A COMPUTER SIMULATOR FOR BALL MILL GRINDING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

AYŞE YASEMİN YEŞİLAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MINING ENGINEERING

SEPTEMBER 2004

Approval of the Graduate School of Natural and Applied Sciences.

_____

Prof. Dr. Canan Özgen

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. Ümit Atalay

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Prof. Dr. Çetin Hoşten

Supervisor

Examining Committee Members

Prof. Dr. Yavuz Topkaya	(METU, METE)	_____
Prof. Dr. Çetin Hoşten	(METU, MINE)	_____
Prof. Dr. Nurkan Karahanoğlu	(METU, GEOE)	_____
Prof. Dr. Cahit Hiçyılmaz	(METU, MINE)	_____
Dr. N. Metin Can	(Hacettepe Uni., MINE)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Ayşe Yasemin Yeşilay

Signature :

ABSTRACT

A COMPUTER SIMULATOR FOR BALL MILL GRINDING

Yeşilay, Ayşe Yasemin
M.Sc., Department of Mining Engineering
Supervisor: Prof. Dr. Çetin Hoşten

September 2004, 91 pages

Ball mill grinding is an important operation in the processing of most minerals, in that it may be used to produce particles of the required size and shape, to liberate minerals from each other for concentration purposes, and to increase the powder surface area. Grinding of minerals is probably the most energy consuming task and optimization of this operation has vital importance in processing plant operations to achieve the lowest operating costs. Predicting the complete product size distribution, mill specifications and power draw are important parameters of this optimization.

In this study, a computer simulation program is developed in MATLAB environment to simulate grinding operations using the kinetic model in which comminution is considered as a process continuous in time. This type of model is commonly and successfully used for tumbling grinding mills having strongly varying residence time as a function of feed rate.

The program developed, GRINDSIM, is capable of simulating a ball mill for a specified set of model parameters, estimating grinding kinetic parameters from experimental batch grinding data and calculating continuous open and closed-circuit grinding behavior with mill power input. The user interacts with the program through graphical user interfaces (GUI's).

Key words: Simulation, batch grinding, closed circuit grinding, parameter estimation, MATLAB, GUI.

ÖZ

BİLYALI DEĞİRMEN İÇİN BİLGİSAYAR SİMÜLATÖRÜ

Yeşilay, Ayşe Yasemin
Yüksek Lisans, Maden Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Çetin Hoşten

Eylül 2004, 91 sayfa

Boyut küçültme, bir çok cevherin hazırlanmasında istenen boyut ve şekilde taneye ulaşmak, minerallerin serbestleştirilmesinde ve yüzey alanının artırılması amacıyla uygulanan çok önemli bir işlemdir. Enerji tüketimi çok yüksek seviyede olduğundan bu işlemin optimizasyonu ekonomi ve zaman açısından önem taşır. Bu optimizasyonda ürün tane boyu dağılımı, değirmen için gerekli güç ve değirmen boyutlarinin tahmini önemli parametreler olarak sıralanabilir.

Bu çalışmada, kinetik model kullanılarak boyut küçültme işleminin simülasyonuna olanak sağlayan bir bilgisayar programı geliştirilmiştir. Bu model bilyalı değirmenler için yaygın bir şekilde kullanılmaktadır.

Değirmen tasarım ve boyutlandırma işlemlerinin cevher hazırlamada önemli bir yeri vardır. Bu çalışmada, boyut küçültme işlemi çalışılmış, kesikli öğütme, parametre tahmini, açık ve kapalı devre değirmen simülasyonları için algoritmalar verilmiştir.

GRINDSIM kullanarak verilen parametre seti ile öğütme simülasyonu yapılabilir, kapalı ve açık devre simulasyon sonuçları ekrandan alınabilir, model parametreleri tahmin edilebilir. Program, kullanıcı odaklı olarak dizayn edilmiştir.

Anahtar Kelimeler: Simülasyon, kesikli öğütme, kapalı devre öğütme, parametre tahmini, MATLAB, GUI

To My Family...

ACKNOWLEDGEMENTS

I would like to thank the following persons for their contributions and support during the course of this study.

My advisor Prof. Dr. Çetin Hoşten for his kind supervision, support, valuable comments, innumerable suggestions and friendship through all stages of this work.

Prof. Dr. Cahit Hiçyılmaz, Prof. Dr. Nurkan Karahanoğlu, Prof. Dr. Yavuz Topkaya and Dr. N. Metin Can for their valuable suggestions and serving on the MSc Thesis committee.

Selvin Yeşilay, Ercan Örücü, Arman Koçal, Onat Başbay and Özlem Deniz Eratak for their friendship, help and support during various stages of this study.

My friends from the department, Koray Önal, Emre Altun, M. Esenay Hacıosmanoğlu, H. Sinan İnal, Savaş Özün, Osman Sivrikaya, Ahmet Karakaş, Ayşe Alpagut, Reza Osgoi, Muharrem Kılıç, Tuğcan Tuzcu, Mehtap Gülsün, G. Meltem Lüle, Serhat Keleş, Necmettin Çetin and Cemil Acar for their friendship.

Finally, and most deeply, I would like to thank my family for always being there.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The number of developments in computer technology in the last two decades has been substantial. These developments have fostered the growth of intelligent technologies applicable to numerous areas in the mining and mineral processing industry. These include intelligent automation and remote control, modeling and simulation, process control systems, and robotics and technology integration. These areas have contributed to safer, more efficient, and more productive operations.

Comminution is a process step for a wide range of industries including cement, ceramics and minerals. Industrial knowledge shows that most of the energy consumed goes to size reduction processes in mineral processing plants. The grinding operation in a ball mill is a capital and energy intensive process. Hence, a marginal improvement in the efficiency of mill operation will be of immense economic benefit to the industry.

A typical procedure for milling simulation involves performing laboratory experiments in smaller size mills under identical operating conditions to obtain the breakage properties (selection and breakage functions) of a particular ore.

Then, these properties are scaled to larger mills using suitable mathematical models introduced by many workers in the area. In the end, the mill dimensions are computed from the feed and the estimated product size distributions.

It has been shown experimentally that the breakage function does not depend upon the grinding environment and can be normalized with respect to the size (Herbst and Fuerstenau, 1980; Herbst et al., 1981). The laboratory experiments are done with nearly identical feed materials and operating conditions. So the size-normalized values obtained for a particular ore in a laboratory-scale mill can be used for an industrial mill.

This thesis aims to present the design and implementation of an initial version of a MATLAB® toolbox for simulating grinding circuits, which should be initially useful at least for educational purposes.

The basic concepts related with grinding are scanned in Chapter 2. Chapter 3 gives information on the models based on in the preparation of the program, in addition the flowsheets of the program are explained in detail, followed by some example applications and program user information presented in Chapter 5. In the appendices, the code and main windows are presented.

CHAPTER II

BASIC CONCEPTS OF GRINDING IN MINERAL PROCESSING

2.1 Introduction

Quantitative modeling and simulation of grinding circuits has been a major component of the research activity in mineral processing for the last several decades. Application of these models to practical engineering situations is not always easy, and information on suitable models must generally be searched for in the technical research literature. Modern simulators allow the user to combine models for many unit operations into a complete flowsheet and to calculate the expected performance. Simulation provides a useful tool that mineral process engineers can use to identify and solve a variety of operational and design problems.

A simulation of a physical model is a mathematical model which behaves on computation in a manner identical to that of the real process. Generally, a simulation is only an approximation to the real behavior, especially for a process as complicated as for a milling, and the mathematical models can be more or less complex depending on how closely one wishes to simulate the real situation.

As the energy is the most critical cost factor in size reduction, comminution modeling was initially approached in terms of the energy-size reduction

relationships, which were concerned with the relationship between the energy consumed by a crusher or a grinding mill and the amount of size reduction that the consumption of this energy brought about. The energy approach, however, does not provide information on the transformation from feed to product size distribution and cannot accommodate the effects of the operating variables, such as feed rate, mill size and mill speed. The transformation relationship between the feed and product size distributions for a comminution machine is necessary for the optimum design and operation of a size-reduction circuit and requires the formulation of mathematical models based on recognition of physical events occurring in size reduction.

Such an approach to comminution modeling allows the prediction of product characteristics for known feed characteristics and values of the operating and design variables.

2.2 Models of Size Reduction

The phenomenological approach to the modeling of size reduction is based on a mechanistic description of the breakage process coupled with mass (or number) balance on each size interval (Lynch, 1977). The feed-product size distribution transformation in a comminution machine is a consequence of a summation of numerous single breakage events which may operate simultaneously or consecutively or both. During the comminution process a certain proportion of each size is selected for breakage at any repetitive step and a complete range of finer breakage product sizes is produced from the single breakage of just one specified particle size. Thus, the breakage event may be conveniently described by two fundamental concepts:

- probability of each size being selected for breakage;
- characteristic size distribution after breakage.

As most comminution machines operate so that sequential breakage events occur while the particles are in the machines, the selection of the particles for breakage at each event is influenced by the type and operating characteristics of comminution machines, and the extent of size reduction depends on the number of successive breakage events. Thus, Lynch (1977) observed that to describe the complete process in a machine requires the introduction of two more concepts:

- internal classification of particles prior to a breakage event within the process (e.g, preferential breakage of coarser particles);
- residence time within the comminution machine.

The phenomenological models of comminution based on the above concepts fall into two main groups defined by Austin and co-workers (1984):

- *matrix type models*, and
- *kinetic type models.*

In the *matrix model*, comminution is considered as a succession of discrete breakage events (cycles or stages of breakage), the feed to each event being the product from the preceding event. The longer the period of size reduction, the greater is the number of events and the size reduction attained. Time is implicit in the model. The matrix model is preferred for fixed residence time machines like jaw and cone crushers where internal material transport depends on the jaw/cone geometry and movement, being independent of feed rate (Austin, 1984). In the *kinetic model*, comminution is considered as a process continuous in time. Time is explicit in the model; the longer the time of grinding, the greater is the size reduction attained. This type of model is commonly and successfully used for tumbling grinding mills having strongly varying residence time as a function of feed rate. There is quite a similarity between the two types of the models as they are

based on common concepts. However, different symbols and names have been used to represent what are effectively the same parameters or concepts. Since the simulation program developed in this thesis has been based on the kinetic model, thus the basic features of the kinetic model of grinding will be presented here.

2.2.1 Kinetic Model

In the kinetic approach, details given by Austin et. al (1982), used mainly for the accurate formulation of tumbling ball mill grinding models, comminution is considered as a rate process continuous in time. Knowledge of rates at which particles of given sizes break (breakage kinetics) and in what sizes their products appear (breakage distribution) and the residence time of particles in the mill (material transport) are the fundamental concepts required in the kinetic description of size reduction. The description of grinding of a given particle size as a rate process contains two parts:

- the specific rate of breakage (or selection function) of that size; and
- the size distribution of daughter particles produced by the fragmentation of each disappearing particle size (primary breakage distribution function or primary progeny distribution).

A description of continuous tumbling mill grinding requires not only a description of the breakage kinetics but also a mathematical representation of material transport through the mill (residence time distribution, RTD function), because the number of breakage actions received by the material in the mill depends on the time it spends in the mill.

2.2.1.1 The Specific Rate of Breakage (Selection Function)

The specific rate of breakage, s_i, of material within a size interval "i" can be defined as the mass of material of that size broken per unit time per unit mass of material of that size present in the mill (Austin et. al, 1982). In other words, s_i gives the fraction of material in the size interval 'i' which will be selected for breakage per unit time, and has units of time^{-1}.

The selection function, s_i, in the models simply describes the probability of breakage of particles of different sizes. If s_i describes the selection function (i.e., the fractional breakage) occurring in a single stage of breakage, $s_i = S_i/\Delta t$, Δt being the time in the single breakage stage.

Experimental work on the grinding of a $\sqrt{2}$ geometric size interval of many homogeneous materials in a completely mixed dry ball mill has shown that the disappearance with time of the material in this finite size interval appears to follow a first-order law, that is rate of disappearance of size j material due to breakage is proportional to the amount of size j material in the mill (Austin et. al 1982):

Rate of breakage (or disappearance) of size i is

$$\frac{dm_i(t)}{dt} = -s_i m_i(t) \qquad (2.1)$$

where m_i is the mass fraction of size i and t, the grind time.

It has been observed that the selection function size dependence frequently follows a power-law relationship (Herbst and Fuerstenau, 1968; Herbst et al., 1973):

$$s_i = s_1 \left[\frac{\sqrt{x_i x_{i+1}}}{\sqrt{x_1 x_2}} \right]^{\alpha} \qquad (2.2)$$

with a slope α approximately equal to the slope of the cumulative breakage function B_{ij} (on a log scale) versus size (on arithmetic scale) plot in the fine size range; s_1 is the experimentally determined rate parameter of the coarsest size interval, x_1 and x_2 being the upper size limits of the coarsest interval. For dry ball milling of a limestone sample with -8+10 mesh feed size Herbst et al. (1973) reported s_1= 0.780 min^{-1} and α=0.75.

As far as the rates of production of fine material are concerned, the b values appear as:

Rate of production of size i from j = (fraction to i from j). (Rate of breakage of size j)
= b_{ij} S_j $m_j(t)M$

and in the same way,

Rate of production of less-than-size i from breakage of larger size j= B_{ij} S_j $m_j(t)M$

2.2.1.2 The Primary Breakage Distribution Function

Grinding of a given feed produces a whole range of product sizes, down to near zero. After the first breakage occurs, these product sizes are mixed back into the mill charge and will continue to be broken with time again and again. If this distribution of fragments can be measured before any of the fragments are re-selected for further breakage, that is before any further breakage occurs, then the resulting distribution is the primary breakage distribution or primary progeny distribution (Austin, 1982).

The breakage distribution functions, b_{ij}, represent the fraction of material that is broken from size interval j and appears in size interval i after breakage where i<j. In this sense, breakage is described as occurring only when the particles are broken out of their original size range throughout this study. Thus in a $\sqrt{2}$ size interval, the material must be broken below the top size interval to be considered as broken and hence the products of breakage are defined as appearing in sizes less than the lower limit of the top size interval. As an example, when a material of size interval 1 is broken in the weight fraction of the products which then occur in the size interval i is called $b_{i,1}$. The set of numbers $b_{i,1}$, where i ranges from 2 to n, which, apparently describes the distribution of fragments produced from size 1.

The second symbolism convenient for characterizing the primary breakage distribution is to accumulate the b_{ij} values from the finest interval, n, and let B_{ij} be the cumulative mass fraction of material broken from size j which appears in size intervals less than the upper size of size interval i.

$$B_{ij} = \sum_{\substack{k=i \\ i>j}}^{n} b_{kj} \tag{2.3}$$

thus

$$b_{ij} = B_{ij} - B_{i+1,j} \tag{2.4}$$

and, by definition, $B_{j+1,j}=1$, and $b_{n,j} = B_{n,j}$.

Unlike the breakage rate parameter, the breakage distribution function is accepted to be an essential property of the material and independent of the mill variables and the mode of operation (wet or dry), in the normal operating range of milling conditions (Austin et. al, 1982).

The cumulative weight fraction of material broken from size 1 into i is denoted by $B_{i,1}$. The matrix of $b_{i,j}$ and $B_{i,j}$ values are needed for a complete understanding of the size reduction process and breakage actions. The values of B can be obtained from one-size-fraction tests at short grinding times, where approximate corrections to allow for reselection for breakage of the primary fragments are reasonably valid. It is inherent in this technique the values of B do not vary with grinding time. This was proved by Gardner and Austin (1962) by the radio tracing experiments.

It seems to be a complicated task to measure the matrix of B values for all materials, under all milling conditions. However it is often found (Shoji, Lohrasb, and Austin, 1980) that the B values are insensitive to the precise mill conditions, at least in the normal operating range of milling conditions. It is often observed that the breakage distribution functions for homogenous materials can be normalized with respect to the feed size material; that is, the fraction of the starting size is independent of the starting size. A breakage distribution function that can be normalized relates all other distribution functions to the feed size distribution function by

$$b_{ij} = b_{i-j+1,1} \text{ or } B_{ij} = B_{i-j+1,1} \qquad \text{for } j=2,3,4,\ldots n; \text{ and } i=j, j+1,\ldots n \qquad (2.5)$$

so that the number of b_{ij} parameters to be estimated is reduced from (n-1)(n-2)/2 to (n-2) number of feed-size parameters can be calculated by the above normalizing equation.

2.2.1.3 Material Transport

The RTD, defined as the fraction per unit time of feed which leaves the mill after a residence time t, is used to construct mass-size balance equations. Two idealized limiting cases of RTD are the plug flow and perfectly mixed (or back-mixed) flow models. Plug flow implies no forward or backward mixing as the material moves along the mill. A perfectly mixed mill

immediately mixes the elements of feed into the bulk of the mill hold-up. The material flow pattern in real mills is, however, generally characterized by a RTD function intermediate to these two limiting cases, which can be modeled satisfactorily by assuming a series of staged tanks, each with perfect mixing and mixed-product removal. For a continuous mill in which conditions along the mill are virtually constant (a reasonable approximation for an overflow ball mill), the ideal plug flow pattern of the material through the mill for a residence time of τ gives exactly the same equation as batch grinding for time τ (Austin et. al., 1984).

If the mill residence time distribution is represented with a mixers-in-series model, the RTD function in dimensionless time takes the form

$$E(\theta) = \frac{N^N}{(N-1)!}\theta^{N-1}\exp(-N\theta) \tag{2.6}$$

Austin and co-workers (1982) claimed that an equivalent mixing model of one large tank followed by two smaller equal tanks would be favorable as a good approximation for wet overflow ball mills:

$$E(t) = \frac{\tau_1}{(\tau_1-\tau_2)^2}\left[\exp\left(-t/\tau_1\right) - \exp\left(-t/\tau_2\right)\right] - \frac{t}{(\tau_1-\tau_2)\tau_2}\exp\left(-t/\tau_2\right) \tag{2.7}$$

where τ_1 is the mean residence time in the large tank and τ_2 is the mean residence time in each of the smaller tanks, the total mean residence time τ = $\tau_1 + 2\tau_2$.

2.2.1.4 The Batch Grinding Equation

A simple batch test mill is a well mixed container where the amount of mass W held is constant and the size intervals are a geometric mean series and

the starting feed is all in the top size, that is $m_i(0)=1$ and the mill is run for a given time t_1. After time t_1, sample is removed from the mill and the fraction still remaining within the original size interval is determined by sieving and weighing methods to analyze the mill conditions. The remaining sample which still reports to the original size fraction is returned to the mill and ground once more to give a total grinding time of t_2, reanalyzed and so on, until the end (Luckie and Austin, 1972).

This analysis results in a first order law of disappearance of size 1 and the rate of disappearance should be proportional to $m_1(t)$ and M.

$$-\frac{[dm_1(t)M]}{dt} \propto m_1(t)M$$

and since the total mass M is constant for a steady state circuit, this becomes

$$dm_i(t) / dt = - S_1 m_1(t)$$

where S_1 is a constant and called the specific rate of breakage, having units $time^{-1}$. Then, if S does not differ with time,

$$m_1(t)=m_1(0)\exp(-S_1 t)$$

then the equation may be written as

$$\log[m_1(t)] = \log[m_1(0)] - S_1 t/2.3 \qquad (2.8)$$

giving a slope of value $(-S_1/ 2.3)$.

Austin (1982) says that when the breakage rate and distribution function parameters are independent of both the size distribution in the mill and time, as treated in the equation above, the equation expresses linear kinetics. However, such a linear plot does not prove that the finer material

generated by breakage of the original size material will also break in a first-order manner. The build-up of the fines does not affect the specific rate of breakage of the original size, until a reasonable amount of original material exists in the mill. The part of the material in any size interval in a batch mill disappears because of breakage, while other material enters the size interval as a result of breakage of particles in larger size intervals. Thus, using the rate process parameters s_i and b_{ij} for grinding, a complete size-mass balance on the fully mixed batch grinding system can be formulated as a set of n differential equations:

$$\begin{pmatrix}\text{the net rate of production}\\ \text{of the size i material}\end{pmatrix} = \begin{pmatrix}\text{the rate of appearance of size i}\\ \text{from breakage of larger sizes}\end{pmatrix} - \begin{pmatrix}\text{the rate of disappeara nce}\\ \text{of size i by breakage}\end{pmatrix}$$

$$\frac{d\left[Hm_i(t)\right]}{dt} = -s_i Hm_i(t) + \sum_{\substack{j=1\\ i>1}}^{i-1} b_{ij} s_j Hm_j(t) \qquad 1 \le i \le n \tag{2.9}$$

where n is the number of size intervals and H is the total mass of material (hold-up) being ground. It is convenient to use geometric size intervals down to (n-1)-th interval and let the nth interval be a sink containing all finer material. Since material cannot be broken out of the finest n-th interval k_n must always be zero; and $b_{nj} = B_{nj}$. For a batch mill H is constant and, therefore, can be dropped from the equation.

The equation may become non-linear if the breakage rate parameters are influenced by the size distribution in the batch mill which is itself a function of time. The rate parameters (and breakage distribution function parameters as well) may have explicit or implicit dependence on time. The explicit dependence on time implies that the operating conditions are changed as a function of time but the material breakage properties are time invariant. The implicit dependence on time implies that the material properties themselves are changing with time, but yet the process kinetics may be linear.

Herbst et. al. (1973) assumed if s_i and b_{ij} do not vary with time, the batch grinding is described by a set of n linear differential equations with constant coefficients, which can be compactly represented by the matrix equation would be:

$$\frac{d}{dt}\left[\underline{m}(t)\right] = -\left[\underline{\underline{I}} - \underline{\underline{b}}\right]\underline{\underline{s}}\,\underline{m}(t) \tag{2.10}$$

where;

$\underline{m}$ is an nx1 matrix (vector) representing the mass fraction of particles in each size interval in the mill.

$\underline{\underline{s}}$ is an nxn diagonal matrix giving the specific rates of breakage.

$\underline{\underline{b}}$ is an nxn lower triangular matrix with diagonal and upper diagonal elements being all zeroes and the lower diagonal elements being the breakage distribution functions.

$\underline{\underline{I}}$ is the nxn identity matrix.

In practice, the grinding of a $\sqrt{2}$ geometric size interval experimentally gives first-order breakage, but this does not mean that the first-order breakage concept can be applied to a wider size interval or to a very narrow size interval (or a differential size interval in the limit when size is considered as a continuous variable like time). In fact, the crushing strength of an irregularly shaped particle depends on its size, and the magnitude and orientation of the applied force as well. Therefore, it is very unlikely that the strengths of a number of particles all of a certain differential size are the same (Austin et. al., 1984).

The time-independent breakage rate parameter, s_i, depends not only on the material properties but also on grinding mill variables such as the mill size, the filling ratio, the rotational speed, the ball size and the load, etc. The specific rates of breakage are lower for smaller particle sizes because smaller particles are inherently stronger and because it is more difficult to nip unit mass of small sizes between the balls. For a given ball size, specific breakage rate values increase up to a certain size, passing through a maximum, and then start to decrease for coarser particles because large particles are too strong to be broken in the mill (Lynch, 1977).

2.2.1.5 Methods for Estimation of Breakage Rate and Distribution Functions

Batch grinding tests performed by Herbst and Rajamani (1982) with narrow size fractions (one-size fraction method) are found to be the most accurate method for obtaining s_i and b_{ij} values for grinding in laboratory scale mills because there are no effects of RTD or variation of hold-up to complicate the analysis, so long as the power input into the batch test mill remains constant.

The one-size-fraction (single size-interval) consists of grinding a starting material which is predominantly of one size interval (e.g. a $\sqrt{2}$ sieve interval). Alternatively, a fresh original-size feed may be used for each grind time.

Austin and co-workers (1984) say that for the determination of s_i values, only the fraction left in the top size needs to be determined, however to determine b_{ij} values, however, very accurate complete size analysis at short grind times is necessary. A complete study on a given material under a given set of conditions normally involves measuring s_i values for three or four starting sizes, getting the size distribution at short times to enable b_{ij} values to be calculated for each of these sizes, and determining the

complete family of size distributions resulting from one or two of the starting sizes.

If the kinetics of disappearance of the single size-interval feed follows the first-order grinding hypothesis, a plot of $m_i(t)$ on a log scale vs. t on a linear scale should give a straight line. It is not correct to force the line to pass through $m_1(0)=1$ at zero time as there may be an incomplete-sieving error with the original-size feed, which should be determined by a blank sieving test before grinding tests. The value of s_i is determined from the slope of the plot, as seen from Figure 2.1 (Austin et. al. 1984).

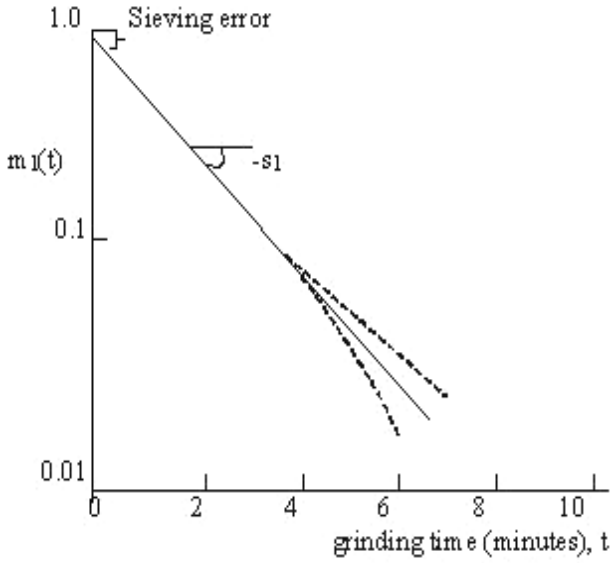

Figure 2.1 Determination of s_1 value

According to Austin and co-workers (1984) non-first-order effects (abnormal breakage) may arise either due to change in properties of the material being ground or the change of grinding environment in the mill, such as cushioning action and altered mechanics of the mill action as fines increase in the mill. It is also likely that the starting material can contain abnormally weak particles due to flaws introduce during feed preparation. In this case,

it has been suggested to precondition the sized feed in a ball mill for a very short grind time and re-sieve to eliminate the broken particles.

The method proposed by Austin and co-workers (1984) say that the best results for B values are obtained when 20 to 30% of the top size material is broken out of their original size. Assuming there is no rebreakage of fragments, then BI method is formulized as:

$$B_{i,1} = \frac{P_i(t) - P_i(0)}{P_2(t) - P_2(0)}, \quad i > 1 \tag{2.11}$$

where $P_i(t)$ is the product amount for size i at time t.

An alternative and a more precise method which uses the compensation condition to correct rebreakage of fragments, BII method based on the batch grinding equation.

For the top size assuming that the compensation conditions applied approximately

$$1 - P_i(t) \cong [1 - P_i(0)]\exp(-B_{i,1}s_1 t) \qquad i > 1$$

For the top size interval, first-order breakage gives

$$1 - P_2(t) = [1 - P_2(0)]\exp(-s_1 t), \text{ since } B_{21} = 1$$

Then

$$-s_1 t = \ln\left[(1 - P_2(t)) \Big/ (1 - P_2(0))\right]$$

$$-B_{i1}s_1 t \cong \ln\left[(1 - P_i(t)) \Big/ (1 - P_i(0))\right]$$

$$B_{i1} \cong \frac{\log\left[\left(1 - P_i(0)\right) / \left(1 - P_i(t)\right)\right]}{\log\left[\left(1 - P_2(0)\right) / \left(1 - P_2(t)\right)\right]} \qquad i > 1 \text{ Method BII} \qquad (2.12)$$

2.3 The Hydrocyclone Classifier

Hydrocyclones are widely used in grinding circuits to remove the unbroken and large particles from the mill product and recycle. The main operating and design variables of a hydrocyclone are the inlet and outlet dimensions, cyclone size, water percent of feed, size distribution, cyclone operating pressure and flow rate of feed solids.

The type of a hydrocyclone model necessary for simulation is that which enables the size distribution and flow rate of the particles in the product streams to be predicted for any change in the feed stream or cyclone parameters. The two major mechanisms which particles undergo are entrainment and classification. The method for model development involves describing these mechanisms by equations which include all operating variables as parameters (Lynch and Rao, 1975).

The hydrocyclone depends on external power for operation, most commonly a continuous flow centrifugal pump, or occasionally gravity feed systems. The feed slurry is introduced near the top of a hydrocyclone, tangential to the periphery. Feed velocity head and pressure head are converted to both angular and linear acceleration, creating a cyclone effect where the angular acceleration increases as the free liquid moves from the outside wall of the cyclone toward the axis of rotation (Wills, 1992).

The separation size of a hydrocyclone is expressed in terms of the d_{50} size. The d_{50} or cut size is that size which has equal (50%) chance of reporting to

either product, overflow or underflow, from the classifier. The d_{50} size is a measure of the separating forces operative on the particles in the cyclone. There are a number of semi-empirical relationships for d_{50} in the literature. One of the widely used is Dahlstrom's equation (1988); a modified form of the equation applicable for slurry densities up to 35% solids by volume is presented below:

$$d_{50} = 3x10 \frac{(D_o D_i)^{0.68}}{Q^{0.53} \sqrt{(\rho_p - \rho_{sl})}} \sqrt{\frac{\mu_{sl}}{\mu_w}} \tag{2.13}$$

where Q is the volumetric feed rate, D_o and D_i are the diameters of vortex finder and feed inlet of the cyclone; ρ_{sl} and μ_{sl} are apparent density and apparent viscosity of slurry in the cyclone, respectively.

The linear velocity of the liquid and solid particles in the cyclone resolves itself into three components as a result of the shape of the vessel and the position of the discharged parts:

- a radial inward velocity component V_R commencing at the cone periphery,
- a downward vertical (axial) velocity component V_V,
- a tangential velocity component V_T.

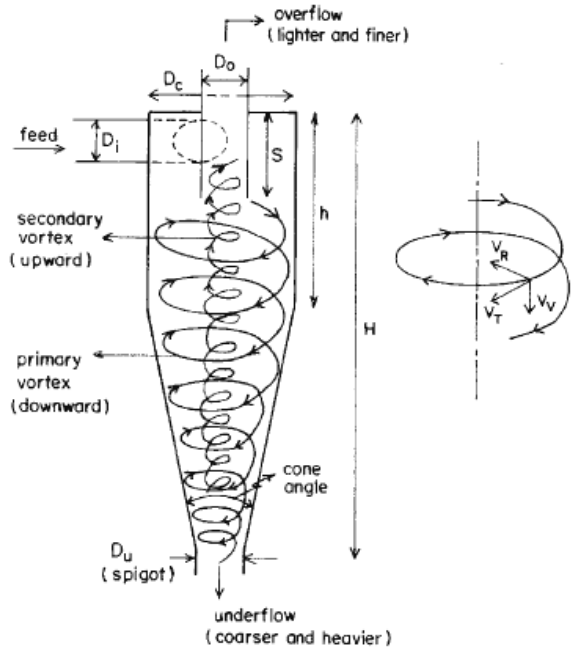

Figure 2.2 The Hydrocyclone (after Hoşten, Ç., 2004)

A particle in a hydrocyclone moves toward the wall of the cyclone if the drag force on it is smaller than the centrifugal force corresponding tangential velocity component; otherwise tends to move radially inward until it reaches a conical envelope (equilibrium orbit) on which drag force is equal to centrifugal force. The particle remains on the envelope and reports to underflow if its envelope is situated outside the zero-vertical-velocity envelope (Figure 2.3), and to overflow if its equilibrium orbit is situated inside the zero-vertical velocity envelope.

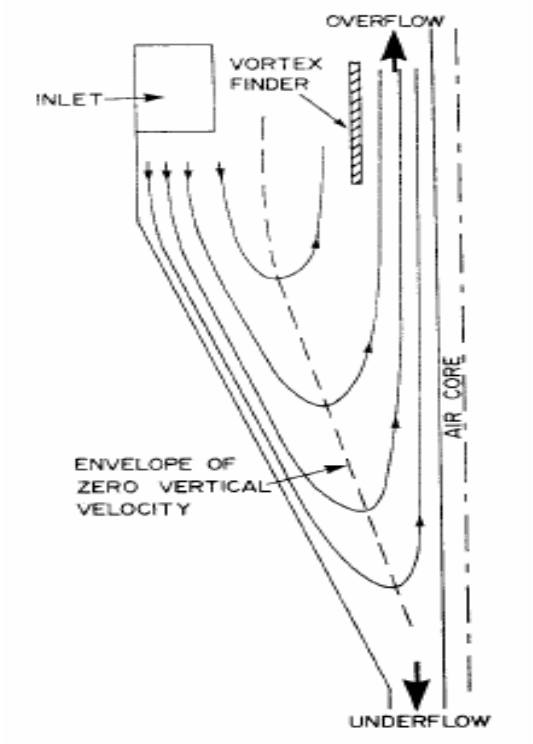

Figure 2.3 Zero Vertical Velocity (after Hoşten, Ç., 2004)

2.3.1 Classifier Efficiency

The description of the performance of a classifier is provided by the classification function, also known as the partition function, and, when shown graphically, as the partition curve. The classification function defines the probability that an individual particle will report to the screen oversize (or undersize) product. This function is a set of efficiency values, or partition coefficients**,** each coefficient referring to a size range, rather than a single efficiency value based on the total amount of misplaced or correctly placed particles in the product streams. Oversize partition coefficients, p_i, are defined as the ratio of the mass of particles in the i-th size interval of the overflow stream to the mass of particles in the same interval of the feed stream in fractions,

$$p_i = \frac{M_o \times m_{oi}}{M_f \times m_{fi}} \tag{2.14}$$

where m_{oi} and m_{fi} are the mass fractions in the i-th size interval for the overflow and feed streams, respectively. M_f and M_o are the mass flow rate of the solids in cyclone feed and overflow respectively. The undersize partition coefficient is $(1-p_i)$. The maximum value of p_i is unity and minimum value is zero.

When the partition coefficients are plotted against particle size or normalized particle size, an S-shaped curve is typical of all practical classifiers, and particularly of hydrocyclones (Figure 2.4).

The classification function is a complicated function of the particle properties and the classifying or separating action of the particular device under consideration. If an average partition curve (or its mathematical formulation) for a separation unit, such as a certain type of screen or a classifier, and the cut size d_{50} are known, the curve or its equation can be used for figuring out the partition coefficients for all the size fractions of the feed to the screen or the classifier. Then, the size distribution in either overflow or underflow of the screen or classifier can be calculated by a simple mass balance over the solids in the i-th size interval (Lynch and Rao, 1975).

Classification in any type of classifiers is of probabilistic nature. It therefore follows that classification is not sharp and some coarse material must be accepted with the fine product and some fine material with the coarse product. The non-zero intercept, say y, on the partition axis at zero size of the actual partition curve in Figure 2.4 is a characteristic of almost all practical classifiers, and is due to short-circuiting of the fine particles into the coarse product stream (underflow). In the hydrocyclone, this is due to

the water carrying fine particles into the boundary layer on the outer wall of the conical section and discharging them with the underflow.

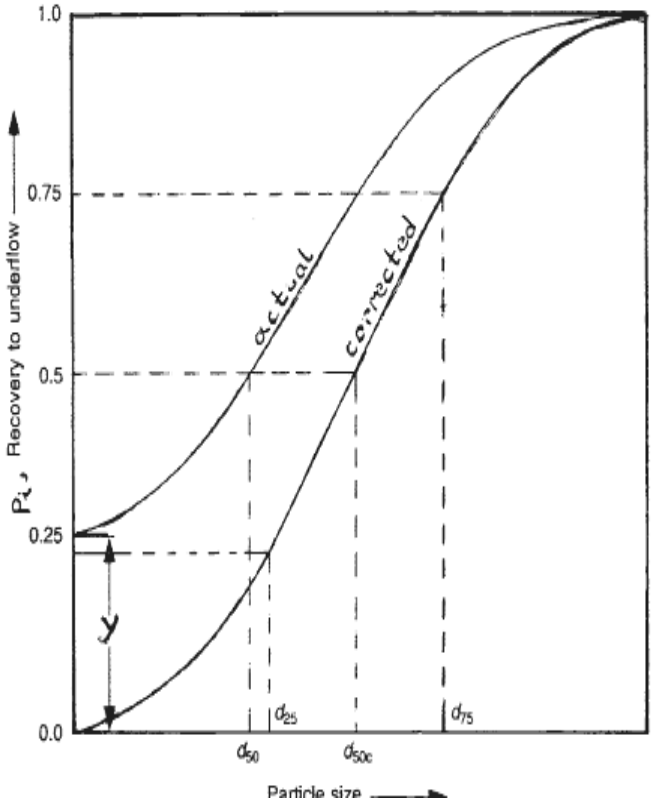

Figure 2.4 Typical Partition Curves for a Hydrocyclone (after Hoşten, Ç., 2004)

The short circuiting of the coarse particles to fine product (overflow) is generally negligibly small. The fraction y is estimated as the fraction of feed water leaving through the underflow:

$$y = \frac{M_{Lu}}{M_{Lf}} \tag{2.15}$$

where M_{Lu} and M_{Lf} are the mass flow rate of liquid (water) in underflow and feed, respectively.

Then the corrected efficiency curve may be constructed by applying the following relationship to each size interval of feed solid particles represents the separation in a correct way:

$$p_{ic} = \frac{p_i - y}{1 - y} \tag{2.16}$$

where p_{ic} is the corrected partition coefficient. The corrected efficiency curve constructed with the use of corrected partition coefficients gives the result of the true classification, eliminating the short-circuiting or bypass effect during classification.

CHAPTER III

GENERAL STRUCTURE AND BASIS OF GRINDSIM

3.1 Introduction

In this study, a simulation program called GRINDSIM has been developed in the scope of simulating grinding circuits using the formerly developed models. The program is capable of simulating a tumbling mill for a specified set of model parameters, estimating the model parameters from experimental data and calculating open and closed circuit grinding product. The program is developed in MATLAB programming environment in a user friendly way, the user can estimate parameters, simulate a mill for batch grinding and steady-state grinding in various circuit arrangements, i.e. open, standard closed or reverse closed circuits, through graphical user interface windows.

3.2 Mathematical Models Used In the Program

The program developed for the simulation of grinding circuits and parameter estimation is set up according to the existing models of various workers in the area.

The simulation results could both be viewed on the screen through graphs and edit boxes and saved, or printed via a printer. Some examples of the simulator run are presented, illustrating the capabilities of the simulator at the end of this chapter. Typical output of the toolbox GRINDSIM consists of a total water and solids balance; size distributions, 80% passing values, graphical outputs and, in particular cases estimated parameters.

3.2.1 Batch Grinding Model

The set of time-continuous, discrete-size batch/plug flow equations was solved first by Reid (1965) for the case of time-independent s_i and b_{ij} values. For the solution of batch grinding equation, Reid's (1965) method has been widely used for its simplicity and popularity, The Reid solution uses recursive integration of the grinding equation to obtain a simple computational form that does not require any iterative procedure or matrix multiplication.

A very simple and straightforward solution to cumulative batch grinding equation for first order breakage has been proposed by Das et. al. (1995). The data required for solution necessitates the parameters for the selection and breakage functions together with cumulative weight fraction of particles and upper size limits. The experimental size distribution data on grinding are commonly described and presented in cumulative form, as in the Rosin-Rammler plot; thus, model results in cumulative form are desirable. The equations are formed under the assumption that the breakage is first order and the selection function does not vary with time (Luckie and Austin, 1972).

The discretized form of equation modified by Das(1995), for the cumulative weight fraction of particles less than or equal to size x_i, where the feed has been represented with n size fractions,

$$\frac{dF_i}{dt} = \sum_{j=i+1}^{n} S(x_j) B(x_i, x_1)(F_j - F_{j-1}) \tag{3.1}$$

Here the fraction of particles of size x_i, which are the daughter particles of size x_j, are represented by $B(x_i,x_j)$; $S(x_j)$ symbolize the breakage rate of particles of size x_j. The n-th size corresponds to the largest size in this form of solution, unlike the usual convention in this thesis, thus F_n equals to unity. The selection and breakage functions are computed from Austin's (1999) equations:

$$S = S_1 \left(\frac{x_i}{x_{ref}} \right)^{\alpha} \tag{3.2}$$

where S_1 is the reference S value, x_i is a row vector containing upper size limits of feed, x_{ref} is a user-defined reference size and α is selection function constant.

and the breakage function can be obtained from,

$$B_{i,j} = \phi(x_i / x_j)^{\gamma} + (1 - \phi)(x_i / x_j)^{\delta} \tag{3.3}$$

where B_{ij} is the cumulative fraction of progenies of size less than or equal to upper size limit of the i-th size interval produced from the breakage of the particles from the j-th parent size interval; x_i is a column vector whose elements are the upper size limits of size intervals; φ(phi), γ(gamma) and β(beta) are the parameters of the functional form for the cumulative breakage function values.

The matrix form of Equation 3.1 is given by

$$\frac{d\underline{F}}{dt} = \underline{\underline{A}}\underline{F} + \underline{C} \tag{3.4}$$

where $\underline{F}$ is a vector containing cumulative mass fraction of feed material, and $\underline{\underline{A}}$ is given as,

$$A_{ij} = \begin{cases} S_j B_{i,j} - S_{j+1} B_{i,j+1} & i < j \\ -S_{j+1} B_{i,j+1} & i = j \\ 0 & i > j \end{cases}$$

and the column vector $\underline{C}$,

$$C_{i,1} = S_n B_{i,n}$$

The matrix $\underline{\underline{A}}$ is upper triangular. Das et al. (1995) used spectral theorem to solve the Equation 3.7, using the eigen values of $\underline{\underline{A}}$, which are symbolically represented as,

$\lambda_r = A_{r,r}$ r=1,2,3,…,n-1

and if $\underline{x}^r$ and $\underline{y}^r$ are the respective eigenvectors of $\underline{\underline{A}}$ and $\underline{\underline{A}}^T$ corresponding to the common eigenvalue λ_r, the solution of Equation 3.4 becomes:

$$\underline{F}(t) = \sum_{i=1}^{n-1} \left\{ \exp(\lambda_i t) \frac{\left\langle \underline{F}_0, \underline{y}^i \right\rangle}{\left\langle \underline{x}^i, \underline{y}^i \right\rangle} - \frac{\left\langle \underline{C}, \underline{y}^i \right\rangle}{\lambda_i \left\langle \underline{x}^i, \underline{y}^i \right\rangle} \left[1 - \exp(\lambda_i t) \right] \right\} \underline{x}^i \tag{3.5}$$

This solution is used in the batch grinding model in GRINDSIM to get batch grinding cumulative solution and plots. The algorithm for this stage of the program is as follows:

1. Compute selection and breakage functions using the related parameters.

2. Compute $\underline{\underline{A}}$ and $\underline{\underline{A}}^T$ values.

3. Find the eigen values of $\underline{\underline{A}}$ and $\underline{\underline{A}}^T$.

4. Solve Equation 3.5 to get F(t) values.

5. Plot mill feed and product size distributions

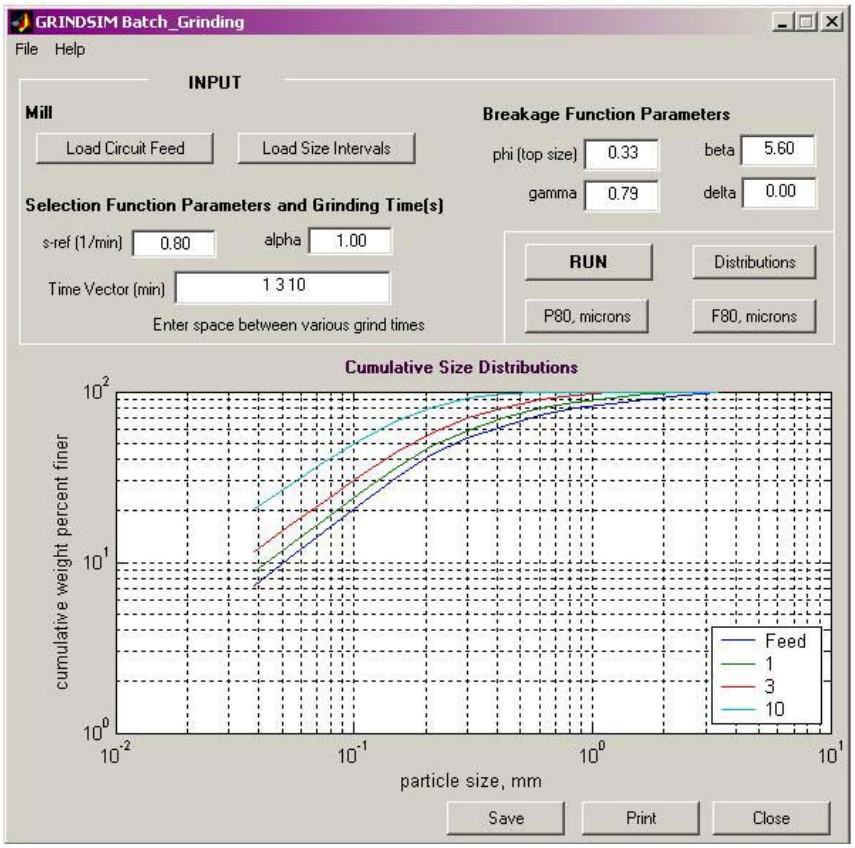

Figure 3.1 Batch Grinding Simulation Sample Run

3.2.2 Parameter Estimation Model

The common procedure for back-calculation of the parameters relies on minimizing the error between the experimental and predicted differential weight fractions for different size intervals and presenting the results in cumulative size distribution. Rajamani and Herbst (1984), Klimpel and Austin(1977), and Das(2001) has studied the methods to estimate mill selection and breakage distribution parameters. These parameters play a vital role in predicting the entire particle size distribution, and accurate values are quite important in design, scale-up and simulation of grinding circuits. The practical incentive for computer simulation of selection function parameters is to shorten the lengthy standard procedure and to perform no more than 2 or 3 experiments. Direct determination of the selection function parameters is possible with a single short time grinding of a mono size feed (Hoşten and Avşar, 2004).

In this option of the program, the parameter estimation calls the MATLAB's minimization function 'fminsearch' for the estimation of grinding kinetic parameters using the cumulative-basis batch grinding model and a set of experimental data for various grind times. The objective function to be minimized is the (mean) sum of squares of the relative deviations between the model-computed and experimental product size distributions of cumulative basis. The maximum iteration number is set to 4000 for this case. The initial estimates of the parameters are input to the algorithm, assuming the same feed size distributions for all tests.

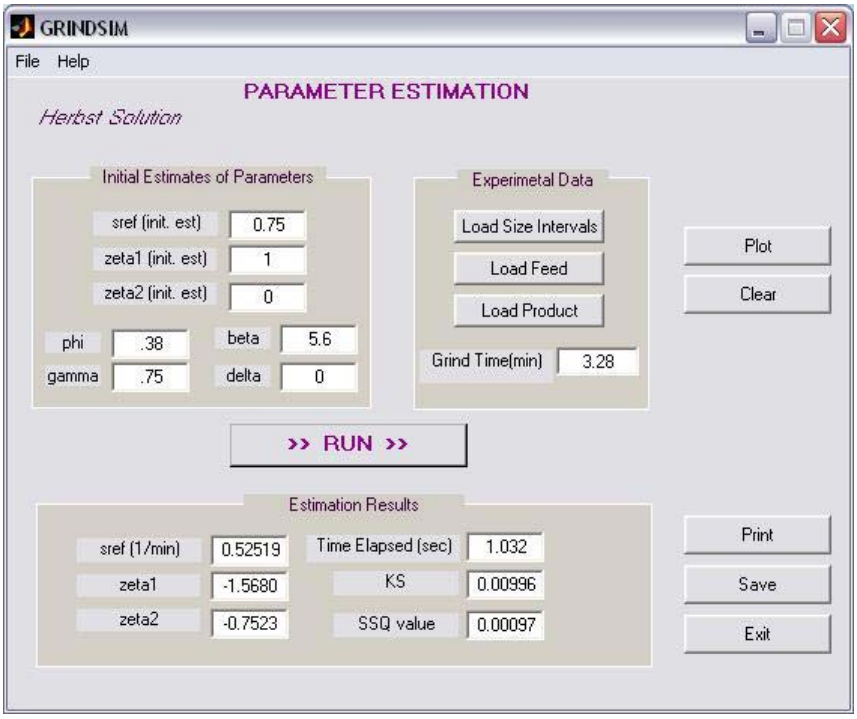

Figure 3.2 Parameter Estimation Sample Run

3.2.3 Normal and Reverse Closed Circuit Models

A procedure which has been proposed for commercial mill design using the population balance approach and specific mill power information is given by Herbst and Fuerstenau (1980), the closed circuit grinding models are formed under their studies.

Milling model calculations require selection and breakage function values, each giving characteristics of the material to be ground. The selection function, S_i, may be calculated from the following equation:

$$S_i = S_1 \exp\left[-\sum_{j=1}^{j} \xi_j \ln(\sqrt{x_i x_{i+1}} / \sqrt{x_1 x_2})\right] \qquad \text{j=1,2,3…} \qquad (3.6)$$

where S_1 and ξ_j are selection function parameters and x_i are upper size limits of feed size intervals.

And the breakage functions can be specified with the following equation,

$$B_{ij} = \alpha_1 (x_i / x_{j+1})^{\alpha_2} + (1-\alpha_1)(x_i / x_{j+1})^{\alpha_3} \tag{3.7}$$

Here, B_{ij} is the cumulative fraction of progenies of size less than or equal to upper size limit of the i-th size interval produced from the breakage of the particles from the j-th parent size interval and, α_1, α_2 and α_3 are parameters of the breakage function.

Herbst and Rajamani (1982) applied the specific selection function hypothesis (Herbst and Fuerstenau, 1973; Herbst et al., 1973) for scaling up the selection function. The selection function is proportional to the mass-specific power input to the mill (Equation 3.8). The constant of proportionality is defined as the ‘‘specific selection function’’. Therefore, knowing the required mill capacity and the power consumption, one can calculate the selection function of the mill using their equation.

The size discretized breakage rate functions, S_i, are proportional to the specific power input to the mill:

$$\mathrm{S_i} = \mathrm{S_i^E}\left(\frac{\mathrm{P}}{\mathrm{H}}\right) \tag{3.8}$$

where the set of proportionality constants, S_i^E, (specific selection function) is independent of mill operating conditions and of mill size; P is the net power input to the mill (energy/time); H is the mass hold-up of material in the mill. Initial estimates of S_i^E can be predicted from S_i values calculated from the runs of the reference mill at varying P/H levels. These initial

estimates can be improved by the use of appropriate non-linear regression algorithms (Herbst et. al, 1973).

The breakage distribution functions (b_{ij}) can be taken as invariant, that is, they are independent of mill design and operating variables. The set of numbers b_{ij} represent the fraction of the breakage product of size interval j which appears in size interval i after one single breakage event, where i is a smaller size than that of j. Hence, breakage is defined as occurring only when particles are broken out of their original size. By definition, $\sum_{i=j+1}^{n} b_{ij} = 1$ hence, the elements of the last row of the matrix (i=n), except for the zero diagonal element, may be calculated by the following set of equations:

$$b_{nj} = 1 - \sum_{k=j+1}^{n-1} b_{k,j} \quad \text{for } j = 1,2,.....,n-1 \tag{3.9}$$

Substitution of breakage rate-power input relationship into the batch grinding model equation yields:

$$\frac{dm_i(\bar{E})}{d\bar{E}} = -s_i^E m_i(\bar{E}) + \sum_{j=1}^{i-1} b_{ij} s_j^E m_j(\bar{E}) \tag{3.10}$$

where $\bar{E}$ is the energy input per ton of ore given by,

$$\bar{E} = \frac{P}{H} \tag{3.11}$$

where P is the net power draw of the mill and H is the mass hold-up of the mill.

Similarly, continuous milling model equations can also be transformed, in which case τ, the mean residence time, is replaced by $\bar{E}$:

$$\bar{E} = \frac{P}{M_{MF}(1+CLR)} \tag{3.12}$$

where M_{MF} is the mass flow rate of the mill feed and CLR is the related circulating load ratio. The form of the solution to the continuous model remains the same with τ replaced by $\bar{E}$, and continuous mill product distributions can be obtained for different energy inputs with the transformed equation.

Assuming the breakage kinetics in the ball mill is linear, the batch response for n size intervals and fractions is given by Herbst and Rajamani (1980):

$$\underline{m}(t) = \underline{\underline{T}}\,\underline{\underline{J}}\,\underline{\underline{T}}^{-1}\,\underline{m}(0) \tag{3.13}$$

in which

$$T_{ij} = \begin{cases} 0, & i < j \\ 1, & i = j \\ \sum_{k=j}^{i-1} \frac{b_{ik} s_k}{s_i - s_j} T_{kj}, & i > j \end{cases}$$

$$J_{ij} = \begin{cases} \exp(-s_i t), & i = j \\ 0, & i \neq j \end{cases} \tag{3.14}$$

Where s_i is the i-th element of selection function and b_{ik} is the individual breakage function of size k which appears in size i after breakage and the size discretized selection functions, T and J are transfer functions. If the user inputs number of mixers (N) in series to represent the commercial mill's RTD (Residence Time Distribution), then we do not need to know the time value, and the J matrix is calculated from:

$$J_{i,i} = \left(1 + \frac{S_i^E P}{M_{MF} N}\right)^{-N}$$

and introducing $\bar{\bar{E}} = \frac{P}{H}$, the equation becomes

$$J_{i,i} = \left(1 + \frac{S_i^E \bar{\bar{E}}}{N}\right)^{-N} \tag{3.15}$$

For closed circuit grinding the mill discharge rate is equal to the circuit product rate and fresh feed rate equal to the mill feed rate, thus

$$\underline{m}_p = \underline{\underline{T}} \underline{\underline{J}}_c(\tau) \underline{\underline{T}}^{-1} \underline{m}_f \tag{3.16}$$

where m_f and m_p are feed and product size distributions, respectively.
A material balance around the classifier for closed circuit grinding and introducing L for $\underline{\underline{T}} \underline{\underline{J}}_c(\tau) \underline{\underline{T}}^{-1}$ yields:

$$\underline{m}_p = \left[\underline{\underline{I}} - \underline{\underline{C}}\right] \underline{\underline{L}} \left[\underline{\underline{I}} - \underline{\underline{C}} \underline{\underline{L}}\right]^{-1} \underline{m}_f \tag{3.17}$$

and for reverse closed circuit

$$\underline{m}_p = \left[\underline{\underline{I}} - \underline{\underline{C}}\right] \left[(\underline{\underline{I}} - \underline{\underline{L}} \underline{\underline{C}})^{-1} \underline{\underline{L}} \underline{\underline{C}} + \underline{\underline{I}}\right] \underline{m}_f \tag{3.18}$$

The identity matrix I is of size nxn and C is the classification matrix, obtained from classifier model and will be discussed in the following sections.

The following summarizes the numerical procedure, which is an iterative algorithm to compute product size distribution for a given feed under given operating conditions.

1. Calculate the necessary energy input to the mill according to

$$\overline{\mathrm{E}} = \frac{\mathrm{P}}{\mathrm{M}_{\mathrm{MF}}(1 + CLR)}$$

2. Compute selection and breakage functions, according to Equations 3.6 and 3.7.

3. Compute the transformation matrix $\underline{\underline{T}}\underline{\underline{J}}_c(\tau)\underline{\underline{T}}^{-1}$ with the Equation 3.14.

4. Form the classifier matrix C, according to the classifier model equations presented in part 3.2.6 of this chapter.

5. Compute the product size distribution with the Equations 3.17 or 3.18, i.e. for normal or reverse closed circuit configurations, respectively.

6. Find cyclone feed, CF, by adding circulating load to the mill fresh feed.

7. Making solids balance around the classifier necessitates finding the circuit product for n size fractions, which are presented by Herbst and Rajamani (1982).

   for normal closed and closed circuits;

$$MP = \frac{M_{MF}}{eL(I - LC)^{-1} w_i}$$

where MP refers to circuit product, e is 1xn identity matrix, I is nxn identity matrix, L is transformation matrix, C is classifier matrix and w_i is the initial fresh feed size distribution.

8. Compute circulating load CL, by subtracting circuit product MP from cyclone feed CF.

9. Compare circuit product with mill fresh feed rate, if the two values are not equal, start from the beginning until circuit product rate converges the fresh feed solids rate by 0.1%.

10. Find cumulative circuit product size distribution found in step 5.

11. Compute 80% passing feed and product sizes.

12. Plot the cyclone efficiency curve and cumulative size distributions of feed and the product.

13. Find water to be added from mill design and classifier equations. (see sections 3.2.5 and 3.2.6)

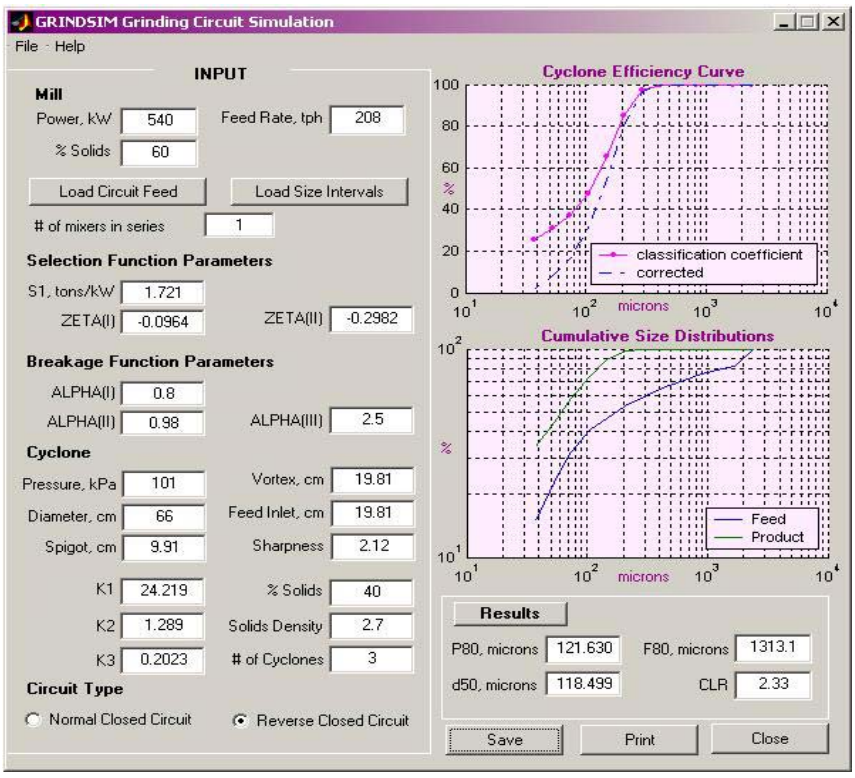

Figure 3.3 Closed Circuit Simulation Sample Run

3.2.4 Open Circuit Grinding Model

Open circuit model is set up on the basis of Herbst and co-workers (1973) model. In the algorithm a known amount of feed enters the mill to give a product size distribution and 80% passing value.

The algorithm of the program is as follows:

1. Calculate the specific energy input to the mill according to:

$$\overline{E} = \frac{P}{M_{MF}}$$

2. Compute selection and breakage functions using equations 3.6 and 3.7.

3. Compute the mill transformation matrix L using $\underline{\underline{L}} = \underline{\underline{T}}\,\underline{\underline{J}}_c(\tau)\underline{\underline{T}}^{-1}$.

4. Find the fractional product size distribution using equation 3.13.

5. Convert the product size distribution to cumulative form.

6. Find the 80% passing size of feed and product of the mill.

7. Plot the results.

3.2.5 Mill Modeling and Scale-Up

Herbst and Rajamani (1981), proposed a procedure for commercial mill design using the population balance approach and specific power information. The models developed in recent years are quite useful and the success of these models can be attributed largely to the fact that each of the major grinding circuit sub-processes, i.e. material breakage and fragment redistribution, material transport and classification, is represented in the explicit fashion in the descriptive equation, this fact helps us to apply these models in simulation applications.

The linear, size discretized, model for mineral breakage kinetics is obtained by dividing the complete feed being ground into n size fractions and making a mass balance at each step of simulation. When the selection and breakage function parameters are independent of the size consist in the mill and time, the kinetic model is linear with constant coefficients (Austin et. al. 1984).

The open circuit mill design calculation necessitates a direct search for the correct energy input that would produce the required product size. The mill

model equations of Herbst and Rajamani (1982) are solved for two energy input levels. In the first iteration, the initial estimate for energy input can be found from Bond Equation (1952, 1960) given by:

$$\overline{E} = 10WI(1/\sqrt{d_{80}^{product}} - 1/\sqrt{d_{80}^{feed}} \tag{3.19}$$

where WI is the work index of the feed material; $d_{80}^{product}$ is the 80 percent passing size of the cyclone overflow stream and d_{80}^{feed} is the 80 percent passing size of fresh feed. The algorithm uses $\overline{E} \pm 0.5$ energy input values, then a simulation is done with the interpolated value of products found from these two energy inputs.

The closed circuit mill design has the objective of achieving a parallelism between mill and cyclone models. Here, the mill model equations and hydrocyclone model equations are solved successively until the mill solids feed rate equals the solids feed rate to the cyclone. The algorithm here also searches for the correct energy input that produces the desired circulating load and required water addition to the circuit that produces the desired product size distribution as well. In the first iteration, the solids feed rate to the cyclone is set to meet the desired mill circulating load. The computations and the algorithm followed in this section of the program are rather complex and force iterative loops to reach the accurate solution.
In open circuit mill design, the algorithm is as follows:

1. Calculate $\overline{E}$ and $\overline{E} \pm 0.5$ values from Equation 3.19.

2. Solve open circuit grinding model equations using Equation 3.15, which involves the knowledge of mill transformation matrix L, described by $\underline{\underline{L}} = \underline{\underline{T}}\underline{\underline{J}}_c(\tau)\underline{\underline{T}}^{-1}$ for two input E values.

3. Find the correct energy input value $\overline{E}^*$ by making an interpolation of $\log d_{80}^{product}$ versus $\log \overline{E}$ for $\overline{E}_1$ and $\overline{E}_2$ cases so as to achieve the desired $\log d_{80}^{product}$.

4. Solve the grinding model equations again with the interpolated value of energy input $\overline{E}^*$. If the resulting product size is not in the convergence limit of 0.1 step back to the third step of this algorithm.

5. After finding the correct energy input level $\overline{E}^*$ compute mill power draw with the equation

$$P = \overline{E}^* \times M_{FF}$$

6. From mill power calculate the mill dimensions from:

$$P_{net} = 2.70\rho_{balls}(L/D)D^{3.3}M_B^*(3.2 - 3M_B^*) \times (1 - 0.1/2^{9-10N^*}) + S_s \quad (3.20)$$

and

$$S_s = 0.393\left(B - \frac{3D}{2O}\right)D^3\left(\frac{L}{D}\right)\rho_{balls}M_B^* \quad (3.21)$$

where ρ_{balls} is the bulk density of ball charge, D is the mill diameter, N* is the fraction of critical speed, M_B^* fraction of mill volume occupied by balls and B is the ball size. S_s is the ball size correction factor.

For the closed circuit mill design, the algorithm followed in the program is as follows:

1. Compute a water feed rate to the sump, WF, to make up a prespecified percent solids in cyclone feed using the equation

$$WF_{j+1} = WF + k_{WF}(d_{80}^{simulation} - d_{80}^{desired})$$

where j is the iteration number, k_{WF} is a specified gain value, and for the first iteration solids feed rate is taken as the rate corresponding to the desired mill re-circulating load.

2. Calculate $\overline{E} \pm 0.5$ values from Equation 3.19.

3. Compute classifier matrix, $\underline{\underline{C}}$ from hydrocyclone model equations.

4. Solve closed circuit model equations using $\overline{E}_1$ and $\overline{E}_2$ values.

5. Find the correct energy input value $\overline{E}^*$ by making an interpolation of $\log d_{80}^{product}$ versus $\log \overline{E}$ for $\overline{E}_1$ and $\overline{E}_2$ cases so as to achieve the desired $\log d_{80}^{product}$.

6. Interpolate between the computed circulating loads from step 4 for the energy input $\overline{E}^*$.

7. Solve closed circuit grinding model equations again with interpolated $\overline{E}^*$, if the resulting circulating load is not within the 1% convergence limit, repeat steps 5 and 6.

8. Find 80% passing size from cyclone overflow product size distribution, if this size is not within the convergence limit, say 2μm of desired product size, adjust WF from the first step of this algorithm, if convergence limit is achieved skip the next step.

9. If convergence limit is not achieved, obtain two new energy input levels 0.5 below and above $\overline{E}*$ value approached before.

10. Repeat steps 3 to 8, until the conditions are satisfied.

11. With the correct energy input value, $\overline{E}*$, compute the mill power draw using modified form of the equation 3.12:

$$P = \overline{E}* \times M_{FF} \times (1 + CL)$$

12. From mill power calculate the mill dimensions from Equations 3.20 and 3.21.

3.2.6 Classifier Model

The model used for hydrocyclone consists of equations given by Lynch and Rao (1975) and Lynch (1977), the computation of classification matrix is the main objective of the algorithm.

The algorithm for the program is as follows:

1. Find the specific gravity of classifier feed slurry with the equation:

$$\rho_s = \frac{1}{\left[1 - \frac{(S_s - 1) \times (\%solidsfeed)}{S_s \times 100}\right]} \tag{3.22}$$

where S_s refers to solids specific gravity.

2. Calculate the flow rate of slurry to the cyclone with the equation:

$$Q = K_1 D_o (\Delta P)^{0.5} (PWF)^{0.25} \quad (3.23)$$

in which K_1 is a cyclone constant, D_0 is the vortex finder diameter, P is the cyclone operating pressure and PWF is the percent water in the cyclone feed.

3. Calculate the amount of classifier feed slurry to a cyclone

$$\text{TPHFEED} = Q \times 60\,\text{min}/hr \times 0.001 m^3/liter \times feedslurrysp.gr.$$

4. Calculate the classifier feed water rate

$$WF = TPHFEED\left[1 - 0.01 \times (1 - PWF)\right]$$

5. Compute the corrected cut size, d_{50c} value from :

$$\log(d_{50c}) = (0.02 \times PSF - 0.083 \times D_u + 0.016 \times D_o \times 0.00076 \times TPHFEED + K_2$$

where PSF is the solids content of the cyclone feed as a percentage, D_u and D_o are spigot and vortex finder diameters, respectively and K_2 is the cyclone operating constant. Parameters K_2 and K_3 have been found to be sensitive to the mill size distribution by Lynch and Rao (1975).

6. The fraction of feed water to underflow, R_f is calculated from

$$R_f = \frac{1.93 D_u - 2.72}{WF} + K_3 \quad (3.24)$$

where K_3 is a cyclone constant.

7. Calculate the corrected partition coefficient from

$$P_{ic} = 1 - \exp\left[-0.693 \left(\frac{d_i}{d_{50c}} \right) \right]^m \tag{3.25}$$

where $\left(\frac{d_i}{d_{50c}} \right)$ is the dimensionless size, the actual size divided by the corrected cut size, d50c, and m is the sharpness of separation.

8. Compute the actual classification coefficients, C_i using

$$C_i = P_{ic}(1 - R_f) + R_f \tag{3.26}$$

9. Convert C_i vector to C diagonal matrix to be able to use in closed circuit grinding equations.

10. Construct the cyclone efficiency curve, by plotting C_i vs. x_i. The corrected efficiency curve constructed with the use of corrected partition coefficients gives the result of the true classification, eliminating the short-circuiting or bypass effect during classification.

3.3 MATLAB® As a Programming Language

MATLAB is a convenient platform for development of simulation and design programs for its effectiveness as a programming language, sophisticated graphics features, and statistics and optimization tools. Through implementation of many toolboxes, the MATLAB user has close control over program development in diverse engineering applications.

MATLAB incorporates a programming language that is similar in structure to many common languages such as FORTRAN, BASIC, PASCAL, and C. One element that makes MATLAB's language unique is that, it uses a

matrix as its basic element. This property confers numerous benefits, that is in most programming languages it becomes time-consuming and weighty to perform the same mathematical operation on a group of numbers because the operation must be repeated (generally via loops) for each number, one number at a time. The greater the dimensionality of the data structure on which such operations are performed, the greater the number of embedded loops required. However, in MATLAB's programming language, data structures are constructed as matrices, and almost any operation can be performed in a single step. Embedded loops for these tasks are thus eliminated in MATLAB, saving the programmer time, and cutting down on the length and complexity of programming routines. A second benefit of MATLAB's matrix-based language pertains to the use and manipulation of images (www.mathworks.com).

MATLAB's abilities can be further utilized through easily programmable Graphical User Interfaces (GUIs). GUI's can serve as a powerful and intuitive tool for organizing and controlling all aspects of a program, including design, data collection, data analysis and theory fitting. There are essentially three ways in which a user can communicate with the computer via MATLAB: through the Command Window, through the use of scripts and functions, and through GUIs (Brian, A, 1995).

A GUI provides an intuitive interface between the user and the programming language and allows the user to bypass MATLAB commands altogether and, instead, to execute programming routines with a simple mouse click or keypress. No knowledge of MATLAB or computer programming is necessary for a user to successfully navigate and use a well-designed GUI (Marchand, P, Holland, T., 2003)

GUIs can range from simple question boxes prompting the user for a Yes/No response, to more complex interfaces, an example of which was provided in Figures 3.1, 3.2 and 3.3.

CHAPTER IV

GRINDSIM PROGRAM

4.1 Introduction

The menus created for GRINDSIM are quite simple and user friendly in nature. A main window containing the all sub menus and logo of the program is the first window the user meets. There are 4 shortcuts to the main options of the program, i.e.

- Batch Grinding Simulation
- Parameter Estimation
- Open Circuit Grinding
- Closed Circuit Grinding

The general structure of the program is explained in the following sections.

4.2 Batch Grinding Simulation Window

This menu simulates a batch ball mill on cumulative size distribution basis which is based on the model and solution equations given by P.K. Das (2001). The window computes the product size distributions for a set of grinding times from a single feed data. The menu reads in input data from

using general graphical user interface (GUI) window, tempting for the data files for feed distributions and size limits. Selection and breakage function parameters are typed from the GUI window, with various time input options.

The 80% passing size of feed and product are computed and a plot of size versus cumulative feed and product size distributions are plotted on the screen. All data input and output may be saved and printed using the File menu on the window.

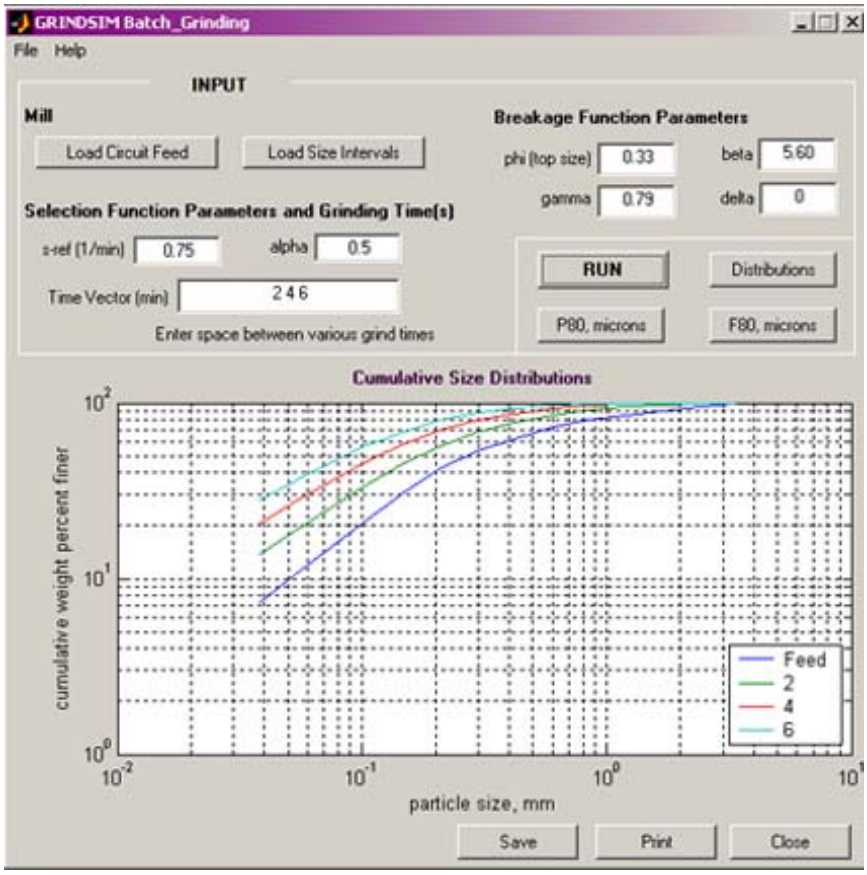

Figure 4.1 GRINDSIM Sample Run for Batch Grinding

The sample run is made with a feed size distribution w_i of:
w_i=[0.0239 0.2256 0.4435 0.5897 0.67 0.7086 0.7484 0.7880 0.8322 0.86 0.9415 1.0000];

and size limits x_i

x_i=[0.038 0.0530 0.075 0.106 0.150 0.212 0.300 0.420 .600 .850 1.200 1.680 2.400 3.353]

and selection function parameters:

s_{ref}=0.75; alpha=0.5

breakage function parameters

phi: 0.33; beta: 5.60; gamma:0.79 and delta:0

The simulation is run for 2, 4 and 6 minutes of grind times giving a fractional cumulative product size distribution which is presented in Table 4.1.

Table 4.1 Batch Grinding Simulation Results for grind times 2, 4 and 6 minutes

	Cumulative Fractional Product after:		
Size intervals (mm)	**t(min)=2**	**t(min)=4**	**t(min)=6**
-0.038	0.1367	0.2057	0.2772
-0.053+0.038	0.1866	0.2727	0.3593
-0.075+0.053	0.2554	0.3605	0.4615
-0.106+0.075	0.3468	0.4677	0.5770
-0.150+0.106	0.4577	0.5856	0.6936
-0.212+0.150	0.5725	0.6981	0.7958
-0.300+0.212	0.6819	0.7970	0.8777
-0.420+0.300	0.7744	0.8738	0.9344
-0.600+0.420	0.8546	0.9305	0.9700
-0.850+0.600	0.9091	0.9658	0.9886
-1.200+0.850	0.9506	0.9867	0.9968
-1.680+1.200	0.9798	0.9965	0.9994
-2.400+1.680	0.9957	0.9996	1.0000
-3.353+2.400	1.0000	1.0000	1.0000

The program gives the 80% passing values for these grind times as 0.4774 mm, 0.3047mm, and 0.2146 mm.

4.3 Parameter Estimation Window

The parameter estimation option helps the user to estimate selection function parameters with the cumulative-basis batch grinding model and a set of experimental data for various grind times. The aim here is to minimize the relative deviations between the model-computed and experimental product size distributions of cumulative basis.

The user is prompted to input initial estimates of parameters of selection and breakage functions. The experimental data is input to the program with the information of grind time. The estimation tool is then run to obtain the selection function parameters of Herbst's solution.

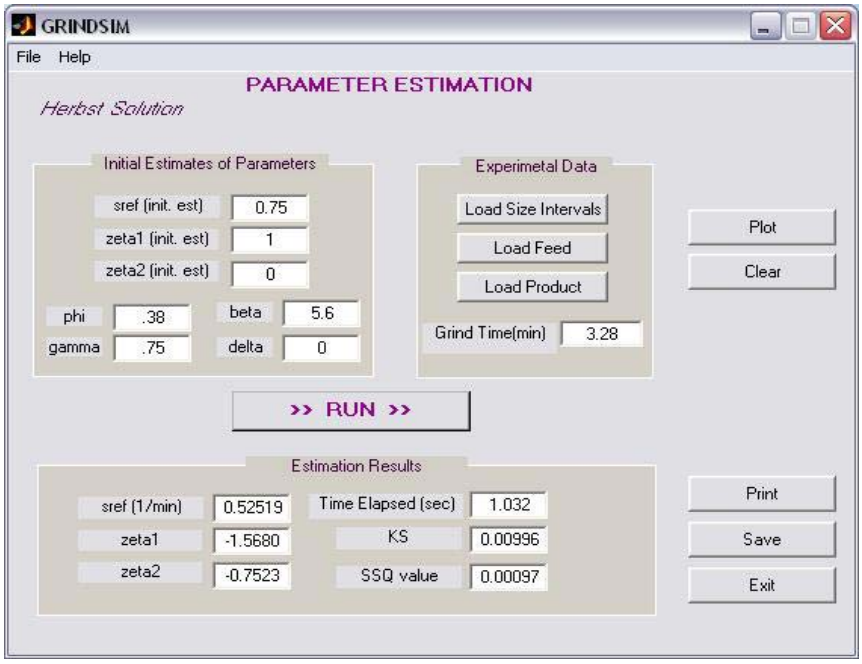

Figure 4.2 GRINDSIM Parameter Estimation Sample Run

4.4 Forward Simulation Window

Forward simulation window performs simulation for existing open and closed circuits of known mill power and feed size distributions. The closed circuit in itself has two solution options for normal and reverse closed circuits.

4.4.1 Open Circuit Grinding Window

This option simulates a ball mill continuously operating at steady state conditions with residence time t. The program requires top size limits for size intervals and feed size distribution with grinding time as input; in return computing product size distribution, graphical representation of the output and 80% passing size, P_{80}, after grind time t.

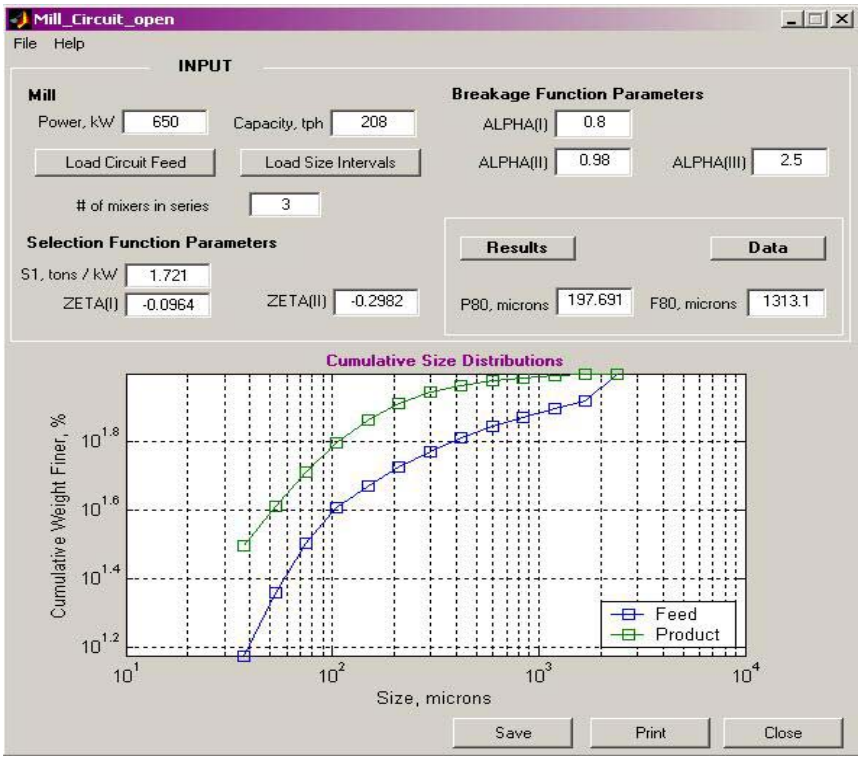

Figure 4.3 GRINDSIM Open Circuit Grinding Sample Run

Open circuit grinding is performed with the data given by Herbst et. al (1980), selection function value here is the energy based model of Herbst. The number of mixers is assumed as three which represent the RTD in the mill as three mills connected in series.

The breakage and selection function values along with feed data are main input to the program and after computation the 80% passing size of product and feed are calculated among with product size distributions. A plot of results is presented, in which cumulative weight percent finer versus particle size limits are plotted (Figure 4.3).

4.4.2 Closed Circuit Grinding Window

This tool simulates a mill and classifier system to grind a known mill power and amount of material to get product size distribution of cyclone overflow. The feed to the circuit consists of 13 size fractions, varying between 37-2380 μm, the 80% passing size, P_{80} is computed as 121 μm after simulation run. Input data to the program is taken from Herbst et. al. (1982) and results coincide with their results. The GUI made and inputs to the closed circuit grinding are shown in Figure 4.4.

Both normal and reverse closed circuit grinding simulations are made based on the model developed by the Herbst et. al.(1982), the typical necessary input for existing closed circuit simulation are fractional feed size distribution, breakage and selection function parameters, mill power and fresh feed rate along with cyclone model parameters. In the model, initially circulating load is set to zero, solids and water go into ball mill and produce a ball mill product according to the mill model and join with water to form the cyclone feed at specified percent solids, the material is then split according to the cyclone model to give final product size distribution and 80% passing size of the product at a calculated circulating load.

The basic principle here is to repeat the procedure until the fresh feed rate to the circuit equals the cyclone overflow solids rate followed by a water balance made with the steady state circuit data. Here water feed to the cyclone is set at a rate to meet prespecified % solids.

The program uses energy based form of the grinding model; therefore the S values through the program have units in tons/kWh. The size distribution of the product stream is presented by weight fractions, p_i, p_1 being the top size and the smallest size being p_n for n size intervals. The cyclone efficiency curve and cumulative size distribution values of feed and product are plotted after the simulation run. The user may select between normal and reverse closed circuit options.

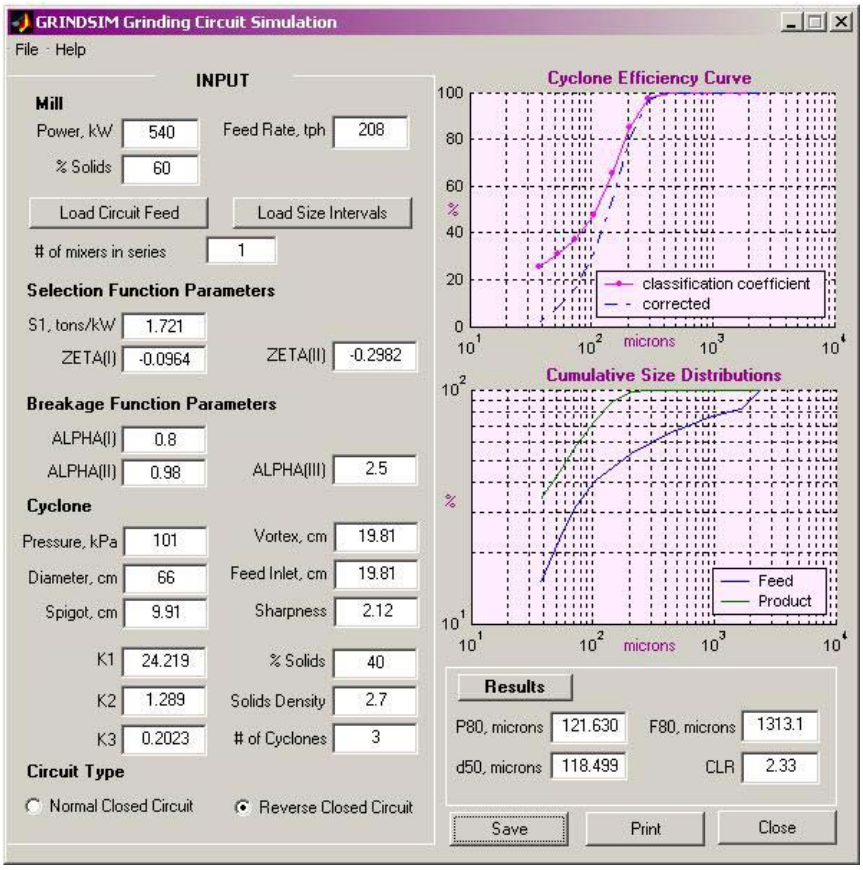

Figure 4.4 GRINDSIM Reverse Closed Circuit Grinding Sample Run

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER STUDY

5.1 Conclusions

In this study, a computer application in mineral processing has been conducted. The toolbox developed in MATLAB programming environment allows user to simulate grinding operations, namely batch grinding, parameter estimation, open and closed circuit grinding.

The Batch Grinding and Open Circuit Grinding section of the toolbox first requires the grinding data for a mill, the initial feed size distribution, size intervals in microns, selection and breakage functions along with related parameters; in turn an output of product size distribution with graphical representation is provided using the related mathematical models of size reduction proposed up to date by various workers.

The Closed Circuit Grinding section of the toolbox requires the grinding data for a mill, the initial feed size distribution, size intervals in microns, selection and breakage functions along with related parameters and mill power and

capacity or feed rate inputs, computing 80% passing sizes of feed and product, product size distribution and graphical outputs.

Main conclusions derived from this study are as follows:

1. The MATLAB toolbox developed will be useful at least for educational purposes.

2. The simulation results agree well with the results reported in the literature from which the input data for our simulator were taken. (See Appendix C).

3. The toolbox will save time and effort in grinding applications, giving the user a starting point.

4. The graphical user interfaces created makes the program easy for users to perform simulations.

5. Using graphical user interfaces in education should be widened to increase understanding in education.

5.2 Recommendations for Further Study

With this study ball mill simulation is studied, also mill design and scale up procedures are revised, a GUI made to perform mill-design and scale up in the next versions would be useful.

For the next versions of this toolbox, help files may be extended to solve possible user or data input related problems.

The developments in computer applications will be for the benefit of mineral processing engineers and education, saving time and effort to create better understanding of the processes. Models used in this thesis link product size distribution with a series of model parameters. The effect of mill parameters are lumped into breakage and selection functions. Breakage kinetics and the mixing efficiency are obviously dependent on the mill size, ball size distribution, mill load and many other parameters.

In future, user-friendly computer applications are expected to be produced to get better understanding of mineral processes in education.

REFERENCES

Austin, L.G., Klimpel, R.R., Luckie, P.T., Rogers, R.S.C., 1982, "Simulation of Grinding Circuits for Design", In: Design and Installation of Comminution Circuits, editors: A.L. Mular and J.V. Jergensen. SME-AIME, New York, Pages 301-324.

Austin, L.G., Klimpel, R.R., Luckie, P.T., 1984, "Back Calculation of Breakage Parameters from Batch and Continuous Mill Data", Process Engineering of Size Reduction, SME-AIME, New York.

Austin, L.G., Klimpel, R.R, Luckie, P.T., 1984, "Process Engineering of Size Reduction: Ball Milling", SME-AIME, New York, NY.

Austin, L.G., 1999, "A Discussion of Equations for the Analysis of Batch Grinding Data", Powder Technology 106, Pages 71-77.

Bond, F.C., 1952, "The Third Theory of Comminution", Trans. AIME 193, Pages 484– 494.

Bond, F.C., 1960, "Crushing and Grinding Calculation", Br. Chem. Eng. 6, Pages 378– 391.

Brian, A, 1995, "MATLAB for Engineers", Addison-Wesley, England

Dahlstrom, D. A., Kam W., 1988," Potential Energy Savings in Comminution by Two-stage Classification", International Journal of Mineral Processing , Volume 22, Issues 1-4 , Pages 239-250

Das, P.K., Khan, A.A., & Pitchumani, B., 1995, "Solution of the Batch Grinding Equation", Powder Technology, Volume 85, Pages 189-192.

Das, P.K., 2001, "Use of Cumulative Size Distribution to Back-calculate the Breakage Parameters in Batch Grinding", Computers and Chemical Engineering, Volume 25, Pages 1235-1239.

D.J. Higham, N.J. Higham, 2000, "MATLAB Guide", Society for Industrial and Applied Mathematics, Philadelphia.

Gardner, R.P., Austin, L.G., 1972, "Use of Radioactive Tracer Technique and a Computer in the Study of the Batch Grinding of Coals", Journal of Institute of Fuel, Volume 35, Pages 174-177.

Herbst, J.A., Fuerstenau, D.W., 1973, "Mathematical Simulation of Dry Ball Mill Using Specific Power Information", Trans., vol. 254. SME Publications, Littleton, Colorado, USA, Pages 343– 348.

Herbst, J.A., Grandy, G.A., Fuerstenau, D.W., 1973, "Population Balance Models for the Design of Continuous Grinding Mills", Proceedings of the 10th International Mineral Processing Congress, Pages 27– 37.

Herbst, J.A. and Fuerstenau, D.W., 1980, "Scale-Up Procedure for Continuous Grinding Mill Design Using Population Balance Models", International Journal of Mineral Processing, Vol. 7, Pages 1-31.

Herbst, J.A., Siddique, M., Rajamani, K., Sanchez, E., 1981, “Population Balance Approach to Ball Mill Scale-Up: Bench and Pilot Scale Investigations”, Trans. SME/AIME 272, Pages 1945–1954.

Herbst, J.A., Rajamani, R.K., 1982, “Developing a Simulator for Ball Mill Scale-Up: A Case Study”, In: Design and Installation of Comminution Circuits, Editors: Mular, A.L., Jergensen, G.V., AIME, New York, Pages 325– 345.

Hoşten, Ç., Avşar, Ç., 2004, “Variation of Back-Calculated Breakage Rate Parameters in Bond Mill Grinding.”, Scandinavian Journal of Metallurgy, in print.

Klimpel and Austin, 1977, “The Back-calculation of Specific Rates of Breakage Distribution Parameters from Batch Grinding Data”, International Journal of Mineral Processing, Volume 4, Issue 1, Pages 7-32.

Luckie, P.T., Austin, L.G., 1972, “A Review Introduction to the Solution of the Grinding Equations by Digital Computation”, Minerals Science Engineering, Vol.4, No.2.

Lynch, A.J., Rao, T.C., 1975, “Modeling and Scale-Up of Hydrocyclone Classifiers”, XI. International Mineral Processing Congress Proceedings, Cagliari.

Lynch, A.C., 1977, “Mineral Crushing and Grinding Circuits”, Elsevier Scientific Publishing Company, Amsterdam.

Marchand, P, Holland, T., 2003, “Graphics and GUI’s with MATLAB”, Boca Raton, FL.

MATLAB Web Site, 2003, “The Mathworks”, http://www.mathworks.com

Mular, A.L., Herbst, J.A., 1982, “Digital Simulation: An Aid for Mineral Processing Design”, In Design and Installation of Comminution Circuits, editors: A.L. Mular and J.V. Jergensen. SME-AIME, New York

Rajamani, R.K. and Herbst, J.A., 1984, “Simultaneous Estimation of Selection and Breakage Functions from Batch and Continuous Grinding Data”, Trans. Instn. Min. Metallurgy, Section C. 93, C74-C85.

Reid, K.J., 1965, “A Solution to the Batch Grinding Equation”, Chemical Engineering Science 20, Pages 953-963.

Shoji, K., Lohrasb, S., Austin, L.G., 1980, “The variation of Breakage Parameters with Ball and Powder Loading in Dry Ball Milling”, Powder Technology, Volume 25, Issue 1, Pages 109-114.

Wills, BA, 1992, “Mineral Processing Technology: an Introduction to the Practical Aspects of Ore Treatment and Mineral Recovery”, Oxford, NY.

APPENDIX A

MAIN SUBFUNCTIONS OF GRINDSIM

1 Code for Batch Grinding

```
function varargout = RUN_Callback(h, eventdata, handles, varargin)
clc
wi=get(handles.Load_Fin,'UserData');
F=wi; F(end)=[];
%Selection fcn parameters are taken from closed circuit example of Herbst
sref=get(handles.S1,'UserData');
alpha=get(handles.alpha,'UserData');
time=get(handles.zeta2,'UserData');

%Breakage fcn parameters
phi=get(handles.phi,'UserData');
gamma=get(handles.gamma,'UserData');
beta=get(handles.beta,'UserData');
delta=get(handles.delta,'UserData');
%Mill parameters
xi=get(handles.Load_Size,'UserData');
```

```
n=length(xi); %number of size intervals
%Selection functions
S=SelFuncA(xi,sref,alpha); %calling the selection function

%call the function for computing a matrix of cum. breakage functions:
   p=[phi,gamma,beta,delta];
B=BijFunc(xi,p);
A=zeros(n-1,n-1); % A is (n-1)-by-(n-1) upper triangular matrix

for i=1:n-1;
   for j=1:n-1;
      if i==j
         A(i,j)=-S(j+1).*B(i,j+1);
      elseif i>j
         A(i,j)=0.0;
      else
         A(i,j)=S(j).*B(i,j)-S(j+1).*B(i,j+1);
      end
   end;
end;
C=S(n).*B(:,n); C(end)=[]; % C becomes (n-1)-by-1 column vector.
%U(i,j) is the matrix for the eigenvectors of A
U=eye(n-1);
for j=2:n-1
```

```
    for i=j-1:-1:1
        sigma1=0;
        for k=1:j-i
            sigma1=sigma1+A(i,j-k+1).*U(j-k+1,j);
        end;
        U(i,j)=sigma1./(A(j,j)-A(i,i));
    end;
end;
%V(i,j) is the matrix for the eigenvectors of A transpose.
V=eye(n-1);
for j=1:n-2
    for i=j+1:n-1
        sigma2=0;
        for m=1:i-j
            sigma2=sigma2+A(j+m-1,i).*V(j+m-1,j);
        end;
        V(i,j)=sigma2./(A(j,j)-A(i,i));
    end;
end;
q=length(time)
P=zeros(n,q); %each column is the product for the relevant grind time.
for t=1:q
    for i=1:n-1
        sigma=0.0;
```

```
        for j=1:n-1
            a=V(:,j)'*F;
            b=V(:,j)'*C;
            c=exp(A(j,j).*time(t));
            d=1-exp(A(j,j).*time(t));
            sigma=sigma+(c.*a-b.*d./(A(j,j))).*U(i,j);
        end
        P(i,t)=sigma;
    end
    P(n,t)=1.0;
end;
%plot mill feed and product size distributions
P80=interp1(P(:,1),xi,.8)
P802=interp1(P(:,2),xi,.8)
P803=interp1(P(:,3),xi,.8)
t=1:q; loglog(xi, [wi*100 P(:,t)*100],'linewidth',1);
grid on;
xlabel ('particle size, mm');
ylabel('cumulative weight percent finer');
time1=num2str(time(1));
time2=num2str(time(2));
time3=num2str(time(3));
legend('Feed',time1,time2,time3,4);
```

```
function pushbutton20_Callback(hObject, eventdata, handles)
%distributions callback

% hObject    handle to pushbutton20 (see GCBO)

% eventdata  reserved - to be defined in a future version of MATLAB

% handles    structure with handles and user data (see GUIDATA)

% Create a figure to work with

A=get(handles.pushbutton20,'UserData');

fig = figure('NumberTitle','off','Name','Product Size Distributions');


% Create the axes

ax = axes('Units','normal','Position',[.1 .1 .8 .8]);

axis off  % Make the axes invisible

axis ij   % Origin at top left


% Define the default values for text

set(fig,'DefaultTextFontName','courier', ...  % So text lines up

      'DefaultTextHorizontalAlignment','left', ...

      'DefaultTextVerticalAlignment','bottom', ...

      'DefaultTextClipping','on')

% Define the matrix

%A = 999*rand(20,25);

% Determine the number of rows and columns

[m,n] = size(A);

% Draw th e first column
```

```
axis([1 m 1 n])
drawnow % forcing MATLAB to update the screen
tmp = text(.5,.5,'t');
ext = get(tmp,'Extent');
dy = ext(4);  % Height of a single character.
wch = ext(3); % Width of a single character.
fw = 8;       % Column width.  Each column can have up to
% 8-digits in it.
wc = 8*wch;   % Width of 8 digit column
dwc = 2*wch;  % Distance between columns
dx = wc+dwc;  % Step used for columns
x = 1;        % Location of first column
delete(tmp)
for i = 1:n   % Column
  y = 1;
  for j = 1:m % Row
    y = y + abs(dy);  % Location of row
    t(j,i) = text(x,y,sprintf('%3.4f',A(j,i)));
  end
  x = x+dx;   % Location of next column
end

% Fixing up the axes
axis([1-dwc/2 1+6*dx-dwc/2 1 n])
```

```
set(gca,'XTick',(1-dwc/2):dx:x)
set(gca,'XGrid','on','GridLineStyle','-','Box','on')
set(gca,'YTick',[],'XTickLabel',[],'Visible','on')
title('Columns 1 through 6')

% Adding a horizontal slider
slide_step = floor(10/dx);  % Number of columns in view
hs = uicontrol('Style','slider','Units','normal', ...
   'Position',[.1 0 .8 .05],'min',1,'max',n, ...
   'UserData',[dx,dwc],'Value',1, ...
            'CallBack',['val = get(gco,"Value");', ...
             'dx = get(gco,"UserData");', ...
             'dwc = dx(2); dx = dx(1);', ...
             'if (val-round(val))>eps,', ...       % Test for + step
             '  val = ceil(val);', ...
             'elseif (val-round(val)) < -eps,', ... % Test for - step
             '  val = floor(val);', ...
             'end,', ...
             'minx = 1+(floor(val)-1)*dx-dwc/2;', ...
             'maxx = minx+6*dx;', ...
             'set(gco,"Value",val),', ...    % Set the step
             'axlim = axis;', ...
             'axis([minx maxx axlim(3:4)]),', ...
             'title(["Columns ",int2str(val)," through ", int2str(val+6)])']);
```

2 Code for Parameter Estimation Main Function

```
%solves the problem using the Herbst Solution
function varargout = RUN_Callback(h, eventdata, handles, varargin)
global Pcom
   p_init1=get(handles.p_init1,'UserData')
   p_init2=get(handles.p_init2,'UserData')
   p_init3=get(handles.p_init3,'UserData')
   p1_init=[p_init1 p_init2 p_init3] %SelFuncH requires row vector
   p_init4=get(handles.p_init4,'UserData')
   p_init5=get(handles.p_init5,'UserData')
   p_init6=get(handles.p_init6,'UserData')
   p_init7=get(handles.p_init7,'UserData')
p2=[p_init4 p_init5 p_init6 p_init7]
xi=get(handles.Load_xi,'UserData')
Fin=get(handles.Load_fi,'UserData')
Pexp=get(handles.Load_pi, 'userData')
Time=get(handles.time, 'UserData')
n=length(xi)
q=length(Time)
B=BijFunc(xi,p2)
clc %clears the command window and homes the cursor
%call the minimization function with an appropriate objective function
tic
options=optimset('MaxFunEvals',4000, 'MaxIter',4000);
```

```
% here we use MATLAB's built in function 'fminsearch'
[x, value]=fminsearch('ssq_H',p1_init,options,xi,Fin,Pexp,Time,n,q,B)
% Calculate the Kolmogorov-Smirnov test statistics, KS
for j=1:q
   KS(1,j)=max(abs(Pexp(:,j)-Pcom(:,j)));
end
KS
toc
t=toc;

set(handles.time_elapsed,'UserData',t);
set(handles.KS,'UserData',KS);
set(handles.x1,'UserData',x(1));
set(handles.x2,'UserData',x(2));
set(handles.x3,'UserData',x(3));
set(handles.SSQ,'UserData',value);

set(handles.KS,'String',KS)
set(handles.time_elapsed,'String',t);
set(handles.x3,'String',x(3));
set(handles.x2,'String',x(2));
set(handles.x1,'String',x(1));
set(handles.SSQ,'String',value);
guidata(gcbo,handles);%saving the data
```

3 Sample Code for Selection Function Computation

```
function RetVal=SelFuncA(y,p1,p2)

n=length(y);

yref=1.42; %user-defined reference size in mm

i=1:n; S=p1.*(y(i)./yref).^p2; % S is 1-by-n vector.

S(1)=0; % sink size interval selection function is zero.

RetVal=S;
```

4 Sample Code for Breakage Function Computation

```
function RetVal=BijFunc(y,p)

%BIJFUNC

% Computes  cumulative breakage distribution functions.

% Bij is the cumulative fraction of progenies of size less than or

% equal to upper size limit of the ith size interval produced

% from the breakage of the particles from the jth parent size interval.

% y is a vector whose elements are the upper size limits of the size

% intervals.First element of y is the finest non-zero size.

% p is a row vector whose elements are the 4 parameters (phi at 5mm, gamma,

% beta, and delta) of the general functional form for the non-normalized

% cumulative breakage function values.

% Delta is zero for normalizable breakage.

n=length(y);

Bmtx=eye(n); %n-by-n identity matrix
```

```
m=length(p);
if m~=4
   disp('!!!You must enter 4 parameter values for Bij!!!')
end;
if p(4)==0  %normalizable breakage
   for j=2:n
      for i=1:j-1
         normy=y(i)./y(j-1);
         Bmtx(i,j)=p(1).*normy.^p(2)+(1-p(1)).*normy.^p(3);
      end;
   end;
elseif p(4)<0
   disp('!!!You must enter a non-negative value for delta!!!')
   return;
else
   %non-normalizable breakage
   phi=zeros(1,n); phi(1,n)=p(1,1);
   for j=2:n-1
      phi(1,j)=p(1).*(y(n)./y(j)).^p(4);
   end
   for j=2:n
      for i=1:j-1
         normy=y(i)./y(j-1);
         Bmtx(i,j)=phi(1,j).*normy.^p(2)+(1-phi(1,j)).*normy.^p(3);
```

```
        end;
    end;
end
RetVal=Bmtx;
%Now Bmtx is upper diagonal with principal diagonal and first minor
%diagonal  elements above the main diagonal are all ones.
```

5 Sample Code for Open Circuit Grinding Simulation

```
function varargout = RUN_Callback(h, eventdata, handles, varargin)
clc
mill_capacity=get(handles.capacity,'UserData'); %tph
mixers=get(handles.mixers,'UserData');

wi=get(handles.Load_Fin,'UserData');

%Selection fcn parameters are taken from closed circuit example of Herbst
S1=get(handles.S1,'UserData');
zeta1=get(handles.zeta1,'UserData');
zeta2=get(handles.zeta2,'UserData');

%Breakage fcn parameters
alpha1=get(handles.alpha1,'UserData');
alpha2=get(handles.alpha2,'UserData');
alpha3=get(handles.alpha3,'UserData');
```

```
%Mill parameters
xi=get(handles.Load_Size,'UserData');
n=length(xi); %number of size intervals
%calculating E
P=get(handles.mill_power,'UserData');%kW
E=P/mill_capacity;%Bond's eqn
%Selection functions
S=SelFun(xi,zeta1,zeta2,S1);

%Breakage functions calculation
%calling breakage matrix function
bmtx=Bij(xi,alpha1,alpha2,alpha3);

%calling function to get L=TJT-1 matrix
[L,J]=L_matrix(n,bmtx,S,E,mixers);
I=eye(n);

%Open Circuit
mp=L*wi;
mp=cumsum(mp);
mp(n,:)=[];
mp=[1;1-mp]
wicum=cumsum(wi);wicum=flipud(wicum);
f80=interp1(wicum,xi,.8);
```

```
set(handles.F80,'String',f80);
%the following lines are coded to make spline function work for our case
for i=1:n-1
   if mp(i)==1 | mp(i)-mp(i+1)<0.00001
      mp(i)=1-0.0001*i;
   end
end;

P80=interp1(mp,xi,.8)
set(handles.P80,'String',P80);
yas=loglog(xi, [wicum*100 mp*100], 'marker','square');
hold on
grid on
legend('Feed','Product',4);
xlabel('Size, microns')%,'FontSize','8','FontAngle','Italic');
wicum
ylabel('Cumulative Weight Finer, %')%,'FontSize','10','FontAngle','Italic');
hold off
handles.sizedist=yas;
```

6 Sample Code for Closed Circuit Grinding Simulation

```
function RUN_Callback(hObject, eventdata, handles)
clc;  tic
mill_capacity=get(handles.feed_rate,'UserData'); %tph
```

```
wi=get(handles.wi,'UserData')
%Selection fcn parameters are taken from closed circuit example of Herbst
S1=get(handles.S1,'UserData');
zeta1=get(handles.zeta1,'UserData');
zeta2=get(handles.zeta2,'UserData');
%Breakage fcn parameters
alpha1=get(handles.alpha1,'UserData');
alpha2=get(handles.alpha2,'UserData');
alpha3=get(handles.alpha3,'UserData');

%Mill parameters
xi=get(handles.xi,'UserData');
n=length(xi); %number of size intervals

%calculating E
Power=get(handles.power,'UserData');%kW
mixers=get(handles.mixers,'UserData');
%calculating E
e=ones(1,n);
CL=0;
CLR=0;
cirprod=10;
indice=0;
while abs(cirprod-mill_capacity)>0.01
```

```
E=Power/mill_capacity/(CLR+1)
%Selection functions
S=SelFun(xi,zeta1,zeta2,S1)
%calling breakage matrix function
bmtx=Bij(xi,alpha1,alpha2,alpha3);

%calling function to get L=TJT-1 matrix
[L,J]=L_matrix(n,bmtx,S,E,mixers)
I=eye(n);

%CLASSIFICATION MATRIX
K1=get(handles.K1,'UserData');
K2=get(handles.K2,'UserData');
K3=get(handles.K3,'UserData');
m=get(handles.m,'UserData');
Ss=get(handles.Ss,'UserData');

P=get(handles.cyc_pressure,'UserData');%kPa
PSF=get(handles.FPS,'UserData');%percent solids in feed
PWF=100-PSF;%in percent by wt feed water
%for the mill % solids we have FPS_mill callback
   for i=1:n-1
      dia(i)=sqrt(xi(i)*xi(i+1));
   end;
```

```
di=[dia,xi(n)/2];% in herbst cyclone model di is the geometric mean of size limits

Dc=get(handles.cyc_dia,'UserData');

Du=get(handles.spigot,'UserData');

Do=get(handles.vortex,'UserData');

Di=get(handles.inlet,'UserData');


slurryden=1/(1-((Ss-1)/Ss*PSF/100))

Q=K1*Do*P^0.5*(PWF)^0.125;%per cyclone

TPHFEED=(Q*60*0.001*slurryden);%tph cyclone feed

WF=TPHFEED*(1-0.01*PSF);%tph of feed water

logd50c=(0.02*PSF-
0.083*Du+0.016*Do+0.00076*TPHFEED+K2);%microns

d50=10^logd50c;%corrected d50

Rf=(1.93*Du/WF)-(2.72/WF)+K3;

Pic=1-exp(-0.693*(di/d50).^m);

Ci=Pic*(1-Rf)+Rf;

%Ci=[1;1;1;1;1;0.9977;0.9451;0.7711;.5624;.4128;.3317;.2907;.2601]

C=diag(Ci);


if(get(handles.normal,'Value')==get(handles.normal,'Max'))

%Normal Closed Circuit

mp=((I-C)*L)*inv(I-C*L)*wi;

mp=cumsum(mp);

mp(n,:)=[];
```

```
mp=[1;1-mp];
%making solids balance around classifier
cycfeed=mill_capacity+CL;
cirprod=cycfeed/(e*L*inv(I-C*L)*wi);
CL=cycfeed-cirprod;
indice=indice+1
CLR=CL/mill_capacity;

else %reverse circuit
mp=(I-C)*(inv(I-L*C)*L*C+I)*wi;
mp=cumsum(mp);
mp(n,:)=[];
mp=[1;1-mp];

%making solids balance around classifier
cycfeed=mill_capacity+CL
cirprod=cycfeed/(e*inv(I-L*C)*L*C*wi)
CL=cycfeed-cirprod
indice=indice+1
CLR=CL/mill_capacity;
end;%if statement
end; %while loop to obtain CLR of the circuit @ steady state
for i=1:n-1
   if mp(i)~=1
```

```
    break
end
end
%the next few lines are to calculate 80% passing sizes correctly with
interp1 fncn
wicum=cumsum(wi);wicum=flipud(wicum);
mp_interp=mp(i+1:end); %will be used for finding P80 value
xi_interp=xi(i+1:end);


p80=interp1(mp_interp,xi_interp,.8);
f80=interp1(wicum,xi,.8);
set(handles.F80,'String',f80);
set(handles.P80,'String',p80);
set(handles.CLR,'String',CLR);
set(handles.d50,'String',d50);


axes(handles.axes5)
semilogx(xi,Ci*100,'m.-'); hold on;
semilogx(xi,Pic*100,'b-.');grid on;
legend('classification coefficient','corrected',4);
set(handles.axes5,'XMinorTick','on')
axes(handles.axes6)
loglog(xi, [wicum*100 mp*100]);legend('Feed','Product',4);
grid on
toc
```

```
%the next few lines are coded to calculate water addition/suction at key
locations

if(get(handles.normal,'Value')==get(handles.normal,'Max'))      %check   the
circuit type selected

else

   waterbeforecyclone=cycfeed*(100+PSF)/100

end

function print_Callback(hObject, eventdata, handles)

A=get(handles.print,'UserData');

fig = figure('NumberTitle','off','Name','Product Size Distributions');


% Create the axes

ax = axes('Units','normal','Position',[.1 .1 .8 .8]);

axis off  % Make the axes invisible

axis ij   % Origin at top left

% Define the default values for text

set(fig,'DefaultTextFontName','courier', ...  % So text lines up

      'DefaultTextHorizontalAlignment','left', ...

      'DefaultTextVerticalAlignment','bottom', ...

      'DefaultTextClipping','on')

% Define the matrix

%A = 999*rand(20,25);

% Determine the number of rows and columns

[m,n] = size(A);
```

```
% Draw the first column
axis([1 m 1 2])
drawnow
tmp = text(.5,.5,'t');
ext = get(tmp,'Extent');
dy = ext(4);  % Height of a single character.
wch = ext(3); % Width of a single character.
fw = 8;       % Column width.  Each column can have up to
              % 8-digits in it.
wc = 8*wch;   % Width of 8 digit column
dwc = 2*wch;  % Distance between columns
dx = wc+dwc;  % Step used for columns
x = 1;        % Location of first column
delete(tmp)
for i = 1:n   % Column
  y = 1;
  for j = 1:m % Row
    y = y + abs(dy);  % Location of row
    t(j,i) = text(x,y,sprintf('%3.4f',A(j,i)));
  end
  x = x+dx;   % Location of next column
end
% Fix up the axes
axis([1-dwc/2 1+6*dx-dwc/2 1 n])
```

```
set(gca,'XTick',(1-dwc/2):dx:x)
set(gca,'XGrid','on','GridLineStyle','-','Box','on')
set(gca,'YTick',[],'XTickLabel',[],'Visible','on')
title('Columns 1 through 6')

% Add a horizontal slider
slide_step = floor(10/dx);  % Number of columns in view
hs = uicontrol('Style','slider','Units','normal', ...
   'Position',[.1 0 .8 .05],'min',1,'max',n, ...
   'UserData',[dx,dwc],'Value',1, ...
            'CallBack',['val = get(gco,"Value");', ...
             'dx = get(gco,"UserData");', ...
             'dwc = dx(2); dx = dx(1);', ...
             'if (val-round(val))>eps,', ...       % Test for + step
             '  val = ceil(val);', ...
             'elseif (val-round(val)) < -eps,', ... % Test for - step
             '  val = floor(val);', ...
             'end,', ...
             'minx = 1+(floor(val)-1)*dx-dwc/2;', ...
             'maxx = minx+6*dx;', ...
             'set(gco,"Value",val),', ...    % Set the step
             'axlim = axis;', ...
             'axis([minx maxx axlim(3:4)]),', ...
             'title(["Columns ",int2str(val)," through ", int2str(val+6)])']);
```

7 Sample Code for L Matrix Computation

```
function [RetVal,J]=L_matrix(n,bmtx,S,E,mixers)
T=eye(n);
   for r=1:n-1
      for k=r+1:n
         sigma=0;
         for jsum=r:k-1
            sigma=sigma + bmtx(k,jsum).*S(jsum).*T(jsum,r);
         end;
         dummy=S(k)-S(r);
           if dummy==0
              dummy=eps;
           end
         T(k,r)=sigma./dummy;
      end;
   end;
jdiag=(1+S.*E/mixers).^mixers;
J=1./jdiag;
J=diag(J);

L=T*J*inv(T);%Transformation matrix is ready
RetVal=L;
```

APPENDIX B

MAIN WINDOWS OF GRINDSIM

1 Main GUI's of GRINDSIM

Figure 1B Main Menu of GRINDSIM

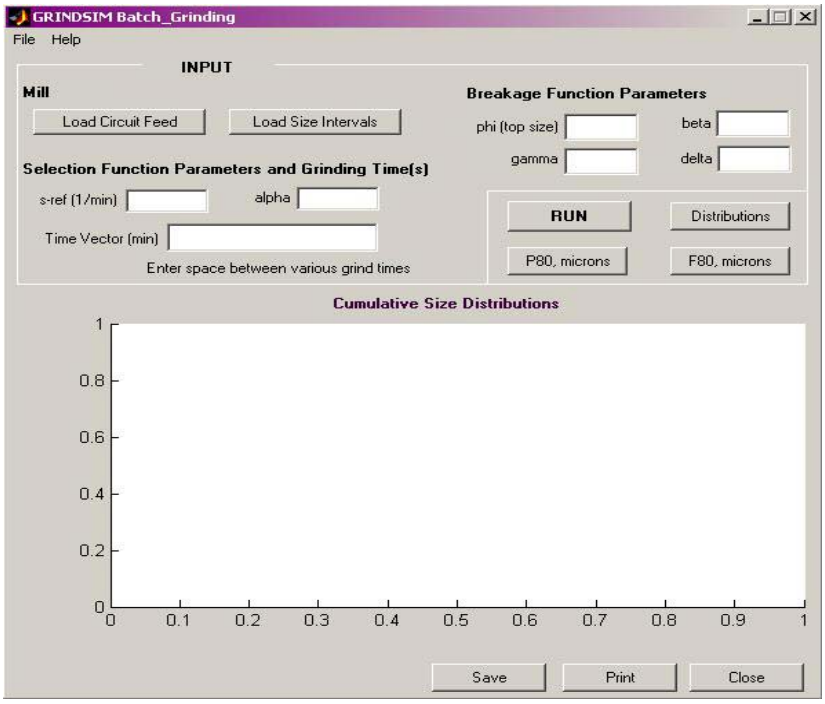

Figure 2B Batch Grinding Simulation Window

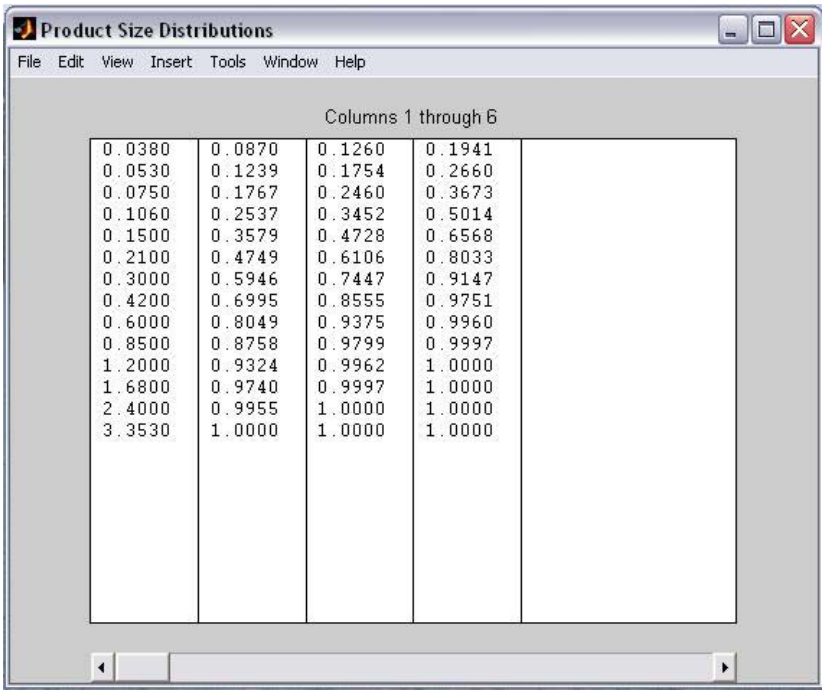

Figure 3B Batch Grinding Product Size Distribution Window

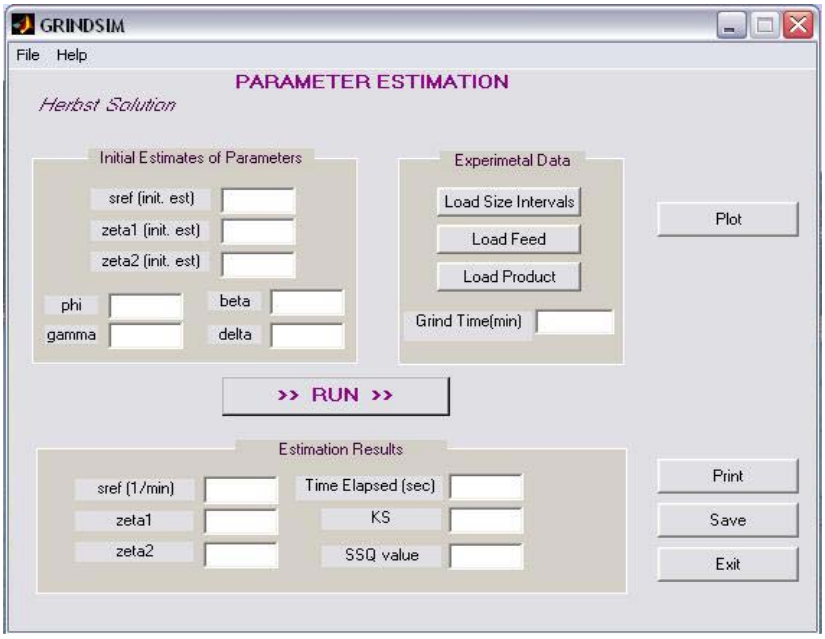

Figure 4B Parameter Estimation Window

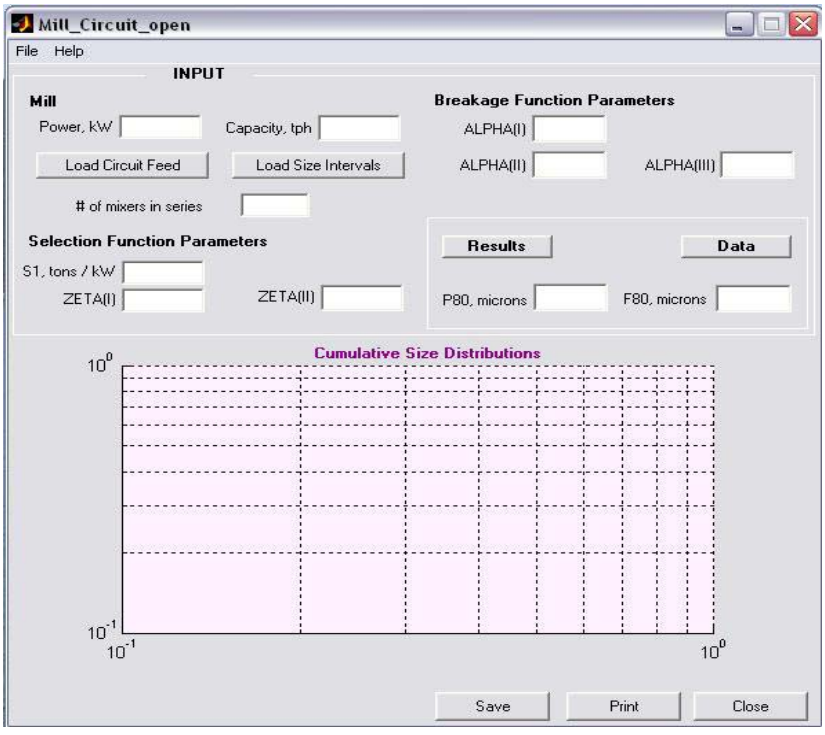

Figure 5B Open Circuit Grinding Simulation Window

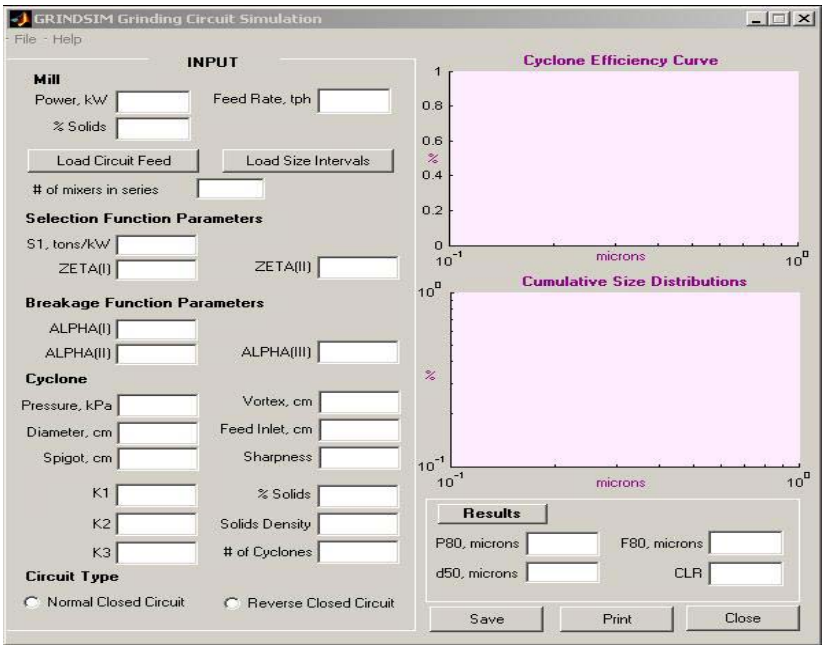

Figure 6B Closed Circuit Grinding Simulation Window

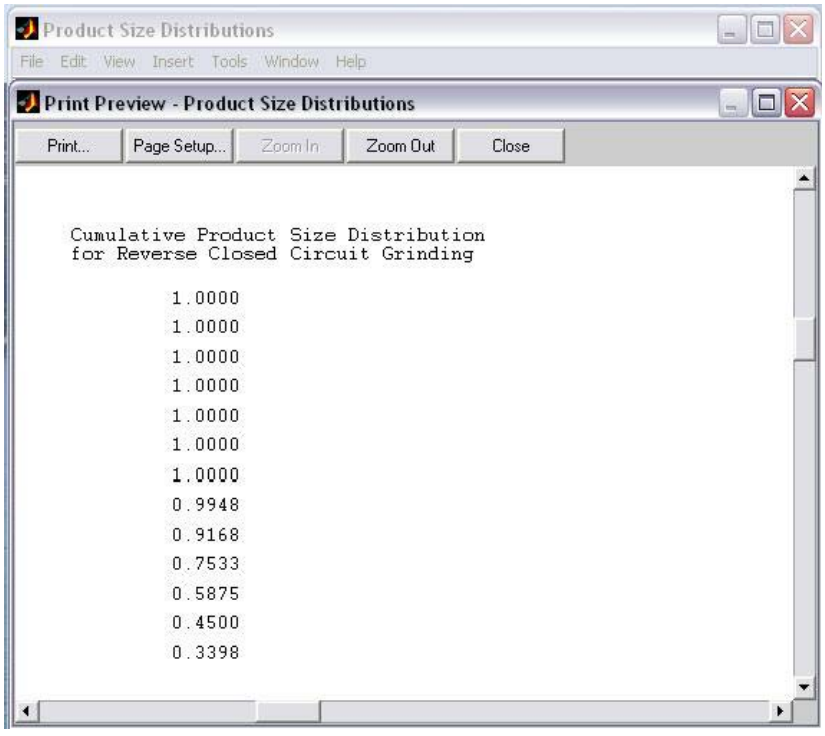

Figure 7B GUI for Printing Product Size Distribution

Figure 8B GUI for Confirmation

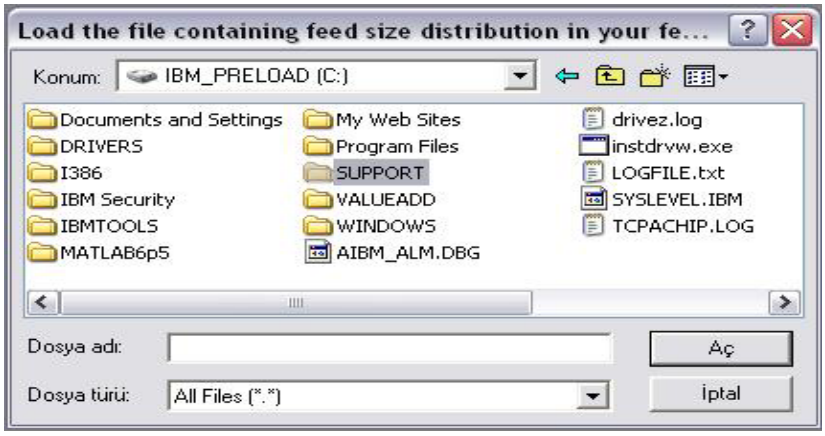

Figure 9B GUI for Uploading Feed Data

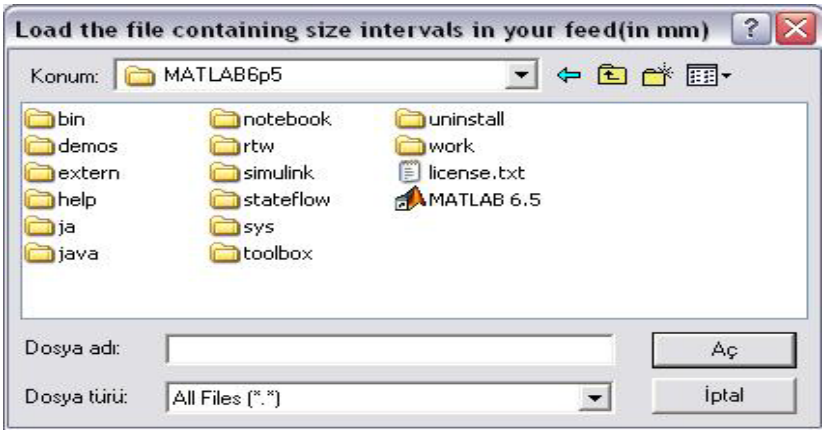

Figure 10B GUI for Uploading Size Interval Data

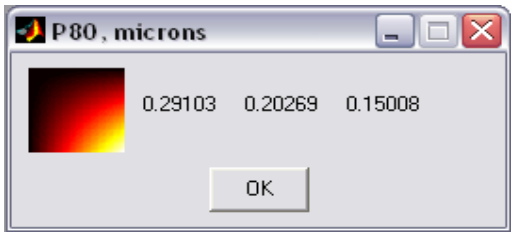

Figure 11B GUI for Presenting 80% Passing Sizes

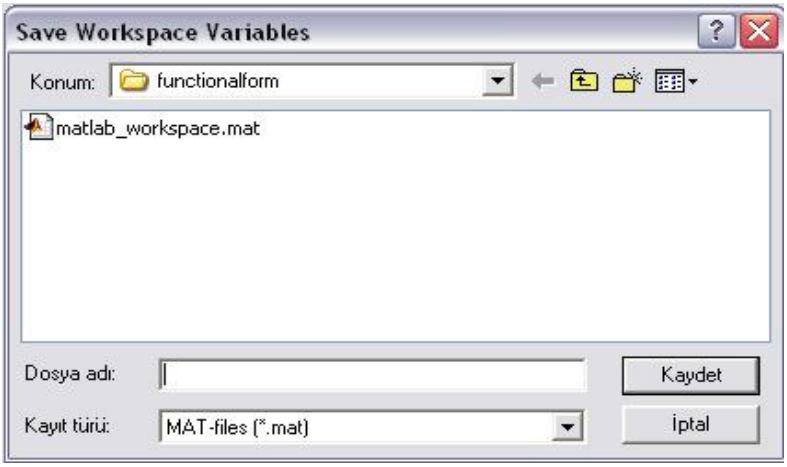

Figure 12B GUI for Saving the Workspace Variables

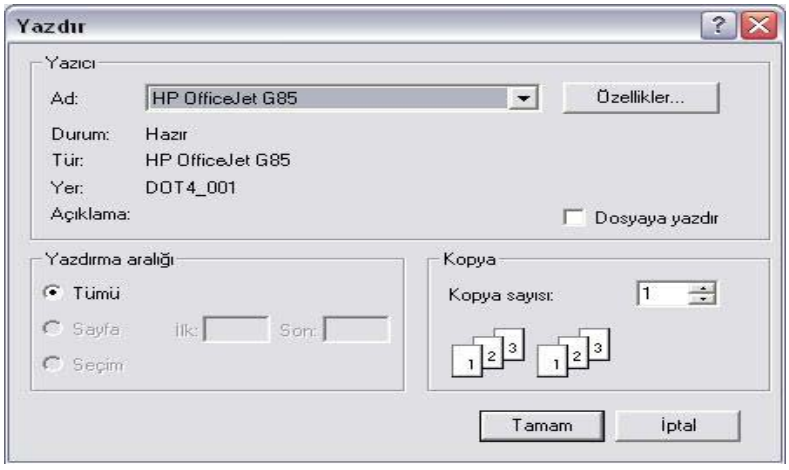

Figure 13B GUI for Printing the Workspace Variables

APPENDIX C

COMPARISON of PROGRAM RESULTS

Table 1C Cumulative Fractional Mill Product for Reverse Closed Circuit Grinding

Top Size (microns)	**Reference data** (Herbst & Rajamani, 1982)	**GRINDSIM Simulator**
2380	1	1
1680	1	1
1190	1	1
841	1	1
595	1	1
420	1	1
297	0.9993	0.9997
210	0.9757	0.9817
149	0.8745	0.8783
105	0.7210	0.7139
74	0.5679	0.5563
53	0.4381	0.4270
37	0.3321	0.3233

Table 2C Selection Functions

Top Size (microns)	**Reference data** (Herbst & Rajamani, 1982)	**GRINDSIM Simulator**
2380	1.721	1.721
1680	1.717	1.717
1190	1.595	1.595
841	1.379	1.379
595	1.109	1.109
420	0.830	0.830
297	0.579	0.579
210	0.376	0.376
149	0.228	0.228
105	0.120	0.120
74	0.067	0.067
53	0.033	0.033
37	0.000	0.000

Table 3C Breakage Functions

Top Size (microns)	**Reference data** (Herbst & Rajamani, 1982)	**GRINDSIM Simulator**
2380	0.0000	0.0000
1680	0.3450	0.3450
1190	0.2135	0.2135
841	0.1373	0.1373
595	0.0923	0.0923
420	0.0628	0.0628
297	0.0437	0.0437
210	0.0304	0.0304
149	0.0219	0.0219
105	0.0150	0.0150
74	0.0105	0.0105
53	0.0080	0.0080
37	0.0190	0.0190