Operability Analysis of an Industrial Comminution Circuit

David Richard Seaman

Submitted in fulfillment of the requirements for the degree of

Master of Science in Engineering

Department of Chemical Engineering

University of Cape Town

Cape Town, South Africa.

Supervisor: A/Prof. C.L.E. Swartz

September 2000

Synopsis

Dynamic operability is a measure of how well a process performs in dynamic operation away from it's nominal operating point when subjected to temporary upsets. Since most minerals process plants are designed according to optimum performance at steady-state, there exists a need to include dynamic considerations at the design stage rather than afterwards.

Typically, the incorporation of dynamic requirements during the design of a process are carried out in an ad-hoc fashion in the post-design stage. The problem with carrying out designs in this manner is that the plant designed for the highest economic gain on a steady-state basis may not be able to maintain operation at this point in the face of disturbances and may in fact result in a lower overall economic gain than a more conservative design which includes better operational characteristics.

An operability analysis is presented in this dissertation, based on an industrial comminution circuit from where the operational objectives are obtained. The industrial comminution circuit under investigation is the primary comminution circuit of the UG-2 plant at Impala Platinum Ltd., Rustenburg, South Africa.

The analysis involves the use of a dynamic model describing an entire comminution circuit. The data and model parameters for which are taken from Rajamani and Herbst (1991a), while information gathered from Impala is used to formulate the problem in the identification of key disturbances, manipulated and control variables, as well as the operational objective. The operational objective considered is to: *"Determine a design that minimizes the variation in the cyclone overflow product size due to spillage water addition and variation in feed characteristics subject to the design and operational constraints being satisfied."*

There are two main areas of ongoing research in the operability field with a means to analyzing and designing processes; the first is based on heuristic approaches, which are essentially simple analysis tools intended for preliminary discrimination of designs; the second, forming the focus of this work, involves the use of various optimization frameworks within which the process under investigation can be designed and/or analyzed.

An operability optimization framework is developed here, based on that presented by Mohideen *et al.* (1996b) which includes a measure of the flexibility of the process over parametric uncertainty and the defined disturbance spaces. This formulation is then used to re-design the process such that feasible operation is guaranteed over all ranges of disturbances and uncertainty, while at the same time optimizing the design in terms of minimizing the product variation when subjected to various disturbances.

Five discrete designs were considered, where each design contained a different multi-loop single loop Proportional Integral (PI) control structure. Each design was optimized within the framework with respect to the following list of design parameters:

- Sump area
- Sump level set-point
- Sump level controller parameters
- Water addition rates to the circuit
- Mill loading
- Controller tuning of the secondary control loop.

Finally the optimal design was chosen according to the minimum objective function value obtained for the five discrete designs.

Acknowledgements

I would like to extend my sincere gratitude to Professor Chris Swartz, for his wholehearted involvement in supervising this project. The professionalism with which he approached the supervision of this work is greatly appreciated, and I am certainly very fortunate for the experience gained both from a research perspective and a personal one.

My thanks go out to my fiancé, Brigitte, for her understanding encouragement and support through the course of my masters. Also to her family for their support and encouragement.

To my parents, thanks for you encouragement towards my studies over the years, especially for the enthusiasm shown towards both my work and me in general. This certainly would not have been possible without you. Thanks also to my sister for the encouragement and motivation you have given me.

To the Minerals Processing Research Unit at UCT, thanks for both the financial support given to this project as well as their interest shown to this work. Special thanks to Peter Gaylard for arranging the collaboration with Impala Platinum Limited, to whom I also extend my gratitude for the plant visits and discussions regarding the application of the study. Also to Malcom Powell and the Mech-Eng Comminution group for their contributions.

Special thanks to my colleagues in the Control Group, for both their friendship and mutual motivation towards our studies. Certainly all those I was fortunate enough to share an office with over the past two years provided a strong work ethic as well as an entertaining atmosphere; Richard MacRosty, Rhoda Baker, Kevin Dunn, Ben Knights and Caleb Hattingh. Special thanks to Richard for the companionship, especially when we were burning the midnight oil, and also the continuous amusement and friendship.

Finally thanks to Warwick for manning the Linux help-desk, and to Craig for all the technical support on the server side.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Operability is defined as the ability of a process to perform satisfactorily under conditions different from the nominal design (Grossmann and Morari, 1984). Operability includes both the steady state flexibility and the recovery of the process from temporary upsets; flexibility is the ability of a process to operate at conditions different from the nominal design. The importance of flexibility is seen by the capacity of a process to adjust to severe persistent changes in operating conditions, for example, different feeds stocks or product requirements.

The dynamic operability of a process defines how well a process responds to temporary disturbances around a particular nominal condition. The key conceptual difference between flexibility and dynamic operability is that the latter refers to the maintenance of satisfactory performance despite adverse conditions while flexibility refers to the ability of handling alternate (*and generally desirable*) operating conditions.

1.1 Motivation and Goals

Process plants are typically designed for a single steady state (*or nominal*) operating point, and the corresponding dynamic operability and flexibility issues are dealt with in an ad-hoc fashion as a post-design project. Examples are the over-sizing of equipment and addition of surge capacity. However, such heuristics are rarely optimal and may in fact worsen the plant's operability. As a result many plants have been designed in such a way that they are inherently difficult to control.

The Process Systems Research Group at the University of Cape Town has been involved for several years in the development of computational strategies for assessing plant operability. The potential of these methods was demonstrated on case studies based on simplified flotation models. The next logical step in this work has been to investigate it's application to industrial systems. The motivation to carry out this study on an industrial process stems from the need to incorporate realistic operational objectives and constraints into the problem formulation. Key issues addressed in this study are the formulation of realistic operational objectives and constraints; sensitivity of the analysis to model inaccuracy (*parametric uncertainty*) and the choice of appropriate analytical techniques.

There are several methods for assessing as well as defining operability, falling largely into two categories, namely, heuristic approaches and rigorous dynamic optimization approaches. The first category finds its' use in simple process selection, where a large number of different designs and alternatives are being compared, whereas the second set involves the use of dynamic optimization with the ultimate goal of carrying out a full design of the process. In this thesis the latter of the two areas is explored with a means of formulating a tight framework for the application of operability studies to the minerals processing industry in general.

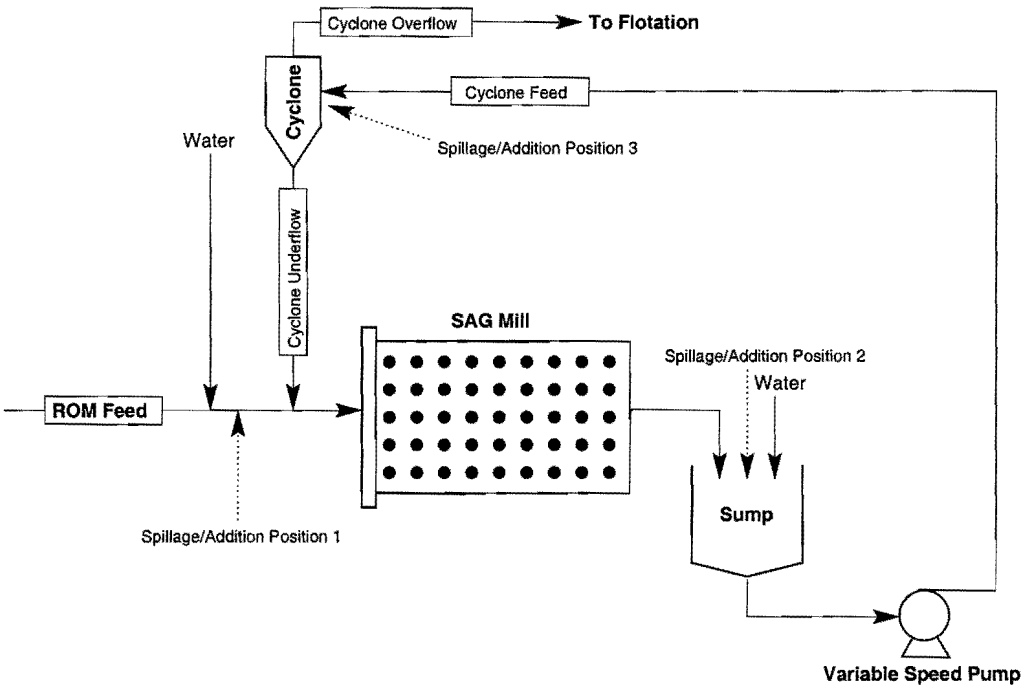

Figure 1.1: Process flow diagram of Impala primary circuit

1.2 Industrial Objectives

The study is based on the UG-2 Concentrator section at the Mineral Processing Section of Impala Platinum Limited, Rustenburg. The process under study is a primary comminution circuit consisting of two parallel Semi Autogenous (SAG) mills in closed circuit with hydrocyclones. A process flow diagram of this primary milling circuit of the UG-2 plant is shown in Figure 1.1.

The study of the Impala process is primarily to include realistic operational objectives from an industrial process into the formulation. The application of the study, however, does not go as far as modelling their process, but rather draws on their operational experience and motivations.

Any spillage from the entire UG2 plant (*flotation section as well*) is added back into the system at the primary milling stage. Indicated on the diagram are three possible alternatives for plant spillage to be returned to the circuit (*for later reference these positions have been labelled 1,2 and 3*). At present plant spillage

is being returned to position 1.

The formulation of any operability problem involves the identification of design variables, control structure, disturbances, and operational constraints. The key disturbances and design variables identified at Impala Platinum Limited are shown in Table 1.1.

Table 1.1: Key disturbances and design variables

Key Disturbances	**Design Variables**
1. Return of plant spillage	1. Mill charge
2. Variation of ROM feed	2. Nominal water addition rates
• Tonnage	3. Position of spillage addition
• Mineralogy	4. Control structures
• Size Distribution	5. Controller tuning
• Density	6. Sump volume

It is important to note that in a design rather than retro-fit case, the design variables would include all elements that are taken into account when performing a steady-state design, for example the actual size of the mill.

The key control objective is to minimize the variation in particle size of the product stream. This variation causes difficulties in operating the flotation circuit downstream in terms of the mineralogy of liberated particles as well as flow considerations around the flotation circuit. The problem has to be solved subject to operational and design constraints.

Based on information gathered at Impala Platinum Limited, a suitable operability problem is formulated:

"Determine a design that minimizes the variation in the cyclone overflow product size due to spillage water addition and variation in feed characteristics subject to the design and operational constraints being satisfied."

1.3 Thesis Overview

Chapter 2 – Operability - A Review

In this chapter, theory behind the application of operability assessment is tackled. First a short review on heuristic approaches to operability assessment is presented, followed by a more thorough review of optimization methods which were used in this application. Included in the this chapter is also a review on different methods of including flexibility into the formulation.

Chapter 3 – Comminution

Some basic theory regarding comminution circuit modelling in terms of both steady-state and dynamic operation, is presented in this chapter. This chapter ends with a full description of a dynamic circuit model selected to carry out the operability assessment, together with some modifications made to this model for incorporating the effect of ore hardness.

Chapter 4 – Optimization Formulation

A set of scenario studies is carried out on the chosen model, and the results presented. These scenarios were carried out to explore the dynamic behaviour of the process as well as to identify a hierarchy of variables. The scenarios are also used to determine which integer designs are to be included in the formulation. The chapter then proceeds to present the inclusion of the industrial problem into the mathematical formulation and is finally concluded with some modelling issues related to the chosen modelling environment, *gPROMS* (Process Systems Enterprise Ltd., 1999a,b). A novel approach to solving the problem is also presented in this chapter.

Chapter 5 – Results and Discussion

A selection of results is presented in this chapter, showing the final solutions achieved through the operability analysis. Some further scenarios are also presented in this section with reference to selecting the optimal spillage addition position.

Chapter 6 – Conclusions

Conclusions based on the work carried out and presented in previous chapters are presented here.

Appendix A – *gPROMS* code

A sample set of *gPROMS* code generated in this thesis is included here.

Chapter 2

Operability Assessment - A Review

2.1 Overview

According to Grossmann and Morari (1984), there are four main operability objectives that should be achieved by a chemical process:

1. Feasibility of steady state operation over a wide range of different feed conditions and plant parameter variations.
2. Fast and smooth changeover and recovery from process disturbances.
3. Safe and reliable operation despite equipment failures.
4. Easy start up and shut down.

Both this application and this review focus on the first two issues on this list.

There are two main areas of ongoing research in the operability arena. The first area utilizes heuristic approaches as a means to assessing plant operability and

making design decisions. The second area is concerned with the use of optimization based approaches, where the actual design of the process can be carried out within the operability framework. This study focuses on the use of optimization methods as a means to achieving an optimal design rather than making use of heuristic approaches which are approximate and thus better utilized for comparing several different designs at a very early design stage. The basic theory behind various heuristic approaches is discussed here for completion and the insight they provide.

2.2 Heuristic Approaches

This review on heuristic approaches is arranged in two sections. First, factors affecting closed loop performance are identified through the Internal Model Control (IMC) framework proposed by Morari (1983), after which the effects of these performance limiting factors are quantified.

2.2.1 Internal Model Control (IMC) Framework

Morari (1983) developed the IMC approach to assessing operability in the form of dynamic resilience. The use of the IMC framework allows the identification of factors limiting closed-loop performance (Morari, 1983). The framework of IMC is described here. Figure 2.1 shows the classical feedback structure. The IMC structure is obtained in Figure 2.2 by passing the input signal through the plant model, $\widetilde{G}$, in two places in such a way that their effects cancel overall. The IMC controller, G_c, is then defined:

$$G_c = C\left(I + \widetilde{G}C\right)^{-1} \tag{2.1}$$

The relation between the classical (C) and IMC (G_c) controllers is given below.

$$C = G_c\left(I + \widetilde{G}G_c\right)^{-1} \tag{2.2}$$

The IMC structure is used in the following analysis.

For a perfect model ($\widetilde{G} = G$), the relationship between the regulated output, y, and external inputs, y_s and d can be derived from Figure 2.2 as:

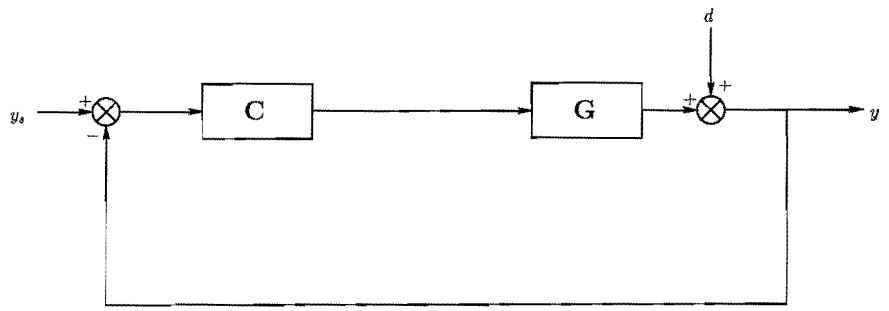

Figure 2.1: Classic Control Structure

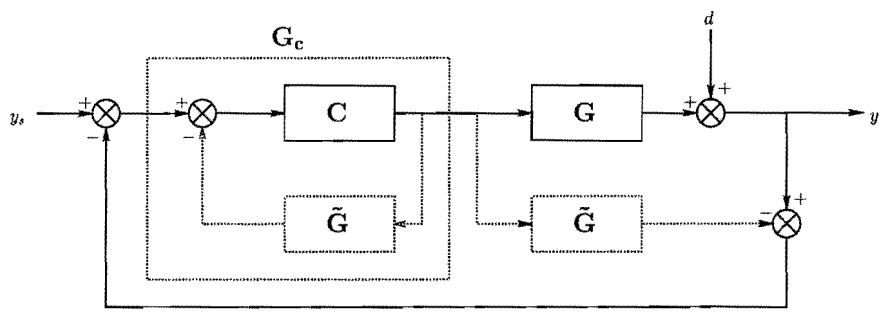

Figure 2.2: Internal Model Control Structure

$$y = GG_c\,(y_s - d) + d \tag{2.3}$$

A key result of using the IMC approach is that for a stable plant, G, the feedback system is internally stable if and only if G_c is stable. The same is not true if the classical structure is considered, i.e. a stable controller C does not guarantee closed-loop stability.

For perfect control, $y = y_s$ at all times irrespective of any disturbance or set point change, which from equation (2.3) requires $G_c = \widetilde{G}^{-1}$. The relation between the IMC and classic controllers implies that even the simplest classical controllers operate through an approximation of the plant inverse. Perfect control is not achievable in practice; equation (2.3) associates failure to achieve perfect control with limitations to plant invertability. The limitations causing imperfect control due to the unattainable plant inverse can be quantified by the following points

(Morari, 1983):

1. Right-half-plane transmission (RHPT) zeros, which would cause unstable poles in the controller, resulting in an unstable feedback system.
2. Time delays which result in prediction terms of the plant inverse which is physically unrealizable.
3. Input constraints, since a strictly proper plant model results in an improper plant inverse, resulting in unbounded input signals.
4. Model uncertainty.

2.2.2 Open Loop Indicators

These indicators are based on a linear (*or linearized*) model of the process and have been shown to give an indication of how inherently amenable a process is to tight control (Barton *et al.*, 1991). The indicators are based on the open-loop model of the process and are thus independent of the type or structure of control to be applied to the process. The minimum and maximum singular values, right half plane transmission zeros and the plant condition number are among the open-loop indicators used in practice. Barton *et al.* (1991) show an application where open loop indicators are used to screen out flotation circuit configurations which are likely to pose operability shortcomings from a set of possible configurations with similar steady-state performance.

Open loop indicators are frequency based and hence it is not possible to include time-domain specifications, and they may only be used to investigate the effect of performance limiting factors individually. They find their place in the operability framework during the early stages of design to screen out inherently poor designs with unfavourable dynamics.

Open loop indicators are readily computed from an open loop transfer function of the process under consideration.

Approach

A typical dynamic model of a process (*or subprocess*) under investigation is of the form given in equations (2.4) and (2.5), where x is the vector of state variables, y is the vector of controlled variables, and u is the vector of manipulated variables. g represents the set of state, controlled, and manipulated variable constraints.

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x, y, u) \tag{2.4}$$

$$g(x, y, u) = 0 \tag{2.5}$$

Linearizing the above model equations transforms the model to a state-space representation shown below in equations (2.6) and (2.7).

$$\frac{dx}{dt} = A\mathrm{x}(t) + B\mathrm{u}(t) \tag{2.6}$$

$$\mathrm{y}(t) = C\mathrm{x}(t) + D\mathrm{u}(t) \tag{2.7}$$

An input-output model in the Laplace domain is thus:

$$\mathrm{y}(s) = G(s)\,\mathrm{u}(s) \tag{2.8}$$

where $G(s)$ is the transfer function matrix relating the process inputs and outputs as follows:

$$G(s) = C(Is - A)^{-1}B + D \tag{2.9}$$

Open Loop Measures

Some selected open loop measures are briefly discussed below.

Singular Values The singular values of G can be used as a measure of the likelihood of input saturation to occur, as they relate the effect of manipulated inputs on regulated outputs.

Condition Number The condition numbers give an indication of the sensitivity to robust stability of a process to any uncertainty in the transfer function matrix. A large condition number could result in a process showing poor robust performance in the event of plant uncertainty.

Right Half Plane Transmission Zeros (RHPT zeros) The closer RHPT zeros are to the origin, the more sever their effect on the process. RHPT zeros exhibit a relationship to the stability of a process.

Relative Gain Array (RGA) The RGA of a process has been widely used to assist with the choice of multi-loop controller loop pairing. It is also a good indication of the steady-state process interaction. The RGA is also a useful pointer to the robust stability of a process. Large RGA elements of a process are a strong indication of control difficulties with a particular process.

2.3 Flexibility

As defined previously, flexibility reflects the ability of a process to sustain feasible operation in the light of severe sustained changes in operating conditions, that could also for example, be extended to unit failure within a process. Ensuring that a process is flexible guarantees feasible operation over a defined range of variation in operating conditions.

2.3.1 Steady State Flexibility

Halemane and Grossmann (1983) present a framework for the inclusion of flexibility in the design of processes under uncertainty. A constraint is formed over all uncertainty requiring that the process be feasible with respect to the design variables (*both integer and continuous variables*).

In order for a process to maintain feasible operation, the set of inequality constraints associated with the process model must be satisfied. The set of such constraints under steady-state conditions are shown in equation (2.10). g_k represents the the indexed vector of inequality constraints, $k \in K$, and x the vector of state variables. The vector d comprises the set of design decision variables which can be manipulated to obtain an optimal design. d comprises all factors of design decisions such as equipment sizing, controller tunings, circuit configurations and nominal operating points and includes both binary/integer and continuous time-invariant parameters. For simplicity, d is said to lie on the space $D^{\dagger}$, which is the set of all binary/integer variable combinations as well as the space within which the continuous variables may lie. $u^{\ddagger}$ represents the set of time-invariant control variables, which in the steady-state case are typically additional degrees of freedom which do not form the set of design variables, but are available for manipulation to ensure feasible operation.

[†]for clarity, it can be assumed that D is a set of both binary/integer variables and continuous variables unless otherwise stated.

[‡]u is used in this text, where Halemane and Grossmann (1983) use the variable z, to avoid confusion between formulations; also $f \rightarrow g'$.

$$g_k(d, x, \theta, u) \leq 0 \tag{2.10}$$

The uncertain parameters, θ, in the model include any infrequent changes to the process as well as possible and unavoidable plant/model mismatch. θ has upper and lower bounds θ^u and θ^l respectively, and Γ defines the region containing all realizations of θ as shown in equation (2.11)[§].

$$\Gamma = \left\{\theta | \theta^l \leq \theta \leq \theta^u\right\} \qquad \theta \in \Gamma \tag{2.11}$$

The feasibility constraint shown in equation (2.10) is then formed independent of the state variables by expressing the state variables as an implicit function of u, the control variables. h is the vector of algebraic equations describing the process under steady state conditions.

$$h\left(d, u, x, \theta\right) = 0 \Rightarrow x = x\left(d, u, \theta\right) \tag{2.12}$$

For a design d, the control variable u must satisfy the vector of inequality constraints, g_k, resulting in a new set of inequality constraints, g_k', as a function of the binary and continuous design variables, d, the control variables, u, and the uncertainty represented by θ as shown below:

$$g_k\left(d, u, x, \theta\right) = g_k\left(d, u, x\left(d, u, \theta\right), \theta\right) = g_k'\left(d, u, \theta\right) \leq 0 \tag{2.13}$$

The flexibility constraint is then formed, where for all realizations of θ on the uncertainty region Γ, there must exist a vector of design variables, d and a control

[§] $T \rightarrow \Gamma$ from Halemane and Grossmann (1983)

variable u, such that for all $k \in K$, the function g'_k, is less than or equal to zero. This is shown in equation (2.14):

$$\forall\theta \in \Gamma \left\{\exists u, d \left(\forall k \in K \left[g'_k \left(d, u, \theta\right) \leq 0\right]\right)\right\} \tag{2.14}$$

This constraint is included in an optimization problem where an optimal design in terms of both the design variables, d, and control variables, u, is sought. This formulation is discussed in more detail in the next section.

Equation (2.14) results in a mixed integer nonlinear semi-infinite programming problem, **MINLSIP**, due to the infinite number of possible realizations of $\theta \in \Gamma$. Halemane and Grossmann (1983) show a direct proof that equation (2.14) is equivalent to (2.15) which circumvents the **MINLSIP**.

$$\max_{\theta\in\Gamma} \min_{\substack{d\in D \\ u\in U}} \max_{k\in K} g'_k \left(d, u, \theta\right) \leq 0 \tag{2.15}$$

The simplified proof goes as follows (Halemane and Grossmann, 1983) (*for a more complete proof see Halemane and Grossmann (1981); Polak and Sangiovanni-Vincentelli (1979)*)

$$\forall\theta \in \Gamma \quad \left\{\exists d, u \left(\forall k \in K \left[g'_k \left(d, u, \theta\right) \leq 0\right]\right)\right\} \tag{2.16}$$

$$\Longrightarrow \quad \forall\theta \in \Gamma \quad \left\{\exists d, u \left(\max_{k\in K} g'_k \left(d, u, \theta\right) \leq 0\right)\right\} \tag{2.17}$$

$$\Longleftrightarrow \quad \forall\theta \in \Gamma \quad \left\{\min_{\substack{d\in D \\ u\in U}} \max_{k\in K} g'_k \left(d, u, \theta\right) \leq 0\right\} \tag{2.18}$$

$$\Longleftrightarrow \quad \max_{\theta\in\Gamma} \min_{\substack{d\in D \\ u\in U}} \max_{k\in K} g'_k \left(d, u, \theta\right) \leq 0 \tag{2.19}$$

Practical Implementation

The max-min-max constraint guaranteeing flexibility shown in equation (2.19) cannot be included directly in a nonlinear programming (**NLP**) or a mixed integer nonlinear programming (**MINLP**) problem. Certain assumptions need to be made in order to solve such problems.

The max-min-max constraint set depicted in equation (2.19) for the purposes of guaranteeing flexibility over parameter uncertainty, can be simplified if one considers a convex model in the uncertainty space. If convexity is guaranteed, the max-min-max construct can be reduced to simply enforcing the feasibility constraints at the vertices of the uncertainty space. This concept is shown graphically in Figure 2.3. It can easily be seen in this diagram that for the case where the feasibility constraints are convex, feasible operation can be guaranteed if the vertices of the uncertainty space boundary do not violate any of the constraints. This approach forms the basis of that taken by Halemane and Grossmann (1983) and Swaney and Grossmann (1985). The feasible region boundaries, shown as ψ_1, ψ_2 and ψ_3, are depicted here as functions of θ only for a specified design and control vector.

Halemane and Grossmann (1983) provide a theorem for ensuring the convexity of parameter uncertainty space, within which a feasible region lies which can be described by the vertices of a polyhedron. A drawback to this method is the computation involved in locating the critical vertices. For p dimensional uncertainty space, the number of vertices under consideration is 2^p (*if ten uncertain parameters are considered, 1024 vertices describe the polyhedron*).

Swaney and Grossmann (1985) build on the Halemane and Grossmann (1983) approach by defining an index of flexibility based on the maximum size of a hyperrectangle from a nominal origin, $(\theta_1^N;\theta_2^N)$ that can be inscribed inside the feasible region boundary, depicted in Figure 2.4 (*in the case of 2-dimensional geometry, which is readily extended to larger dimensions*). A hyperrectangle rather than a hypercube is inscribed within the feasible region, because in practice, uncertain parameters do not necessarily vary over arbitrary ranges. Hence the dimensions

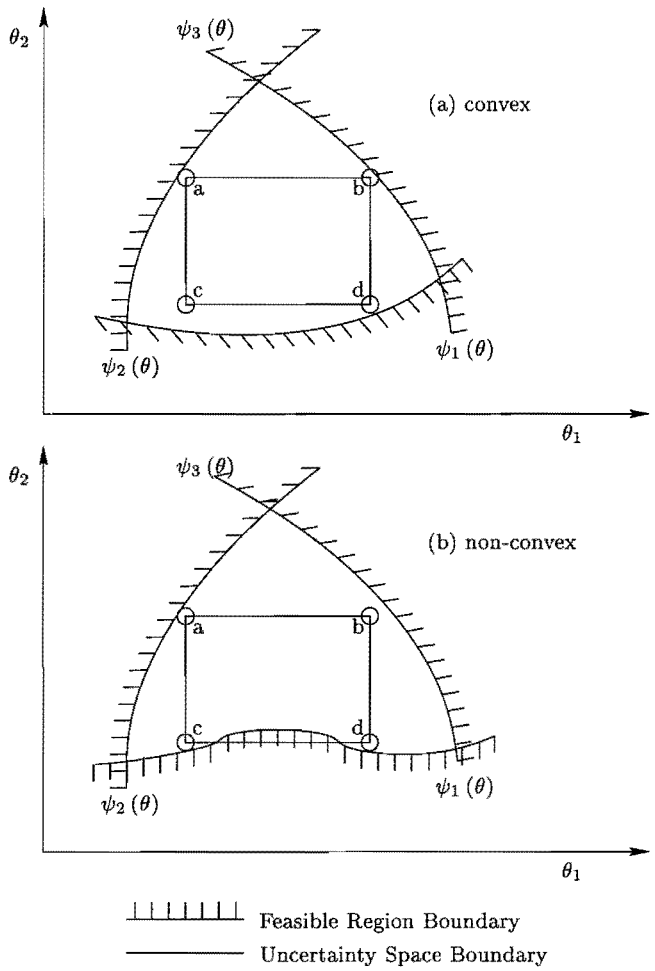

Figure 2.3: Convex vs. non-convex feasibility constraints

of such a hyperrectangle are scaled according to the relative expected deviations of the uncertain parameters, which do not have to be the same in both directions (*positive and negative*) for each parameter. The index of flexibility is defined by the dimensions of the largest hyperrectangle.

The location of the critical point limiting the size of such a hyperrectangle may lie on a vertex of a hyperrectangle for non-convex feasible region shapes as well. Swaney and Grossmann (1985) go on to show that the feasibility region only needs to be one-dimensional quasi-convex (**1-DQC**) as depicted in Figure 2.5, where the feasible region is non-convex over the entire uncertainty space, but is

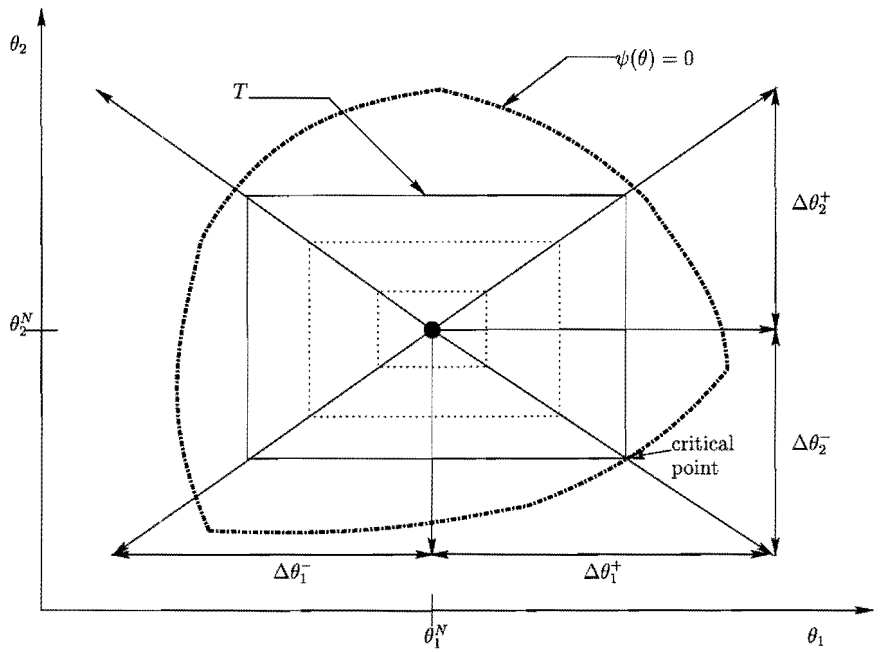

Figure 2.4: Maximum scaled hyperrectangle T inscribed within feasible region (Swaney and Grossmann, 1985)

1-DQC in the direction of the expansion of the vertices over the space (*shown by the arrows going through the expanding hyperrectangle*).

Grossmann and Floudas (1987) present a number of formulations to simplify the flexibility computations, including those presented by Halemane and Grossmann (1983). By discretizing the uncertainty space, Grossmann and Floudas (1987) present a different method to Halemane and Grossmann (1983) which requires less stringent convexity assumptions over the feasibility region. In this manner convexity need only hold between such points in the discretized space, and flexibility can be enforced over the smaller periods. Grossmann and Floudas (1987) extend this concept by identifying the critical uncertain parameters that may violate the feasibility constraints, and enforce the constraints at such points in the routine rather than arbitrarily choosing discretized points. This is implemented by solving an overall optimization problem in an iterative manner. If a design is arrived at where any constraint is violated, the critical point where the violation occurs is set as the new *active constraint*, from which the design

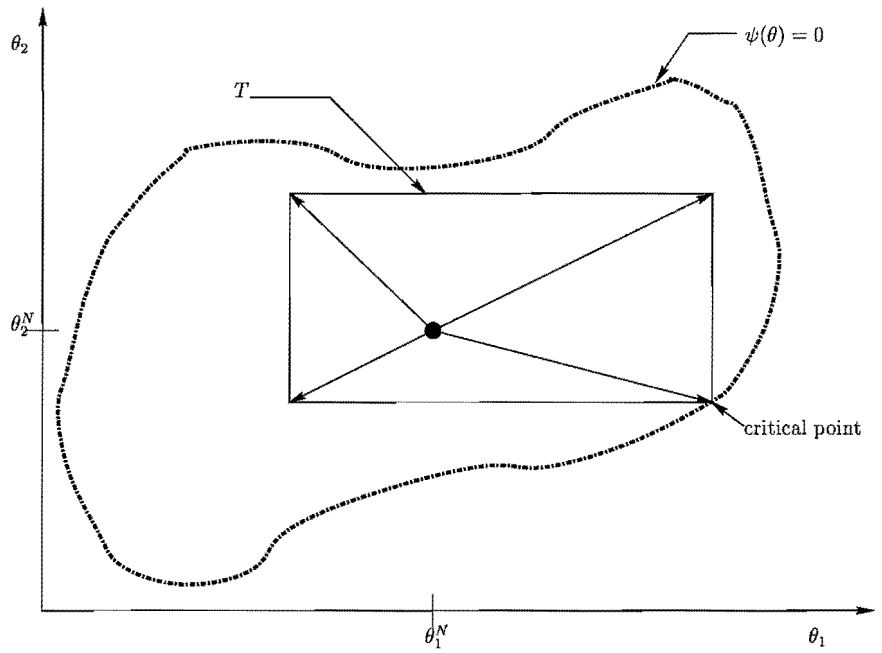

Figure 2.5: One-dimensional convex region. (Swaney and Grossmann, 1985)

is changed to ensure that this particular constraint is no longer violated. This iterative procedure is continued until no constraints are violated. This method reduces the number of constraints included in the design process. This method was also implemented by Mohideen *et al.* (1996a) in a dynamic case which is discussed further in the following section.

The key difference between these two approaches discussed here is that the Swaney and Grossmann (1985) approach requires the convexity assumptions to be satisfied, whereas the discretized approach eliminates this requirement. Both methods use a similar active constraint set, where the design parameters are backed off according to any violated feasibility constraint. The importance of the Swaney and Grossmann (1985) approach can be seen in the application of this method to a possible design methodology which synthesizes a process with a maximum degree of flexibility. The discretization approach finds its niche in applications where a defined amount of flexibility for individual uncertain parameters is considered in the design.

2.3.2 Dynamic Flexibility

The dynamic model of the process will be described as in equations (2.20) to (2.22). The differential equations are formed as a vector of time dependent states, $x(t)$, and the time dependent differential of these states is represented by $\dot{x}(t)$. h represents the set of differential and algebraic equations describing the plant and it's associated control system, as shown in equation (2.20). h is a function of all design decision variables, d, time dependent states, $x(t)$, disturbance variables, $v(t)$, uncertain parameters, θ and time, t on a finite horizon, H. Control variables consumed by controllers in order to reject disturbances are represented by the vector $u(t) \in U(t)$, where $U(t)$ is bounded by $u^u(t)$ above and $u^l(t)$ below. The representation of the model shown here represents the open-loop model of a process. For the closed-loop case, the control variables, $u(t)$, will be consumed by controller equations, leaving the tuning parameters of these controllers available for manipulation. It is also assumed that at time $t = 0$ the system is at rest (*steady-state*), with initial states represented by the vector x_0, which are themselves a function of the design variables and any parametric uncertainty:

$$h(d, x(t), \dot{x}(t), u(t), v(t), \theta, t) = 0 \tag{2.20}$$

$$x(0) = x_0(d, \theta) \;;\; \dot{x}(0) = 0 \tag{2.21}$$

The inequality constraints making up the remainder of the model are represented as indexed vectors g_k with $k \in K$, as shown in equation (2.22).

$$g_k(d, x(t), u(t), v(t), \theta, t) \leq 0 \qquad k \in K \tag{2.22}$$

Equations 2.20-2.22 fully describe the process under dynamic operation.

The set of vectors $v(t) \in V(t)$ represent possible disturbance profiles to which the process is subjected. These disturbances do not necessarily have any consistent pattern with respect to time except that they lie within specified upper and lower bounds $v^u(t)$ and $v^l(t)$ respectively as shown in equation (2.23).

$$V(t) = \left\{v(t) | v^l(t) \leq v(t) \leq v^u(t)\right\} \qquad v(t) \in V(t) \qquad (2.23)$$

Realistically, the disturbance profiles are continuous and can be modeled on a frequency based function of varying form[¶].

Previously the flexibility problem had been considered as a steady-state problem, but in recent years has been extended to the dynamic case. The need for extending flexibility from a steady-state requirement to a dynamic one is illustrated by Dimitriadis and Pistikopoulus (1995), where a region of feasibility is defined as an area in θ space, where θ is the vector comprising uncertain parameters and infrequent disturbance variables. Dimitriadis and Pistikopoulus (1995) go on to show that because the feasibility constraints are in fact time-dependent, the *feasible region*, which is defined as the region within which feasible operation is guaranteed, is shown to shift over time thus rendering operation of the process infeasible. Figure 2.6 illustrates this concept for the two dimensional case, where θ is comprised of two uncertain parameters θ_1 and θ_2.

The mapping of θ_1 and θ_2 is shown at two different times, with the uncertainty space boundary superimposed on this feasible region. It is shown in Figure 2.6 that at $t = t_1$, the uncertainty space boundary lies within the feasible region boundary, and the process can be said to be feasible for all combinations of uncertainty; conversely, when $t = t_2$, the uncertainty space boundary lies outside of the feasible region boundary, illustrating that the chosen design is infeasible for part of the uncertainty region at this time. This example illustrates the concept that flexibility needs to be incorporated in the time domain formulation if one

[¶]The nomenclature presented up to this point is maintained for purposes of consistency through this text.

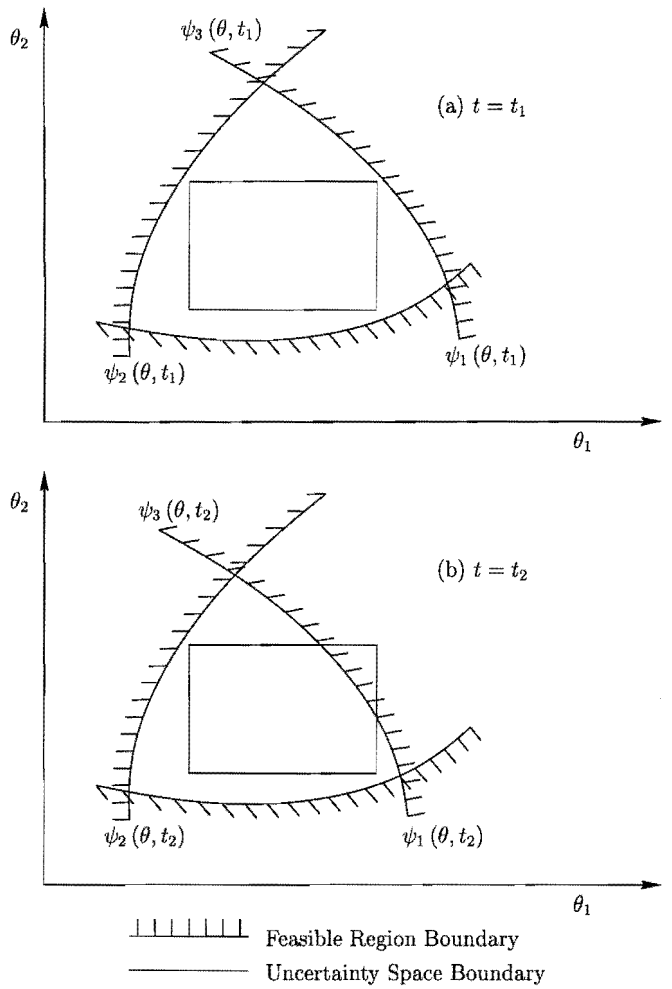

Figure 2.6: Variation of feasible region with time

wishes to choose a design guaranteeing feasibility over both time and parametric uncertainty.

The feasibility region shown in Figure 2.6 is bounded by three feasibility constraints, ψ_1, ψ_2 and ψ_3. This feasible region, R, is defined in equation (2.24) where g_k represents the set of $k \in K$ inequality (*feasibility*) constraints and d is the vector of design variables, in this case containing only continuous time-invariant variables.

$$R = \{\theta(t), d \mid \forall k \in K \; g_k(d, \theta(t)) \leq 0\} \tag{2.24}$$

From equation (2.24), the bounds ψ_1, ψ_2 and ψ_3, shown in Figure 2.6, can be defined:

$$\psi_k(d, \theta(t)) = \{d, \theta(t) \mid g_k(d, \theta(t)) = 0\} \tag{2.25}$$

Mohideen *et al.* (1996b) present the max-min-max construct of Halemane and Grossmann (1983) shown in equation (2.15) with the extension of steady-state flexibility to the dynamic case:

$$\max_{\substack{\theta \in \Gamma \\ v(t) \in V(t)}} \min_{\substack{d \in D \\ u(t) \in U(t)}} \max_{\substack{k \in K \\ t \in H}} g_k(d, x(t), u(t), \theta, v(t), t) \leq 0 \tag{2.26}$$

In order for the max-min-max construct to be used to test whether or not a particular design is feasible over parametric uncertainty and a defined disturbance space, the design and control variables are fixed at this point, and feasibility is checked as follows; in the case of dynamic optimization, this construct forms the constraints to be satisfied:

$$\max_{\substack{\theta \in \Gamma \\ v(t) \in V(t)}} \max_{\substack{k \in K \\ t \in H}} g_k(d, x(t), u(t), \theta, v(t), t) \leq 0 \tag{2.27}$$

Bahri *et al.* (1997) present a similar method for ensuring feasible operation for a given set of independent variables (*decision variables*), d. In order for feasible operation to be guaranteed, the equality and inequality constraints of the

process model must be satisfied for all defined variations of disturbances and parameter uncertainty arising from plant/model mismatch. Bahri *et al.* (1997) include parametric uncertainty in their formulation as a single step-up or down disturbance, and investigate the long term effects of this change.

Bahri *et al.* (1997) also go on to conduct an integrated flexibility and controllability analysis.

Practical Implementation

The inclusion of dynamic flexibility in an operability problem formulation is very similar to that of steady-state flexibility. The feasibility constraints now include both short-term time dependent disturbances as well as time. Thus all the methods presented in the previous section on the implementation of steady-state flexibility are readily extended to the dynamic case as well.

Mohideen *et al.* (1996a) use the active set strategy over discretized uncertainty and disturbance space, presented in the previous section from Grossmann and Floudas (1987). The key difference is that Mohideen *et al.* (1996a) extends the uncertainty to include the disturbance variables with variation over time. In addition they use orthogonal collocation on finite elements to convert the set of differential equations to algebraic residual equations, and enforce external stability by way of interior point constraints over a finite time-horizon. A measure of the Jacobian matrix is utilized as a general estimate of stability.

2.4 Optimization Based Approaches

Optimization based approaches to operability typically involve the search for an optimal design of a process with a measure of feasibility and/or stability included in the formulation. There are two main arenas where optimization based approaches are exploited. In the first, an optimization problem is formulated based on an existing design of a process where optimal operating points, control structures and tuning are being sought. In the second, the actual design of all aspects of a process are included within the formulation.

The inclusion of feasibility in an overall design problem is generally seen in one of two forms. Firstly, as part of the objective function, where one might be searching for a design with optimal (*or maximum*) flexibility; and secondly, where feasibility is included as a constraint in the optimization, requiring that any design satisfies the requirements of feasible operation in light of disturbances and parametric uncertainty.

In most cases the feasibility constraints discussed in the previous section are included in optimization problems, where an objective function is to be optimized over a range of decision variables (*typically formed of time dependent and/or time independent design variables*).

How the cost function is made up is of little importance to the mathematical formulation itself but is the major design goal, and is thus very important in the design stage in terms of knowing exactly what the design is trying to achieve. Φ is used to represent a measure of the objective under investigation, where it is desired to minimize the value of Φ. Equation (2.28), slightly modified in terms of nomenclature from the approach taken by Mohideen *et al.* (1996a) shows how the expected cost function would be minimized over the range of design variables. As shown in equation (2.28), the expected value of the objective function, E, is minimized over both integer and continuous design variables. The expected function takes into account the value of a measure of the objective function over the range of uncertain parameters, disturbances and time, where time is taken to be finite over a horizon, H. This expected function is used to find the optimum

across all operating conditions so as not to optimize the process at a particular (*and arbitrary*) realization on the uncertainty and disturbance space.

$$\min_{d\in D} \underset{\substack{\theta\in\Gamma \\ v(t)\in V(t) \\ t\in H}}{\mathrm{E}} \{\Phi(d, x(t), v(t), \theta, t)\} \tag{2.28}$$

The expected function represents an average expected cost estimate which can be determined through a suitable integration scheme.

Another method of ensuring the design is not driven by some arbitrary realization of uncertainty and disturbance variables is to consider the maximum value of the objective function over these uncertainty and disturbance spaces, resulting in the min-max construct below.

$$\min_{d\in D} \max_{\substack{\theta\in\Gamma \\ v(t)\in V(t) \\ t\in H}} \{\Phi(d, x(t), v(t), \theta, t)\} \tag{2.29}$$

Equation (2.29) shows how the primary optimization routine in the formulation is to minimize the objective function over the design variables, where the objective function value is taken as the maximum value over the uncertainty and disturbance space (*i.e. the highest value of the objective function for all realizations of* $\theta \in \Gamma$, $v(t) \in V(t)$ *and* $t \in H$).

Mohideen *et al.* (1996a) show an objective function comprising of the expected value of a cost function over the uncertain parameter set, the disturbance range and all time on a finite horizon. They also include in the optimization decision space all possible PI controller structures together with their individual tuning parameters. The optimization objective follows (Mohideen *et al.*, 1996a):

$$\min_{\substack{d,X \\ u_o,KC,\tau}} \mathop{\mathrm{E}}_{\substack{\theta\in\Gamma \\ v(t)\in V(t) \\ t\in H}} \left\{ \min_{\substack{d\in D \\ y_{set}}} P\left[d, x(t), u(t), \theta, v(t), t\right] + C^T X \right\} \tag{2.30}$$

The objective function can take many forms; in the case study by Mohideen *et al.* (1996a), the objective is to minimize an overall function comprising a measure of economics. This objective is shown in equation (2.30), where the function P relates to operating costs, while C^T relates to the cost of additional controller loops, X is a binary vector denoting whether (*or not*) a particular control loop is included, and u_o, KC, τ denote the controller tuning parameters, all of which are included in the search space. Mohideen *et al.* (1996b) approximate the expected value function shown in equation (2.30) with a weighted summation over a set of specified periods/critical parameters.

Mohideen *et al.* (1996a) apply this formulation together with the flexibility constraints discussed in the previous section to both a stirred tank reactor requiring hold-up and temperature control and a distillation column demonstrating the ability of their approach to attain solutions with significant cost savings. In this example, the objective function is first optimized over the design variables. The resulting design is then passed to the flexibility analysis in order to locate any critical uncertain parameters (*those that cause the uncertainty space to violate the feasibility region; see the previous section*), if any are found they are fed back into the primary optimization routine as the *new* set of constraints. This iterative process is continued until a design is achieved which satisfies the flexibility requirements.

Bahri *et al.* (1997) present a similar method to that of Mohideen *et al.* (1996a), although this formulation does not consider the simultaneous design and flexibility considerations included in the Mohideen *et al.* (1996a) approach. Bahri *et al.* (1997) first consider the amount of back-off required, in terms of the objective function from the optimal design, in order to ensure feasible operation on the open-loop process at steady state. They go on to compare this amount of open-

loop back-off required to the amount of open-loop back-off required for the dynamic case and finally compare this to closed-loop back-off required when different control structures are implemented. This routine provides a measure of both *flexibility* and *controllability* (*the inherent ability of a process to be controlled*) Parametric uncertainty as such is not included in this formulation, however, time dependent disturbances in the form of step functions are included where the magnitude of this step function is allowed to vary between given bounds.

Narraway and Perkins (1994) show a similar formulation in terms of the objective function and search space investigated. In this example, the search space comprised integer variables making up a set of different PI controller loop pairings, while an economic objective was optimized. The set of PI controller loops comprised all combinations of defined manipulated to output variable pairings.

Ross *et al.* (1999) show an application of simultaneous design and control to both a double-effect distillation system and a high-purity industrial distillation column. They highlight the key difference between sequential design and control, and simultaneous design and control strategies. Ross *et al.* (1999) show that the simultaneous approach is superior in that it exploits the interaction between design and control in achieving an optimal solution.

In all cases the optimization is carried out over a finite time horizon[‖] and will always be carried out subject to a set of differential and algebraic equations constituting a process model of one form or another.

Grossmann and Floudas (1987) present a similar construct to those presented here, except they include flexibility in the objective function rather than as a constraint. In such a formulation, a resulting design will be one with a maximum degree of flexibility. Grossmann *et al.* (1983) also present two scenarios, the first method assumes a fixed degree of flexibility is required, in much the same manner as Mohideen *et al.* (1996b), while in the second scenario they take a similar approach to Grossmann and Floudas (1987) by formulating a method for determining a design with an optimal degree of flexibility.

[‖]denoted by H in this text

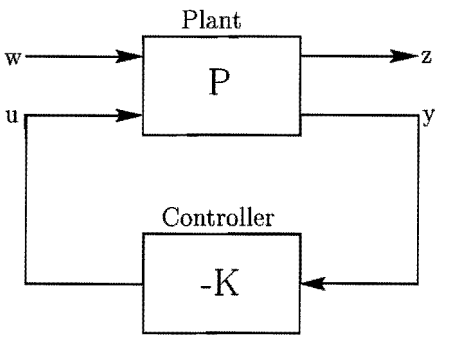

Figure 2.7: Feedback Framework

2.4.1 Q-Parametrization

The Q-parametrization approach is well developed in Ross (1997), from where a description of Q parametrization is formulated. Figure 2.7 shows the structure of a feedback system considered in Q-Parametrization. The plant inputs are partitioned into two vectors, control inputs, u, and exogenous inputs, w. The control inputs are inputs available for manipulation by the controller, while the exogenous inputs are all other inputs, typically process output set-points and disturbances. Similarly the plant outputs are divided into regulated variables, z, and sensor outputs, y. The regulated variables are those which may be required to meet certain process operation specifications, and the sensor outputs are those which can be measured directly for use by the controller.

P is a transformation matrix relating the plant outputs to the plant inputs and is thus independent of control structure. P can be expressed in terms of inputs , w and u, and outputs, z and y as in equation (2.31). Note that P is not necessarily the plant model itself but will include the plant model.

$$\begin{bmatrix} z \\ y \end{bmatrix} = P \begin{bmatrix} w \\ u \end{bmatrix} \tag{2.31}$$

If P is linear it can then be expressed as a matrix of transfer functions as in

equation (2.32).

$$P = \begin{bmatrix} P_{zw} & P_{zu} \\ P_{yw} & P_{yu} \end{bmatrix} \tag{2.32}$$

For a linear, time-invariant controller K, the closed-loop transfer function relating exogenous inputs, w, to regulated outputs, z, is as follows:

$$H_{zw} = P_{zw} - P_{zu}K(I + P_{yu}K)^{-1}P_{yw} \tag{2.33}$$

Vidyasagar (1985) used the theory of coprime factorization to show that any stabilizing H_{zw} can be representing by the parametrization shown in equation (2.34),

$$H_{zw} = T_1 + T_2QT_3 \tag{2.34}$$

where Q is any stable transfer function matrix, and is also the free parameter in the above equation. T_1, T_2 and T_3 are stable transfer matrices that may be determined from coprime factorizations of the plant model P_{yu} and a nominal stabilizing controller (Ross and Swartz, 1995). Since H_{zw} is affine in Q, closed loop specifications convex in H_{zw} are convex in Q. Many closed-loop specifications are convex in H_{zw} and hence convex in Q (Boyd *et al.*, 1990).

The above theory forms the basis for the formulation of a convex objective function reflecting a measure of closed loop performance which is then optimized over Q, subject to a set of convex constraints on the inputs, the outputs and robust stability. Because Q parametrizes all stabilizing controllers, the performance at the optimum represents the best possible performance for any linear feedback controller, while the convexity of the formulation guarantees that the optimum found is indeed the global optimum.

Using the above expressions, it is then possible to formulate a convex optimization problem to calculate the limit of achievable performance for some linear controller Q, as in equations (2.35 to 2.36).

$$\min_{Q \in \bar{\Omega}} \ \bar{\Phi}(Q) \tag{2.35}$$

$$\text{where} \quad \bar{\Omega} = \{Q \,|T_1 + T_2 Q T_3 \in \Omega, \ \ Q \text{ stable}\} \tag{2.36}$$

Here $\bar{\Phi}(Q)$ is a convex function of Q, and Ω is the set of all H_{zw} satisfying the performance constraints.

The set of all stable transfer matrices, Q is of infinite dimension and must be reduced to a finite dimension for practical reasons. A discrete time formulation is used to facilitate the handling of dead time and time constraints to the problem. The (i, j) element of matrix Q can thus be represented as in equation (2.37), where q_{ij} become the search variables.

$$Q_{ij} = \sum_{k=0}^{L} q_{ij}(k) z^{-k} \tag{2.37}$$

From the development of Q-parametrization shown here, an objective function can be formulated relating to the desired characteristics of the process. This formulation can include set point tracking, disturbance rejection as well as some form of economic objective to name a few. Ross and Swartz (1995) show the formulation of an objective function in which the sum of squared errors in the outputs of various flotation circuits was to be minimized for a step change in the feed solids rate (*a disturbance variable*).

The purpose of the approach was to reject any flotation circuits that may show undesirable controllability characteristics at the design stage. The analysis involved six three-bank flotation circuit configurations in the presence of input

constraints. A set of *non-inferior* flotation circuits was generated, where the term *non-inferior* was taken to mean that neither circuit had both better steady state recovery and better dynamic operability than the others. The flotation circuits analyzed are the same as those analyzed by Barton *et al.* (1991). Both methods of analysis rejected similar flotation circuit designs; however Barton *et al.* (1991) used open-loop indicators for their analysis.

2.4.2 Operability Problem Formulation

The formulation of an operability problem requires an understanding of the design and operational objectives to be met. The problem formulation is in itself an analysis of the process where both desirable and undesirable characteristics of the process are identified.

The design variables of the problem need to be clearly identified. They are typical of the process, for example: pump sizing, sump volumes, configuration of circuits etc.. In the case of a green fields project, the design variables could also include different processes or major equipment elements. In the case of flotation circuits, one of the key design variables available for manipulation is the configuration of the different flotation circuits (Ross and Swartz, 1995; Swartz, 1997; Seaman, 1998).

The manipulated, output and disturbance variables need to be determined. However the methodology of control is not as important in the scope of an operability analysis as the control structures are. For example, a process that lends itself to more control structures may be preferable to one with fewer, as this allows for more flexibility in the latter design of detailed controllers.

The design and operating objectives of any process must be clearly identified and formulated into an objective function with an associated set of constraints. Typically the most important objective in any process is the maximization of economic gain, be it directly or indirectly. The optimization problem will also include constraints of varying forms such as product quality and input constraints.

Modelling/Simulation

The first step in analyzing a process with respect to operability of any sort is to arrive at a reasonable and practical process model of the process. A process model is a set of mathematical equations that relate the process performance variables to the independent process variables (Smith, 1984). Depending on the particular characteristic of the process one wishes to investigate, the model may take on different forms. The first form of model is a steady-state model describing the outputs of a process given a set of inputs at steady-state. The second form of model is dynamic in the sense that it considers the variations with time of the outputs given the time dependent input variations. A mathematical model allows the process to be analyzed 'off-line' or during the design process of the plant in question.

The investigation of flexibility requires only a steady-state model, while the analysis of dynamic flexibility requires a dynamic model. The chosen model can later be used for simulation purposes to investigate and compare the results of the analysis carried out.

It is important that the relevant model allows the required degrees of freedom with respect to the design variables to be investigated (Manlapig, 1977), enabling choices and or deductions to be made relating to the particular variable. For example, if the purpose of the analysis was to decide from an operability standpoint on the configuration of a flotation circuit, the model would have to be independent of circuit configuration.

Chapter 3

Comminution

The term comminution applies to any device involved in size reduction of minerals. This covers a vast number of devices which fall predominantly into the following categories:

- Crushers
- Stirred Mills
- Tumbling Mills
- Sizing Processes

The scope of this review is such that it only deals with the last two categories, tumbling mills and sizing processes. While sizing processes do not involve the physical reduction of minerals, they are an integral part of any comminution circuit. Therefore they justifiably fall under the umbrella of comminution.

There are four distinct classes of tumbling mills used in the minerals processing industry at present, namely:

1. Rod Mills

2. Ball Mills
3. Autogenous (AG) Mills
4. Semi-Autogenous Mills (SAG Mills)

The key difference between these four tumbling mills is in the grinding media used. In the case of AG mills, the grinding medium inside the mill is the ore itself. While in the case of SAG mills, the grinding medium is made-up of feed ore as well as steel balls.

This review will further focus on SAG mills in terms of circuit design, modeling and operation.

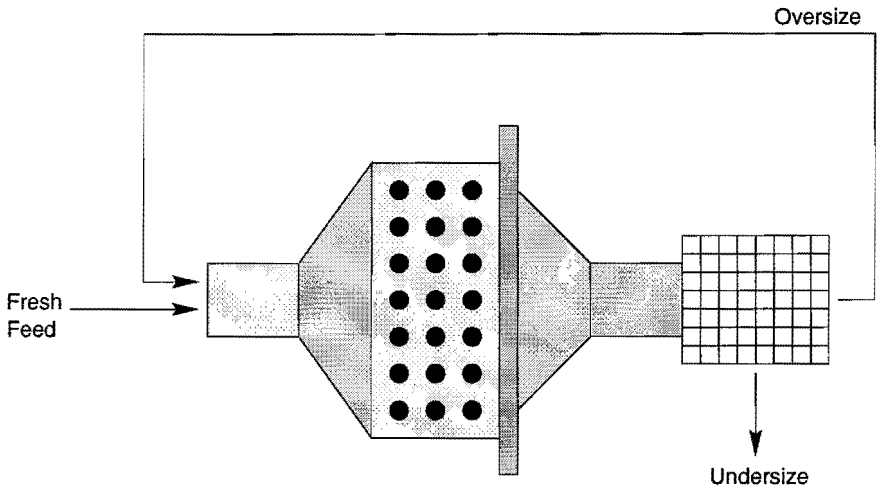

Figure 3.1: Open-Circuit Configuration - external return of oversize

3.1 Circuit Design

In most cases the feed to SAG mills is primary jaw crusher product. ROM (*Run of Mine*) feed and secondary jaw crusher product are also found as feed to SAG mills in industry, especially in South Africa. At a basic level, SAG mills are operated in so called open-circuit configuration, as shown in Figure 3.1 (Napier-Munn *et al.*, 1996). In this case a coarse classifier is used, such as a trommel attached to the mill. The oversize particles leaving the mill are either stock-piled and then added to the feed in a batch type process or continually transported to the feed using a conveyer.

A similar open circuit design is the use of internal oversize return, making use of a trommel, where the oversize material is conveyed back into the mill by a reverse spiral on the trommel or via the use of a water jet. This type of circuit is shown in Figure 3.2 (Napier-Munn *et al.*, 1996).

Figure 3.3 (Napier-Munn *et al.*, 1996) shows a closed circuit configuration where a classifier, typically a cyclone or a DSM screen, is used. This type of circuit will have a finer resulting product, but in most cases will have a lower throughput than an open circuit type configuration under the same conditions (*of design and feed characteristics*). This configuration does however allow for a much

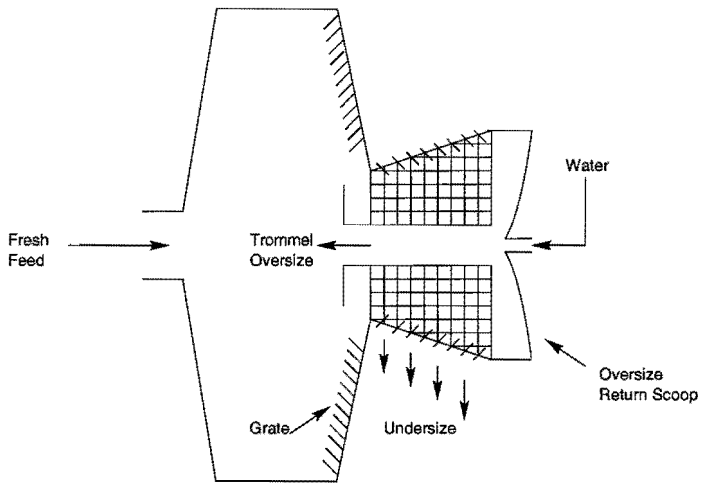

Figure 3.2: Open-Circuit Configuration - internal return of oversize

finer product and thus a trade-off can be seen between choosing on open or closed circuit configuration.

The final circuit design to be presented here is that depicted in Figure 3.4 of closed circuit type with recycle to a pebble crusher. This configuration has become increasingly popular in recent years as it usually gives significantly increased throughput than other configurations (Napier-Munn *et al.*, 1996).

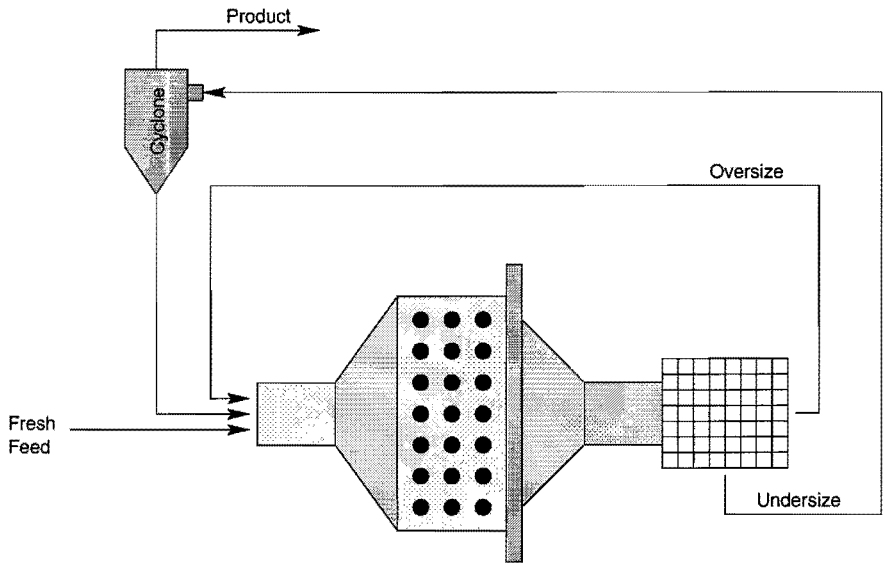

Figure 3.3: Closed-Circuit Configuration

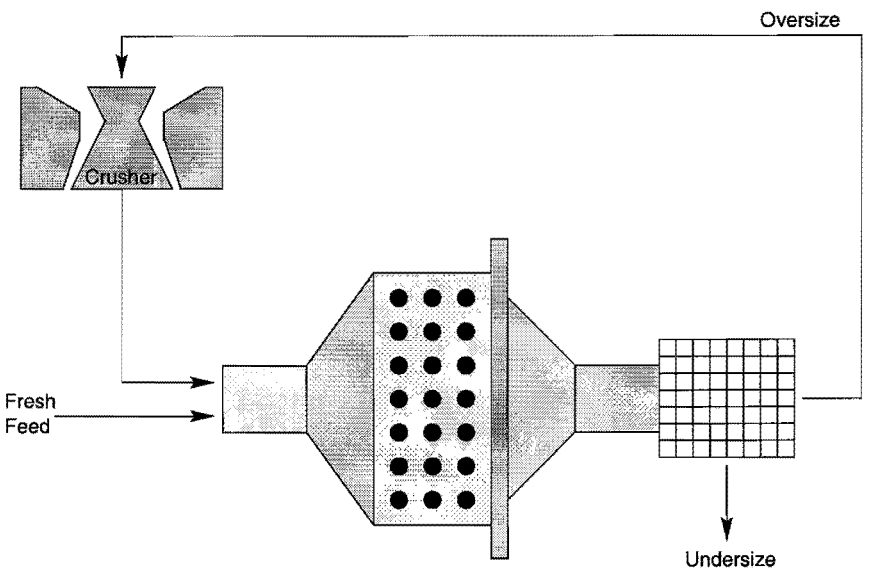

Figure 3.4: Recycle Crusher Configuration

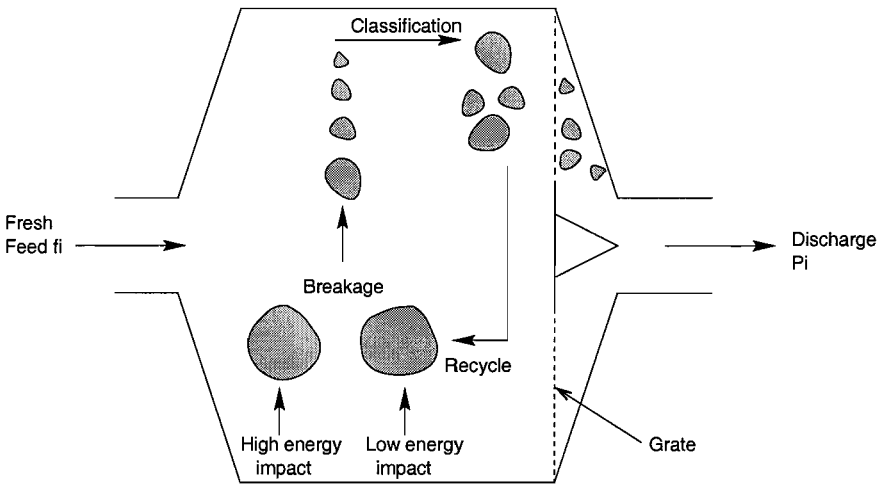

Figure 3.5: Schematic Diagram of SAG mill process mechanisms

3.2 Modeling

3.2.1 A steady state model

The development of fundamental or empirical approaches to the modeling of comminution circuits has become very well advanced in recent years. The models developed thus far are mainly static (*steady state*) models that can be modified to represent the dynamic case in a milling/comminution circuit.

The processes occurring in a SAG mill are shown schematically in Figure 3.5 (Morrell *et al.*, 1992). Feed enters the mill and is subject to breakage from collision with other particles and/or the mill shell. The product either exits via the grate or remains to undergo further collisions. The processing of particles within a SAG mill may be described through essentially three components:

1. Collision Frequency (*breakage rate*)
2. Ore size distribution after collision (*appearance distribution function*)
3. Particle transport out of the mill (*discharge rate*)

A simple way of reducing a continuous size distribution is to subdivide it into a set of n finite size classes, where the largest size class is $i = 1$ and the smallest is $i = n$. At steady state these components are combined in the perfect mixing model mass balance equations (Whiten, 1974):

$$0 = f_i - p_i + \sum_{j=1}^{i} r_j s_j a_{ij} - r_i s_i \tag{3.1}$$

and

$$p_i = d_i s_i \tag{3.2}$$

where:

f_i	feed rate of particles of size i
p_i	product rate of particles of size i
r_i	breakage rate of particles of size i
s_i	mill contents of particles of size i
d_i	discharge rate of particles of size i
a_{ij}	appearance of breakage distribution function
n	total number of size classes.

subscript:

i	size class of balance

An important aspect of this model is that the breakage rate of particles and the appearance of breakage distribution function can be split into functions of two distinct variables, namely the properties of the ore and the characteristics of the breakage machine, i.e. the SAG mill. In this way the model can be seen to be

dependent on the ore properties as well as the mill, and hence variations of either can be considered during simulation. These properties are discussed further in section 3.2.2.

The appearance function of the feed is determined using laboratory breakage and abrasion tests, while the rest of the parameters shown above can be developed according to the particular mill in question (Napier-Munn *et al.*, 1996).

The rate of discharge, d_i, of particles of size, i, is considered to be the product of transport to and classification by the grate as given in equation (3.3).

$$d_i = dc_i \tag{3.3}$$

where:

d	maximum discharge rate
c_i	classification function value

A mass transfer function is used to determine the relation between the hold up in the mill and the total volumetric discharge rate in an iterative manner:

$$L = m_1 F^{m_2} \tag{3.4}$$

where:

m_1, m_2	constants
L	Fraction of mill occupied by material less than grate aperture
F	Volumetric discharge rate in mill fills per minute

The material transport out of the mill is discussed further in section 3.2.3.

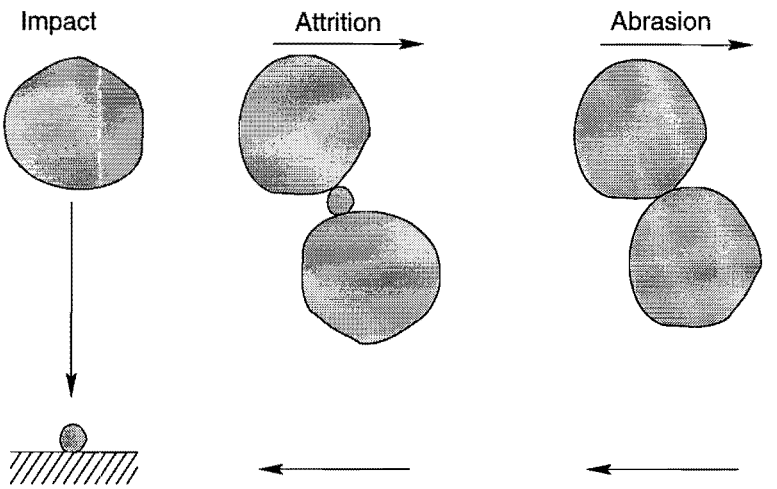

Figure 3.6: Breakage Mechanisms

The energy level within the mill (*or total energy available for breakage in the mill*), E_i, is assumed to be related to the specific energy for size class i by equation (3.5), where x_i is the mean particle size in size class i.

$$E_{cs_i} \propto E_i x_i^{1.5} \tag{3.5}$$

3.2.2 Breakage

Three mechanisms are responsible for breakage of ore within a SAG mill, namely:

1. Abrasion
2. Attrition
3. Impact

Figure 3.6 (Napier-Munn *et al.*, 1996), describes these mechanisms graphically.

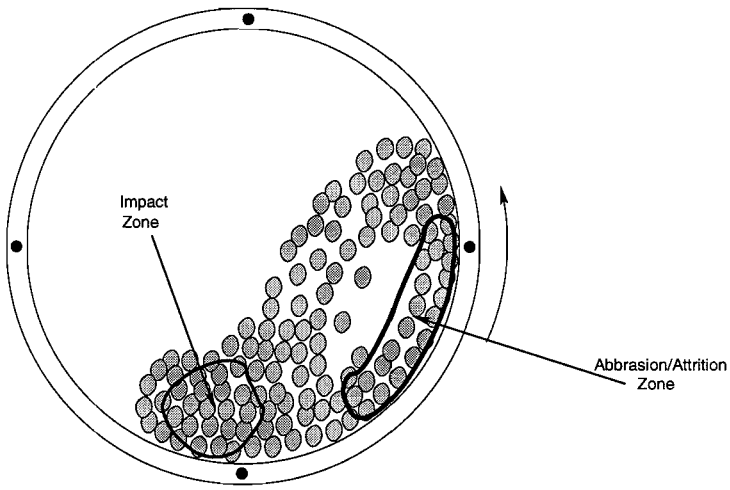

Figure 3.7: Breakage Zones

The mechanism of abrasion is largely a surface phenomenon resulting when two particles move parallel to one another (*along their point of contact*). The parent particles are left mostly intact as only small particles are torn from their surfaces.

Attrition results in a similar manner to abrasion, except that a small particle is trapped between the two large particles. The smaller particle is broken down at a far higher rate than the larger ones, again leaving the larger particles relatively unchanged in size.

Impact breakage results from the impacting particle moving perpendicular to the plane of contact with either the mill shell or another particle/ball. The specific energy, or energy per unit mass which the target particle is subjected is directly related to the amount of resulting breakage of the particle. If the specific energy of an impacted particle is low, the size distribution of the resulting breakage is similar to that of an abrasion type of breakage.

There are essentially two distinct regions within a mill where the breakage mechanisms occur. These are shown in Figure 3.7 (Napier-Munn *et al.*, 1996). Impact breakage occurs in the toe region of the mill, while abrasion/attrition occurs in the body of the charge as it is lifted by the rotation of the mill. The amount of breakage that occurs in the former area is directly influenced by the frequency with which the charge turns over, and the energy generated on impact. The

breakage due to abrasion/attrition in the body of mill charge occurs as the relevant body of charge moves by a series of layers within this region which slip over one another. This slipping develops an angular velocity gradient in the charge. This motion promotes breakage due to abrasion and attrition and is a direct relationship of the mill speed and size distribution of the charge.

3.2.3 Material Transport

Due to the variation of particle size of material flowing out the mill, it is not accurate to describe this flow by fluid dynamics, although a simplification can be made by considering two fractions of slurry; the first containing size particles up to and including size, x_m, the second containing larger particles. The smaller particles can be considered to behave like a fluid within the slurry, while the larger particles do not. Napier-Munn *et al.* (1996) describe the flow of the first class as flow through *the pool zone* and the second class as flow through the *grinding media.* These authors also develop an empirical relation for the two descriptions of fluid flow, given below for the respective fractions:

$$\begin{aligned} Q_m &= 6100 J_{pm}^2 \gamma^{2.5} A \phi^{-1.38} D^{0.5} \\ J_{pm} &\leq J_{\max} \end{aligned} \tag{3.6}$$

$$Q_t = 935 J_{pt} \gamma^2 A D^{0.5} \tag{3.7}$$

$$J_p > J_{\max} \tag{3.8}$$

$$J_{pt} = J_p - J_{\max} \tag{3.9}$$

$$J_{\max} = 0.5 J_t - J_{po} \tag{3.10}$$

$$J_{po} = 0.33 \left(1 - r_n\right) \tag{3.11}$$

$$J_p = J_{pg} - J_{po} \tag{3.12}$$

where:

A	total open area of apertures (m^2)
D	mill diameter (m)
J_{pg}	gross fraction of mill volume occupied by slurry
$J_{\max}$	maximum net fraction slurry hold-up in the grinding media zone
J_{po}	dead fraction of mill volume which has to be occupied by slurry
J_p	net fraction of mill volume occupied by slurry
J_{pm}	net fraction slurry hold up in the grinding media interstices
J_{pt}	net fraction slurry hold up in the slurry pool
J_r	fraction of mill volume occupied by grinding media (balls plus coarse rocks) including associated interstices
k_g	factor to account for coarse material
Q_m	flowrate through grinding media zone
Q_t	flowrate through the pool zone
r_n	radius of the outermost row of grate apertures as a fraction of the mill radius.
f	mill speed (fraction of critical)
g	mean relative radial position of grate apertures

The total flowrate out of the mill can now be expressed by the following equation:

$$Q = k_g \left(Q_m + Q_t\right) \tag{3.13}$$

where:

k_g	factor to account for coarse material
Q	total slurry flowrate out of the mill

The factor k_g is incorporated to account for the coarse solids which also flow out of the grate. Typical values of k_g suggested by Napier-Munn *et al.* (1996) are showing in Table 3.1.

Table 3.1: Typical values of k_g

Apertures	k_g
grates only (>19mm)	1.05-1.1
grates only (19-38mm)	1.1-1.5
grates > 38mm and pebble ports	1.15-1.25

As the number of pebble ports increases the value of k_g will also increase. At the point where the open area of pebble ports is equal to that of the grates, the value of k_g would be about 1.25.

3.3 A Dynamic Comminution Circuit Model

This section serves to present a full dynamic model of a comminution circuit to be used in the operability analysis. The circuit modelled is identical in structure to the Impala Primary Comminution Circuit, however model-parameters and unit sizes are taken from a pilot plant investigated by Rajamani and Herbst (1991a,b). A circuit diagram of the comminution circuit under investigation is shown in Figure 3.8.

It can be seen in the brief discussion of the steady state model, that a dynamic model can be derived from the steady state model by essentially adding appropriate accumulation terms.

Rajamani and Herbst (1991a) use a convenient model expression splitting the continuous size distribution ranges of streams in a comminution circuit into a set of n finite size classes, with a maximum size, d_1 and a minimum size d_n. The ith size interval is bounded by d_i above and d_{i+1} below. This is the same approach as that taken by Napier-Munn *et al.* (1996).

For the purpose of considering the size distribution of all streams around the circuit, the continuous size range is sub-divided into 15 discrete sizes, each with an upper and lower bound. Furthermore it is assumed that all particles within a size range are of the same size (that of the mid-size).

The model was coded in *gPROMS* (Process Systems Enterprise Ltd., 1999a,b), where the connectivity of the units are handled using *streams*. Each stream connecting the units contained the mass flowrate of each discrete size fraction, the solids concentration (kg/(m^3 slurry)), and average ore density of that stream.

3.3.1 SAG-Mill Model

Having divided the particulate assembly into n size intervals, size interval 1 contains the largest particles and size interval n containing the finest particles,

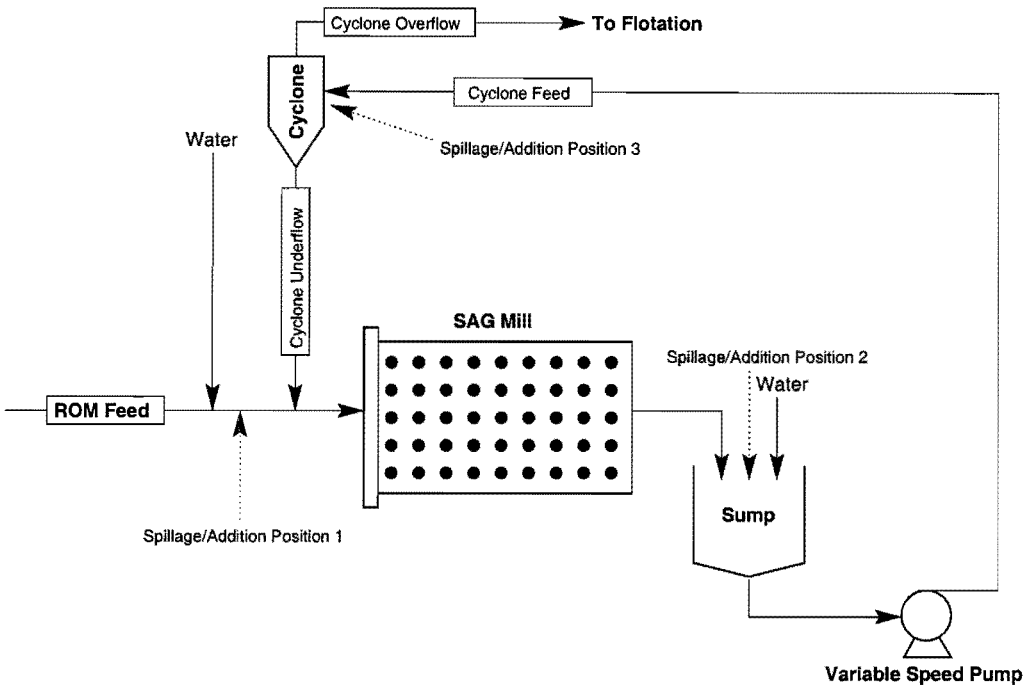

Figure 3.8: Figure showing comminution circuit under investigation.

a mass balance can be carried out for all material in each size class, i. There are two key processes occurring in the mill; firstly the process of breakage, where the rate of breakage of a particular size class is defined; secondly the breakage selection, which refers to the weighted fraction of material reporting to different size classes. Figure 3.9 illustrates these two concepts of breakage and breakage selection.

For a continuous mill, a mass balance for material in the ith size interval is shown below.

$$accumulation = input - output + generation \tag{3.14}$$

The accumulation referred to in equation (3.14) describes the mass build-up of size class i in the mill; while the generation term describes the movement of material both into and out of size class i through the breakage mechanism. Thus

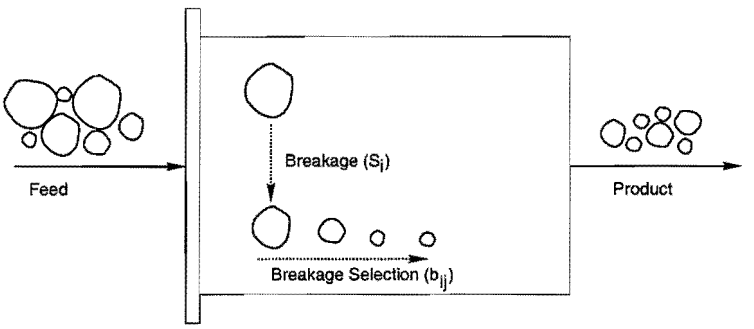

Figure 3.9: Figure showing breakage and breakage selection.

the mass balance is shown in equation (3.15).

$$\frac{\mathrm{d}H\left(t\right)m_{MP,i}\left(t\right)}{\mathrm{d}t} = M_{MF}m_{MF,i}\left(t\right) - M_{MP}m_{MP,i}\left(t\right) - S_iH\left(t\right)m_{MP,i}\left(t\right) + \sum_{j=1}^{i-1} b_{ij}S_jH\left(t\right)m_{MP,i}\left(t\right) \quad (3.15)$$

where:

$H(t)$	total mass hold up of solids in the mill
$m_{MP,i}(t)$	mass fraction of size class i in stream MP
$M_{MP,i}(t)$	mass flowrate of stream MP
S_i	size discretized breakage function
$b_{i,j}$	size discretized breakage selection function

subscript:

MP	mill product stream
MF	mill feed stream
i, j	size fraction indices

The size discretized breakage function, S_i, refers to the fractional rate at which material in size class i is broken out of that size fraction. $b_{i,j}$, the size discretized breakage selection function refers to the amount of material that is broken out

of size fraction i reports to size fraction j; it is thus a lower triangular matrix of dimension n, the total number of discretized size fractions.

The breakage function for a given size class i and a specific material is proportional to the specific power draft of the mill as shown in equation (3.16).

$$S_i = S_i^E \left(\frac{P}{H} \right) \tag{3.16}$$

where:

S_i	breakage function
S_i^E	specific breakage function
P	power draw of the mill
H	total mass hold up in the mill

The breakage function is a function of both the mill characteristics and the specific ore type being milled. It can be assumed that there is a direct relationship between ore hardness and the specific breakage function S_i^E. In order to investigate the effect of varying ore hardness, section 3.3.4 describes modifications made to the SAG mill model to account for the variation of ore hardness in the ROM feed stream to the circuit.

The solids concentration within the mill is governed by equation (3.17), which also accounts for the hold-up of water in the mill.

$$H(t) = V_m C_{S,MP} \tag{3.17}$$

where:

V_M	volume of mill occupied by slurry (m^3)
$C_{S,MP}$	solids concentration of the mill product (kg_{ore}/m^3_{slurry})

The solids concentration is defined as the mass of solids per unit volume of slurry (*slurry volume is the combined volume of water and ore*). Assuming that the slurry in the mill is perfectly mixed, the variation of solids concentration in the mill as a function of time is shown in 3.18.

$$V_M \frac{\mathrm{d}C_{S,MP}}{\mathrm{d}t} = Q_{MF}(t)\left(C_{S,MF}(t) - C_{S,MP}(t)\right) \tag{3.18}$$

where:

$C_{S,MP}$	solids concentration of the mill feed
Q_{MF}	volumetric flowrate of the mill feed

The breakage and breakage selection functions were determined empirically by Rajamani and Herbst (1991a).

For complete dynamic description of the mill, $n+1$ differential equations are required, i.e. n differential equations for the description of mass and one for solids concentration/water retention.

A key assumption made in this model is that the volumetric slurry hold up in the mill, V_m, remains constant through time. The total mass hold up, H does however vary with time as the solids concentration in the mill varies.

3.3.2 Sump Model

The sump is modelled as a perfectly mixed tank where no breakage of particles through collision with each other, the impeller or the tank itself occur. A mass balance over the sump yields equation (3.19), in terms of the accumulation of the solids in the ith size interval present in the sump.

$$\frac{\mathrm{d}H_s(t)m_{SP,i}(t)}{\mathrm{d}t} = M_{MP}(t)\, m_{MP,i}(t) - M_{SP}(t)\, m_{SP,i}(t) \tag{3.19}$$

where

H_s	mass hold up in the sump
subscript:	
MP	mill product
SP	sump product

The relationship of solids concentration in the sump (*i.e. water hold up*) is obtained from an overall mass balance around the sump:

$$\frac{\mathrm{d}\left(V_s C_{S,SP}\right)}{\mathrm{d}t} = Q_{MP} C_{S,MP} - Q_{SP} C_{S,SP} \tag{3.20}$$

where:

V_s	total volume of slurry hold up in the sump

3.3.3 Hydrocyclone Model

As mentioned previously the dynamic modelling of a hydrocyclone may be simplified by assuming that the hold up in the cyclone is negligible, and thus the cyclone responds instantaneously to any changes in the feed characteristics. To this end a simple steady state model is all that is required to fully describe the dynamic operation of the hydrocyclone. The hydrocyclone model used here is adapted from an empirical approach proposed by Lynch and Rao (1975). The model equations are as follows:

$$\begin{aligned} WOF &= a_1 WF + a_2 \\ \log_e &= a_3 + a_4 Q_c + a_5 f_v \\ Y_i &= 1 - \exp\left[-0.693\left(\frac{d_i}{d_{50}}\right)^{a_6}\right] \\ R_f &= a_7 + a_8 \frac{WOF}{SF} \\ E_i &= Y_i\,(1 - R_f) + R_f \end{aligned} \tag{3.21}$$

where:

WOF	water flowrate in the cyclone overflow
WF	water flowrate in the cyclone feed
d_{50}	size at which 50% of the solids of that size reports to the cyclone underflow
Q_c	volumetric flowrate of cyclone feed
f_v	volumetric fractions of solids in the cyclone feed
R_f	fraction of fines reporting the cyclone underflow
E_i	fraction of particles in size class i that report to the cyclone underflow
Y_i	fraction of particles reporting to the underflow corrected for those carried off (*entrained*) by water
$a_{1,\ldots,8}$	empirical constants

The eight empirical constants $a_{1,\ldots 8}$ must be determined experimentally or obtained from sampled data, and the water split equation used to calculate the value of WOF is shown below:

$$\begin{aligned} &\text{if} && WF > 21.4 \\ &&& WOF = 1.363WF - 10.75 \\ &\text{else} \\ &&& WOF = 0.837WF + 0.35 \end{aligned} \tag{3.22}$$

3.3.4 Circuit Model Modifications

Varying ore hardness

In order to account for a variation in ore hardness in the feed stream to the mill, the model is duplicated into two separate parallel models. In the first model a lower breakage function is assumed, while in the second a higher breakage function is assumed. The total mill volumetric hold up, V_m is kept the same, but is now made up of two components. In this way there are also two ROM feed streams to the circuit, where the rate of each can be varied simultaneously, thus simulating a variation in the ore hardness of the feed stream. The constant mill volumetric hold-up is still maintained however the relative volumes of 'hard' and 'soft' ore hold up are allowed to vary. The split streams remain separate entities throughout the circuit; however the combined mass flowrates of specific size classes are used in the model equations of the cyclone.

SAG Mill Model

The mass balance around the mill, shown in equation (3.15), is now split up into two equations as follows:

$$\frac{\mathrm{d}H_1(t)\,m_{1,MP,i}(t)}{\mathrm{d}t} = M_{1,MF}m_{1,MF,i}(t) - M_{1,MP}m_{1,MP,i}(t) - S_{1,i}H_1(t)\,m_{1,MP,i}(t) + \sum_j^{i-1} b_{ij}S_{1,j}H_1(t)\,m_{1,MP,i}(t) \quad (3.23)$$

$$\frac{\mathrm{d}H_2(t)\,m_{2,MP,i}(t)}{\mathrm{d}t} = M_{2,MF}m_{2,MF,i}(t) - M_{2,MP}m_{2,MP,i}(t) - S_{2,i}H_2(t)\,m_{2,MP,i}(t) + \sum_j^{i-1} b_{ij}S_{2,j}H_2(t)\,m_{2,MP,i}(t) \quad (3.24)$$

where

subscript:

1	stream 1
2	stream 2

with:

$$H(t) = H_1(t) + H_2(t) \quad (3.25)$$

The balance for the solids concentration hold-up remains unchanged, as it is assumed that the streams 1 and 2 are perfectly mixed and thus both carry the same effective solids-concentration.

Equation (3.16) is also split into two different equations as shown in equation (3.26) and (3.27). The assumption is also made that the power draft of the mill is distributed proportionally to the mass hold up of each stream in the mill.

$$S_{1,i} = S_{1,i}^E\left(\frac{P}{H}\right) \quad (3.26)$$

$$S_{2,i} = S_{2,i}^E\left(\frac{P}{H}\right) \quad (3.27)$$

Sump Model

The sump model remains almost unchanged, except that there are now twice as many 'size fractions' to consider in the mass balance. The similar size fractions carried in the two split streams are treated as separate size classes altogether.

Varying ore density

The inclusion of varying ore density in the feed stream is possible using the relationship between solids concentration of a particular stream, C_s $(kg_{\text{ore}}/m^3_{\text{slurry}})$, to ore density, ρ $(kg_{\text{ore}}/m^3_{\text{ore}})$, mass flowrate, M (kg_{ore}/s), and water flowrate, Q_w (m^3/s):

$$C_s = \frac{M}{\frac{M}{\rho} + Q_w} \tag{3.28}$$

All variables in the above expression are time dependent variables.

3.4 Operation and Control of Comminution Circuits

3.4.1 Operation of Comminution Circuits

Details of some of the process interactions involved in a comminution section are outlined below (Napier-Munn *et al.*, 1996).

Throughput The throughput of the mill is mainly affected by three process variables, namely the speed and power draw of the mill and the mill charge (*volumetric charge*). An increase in mill speed will result in a higher throughput, as will a higher mill charge which requires a higher power draw of the mill to maintain the same mill speed. In terms of optimizing a mill, the mill load (*power draw*) will have to be monitored and compared to the maximum power draw possible.

An increase in throughput also has an effect on the product size of the resulting stream, so some form of compromise must be found.

Grind Size The Grind size achieved by a mill is a function of many parameters and is not easy to manipulate. Some of the process variables affecting the grind size of product are as follows

- Screen/Trommel Apertures
- Cyclone Size (*in the case of a closed circuit configuration*)
- Mill Charge
- Ball Loading
- Mill Speed/Power draw

- Feed Mineralogy

In open circuit configurations it is very difficult to change or maintain the grind size of product, whereas in closed circuit operation this process is easier due to the cyclone or screen on the recycle stream.

Power Consumption Specific power consumption is often used to indicate the efficiency of the mill. The power consumption of the mill is dependent on the ore type and mineralogy as well as feed size, ball loading and mill charge.

Steel Consumption Steel consumption is dependent on ore type as well as liner/ball material. However certain operating conditions will lead to an increase in ball consumption. If the mill charge is too low or too high, this will give rise to a higher steel consumption. Mill speed also affects the steel consumption rate of both liners and balls. If the mill charge is very low, the liners and lifters on the inside of the mill are left unexposed leading to an unnecessarily high wear rate.

3.4.2 Control of Comminution Circuits

Various authors have presented different methods of advanced control for industrial comminution circuits. No single method appears dominant in the literature. The specifics of the control systems do not form the scope of this review. The table below shows some of the authors and the types of control they have implemented to Comminution Circuits.

Commonly used controlled and manipulated variables in comminution circuits (*for the general case of a comminution circuit inclusive of a mill, cyclone and sump*) are shown in Table 3.2.

Rajamani and Herbst (1991b) implemented optimal control to a pilot scale grinding circuit containing a ball mill, cyclone and sump. A simplified version of the

Table 3.2: Common controlled and manipulated variables

Controlled Variables	Manipulated Variables
Mill throughput rate	Fresh feed solids rate
Mill discharge density	Fresh water rate
Cyclone feed density	Sump water rate
Cyclone mass feed rate	Cyclone Feed rate
Sump level	
Overflow product particle size	
Mill power	

model derived in the precious section of this report was used for the controller tuning and pre-simulation of the process. The sump water addition rate and the fresh solids flow rate were used to try and control the product particle size and the mill throughput.

Herbst *et al.* (1992) show the application of model-based control to three grinding circuits, namely a sag mill circuit, a rod and ball mill circuit and a ball mill circuit. In all three the control implemented is of a supervisory nature. The model used for the SAG mill is described in the previous section.

In the first instance, a Kalman filter was used to estimate ore hardness, volume filling and mill percent solids, which were then used to control the mill power draw by determining the feed rate.

In the second instance an optimal control law was applied that minimizes the cumulative control error. The controlled variables were overflow particle size and slurry volume in the sump. The manipulated variables used were fresh ore feedrate and sump water addition rate.

In the third example, a Kalman filter was employed to estimate the percent solids of mill charge and the closeness of approach to mill/cyclone overload and ore hardness. These parameters were used in conjunction with an 'expert' system to drive the system to it's nominal operating point by adjusting the feed rate to the mill.

Hulbert *et al.* (1990) implemented a multi-variable control structure on a milling

circuit at a gold mine. The manipulated variables used in this implementation were the feed rate of water to the sump, feed rate of solids to the mill, flowrate of slurry to the cyclone and feed rate of fresh water to the mill inlet. The plant outputs controlled were the product size of the cyclone overflow, the mill load, the slurry level in the sump and the density of feed to the cyclone.

There are numerous other authors describing different types of control applications to comminution circuits, but most of them use similar approaches to those described above, making use of similar control objectives through the use of similar manipulated variables, eg. (Hales *et al.*, 1988; Hulbert and Woodburn, 1983; Desbiens *et al.*, 1994).

Chapter 4

Optimization Formulation

This chapter serves to present the operability optimization formulation based on scenario studies carried out and information gathered relating to the operational objectives of the process.

The formulation of the problem is presented in two major sections building up to the final problem specification. In the first section a number of scenarios are presented based on the model of the process, where various aspects of the design and control of the comminution circuit are considered, with a means to formulating a realistic problem. The second section presents the final formulation in terms of the application to an industrial comminution circuit. Finally this chapter is concluded with a description of the implementation of the formulation in the chosen modeling and optimization environment *gPROMS* (Process Systems Enterprise Ltd., 1999a,b).

4.1 Scenario Studies

4.1.1 Introduction

The model described in the previous chapters was used to perform a number of scenario studies for comparative purposes. The main purpose of performing these scenario studies is to determine possible and realistic controller loop pairings, as well as to investigate the dependence of selected outputs on model parameters. A number of studies were carried out using the model in standard form with a PI controller controlling the level of slurry in the sump by manipulating the sump outflow rate. An operational objective described in the introduction to this thesis is to maintain as near constant particle size in the product stream in the face of disturbances to the process; to this end, the product stream is monitored in terms of the % passing 40μm in the cyclone overflow stream.

A process flow diagram is shown in Figure 4.1 describing the positions of all streams discussed in this section.

The results of the scenarios presented here are all based on a comparison to the base case, where only sump control is implemented. For each possible scenario, the output was monitored when the model was subjected to a number of different disturbances. In all, 6 different disturbances were considered. The details of each disturbance examined are shown below.

- Mass feed rate
  - Mass feed rate to the circuit was varied sinusoidally with an amplitude of 40 kg/h and a period of oscillation of 1 hour.
  - Mass feed rate was subjected to a pulse disturbance (step-up step-down) of amplitude 20 kg/h and a duration of 2 hours.

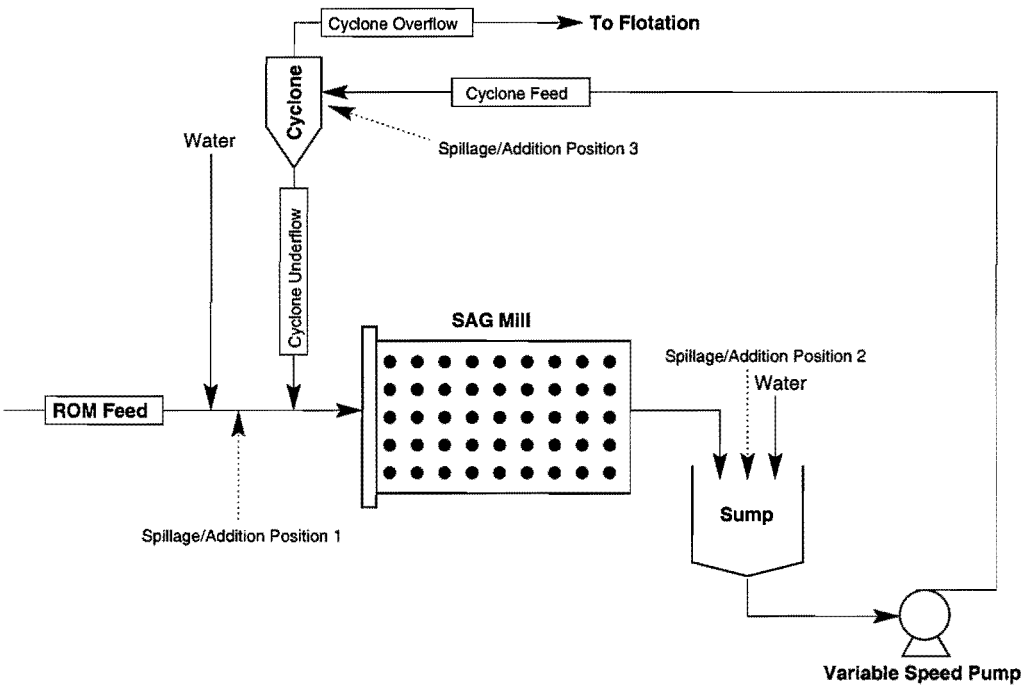

Figure 4.1: Figure showing comminution circuit under investigation.

- Spillage
  - Spillage was simulated as a pulse disturbance of water with an amplitude of 50e-3 (m^3/h) and a duration of 2 hours, at each of the addition points (1,2 and 3 – see Figure 4.1) individually (separate simulations).
- Feed size distribution
  - The ROM feed size distribution was varied sinusoidally with an amplitude of 20% and a period of oscillations of 1 hour.

The scenario studies are split up into two distinct sections, namely scenarios concerning discrete variables and those concerning continuous variables.

4.1.2 Discrete variables

Plant Spillage

The first of the discrete variables investigated is the position of spillage addition to the process. In this scenario, the spillage was modeled as a pulse disturbance to the three spillage addition points individually. The pulse disturbance was of magnitude 50e-3 (m^3/h) for a period of two hours. The simulations ran for a total of 4 hours. In all cases a PI controller was used to control the sump level, manipulating the volumetric flowrate out the sump.

Figure 4.2 shows how the spillage disturbance at the different positions affects the cyclone product stream. As can be seen from this response, the best possible placement for the return of plant spillage appears to be at position 1. This is also intuitive as the circuit now has a larger "buffer" to flow variations than if the spillage is returned closer to the cyclone, the primary particle size splitter in this circuit. It can also be seen that the worst possible position appears to be position 3, which feeds directly into the cyclone. However, this simulation is not carried out at optimal operating conditions or controller tuning around the sump, so the addition of spillage to the circuit is revisited in the results section.

Controller Configuration

A number of single loop controller pairings were investigated. In each case the controller tuning around the sump was kept constant, and each scenario was investigated for the list of disturbances shown on page 63.

In each case the single loop controller was tuned using the direct synthesis method (Seborg *et al.*, 1989). It is very important to note that the tuning is not necessarily optimal, and that the purpose of the selected scenario studies is to determine which controller pairings are possible and to distinguish them from those, if any, that would result in infeasible operation regardless of tuning. In particular this exercise also shows which intermediate variables could be maintained at a

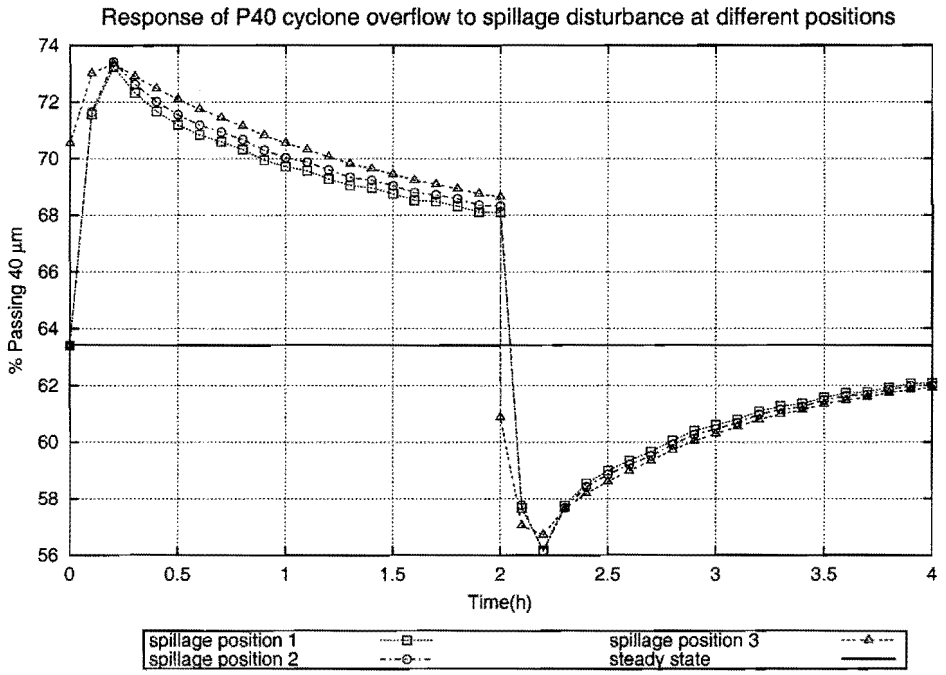

Figure 4.2: Graph showing the effect of spillage addition position on P_{40} cyclone overflow

set-point to achieve better operation of the process. The selection of the best controllers in terms of configuration and tuning are carried out in the dynamic optimization framework.

In one case a property of the product stream (cyclone overflow) was maintained by manipulating one of the inputs to the process. The two possible manipulated inputs to the process were chosen to be:

1. Feed water to the SAG Mill (*water addition position 1*)
2. Feed water to the Sump (*water addition position 2*)

In a second case an intermediate stream was chosen as the controlled variable, namely the properties of the cyclone feed stream. It was decided not to include feed water to the cyclone as a manipulated input, due to physical limitations of the process.

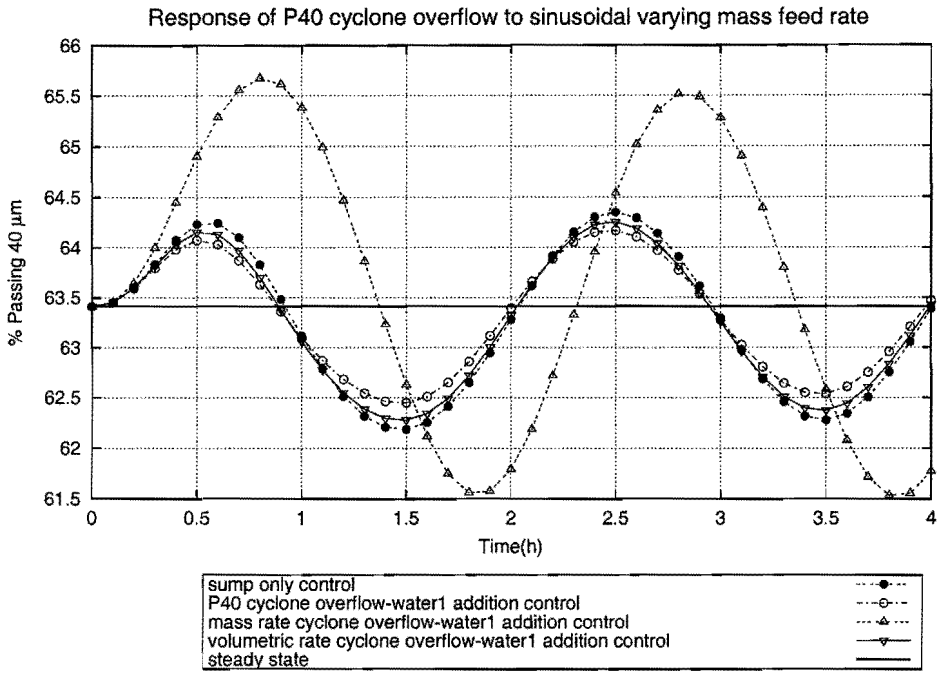

Figure 4.3: Effect of a sinusoidally varying feed mass rate on the product size.

Product Variables Table 4.1 shows the list of single loop pairings investigated, where a property of the product stream was controlled.

Table 4.1: Table showing investigated single loop pairings of the product stream

	Manipulated Variable	Controlled Variable
1	Feed water to position 1	% Passing 40 μm of Cyclone Overflow
2	Feed water to position 2	% Passing 40 μm of Cyclone Overflow
3	Feed water to position 1	Mass flowrate of Cyclone Overflow
4	Feed water to position 2	Mass flowrate of Cyclone Overflow
5	Feed water to position 1	Volumetric flowrate of Cyclone Overflow
6	Feed water to position 2	Volumetric flowrate of Cyclone Overflow

Due to the large number of possible combinations, only selected figures are shown in this and subsequent sections in order to best illustrate the results.

For each controller pairing shown in Table 4.1, the model was subjected to all six types of disturbances individually, and the cyclone overflow stream monitored.

Figure 4.3 displays the effect of a sinusoidally varying mass feed rate on the

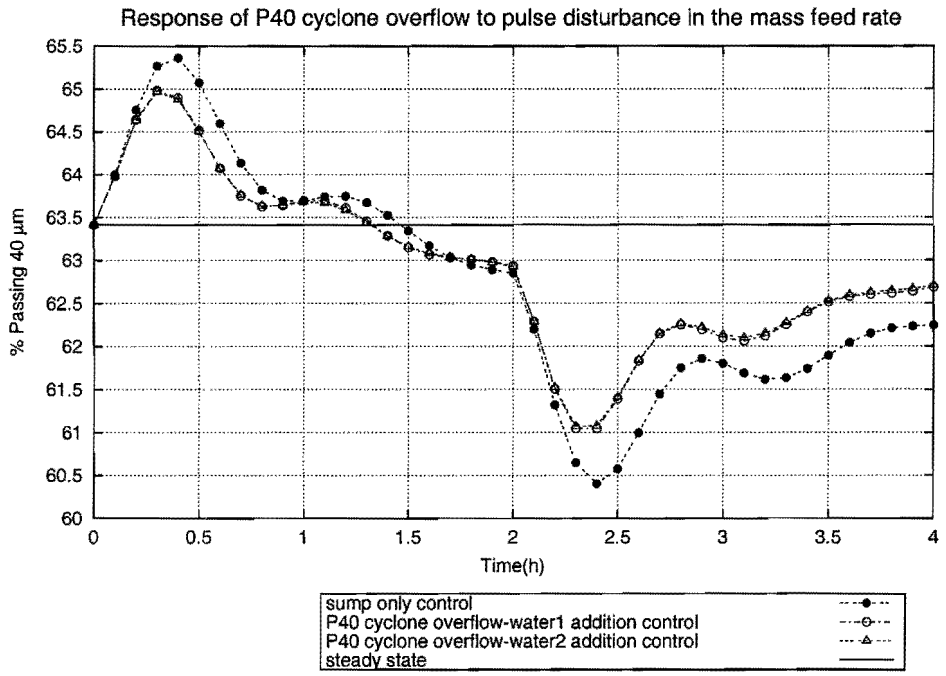

Figure 4.4: Effect of a pulse mass disturbance in the mass feed rate on the particle size of cyclone overflow

product size. It appears from this response that controlling the mass product rate of the cyclone overflow by manipulating the water addition at position 1 seems a poor strategy with respect to maintaining a consistent product size. Controlling the volumetric rate and particle size of the product stream appear reasonable and improve the dynamic response of the process in terms of the volumetric flowrate and product size. It is also apparent from Figure 4.3 that the product size and volumetric flowrate of the cyclone overflow are closely tied, i.e. if you maintain a more consistent volumetric flowrate in the product stream, the variation in product size will be reduced. When the mass rate in the cyclone overflow is controlled the particle size is seen to fluctuate from it's nominal state more than when there is no additional controller present, and thus using the mass rate as a controlled variable does not appear viable.

Figure 4.4 shows the difference between using water addition position 1 and water addition position 2 as manipulated variables when controlling the volumetric rate of the cyclone overflow stream when subjected to a pulse disturbance in the

mass feed rate. As can be seen from this graph, there is little difference when comparing the water 1 and water 2 positions as manipulated variables with the same control structure. The SAG mill model being used assumes a constant volumetric hold up within the mill, so any change in volumetric rate to the mill feed will immediately cause a similar change in the discharge, therefore it is not expected that the addition position of water will have a pronounced effect on the particles size of the cyclone overflow stream.

Intermediate Variables The effect of all 6 disturbances shown on page 63 is investigated for each controller pairing shown in Table 4.2. As before the cyclone overflow stream is monitored in terms of the particle size. Previously the product stream was controlled directly, in this set of scenarios, an intermediate variable is controlled that does not directly involve the product stream.

Table 4.2: Table showing investigated single loop pairings of an intermediate stream

	Manipulated Variable	Controlled Variable
1	Feed water to position 1	% Passing 40 μm of Cyclone Feed
2	Feed water to position 2	% Passing 40 μm of Cyclone Feed
3	Feed water to position 1	Mass flowrate of Cyclone Feed
4	Feed water to position 2	Mass flowrate of Cyclone Feed
5	Feed water to position 1	Volumetric flowrate of Cyclone Feed
6	Feed water to position 2	Volumetric flowrate of Cyclone Feed

Figure 4.5 shows the effect of a sinusoidally varying mass feed rate on the particle size of cyclone overflow, with water 1 addition as the manipulated variable. The use of three different controlled variables are compared, namely the mass flowrate, volumetric flowrate and size of the cyclone feed stream. This graph shows that using both the volumetric flowrate and the particle size as controlled variables produces a lower product variation. This scenario also shows that as with the product variables, the use of mass feedrate the the cyclone as a controlled variable appears to be poor as it results in a larger variation in product particle size of the product stream.

Figure 4.6 illustrates the effect of water addition position as the manipulated variable when the process is subjected to a pulse spillage addition at position 3. It can be seen from this response that the chosen water addition position makes little or no difference to the resulting particle size variation. As with the product variables this comparison can only be made when the optimal tunings of all controllers and process units has been carried out.

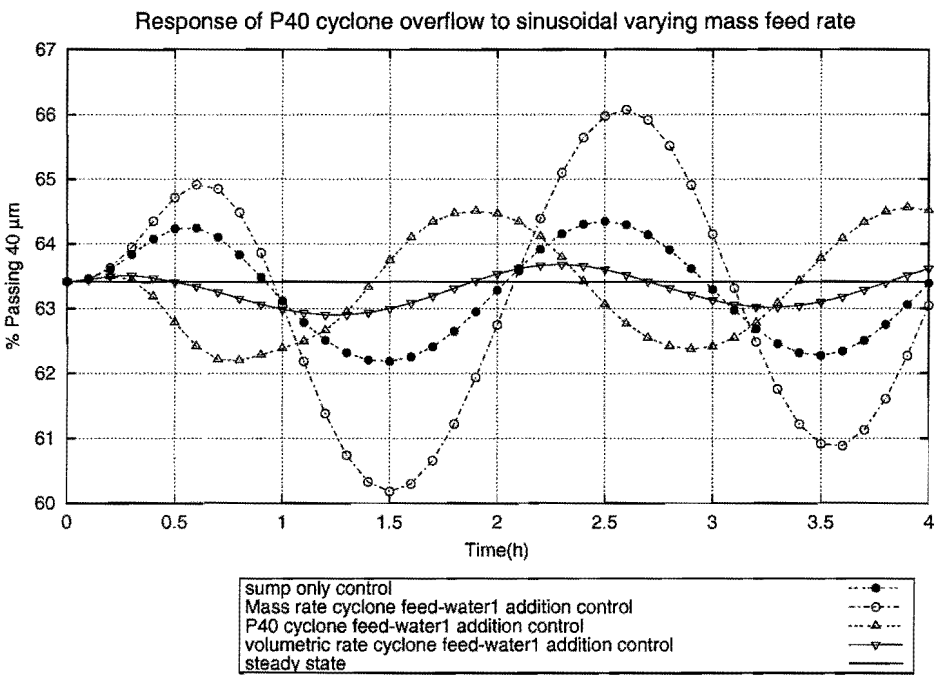

Figure 4.5: Effect of a sinusoidally varying mass feed rate on the particle particle size of cyclone overflow

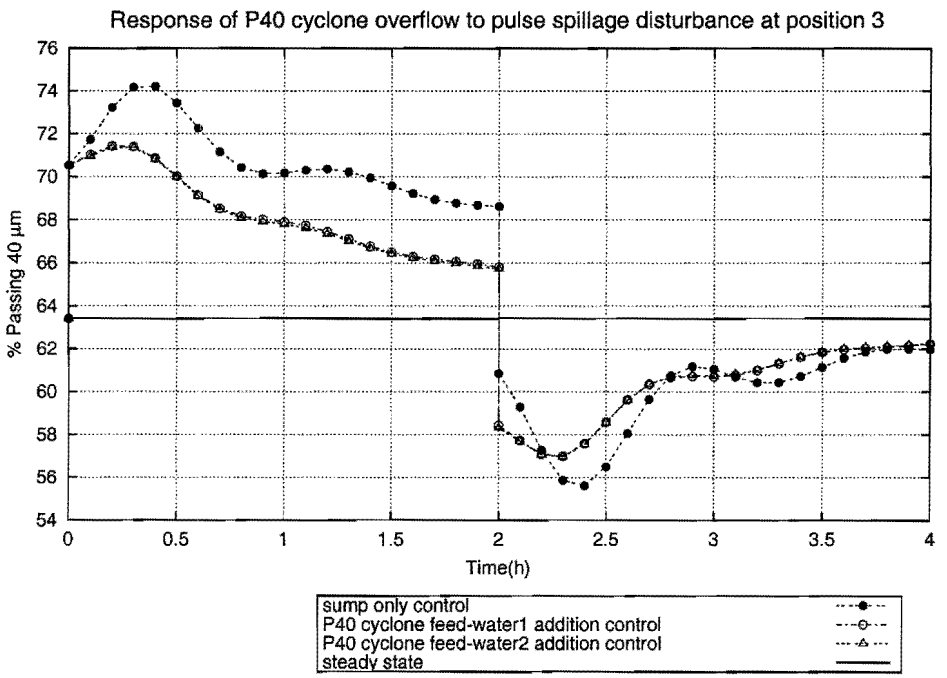

Figure 4.6: Effect of a pulse spillage disturbance at position 3 on the particle particle size of cyclone overflow

Controller Pairing Summary From the scenario studies regarding the choice of appropriate single loop controller loop pairs, it is clearly evident that the choice of mass flowrate in either the cyclone feed or the cyclone product be disregarded at this point if the object of operation is to maintain a consistent grind size. From a practical standpoint, the use of the particle size as a controlled variable in any stream is not considered due to the difficulties associated with making such a measurement on a full scale plant. It was also shown that the use of water addition at positions 1 and 2 both be included as candidates for single loop controllers as they both exhibit fairly similar behaviour, with neither being superior. This leaves the process with four possible combinations of single-loop controllers as shown in Table 4.3, in addition to sump control.

Table 4.3: Table showing single loop candidates

Manipulated Input		**Controlled Output**
water - position 1	$\rightarrow$	cyclone volumetric discharge
water - position 2	$\rightarrow$	cyclone volumetric discharge
water - position 1	$\rightarrow$	cyclone volumetric feedrate
water - position 2	$\rightarrow$	cyclone volumetric feedrate

4.1.3 Continuous Variables

Table 4.4 shows a list of scenarios carried out where the effect of continuous variables are investigated. These are design and/or operational variables. As with the control loop pairing scenarios, the purpose of carrying out these studies is to determine which variables have an appreciable effect on the product stream and hence should be included in the optimization formulation. These studies do not show any optimal condition.

Table 4.4: Table showing investigated continuous variables

	Variable
1	Sump controller tuning
2	Sump level set-point
3	Mill Loading
4	Ore density
5	Ore Hardness

Sump Controller Tuning

Figure 4.7 shows the variation of cyclone overflow when subjected to a pulse spillage disturbance at position 1 in terms of the particle size of cyclone overflow. In each case, a 'tight' vs 'loose' controller around the sump level is considered. It is evident from the graphs, that the response to a disturbance is more rapid with a 'tighter' controller (a PI controller with a higher gain and smaller integral time constant) than a 'looser' one but with a larger overshoot. This is due the damping effect of the sump, which will be greater when the sump level is allowed to fluctuate more. The extent to which one could reduce the gain on the sump is limited by the capacity of the sump in terms of not allowing the sump to overflow or to run-dry. These scenarios here show two extremes in terms of the sump controller tuning, and the optimal controller tunings should be determined within the dynamic optimization formulation.

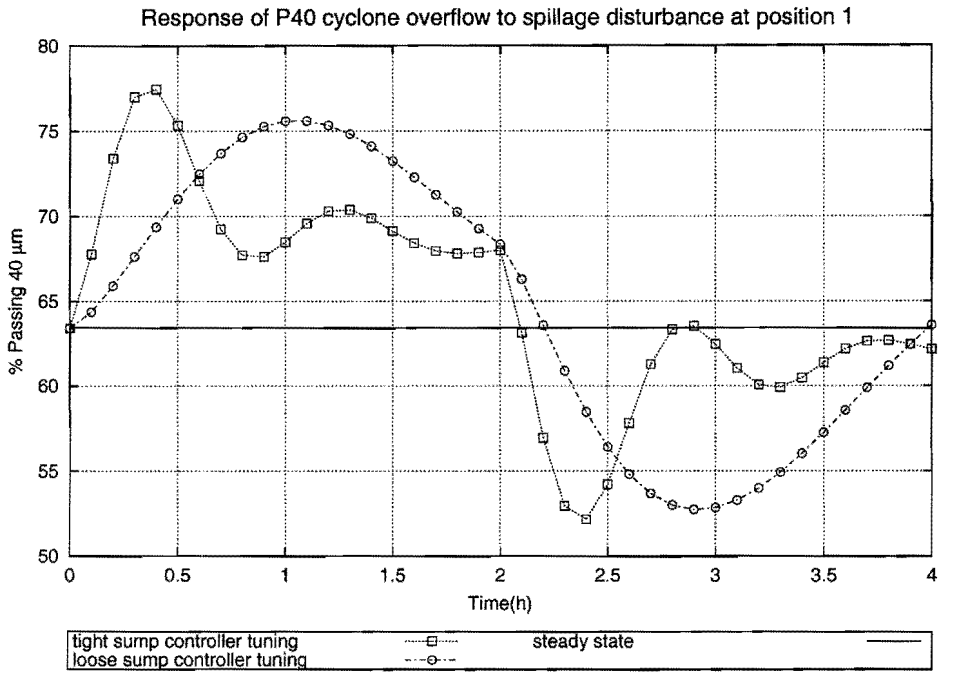

Figure 4.7: Effect of a pulse spillage disturbance at position 1 on the particle size of cyclone overflow

Sump level set-point

Figure 4.8 shows the variation of cyclone overflow when subjected to a pulse spillage disturbance at position 1 in terms of the particle size of cyclone overflow. In each case a change in the sump level set-point is considered, firstly a set-point of 0.5 m and secondly a higher value of 0.75 m. In both cases it can be seen that a larger value of set-point produces a larger variation in the product output for a spillage disturbance at position 1.

The noted variations in the product stream to variations of the sump-level, sump set-point and controller sump controller tuning, suggest that all three variables be included in the set of design variables of the operability optimization formulation. From these observations it also noted that the actual size of the sump be included as a design variable, based on the difference in performance observed with a change of sump set-point; as a change in the total sump volume would have a similar effect to changing the set-point of the sump-level controller.

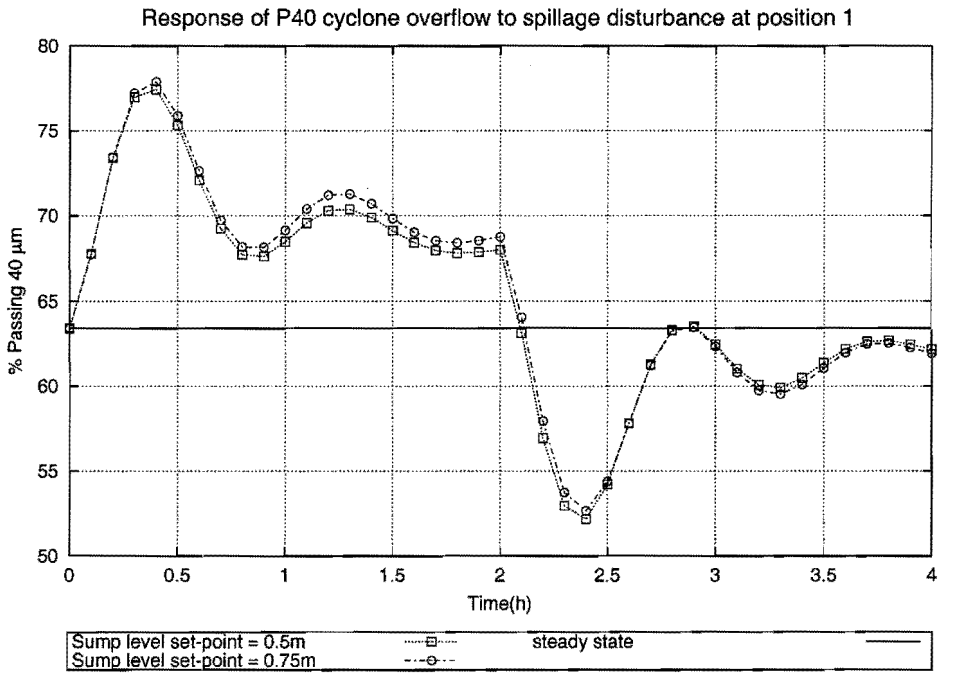

Figure 4.8: Effect of a pulse spillage disturbance at position 1 on the particle size of cyclone overflow

Mill Loading

In these scenarios two different mill loadings are considered, firstly a loading of $0.05m^3$ (which is the mill loading used in all other cases), and secondly a mill loading of $0.1m^3$. Figure 4.9 shows the variation in the product stream when the circuit is subjected to a sinusoidally varying feed size distribution.

In all cases the variation in cyclone overflow appears to be reduced when the mill loading is increased, and thus the mill loading should be included in the dynamic optimization formulation. As before this result does not mean that the mill loading should be increased to the maximum possible load, but rather that the optimal loading be sought in conjunction with all the other design variables.

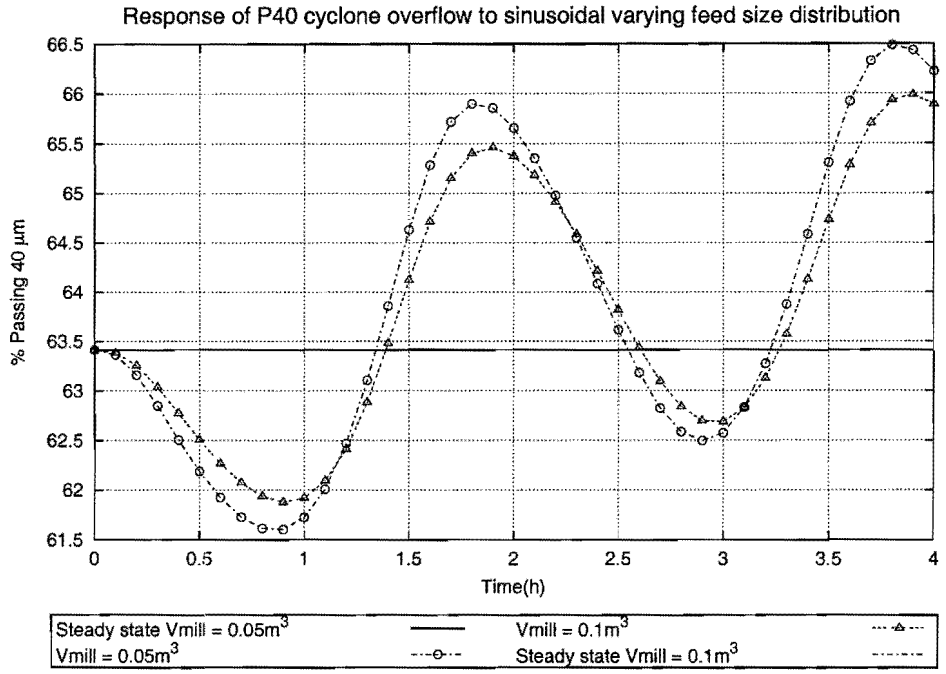

Figure 4.9: Effect of a sinusoidally varying feed size distribution on the particle size of cyclone overflow

Ore Hardness

Assuming that there is a direct relationship between ore hardness and the specific energy within the mill, the effect of changing the ore-hardness on the performance of the circuit was analyzed. A decrease of 10% in the ore hardness was simulated for the 6 types of disturbances presented on page 63. A selected result is shown in Figure 4.10, when the model was subjected to a pulse spillage addition at position 1. For both the standard case and the case where the ore hardness was reduced by 10%, the simulation was started off at steady-state so that the nominal condition could also be compared. While the variation in ore hardness is not necessarily a slow varying parameter, this section serves only to present the overall effect of a change in the ore hardness of the feed stream.

As expected the steady state mass rate and volumetric rate of the cyclone overflow stream remained the same by simple mass-balancing laws. A change was seen in the nominal value of % passing 40 μm, as can be expected, when the ore

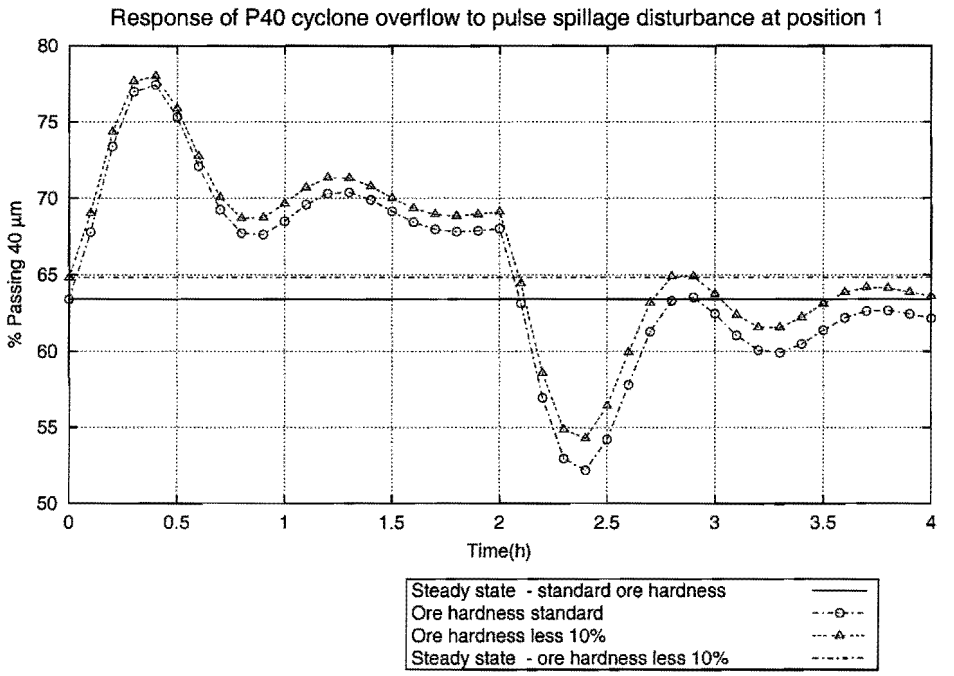

Figure 4.10: Effect of a pulse spillage disturbance at position 1 on the particle size of cyclone overflow

hardness is decreased the cyclone overflow stream will be finer.

As noted previously, variation of ore hardness is a frequently varying disturbance and should be included in the formulation as such. This set of simulations does however highlight the fact that ore hardness should indeed be included in the formulation and be considered as an infrequent disturbance variable.

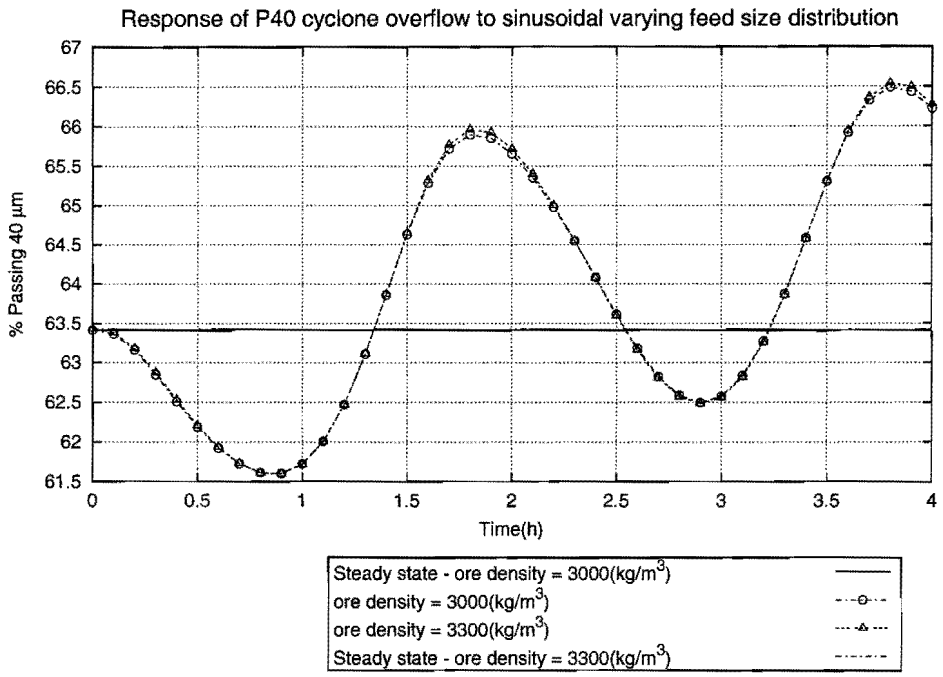

Figure 4.11: Effect of sinusoidally varying feed size distribution on the particle size of cyclone overflow

Ore Density

Ore density was increased by 10%, and the simulations started at steady state; i.e. this variation of ore density was considered to be slow varying change. Figure 4.11, shows the effect of increasing the ore density in the circuit by 10% when the model is subjected to a disturbance in the form of a sinusoidally varying feed size distribution. It is seen in these responses that there is a very small change in steady-state values of the product size.

This change is however still included in the optimization formulation.

4.2 Problem Formulation

4.2.1 Objective

It is desired to minimize the product variation when the plant is subjected to various disturbances, and for feasible operation to be satisfied over all parametric uncertainty.

Process Objectives

An operational objective derived from process analysis at Impala indicates that the dynamic requirements of the comminution circuit are to minimize the variation of the particle size of the cyclone overflow stream in light of disturbances to the process. This minimization of variation in the product stream may take several forms in terms of defining variation.

The first way to define variation is by the absolute deviation of the output and secondly by calculating the Integral Square Error (ISE) of the output around the nominal operating point over a pre-defined time-horizon. These methods are shown in Figure 4.12.

The specific choice of objective function would have to be made based on the operating philosophy of downstream process and/or product requirements. In this thesis it is chosen to minimize the absolute deviation of the particle size in the cyclone overflow stream (*see equation (4.1)*). In different problem specifications it is certainly possible to include a measure of operating and equipment cost in the objective function.

$$\Phi = \left(\max_{t \in H} \left(P_{40,\text{product}} \right) - \min_{t \in H} \left(P_{40,\text{product}} \right) \right) \tag{4.1}$$

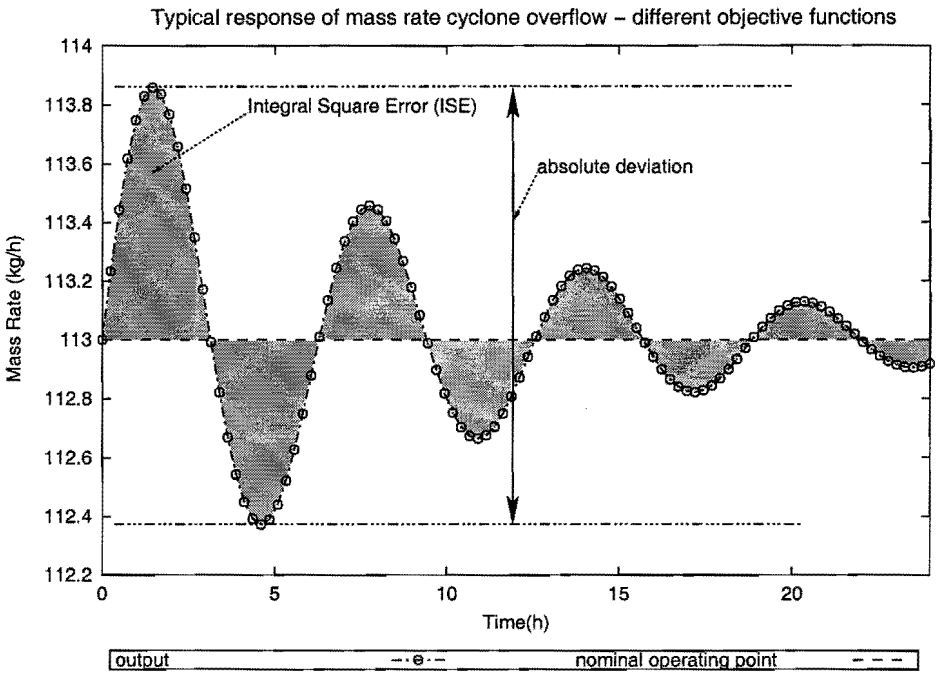

Figure 4.12: Different objective functions

P_{40} is chosen as a measure of the particle size, as such the objective function does not contain any measure of the size distribution but rather a measure of the actual size of the particles or the *grind size.*

In order to account for the expected value of the objective function over parametric uncertainty the primary objective is now written in terms of a newly defined objective shown in equation (4.2).

$$\min_{d\in D} \max_{\substack{\theta\in\Gamma \\ v(t)\in V(t)}} \left\{ \left(\max_{t\in H} \left(P_{40,\text{product}}\right) - \min_{t\in H} \left(P_{40,\text{product}}\right) \right) \right\} \tag{4.2}$$

It is also important to note that the search over the time domain is carried out within the calculation of the objective function, Φ.

4.2.2 Disturbances

The disturbances to the process identified at Impala are listed below:

- ROM feed distribution
- ROM mass rate
- ROM ore density
- ROM ore hardness
- plant spillage addition

The plant spillage addition is a one-time infrequent disturbance to the process and is not included directly in the problem formulation, but rather the optimum position for spillage addition is analyzed using the optimal operating conditions found through the optimization process. The choice of spillage addition for a given design is thus chosen on the basis of the position which produces the least variation in the particles size of the product stream as well as that which results in the least constraint violations if any.

The remaining disturbances are modelled as two discrete types each, firstly a sinusoidal variation, and secondly a pulse function. In both specification the frequency of the wave form and the amplitude are included as variables defining the space $V(t)$ on which $v(t)$ is realized.

The maximum amplitudes for both the sinusoidal and the pulse function disturbances are specified as a variation of 20% from the nominal condition. In terms of the pulse disturbance this allows for the square wave to be realized with any amplitude ranging from +10% to -10% form it's nominal operating point.

The frequency range allowed for in the sinusoidal disturbance specification ranges from a frequency of 1hr^{-1} to $\frac{1}{6}\text{hr}^{-1}$. This specification is chosen on the basis that it allows for typical plant disturbance which generally range from a time length

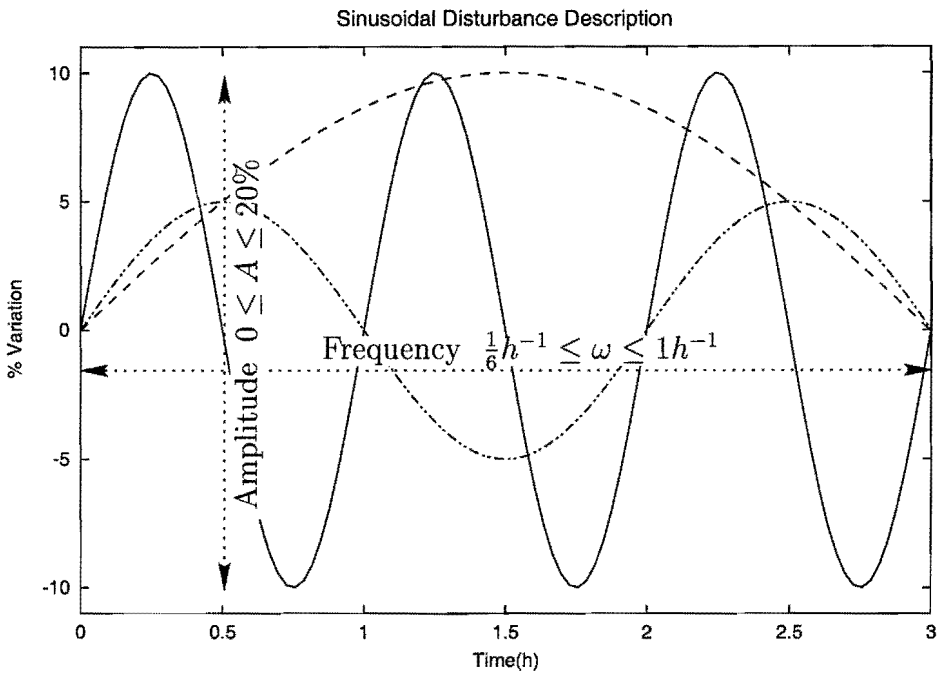

Figure 4.13: Sinusoidal disturbances

of 1hr to 6hr for completion. Similarly the square step function is specified for a range of time from 1hr to 3hr, with the time horizon in all cases being 7 hours. The amplitude of the square step function is included as a variable ranging from -10% to +10% in variation, and is thus considered to be a positive as well as negative pulse function.

This time horizon allows for the effect of both the upward and downward steps of the pulse function to be included in the time frame under investigation. Figures 4.13 and 4.14 show graphically the two disturbance profiles chosen, along with their respective bounding values.

Constraints

There are a number of constraints imposed on the process. The first of which is a path constraint on the sump level to ensure that the sump level does not vary past a minimum or maximum point when subjected to any of the disturbances discussed in section 4.2.2. This path constraint is included to prevent any

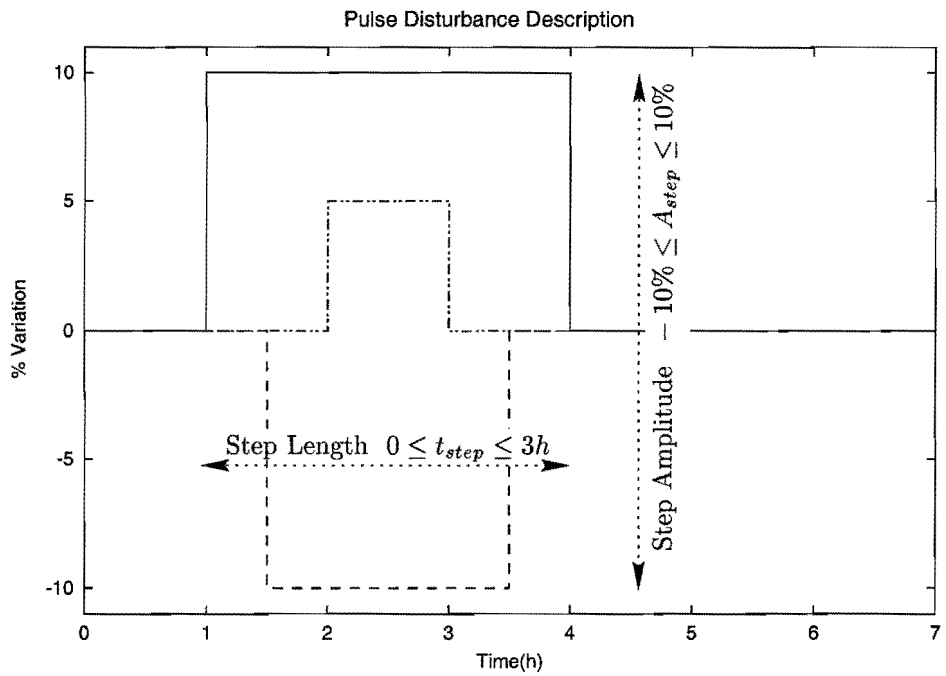

Figure 4.14: Pulse disturbance

possibility of the sump overflowing or the variable speed pump cavitating in the normal operation of the process. These constraints form the set g_k $(k \in K)$.

Other constraints include the sump outflow rate which is allowed to vary only up to 20% either side of nominal operating point. This constraint is included to model a typical varying speed pump which will be rated for a median flowrate. In a similar manner, the flowrates of the control valves used for water addition at positions 1 and 2 also have bounds on their maximum and minimum flowrates. The minimum flowrate is set to zero, while the maximum is set to 40 l/h. These constraints form a set of forced operational constraints that are enforced within the model, as opposed to those of the sump level which form the feasibility constraints of the flexibility considerations.

Uncertainty

Two uncertain parameters are included in the problem formulation; the first being the calculation of d_{50} in the hydrocyclone model, and the second the power

draw of the mill. In each case it is assumed that the the values of these two parameters are uncertain over a range of +/- 10% from their nominal or calculated values respectively. This uncertainty is typical of that which could arise in plant/model mismatch as well as any errors involved when fitting the model to plant data.

4.2.3 Continuous Decision Variables

The decision variables in the problem formulation are those which constitute both the design and operating variables. These decision variables are divided into two subsets, continuous variables and discrete decision variables. The first category identified are listed below:

1. Controller settings of all PI controllers.
2. Controller set-points of all PI controllers.
3. Nominal water addition at positions 1 and 2.
4. Volume of slurry hold-up in the mill.
5. Total sump volume.

4.2.4 Discrete Decision Variables

Various controller configurations and the optimal position of spillage addition form the set of discrete decision variables in the problem formulation. Five scenarios are considered in terms of differing controller loop pairings, and for each of these the ideal spillage addition position is also considered. This results in a total of fifteen discrete designs to consider simultaneously. As mentioned previously, the spillage addition is considered as a one-time infrequent disturbance, and is as such not included in the formulation itself, but rather the optimal position is to be found through simulations of the five different optimal designs.

The comparison of the five scenarios is carried out by determining the optimal design for each case and then drawing conclusions based on the objective functions for each. If the problem contained a far higher number of integer variables, a more systematic approach than the combinational one used would be to consider formulating the problem as a mixed-integer approach for considerations of computational time.

In all cases a controller around the sump is included. In addition to the sump controller, a further 4 possible combinations of single-input single-output controllers are considered. These 4 combinations constitute all permutations of manipulated input to controlled output variables shown in Table 4.5.

Table 4.5: Table showing available manipulated inputs and controlled output streams

Manipulated Variables	**Controlled Variables**
Feed water to position 1	Mass flowrate of cyclone overflow
Feed water to position 2	Mass flowrate of cyclone feed

4.2.5 Formulation Summary

A summary of the formulation is now shown in Equations (4.3) to (4.7).

Minimize the objective function over a specified time horizon H with z and d as decision variables, with $v(t)$ chosen as the $v(t)$ producing the the maximum objective function:

$$\min_{d\in D} \max_{\substack{\theta\in\Gamma \\ v(t)\in V(t)}} \left\{ \left(\max_{t\in H} \left(P_{40,product}\right) - \min_{t\in H} \left(P_{40,product}\right) \right) \right\} \tag{4.3}$$

The optimization is to be carried out such that the process model differential and algebraic equations and inequality constraints are valid, and the system is at rest at $t = 0$:

$$h\left(d, x\left(t\right), \dot{x}\left(t\right), v\left(t\right), \theta, t\right) = 0 \tag{4.4}$$

$$x\left(0\right) = x_0\left(d, \theta\right) \; ; \; \dot{x}\left(0\right) = 0 \tag{4.5}$$

$$g_k\left(d, x\left(t\right), v\left(t\right), \theta, t\right) \leq 0 \qquad k \in K \tag{4.6}$$

The set of differential and algebraic equations, h, corresponds to the equations describing the model as a whole including the controller equations relating manipulated and controlled variables.

The vector of inequality constraints, g_k, is the set of equations describing the sump level path constraint, where the sump level is to remain within given bounds over all time.

The vector of state variables, x, comprises all state variables in the model, and as such includes the state variables in the mill and sump models. The state variables corresponding to the mass hold up within the mill are shown in equations (3.23)

and (3.24) and those for the hold up in the sump are shown in equation (3.19), while the states corresponding to the mass of water hold up in these two units can be found equations (3.18) and (3.20) for the mill and sump respectively. Included in the vector of state variables are also the states corresponding to controller equations.

The parametric uncertainty, θ, represents the two variables, d_{50}, the cut size of the cyclone and P, the power draw of the mill. These two variables are described by equations (3.21) and (3.16) respectively.

The vector of design variables, d forms the search space for the design and contains integer and continuous variables. The integer variable description is summarised by Table 4.5, and contains the choice of multi-loop controllers considered. The continuous design variables are summarized in the list below:

- Controller settings of all PI controllers.
- Controller set-points of all PI controllers.
- Nominal water addition at positions 1 and 2.
- Volume of slurry hold-up in the mill.
- Total sump volume.

Finally the vector of disturbance variables, $v(t)$, is formed considering two types of variation over time; namely sinusoidal and pulse functions. Each description comprises a further two variables describing the frequency and amplitude of these variations. The disturbances represented by these variations are as follows:

- ROM feed distribution
- ROM mass rate
- ROM ore density
- ROM ore hardness

Dynamic Flexibility

An additional constraint is incorporated to guarantee feasibility in the face of parametric uncertainty θ and all possible disturbances on the defined range $V(t)$:

$$\max_{\substack{k\in K \\ \theta\in\Gamma \\ v(t)\in V(t) \\ t\in H}} \left\{\left(g_k\left(d, x\left(t\right), v\left(t\right), \theta, t\right)\right) \leq 0\right\} \tag{4.7}$$

4.3 Computational Issues

The problem is solved within the *gPROMS* (Process Systems Enterprise Ltd., 1999a,b) environment, which is a dynamic modelling package capable of solving both lumped and distributed systems. *gPROMS* uses a direct algebraic equation (DAE) solver to solve the set of algebraic equations within a given model. The DAE solver eliminates the need to carry out the laborious task of modifying the code when changing the set of decision variables. Thereafter *gPROMS* uses a numerical ordinary differential equation (ODE) solver to simulate the process of the given model over a specified amount of time (*or any other space*). *gOPT* is a complimentary dynamic optimization front-end to *gPROMS*. Within the *gOPT* environment, the user specifies the objective function, constraints, decision variables and time horizon (*which may also be specified as a decision variable*). Thereafter *gOPT* successively searches for an optimal set of design parameters by simulating the model in *gPROMS* and obtaining the objective function value for each iteration step within the optimization procedure.

Although not used in this study, *gOPT* may be used to locate the best input trajectory for a process in order to minimize an objective function. The *gPROMS*/*gOPT* route was chosen due to it's powerful dynamic optimization capabilities. Another possible route would be to implement collocation, eliminating the differential states, and then any optimization package could be used (Cuthrell and Biegler, 1989; Barton *et al.*, 1998).

gOPT does not handle nested optimization routines directly as are present in the problem formulation. To this end, an outer loop is implemented to break the nested routines. This solution path is shown in Figure 4.15. For the purposes of this explanation, the decision variables are split into integer decision variables, d and continuous decision variables, z. The integer variables are dealt with individually, i.e. each integer set of variables ($d_i \in D$) is used to obtain a solution, and then the resulting objective functions compared. This method is chosen rather than taking a mixed integer approach due to the small number of integer sets in the problem. The block labelled **1** in the diagram represents the second part of the primary objective, where the objective function is the absolute deviation

of the chosen product variable and is maximised over the uncertainty and disturbance space. The resulting set of disturbances and uncertain parameters are then simultaneously fed into blocks **2** and **3**. In block **2**, the flexibility constraint is considered, where the vector of inequality constraints is maximized over the disturbance and uncertainty space. In addition, the maximum likely constraint violation is also sought over the indexed set of inequalities $k \in K$. For clarity, the uncertain parameters, disturbance variables, and set of inequality constraints are labelled with an addition subscript g. In block three the secondary part of the primary objective is considered, where the design variables form the search space, and the objective function is minimized subject to the constraints formed in block **2**; with the vectors of uncertainty and disturbance variables in the objective function, taken from those computed in block *1*, thus forming the expected value of the objective function. In block **4**, the set of continuous design variables z is compared against those in the last iteration loop; if they are the same, an integer solution has been found on D, and the the routine moves on to the next d_i; otherwise the inner loop iteration is carried out again. Finally once all integer solutions are found, these integer solutions are compared arriving at the final optimal design.

This routine does not necessarily guarantee that any or all integer solutions will be found. To this end a maximum number of iterations (*in this case 100*) is imposed, and if a solution is not found, the flexibility constraints are discretized over the spaces Γ and $V(t)$ in block **2**, as are the similar spaces in block **1**. This method may be used to generate an initial guess which will be closer to the final solution, the higher the order of the discretation. Unfortunately, although this is a brute force method guaranteeing solution, it is very computationally intensive as the order of discretation is increased.

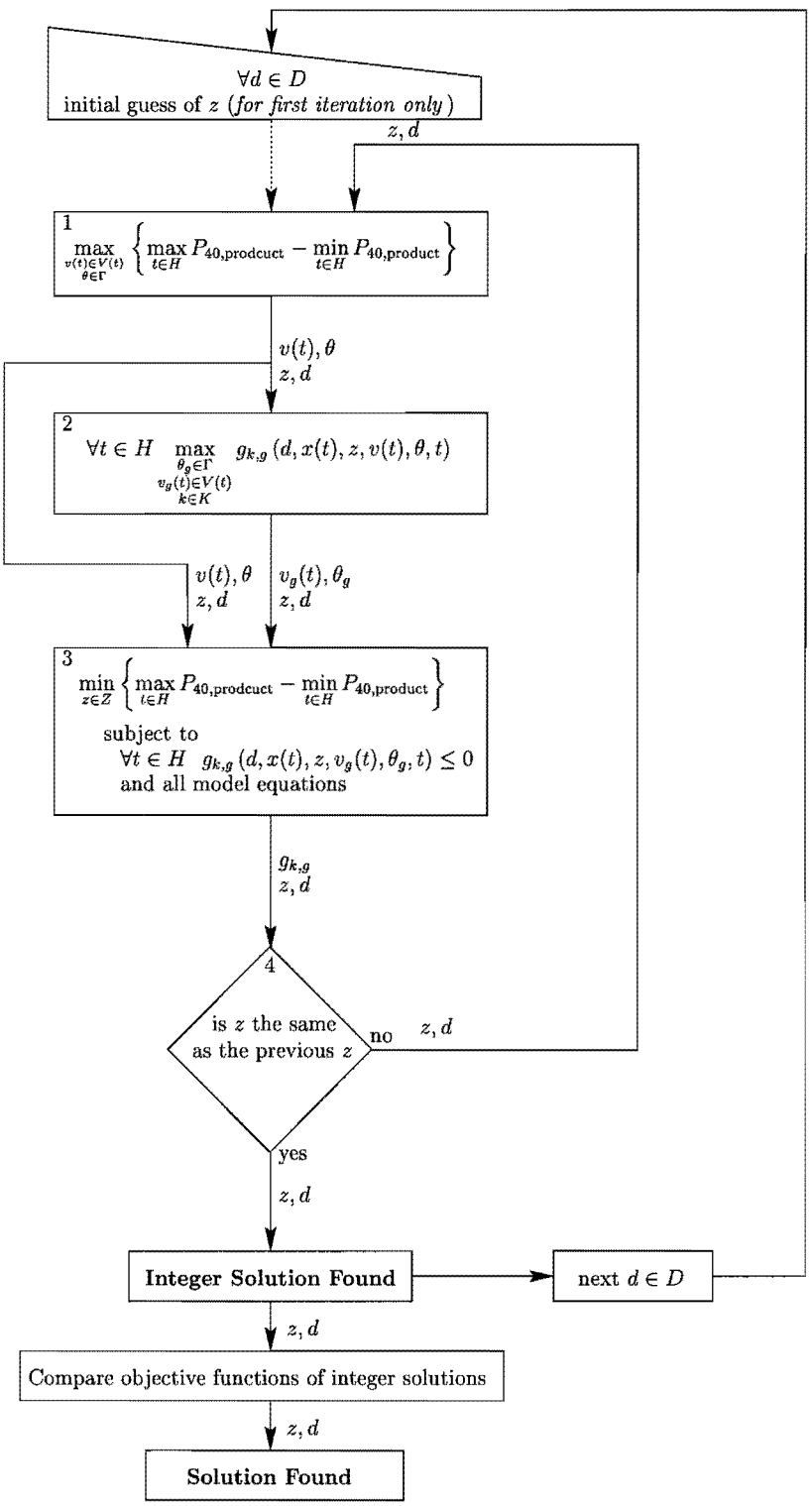

Figure 4.15: Routine to eliminate nested optimization routines

For simultaneous consideration of the discrete set of disturbance variables, a vector of models is formed, where each model in the vector describes the entire circuit with one single discrete disturbance. This vector is further increased for purposes of the flexibility constraint set, where additional model elements are included in the vector to ensure constraint satisfaction. The design variables are then made common to all models, and the objective function is comprised as a summation of discrete objectives over selected models (*i.e. those which comprise the objective function*).

In this case a single model comprises 69 differential states, 478 algebraic equations, and 547 unknown variables. The complete vector of models is comprised of 1104 differential states, 7847 algebraic equations, with 8951 model equations in total. On average each outer loop shown in Figure 4.15 took approximately 5-6 hours processor time on a Linux based dual PIII-650 processor system with 640MB RAM.

4.3.1 *gPROMS* Implementation

Path Constraint

Path constraints are not handled directly in the *gPROMS* optimization specification; only end-point inequalities and/or end-point equalities. Thus path constraints are entered into the problem formulation as an end-point equality. For example, the path constraint governing the sump level is bounded below by $h_{\min}$ below and above by $h_{\max}$, and must be satisfied for all time on the horizon $t = [0, H]$:

$$\forall t \in H \quad h_{\min} \leq h_{\text{sump}}(t) \leq h_{\max} \tag{4.8}$$

A dummy state variable h_{sumpcon} is introduced and set to zero for $t = 0$. This variable is now set as an end-point constraint in the optimization specification such that it must equal zero at the final time. h_{sumpcon} is defined in equation (4.9), leading to equation (4.10).

$$h_{\text{sumpcon}} = \int_0^H \max\left(0, h_{\min} - h\left(t\right), h\left(t\right) - h_{\max}\right) \mathrm{d}t \tag{4.9}$$

$$\Longrightarrow$$

$$\frac{\mathrm{d}h_{\text{sumpcon}}}{\mathrm{d}t} = \max\left(0, h_{\min} - h\left(t\right), h\left(t\right) - h_{\max}\right) \tag{4.10}$$

Integral expressions are handled in *gPROMS* through the use of their corresponding differential, which in this case is shown in equation (4.10).

Figure 4.16, shows the significance of equation (4.9), where the integral is shown as the shaded area in the Figure. Equation (4.9) calculates the sum of the shaded areas in Figure 4.16, (*where the shaded areas show path constraint violation.*)

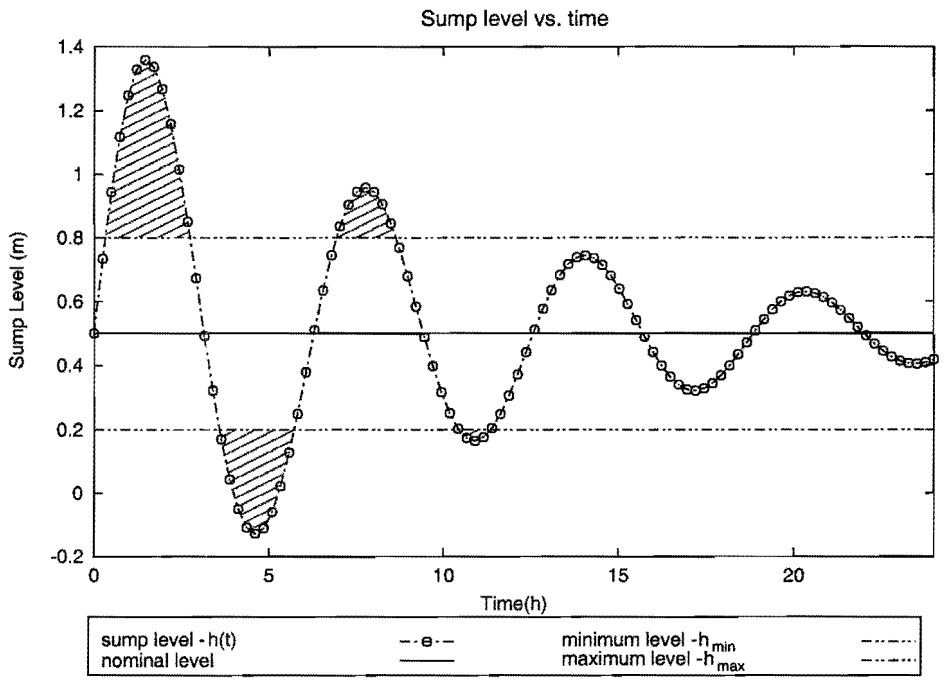

Figure 4.16: Path constraint implementation

The inclusion of the path constraint in the problem simply requires that the value of h_{sumpcon} is equal to zero at $t = H$, in order for the path constraint not to be violated.

Case Statements

Similarly to path constraints, case statements cannot be used in the *gPROMS* environment for optimization routines, thus one is unable to use any case and/or if statements within the specified model. This subsection serves to present a simple way of converting case statements to continuous functions that may easily be specified within *gPROMS*.

Figure 4.17 shows a hyperbolic tan function, **htan**, which is used to replace case statements. The variable β is used to compress the **htan** function horizontally (*where* $y = \text{htan}(\beta x)$) shown in Figure 4.17.

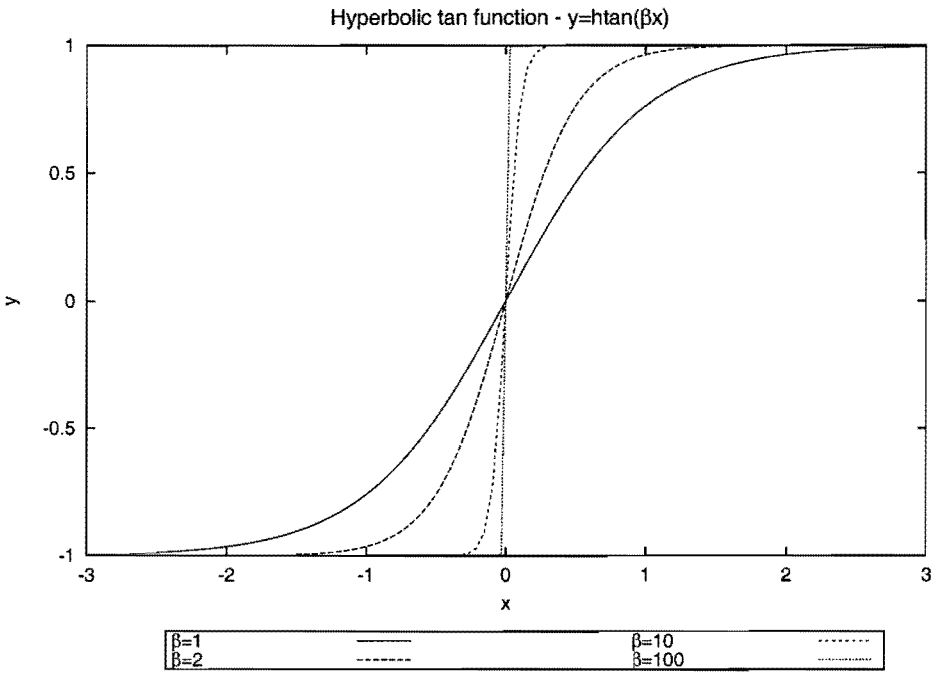

Figure 4.17: Hyperbolic tan functions

In a similar fashion the hyperbolic function may be scaled vertically to lie between 0 and 1, via equation (4.11), thereby providing a *case* function when $\beta \gg 1$.

$$y = \frac{1 + \text{htan}(\beta x)}{2} \tag{4.11}$$

In the *gPROMS* environment, it is recommended to use a value of βx no greater than 100 to prevent the accuracy required in the **htan** evaluation from decreasing below machine precision. In order to ensure that the magnitude of the argument of **htan** is never greater than 100, a max/min construct is used:

$$y = \frac{1 + \text{htan}\left(min\left(100, max\left(-100, (\beta x)\right)\right)\right)}{2} \tag{4.12}$$

Equation (4.12) is an approximation of a continuous step function from $y = 0$ to

$y = 1$ at $t = 0$, shown also by the construct:

$$y(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x \geq 0 \end{cases} \tag{4.13}$$

The inclusion of Equation (3.22) in the cyclone model used had to be implemented in the *gPROMS* code as such a case system. Equation (3.22) is repeated below for convenience:

$$\begin{aligned} &\text{if} && WF > 21.4 \\ & && WOF = 1.363WF - 10.75 \\ &\text{else} && \\ & && WOF = 0.837WF + 0.35 \end{aligned} \tag{4.14}$$

The resulting case statement is now written in respect of the hyperbolic tan function from equation (4.11) as follows:

$$\begin{aligned} WOF = &\frac{1 + \text{htan}\left(10^3\left(WF - 21.4\right)\right)}{2}\left(1.36WF - 10.75\right) + \\ &\frac{1 + \text{htan}\left(10^3\left(21.4 - WF\right)\right)}{2}\left(0.837WF + 0.35\right) \end{aligned} \tag{4.15}$$

In order to ensure that the magnitude of the argument of the htan expression never exceeds 100, equation (4.15) is now extended in the same way as equation (4.12):

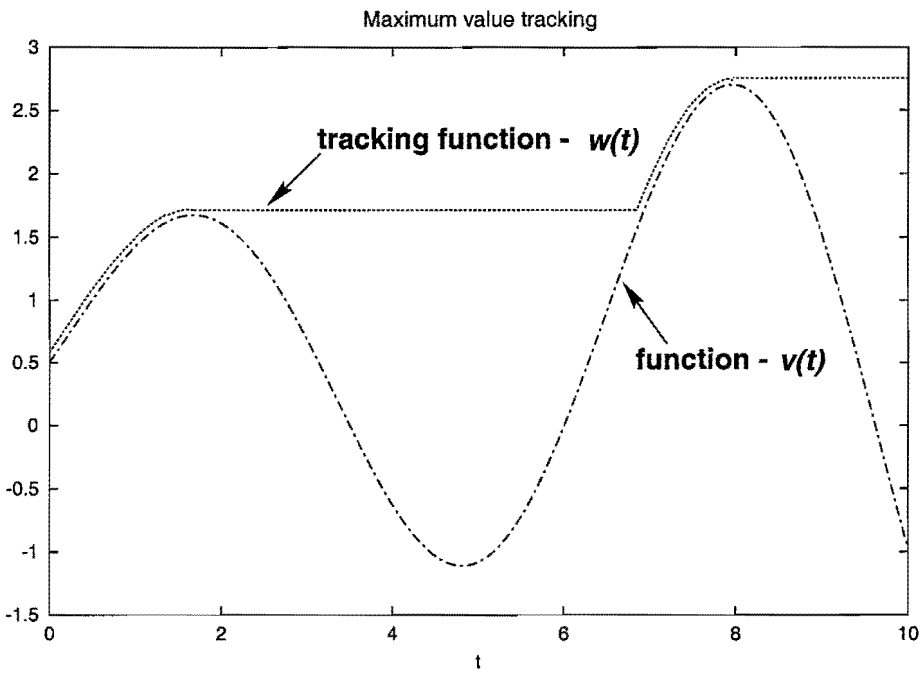

Figure 4.18: Maximum Tracking Function

$$WOF = \frac{1 + \text{htan}\left(min\left(100, max\left(-100, \left(10^3\left(WF - 21.4\right)\right)\right)\right)\right)}{2}\left(1.36WF - 10.75\right) + \frac{1 + \text{htan}\left(min\left(100, max\left(-100, \left(10^3\left(21.4 - WF\right)\right)\right)\right)\right)}{2}\left(0.837WF + 0.35\right) \tag{4.16}$$

Max/Min value tracking

gPROMS does not handle direct arguments for determining the maximum or minimum value of a trajectory over time, in addition non-continuous functions are not allowed within any model that is to be used in the *gOPT* environment. To this end a dummy function is formed to track the maximum or minimum value of a trajectory over time.

Figure 4.18 shows a typical function, $v(t)$ that is to be tracked using a tracking function, $w(t)$. This is achieved by specifying that the differential with respect

to time of $w(t)$ is equal the differential of $v(t)$ when $v(t)$ is increasing and $v(t) \geq w(t)$; conversely if $v(t)$ is decreasing or $v(t) < w(t)$, then the differential of w must equal to zero. This is clear in the mathematical description shown in equation (4.17).

$$\frac{\mathrm{d}w}{\mathrm{d}t} = \begin{cases} \frac{\mathrm{d}v}{\mathrm{d}t} & \frac{\mathrm{d}v}{\mathrm{d}t} \geq 0 \text{ and } v(t) \geq w(t) \\ 0 & \frac{\mathrm{d}v}{\mathrm{d}t} < 0 \text{ or } v(t) < w(t) \end{cases} \tag{4.17}$$

The continuous case function described previously (*see equation (4.11)*) is now used to handle the discontinuity in equation (4.17):

$$\frac{\mathrm{d}w}{\mathrm{d}t} = \frac{1 + \mathrm{htan}\left(\beta\left(v(t) - w(t)\right)\right)}{2} \max\left(0, \frac{\mathrm{d}v}{\mathrm{d}t}\right) \tag{4.18}$$

Equation (4.18) requires that the initial values ($t = 0$) of $w(t)$ and $v(t)$ are the same. Since the simulations within the optimization routines are started with all state differentials equal to zero, this further constraint may be added to equation (4.18) by including the difference between $v(t)$ and $w(t)$ in the max function, ensuring that the initial values of $w(t)$ and $v(t)$ will be the same as shown below:

$$\frac{\mathrm{d}w}{\mathrm{d}t} = \frac{1 + \mathrm{htan}\left(\beta\left(v(t) - w(t)\right)\right)}{2} \max\left(0, \frac{\mathrm{d}v}{\mathrm{d}t}, \left(v(t) - w(t)\right)\right) \tag{4.19}$$

$$\text{at } t = 0 \qquad \frac{\mathrm{d}w}{\mathrm{d}t} = 0 \tag{4.20}$$

A new variable $y(t)$ is now defined as the difference between $v(t)$ and $w(t)$, $y(t)$ is set equal to zero at $t = 0$, which ensures that $w(0) \equiv v(0)$.

$$y(t) = v(t) - w(t) \tag{4.21}$$

$$y(0) = 0 \tag{4.22}$$

A similar approach could also be taken for the consideration of time-varying constraints.

Multiple Disturbances

The inclusion of multiple simultaneous disturbances in the problem specification is implemented in *gPROMS* by simulating and optimizing parallel models simultaneously, each with a different disturbance description. The parallel models are formed in an array, where the objective function and path constraint equations are formed by a summation across the array of the individual components.

Chapter 5

Results and Discussion

This chapter serves to present results obtained from the operability analysis and design discussed in the previous chapter. The chapter begins with a comparative review of the results obtained, followed by a section detailing further simulations addressing the plant spillage addition problem.

5.1 Optimization Results

A total of five optimization routines were carried out, one for each of the five sets of integer variables, where different multi-loop controllers were considered. Table 5.1 shows the details of each integer set examined. In all cases a PI controller, controlling the sump level by manipulating the flowrate out of the sump was used. A solution was found using the method shown in Figure 4.15, within a maximum of 21 outer loop iteration steps, and there was thus no need to discretize the flexibility and expected value space.

Table 5.1: Single Loop Controllers

Scenario	Manipulated Variable	Controlled Variable
1	Sump Only Control	
2	Water addition at position 1	Volumetric rate of Cyclone Overflow
3	Water addition at position 2	Volumetric rate of Cyclone Overflow
4	Water addition at position 1	Volumetric rate of Cyclone Feed
5	Water addition at position 2	Volumetric rate of Cyclone Feed

The objective function minimized in the optimization routine, namely the absolute deviation (max - min) of the cyclone overflow particle size gives a measure of the success of each of the designs listed in Table 5.1. Table 5.2 reports the final objective value obtained for each scenario.

Table 5.2: Final Objective function Values

Scenario No.	Objective Function Value
1	6.58 E-02
2	5.57 E-02
3	**4.98 E-02**
4	5.29 E-02
5	6.23 E-02

It is interesting to note that all of the addition controllers investigated improved the performance of the process in terms of reducing the objective function. In particular, scenario **3** (*water addition at position 2* $\longrightarrow$ *volumetric rate of cyclone overflow*) resulted in the lowest objective function of all multi-loop controllers compared, suggesting that this controller pair is the optimal.

The results shown in Table 5.2 do not include the effect of plant spillage addition; it was shown in section 4.2.4 that the addition of plant spillage is considered a one-time infrequent disturbance, and that the optimal position for spillage addition is to be found based on the optimal design obtained through the optimization process. To this end, further scenarios were carried out where for each of the designs shown in Table 5.1, spillage addition was simulated at the three possible positions individually. Although the choice of design is to be based on the optimization results, these scenarios were carried out for all the designs to investigate any trend in terms of the spillage addition position in order to draw more general conclusions. The following section proceeds to present the results obtained from these scenarios.

5.2 Spillage Addition

As mentioned previously this section serves to present the results of scenarios carried out on the five different designs discussed in section 5.1. The spillage addition was modeled as a pulse function of 50 l/h water being added at the three positions, where the pulse has a duration of 3 hours, and the simulations were carried out for 7 hours of simulated time. For each scenario, the comparison between different spillage addition positions is compared in Figures 5.1 to 5.5.

It can easily be seen in all scenarios that the optimal position for spillage addition is position 3. Contrary to the preliminary findings shown in the scenario studies section, where it was noted that the optimal spillage position appeared to be either at positions 1 or 2, it is clear that the optimal design has different characteristics. Thus no generalization can be made as to the optimal spillage addition position until the process has been fully analyzed, in other words the optimal position is dependent on the design. It can also be seen on Figures 5.1 to 5.5 that spillage addition positions 1 and 2 show very similar characteristics, and this can likely be ascribed to the constant mill volumetric hold-up assumption used.

Figure 5.6 shows a comparison of the different scenarios when subjected to the same spillage disturbance at position 3. It is evident in this response that the additional single loop PI controllers have a detrimental effect on the response to the spillage addition. This highlights the importance of not including spillage addition as a frequent disturbance in the optimization formulation, as it would appear that the inclusion of this variable would have favoured the use of only sump control and the design could well have been optimized for spillage addition alone.

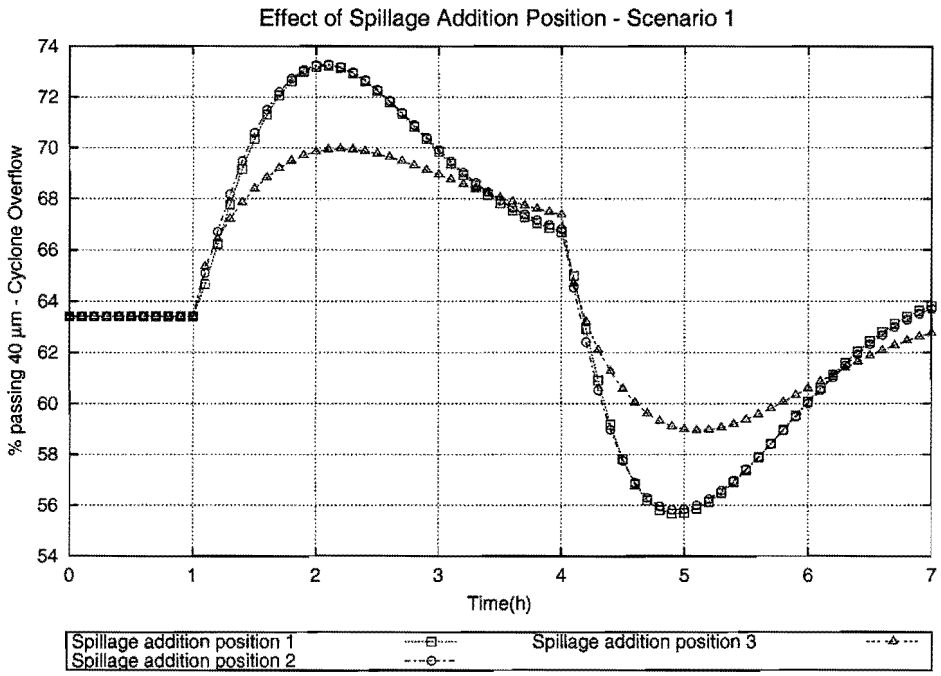

Figure 5.1: Figure showing different spillage addition positions for scenario 1

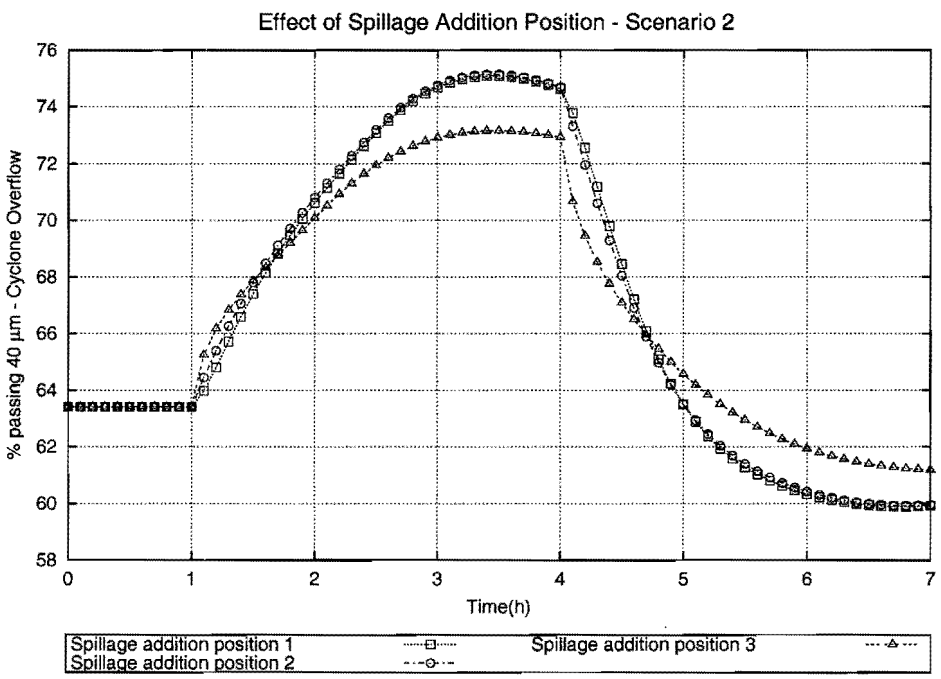

Figure 5.2: Figure showing different spillage addition positions for scenario 2

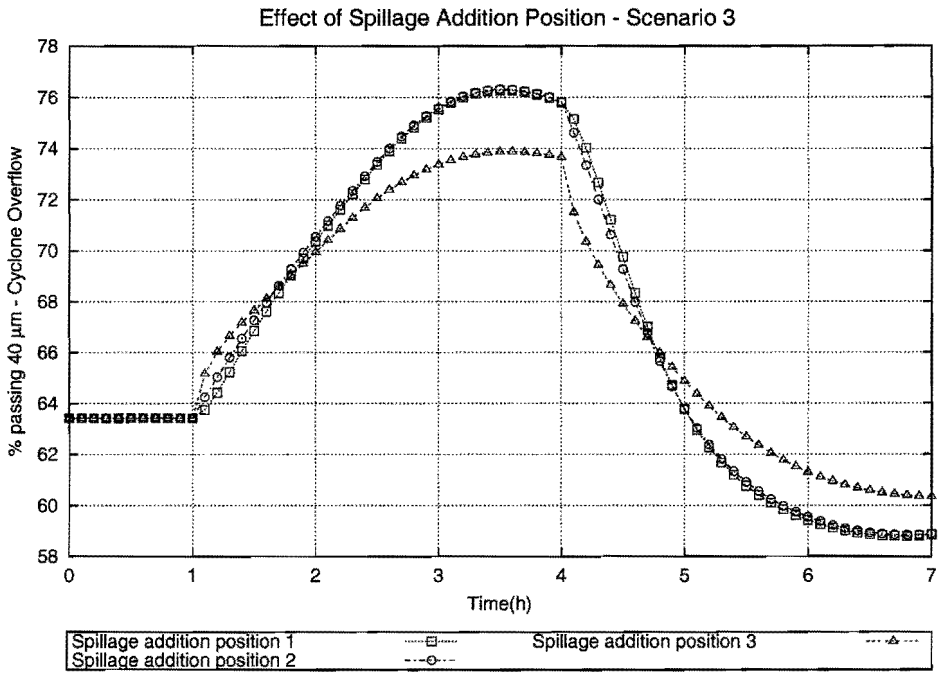

Figure 5.3: Figure showing different spillage addition positions for scenario 3

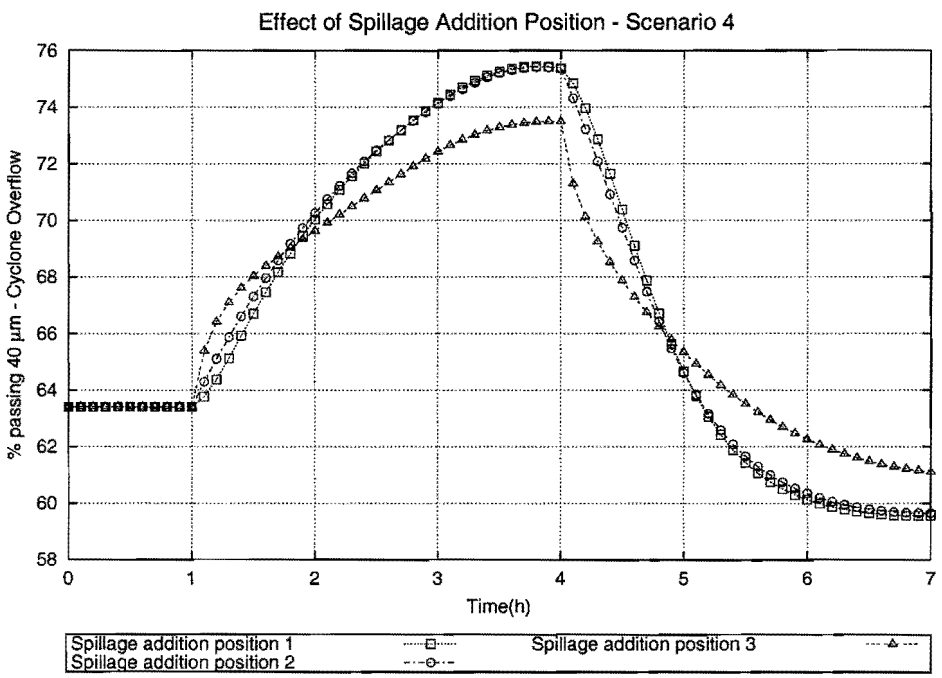

Figure 5.4: Figure showing different spillage addition positions for scenario 4

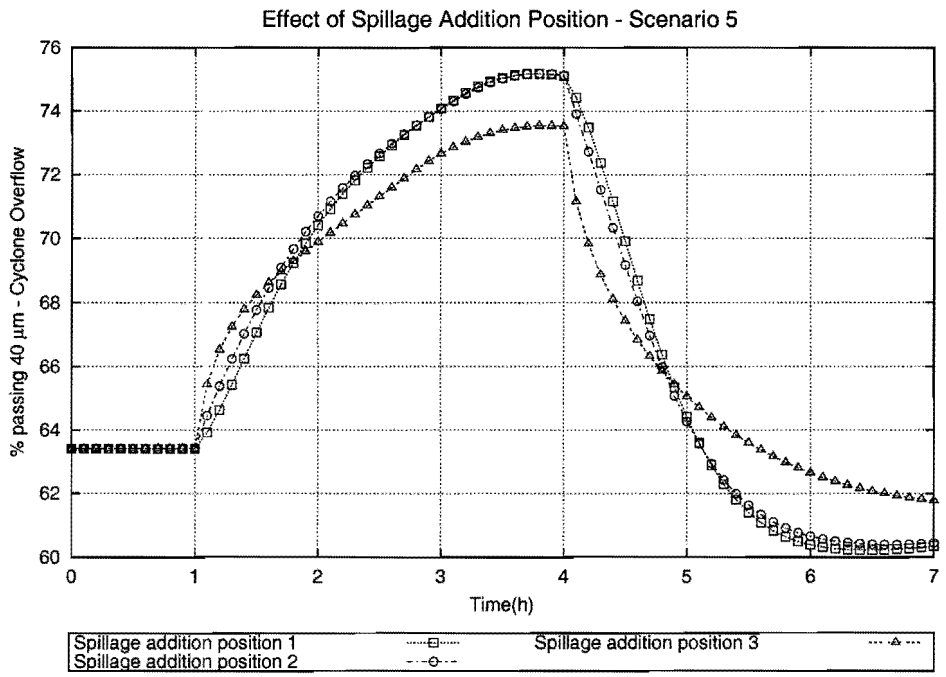

Figure 5.5: Figure showing different spillage addition positions for scenario 5

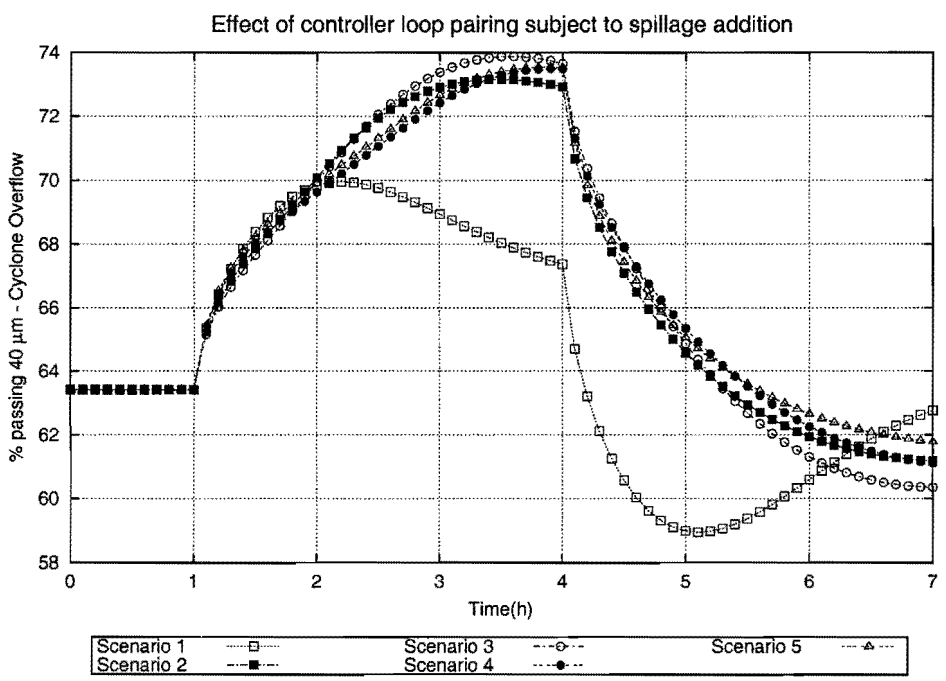

Figure 5.6: Figure showing effect of spillage added at position 3 for the different scenarios

5.3 Analysis of Optimal Solution

Table 5.3 † shows the values of the design variables found at the optimal solution, together with their respective units, previous design values and specified search space.

Of interest, the sump area is found to be at the maximum value allowed in the optimization, further validating the findings in the scenario studies where it was also noted that an effective increase in sump volume showed improved disturbance rejection.

Another point of interest shown in Table 5.3 is the mill loading. It was found during the scenario studies that an increase in mill loading appeared to decrease product variation. While this is true, it must also be noted that this correlation derived in the scenario studies does not show any global optimum, and in fact it was found that a lower mill load with the combination of other parameter changes produces least product variation. This emphasises the need for such a study that considers all parameters and design variables simultaneously within a single framework, rather than making ad-hoc decisions based on observation of single variables.

Table 5.3: Optimal Design Parameters

Variable		Optimal	Previous	Lower/Upper Bound	
Sump Area	(m^3)	3.00 e – 01	2.25 e – 01	1.50 e – 01	3.00 e – 01
Water rate 1	(l/h)	3.30 e +01	3.50 e +01	3.30 e +01	3.70 e +01
Water rate 2	(l/h)	1.40 e +01	1.20 e +01	1.00 e +01	1.40 e +01
Mill load	(m^3)	3.28 e – 02	5.00 e – 02	3.00 e – 02	1.00 e – 01
Sump gain	(m^2/h)	-1.49 e – 01	-	$-\infty$	∞
Sump reset time	(h)	2.91 e – 01	-	0.00 e +00	∞
Sump set-point	(m)	5.85 e – 01	5.00 e – 01	4.00 e – 01	6.00 e – 01
PI gain	$(-)$	-1.24 e +02	-	$-\infty$	∞
PI reset time	(h)	3.82 e – 01	-	0.00 e +00	∞

The disturbance variables resulting in the maximum (*worst*) value of the objective function are shown in Table 5.4. The disturbance creating the maximum

†The upper and lower bounds of ∞ and -∞, respectively, imply no upper or lower bounds specified. In the *gPROMS* environment they are specified as +1e20 and -1e20 respectively.

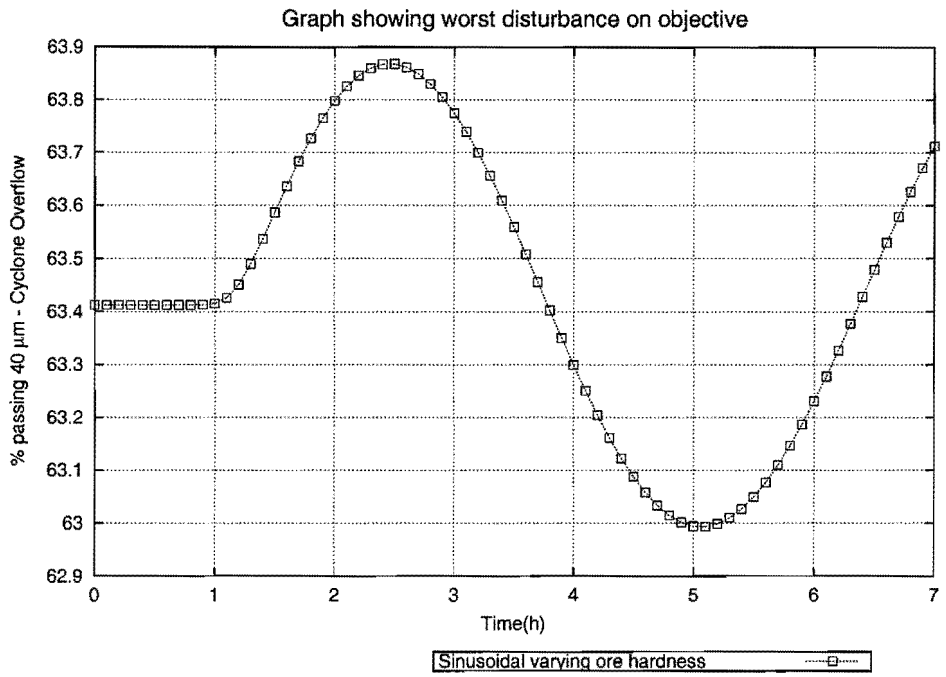

Figure 5.7: Figure showing the largest effect of a disturbance on the objective

effect on the product size is that of the sinusoidally varying ore hardness. A graph showing the effect of this disturbance is shown in Figure 5.7.

In all cases the biggest effect on product by disturbances of the sinusoidal shape were found to be those of low frequency; similarly the step function lengths were all at the maximum allowed of three hours.

Table 5.4: Disturbance Variables Constituting the Expected Objective Function

ROM mass feed rate			
Variable	**Optimal**	**Lower/Upper Bound**	
step amplitude (kg/h)	-1.00 e +01	-1.00 e +01	1.00 e +01
step length (h)	3.00 e +00	0.00 e +00	3.00 e +00
sine amplitude (kg/h)	1.00 e +01	0.00 e +00	1.00 e +01
sine frequency (h^{-1})	1.67 e − 01	1.67 e − 01	1.00 e +00
ROM feed density			
Variable	**Optimal**	**Lower/Upper Bound**	
step amplitude (kg/m^3)	3.00 e +02	-3.00 e +02	3.00 e +02
step length (h)	3.00 e +00	0.00 e +00	3.00 e +00
sine amplitude (kg/m^3)	3.00 e +02	0.00 e +00	3.00 e +03
sine frequency (h^{-1})	1.67 e − 01	1.67 e − 01	1.00 e +00
ROM Ore Hardness			
Variable	**Optimal**	**Lower/Upper Bound**	
step amplitude (%)	1.00 e +01	-1.00 e +01	1.00 e +01
step length (h)	3.00 e +00	0.00 e +00	3.00 e +00
sine amplitude (%)	1.00 e +01	0.00 e +00	1.00 e +01
sine frequency (h^{-1})	1.82 e − 01	1.67 e − 01	1.00 e +00
ROM feed size distribution			
Variable	**Optimal**	**Lower/Upper Bound**	
step amplitude $(\%P_{40})$	-1.00 e +01	-1.00 e +01	1.00 e +01
step length (h)	3.00 e +00	0.00 e +00	3.00 e +00
sine amplitude $(\%P_{40})$	1.00 e +01	0.00 e +00	1.00 e +01
sine frequency (h^{-1})	1.67 e − 01	1.67 e − 01	1.00 e +00

The constraints evaluated at the values of disturbance variables and uncertain parameters shown in Table 5.5 result in the worst case set (*i.e. the closest possible to violation*). The entry highlighted by bold font (*the sinusoidal mass feed rate disturbance*) shows the disturbance that results in constraints being closest to their limits.

It was found that the uncertain parameters for each disturbance in the flexibility analysis area of the optimization process were all of the same value (*see Table 5.5, where only one set has been reported*). This suggests that these uncertain parameters are fairly independent of the disturbance variables, or certainly that the cumulative effects are similar.

It can also be seen from Table 5.5 that the disturbance variable values producing worst case constraint considerations are very similar to those having the worst effect on the product stream. This highlights a strong correlation between the constraint variables and those constituting the product stream in this particular study.

Figure 5.8 shows the response of the sump level to the sinusoidal mass feed rate with a corresponding line marking the maximum allowable sump level. It can be seen in this Figure that the constraints are still satisfied for the worst case disturbance and parameter uncertainty as is guaranteed in the problem formulation. In Figure 5.8, the maximum sump level is not violated (*simply showing that the formulation does indeed guarantee this*) and that this worst case constraint approaches the maximum allowable limit. This suggests that the optimum sump level set-point in conjunction with the controller tuning parameters found, appear to make use of the maximum available sump volume. In addition, the optimal sump cross sectional area found is the maximum area specified, further strengthening the hypothesis that a larger sump volume improves disturbance rejection of the circuit.

Table 5.5: Disturbance and Uncertainty Constituting the Worst Case Constraint Set

ROM mass feed rate				
Variable		**Optimal**	**Lower/Upper Bound**	
step amplitude	(kg/h)	-1.00 e +01	-1.00 e +01	1.00 e +01
step length	(h)	3.00 e +00	0.00 e +00	3.00 e +00
sine amplitude	(kg/h)	1.00 e +01	0.00 e +00	1.00 e +01
sine frequency	(h^{-1})	1.67 e – 01	1.67 e – 01	1.00 e +00
ROM feed density				
Variable		**Optimal**	**Lower/Upper Bound**	
step amplitude	(kg/m^3)	3.00 e +02	-3.00 e +02	3.00 e +02
step length	(h)	3.00 e +00	0.00 e +00	3.00 e +00
sine amplitude	(kg/m^3)	3.00 e +02	0.00 e +00	3.00 e +03
sine frequency	(h^{-1})	1.67 e – 01	1.67 e – 01	1.00 e +00
ROM Ore Hardness				
Variable		**Optimal**	**Lower/Upper Bound**	
step amplitude	(%)	1.00 e +01	-1.00 e +01	1.00 e +01
step length	(h)	3.00 e +00	0.00 e +00	3.00 e +00
sine amplitude	(%)	1.00 e +01	0.00 e +00	1.00 e +01
sine frequency	(h^{-1})	1.67 e – 01	1.67 e – 01	1.00 e +00
ROM feed size distribution				
Variable		**Optimal**	**Lower/Upper Bound**	
step amplitude	$(\%P_{40})$	-1.00 e +01	-1.00 e +01	1.00 e +01
step length	(h)	3.00 e +00	0.00 e +00	3.00 e +00
sine amplitude	$(\%P_{40})$	1.00 e +01	0.00 e +00	1.00 e +01
sine frequency	(h^{-1})	1.67 e – 01	1.67 e – 01	1.00 e +00
Uncertain Parameters				
Variable		**Optimal**	**Lower/Upper Bound**	
d_{50} Cyclone	(%)	-1.00 e +01	-1.00 e +01	1.00 e +01
Mill Power	(kW)	5.00 e +01	5.00 e +01	6.00 e +01

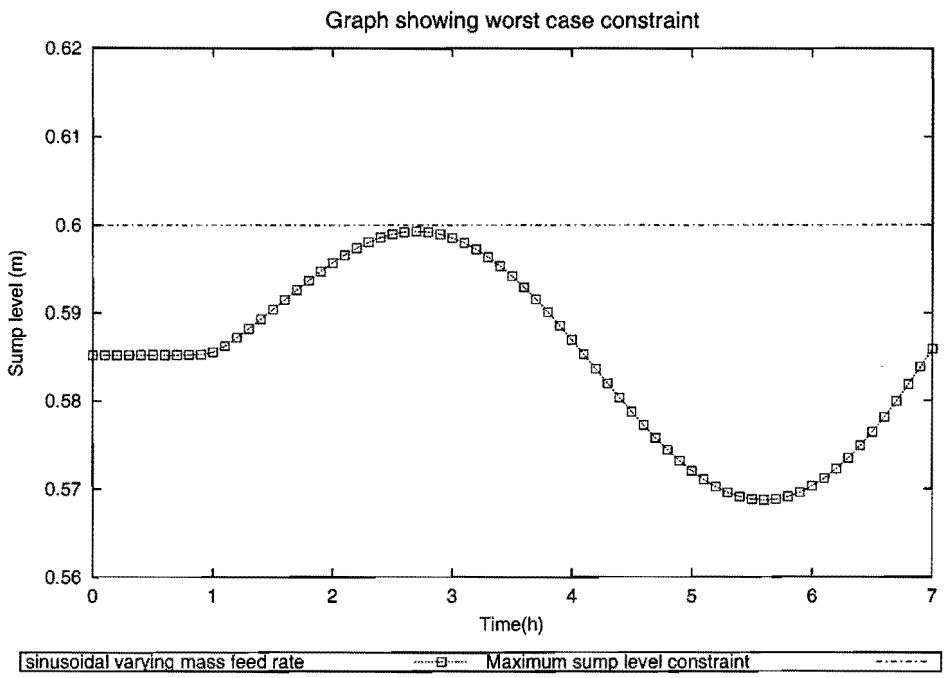

Figure 5.8: Figure showing the worst constraint

Chapter 6

Conclusions

A framework for optimal design that incorporates dynamic flexibility constraints has been formulated. Within this framework it has been shown that a full design of a process can be carried out as well as a retro-fit design. In terms of the design of the process, it was also shown that the objective of the design can easily be changed within the framework, and can be extended to include various measures of economic performance, including for example the cost of additional controller loops, measuring devices and so forth.

The framework presented here, is a general framework that can be extended to other applications both within and out of the minerals processing field. The formulation can also be extended to more complex process models and systems. The inclusion of binary decision variables could for example be useful in designing flotation circuit configurations.

A novel methodology for solving the proposed framework was developed and used (*see Figure 4.15*) where a manual iteration step had to be introduced in order to solve the nested optimization routine included for the purposes of incorporating the dynamic flexibility constraints.

This framework was applied to a comminution circuit model, with the operational objectives of the process derived from the primary comminution circuit of the

UG2 plant at Impala Platinum Limited, Rustenburg. The operational objectives of the plant are to: *"Determine a design that minimizes the variation in the cyclone overflow product size due to spillage water addition and variation in feed characteristics subject to the design and operational constraints being satisfied."* A number of design features were included in the framework for manipulation in obtaining an optimal design. They included the sump volume and corresponding controller tuning (*including the nominal level*), nominal water addition around the circuit, mill loading and controller tuning of additional control loops. A set of five binary variables were included in the optimization problem, representing the different possible multi-loop PI controllers used to control the circuit.

Various disturbances to the process were considered in the form of both sinusoidal and pulse variation. Both types of disturbance included varying frequency and amplitude of these functions. These short-term disturbances included in the problem formulation were variations in feed size distribution, ore hardness, mass feed rate and ore density. Parametric uncertainty relating to possible plant/model mismatch was also included.

The sump-level of the circuit was constrained in the formulation so as not to overflow or run dry despite any or all of the disturbances and parametric uncertainty considered.

As a post-design exercise, the optimal position for spillage addition to the circuit was found by conducting simulations of these optimal designs.

In all cases adding the plant spillage directly to the feed of the cyclone produced the least variation in cyclone product particle size.

The design with only a single controller, maintaining a sufficient sump level, produced the least variation in face of the spillage addition. The plant spillage was considered to be an infrequent disturbance, and as such these results are only used to make a decision based on where to return plant spillage.

The importance of the interaction between design and control has been highlighted in the case study presented here, where the optimal spillage addition

position found by running simulations on the nominal design was different to that found when optimal designs were simulated, highlighting the importance of including operability in the design stage, rather than basing dynamic decisions on isolated simulations.

Another example of the interaction between design and control presented here is the effect of mill loading on the dynamic performance of the circuit. In the scenario studies it was found that increasing the mill loading appeared to improve the performance of the circuit. However, the optimal mill loading found in the operability design was in fact a much lower mill loading. This further emphasises the importance of using a dynamic model of a process to carry out the design within an operability framework, rather than making general design decisions based on secluded scenarios which do not show the interaction between design and control.

The optimal controller configuration of those investigated was to control the volumetric flowrate of the cyclone feed stream by manipulating the water added into the sump. Through scenarios studies, it was found that the choice of using the mass flowrate of the cyclone feed or product as a controlled variable is detrimental to maintaining a consistent grind in the product stream.

It was shown that excluding one-time infrequent disturbances to the process, such as the plant spillage addition, is important when formulating such problems. In the case of plant spillage addition it was found that the design with only a single PI controller showed least variation in product size. However, the design should not sacrifice performance to frequent disturbances based on the effects caused by an infrequent one.

From the study carried out, it is recommended that the formulation be extended to handle binary variables in a mixed integer environment, and that an optimization routine be formulated to incorporate nested optimization routines in an automated manner, thus eliminating the need for a manual iteration procedure used here, or the discretation of variables as other authors have made use of.

References

Bahri, P., Bandoni, J., and Romagnoli, J. (1997). Integrated Flexibility and Controllability Analysis in Design of Chemical Processes. *AIChE Journal*, 43:997–1015.

Barton, G., Chan, W., and Perkins, J. (1991). Interaction Between Process Design and Process Control: The Role of Open-Loop Indicators. *Journal of Process Control*, 1:161–170.

Barton, P. I., Allgor, R. J., Feehery, W. F., and Galan, S. (1998). Dynamic Optimization in a Discontinuous World. *Industrial and Engineering Chemistry Research*, 37:966–981.

Boyd, S., Barratt, C., and Norman, S. (1990). Linear Controller Design: Limits of Achievable Performance Via Convex Optimization. In *Proceedings of the IEEE, 78(3)*, pages 529–564.

Cuthrell, J. E. and Biegler, L. T. (1989). Simultaneous optimization and solution methods for batch reactor control profiles. *Computers and Chemical Engineering*, 13(1/2):49–62.

Desbiens, A., Pomerleau, A., and Najim, K. (1994). Adaptive Predictive Control of a Grinding Circuit. *International Journal of Mineral Processing*, 41:17–31.

Dimitriadis, V. and Pistikopoulus, E. (1995). Flexibility Analysis of Dynamic Systems. *Industrial and Engineering Chemistry Research*, 34:4451–4462.

Grossmann, I. and Floudas, C. (1987). Active Constraint Strategy for Flexibility Analysis in Chemical Processes. *Computers and Chemical Engineering*, 11(6).

Grossmann, I., Halemane, K., and Swaney, R. (1983). Optimization Strategies for Flexible Chemical Processes. *Computers and Chemical Engineering*, 7(4).

Grossmann, I. and Morari, M. (1984). Operability, Resiliency and Flexibility - Process Design Objectives for a Changing World. *Proceedings of the Second International Conference on Foundations of Computer Aided Process Design, A.W Westerberg and H.H. Chien, eds.*

Halemane, K. and Grossmann, I. (1981). A Remark on the Paper, Theoretical and Computational Aspects of the Optimal Design Centering, Tolerance and Tuning Problems. *IEEE Transactions on Circuits and Systems*, CAS-28(2).

Halemane, K. and Grossmann, I. (1983). Optimal Process Design under Uncertainty. *AIChE Journal*, 29(3):425–433.

Hales, L., Vanderbreek, J., and Herbst, J. (1988). Supervisory Control of a Semi-Autogenous Grinding Circuit. *International Journal of Mineral Processing*, 22:297–312.

Herbst, J., Pate, W., and Oblad, A. (1992). Model-Based Control of Mineral Processing Operations. *Powder Technology*, 69:21–32.

Hulbert, D., Craig, I., Coetzee, M., and Tudor, D. (1990). Multivariable Control of a Run-of-Mine Milling Circuit. *Journal of the South African Institute of Minerals and Metallurgy*, 7:173–181.

Hulbert, D. and Woodburn, E. (1983). Multivariable Control of a Wet Grinding Circuit. *AIChE Journal*, 29:186–191.

Lynch, A. and Rao, T. (1975). Modelling and Scale Up of Hydrocyclone Classifiers. *Proceedings of the 11th International Mineral Processing Congress*, pages 245–269.

Manlapig, E. (1977). *The Dynamic Behaviour and Automatic Control of a Chalcopyrite Flotation Plant.* PhD thesis, University of Queensland.

Mohideen, M., Perkins, J., and Pistikopoulos, E. (1996a). Optimal Design of Dynamic Systems Under Uncertainty. *AIChE Journal*, 42(8):2251–2272.

Mohideen, M., Perkins, J., and Pistikopoulos, E. (1996b). Optimal Synthesis and Design of Dynamic Systems Under Uncertainty. *Computers and Chemical Engineering*, 20(Supplement):S895–S900.

Morari, M. (1983). Design of Resilient Processing Plants - III. A General Framework for the Assessment of Dynamic Resilience,. *Chemical Engineering Science*, 38(1):1881–1891.

Morrell, S., Napier-Munn, T., and Andersen, J. (1992). The Prediction of Power Draw for Comminution Machines. *Comminution: Theory and Practice, AIME*, pages 405–426,233–248.

Napier-Munn, T., Morrell, S., Morrison, R., and T., K. (1996). *Mineral Comminution Circuits - Their Operation and Optimisation.* Julius Kruttschnitt Mineral Research Centre, Isles Road, Indooroopilly, Queensland 4068, Australia.

Narraway, L. and Perkins, J. (1994). Selection of Process Control Structure Based on Economics. *Computers and Chemical Engineering*, 18(S511).

Polak, E. and Sangiovanni-Vincentelli, A. (1979). Theoretical and Computational Aspects of the Optimal Design Centering, Tolerance and Tuning Problems. *IEEE Transactions on Circuits and Systems*, CAS-26(6).

Process Systems Enterprise Ltd. (1999a). *gPROMS Advanced User Guide.* Bridge Studios, 107a Hammersmith Bridge Road, London W6 9DA, United Kingdom.

Process Systems Enterprise Ltd. (1999b). *gPROMS Introductory User Guide.* Bridge Studios, 107a Hammersmith Bridge Road, London W6 9DA, United Kingdom.

Rajamani, R. and Herbst, J. (1991a). Optimal Control of a Ball Mill Grinding Circuit - I. Grinding Circuit Modelling and Dynamic Simulation. *Chemical Engineering Science*, 46:861–870.

Rajamani, R. and Herbst, J. (1991b). Optimal Control of a Ball Mill Grinding Cricuit - II Feedback and Optimal Control. *Chemical Engineering Science*, 46:871–879.

Ross, R. (1997). *Dynamic Operability Assessment: A Mathematical Approach Based on Q-Parametrization.* PhD thesis, Department of Chemical Engineering, University of Cape Town.

Ross, R., Bansal, V., Perkins, J., and Pistikopoulos, E. (1999). Simulataneous Process Design and Process Control: Application to Complex Separation Systems. *The 7th IEEE Mediterranean Conference on Control and Automation (MED99).*

Ross, R. and Swartz, C. (1995). The Application of Dynamic Operability Assessment to Flotation Circuit Design. In Barker, I., editor, *8th IFAC Interantional Symposium in Mining, Mineral and Metal Processing, Sun City.*

Seaman, D. (1998). Controllability Analysis of Flotation Circuits. BSc Thesis, University of Cape Town.

Seborg, D., Edgar, T., and Mellichamp (1989). *Process Dynamics and Control.* John Wiley and Sons.

Smith, C. (1984). *Dynamic Simulation Development and Application to Sulphide Flotation Circuits.* PhD thesis, University of Queensland.

Swaney, R. and Grossmann, I. (1985). An Index for Operational Flexibility in Chemical Process Design. *AIChE Journal*, 31(4).

Swartz, C. (1997). Operability Analysis and its Application to Minerals Processing Plants. In *SAIMM Western Cape Branch Process Control Workshop, 6 August 1997.*

Vidyasagar, M. (1985). *Control System Synthesis: A Factorisation Approach.* MIT Press, Cambridge, Massachusetts.

Whiten, W. (1974). A Matrix Theory of Comminution Circuits. *Chemical Engineering Science*, 29:588–599.

Appendix A

gPROMS code

variable declaration

```
#=============================================================================================#
DECLARE

  TYPE
     NoType                =    0.5  :    -1E20:    1E20
  STREAM
     MCsSIro               IS             NoType, NoType, NoType, NoType
     Notype                IS             Notype
END
#=============================================================================================#
```

SAG Mill Model

```
#=============================================================================================#
MODEL MILL
#<-------------------------------------------------------------------------------------------->#
     VARIABLE
          M_MillFeed           AS        ARRAY(15)          OF   NoType
          M_MillFeed_T         AS                           OF   NoType
          Q_MillFeed           AS                           OF   NoType
          P40_MillFeed         AS                           OF   NoType
          Cs_MillFeed          AS                           OF   NoType
          xSIfeed              AS        ARRAY(15)          OF   NoType
          roFeed               AS                           OF   NoType

          P40_MillProduct      AS                           OF   NoType
          M_MillProduct_T      AS                           OF   NoType
          M_MillProduct        AS        ARRAY(15)          OF   NoType
          Cs_MillProduct       AS                           OF   NoType
          xSIproduct           AS        ARRAY(15)          OF   NoType
          roProduct            AS                           OF   NoType

          H                    AS        ARRAY(15)          OF   NoType
          Ht                   AS                           OF   NoType
          P                    AS                           OF   NoType
          V_Mill               AS                           OF   NoType
          b                    AS        ARRAY(15,15)       OF   NoType
          SiE                  AS        ARRAY(15)          OF   Notype
          SiE1                 AS        ARRAY(15)          OF   Notype
          SiE2                 AS        ARRAY(15)          OF   Notype
          Si                   AS        ARRAY(15)          OF   NoType


     STREAM
Inlet     :    M_MillFeed, Cs_MillFeed, xSIfeed, roFeed                AS   MCsSIro
Outlet    :    M_MillProduct, Cs_MillProduct, xSIproduct, roProduct    AS   MCsSIro
```

```
#<-------------------------------------------------------------------------------------------->#
     EQUATION

     # ALGEBRAIC EQUATIONS:-

          # Average Specific Energy in the Mill
          SiE                 =    xSIproduct*SiE1 + (1-xSIproduct)*SiE2                    ;

          # Total hold up in mill:
          H(15)               =     V_Mill*Cs_MillProduct - sigma(H(1:14))                  ;

          # Breakage selection
          Si                  =    SiE*P/sigma(H(1:15))                                     ;

          # Calculate volumetric Feed Rate
          Q_MillFeed          =    sigma(M_MillFeed)/Cs_MillFeed                            ;
          M_MillProduct       =    (H/sigma(H(1:15)))*Q_MillFeed*Cs_MillProduct             ;
          M_MillProduct_T     =    sigma(M_MillProduct)                                     ;
          M_MillFeed_T        =    sigma(M_MillFeed)                                        ;
          Ht                  =    sigma(H(1:15))                                           ;
          P40_MillProduct     =    sigma(M_MillProduct(12:15))/sigma(M_MillProduct)         ;
          P40_MillFeed        =    sigma(M_MillFeed(12:15))/sigma(M_MillFeed)               ;

     # DIFFERENTIAL EQUATIONS

     # Hold up in the mill:
          FOR  i    :=   2    TO   14   DO
               $H(i)          =    M_MillFeed(i) - M_MillProduct(i) - Si(i)*H(i)            ;
          END
          $H(1)               =    M_MillFeed(1) - M_MillProduct(1) - Si(1)*H(1)            ;

      # Pulp Density in the mill
           $Cs_MillProduct     =    Q_MillFeed/V_Mill*(Cs_MillFeed-Cs_MillProduct)           ;

      # Specific Energy mixing, and density
          $xSIproduct         =    (M_MillFeed_T*xSIfeed - M_MillProduct_T*xSIproduct)/Ht   ;
          $roProduct          =    (M_MillFeed_T*roFeed  - M_MillProduct_T*roProduct )/Ht   ;

END #MILL
#============================================================================================#
```

Sump Model

```
#=============================================================================================#
MODEL     Sump

     VARIABLE
          M_SumpFeed          AS        ARRAY(15)      OF   NoType
          Cs_SumpFeed         AS                            NoType
          Q_SumpFeed          AS                            NoType
          Q_SumpProduct       AS                            NoType
          M_SumpProduct       AS        ARRAY(15)      OF   NoType
          Cs_SumpProduct      AS                            NoType

          V_SumpFill          AS        ARRAY(15)      OF   NoType
          V_SumpWater         AS                            NoType
          V_SumpFill_T        AS                            NoType

          P40_SumpProduct     AS                            NoType
          P40_SumpFeed        AS                            NoType

          M_SumpProduct_T     AS                            NoType
          M_SumpFeed_T        AS                            NoType

          H_Sump              AS                            NoType
          H_Sumpmin           AS                            NoType
          H_Sumpmax           AS                            NoType
          H_SumpCon           AS                            NoType

          A_Sump              AS                            NoType

          roFeed              AS                            NoType
          roProduct           AS                            NoType

          xSIfeed             AS        ARRAY(15)      OF   NoType
          xSIproduct          AS        ARRAY(15)      OF   NoType


          STREAM

Inlet     :    M_SumpFeed, Cs_SumpFeed, xSIfeed, roFeed              AS   MCsSIro
Outlet    :    M_SumpProduct, Cs_SumpProduct, xSIproduct, roProduct  AS   MCsSIro
PI_con    :    H_Sump                                                AS   Notype
PI_man    :    Q_SumpProduct                                         AS   Notype

#<-------------------------------------------------------------------------------------------->#
     EQUATION

#PATH CONSTRAINT monitoring:
$H_sumpcon               =    (1E1*max(0,H_Sumpmin-H_sump,H_sump-H_sumpmax))^2                ;

#ALGEBRAIC EQUATIONS
H_Sump                   =    V_SumpFill_T/A_Sump                                             ;
Q_SumpFeed               =    sigma(M_SumpFeed)/Cs_SumpFeed                                   ;
M_SumpProduct            =   (Q_SumpProduct*Cs_SumpProduct)*(V_SumpFill/sigma(V_SumpFill))    ;
V_SumpFill_T             =    sigma(V_SumpFill) + V_SumpWater                                 ;
Cs_SumpProduct           =    sigma(V_SumpFill)*roProduct/V_SumpFill_T                        ;
P40_SumpProduct          =    sigma(M_SumpProduct(12:15))/sigma(M_SumpProduct)                ;
P40_SumpFeed             =    sigma(M_SumpFeed(12:15))/sigma(M_SumpFeed)                      ;
M_SumpProduct_T          =    sigma(M_SumpProduct)                                            ;
M_SumpFeed_T             =    sigma(M_SumpFeed)                                               ;

#    DIFFERENTIAL   EQUATIONS
$V_SumpFill_T            =    Q_SumpFeed     -    Q_SumpProduct                               ;
$V_SumpFill              =    M_SumpFeed/roFeed   -    M_SumpProduct/roProduct               ;
$roProduct               =    (M_SumpFeed_T*roFeed  - M_SumpProduct_T*roProduct
                               )/V_SumpFill_T                                                ;
$xSIproduct              =    (M_SumpFeed_T*xSIfeed -
                               M_SumpProduct_T*xSIproduct)/V_SumpFill_T                      ;

END  # Sump
#=============================================================================================#
```

Cyclone Model

```
#=============================================================================================#
MODEL     Cyclone

     VARIABLE
# FEED VARIABLES #
M_CycloneFeed             AS        ARRAY(15)      OF   NoType
M_CycloneFeed_T           AS                             NoType
Cs_CycloneFeed            AS                             NoType
Q_CycloneFeed             AS                             NoType
W_CycloneFeed             AS                             NoType
P40_CycloneFeed           AS                             NoType
xSIfeed                   AS        ARRAY(15)      OF   NoType
roFeed                    AS                             NoType

# OVERFLOW VARIABLES #
M_CycloneOverflow         AS        ARRAY(15)      OF   NoType
M_CycloneOverflow_T       AS                             NoType
Cs_CycloneOverflow        AS                             NoType
W_CycloneOverflow         AS                             NoType
Yi_CycloneOverFlow        AS        ARRAY(15)      OF   NoType
P40_CycloneOverFLow       AS                             NoType
Q_CycloneOverFlow         AS                             NoType

# UNDERFLOW VARAIABLES #
M_CycloneUnderflow        AS        ARRAY(15)      OF   NoType
M_CycloneUnderflow_T      AS                             NoType
Cs_CycloneUnderflow       AS                             NoType
W_CycloneUnderflow        AS                             NoType
Yi_CycloneUnderFlow       AS        ARRAY(15)      OF   NoType
FW_CycloneUnderFlow       AS                             NoType
P40_CycloneUnderFlow      AS                             NoType

# CYCLONE VARIABLES #
fv_Cyclone                AS                             NoType
d50_Cyclone               AS                             NoType
d50_uncertain             AS                             NoType
Rf_Cyclone                AS                             NoType
D                         AS        ARRAY(15)      OF   NoType
m                         AS                             NoType

MINMAXFUN_SUM             AS                             NoType
MAXFUN_P                  AS                             NoType
MINFUN_P                  AS                             NoType
DP40_CYCLONEOVERFLOW      AS                             NoType


STREAM

Inlet          :    M_CycloneFeed , Cs_CycloneFeed, xSIfeed, roFeed          AS MCsSIro
Outlet1        :    M_CycloneUnderFlow , Cs_CycloneUnderFlow, xSIfeed, roFeed AS MCsSIro
Outlet2        :    M_CycloneOverFlow , Cs_CycloneOverFlow, xSIfeed, roFeed   AS   MCsSIro

CsFeed         :    Cs_CycloneFeed                                           AS   NoType
QFeed          :    Q_CycloneFeed                                            AS   NoType
MFeed          :    M_CycloneFeed_T                                          AS   NoType
P40Feed        :    P40_CycloneFeed                                          AS   NoType

CsProduct      :    Cs_CycloneOverFlow                                       AS   NoType
QProduct       :    Q_cycloneOverFlow                                        AS   NoType
MProduct       :    M_CycloneOverFlow_T                                      AS   NoType
P40Product     :    P40_CycloneOverFlow                                      AS   NoType

#<-------------------------------------------------------------------------------------------->#

     EQUATION

# Calculate the water flow rate in kg/min:
W_CycloneFeed       =         (sigma(M_CycloneFeed)/Cs_CycloneFeed -
                              sigma(M_CycloneFeed)/roFeed)/60*1000                              ;
# Calculate fv, fraction of solids in feed:
fv_Cyclone          =         Cs_CycloneFeed/roFeed                                             ;

# Calculate  Q, volumetric feed rate (l/min):
Q_CycloneFeed       =    sigma(M_CycloneFeed)/Cs_CycloneFeed*1000/60                            ;

# Declare size class top sizes:

W_CycloneOverFlow = (1+(min(100,max(-100,(1E3*(W_CycloneFeed-21.4)))))/2*
                    (1.36W_CycloneFeed-10.75) +
```

```
                    (1+(min(100,max(-100,(1E3*(21.4-W_CycloneFeed-21.4)))))/2*
                    (0.837W_CycloneFeed+0.35)                                          ;
W_CycloneUnderFlow                  =    W_CycloneFeed  -    W_CycloneOverFlow         ;
# Calculate d50:
d50_Cyclone                         =    exp(3.616-15.006e-2*Q_CycloneFeed+2.3*fv_Cyclone)  ;

# Calculate WS, fraction of feed water reporting to underflow:
FW_CycloneUnderFlow                 =    (W_CycloneFeed-W_CycloneOverFLow)/W_CycloneFeed    ;


# Calculate Rf:
Rf_Cyclone                          =    0.818-0.7932*FW_CycloneUnderFlow                   ;
# Calculate Yi's:

Yi_CycloneOverFLow                  =
                           exp(-0.6931*D/SQRT((d50_uncertain*d50_Cyclone)^2)^m)            ;

Yi_CycloneUnderFlow                 =    1 - Yi_CycloneOverFlow                             ;
M_CycloneUnderFlow                  =    Yi_CycloneUnderFlow*M_CycloneFeed                  ;
M_CycloneOverFlow                   =    Yi_CycloneOverFlow *M_CycloneFeed                  ;
Cs_CycloneUnderFlow                 =
                                          sigma(M_CycloneUnderFlow)/(W_CycloneUnderFlow*60/1000 +
                           sigma(M_CycloneUnderFlow)/roFeed)                               ;

Cs_CycloneOverFlow                  =    sigma(M_CycloneOverFlow)/
                                         (W_CycloneOverFlow*60/1000 +
                                         sigma(M_CycloneOverFlow)/roFeed)                  ;

Q_CycloneOverFlow                   =    M_CycloneOverFlow_T/Cs_CycloneOverFlow             ;
P40_CycloneUnderFlow                =
                               sigma(M_CycloneUnderflow(12:15))/sigma(M_CycloneUnderflow)  ;
P40_CycloneOverFlow                 =   (M_CycloneOverflow(12:15))/sigma(M_CycloneOverflow) ;
P40_CycloneFeed                     =    sigma(M_CycloneFeed(12:15))/sigma(M_CycloneFeed)   ;

M_CycloneFeed_T                     =    sigma(M_CycloneFeed)                               ;
M_CycloneOverFlow_T                 =    sigma(M_CycloneOverFlow)                           ;
M_CycloneUnderFlow_T                =    sigma(M_CycloneUnderFlow)                          ;


# OBJECTIVE FUNCTION CALCULATIONS:



# Max value tracking:
$P40_CYCLONEOVERFLOW                =    DP40_CYCLONEOVERFLOW                               ;
$MAXFUN_P                           =
          (1+tanh(min(100,max(-100,1E20*(P40_CYCLONEOVERFLOW-MAXFUN_P)))))/2*
          max(0,DP40_CYCLONEOVERFLOW,P40_CYCLONEOVERFLOW-MAXFUN_P)                         ;
# Min value tracking:
$MINFUN_P                           =
        (1+tanh(min(100,max(-100,1E20*(MINFUN_P-P40_CYCLONEOVERFLOW)))))/2*
     min(0,DP40_CYCLONEOVERFLOW,P40_CYCLONEOVERFLOW-MINFUN_P)                            ;

# Minmax sum:
MINMAXFUN_SUM                       =    (MAXFUN_P-MINFUN_P)                                ;

END  # Cyclone
#=============================================================================================#
```

Mixer Model for water addition

```
#=============================================================================================#
MODEL MIXER_x
     # Model to simulate water addition and stream mixing at any point in the circuit #


     VARIABLE

# FEED VARIABLES #
M_MixerFeed1              AS        ARRAY(15)      OF   NoType
M_MixerFeed1_T            AS                            NoType
x_MixerFeed1              AS        ARRAY(15)      OF   NoType
Cs_MixerFeed1             AS                            NoType
P40_MixerFeed1            AS                            NoType
roFeed1                   AS                            NoType
xSIfeed1                  AS        ARRAY(15)      OF   NoType

M_MixerFeed2              AS        ARRAY(15)      OF   NoType
M_MixerFeed2_T            AS                            NoType
Cs_MixerFeed2             AS                            NoType
P40_MixerFeed2            AS                            NoType
roFeed2                   AS                            NoType
xSIfeed2                  AS        ARRAY(15)      OF   NoType

W_MixerFeed               AS                            NoType
W_MixerFeedman            AS                            NoType


# PRODUCT VARIABLES #
M_MixerProduct            AS        ARRAY(15)      OF   NoType
Cs_MixerProduct           AS                            NoType
P40_MixerProduct          AS                            NoType
roProduct                 AS                            NoType
xSIProduct                AS        ARRAY(15)      OF   NoType


          STREAM

Inlet1    : M_MixerFeed1, Cs_MixerFeed1, xSIfeed1, roFeed1          AS   MCsSIro
Inlet2    : M_MixerFeed2, Cs_MixerFeed2, xSIfeed2, roFeed2          AS   MCsSIro
Outlet    : M_MixerProduct, Cs_MixerProduct, xSIproduct, roProduct AS   MCsSIro

Waterin     : W_MixerFeed                                            AS   NoType
Waterinman : W_MixerFeedman                                          AS   NoType

#<-------------------------------------------------------------------------------------------->#
     EQUATION

M_MixerFeed1   =    M_MixerFeed1_T*x_MixerFeed1/sigma(x_MixerFeed1)                          ;

M_MixerProduct =    M_MixerFeed1   +    M_MixerFeed2                                        ;

Cs_MixerProduct=    sigma(M_MixerProduct)/((sigma(M_MixerFeed1)/Cs_MixerFeed1)
                                  + (sigma(M_MixerFeed2)/Cs_MixerFeed2)
                                  + W_MixerFeed/1000 + W_MixerFeedman/1000)              ;

M_MixerFeed2_T =    sigma(M_MixerFeed2)                                                 ;

P40_MixerFeed1          =    sigma(M_MixerFeed1(12:15))/sigma(M_MixerFeed1)             ;
P40_MixerFeed2          =    sigma(M_MixerFeed2(12:15))/sigma(M_MixerFeed2)             ;
P40_MixerProduct        =    sigma(M_MixerProduct(12:15))/sigma(M_MixerProduct)         ;

roProduct               =    ( M_MixerFeed1_T*roFeed1 + M_MixerFeed2_T*roFeed2
                             )/(sigma(M_MixerProduct))                                  ;
xSIproduct              =    ( M_MixerFeed1_T*xSIfeed1 + M_MixerFeed2_T*xSIfeed2
                              )/(sigma(M_MixerProduct))                                 ;

END  # Mixer_x
#=============================================================================================#
```

Mixer Model for stream mixing

```
#=============================================================================================#
MODEL MIXER
     # Model to simulate water addition and stream mixing at any point in the circuit #

     VARIABLE

          # FEED VARIABLES #
          M_MixerFeed              AS        ARRAY(15)      OF   NoType
          M_MixerFeed_T            AS                            NoType
          Cs_MixerFeed             AS                            NoType
          W_MixerFeed              AS                            NoType
          P40_MixerFeed            AS                            NoType
          W_MixerFeedman           AS                            NoType

          roFeed                   AS                            NoType
          xSIFeed                  AS        ARRAY(15)      OF   NoType

          # PRODUCT VARIABLES #
          M_MixerProduct           AS        ARRAY(15)      OF   NoType
          M_MixerProduct_T         AS                            NoType
          Cs_MixerProduct          AS                            NoType
          P40_MixerProduct         AS                            NoType

          STREAM

               Inlet     :    M_MixerFeed, Cs_MixerFeed, xSIfeed, roFeed        AS   MCsSIro
               Outlet    :    M_MixerProduct, Cs_MixerProduct, xSIfeed, roFeed  AS   MCsSIro

               Waterin   :    W_MixerFeed                                       AS   NoType
               Waterinman:    W_MixerFeedman                                    AS   NoType

     EQUATION


          M_MixerProduct           =    M_MixerFeed                                        ;

          Cs_MixerProduct          =
                                    sigma(M_MixerProduct)/((sigma(M_MixerFeed)/Cs_MixerFeed)
                                    + W_MixerFeed/1000 + W_MixerFeedman/1000)              ;

          M_MixerFeed_T            =    sigma(M_MixerFeed)                                 ;
          M_MixerProduct_T         =    sigma(M_MixerProduct)                              ;

          P40_MixerFeed            =    sigma(M_MixerFeed(12:15))/sigma(M_MixerFeed)       ;
          P40_MixerProduct         =    sigma(M_MixerProduct(12:15))/sigma(M_MixerProduct) ;

     END  # Mixer
#=============================================================================================#
```

PI Controller

```
#=============================================================================================#
MODEL AnalogPIController

     VARIABLE

          SetPoint             AS   NoType
          Gain                 AS   NoType
          ResetTime            AS   NoType
          MaxSignal            AS   NoType
          MinSignal            AS   NoType
          Error                AS   NoType
          IntError             AS   NoType
          Measurement          AS   NoType

          Signal               AS   NoType
          CalcSignal           AS   NoType


     STREAM

         Input  : Measurement  AS  Notype
         Output : Signal       AS  Notype

#<-------------------------------------------------------------------------------------------->#
     EQUATION

          # Controller error:
          Error = SetPoint - Measurement                                                    ;
          # Definition of integral term of error:
          $IntError = Error                                                                 ;
          # Definition of output signal:
          CalcSignal = Gain * (Error + IntError / ResetTime)                                ;
          Signal          =    max(min(CalcSignal,MaxSignal),MinSignal)                     ;

END # Model AnalogPIController
#=============================================================================================#
```

Full Model Description

```
#=============================================================================================#
MODEL     Sumponly
     UNIT

          SagMill                  AS        Mill
          HoldingTank              AS        SUMP
          Cyclone1                 AS        Cyclone
          Mixer1                   AS        Mixer_x
          Mixer2                   AS        Mixer
          Mixer3                   AS        Mixer
          PI_sump                  AS        AnalogPIController
#         PI_1                     AS        AnalogPIController


     EQUATION

          SagMill.Outlet           =         Mixer2.Inlet                                    ;
          HoldingTank.Inlet        =         Mixer2.Outlet                                   ;
          HoldingTank.Outlet       =         Mixer3.Inlet                                    ;
          Cyclone1.Inlet           =         Mixer3.Outlet                                   ;
          Mixer1.Inlet2            =         Cyclone1.Outlet1                                ;
          SagMill.Inlet            =         Mixer1.Outlet                                   ;
          HoldingTank.PI_man       =         PI_Sump.Output                                  ;
          HoldingTank.PI_con       =         PI_Sump.Input                                   ;

#         Mixer1.Waterinman        =         PI_1.Output                                     ;
#         Cyclone1.QProduct        =         PI_1.Input                                      ;

     # Monitor minmax of sump-level:

     # Max value tracking:

         H_SUMPSET                          =    PI_SUMP.SETPOINT                           ;
         HoldingTank.$H_SUMP                =    DHSUMP                                     ;
         $MAXFUN_HSUMP                      =
         (1+tanh(min(100,max(-100,1E4*(HOLDINGTANK.H_SUMP-MAXFUN_HSUMP)))))/2*
         max(0,DHSUMP,10*(HOLDINGTANK.H_SUMP-MAXFUN_HSUMP))                                    ;


     # Min value tracking:

         $MINFUN_HSUMP                      =
         (1+tanh(min(100,max(-100,1E4*(MINFUN_HSUMP-HOLDINGTANK.H_SUMP)))))/2*
         min(0,DHSUMP,10*(HOLDINGTANK.H_SUMP-MINFUN_HSUMP))                                    ;

     # Minmax sum:
         MINMAX_HSUMP                       =    ((MAXFUN_HSUMP-H_SUMPSET)+
                                                  (H_SUMPSET-MINFUN_HSUMP))                  ;
END
#=============================================================================================#
```

```
_____ 16 Parrallel Mill Circuit Models for flexibility constraint inclusion _____
#=============================================================================================#
MODEL SUMPONLY_ARRAY_16

UNIT
     CIRCUIT                  AS   ARRAY(16) OF SUMPONLY

VARIABLE

     A_Sump                   AS   NoType
     SUMPGAIN                 AS   NoType
     SUMPRESETTIME            AS   NoType
     SUMPSETPOINT             AS   NoType

#    PI_1GAIN                 AS   NoType
#    PI_1RESETTIME            AS   NoType
#    PI_1SETPOINT             AS   NoType

     W_MixerFeed1             AS   NoType
     W_MixerFeed2             AS   NoType
     V_mill                   AS   NoType

     xSI1                     AS   NoType
     xSI2                     AS   NoType

     xSI1_g                   AS   NoType
     xSI2_g                   AS   NoType

     D50scale                 AS   NoType
     D50scale_g               AS   NoType

     MILL_P                   AS   NoType
     MILL_P_g                 AS   noType

#    ISE                      AS   NoType
#    MINMAXFUN_DEV            AS   NoType
     MINMAXFUN_SUM            AS   NoType
     HSUMPCON                 AS   NoType

     M_MixerFeed1_T_Asin           AS   NoType
     M_MixerFeed1_T_Wsin           AS   NoType

     M_MixerFeed1_T_Astep          AS   NoType
     M_MixerFeed1_T_Wstep          AS   NoType


     ROFEED1_Asin                  AS   NoType
     ROFEED1_Wsin                  AS   NoType

     ROFEED1_Astep                 AS   NoType
     ROFEED1_Wstep                 AS   NoType


     xSIFeed1_Asin                 AS   NoType
     xSIFeed1_Wsin                 AS   NoType

     xSIFeed1_Astep                AS   NoType
     xSIFeed1_Wstep                AS   NoType


     x_Asin                        AS   NoType
     x_Wsin                        AS   NoType

     x_Astep                       AS   NoType
     x_Wstep                       AS   NoType

     M_MixerFeed1_T_Asin_g         AS   NoType
     M_MixerFeed1_T_Wsin_g         AS   NoType

     M_MixerFeed1_T_Astep_g        AS   NoType
     M_MixerFeed1_T_Wstep_g        AS   NoType

     ROFEED1_Asin_g                AS   NoType
     ROFEED1_Wsin_g                AS   NoType

     ROFEED1_Astep_g               AS   NoType
     ROFEED1_Wstep_g               AS   NoType

     xSIFeed1_Asin_g               AS   NoType
     xSIFeed1_Wsin_g               AS   NoType

     xSIFeed1_Astep_g              AS   NoType
```

```
    xSIFeed1_Wstep_g              AS   NoType

    x_Asin_g                      AS   NoType
    x_Wsin_g                      AS   NoType

    x_Astep_g                     AS   NoType
    x_Wstep_g                     AS   NoType

#<-------------------------------------------------------------------------------------------->#
    EQUATION

        for i := 1 to 8  do
            CIRCUIT(2*i-1).CYCLONE1.d50_uncertain   =    d50scale                          ;
            CIRCUIT(2*i).CYCLONE1.d50_uncertain     =    d50scale_g                        ;
            CIRCUIT(2*i-1).SAGMILL.P                =    MILL_P                            ;
            CIRCUIT(2*i).SAGMILL.P                  =    MILL_P_g                          ;

        end

        W_MixerFeed2                          =    470 - W_MixerFeed1                     ;

        CIRCUIT(1).Mixer1.M_MixerFeed1_T      =    113+
                                                   M_MixerFeed1_T_Asin*
                                                   (1+tanh(min(100,max(-100,10*(TIME-1)))))/2*
                                                   SIN(TIME*3.142*2*M_MixerFeed1_T_Wsin)  ;

        CIRCUIT(2).Mixer1.M_MixerFeed1_T      =    113+
                                                   M_MixerFeed1_T_Asin_g*
                                                   (1+tanh(min(100,max(-100,10*(TIME-1)))))/2*
                                                   SIN(TIME*3.142*2*M_MixerFeed1_T_Wsin_g)  ;


        CIRCUIT(3).Mixer1.M_MixerFeed1_T      =    113+
                                                   M_MixerFeed1_T_Astep*
                                                   (1+tanh(min(100,max(-100,10*(TIME-1)))))/2*
                              (1+tanh(min(100,max(-100,10*(M_MixerFeed1_T_Wstep-TIME+1)))))/2  ;

        CIRCUIT(4).Mixer1.M_MixerFeed1_T      =    113+
                                                   M_MixerFeed1_T_Astep_g*
                                                   (1+tanh(min(100,max(-100,10*(TIME-1)))))/2*
                          (1+tanh(min(100,max(-100,10*(M_MixerFeed1_T_Wstep_g-TIME+1)))))/2     ;


        CIRCUIT(5).Mixer1.ROFEED1             =    3000+
                                                   ROFEED1_Asin*
                                                   (1+tanh(min(100,max(-100,10*(TIME-1)))))/2*
                                                   SIN(TIME*3.142*2*ROFEED1_Wsin)          ;

        CIRCUIT(6).Mixer1.ROFEED1             =    3000+
                                                   ROFEED1_Asin_g*
                                                   (1+tanh(min(100,max(-100,10*(TIME-1)))))/2*
                                                   SIN(TIME*3.142*2*ROFEED1_Wsin_g)        ;


        CIRCUIT(7).Mixer1.ROFEED1             =    3000+
                                                   ROFEED1_Astep*
                                                   (1+tanh(min(100,max(-100,10*(TIME-1)))))/2*
                                                   (1+tanh(min(100,max(-100,10*(
                                                   ROFEED1_Wstep
                                                   -TIME+1)))))/2 ;


        CIRCUIT(8).Mixer1.ROFEED1             =    3000+
                                                   ROFEED1_Astep_g*
                                                   (1+tanh(min(100,max(-100,10*(TIME-1)))))/2*
                                                   (1+tanh(min(100,max(-100,10*(
                                                   ROFEED1_Wstep_g-TIME+1)))))/2           ;


        xSI1                                  =    0.5+xSIFeed1_Asin*
                                                   (1+tanh(min(100,max(-100,10*(TIME-1)))))/2*
                                                   SIN(TIME*3.142*2*xSIFEED1_Wsin)         ;

        xSI1_g                                     =    0.5+xSIFeed1_Asin_g*
                                                   (1+tanh(min(100,max(-100,10*(TIME-1)))))/2*
                                                   SIN(TIME*3.142*2*xSIFEED1_Wsin_g)       ;
```

```
CIRCUIT(9).Mixer1.xSIfeed1(1)          =    xSI1                                    ;
.... # omitted
CIRCUIT(9).Mixer1.xSIfeed1(15)         =    xSI1                                    ;

CIRCUIT(10).Mixer1.xSIfeed1(1)         =    xSI1_g                                  ;
.....# omitted
CIRCUIT(10).Mixer1.xSIfeed1(15)        =    xSI1_g                                  ;

xSI2                                =    0.5+xSIFeed1_Astep*
                                         (1+tanh(min(100,max(-100,10*(TIME-1)))))/2*
                                         (1+tanh(min(100,max(-100,10*(
                                         xSIFeed1_Wstep-TIME+1)))))/2               ;

xSI2_g                              =    0.5+xSIFeed1_Astep_g*
                                         (1+tanh(min(100,max(-100,10*(TIME-1)))))/2*
                                         (1+tanh(min(100,max(-100,10*(
                                         xSIFeed1_Wstep_g-TIME+1)))))/2             ;


CIRCUIT(11).Mixer1.xSIfeed1(1)              =    xSI2                               ;
.... # omitted
CIRCUIT(11).Mixer1.xSIfeed1(15)             =    xSI2                               ;

CIRCUIT(12).Mixer1.xSIfeed1(1)              =    xSI2_g                             ;
..... # omitted
CIRCUIT(12).Mixer1.xSIfeed1(15)             =    xSI2_g                             ;

CIRCUIT(13).Mixer1.x_MixerFeed1(1 )         =    (1+x_Asin*(1+tanh(min(100,
                                                 max(-100,10*(TIME-1)))))/2*
                                       (sin(TIME*3.142*2*x_Wsin)*(16-7 )/15))*0.062323;
..... # omitted
CIRCUIT(13).Mixer1.x_MixerFeed1(15)         =    (1+x_Asin*(1+tanh(min(100,max
                                                 (-100,10*(TIME-1)))))/2*
                                        (sin(TIME*3.142*2*x_Wsin)*(16-15)/15))*0.10228;
..... # omitted
CIRCUIT(14).Mixer1.x_MixerFeed1(15)         =    (1+x_Asin_g*(1+tanh(min(100,
                                                 max(-100,10*(TIME-1)))))/2*
                                      (sin(TIME*3.142*2*x_Wsin_g)*(16-15)/15))*0.10228;
..... # omitted
CIRCUIT(15).Mixer1.x_MixerFeed1(15)         =    (1+x_Astep*((1+tanh(min(100,
                                                 max(-100,10*(TIME-1)))))/2*
                                   (1+tanh(min(100,max(-100,10*(x_Wstep-TIME+1)))))/2)*
                                                 (16-15)/15)*0.10228;
..... # omitted
CIRCUIT(16).Mixer1.x_MixerFeed1(15)         =    (1+x_Astep_g*((1+tanh(min(100,max
                                                 (-100,10*(TIME-1)))))/2*
                                  (1+tanh(min(100,max(-100,10*(x_Wstep_g-TIME+1)))))/2)*
                                                                  (16-15)/15)*0.10228;


FOR i := 1 to 4 DO
          FOR n := 1 TO 15 DO
               CIRCUIT(i).Mixer1.xSIfeed1(n) =    0.5  ;
          END
          CIRCUIT(i).Mixer1.ROFEED1          =    3000 ;

          CIRCUIT(i).Mixer1.x_MixerFeed1(1)  =    0.01468   ;
          ..... # omitted
          CIRCUIT(i).Mixer1.x_MixerFeed1(15) =    0.10228   ;
END


FOR i := 5 TO 8 DO
          CIRCUIT(i).Mixer1.M_MixerFeed1_T   =    113  ;
          FOR n := 1 TO 15 DO
               CIRCUIT(i).Mixer1.xSIfeed1(n) =    0.5  ;
          END

          CIRCUIT(i).Mixer1.x_MixerFeed1(1)  =    0.01468   ;
          CIRCUIT(i).Mixer1.x_MixerFeed1(2)  =    0.026064  ;
          CIRCUIT(i).Mixer1.x_MixerFeed1(3)  =    0.020528  ;
          CIRCUIT(i).Mixer1.x_MixerFeed1(4)  =    0.036447  ;
          CIRCUIT(i).Mixer1.x_MixerFeed1(5)  =    0.036504  ;
          CIRCUIT(i).Mixer1.x_MixerFeed1(6)  =    0.043167  ;
          CIRCUIT(i).Mixer1.x_MixerFeed1(7)  =    0.062323  ;
          CIRCUIT(i).Mixer1.x_MixerFeed1(8)  =    0.076212  ;
          CIRCUIT(i).Mixer1.x_MixerFeed1(9)  =    0.060025  ;
          CIRCUIT(i).Mixer1.x_MixerFeed1(10) =    0.10657   ;
```

```
                CIRCUIT(i).Mixer1.x_MixerFeed1(11) =    0.083937  ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(12) =    0.095984  ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(13) =    0.17041   ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(14) =    0.064861  ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(15) =    0.10228   ;
        END

        FOR i := 9 TO 12 DO
                CIRCUIT(i).Mixer1.M_MixerFeed1_T   =    113  ;
                CIRCUIT(i).Mixer1.ROFEED1          =    3000 ;

                CIRCUIT(i).Mixer1.x_MixerFeed1(1)  =    0.01468   ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(2)  =    0.026064  ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(3)  =    0.020528  ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(4)  =    0.036447  ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(5)  =    0.036504  ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(6)  =    0.043167  ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(7)  =    0.062323  ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(8)  =    0.076212  ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(9)  =    0.060025  ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(10) =    0.10657   ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(11) =    0.083937  ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(12) =    0.095984  ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(13) =    0.17041   ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(14) =    0.064861  ;
                CIRCUIT(i).Mixer1.x_MixerFeed1(15) =    0.10228   ;
        END

        FOR i := 13 TO 16 DO
                CIRCUIT(i).Mixer1.M_MixerFeed1_T   =    113  ;
                CIRCUIT(i).Mixer1.ROFEED1          =    3000 ;
                FOR n := 1 TO 15 DO
                    CIRCUIT(i).Mixer1.xSIfeed1(n) =    0.5  ;
                END
        END

# SPECIFICATION OF TIME INVARIANT PARAMETERS TO BE THE SAME FOR ALL CIRCUITS
    FOR i := 1 TO 16 DO
            CIRCUIT(i).PI_SUMP.GAIN             =           SUMPGAIN  ;
            CIRCUIT(i).PI_SUMP.RESETTIME        =           SUMPRESETTIME  ;
            CIRCUIT(i).PI_SUMP.SETPOINT         =           SUMPSETPOINT   ;

#           CIRCUIT(i).PI_1.GAIN                =           PI_1GAIN  ;
#           CIRCUIT(i).PI_1.RESETTIME           =           PI_1RESETTIME  ;
#           CIRCUIT(i).PI_1.SETPOINT            =           PI_1SETPOINT   ;

            CIRCUIT(i).HoldingTank.A_Sump       =           A_Sump         ;

            CIRCUIT(i).Mixer1.W_MixerFeed       =           W_MixerFeed1   ;
            CIRCUIT(i).Mixer2.W_MixerFeed       =           W_MixerFeed2   ;

            CIRCUIT(i).SagMill.V_mill           =           V_Mill         ;
    END

        MINMAXFUN_SUM           =   (CIRCUIT(1 ).CYCLONE1.MINMAXFUN_SUM)    +
                                    (CIRCUIT(3 ).CYCLONE1.MINMAXFUN_SUM)    +
                                    (CIRCUIT(5 ).CYCLONE1.MINMAXFUN_SUM)    +
                                    (CIRCUIT(7 ).CYCLONE1.MINMAXFUN_SUM)    +
                                    (CIRCUIT(9 ).CYCLONE1.MINMAXFUN_SUM)    +
                                    (CIRCUIT(11).CYCLONE1.MINMAXFUN_SUM)    +
                                    (CIRCUIT(13).CYCLONE1.MINMAXFUN_SUM)    +
                                    (CIRCUIT(15).CYCLONE1.MINMAXFUN_SUM)    ;


        HSUMPCON                =   (CIRCUIT(1 ).HOLDINGTANK.H_SUMPCON)     +
                                    ..... # omitted
                                    (CIRCUIT(15).HOLDINGTANK.H_SUMPCON)     +
                                    (CIRCUIT(16).HOLDINGTANK.H_SUMPCON)     ;

END
#  ================================================================================#
```

Process for execution/optimization

```
#=============================================================================================#

     UNIT

          SM01 AS    SUMPONLY_ARRAY


     ASSIGN


          SM01.SUMPGAIN                        :=  -3.68019E-01    ;
          SM01.SUMPRESETTIME                   := 1                ;
          SM01.A_Sump                          := 0.3              ;
          SM01.SUMPSETPOINT                    :=  5.91077E-01     ;
          SM01.W_MixerFeed1                    := 330              ;
          SM01.V_mill                          :=    5.94433E-02   ;

          SM01.M_MixerFeed1_T_Asin             :=   10         ;
          SM01.M_MixerFeed1_T_Wsin             :=   .167       ;
          SM01.M_MixerFeed1_T_Astep            :=   -10        ;
          SM01.M_MixerFeed1_T_Wstep            :=   3          ;
          SM01.ROFEED1_Asin                    :=   300        ;
          SM01.ROFEED1_Wsin                    :=   .167       ;
          SM01.ROFEED1_Astep                   :=   300        ;
          SM01.ROFEED1_Wstep                   :=   3     ;
          SM01.xSIFeed1_Asin                   :=   0.4        ;
          SM01.xSIFeed1_Wsin                   :=   .167       ;
          SM01.xSIFeed1_Astep                  :=   .4         ;
          SM01.xSIFeed1_Wstep                  :=   3          ;
          SM01.x_Asin                          :=   .1         ;
          SM01.x_Wsin                          :=   .167       ;
          SM01.x_Astep                         :=   .1         ;
          SM01.x_Wstep                         :=   3          ;
          SM01.d50scale                        :=   1          ;
          SM01.MILL_P                          :=   55         ;


          SM01.M_MIXERFEED1_T_WSIN_g:=     1.67000E-01;
         SM01.ROFEED1_WSIN_g          :=     1.67000E-01;
         SM01.XSIFEED1_WSIN_g         :=     1.67000E-01;
         SM01.X_WSIN_g                :=     1.67000E-01;
         SM01.D50SCALE_g              :=     9.00000E-01;
         SM01.MILL_P_g                :=     6.00000E+01;

          SM01.M_MixerFeed1_T_Asin_g           :=   10         ;
#         SM01.M_MixerFeed1_T_Wsin_g           :=   .167       ;
          SM01.M_MixerFeed1_T_Astep_g          :=   -10        ;
          SM01.M_MixerFeed1_T_Wstep_g          :=   3          ;
          SM01.ROFEED1_Asin_g                  :=   300        ;
#         SM01.ROFEED1_Wsin_g                  :=   .29767     ;
          SM01.ROFEED1_Astep_g                 :=   300        ;
          SM01.ROFEED1_Wstep_g                 :=   2.235      ;
          SM01.xSIFeed1_Asin_g                 :=   0.4        ;
#         SM01.xSIFeed1_Wsin_g                 :=   1          ;
          SM01.xSIFeed1_Astep_g                :=   .4         ;
          SM01.xSIFeed1_Wstep_g                :=   3          ;
          SM01.x_Asin_g                        :=   .1         ;
#         SM01.x_Wsin_g                        :=   .167       ;
          SM01.x_Astep_g                       :=   .1         ;
          SM01.x_Wstep_g                       :=   3          ;
#         SM01.d50scale_g                      :=   1          ;
#         SM01.MILL_P_g                        :=   55         ;


     FOR i := 1 TO 16 DO

     WITHIN SM01.CIRCUIT(i)  DO

     # Mill Settings

SagMill.b              :=        [0,  0,   0,   0,   0,   0,   0,   0,   0,   0,
0,   0,   0,   0,   0,
.........# omitted

SagMill.SiE2(1:15)       :=   [238.3304,171.4681,228.2171,247.4073,3.8848,4.2105,2.1719,
                                2.0370,0.8406,1.7213,0.5414,1.1202,0.0147,1.0211,
                                0.8149];
```

```
SagMill.SiE1(1:15)        :=   [291.2928,209.5721,278.9320,302.3867,4.7481,5.1461,2.6545,
                                2.4896,1.0274,2.1039,0.6617,1.3692,0.0179,1.2480,
                                0.9960];


#         SagMill.P                       :=   55                          ;

     # PI CONTROLLER SETTINGS ON SUMP:
          PI_Sump.MaxSignal               :=   1                           ;
          PI_Sump.MinSignal               :=   0                           ;

#         PI_1.MaxSignal                  :=   20                          ;
#         PI_1.MinSignal                  :=   -20                         ;

#         PI_2.MaxSignal                  :=   20                          ;
#         PI_2.MinSignal                  :=   -20                         ;

     # Mixer 1 Paramters
#         Mixer1.x_MixerFeed1             :=   [0.01468, 0.026064, 0.020528, 0.036447, 0.036504,
 0.043167,
#              0.062323, 0.076212, 0.060025, 0.10657,  0.083937, 0.095984, 0.17041,  0.064861,
0.10228];
          Mixer1.Cs_MixerFeed1            :=   2999                        ;
          Mixer1.W_mixerFeedman           :=   0                           ;


     # Holding Tank

          HoldingTank.H_Sumpmin           :=   0.4                         ;
          HoldingTank.H_Sumpmax           :=   0.6                         ;


     # Mixer 2 and 3:
          Mixer2.W_mixerFeedman           :=   0                           ;
          Mixer3.W_MixerFeed              :=   0                           ;
          Mixer3.W_mixerFeedman           :=   0                           ;


     # Cyclone
          Cyclone1.D                      :=   [2400, 1700, 1200, 810, 575, 400, 300, 210,
          144, 100, 73, 50, 36, 26, 18];
          Cyclone1.m                      :=   1.6                         ;

          END
     END

     PRESET

     INITIAL

          STEADY_STATE

     SOLUTIONPARAMETERS

          gPLOT                                        :=   ON          ;
#         Integrator                                   :=   "SRADAU"    ;
          LASolver                                     :=   "MA48"      ;
          BlockDecomposition                           :=   OFF         ;
     SCHEDULE


   CONTINUE FOR     7



END # PROCESS Model
#=============================================================================================#
```