Towards More Effective Simulation Of Minerals Processing Systems

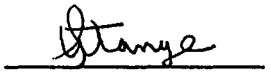
Wayne Stange

A dissertation submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering

Johannesburg, 1988

DECLARATION

I declare that this dissertation is my own, unaided work except where otherwise stated. It is being submitted for the Degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has never been submitted before for any degree or any examination in any other University.



11 day of August 1988

ABSTRACT

Two aspects of the computer simulation of minerals processing systems were investigated in order to facilitate more effective use of simulation technology.

A user-interface was designed and combined with an existing simulator executive, resulting in the implementation of a user-friendly microcomputer based minerals processing simulator, MicroSim. Ease of use was achieved by consideration of the needs of the user of such a program. This resulted in the use of graphical methods for information input and output. Efficient form-filling techniques were developed for numerical data entry and editing.

Models for the carbon-in-pulp adsorption process and for continuous gold leaching were derived. The CIP models were derived using a population balance approach. The method of characteristics and the method of moments were found to be particularly useful in solving the resulting equations. Besides being important processes in themselves, the integration of these models into MicroSim provided valuable experience regarding the use of such models in a simulator.

ACKNOWLEDGEMENTS

I would like to express my thanks to the following:

- My supervisor, Professor R.P. King for his assistance during the course of this project;
- The Anglo American Corporation and the CSIR for financial assistance received;
- My wife Debbie and my family for their support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Symbol Quantity

A	Parameter in linear isotherm
a,b	Freundlich isotherm parameters
b_1,b_2	Parameters in Brittan leaching rate expression
B	Birth rate due to discrete events
C	Concentration of gold in bulk of solution (mass of gold per unit mass of solution)
D	Death rate due to discrete events
E(t)	Vector of particle coordinates
F	Flowrate of ore solids
J	Number of internal particle properties
k	Mass transfer coefficient (Fleming-Nicol adsorption rate expression)
k_1, k_2	Adsorption and desorption rate constants (Dixon adsorption rate expression)
k_{max}	Maximum zero order leaching rate constant, Loveday expression.
k_p	Leaching rate constant, MINTEK expression
K	Mass transfer coefficient (Johns adsorption rate expression)
L(S)	Rate of leaching (mass of gold per unit mass of ore per unit time)
M^c	Mass of carbon in contactor
n	Schumann shape factor, Loveday leaching rate expression
N	Total number of stages in a series of CSTRs
NDC	Number of carbon size classes
$\hat{n}$	Unit normal vector orthogonal to surface S
P	Particle mass density function
$\hat{P}$	Mass density at fixed loading bound
Q	Slurry or aqueous phase volumetric flowrate
R	Region of phase space
R(y,C)	Rate of gold loading onto carbon (mass of gold loading per mass of carbon per unit time)
S	Surface enclosing volume V OR Concentration of gold by mass in ore.
S_m	Concentration by mass of refractory gold (not recoverable by cyanidation) in ore
S_o	Concentration of gold in ore at time zero.
t	Time coordinate

V	Reactor volume
V_a	Volume of aqueous phase in contactor
x,y,z	Spacial coordinates of particle
y	Concentration of gold on carbon (mass of gold per unit mass of carbon)
$\hat{y}$	Fixed value of loading class boundary
y^*	Capacity factor (Dixon adsorption rate expression)

Greek Quantity

δ	Carbon particle size
$\varepsilon_i(t)$	Magnitude of internal property i at time t.
$\varepsilon(t,\delta)$	Mass of carbon per unit volume of slurry at time t having particle size δ
Ψ	Particle number density function
$\bar{v}$	Vector of time rate of change of particle properties or position

Superscript Quantity

e	Denotes external particle coordinate
i	Denotes internal particle coordinate
-	Denotes averaging of quantity over reactor volume

Subscript Quantity

e	Denotes equilibrium conditions
i	Denotes i'th characteristic
x	Denotes stage number in a series of CSTRs
_	Denotes vector quantity

CHAPTER 1

INTRODUCTION

Much research regarding the use of steady-state simulation in investigating the behaviour of minerals processing systems has been performed. The benefits of simulation[1] are many (Ford and King 1984; King 1984; Lynch 1984; McKee 1984):

- The financial implications for different processing routes can be determined inexpensively and with a minimum of data.
- Flowsheet alternatives for existing plants as well as plants being designed, can be evaluated at minimum cost.
- The intricate interactions between the units making up a flowsheet can be studied, perhaps leading to a better understanding of the processes taking place.
- Interactive simulators make good training and educational tools.

The use of simulators in minerals processing has been made viable by advances made in the mathematical modelling of unit processes. At present, reliable mathematical models for a number of unit processes exist. A brief review of some of the modelling and simulation studies which have been published follows.

1 Note that in this dissertation the term simulation implies steady-state simulation unless otherwise stated.

1.1 Steady-State Simulation In Minerals Processing : A Brief Review

King (1974) developed a flotation plant simulator which could be used to design new plants as well as to improve the performance of existing plants. The flexibility of the simulator allowed many flowsheet configurations to be evaluated at much lower cost than that of performing experimental work.

Mular and Herbst (1980) reviewed some minerals processing modelling and simulation techniques. Examples showing how these techniques could be used to increase the efficiency of existing grinding and crushing circuits were presented.

Sutherland and Weller (1983) described how simulation techniques were used to evaluate the performance of a tin concentrator.

King (1984) presented several examples illustrating how simulation could be used to examine the performance of various circuits under different conditions. Examples for milling, hydrocyclone classification, coal classification (using dense medium cyclones) and sulphide flotation were given.

Kavetsky and Mckee (1984) published case studies demonstrating how simulation can be used to solve industrial problems. The investigation dealt with grinding and fines classification (using hydrocyclones). For both cases simulation helped in establishing the optimum flowsheet.

Tucker (1986) presented results showing how simulation aided in finding the optimum operating point for the preconcentration stage of a tin concentrator. The changes which simulation indicated should be made were implemented on the plant. These changes resulted in the benefits predicted by simulation.

A number of steady-state simulation packages have also been described in the literature. Many of these packages can be classified as flowsheet solvers. These packages do not predict the performance of plants, but utilize the power of the computer to perform flowsheet mass balance calculations in much shorter times than manual methods require. Examples of these packages are

GEMS (Izurieta and Edwards 1980), FLEXMET (Coles 1982) and Genflow (Woollacott 1982). As these packages are not simulators in the true sense of the word, they will not be considered further.

The Department of Energy, Mines and Resources, Canada (CANMET) has developed a suite of modelling and simulation software called SPOC (CANMET 1981). Written in FORTRAN, these programs run on both mainframe computers and the IBM PC type microcomputer. The programs are specifically for use in the areas of coal preparation and ore-dressing.

Ritchie and Spencer (1984) extended the chemical engineering flowsheeting system ASPEN PLUS (Evans et al 1979) so that the simulator can simulate minerals extraction processes. This is a powerful simulator and can simulate ore-dressing and hydrometallurgical processes. It is a large system (320 000 lines of FORTRAN code), and at present runs only on mainframe and minicomputers.

The Julius Kruttschnitt Mineral Research Centre has developed a package which allows simulation of ore-dressing flowsheets (Mckee 1984, Hess and Wiseman 1984). This simulator runs on IBM AT type microcomputers.

MODSIM (Ford and King 1984) is a sequential-modular ore-dressing simulator written in FORTRAN available for microcomputers and mainframes. The simulator can accommodate very complex as well as simple unit process models. Interactive graphics are used to describe the process flowsheet.

Warren Spring Laboratory are developing a simulation package, MINSIM (Cutting 1984, Tucker 1986) written in FORTRAN 77 for IBM compatible microcomputers. Each unit model in the package can be used as a stand-alone program or the models can be linked together to simulate flowsheets. Only ore-dressing processes can be simulated.

1.2 Use Of Simulation In Industry

It is obvious that simulation can most profitably be exploited by industrial users. This is due to the significant economic benefits which simulation has the potential of providing. These benefits take the form of more efficient designs as well as facilitating more efficient plant operation and off-line optimization.

Despite the many benefits listed above which simulation can provide and despite the fact that suitable models for many unit processes exist, simulation is not widely used in the minerals processing industry (Cutting 1984; King 1984; Lynch 1984; McKee 1984).

A number of reasons are responsible for the limited use of simulation. The most obvious is that specialised skills are required to utilise simulation effectively. Due to a shortage of engineers who understand the application of simulation and modelling technology to minerals processing, these skills are not widely available in industry.

Sophisticated simulation programs such as those described in section 1.1, contain the required technology in a form which can be used without specialized knowledge. The necessary technology transfer can be effected by making such simulation programs readily available to industry.

However, any software which is provided for mass use must be easy to use. Many authors (Adel and Sastry 1982; Cutting 1984; Mckee 1984; Sastry and Adel 1984) are of the opinion that simulation software is of limited use if metallurgists find the packages difficult to use. In such a case, traditional methods will continue to be used and the benefits of the use of computer methods will not be experienced.

Another factor which limits the use of simulation in industry, particularly in South Africa, is the limited range of unit processes for which simulation packages cater. All the packages described in 1.1 (with the exception of the ASPEN based simulator) only allow the simulation of ore-dressing circuits.

Due to the large number of hydrometallurgical unit processes used by the South African industry, the available simulation packages are obviously not very useful.

1.3 Objectives Of The Research Project

Summarising the literature discussed above, it is concluded that:

- Modelling techniques for many ore-dressing unit operations are well established. The use of these models can provide real benefits even for complex industrial problems.

- A number of simulation programs do exist. However, the majority of these are capable of simulating only minerals dressing unit operations.

- If a simulator is to be used effectively in the industrial environment, the simulator must be designed so that no special simulation skills are required by the user of the program. Due to the relatively short time which computers have been available to non-specialists, simulation packages should be designed so that even people having minimum computer experience can use the simulator.

The objective of this research project is to make some contribution to techniques which will allow more effective use of simulation, particularly in industry. The work described in this dissertation concentrates on two areas. The development of these areas are seen as essential to the attainment of this objective.

1.3.1 A user-friendly simulator

Increasingly the most important factor determining the usefulness of a software package is the user-friendliness of the program (Maguire 1982; Gaines and Shaw 1986; Tucker 1986). This dissertation describes how established principles of human-computer interface design can be used to construct a user-friendly interface for a minerals processing simulator.

More specifically, it will be shown how these techniques can be used to develop a highly interactive, integrated microcomputer based simulation package. This results in a package which is easy to use for people with little or no expertise in simulation technology and/or computing, while allowing people with more knowledge of simulation to customize and expand the simulator to meet their requirements.

1.3.2 Techniques for hydrometallurgical process simulation

In section 1.2 it was noted that few simulators are capable of simulating hydrometallurgical processes. If simulators are to become popular in South Africa, the programs must be capable of simulating all unit processes which are found in typical flowsheets.

This research contributes toward the development of hydrometallurgical modelling by the derivation of models for the carbon-in-pulp (CIP) adsorption process. The integration of the CIP model into a simulator is also described. This process was chosen as CIP has become the favoured route for gold extraction in countries like South Africa, Australia and the U.S.A. Thus significant economic benefits would result from better design and operation of such plants.

It was hoped that experience gained in the derivation of this model and the subsequent integration of the model into a general simulator would provide some guidelines for the further development of hydrometallurgical models for use in a general simulation program.

1.4 Strategy Used To Meet Objectives

Cilliers (1987) has developed an executive structure for a general purpose sequential-modular simulator. This program is written in Pascal and utilises flexible data structures. These structures were designed to facilitate the design of records which allow adequate description of the various stream components (ore phase, aqueous phase containing lixiviants, carbon phase, organic phase and so on) which may be present in minerals processing flowsheets. Cilliers also developed the flowsheet tearing and ordering routines required by a sequential-modular simulator.

The executive developed by Cilliers was used as the basis for the implementation of a microcomputer based simulator called MicroSim (Stange, Cilliers and King 1988a,b). This was accomplished as follows.

- The requirements for an efficient user-interface and current interface technology were reviewed. This is discussed in Chapter 2.

- The implementation of the MicroSim user-interface based on the principles presented in Chapter 2 was carried out. This is discussed in Chapter 3.
- A review of current CIP models was carried out. This is described in Chapter 4. It was discovered that existing modelling techniques were not entirely suitable for the development of a CIP model for MicroSim. Thus a model, based on the population balance technique (Hulbert and Katz 1964), was derived for the CIP process. The derivation of this general model is presented in Chapter 4.
- As the model derived in Chapter 4 requires much computational effort, a CIP model more suited to industrial use was derived. The derivation and integration of the model into MicroSim is described in Chapter 5.
- Chapter 6 presents the conclusions drawn from the above research, as well as recommendations for further research.

CHAPTER 2

METHODS FOR THE DESIGN OF AN EFFICIENT USER-INTERFACE FOR A MINERALS PROCESSING SIMULATOR

In this chapter, methods described in the literature which can be used for the design of efficient user-interfaces, are reviewed. Based on these methods, some techniques which can be used specifically for a minerals processing simulator will be discussed.

2.1 Introduction

The term "user-interface" has appeared often in recent literature describing minerals processing simulators (Hess and Wiseman 1984; Lynch 1984; Mckee 1984). The user-interface is however much more than the "look" and "feel" of a particular piece of software. It is also more than the way in which the program allows the user to enter numeric data, or the way in which menus are implemented.

The user-interface may be viewed as that part of the program which communicates the actions of the user to the rest of the package so that the desired user action may be performed. A good user-interface will allow the user to communicate desired actions in a simple and intuitive manner.

The user-interface provides the objects and methods which a user requires for the efficient execution of the tasks that the program is designed to carry out. These objects and methods can be provided by the interface only after a careful analysis of the tasks a user needs to perform has been carried out (Gaines 1981; Rich 1983). A user-interface may also incorporate such elements as the computing environment and documentation (Evans 1980).

2.2 Methods For The Design Of The User-Interface

2.2.1 The user model : Characterization of the user of a minerals processing simulator

The accurate characterization of the user of the software is the most important single step in the design of an efficient user-interface. This is called the user model. The user model encapsulates the procedures a user would normally use to solve the problem. It also considers the users requirements, expectations, and terminology.

The importance of incorporating the users model into the design of an interface is emphasized by a number of researchers in this field (Gaines 1981; Maguire 1982; Rich 1983). In order to design an efficient interface, it is thus necessary to characterize the expected user of the system.

The users of a steady-state simulation package such as MicroSim can be split into two groups. One group consists of engineers using the package for routine simulation studies. They would be satisfied with using the models and facilities provided with the simulator. In general they would not be interested in extending the capabilities of the simulator. If this needed to be done, they would have recourse to a person classified as being in the second group. Computer skills of members of the first group of users would range from unskilled to expert.

The second group consists of people actively involved in simulation technology. For example, the modelling of various unit processes and the development of new simulation techniques (e.g. tearing and ordering or convergence algorithms). A simulator which is easy to extend would prove useful to this group of users. Models and algorithms could then be developed, inserted into the simulator and used without the need to develop supporting code. In general this group of users would be more familiar with computers, especially the programming aspect, than the first group. This does not imply that the use of the package (including the insertion of new

models) for the second group should be considerably more complicated than the difficulties experienced by the first group when using the package for routine simulation studies.

The dual nature of the potential user of MicroSim has far reaching implications for the design of an efficient interface. Both groups of users require a package which communicates effectively while the simulator is being used in the conventional manner (i.e. as any packaged software would be used).

The second group require facilities which allow the insertion of new models into the existing simulation structure with a minimum of effort. Also, existing features of the package should be accessible, so that existing code may be used effectively.

In effect a dual interface exists, the external interface being that which is evident when the package is being used for routine simulation studies, the internal interface being the structure of the source code and quality of technical documentation i.e. the part of the simulator that users in the second group need to "communicate" with when extensions to the simulator are being developed[2].

For the second group of users, the minimum requirements for the internal interface are that users should be able to insert their own unit models and graphics icons into the simulator. For a simulator such as MicroSim which has flexible and powerful data structures to describe the type of material which exists in a process stream (Cilliers 1987), users should be given the facility to design and implement their own streams data structures.

Characterization of the user of a program does not imply that the user-interface for the program has been defined. A number of complex issues still have to be resolved. The following section describes some methods which may be used to deal with these issues on a more formal basis.

---

2 Note that the term "user-interface" refers only to the external interface in this dissertation. When referring to the internal interface this will be stated specifically.

2.2.2 Interface structure

In the design of a user-interface it is convenient to view the interface as being made up of two parts; architectural presentation and architectural form (Carroll 1983).

Architectural presentation

Architectural presentation is concerned with the nature of the elements which make up an interface. The program functions available to the user are presented through these interface elements. The interface elements are therefore the actions and the objects with which the user interacts while using the software. These actions and objects are often called metaphors.

An architectural presentation issue may be whether to use graphically presented interface metaphors or to use text to present certain interface elements. For example, at a certain stage in a program a user might have the option to save a document to disk, edit the document or print the document.

If graphically presented metaphors were used, the user would be presented with a number of icons representing the choices available (saving, editing and printing). In order to select the correct choice, the user is required to move a cursor (normally using a mouse) to the icon representing the desired action.

If text metaphors were used, the user might be presented with a menu containing a text description of the available options. Selecting the correct menu option would result in the appropriate action.

The success of the graphic icon approach depends on how effectively the icons represent the actions to be performed as well as the graphics capability of the computer on which the program is run. If efficient graphics input devices (mouse, light-pen) and high-resolution screens are available, this method of presentation can be very efficient.

Some cases of architectural presentation issues in flowsheeting software will be discussed so as to elucidate the factors influencing the choice of method used in resolving these issues.

Woollacott (1982) has developed a mass balance package which uses "concepts" as the presentation objects used to communicate with the user. He identifies three types of concepts:

a) Graphical concepts used during flowsheet drawing.

b) Functional concepts such as "retrieve a data file", "print the results" and so on.

c) Technical concepts which describe the metallurgical aspects of the package.

These concepts are activated using labelled function keys. Due to the complexity of the concepts which the user uses in communicating with the package, it is not likely that the icon approach would work as it would be difficult to provide icons which would be instantly recognisable by all users to represent the required concepts.

Of the packages which have flowsheet drawing capabilities and which are described in sufficient detail in the literature (Woollacott 1982; Hess and Wiseman 1984; King 1985; Preece 1986), three use a menu approach to select graphical and other operations to be performed.

In the case of the JKMRC simulator (Hess and Wiseman 1984) and PFG/PIG (Preece 1986), the menus are displayed alongside the graphical representation of the flowsheet. Genflow (Woollacott 1982) also uses a menu based system for flowsheet drawing. Function keys below the graphics area are used to select the various graphics operations required. Unit selection however, is done from a separate full screen menu, necessitating the erasing and reconstruction of the flowsheet.

MODSIM (King 1985) uses command-driven dialogue during the flowsheet drawing phase. The graphics screen is then cleared and input of alphanumeric data proceeds using a question-answer type dialogue.

The above are all examples of different architectural presentation approaches used for various packages which allow a graphic description of a flowsheet to be specified by the user.

Architectural presentation issues are normally resolved on a case-by-case basis (Carroll 1983). It is difficult to make any specific recommendations concerning the presentation aspect of the user-interface as these aspects are influenced by available software tools (graphics routines etc.), the hardware on which the software is to run (microcomputer versus mainframe), the complexity of the software and so on.

Architectural form

Architectural form deals with the interrelation between the various functions provided by the system for a typical user scenario (the 'user model' mentioned in section 2.2.1). It is possible to address the issues involved in the organization of the interface (architectural form) on a more general level than for architectural presentation (Carroll 1983).

For example, should an interface be sequential, user-driven[3], or employ elements from both domains? The architectural form chosen has a significant effect on the flexibility and ease-of-use of the program as well as on the expansion capabilities of the system. Careful consideration of the user model allows such issues to be resolved objectively. This will be demonstrated in Chapter 3, when the design and implementation of the MicroSim interface is discussed.

Methods of numeric data entry can also be considered as part of architectural form in packages such as simulators which require large amounts of numeric data to be entered and edited. At first it may seem more natural to relegate data entry to architectural presentation. Some explanation as to why data entry is considered an architectural form issue will be given.

3 User-driven software is software which allows the user to access any of the available functions in a random order at any time. Sequential software performs the required functions in an order over which the user has little or no control.

Two basic techniques may be used for data entry, these being form-filling and question-answer dialogue.

In form-filling, the user is presented with a facsimile of a 'form' on the computer screen. The form may resemble a document with which the user is familiar. The form contains text prompts which describe the numeric fields on the form. Default values are normally presented in these numeric fields. The user is able to position the cursor at any of the fields and enter appropriate data. Once the user is satisfied with all the information entered on the the form, it is submitted to the program.

Question-answer dialogue normally relies on a sequential series of text prompts to which the user responds by entering the required data. Data is normally entered item by item.

It is not only the architectural presentation of these techniques which are different. They differ significantly with respect to the way in which they influence the confidence of a user in his/her ability to use the package, especially during the early learning stages. Form-filling has significant advantages over question-answer dialogue.

Miller and Thomas (1977) state that the provision of a default when prompting for data is "one of the most powerful of existing computer system concepts for achieving a user-oriented environment". Form-filling is the ideal vehicle for providing defaults. These default values give some indication as to the type of values the system expects. The user may easily replace these values by data specific to the problem being solved. These defaults form a valuable guide for the inexperienced user. Using this system a simulation can be set up and executed with very little data supplied by the user as defaults suffice when the user is inexperienced or merely exploring the capabilities of the system (without wishing to solve a specific problem).

Kennedy (1974) also recommends providing defaults where possible. However he expresses misgivings concerning form-filling which are related to the problems inherent to the implementation of this dialogue style on mainframe systems of that era. These problems arise if no

interaction between the user and program is possible while the user is filling in the form. Error checking is then only possible when the completed form is submitted to the program. If an error is detected, it may be necessary to clear the screen to inform the user of the error, who cannot then see the original input, making error correction difficult.

These problems are easily overcome on microcomputers as they are totally interactive and each keystroke made by the user can be checked almost instantaneously. Error correction is therefore made simple. For example, if the field the user is filling in is a numeric field, then all non-numeric keystrokes can be filtered out, aiding the inexperienced user who may be having "finger trouble". Thus there is normally no need to clear the screen when the user makes an error. Instead the computer may produce a warning message when the user strikes an incorrect key.

If the data entered must satisfy some constraint (e.g. if a size distribution is being entered, the sum of the entries is well defined) and if the data entered does not satisfy the constraint it is possible to alert the user to the error and then display a form whose defaults are the original incorrect data entered by the user. The incorrect entry can then be rectified quickly and easily. Question-answer style dialogue would normally require all the data to be re-entered.

The form-filling technique is also very powerful when applied to the editing of data currently being used by the program. The data item(s) to be edited may be chosen from the form, avoiding the need to re-specify all the data items in the particular data record being edited. For example, assume that the parameters defining the operation of a milling model are to be edited. Form-filling allows all the model parameters to be presented to the user simultaneously. The user simply moves the cursor to the appropriate item and changes it. With question-answer dialogue, the user would first have to indicate which parameter is to be changed and then enter the new parameter.

The arguments presented above show how form-filling may be used as an integral part of a user-interface. It can be concluded that where hardware allows, form-filling should be used for data entry due to the significant advantages it has over more conventional methods.

2.2.3 Heuristic issues in interface design

Much of the design of a good user-interface is still an art acquired after much experience. Some empirical rules which have been recommended for interface design will now be discussed. Where possible, interpretation of these rules for the interface of a minerals processing simulator will be made.

Many of the more practical guidelines were obtained from the paper **Hints for Computer System Design** (Lampson 1984). To quote from the paper "This collection of good advice and anecdotes draws upon the folk wisdom of experienced designers".

Accordingly an interface should:

1 Be intuitive and consistent. As far as possible command sequences in different parts of the program should be functionally equivalent. The next action to be taken by the user should be almost immediately obvious. This reduces the chances of the user making errors.

2 Not impose alien work habits on the user. This implies that the terminology and techniques which a metallurgist normally uses should be supported by the simulator. Graphical techniques must be used to input the flowsheet to be simulated, as this is the most natural way for a metallurgist to describe the process. In other areas of data input, such as when defining the properties of the ore, familiar terminology must be used. When alternatives are available a choice should be provided if possible. It should be noted that it may be impossible not to include some terminology and concepts which will be unfamiliar to some users. For example, the user may have to specify "a bound for the Wegstein convergence method." Appropriate defaults must be provided so that the user may accept these if it is not understood what information is required.

3 Output information which may be used by the metallurgist without further data processing. For example, plots of size distributions and partition curves should be available, not just tabulations of size distributions and recoveries. This enables the rapid interpretation and evaluation of the data output by the simulator. Quantities should be used with which the user is most familiar eg. "recovery", "percent solids" etc.

4 Be flexible. One of the strengths of steady-state simulation is the quantification of the "what-if" situation where various flowsheet configurations and operating parameter settings are investigated. Changes to the data defining the problem need to be carried out quickly and easily for the simulator to be of maximum efficiency in this application.

5 Be totally independent of operating system software. The user should not have to exit the package in order to perform such tasks as file copying and deleting. Once the package is loaded it should function as an integrated system with all features needed by the user being accessible from within the package. This leads to fast turnaround times, improved productivity, as well as a high degree of interaction between the user and the computer.

6 Have high contextual dependence. That is, the range of functions or choices available at any time is restricted by the mode in which the user is currently working. This leads to less confusion on the part of the user and promotes ease of use (Carroll, 1983).

7 Be fault tolerant. For example, if a letter is entered in place of a number the program should not crash. This may occur frequently for new users with "finger trouble" and can lead to frustration if the program has to be restarted every time a minor error such as this occurs.

8 Have an acceptably fast response time. A fast response time causes the least disruption in the thought processes of the user. The definition of "acceptably fast" depends on the complexity (as the user sees it) of the task the computer has been asked to perform. This can be linked to the concept of "psychological closure" which is the feeling of satisfaction

experienced when a task has been completed (Maguire 1982). The more complicated the task being performed, the stronger the closure and the more tolerant the user will be of a delay. For example, the user would be prepared to accept a fairly lengthy delay when the simulator is calculating the circuit mass-balance, but would expect an almost instantaneous response when issuing a command to draw a unit icon with the flowsheet editor, even though the internal coding required to draw the icon may not be trivial. If a long response time is unavoidable, for example during the mass balance calculation of a complex circuit, interim messages should be provided in order to assure the user that the computer is working. This is particularly important for microcomputer systems with their limited processing power.

9 Not give the impression that the response to a question is final and unchangeable. If possible, a user should be able to change any data (graphical or alphanumeric) without restarting. It may be exceedingly difficult to design a program where any data item can be changed. This is especially true in a simulator where changing one data item may have a snowball effect due to the dependency that much of the other data have on the item to be changed. For example, if the user decides at an advanced stage to increase the number of size classes used to define the problem, the distribution of the solids in their various classes (size, particle-type) in each feed stream needs to be re-specified. However, it is reasonable to expect the system to have the capability of editing (from within the package) any data item which will not cause this snowball effect.

10 Not be acausal. Each action by the user should cause a response. If the chosen function is valid, the interface must be seen to be responding. If the function is not valid, the user should be informed via a suitable error message. Certain functions may be valid only in a certain mode. These functions should not be accessible from anywhere but the correct mode.

11 Allow the user to perform the task for which the program is designed with the minimum of unnecessary distractions. For example, a user must

be able to perform any task with the minimum number of keystrokes and without having to perform a number of actions in order to accomplish a simple task.

2.2.4 Internal interface requirements

Sections 2.2.2 and 2.2.3 have described requirements and design methods for the external interface; the requirements for the internal interface will be discussed briefly here.

Three separate but overlapping areas influence the efficiency of the internal interface. These areas all affect the ability of a user to understand the structure of the simulator at the source-code level. These three areas are:

Technical documentation which illustrates the structure of the simulator, the relevant data structures, the methods used to construct graphics icons, and the way in which mathematical models are inserted into the source-code must be made available. The above concepts should be illustrated in the manual step by step using simple examples.

Structured programming techniques must be used. This enables the program to be broken down into a number of separate modules which do not interact. Changes to any module should not require changes to the others (if possible). This is difficult to accomplish when changes or additions to streams data structures are to be made as these data structures are applicable throughout the entire program. However when unit models and icons are to be inserted the user should have to modify only the appropriate modules.

Source-code must be written with the aim of making the code easily readable and understandable. Methods of doing this include the use of profuse and meaningful comment statements as well as descriptive variable, procedure and function names.

2.3 Conclusions

This chapter has summarised some of the techniques for user-interface design found in the literature. Heuristic and more formal methods were described. The single most important aspect is the accurate characterization of the user.

The next chapter shows how the user model may be developed based on the characterization of the user in section 2.2.1. This model is then used as the basis for the design of the structure of the MicroSim user-interface.

CHAPTER 3

THE DESIGN AND IMPLEMENTATION OF THE MicroSim USER-INTERFACE

This chapter will describe the implementation of the user-interface for the MicroSim simulator. The chapter begins with the derivation of a user model for a minerals processing simulator. It will also be shown how the principles of interface design discussed in Chapter 2 were used in the design and implementation of MicroSim.

3.1 A User Model For A Steady-State Process Simulator

As discussed in 2.2, the most important step in the design of an efficient user-interface is the derivation of a valid user model. A typical user of MicroSim was described in section 2.2.1. The user model may be seen as the construction of a user scenario which will be typical of most conditions under which the program will be used. The user model then allows the interrelationships between the system functions (the architectural form of the interface) to be designed in a rational manner.

For a minerals processing simulation program the following user scenario is proposed.

For a new simulation problem, the steps a user is assumed to follow are:

1) The process flowsheet to be simulated is defined.

2) The data defining the simulation is entered.

3) The simulator is then used as a "what-if" tool. That is, the effects which changes in the various data items that define the simulation have on circuit performance are evaluated. The process flowsheet is not changed.

4) Exit.

For second and subsequent sessions, the steps a user follows are as follows:

1) The process flowsheet data is recalled and displayed. If desired, changes are made to the process topology.

2) If changes to the flowsheet were made and this results in the need for more data (e.g. the addition of a new unit) the user supplies the required process data.

3) The "what-if" situation is entered.

4) Exit.

In effect the user model defines the form of the interface. The user model described above results in the interface structure illustrated in Figure 3.1.

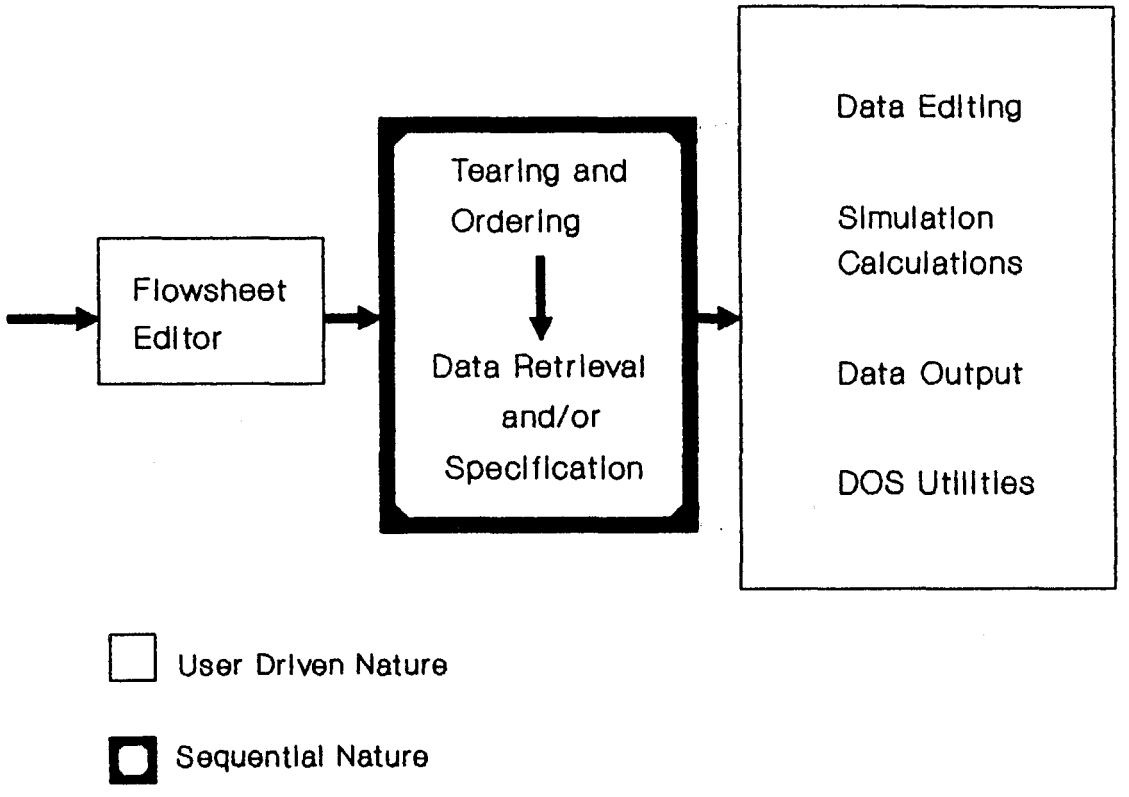
As can be seen from Figure 3.1, seven separate modules make up the simulator. The flowsheet editor and the tearing and ordering modules are used only in the first stage of the user scenario; flowsheet generation or modification, and analysis.

The next module, the data input/retrieval section, is then invoked. The other modules, namely the data editing routine, the calculation routine, the output routine and a collection of operating system utility commands are then all available for the user to use in the "what-if" phase.

These modules were implemented as three separate programs; a flowsheet editor, a tearing and ordering program (Cilliers 1987), and then the body of the simulator which has routines for data specification, retrieval, editing and output, simulation calculations and utility commands.

Data is transferred from program to program via disk files. Provision is made so that the user does not have to know anything about the structure and contents of these files.

Figure 3.1 : MicroSim Interface Structure



The simulator is also structured so that it is difficult for a situation to develop where there is inconsistent data in the files. If the files were to contain inconsistent data, it is very likely that the system would crash during the calculation stage.

The simulator has only one entry point, through the flowsheet editor. This has the disadvantage that even if no flowsheet changes are to be made, the flowsheet is still displayed. Thus the user is always made aware of the topology of the flowsheet being simulated.

Tearing and ordering always takes place once the flowsheet editor is completed. Output from the tearing and ordering phase will thus always be consistent with the flowsheet which the user has constructed or modified.

A sequential data input/retrieval phase is then entered. Any changes to the flowsheet are noted, and if new data is required the user is prompted for the data.

The three steps above ensure that by the time the user reaches the interactive section of the simulator, where most time will be spent performing the simulation calculations, a consistent data set has been defined. Although this program structure is rigid and may cause minor irritation, it ensures that a consistent data set is defined, eliminating many problems which could cause the calculation phase to crash or to diverge.

Before the design and implementation of each major section of the simulator is described, the hardware on which the simulator is implemented and the reasons for this particular choice of hardware will be reviewed.

3.2 Hardware Choice For The MicroSim Simulator

The correct choice of hardware is an important requirement if the simulator is to become widely used in industry. It is not likely that many industrial organizations would purchase dedicated hardware in order to run the simulator package. Thus the hardware chosen had to be widely available and have the capability of running other useful software. The hardware chosen should also have sufficient power, in terms of memory capacity, disk storage capacity, graphics capability, and processing speed.

Microcomputers have several advantages over mainframes and minicomputers. These include low cost and ease-of-use. Microcomputer users have full control over the computing environment, resulting in less stress for unskilled computer users.

The importance of the user-friendliness of software has already been stressed (sections 1.2 and 1.3.1). Microcomputers have ideal properties (both in hardware and software terms) which allow the development of extremely interactive, easy-to-use software.

Microcomputers also have disadvantages, the most important being their slow processing speed as compared to mainframes and minicomputers. However, the processing power of microcomputers is increasing at a rapid rate, with recent developments leading to machines which can only be described as "desktop supercomputers" (Christ and Terrano 1986). The current generation of microcomputers are capable of solving complex simulation problems in relatively short times. This is well illustrated in Appendix 2, where some examples of complex simulation calculations are presented.

Hardware which best fulfils the above requirements is the IBM PC type microcomputer. The IBM PC and compatibles have become very popular and most industrial and research organizations have ready access to this type of computer. There is also a extensive base of very useful software available for this type of computer.

Due to the large investment in existing software, it is likely that most new microcomputers developed will be downwardly compatible with IBM PC software, guaranteeing a long life for IBM PC software. This has been confirmed with the appearance of the IBM AT type computer as well as the newer 32 bit systems such as the IBM PS/2 model 80. These computers are significantly more powerful than the PC family. However they exhibit a high degree of compatibility with previously developed software.

It was thus decided to implement MicroSim specifically for IBM PC type microcomputers.

Concerning the choice of language, as the simulator executive was coded in Pascal (Cilliers 1987), the language in which the interface was to be implemented was for all intents and purposes, pre-specified. Due to the availability of good software tools for this language on the IBM PC, and due to the structured, strictly typed nature of Pascal this was an advantage rather than a disadvantage.

The following sections of the chapter will describe how MicroSim was implemented. The flowsheet editor will be described first.

3.3 The Flowsheet Editor

The use of graphics has become widespread in modern microcomputer software due to the availability of inexpensive graphics hardware. This is one of the most effective means of communicating complex data. A graphics based flowsheet editor was thus designed for the MicroSim simulator, enabling the metallurgist to describe the flowsheet in a natural manner. This editor is a stand-alone program capable of providing flowsheet data to other metallurgical software.

At present, the PC version of MODSIM (King 1985), the well known ore-dressing simulator, uses this editor to obtain flowsheet information. This editor is also suited for use with other metallurgical programs such as mass-balance smoothing packages.

3.3.1 Design considerations

Two requirements exist for a general purpose flowsheet editor. Firstly, the editor must be capable of deriving all the required metallurgical information from the graphic information provided by the user. Secondly, the objects and actions which the user requires to complete the task of drawing a flowsheet must be provided. These capabilities must be provided in a way which is efficient and intuitive.

Flowsheet process data required

Firstly, basic information such as the number of streams and the number of units which make up the flowsheet and the stream connection matrix must be acquired. This data is trivial to obtain and will not be discussed further.

The characteristics of each unit in the flowsheet must also be obtained. The amount of detail in the information required varies depending on which software is linked to the editor. A mass balance smoothing program only needs to know if a particular unit is destructive[4] or not, while a simulation program needs to know the precise characteristics of a unit. These include the number of stages which make-up the unit if the unit process is a staged one (flotation, leaching, CIP etc.), as well as which streams are connected to each feed and product point of the unit.

Streams may also have complex characteristics. Consider for example a hydrometallurgical flowsheet. The phases which may be present include ore particulate solids, organics liquids, carbon particulate solids and the aqueous phase which may contain many dissolved species. The MicroSim data structures are based on the concept of substreams (Cilliers 1987). That is, each stream in the flowsheet consists of a combination of substreams. For example, a slurry stream is made up of solids and water substreams. A carbon stream however is made up of only the carbon substream type.

Each stream in the process may consist of only one or a number of the above mentioned substream types. The strength of the data structures derived by Cilliers (1987), is the efficient way in which the information for complex stream types may be stored in memory.

It is only possible to use this scheme efficiently if the procedure which allocates storage space for each stream knows exactly which substream types make up each particular stream. If this is not known, it may be

4 A unit may be destructive with respect to quantities which are measured in order to establish the mass balance. For example, particle size is not conserved around a comminution unit, therefore size distribution data cannot contribute to the establishment of a mass balance around a comminution unit. Comminution units are thus destructive with respect to particle size.

necessary to allocate each substream type to each stream. Inefficient stream type classification may have serious effects on both memory utilization and calculation convergence (Cilliers 1987).

The ASPEN simulator, which also uses the substream concept, requires the user to define the substream types (Evans et al 1979). Although this is an acceptable solution, it requires more competence on the part of the user. A solution which allocates the exact substream types to each process stream without the need for intervention by the user is desirable.

In principle it is possible to allocate the correct substream types at the flowsheet drawing stage if it is assumed that the required substream types are a function only of the behaviour of the unit process. That is, the mathematical model used to simulate the unit process has no influence over which substream types should be allocated. Cilliers illustrates the difficulties which may arise when two different mathematical models are used to simulate the filter unit process. One assumes solids are present in the filtrate while the other does not. This leads to the problem that one model expects the filtrate stream to consist of the ore solids and lixiviant substream types, while the other expects the filtrate stream to have only lixiviant characteristics.

This problem can be resolved by defining the stream types which connect to each feed/product position of all units in a sufficiently general way so that all possible mathematical models for that unit process can be accommodated. The filter unit would thus be defined as having ore solids and lixiviants in the filtrate stream. The model which assumes no solids in the filtrate would then place zero values in the solids record of the filtrate stream.

By taking this approach, the requirements for each unit model are well defined, reducing possible confusion when new models are inserted into the simulator. This approach makes it possible for the flowsheet editor to allocate the correct combination of substream types to each stream in the flowsheet. The methods used to achieve this are presented in section 3.3.2.

Flowsheet graphical editing functions required

The graphical representation of a flowsheet consists of three basic elements. These are the unit icons, stream vectors and labels. Although labels provide no metallurgical information from a mathematical or simulation point of view, the ability to label a flowsheet quickly and neatly is a basic requirement of any flowsheet editor.

The basic functions needed to manipulate labels, streams and units are not numerous and are of two types i.e. the generation of streams, labels and units and the deletion of streams, units, and labels. Repositioning of these elements is a combination of the generation and deletion primitives. More important is the way in which these functions interact as this determines the architectural form of the flowsheet editor (see section 2.2.2).

While drawing the flowsheet, the user will not use any well defined sequential procedure. This means that the flowsheet editor needs to be totally user-driven. That is, the interface remains passive, action being taken only when a user issues a command. The implications of this are that any and all commands related to the manipulation of units, streams, or labels need to be accessible simultaneously, i.e. a hierarchical series of commands must not be traversed to get to the function required. Commands should be as consistent as possible, the command procedure which deletes a label being similar to that which deletes a unit or stream and so on.

3.3.2 Implementation of the flowsheet editor

The considerations mentioned above must be implemented on different levels. The area of process data requirements can be attacked only at a fundamental data structure level. That is, the efficient design of data structures for the representation of the process data has a major influence on the flexibility and ease of maintenance of the editor. As little precise information concerning the requirements for hydrometallurgical simulation

was available during the development of the editor, the principal consideration was to develop a data structure which is flexible enough to cope with almost any situation.

Editing functions which are powerful (in that the user is able to perform any sub-task with a small number of keystrokes) but simple to use are required. The flowsheet editor was provided with a small but complete set of editing functions.

Data structures for the representation of process data during the flowsheet drawing phase

In order to meet the requirements that the editor be capable of coping with almost any situation, the MicroSim editor uses the concept of "intelligent units" and "latch types". When a unit is drawn, the editor associates a number of latches with that unit. The number of latches associated with the unit is a function only of the unit type.

These latches contain the graphical coordinates of the latch, as well as the latch type. The latch type is simply a descriptor of which stream type may be attached to that unit at that particular feed/product position. There is no restriction on the number and types of latches which a particular unit may posses. This allows the editor to cope with units having many feed and product streams of varying types.

Consider Figure 3.2. This illustrates how latch types would be allocated for a hydrocyclone and a CIP adsorption cascade. By convention, each latch type represents a well defined stream type and the associated feed or product position. For example the number 1 represents a unit feed stream which contains ore solids and water, while 3 represents a stream which also contains ore solids and water but which leaves the unit as a unit product (3 signifying the underflow in the case of the hydrocyclone).

Thus all information required for the classification of substream types in each process stream is available during the flowsheet drawing phase.

Figure 3.2 : Illustration Of Latch Scheme

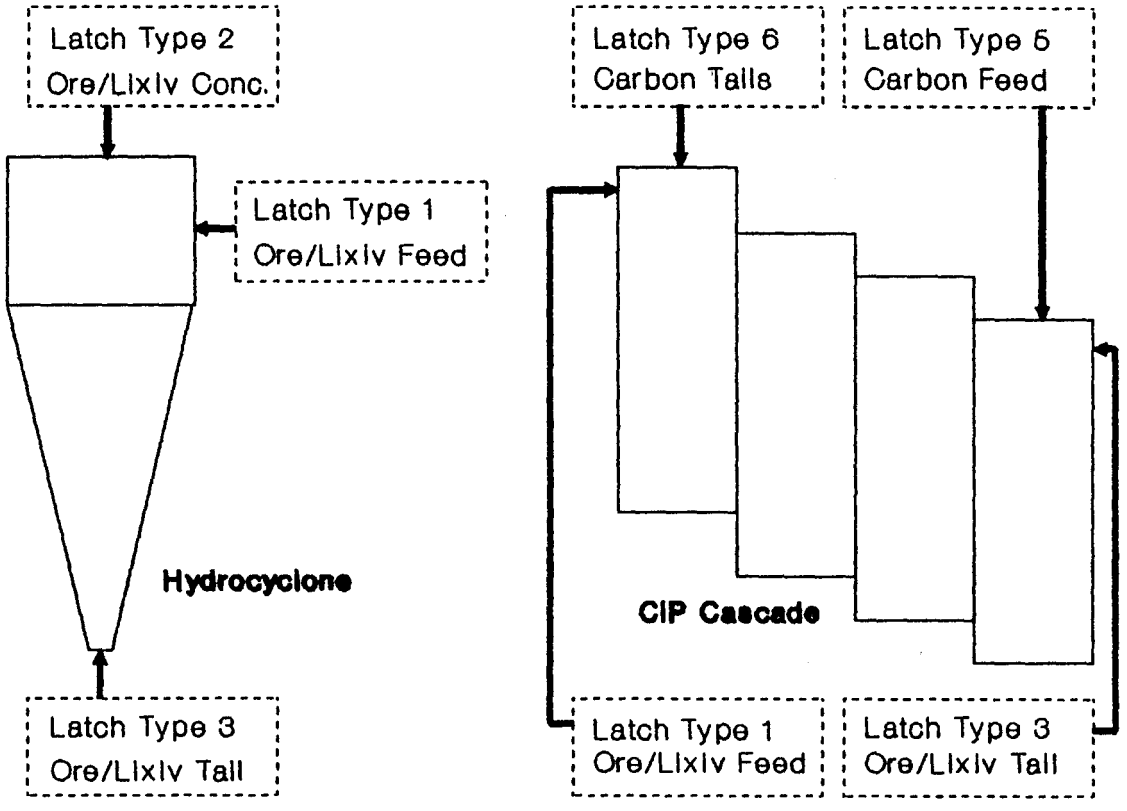


Figure 3.3 : Architectural Presentation Of Flowsheet Editor

The flexibility provided by the latch data structure should enable all future requirements to be met. The specific data structures used in the MicroSim flowsheet editor and their Pascal type declarations are described in Appendix 1.

Implementation of architectural presentation in the flowsheet editor

The appearance of the flowsheet editor user-interface is illustrated in Figure 3.3. As stated in 2.2.2, there are few general principle for the architectural presentation of a user-interface. In this case, hardware considerations played a dominant role in the implementation of the flowsheet editor.

Due to the low graphics resolution of the IBM PC type microcomputers, the screen area used for flowsheet drawing was maximized. This is labelled as area 1 in Figure 3.3. Area 2 is the status line which provides information such as the number of units and streams in the flowsheet, and the last command issued by the user. Area 3 displays function key labels for commands accessed via the function keys. These consist of commands to access the help screens and commands which allow the user to move the graphics cursor to the nearest stream, unit, latch or label ('find' commands').

Means of starting and ending streams, positioning, scaling and deleting units, and placing and deleting labels were implemented. Each operation is carried out at the position defined by a graphics cursor, which is moved using the cursor keys. Although many superior graphics input devices exist, such as light pens and mice, it was decided to implement the interface using facilities available on a "standard" PC.

Experience with the use of the editor by metallurgists has shown that these facilities are more than adequate.

Implementation of architectural form in the flowsheet editor

Although many of the latest flowsheeting packages (Woollacott 1982; Hess and Wiseman 1984; Preece 1986) have a menu-based graphical

interface where the operation to be selected is chosen from a menu, a command driven interface was implemented for the MicroSim flowsheet editor. This allowed the design consideration (discussed in the second part of section 3.3.1) that the flowsheet editor be totally user-driven to be implemented.

In order to perform any operation, the appropriate keystroke must be entered. The initial learning period is made easier for the user by the provision of on-line help screens, as well as by limiting the commands to one keystroke wherever possible.

A command driven approach was adopted for the following reasons :

- As no space has to be provided on the screen for the menus, the full area is available for the flowsheet drawing. This is important as the graphics resolution provided by the IBM PC type computers is not good.
- If a menu based interface is implemented, several menu choices may need to be made in order to perform a fairly simple operation. For example, if a unit is to be drawn the "draw unit" mode must first be selected, the unit to be drawn must be selected, and the unit may then be drawn at the desired position. The command driven approach requires the issuing of one command (all commands being a single keystroke) to draw a unit.
- Using the menu approach, it may not be possible to conveniently provide all possible combinations of desired actions. For example, a user may be drawing a stream when it is realised that the stream needs to end at a unit which has not yet been drawn. If the menu approach is used, it may be necessary to first exit the "draw stream" mode and then to enter the "draw unit" mode before the unit can be drawn. In the command driven approach, only a single keystroke is needed to draw the required unit as all commands and combinations of commands are accessible. This implements one of the suggested principles for good

interface design mentioned in Chapter 2, i.e. the computer should not divert the user from performing the task for which the system is designed.

Command driven systems may be criticised in that they are inherently more difficult to learn initially, as the user is required to memorise a number of commands before any use can be made of the system. This problem was minimised in the MicroSim flowsheet editor by providing on-line help and keeping the commands short.

3.4 Data Input Module

The design and implementation of the data input module will be discussed on two levels; architectural presentation and architectural form.

3.4.1 Architectural form for the data input module

The data input module must obtain information from the flowsheet editor and the tearing and ordering phase and then use this data to prompt the user for the remaining information defining the simulation.

Emphasis will be placed on the case where a new flowsheet is being defined; for an old flowsheet the required information is simply retrieved from disk storage.

There are three types of information which a user must enter to define a simulation. These are:

System Data : This data defines properties common to all streams in the flowsheet. This consists of information such as the number of size classes to be used, the number of minerals in the ore, the specific gravity of each mineral and so on.

Streams Data : This data defines the properties of the flowsheet feed streams. For example, solids and water mass flows, size distributions and so on.

Unit Data : This data defines the operating characteristics as well as the mathematical model used to describe each unit process in the flowsheet.

A sequential interface form for the data input module was adopted due to the well defined structure of the data input phase. That is, the user is first prompted for system data, then streams data and finally unit data.

This is in contrast to the user-driven approach adopted for the flowsheet editor. However, the process of drawing a flowsheet is non-sequential. Data input does not necessarily have to be sequential but it was felt that the sequential approach would guide users into entering the correct data at the correct time.

Experience with a totally user-driven material balancing package, Genflow (Woollacott 1982), illustrated the difficulty of having a user-driven data input phase. Using such an approach, the user is often in doubt as to which item of data must be selected and entered next when still inexperienced in the use of the program.

Thus the natural sequential structure of the data input phase for a simulation problem defined the architectural form of the data input module.

One of the most important considerations in the design of the data input module, was that there be no defaults assigned, especially to unit model parameters, of which the user is not made aware. These defaults should be as visible as possible, as the consequences which may arise if invalid defaults are used during simulation may be disastrous. For the user to take any responsibility for the simulator predictions the user must be made fully aware of all the parameters which influence the predictions made by the simulator. An ideal way of achieving this is by the use of form filling; the defaults for all parameters being provided in the form. The user may then substitute values appropriate to the particular problem in place of the defaults. Form filling is described in more detail in the following section.

3.4.2 Architectural presentation for the data input module

For a simulation program, data input consists mostly of entering numeric information. Another action which the user will carry out often is the

selection of the appropriate item or choice from a number of choices. The design of these two most commonly used presentation methods, numeric data input and menu selection, will now be discussed.

Form filling

Maximum use is made of form filling in the data input section. This gives the interface a consistent appearance. All forms have appropriate defaults in each field. This enables users to carry out a simulation without having to go into every detail when still learning to use or evaluating the simulator. However, changing default items to problem specific values is easy.

Error trapping is performed while the user is filling in the form. In the case where the numbers entered by a user must meet some constraint, for example a size distribution must sum to 100%, checking is performed to see if the constraint is met. If an error is detected, the appropriate forms are displayed, with the numbers previously entered by the user displayed in the default position. Error correction is thus made simple.

Keystroke filtering is also used so that it is only possible to enter numeric data where numeric data is required.

Where possible, well defined and widely used terminology is used in the forms. Figure 3.4 illustrates the way in which a user would be asked to enter a size distribution in the data input or editing section of MicroSim.

Menus

Menus are used to select an appropriate choice when there are several. MicroSim menus were implemented using a pointing mechanism. The user moves a pointer until it points to the appropriate choice, which is then activated by striking the return key. All choices are described using text not graphics icons.

This is an attempt to use the "pointing to an icon" technique discussed in 2.2.2, using "standard" hardware. However, instead of pointing to a

graphical icon, the user points at a textual item which describes the action he wishes to carry out. The pointing technique is considered to be more intuitive than typing in a number or letter to select the appropriate choice.

Enter particle size distribution (in %) in stream 1
Push <ESC> to exit or for next screen

+	2360.0 Microns		2.4
-	2360.0 + 1700.0		3.1
-	1700.0 + 1180.0		4
-	1180.0 + 850.0		5
-	850.0 + 600.0		6.6
-	600.0 + 425.0		9.1
-	425.0 + 300.0		13.1
-	300.0 + 212.0		16.4
-	212.0 + 150.0		12.7
-	150.0 + 106.0		7.6
-	106.0 + 75.0		4.9
-	75.0 + 53.0		3.7
-	53.0 + 38.0		2.8
-	38.0 + 27.0		1.8
-	27.0 + 0	6.8	

Figure 3.4 : Size distribution input form

3.5 Data Output

The primary requirement for the data output section is that it must be able to present the large amounts of information output by the calculation phase concisely and in an understandable and directly usable form. The MicroSim data output module presents information in three ways:

- Mass balance and size distribution information may be tabulated and output to the screen, disk file or printer. The user has some control over the output format, for example whether water flowrates should be displayed or not, and the units which grades and flowrates must be reported in (eg. tons per hour, tons per month, mass percent, parts per million and so on).

- Graphical representations of size distributions and partition curves may be plotted on the screen and then output to a plotter. Four commonly used forms of axes (linear-linear, log-linear, log-log and Rosin-Rammler) are available. Labels may also be placed on the graphs before they are plotted.

This allows information to be produced in a directly usable form without the need for any intermediate steps, such as the use of a dedicated plotting program.

- Certain units, such as screens and crushers, provide design information such as power rating etc. This concept was adopted from the MODSIM simulator (King 1985). The design report produced during simulation may be previewed on the screen before being printed.

Figures 3.5, 3.6 and 3.7 show examples of each of the three output formats for a simple simulation. The flowsheet simulated consisted of a single hydrocyclone with stream 1 the cyclone feed, stream 2 the overflow and stream 3 the underflow.

Figure 3.5 : Tabulated output

Strm No	Mass Flowrate Tonnes/Hour	Water Flow m^3/Hour	Mass Percent Solids
1	5.00	20.00	20.00
2	0.23	1.95	10.40
3	4.77	18.05	20.92

Size Distribution for stream 1

SIZE (Microns)	% Retained	% Passing
162.000	0.80	99.20
121.000	6.30	93.70
89.900	15.10	84.90
66.900	24.70	75.30
49.800	32.30	67.70
37.100	37.60	62.40
27.600	44.60	55.40
20.500	51.80	48.20
15.300	60.00	40.00
11.400	67.80	32.20
8.500	75.70	24.30
6.300	82.60	17.40
4.100	91.40	8.60
3.000	95.40	4.60

Size Distribution for stream 2

SIZE (Microns)	% Retained	% Passing
162.000	0.56	99.44
121.000	4.74	95.26
89.900	11.78	88.22
66.900	19.83	80.17
49.800	26.50	73.50
37.100	31.36	68.64
27.600	38.05	61.95
20.500	45.20	54.80
15.300	53.65	46.35
11.400	61.98	38.02
8.500	70.71	29.29
6.300	78.58	21.42
4.100	89.00	11.00
3.000	93.92	6.08

Size Distribution for stream 3

SIZE (Microns)	% Retained	% Passing
162.000	0.81	99.19
121.000	6.37	93.63
89.900	15.26	84.74
66.900	24.93	75.07
49.800	32.58	67.42
37.100	37.90	62.10
27.600	44.91	55.09
20.500	52.11	47.89
15.300	60.30	39.70
11.400	68.08	31.92
8.500	75.94	24.06
6.300	82.79	17.21
4.100	91.51	8.49
3.000	96.36	3.64

Figure 3.6 : Report file output

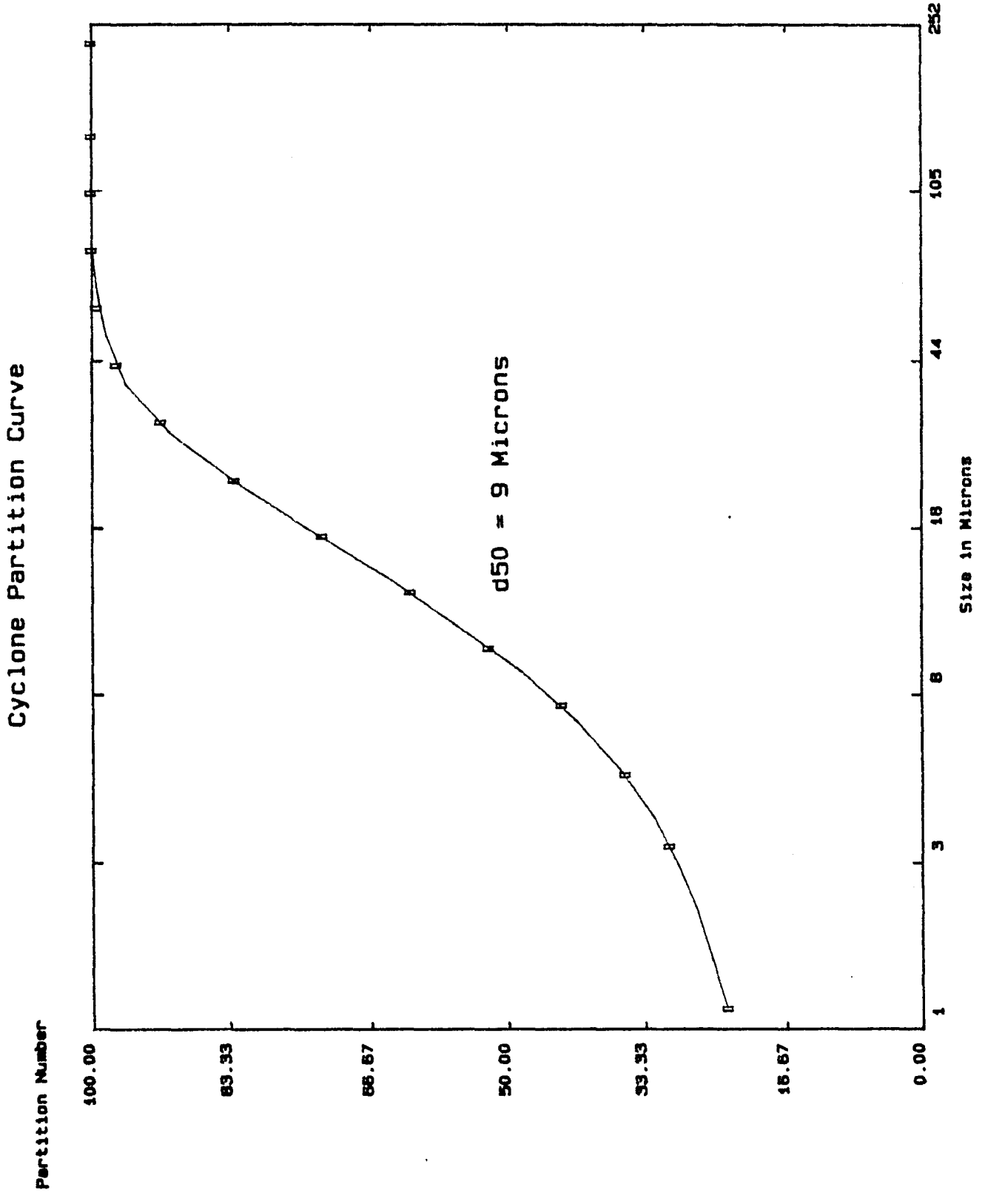
```
------------------------------------------------
          MICROSIM Report For Unit 1
          Unit Of Type : HYDROCYCLONE
             Model Used   : Plitt
------------------------------------------------

Cyclone parameters are
 Cyclone Diameter                          :  10.00 cm
 Vortex-Spigot Height                      : 130.70 cm
 Inlet Equivalent Diameter                 :   4.00 cm
 Vortex Finder Diameter                    :   4.10 cm
 Spigot Diameter                           :   3.00 cm
 Presure Drop                              :  50.00 kPa
 Number Of Cyclones                        :   1
 Exponent For Solids Density               :   0.500
 Viscosity Of Carrier Fluid                :   1.000 centipoise
 D50 Correction Factor F1                  :   1.373
 Sharpness Correction Factor F2            :   1.380
 Pressure Correction Factor F3             :   1.000
 Split Correction Factor F4                :   0.148

Cyclone performance
 Volumetric Flow Split (S)                 :    1.441
 Mass Recovery Of Solids To U/F            :   81.16 percent
 Mass Recovery Of Water To U/F             :   56.97 percent
 Partition Curve Slope                     :    0.543
 Corrected Cut Size, D50c                  :   16.9 microns

Percentage Of Feed Partitioned to U/F
 Size Class              Partition number
    1                         97.5
    2                         95.1
    3                         93.3
    4                         91.2
    5                         88.8
    6                         86.4
    7                         83.8
    8                         81.3
    9                         78.9
   10                         76.5
   11                         74.3
   12                         72.3
   13                         70.0
   14                         68.0
   15                         64.3

------------------------------------------------
NOTES :


------------------------------------------------
```

3.6 Data Editing Functions

In order to make the simulator as flexible as possible in the "what-if" situation, a data editing module which allows access to most data defining the simulation was implemented.

Data editing is made easy by the use of form filling. Thus the data editing section appears very similar to the data input section making the overall interface very consistent.

Figure 3.7 : Graphical Output



High contextual dependence was also enforced, allowing the user to edit only the item of data selected, alleviating the need to re-enter all the data in a section. For example, the user is able to edit a particular unit parameter for one unit. There is no need to re-specify all unit parameters for all units or for a single unit.

The item of data to be edited is selected by a hierarchical system of menus. Care has been taken not to nest the menus too deeply. In most cases the menus are nested to two levels; the maximum number of levels used being four. Form filling is used for entering the numeric data, the form defaults being the data which existed before editing takes place.

3.7 Operating System Access

As discussed earlier, the user should be able to perform all required tasks without leaving the simulator. To facilitate this a "DOS utilities" option was implemented. This gives the user access to basic file manipulation commands. These include operations such as file copying, renaming and deleting.

3.8 The Internal Interface

The three areas which influence the user-friendliness of the internal interface have been stated previously viz. the availability of technical documentation, the use of structured programming techniques, and commenting of the source code. The first two of these aspects will be discussed briefly; the third can only be demonstrated by examination of the source code.

3.8.1 Technical documentation

A technical manual was written for MicroSim (Stange, Cilliers and King 1988b). This document describes the following in detail:

- The structure of the MicroSim system and the more important data structures used by the system.
- The addition of new unit icons and new unit models to the system.
- Changing drivers for different graphics cards.

- Adding new stream data structures to MicroSim. This allows the user to add new substream types to simulate processes not available in the simulator at present. For example, a gas stream type may be added if pyrometallurgical processes are to be simulated.

Where possible, these concepts are all illustrated using simple examples from the existing source code.

3.8.2 Structured and other programming techniques

MicroSim has been designed to consist of as many independent modules as possible. These modules may be separate programs, such as the flowsheet editor and the tearing and ordering routines, or may be well isolated routines in the main program.

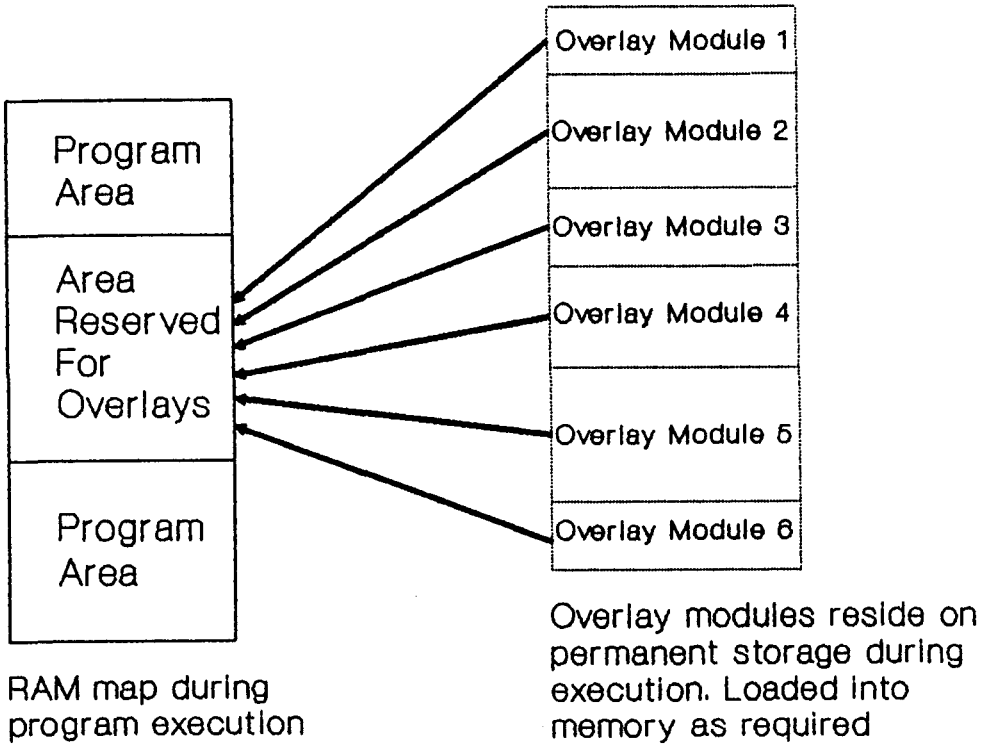
This facilitates modification of the system by users as it is not necessary for users to be familiar with the entire program before modifying the program. The changes which need to be made for the insertion of new models etc. are localised in modules and described in the Technical Reference Manual (Stange, Cilliers and King 1988b).

The concept of overlaying (illustrated in Figure 3.8) has been used extensively in the coding of MicroSim. An overlay is a section of code which resides on disk and is only loaded into memory when needed. This is facilitated by the modularity of the simulator.

This causes some degradation in performance, as the overlays need to be loaded from disk. This degradation has been minimised by careful design in that the execution time for the overlay code is typically much longer than the time required to load the overlay from disk. This degradation is hardly noticeable when the program is run from a hard disk. The advantages of using overlays more than outweighs the disadvantages.

The primary advantage is that of memory utilization. As only the module which is required at that moment needs be in memory, the total executable code size may be much larger than the available memory space. Although

Figure 3.8 : Overlaying



the total executable code size is about 300 Kb, MicroSim may be run on a machine having only 192 Kb of memory, 64 Kb being used for program space.

This means that up to 448 Kb may be used for data space (assuming a maximum memory capacity of 640 Kb). This allows the storage of about 74 000 real numbers (assuming 6 bytes per real number) which describe the streams properties. Assuming 10 size classes, 10 particle type classes, and 10 flotability classes, there is memory space for approximately 70 streams. This is large enough for the simulation of realistic, complex flowsheets.

Overlaying has been most effective in the unit model library. Each unit model subroutine is an overlay procedure. This is possible because in the sequential-modular calculation method only the subroutine needed for the calculation of the current unit model needs to be in memory. The consequences are that each unit model subroutine may be fairly large (in terms of lines of code) allowing the insertion of complex models, and that numerous unit models may be inserted[5].

The number of unit models which can be inserted is limited only by the available disk space. If a hard disk machine is used, disk space of 10 megabytes or more would be available, implying that there is no practical limit on the number of unit models which may be inserted into the simulator. This provides the potential for a totally integrated simulator i.e. models for ore-dressing, coal-washing, pyro- and hydrometallurgical unit processes are available for the simulation of a single flowsheet if required.

Another interesting technique used is the design of a database which stores all the prompts and numeric defaults for the unit model parameters. When a new model is added, no code needs to be written to prompt the user for unit model parameters. The person making the modifications simply executes a database program supplied which allows the model database to be manipulated when models are added or changed. This saves code space in the main program, and makes the addition of new models as simple as possible.

---

5 All ore-dressing models available in MODSIM (King 1985) were inserted into MicroSim. Some models were inserted by Cilliers (1987), the rest were inserted during this study by the author.

3.9 The Effectiveness Of The MicroSim Interface

The effectiveness of the MicroSim interface will be discussed on two levels; the external and internal interfaces.

3.9.1 The external interface

An opportunity to evaluate the effectiveness of the external interface arose when MicroSim was used during a Computer Aided Engineering course (King and Woollacott 1986) organised by the Department of Metallurgy and Materials Engineering, University of the Witwatersrand. This course was attended by metallurgists, mostly from industry, with levels of computer literacy varying from beginner to expert. As no quantitative study allowing the efficiency of the user-interface to be measured could be performed, the attitude of these users towards MicroSim can only be described in qualitative terms.

Attention was focused mostly on the flowsheet editor, as this is the most complex part of the user-interface. In general, users coped well with the flowsheet editor. After one to two hours practice, users were able to construct flowsheets of varying complexity.

In a questionnaire (Woollacott 1986) in which users were asked to comment on the differences in design philosophy between the Genflow (Woollacott 1982) and MicroSim flowsheet editors, it was found that when selecting units the majority of users preferred the MicroSim command driven approach to the Genflow menu driven approach. This justifies the decision to implement a command driven editor rather than a menu driven system.

Form filling proved very effective, and all users were able to complete a simulation exercise without any major problems.

At the time of writing (July 1988) MicroSim has been generally available for approximately 2 years. During this time, MicroSim has been supplied to 19 installations in South Africa, Australia, Europe and the United Kingdom. The majority of these (15 out of the 19) are industrial sites.

No major interface related problems have been reported from any of the installations.

3.9.2 The internal interface

Due to the youth of the MicroSim system, no reports of users outside the University environment who have attempted any additions to the simulator have been received. This makes the evaluation of the internal interface difficult. However it is worthwhile describing the experiences encountered by people using and extending MicroSim in the University environment.

The availability of detailed technical documentation has made the insertion of subroutines for new unit icons almost trivial, even for people not fluent in Pascal. This indicates that no problems should be encountered in this area by any users of the system having a reasonable knowledge of Pascal.

On a more technical level, streams data structures have been designed and implemented in order to simulate a gas phase. This was done so that the simulator could be used for the development of models used in air pollution control, such as cyclones and electrostatic precipitators. No details regarding the progress of this work are available.

Work involving the addition of optimization capabilities to MicroSim has been carried out. This is an extremely important area, which once fully developed, will allow the user to solve complex and realistic optimization and design problems.

It must be stated that all work done concerning the addition of an optimization routine was not an exercise in the development of optimization techniques. This exercise was performed to see how easily the MicroSim program structure could be extended to include complex new modules which were not envisaged during the design of the program, a situation which may arise easily and frequently.

A search algorithm (Caceci and Cacheris 1984) was easily incorporated into the existing program. No changes were required to existing program

modules. The modularity of the program proved to be very effective; a single subroutine call inside the search routine being all that was required in order to initiate a complete simulation calculation.

This simple approach proved remarkably effective. The optimization loop allows parameters for complex unit models (such as milling and flotation) to be estimated from plant data, as well as allowing the solution of design and optimization problems. The optimization abilities of MicroSim are discussed in more detail in Appendix 2, which provides some examples of the capabilities of this component of MicroSim. Experience thus far has shown that the optimization capabilities of MicroSim significantly increase the usefulness of the program.

From this exercise it can be concluded that it is feasible (even on a PC) to integrate optimization techniques into a sequential-modular simulator, although much research still needs to be done in the search for more efficient multi-variable, constrained optimization algorithms. Highly modular simulators, such as MicroSim, facilitate the inclusion of new features such as these.

Regarding software maintenance, this has been a much smaller problem than anticipated with very few serious bugs reported by users. This can be attributed to:

- The use of a powerful, structured and strictly typed language such as Pascal. This facilitates the production of robust code. The debugging stage of program development is very effective, with most bugs being eliminated during this phase.
- Implementation is on standard, widespread hardware leading to few installation problems.

3.10 Conclusions

A powerful, easy-to-use interface for MicroSim has been developed by careful consideration of the requirements of the broad spectrum of potential users of a general minerals processing simulation program. Particular attention was paid to the areas of interactive graphics, data entry and editing and data

output. The simulator was implemented so that it can function as an integrated package without dependence on operating system software or other packages, such as plotting programs.

Experience has shown that even the most inexperienced users (in terms of both computer experience and knowledge of simulation) are able to use MicroSim for simulation studies. There is also good evidence that the modular structure of the program will facilitate the inclusion of any future requirements.

CHAPTER 4

THE DEVELOPMENT OF A MODEL FOR THE CARBON-IN-PULP PROCESS

It was shown in Chapter 1 that little work concerning the development of hydrometallurgical models for use in a general process simulator has been published. In addition, few simulators capable of simulating hydrometallurgical processes are available. The remainder of this dissertation discusses the derivation of a model for a complex and economically important hydrometallurgical process; namely the carbon-in-pulp (CIP) process.

As will be seen later in this chapter, the CIP process has become widely used in the gold mining industry. Any techniques which result in more efficient design and operation of such plants could result in significant economic benefits in most gold producing countries.

A good model for the CIP process can be considered as being a valuable part of the techniques which can be used to improve understanding, design and operation of such plants. Industry is only now beginning to use modelling routinely in the design and operation of CIP plants. Some models in use by industry (Bailey 1988) are almost dangerously simplistic and relatively inflexible. This situation has arisen due to the complexity of the CIP adsorption process; realistic models for this process being necessarily complex. Industry has thus preferred models which are less complex, with a corresponding loss in model accuracy.

This situation could be improved if a user-friendly simulator such as MicroSim was capable of simulating the CIP process with sufficient flexibility. The user-friendly nature of the simulator would hide the complex nature of the CIP model. The user of the simulator would thus not have to be an expert in CIP modelling in order to utilise the CIP model effectively.

It was thus decided that the development of a CIP model[1] for the MicroSim simulator would result in the following benefits:

- The availability of such a model would facilitate better design and operation of such plants. This could have significant economic benefits for the mining industry.
- As the CIP process is complex, the development of a model for MicroSim for this process could provide guidelines for the development of further hydrometallurgical models for simulators such as MicroSim.

This chapter will provide some background on the CIP process and will review some existing models for the process. As the overall aim of the project is to facilitate the use of simulation in an industrial environment, the requirements for a CIP model for MicroSim are that the model describe the industrial-scale process realistically. It was found that no existing models meet this criterion (discussed in section 4.4).

It was thus necessary to develop a suitable model for the CIP process. It will be shown that the population balance technique (Hulbert and Katz 1964) can be used as the basis for the development of a CIP adsorption model. Details of the solution of the resulting equations and some simulation results will then be presented.

Chapter 5 will then describe how the modelling techniques developed in this chapter can be simplified to make the solution of the model on a microcomputer practical. Chapter 5 also describes the integration of the resulting model into the MicroSim simulator.

4.1 History And Description Of The CIP Process

The history of the CIP process has been described by a number of authors (Fleming 1983; Laxen 1986, Bailey 1988). The following summary has been taken from these works.

---

1 Note that in this dissertation the term "CIP model" refers to a model for the adsorption section of the process; not to all unit processes in a CIP flowsheet.

The readiness with which activated carbon adsorbs gold in cyanide solution was discovered by Johnson in 1894. By 1916, the Yuanmi mine in Western Australia was using fine carbon to recover gold from cyanide solutions by pumping the gold bearing solution through carbon filters. The gold was recovered from the carbon by ashing the carbon and smelting the ash.

The absence of an efficient elution process (one which did not destroy the carbon) which could be used to remove the adsorbed gold from the carbon and the efficiency of the competing zinc precipitation procedure resulted in the use of the zinc precipitation procedure for many years.

After World War II, the United States Bureau of Mines (USBM) played a leading role in the development of the technology required for the implementation of an industrial scale CIP flowsheet. The development of the Zadra elution procedure by the USBM, which allowed the loaded carbon to be stripped of gold (by means of a hot caustic cyanide solution) and reused for adsorption, led to the establishment of a pilot CIP plant at the Getchell mine. At this plant, carbon was used for 10 to 15 adsorption cycles (without thermal regeneration at the end of each elution) before being replaced.

By 1961, the basic CIP flowsheet in use today, had been implemented at the Carlton Mill at Cripple Creek, Colorado. This flowsheet is shown in Figure 4.1.

In this flowsheet, a conventional cyanide leach circuit preceded the CIP circuit. In the CIP circuit, carbon was moved counter-current to the cyanided pulp. This counter-current carbon movement resulted in high adsorption efficiencies. The carbon in each stage was separated from the pulp using external vibrating screens in order to move the carbon counter-current to the pulp. The loaded carbon was eluted using the Zadra procedure and then thermally regenerated by heating the eluted carbon to a temperature of 650 °C in a steam atmosphere before being returned to the adsorption circuit. Electrowinning was used to recover the gold from the eluate.

Although the modern CIP flowsheet is very similar in principle to that described above, considerable development has taken place, leading to improved efficiency for most of the equipment in a typical CIP flowsheet. This includes the development of more efficient inter-stage screens, elution,

Figure 4.1 : Typical CIP Flowsheet

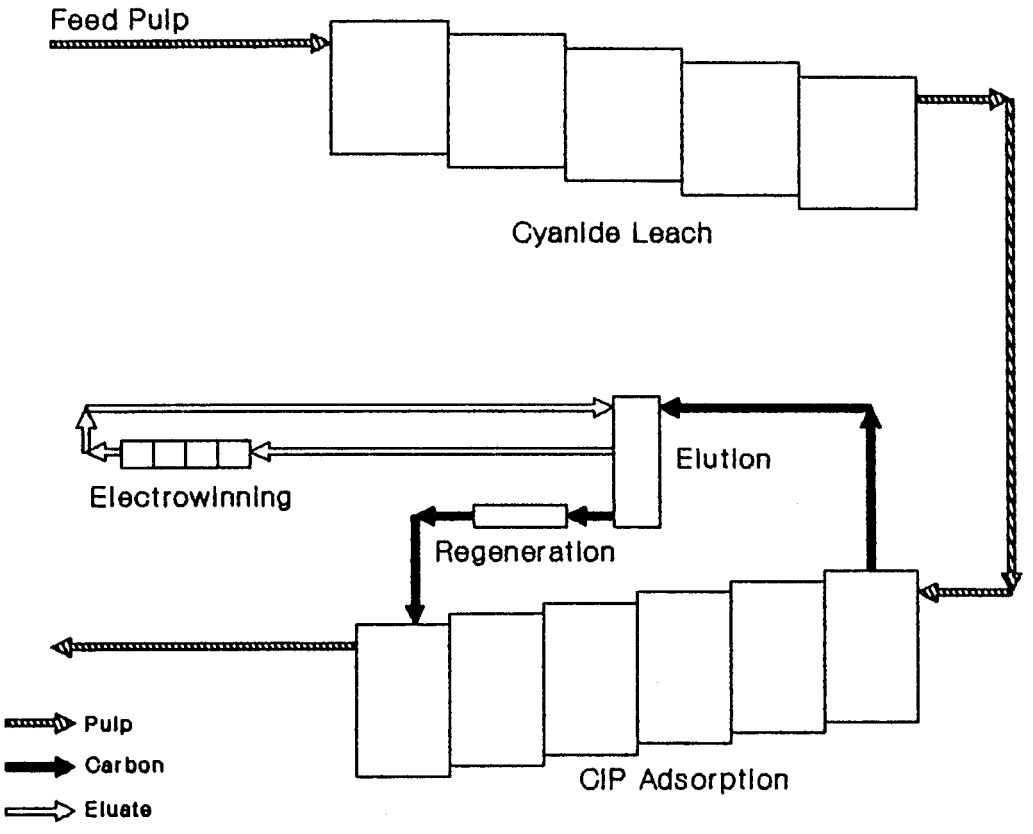
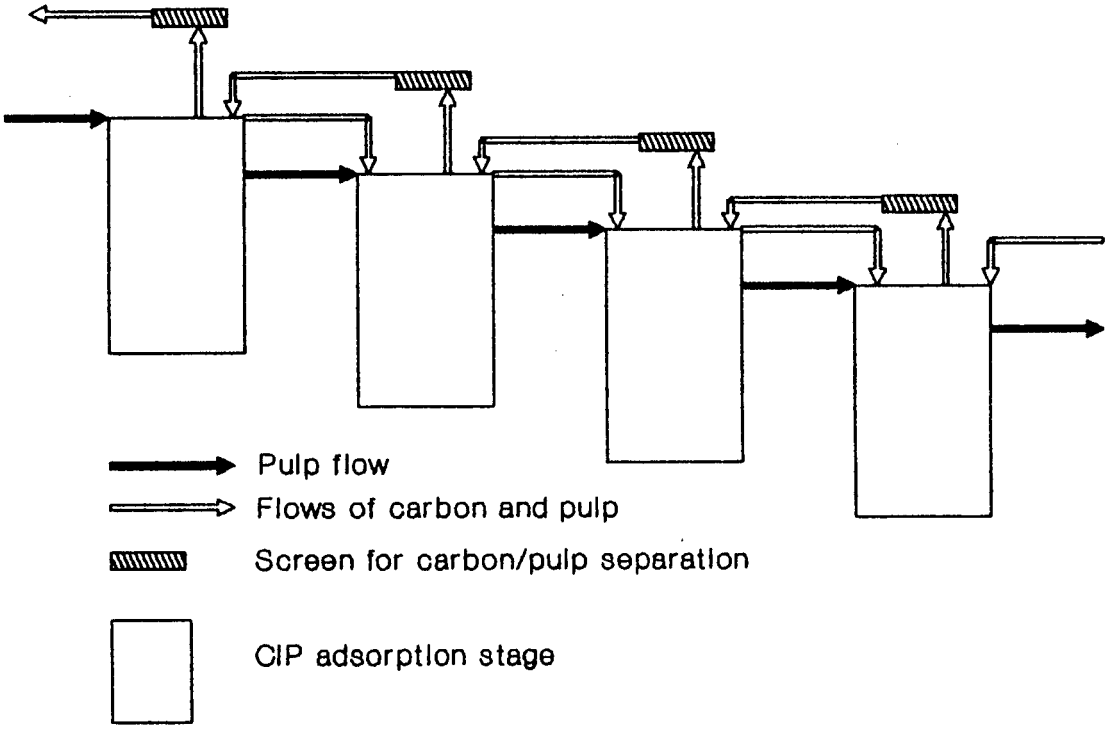


Figure 4.2 : The CIP Adsorption Process



regeneration and electrowinning procedures. Acid washing of the carbon, in order to remove precipitates such as calcium carbonate, has also been implemented. A major role in these developments have been played by South African research groups, namely the Council for Mineral Technology (MINTEK) and the Anglo American Research Laboratories (AARL).

CIP technology is firmly entrenched in all gold producing countries. In South Africa, CIP has become the preferred route for the recovery of gold from leached pulps (Bailey 1986a). At present, 11 plants exist which handle more than 100 000 tons of ore per month. Many smaller scale operations also exist, as well as larger plants such as ERGO, which treats up to 1.9 million tons of ore per month. At present, 4.5 million tons of gold ore per month are treated by the CIP route in South Africa, while plants which will treat an additional 1.7 million tons per month are planned.

Bailey (1986b) is of the opinion that the successful operation of a CIP plant depends largely on the performance of the adsorption section. It is for this reason that the model developed in this dissertation describes the adsorption circuit.

4.2 Physical Characteristics Of The CIP Adsorption Process

Before a model is derived for a process, it is necessary to have a good understanding of the physical characteristics of the process. In this project, an attempt to derive a model which is useful in the industrial environment is attempted. Thus it is not sufficient to model the CIP process as it would work in an ideal world. Instead the model should be capable of simulating the process as it behaves in reality.

The CIP adsorption process is illustrated in Figure 4.2. The equipment consists of a number of CSTRs in series. Pulp gravitates through this series of reactors. Each contactor is fitted with a screen which allows a mass of carbon to be retained in the reactor while the pulp flows through the tank. As the pulp passes through the train of contactors, gold and other species are adsorbed from the pulp onto the activated carbon.

Counter-current movement of the carbon is normally achieved by pumping pulp and the carbon it contains forward to the adjacent upstream stage. As the gold loadings on the carbon particles depend on the position of the carbon in the circuit, a distribution of loadings arises in each stage due to the countercurrent transfer of carbon and the resulting mixing of carbon having different loadings.

The length of time needed to transfer the carbon depends on various plant parameters such as the concentration of carbon in the pulp (the lower the concentration, the greater the amount of pulp that needs to be transferred in order to move a fixed amount of carbon for a given pumping rate). Plant practice varies widely so that carbon transfer may be active for 10 to 100 percent of the time depending on the constraints (pump capacities and so on) faced by the operating staff.

Due to the nature of the carbon, a particle size distribution is also present in each stage. As the carbon loading and particle size both have a significant effect on the rate of adsorption, a CIP model must take into account the effect of these factors on the overall performance of the system.

As the screens in the adsorption stages can never be 100 percent efficient, plants experience some degree of co-current carbon flow as the carbon leaks past the screens with the pulp. This screen leakage may be due to:

- Overflows which result in the pulp (and the carbon it contains) bypassing the screen units.
- Undersize carbon less than the aperture size of the screen passing through the screen.
- Platelet shaped carbon which passes through wedge-wire type screens.

Co-current flow of carbon may result in a serious loss of efficiency for the adsorption section (Whyte et al 1987). Carbon abrasion in the adsorption section is common in most South African CIP plants (Bailey 1988). This results in carbon fines which flow easily through inter-stage screens. However, no

model for carbon abrasion/breakage has been published. The development of such a model was considered to be beyond the scope of this work. Therefore the carbon abrasion aspect was ignored in this work.

Models which have been used to simulate the CIP adsorption process will now be discussed.

4.3 Existing Models For The CIP Adsorption Process

It should be obvious that the adsorption system is in general fully dynamic in nature due to the non-continuous (in general) movement of the carbon phase. Thus, any model for the system will consist of a number of differential equations which must be integrated over a period of time in order to simulate the behaviour of the process. The difficulty which arises when a dynamic model is used in a steady-state simulator is explored in more detail in the next chapter.

Much work on the modelling of the adsorption section of the CIP process has been reported recently. Efforts have been directed primarily towards the derivation of appropriate equations which describe the rate of loading of the aurocyanide ion onto activated carbon under various conditions.

This work ranges from complicated expressions derived from consideration of the postulated physical behaviour of the system (Van Deventer 1984) to more empirical expressions (Menne 1982a). In general, the methods used by each author to insert the adsorption rate expression into the material balance which describes the CIP situation has varied depending on the complexity of the rate expression used. The more important models will now be discussed, in order of increasing complexity.

The rate expression developed by Nicol, Fleming and Cromberge (1984) is based on the assumption that the rate of transport of gold into the carbon particle is characterized by a single pseudo mass-transfer coefficient. Transport in the solution across the boundary film is also assumed to be represented by a single pseudo mass-transfer coefficient. A linear isotherm is used to describe equilibrium at the particle surface.

These assumptions result in a rate expression having the following form:

$$R(y,c) = k\left(C - \frac{y}{A}\right)$$

Where:

y = Loading of gold on activated carbon particle (mass of gold per unit mass of carbon).
C = Concentration of gold in bulk solution (mass of gold per unit mass of solution).
R(y,c) = Rate of loading of gold onto a carbon particle having a loading y from a solution having a tenor C (mass of gold per unit mass of carbon per unit time).
k = Mass transfer coefficient dependent on mixing conditions and carbon particle size.
A = Parameter for linear isotherm. That is $y_e = AC_e$ where C_e is the solution concentration in equilibrium with carbon having a loading y_e.

Note that the expression is linear in the gold loading term, y. This allows the use of average loadings instead of having to take into account the effects of the distribution of gold loadings in each stage (Williams and Glasser 1985). This will also be demonstrated in the next chapter.

Nicol, Fleming and Cromberge (1984) demonstrated that this model can be used for the batch and continuous situations. They showed that acceptable model fits can be obtained for both the batch and continuous situations.

Williams and Glasser (1985) use an empirical rate expression suggested by Dixon, Cho and Pitt (1978) to develop a simulation model for the CIP process. The rate expression has the form:

$$R(y,c) = k_1 C(y^* - y) - k_2 y$$

Where:

y^* = Parameter with same units as y.
k_1 k_2 = Forward and reverse rate constants with units compatible with the above definitions.

Menne (1982a) fitted a wide range of data to this expression and found good fits for both the batch and continuous situation (Williams and Glasser 1985). Note that this rate expression reduces to a Langmuir type isotherm at equilibrium (when R(y,C)=0). Van Deventer (1984) has shown that the Langmuir isotherm is a good model for the equilibrium of the aurocyanide-carbon system.

Williams and Glasser (1985) also describe assumptions which allow the liquid phase mass balance to be written as algebraic instead of differential equations. This reduces the amount of numeric calculation needed for the dynamic description for the carbon and solution phases to a minimum, allowing this model to be used on a microcomputer. No attempt at incorporating the effects of a carbon particle size distribution is reported by Williams and Glasser.

Johns (1987) showed that under certain situations, typical of plant conditions, the rate of adsorption of gold onto activated carbon is controlled by the rate of diffusion of the aurocyanide ion through the film boundary surrounding the carbon particle. This mechanism was found to control the rate of adsorption until the carbon had reached approximately 70 per cent of its equilibrium loading capacity. He also shows that equilibrium for the carbon-aurocyanide system can be described using the Freundlich isotherm.

The rate expression developed by Johns thus has the following classical film diffusion form:

$$R(y,C) = K(C - a\,y^b)$$

Where:

K = Mass transfer coefficient.
a,b = Parameters in the Freundlich isotherm $C_e = a y_e^b$ which relates the solution concentration, C_e in equilibrium with carbon having a loading y_e.

The effect of carbon particle size is incorporated in the mass transfer parameter K. This predicts that the rate of adsorption R will be inversely proportional to particle size d, a fact which has been verified by a number of investigators (Fleming and Nicol 1984).

Johns (1987) obtains acceptable fits for rolling bottle batch tests using this expression.

Because this rate expression is highly non-linear (in the loading term y), a strategy to simulate the effects of the loading distribution present in each stage of a CIP plant should be used in order to make the procedure mathematically correct. However, Johns (1986) maintains that average loading values can be used in order to simulate continuous CIP plants and does not describe any method to take account of loading distributions.

Carrier, Hououin and Courchesne (1987) describe a dynamic simulator for the CIP process. The rate expression used to describe adsorption is the same as used by Menne (1982a), Williams and Glasser (1985) and suggested by Dixon, Cho and Pitt (1978). The authors develop the differential equations describing the dynamics of the process, describe methods for parameter estimation and present some simulation results for a plant study.

Although no account of the effect of carbon particle size is included in their analysis, the model developed by Carrier, Hououin and Courchesne is unique in one important respect. The authors take into account the effects of inter-stage carbon transfer; all other studies published to date ignore these effects as they assume that carbon transfer takes place instantaneously. However, on an industrial scale carbon transfer may take up anything from 1 to 24 hours per day. It is clear that assuming instantaneous transfer is far from correct for industrial scale plants.

The analysis presented by the authors (Carrier, Hououin and Courchesne 1987) although correct for the rate expression used in their study is not correct in the general sense. This is because no account was taken of the distribution of gold loadings on the carbon. This issue will be discussed in more detail in the next chapter (section 5.2.1). Good fits for the dynamic behaviour of a CIP plant are obtained by the authors.

Cho, Dixon and Pitt (1979) describe a model in which it is assumed that the rate of adsorption is limited by pore diffusion with local equilibrium existing between carbon and solution. The Freundlich isotherm is assumed to represent this equilibrium. The partial differential equation resulting from these assumptions are solved using a finite difference scheme. Batch adsorption data is shown to fit these equations reasonably well. No analysis of the batch counter-current configuration is presented.

Van Deventer (1984) has proposed a dual-rate model for the adsorption of gold onto carbon. He assumed that a carbon particle consists of two areas in which adsorption takes place; a macropore region and a micropore region, the dimensions of the micropores being much smaller than the macropores.

In the derivation of the model, Van Deventer considers diffusion of the adsorbed species through a stagnant film boundary around the carbon particle as well as intraparticle diffusion in both the macropore and micropore regions. It is assumed that intraparticle diffusion takes place by the mechanism of surface diffusion only. It is also assumed that the actual adsorption reaction is much faster than the diffusion processes, resulting in local equilibrium between the gold species in the liquid phase and the gold species adsorbed on the surface of the carbon particle.

Van Deventer also presents mass balance equations for both the carbon phase and the aqueous phase for the batch and CSTR situations. This results in a large set of equations (at least 3 differential and 2 non-linear equations for each CIP contactor) which must be solved to simulate the behaviour of a CIP plant. The number of equations grows significantly if a particle size distribution and a loading distribution is used. At each step of the integration, it is necessary to solve a non-linear set of equations, making the solution procedure slow for all but very powerful computers.

Although Van Deventer obtains good predictions of continuous plant behaviour based on parameters measured in batch tests, the structure of the model is complex. This means that at least a minicomputer is required in order to solve the model in reasonable times. The model also contains 5 parameters which must be estimated. This makes it difficult to use this model in an industrial environment.

The suitability of these models for use in industrial applications in a general process simulator such as MicroSim will now be discussed.

4.4 Suitability Of Existing Models

Some of the work reviewed above (Cho et al 1979; Menne 1982a; Nicol et al 1984; Johns 1987) are not simulation models but rather rate expressions. That is, the authors have presented functional forms which quantify the rate of adsorption of gold onto activated carbon under different solution tenor-carbon loading conditions. No detailed attempt at using these rate expressions with appropriate mass balances in order to derive a model for the CIP adsorption system is presented in these works.

The other works (Van Deventer 1984; Williams and Glasser 1985; Carrier et al 1987) combine rate expressions with the equipment mass balance to derive models which can be used to simulate the CIP adsorption process.

Some criterion for the suitability of existing models and rate expressions had to be formulated.

Due to the nature of this work, it was decided that any model or rate expression which is able to simulate the physical situation typical of the industrial process (described in 4.2) would be suitable. In particular, the models should be capable of describing (if required):

- Carbon transfer in detail.
- The co-current flow of carbon due to screen leakage.
- The effects of a carbon particle size distribution.
- The effects of a carbon loading distribution.

The criterion for rate expressions are simply that they provide accurate fits and predictions for the measured rates of gold adsorption by activated carbon.

4.4.1 Suitability of existing rate expressions

From the discussion of existing rate expressions presented in section 4.3, it can be seen that even simple rate expression such as that presented by

Nicol, Fleming and Cromberge (1984) provide adequate descriptions of the kinetics of adsorption. However, Nicol et al assume a linear isotherm to describe equilibrium. Many authors have shown however that the equilibrium between the carbon-gold aurocyanide system is highly non-linear; even in the low solution concentration regime.

The rate expression due to Dixon, Cho and Pitt (1978) also produces good fits for kinetic adsorption data. In addition, the expression reduces to a Langmuir type isotherm which fits observed data well (Van Deventer 1984).

The rate expression due to Johns (1987) is also considered suitable as goods fits have been obtained for both kinetic and equilibrium data.

The expressions due to Van Deventer (1984) and Cho, Dixon and Pitt (1979) also provide good fits to data. In addition, these models were derived by considering in detail the physical system being modelled. The predictive abilities of these expressions are expected to be better than the other expressions described. However, due to the complexity of these expressions, they are not suitable for use on microcomputers with their relatively limited numeric processing abilities.

4.4.2 Suitability of existing CIP adsorption models

From the requirements for suitable models discussed in the introduction to this section and comments made concerning the characteristics of existing models (section 4.3), it can be concluded that none of the existing models meet all the requirements for an industrially useful model. In all cases this is because a careful analysis of the characteristics of the process as it behaves in practice was not performed.

4.4.3 Conclusions

It can be concluded that a number of rate expressions suitably describe the kinetic and equilibrium behaviour of the carbon-gold aurocyanide adsorption system. However, no simulation model meeting the criteria described in the

beginning of this section exists. It is thus necessary to develop a model framework which will meet these requirements. The next section describes the method used to achieve this.

4.5 The Population Balance Technique

The adsorption section of a CIP plant can be regarded as a particulate process, where the discrete particles (carbon) exhibit distributed properties (size and metal loading). Due to the nature of the process, the movement of these particles (due to carbon transfer and screen leakage) may be extremely complex.

The population balance technique (Hulbert and Katz 1964) was developed in order to model complex particulate systems such as the system just described.

A particular advantage of this technique is that it allows rate parameters to be estimated using simple experimental techniques. For example in the CIP case, rate parameters may be estimated using monosize carbon particles and batch experiments (where a distribution of loadings is not present). As the rate dependence on particle size and loading is known (via the rate expression), this data may now be used to simulate the performance of the batch counter-current situation (where a distribution of both loading and particle size is present) using the population balance technique (assuming no scale-up/down problems).

The following section reviews the population balance technique in more detail.

4.5.1 Introduction

In the early 1960s it was discovered that many particulate systems could not be modelled adequately using the normal mass, energy and momentum balance approach. Consequently, the population balance concept was developed (Hulbert and Katz 1964; Randolph and Larson 1971) for systems which consist of countable entities.

This concept has been successfully applied to the modelling of a number of diverse systems. These include crystallization (Hulbert and Katz 1964), grinding (Herbst and Fuerstenau 1981), liquid-liquid extraction (Valentas

and Amundsen 1966a; Valentas and Amundsen 1966b), polymerization reactors (Kiparissides and Ponnuswany 1981), solid phase reactions (Bhatia and Perlmutter 1979) and leaching (Sepulveda 1981). Herbst (1979) presents an excellent review of the derivation of the population balance equations as well as the use of the population balance for various mineral processing unit operations.

Central to the population balance concept is the existence of a particle phase space i.e. a coordinate system which identifies the location of each particle as well as the magnitude of the properties of interest (for the specific problem) for that particle. The phase coordinates are divided into a set of external coordinates, specifying the location of the particle, and internal coordinates which quantify the particle properties of interest.

If the ordinary differential equations (i.e. kinetic expressions for each internal and external property) which describe how these phase coordinates vary with time are defined, then the population balance can be used to find the number density of particles at all points in the phase space.

For example, a plug-flow mill at steady state may be modelled using two coordinates; an external coordinate denoting the position of the particle along the length of the mill, and the internal coordinate, particle size. Solving the relevant equations allows the size distribution to be calculated as a function of length along the mill.

4.5.2 Derivation of the population balance equations

The derivation and notation used in this section is due to Sepulveda (1981).

Consider an assembly of particles in J+3 dimensional phase space, J being the number of internal coordinates or properties which completely define the behaviour of each particle for the system being modelled.

Each particle is thus characterized by the set of appropriate internal properties $\varepsilon_1(t)$, $\varepsilon_2(t)$ ... $\varepsilon_J(t)$ and a set of external coordinates x(t), y(t) and z(t).

Thus at any time t, the state of a particle is defined by the phase space coordinates E(t), where:

$$E(t) = (\varepsilon_1(t),\varepsilon_2(t),\ldots,\varepsilon_J(t),x(t),y(t),z(t))$$

Thus in order to characterize the behaviour of a particulate system, it is necessary to describe the trajectory of different particles E(t) in the J+3 dimensional phase space.

For continuous events this can be done based on information concerning the vector of property and spatial velocities:

$$\frac{dE(t)}{dt} = (\frac{d\varepsilon_1}{dt},\frac{d\varepsilon_2}{dt},\ldots,\frac{d\varepsilon_J}{dt},\frac{dx}{dt},\frac{dy}{dt},\frac{dz}{dt})$$

$$= [\,v_1^i, v_2^i, \ldots, v_J^i, v_x^e, v_y^e, v_z^e\,]$$

Changes in the system may also take place due to discrete events. For example, when an ore particle in a grinding mill is broken, there is a "death" event as the particle breaks and there are also "birth" events due to the instantaneous production of at least two new particles. Thus define:

a) The birth rate function, $B(\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_J,x,y,z,t)$ such that:

   BdR = Rate (number of particles per unit time) at which particles enter the incremental region of phase space dR, at spatial position (x,y,z) at time t, due to discrete events.

b) The death rate function, $D(\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_J,x,y,z,t)$ such that:

   DdR = Rate (number of particles per unit time) at which particles leave the incremental region of phase space dR, at spatial position (x,y,z) at time t, due to discrete events.

with:

$$dR = d\varepsilon_1 d\varepsilon_2 \ldots . d\varepsilon_J dx dy dz$$

Define the population density function $\Psi(\varepsilon_1,\varepsilon_2,.....,\varepsilon_J,x,y,z,t)$ so that ΨdR represents the number of particles in the region dR. Thus the total number of particles in any region R(t) is given by:

$$N_{R(t)} = \int_{R(t)} \Psi dR$$

The population balance equation may take one of two forms; namely the microscopic or macroscopic form.

The microscopic population balance equation

This form of the equation is simply a number balance at a specified position (x,y,z) in a reactor vessel at time t. This form is used when the particle properties are dependent on the spatial position of the particles, such as in a plug flow reactor. For this form, knowledge of the spatial dependencies of the property velocities $v_k^i (k = 1..J)$ and the functions B and D at any time t must be available.

For an arbitrary region R(t) apply the conservation equation:

Rate at which particles accumulate in R(t) =
Rate at which particles enter R(t) -
Rate at which particles leave R(t) +
Rate of generation of particles in R(t)

If it is assumed that the control region moves convectively across the phase space with velocity dE(t)/dt then the input and output terms are zero.

The accumulation term is :

$$\frac{dN_{R(t)}}{dt} = \frac{d}{dt} \int_{R(t)} \Psi dR$$

The generation is simply:

$$\int_{R(t)} (B - D) dR$$

Thus:

$$\frac{d}{dt} \int_{R(t)} \Psi dR = \int_{R(t)} (B - D) dR$$

Expanding this equation using Leibnitz's rule and recognizing that the integral over the region R(t) must vanish (as this region R(t) was selected arbitrarily) provides the final form of the microscopic equation:

$$\frac{\partial \Psi}{\partial t} + \nabla \cdot (\Psi \underline{v}^i) + \nabla \cdot (\Psi \underline{v}^e) - (B - D) = 0 \qquad \{4.1\}$$

where:

$\underline{v}^e = (v_x^e, v_y^e, v_z^e)$ = vector of external coordinate velocities.

$\underline{v}^i = (v_1^i, \ldots, v_j^i)$ = vector of internal property velocities.

In order to use the microscopic equation, a detailed description of the influence of spatial position on the property velocities and birth and death functions must be available. This may be very difficult for systems such as the CIP process, where the particles may be regarded as well mixed, i.e. the behaviour of the particle is not influenced by the position of the particle in the reactor. For this case, the macroscopic population balance equation can be used.

The macroscopic population balance equation

For this case, an averaging of the various quantities over the reactor vessel is utilized:

$$\overline{\Psi} = \frac{1}{V}\int_V \Psi dV$$

$$\overline{B} = \frac{1}{V}\int_V B\,dV$$

$$\overline{D} = \frac{1}{V}\int_V D\,dV$$

where:

V = Active volume of reactor vessel.

$\Psi d\varepsilon_1 \ldots d\varepsilon_J$ = number of particles per unit volume having internal property values in the range ε_1 to $\varepsilon_1 + d\varepsilon_1$, ε_2 to $\varepsilon_2 + d\varepsilon_2$,...., ε_J to $\varepsilon_J + d\varepsilon_J$

$\overline{B} d\varepsilon_1 \ldots d\varepsilon_J$ = number of particles created per unit volume by discrete events per unit time having internal property values in the range ε_1 to $\varepsilon_1 + d\varepsilon_1$, ε_2 to $\varepsilon_2 + d\varepsilon_2$,...., ε_J to $\varepsilon_J + d\varepsilon_J$

$\overline{D} d\varepsilon_1 \ldots d\varepsilon_J$ = number of particles destroyed per unit volume by discrete events per unit time having internal property values in the range ε_1 to $\varepsilon_1 + d\varepsilon_1$, ε_2 to $\varepsilon_2 + d\varepsilon_2$,...., ε_J to $\varepsilon_J + d\varepsilon_J$

Integrating {4.1} over the volume V:

$$\int_V \left[\frac{\partial \Psi}{\partial t} + \nabla \cdot (\Psi \underline{v}^i) + \nabla \cdot (\Psi \underline{v}^e) - (B - D) \right] dV = 0$$

{4.2}

The first term in {4.2} can be written:

$$\int_V \frac{\partial \Psi}{\partial t} dV = \frac{\partial (V\overline{\Psi})}{\partial t} - \overline{\Psi}\frac{dV}{dt}$$

{4.3}

As it has been assumed that the internal property velocities do not depend on the spatial position of the particle, the second term in {4.2} can be written:

$$\int_V \nabla \cdot (\underline{v}^i \Psi) dV = V \nabla \cdot (\underline{v}^i \overline{\Psi})$$

{4.4}

The third term in {4.2} is an integration over the volume of the spatial divergence of the population flux, and can be converted into a surface integral of the population flux flowing through the moving surfaces of the system using Gauss' divergence theorem:

$$\int_V \nabla \cdot (\underline{v}^e \Psi) dV = \int_S \hat{n} \cdot (\underline{v}^e \Psi) dS$$

{4.5}

Here S is the surface enclosing the volume V and $\hat{n}$ is the unit normal vector orthogonal to the surface. The RHS of {4.5} is an integral over the surface S of the flux of particles across the boundaries of the reaction vessel. This integral can be split into several components i.e. the areas of the inlets and outlets of the vessel and the remaining area enclosing the vessel. Thus {4.5} is written:

$$\int_V \nabla \cdot (\underline{v}^e \Psi) dV = -\sum_{p=1}^{P} Q_{\text{in}}^p \overline{\Psi}_{\text{in}}^p + \sum_{q=1}^{Q} Q_{out}^q \overline{\Psi}_{out}^q + \overline{\Psi} \frac{dV}{dt}$$

{4.6}

where:

Q_{in}^p = volume flowrate of suspension entering the vessel through the p'th inlet.

Q_{out}^q = volume flowrate of suspension leaving the vessel through the q'th inlet.

P = number of inlets in vessel.

Q = number of outlets in vessel.

Substituting {4.3}, {4.4} and {4.6} into {4.2} gives:

$$\frac{1}{V}\frac{\partial(V\overline{\Psi})}{\partial t}+\nabla\cdot(\underline{v}^{i}\overline{\Psi})-(\overline{B}-\overline{D})=\frac{1}{V}\left[\Sigma_{p}Q_{\text{in}}^{p}\overline{\Psi}_{\text{in}}^{p}-\Sigma_{q}Q_{out}^{q}\overline{\Psi}_{out}^{q}\right] \qquad \{4.7\}$$

Equation {4.7} is the standard form of the macroscopic population balance equation. The next section will demonstrate how this equation can be used as a basis for a model for the CIP adsorption system.

4.6 A CIP Model Based On The Macroscopic Equation

This section will demonstrate how the macroscopic population balance equation can be used to derive a model for the CIP adsorption process. It will be shown how this method provides a framework which considers the effects of the carbon particle size distribution and the gold loading distribution. It will be seen that the method can be used with an arbitrary rate expression.

The model presented here is not intended to meet all the criteria for an acceptable CIP model as discussed in sections 4.2 and 4.4, but is presented as an example in the use of the population balance for adsorption modelling. The next chapter will deal with the derivation of a more suitable model in detail.

4.6.1 Model assumptions

It is assumed that:

a) During an adsorption cycle no carbon enters or leaves a contactor.

b) The time taken to transfer the carbon is small compared to the length of the adsorption period, i.e. carbon transfer is instantaneous.

c) No carbon abrasion or breakage occurs.

d) The carbon phase is sufficiently well mixed to allow use of the macroscopic population balance equation.

As seen from sections 4.2 and 4.4 assumptions a), b) and c) are certainly not valid. The population balance technique can still be applied if assumptions a) and b) are not made. This will be demonstrated in the next chapter. These assumptions are made here so that the technique can be demonstrated without making the resulting equations too complex.

Due to the fact that no model for the breakage and abrasion of carbon has been published, there was no choice but to make assumption c). If and when such a model becomes available, the population balance framework could still be used.

Assumption d) is valid if the mixing in the adsorption section is sufficient; this is the case in most CIP plants, especially the newer ones. Thus assumption d) is reasonable.

4.6.2 The carbon mass balance

The macroscopic equation {4.7} can be used to describe the mass balance for the carbon under the assumptions described above. The distributed properties of interest are particle size and the gold loading on the carbon. Thus the property velocities are:

$$\nu_1 = \frac{dy}{dt} = R(y,\delta,C)$$

$$\nu_2 = \frac{d\delta}{dt} = 0$$

Where $R(y,\delta,C)$ is the rate of gold loading (mass of gold per mass of carbon per unit time) onto a carbon particle having a gold loading of y, a particle size of δ in a solution having a bulk gold tenor of C. Any suitable functional form (as described in section 4.3) can be chosen for the rate expression.

As it has been assumed that no carbon breakage occurs, the birth and death functions, $\overline{B}$ and $\overline{D}$, are both equal to zero.

Thus, carbon particle size is not a variable as the particle size does not change. If the effects of particle size are to be considered a suitable number of discrete carbon particle sizes δ_j are chosen. The problem is then solved for the specified number of carbon particle sizes.

It will also be assumed that the effective reactor volume V(t) is constant. This allows {4.7} to be written for each contactor x as:

$$\frac{\partial \overline{\Psi}_x}{\partial t} + \frac{\partial}{\partial y}\{R(y,\delta_j,C_x)\overline{\Psi}_x(y,\delta_j,t)\} = 0 \qquad \{4.8\}$$

As it has been assumed that no size degradation occurs, it is possible to substitute the more convenient mass fraction P_x for the number density Ψ_x. Thus $P_x(y,\delta_j,t)dy$ is the fraction of the total mass of carbon in contactor x having a loading between y and y+dy and representative particle size δ_j at time t. It is easily shown that {4.8} can be written as:

$$\frac{\partial P_x(y,\delta_j,t)}{\partial t} + \frac{\partial}{\partial y}\{R(y,\delta_j,C_x)P_x(y,\delta_j,t)\} = 0 \qquad \{4.9\}$$

Equation {4.2} is a system of first-order partial differential equations (PDEs) which describe the evolution of the mass fraction density function $P_x(y,\delta_j,t)$ with time for each contactor x=1..N. This system is subject to the initial conditions $P_x(y,\delta_j,t_o) = \phi_x(y,\delta_j)$.

In order to completely describe the behaviour of the adsorption circuit, the aqueous phase mass balance must also be considered.

4.6.3 The aqueous phase mass balance

The aqueous phase mass balance can be derived by considering the differential conservation equation:

Gold in solution flowing into vessel x
- Gold in solution flowing out of vessel x
= Rate of gold adsorbing onto carbon in vessel x
+ Rate of accumulation of solution in vessel x

Assuming that the aqueous phase is well mixed, the solution mass balance is expressed mathematically as:

$$Q(C_{x-1}-C_x)=M_x^c\sum_{j=1}^{NDC}\int_0^{\infty}P_x(y,\delta_j,t)R(y,\delta_j,C_x)dy+V_x^s\frac{dC_x}{dt} \qquad \{4.10\}$$

Here Q is the flowrate of the aqueous phase, V_x^s is the volume of solution in contactor x and M_x^c is the mass of carbon in contactor x and NDC is the number of carbon particle sizes being used.

Equation {4.10} is a system of ordinary differential equations which describes how the bulk concentration C_x varies with time in each contactor x. This system of equations is subject to the initial conditions $C_x(t_0)=C_{0x}$ and must be solved simultaneously with {4.9}.

In order to simulate the dynamic behaviour of a CIP adsorption system, it is necessary to solve simultaneously the system of equations {4.9} and {4.10}. As R is chosen arbitrarily, it is not possible to solve these equations analytically for the general case, although it may be possible to find analytic solutions for particular functional forms used for R. The next section will describe numerical methods which can be used to solve these equations.

4.6.4 Numerical methods

A robust method which works well for all situations is required for the integration of the system of PDEs {4.9} and {4.10}. However, numerical techniques for solving PDEs are not as well established or as robust as those for ODEs.

The system of equations to be solved {4.9} and {4.10} may be simplified using the method of characteristics. This method finds directions in the relevant plane(s) which transform the integration of a system of PDEs into the integration of a system of ODEs along the characteristic directions (Ames 1977).

The characteristic directions are determined by integrating the ODEs defining the property velocities. Thus the method of characteristics seems well suited to PDEs arising from the application of the population balance, especially in the case where only convective changes in the relevant properties occurs.

Applying the method of characteristics to the system {4.9} and {4.10}, the characteristic directions are given by:

$$\frac{dy}{dt}\Big|_I = R(y,\delta_j,C_x) \qquad \{4.11\}$$

$$\frac{dy}{dt}\Big|_{II} = 0 \qquad \{4.12\}$$

Along the I characteristic directions (defined by the integration of the system of equations {4.11}) integrate:

$$\frac{dP_x}{dt}\Big|_I = -P_x \frac{\partial R(y,\delta_j,C_x)}{\partial y} \qquad \{4.13\}$$

Along the II characteristic directions (defined by the integration of the system of equations {4.12}) integrate:

$$\frac{dC_x}{dt} = \frac{Q}{V_x^s}(C_{x-1} - C_x) - \frac{M_x^c}{V_x^s} \sum_{j=1}^{NDC} \int P_x(y,\delta_j,t) R(y,\delta_j,C_x) dy \qquad \{4.14\}$$

Thus the problem of solving the system of PDEs {4.9} and {4.10} is equivalent to solving the simultaneous system of ODEs {4.11}, {4.12}, {4.13} and {4.14}.

The problem has now been reduced to that of solving a set of ODEs. Stable and robust algorithms exist which can be used for this problem. As the equations to be solved are not stiff, Runge-Kutta type algorithms can be used.

The Runge-Kutta algorithms also lend themselves to control of the integration step length as the local truncation error can be estimated at each step. Use of an algorithm which provides step length control may significantly reduce computation effort when simulating adsorption. At the start of an adsorption period, rates are high, requiring small step lengths for sufficiently accurate answers. However, as the carbon loads up the rate of adsorption decreases allowing the same accuracy of integration to be maintained for bigger step lengths.

Normal practice is to simulate the adsorption behaviour over a number of adsorption cycles (the final conditions of one cycle being the initial conditions for the next) until the adsorption behaviour is the same as on the preceding cycle i.e. until a pseudo steady-state is reached. This may require the simulation of many adsorption cycles, thus any saving on computation effort on each cycle may lead to significant decreases in overall computation time.

The integration algorithm implemented to solve the system of differential equations defining the behaviour of the adsorption system is a variable step length algorithm proposed by Fehlberg (1970). This method is based on a fourth-order Runge-Kutta algorithm. The local truncation error is estimated using a higher order approximation. The Fehlberg derivation of the

Runge-Kutta scheme also has the advantage of having considerably smaller truncation errors (due to the coefficient values derived by Fehlberg) than the standard implementation.

Practical methods of implementing variable step length integration algorithms were adopted from the work of Thomas (1986). These algorithms were implemented as a general purpose numerical integration subroutine. The Pascal code for this subroutine, as well as tests of the accuracy of the method, can be found in Appendix 3.

4.6.5 Implementation of the method of characteristics

Using characteristics the problem consists of integrating the system of ODEs {4.11}, {4.13} and {4.14} for the chosen number of characteristics. The number of characteristics used depends on the accuracy required. Ten to two hundred characteristics have been used, and it has been found that the solution obtained using forty characteristics approaches that using two hundred characteristics. Thus it appears that forty characteristics per contactor is sufficient, although this is obviously dependent on the particular case being simulated.

The initial values for each characteristic are chosen to span the range of loading values appropriate to the problem. When integrated, the characteristics define the trajectory of a particle in the phase space with the given initial loading and particle size.

Equation {4.13} is integrated along these characteristics, with the integration of {4.14} performed with values defined by integration of {4.11} and {4.14}. The initial values for {4.13} are the initial mass fraction densities at the corresponding loading values used as the initial conditions for the integration of {4.11}. As the values defined by the integration of {4.11} and {4.13} are not equally spaced, the integral in {4.14} was evaluated using the trapezoidal rule.

The major problem resulting from the use of the method of characteristics is that values of P_x at the end of an adsorption cycle are determined at

loading (y) coordinates defined by the integration of {4.11}; not at any regularly spaced points. In general, these points will be different for each contactor in the cascade.

The simulation of the mixing of carbon having different loading distributions cannot be accomplished unless the loading distribution functions $P_x(y_i,\delta_j,t)$ for each stage and each particle size δ_j are known at the same loading values $\hat{y}_i$ (i=1 ... Number of characteristics) at the end of an adsorption cycle. This problem is illustrated in Figure 4.3.

It is thus necessary to develop an interpolation technique which will allow the loading distribution function $\hat{P}_x$ to be determined at a given set of loading values $\hat{y}_i$ (i=1 ... Number of characteristics) from information about P_x at the y_i values defined by the integration of {4.11} and {4.13}

The requirement for the interpolation technique is that the integral:

$$I = \int_0^\infty P_x(y,\delta_j,t)dy = f_x(\delta_j)$$

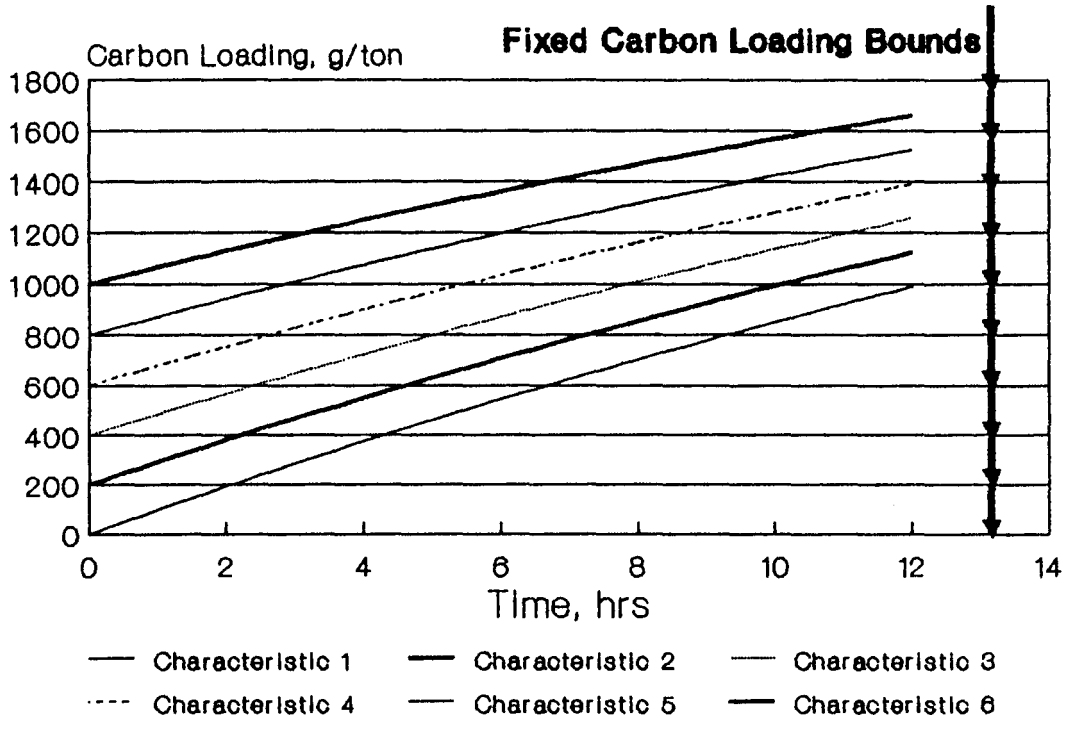
where $f_x(\delta_j)$ is the mass fraction of carbon in tank x having particle size δ_j.

when evaluated at the y, δ_j, P_x values defined by the integration of {4.11} and {4.13} be the same as the value of the integral calculated at the given $\hat{y}_i$ values and the calculated $\hat{P}_x$. This is most critical as the above integral represents the integration of the mass fraction density over all loading values and has a well defined value ($f_x(\delta_j)$). Errors in this quantity will have serious effects on the accuracy of the simulation.

Although standard linear and polynomial interpolation techniques were tried, none met the above requirement. Thus a method based on constrained minimization was developed.

The constraint is that the integral I at the fixed loading values $\hat{y}_i$ and the calculated $\hat{P}_x$ values be equal to $f_x(\delta_j)$ for each carbon particle size.

Figure 4.3 : The Characteristics Interpolation Problem



The relative difference between the $\hat{P}_x$ values at the fixed $\hat{y}_i$ loading values that satisfies the constraint (i.e. the values being sought) and those P_x values calculated at the fixed loading values $\hat{y}_i$ by linear interpolation of the y, δ_j and P_x values defined by the integration of {4.11} and {4.13} are then minimized.

This allows the constraint to be met while preserving the shape of the original P_x distribution. Appendix 4 presents the mathematics of this method and also shows an example of results obtained using this interpolation technique.

The next section provides some examples of CIP simulations obtained using the methods developed.

4.6.6 Simulation results

Although many simulations could be performed showing the effect of various parameters on CIP plant performance, these predictions would be different for different rate expressions. (Hopefully only the magnitudes of the effects would be different with different rate expressions and not the directions of the predicted trends.)

This chapter is not concerned with the predictions of a particular rate expression, but rather how well the population balance framework accommodates the evolution of the loading distributions with time. Thus only examples illustrating this point will be presented.

All simulations were performed using the rate expression developed by Dixon, Cho and Pitt (1978). As the dependency of the rate expression parameters on carbon particle size have not been determined, only one size class was used in all simulations. The following parameter values were used:

k_1	0.012
k_2	0.0019
y^*	3328 g/ton
M_c	4.32 tons
V_t	75 m^3

Q 75 m^3 per hour
N 5 contactors
C_{in} 1.9 g per m^3

Adsorption period = 24 hours

Figure 4.4 illustrates the loading density distributions used as the initial condition in all 5 CIP contactors.

Figures 4.5, 4.6 and 4.7 illustrate how the cumulative loading distributions for each stage converges as a pseudo steady-state is reached. In these illustrations, 50% of the carbon in each stage was moved forward to the next stage after each adsorption cycle.

Figure 4.8 is a three dimensional plot showing the movement of the loading distribution in stage 1 as the system approaches pseudo steady-state.

Figures 4.5, 4.6, 4.7 and 4.8 all show the loading distribution after carbon has been transferred and mixed at the end of the relevant adsorption cycle. Figure 4.9 illustrates the effect that mixing carbon from stage 2 into stage 1 at the end of the adsorption cycle has on the loading distribution in stage 1 after 1 adsorption cycle.

One of the most important factors affecting the performance of the adsorption section of a CIP plant is the rate at which the carbon is moved through the plant. Assuming the mass of carbon in each contactor remains constant, the rate of carbon movement can be increased by increasing the percentage transferred from each contactor.

This percentage is often limited by factors such as equipment capacities. The higher the mass flow of carbon moved through the system, the larger the elution and regeneration facilities have to be. Thus some balance between increased adsorption efficiency and increased capital and operating costs in the elution-regeneration section must be reached in an optimum plant design.

Figure 4.10 illustrates how the mass of carbon transferred influences the loading distribution in stage 1 after 15 adsorption cycles. For this case, 15 cycles was very close to the pseudo steady-state. It can be clearly seen that

Figure 4.4 : Initial Distribution Of Loadings

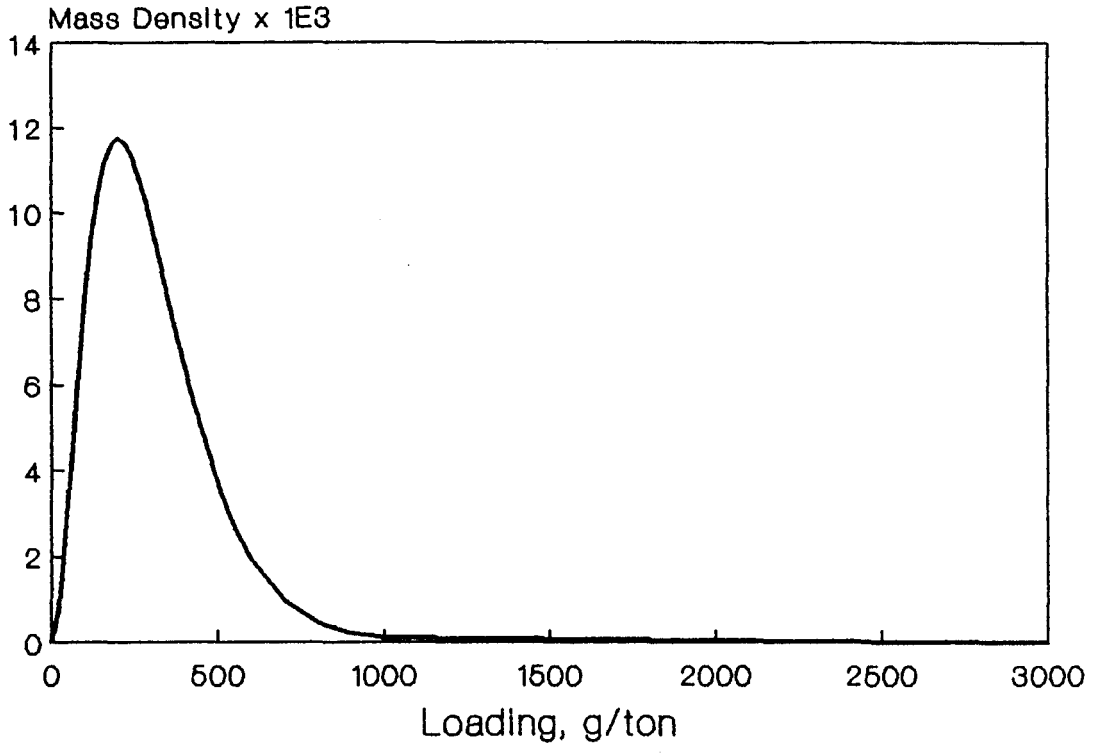


Figure 4.5 : Loading Distributions After 1 Cycle

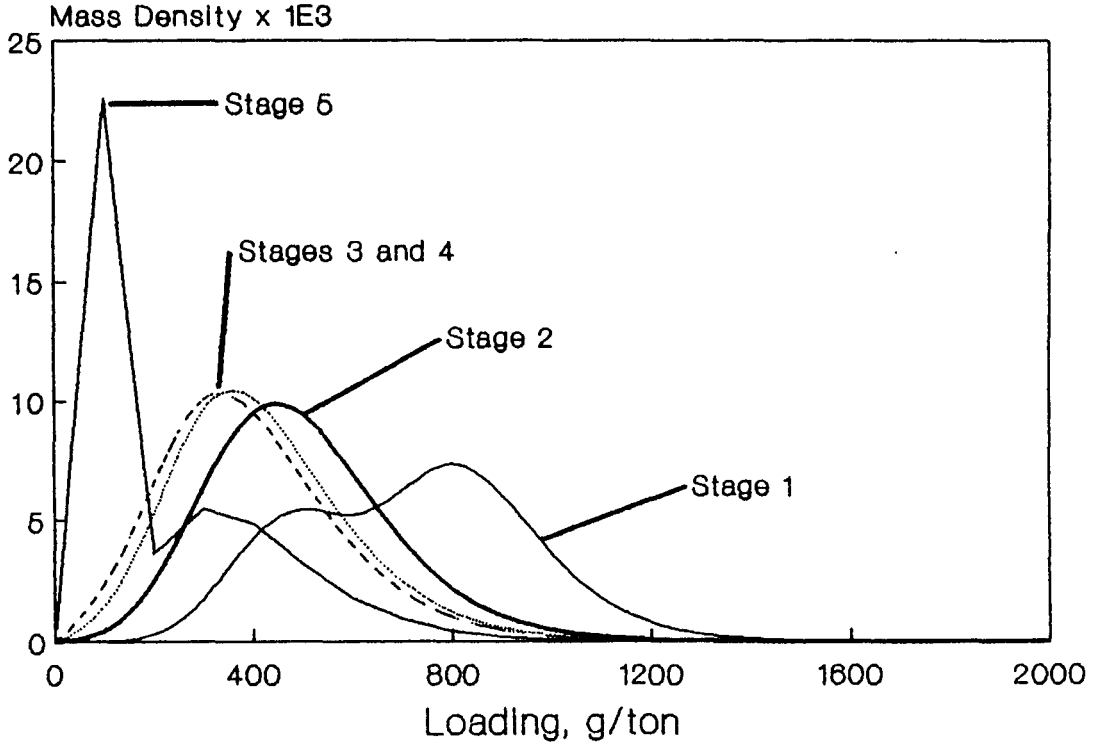


Figure 4.6 : Loading Distributions After 5 Cycles

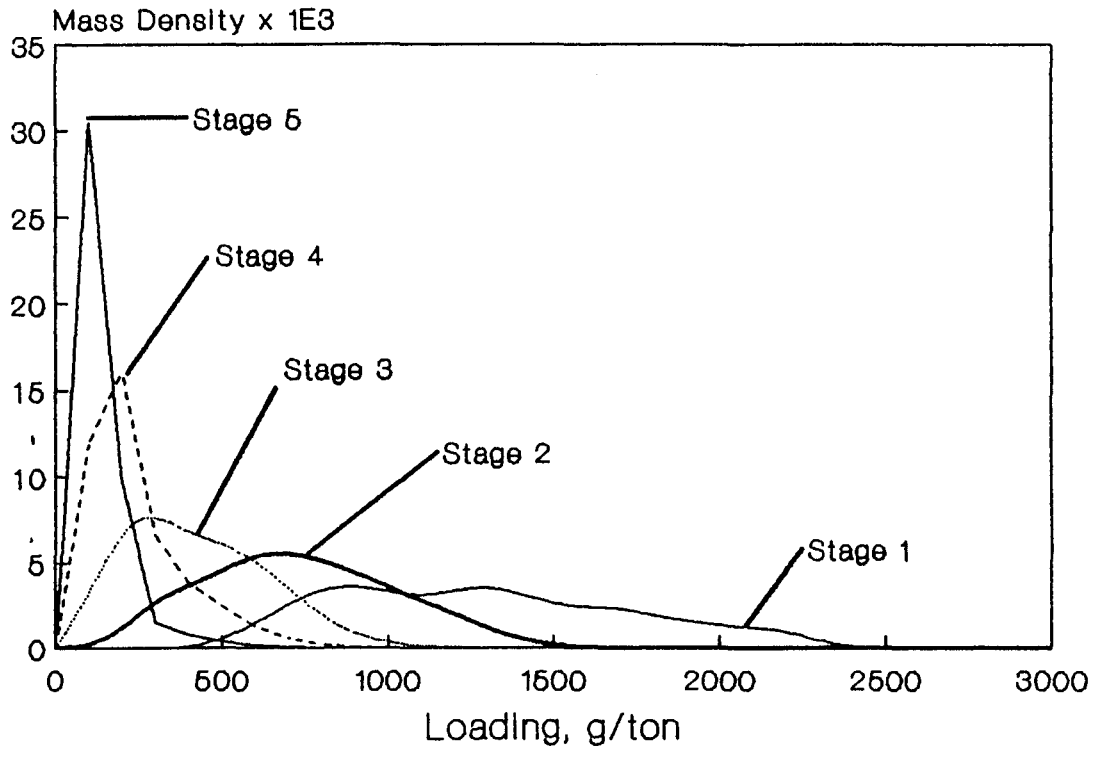


Figure 4.7 : Loading Distributions After 10 Cycles

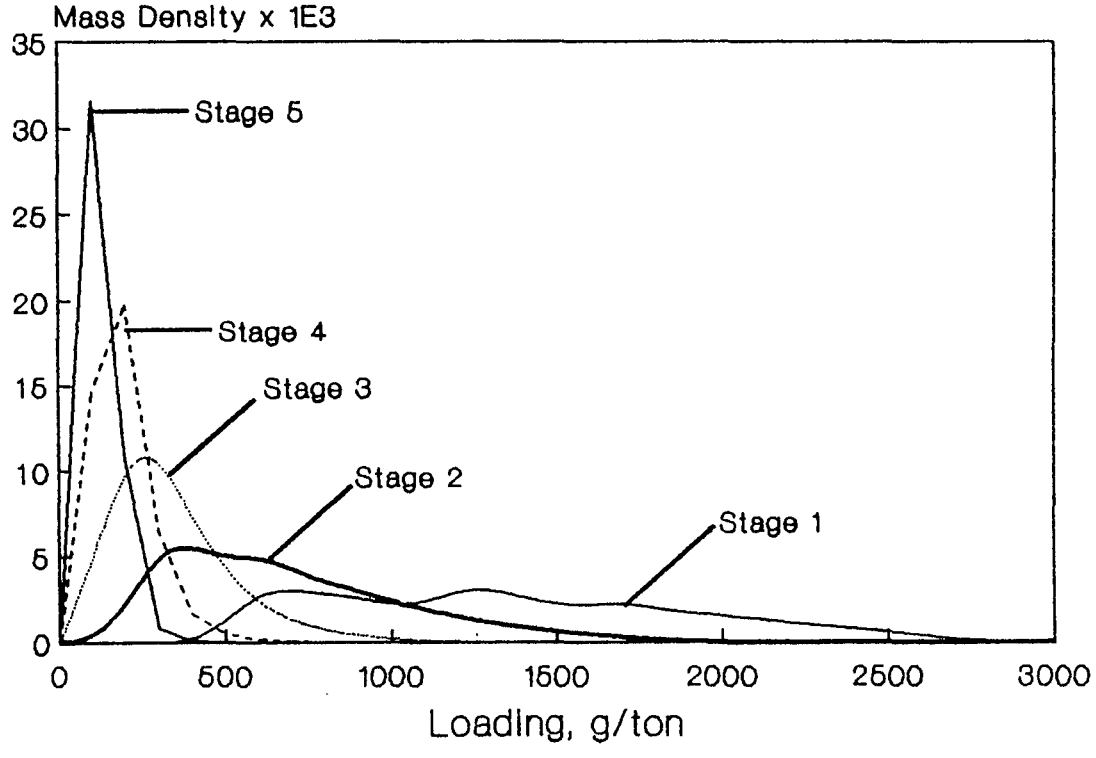


Figure 4.8 : Movement Of Loading Distribution In Stage 1 As Steady-State Approaches

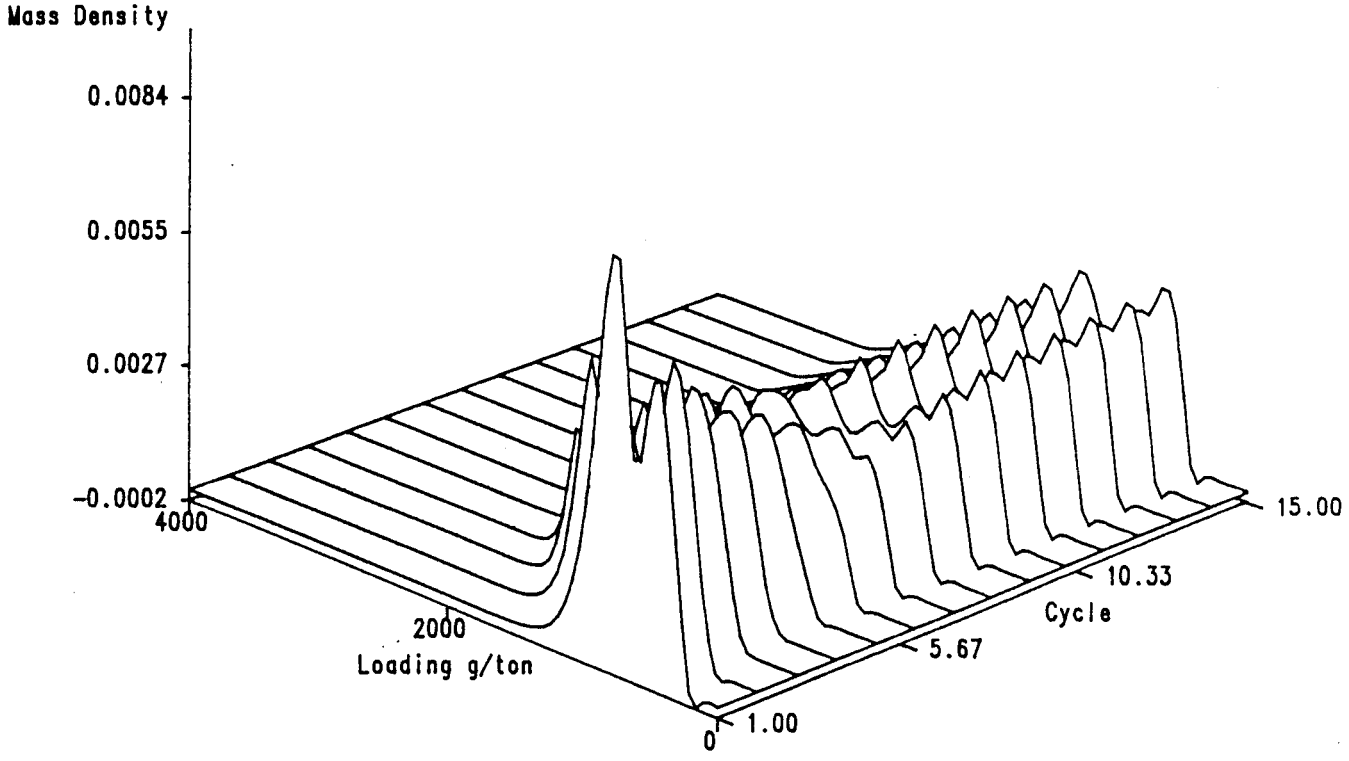


Figure 4.9 : Mixing Of Carbon From Stages 1 And 2 After 1 Cycle

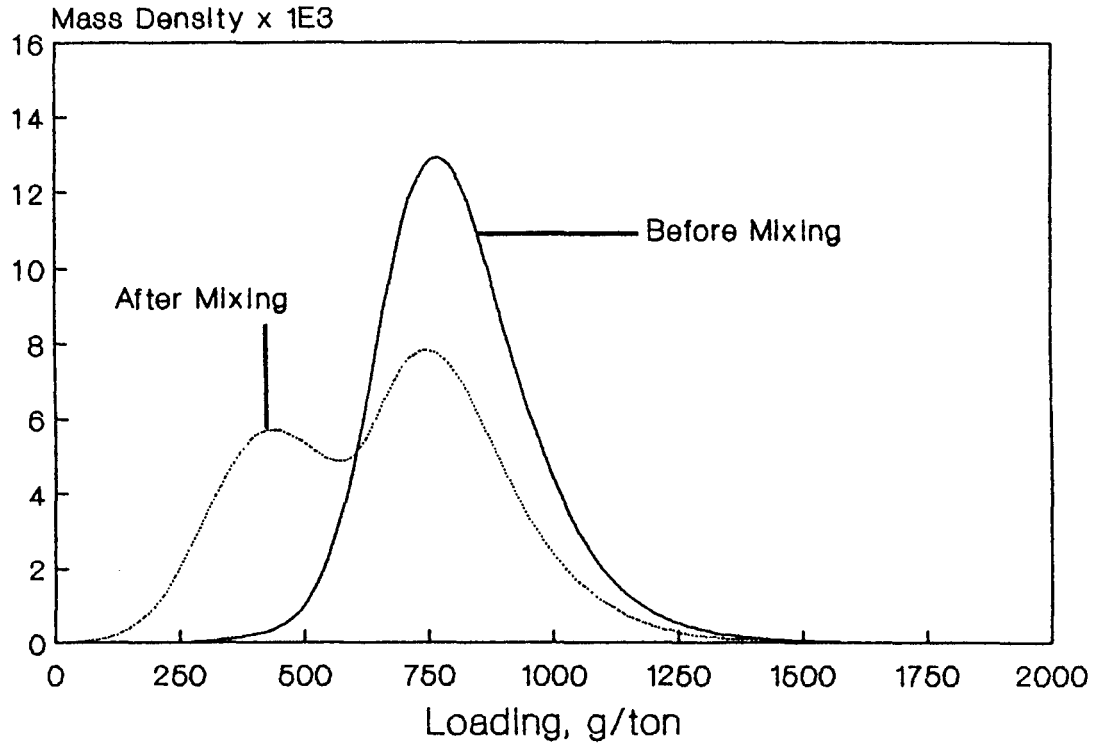
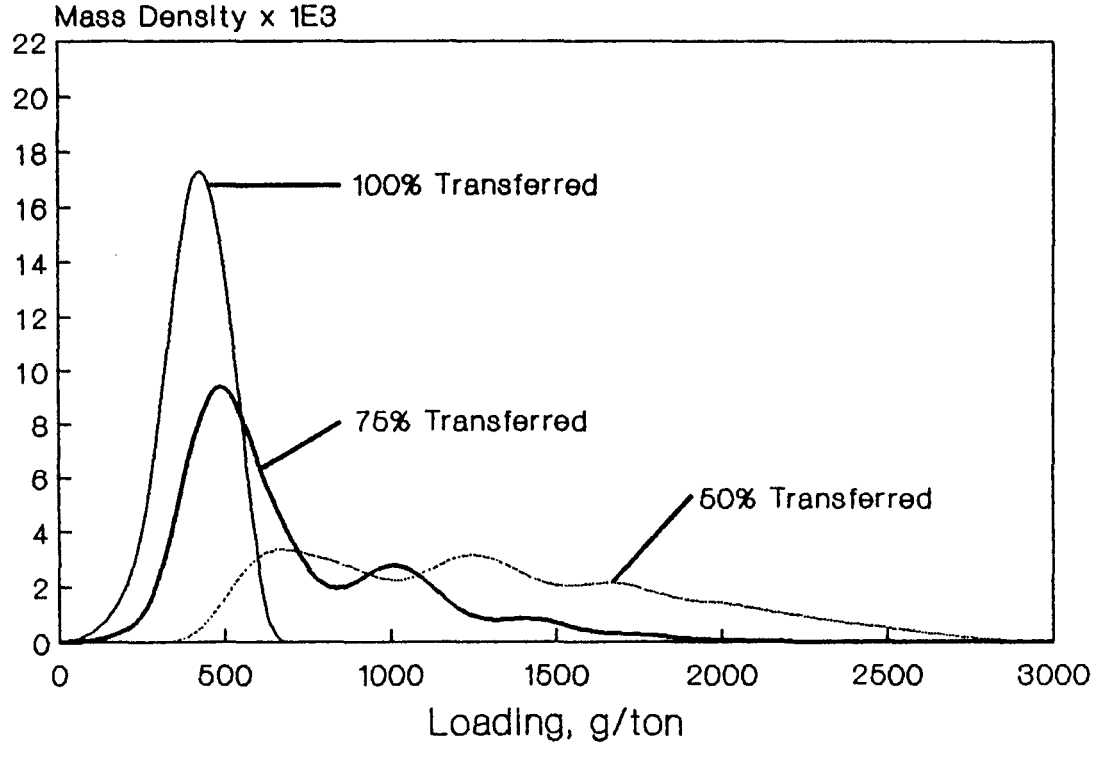


Figure 4.10 : Effect Of Movement Rate On Distribution In Stage 1 (15 Cycles)



adsorption efficiency increases markedly as the amount of carbon transferred increases. This can be concluded as the average loading for the highest transfer rate is the lowest implying a faster rate of adsorption and corresponding decrease in soluble loss.

4.7 Conclusions

This chapter has discussed the history of the CIP adsorption process and reviewed published attempts at the modelling of this process. Most of these attempts are not suitable for use in a general simulator to be used in an industrial environment. This is because the models are either complex, making parameter estimation difficulty and requiring much computer time, or the models do not reflect reality or both.

It was demonstrated that the macroscopic population balance equations are well suited to use in the modelling of distributed particulate systems, such as the CIP adsorption process.

Numerical techniques were developed to solve the equations resulting from the application of the population balance. The method of characteristics allowed the original system of PDEs to be transformed into a set of ODEs. An efficient ODE solver with step length control was implemented to solve these equations.

It was shown that the model derived can provide quantitative descriptions (for an arbitrary rate expression) of the evolution of the carbon loading distributions in each contactor as adsorption proceeds.

However, the computational cost of obtaining a detailed description of the loading distributions is high; a large system of ODEs (at least 40 per contactor) must be solved to obtain the information. Thus a mainframe computer is required in order to obtain results in reasonable times. This makes this modelling approach unsuitable for use in a microcomputer based simulator.

The next chapter will show how the method of moments can be used with the macroscopic equation to reduce the numeric work to such an extent that the resulting model can be easily solved on a microcomputer. In addition, it will be

shown how the macroscopic equation can be used to model the real world phenomenon taking place on a CIP plant which an industrially orientated model should be able to describe.

CHAPTER 5

A FLEXIBLE CARBON-IN-PULP MODEL FOR INDUSTRIAL USE

5.1 Introduction

This chapter will describe the development of a CIP adsorption model based on the macroscopic population balance equation and the method of moments which meets the criteria set for an industrially acceptable model discussed in section 4.4.

An additional complication which may exist on industrial plants, not mentioned yet, which will be considered in this chapter is the possibility of gold leaching in the adsorption section. Laxen (1986) has described how gold may continue to be leached in the CIP contactors. This extra gold dissolution may make an important contribution towards the feasibility of a particular CIP operation, particularly for low grade ores. The extra leaching also has a significant influence on the adsorption efficiency, as will be shown later in the chapter. A realistic CIP model should be able to quantify the effects of this dissolution. The model developed in this chapter is in effect a carbon-in-leach (CIL) model. When leaching is not considered or does not take place, the model becomes a CIP model.

Various rate expressions which have been used to describe the leaching of low-grade ores will be discussed in this chapter. Models for the CSTR leaching process based on appropriate rate expressions will then be presented.

The integration of the resulting CIP/CIL model in the MicroSim simulator will also be discussed in this chapter. The chapter will conclude with some simulation results for the CIP/CIL process using MicroSim.

5.2 Development Of A Flexible Adsorption Model

Figure 4.2 illustrates the flows of pulp and pulp containing carbon which may exist on a real CIP plant. It should be noted that the configuration presented in Figure 4.2 is a general representation of various types of circuits. These are:

a) The conventional batch counter-current CIP-CIL-RIP[1] type circuits.

b) Continuous counter-current CIP-CIL-RIP type circuits where the adsorbent is moved continuously.

c) Circuits in which the particulate and aqueous phases flow co-currently where there is mass transfer between the aqueous and particulate phases such as leaching.

d) Circuits in which the particulate phase flows both counter-current and co-current to the aqueous phase.

Thus, if the rate expression allows it, the techniques to be described may be used to model any of the above circuits.

5.2.1 The carbon mass balance

In order to derive a model for the CIP process as it behaves industrially, the macroscopic population balance equation {4.7} will be applied to the general situation illustrated in Figure 4.2.

For this case it will be assumed that carbon may be present in the slurry streams into and out of a particular contactor(s). It will be assumed, as in the previous case, that no carbon abrasion or breakage occurs. As described before this assumption is made because of the lack of quantitative information regarding carbon breakage and abrasion. If this information were available, the macroscopic population balance would still be an ideal basis for the development of the adsorption model.

1 RIP is an abbreviation for the resin-in-pulp process where resins are used as the adsorbent.

Again, the properties of interest will be the loading y and the carbon particle size δ. Under the conditions illustrated in Figure 4.2, the macroscopic equation {4.7} can be written for contactor x as:

$$\frac{\partial\{V_x(t)\overline{\Psi}_x(y,\delta,t)\}}{\partial t} = Q_x^{in}\overline{\Psi}_x^{in} - Q_x^{out}\overline{\Psi}_x^{out} - V_x(t)\frac{\partial\{R(y,\delta,C_x)\overline{\Psi}_x(y,\delta,t)\}}{\partial y} \qquad \{5.1\}$$

Equation {5.1} can be expanded to:

$$\overline{\Psi}_x\frac{\partial V_x}{\partial t} + V_x\frac{\partial\overline{\Psi}_x}{\partial t} = Q_x^{in}\overline{\Psi}_x^{in} - Q_x^{out}\overline{\Psi}_x^{out} - V_x\left(\overline{\Psi}_x\frac{\partial R}{\partial y} + R\frac{\partial\overline{\Psi}_x}{\partial y}\right) \qquad \{5.2\}$$

It should be noted that any number of streams flowing into a contactor (see Figure 4.2), can be represented as a single stream for the purposes of simulation. The single stream is obtained by performing a mixing operation on all streams entering the contactor. The same principle applies to streams leaving a contactor.

As no size changes occur, the mass density function can be substituted for the number density function. Thus {5.2} is rewritten:

$$P_x\frac{\partial V_x}{\partial t} + V_x\frac{\partial P_x}{\partial t} = Q_x^{in}P_x^{in} - Q_x^{out}P_x^{out} - V_x\left(P_x\frac{\partial R}{\partial y} + R\frac{\partial P_x}{\partial y}\right) \qquad \{5.3\}$$

For this case it is more convenient to define $P(y,\delta,t)dy$ as the mass of carbon per unit volume having a loading between y and y+dy and carbon particle size δ. This is done as the mass of carbon in each contactor is now a function of time.

In order to proceed with the development, a functional form for the rate expression R needs to be chosen. Although this leads to less generality,

simplifications can be made and thus the amount of computation is reduced substantially making it feasible to carry out the simulations using a microcomputer.

Hulbert and Katz (1964) describe a method to simplify the population balance PDE by transforming it into an ODE (or system of ODEs) which represent the changes of the moments of the distributions with time. Some information is lost, as the full distribution is not described but only the moments of the distribution. However, in the industrial situation it would not be necessary to have the full description of the density function; in a CIP application knowledge of the first moment (the average loading value) for each particle size would normally suffice. Indeed no work regarding the measurement of these distributions has been published. Thus even if a model could predict them, it would be very difficult to ascertain the accuracy of the predictions.

In a complex situation such as CIP, the validity of the rate expression may not be the most critical influence on the accuracy of the resulting model. In their work, Williams and Glasser (1985) show that parameters such as liquid bypassing, which reflect the imperfections of the physical system, have a major effect on their model predictions. They conclude that it is more important to model the physical characteristics of the system properly, rather than concentrate on a complex "correct" rate expression. In section 4.4.1 it was shown that even the simple rate expressions fit observed adsorption rate data well.

Due to the heterogeneous nature of activated carbon and the experimental difficulties in investigating the gold-carbon adsorption system, it is not likely that a "correct" rate expression will ever be developed. Criteria for the selection of a suitable rate expression used in this development are that the expression fits kinetic data acceptably and that the expression accommodates the phenomena which are known to occur. For example, the gold-carbon equilibrium isotherm is strongly non-linear, thus the rate expression should reduce to a non-linear isotherm at equilibrium.

In order to successfully apply the method of moments to {5.3} a rate expression which is linear in the gold loading term(s) needs to be chosen. A rate expression which has loading terms y^n where n is a positive integer could also be used. For the latter case a moment truncating procedure developed by Hulbert and Katz (1964) would need to be used.

Two rate expressions exist which are linear in the gold loading term, that suggested by Dixon, Cho and Pitt (1978) and that due to Nicol, Fleming and Cromberge (1984). As Nicol et al assume a linear isotherm, the Dixon expression is considered to be more realistic. The form of this expression is described in section 4.3.

For the development that follows, it will be assumed that the rate constants k_1 and k_2 in the Dixon rate expression are functions of the carbon particle size.

Multiplying {5.3} by y and integrating with respect to y between the limits of zero and infinity, the differential equation for each particle size δ is:

$$\frac{dV(t)_x}{dt} I_1 + V(t)_x I_2 = Q_x^{\text{in}} I_3 - Q_x^{out} I_4 - V_x(I_5 + I_6) \qquad \{5.4\}$$

where:

$$I_1 = \int_0^\infty y P_x(y,\delta,t)\,dy$$

$$I_2 = \int_0^\infty y \frac{\partial P_x(y,\delta,t)}{\partial t}\,dy$$

$$I_3 = \int_0^\infty y P_x^{\text{in}}(y,\delta,t)\,dy$$

$$I_4 = \int_0^{\infty} y P_x^{out}(y,\delta,t)dy$$

$$I_5 = \int_0^{\infty} y P_x(y,\delta,t)\frac{\partial R}{\partial y}dy$$

$$I_6 = \int_0^{\infty} y R \frac{\partial P_x(y,\delta,t)}{\partial y}dy$$

I_1 is the total amount of gold per unit volume on carbon having particle size δ in contactor x at time t and can be expressed as:

$$I_1 = \varepsilon(t,\delta)\overline{y}(\delta)$$

Here $\overline{y}(\delta)$ is the average loading of gold on carbon particles of size δ. $\varepsilon(t,\delta)$ is the mass of carbon of size δ per unit volume of contactor x.

I_3 is the total amount of gold on carbon of size δ per unit volume of pulp flowing into contactor x at time t and can be expressed as:

$$I_3 = \varepsilon^{in}(t,\delta)\overline{y}^{in}(\delta)$$

Similarly, I_4 is the total amount of gold on carbon of size δ per unit volume of pulp flowing out of contactor x at time t and can be expressed as:

$$I_4 = \varepsilon^{out}(t,\delta)\overline{y}^{out}(\delta)$$

Here:

$\varepsilon^{in,out}(t,\delta)$ is defined as the mass of carbon having size δ per unit volume of the pulp in the input or output stream of contactor x.

$\overline{y}^{in,out}(\delta)$ is the average loading of gold on carbon particles of size δ.

$Q_x^{in,out}$ is the volumetric flowrate of pulp into or out of contactor x. These flowrates are assumed to be constant and are therefore not functions of time.

I_2 can be evaluated by changing the order of integration and differentiation:

$$I_2 = \frac{d}{dt}\int_0^\infty yP_x(y,\delta,t)dy$$

$$= \frac{d\{\varepsilon(t,\delta)\bar{y}(d)\}}{dt}$$

$$= \varepsilon(t,\delta)\frac{d\bar{y}(\delta)}{dt} + \bar{y}(\delta)\frac{d\varepsilon(t,\delta)}{dt}$$

Substituting the Dixon rate expression for R, I_5 is written:

$$I_5 = \int_0^\infty yP_x(y,\delta,t)\frac{\partial}{\partial y}\{k_1(\delta)C_x(t)(y^* - y) - k_2(\delta)y\}dy$$

$$= -\{k_1(\delta)C_x(t) + k_2(\delta)\}\varepsilon(t,\delta)\bar{y}(\delta)$$

Now I_6 is expanded to:

$$I_6 = \int_0^\infty \{k_1(\delta)C_x(t)(y^* - y) - k_2(\delta)y\}y\frac{\partial P_x(y,\delta,t)}{\partial y}dy$$

Integrating by parts:

$$I_6 = yRP\,|_0^\infty - \int_0^\infty P\frac{d(Ry)}{dy}dy$$

Performing the necessary manipulation and noting that the first term of the above expression is zero at both ends leads to:

$$I_6 = -\varepsilon(t,\delta)\{k_1(\delta)C_x(t)y^* - 2[k_1(\delta)C_x(t)\bar{y}(\delta) + k_2(\delta)\bar{y}(\delta)]\}$$

Substituting the expressions for the integrals into equation {5.4} gives:

$$\bar{y}(d)\varepsilon_x(t,\delta)\frac{dV_x}{dt} + V_x\bar{y}(\delta)\frac{d\varepsilon_x(t,\delta)}{dt} + V_x\varepsilon_x(t,\delta)\frac{d\bar{y}(\delta)}{dt}$$

$$= Q_x^{\text{in}}\varepsilon_x^{\text{in}}(t,\delta)\bar{y}(\delta)^{\text{in}} - Q_x^{out}\varepsilon_x^{out}(t,\delta)\bar{y}(\delta)^{out}$$

$$+V_x\varepsilon_x(t,\delta)\{k_1(\delta)C_x(t)[y^* - \bar{y}(\delta)] - k_2(\delta)\bar{y}(\delta)\}$$

{5.5}

It will be assumed that the dynamics of the system are such that the rate of change of total active volume dV/dt, is negligible. This does not imply that the rate of change of the mass of carbon in the contactor is also negligible.

This is because the volume of carbon in the contactor is much smaller than the total reactor volume. Thus a significant change in the carbon volume may only produce a small change in the overall active volume. This assumption would be true as most plants operate at a carbon concentration of less than 60 g/l (approximately 4% by mass).

Assuming dV/dt equals zero, then $Q^{in} = Q^{out} = Q$. However, if the dependence of output stream volumetric flowrate Q^{out}, as a function of reactor volume V is known, then these assumptions do not need to be made, but an extra differential equation, which expresses the volume of each contactor V as a function of time, would be needed.

Due to the assumptions made about dV/dt and the resulting condition $Q^{in} = Q^{out} = Q$, a restriction on Q has to be imposed i.e. that Q is time invariant. Consider if this were not the case. A simulated change of feed rate (Q) to the entire system would result in an instantaneous change in the flowrate through each contactor, which is not a good approximation of what would happen in reality.

A mass balance on the carbon phase gives:

$$\frac{dV_x\varepsilon_x(t,\delta)}{dt} = Q\{\varepsilon_x^{in}(t,\delta) - \varepsilon_x^{out}(t,\delta)\}$$

{5.6}

Substituting {5.6} into {5.5} and noting that $\bar{y}(\delta)^{out} = \bar{y}(\delta)$ as the system is assumed to be well mixed gives:

$$\frac{d\bar{y}(\delta)}{dt} = \frac{Q\varepsilon_x^{in}(t,\delta)}{V\varepsilon_x(t,\delta)}\{\bar{y}(\delta)^{in} - \bar{y}(\delta)\} + k_1(\delta)C_x(t)\{y^* - \bar{y}(\delta)\} - k_2(\delta)\bar{y}(\delta)$$

{5.7}

Note that {5.7} is a system of ODEs which describe how the average loading values for each carbon particle size in each contactor changes with time. Equation {5.7} is similar to the equation obtained by Carrier et al (1987) (see section 4.3). However, Carrier et al obtain their result by simply writing the carbon loading balance as "In - Out = Adsorbed + Accumulated". As the rate expression used by Carrier et al (that due to Dixon et al (1978)) is linear in the loading term y, the result obtained by Carrier et al is correct. However, if this simple approach was used with a non-linear rate expression, such as that due to Johns (1987), the result would not be correct. This can be dedeuced from the analysis presented above. This is because in general:

$$\int_0^\infty yR(y,\delta,C)P(y,\delta,t)dy \neq \varepsilon R(\bar{y},\delta,C)$$

if R contains non-linear forms of the loading term y. A similar analysis as presented here would have to be carried out for the particular rate expression R.

Note that if the rate expression due to Nicol, Fleming and Cromberge (1984) is used instead of that due to Dixon, Cho and Pitt, equation {5.7} would have the following form:

$$\frac{d\bar{y}(\delta)}{dt} = \frac{Q\varepsilon_x^{in}(t,\delta)}{V\varepsilon_x(t,\delta)}\{\bar{y}(\delta)^{in} - \bar{y}(\delta)\} + k(\delta)\left(C_x(t) - \frac{\bar{y}(\delta)}{A}\right)$$

In order to complete the mathematical description of the process, equations defining the dynamic behaviour of the aqueous phase need to be derived.

5.2.2 The aqueous phase mass balance

Assuming that the carbon is discretised into NDC size classes, with $\varepsilon_x(t,\delta)$ = mass of carbon in contactor x, per unit volume having representative size δ at time t, the solution mass balance for stage x can be written as:

$$\frac{dC_x(t)}{dt} = \frac{Q}{V_s^x}(C_{x-1} - C_x) + \frac{F}{V_s^x}(S_{x-1} - S_x) - \frac{V_x}{V_s^x}\sum_{i=1}^{NDC}\varepsilon_x(t,\delta)R(\bar{y}(\delta_i),\delta_i,C_x)$$

{5.8)

based on the solution balance as written in section 4.6.3. Here F is the mass flowrate of ore and S_x is the concentration of gold in the ore in stage x.

Models for calculating the leaching term $F(S_{x-1}-S_x)$ in equation {5.8} will now be discussed.

5.3 Models For Gold Leaching

5.3.1 Introduction

In order to model the extra dissolution which may occur in the adsorption section of a CIP plant, various rate expressions which have been used to model the rate of leaching were investigated. Three rate expressions were investigated. Regression analysis for each expression, for twenty five sets of batch leaching data, were carried out.

5.3.2 Experimental

Leach data from Vaal Reefs, Harmony, President Brand, President Steyn and Western Holdings gold mines were obtained from the Anglo American Research Laboratories, MINTEK, and the Department of Chemical Engineering, University of the Witwatersrand.

Head grades ranged from 2.6 to 15.2 g/t. Leaching times ranged from 24 to 72 hours. Leach vessels were both rolling bottles and batch STR's.

5.3.3 Rate expressions

The three rate expressions investigated were:

The MINTEK expression

This is an experimental expression used by MINTEK in the modelling of gold leaching (Nicol, Fleming and Cromberge 1984). The rate of leaching, L(S) is described by the following functional form:

$$L(S) = k_p(S - S_m)^2$$

Integrating for a batch configuration gives:

$$S(t) = \frac{S_o + k_p S_m (S_o - S_m)t}{1 + k_p (S_o - S_m)t}$$

Here S_m is the concentration of gold in the ore after an infinite leaching time and k_p is an empirical rate constant. S_o is the concentration of gold in the ore at time zero.

Rate expression due to Brittan (1975)

This is a variable activation energy model. The expression is derived by lumping together the rate-limiting effects as an Arrhenius activation energy barrier. As reaction proceeds, the more reactive material is leached

out first leaving behind material which is more refractory. Brittan (1975) models this effect by increasing the activation energy during leaching, hence the rate "constant" decreases as conversion increases.

The rate expression is written as :

$$L(S) = (S - S_m)e^{b_1\left(S - S_m\right) - b_2}$$

Here S_m has the same meaning as for the MINTEK expression. The parameters b_1 and b_2 are determined experimentally. This rate expression cannot be integrated analytically for the batch situation.

Rate expression due to Loveday et al (1970)

Loveday et al developed a model to quantify the kinetics of leaching of a low grade uranium oxide ore. It was assumed that the rate of dissolution of individual sites is a zero order process and that there is a distribution of the zero order rate constants.

The Schumann distribution function was used to describe the way in which the initial zero order rate constants are distributed. Good fits to experimental data over a range of times and temperatures were obtained by the authors. The effects of particle size distribution on the model parameters were also investigated.

The equations for the batch system can be written as:

$$S(t) = (S_o - S_m)\left(1 - \frac{n}{n+1}k_{\max}t\right) + S_m \qquad t < \frac{1}{k_{\max}}$$

$$S(t) = \frac{(S_o - S_m)}{n+1}\left(\frac{1}{k_{\max}t}\right)^n + S_m \qquad t > \frac{1}{k_{\max}}$$

Here S_o and S_m have the same meaning as above while k_{max} is the maximum rate constant and n is the shape factor in the Schumann distribution.

5.3.4 Parameter estimation techniques

For all 3 equations, the minimization subroutine ZXSSQ from the IBM IMSL (1982) mathematical software library was used to find the minimum sum of squares between predicted and experimental concentrations. The sum of squares of the relative errors was used in order to give equal weighting to all points. In the case of the Brittan rate expression, the subroutine DGEAR from the same software library, was used to solve the differential equation within the minimization loop of ZXSSQ.

5.3.5 Results

It was found that the Loveday rate expression fits the data best, with the MINTEK expression providing better fits than the Brittan expression. The averages and standard deviations for the sum of squares of the relative errors per data point for the entire data set for the three expressions are:

Loveday : 0.0074 ± 0.0083
MINTEK : 0.043 ± 0.084
Brittan : 0.10 ± 0.15

It can be seen that the sum of squares error for the Loveday expression is almost an order of magnitude less than for the other models. The error for the Brittan model is twice as large as for the MINTEK expression, even though the Brittan expression has three parameters while the MINTEK expression has two.

Typical plots of the data and the fitted points are shown in Figures 5.1 to 5.3. When graphs of the comparison between the fits obtained for the three expressions and the exeprimental points for the entire data set were examined it was seen that the rate expression due to Loveday describes the shape of the experimental data better than any of the other expressions, especially towards the end of the leaching period when the residue values are asymptoting to a constant value. It is this portion of the leaching curve which would be relevant in a CIP operation. Thus, the leaching model due to Loveday is the most suitable expression of those evaluated to model gold leaching.

Figure 5.1 : Leaching Model Fits (Data Set 1)

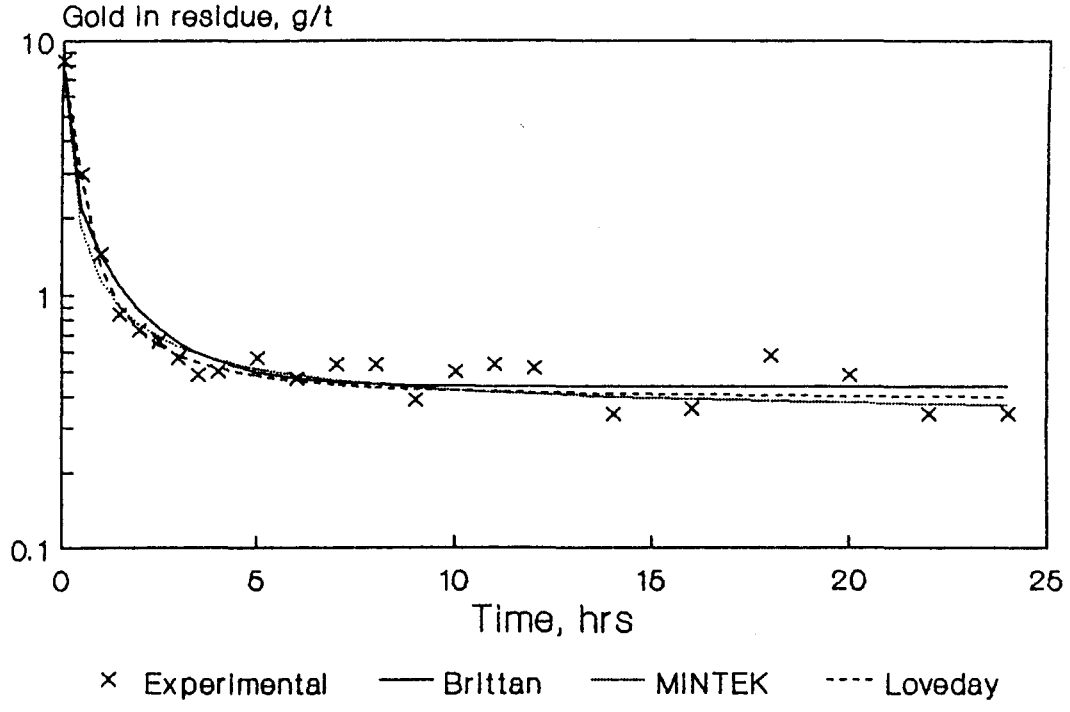


Figure 5.2 : Leaching Model Fits (Data Set 5)

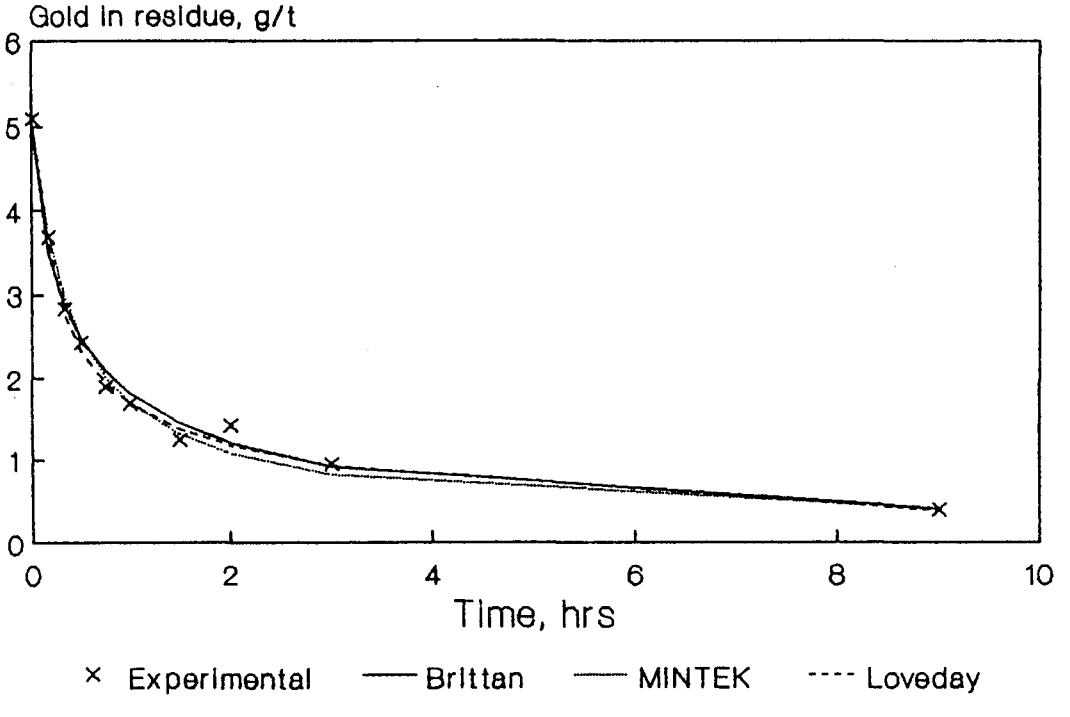
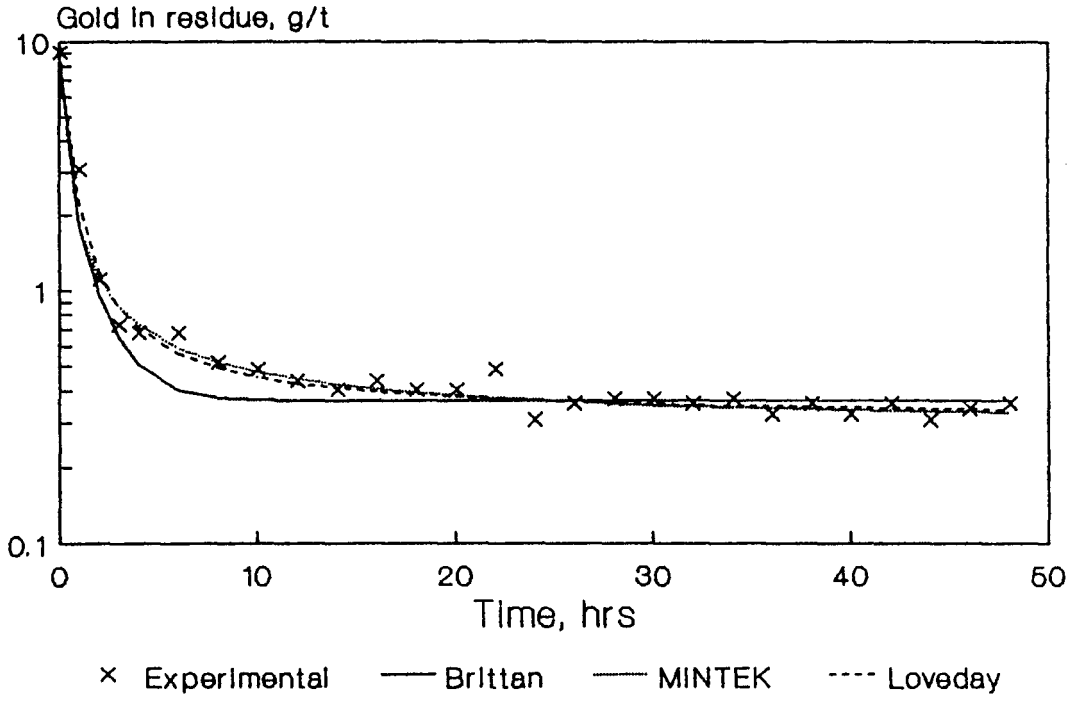


Figure 5.3 : Leaching Model Fits
(Data Set 11)



None of the rate expressions include the solution tenor as a variable. This implies that one of the above rate expressions may be used to calculate the values of S_x x=1..N in equation {5.8} without any knowledge of the solution profile. This independence implies that although the behaviour of the solution and carbon profiles is transient, the ore concentration values are steady-state values and only need to be calculated once (assuming no changes in the ore head grade or ore flow rate).

5.3.6 Continuous leaching models

Section 5.3.3 described the form of the leaching models for the batch configuration. In order to be used to predict leaching in a CIP/CIL situation, models based on these rate expressions need to be developed for the CSTR situation. This section develops the mathematics for the continuous situation for the two expressions found to fit the data well; the MINTEK and Loveday models.

The simple form of the MINTEK expression, and the fact that time does not appear explicitly in the expression allows the following model to be derived for a CSTR leaching circuit by performing a simple gold mass balance across CSTR x (Nicol 1986):

$$S_x = \frac{(2S_m - \alpha) + \sqrt{\alpha^2 + 4\alpha(S_{x-1} - S_m)}}{2} \qquad \{5.9\}$$

where:

$$\alpha = \frac{1}{\tau k_p}$$

and τ is the nominal ore residence time in a single STR. S_x is the concentration of gold in the ore in the exit of CSTR x.

Due to the fact that time appears explicitly in the Loveday expression, the batch expression must be weighted by the CSTR residence time distribution and integrated over all possible times which a single particle may spend in a series of CSTRs to find the bulk ore concentration in the exit stream of the last CSTR stream in the series.

When this is done it is found that the concentration of gold in the exit of CSTR x is given by:

$$S_x = (S_o - S_m)A^x + S_m \qquad \{5.10\}$$

where:

$$A = (1-\beta) - \frac{nk_{\max}}{n+1}\left(\tau(1-\beta) - \frac{\beta}{k_{\max}}\right) + \frac{1}{n+1}\left(\frac{1}{k_{\max}}\right)^n \beta I$$

$$\beta = e^{\frac{-1}{\tau k_{\max}}}$$

$$I = \int_o^1 \frac{dz}{\left(-\tau\log(z) + \frac{1}{k_{\max}}\right)^n}$$

Note that "log(z)" represents the natural logarithm of z.

A more detailed derivation of this equation can be found in Appendix 5.

5.4 Numerical Methods

From section 5.3.6 it can be seen that the continuous leaching models can be used to calculate the concentration of gold in the ore, S_x, in the exit of each CSTR given the ore headgrade, the residence time and the rate parameters for the leaching system. That is, no knowledge of the solution tenors are required

in the calculation. Note that the rate of leaching is not affected by the presence of carbon if all other factors are kept constant (Nicol, Fleming and Cromberge 1984).

This implies that the algebraic equations {5.9} or {5.10} can be solved quickly and easily to simulate the leaching behaviour of a CIP/CIL plant. Simulation runs have shown that values for the definite integral in equation {5.10} can be obtained using Simpsons rule with about 20 intervals.

Equation {5.9} or {5.10} can be used to provide values for S_x for all stages. These values must be used in the aqueous phase mass balance {5.8}. The initial value problem defined by ODEs {5.7} and {5.8} can then be solved simultaneously to model the dynamic behaviour of the aqueous and carbon phases. The ODE solution methods discussed in section 4.6.4 can be used for this.

5.5 Integration Of The CIP/CIL Model Into MicroSim

Integrating the CIP and leaching models into MicroSim served as a test for the streams data structures designed and implemented by Cilliers (Cilliers 1987). As coded by Cilliers, the executive for MicroSim made provision for slurry streams only. In addition, the aqueous phase was characterised using only a water flowrate. It was thus necessary to design data structures for carbon streams and to provide facilities for the description of the dissolved species which may be present in the aqueous phase of slurry streams.

The integration of the adsorption model will first be described.

5.5.1 The adsorption model

Before describing how the adsorption model was inserted into MicroSim, some remarks concerning the use of a dynamic model in a steady-state simulator will be made.

Dynamic nature of the model

Due to the nature of the CIP/CIL process, the model developed in section 5.2 was necessarily of a fully dynamic nature. MicroSim is a

steady-state simulator and none of the other models in MicroSim display time dependent behaviour. There is thus a need to justify the decision made to insert a dynamic model into a steady-state simulator.

The major reason for including the adsorption model in MicroSim, rather than developing a separate CIP software package, was that this allows the CIP operation to be simulated as part of a larger flowsheet.

For example, assume that modifications to the crushing circuit at a certain gold mine are to be made. This will affect the feed size distribution of the milling circuit and thus the grinding circuit product size distribution. This may then affect leaching and thus adsorption performance. If the adsorption model is integrated into MicroSim, the whole circuit can be simulated easily, allowing the effects of changes made to individual circuits on overall plant performance to be quantified.

If there are no recycles between the size reduction and leaching/adsorption circuits, it can be argued that the problem can be solved just as easily by first simulating the crushing/grinding circuit and using the information from this study to quantify the behaviour of the adsorption/leaching circuit (using a separate computer program) under the various conditions. However, if there is a recycle stream (or streams) between the processes, then the approach making use of separate computer programs is at best extremely tedious and at worst, will fail completely. Thus it was concluded that it is desirable to integrate the adsorption model into the simulator.

It should be noted that under steady feed and operating conditions, a CIP plant will reach a state where the dynamic behaviour of the process for any cycle is the same as for the previous cycle. That is, a point is reached where the way in which the various quantities (solution tenors, carbon loadings, carbon masses) vary with time on each cycle is exactly the same as for the previous cycle. This is normally termed pseudo steady-state.

At pseudo steady-state, the solution tenors, carbon loadings and carbon masses averaged over each adsorption cycle will be time invariant. For most simulation studies, the quantities of interest will be these time invariant average quantities. The adsorption model was implemented so that the user can instruct the model to iterate until the pseudo steady-state is reached. The model then passes the time invariant average quantities to the output streams. For this case, the adsorption model is essentially a steady-state model as all the other models in MicroSim are; for a given set of steady inputs the model calculates the corresponding steady outputs.

However, facilities were also provided which enable users to output dynamic data (to a separate data file) if required. The model is thus flexible enough to be used as part of a larger steady-state simulation exercise, or it can be used as a stand alone model capable of producing dynamic output.

Data structures

The data structures were extended using the method described in the MicroSim Technical Reference manual (Stange, Cilliers and King 1988b).

Changes were made to the record used to store the aqueous phase characteristics. This record was extended to allow the storage of the concentrations of a number of aqueous phase species. The unit of concentration chosen was grams of aqueous species per ton of solution. This was chosen due to the wide use of this unit in gold mining. Choosing mass based units of concentration instead of molar units is more suitable for the simulation of gold leaching and adsorption.

A record which allows the properties of carbon streams to be stored was also added. The record allows the following properties of a carbon stream to be stored:

- The carbon mass flowrate in kilograms per second
- The size distribution of the carbon, with the mass flowrate (kilograms per second) of carbon in each size class stored.

- As carbon particle size affects the rate of adsorption, it is likely that the loadings of each species will vary with carbon particle size. Thus the data structure was designed to store the concentration (in grams of adsorbate per ton of carbon) of each adsorbate in each size class. Again concentration units of grams per ton were chosen as they are most suitable for simulation of gold recovery processes. The adsorbates in the carbon record correspond to the aqueous phase components defined in the aqueous phase record.

Appendix 6 contains the Pascal type definitions used for the data structures as well as describing utility functions implemented to allow easy manipulation of these data structures.

The changes made to MicroSim in order to integrate these extended data structures were:

- The flowsheet editor (as described in section 3.3.1) was modified so that carbon containing streams are automatically identified by the editor.

- The data input phase was modified so that if hydrometallurgical units exist in the flowsheet being simulated, the user is required to specify the number of aqueous phase species to be used in the simulation. This is done in the same way as the way in which the user identifies the various minerals present in the ore. Names are given to the aqueous species by the user.

- If carbon streams exist in the flowsheet being simulated, the user defines the number of size classes and the class bounds used to describe the distribution of carbon particle size. This is done in a similar way to which the user defines the size distribution for the ore.

- If aqueous phase components exist, the user must instruct MicroSim as to which aqueous phase species are associated with which minerals in the ore. For example, assume that the user has defined two components in the ore 'gangue' and 'gold' and one species in solution, '[Au]'. The user would then associate '[Au]' with 'gold' by selecting the appropriate choice from a menu.

- If necessary the adsorption model asks the user to identify which aqueous species is adsorbed[2] and the leaching model asks the user to identify which mineral is leached. Together with the modifications described above, this allows the leach and adsorption models to identify the components (in the aqueous and ore phases) which must be considered in the mass balances performed by these models, in a user-friendly and general manner.

- The data input, editing, output, storage and retrieval phases were also modified to handle the extended structures in a general manner, transparent to the user.

In summary, the MicroSim data structures were extended to cope with carbon streams and aqueous species. The program was modified so that the simulation of processes requiring use of these data structures are easy and natural for the user to carry out.

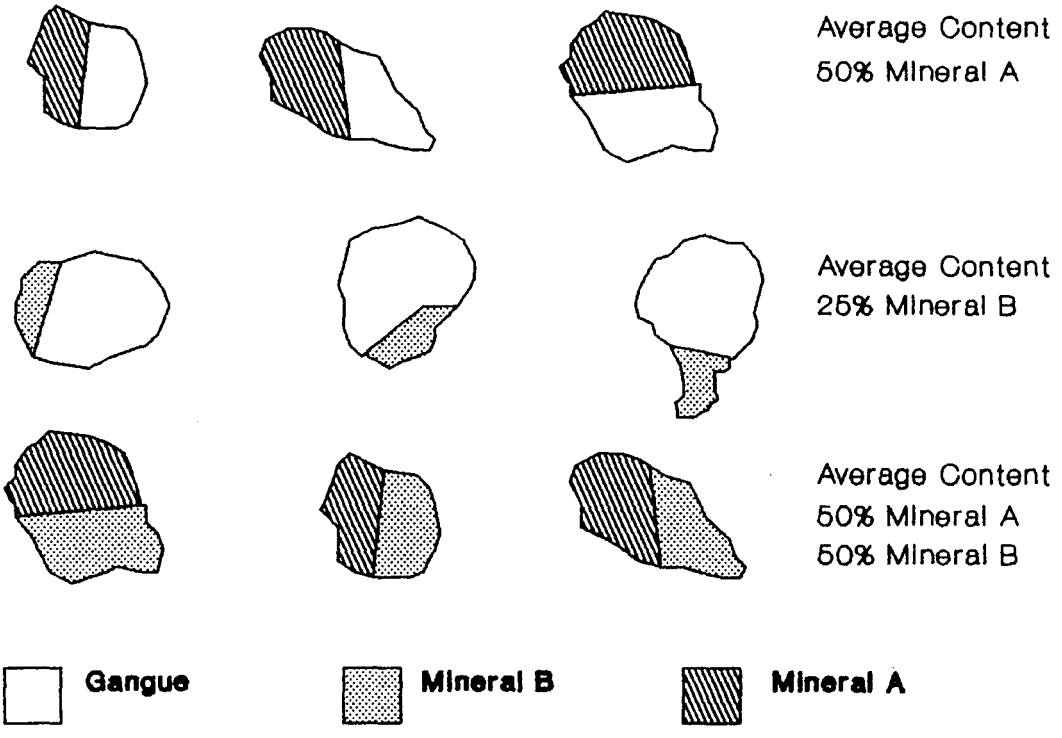
5.5.2 The leach model

Although the leach model is a steady-state model and due to the extensions made to the data structures (described in section 5.5.1 above), insertion of the model into MicroSim was expected to be straight forward. However, a serious flaw in the way in which the MicroSim streams data structures for the ore phase have been designed and implemented was encountered.

Figure 5.4 illustrates a number of possible particle types for a ternary ore. This figure illustrates the various way in which the three minerals making up the ore may be combined for various ore particles. The composition of the ore particle influences the behaviour of the particle in many minerals processing units. Thus a good simulator should be able to describe the composition of the ore particles in sufficient detail.

2 The adsorption model only simulates single solute adsorption and the leaching model only allows one mineral to be leached. However the data structures allow storage of more than one adsorbate or aqueous species.

Figure 5.4 : Ternary Particle System (After King, 1974)



MicroSim uses the method described by King (1974) to store the data describing particle compositions. The concept of G classes is used. Each G class represents a typical or average particle composition. The user is responsible for entering these average compositions. For the example presented in Figure 5.4, the user may choose to define 6 G classes representing the following compositions:

G Class	% Gangue	% Mineral A	% Mineral B
1	100	0	0
2	0	100	0
3	0	0	100
4	50	50	0
5	75	0	25
6	0	50	50

This descriptive data is common to all streams and is stored only once, not for all streams. Only the amount or distribution of material in the G classes in the various streams may change. For example, a flotation concentrate stream may have most of the stream mass distributed in G classes 2,4 and 6 (assuming preferential flotation of mineral A) while the tailings stream may have most of the stream mass distributed in G classes 1,3 and 5.

This is a very efficient method for simulation of ore-dressing processes. It works well because no ore-dressing unit process modifies the composition of particles passing through that unit[9].

If it is imagined that the particles illustrated in Figure 5.4 are subjected to a leaching process with mineral A being preferentially leached, the weakness of the present G class system for use in hydrometallurgical simulation will soon be realised. The process of leaching causes particle compositions (and perhaps particle sizes) to change. There is no guarantee that the particle

9 This is of course not true for comminution units. These units modify both particle size and particle composition. However the state of liberation modelling is such that liberation is mostly ignored when modelling comminution units.

compositions after leaching will be aptly described by one or more of the G class compositions defined by the user of the simulation package. In fact, it would be most unlikely that this would happen.

Any process which modifies the composition of an ore particle (such as leaching, roasting, comminution) cannot in general be modelled using the existing G class system. This is because there is nothing which precludes the particle composition changing to a composition which the user has not defined in the vector of particle compositions.

Further, the particle compositions in various streams in a flowsheet containing unit processes which modify particle compositions, are likely to be very different. In general, the particle compositions may also be size class dependent. Thus using one vector to describe particle compositions in various streams and size classes does not appear to be feasible.

The information which must be contained in a stream data structure is dictated by the needs of the unit models which need to perform transformations on inlet streams in order to predict the composition of output streams (Brit 1980; Ritchie 1983). This statement illustrates the fact that the fixed G class method is sufficient (as well as efficient) for use in an ore-dressing simulator, where particle compositions are not changed by any unit processes (ignoring liberation during comminution), but will fail when processes which result in a change in particle composition are simulated.

Due to the reasons discussed above, the leaching model was implemented using the restricting assumption that only the leaching of perfectly liberated ores can be simulated. This assumption was made because the formulation and implementation of more suitable data structures for storing particle composition in streams was considered beyond the scope of this project.

Assuming perfect liberation, no difficulty concerning changing particle composition is encountered. The model merely calculates the mass of the mineral (for the mineral being leached) in the appropriate particle classes in the tailings stream according to the expressions derived in 5.3.6. The model then calculates the concentration of the appropriate species in solution in order to complete the mass balance.

Although this is not an entirely satisfactory solution, the model should still be useful for many practical simulation studies, especially when used together with the adsorption model. More importantly, a major problem area which needs to be investigated thoroughly before a general simulator can be implemented has been identified.

Figure 5.5 shows a MicroSim report file for a typical CIL simulation. This Figure illustrates the output which the leach and adsorption models provide for the user for a 5 stage leach followed by a 5 stage CIP adsorption plant. The parameters for the simulation are taken from the CIP tutorial in the MicroSim Users Guide (Stange, Cilliers and King 1988a).

Figure 5.5 : MicroSim report file for leach/CIP simulation

```
------------------------------------------------
MICROSIM Report For Unit 1
Unit Of Type : LEACH_VESSELS
Model Used   : LOVEDAY
------------------------------------------------

Average solids density              : 2700 kg/m3
Solids hold up                      :  420 tons
Residence time                      : 1.00 hrs
```

Stage	[Au] (ppm)	Gold (g/t)
Head	0.000	5.000
1	2.124	2.876
2	3.245	1.755
3	3.838	1.162
4	4.150	0.850
5	4.315	0.685

```
------------------------------------------------
NOTES :



------------------------------------------------


------------------------------------------------
MICROSIM Report For Unit 2
Unit Of Type : CIP_ADSORPTION
Model Used   : FLEMING
------------------------------------------------

Number of Stages                              :   5
Tank Size          (m^3)                      : 576.00
Carbon Mass per Elution (Tons)                :  10.00
Elution Cycle = Once per 24.0 hours
Loaded Carbon Value      (g/t)                : 4528
Carbon Inventory         (Tons)               :  72.00
Metal Lock Up            (kg)                 :  96.45

Average profiles are :
```

Stage	Carbon Conc, g/l	[Au] (Soln, ppm)	[Au] (Carbon, g/t)	Gold (Ore, g/t)
1	25.0	1.256	4595	0.598
2	25.0	0.3747	1394	0.551
3	25.0	0.1173	456	0.527
4	25.0	0.0403	172	0.514
5	25.0	0.0163	82	0.508

During phase 1 the following conditions exist :

Effective pulp flowrate through stage 1 : 575.55 m^3/hr
Effective pulp flowrate through stage 2 : 575.55 m^3/hr
Effective pulp flowrate through stage 3 : 575.55 m^3/hr
Effective pulp flowrate through stage 4 : 575.55 m^3/hr
Effective pulp flowrate through stage 5 : 575.55 m^3/hr

No carbon transfer during this phase

During phase 2 the following conditions exist :

Effective pulp flowrate through stage 1 : 575.55 m^3/hr
Effective pulp flowrate through stage 2 : 575.55 m^3/hr
Effective pulp flowrate through stage 3 : 575.55 m^3/hr
Effective pulp flowrate through stage 4 : 575.55 m^3/hr
Effective pulp flowrate through stage 5 : 475.55 m^3/hr

Carbon transfer is as follows :

Transfer from stage 1 to loaded screen at 100.00 m^3/hr
Transfer from stage 2 to stage 1 at 100.00 m^3/hr
Transfer from stage 3 to stage 2 at 100.00 m^3/hr
Transfer from stage 4 to stage 3 at 100.00 m^3/hr
Transfer from stage 5 to stage 4 at 100.00 m^3/hr
Transfer from regen bin to stage 5 at 2.50 TPH

NOTES :

5.6 Simulation Results

In order to illustrate the flexibility of the models developed, this section will illustrate how the models can be used to study the effect of various process operating parameters on the CIL process.

The simulations were carried out using the rate expression due to Nicol, Fleming and Cromberge (1984) as described in section 4.3. That is,

$$R(y,c) = k\left(C - \frac{y}{A}\right)$$

The values used for the parameters were as reported by Nicol, Fleming and Cromberge. That is k = 292.95 hr^{-1} and A = 10850.

Leaching behaviour was modelled using the MINTEK expression described in section 5.3. The parameters used were $k_p = 0.9$, $S_m = 0.13$. These values are for units of concentration in grams per ton and time in hours.

The plant parameters for the base case simulated are:

Ore headgrade	: 1.0 grams/ton
Ore flowrate	: 700 TPH
Ore S.G.	: 2.67
Pulp Density	: 50% solids by mass
Number of adsorption stages	: 6
Stage volume	: 960 m³
	(1 hour residence time)
Carbon concentration	: 20 grams per litre
	(19.2 tons per stage)
Mass of carbon eluted	: 12 tons per day
Loading on eluted carbon	: 50 g/t

Due to the high tonnage (500 000 tons/month) treated together with the low grade of the material, the plant configuration simulated is characteristic of the type of operation used to treat reclaimed dump material.

The simulated method of carbon transfer for the base case was the sequential mode, with regenerated carbon added to tank 6, carbon from tank 6 transferred to tank 5, carbon from tank 5 transferred to tank 4 and so on with the transfer of carbon from tank 1 to elution the last step in the sequence. Each transfer step was simulated to take 2 hours, carbon transfer thus lasting 14 hours every 24 hours.

5.6.1 Effect of leaching

Most adsorption plants treating dump material are of the CIL type where a significant part of the leaching takes place in the adsorption section of the plant. In order to minimize capital expenditure, dump treatment plants may have only one or two leach stages before the pulp enters the adsorption plant. Experimental work has shown that "pushing" the leach into adsorption leads to a decrease in adsorption efficiency (Davidson 1988).

A series of simulations where the number of preleach stages were varied, was carried out. As the number of preleach stages are increased, less leaching takes place in the adsorption section. Figures 5.6 and 5.7 show the carbon loading and solution tenor profiles for various numbers of preleach stages. These profiles represent average values (averaged over 1 adsorption cycle, i.e. 24 hours) at pseudo-steady state. Figures 5.6 and 5.7 show results obtained for carbon concentrations of 20 g/l in all stages.

Nicol, Fleming and Cromberge (1984) use their model to show that CIP circuits perform optimally for a given total carbon inventory when the carbon concentration in each stage is the same (as in Figures 5.6 and 5.7). Figures 5.8 to 5.11 illustrate the effect that the number of preleach stages has on adsorption performance when the carbon concentrations are not evenly distributed. The carbon concentrations for the case shown in Figures 5.8 and 5.9 are 25 g/l in stages 1 and 2, 20 g/l in stages 3 and 4 and 15 g/l in stages 5 and 6. The total carbon inventory is thus the same as for the base case.

Figures 5.10 and 5.11 show the results when the carbon concentrations are 15 g/l in stages 1 and 2, 20 g/l in stages 3 and 4 and 25 g/l in stages 5 and 6.

It can be seen that in all cases, as more preleach stages are added, the carbon loading and solution tenor profiles become steeper, indicative of increased adsorption efficiency.

Table 1 shows the values obtained for the average carbon and solution profiles as well as gold lock up for each run. Note that carbon loadings are reported in grams of gold per ton of carbon, while the solution tenor is in grams of gold per ton of solution.

The lock up is the average amount of gold on the carbon in the adsorption plant. Gold lock up may be treated as an operating cost (Menne 1982b), with the cost considered to be proportional to the amount of gold locked up.

Figure 5.6 : Effect Of Preleach Stages On Loading Profile (20 g/l Carbon Conc)

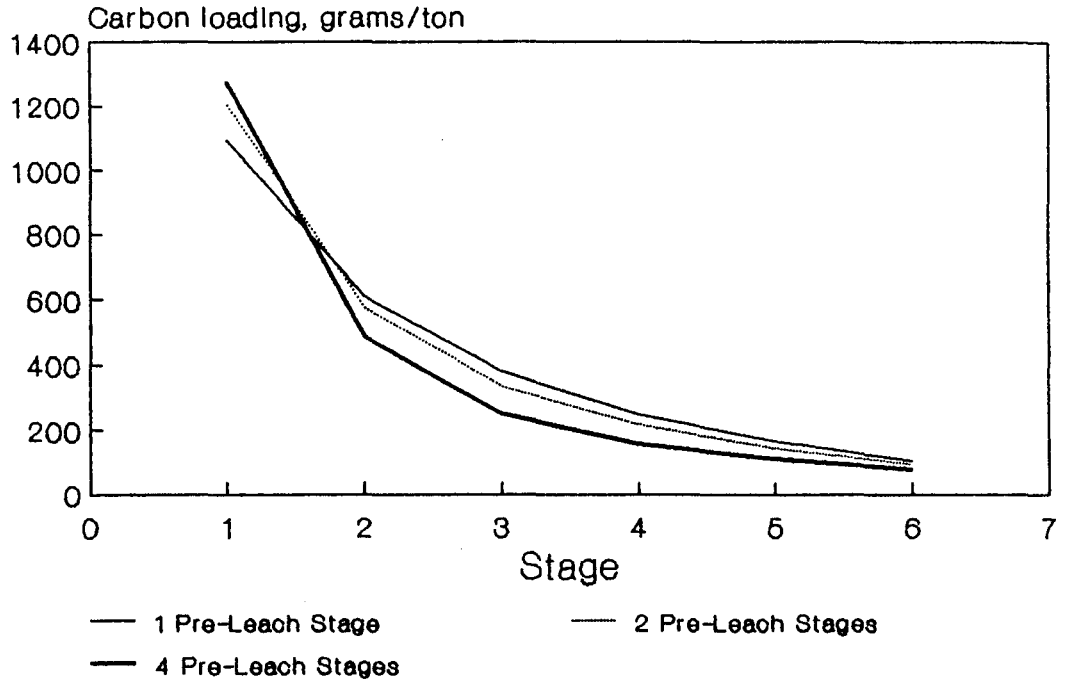


Figure 5.7 : Effect Of Preleach Stages On Solution Profile (20 g/l Carbon Conc)

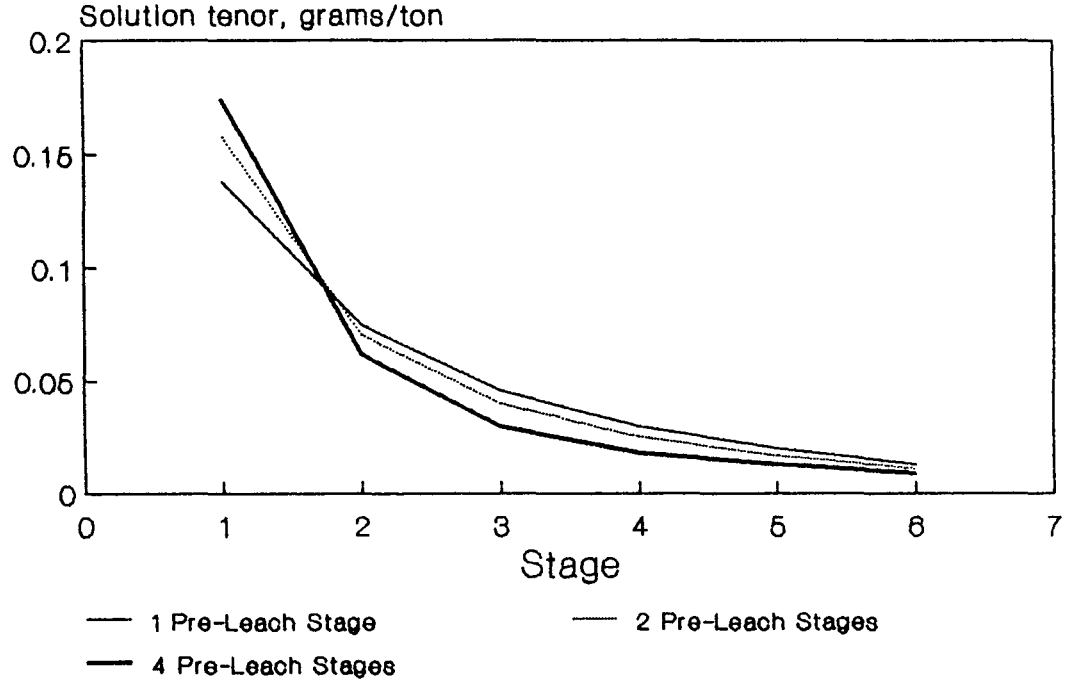


Figure 5.8 : Effect Of Preleach Stages On Carbon Profile (25,20,15 g/l)

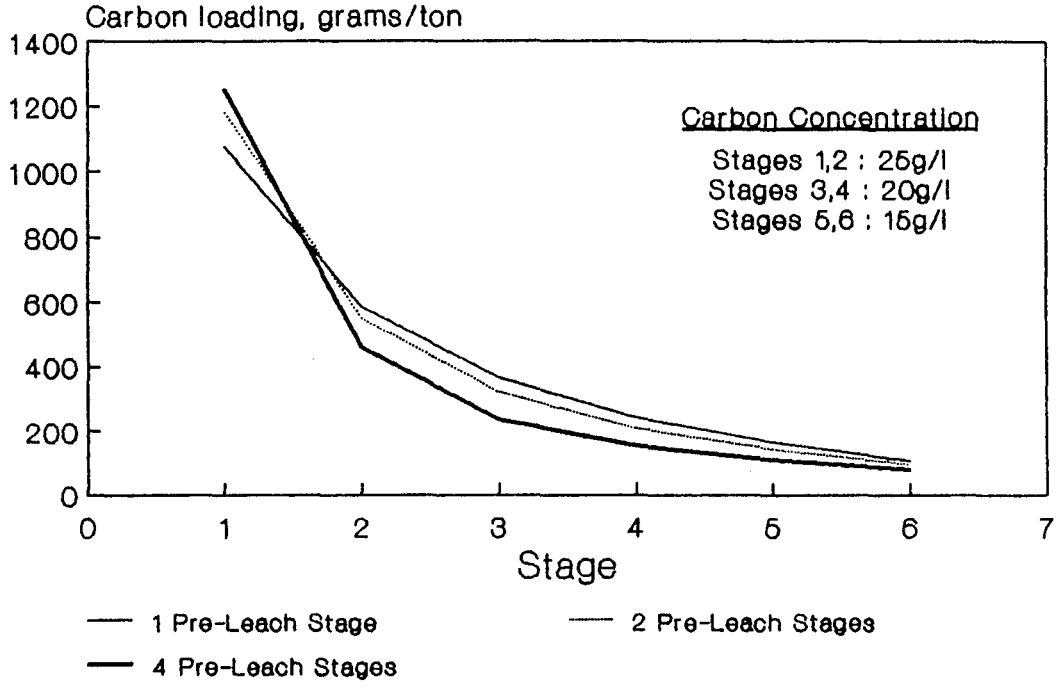


Figure 5.9 : Effect Of Preleach Stages On Solution Profile (25,20,15 g/l)

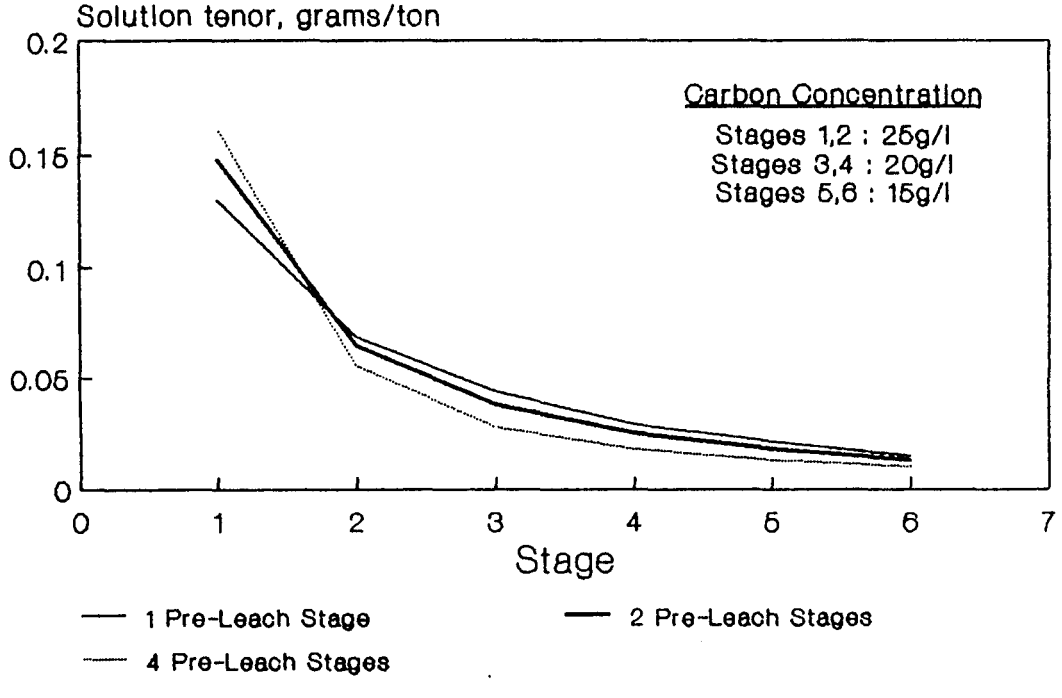


Figure 5.10 : Effect Of Preleach Stages On Carbon Profile (15,20,25 g/l)

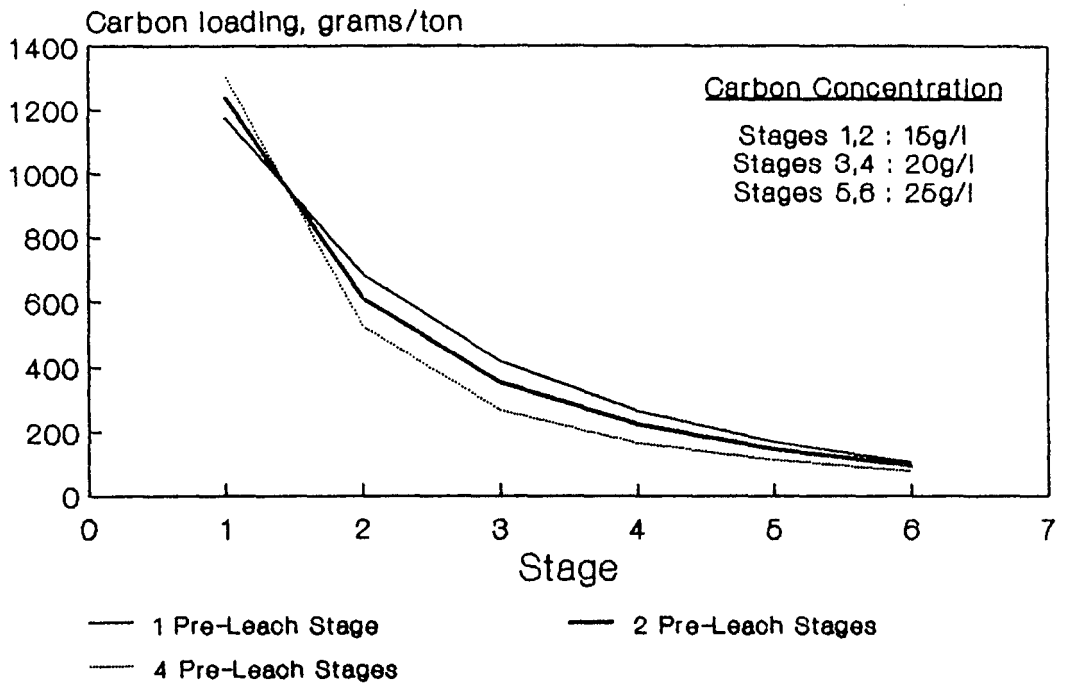


Figure 5.11 : Effect Of Preleach Stages On Solution Profile (15,20,25 g/l)

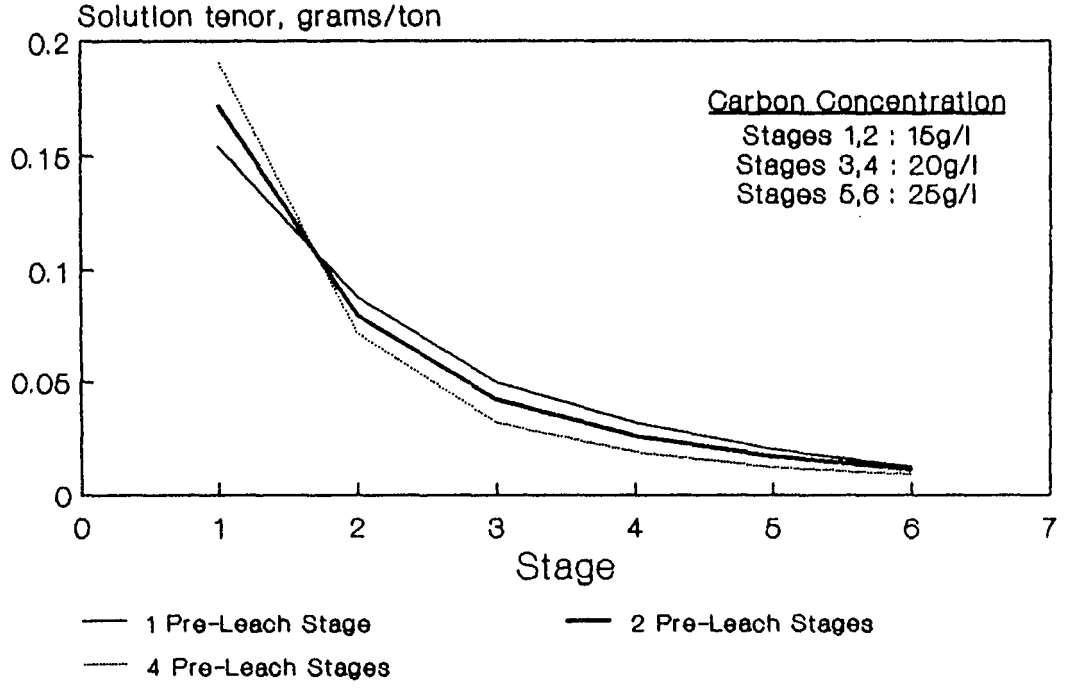


Table 1 : Effect of leaching on adsorption performance for various carbon concentration profiles

20 g/l in all stages						
	1 Leach Stage (Lock Up = 52.4 kg)					
Carbon	1092	610	379	248	164	103
Solution	.138	.075	.046	.030	.020	.013
	2 Leach Stages (Lock Up = 51.6 kg)					
Carbon	1207	575	335	216	143	93
Solution	.158	.071	.040	.025	.017	.011
	3 Leach Stages (Lock Up = 49.0 kg)					
Carbon	1248	521	284	182	123	84
Solution	.168	.065	.034	.021	.014	.010
	4 Leach Stages (Lock Up = 47.4 kg)					
Carbon	1274	486	250	158	110	78
Solution	.174	.062	.030	.018	.013	.009

25g/l in stages 1 and 2; 20 g/l in stages 3 and 4; 15 g/l in 5 and 6						
	1 Leach Stage (Lock Up = 57.9 kg)					
Carbon	1075	586	367	243	164	106
Solution	.130	.069	.044	.029	.021	.015
	2 Leach Stages (Lock Up = 57.6 kg)					
Carbon	1185	550	323	210	142	95
Solution	.148	.065	.038	.025	.018	.013
	3 Leach Stages (Lock Up = 55.1 kg)					
Carbon	1227	495	271	176	122	86
Solution	.157	.059	.032	.020	.015	.011
	4 Leach Stages (Lock Up = 53.4 kg)					
Carbon	1251	459	238	153	109	79
Solution	.162	.056	.028	.018	.013	.010

15g/l in stages 1 and 2; 20 g/l in stages 3 and 4; 25 g/l in 5 and 6						
	1 Leach Stage (Lock Up = 49.6 kg)					
Carbon	1174	687	420	265	168	103
Solution	.154	.088	.050	.032	.020	.012
	2 Leach Stages (Lock Up = 46.4 kg)					
Carbon	1238	612	353	222	145	92
Solution	.172	.080	.042	.026	.017	.011
	3 Leach Stages (Lock Up = 43.7 kg)					
Carbon	1277	559	301	188	125	84
Solution	.184	.075	.036	.022	.014	.010
	4 Leach Stages (Lock Up = 42.0 kg)					
Carbon	1306	526	268	164	112	78
Solution	.192	.072	.032	.019	.012	.009

It can be seen that the first scheme, with the carbon inventory distributed at 20 g/l in all stages, provides better performance than the second scheme (25 g/l in stages 1 and 2, 20 g/l in stages 3 and 4 and 15 g/l in stages 5 and 6). The solution tenor in the tails as well as the gold lock up for scheme 1 is lower than for scheme 2. This is in agreement with the findings of Nicol, Fleming and Cromberge (1984).

Comparing the results for schemes one and three (15 g/l in stages 1 and 2, 20 g/l in stages 3 and 4 and 25 g/l in stages 5 and 6) it can be seen that the third scheme provides better performance than the first. For the 1 preleach stage case, the solution tails as well as the lock up is lower for scheme 3 than for scheme 1. For the other cases (2, 3 and 4 leach stages) the solutions tails for schemes 1 and 3 are equivalent, but the lock ups for scheme 3 are lower than for scheme 1.

It can thus be concluded that the less leaching which takes place in an adsorption circuit, the lower the solution tails in the exit of the circuit will be. In addition, if leaching in the adsorption circuit does take place, an evenly distributed carbon concentration profile is not necessarily best.

5.6.2 Effect of eluted loadings

A number of early CIP plants encountered problems with the elution unit process. Various operational problems resulted in high gold loadings on the eluted carbon. The eluted carbon serves as the counter-current carbon feed stream in a CIP plant.

A number of simulations considering the effect of the eluted carbon loading on adsorption performance were performed. All simulations were run with 2 preleach stages. As in section 5.6.1 three schemes of carbon concentration distribution was used. The first scheme has 20g/l of carbon in each stage, scheme 2 has 25 g/l in stages 1 and 2, 20 g/l in stages 3 and 4 and 15 g/l in stages 5 and 6. Scheme 3 has 15 g/l in stages 1 and 2, 20 g/l in stages 3 and 4 and 25 g/l in stages 5 and 6.

The average simulated solution and carbon profiles for various eluted loadings are shown in Figures 5.12 to 5.17. This data and the lock up are also reported in Table 2.

Table 2 : Effect of eluted loadings on adsorption performance for various carbon concentration profiles

20 g/l in all stages						
	Eluted Loading 10g/t (Lock Up 44.9kg)					
Carbon	1140	502	266	164	98	51
Solution	.152	.064	.033	.020	.013	.008
	Eluted Loading 100g/t (Lock Up 54.6kg)					
Carbon	1200	577	353	248	185	141
Solution	.157	.071	.041	.028	.021	.016
	Eluted Loading 200g/t (Lock Up 65.7kg)					
Carbon	1275	664	446	344	284	240
Solution	.163	.079	.050	.037	.030	.025
	Eluted Loading 300g/t (Lock Up 76.9kg)					
Carbon	1352	753	539	440	382	340
Solution	.170	.087	.058	.046	.039	.034

	25g/l in stages 1 and 2; 20 g/l in stages 3 and 4; 15 g/l in 5 and 6					
	Eluted Loading 10g/t (Lock Up 53.0kg)					
Carbon	1162	511	279	167	100	54
Solution	.146	.061	.034	.021	.014	.009
	Eluted Loading 100g/t (Lock Up 59.7kg)					
Carbon	1173	546	339	243	185	143
Solution	.147	.064	.039	.028	.022	.017
	Eluted Loading 200g/t (Lock Up 70.7kg)					
Carbon	1246	635	434	341	285	243
Solution	.153	.072	.048	.036	.031	.026
	Eluted Loading 300g/t (Lock Up 81.7kg)					
Carbon	1323	723	528	437	383	342
Solution	.159	.080	.057	.045	.040	.035

	15g/l in stages 1 and 2; 20 g/l in stages 3 and 4; 25 g/l in 5 and 6					
	Eluted Loading 10g/t (Lock Up 41.3kg)					
Carbon	1200	569	309	179	103	52
Solution	.169	.076	.038	.022	.013	.007
	Eluted Loading 100g/t (Lock Up 49.2 kg)					
Carbon	1229	613	367	251	184	139
Solution	.171	.080	.043	.029	.020	.015
	Eluted Loading 200g/t (Lock Up 60.7kg)					
Carbon	1307	704	462	348	283	239
Solution	.178	.088	.052	.038	.029	.024
	Eluted Loading 300g/t (Lock Up 72.0kg)					
Carbon	1385	794	556	444	381	338
Solution	.184	.096	.061	.046	.038	.033

It can be seen from the graphs and Table 2 that the loading of gold on the eluted carbon has a strong effect on adsorption performance. As the gold loading on the eluted carbon rises, both the tenor of the tails solution and the gold lock up rises sharply.

Figure 5.12 : Effect Of Eluted Loading On Carbon Profile (Carbon Conc 20g/l)

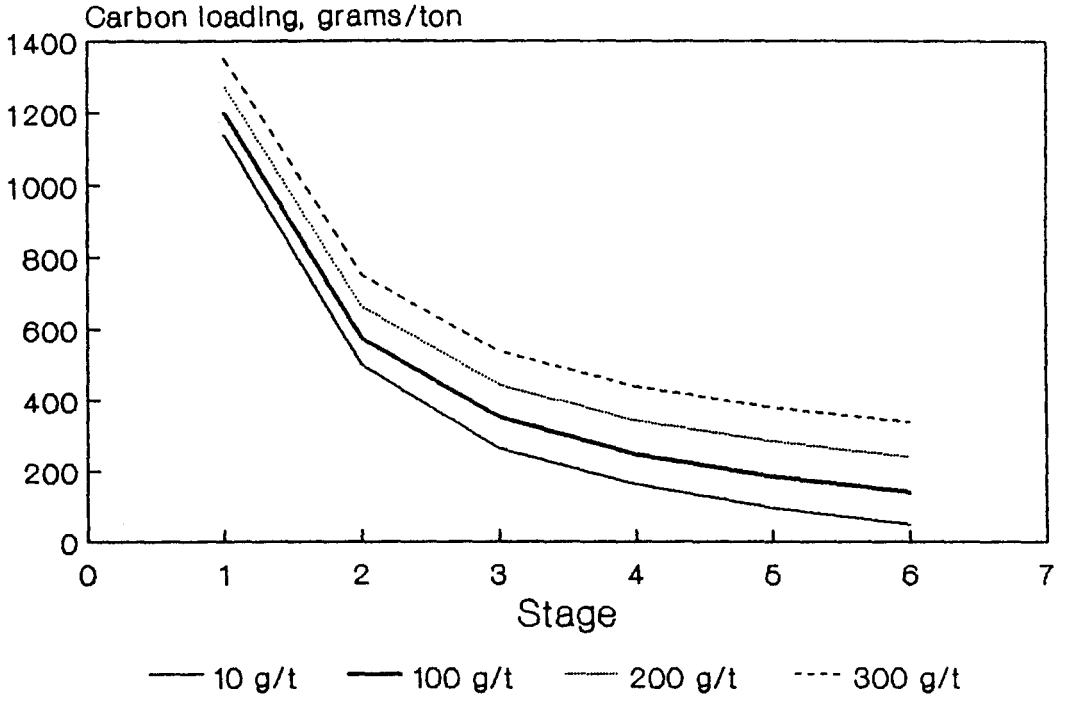


Figure 5.13 : Effect Of Eluted Loading On Solution Profile (Carbon Conc 20g/l)

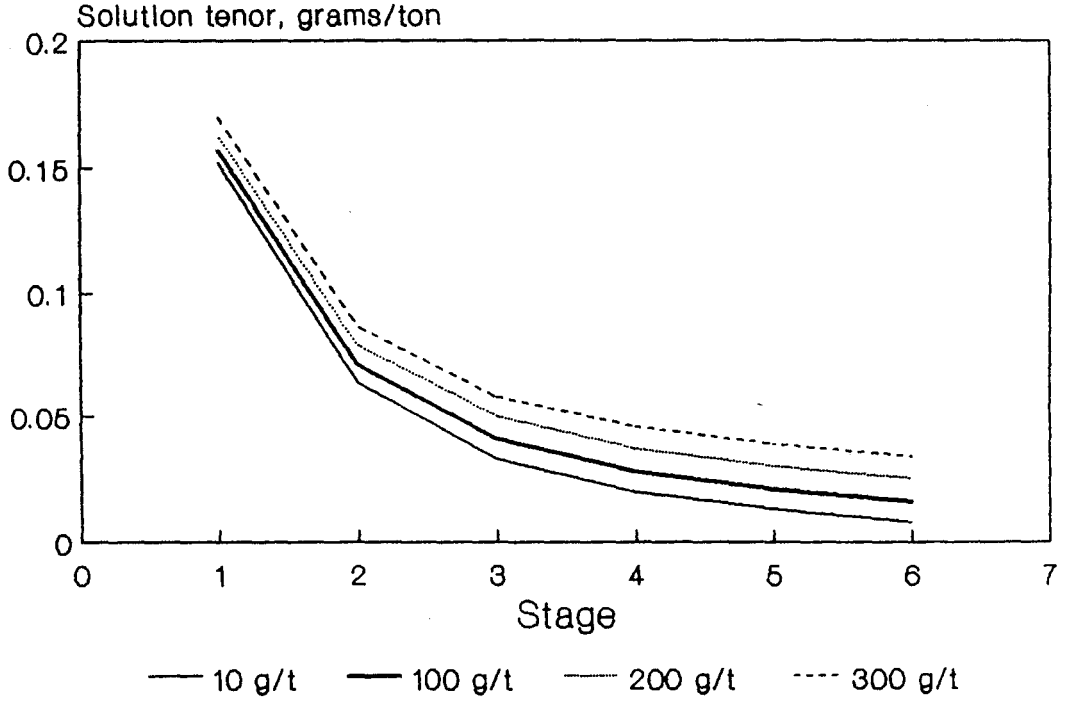


Figure 5.14 : Effect Of Eluted Loadings On Carbon Profile (25,20,15 g/l)

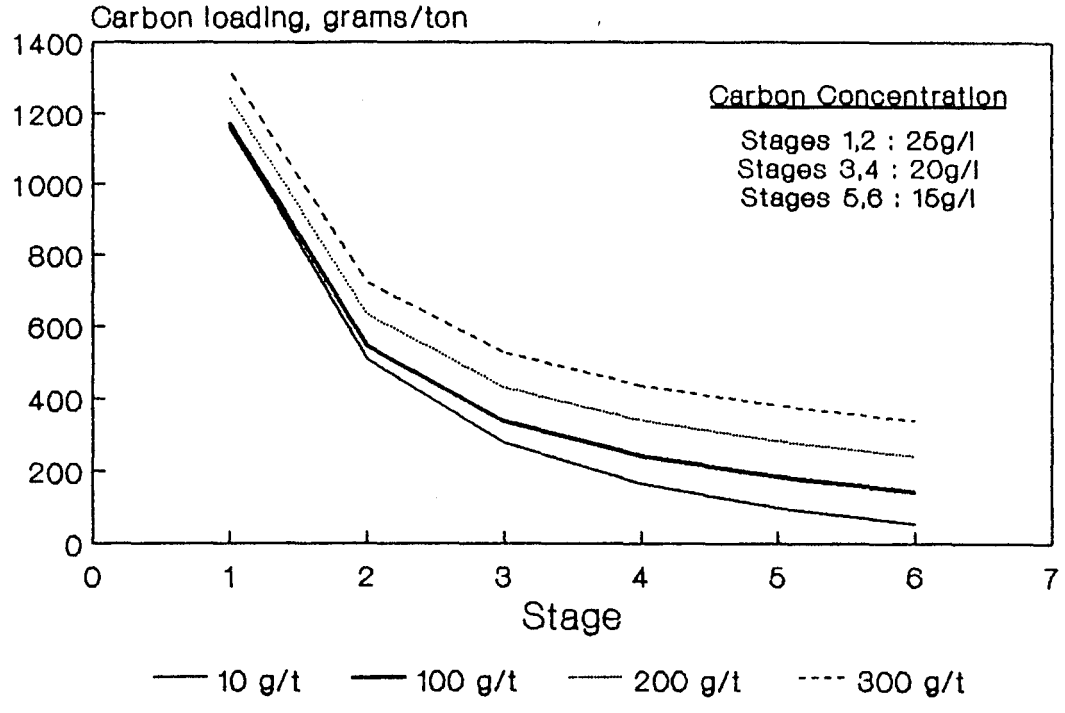


Figure 5.15 : Effect Of Eluted Loadings On Solution Profile (25,20,15 g/l)

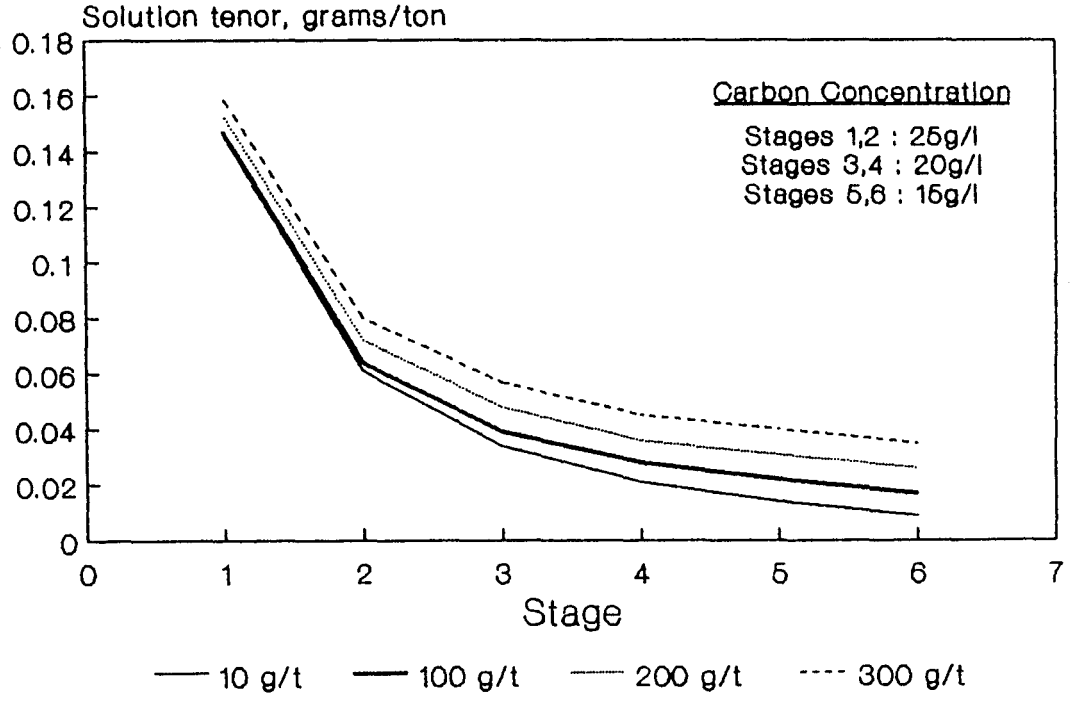


Figure 5.16 : Effect Of Eluted Loadings On Carbon Profile (15,20,25 g/l)

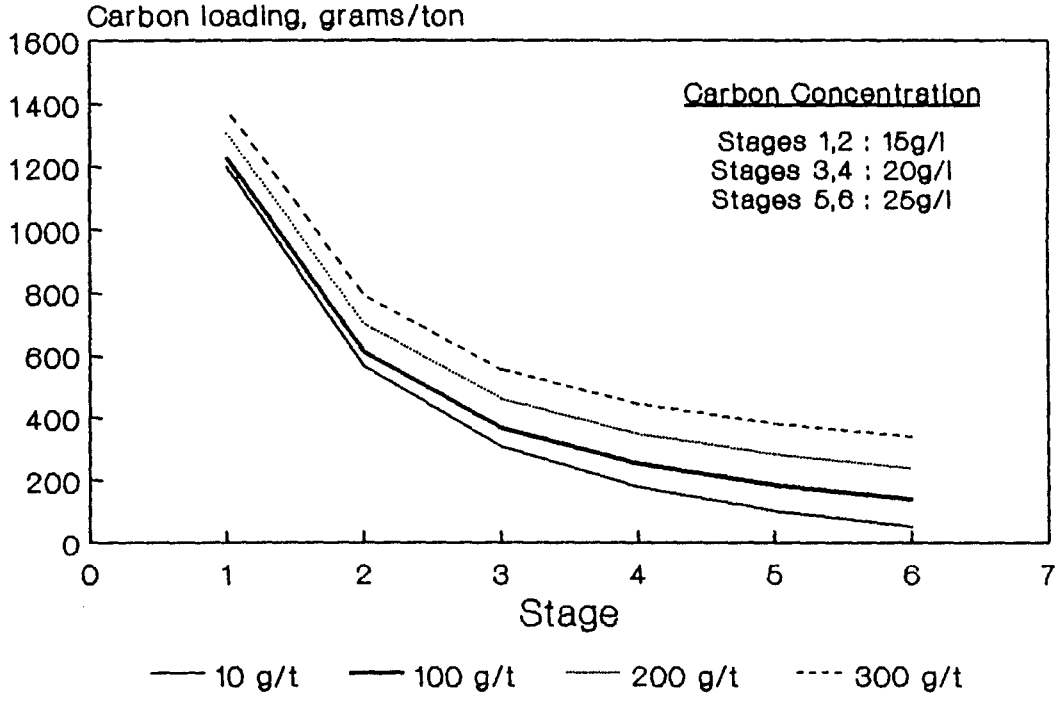
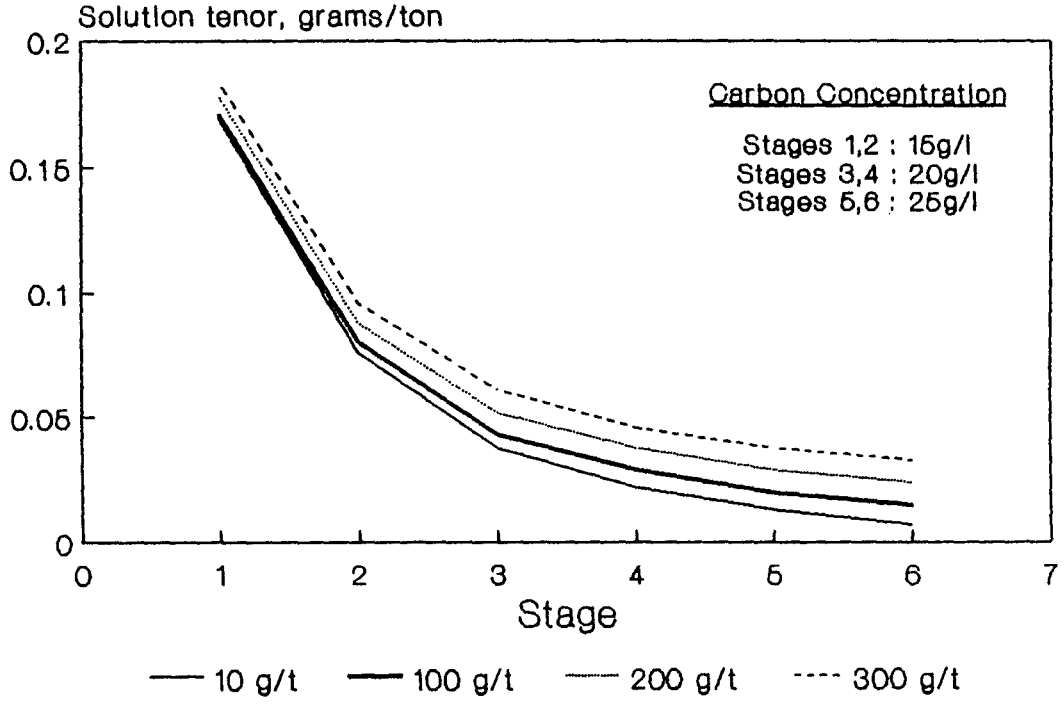


Figure 5.17 : Effect Of Eluted Loadings On Solution Profile (15,20,25 g/l)



Again, the scheme 1 distribution is superior to the scheme 2 carbon distribution. The scheme 3 distribution is significantly better than the scheme 1 distribution with respect to gold lock up and slightly better with respect to the gold tenor in the tailings stream.

5.6.3 Effect of carbon transfer method

As pointed out in section 4.3 only one group of researchers (excluding this work) have developed a model which includes the effect of carbon transfer. The inclusion of the effects of carbon transfer complicate the model significantly. It is thus necessary to establish whether this added complexity results in a better model.

In order to answer this question, a number of different transfer methods were simulated to determine whether the method of carbon transfer has a significant effect on circuit performance.

All simulations were carried out using 2 stages of preleaching and carbon concentrations of 20 g/l in each stage. Four transfer methods were simulated. They are:

Method A

For this method, carbon transfer is assumed to take place instantaneously. This assumption is made in most adsorption models (see sections 4.3 and 4.4). As 12 tons of carbon is eluted per day, the percentage of the carbon per stage transferred is 12/19.2*100 = 62.5.

Methods B and C

For these methods sequential transfer was assumed. The sequence used for each method and the base case are summarised in table 3.

Table 3 : Sequential transfer methods simulated

Phase	Base	B	C
0	No Transfer	No Transfer	No Transfer
1	Regen -> 6	1 -> Elution	6 -> 5
2	6 -> 5	2 -> 1	Regen -> 6
3	5 -> 4	3 -> 2	5 -> 4
4	4 -> 3	4 -> 3	4 -> 3
5	3 -> 2	5 -> 4	3 -> 2
6	2 -> 1	6 -> 5	2 -> 1
7	1 -> Elution	Regen -> 6	1 -> Elution

The average carbon and solution profiles for these transfer schemes are shown in Figures 5.18 and 5.19. The data are tabulated in table 4.

Table 4 : Effect of transfer scheme on adsorption performance

	Base Transfer Method (Lock Up 51.6kg)					
Carbon	1207	575	335	216	143	93
Solution	.158	.071	.040	.025	.017	.011
	Transfer Method A (Lock Up 35.7kg)					
Carbon	861	403	245	165	111	72
Solution	.138	.054	.031	.020	.014	.010
	Transfer Method B (Lock Up 33.7kg)					
Carbon	888	420	254	170	115	74
Solution	.139	.059	.033	.022	.015	.010
	Transfer Method C (Lock Up 48.6kg)					
Carbon	1159	532	305	200	133	75
Solution	.154	.067	.037	.024	.016	.010

It can be seen that although the solution tenor in the tails stream is similar for all transfer methods, a significant difference in the shape of the profiles and in the lock up exists for the various methods.

Figure 5.18 : Effect Of Transfer Scheme On Carbon Profile

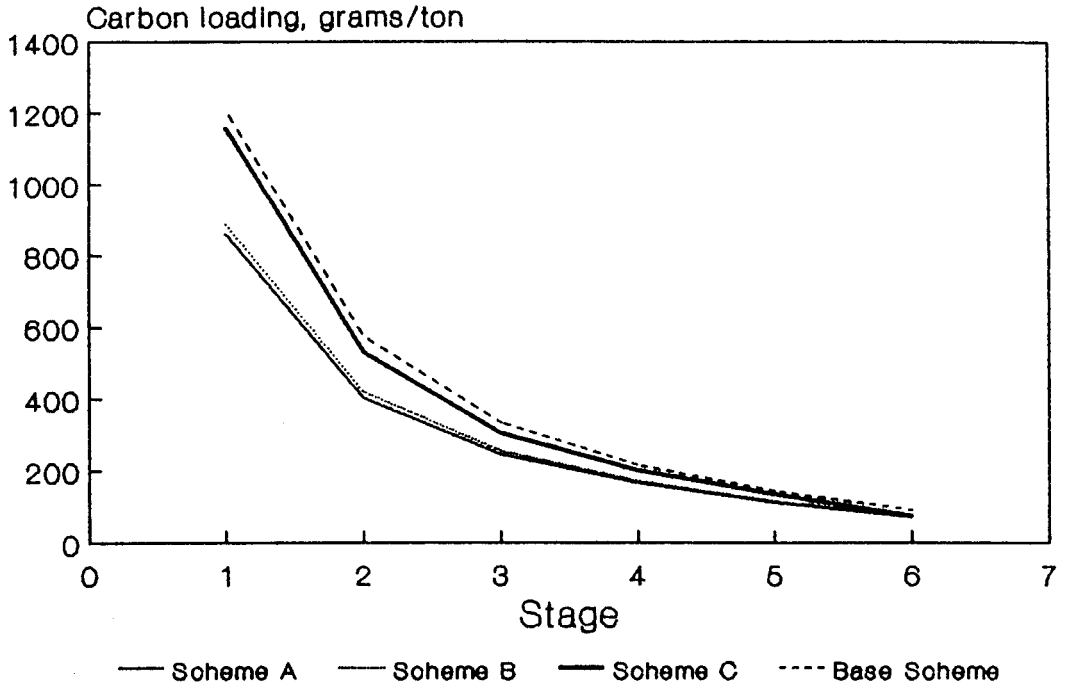
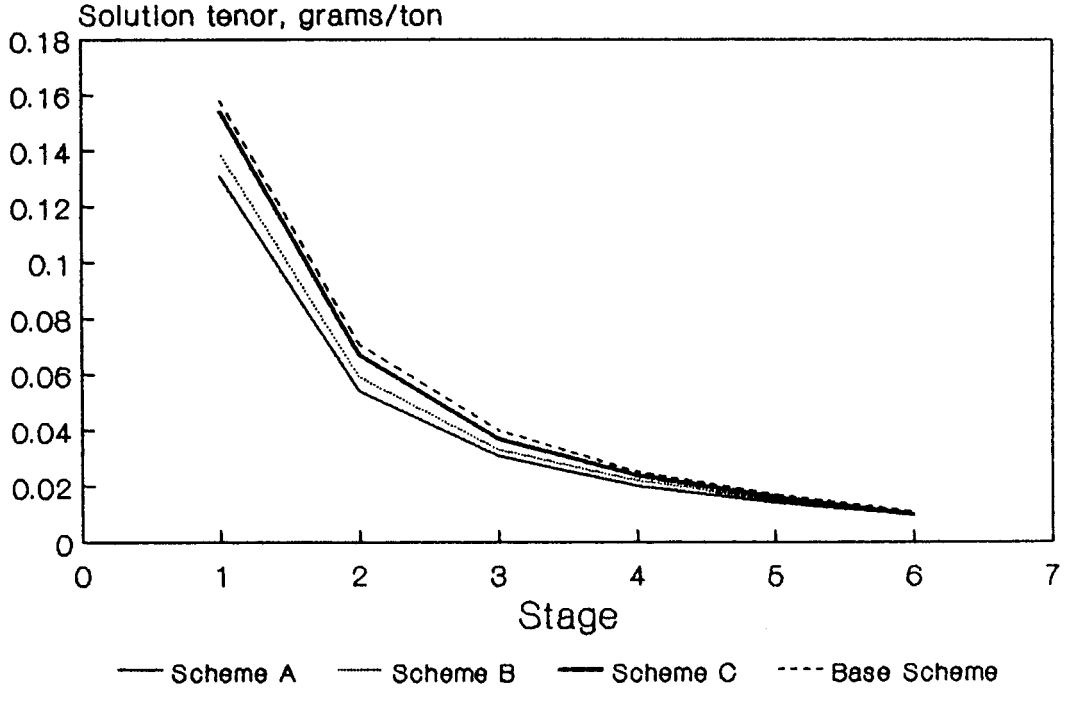


Figure 5.19 : Effect Of Transfer Scheme On Solution Profile



It can thus be concluded that in general, the simple "instantaneous transfer" method will not provide accurate predictions of full scale plant behaviour.

5.6.4 The effect of carbon leakage

The development of efficient inter-stage screens has played an important role in the overall improvement of CIP plant performance. However, older CIP plants may have screens installed which leak and/or overflow easily. Thus a number of simulations were carried out to investigate the effect of carbon leakage (or co-current carbon flow) on adsorption efficiency.

All simulations were performed using 2 preleach stages, 20 g/l of carbon per stage and the base case sequential transfer method. The effect of carbon leakage was simulated by assuming various carbon concentrations in the inter-stage pulp. Levels of 0.05, 0.1 and 0.25 g/l were used. It was assumed that all carbon that leaks is recovered and recycled back into the adsorption system. Thus any difference in adsorption performance is due to the co-current flow of carbon and not because of a depletion of the mass of carbon in the system.

The variations in the average carbon loading profile for these conditions is shown in Figure 5.20. As the effects on the solution profile were small, the solution tenors were not plotted. The data are summarised in Table 5.

Figure 5.20 : Effect of Screen Leakage On Carbon Profile

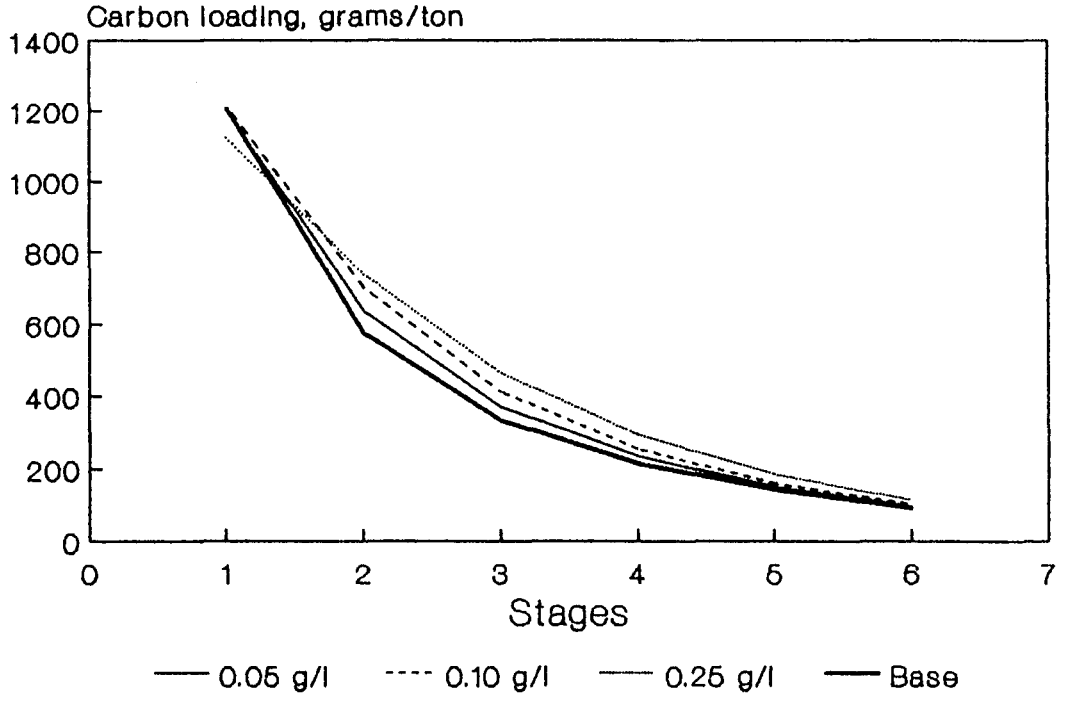


Table 5 : Effect of carbon leakage on adsorption performance

	Base Case (Lock Up 51.6 kg)					
Carbon	1207	575	335	216	143	93
Solution	.158	.071	.040	.025	.017	.011
	Leakage of 0.05g/l (Lock Up 55.5 kg)					
Carbon	1201	637	371	234	152	97
Solution	.156	.075	.043	.027	.018	.012
	Leakage of 0.10g/l (Lock Up 55.8kg)					
Carbon	1213	702	413	256	162	101
Solution	.163	.081	.047	.029	.019	.012
	Leakage of 0.25g/l (Lock Up 59.0kg)					
Carbon	1123	739	466	294	185	113
Solution	.155	.083	.051	.033	.021	.013

It can be seen that leakage (for this particular situation) has fairly small deleterious effects on both lock up and the tenor of the tails solution. However, the effect is particular to the plant configuration simulated. Other work (Whyte et al 1987) has shown that carbon leakage may lead to a 50% increase in the gold lock up and the tenor of the tails solution. The model presented in this chapter allows these effects to be quantified for the particular situation.

5.6.5 Computational aspects

The simulations were all run on an IBM PC compatible microcomputer (approximately 1.7 times the speed of a standard IBM PC) without an 8087 numeric coprocessor. Times required for each simulation run ranged from 3 to 15 minutes. The adsorption model requires much more computation effort than the leaching model. For each simulation run, the adsorption model was set to iterate until pseudo steady-state was achieved. The tolerance for convergence was set at 0.5 percent. That is, the relative change in each quantity from one cycle to the next had to be 0.5 percent or less before convergence was achieved.

Computation time is influenced by the initial conditions chosen for the computation. If guesses far from the pseudo steady-state values are chosen, longer computation times, of the order of 15 minutes, are required. This may take place for the first run performed. However, if the final values of the last calculation are used as initial values for the next run, convergence is quicker leading to computation times of 3 to 10 minutes.

Although the calculation times are much longer than for the ore-dressing models used in MicroSim, they are not long enough to make use of the model impractical, particularly when the detail which the model includes is considered. If the length of the computation time is a problem, significant reduction in time can be achieved by using a faster computer such as the AT or PS/2 with a numeric coprocessor.

5.7 Conclusions

This chapter has shown that using the macroscopic population balance equation as the basis for a CIP adsorption model allows complex effects such as carbon transfer, carbon leakage and leaching which occur on full-scale plants to be modelled realistically. A number of simple but adequate models for gold leaching were also presented.

Applying the method of moments reduces the amount of numeric work required to solve the model equations by reducing the number of differential equations to be solved. This has a disadvantage in that only information concerning the values of average loadings rather than the complete distribution of loadings is available from the calculations. However, for most practical adsorption simulation exercises a knowledge of these average quantities should suffice. It is felt that the information lost using the moments approach is more than compensated for by the considerably reduced computation effort required to solve the reduced set of equations.

The adsorption model was integrated into the MicroSim simulator by the design and implementation of a carbon stream data structure and by extending the data structures used to store aqueous phase data.

Some interesting problems were encountered when integrating the leaching model into MicroSim. These problems were not solved generally, as they were considered beyond the scope of this work. These problems arise due to the manner in which MicroSim stores the data describing particle composition in process streams. A general solution to this problem is required before MicroSim can evolve into a general simulator, capable of simulating all unit processes used in extractive metallurgy.

Finally, a number of simulations were carried out using the models developed. The effects of leaching, eluted loadings, method of carbon transfer and carbon leakage for a particular set of parameters was illustrated. The model should be useful for both the design of new plants and the optimization of existing plants.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Summary and Conclusions

Much literature has been published which illustrates the power of computer simulation techniques applied to minerals processing systems. Despite the potential benefits which this technology offers, simulation is not used widely in industry at present. This can be ascribed to the fact that there are relatively few engineers who are experts in simulation and modelling technology. The availability of inexpensive computing hardware implies that it is feasible to provide this technology in the form of computer software which can be used by engineers having no specialised skills.

In order for this approach to be successful, the software must meet certain prerequisites. The most important criterion is that the software be easy to use. If this is not the case, engineers will continue to use alternative methods. Secondly, the software must be extendable so that it can be modified to meet the specialized needs of individual users. In addition, the software must be capable of simulating most of the common extraction processes. Of the programs available for simulation of minerals processing systems, the majority can simulate only ore-dressing and/or coal preparation flowsheets. This is far from ideal as hydrometallurgical (and pyrometallurgical) unit processes form a very important part of many flowsheets, as in the gold mining industry.

This project demonstrates that a user-friendly interface for a steady-state simulator can be designed by carefully consideration of the needs and expectations of the users of such a program. Using the sequential-modular executive developed by Cilliers (1987) as a basis, a microcomputer based simulator, MicroSim, was designed and implemented on IBM PC compatible hardware. Reports received from users of the program indicate that MicroSim

has met the specifications set for the system. These specifications are that the simulator can be used effectively by engineers having minimal knowledge of computers or simulation and that the simulator may be easily extended by users who wish to do so.

The feasibility of extending the simulator to cope with hydrometallurgical processes was also investigated. This was accomplished by the derivation of models for the carbon-in-pulp adsorption process and the gold leaching process. These models were then integrated into the MicroSim simulator. A more detailed summary of the work performed follows.

6.1.1 The MicroSim interface

The interface was developed on two levels by careful consideration of the user requirements. The first level of the interface can be termed the external interface and is that part of the simulator which communicates with a user who is using the program in the normal manner. The second level is termed the internal interface and is that part of the simulator which a user modifying and extending the simulator would be exposed to.

The external interface

The specification for this level of the interface is that any user, irrespective of his level of computer literacy or the extent of his knowledge of simulation, be able to set up and carry out a realistic simulation exercise. This includes the understanding and evaluation of the results of the simulation.

In order to provide means of accomplishing the above, a user model was derived for a typical user of such a simulation program. Together with other interface design techniques, this user model was used to design the MicroSim interface. Four areas important to the successful design of the interface became apparent.

Flowsheet Specification : An interactive graphics flowsheet editor was designed to allow the user to describe the flowsheet to be simulated in the most natural way possible. The editor was designed to be totally

user-driven, with all functions accessible simultaneously. This provides the flexibility required when drawing or modifying flowsheets. In addition, rigorous error checking is implemented making it difficult for the user to create an incomplete or inconsistent flowsheet. This helps make the simulator robust.

Data Entry and Editing : As much numeric data needs to be entered to define a simulation problem, considerable attention was paid to efficient means of data entry. In addition, the interface was designed so that most of the data defining the simulation can be edited without leaving the program. An ideal way of implementing data editing and entry was found in the use of form-filling with default parameter values displayed in the form. This reduces confusion, allows data to be entered quickly and accurately, and is extremely interactive.

Data Output : In this area the aim was to provide output which requires minimal additional processing in order to satisfy the needs of the user. Three methods of output were implemented. These are:

1) A tabulation of the mass balance expressed in terms of mass flowrate, grade and recovery as well as tabulations of size distributions.

2) Plots of size distributions and partition curves with a choice of four commonly used axes.

3) Report files for the output of design oriented information.

All tables and graphs may be previewed on the screen before being sent to a hard-copy device.

Integrated System : The simulator was implemented as a totally integrated system. All tasks which a user may need to perform during the simulation run may be carried out without leaving the program, leading to high productivity. This includes means of carrying out operations such as copying, erasing or renaming files without exiting the program to make use of operating system software.

The internal interface

The internal interface needs to be structured so that the simulator may be extended without the need for understanding how the whole simulator is put together. This was achieved by:

Technical Documentation : A technical manual, explaining how to add new models and unit icons as well as how to extend the streams data structures was written for MicroSim. Extensive use of simple illustrative examples are included in the manual.

Structured Programming Techniques : These techniques were used to make the internal structure of MicroSim as modular as possible, with little or no interaction between the different modules. This facilitated maintenance and development of the code.

Source Code : All source code was written with the intention of making it as readable as possible. Comment statements were used where additional explanation was neccessary.

6.1.2 Models for CIP and gold leaching

A model based on the macroscopic population balance equation was developed for the adsorption section of a CIP plant. This is a general model, and is able to accommodate any adsorption rate expression. The technique includes the effects of the distributed properties of the carbon (loading and particle size) on adsorption performance.

The model described above consists of a system of PDEs. The method of characteristics was used to transform this system of PDEs into an equivalent system of ODEs. These ODEs were then easily solved using a variable step length Runge-Kutta type integration algorithm.

The model described above is powerful in that it provides a detailed description of the loading distributions present in each CIP contactor and can be used with any adsorption rate expression. However due to the complexity of the model, the model is computationally intensive, making it unsuitable for use in a microcomputer based simulator.

For a particular class of rate expressions (those linear in the loading term) it was shown that a simulation model with much more flexibility than existing models could be developed, based on the general adsorption model. This was done by applying the method of moments. This model is suitable for the simulation of full scale CIP plants. The model can be solved in reasonable times on a microcomputer, making it suitable for integration into MicroSim.

Gold leaching models were developed by fitting a number of rate expressions found in the literature to a number of sets of batch leaching data. The mathematics for the continuous leaching situation were developed (where neccessary) for the rate expressions which fitted experimental data well. These models were then integrated into MicroSim.

The integration of the adsorption model into MicroSim was relatively simple. The MicroSim streams data structures proved easy to extend, making it easy to add carbon and more detailed lixiviant streams to the simulator. When implementing the leach models in MicroSim it became obvious that serious flaws exist in the method used by MicroSim for storing the properties of an ore in each stream. This will be discussed in more detail in the next section.

6.2 Recommendations for Future Work

MicroSim stores the mineral composition of ore particles in each stream using the G-class method. In this method, a single vector is used to store a number of typical particle compositions, or G classes. The distribution of particles amongst these classes may change from stream to stream. However the particle compositions represented by the vector are fixed.

This scheme works well for ore-dressing simulation as ore-dressing unit processes do not modify particle compositions (discounting liberation during comminution), but merely transform the distribution of particle types in the feed to appropriate distributions in the product streams. In general, the fixed G-class scheme cannot be used for the simulation of unit processes which modify particle composition. This is because the particles may be modified to have compositions not provided for in the G-class descriptor vector.

Unit processes which modify particle types include leaching, comminution, and pyrometallurgical units such as roasters. If a general purpose simulator capable of simulating all such unit processes is to be developed, it is important that this problem be solved in general terms. This is considered the single largest obstacle hindering the development of a general purpose process simulator. Although a general solution to this problem was considered to be beyond the scope of this project, the elucidation of this problem is regarded as an important finding of this project. It is thus important to direct research effort to this area if progress towards the development of a general purpose minerals processing simulator is to be made.

Another important area which was encountered during this work is the relative inflexibility of sequential-modular simulators. Such simulators are incapable of solving the design problem, where the value of design variables must be found to meet design constraints, or the optimization problem, where values of free variables must be found which optimize an objective function. These more complex problems are often encountered while attempting to solve realistic simulation problems.

A simple but not entirely satisfactory solution to this problem was implemented during this project. This was achieved by integrating a simple optimization search routine into MicroSim. This relatively simple solution increased the flexibility of MicroSim significantly, making it much more useful for many applications. The implementation uses simple search techniques and crude methods for solving the design problem.

Research directed at more efficient methods of integrating optimization methods into sequential-modular simulators should be undertaken. This approach will retain the benefits of a sequential-modular simulator i.e. robust convergence techniques, ease of developing and adding new unit models, but will allow as much flexibility as an equation-oriented simulator, without the disadvantages associated equation based simulator. The existence of such a simulator will be immensely beneficial to the minerals processing industry, allowing more complex and meaningful simulation problems to be solved than at present.

The implementation of the simple search technique showed that the modular structure of MicroSim, as well as the flexible Pascal data structures used in MicroSim allow the integration of new features into the simulator very easily. Thus MicroSim should prove relatively simple to extend to allow particle modifying unit processes to be simulated, as well as for the incorporation of sophisticated optimization techniques.

APPENDIX A

DATA STRUCTURES FOR THE MICROSIM FLOWSHEET EDITOR

Introduction

As described in Chapter 2, there are three objects which need to be provided in a flowsheet editor; streams, units and labels. Each object has different requirements thus a separate data structure was designed for each of these objects. The Pascal data structures and implementation for each object will be discussed briefly.

Streams Data Structures

The type declarations used for the streams are:

```
CONST
maxnostr=100;       {Maximum number of streams }

TYPE
strrange  = 0..maxnostr;
co_ordptr = ^co_ords;

co_ords  = RECORD  {Stream Coordinates}
        x,y    : INTEGER;
        next_co : CO_ORDPTR;
       END;

strm    = RECORD
        strm_head  : CO_ORDPTR;
        xl,yl      : INTEGER; {Label coordinates}
```

```
        END;

strm_mat = RECORD
        unit_from,
        latch_from,
        unit_to,
        latch_to : BYTE;  {Stream connection data}
        id       : ARRAY[1..2] OF BYTE;
        END; {Stream and substream types, Cilliers 1987}

VAR  {Global variables}
 scma    : ARRAY[0..maxnostr] OF STRM_MAT;
 streams : ARRAY[0..maxnostr] OF STRM;
```

Graphical data

A stream may be represented graphically as a number of X,Y graphical coordinates. These X,Y points represent the coordinates of the stream start, the corners and the end. Associated with each stream is a stream identifying number which must also be displayed by the flowsheet editor.

As the number of coordinates which make up a stream is not fixed, but depend on the routing of the stream, a linked list was used for optimal storage of these coordinates. This linked list is declared as record *co_ords*. The head of this linked list is the field *strm_head* of the record *strm*. The fields *XL,YL* also in the *strm* record store the graphical coordinates of the stream number label. The array *streams* thus contains all the graphical data required to reconstruct any stream. The index of this array is the same as the stream number.

Process data

The process data for each stream is stored in the *strm_mat* record. This record contains the number of the unit from which the stream originates as

well as the latch type from the product position at which the stream originates, and the unit number which the stream feeds as well as the latch type at the feed position where the stream terminates. This is in effect a stream connection matrix with latch information added. The *id* array contains the numeric codes which identifies the stream type and the associated substream types which need to be communicated to the rest of the simulator. This code follows the convention implemented by Cilliers (1987) and is derived from consideration of the *latch_to* and *latch_from* fields. Thus array *scma* stores all streams process data, with the index number of this array being the same as the stream number.

Units Data Structures

The relevant declarations for the units are:

```
CONST
 maxnounits = 100;     {Maximum number of units  }

TYPE
 latptr   = ^lat;
 unitrange = 0..maxnounits;

lat = RECORD
     x,y     : INTEGER; {Latch Coordinates}
     lat_num  : BYTE;   {Latch type}
     next_lat : LATPTR;
     END;

unt = RECORD
     scale   : REAL;   {Graphical scale factor}
     unittype : BYTE;   {Unit type }
     xc,yc   : INTEGER; {Unit coordinates}
```

```
    latch    : LATPTR;  {Linked list of latches}
    genprop  : INTEGER; {Stages, if required}
    END;

VAR {Global variables}
units : ARRAY[0..maxnounits] OF UNT;
```

When a user issues a command to draw a unit icon, the MicroSim editor selects the appropriate section of code to execute via a CASE statement. The unit icon is drawn, scaled and moved to the position required by the user. Once this operation is complete, another segement of code is used to supply all the information describing the position and type of the unit as well as the associated latches.

As the unit is drawn by a sequence of coded instructions, only a reference position, scale and unit type needs to be stored to fully describe the icon graphically. This is done in the *unt* record using fields *xc,yc, scale* and *unittype* respectively.

In some cases, the number of stages making up the unit also needs to be stored (flotation, leaching and so on). This is recorded in the *genprop* field.

As minimal information regarding the requirements for hydrometallurgical simulation was available during the design stage of the editor, the data structures designed to accomodate the latch types of each unit were made extensible and flexible. The latch types for each unit are stored as a linked list, allowing as many or as few latches for each unit as required to be accomodated efficiently. The structure of this linked list is as in record *lat*. This record stores the graphical coordinates of each latch *x,y*, the type of the particular latch *lat_num* i.e. which combination of substream types may be attached to a particular unit at this position, and a pointer to the next record in the list *next_lat*. The head of this list is accessible from the record *unt* via the pointer *latch*. Thus all the information describing the units may be easily obtained from the array *units*. Again, the index number of this array is the same as the unit number.

Labels Data Structures

The relevant declarations for the labels are :

```
CONST
maxlabels = 50;  {Maximum number of labels}

TYPE
strng    =STRING[35];

labl_info = RECORD
        x,y   : INTEGER; {Label coordinates}
        kind  : CHAR;    {Label type}
        size  : BYTE;    {Label size}
        direc : INTEGER; {Label orientation}
        labl  : STRNG;   {Label text}
       END;

VAR
labels : ARRAY[1..maxlabels] OF LABL_INFO;
```

The information for each label is stored in the *labl_info* record. This describes the coordinates of the label *x,y*, the font used for that label *kind*, the size and orientation of the label *size* and *direc* as well as the label text *labl*. An array *labels* is used to store all the labels for each flowsheet.

Conclusions

It has been shown how the custom data structure features available in the Pascal language have been used to design efficient structures for various requirements of the MicroSim flowsheet editor. Use of these structures lead to easily maintainable as well as extendable code.

APPENDIX B

AN OPTIMIZATION MODULE FOR MICROSIM

Introduction

As described in the main body of the dissertation, there are currently 19 MicroSim installations, the majority of these at industrial sites. Although MicroSim has performed as designed, a number of problem areas have been identified by users of the program. The first and most obvious is that of parameter estimation.

Parameter estimation

Before a meaningful simulation can be performed, it is necessary to have an estimate of the operating parameters for all the units in the flowsheet being simulated. In some cases these parameters may be trivial to establish e.g. the medium density of a dense medium cyclone. For other units, it may be very difficult.

Consider the use of a breakage and selection function mill model. Although well established methods for the estimation of breakage and selection function parameters exist, the user of the simulation package may not have the time or the resources to implement these algorithms. It also makes little sense providing an easy-to-use simulator without giving the user the means to easily estimate parameters for the models in the simulator.

Experience has shown that users normally follow one of two routes when the parameter estimation problem is encountered. The first route is to abandon simulation and revert to more traditional methods. The other is to use inappropriate parameter values in the belief that any numbers the computer generates must have some value. The latter route can be extremely costly.

The design problem

Often a minerals processor may want to solve a design rather than a simulation problem. In the simulation problem, all the properties of the streams entering the flowsheet and the parameters of each unit operation are defined by the user. The simulator then predicts the performance of the circuit. Sequential-modular simulators solve this kind of problem well, finding a solution in reasonable times for most cases.

For the design problem, the user is faced with a situation having a number of specifications that must be met. The values of free or design variables are sought to meet these specifications. Examples of such situations may be:

- What crusher settings are required to give a 200% recirculating load for a certain crushing circuit ?
- What must the feed rate to the circuit be to produce a product flowrate of X tons per hour ?

The optimization problem

The optimization problem is an extension of the design problem. However for this case, the metallurgist wishes to set the free variables to levels which optimize some measure of the efficiency of the process. For this problem, equality and inequality constraints may also exist. For example "What size flotation cells must be used to maximize recovery while maintaining the grade of the concentrate above Y% ?"

The parameter estimation and design problems can also be expressed as optimization problems. In the parameter estimation problem, values of variables which minimize the difference between measured and simulated quantities are sought. In the design problem, the free variables must be manipulated until the differences between the specified and simulated values are minimized.

It was thus concluded that providing MicroSim with optimization capabilities would significantly increase its usefulness in the industrial situation. The next section describes how this was achieved and also presents some example problems illustrating the ability and limitations of the approach adopted.

Optimization Capabilities Of MicroSim

As a first step it was decided to integrate methods into MicroSim which would allow the user to solve the parameter estimation and design problems. In keeping with the philosophy used for the design of MicroSim, the criterion was that these methods be extremely easy to use. A brief description of the implementation will be given before the example problems are described. All the examples except the optimization example have been contributed by industrial users of MicroSim and thus represent the kinds of real problems which arise in industry.

The optimization algorithm

The Simplex algorithm (not to be confused with the linear programming algorithm of the same name) as described by Caceci and Cacheris (1984) was used. This is a search algorithm and does not require the evaluation of derivatives. The Simplex algorithm works by generating a geometric figure called a simplex which has one more vertex than the space in which it is defined has dimensions. For example, when there are two variables to be found, the simplex would have three vertices i.e. a triangle.

The objective function is evaluated at each vertex. The algorithm then finds the vertex with the worst response (highest value if a minimum is being sought) and rejects this vertex, computing a new vertex at which the response should be lower. This process is repeated until convergence is achieved.

This algorithm was chosen because it is rugged, requires no derivatives, no matrix operations are involved and can be implemented with relatively few lines of code. This last factor is important as MicroSim is written in Turbo Pascal v3.0 (Copyright Borland International) which allows a maximum of 64kb of code space to be used.

The Simplex algorithm is an unconstrained search algorithm, that is the algorithm does not constrain the search variables in any way. For an engineering problem, the variables would always be constrained in some way, for example, a flotation cell must have a volume of greater than zero and a size less than the largest cell manufactured. Single valued transformation methods (Atwood and Foster 1973) were used to transform the real bounded variables to unbounded search variables for the Simplex routine.

The user interface

Westerberg (1981) has described some desirable features which a flowsheeting program designed to solve design and optimization problems should have. Above all, the program should be flexible and interactive, allowing the user to evolve towards an optimal solution. To facilitate this, the problem should be easy to modify so that changes are not inhibited.

The user-interface for the optimization section of MicroSim employs the same features as the data input and editing sections to make the system consistent and easy-to-use. Form filling with appropriate defaults are used for numeric data input, while the standard MicroSim "pointing menus" are used when choices must be made.

The user specifies the problem by selecting streams where specifications are to be made. For the design problem, the design specifications may be entered while for the parameter estimation problem the measured properties of the required streams may be entered. Specifications may take the form of any or all of:

- Water flowrates

- Solids flowrates
- Liquid to solid ratio
- Size distribution
- Mineral grades

in the chosen streams.

Manipulated variables may consist of:

- Any unit parameter (including flotation rate constants)
- Water flowrate in a flowsheet feed stream
- Ore flowrate in a flowsheet feed stream

The optimisation calculation is run interactively, with the user having the ability to terminate the calculation at any stage. The progress of the calculation (sum of squares error, variable values) is displayed on the screen as the calculation runs. This allows the flexibility described by Westerberg, as the user may stop the calculation at any stage, evaluate the status of the calculation, modify variable values and bounds or termination criteria and then restart the calculation. This facilitates the "steering" of the calculation by the user until the optimal solution is attained.

Some examples of parameter estimation, design and optimization problems solved using MicroSim will now be presented. Note that all problems were run on an IBM AT compatible with an 80287 numeric processor. Where appropriate, recycle calculations were converged to relative tolerances of 0.0001. The optimization algorithm was terminated at relative tolerances in the variables and objective value of 0.001.

Parameter Estimation Problems

Two of the more difficult unit operations to obtain parameter estimates for are flotation models and breakage and selection function mill models (assuming no dedicated program for finding these parameters is available). The first

example described deals with estimation of flotation rate constants, while the second describes the estimation of parameters for a breakage and selection function mill model.

Flotation example

Data from a 5 TPH pilot plant were collected to use for the estimation of rate constants. The process recovers three minerals, and mineral grades and mass flowrates in each of 12 consecutive cell concentrate streams for the rougher/scavenger bank were used for parameter estimation. No material balance smoothing was performed on the data.

The MicroSim simulation was set up as consisting of 12 flotation cells in series. Flotation was modelled using the methods described by Sutherland (1977). Specifications consisting of water and solids flowrates and mineral grades based on the measured data, were set on each of the 12 concentrate streams. Manipulated variables were chosen to be the flotation rate constants.

For the first run, it was assumed that the ore could be described by 4 rate constants; one for gangue and one for each of the minerals A, B and C. The results of this parameter estimation run are show in Figures B1 and B2. Figure B1 shows the measured and fitted cumulative mass recoveries for mineral A, the water, and the ore. Acceptable fits are obtained for these quantities. However, from Figure B2 it can be seen that the fits for minerals B and C are far from acceptable.

From the form of discrepancies between measured and fitted values for minerals B and C in Figure B2, it was deduced that minerals B and C contain fast and slow floating components, requiring 2 rate constants for mineral B and 2 for mineral C. Thus, for the second run 6 parameters were estimated; one rate constant each for gangue and mineral A and 2 each for minerals B and C.

Figure B1 : Fitted And Experimental Recoveries For Mineral A, Water And Ore

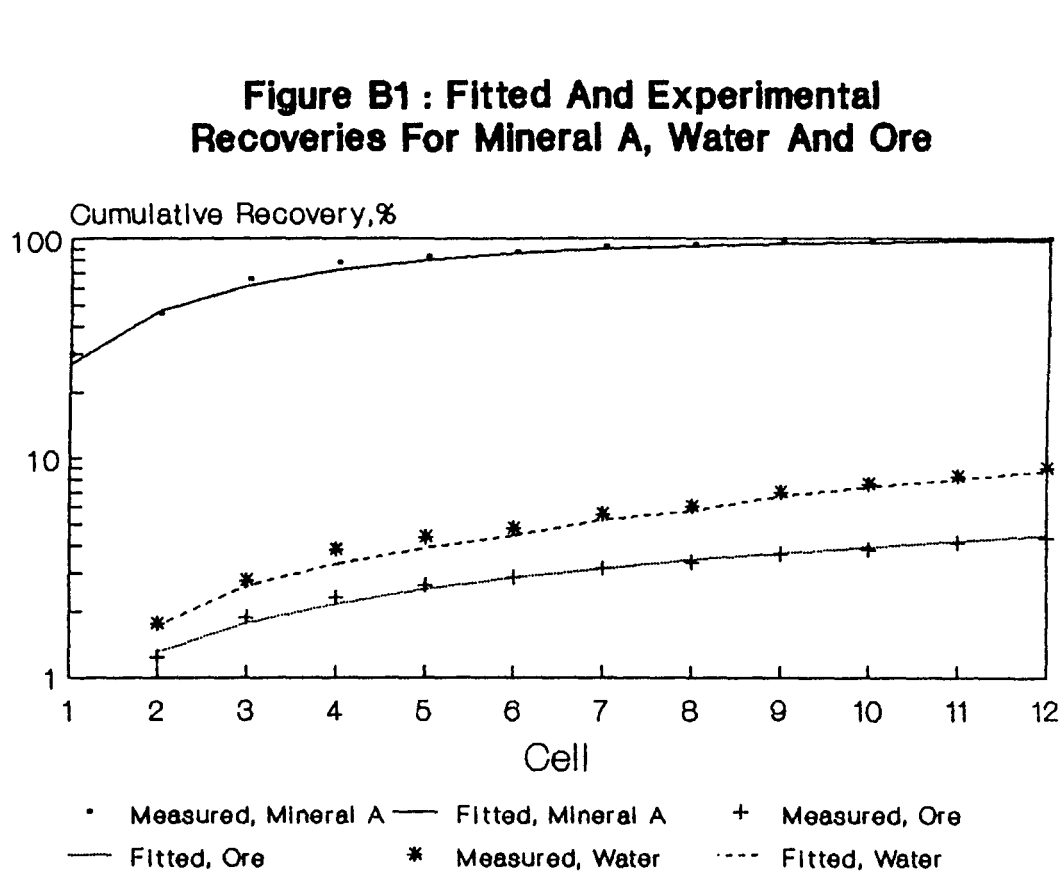


Figure B2 : Fitted And Experimental Recoveries For Minerals B and C

Using 4 Rate Constants

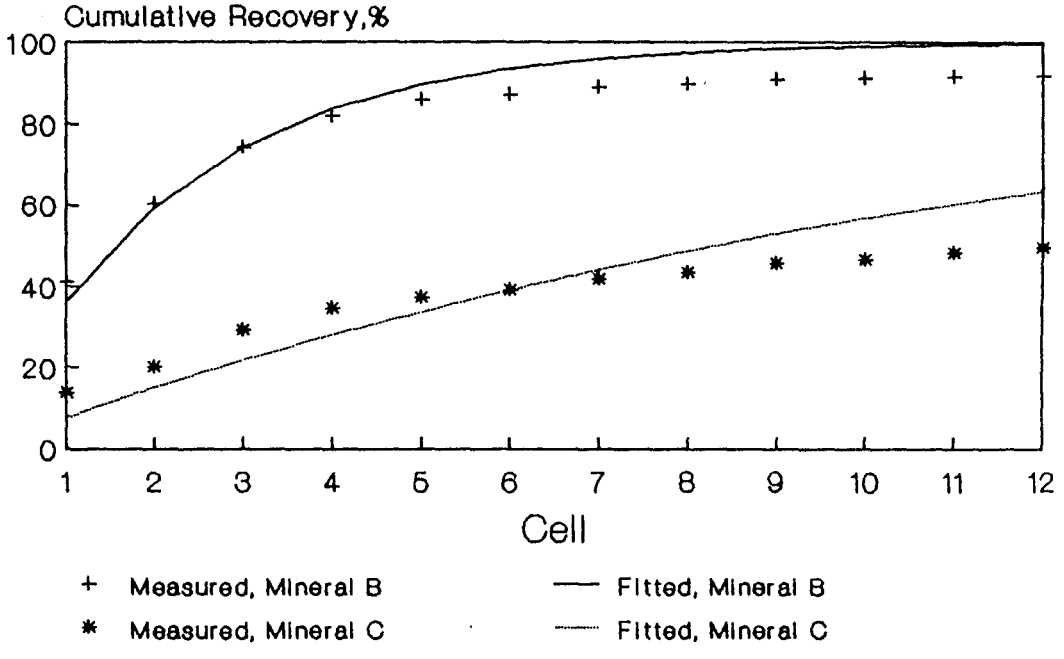


Figure B3 : Fitted And Experimental Recoveries For Minerals B And C
Using 6 Rate Constants

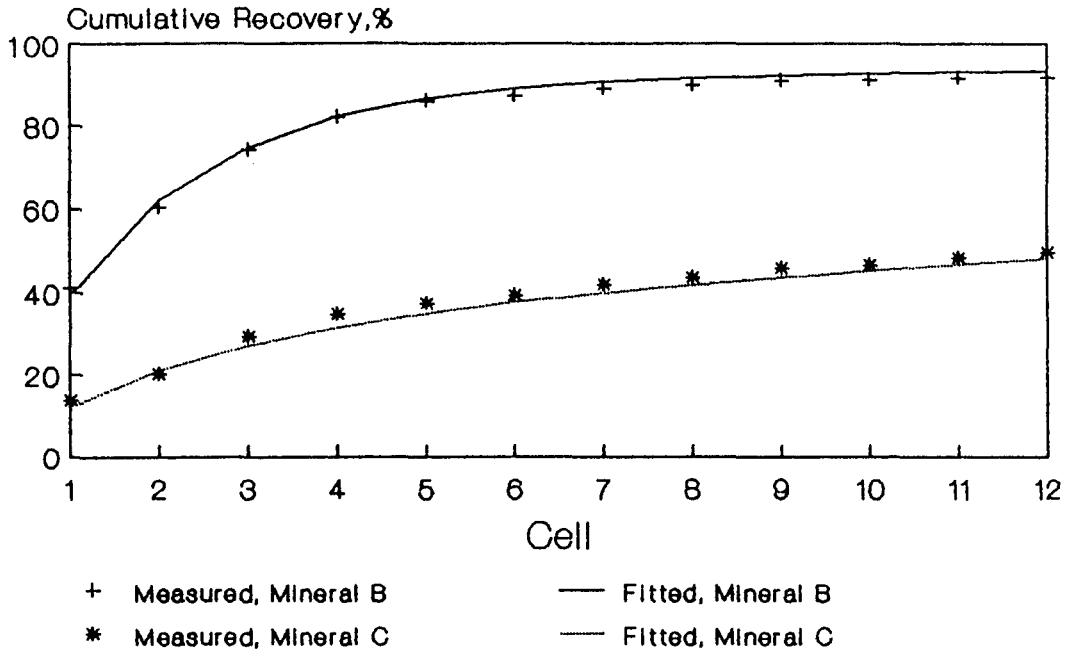
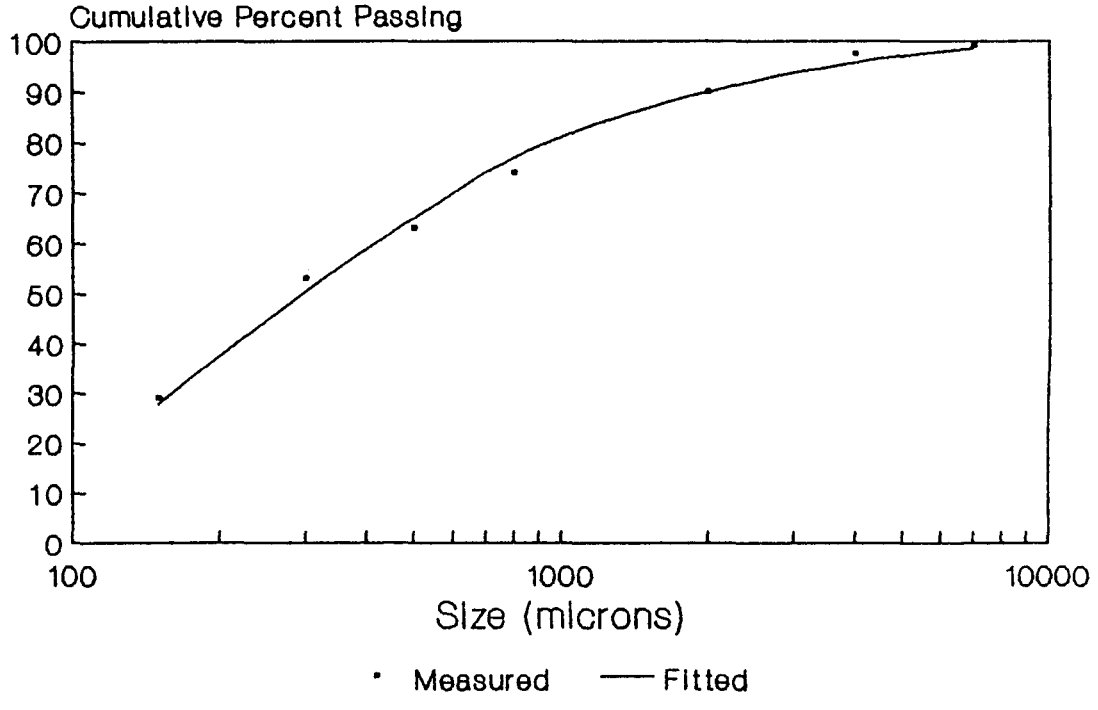


Figure B4 : Fitted And Measured Product Size Distributions



The results of this run for minerals B and C are shown in Figure B3. It can be seen that acceptable fits for these minerals were obtained. The quality of the fits for the other quantities (recoveries of mineral A, total mass, water) were as good if not better than in run 1.

Thus it can be concluded that for this problem, MicroSim was able to estimate 6 parameters successfully.

Milling example

As part of an exercise to improve the performance of a 140 TPH milling circuit, it became necessary to estimate the breakage and selection function parameters for the mill. Thus size distributions of the mill feed and product and an estimate of the residence time was obtained from the plant.

A breakage and selection function model with the breakage and selection function correlations due to Austin (1977) was used to simulate the mill performance. The selection function correlation is:

$$S_i = S_1\left(\frac{D_i}{D_1}\right)^{\alpha}$$

where S_1 is the selection function for the top size.

The cumulative breakage function is modelled by:

$$B(l,x) = \phi\left(\frac{l}{x}\right)^{\gamma} + (1-\phi)\left(\frac{l}{x}\right)^{\beta}$$

$$\phi = \phi_1\left(\frac{x}{D_1}\right)^{-\delta}$$

The appropriate parameters to estimate were judged to be α,β and ϕ_1 as the model predictions were not sensitive (for this case) to the other parameters, the MicroSim default values being accepted for these parameters.

The fit obtained is shown in Figure B4. It can be seen that a good fit for the mill product size distribution was obtained.

A Design Problem

This problem was encountered by a MicroSim user during the design of a bulk sampling plant. The plant flowsheet is shown in Figure B5. As the plant is to be built in an arid area, the plant was designed to minimize the use of fresh water. Dry feed is added in stream 1, with wash water added in stream 2. A dilute slurry stream, stream 8, is also added to the circuit. The product streams 5 and 14 are treated by separate downstream processes, while stream 11 is first thickened before further processing.

Three specifications had to be met. Due to the design and size of the screen to be used, the water and solids flowrate in the screen feed (stream 3) had to have values of 470 and 120 TPH respectively. The product stream 14 needed to meet a specification of 25 percent solids, required by the downstream process.

In order to specify the problem uniquely, three manipulated variables were identified. These were the mass flow in feed stream 1 and the split produced by two way splitters C and D. It can be seen that the presence of the two splitters C and D make the circuit behaviour extremely interactive. This is due to the variance in the performance of cyclone B at the different feed solids concentrations in stream 4 brought about by the manipulation of the splits of C and D.

The problem was described to MicroSim by setting the described specifications on streams 1 and 3. The required manipulated variables were also identified and initial values and bounds were set for the problem. The calculation phase was then initiated. The mass balance calculated by MicroSim at the solution is shown in table B1.

Figure B5 : Flowsheet For Design Problem

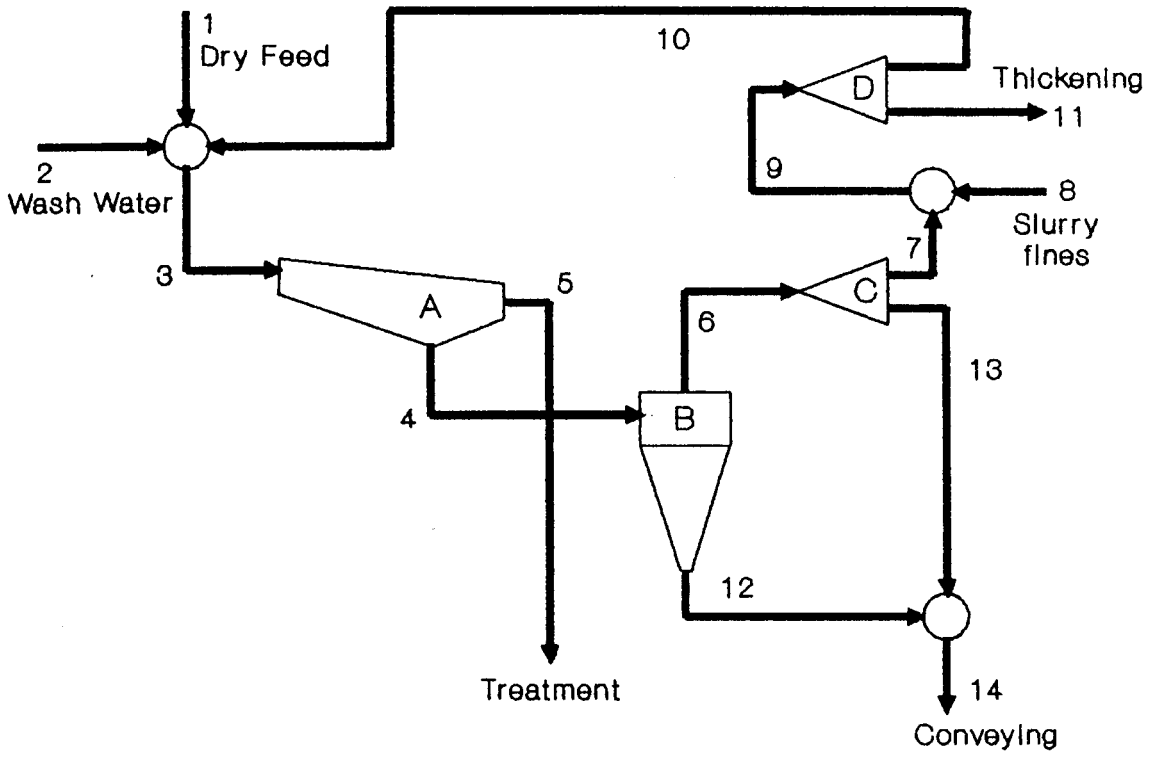


Figure B6 : Flowsheet For Optimization Problem

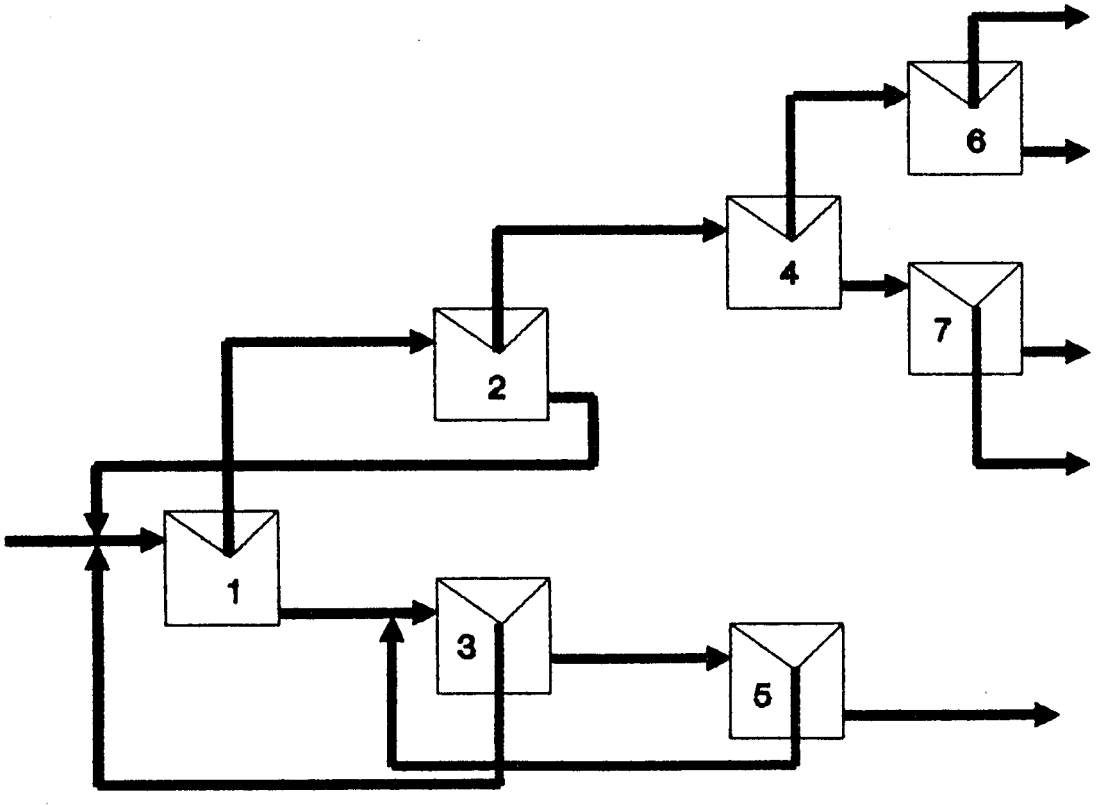


Table B1
Mass Balance At Solution

Strm No	Solids Flow TPH	Water Flow TPH	Mass Percent Solids
1	88.18	0.00	100.00
2	-	110.00	-
3	120.00	470.03	20.34
4	76.26	446.53	14.59
5	43.74	23.50	65.05
6	36.39	417.03	8.03
7	25.74	294.98	8.03
8	12.00	132.00	8.33
9	37.74	426.98	8.12
10	31.82	360.03	8.12
11	5.92	66.94	8.12
12	39.87	29.51	57.47
13	10.65	122.05	8.03
14	50.52	151.56	25.00

It can be seen that the required specifications were met.

An Optimization Problem

In order to evaluate the optimization capabilities of MicroSim more fully, an optimization problem described in the literature (Chan and Prince 1986) was attempted. The flowsheet for the problem is shown in Figure B6. A hypothetical ore consisting of two minerals (value and gangue) is fed to the flotation circuit. The objective is to find values for the seven flotation cell volumes which optimizes the circuit profit. The profit is calculated from the value of the output streams less the value of the input stream and the unit operating costs. The operating costs are calculated as a function of cell volume. See Chan and Prince for a detailed description of the problem and the models used.

At present it is not possible to set up arbitrary optimization problems using MicroSim. Thus some code was added to allow the evaluation of the objective function for this problem. The results obtained by MicroSim are compared to those obtained by Chan and Prince for this problem in table B2.

Table B2
Results Of Optimization Problem

	MicroSim	Chan & Prince
Profit $/hr	437.53	439.75
V1	8.42	7.91
V2	0.88	0.80
V3	19.10	17.80
V4	0.63	0.41
V5	16.05	16.74
V6	0.25	0.19
V7	0.22	0.29

Note that the volumes for the cells given above are in cubic meters.

Discussion Of Results

Table B3 below presents a summary of the performance of MicroSim on the problems described above.

Table B3
Performance Summary

Problem	Simplex Iterations	Total Time (seconds)	STEs
Mill	73	131	172
Flotation	128	1103	221
Design	149	415	80
Optimization	152	1066	47

A simulation time equivalent (STE) is defined as the time taken to solve the design (or optimization) problem divided by the time taken for a single converged sequential-modular simulation of the flowsheet. It is a measure of how much more time the the optimization problem requires for solution compared to the time required to solve the simulation problem.

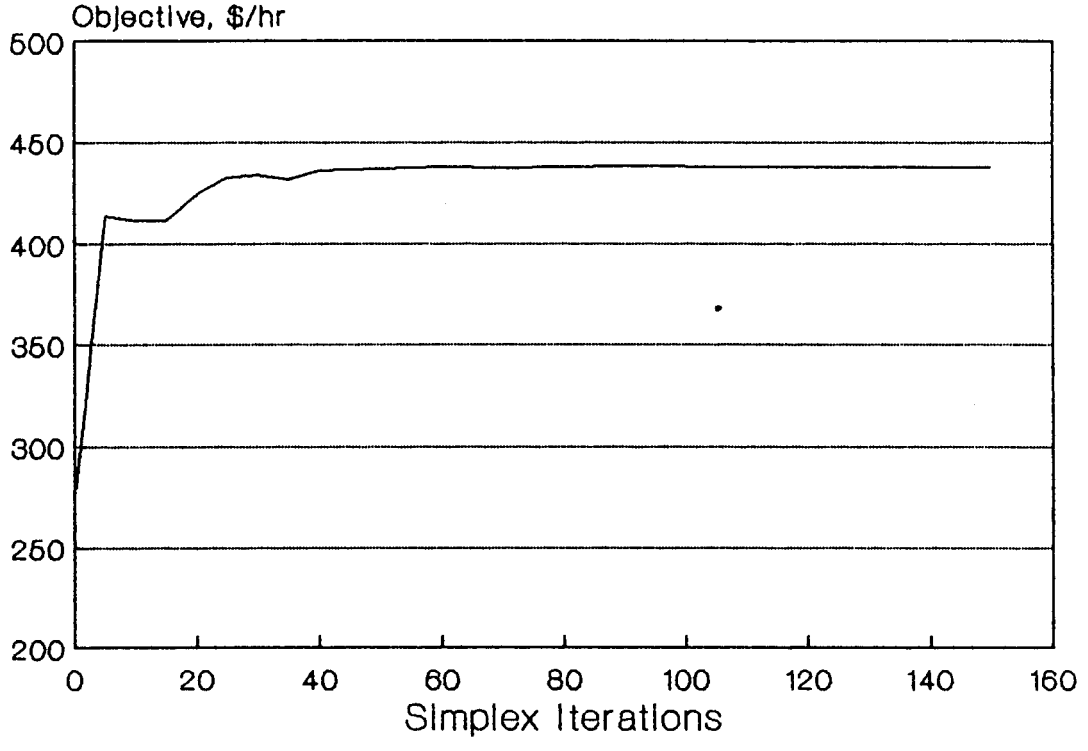
From table B3, it can be seen that the time required to solve the problems are fairly reasonable, ranging from about 4 to 20 minutes. As microcomputer time is extremely inexpensive, these times can be tolerated. However, the STEs colum shows that it may take anything from 50 to 250 times longer to solve the more complex optimization or design problem than required to solve the simulation problem for the same flowsheet.

Even though the method used is inefficient, it can be seen that MicroSim is capable of solving realistic and complex parameter estimation, design, and optimization problems. The greatest strength of this facet of the simulator is the easy way in which such problems can be set up for design and parameter estimation problems. This allows problems to be set up and solved by people with little detailed knowledge of simulation. No knowledge of computer programming is required. At present, optimization problems require some coding to be performed.

However, the optimization algorithm used is far from ideal. The Simplex algorithm requires many iterations to converge. Computationally this is very expensive as each iteration requires at least one fully converged flowsheet calculation to be performed. This results in the long computing times reported above.

Another problem with the Simplex algorithm is its poor convergence properties. Figure B7 shows the value of the objective function as a function of the number of Simplex iterations for the optimization problem discussed above. It can be seen that the algorithm locates the approximate optimum rapidly. However, it requires many iterations to locate the optimum more precisely. In addition, when the number of variables is greater than about three, the algorithm does not locate the optimum very precisely. This can be seen from the results in table B2; even though the Simplex came very close to the

Figure B7 : Objective Value As A Function Of Simplex Iterations



optimum value of the objective function, it was not able to locate the optimum variable values very precisely. This is a characteristic of the Simplex algorithm (Dixon 1973).

General constraints, as described in the beginning of the paper, are not handled well by Simplex type methods, although some work in providing Simplex type algorithm with this ability has been reported (Dixon 1973; Box 1965).

Conclusions

It was demonstrated that the MicroSim program structure is such that major extensions and modifications to the program can be made with relative ease. This is attributed to the use of Pascal, which encourages the writing of modular code.

The optimization capabilities which have been added to MicroSim make the program significantly more useful. The present implementation is not very efficient, but has provided a glimpse of the power and flexibility a simulator capable of solving design and optimization problems would have.

It is felt that this is a prime area for further research, with the implementation of more efficient algorithms having priority. The simultaneous-modular successive quadratic programming (SQP) approach has proved to be very efficient for solving chemical engineering optimization and design problems (Chen and Stadtherr 1985a, 1985b 1985c). The SQP approach is currently one of the most efficient methods for solving general constrained multivariable nonlinear optimization problems. Research into the feasibility and implementation and of SQP methods for MicroSim is strongly recommended.

APPENDIX C

A VARIABLE STEP LENGTH ALGORITHM FOR THE INTEGRATION OF SYSTEMS OF DIFFERENTIAL EQUATIONS

Introduction

Although many numerical analysis textbooks provide Runge-Kutta type integration algorithms which use fixed step lengths, variable step length algorithms seem to be much more difficult to find. In addition, variable step length algorithms require the inclusion of many empirical rules which are not widely published. Thus the Pascal coding for the ODE solver used in this work is provided in the hope that it may be of general use.

Numerical Techniques

The algorithm is based on a fourth order Runge-Kutta scheme derived by Fehlberg (1970). The coefficients in this scheme give much lower truncation errors than other fourth order schemes due to the coefficient values derived by Fehlberg. A fifth order formula is used to estimate the truncation error, forming the basis of the variable step length algorithm. Practical implementation techniques were obtained from a paper by Thomas (1986). These techniques were used to develop a reliable and robust means of adjusting the step length.

Test Problems

In order to test the performance of the algorithm, various test problems were solved. The results from two of these problems will be presented.

Problem 1

This problem was taken from Conte and De Boor (1972) who have solved this problem using the subroutine DVERK from the IMSL mathematical library. The problem consists of the following four differential equations, solved for x=0 to 1.0.

$$\frac{dy_1(x)}{dx} = y_3(x)$$

$$\frac{dy_2(x)}{dx} = y_4(x)$$

$$\frac{dy_3(x)}{dx} = y_1(x)^2 - y_2(x) + \exp x$$

$$\frac{dy_4(x)}{dx} = y_1(x) - y_2(x)^2 - \exp x$$

The following initial conditions apply:

$$y_1(0.0) = 0.0$$

$$y_2(0.0) = 1.0$$

$$y_3(0.0) = 0.0$$

$$y_4(0.0) = -2.0$$

Results obtained using the Fehlberg based ODE solver for this problem were the same (to seven significant digits) as those reported by Conte and De Boor. The Fehlberg routine performed 138 function evaluations to obtain this accuracy while DVERK used approximately 160 function evaluations.

Problem 2

This problem was taken from the IMSL manual and is used in the IMSL manual to illustrate use of the well known ODE solver DGEAR. The problem consists of two differential equations, solved over the range x=0.0 to 10.0. The equations are :

$$\frac{dy_1(x)}{dx} = 2y_1(x)(1.0 - y_2(x))$$

$$\frac{dy_2(x)}{dx} = y_2(x)(y_1(x) - 1.0)$$

The following initial conditions apply:

$$y_1(0.0) = 1.0$$

$$y_2(0.0) = 3.0$$

Results obtained using the Fehlberg algorithm were the same (to the number of digits reported in the IMSL manal) as those obtained by DGEAR. The Fehlberg routine required 312 function evaluations to perform the integration, while the number required by DGEAR was not reported.

From the above, it can be concluded that the variable step-length implementation of the Fehlberg algorithm performs satisfactorily.

Pascal Source Code

The source code for the algorithm follows. The source code illustrates how Problem 1 was set up.

```
{=========================================================}
{ This program solves a set of simultaneous ODE's using Fehlbergs  4th order RK scheme
with stepsize control. Dynamic memory techniques are used to store all VECTORS.}
{=========================================================}

CONST
 length1=100;
 length2=10;
 max_space=1000;        {length1*length2}

TYPE
 stor_array=ARRAY[1..length2] OF REAL;
 to_stor   =^stor_array;
 base_array=ARRAY[1..length1] OF TO_STOR;
 max_range =0..max_space;

VAR
 evals     : INTEGER;

{=========================================================}
{ This procedure allocates N real storage spaces to variable VEC}
{=========================================================}

PROCEDURE MAKE_SPACE(VAR vec : BASE_ARRAY;
                     n   : MAX_RANGE);

VAR
 i,l1,l2   : MAX_RANGE;

BEGIN
 l1:=n DIV length2;        l2:=n MOD length2;
 IF (l2>0) THEN l1:=l1+1;
 FOR i:=1 TO l1 DO
  NEW(vec[i]);
END;
{---------------------------------------------------------}

{=========================================================}
{ This procedure removes storage space of variable VEC from memory}
{=========================================================}

PROCEDURE KILL_SPACE(VAR vec : BASE_ARRAY;
                     n   : MAX_RANGE);

VAR
 i,l1,l2   : MAX_RANGE;
```

```
BEGIN
 l1:=n DIV length2;       l2:=n MOD length2;
 IF (l2>0) THEN l1:=l1+1;
 FOR i:=1 TO l1 DO
  DISPOSE(vec[i]);
END;
{-------------------------------------------------------------}

{=============================================================}
{ This procedure stores value VAL at index value I in variable VEC}
{=============================================================}

PROCEDURE S(vec : BASE_ARRAY;
        i  : MAX_RANGE;
        val : REAL);

VAR
 l1,l2  : MAX_RANGE;

BEGIN
 l1:=i DIV length2;       l2:=i MOD length2;
 IF (l2>0) THEN l1:=l1+1
 ELSE l2:=length2;
 vec[l1]^[l2]:=val;
END;
{-------------------------------------------------------------}

{=============================================================}
{ This returns the value at index value I in variable VEC}
{=============================================================}

FUNCTION  R(vec : BASE_ARRAY;
        i  : MAX_RANGE) : REAL;
VAR
 l1,l2  : MAX_RANGE;

BEGIN
 l1:=i DIV length2;       l2:=i MOD length2;
 IF (l2>0) THEN l1:=l1+1
 ELSE l2:=length2;
 R:=vec[l1]^[l2];
END;
{-------------------------------------------------------------}

FUNCTION POW(y,x : REAL) : REAL;
BEGIN
 IF (y=0) THEN POW:=0
 ELSE POW:=EXP(x*LN(y));
END;
{-------------------------------------------------------------}
```

```
PROCEDURE MACHINE_EPSILON(VAR eps : REAL);
BEGIN
 eps:=1;
 WHILE ((eps+1)>1) DO
  eps:=0.5*eps;
END;
{-------------------------------------------------------------}

{=============================================================}
{ This procedure describes the users system of equations. The parameters are :
  t    : Value of independent variable at which DX_DT must be  calculated.
  X    : Vector of dependent variables at which DX_DT must be calculated.
  deriv : Vector of derivative values.}
{=============================================================}

PROCEDURE DX_DT(t    : REAL;
                x    : BASE_ARRAY;
                deriv : BASE_ARRAY);

VAR
 i : INTEGER;

BEGIN
 evals:=evals+1;
 S(deriv,1,R(x,3));
 S(deriv,2,R(x,4));
 S(deriv,3,SQR(R(x,1))-R(x,2)+EXP(t));
 S(deriv,4,-SQR(R(x,2))+R(x,1)-EXP(t));
END;
{-------------------------------------------------------------}

{=============================================================}
{ These procedures integrate a system dX/dt=F of ODE's numerically using the RK4 method
due to FEHLBERG. Control of the steplength is implemented using techniques described by
Thomas in BYTE, April 1986. The variables passed to the procedure are :
   X    : Vector of variables. Upon entry they contain initial values, upon exit they contain final
values.
  tstart : Time or independent variable value at which integration starts.
    tend : Time or independent variable value at which integration must be stopped. Updated to
reflect where integration stopped.
  hstart : Users guess for initial value of steplength. Upon exit, updated to last value of
steplength used.
  hmin   : User specified limit for step length.
    N    : Number of equations in the system.
   tola  : Absolute tolerance required
   tolr  : Relative tolerance required
   round : Flag set to TRUE if rounding error encountered
   stiff : Flag set to TRUE if stiffnes detected. If stiffnes     detected, routine is exited with tend
indicating the value of the independent variable where stiffnes was detected. X contains the
values of the dependent variables at this point.
The function DX_DT must be modified by the user for his specific problem.}
 {=============================================================}
```

```
PROCEDURE PREDICTOR(   x      : BASE_ARRAY;
                       xnew   : BASE_ARRAY;
                       t      : REAL;
                       h      : REAL;
                       n      : MAX_RANGE;
                       errors : BASE_ARRAY);

VAR
 k0,k1,k2,k3,k4,k5            : BASE_ARRAY;
 i                            : MAX_RANGE;

 BEGIN
  MAKE_SPACE(k0,n);                MAKE_SPACE(k1,n);
  MAKE_SPACE(k2,n);                MAKE_SPACE(k3,n);
  MAKE_SPACE(k4,n);                MAKE_SPACE(k5,n);

  DX_DT(t,x,k0);

  FOR i:=1 TO n DO
   S(errors,i,R(x,i)+h*R(k0,i)/4);

  DX_DT(t+h/4,errors,k1);

  FOR i:=1 TO n DO
   S(errors,i,R(x,i)+h*(R(k0,i)*0.09375+R(k1,i)*0.28125));

  DX_DT(t+h*0.375,errors,k2);

  FOR i:=1 TO n DO
   S(errors,i,R(x,i)+h*(R(k0,i)*0.879381-R(k1,i)*3.2771962
                +R(k2,i)*3.3208921));

  DX_DT(t+h*0.9230769,errors,k3);

  FOR i:=1 TO n DO
   S(errors,i,R(x,i)+h*(R(k0,i)*2.0324074-R(k1,i)*8.0
                +R(k2,i)*7.1734893-R(k3,i)*0.2058967));

  DX_DT(t+h,errors,k4);

  FOR i:=1 TO n DO
   S(errors,i,R(x,i)+h*(-R(k0,i)*0.2962963+R(k1,i)*2.0
                 -R(k2,i)*1.3816764+R(k3,i)*0.4529727
                 -R(k4,i)*0.275));

  DX_DT(t+h/2,errors,k5);

  FOR i:=1 TO n DO
  BEGIN
   S(xnew,i,R(x,i)+h*(0.1185185*R(k0,i)+0.5189864*R(k2,i)
               +0.5061315*R(k3,i)-0.18*R(k4,i)
               +0.0363636*R(k5,i)));
```

```
 S(errors,i,h*ABS(R(k0,i)/360+0.0299415*R(k2,i)+0.0291999*R(k3,i)
          -0.02*R(k4,i)-0.0363636*R(k5,i)));
 END;

 KILL_SPACE(k0,n);              KILL_SPACE(k1,n);
 KILL_SPACE(k2,n);              KILL_SPACE(k3,n);
 KILL_SPACE(k4,n);              KILL_SPACE(k5,n);

 END;
{-------------------------------------------------------------}

PROCEDURE FEHLBERG(   x        : BASE_ARRAY;
              n        : MAX_RANGE;
              tstart   : REAL;
            VAR tend       : REAL;
            VAR hstart     : REAL;
              hmin     : REAL;
              tola     : BASE_ARRAY;
              tolr     : REAL;
            VAR round,stiff : BOOLEAN);

VAR
 errors,xnext                : BASE_ARRAY;
 i                       : MAX_RANGE;
 h_reduced               : BOOLEAN;
 t,hdiv2,maxerror,minerror,error  : REAL;
 ss,slim,slim_meth,v,vlim,h,hmax  : REAL;
 hmax_set,q,h_old,vmin,eps,xi,hnew : REAL;

LABEL
 exit;

BEGIN
 MAKE_SPACE(errors,n);          MAKE_SPACE(xnext,n);
 ss:=POW(0.5,1/(4+1));          slim:=3/ss;
 hmax_set:=0.5;                 q:=6;
 h_old:=hstart;                 vlim:=3/ss;
 round:=FALSE;                  stiff:=FALSE;

 MACHINE_EPSILON(eps);
 h_reduced:=FALSE;

 IF (ABS(tstart-tend)>hmax_set) THEN hmax:=ABS(tstart-tend)
 ELSE hmax:=hmax_set;

 t:=tstart;

 REPEAT
  h_reduced:=FALSE;

  REPEAT   {Integrate until error OK}
   PREDICTOR(x,xnext,t,h_old,n,errors);
   vmin:=1e30;
```

```
FOR i:=1 TO n DO
BEGIN
 xi:=ABS(R(x,i));
 error:=ABS(R(errors,i));
 IF (error<>0) THEN v:=POW((R(tola,i)+tolr*xi)/error,0.2)
 ELSE v:=vlim;
 IF (v<vmin) THEN vmin:=v;
END;


IF (vmin<1) THEN                    {Tolerance not met}
BEGIN
 h_old:=ss*vmin*h_old;
 h_reduced:=TRUE;

 IF (h_old<hmin) THEN    {Stiffnes/discontinuity encountered}
 BEGIN
  stiff:=TRUE;
  tend:=t;
  GOTO exit;
 END;

END;
UNTIL (vmin>=1);

t:=t+h_old;
FOR i:=1 TO n DO
 S(x,i,R(xnext,i));

IF (vmin<vlim) THEN hnew:=h_old*ss*vmin
ELSE hnew:=h_old*ss*vlim;

IF (hnew>hmax) THEN hnew:=hmax;

IF (NOT h_reduced) THEN
 IF (hnew>h_old) THEN h_old:=hnew;

IF (t<>0) THEN
 IF (hnew/(18*ABS(t))<eps) THEN round:=TRUE;

IF (h_old<hmin) THEN       {Stiffnes/discontinuity encountered}
BEGIN
 stiff:=TRUE;
 tend:=t;
 GOTO exit;
END;

UNTIL ((t+h_old)>=tend);

h_old:=tend-t;
PREDICTOR(x,xnext,t,h_old,n,errors);
t:=t+h_old;
```

```
FOR i:=1 TO n DO
 S(x,i,R(xnext,i));

hstart:=h_old;
tend:=t;

EXIT :
 KILL_SPACE(errors,n);          KILL_SPACE(xnext,n);
END;
{-------------------------------------------------------------}

{-------------------------------------------------------------}
{            M A I N   P R O G R A M                          }
{-------------------------------------------------------------}

VAR
 y,tol      : BASE_ARRAY;
 n          : MAX_RANGE;
 hstart,tend  : REAL;
 i,j        : INTEGER;
 round,stiff  : BOOLEAN;

BEGIN
 n:=4;
 MAKE_SPACE(y,n);          MAKE_SPACE(tol,n);

 S(y,1,0.0);               S(tol,1,0.001);
 S(y,2,1.0);               S(tol,2,0.001);
 S(y,3,0.0);               S(tol,3,0.001);
 S(y,4,-2.0);              S(tol,4,0.001);

 hstart:=0.05;             tend:=1.0;
 round:=FALSE;             stiff:=FALSE;
 evals:=0;

 FEHLBERG(y,n,0,tend,hstart,0.0001,tol,1e-6,round,stiff);
 IF stiff THEN WRITELN('Stiffness encountered');
 IF round THEN WRITELN('Round off error occurred');
 WRITELN('Y1  = ',R(y,1));
 WRITELN('Y2  = ',R(y,2));
 WRITELN('Function evaluations =',evals);
END.
```

APPENDIX D

AN INTERPOLATION TECHNIQUE FOR FINDING MASS DENSITY ON FIXED CLASS BOUNDARIES

Introduction

The need for an interpolation technique arises because of the method used to solve the macroscopic population balance viz. the method of moments. Using this method, the differential equations defining the rate of change of the carbon loading mass density function P, are integrated along a number of characteristics. The characteristics define the path of the particles in the y-t plane. Thus, the P values are calculated at y values defined by the integration of the characteristic equations, and not at the fixed y class boundaries. As it has been assumed that the particle sizes do not change, characteristics originating from a certain size class remain at that size during the integration, thus the P values are defined at the fixed size class boundaries.

To facilitate transfer of streams data in a general simulator, the P values need to be defined at the fixed y class values common to all carbon streams in the flowsheet being simulated. Thus an interpolation method is required to establish the values of P at these boundaries from the values defined by the integration of the characteristics. The interpolation method has to satisfy the criterion that the integral :

$$I(\delta) = \int_{o}^{\infty} P(y,\delta)dy$$

evaluated at the interpolated points and evaluated at the points defined by the integration of the characteristics be identical. As $I(\delta)$ is the mass or mass fraction of material in size class δ, any errors in this quantity would lead to a gain or loss of mass with serious implications for simulation accuracy.

Although standard linear and polynomial interpolation techniques were tried, none met the above requirement. Thus an interpolation method which ensured the conservation of the integral $I(\delta)$ was derived.

The Interpolation Technique

As particle size does not vary, it is sufficient to construct an interpolation scheme which can be used for a single size class. The entire problem is solved by repeating the scheme for the number of size classes. Assuming that there exists N+1 loading or y classes, define the following notation :

y_i = y value defined by integration of characteristic equation i at time t. i=0...N.

$\hat{y}_i$ = Fixed y value at class boundary i. i=0...N.

P_i = Mass density or mass defined by integration **along** characteristic i at time t. i=0...N.

$\hat{P}_i$ = Interpolated value of mass density or mass at $\hat{y}_i$.

At any time t, it is possible to find λ and ρ defined by:

λ = Maximum λ such that $\hat{y}_\lambda < y_o$

ρ = Minimum ρ such that $\hat{y}_\rho > y_N$

All characteristics thus lie between $\hat{y}_\lambda$ and $\hat{y}_\rho$

The algorithm proceeds as follows:

1) $\hat{P}_\lambda = 0$ and $\hat{P}_\rho = 0$. Also set all other $\hat{P}_i, i < \lambda$ and $i > \rho$ to zero. That is, all $\hat{P}_i$ **outside** the range defined by λ and ρ are set to zero.

2) Using the trapezoidal rule for integration (as the y values are not, in general, equally spaced, the integral to be preserved can be expressed as:

$$I(\delta) = \frac{1}{2} \sum_{i=\lambda}^{\rho-1} (\hat{P}_i + \hat{P}_{i+1})(\hat{y}_{i+1} - \hat{y}_i)$$

Thus

$$2I(\delta) - \sum_{i=\lambda+1}^{\rho-1} \hat{P}_i(\hat{y}_{i+1} - \hat{y}_i) = 0$$

$$= C \qquad \{D1\}$$

as $\hat{P}_\lambda = \hat{P}_\rho = 0$

Equation {D1} represents the constraint that must be met. The integral $I(\delta)$ in {D1} is evaluated using the trapezoidal rule at the calculated (by integration of the system of ODEs) values P_i and y_i.

The unknowns to be found are thus $\hat{P}_j$ for $j = \lambda+1$ to $\rho-1$

In order to preserve the shape of the calculated P-y curve as closely as possible, the following function is minimized

$$M = \sum_{\lambda+1}^{\rho-1} \left(1 - \frac{\hat{P}_i}{\overline{P}_i}\right)^2 \qquad \{D2\}$$

Equation {D2} holds for all $\overline{P}_i \neq 0$.

$\overline{P}_i$ represents the value of P calculated at $\hat{y}_i$ using linear interpolation of the P_i and y_i values. Thus M is an expression of the relative error between the linearly interpolated P values and those calculated P values that satisfy the constraint {D1}.

It can be seen from {D1} that if $\hat{P}_i = 0$ then that value makes no contribution to the integral. Thus only the classes having non-zero $\overline{P}_i$ are considered in {D2}. If a class i has $\overline{P}_i = 0$, then $\hat{P}_i$ is set to zero, reducing the number of unknowns. (Note : this applies for $i = \lambda+1$ to $\rho-1$).

There is an additional constraint on $\hat{P}_i$, that is $\hat{P}_i \geq 0$.

Thus the transformation $\hat{P}_i = \phi_i^2$ is made.

Using the Lagrange multiplier method for constrained minimisation, form the objective function H:

$$H = M + \Lambda C$$

where Λ is a Lagrange multiplier.

To perform the minimisation subject to the constraint set:

$$\frac{\partial H}{\partial \Lambda} = 0$$

and

$$\frac{\partial H}{\partial \phi_i} = 0$$

This implies that:

$$2I(\delta) - \sum_{j=\lambda+1}^{\rho-1} \phi_j^2 (\hat{y}_{j+1} - \hat{y}_j) = 0 \qquad \{D3\}$$

and

$$-\frac{1}{\overline{P}_i} + \left(\frac{\phi_i}{\overline{P}_i}\right)^2 + \frac{\Lambda}{2}(\hat{y}_{j+1} - \hat{y}_j) = 0 \qquad \{D4\}$$

Equation {D4} holds for $i = \lambda + 1$ to $\rho - 1, \overline{P}_i \neq 0$

After some manipulation of {D3} and {D4}:

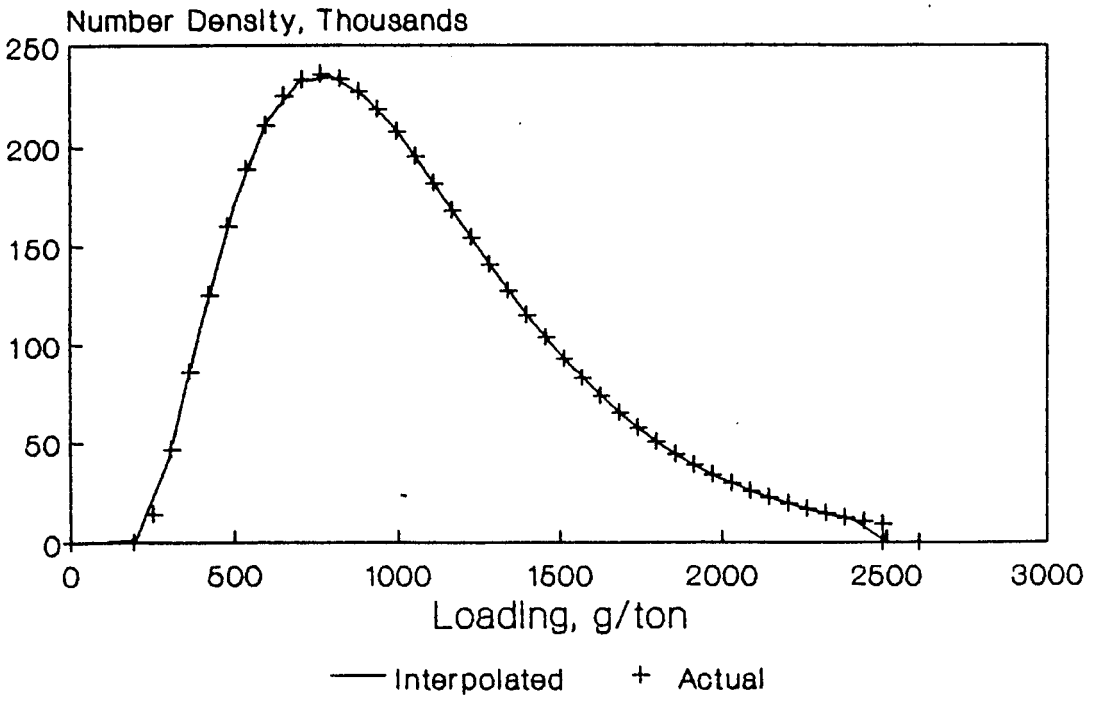
$$\Lambda = \frac{2\sum_{i=\lambda+1}^{\rho-1} \overline{P}_i(\hat{y}_{i+1} - \hat{y}_{i-1}) - 4I}{\sum_{i=\lambda+1}^{\rho-1} (\overline{P}_i(\hat{y}_{i+1} - \hat{y}_{i-1}))^2} \qquad \{D5\}$$

Once Λ is calculated from {D5}, the ϕ_i values can be calculated from {D4}. Thus all the $\hat{P}_i$ can be determined sequentially without the time consuming solution of a system of equations.

Results

Figure D1 illustrates a typical result obtained using this algorithm.

Figure D1 : Actual And Interpolated Distributions



APPENDIX E

DERIVATION OF CONTINUOUS LEACHING MODEL FOR LOVEDAY RATE EXPRESSION

The Loveday rate expression for the batch situation is written:

$$S(t) = (S_o - S_m)\left(1 - \frac{n}{n+1} k_{\max} t\right) + S_m \qquad t < \frac{1}{k_{\max}}$$

{E1a}

$$S(t) = \frac{(S_o - S_m)}{n+1}\left(\frac{1}{k_{\max} t}\right)^n + S_m \qquad t > \frac{1}{k_{\max}}$$

{E1b}

If leaching takes place in a series of CSTR tanks, the particles will undergo leaching for a range of various times. The average concentration of gold in the exit of the first stage is given by:

$$\overline{S}_1 = \int_0^\infty S(t_1) E(t_1) dt_1$$

{E2}

where E(t) is the residence time distribution of the tank. That is, E(t)dt is the fraction of solids spending a time between t and (t+dt) in the tank. Assuming perfect CSTR behaviour:

$$E(t) = \frac{1}{\tau} e^{\frac{-t}{\tau}}$$

{E3}

where τ is the nominal residence time of ore in the tank.

Substituting {E1} and {E3} into {E2} and performing some manipulation:

$$\overline{S}_1 = (S_o - S_m)A(\tau_1) + S_m$$

{E4}

where:

$$A(\tau) = \{1 - \alpha(\tau)\} - \frac{n k_{\max}}{(n+1)}\left(\tau\{1 - \alpha(\tau)\} - \frac{\alpha(\tau)}{k_{\max}}\right) + \frac{\alpha(\tau)}{(n+1)}\left(\frac{1}{k_{\max}}\right)^n I(\tau)$$

$$\alpha(\tau) = e^{\frac{-1}{k_{\max}\tau}}$$

$$I(\tau) = \int_o^1 \frac{dx}{\left(-\tau \ln(x) + \frac{1}{k_{\max}}\right)^n}$$

Now consider a particle which spends time t_1 in CSTR 1 and time t_2 in CSTR 2.

$$S_1(t_1) = (S_o - S_m)\left(1 - \frac{n}{n+1}k_{\max}t_1\right) + S_m \qquad t_1 < \frac{1}{k_{\max}}$$

{E5a}

$$S_1(t_1) = \frac{(S_o - S_m)}{n+1}\left(\frac{1}{k_{\max}t_1}\right)^n + S_m \qquad t_1 > \frac{1}{k_{\max}}$$

{E5b}

$$S_2(t_1,t_2) = (S_1 - S_m)\left(1 - \frac{n}{n+1}k_{\max}t_2\right) + S_m \qquad t_2 < \frac{1}{k_{\max}}$$

{E6a}

$$S_2(t_1,t_2) = \frac{(S_1 - S_m)}{n+1}\left(\frac{1}{k_{\max}t_2}\right)^n + S_m \qquad t_2 > \frac{1}{k_{\max}}$$

{E6b}

Now, the average concentration of gold in the exit of the second stage is given by:

$$\overline{S}_2 = \int_o^\infty \int_o^\infty C_2(t_1,t_2)E(t_2)E(t_1)dt_2dt_1$$

{E7}

Substituting equations {E3}, {E5} and {E6} into {E7} and after some manipulation:

$$\overline{S}_2 = (S_o - S_m)A(\tau_1)A(\tau_2) + S_m$$

{E8}

Now as the amount of gold in the ore is small, the solids mass flowrate is not changed significantly by leaching. If the CSTR vessels are all of the same size, the residence time in each vessel is the same. Thus:

$$\overline{S}_2 = (S_o - S_m)A(\tau)^2 + S_m$$

{E9}

Although no formal proof is offered, it can be seen that from the structure of the expressions, the concentration of gold in the ore in the exit of tank N can be written:

$$\overline{S}_N = (S_o - S_m)A(\tau)^N + S_m$$

{E10}

which is the form of the equation presented in the body of the dissertation.

APPENDIX F

DATA STRUCTURES FOR CARBON AND LIXIVIANT STREAMS

Introduction

As described in Chapter 5, data structures which allow the properties of carbon streams and dissolved species in aqueous streams to be integrated into the simulator, were developed. This appendix describes the data structures in more detail.

Data Structures

The following Pascal definitions define the data structures developed for the carbon and aqueous streams.

```
CONST
 MetalMax = 5;   {Maximum # of dissolved metals }
 SizMax   = 20;  {Maximum number of size classes }

TYPE
 SizRange    = 1..SizMax;
 MetalRange  = 0..MetalMax;

 ToMetals = ^Metals;
 ToSize   = ^Sizes;

 Adsorbents = ARRAY[MetalRange] OF ToSize;
 Metals     = ARRAY[MetalRange] OF REAL;
 Sizes      = ARRAY[SizRange]   OF REAL;
```

```
r7 = RECORD {Characteristic of Carbon streams}
    tonnes  : REAL;   {Solids flow kg/s}
    distrib : SIZES;  {Mass flow in each size class}
    adsorb  : ADSORBENTS; {Matrix describing loading}
    END;              {of each adsorbate in each}
                      {size class}

r6 = RECORD {Characteristics of Lixiviant streams}
    flow    : REAL;  {Aqueous flow kg/s}
    aq_metal : ToMetals; {Pointer to dissolved metal}
    END;
```

The capabilities of the data structures for each stream type will now be described.

Carbon Streams

The carbon streams record, *r7*, has three components which store all properties required by the model developed in Chapter 5. The *tonnes* field stores the flowrate (in kg/s) of the carbon in the stream, while the *distrib* field stores the mass flowrate (also in kg/s) of carbon in each size class.

The *adsorb* field is used to store the loadings on the carbon (in grams of adsorbate per ton of carbon) in each size class. Note that pointers are used for this field so that only the amount of memory space needed (which depends on how many dissolved species the user specifies) is allocated.

Aqueous Streams

The record *r6* is used to store the properties of aqueous streams. The *flow* field is used to store the aqueous flowrate (in kg/s) in the stream. The *aq_metal* field stores the concentration (in grams of dissolved species per ton of liquid) of the dissolved species in the aqueous phase. Again pointer techniques are used so that memory is utilised efficiently. For this case, the

memory is only allocated if the flowsheet being simulated contains hydrometallurgical units. That is, if provision needs to be made for dissolved species in aqueous streams.

Utility Procedures

In keeping with the MicroSim philosophy of user-friendliness for both the internal and external interfaces, utility procedures which make the data structures easy to use were added to the program. The function of these procedures will be briefly described.

Procedures *CarbonPut* and *CarbonValue* store and retrieve the loadings of the adsorbate *j* in size class *i* for a specified carbon stream.

Procedure *CarbonAdd* performs a mixing operation on two carbon streams. In addition, the *LixivAdd* procedure was extended to mix the dissolved species (if present) in aqueous streams.

The procedure *CarbonZero* sets all fields in the carbon record, *r7*, to zero. The *LixivZero* procedure was also extended to cope with the extensions made to the aqueous record, *r6*.

Conclusions

It was found that the methodology developed by Cilliers (1987) for the addition and extension of streams structures allowed data structures required for the simulation of leaching and adsorption to be added to MicroSim easily and efficiently. All changes required were localised and well defined, leading to faster development times and minimum debugging.

The success of the approach used for the MicroSim streams data structures can again be attributed mostly to the power and flexibility inherent to the way in which data structures can be created and manipulated by the Pascal language.

References

Adel, G.T. and Sastry, K.V.S (1982) Design Aspects Of A Mineral Processing Simulation Package. *17th International APCOM Symposium.* Johnson, T.B. and Barnes, R.J. ed. SME-AIME, New York, pp.681-692.

Ames, W.F. (1977) *Numerical Methods For Partial Differential Equations.* 2nd Edition. Academic Press, New York.

Atwood, G.R., and Foster, W.W. (1973) Transformation Of Bounded Variables In Simplex Optimization Techniques. *Ind. Eng. Chem. Process Des. Develop.*, Vol 12 , No 4, 1973, pp. 485-486.

Bailey, P. (1986a) Adsorption Plant Operations, Lecture 11. *SAIMM School : Design And Operation Of Plants For The Recovery Of Gold By Activated Carbon,* MINTEK, Randburg, February 1986.

Bailey, P. (1986b) Adsorption Design, Lecture 6. *SAIMM School : Design And Operation Of Plants For The Recovery Of Gold By Activated Carbon,* MINTEK, Randburg, February 1986.

Bailey, P. (1987) Application Of Activated Carbon To Gold Recovery. *In : The Extractive Metallurgy Of Gold In South Africa, Chapter 9.* G.G.Stanley, ed. SAIMM Monograph M7, Johannesburg.

Bhatia, S.K. and Perlmutter D.D. (1979) A Population Balance Approach To The Modelling Of Solid Phase Reactions. *AIChE Journal,* vol.25, no.2, pp.298-306.

Box, M.J. (1965) A New Method Of Constrained Optimisation And A Comparison With Other Methods. *Comput. J.* vol.8.

Brit, H.I. (1980) Multiphase Stream Structures in the ASPEN Process Simulator. *First Conference On Foundations Of Computer Aided Process Design,* AIChE, Heniker.

Brittan, M.I. (1975) Variable Activation Energy Model For Leaching Kinetics. *International Journal of Mineral Processing,* vol.2, pp.321-331.

Caceci, M.S. Cacheris, W.P. (1984) Fitting Curves To Data. *BYTE,* vol.9, no.5, pp.340-362.

CANMET - Canadian Centre for Mineral and Energy Technology (1981). *Simulated Processing of Ore and Coal Manual.* Ontario, Canada.

Carrier, G. Hodouin, D and Courchesne M. (1987) Dynamic Simulation Of The CIP Gold Recovery Process. *26th Annual CIM Conference Of Metallurgists.* Satter. R.S. Wyslouzi, D.M. and McDonald, G.W. ed. Winnipeg, Canada.

Carroll, J.M. (1983) Presentation And Form In User-Interface Architecture. *BYTE,* vol.8, no.12, pp.113-122.

Chan, W.K. and Prince, R.G.H. (1986) Application Of The Chain Rule Of Differentiation To Sequential Modular Flowsheet Optimization. *Computers & Chemical Engineering*, vol.10, no.3, pp.223-240.

Chen, H.S. and Stadtherr, M.A. (1985) A Simultaneous Modular Approach To Process Flowsheeting And Optimization. Part I : Theory And Implementation. *AIChE Journal*, vol.31, no.11, pp.1843-1856.

Chen, H.S. and Stadtherr, M.A. (1985) A Simultaneous Modular Approach To Process Flowsheeting And Optimization. Part II : Performance On Simulation Problems. *AIChE Journal*, vol.31, no.11. pp.1857-1867.

Chen, H.S. and Stadtherr, M.A. (1985) A Simultaneous Modular Approach To Process Flowsheeting And Optimization. Part III : Performance On Optimization Problems. *AIChE Journal*, vol.31, no.11, pp.1868-1881.

Christ, N.H. and Terrano A.E. (1986) A Micro-Based Supercomputer. *BYTE*, Vol.11, no.4, pp.145-160.

Cilliers, J.J.L (1987) Hydrometallurgical Simulation : A Viable Program Structure. M.Sc. Dissertation, University Of The Witwatersrand, Johannesburg.

Cho, E.H. Dixon, S. and Pitt, C.H. (1979) The Kinetics of Gold Cyanide Adsorption on Activated Charcoal. *Metallurgical Transactions B*, vol.10B, pp. 185-189.

Coles, D.R. (1982) Metallurgical Flowsheet Solution By A New Sequential Modular Programming Technique. *17th International APCOM Symposium.* Johnson, T.B. and Barnes, R.J. ed. SME-AIME, New York.

Conte, S.D. and De Boor, C. (1972) *Elementary Numerical Analysis.* McGraw-Hill Book Company, New York. 3rd Edition, pp.400.

Cutting, G.W. (1984) Simulation Package For Mineral Processing Flowsheet Design : Factors That Influence The Development Of A Practical System For Everyday Use. *18th International APCOM Symposium*, IMM, London.

Davidson, R.J. (1988) Anglo American Research Laboratory, Unpublished work.

Dixon, L.C.W. (1973) ACSIM - An Accelerated Constrained Simplex Technique. *Computer Aided Design*, vol.5, no.1, pp.22-32.

Dixon, S. Cho, E.H. and Pitt, C.H. (1978) The Interaction Between Gold Cyanide, Silver Cyanide And High Surface Area Charcoal. *AIChE Symposium Series*, vol.74, no.173.

Evans, L.B. Boston, J.F. Britt, H.I. Gallier, P.W. Gupta, P.K. Joseph, B. Mahalec, V. Ng, E. Seider, W.D. and Yagi, H. (1979) ASPEN : An Advanced System For Process Engineering. *12th Symp. Of Computer Applications in Chem. Eng.* Montreux.

Evans, L.B. (1980) Advances In Process Flowsheeting Systems. *First Conference On Foundations Of Computer Aided Process Design*, AIChE, Heniker.

Fehlberg, E. (1970) Klassiche runge-kutta-formeln vierter und niedrigerer ordnung mit schrittweiten-controlle und ihre anwendung auf warmeleitungsprobleme. *Computing J.* vol.6, pp.61-71.

Fleming, C.A. (1983) Recent Developments In Carbon-In-Pulp Technology In South Africa. *In : Hydrometallurgy : Research, development and plant practice.* Osseo-Assare, K. and Miller, J.D. ed. Proceedings, 3rd Symposium on Hydrometallurgy, Met. Soc. AIME, pp.839-858.

Fleming, C.A. and Nicol, M.J. (1984) The Adsorption Of Gold Cyanide Onto Activated Carbon. III. Factors Influencing The Rate Of Loading And The Equilibrium Capacity. *J. S. Afr. Inst. Min. Metall.*, vol.84, no.4, pp.85-93.

Ford, M.A. and King, R.P. (1984) The Simulation Of Ore-Dressing Plants. *International Journal of Mineral Processing*, vol.12, pp.285-304.

Gaines, B.R. (1981) The Technology Of Interaction - Dialogue Programming Rules. *Int. J. Man-Machine Studies*, vol.14, pp.133-150.

Gaines, B.R. and Shaw, M.L.G (1986) Foundations Of Dialog Engineering: The Development Of Human-Computer Interaction, Part II. *Int. J. Man-Machine Studies*, vol.24, pp.101-123.

Herbst, J.A. (1979) An Approach to the Modeling of Rate Processes in Multiparticle Systems. *In : Rate Processes in Extractive Metallurgy, Chapter 2.* Sohn, H.Y and Wadsworth, M.E. ed. Plenum Press, New York.

Herbst, J.A. and Fuerstenau, D.W. (1980) Scale-Up Procedures For Continuous Grinding Mill Design Using Population Balance Models. *Int. Journal Min. Proc.* vol.7, pp.1-31.

Hess, F.W. and Wiseman, D.M. (1984) Interactive Colour Graphics Process Analyser And Simulator For Mineral Concentrators. *18th International APCOM Symposium*, IMM, London.

Hulbert, H.M. and Katz, S. (1964) Some Problems In Particle Technology - A Statistical Mechanical Formulation. *Chem. Eng. Science*, vol.19. pp.555-574.

IMSL Library, The, (1982). Published by International Mathematical and Statistical Libraries Inc., Houston, Texas.

Izurieta, H. and Edwards, L.L. (1980) The General Energy and Material-Balance Computer System (GEMS) Applied To Hydrometallurgical Processes. *Journal Of Metals*, December, pp.23-31.

Johns, M.W. (1986) Model Application, Lecture 5, *SAIMM School : Design And Operation Of Plants For The Recovery Of Gold By Activated Carbon*, MINTEK, Randburg, February 1986.

Johns, M.W. (1987) The Simulation Of Gold Adsorption By Carbon Using A Film Diffusion Model. M.Sc. Dissertation. University Of The Witwatersrand, Johannesburg.

Kavetsky, A. and Mckee, D.J. (1984) Analysis And Design Of Industrial Grinding And Classification Circuits By Use Of Computer Simulation. *18th International APCOM Symposium*, IMM, London.

Kennedy, T.C.S. (1974) The Design Of Interactive Procedures For Man-Machine Communication. *Int. J. Man-Machine Studies,* vol.6, pp.309-334.

King, R.P. (1974) Simulation Of Flotation Plants. *Presented at 102nd. Meeting of A.I.M.E.*, Dallas.

King, R.P. (1984) Application Of Computer Simulation Techniques In Ore Dressing Plant Design And Operation. *SAIMM Colloquium, Control and Simulation In The Minerals Industry*, Johannesburg.

King, R.P. (1985) MODSIM : A Modular Method For The Design, Balancing And Simulation Of Ore Dressing Plant Flowsheets. *Report G9*, Department Of Metallurgy, University Of The Witwatersrand, Johannesburg.

King, R.P. and Woollacott, L.C. (1986) Computer Aided-Engineering in Minerals Processing, Department of Metallurgy, University of the Witwatersrand, Johannesburg.

Kiparissides, C. and Ponnuswamy, S.R. (1981) Application Of Population Balance Equations To Latex Reactors. *Chem. Eng. Commun.* vol.10, pp.283-291.

Lampson, B.W. (1984) Hints For Computer System Design. *IEEE Software*, vol.1, no.1, pp.11-28.

Laxen, P. (1986) General Introduction, Lecture 1, *SAIMM School : Design And Operation Of Plants For The Recovery Of Gold By Activated Carbon*, MINTEK, Randburg, February 1986.

Loveday, B.K. Robinson, C.G. and Paynter, J.C. (1970) A Model For The Prediction Of Leaching Curves And For The Determination Of Activation Energies In The Leaching Of A Low-Grade Ore. *1st National meeting, S.A. Inst. Chem. Eng.* vol.2, No.5.3 pp.1-12.

Lynch, A.J. (1984) Computers In Mineral Processing - The First Twenty-Five Years. *18th International APCOM Symposium*, IMM, London.

Maguire, M. (1982) An Evaluation Of Published Recommendations On The Design Of Man-Computer Dialogues. *Int. J. Man-Machine Studies,* vol.16, pp.237-261.

Mckee, D.J. (1984) The Use Of Simulation In The Operation And Design Of Mineral Processing Plants. *MINTEK 50,* South African Institute Of Mining and Metallurgy, Johannesburg.

Menne, D. (1982a) Predicting And Assessing Carbon-In-Pulp Circuit Performance. *XIV International Mineral Processing Congress,* CIM, Toronto Canada.

Menne, D. (1982b) Optimization Of Full-Scale Circuits For The Carbon-In-Pulp Recovery Of Gold. Glen, H.W. ed. *Proceedings, 12th CMMI Congress,* SAIMM, Johannesburg 1982.

Miller, L.A. and Thomas, J. (1977) Behavioural Issues In The Use Of Interactive Systems. *Int. J. Man-Machine Studies,* vol.9, pp.509-536.

Mular, A.L. and Herbst, J.A. (1980) Digital Simulation : An Aid For Mineral Processing Plant Design. *In : Mineral Processing Plant Design, Chapter 14.* Second edition, Mular, A.L. and Bhappu, R.B. ed. Published by A.I.M.E.

Nicol, M.J. (1986) Carbon In Leach, Lecture 13, *SAIMM School : Design And Operation Of Plants For The Recovery Of Gold By Activated Carbon,* MINTEK, Randburg, February 1986.

Nicol, M.J. Fleming, C.A. and Cromberge, G. (1984) The Adsorption Of Gold Cyanide Onto Activated Carbon. I. The Kinetics Of Adsorption From Pulps. *J. S. Afr. Inst. Min. Metall.,* vol.84, no.2, pp.50-54.

Preece, P. (1986) The making of PFG and PIG. *The Chemical Engineer,* June, pp.87-88.

Randolph, A.D. and Larson, M. (1971) *Theory Of Particulate Processes,* Academic Press, New York.

Rich, E. (1983) Users Are Individuals : Individualizing User Models. *Int. J. Man-Machine Studies,* vol.18, pp.199-214.

Ritchie, I.C. (1983) A Methodology For The Design And Evaluation Of Minerals Extraction Processes. *Ph.D. Thesis,* Imperial College.

Ritchie, I.C. and Spencer, R. (1984) Economic Evaluation Of Minerals Extraction Processes By Use Of A Flexible Process Simulation Program. *18th International APCOM Symposium,* IMM, London.

Sastry, K.V.S. and Adel, G.T. (1984) A Survey of Computer Simulation Software for Mineral Processing Systems. *Proceedings of Control 84,* AIME.

Sepulveda, J.E. (1981) A Fundamental Approach To The Design And Optimization Of Multistage, Agitated Leaching Systems. *Ph.D. Thesis*. University Of Utah.

Stange, W. Cilliers, J.J. and King, R.P. (1988) MicroSim V3.0 User Manual. *Report GEN 7*, Department Of Metallurgy and Materials Engineering, University of the Witwatersrand, Johannesburg.

Stange, W. Cilliers, J.J. and King, R.P. (1988) MicroSim V3.0 Technical Reference Guide. *Report GEN 10*, Department Of Metallurgy and Materials Engineering, University of the Witwatersrand, Johannesburg.

Sutherland, D.N. (1977) An Appreciation Of Galena Concentration Using A Steady-State Flotation Model. *Int. Journal Min. Proc.* vol.4. pp.149-162.

Sutherland, D.N. and Weller, K.R. (1983) The Use Of Computers In The Analysis And Steady State Optimization Of Metallurgical Concentrators. *Aus. I.M.M. Southern Queensland Branch*, Computers in Mining Symposium.

Thomas, B. (1986) The Runge-Kutta Methods. *BYTE*, vol.11, no.4, pp.191-210.

Tucker, P. (1986) Development Of A Simulation Package For Flowsheet Design In The Minerals Processing Industries. *Scand. J. Metallurgy*, vol.15, pp.143-150.

Valentas, K.J. and Amundson, N.R. (1966) Breakage And Coalescence In Dispersed Phase Systems. I. & E.C. Fundamentals, vol.5, no.4, pp.533-542.

Valentas, K.J. and Amundson, N.R. (1966) Analysis Of Breakage In Dispersed Phase Systems. I. & E.C. Fundamentals, vol.5, no.2, pp.271-279.

Van Deventer, J.S.J. (1984) Kinetic Model For The Adsorption Of Metal Cyanides On Activated Carbon. *Ph.D. Thesis*, University of Stellenbosch. 337 pp.

Westerberg, A.W. (1981) Optimization in Computer Aided Design. *Foudations Of Computer Aided Design*, Mah, R.S.H and Seider, W.D. ed. vol.1, Engineering Foundation, New York.

Whyte, R.M., Morris, D.J., Smith, P. and Davidson, R.J. (1987) The Development of Interstage Screens for Carbon in Pulp Applications Within the Gold Mines of the Anglo American Corporation. Presented at *Separation Processes In Hydrometallurg*, Davies, G. ed. Society Of Chemical Industries, Manchester 1987.

Williams, D.F. and Glasser, D. (1985) The Modelling And Simulation Of Processes For The Adsorption Of Gold By Activated Charcoal. *J. S. Afr. Inst. Min. Metall.*, vol.85, no.8, pp. 237-243.

Woollacott, L.C. (1984) End-User Oriented Design Of A Flowsheeting Computer Aid In Minerals Processing. *CAD84, 6th International Conference and Exhibition on Computers In Design Engineering*. Wexler, J. ed. pp. 348-359. Organized by Computer-Aided Design Journal.

Woollacott, L.C. (1986) Department of Metallurgy and Materials Engineering, University of the Witwatersrand, Personal communication