving the cells in the concentrate (Frothflow_fm) after a step in mass pull of 15% for all banks (The trends number corresponds to the unit numbers which is consistent for all graphs unless otherwise indicated.)

The graphs are displayed in colour along with a coloured key to match the trend to the variables. These graphs can then be exported as image files from within PlantStar. The data can be saved to a database system by a “plugin” module of PlantStar’s known as mInfoStar, however, the dynamic observer model is too processor intensive to allow all of the variables to be saved in real time. All results obtained have therefore been saved by exporting the graphs to file and then viewed as an image.

6.3.4 DEBUGGING THE SYSTEM

When starting up PlantStar, an indication is given of what it has done and what is running, in a “Responses” window. This window tells the operator the status of strategies, whether true or false, for running, whether or not real time data storage is operational etc. This is best

depicted by an image of the responses window showing the responses after the dynamic observer model has been loaded and is running.

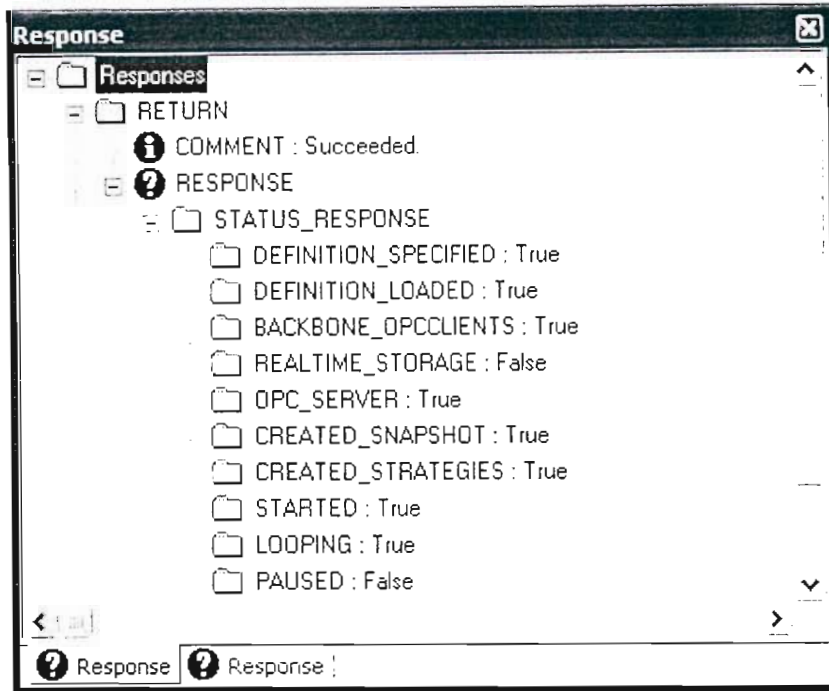


Figure 6-8: The responses window from PlantStar used for debugging

As can be seen, most of the responses are true, which bodes well when debugging the DLL and the plant definition file. If there were any errors with the system they would be displayed in this window with a small comment as to what the problem is. The main problems found when developing the system were variable arrays and their indexing within the plant definition file being incorrect. This resulted in PlantStar being unable to connect to the DLL and hence the system would not function.

6.3.5 THE MODEL EXECUTION FLOWSHEET

Below is a flowsheet of the completed dynamic observer model DLL execution through PlantStar. The parameter $n \geq 1$ allows the model to subdivide a given execution interval Δt to satisfy the maximum integration time-step size specified by the user.

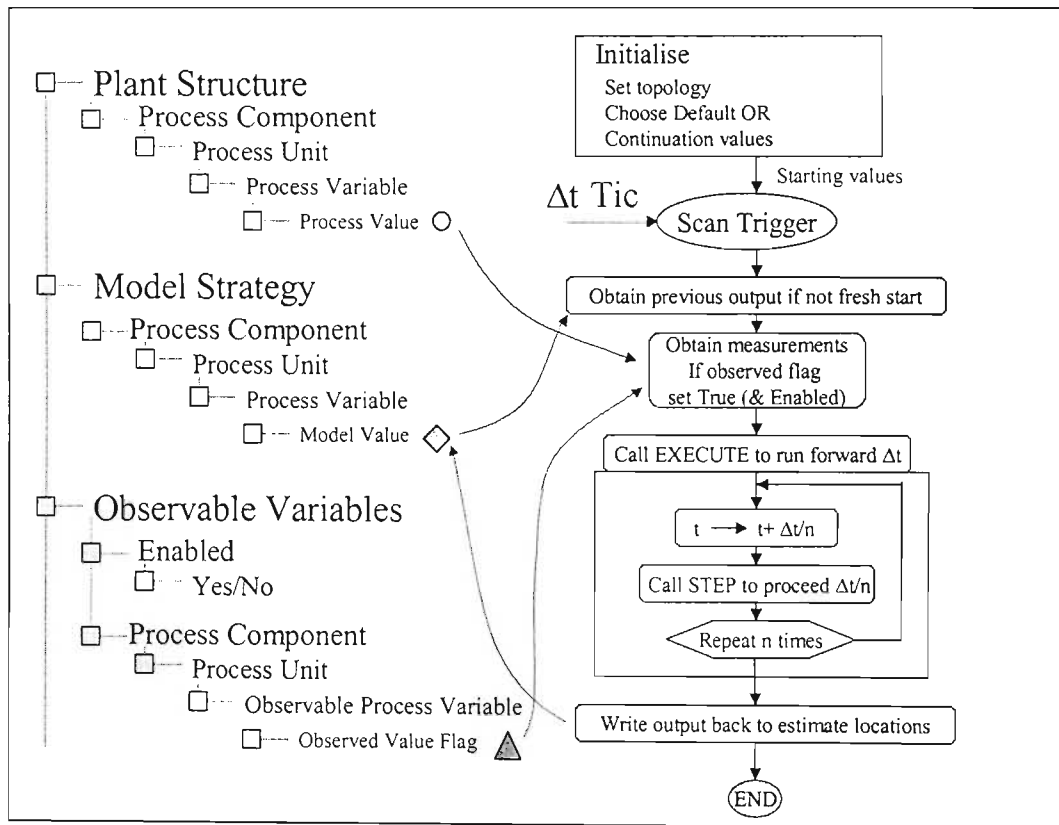


Figure 6-9: The execution loop of the Dynamic observer model through PlantStar.

6.4 USER INTERFACES AND MIMICS

A few of the figures earlier in the chapter show how the process tree structure contains all relevant information about the flotation circuit and its conditions. However this type of tree structure is difficult to understand, and use, without an intimate knowledge of the system and how it is set up. To make the simulator more user-friendly and easy to understand, user interfaces were created that depict the plant in a more readable fashion and display relevant information to an operator which might not necessarily have prior knowledge of the PlantStar system. The complex flotation circuit at LONMIN eastern platinum B-stream has many process units making the plant diagram (figure 4-1) very complicated. There is also a large amount of information available about all of the process units through the dynamic observer model, which could not possibly be displayed on a single plant diagram. Therefore the approach to the graphical user interfaces (GUI's) was to split the overall plant up into main four sections, with the detailed process unit's GUI's linked to the corresponding main section. The four sections are as follows, the primary mill, the primary flotation circuit, the secondary mill and the secondary flotation circuit. Some of the stream properties of pertinent streams are displayed on these four interfaces with the bulk of the information displayed on individual user interfaces made for each of the process units. Images of these GUI's were captured and some will be displayed here while the complete set of all the GUI's is available in Appendix I.

The information displayed on the user interface in figure 6-10 is the mill power, the exit stream conditions from the primary surge tanks and the plant feed conditions. On the bottom right hand corner are five buttons that take the user to the indicated interfaces. This makes navigation around the user interfaces quick and simple.

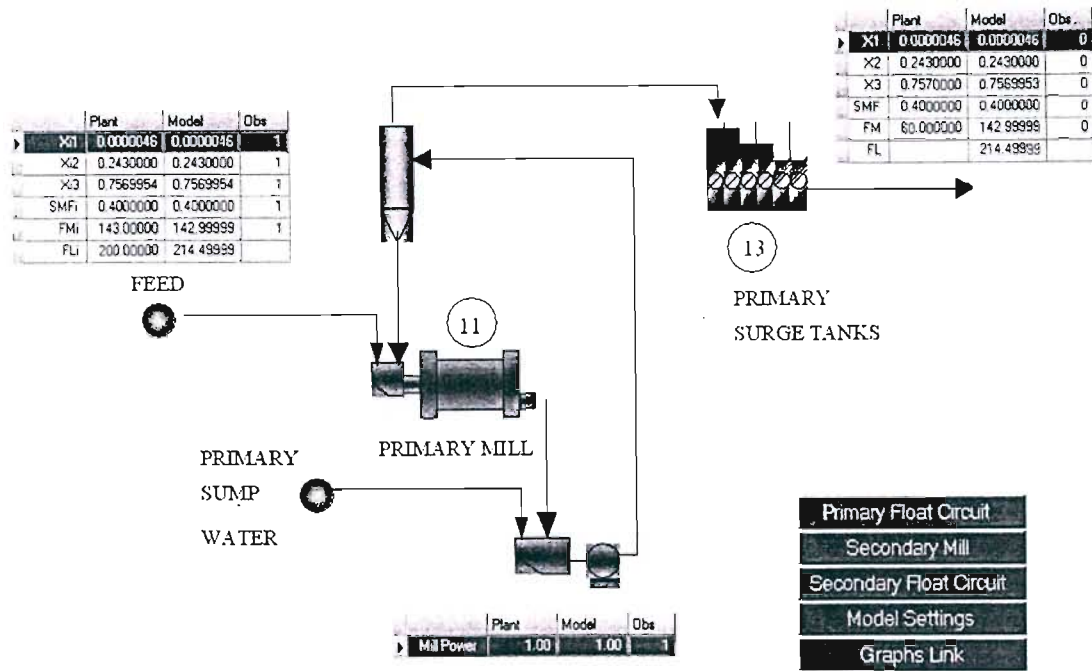


Figure 6-10: The user interface for the primary mill

The information displayed in the user interfaces is set up in such a way that the plant's current position or state, obtained from a measurement device or manually set as an input, is displayed next to the dynamic observer model's predicted value. The third column displays whether or not the model is currently observing the variable. The plant inputs and the observation status can be changed directly on these user interface windows by simply clicking on the variable and editing it.

	Plant	Model	Obs
X1	0.0000046	0.0000046	1
X2	0.2430000	0.2430000	1
X3	0.7569954	0.7569954	1
SMFi	0.4000000	0.4000000	1
FMi	143.00000	142.99999	1
FLi	200.00000	214.49999	

Figure 6-11: The feed condition to the simulated plant.

Looking at the feed conditions to the plant from figure 6-11, it can be seen that the model values closely represent the plant values because they are currently being observed as an input to the model.

The primary flotation circuit is made up of two primary roughers and three cleaners. Its user interface is shown below.

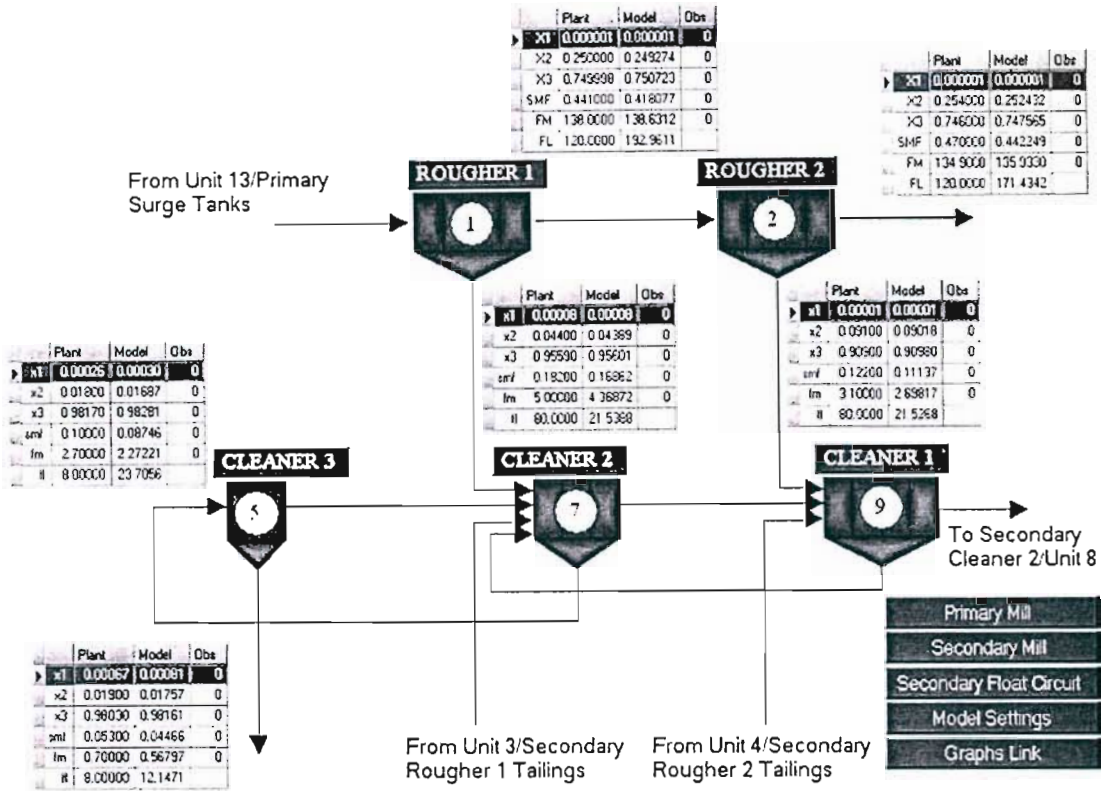


Figure 6-12: The primary flotation circuit user interface

Each of the flotation cell bank names, are link buttons to the individual unit's user interface, where all of the stream properties and process variables are available. For example, unit 1 or primary rougher 1's detailed GUI is displayed in figure 6-13.

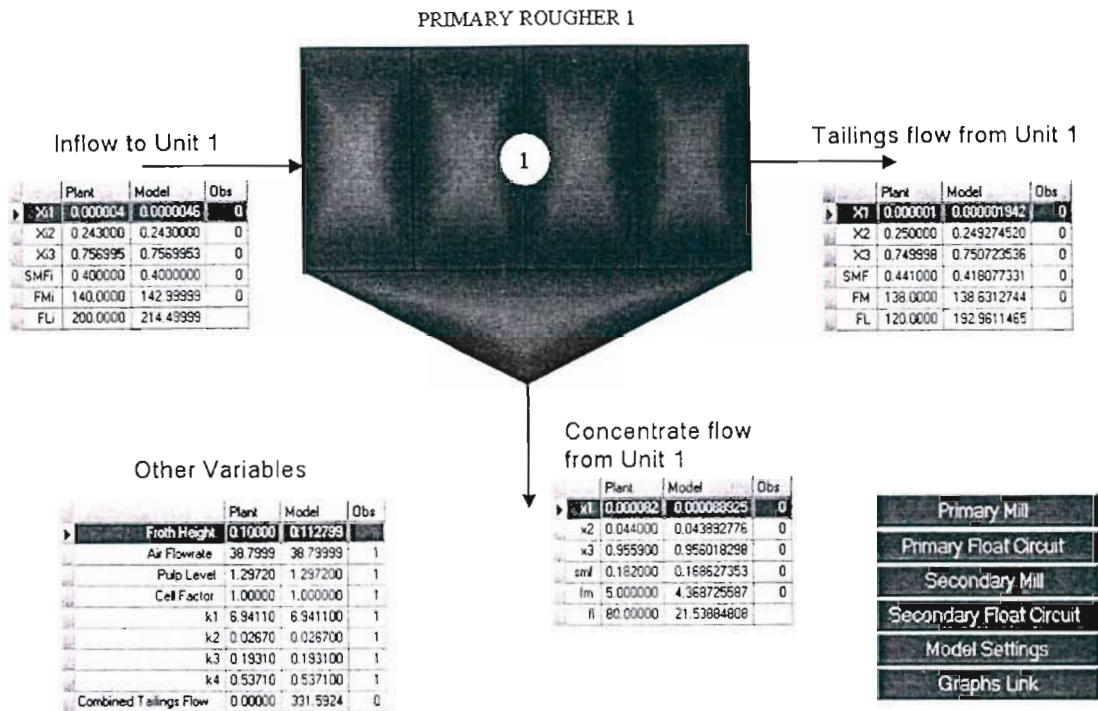


Figure 6-13: The detailed user interface of the primary Rougher 1

The “other variables” contains information other than stream properties, such as pulp height, airflow rate, pulp level, and the effective flotation rate constants.

Other Variables

	Plant	Model	Obs
Froth Height	0.10000	0.112799	0
Air Flowrate	38.7999	38.79999	1
Pulp Level	1.29720	1.297200	1
Cell Factor	1.00000	1.000000	1
k1	6.94110	6.941100	1
k2	0.02670	0.026700	1
k3	0.19310	0.193100	1
k4	0.53710	0.537100	1
Combined Tailings Flow	0.00000	331.5924	0

Figure 6-14: Cell bank properties other than flow rate conditions

The rest of the cell banks user interfaces have the same layout as figure 6-13 and can be found along with the secondary mill and secondary float circuits in Appendix I.

The model parameters user interface however warrants mentioning here. It contains information about the model tuning, operation mode, performance, messages, start-up conditions and shutdown settings.

Model Settings and Tunings parameters

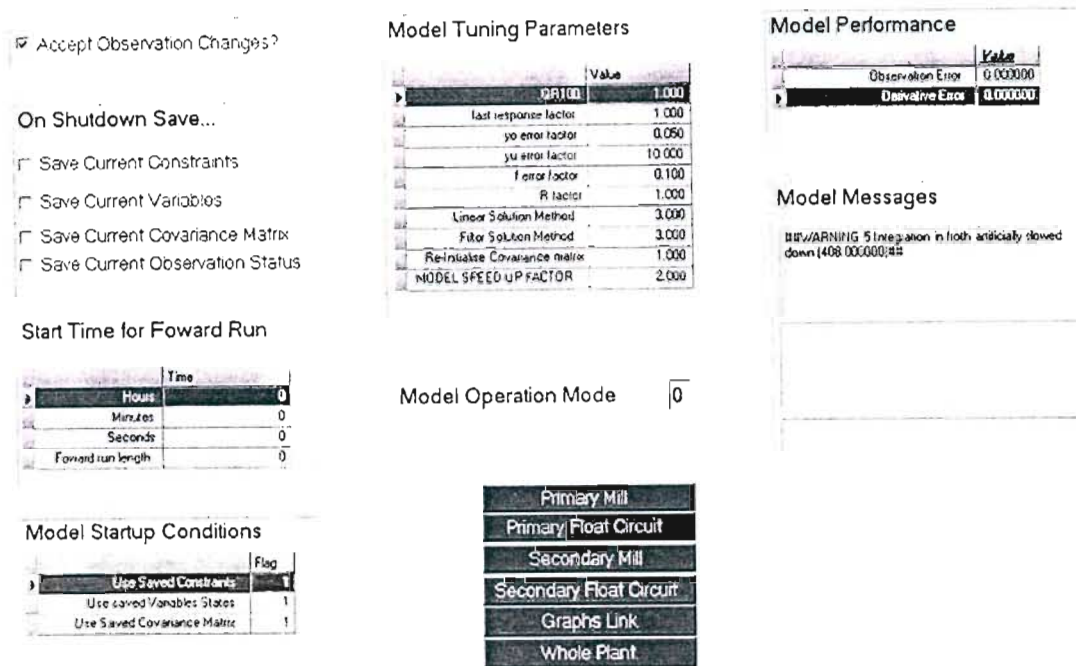


Figure 6-15: Model Parameters user interface

All of the model tuning parameters can be changed during a simulation if need be, however once a good operating set of parameters is found, the need for changing them becomes rare. The messages tell the operator or user about the model's performance, any errors that may have occurred, and generally useful information about the model solution. The model performance gives an indication of how well the solution is converging at any time. The switch for "Accept Observation Changes?" in the top left corner is useful when a large number of variable's observation status is changed at one time. When the switch is in the on position the solution accepts any changes made to the observation set, as they are made. When the switch is off, the solution keeps its last set of observations until the switch is turned on again, when it then accepts the new status of all the observation flags. This feature stops

the solution from trying to converge continuously, while a lot of changes to the observations flags are being made at one time. It will only converge on the new complete set of observation flags once the switch is turned back on. There is also a facility to store Kalman filter covariance matrices appropriate to different combinations of observation flags set “on” if the combinations are found to be repetitive. . These are re-selected when the pattern recurs, thus saving re-convergence time. The above flag thus prevents the generation of intermediate covariance matrices transitional observation patterns. The “Model Operation mode” shows which of the modes the model is operating in, i.e.

MODE 0 : REAL-TIME operation with speed-up factor = texmode 0

MODE 1 : REAL-TIME operation = texmode 1

MODE 2 : HISTORICAL Mode = texmode 2

MODE 3 : RUN-FORWARD Mode = texmode 3

The start-up conditions are also displayed, which indicates what type of start-up was performed, whether from an already converged data set or from a fresh start. There are options to save all the data sets on shutdown as well as settings for the start time of a forward run or historical data run. A detailed description of each of the model tuning parameters can be found in Appendix H.

All of the graphs that were mentioned in the earlier section are available through the graphs links interface, which takes the user directly to the graph of interest by clicking on the corresponding button.

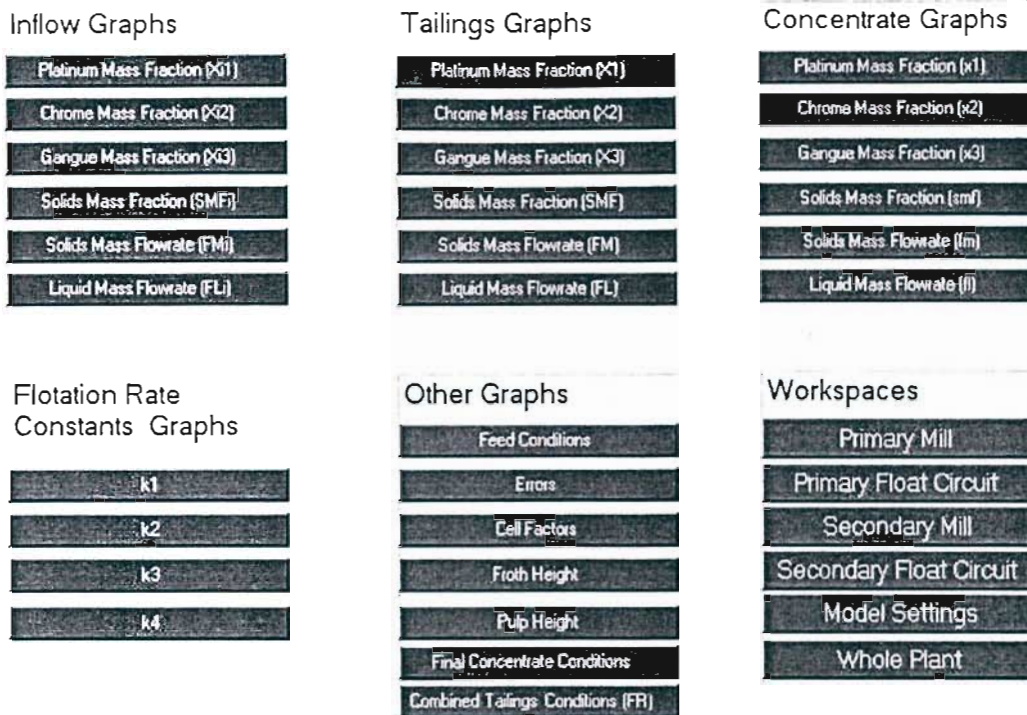


Figure 6-16: The graph links user interface

7 DISCUSSION AND RESULTS

The first chapter described many different techniques of separation used in the mineral processing industry, such as magnetic separators, gravity separators and optical separation. However, for platinum concentration, the technique most widely used and of the most interest to this thesis is froth flotation. Froth flotation is not entirely understood and therefore much research is still being carried out in this field. There are many typical problems faced by industrial flotation circuits that need to be overcome if better performance is to be achieved. The nature of froth flotation results in poor efficiency of the flotation process and therefore recycles are introduced in order to increase the ultimate recovery of valuable materials, however this can result in large internal recycles and mass hold-ups around the circuit. Feed conditions and characteristics can vary greatly, which reduces consistency and makes control difficult. Certain disturbances around the circuit are difficult to detect and can result in reduced circuit performance. The small number of access points on flotation circuits limits the number of meaningful variables that can be measured.

A dynamic observer model can provide information that is normally unavailable to plant operators and would potentially provide a means of improving control strategies on flotation circuits. MINTEK therefore proposed a project whereby a dynamic observer model for the LONMIN Eastern Platinum B-Stream flotation circuit is developed for use with their new expert system. Their objectives for the dynamic observer model were to explain plant behaviour, use the model for simulation and playback, and to provide a better insight into the process.

Many different flotation modelling strategies, such as kinetic rate models, collision models and probability models, were reviewed in chapter 2, each having specific advantages and disadvantages. Other factors researched by previous investigators were: flotation rate order, sub-systems in the froth, froth residence time, gangue entrainment, froth depth, chemical factors, particle size and air addition rate amongst others. However, it was deemed necessary to keep the flotation modelling for the dynamic observer model as simple as possible due to the already large complexity of the circuit layout and therefore of the system of equations describing it. As a result, the first order kinetic rate flotation model was used to determine the concentrate mass flowrates in the dynamic observer model.

As discussed in section 4.3, the kinetic flotation rate constant used in the froth modelling of the dynamic observer model takes into account the air addition rate, the mill power and the froth depth as well as a cell factor, which is used to model the effect of increased mass pull on the cells. The air addition, mill power and froth depth effects were considered as a fraction of their standard operating conditions. The concentrate mass recovery was based solely on the first order rate equation, however the flotation rate constant in this equation (equation 2-9) was replaced by the calculated *effective* flotation rate constant (equation 4-1), allowing for greater accuracy of the flotation rate constant during regression. A better model fit can be achieved by taking into account relevant plant data that is available.

The method of solution of the system of Differential and Algebraic equations, resulting from this type of flowsheet modelling of a mineral processing circuit, an Extended Kalman Filter, will be described in detail in Appendix J. This type of solution is very mathematically intensive as mentioned in section 6.2.3, due to the very large linear systems (398 variables and 163 equations) that must be solved and manipulated in order to find the next time step's solution. As a result, this poses the problem of large processor time usage, which becomes limiting when using the model in a real-time scenario. The three methods of solution that were made available for the model described in section 6.2.3 were an attempt to address this issue. The current solution technique is fast enough to be used for real-time updating, but uses a licensed mathematical software package, "MATLAB" to find the model solution. This results in a portability problem with the model, as any new installation of the model would require a licensed copy of MATLAB to function properly. This issue was discussed in Appendix E, but more research into finding an efficient solution technique for large linear systems is required in order to make the model more portable. An option would be to model each of the units individually, using an extended Kalman filter, and then creating the plant topology by means of interconnection matrices. This would result in less computational time, as the matrices inverted would have smaller dimensions. More than one inversion could also be carried out at a time, using separate processing threads.

The dynamic observer model structure has been described in detail in section 4.3, but the key features will be reiterated here. The flotation circuit that the model describes has 10 flotation banks consisting of various numbers of cells in each bank. In order to minimise the number of equations required to describe the system it was decided to treat each of the banks of cells as an individual cell. This type of assumption is common amongst the approaches of many researches (Yingling, 1993) and reduces the number of cells to be considered. Four species of material were considered within the feed material, these being gangue, chrome, platinum and

water, each having its own specific flotation rate. However, the model is not limited to these four species and can accommodate any number of species should it be necessary to describe the process more accurately. The flotation cells were modelled as perfectly mixed tanks with material reporting to the concentrate determined by the linear first order flotation rate equation operating on the total mass of the species in the pulp phase, with the remainder of the material passing out through the tailings. Mills and Surge tanks were modelled as stirred vessels having a residence time, whilst spirals and settling tanks were modelled with splits to route different fractions of the incoming species to different exit streams.

The dynamic observer model created for this project has many features, as described in section 4.5 but its main use is the online inference of unmeasurable variables that are made available by the extended Kalman filter. A set of observation flags is available to the user, allowing him to select which of the plant variables are currently being observed by the model, in other words which of the variables are actual plant measurements or user inputs. The extended Kalman filter then tries to force the model variables to equal those being observed, whilst balancing the remaining system of equations so as to achieve a best fit for all of the variables. The dynamic observer model has three modes of operation. Firstly there is the standard operating mode which would be most commonly used, that is the real-time mode in which the model uses available plant data to predict all plant variables in real-time and constantly updates itself to keep in line with actual plant measurements. Secondly, a “Run-Forward” mode for the prediction of the future state of the plant given a set of changes made to the model, and thirdly an historical mode in which saved plant data can be used to simulate “what if” scenarios without interrupting the current state and performance of the actual plant.

The dynamic observer model was implemented on the control platform which is available at the plant and which was created and is maintained by MINTEK, called PlantStar®. The dynamic observer model was integrated into a dynamic link library (DLL), which connects to a plant definition file created for PlantStar describing the entire plant topology. The model receives all plant variables and solution parameters through the DLL, which in turn receives them from PlantStar, which is connected to the plant via some form of low-level SCADA or PLC system (Chapter 5). A detailed description of how the DLL and the accompanying plant definition files were created, and their features can be found in chapter 6. PlantStar plots all of the variables describing the circuit that the model simulates in real-time, and a user-friendly graphical interface was created to describe this circuit and make available the variables that affect the model solution. The complete set of GUI interfaces can be found in Appendix I.

One of the difficulties experienced whilst creating the dynamic link library of the model for PlantStar, concerned the control platform itself, as it was constantly being upgraded and updated to meet the changing needs of the industry. Three different versions of PlantStar were used during the project, but fortunately the changes to the model DLL were insignificant. However, on a few occasions the plant definition files had to be recreated to function properly with the newer versions of PlantStar. The software versions used for the model DLL creation and implementation were all *beta* releases of PlantStar, and help was required from the MINTEK staff in order to keep up to date with any nuances or special treatment required in order to keep the program operational. The hierarchical method of describing the plant topology using plant definition files provided by PlantStar had to be learnt, and much time was spent discovering how to create them and implement simple DLL's into them before any attempt could be made at creating the full simulator of the LONMIN B-Stream circuit. As such, much trial and error work was carried out before a good understanding of the system was established and the B-Stream circuit could be created.

Several versions of the DLL itself were also tested with a program that was created specifically for this purpose, as PlantStar's connections to the DLL and start-up time are fairly lengthy. A detailed description of this testing program can be found in section 6.2.6, along with the source code in Appendix D. This testing program facilitated quicker debugging of the DLL to test if the model was functioning correctly within it, and whether all of the variables were being passed into and out of the model. Some limitations of the Microsoft Visual Basic 6 programming language used to create the testing program were discovered during this phase, as there is a limit to the number of variables that can be passed into and out of a program created in this language (section 6.2.6). The DLL was therefore tested in stages with some variables set statically while others were changed.

Once the model DLL and accompanying plant definition file were created and operating correctly within PlantStar, a series of test runs were carried out to view the performance of the model and the functioning of all the graphs and user interfaces created in PlantStar. As the program is not currently installed at the actual plant, all of the plant measurement data that the model would receive from the plant were static measurement inputs. In normal operation at the plant these values would be constantly changing as the plant's conditions changed, however the changes can be made to the corresponding variables manually in order to simulate plant fluctuations, for example a step in the feed flow rate to the plant or a step in the feed grade. The user would then select the necessary model observation flags to indicate that these values are being observed or are available.

A complete set of steady state plant data was obtained from LONMIN for the B-Stream circuit and was entered into the model as the standard set of operating conditions or values for all of the streams. The model was then run in a simulation mode setting all output streams to these values for all of the unit operations and setting all of them as being observed. The specific flotation rate constants for all the species were then allowed to move in a real-time scenario in order to find their values for this set of operating conditions. Once the model had converged, these flotation rate constants were then compared to a set obtained by dividing the cell inventories into the concentrate component flows. The results are as follows,

Table 7-1: A comparison between the converged set of flotation rate constants obtained using the extended Kalman filter model solution compared to the same flotation rate constants obtained from the cell inventories and the plant data set

Cell	Pt		Cr		Gangue		Water	
	Inventory	Converged	Inventory	Converged	Inventory	Converged	Inventory	Converged
1	7.350516	6.947952	0.028231	0.028372	0.20444	0.205119	0.568751	0.568188
2	0.860514	0.857097	0.032345	0.032019	0.110008	0.109059	0.576175	0.574865
3	2.486321	2.576024	0.016952	0.01674	0.241424	0.23946	0.331882	0.330846
4	1.400577	1.374101	0.028594	0.028336	0.31068	0.307985	0.644204	0.642699
5	3.419507	3.461442	0.613507	0.614024	0.580211	0.569235	1.832672	1.834304
6	1.077626	1.076883	0.385488	0.385513	0.440822	0.442986	1.850441	1.850146
7	15.08853	14.96125	0.39465	0.392563	0.780655	0.805497	2.477287	2.469814
8	3.666391	3.632696	0.585183	0.585429	0.496195	0.496166	1.470885	1.470887
9	10.45202	10.30021	0.762661	0.761588	0.803361	0.809095	0.72504	0.72412
10	1.183269	1.18882	0.272768	0.272889	0.221571	0.221573	0.65489	0.654892

Once this data set of flotation rate constants was converged they were set as the flotation rate constants for the plant. The operating conditions of the plant or plant variables would then be calculated using these flotation rate constants, provided the flotation rate constants are set as being observed. Once steady state operation of the plant has been reached for a different set of plant operating conditions, the flotation rate constants can be re-identified in this way and then set to the new converged values for normal open-loop modelling.

A step test of 30% increase for the feed flowrate was performed and results obtained show the response of each of the plant variables. There are 29 graphs that plot all of the model's output variables in real-time, but many of them remain constant depending upon what type of tests are being carried out. Only the relevant trends that show variable responses will be displayed.

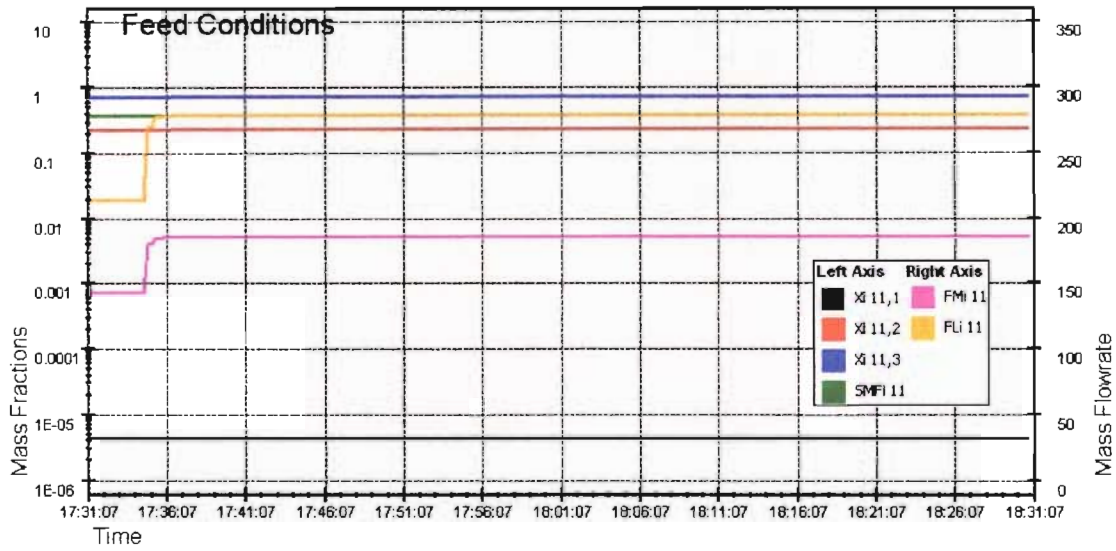


Figure 7-1: A step in the feed flow rate of 30% to the plant (left axis is the mass fractions and the right axis, the mass flowrates)

Figure 7-1 shows the simulated step in the feed flowrate to the plant. As can be seen, only the solid mass flowrate was stepped up, which results in the step in liquid flowrate as the solids mass fraction in the feed remains constant (trends FMi 11 and FLi 11 respectively).

Figure 7-2 shows the effect that the step in feed flowrate had on the tailings flowrates from the roughers (trends 1-4), mills (trends 11 and 13) and surge tanks (trends 12 and 14).

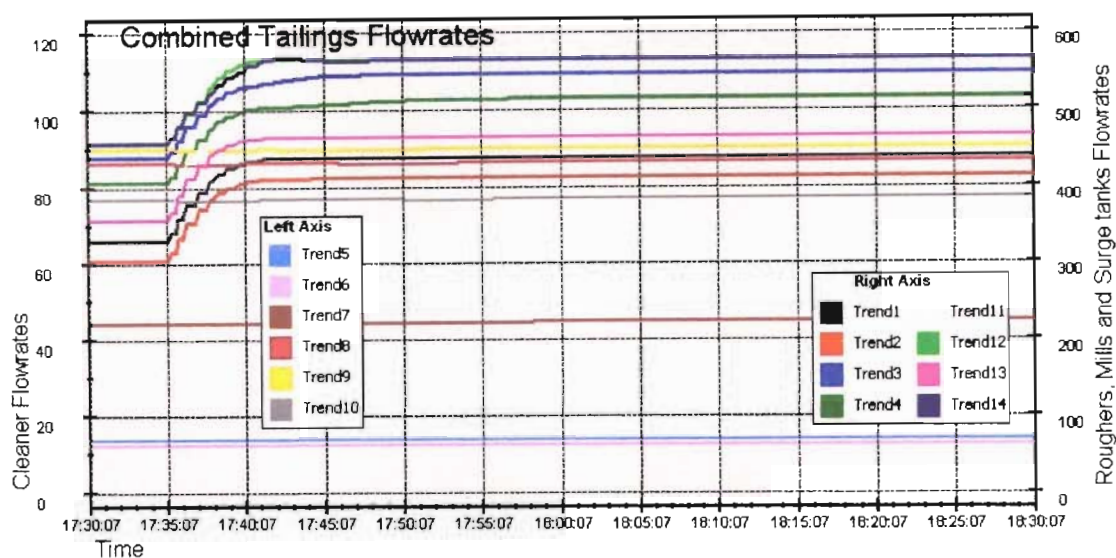


Figure 7-2: Effect of the 30% feed flowrate step on the tailings flows (Left axis is the cleaners and right axis, the roughers, mills and tanks) .

Most of the increased material flow through the circuit passed straight through each of these cells and has effectively just reduced the residence time of material in the cells.

The cleaner cells do not show an increase in their tailings flowrate, as there is little increase to their inflows as these are concentrates. Trends 5-10, in figure 7-2, show the cleaner tailings flowrates, which remain unchanged.

The reduced residence time in the roughers results in a loss of platinum to the tailings as there is less time available to recover the valuable mineral in the cells. This can be seen in figure 7-3, which shows the platinum mass fraction of material leaving the cells in the tailings.

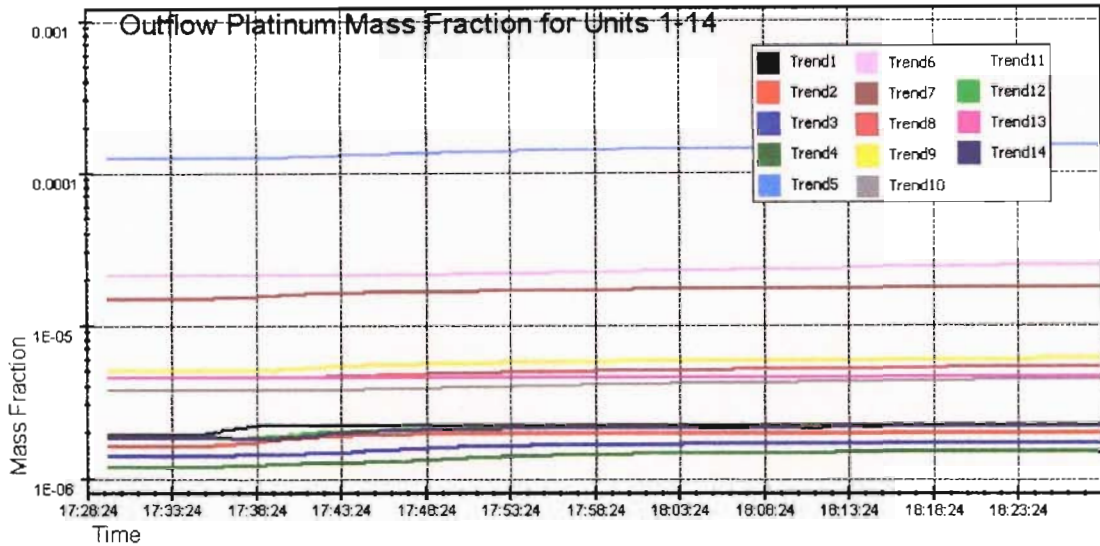


Figure 7-3: The platinum mass fraction leaving in the tailings flows as a result of the reduced residence time in the cells after the increase in feed flowrate.

There is, however, a slight increase in the mass fraction of platinum recovered in the concentrate due to the increase in the amount of platinum in the pulp phase (figure 7-4), but the amount of platinum lost to the tailings is considerably larger than before.

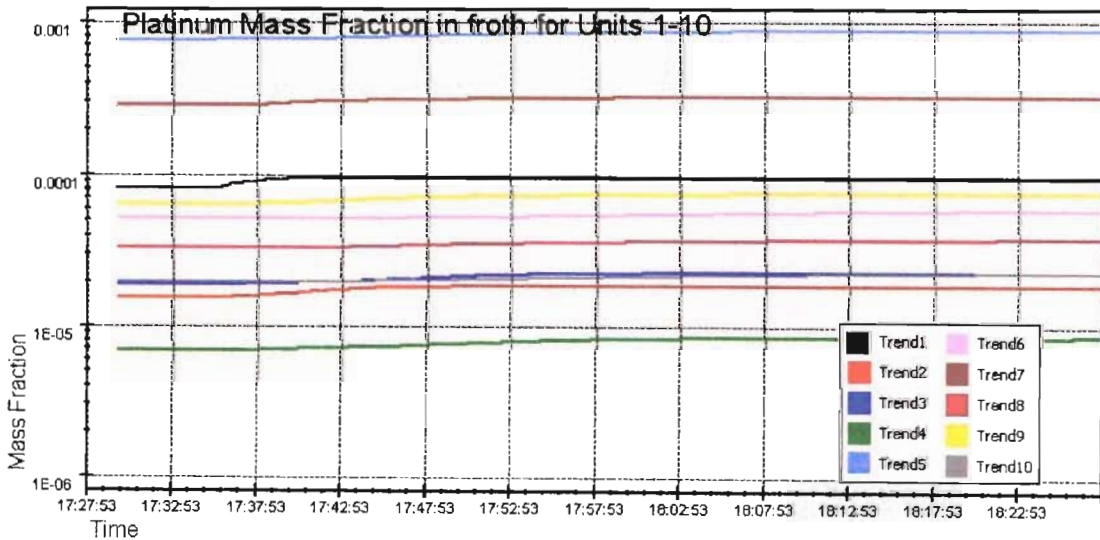


Figure 7-4: The slight increase in the platinum mass fraction in the concentrate after the feed flowrate step.

The mass flowrate of the concentrate should remain constant, as the flotation rates of the material have not changed within the cells, only the throughput of material has increased. This is depicted in figure 7-5, with the obvious exception of trends 3 and 4, which apply to the secondary roughers.

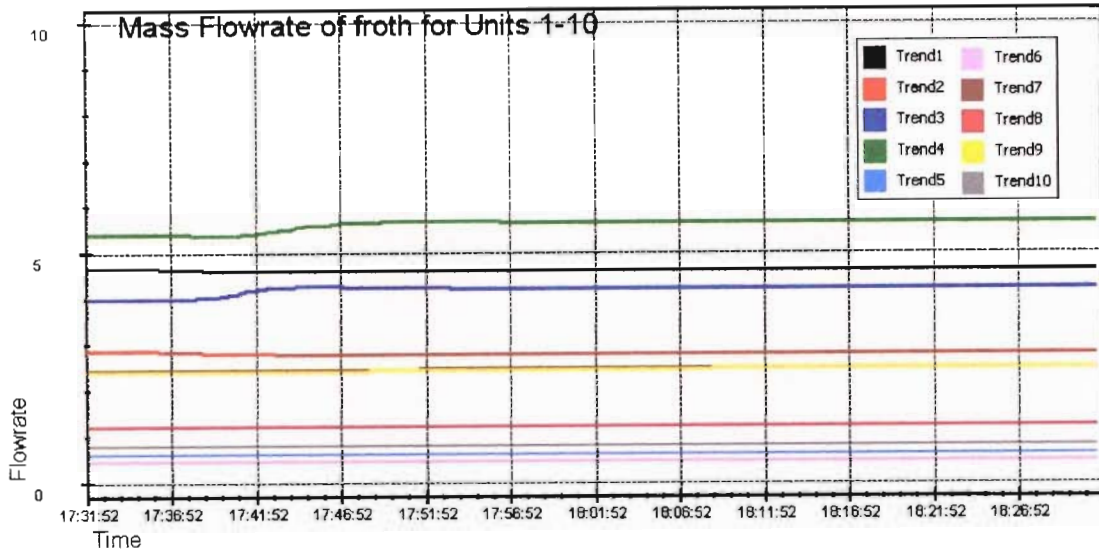


Figure 7-5: The concentrate mass flowrates after the step in feed flow showing the slight increase in flow of concentrate from the secondary roughers.

These show an increase in their concentrate mass flowrates resulting from the increase in the amount of fast floating material passing through the first two roughers as a result of the reduced residence time, and therefore reported to the secondary roughers.

The final combined concentrate and tailings conditions both show an increase in the platinum mass fraction leaving the circuit. Due to the increase in the platinum mass fraction and the constant concentrate mass flowrate, more platinum is recovered. However, an increased loss of platinum to the tailings also occurs due to its increased mass fraction and mass flowrate.

The ratio of the increase in recovered platinum to that of increased platinum lost to the tailings shows that this step would clearly impact on the economics of such a change (figures 7-6 and 7-7).

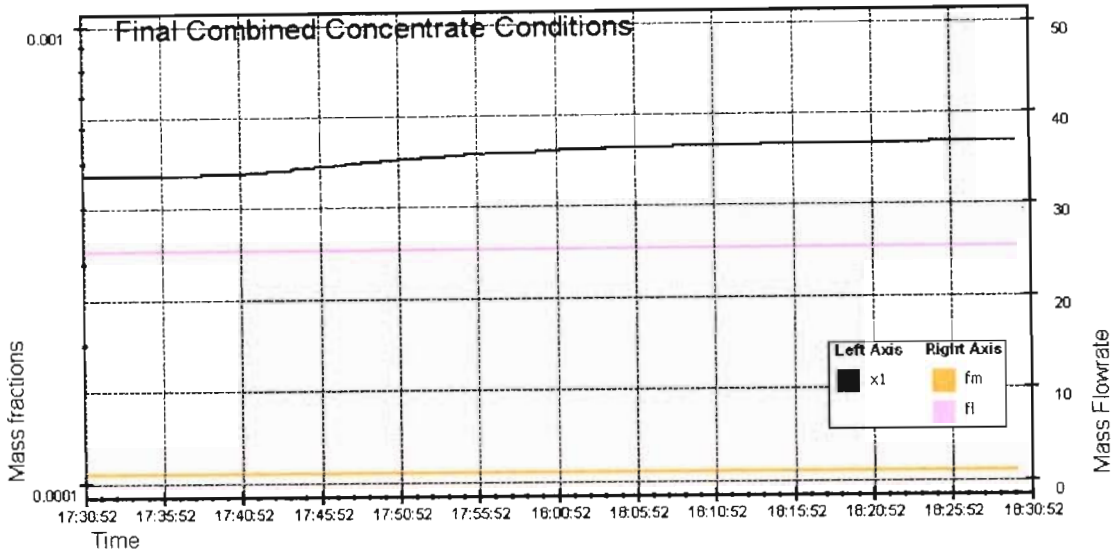


Figure 7-6: The final combined concentrate flows and mass fractions after the step.

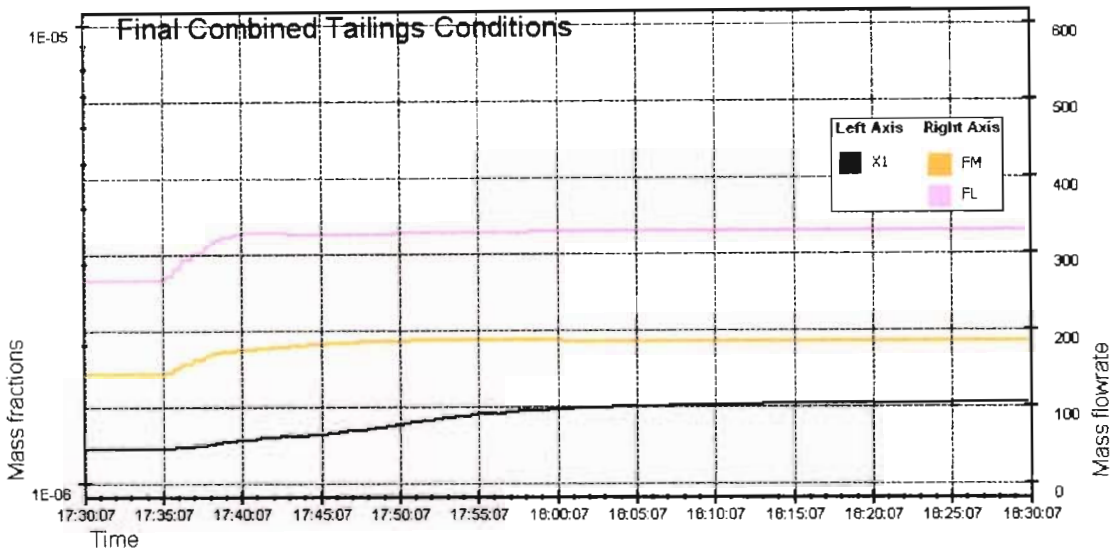


Figure 7-7: The final combined tailings conditions after the feed flow step.

Another test simulation performed was a step test in the cell factors for all the banks to observe their effect, which was introduced to simulate an increase in the mass pull on the cells. These cell factors are observed variables by default, therefore no changes to the observation parameters of the model were required.

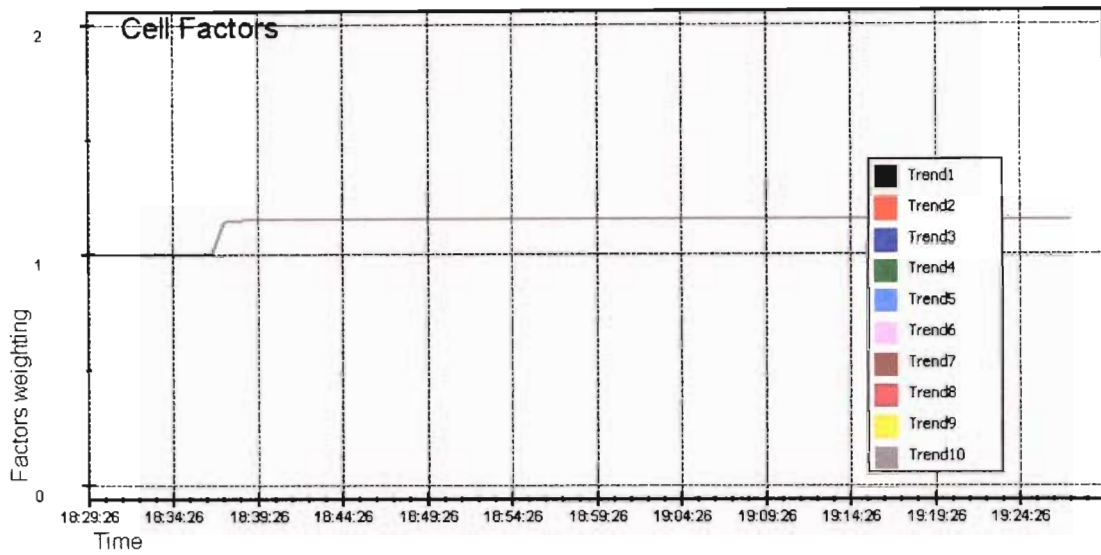


Figure 7-8: The step in cell factors of 15% for all of the cells

Figure 7-8 depicts the step in cell factors and it can be seen that the most immediate effect is the change in the concentrate flowrates (figures 7-9 and 7-10), which is to be expected as the cell factors have a direct influence on the effective flotation rate constant as was discussed in section 4.3.

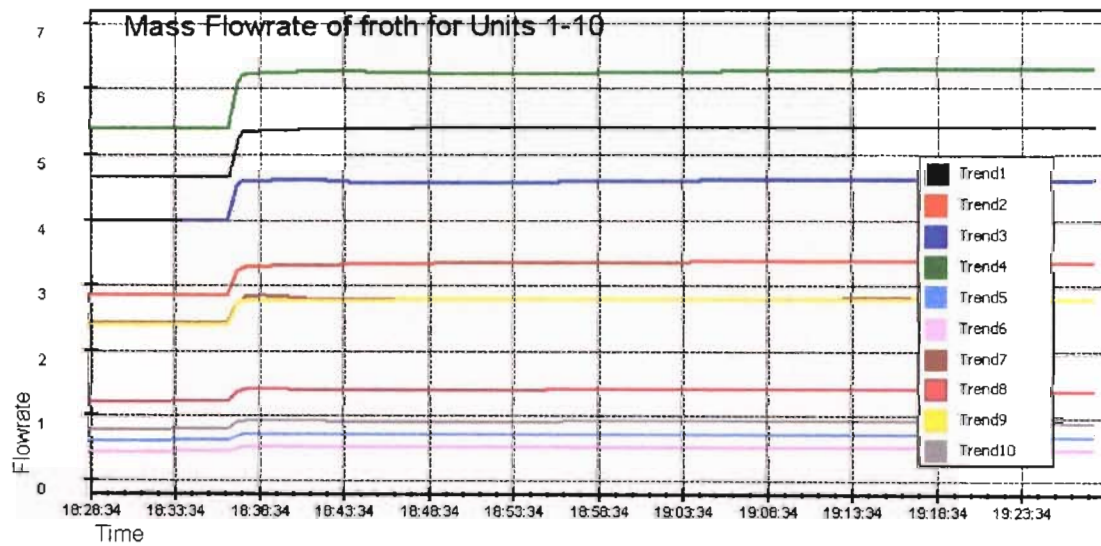


Figure 7-9: Mineral mass flowrate of the concentrates exiting the 10 flotation cells after the step in cell factors.

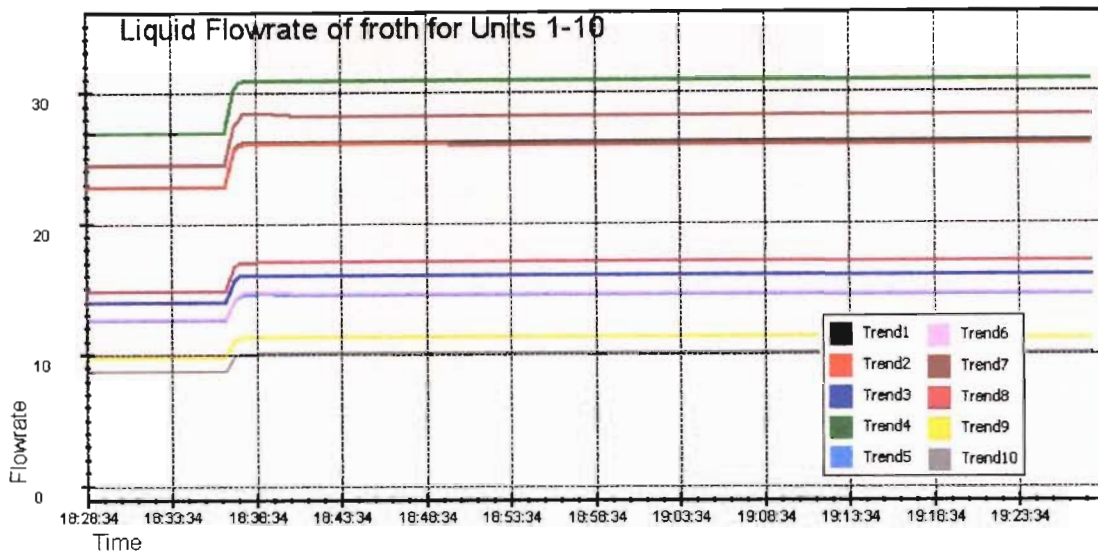


Figure 7-10: The liquid concentrate mass flowrate leaving the 10 flotation cells after the step test.

From figures 7-9 and 7-10, it can be seen that each of the concentrate mass flowrates of minerals and liquid have stepped up by a factor of 15%.

The sudden increase in the concentrate flows should result in a drop in the tailings flowrates due to some of this material now reporting to the concentrates, as can be seen in the following graphs,

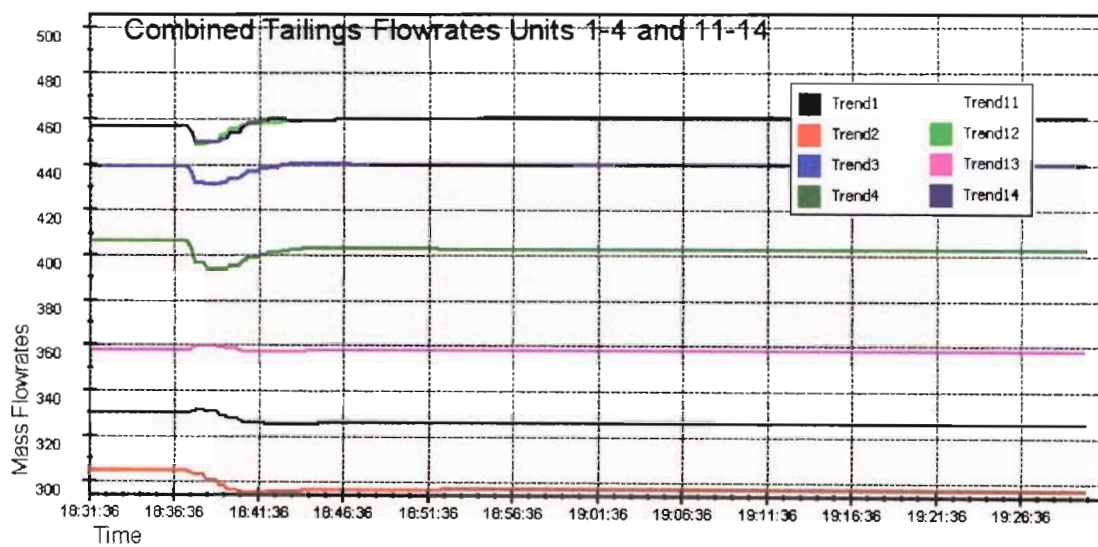


Figure 7-11: The minerals mass flowrates and the liquid mass flowrates combined for the tailings streams of unit operation 1 to 4 and 11 to 14.

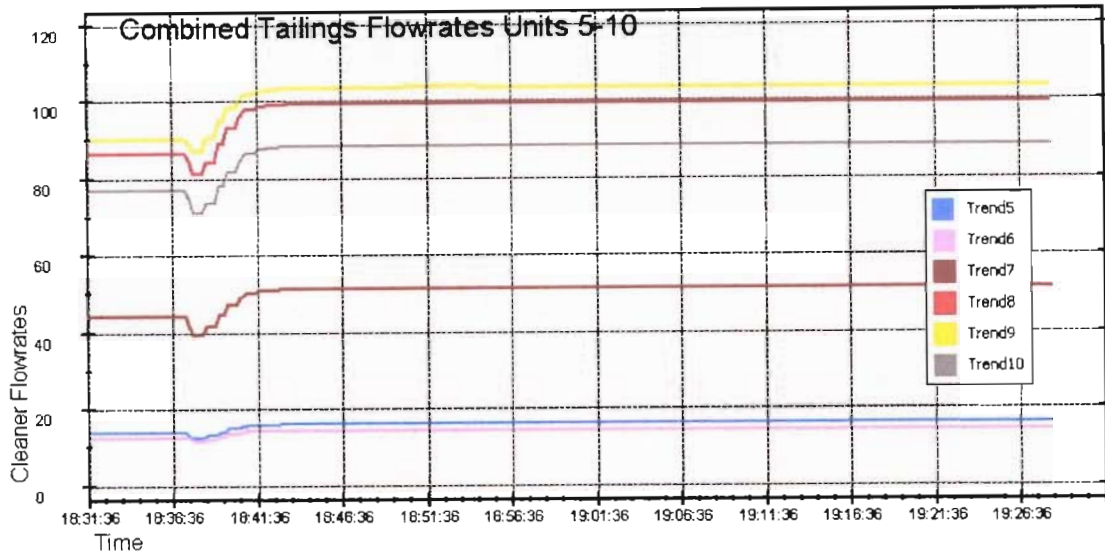


Figure 7-12: The minerals mass flowrates and the liquid mass flowrates combined for the tailings streams of unit operation 5 to 10.

Due to recycles around the system and the fact that some cells are in series, the increased concentrate flow from some of the units results in a far larger than normal mass inflow to other units, resulting in their tailings streams being larger than before, as can be seen in units 5-10 with the tailings flows for the cleaner cells. It can also be seen that initially their tailings flows dropped slightly as in the case of the trends in figure 7-11. The tailings flows dropped slightly and remained lower than before for units 1 and 2, which do not receive any recycle flows (figure 7-11). Units 11 and 13 are largely unaffected by these changes as they are upstream of the step change, as shown in figure 7-11 (trend 11 is largely covered by trend 13 and is therefore not visible because these values are very similar, which is to be expected).

Figures 7-13 and 7-14 show the effect of the increased mass pull on the platinum mass fractions throughout the plant. Considering the effect on the platinum mass fraction in the tailings flows from the cells (figure 7-13), it is clear that the platinum mass fraction would be reduced in the tailings as the flotation process is most selective towards platinum and an increase in the amount of concentrate removed would subsequently increase the amount of platinum removed from the pulp.

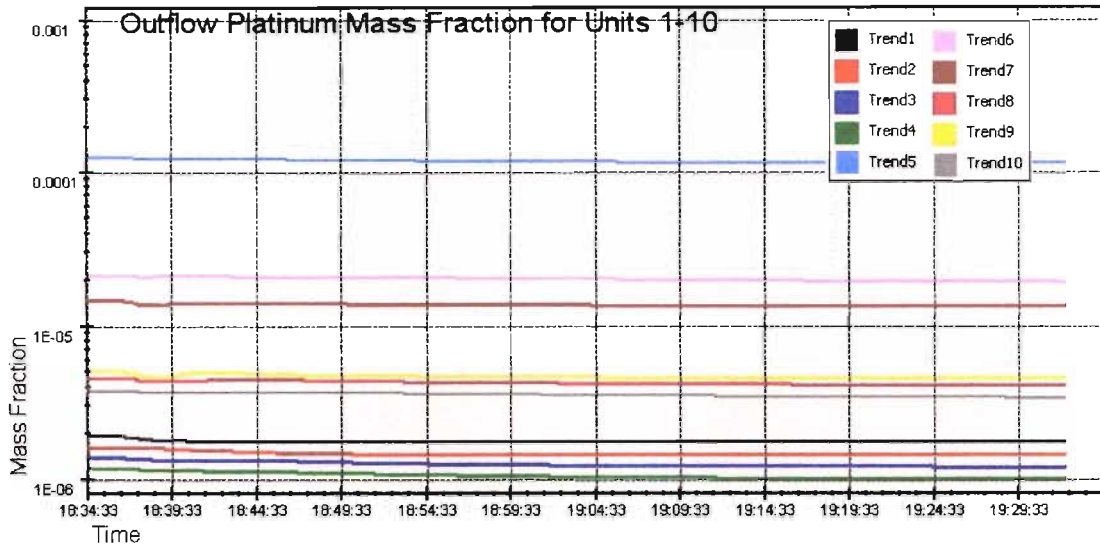


Figure 7-13: This graph shows the reduction in the platinum mass fraction of the tailings flows as a result if the increased concentrate mass flows.

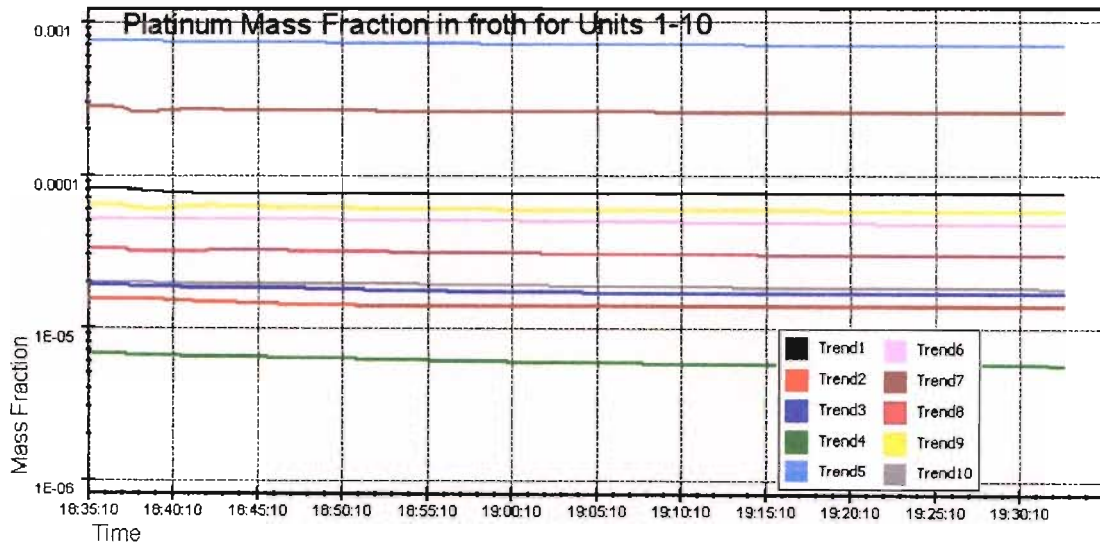


Figure 7-14: Platinum mass fraction in the concentrate flows leaving cells after the cell factor step .

It can be seen in figure 7-14, that the platinum mass fractions in the concentrates have decreased, which is counter-intuitive. However, as the residual amount of platinum in the pulp has dropped by a greater fraction than that of the chrome and gangue fractions, there is less

platinum available to be floated. Therefore even though the actual mass flow of platinum will have increased slightly as a result of the increased mass pull, the overall mass fraction of platinum in the concentrate flow has been reduced because of the proportional increase in the gangue and chrome mass flowrates.

From these graphs it can be concluded that a step in the mass pull of 15% for all the cells results in an increase in the overall amount of platinum recovered, but the grade of the concentrate produced would be lower, which would not result in an increase in plant performance.

The cell factors were then allowed to move (no longer an observed variable in the model) and a step test in the tailings flowrate (new observed variables) was performed to check their converged positions.

The step in the tailings flowrate for the primary cleaner 3 (unit 5) can be seen in figure 7-15. The step size was 50% of the original value so that appreciable responses in the cell factors would result.

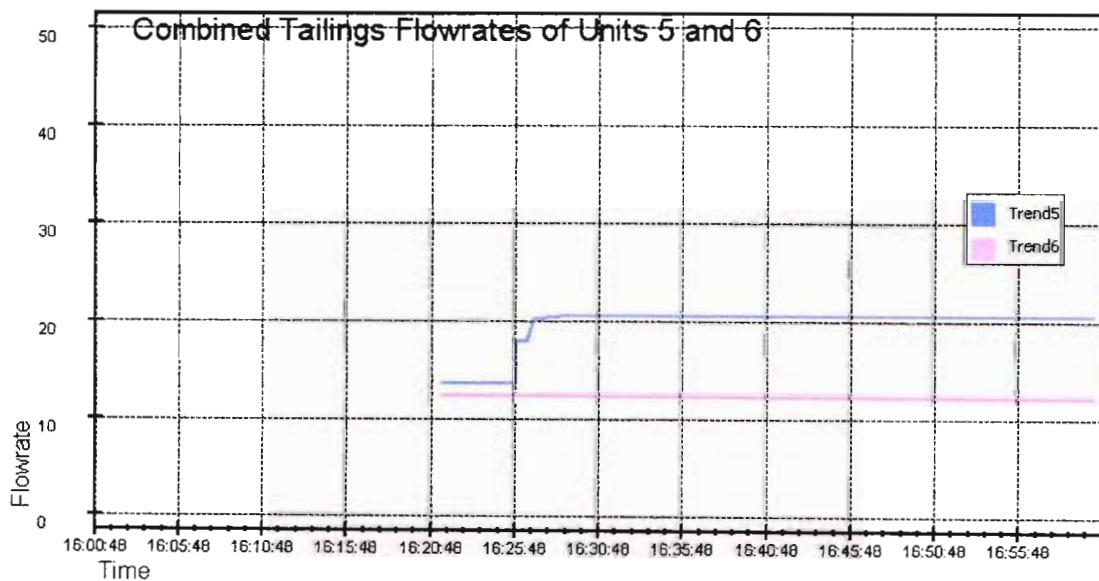


Figure 7-15: The step in the tailings flowrate of 50%

Figure 7-16 shows the effect that this step had on the cell factors and their new converged positions. Unit 5's cell factor initially drops so that less concentrate is removed from the cell,

thereby increasing the tailings flow, whilst unit 7's cell factor increases to produce more concentrate, which in turn is the feed to unit 5. Due to this increase in feed flow to unit 5 the cell factor for unit 5 no longer needs to reduce the concentrate removed to achieve the higher tailings flowrate and it therefore returns to its original position.

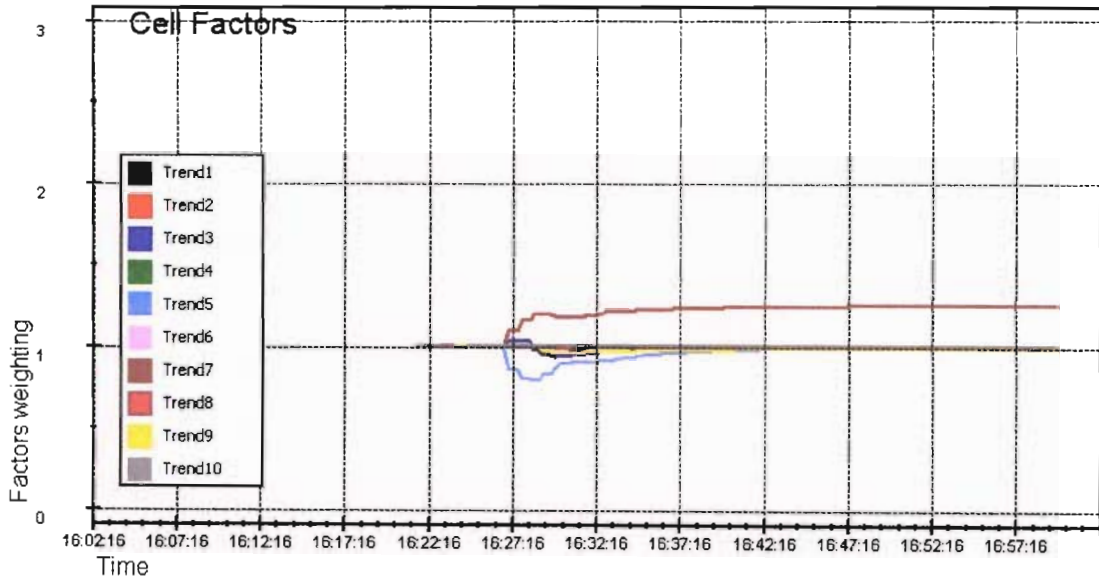


Figure 7-16: The cell factors converged positions after the step in tailings flow for unit 5.

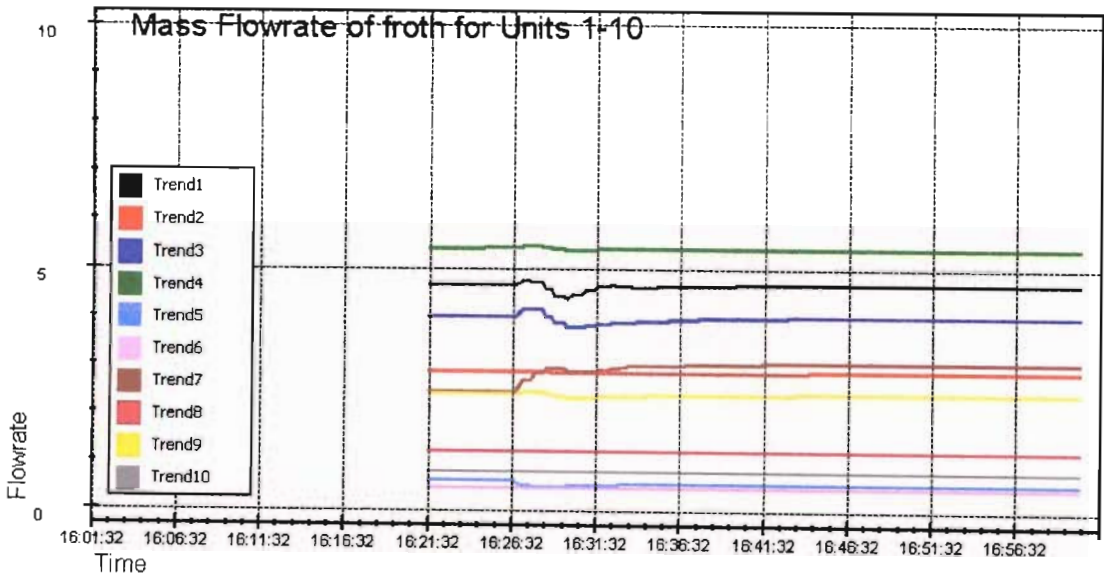


Figure 7-17: Mass flowrates of concentrates resulting from the new cell factors positions.

Figure 7-17 shows the mass flowrates of the concentrates for all the cells. Clearly, unit 7's concentrate flowrate has increased considerably due to the cell factor's increase. The slight changes to the mass flowrates of the other streams correspond to the small changes that can be seen in their cell factors in figure 7-16, before the model converges, due to the number of degrees of freedom as they are all unbound and able to move to try to accommodate the step change.

A final simulation was performed whereby the feed grade to the plant was doubled (feed grade is an observed variable by default at startup). In this simulation the effect of residence time in the cells can be seen as the increase in the platinum mass fractions of flows throughout the circuit is delayed by each of the residence times. Firstly, in figure 7-18 it can be seen that the platinum concentration is doubled, from 4.6g/ton to 9.2g/ton.

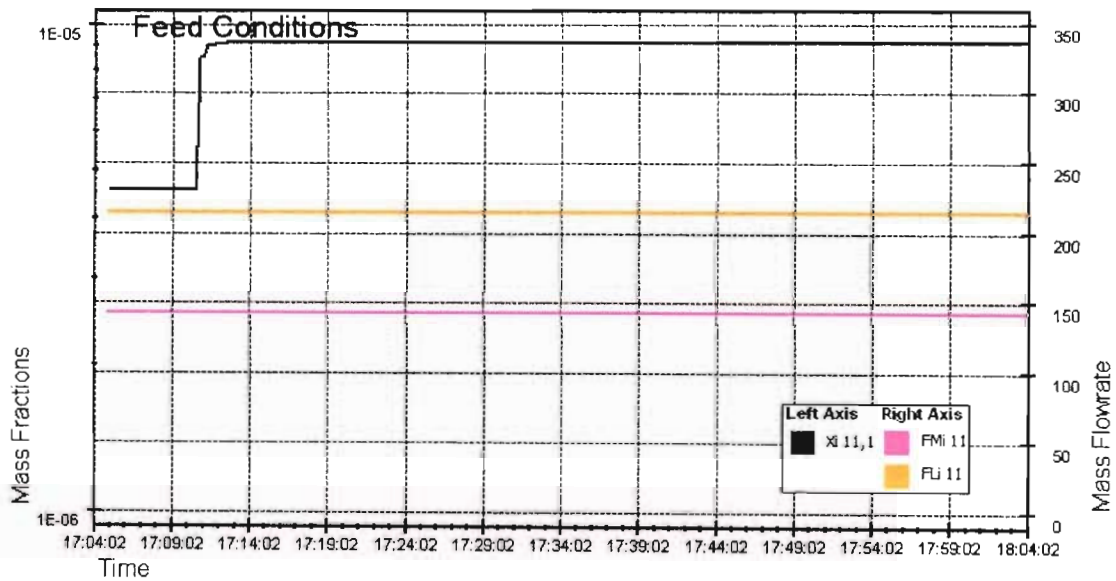


Figure 7-18: The step in feed grade from 4.6 g/ton to 9.2 g/ton.

The time at which the step was performed should be noted in order to observe the lag due to the residence time in the cells. The flowrates around the plant would obviously remain constant, as only the grade has changed, and as there is no change in the trends these graphs have been omitted. Figure 7-19 illustrates the increase in platinum concentration in the concentrates from all the cells. Note that unit 1's platinum mass fraction increases, as it is the first to receive the higher-grade pulp. The increase occurs at approximately 17:13 as the feed

passes through the primary mill and surge tanks before entering this cell. This time appears very short when considering the actual plant, however the simulations are performed with the model sped up by a factor of 2, in order to obtain the results in an appreciable amount of time. The ratios of the times between cells changes would, however, remain constant on a real time scale.

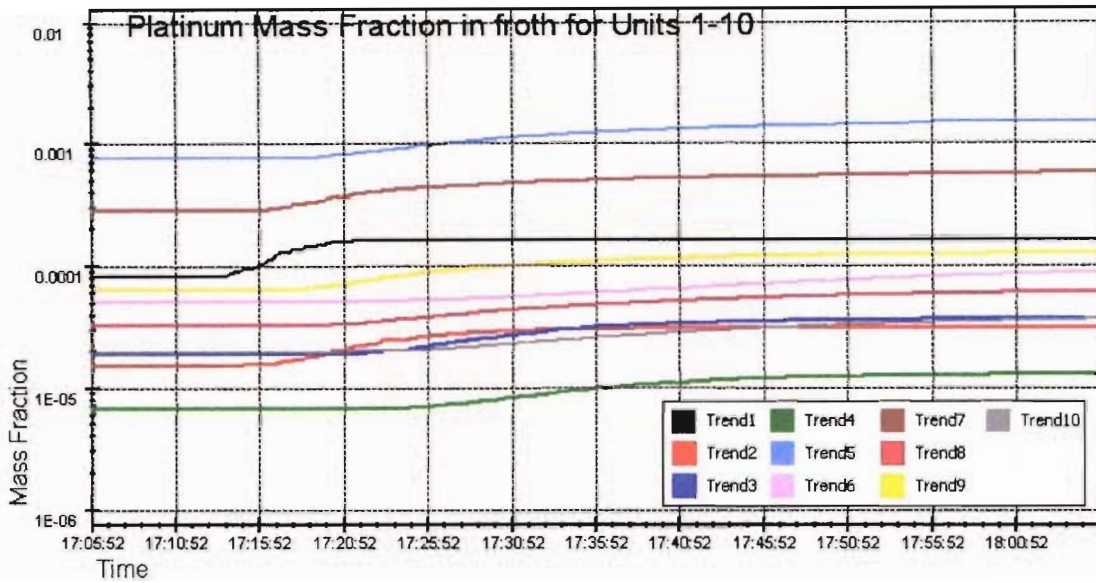


Figure 7-19: The increase in platinum mass fraction in the concentrate due to the step in feed grade.

The order in which the platinum mass fractions in the concentrates increases with respect to time clearly follows the topology of the plant, with firstly the primary rougher 1 (trend 1, black) increasing and lastly the secondary cleaner 3 (trend 6, pink). The same pattern can be seen in figure 7-20 with regard to the increased platinum concentrations leaving the plant in the tailings flows. The low efficiency of the froth flotation process leads to this increase, as not all of the platinum can be recovered from the pulp, and the higher grade of feed would result in a higher loss to the tailings.

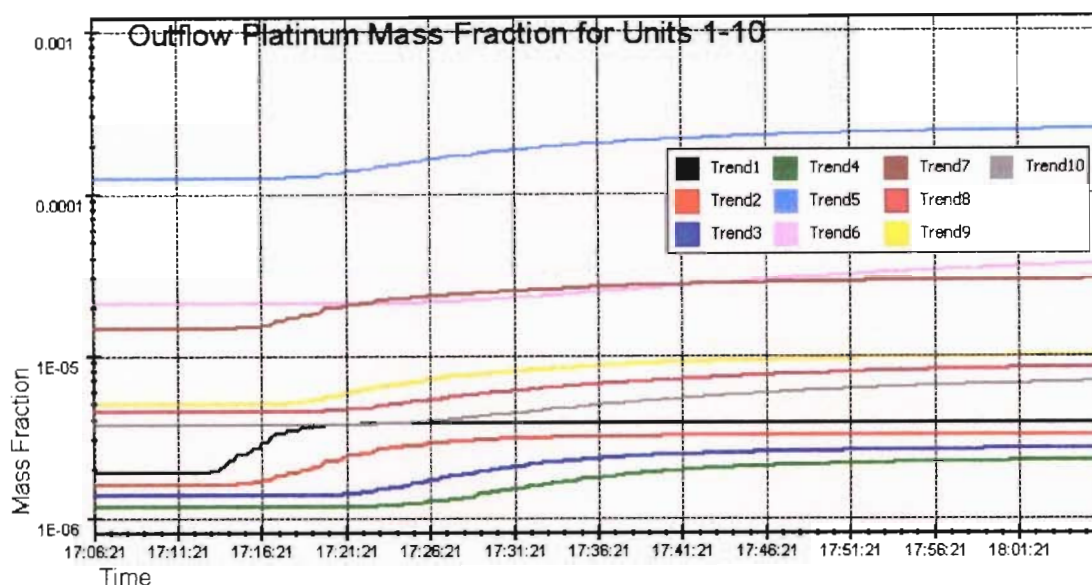


Figure 7-20: Trends showing the increased loss to the tailings of platinum due to the higher feed grade.

In a real plant situation it would be ideal to increase the concentrates platinum mass fraction whilst keeping the tailings constant or reducing them. This would be achieved through changes on the plant and therefore in an on-line application of the model it would track these changes and correct itself. The model might be used to determine what changes are to be made to the plant in order to achieve this, by running simulations and making the changes to the model as apposed to the plant.

The dynamic observer model has two variables which indicate the performance of the solution in terms of its derivative errors and observation errors. Once these two variables are zero, the model is considered to be converged and the plant to be in a steady state condition. These two variables are plotted against time to show the performance of the model and its convergence towards steady state.

Figure 7-21 shows the performance of the model whilst attempting to reach the steady state conditions for the first simulation (the step in feed flowrate of 30%). The errors drop until zero, at which time the plant is considered to be at a steady state condition.

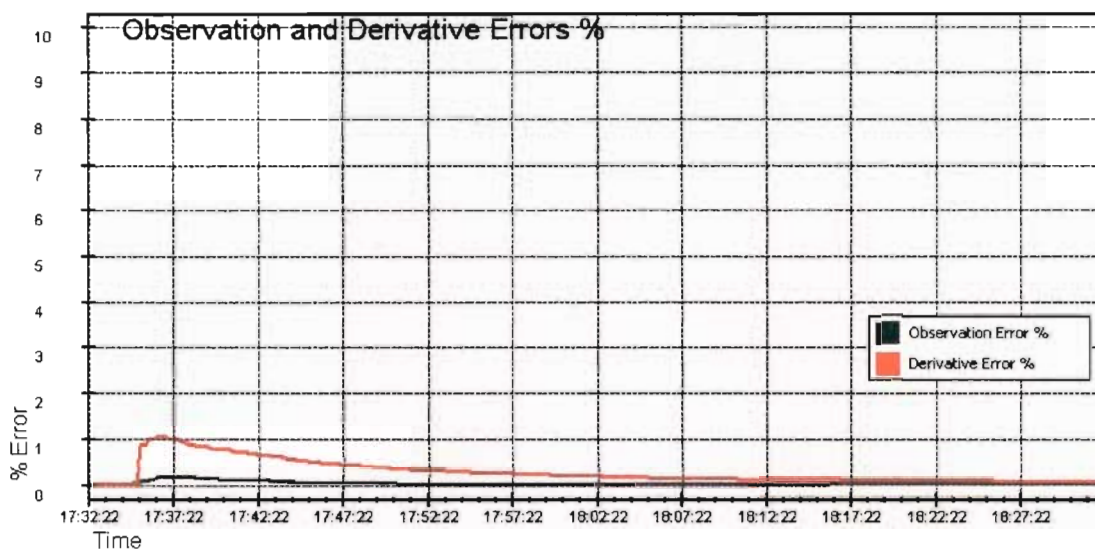


Figure 7-21: Indicators of the performance of the model solution and its convergence towards steady state.

The model's simulations are not limited to those discussed here and can accommodate any set of observations and step changes to the plant. These tests were performed using a set of initialisation conditions from a steady state data set obtained from plant assays, which provide a fairly good indication of normal operating conditions at the plant, however, future work would include installing the model at the plant and tuning it to the plants operating conditions, after which it would update its position relative to the plant automatically, so as to track it more tightly.

8 CONCLUSION

Mineral processing circuits face many problems, particularly the lack of on-line measurements due to limited access points, resulting in measures of plant performance only becoming available after laboratory analysis. A dynamic observer model of the LONMIN Eastern platinum B-Stream flotation circuit was developed in order to provide better insight into the process and explain plant behaviour by inferring unavailable plant variables in real time.

Numerous flotation models were reviewed, with special attention to the kinetic rate modelling of flotation sub processes. However, the dynamic observer model's flotation rate kinetics are based upon the first order flotation rate equation in order to simplify the strenuous calculations performed by the dynamic observer.

The model solution, based upon an extended Kalman filter, is able to reconcile dynamically available measurements as they are presented to it, and to infer those variables that are unavailable. The measurement selection set may vary in real time allowing for circuit changes and model operation flexibility. The model solution can be run in a number of different operational modes depending upon the information required by the user. Features of the model allow its solution characteristics to be changed in real time allowing for different scenarios to be simulated in succession. The default set of solution parameters found in the model are optimised for the LONMIN Eastern platinum B-Stream flotation circuit and further tuning would be required if the model is applied to another circuit.

The dynamic observer model was first tested in the Microsoft C++ 6 environment before it was integrated into a dynamic link library (DLL). This DLL was in turn debugged using a Microsoft Visual Basic 6 testing program created to imitate the control platform in which the model was finally implemented, namely, PlantStar®, developed by MINTEK. The modular design of PlantStar allows the observer model to be integrated into a DLL which connects to a plant definition file created for PlantStar and handles all data transfers between them. PlantStar makes available all online measurements to the model via the DLL, whilst all model outputs are returned to the plant definition file in the same manner. The performance of the model variables are then displayed next to the plant's operating variables in a graphical user interface. Comparing the variables in this manner allows the user to observe the model's performance at tracking the plants conditions.

A graphical user interface was created to facilitate easy manipulation of variables and to give a clearer view of the operation of the dynamic observer model within PlantStar. The plant topology and key processing units are displayed in detail along with a set of model solution parameters which can be manipulated to change the model solution performance and operation. The plant topology, however, cannot be modified in real-time at present, as it is set-up within the model section of the “plugin” module. This may be added to the “plugin” module in future, so as to make it more versatile and easy to configure to different plant topologies, however, as this work focused only on one plant, it was considered unnecessary to go to the expense of creating this feature at present.

A set of graphs track all available plant variables in real time, giving an indication of what condition the plant is in at any moment. This feature also shows the trend of plant characteristics over a set time span in the past so that the user can identify in which direction the plant is moving and perhaps take corrective action before the plant’s performance is affected.

Step tests and convergence tests on a steady state plant data set were performed to observe the model’s effectiveness at simulating changes in plant conditions and reconciling all of the plant variables. These showed promising results, though only a small number of tests were performed. The nature of the solution of the model allows for any selection of variables to be changed or stepped and the new converged steady state of the plant observed. Many other tests can be subsequently performed to simulate any plant situation which may arise and determine what possible effects it may have on the plant. Another possibility would be to make changes to the simulated plant in order to obtain a given state of the plant and from these determine which actions are required on the real plant in order to achieve this state.

An on-line application of the model would give a better indication of its performance and functionality. Plans are in place to install it on site in the near future at which time these tests may be performed and compared to laboratory assays.

REFERENCES

Abu-Ali, M.H. and Abdel Sabour, S.A., 2003. "Optimizing the design of flotation circuits: an economic approach", *Minerals Engineering*, Vol. 16, pages 55-58.

*Ahmed, N and Jameson, G. J., 1989. "Flotation Kinetics", *Mineral Processing and Extractive Metallurgy Review*, Vol. 5, pages 77-99.

*Alexander, D.J. and Morrison, R.D., 1998. "Rapid estimation of floatability components in industrial flotation plants", *Minerals Engineering*, Vol. 11, No. 2, pages 133-143.

Anfruns, J.F. and Kitchener, J.A., 1977. "Rate of capture of small particles in flotation", *Trans. IMM*, March, pages C9-19.

Arbiter, N. and Harris, C.C., 1962. "Flotation Kinetics", In "Froth Flotation", ed. D.W. Fuerstenau., 50th Anniversary Volume, AIME, New York, pages 215-246.

Bisshop, J.P. and White, M.E., 1976. "Study of particle entrainment in flotation froths", *Trans. IMM*, Vol. 85, pages C191-194.

Bloom, F. and Heindel, T.J., 2002. "On the structure of collision and detachment frequencies in flotation models", *Chemical Engineering Science*, Vol. 57, pages 2467-2473.

*Casali, A., Gonzalez, G., Agosto, H. and Vallebuona, G., 2002. "Dynamic simulator of a rougher flotation circuit for a copper sulphide ore", *Minerals Engineering*, Vol. 15, pages 253-262.

Dai, Z., Dukhin, S., Fornasiero, D. and Ralston, J., 1998. "The inertial hydrodynamic interaction of particles and rising bubbles with mobile surface", *J. Colloid Interface Sci.*, Vol. 197, pages 275-292.

Dai, Z., Fornasiero, D. and Ralston, J., 2000. "Particle-bubble collision models – a review", *Advances in Colloid and Interface Science*, Vol.85, pages 231-256.

Deglon, D.A., Sawyerr, F. and O'Connor, C.T., 1999. "A model to relate the flotation rate constant and the bubble surface area flux in mechanical flotation cells", *Minerals Engineering*, Vol. 12, No.6, pages 599-608.

- Edwards, R.P. and Mular, A. L., 1992. "An expert system supervisor of a flotation circuit", CIM Bulletin, Vol. 84 No. 959, April, pages 69-76.
- Ekmekci, Z., Bradshaw, D.J., Allison, S.A. and Harris, P.J., 2003. "Effects of frother type and froth height on the flotation behaviour of chromite in UG2 ore", Minerals Engineering, Vol.16, pages 941-949.
- Feteris, S.M., Frew, J.A. and Jowett, A., 1987. "Modelling the effect of froth depth in flotation", Int. J. Miner. Process., Vol. 20, pages 121-135.
- *Fichera, M.A. and Chudaseck, M.W., 1992. "Batch cell flotation Models - A review", Min. Engng., Vol. 5, No. 1, pages 41-55.
- Flint, L. R. and Howarth, W. J., 1971. "The collision efficiency of small particles with spherical air bubbles", Chem. Eng. Sci., Vol. 26, pages 1155-1168.
- *Flint, L.R., 1974. "A mechanistic approach to flotation kinetics", Trans. IMM, Vol. 83, pages C90-C95.
- *Frew, J.A., and Trahar, W.J., 1982. "Roughing and cleaning flotation behaviour and the realistic simulation of complete plant performance", Int. J. Miner. Process., Vol. 9, pages 101-120.
- Gaudin, A.M., 1957. "Flotation", 2nd Edition. McGraw-Hill, New York.
- Gonzalez, G.D., Orchard, M., Cerda, J.L., Casali, A. and Vallebuona, G., 2003. "Local models for soft-sensors in a rougher flotation bank", Minerals Engineering, Vol. 16, pages 441-453.
- Gorain, B.K., Franzidis, J.P. and Manlapig, E.V., 1997. "Studies on impeller type, impeller speed and air flow rate in an industrial scale flotation cell. Part 4: Effect Of Bubble Surface Area Flux On Flotation Performance", Minerals Engineering 10, Vol. 4, pages 367-379.
- Gorain, B.K., Harris, M.C., Franzidis, J.P. and Manlapig, E.V., 1998. "The effect of froth residence time on the kinetics of flotation", Minerals Engineering, Vol. 11, No. 7, pages 627-638.
- Greaves, M. and Allan, B.W., 1974. "Steady-State and Dynamic Characteristics of flotation in a single cell", Trans. Instn Chem. Engrs., Vol. 52, pages 136-148.
-

- *Haber, R., and Unbehauen, H., 1990. "Structure Identification of non linear Dynamic Systems – A survey of Input/Output Approaches", *Automatica*, Vol. 26, No. 4, pages 651-677.
- Hanumath, G. S. and Williams, D.J.A., 1992. "A three phase model of froth flotation", *Int J. Miner. Process.*, Vol. 34, pages 261-273.
- Harris, C.C. and Rimmer, H.W., 1966. "Study of a two phase model of the froth flotation process", *Trans. IMM*, Vol. 79, pages C153-C162.
- *Harris, C.C., 1976. "A recycle flow flotation machine model: Response of a model to parameter changes", *Int. J. Miner. Process.*, Vol. 3, pages 9-25.
- Harris, C.C., 1978. "Multiphase models of flotation machine behaviour", *Int. J. Min. Proc.* 5, pages 107–129.
- Herbst, J. A., Pate, W.T. and Oblad, A.E., 1992. "Model-based control of mineral processing operations", *Powder Technology*, Vol. 69, pages 21-32.
- Honaker, R.Q. and Ozsever, A.V., 2003. "Evaluation of the selective detachment process in flotation froth", *Minerals Engineering*, Vol.16, pages 975-982.
- Kalman, R.E., 1960. "A New Approach to Linear Filtering and Prediction Problems", *Trans. ASME*, Vol. 82, Series D, pages 35-45.
- Kelsall, D.F., 1961. "Application of Probability in the Assessment of Flotation Systems", *Trans. IMM*, Vol. 70, pages 191-204.
- *King, R.P., 1975. "Simulation of Flotation Plants", *Trans. Soc. Min. Eng. AIME*, Vol. 58, pages 286-293.
- *Klimpel, R.R., 1980. "Selection of chemical reagents for flotation", In: Mullar, A.L., Bhappu, R.B. Eds., "Mineral Processing Plant Design", 2nd edn., AIME, New York, pages 907–934.
- Leal Filho, L.S., Seidl, P.R., Correia, J.C.G. and Cerqueira, L.C.K., 2000. "Molecular modelling of reagents for flotation processes", *Minerals Engineering*, Vol. 13, No. 14-15, pages 1495-1503.
-

Loveday, B.K. and Brouckaert, C.J., 1995. "An analysis of flotation circuit design principles", *The Chemical Engineering Journal*, Vol. 59, pages 15-21.

Luyben, W.L., 1990. "Process Modelling, Simulation and Control for Chemical Engineers", 2nd Edition, McGraw-Hill.

*Mao, L. and Yoon, R.H., 1997. "Predicting flotation rates using a rate equation derived from first principles", *Int. J. Miner. Process.*, Vol. 50, pages 171-181.

Mathe, Z.T., Harris, M.C., O'Connor, C.T. and Franzidis, J.P., 1998. "Review of froth modelling in steady state flotation systems", *Minerals Engineering*, Vol. 11, No. 5, pages 397-421.

*McKee, D.J., 1991. "Automatic flotation control-A review of 20 years of effort", *Minerals Engineering*, Vol. 4, Nos 7-11, pages 653-666.

*Moys, M.H.M., 1978. "A study of a plug flow model for flotation froth behaviour", *Int. J. Miner. Process.*, Vol. 5, pages 221-238.

Mulholland, M., Vosloo, J-R.I and Loveday, B.K., 2003. "Real-time Observer model for mineral processing circuits", School of Chemical Engineering, University of KwaZulu-Natal, unpublished technical report for DACST Innovations project.

*Murphy, D.G., Zimmermann, W. and Woodburn, E.T., 1996. "Kinematic model of Bubble motion in flotation froth", *Powder technology*, Vol. 87, pages 3-12.

Neethling, S.J. and Cilliers, J.J., 1998. "The effect of weir angle on bubble motion in a flotation froth: visual modelling and verification", *Minerals Engineering*, Vol. 11, No. 11, pages 1035-1046.

Neethling, S.J. and Cilliers, J.J., 2002. "The entrainment of gangue into a flotation froth", *Int. J. Miner. Process.*, Vol. 64, pages 123-134.

Nguyen, A.V., Evans, G.M. and Jameson, G.J., 2001. "Bubble-Particle Attachment in Froth Flotation", 6th World Congress of Chemical Engineering, Melbourne, Australia.

Nguyen, A.V., Ralston, J. and Schulze, H.J., 1998. "On modelling of bubble-particle attachment probability in flotation", *Int. J. Miner. Process.*, Vol. 53, pages 225-249.

- *Polat, M. and Chander, S., 2000. "First-order flotation kinetics models and methods for estimation of the true distribution of flotation rate constants", *Int. J. Miner. Process.*, Vol. 58, pages 145-166.
- *Runge, K.C., Harris, M.C., Frew, J.A. and Manlapig, E.V., 1997. "Floatability of Streams Around the Cominco Red Dog Lead Cleaning Circuit", Sixth Mill Operators' Conference.
- Savassi, O.N., Alexander, D.J., Franzidis, J.P. and Manlapig, E.V., 1998. "An Empirical Model for Entrainment in industrial flotation circuits" *Minerals Engineering*, Vol. 11, No. 3, pages 243-256.
- Schroder, A. J., 1999. "Modelling, Optimisation and Control of Mineral Flotation Circuits", Phd thesis, Department of Chemical Engineering, University of Queensland, Australia.
- Schuhmann, R., 1942. "Flotation kinetics: I. Methods for steady-state study of flotation problems", *J. Phys. Chem.*, Vol. 64, pages 891-902.
- Schulze, H.J., 1989. "Hydrodynamics of bubble mineral particle collisions", *Min. Process. Extractive. Metall. Rev.* 5, pages 43-76.
- Stechemesser, H. and Nguyen, A.V., 1999. "Time of gas-solid-liquid three-phase contact expansion in flotation", *Int. J. Miner. Process.*, Vol. 56, pages 117-132.
- Sutherland, K.L., 1948. "Kinetics of flotation process", *J. Phys. Chem.*, Vol. 52, pages 386-390.
- Trahar, W.J. and Warren, L.J., 1976, "The floatability of very fine particles - A review", *Int. J. Miner. Process.*, Vol. 3, pages 103-131.
- van Deventer, J.S.J., van Dyk, W.A., Lorenzen, L. and Feng, D., 2002. "The dynamic behaviour of coarse particles in flotation froths Part I: Model", *Minerals Engineering*, Vol. 15, pages 635-645.
- Vera, M.A., Mathe, Z.T., Franzidis, J.P., Harris, M.C., Manlapig, E.V. and O'Connor, C.T., 2002. "The modelling of froth zone recovery in batch and continuously operated laboratory flotation cells", *Int. J. Miner. Process.*, Vol. 64, pages 135-151.
- Wan, E.A. and van der Merwe, R., 1999. "The Unscented Kalman Filter for Nonlinear Estimation", Mathematics Department, Macquarie University, Sydney.
-

Weber, M.E. and Paddock, D., 1983. "Interceptional and gravitational collision efficiencies for single collectors at intermediate Reynolds numbers", *J. Colloid Interface Sci.*, Vol. 94, pages 328–335.

Welch, G. and Bishop, G., 2001. "An Introduction to the Kalman Filter", University of North Carolina at Chapel Hill, Department of Computer Science,
<http://www.cs.unc.edu/~{Welch,gb}>

*Woodburn, E.T. and Wallin, P.J., 1984. "Decoupled kinetic model for simulation of flotation networks", *Trans. Instn Min. Metall.*, Vol. 93, pages C153-C161.

Woodburn, E.T., 1970. "Mathematical modelling of flotation process", *Minerals Sci. Eng.*, Vol. 22, pages 3–17.

Yingling, J.C., 1993. "Parameter and configuration optimisation of flotation circuits, part I. A review of prior work", *Int. J. Miner. Process.*, Vol. 38, pages 21-40.

Yoon, R.H. and Luttrell, G.H., 1989. "The effect of bubble size on fine particle flotation", *Min. Process. Extractive Metall. Rev.*, Vol. 5, pages 101-122.

Yoon, R.H., 2000. "The role of hydrodynamic and surface forces in bubble-particle interaction", *Int. J. Miner. Process.*, Vol. 58, pages 129-143.

Note that those marked with an (*), are indirect references and were not quoted in the text.

APPENDIX A GLOSSARY OF TERMS

The following list gives the meanings of some of the terms used in this thesis that may not be familiar to the reader. These terms are not specific definitions but rather the understanding of the author.

<i>Bank</i>	Flotation cells in series, where the tailings from one cell are the feed to the next, are known as a bank. A bank may also be one large trough with baffles separating it into cells with a single launder.
<i>Beta</i>	A beta program is one that is still in the developmental stage and is used for testing and debugging before the final release.
<i>Cell</i>	A cell is the basic processing unit in froth flotation, which is a stirred and aerated tank encouraging the hydrophobic minerals to leave via the overflow while the remaining minerals leave by the underflow.
<i>Cleaners</i>	Cleaners receive concentrate from the roughers and attempt to increase the grade of the material leaving the circuit. The final concentrate leaves this part of the circuit.
<i>Collector</i>	Collector is a term for a type of reagent added to an ore suspension, which renders certain particles hydrophobic so that they may attach to air bubbles and leave the cell via the overflow.
<i>Concentrate</i>	The flow of minerals and water that leave a flotation cell in the overflow
<i>Dynamic Link Library</i>	A program file that is used to transfer data to and from applications.
<i>Froth</i>	One of the phases present in a flotation cell comprised of air bubbles floating on the surface of the <i>pulp</i> . This is then collected in a <i>launder</i> to produce the <i>concentrate</i> .

<i>Frother</i>	A type of reagent added to the pulp, which promotes the formation of bubbles and controls the characteristics of the air bubbles to reduce collapse.
<i>Gangue</i>	The non-valuable materials in the ore, which must be removed to achieve a high grade concentrate.
<i>Grade</i>	The content of a particular mineral in a stream, usually the valuable mineral.
<i>Launder</i>	The channel on a flotation cell that catches the froth as it passes over the weir, producing the concentrate.
<i>Member functions</i>	Parts of a DLL class that a program connects to by calling the corresponding member function name along with its variables to be passed.
<i>Plant definition file</i>	The program file used by the PlantStar control platform to describe the entire plant topology and control algorithms. It connects to low level plant control systems such as SCADA through PlantStar and allows connections to more sophisticated algorithms such as the dynamic observer model through dynamic link libraries.
<i>Plugin</i>	A program file that can connect to PlantStar in the form of a module.
<i>Pulp</i>	One of the phases present in flotation cells comprised of a suspension of ground ore particles. This phase leaves the cell via the underflow or <i>tailings</i> .
<i>Rougher</i>	Roughers are the first part of the flotation circuit and remove most of the valuables. The concentrate from this part of the circuit is sent to the cleaners
<i>Tailings</i>	The underflow from a flotation cell is known as the tailings. It is comprised of the material that has not floated, mainly gangue and little valuable mineral.

APPENDIX B CLASSIFICATION OF FLOTATION CIRCUIT PROCESS VARIABLES

Table B-1: Typical Controlled Variables in Flotation Circuits (Schroder, 1999)

Controlled Variable	Setpoint	Reason for Control
Final concentrate grade (for cleaner circuits)	Determined by product specifications (smelter requirements)	To maintain product quality (this objective competes with that of controlling recovery)
Valuable mineral recovery (or final tailings grade)	Determined by economic considerations	To maintain circuit efficiency (this objective competes with that of controlling product grade)
pH	Chosen to maximise flotation selectivity for the valuable mineral	To regulate the chemical environment, which affects the selectivity of separation (promote steady and efficient operation)
Reagent addition rates	Chosen to maximise valuables recovery	To regulate the concentration of reagents in the circuit (these control the selectivity of the separation)
Bank pulp level	Chosen to maximise the separation capacity of each bank	To maintain steady operation and to prevent pulp overflow from cells into the concentrate
Sump level	Usually 50% (to give maximum buffering against changes in concentrate flowrate)	To prevent the sumps from running dry or overflowing
Recirculating load	Determined by pumping capacity of the sump pumps and by the degree of control over bank concentrate flowrate	To limit concentrate flows to the sumps so that sump pumping capacity is not exceeded

Table B-2: Typical Disturbance Variables in Flotation Circuits (Schroder, 1999)

Disturbance Variable	Primary Effect	Comments
Feed flowrate	Flow volume determines the residence time in the circuit	Can vary dramatically (feed is the product of a grinding circuit or another flotation circuit, so fluctuations are passed on), surges in flowrate from a grinding circuit due to poor classification control may be accompanied by increased particles sizes and reduced liberation
Feed composition	Feed composition determines the amount of separation 'work' which needs to be done to produce concentrate of the target grade	If concentration of valuable mineral in feed is low, then a circuit may not be able to deliver the desired concentrate at reasonable recoveries because of limitations on residence time and flotation selectivity
Mineral liberation in feed	Liberation sets the maximum physical separation of minerals which can be achieved without further grinding	Liberation depends mostly on the size of the mineral grains in the ore body, and the ability of the grinding circuit to reduce the ore to particles of less than this size degree of liberation can be highly variable if there is poor control of product particle size in the grinding circuit
Feed flotation properties	Flotation properties determine the relative flotation rates of minerals	Properties are determined by the particle size distribution, liberation, the concentration of reagents on the particle surfaces, the microscopic characteristics and surface properties of the particles

Table B-3: Typical Measurements Available in Flotation Circuits (Schroder, 1999)

Measured Variable	Instrument	Positioning	Typical Accuracy	Notes
Stream composition	On-line stream analyser (OSA) or in-line analyser (ISA)	Typically new feed, final concentrate and final tails	$\pm 2-5 \%$	Uses x-ray spectrometry or nucleonics to measure the elemental composition. Sampling interval is usually between 1 and 15 minutes, depending on the instrument.
Stream flowrate	Magnetic flowmeter	Often only final tailings because of restrictions on positioning	$\pm 1-2 \%$	Measures the electric flux of a stream and estimates the velocity. Must be installed on a straight run of vertical pipe, so that there is no segregation of solids across the pipe profile.
Stream density	Nuclear density gauge	Often only final tailings because of restrictions on positioning	$\pm 1-3 \%$	Infers density from attenuation of a beam of radiation through the stream. Must be installed on a straight run of vertical pipe, as for magnetic flowmeters.
pH	Industrial pH probe	Downstream of lime addition points	± 0.1 units of pH	Sensor must be located so that lag in pH measurement is small. Suffers from drift, and requires frequent re-calibration.
Pulp level	Bubble tube, float with position sensor or conductivity probe	Last cell of each bank (near to the tailings valve)	$\pm 5-10 \%$	Pulp level can be difficult to measure because the pulp-froth interface is not well defined. Bubble tubes estimate pulp level from back pressure applied by the pulp liquid, and this changes with pulp density.
Sump level	Bubble tube, float with position sensor or conductivity probe	In each sump	$\pm 2-5 \%$	Measurements are reasonably accurate because the froth has collapsed and the sump contents therefore presents a clean liquid surface.

Table B-4: Typical Manipulated Variables in Flotation Circuits (Schroder, 1999)

Manipulated Variable	Aspect of Circuit Performance Affected	Speed of Response	Comments
Additions of reagent to feed	Whole circuit performance	Slow	<p>Collector, which selectively absorbs onto particles, rendering them hydrophobic.</p> <p>Frother, which reduces surface tension at the air-water interface, resulting in finer bubbles and a more stable froth.</p> <p>pH modifiers, pH determines the selectivity of collector adsorption by minerals.</p> <p>Activators and depressants, which modify the tendency of specific minerals to float.</p> <p>Reagents are commonly added to the feed, although they may be supplemented by additions at other points.</p>
Bank air additions	Rate of recovery of each mineral from the bank	Fast	<p>Air addition to each bank is split into a number of arterial lines that supply the individual cells.</p> <p>Bank air rate is normally controlled remotely, while supply to individual cells is controlled manually.</p> <p>Increasing air additions will increase the recovery of all minerals, and may also increase the amount of entrained material, leading to increased recovery at a cost to concentrate grade.</p> <p>Most often used to control the performance of individual banks.</p>
Bank tailings valve opening	Bank pulp level	Fast	<p>Valve position determines how fast tailings flow from a bank, and hence the depth of pulp in the bank.</p> <p>Valve position controls both the residence time of pulp in the bank and the froth depth.</p> <p>Sometimes used to control bank performance, less often than air addition.</p>
Water additions	Bank concentrate density, bank froth	Fast	<p>Water is added to the concentrate from each bank to make it amenable to pumping.</p> <p>Water may be sprayed over the froth to encourage drainage of hydrophilic gangue back to the pulp.</p> <p>Water additions effectively reduce the pulp density if the concentrate is retreated by other banks: this can improve separation but also reduces the pulp residence time and increases the pumping requirements.</p>

Speed of the sump pumps	Sump level	Fast	Pumping rate is set to meet the dual requirements of maintaining the sump level and maintaining a steady flow of concentrate to other units.
Froth diverters	Circuit configuration	Fast	Allow the operator to change where a concentrate stream flows, to reasonably rare in flotation circuits.

APPENDIX C OTHER METHODS OF DAE SOLUTION

DAE's properties are a mixture of those of ordinary differential equations and algebraic equations. In general they may require a combination of solution techniques. Some of these techniques will be discussed in the following sections for completeness. The general form of a DAE system is as follows:

$$\frac{dx}{dt} = f(x, y, t): \text{ Ordinary Differential Equations} \quad \text{Eq C-1}$$

$$g(x, y, t) = 0: \text{ Algebraic equations} \quad \text{Eq C-2}$$

where:

- x Variables whose time derivatives appear in the model (differential variables)
- y Variables whose time derivatives do not appear in the model (algebraic variables)

For this system to be easy to solve, it must be possible to solve the AE's,

$$g = 0 \quad \text{Eq C-3}$$

for y , if fixed values of x and t are available.

C.1.1 A SIMPLE DAE CASE

If the following set of conditions is met, then the solution of the DAE's can be handled quite simply:

- The system contains only algebraic equations and ODE's with a single independent variable, usually time.
- There are as many equations in total, whether difference or algebraic, as there are variables excluding the independent variable.

- Initial values are given for as many variables as there are ODE's.
- All variables which do not appear as derivatives occur at least once in the algebraic equations.
- The algebraic equation subset may be solved using the normal methods available for AE's if values are known for all variables which appear as derivatives.
- Initial values are available for all variables which appear as derivatives.
- The algebraic subset is amenable to a non-iterative direct solution.

Consider the following arbitrary set of 4 DAE's:

$$\frac{dx}{dt} = w - y \quad \text{Eq C-4}$$

$$\frac{dy}{dt} = y - x \quad \text{Eq C-5}$$

$$z = y - x \quad \text{Eq C-6}$$

$$w = z - x^0 \quad \text{Eq C-7}$$

with initial conditions: $x=1$ and $y=0$ at $t=0$

Note that:

- There are only AE's and ODE's with one independent variable t .
- There are 4 equations and 4 variables w, x, y, z .
- There are 2 ODE's and two initial conditions are given.

This satisfies the first three conditions given above. Furthermore:

- In the 2 ODE's, x and y , appear in derivatives. These will be referred to as the differential variables.
- The remaining variables, w and z , which we will call the algebraic variables, both appear in the two AE's.
- If values were known for the differential variables, the algebraic subset would consist of two equations in two unknowns and would, in principle, be solvable.

Thus we have thus satisfied all the main conditions. Additionally, in this example:

- Initial conditions are given for both the differential variables.
- The AE's are already both written as formulas, which enables the two algebraic variables to be calculated directly.

The solution algorithm involves three steps, one of which is performed only once, at the start of the solution procedure. The other two must each be carried out at each timestep.

Initialisation (once only):

- Initial conditions being given for the differential variable, the AE's can be solved for the algebraic variables.
- The values of all variables are now known at time $t=0$.

The next two steps must be performed at each timestep. The procedure starts with values for all variables known at a general time t , and finishes with them calculated at the next time increment ($t+dt$).

O.D.E solution:

- All variables are known at time t (time $t = 0$ on the first step)
- The right hand sides of the ODE's, from which values of derivatives can be calculated, are evaluated and the derivatives determined.
- The new values of the differential variables are calculated by a suitable ODE solving procedure, for example using:

$$y(t + dt) = y(t) + dy = y(t) + \frac{dy(t)}{dt} dt \quad \text{Eq C-8}$$

- After this has been performed for all ODE's, the values of the differential variables are known at time ($t+dt$).

Algebraic equation solution:

- The differential variable values being calculated above, the AE's now involve only the algebraic variables as unknowns.
- Solve the AE's
- All variables are now known at time ($t+dt$).

C.1.2 EXPLICIT NUMERICAL INTEGRATION METHODS

C.1.2.1 Explicit Euler

The key to any numerical ODE solution is to make a finite difference approximation to the derivative. The explicit Euler method makes use of forward difference as follows:

$$\left. \frac{dx}{dt} \right|_{t=t_0} \approx \frac{x_1 - x_0}{t_1 - t_0} \quad \text{Eq C-9}$$

therefore to solve:

$$\frac{dx}{dt} = f(x, t) \quad \text{Eq C-10}$$

specify x_0 at initial time t_0 and calculate $f(x)$ at t_0 , then the finite difference formula becomes (notation as that of diagram):

$$x_1 = x_0 + hf_0 \quad \text{Eq C-11}$$

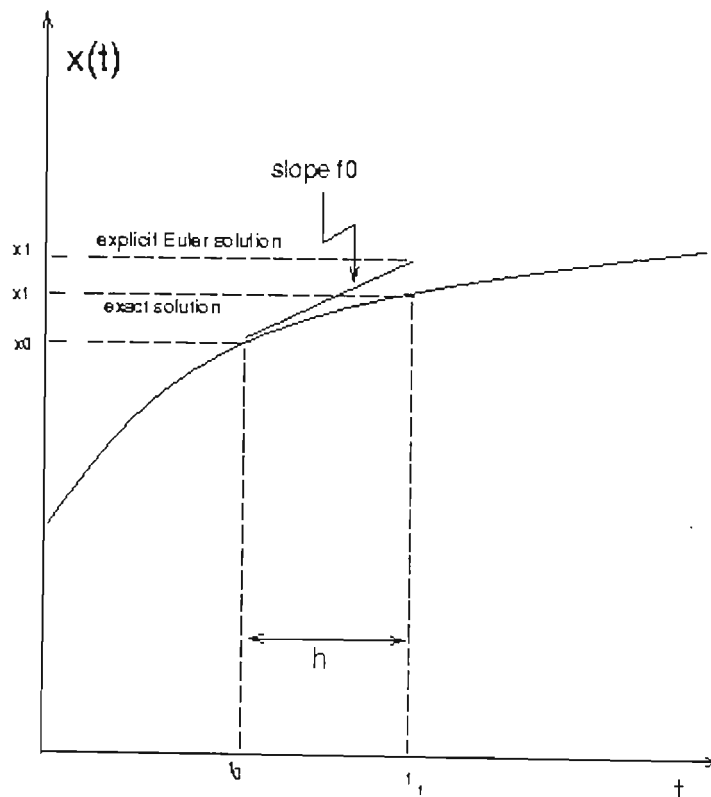


Figure C-1: Explicit Euler solution as compared to exact solution

where h refers to the interval step size. If this step size is small enough, the estimate of x_l will be very close to the actual value.

In terms of the solution of a DAE, once the ODE has been discretised in this fashion, values are required for all the algebraic variables at t_n . The integration gives only the differential variables at t_{n+1} . The AE's are then solved for the algebraic variables at t_{n+1} using the values calculated for x_{n+1} by explicit Euler. Consistent initial values must be found by solving the AE's at time t_0 with specified values $x(0)$ for the differential variables, i.e. solve:

$$\mathbf{x} - \mathbf{x}(0) = 0 \quad \text{Eq C-12}$$

$$\mathbf{g}(\mathbf{x}, \mathbf{y}, t_0) = 0 \quad \text{Eq C-13}$$

C.1.2.2 Runge-Kutta (Fourth Order)

Another widely used numerical method of integration of ODE's is the Runge-Kutta method. The Runge-Kutta algorithm is as follows:

$$\begin{aligned} k_1 &= \Delta t f(x_n, t_n) \\ k_2 &= \Delta t f\left(x_n + \frac{1}{2}k_1, t_n + \frac{1}{2}\Delta t\right) \\ k_3 &= \Delta t f\left(x_n + \frac{1}{2}k_2, t_n + \frac{1}{2}\Delta t\right) \\ k_4 &= \Delta t f(x_n + k_3, t_n + \Delta t) \\ x_{n+1} &= x_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \quad \text{Eq C-14}$$

If an accurate integration is required, the fourth order Runge-Kutta is superior to Euler. However, if computing speed is the main criterion then Euler is faster as it has four times less computations per time step to perform.

C.1.3 **IMPLICIT EULER**

The Implicit Euler method uses a backward difference method and simultaneous solution of the ODE's and the AE's. The backward difference is as follows:

$$\left. \frac{dx}{dt} \right|_{t=t_1} \approx \frac{x_1 - x_0}{t_1 - t_0} \quad \text{Eq C-15}$$

The ODE is discretised in the following manner:

$$x_{n+1} = x_n + hf(x_{n+1}, t_{n+1}) \quad \text{Eq C-16}$$

As can be seen, the function f_{n+1} and variable x_{n+1} are all unknown. It therefore makes sense to solve the discretised ODE and the AE's simultaneously at t_{n+1} . The unknowns in the AE's are the differential and algebraic variables at time t_{n+1} :

$$\mathbf{x}_{n+1}, \mathbf{y}_{n+1} \quad \text{Eq C-17}$$

Solution of x_{n+1} involves an iterative procedure if f is non-linear. It is necessary to solve:

$$g(x_{n+1}) = x_{n+1} - x_n - hf(x_{n+1}, t_{n+1}) = 0 \quad \text{Eq C-18}$$

for x_{n+1} . Each iteration involves the calculation of f at an iterated value of $x^{(k)}$ for x_{n+1} .

Consider the set of N linear ODE's

$$\frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{12} & a_{22} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ a_{N1} & \dots & \dots & a_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{bmatrix} \quad \text{Eq C-19}$$

$$\frac{d\mathbf{x}}{dt} = \underline{\underline{\mathbf{A}}}\mathbf{x} \quad \text{Eq C-20}$$

where:

\mathbf{x} Vector of variables

$\underline{\underline{\mathbf{A}}}$ Square matrix of constants

Applying the implicit Euler algorithm to these equations gives:

$$\underline{\mathbf{x}}_{n+1} = \underline{\mathbf{x}}_n + \left[\underline{\mathbf{A}} \underline{\mathbf{x}}_{n+1} \Delta t \right] \quad \text{Eq C-21}$$

and solving for $\underline{\mathbf{x}}_{n+1}$

$$\underline{\mathbf{x}}_{n+1} = \left[\underline{\mathbf{I}} - \underline{\mathbf{A}} \Delta t \right]^{-1} \underline{\mathbf{x}}_n \quad \text{Eq C-22}$$

Using this technique for the set of ODE's, the solution of the set of DAE's can be found as above.

APPENDIX D MICROSOFT VISUAL BASIC 6

TESTING PROGRAM CODE

Below is the code used to create the testing program which was used to test the functionality of the PlantStar DLL in its early stages of development. The tester was necessary in order to speed up the debugging time required when using PlantStar. It simply imitates PlantStar and puts in the same calls to the member functions of the DLL that PlantStar would whilst operating.

D.1.1 PROGRAM DECLARATIONS

```
Dim x As New PSTest81BLib.TestClass

'Input section for variables
Private Inflow_X1(1 To 23) As Variant
Private Inflow_X2(1 To 23) As Variant
Private Inflow_X3(1 To 23) As Variant
Private Inflow_SMF(1 To 23) As Variant
Private Inflow_FM(1 To 23) As Variant
Private Outflow_X1(1 To 23) As Variant
Private Outflow_X2(1 To 23) As Variant
Private Outflow_X3(1 To 23) As Variant
Private Outflow_SMF(1 To 23) As Variant
Private Outflow_FM(1 To 23) As Variant
Private Frothflow_x1(1 To 10) As Variant 'this isn't 23 big?
Private Frothflow_x2(1 To 10) As Variant
Private Frothflow_x3(1 To 10) As Variant
Private Frothflow_smf(1 To 10) As Variant
Private Frothflow_fm(1 To 10) As Variant
Private Ao(1 To 10) As Variant
Private ho(1 To 10) As Variant 'this should be ho
Private H_o(1 To 14) As Variant 'This should be Ho but because of VB had
to make it H_o
Private ko(1 To 10, 1 To 4) As Variant
```

Private Pmo(1 To 14) As Variant

'Observe section

'Cells

Private Observe_Unit_1(1 To 19) As Variant

Private Observe_Unit_2(1 To 19) As Variant

Private Observe_Unit_3(1 To 19) As Variant

Private Observe_Unit_4(1 To 19) As Variant

Private Observe_Unit_5(1 To 19) As Variant

Private Observe_Unit_6(1 To 19) As Variant

Private Observe_Unit_7(1 To 19) As Variant

Private Observe_Unit_8(1 To 19) As Variant

Private Observe_Unit_9(1 To 19) As Variant

Private Observe_Unit_10(1 To 19) As Variant

'Mills

Private Observe_Unit_11(1 To 12) As Variant

Private Observe_Unit_12(1 To 12) As Variant

'Surge tanks

Private Observe_Unit_13(1 To 11) As Variant

Private Observe_Unit_14(1 To 11) As Variant

'Product Streams

'Private Observe_Unit_15(1 To 5) As Variant

'Private Observe_Unit_16(1 To 5) As Variant

'Private Observe_Unit_17(1 To 5) As Variant

'Private Observe_Unit_18(1 To 5) As Variant

'Private Observe_Unit_19(1 To 5) As Variant

'Private Observe_Unit_20(1 To 5) As Variant

'Private Observe_Unit_21(1 To 5) As Variant

'Private Observe_Unit_22(1 To 5) As Variant

'Private Observe_Unit_23(1 To 5) As Variant

```
'Output section
Private Unit_1(1 To 22) As Variant
Private Unit_2(1 To 22) As Variant
Private Unit_3(1 To 22) As Variant
Private Unit_4(1 To 22) As Variant
Private Unit_5(1 To 22) As Variant
Private Unit_6(1 To 22) As Variant
Private Unit_7(1 To 22) As Variant
Private Unit_8(1 To 22) As Variant
Private Unit_9(1 To 22) As Variant
Private Unit_10(1 To 22) As Variant
Private Unit_11(1 To 14) As Variant
Private Unit_12(1 To 14) As Variant
Private Unit_13(1 To 13) As Variant
Private Unit_14(1 To 13) As Variant
Private Unit_15(1 To 6) As Variant
Private Unit_16(1 To 6) As Variant
Private Unit_17(1 To 6) As Variant
Private Unit_18(1 To 6) As Variant
Private Unit_19(1 To 6) As Variant
Private Unit_20(1 To 6) As Variant
Private Unit_21(1 To 6) As Variant
Private Unit_22(1 To 6) As Variant
Private Unit_23(1 To 6) As Variant

Private CONSTRAINTFROMLAST As Long
Private STARTATLAST As Long
Private LOADLASTMT As Long
```

D.1.2 THE FORM LOAD CODE

```
Private Sub Form_Load()
Dim i As Long
'Create instance of DLL
Set x = New PSTest81BLib.TestClass
```

'Initialise arrays, so that the Variants in the arrays will have the CORRECT TYPES

```
For i = 1 To 22
    Unit_1(i) = CDbl(0)
    Unit_2(i) = CDbl(0)
    Unit_3(i) = CDbl(0)
    Unit_4(i) = CDbl(0)
    Unit_5(i) = CDbl(0)
    Unit_6(i) = CDbl(0)
    Unit_7(i) = CDbl(0)
    Unit_8(i) = CDbl(0)
    Unit_9(i) = CDbl(0)
    Unit_10(i) = CDbl(0)
Next
```

```
For i = 1 To 14
    Unit_11(i) = CDbl(0)
    Unit_12(i) = CDbl(0)
Next
```

```
For i = 1 To 13
    Unit_13(i) = CDbl(0)
    Unit_14(i) = CDbl(0)
Next
```

```
For i = 1 To 6
    Unit_15(i) = CDbl(0)
    Unit_16(i) = CDbl(0)
    Unit_17(i) = CDbl(0)
    Unit_18(i) = CDbl(0)
    Unit_19(i) = CDbl(0)
    Unit_20(i) = CDbl(0)
    Unit_21(i) = CDbl(0)
    Unit_22(i) = CDbl(0)
    Unit_23(i) = CDbl(0)
Next
```



```
Next
CONSTRAINTFROMLAST = CLng(0)
STARTATLAST = CLng(0)
LOADLASTMT = CLng(0)

'Initialise DLL
Call x.Initialise(CONSTRAINTFROMLAST, STARTATLAST,
LOADLASTMT)

End Sub
```

D.1.3 THE RUN CODE

```
Private Sub Command1_Click()
Call x.Execute(Inflow_X1, Inflow_X2, Inflow_X3, Inflow_SMF,
Inflow_FM, Outflow_X1, Outflow_X2, Outflow_X3, Outflow_SMF,
Outflow_FM, Frothflow_x1, Frothflow_x2, Frothflow_x3, Frothflow_smf,
Frothflow_fm, Ao, ho, H_o, ko, Pmo, Observe_Unit_1, Observe_Unit_2,
Observe_Unit_3, Observe_Unit_4, Observe_Unit_5, Observe_Unit_6,
Observe_Unit_7, Observe_Unit_8, Observe_Unit_9, Observe_Unit_10,
Observe_Unit_11, Observe_Unit_12, Observe_Unit_13, Observe_Unit_14,
Unit_1, Unit_2, Unit_3, Unit_4, Unit_5, Unit_6, Unit_7, Unit_8, Unit_9,
Unit_10, Unit_11, Unit_12, Unit_13, Unit_14, Unit_15, Unit_16, Unit_17,
Unit_18, Unit_19, Unit_20, Unit_21, Unit_22, Unit_23)
End Sub
```

It should be noted that the testing program only contains 14 of the 23 element arrays as far as the observations are concerned. This was necessary to ensure that the number of variables passed between the DLL and the testing program was kept below the maximum number that the Microsoft Visual Basic program can handle.

D.1.4 SHUTDOWN CODE

```
Private Sub Command25_Click()
Picture1.Cls
Call x.Shutdown
End Sub
```

D.1.5 EXAMPLE OF THE DISPLAY CODE FOR PRIMARY ROUGHER 1

```
Private Sub Command2_Click()
```

```
Picture1.Cls
For i = 1 To 22
    Picture1.Print Unit_1(i); ",";
    Picture1.Print
Next
End Sub
```

APPENDIX E MATLAB COMPILER INFORMATION

MATLAB is a very powerful mathematical program that is simple and easy to operate. It is therefore very popular when creating mathematically intensive programs or simulations, as it has optimised matrix and vector handling routines that are difficult to match in speed and efficiency using other software such as C++ or Visual Basic. As was discussed earlier in section 6.2.3 the dynamic observer model uses MATLAB to perform its large matrix inversions needed by the Kalman filter solution. The DOM has the option of using the MATLAB engine directly, which it achieves through command line calls to an open instance of the MATLAB command window or by the use of stand alone applications that read and write from files saved to the hard disk. The later method makes use of an M-File that has been compiled using the MATLAB compiler to create a stand-alone application. This allows for the M-File code that was created inside MATLAB and used by MATLAB to be portable and run independently of the MATLAB program.

An example of this method can be seen below which was created to find the average values of a large amount of data from a database using a centred average of a moving widow:

The M-File Code:

```
function [] = data_reduction()

%Coded by John-Roy Vosloo
%03/04/2003
data_in = dlmread('11_02_2003_data.txt');
mSize = size(data_in);
nrows = mSize(1,1);
ncolumns = mSize(1,2);
window_size = 20;
sample_int = 20;
temp = zeros(window_size,ncolumns);
temp_sum = zeros(1,ncolumns);
data_avg = zeros(window_size,ncolumns);
```

```
n=0;
i=0;
j=0;
k=0;
h=0;
while n < nrows
    n=n+1;
    for i=1:window_size
        k=max(1,n-window_size+i);
        temp(i,:)=data_in(k,:);
    end
    temp_sum = sum(temp);
    data_avg = temp_sum/window_size;
    if (mod(n,sample_int)==0)
        h=h+1;
        data_out(h,:) = data_avg(1,:);
    end
end
dlmwrite('11_02_2003_20_20.txt',data_out);
```

It should be noted here that for the compilation to be successful the file name of the M-File must correspond to the function name. In the MATLAB command window the following is inputted in order to call the MATLAB compiler and to create the stand-alone application:

```
mcc data_reduction -c
```

This would then create a file called data_reduction.exe, which would be accessible from the windows command prompt. The stand-alone applications for solving the linear solution and filter solutions for the EKF in the DOM were created in a similar way.

Another useful feature of the MATLAB compiler is an add-in toolbox for Visual C++, which allows for the addition of M-Files to C++ projects with all the necessary libraries and functions used by MATLAB automatically included into the project. This method was successfully carried out for small projects and manipulations of matrices, however it was not possible to implement into the dynamic link library for use with PlantStar. Further research into this method of utilising the MATLAB compiler is required in order to successfully use

this tool and end the dependence of the solution on the use of the MATLAB engine as this creates portability issues.

APPENDIX F DETAILED DESCRIPTION OF VARIABLE HANDLING BETWEEN PLANTSTAR AND MODEL CLASS

Looking at figure 6-2, The structure which a typical process unit of the plant takes, can be seen i.e. all its relevant process variables required to fully describe it at any point in time. These would then be updated if available from any plant measuring devices installed on the plant. PlantStar would then transfer this information to the DOM DLL through a referenced array. For example consider the simple case, which should be available on most plants including the one in question, the pulp level, H_o on the tree. The name, H_o, was chosen so as to maintain consistency with the model's definitions of variables and stands for Pulp Height observed on the plant.

In the inputs section of the process strategies algorithm, the input tree structure is displayed in figure 6-6 on which the H_o referenced variable is visible. The value property of this variable is as follows:

```
Plant Definition\Flotation circuit*\H_o\Value
```

which is similar to the Inflow_X1 variable that is shown in the diagram. This means that the referenced array, H_o, will look for the H_o value for each process unit in the Flotation circuit tree structure and place it into the H_o array in the order in which they are found. Therefore the first element in the H_o array would be the value of the Primary Rougher 1 pulp height and the last element in this array would be the last process unit that has a process variable named H_o and has an accompanying value set.

This array is then passed into the DLL on each time step and is converted into a cMatrix<double> format by the following line of code:

```
VariantToInput(&H_o, m.HoPS);
```

The "m" indicates the instance of the model class and "HoPS" is the pulp height value received from PlantStar, hence the "PS" suffix. All of the inputs to the model from the plant are processed in a similar way inside the DLL before being passed onto the model class.

If the pulp height of the Primary Rougher 1 was being observed, then the observation tree structure would have a 1 value against this element, namely element 18, in the observation parameter array of Unit 1. Refer to the diagram below and the highlighted value.

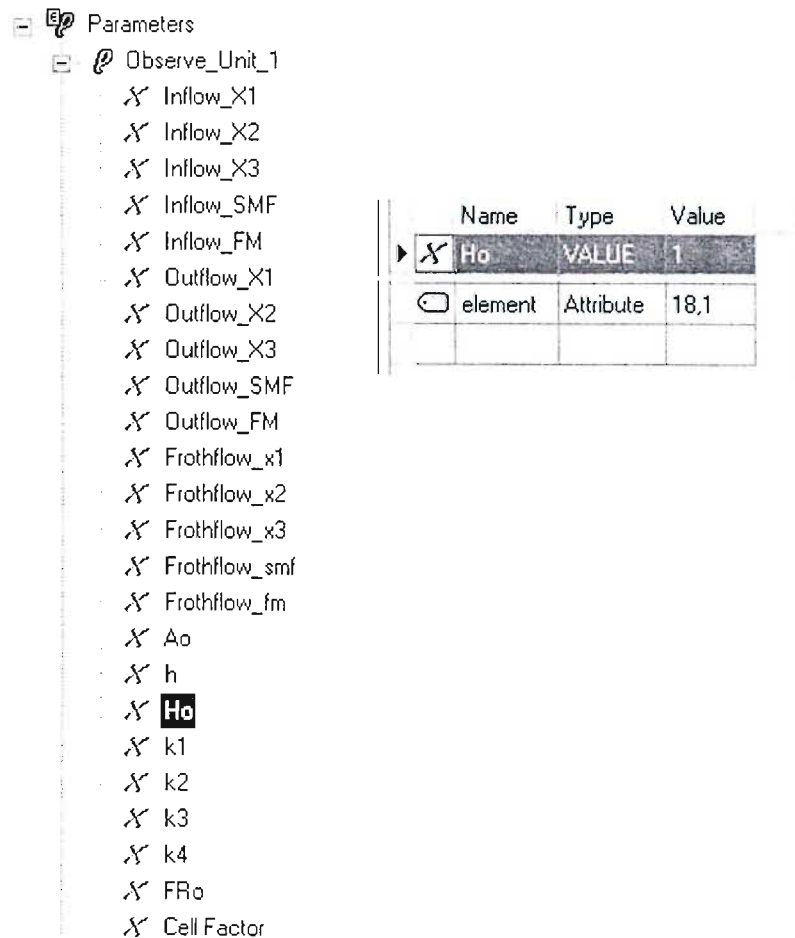


Figure F-1: The Observation Tree structure for Unit 1

It can be seen that the Observe_Unit_1 array has 24 elements in it and that the value of the 18th element is 1.

The model class uses the array of 'HoR', indexed by unit. As a result, there is no 18th element in this array and the entry regarding the observation of the pulp level for the Primary Rougher 1 or Unit 1 is in fact:

$$\text{HoR}(1,1) = 1$$

The first Unit is Primary Rougher 1 and therefore the first entry in the array is for Unit 1.

To overcome this, a large Observation matrix was created inside the DLL's class and all the observations were re-indexed in the following manner:

- 1) The observation array from PlantStar is converted into a `cMatrix<double>`, for example:

```
cMatrix<double> cObserve_Unit_1;  
VariantToInput(&Observe_Unit_1, cObserve_Unit_1);
```

- 2) The `cMatrix` observation array is placed into a large `cMatrix<double>` matrix, called `Observation_Matrix`, for example:

```
long num_var_Obs_Cells, num_var_Obs_Mills, num_var_Obs_Surge,  
num_var_Obs_PStreams; .
```

```
num_var_Obs_Cells = num_Observe_Unit_1; // = 24;  
num_var_Obs_Mills = 10;  
num_var_Obs_Surge = 10;  
num_var_Obs_PStreams = 5;
```

```
for (i=1;i<=num_var_Obs_Cells;i++)  
{  
    Observe_Matrix(1,i) = cObserve_Unit_1(i,1);  
}
```

This places the `c_Observe_Unit_1` array into the first row of the `Observe_Matrix`. All other observation arrays from PlantStar are handled in the same way and are placed into this one large matrix.

- 3) The arrays used by the model class are then referenced out of the `Observe_Matrix`, taking for example the observed variable of `Ho` for Unit 1:

```
for (i=1;i<=(m.C+m.MT);i++)  
{  
    m.HoRPS(i,1) = (Observe_Matrix(i,18)!=0);  
}
```


The value of `Observe_Matrix(1,18)` would be 1 and therefore the value of `m.HoRPS(1,1)` would also be 1 and the model class array would then correlate to the PlantStar array.

Once new values for all the variables are calculated by the extended Kalman filter, they are stored in variable arrays inside the Model class. Therefore a similar method to that used to re-index the observation arrays coming from PlantStar, is required to pass the data out of the DLL as arrays for each process unit.

Take for example the new predicted level of the pulp in the Primary Rougher 1 or Unit 1. This new value is in the array `H(1,1)` and must be placed into `cMatrix<double> Unit_1` as the 21st element so that it corresponds to the plant definition file structure as depicted below:

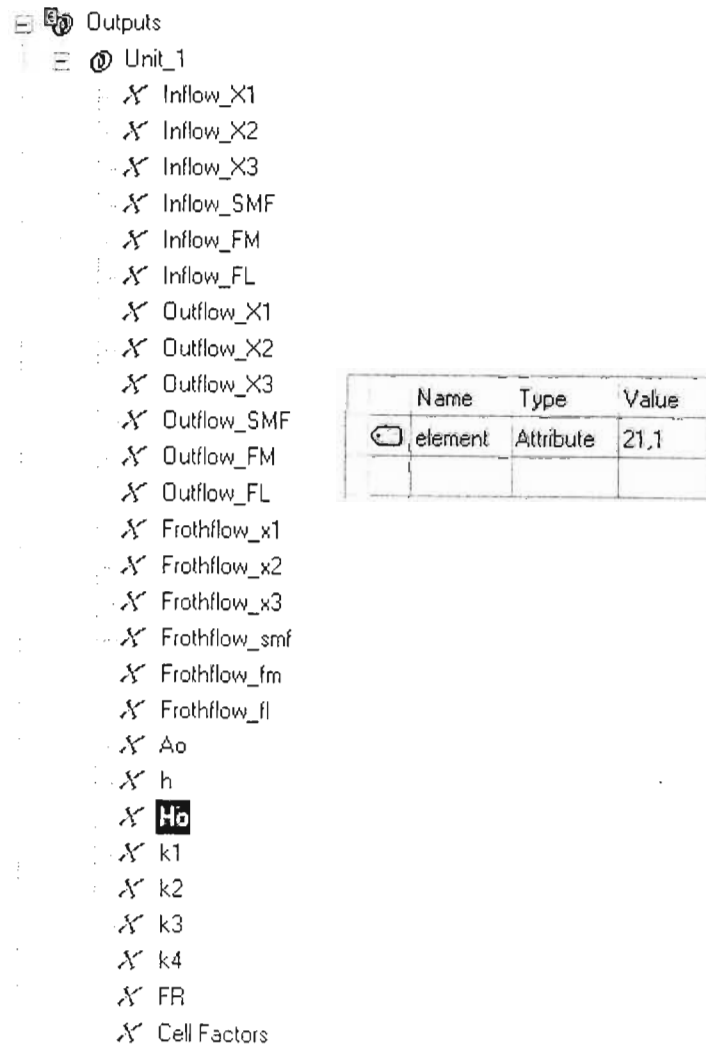


Figure F-2: The outputs tree structure in the plant definition file for Unit 1

To achieve this, a large output matrix is created called Output_Matrix. All of the process variable arrays from the model class are then placed into this one large matrix and the individual Unit's arrays are indexed out of it. For example, in the case of the predicted value for Unit 1 pulp height:

- 1) The model class arrays are placed into Output_Matrix as follows

```
for (i=1;i<=(m.C+m.MT);i++)
{
    if (m.HPS(i,1)) Output_Matrix(i,21)= m.HPS(i,1);
}
```

- 2) Once all of the model class arrays have been placed into the Output_Matrix each individual Unit can be taken out as the rows of Output_Matrix. The Units arrays are placed into cMatrix<double> arrays as follows for Unit 1 or cUnit_1:

```
cUnit_1.Init(27,1,0.0);
for (i=1;i<=27;i++)
{
    cUnit_1(i,1) = Output_Matrix(1,i);
}
```

- 3) Now cUnit_1 can be converted to a Variant to be passed back to PlantStar as follows:

```
OutputToVariant(Unit_1, cUnit_1);
```

Unit_1 is the Variant pointer defined in the Execute member function of the DLL and cUnit_1 is the internal array used to store the Primary Rougher 1 information.

APPENDIX G DART VALVE CALIBRATION

Flotation circuits usually use dart valves to control the pulp level and the flow of material from one flotation cell to the next. These dart valves are normally on some form of computer control and therefore are an opportunity to obtain some idea of material flows around the plant. MINTEK developed a program known as FloatStar® that controls pulp level of flotation circuits and minimises disturbances. This program is available as a “plugin” module with PlantStar and therefore provides valve positions to the control software. Some testing was carried out on the pilot plant at LONMIN in order to calibrate their dart valves. The problem with this form of calibration is that it is empirical and dart valve specific. Also the material flow through the dart valve is not uniform and may vary depending upon the amount of agitation in the cell. However to obtain some indication of whether this is a viable indicator of material flows, water was passed through the cells and the flowrates measured at different dart valve positions.

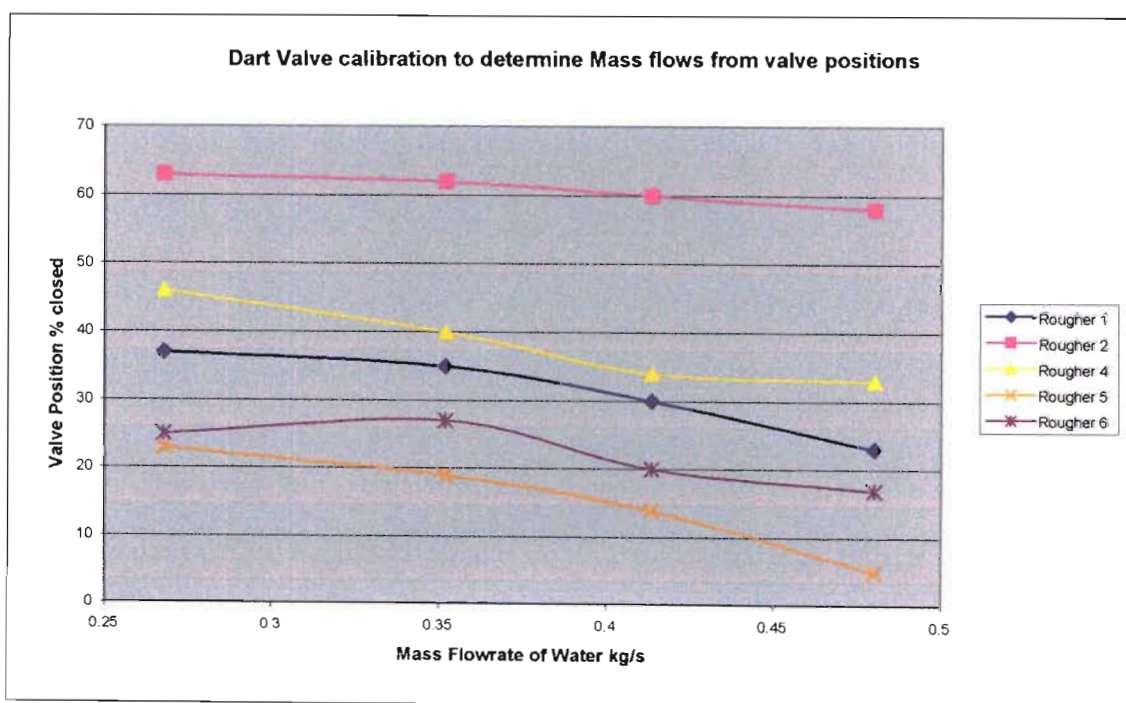


Figure G-1: Dart valve calibration to determine mass flowrates from valve positions on LONMIN’s pilot plant

The results above are promising, however more tests need to be carried out with pulp flow through the valves instead of water, to obtain a more accurate representation. If all of the dart valves on the plant are calibrated in this way, the mass flows around the circuit become identifiable by simply looking at the dart valve position.

APPENDIX H MODEL TUNING AND SOLUTION PARAMETERS

In order to accommodate the wide range of operating conditions on the plant, the model's solution is flexible. This accommodates real-time model solution tuning by making use of a pertinent set of model tuning parameters that are available in real-time through the model tuning parameter GUI. All of the solution parameters will be described here for completeness.

H.1 MODEL SOLUTION PARAMETER SET

Within the "model.cpp" file mentioned in section 6.2.2.2 there is a solution parameter block in which key solution control parameters are defined. Some of these solution control parameters are made available to the user by means of the model tuning parameters GUI, but most of them are set statically in this section of code. What follows is a short description of each of them obtained from, Mulholland et al. (2003) along with its default value used in the modelling of LONMIN Eastern Platinum's B-Stream flotation circuit.

- dtmax = 0.02: The maximum time step to be used in the solution (h).
- pcmove_tol=10: Maximum percentage move for any one variable before re-evaluation of Jacobians (%).
- pertfr = 0.01: Fraction of *Range* (upper constraint - lower constraint) to be used as a perturbation step in periodic re-determination of system Jacobians
- minrespfr = 0.0*pertfr: Minimum recognised fractional value response of right-hand-sides of differential and algebraic equations on perturbation (otherwise term omitted, which has a benefit in Sparse modes: $0.01 * pertfr$ has been used at times).
- minrange=1e-8: Protection against convergent constraints: the minimum value of *Range* is restricted to this for determination of diagonal elements of

the Kalman Filter error covariance matrices **Q** and **R** (units are those of the variable concerned).

nexpm_max=1000: Maximum number of iterations allowed for convergence of state transition matrices $[e^{\phi\Delta t} - \mathbf{I}]\phi^{-1}$ equation 4-22 and $e^{\phi\Delta t}$ equation 4-21.

tol = 1e-3: Fractional change tolerance for state transition matrix (matrix exponential) convergence.

SM = 1e-8: Smallest allowed Flow or Mass Inventory value to protect against divide-by-zero (units are those of the variable concerned: ie. th^{-1} or t).

SMM = 1e-12: Smallest allowed dy_{jj} (perturbation response), or g (equation sensitivity based on largest linear term), or variable size used in fractional observation error report (units are those of the variable concerned: ie. th^{-1} , t, m (level), m^3h^{-1} (air flow), or h^{-1} (k values)).

SMMM = 1e-14: Smallest value allowed for perturbation response or corresponding Jacobian term (units are effectively those of the variable concerned: i.e. th^{-1} , t, m (level), m^3h^{-1} (air flow), or h^{-1} (k values) OR the ratio of any two of these). *SMMM* is also used as the weeding threshold in *changemat* in the sparse calculation (section 6.2.3.3) of the state transition matrices.

SMweedMt = 0: When the Kalman Filter covariance matrix *Mt* is calculated using sparse matrix techniques, this is the smallest size of element left in the matrix at the weeding step. If this parameter is set as a negative, a minimum of two elements is left in any one row regardless of size (units depend on the variables concerned: ie. the product of any two of: th^{-1} , t, m (level), m^3h^{-1} (air flow), or h^{-1} (k values)).

SMweedCM = 0: When the Kalman Filter covariance matrix *Mt* is updated using sparse matrix techniques (section 6.2.3.3), this is the smallest size of element left in the intermediate *CM* matrix at the weeding step. If this parameter is set as a negative, a minimum of two elements is left in any one row regardless of size (units depend on the variables concerned).

SMweedCMCI = 0: When the Kalman Filter covariance matrix **Mt** is updated using sparse matrix techniques (section 6.2.3.3), this is the smallest size of element left in the intermediate *CMCI* matrix at the weeding step. If this parameter is set as a negative, a minimum of two elements is left in any one row regardless of size (units depend on the variables concerned).

SMc = 1e-7: Smallest allowed upper constraint (units are those of the variable concerned: ie. th^{-1} , t, m (level), m^3h^{-1} (air flow), or h^{-1} (*k* values)).

SMcf = 0.01: Factor to determine smallest allowed upper constraint as a fraction of the actual or default observation value.

SMsf=0.000001: Smallest solids flow allowed, used e.g. to set non-zero value of solids flow in a pure water stream, for divide-by-zero protection (th^{-1}).

SMlf=0.0001: Smallest liquid flow allowed, used to prevent divide-by-zero (th^{-1}).

BIG = 1e8: Maximum allowed Jacobian term (function derivatives by variables) (units depend on the variables concerned: ie. the ratio of any two of: th^{-1} , t, m (level), m^3h^{-1} (air flow), or h^{-1} (*k* values)).

Matrix_Solution_TOLERANCE_fraction = 1e-3: Convergence tolerance of Conjugate Gradient method used in the sparse option for solution of the linear system of equations for the Kalman filter gain.

Matrix_Solution_Shrink_Tol = 1e-8: Sparse matrix weeding threshold invoked optionally in Conjugate Gradient method used in the sparse option for solution of the linear system of equations for the Kalman filter gain.

Mt_START=0.01: Value used to initialise the entire diagonal of the Kalman filter covariance matrix \mathbf{M}_t at startup, or, if requested, when the observation pattern changes. A bigger value will give lower filter gain initially, and slower adjustment of the variables initially.

nFirstKrecalc = 1: On the first occasion that the Kalman filter gain is calculated, ie. after initialising \mathbf{M}_t above, using the recursion of equations 4-23 and 4-25, the sequence 4-23, 4-25 is repeated *nFirstKrecalc* times, aiming to achieve the steady-state Kalman gain which would apply under the initial prevailing conditions.

Kint = 10: Once the process is quite steady, new evaluations of the Kalman gain matrix \mathbf{K}_t (and \mathbf{M}_t) will not be triggered e.g. by *pcmove_tol* above. To ensure that this matrix is up-to-date and convergent, it is forced to be re-evaluated *Kint* time-steps after the last evaluation.

QR100 = 1: Denominator factor of square-root of diagonal elements of both \mathbf{Q} and \mathbf{R} error covariance matrices used in Kalman filter. Complete \mathbf{Q} and \mathbf{R} matrices are scaled equally, so there is no net effect on the Kalman filter gain. However, when solutions are found using the sparse option, this factor can be used to manipulate the average size of terms appearing in certain equations, affecting sparseness and accuracy.

fast_response_factor=0.5: This is a factor applied to the basic model time-step dt , (which is close to the maximum allowed step $dtmax$), in order to determine a relatively fast time-constant τ for use in equation 4-12, to determine the speed of response of the “incidental variables” \mathbf{z} (flows etc.). The same τ is used to determine the minimum allowed time-constant for the only other short-response equations, the froth mass-balance differential equations. The maximum of either τ or the correct time-constant (mass inventory / mass throughput) is used. When this is restricted to τ for any one cell, the warning “*Integration in froth artificially slowed down*” is reported (if the fractional adjustment is below *SlowReportLevel*). Bearing in mind the type of time-resolution required of the model, this minor interference to reduce the stiffness of the set is certainly justified by the larger integration time-step permitted.

yo_err_factor =0.05: This is a factor used as a multiplier in the square-root of all \mathbf{Q} and \mathbf{R} terms which concern the tracking of observed variables (For state variables these will appear in the first terms of \mathbf{R} and for observed incidental variables they will appear in the latter terms of \mathbf{Q}). It is used as an overall adjustment on the observation (y_o) tracking: smaller values will cause closer tracking (units are effectively the inverse of the variable concerned).

yu_err_factor=1.0: This is a factor used as a multiplier in the square-root of all \mathbf{Q} terms which concern the unobserved incidental variables. It is used as an overall adjustment on the rate of migration of these values as they attempt to balance the algebraic equations. For smaller values, greater “move-suppression” is achieved (units are effectively the inverse of the variable concerned).

- f_err_factor=0.1: This is a factor used as a multiplier in the square-root of all **R** terms which concern the algebraic equations (ie. to force these equations to balance to zero). It is used as an overall adjustment on the importance allocated by the filter to balancing the algebraic equations - the smaller this term, the more accurately they will balance (units are effectively the inverse of the variable concerned).
- Rfactor = 1: This is a factor used as a multiplier in the square-root of all **R** terms. It is used as an overall adjustment on the importance allocated to what would normally be the “observation” part of a Kalman filter, and thus has a direct bearing on the overall filter gain - the smaller *Rfactor*, the greater the filter gain, in general (units are effectively the inverse of the variable concerned).
- MinFrothHeight=0.01: Minimum allowed froth height, to maintain a minimal inventory (m).
- FrScale=1000: Scaling factor used to obtain larger numerical values for froth inventories. At 1000 it represents these as kg instead of t. This adjustment is useful in the case of *sparse* solution options, to reduce numerical round-off error (kg/t).
- SlowReportLevel=0.80: As discussed under *fast_response_factor* above, some froth derivatives may be slowed down to reduce stiffness. When any one froth is reduced to less than *SlowReportLevel* of its correct speed of response, a warning message reports “*Integration in froth artificially slowed down*”.
- ncalls_until_reevaluation=30: Once the process is quite steady, new evaluations of the Jacobians and the **Q** and **R** matrices will not be triggered e.g. by

pcmove_tol above. To ensure that these matrices are up-to-date, they are forced to be re-evaluated after *ncalls_until_reevaluation* time-steps after the last evaluation. This also leads to a Kalman Filter covariance \mathbf{M}_t and gain \mathbf{K}_t re-evaluation.

autoexpand=0.4: See section H-2. The current value of the 'c' parameter of any one variable is recalculated on each time-step to be the maximum of either *shrink* × (present 'c' value) or $(1 + \text{autoexpand}) \times (\text{present value of the variable})$. The upper constraint is in general subsequently defined as $2 \times (\text{'c' parameter})$.

shrink=0.95: Factor for shrinking constraints if a variable decreases - see section H-2 and *autoexpand*.

MAX_av_obserr_pc=5: Average percentage deviation from observations (and percent error in algebraic equations) before reporting using "*Av Observation error PC greater than limit*". The average deviation is calculated as the arithmetic average of percentage deviation for just those variables which are presently being observed, PLUS the imbalance for each algebraic equation, expressed as a percentage of the largest linear term in each equation. Thus, for example if there are 10 observations each deviating by 10%, and 10 equations, each deviating by 5%, a 7.5% error will be reported (The *av_obserr_pc* is also continuously available for plotting).

MAX_av_deriverr_pc = 5: If the errors specified for the first few terms of the \mathbf{Q} matrix, affecting the state variables, are too large, the states can appear to be steady, yet the right-hand-sides of the differential equations do not evaluate to zero. To detect this situation, the *MAX_av_deriverr_pc* defines a threshold for *av_deriverr_pc*, above which "*Av Derivative*

error PC greater than limit" is reported. Clearly the *av_deriverr_pc* term, which is continuously available for plotting, etc, is expected to be non-zero during transients, but it should never be persistently non-zero. It is evaluated as the percentage ratio of the derivative to the largest linear term on the right-hand-side, averaged for all differential equations. One notes that if both *av_obserr_pc* and *av_deriverr_pc* go to zero, one has a fully reconciled steady-state system, without any compromise between observations.

LinSolnMethod = 3: There is a choice of 4 solution methods for the most time-consuming task on a single time-step. This is the evaluation of the Kalman gain matrix **K** using the equation

$$\mathbf{K} = \mathbf{M} \mathbf{C}^T [\mathbf{C} \mathbf{M} \mathbf{C}^T + \mathbf{R}]^{-1}$$

The inverse is not required, so this is effectively solved directly as a linear set, using MATLAB notation (see equations 4-28 and 4-29):

$$\mathbf{K}^T = \mathbf{C} \mathbf{M}^T / \mathbf{C} \mathbf{M} \mathbf{C}^T$$

CM and **CMCI** each representing a single matrix. We note that the arrays are generally large. In the accompanying example, LONMIN Eastern Platinum circuit, there are over 200 variables, and over 200 equations, plus the possibility of a large number of observations.

Method 0: NONSPARSE FILES PASSED TO EXTERNAL MATLAB .exe for NONSPARSE SOLUTION: The matrices **CM**, **CMCI** are written in full to text files in the C:\ root directory. Then the standalone pre-compiled MATLAB program, *SolveFilesNS_Croot.exe*, is executed through a *_spawnlp* command with a *_P_WAIT* for completion. It uses full non-sparse matrix methods. Before that program ends, it writes the result file for **K** as text to the C:\. This is then read for use in the model. NOTE: The files **M,CM,CMCI** are still subject outside of MATLAB to the

weeding thresholds $SMweedMt$, $SMweedCM$ and $SMweedCMCI$ respectively, so these must be set to zero for a completely non-sparse solution.

Method 1: SPARSE FILES PASSED TO EXTERNAL MATLAB .exe for SPARSE SOLUTION: The matrices **CM**, **CMCI** are written in sparse form to text files in the C:\ root directory. Then the standalone pre-compiled MATLAB program, *SolveFiles_Croot.exe*, is executed through a *_spawnlp* command with a *_P_WAIT* for completion. It uses full sparse matrix methods. Before that program ends, it writes the sparse result file for **K** as text to the C:\. This is then read for use in the model. NOTE: The files **M,CM,CMCI** are subject outside of MATLAB to the weeding thresholds $SMweedMt$, $SMweedCM$ and $SMweedCMCI$ respectively - these must be set above zero to invoke some sparseness.

Method 2: SPARSE METHODS USING OWN SPARSE MATRIX HANDLING ROUTINES AND CONJUGATE GRADIENT LINEAR SOLUTION: The files **M,CM,CMCI** are subject to the weeding thresholds $SMweedMt$, $SMweedCM$ and $SMweedCMCI$ respectively - these must be set above zero to invoke some sparseness. Then, within the Conjugate Gradient routine (overridden "operator /") another weeding threshold operates: *Matrix_Solution_Shrink_Tol* (see above). A convergence tolerance *Matrix_Solution_TOLERANCE_fraction* must also be set (see above).

Method 3: PASSING OF FULL MATRICES THROUGH MEMORY TO CO-RESIDENT MATLAB ENGINE, EMPLOYING FULL-MATRIX TECHNIQUES: This is the fastest and most accurate method, employing a sophisticated linear solution optimiser within MATLAB (Method 1 presumably also uses this, but is

apparently slowed down by the file transfers). However, Method 3 does require that MATLAB itself is running on the computer: The Model program starts and stops MATLAB using *engOpen* and *engClose* commands.

FilterMethod = 3: Amongst the top three consumers of computation time was the evaluation of the matrix equation:

$$\mathbf{M} = \mathbf{A} [\mathbf{I} - \mathbf{K} \mathbf{C}] \mathbf{M} \mathbf{A}^T + \mathbf{Q}$$

Again, the user can select from several solution methods:

Method 0: NONSPARSE FILES PASSED TO EXTERNAL MATLAB .exe for NONSPARSE SOLUTION: The matrices are written in full to text files in the C:\ root directory. Then the standalone pre-compiled MATLAB program, *SolveFilterNS_Croot.exe* is executed through a *_spawnlp* command with a *_P_WAIT* for completion. It uses full non-sparse matrix methods. Before that program ends, it writes the result file for **M** as text to the C:\. This is then read for use in the model. NOTE: The file **M** is still subject outside of MATLAB to the weeding threshold *SMweedMt.* respectively, so this must be set to zero for a completely non-sparse solution.

Method 1: NONSPARSE MATRICES PROCESSED USING M.MULHOLLAND'S FULL MATRIX ROUTINES for NONSPARSE SOLUTION: The overridden operators available with the cMatrix Class provided by others, allows the direct evaluation of the expression.

Method 2: SPARSE METHODS USING M.MULHOLLAND'S SPARSE MATRIX HANDLING ROUTINES: The files **A**, **K**, **C** and **Q** are not subjected to weeding, but **M** can be weeded outside of this calculation depending on *SMweedMt*.

Method 3: PASSING OF FULL MATRICES THROUGH MEMORY TO CO-RESIDENT MATLAB ENGINE, EMPLOYING FULL-MATRIX TECHNIQUES: This is the fastest method. However, it does require that MATLAB itself is running on the computer: The Model program starts and stops MATLAB using *engOpen* and *engClose* commands.

REINITIALISE Mt FOR NEW OBSMODE = 1: The matrix **M** has a fixed dimension (side = number of variables), and does not change in size as the observation pattern changes (unlike **C**, **K**). A few observation changes will have a limited impact on the elements, so it will generally be less disruptive to continue updating (equations 4-23, 4-25) from the last value of **M** after an observation pattern change, rather than to re-initialise this Kalman filter covariance matrix (diagonal reset to *Mt_START*). Nevertheless, this option is provided here for those who think differently. In tests done, little noticeable disruption occurs by either option.

speedupfactor = 1.0: For normal real-time pace, this should be 1.0. It can be set differently directly from the PlantStar GUI. Once the "Historical Playback" and "Play forward" options are implemented, there will be a case to operate the solution faster than real-time. This feature has proved useful in development and diagnosis.

The following are the tuning and solution parameters which were deemed the most pertinent to the solutions and as such were made available in real-time to the user through the model tuning parameters graphical user interface (see figure 6-15):

QR100

Fast response factor

yo error factor

yu error factor

f error factor

R factor

Linear solution method

Filter solution method

Re-Initialise Covariance matrix

Model Speed-Up factor

H.2 AUTOMATIC COVARIANCE MATRIX TECHNIQUE

Some form of an expression was required to determine the likely maximum value of each variable automatically, for example

$$SSc(i,j) = _max(Xc(i,j)*Hw(i)*AF(i)*rhoSmax$$

where the maximum inventory of component j in cell i is its maximum mass fraction multiplied by the pulp inventory based on the maximum pulp density of material at the plant.

The 'c' suffix denotes that a variable is a range value, whilst the 'o' suffix denotes an observed value. This made the handling of variables inside the code consistent and easily readable. At startup the range variables can be set to the default observation set or the last saved set in the data file "LASTCONSTRAINTS.bin", after which the range is shrunk or expanded every model time step according to


```

for (i=1; i<= Xc.nRow; i++)
  for (j=1; j<=Xc.nCol; j++)
    Xc(i,j) = __max( __max(SMc*f*__max(Xo(i,j),Xtyp(j)),
      __max(shrink*Xc(i,j), (1+autoexpand)*X(i,j))) , SMc);

```

Here *shrink* continuously attempts to reduce the ‘c’ variable, whilst *autoexpand* will not let it drop below a point 40% above its present value. Moreover if a variable finds itself higher, the ‘c’ value will expand on each step until it is itself 40% higher than the new value. On the other side of the scale, the ‘c’ variable is prevented from dropping below a maximum of either a fixed minimum value, *SMc* or a set small fraction of the actual or default observation, *SMcf* (Mulholland et al., 2003).

The constraint variables (‘c’) thus define a value, which the states and incidental variables (such as flows) are not expected to exceed. For all the model variables, the corresponding ‘c’ values are multiplied by 2 to determine an *upper constraint*. A *lower constraint* is also defined for all variables as zero, except in the cases of the flotation rate constants, which are prevented from dropping below 0.1% of their nominal observation values. In the operation of the Kalman filter, those variables that attempt to move outside the constraint range are ‘clipped’ back to the corresponding limit. The upper constraint will then expand to accommodate such variables on subsequent time steps. In this way a meaningful range of each variable is maintained to form the basis of the calculation of the expected modelling or measurement errors, **Q** and **R** respectively, as a nominal error fraction of the present constraint range. This form of internal “normalisation” will result in, for example, a 10% adjustment of platinum mass fraction of 4.6×10^{-6} a 10% adjustment in chrome mass fraction of 0.24 (Mulholland et al., 2003).

In a typical solution, the following **Q**, **R** policy has been found effective:

observed state: **R** diagonal element as: $(Error * y_{o_err_factor} * Rfactor * Range / QR100)^2$

equations **R** diagonal element as: $(Error * f_err_factor * Rfactor * Range / QR100)^2$

states: **Q** diagonal element as: $(Error * f_err_factor * Range / QR100)^2$

observed incidental: **Q** diagonal elements as: $(Error * y_{o_err_factor} * Range / QR100)^2$

unobserved incidental: Q diagonal elements as: $(Error * yu_err_factor * Range / QR100)^2$

Typical values used for *states* and *incidental variables* have been:

Error: 0.1 for OBSERVED variables and 1.0 for UNOBSERVED variables

yo_err_factor: 0.05 (for overall adjustment of observation tracking)

f_err_factor: 0.10 (for overall adjustment of compliance with differential and algebraic equations)

yu_err_factor: 1.00 (move suppression of unobserved incidental variables)

Rfactor: 1.00 (overall adjustment of all **R** elements)

QR100: 1.00 (overall adjustment of all **Q** and **R** elements for numerical reasons)

It is only in the case of the flotation rate constants k that *Error* goes to 10.0 instead of 1.0 for the UNOBSERVED case, in order to enhance the mobility of k in identification mode. An option also exists to set a fixed absolute error for any variable (by setting it as the negative). In the cases of the *algebraic and differential equations*, the *Range* is determined by the largest linear term in the equation expression (whilst for the variables it is the difference between upper and lower constraints). The *Error* values used in the *equations* are typically 0.1 for the differential equations, and 1.0 for the rest (Mulholland et al, 2003).

APPENDIX I COMPLETE MIMICS AND GRAPHICAL USER INTERFACES (GUI)

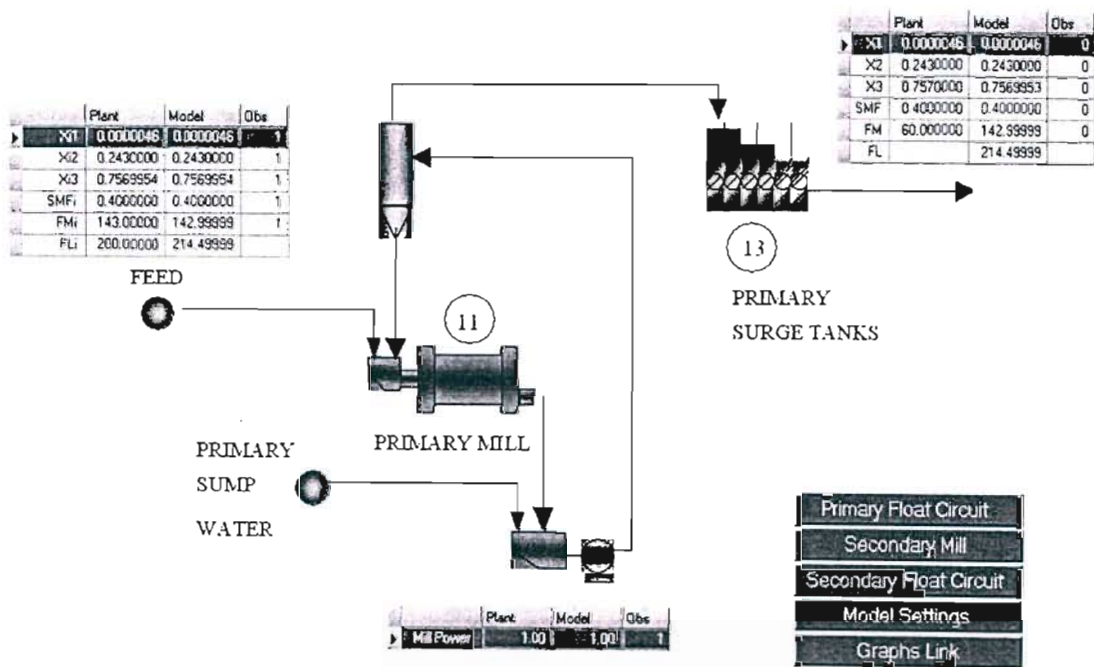


Figure I-1: The Primary Mill GUI

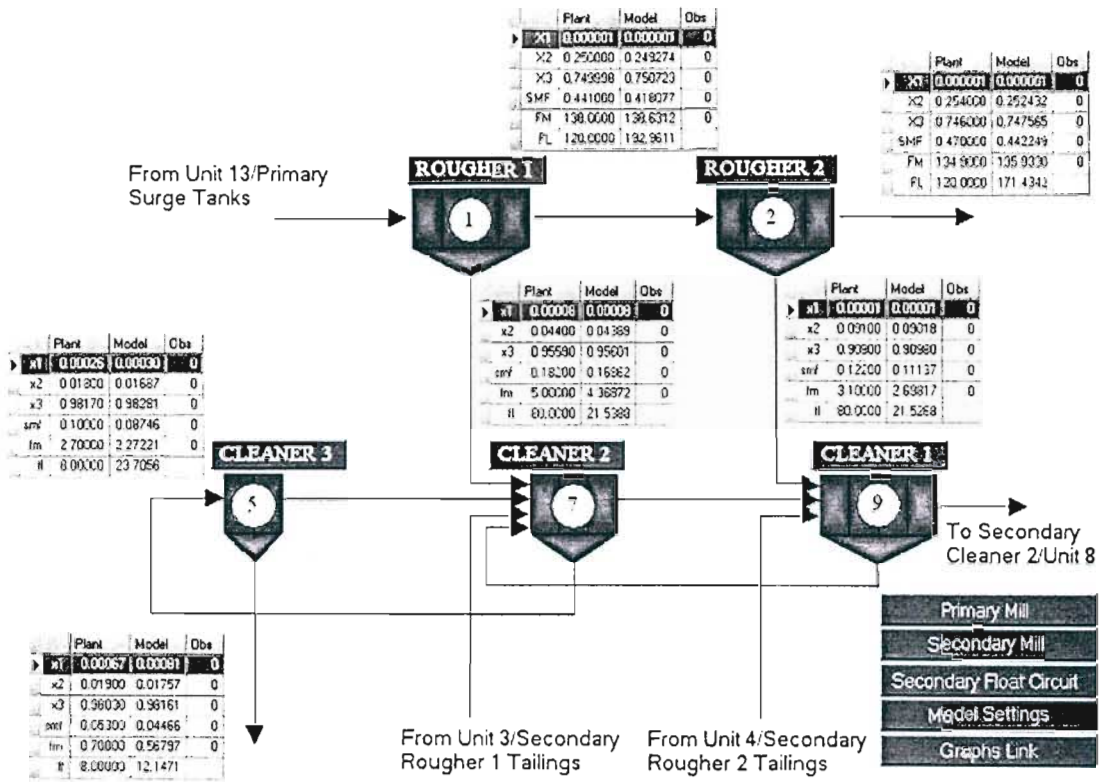


Figure I-2: The Primary Flotation Circuit GUI

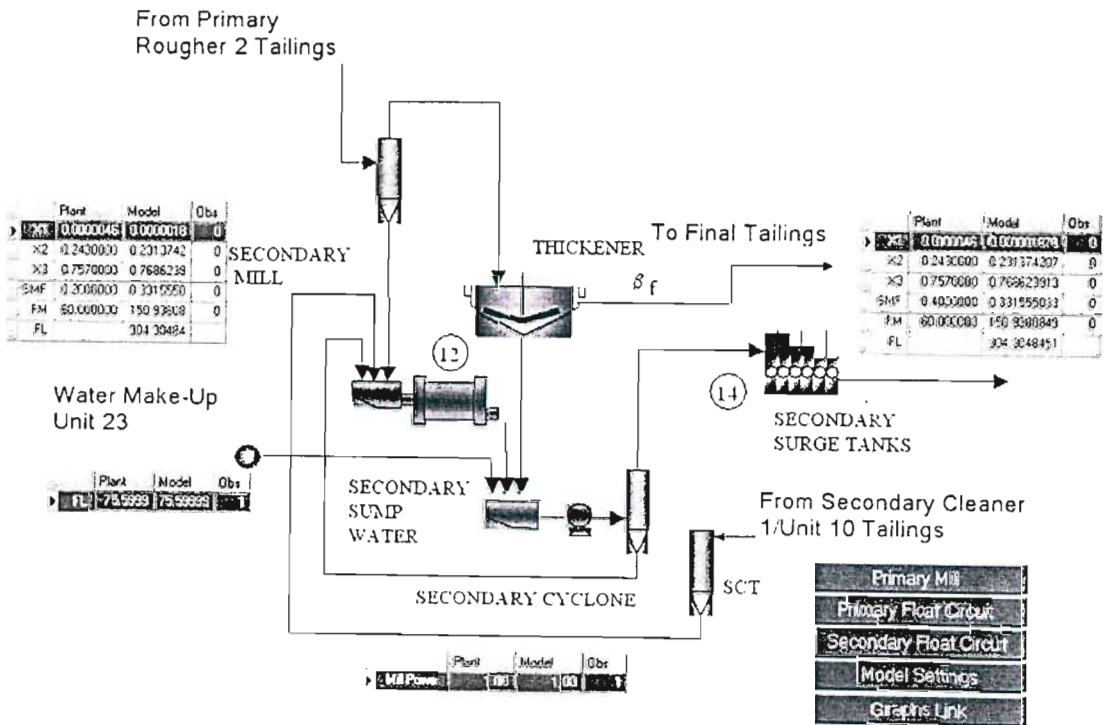


Figure I-3: The Secondary Mill GUI

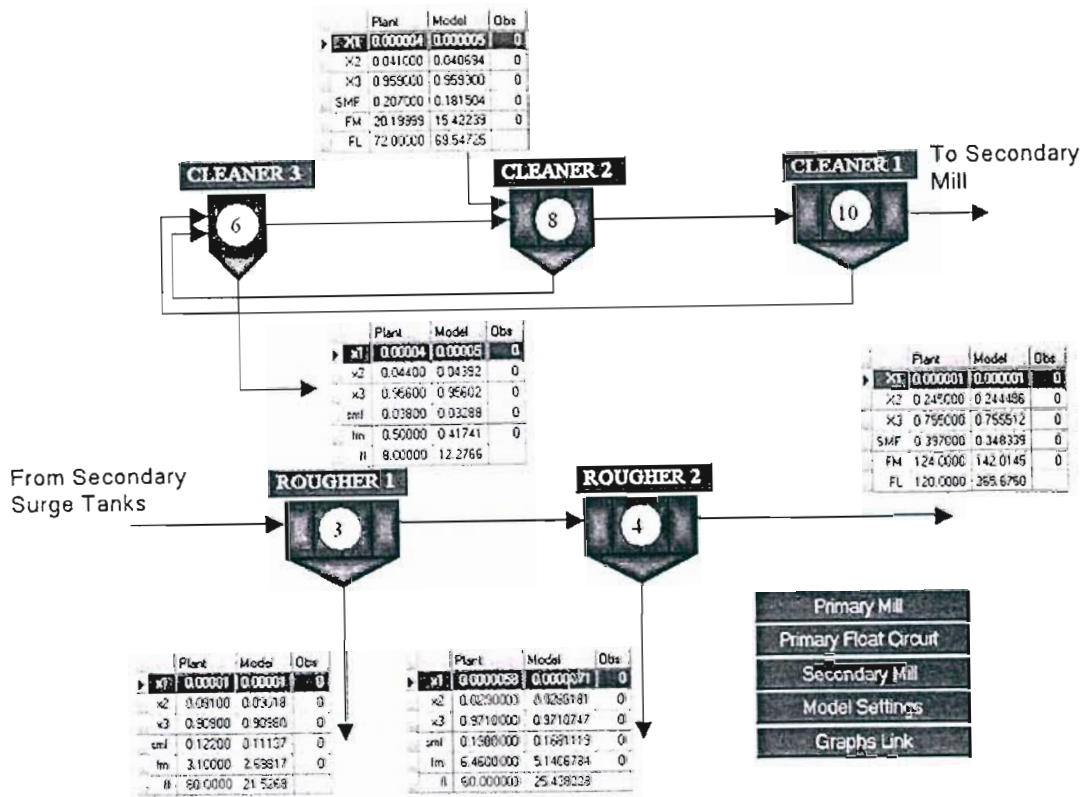


Figure I-4: The Secondary Flotation Circuit GUI

Model Settings and Tunings parameters

Accept Observation Changes?

On Shutdown Save...

Save Current Constraints

Save Current Variables

Save Current Covariance Matrix

Save Current Observation Status

Start Time for Forward Run

Time	Value
Hours	0
Minutes	0
Seconds	0
Forward run length	0

Model Startup Conditions

Flag	Value
Use Saved Constraints	1
Use Saved Variables States	1
Use Saved Covariance Matrix	1

Model Tuning Parameters

Parameter	Value
QR100	1.000
fast response factor	1.000
yo error factor	0.050
yu error factor	10.000
i error factor	0.100
R factor	1.000
Linear Solution Method	3.000
Filter Solution Method	3.000
Retrain/cov Covariance matrix	1.000
MODEL SPEED UP FACTOR	2.000

Model Operation Mode: 0

Model Performance

Value	
Observation Error	0.00000
Derivative Error	0.00000

Model Messages

WARNING: State integral is being artificially slowed down (409.00000)

Primary Mill
Primary Float Circuit
Secondary Mill
Secondary Float Circuit
Graphs Link
Whole Plant

Figure I-5: The Model Parameters GUI

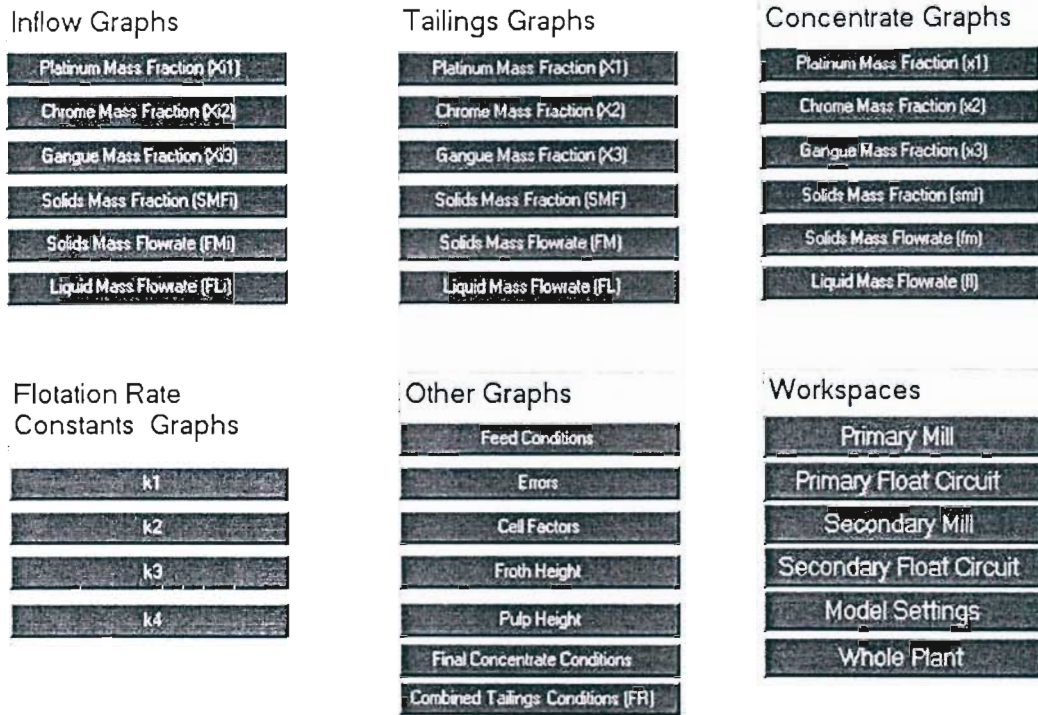


Figure I-6: The Graph Links GUI

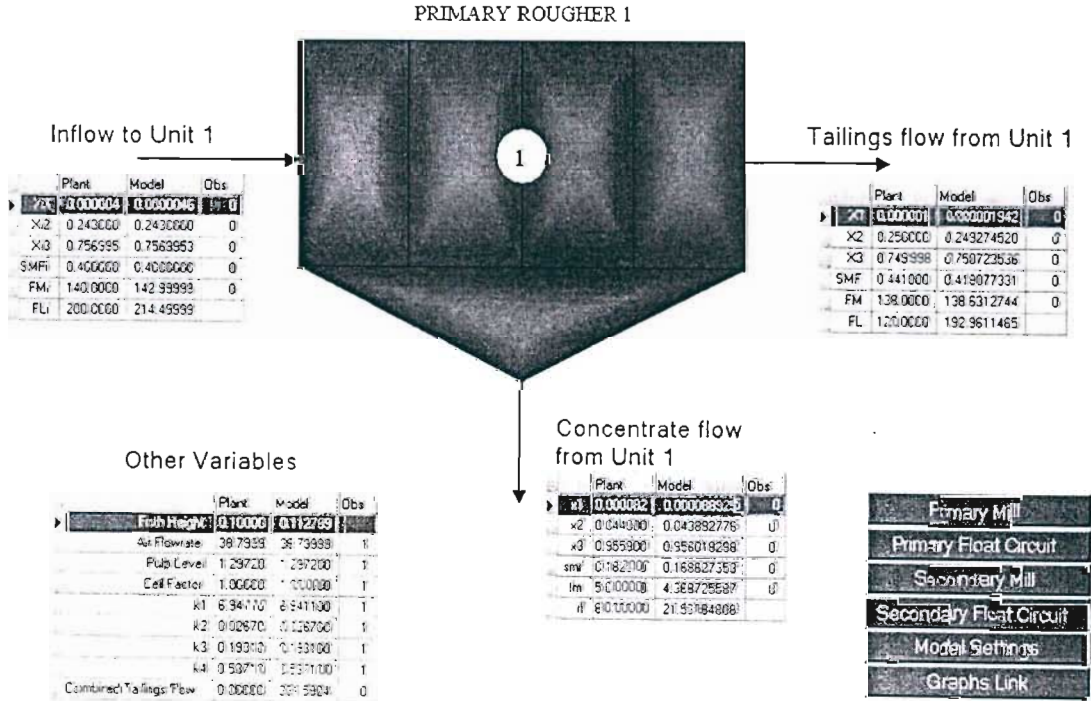


Figure I-7: Detailed GUI of Unit 1

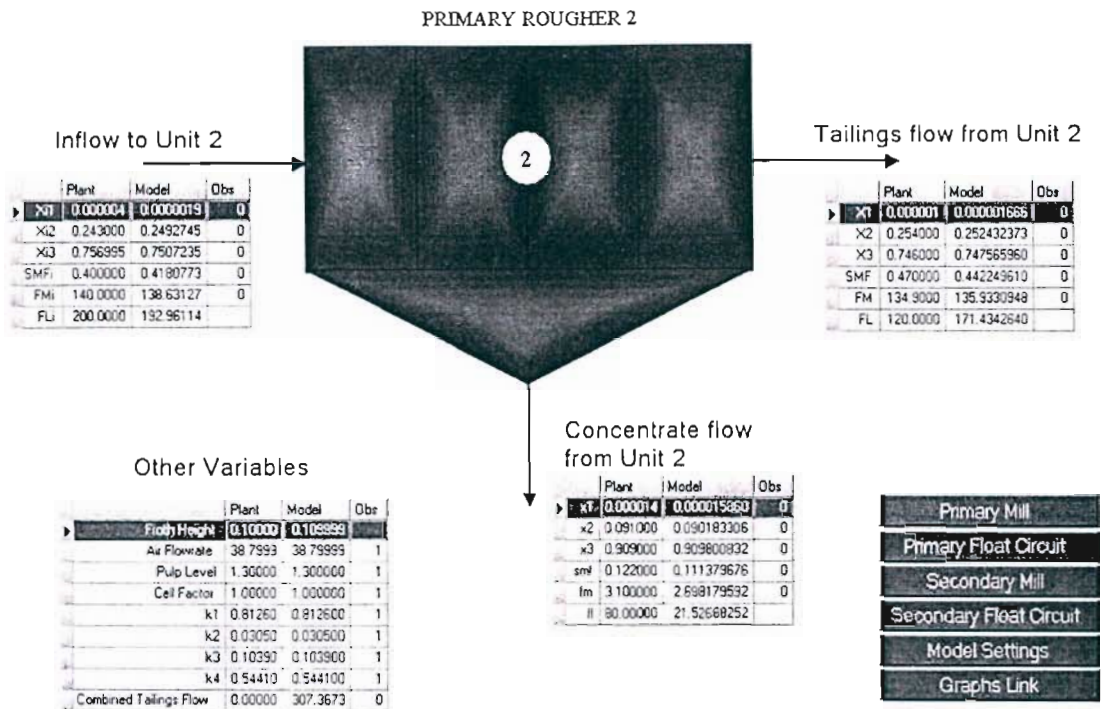


Figure I-8: Detailed GUI of Unit 2

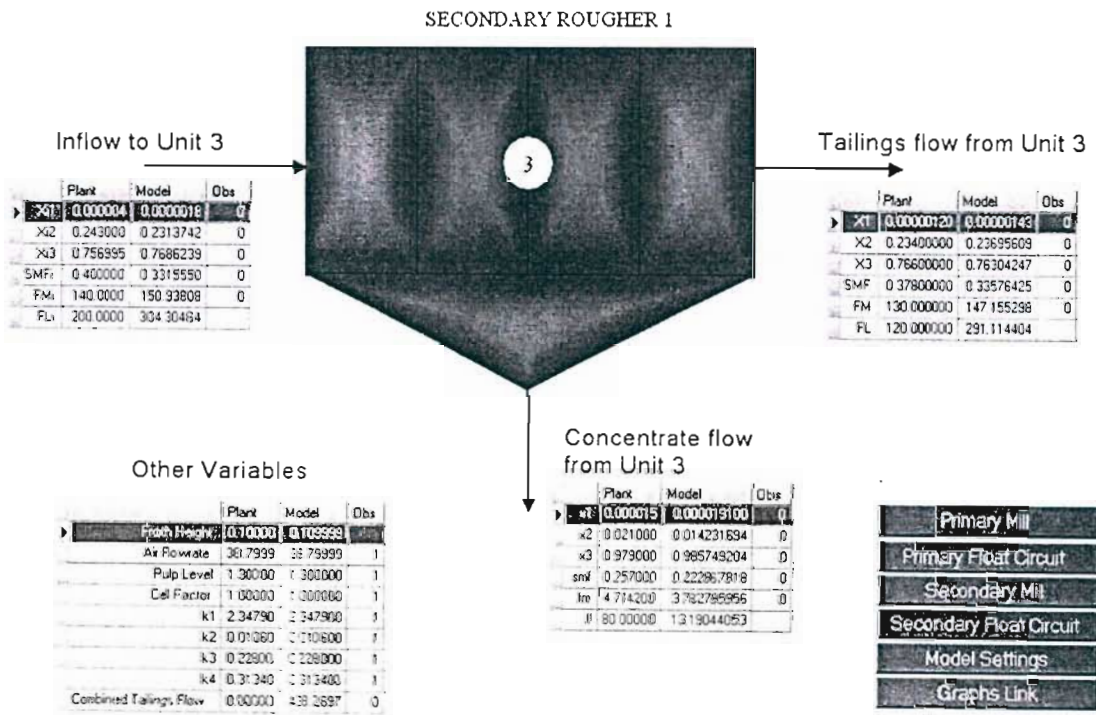


Figure I-9: Detailed GUI of Unit 3

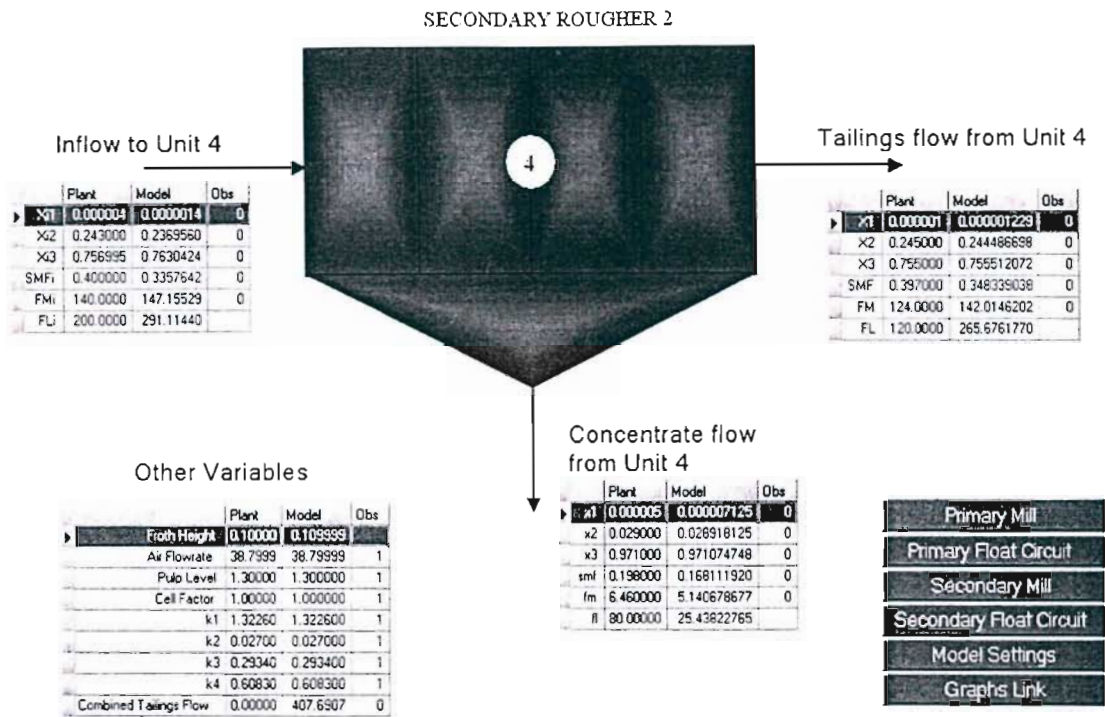


Figure I-10: Detailed GUI of Unit 4

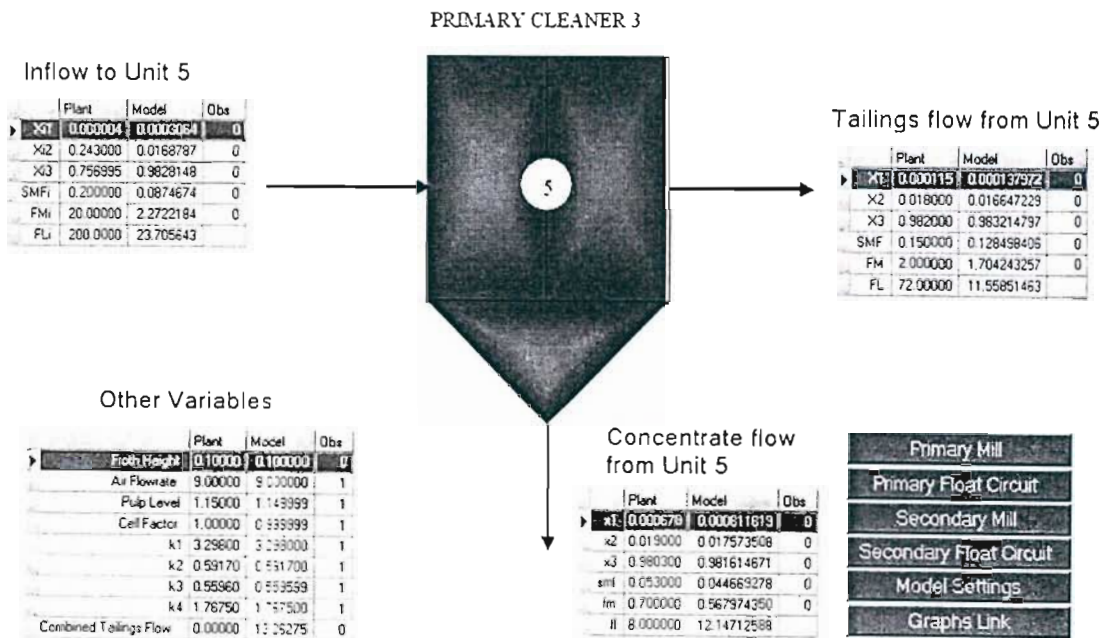


Figure I-11: Detailed GUI of Unit 5

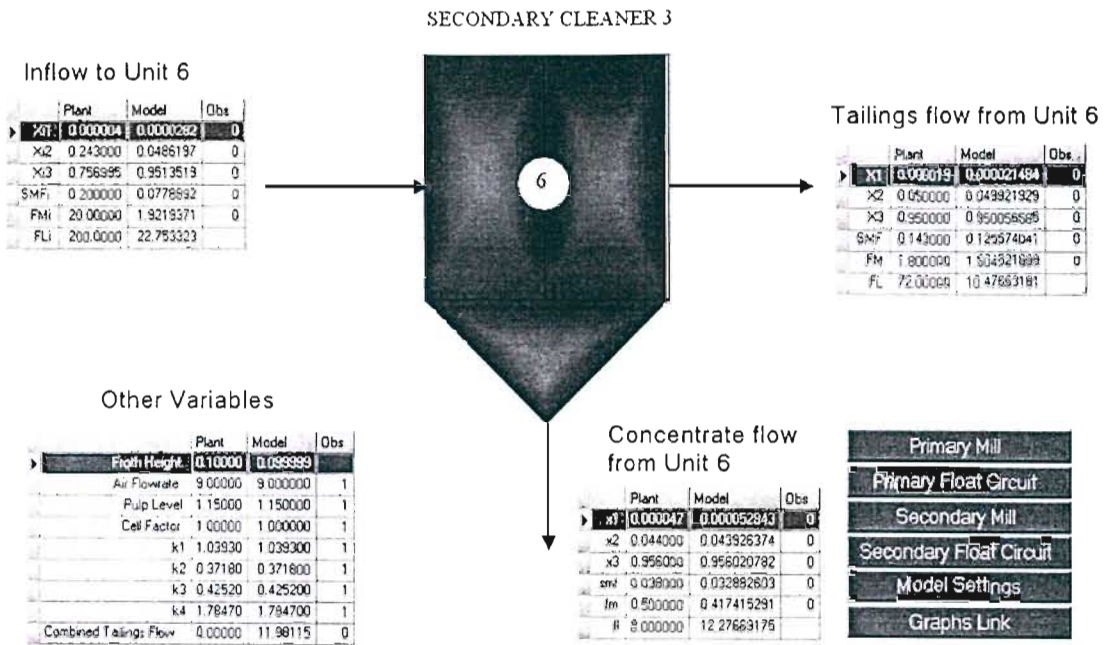


Figure I-12: Detailed GUI of Unit 6

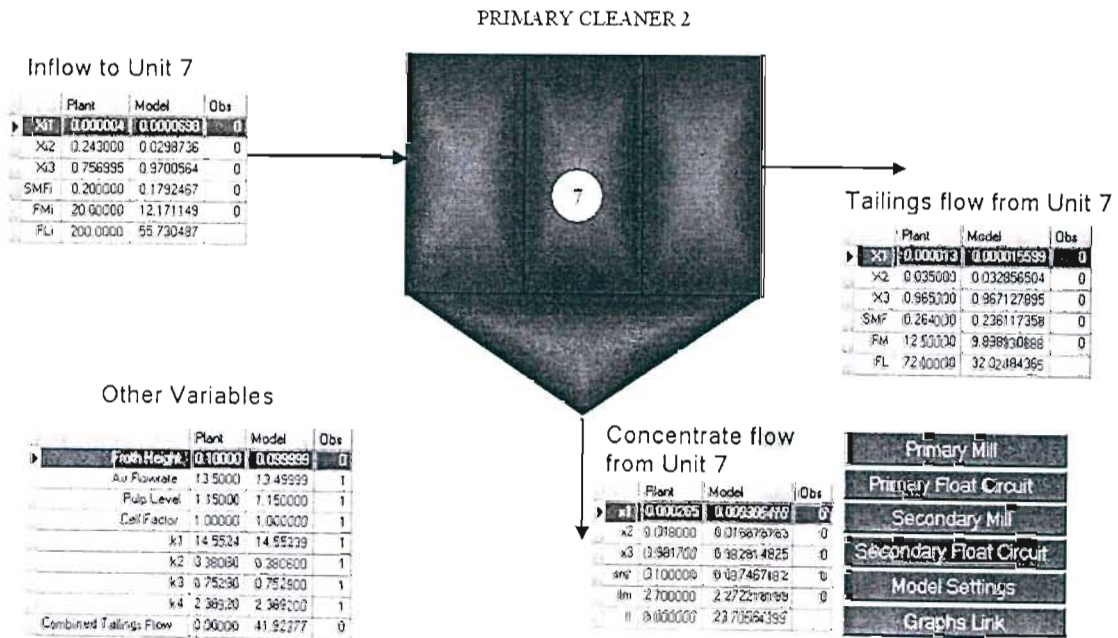


Figure I-13: Detailed GUI of Unit 7

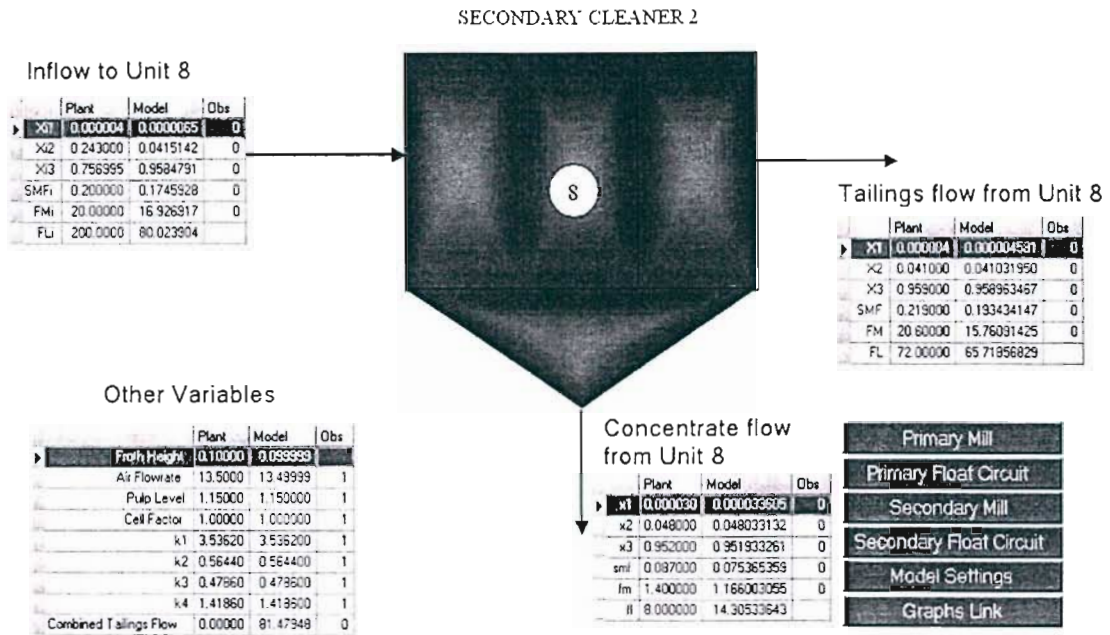


Figure I-14: Detailed GUI of Unit 8

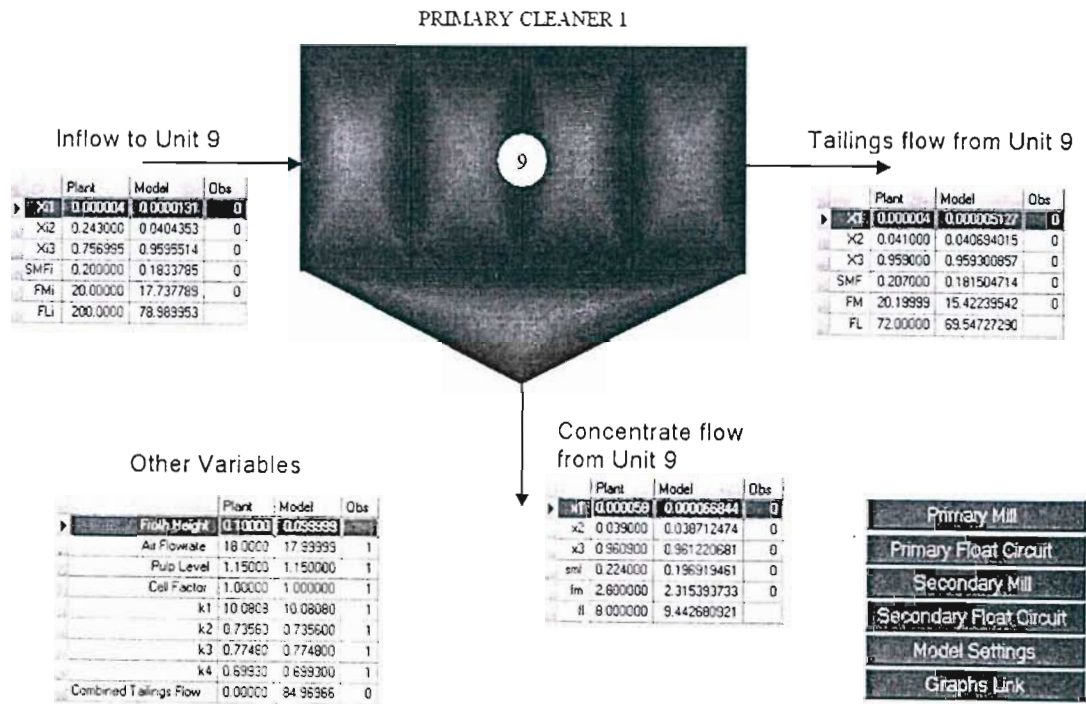


Figure I-15: Detailed GUI of Unit 9

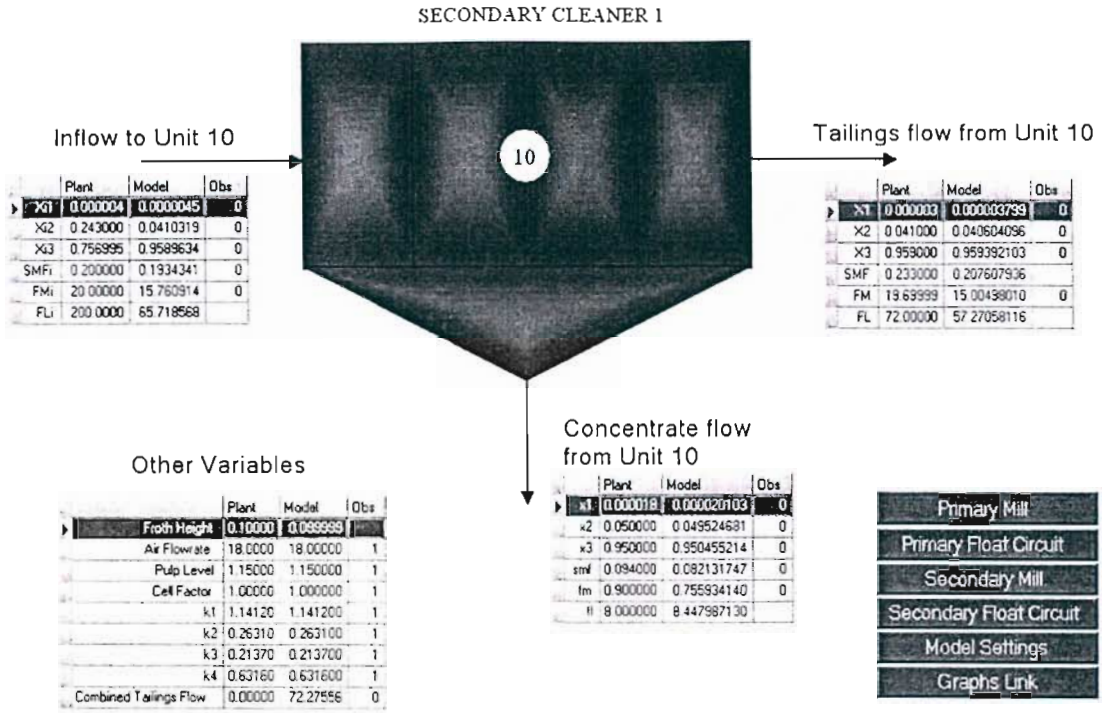


Figure I-16: Detailed GUI of Unit 10

APPENDIX J THE KALMAN FILTER

J.1 INTRODUCTION TO THE KALMAN FILTER

The Kalman filter is named after Rudolph E. Kalman, who published his famous paper describing a recursive solution to the discrete-data linear filtering problem (Kalman, 1960). The Kalman filter is, essentially, a set of mathematical equations that implement a predictor-corrector type estimator that is optimal in the sense that it minimises the estimated error covariance when some presumed conditions are met.

If a dynamic process is considered, described by an n -th order difference equation of the form

$$y_{i+1} = a_{0,i}y_i + \dots + a_{n-1,i}y_{i-n+1} + u_i, i \geq 0 \quad \text{Eq J-1}$$

where

u_i Zero-mean white random noise

and initial values (y_0, y_1, \dots, y_{n+1}) are zero-mean random variables with a known $n \times n$ covariance matrix. Using some basic conditions, this difference equation can then be rewritten as:

$$\bar{x}_{i+1} \equiv \begin{bmatrix} y_{i+1} \\ y_i \\ y_{i-1} \\ \vdots \\ y_{i-n+2} \end{bmatrix} = \underbrace{\begin{bmatrix} a_{0,i} & a_{1,i} & \dots & a_{n-2,i} & a_{n-1,i} \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}}_A \underbrace{\begin{bmatrix} y_i \\ y_{i-1} \\ y_{i-2} \\ \vdots \\ y_{i-n+1} \end{bmatrix}}_{\bar{x}_i} + \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_G u_i$$

Eq J-2(Welch and
Bishop, 2001)

which leads to the state-space model

$$\bar{x}_{i+1} = A\bar{x}_i + Gu_i \quad \text{Eq J-3}$$

$$\mathbf{y}_i = [1 \quad 0 \quad \dots \quad 0] \bar{\mathbf{x}}_i \quad \text{Eq J-4}$$

or in the more general form

$$\bar{\mathbf{x}}_{i+1} = \mathbf{A} \bar{\mathbf{x}}_i + \mathbf{G} \mathbf{u}_i \quad \text{Eq J-5}$$

$$\bar{\mathbf{y}}_i = \mathbf{H}_i \bar{\mathbf{x}}_i \quad \text{Eq J-6}$$

Equation J-3 represents the way in which a new state, $\bar{\mathbf{x}}_{i+1}$, is modelled as a linear combination of both the previous state, $\bar{\mathbf{x}}_i$, and some process noise \mathbf{u}_i . Equation J-4 describes the way in which the process measurements or observations, \mathbf{y}_i , are derived from the internal state, $\bar{\mathbf{x}}_i$. These two equations are sometimes referred to as the process model and the measurement model respectively, and serve as the basis for most linear estimation methods. The basic problem of determining or estimating the internal states of a linear system, given access only to the systems outputs, is known as the observer design problem. Typically, the approaches to this basic problem are based on the state space models described above (Welch and Bishop, 2001). One of these approaches is the Kalman filter, which will be described in the next section.

J.2 THE DISCRETE KALMAN FILTER

The original formulation put forward by Kalman in 1960 was the Discrete Kalman filter, where measurements occur and the state is estimated at discrete points in time. The Kalman filter addresses the general problem of trying to estimate the state, $\bar{\mathbf{x}} \in \mathfrak{R}^n$, of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$\bar{\mathbf{x}}_k = \mathbf{A} \bar{\mathbf{x}}_{k-1} + \mathbf{B} \bar{\mathbf{u}}_k + \bar{\mathbf{w}}_{k-1} \quad \text{Eq J-7}$$

with a measurement, $\bar{\mathbf{z}} \in \mathfrak{R}^m$, that is

$$\bar{\mathbf{z}}_k = \mathbf{C} \bar{\mathbf{x}}_k + \bar{\mathbf{v}}_k \quad \text{Eq J-8}$$

where:

\mathbf{A} $n \times n$ matrix relating the state from th previous time step, $k-1$, to the state at the current time step, k

- B $n \times l$ matrix relating the optional control input, $\bar{\mathbf{u}} \in \mathfrak{R}^l$, to the state $\bar{\mathbf{x}}_k$
- C $m \times n$ matrix relating the states to the measurements, $\bar{\mathbf{z}}_k$
- $\bar{\mathbf{w}}_k$ Random variable representing the process noise
- $\bar{\mathbf{v}}_k$ Random variable representing the measurement noise

In practise the process noise covariance, \mathbf{Q} , and the measurement noise covariance, \mathbf{R} , matrices might change with each time step or measurement, however here they are assumed to be constant along with, \mathbf{C} .

$\hat{\mathbf{x}}^- \in \mathfrak{R}^n$ is defined as the *a priori* state estimate at step k , given knowledge of the process priori to step k , and $\hat{\mathbf{x}} \in \mathfrak{R}^n$ to be the *a posteriori* state estimate at step k , given measurement $\bar{\mathbf{z}}_k$. The *a priori* and *a posteriori* estimate errors can then be defined as

$$\bar{\mathbf{e}}_k^- \equiv \bar{\mathbf{x}}_k - \hat{\mathbf{x}}_k^- \tag{Eq J-9}$$

$$\bar{\mathbf{e}}_k \equiv \bar{\mathbf{x}}_k - \hat{\mathbf{x}}_k \tag{Eq J-10}$$

The *a priori* estimate error covariance is then

$$\mathbf{M}_k^- = \mathbf{E}[\bar{\mathbf{e}}_k^- \bar{\mathbf{e}}_k^{-T}] \tag{Eq J-11}$$

and the *a posteriori* estimate error covariance is

$$\mathbf{M}_k = \mathbf{E}[\bar{\mathbf{e}}_k \bar{\mathbf{e}}_k^T] \tag{Eq J-12}$$

In deriving the Kalman filter equations an equation is first found that computes an *a posteriori* state estimate, $\hat{\mathbf{x}}_k$, as a linear combination of an *a priori* estimate, $\hat{\mathbf{x}}_k^-$, and a weighted difference between an actual measurement, $\bar{\mathbf{z}}_k$, and a measurement prediction, $\mathbf{C}\hat{\mathbf{x}}_k^-$, depicted below

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}(\bar{\mathbf{z}}_k - \mathbf{C}\hat{\mathbf{x}}_k^-) \tag{Eq J-13}$$

The difference $(\bar{z}_k - C\hat{x}_k^-)$ in equation J-13 is referred to as the measurement innovation or residual. The residual reflects the discrepancy between the predicted measurement, $C\hat{x}_k^-$, and the actual measurement, \bar{z}_k . A residual of zero means that the two are in complete agreement.

The $n \times m$ matrix, \mathbf{K} , in equation J-13 is chosen as the gain or blending factor which minimises the *a posteriori* error covariance, equation J-12. With some manipulations of the above equations, the form of \mathbf{K} can be found that minimises equation J-12 and is given by:

$$\mathbf{K}_k = \mathbf{M}_k^- \mathbf{C}^T (\mathbf{C} \mathbf{M}_k^- \mathbf{C}^T + \mathbf{R})^{-1} \quad \text{Eq J-14}$$

Looking at equation J-14 it can be seen that as the measurement error covariance, \mathbf{R} , approaches zero, the gain, \mathbf{K} , weights the residual more heavily:

$$\lim_{\mathbf{R}_k \rightarrow 0} \mathbf{K}_k = \mathbf{C}^{-1} \quad \text{Eq J-15}$$

It is also noted that as the *a priori* estimate error covariance, \mathbf{M}_k^- , approaches zero, the gain, \mathbf{K} , weights the residual less heavily:

$$\lim_{\mathbf{M}_k^- \rightarrow 0} \mathbf{K}_k = 0 \quad \text{Eq J-16}$$

A more simplistic way of thinking about the weighting, \mathbf{K} , is that as the measurement error covariance matrix, \mathbf{R} , approaches zero, the actual measurement, \bar{z}_k , is trusted more and more, while the predicted measurement, $C\hat{x}_k^-$, is trusted less and less. On the other hand as the *a priori* estimate error covariance, \mathbf{M}_k^- , approaches zero the actual measurement, \bar{z}_k , is trusted less and less while the predicted measurement, $C\hat{x}_k^-$, is trusted more and more.

The Kalman filter equations fall into two groups, time update equations and measurement update equations. Time update equations are responsible for projecting forward the current state and error covariance estimates to obtain the *a priori* estimates for the next step. The measurement update equations are responsible for the feedback, or incorporating the new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate. The time update equations can be thought of as predictor equations, whilst the measurement equations can be thought of as corrector equations. The specific equations for the time and measurement updates are presented below.

Discrete Kalman filter time update equations:

$$\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\bar{\mathbf{u}}_{k-1} \quad \text{Eq J-17}$$

$$\mathbf{M}_k^- = \mathbf{A}\mathbf{M}_{k-1}\mathbf{A}^T + \mathbf{Q} \quad \text{Eq J-18}$$

Discrete Kalman filter measurement update equations:

$$\mathbf{K}_k = \mathbf{M}_k^- \mathbf{C}^T (\mathbf{C}\mathbf{M}_k^- \mathbf{C}^T + \mathbf{R})^{-1} \quad \text{Eq J-19}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\bar{\mathbf{z}}_k - \mathbf{C}\hat{\mathbf{x}}_k^-) \quad \text{Eq J-20}$$

$$\mathbf{M}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{C}) \mathbf{M}_k^- \quad \text{Eq J-21}$$

After each time and measurement update pair, the process is repeated with the previous *a posteriori* estimates used to project or predict the new *a priori* estimates. In this way the discrete Kalman filter recursively conditions the current estimate on all of the past measurements (Welch and Bishop, 2001).

J.3 THE EXTENDED KALMAN FILTER

The Discrete Kalman filter addresses the general problem of trying to estimate the state, $\bar{\mathbf{x}} \in \mathfrak{R}^n$, of a discrete time controlled process that is governed by a linear stochastic difference equation. A Kalman filter that linearizes about the current mean and covariance is referred to as an extended Kalman filter (EKF). The estimation is linearized around the current estimate using the partial derivatives of the process and measurement functions even with non-linear relationships. In order to do this, some modifications to the discrete Kalman filter are required. If it is assumed that the process has a state vector, $\bar{\mathbf{x}} \in \mathfrak{R}^n$, as before, but that the process is governed by the non-linear stochastic difference equation

$$\bar{\mathbf{x}}_k = \mathbf{f}(\bar{\mathbf{x}}_{k-1}, \bar{\mathbf{u}}_k, \bar{\mathbf{w}}_{k-1}) \quad \text{Eq J-22}$$

with measurement $\bar{\mathbf{z}} \in \mathfrak{R}^m$

$$\bar{\mathbf{z}}_k = \mathbf{h}(\bar{\mathbf{x}}_k, \bar{\mathbf{v}}_k) \quad \text{Eq J-23}$$

where:

$\bar{\mathbf{w}}_k$ Random variable representing the process noise

$\bar{\mathbf{v}}_k$ Random variable representing the measurement noise

In this case the non-linear function, \mathbf{f} , in the difference equation (equation J-22) relates the state at the previous time step, $k-1$, to the state at the current time step, k . It includes as parameters any driving function, $\bar{\mathbf{u}}_k$, and the zero-mean process noise, $\bar{\mathbf{w}}_k$. The non-linear function, \mathbf{h} , in the measurement equation (equation J-23) relates the state, $\bar{\mathbf{x}}_k$, to the measurement, $\bar{\mathbf{z}}_k$.

In practise the actual noise, $\bar{\mathbf{w}}_k$, and, $\bar{\mathbf{v}}_k$, is not known at each time step, however the state and measurement vector can be approximated without them, as

$$\tilde{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{u}}_k, \mathbf{0}) \quad \text{Eq J-24}$$

$$\tilde{\mathbf{z}}_k = \mathbf{h}(\tilde{\mathbf{x}}_k, \mathbf{0}) \quad \text{Eq J-25}$$

where:

$\hat{\mathbf{x}}_k$ Some *a posteriori* estimate of the state (from the previous time step k)

To estimate a process with non-linear difference and measurement relationships, we write new governing equations that linearize an estimate about equation J-24 and equation J-25,

$$\bar{\mathbf{x}}_k \approx \tilde{\mathbf{x}}_k + \mathbf{A}(\bar{\mathbf{x}}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \mathbf{W}\bar{\mathbf{w}}_{k-1} \quad \text{Eq J-26}$$

$$\bar{\mathbf{z}}_k \approx \tilde{\mathbf{z}}_k + \mathbf{H}(\bar{\mathbf{x}}_k - \tilde{\mathbf{x}}_k) + \mathbf{V}\bar{\mathbf{v}}_k \quad \text{Eq J-27}$$

where:

- $\bar{\mathbf{x}}_k$ and $\bar{\mathbf{z}}_k$ Actual state and measurement vectors
- $\tilde{\mathbf{x}}_k$ and $\tilde{\mathbf{z}}_k$ Approximate state and measurement vectors from equations J-24 and J-25
- $\hat{\mathbf{x}}_k$ An *a posteriori* estimate of the state at step k
- $\bar{\mathbf{w}}_k$ and $\bar{\mathbf{v}}_k$ Process and measurement noise as in equations J-7 and J-8.
- A Jacobian matrix of partial derivatives of \mathbf{f} with respect to $\bar{\mathbf{x}}$, that is

$$\mathbf{A}_{k[i,j]} = \frac{\partial \mathbf{f}_{[i]}}{\partial \mathbf{x}_{[j]}}(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{u}}_k, \mathbf{0}) \quad \text{Eq J-28}$$

- W Jacobian matrix of partial derivatives of \mathbf{f} with respect to $\bar{\mathbf{w}}$,

$$\mathbf{W}_{k[i,j]} = \frac{\partial \mathbf{f}_{[i]}}{\partial \mathbf{w}_{[j]}}(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{u}}_k, \mathbf{0}) \quad \text{Eq J-29}$$

- H Jacobian matrix of partial derivatives of \mathbf{h} with respect to $\bar{\mathbf{x}}$,

$$\mathbf{H}_{k[i,j]} = \frac{\partial \mathbf{h}_{[i]}}{\partial \mathbf{x}_{[j]}}(\tilde{\mathbf{x}}_k, \mathbf{0}) \quad \text{Eq J-30}$$

- V Jacobian matrix of partial derivatives of \mathbf{h} with respect to $\bar{\mathbf{v}}$,

$$\mathbf{V}_{k[i,j]} = \frac{\partial \mathbf{h}_{[i]}}{\partial \mathbf{v}_{[j]}}(\tilde{\mathbf{x}}_k, \mathbf{0}) \quad \text{Eq J-31}$$

The references [i], [j], and [i,j] have been included to clarify the construction of the Jacobian matrices.

Redefining the notation for the prediction error,

$$\tilde{\mathbf{e}}_{\mathbf{x}_k} \equiv \bar{\mathbf{x}}_k - \tilde{\mathbf{x}}_k \quad \text{Eq J-32}$$

and the measurement residual,

$$\tilde{\mathbf{e}}_{\mathbf{z}_k} \equiv \mathbf{z}_k - \tilde{\mathbf{z}}_k \quad \text{Eq J-33}$$

In practise, $\bar{\mathbf{x}}_k$, in equation J-32 is not available, as it is the actual state vector, however, $\bar{\mathbf{z}}_k$, is available as it is the actual measurement. Therefore the governing equations for an error process can be written as

$$\tilde{\mathbf{e}}_{x_k} \approx \mathbf{A}(\bar{\mathbf{x}}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \bar{\boldsymbol{\epsilon}}_k \quad \text{Eq J-34}$$

$$\tilde{\mathbf{e}}_{z_k} \approx \mathbf{H}\tilde{\mathbf{e}}_{x_k} + \bar{\boldsymbol{\eta}}_k \quad \text{Eq J-35}$$

where:

$\bar{\boldsymbol{\epsilon}}_k$ New independent variable with zero-mean and covariance matrix $\mathbf{W}\mathbf{Q}\mathbf{W}^T$

$\bar{\boldsymbol{\eta}}_k$ New independent variable with zero-mean and covariance matrix $\mathbf{V}\mathbf{R}\mathbf{V}^T$

By making some substitutions of the above equations and rearranging them, the following equation is obtained,

$$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k + \mathbf{K}_k(\bar{\mathbf{z}}_k - \tilde{\mathbf{z}}_k) \quad \text{Eq J-36}$$

This equation can now be used for the measurement update in the extended Kalman filter, with $\tilde{\mathbf{x}}_k$ and $\tilde{\mathbf{z}}_k$ resulting from equations J-24 and J-25 respectively. The Kalman gain, \mathbf{K}_k , is obtained from equation J-19 with an appropriate substitution for the measurement error covariance.

The complete set of EKF equations is shown below, with $\hat{\mathbf{x}}_k^-$ substituted for $\tilde{\mathbf{x}}_k$ to remain consistent with the earlier *a priori* notation,

The EKF time update equations:

$$\hat{\mathbf{x}}_k^- = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{u}}_k, 0) \quad \text{Eq J-37}$$

$$\mathbf{M}_k^- = \mathbf{A}_k \mathbf{M}_{k-1} \mathbf{A}_k^T + \mathbf{W}_k \mathbf{Q}_{k-1} \mathbf{W}_k^T \quad \text{Eq J-38}$$

The EKF measurement update equations:

$$\mathbf{K}_k = \mathbf{M}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{M}_k^- \mathbf{H}_k^T + \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^T)^{-1} \quad \text{Eq J-39}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\bar{\mathbf{z}}_k - \mathbf{h}(\hat{\mathbf{x}}_k^-, 0)) \quad \text{Eq J-40}$$

$$\mathbf{M}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{M}_k^- \quad \text{Eq J-41}$$

The above summary of the extended Kalman filter was obtained from prior work by Welch and Bishop (2001).