PREDICTION OF LEACHATE GENERATION FROM MINERALS PROCESSING WASTE DEPOSITS

by

Graham Mark Davies

Dissertation submitted to the University of Cape Town
in fulfilment of the requirements for the degree of
Master of Science in Engineering

Department of Chemical Engineering
University of Cape Town
Rondebosch
7700.

September 1995

The University of Cape Town has been given the right to reproduce this thesis in whole or in part. Copyright is held by the author.

The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

University of Cape Town

Synopsis.

The minerals processing industry in South Africa produces significant tonnages of waste material which are disposed of commonly in dedicated waste depositories. These deposits pose a potential to pollute the environment if leachate is generated within the deposit and released to the surroundings. Leachate generation is generally investigated using laboratory columnar experiments which attempt to mimic the physical and chemical processes which occur in the deposit. These experiments, termed lysimeter experiments, are time consuming in that they typically last for at least a few months and can last for up to three years. Lysimeter experiments are also costly to conduct. Because of restrictions such as these, relatively few deposits have been characterised to determine the leachate which they generate and thus the risk which they pose to the environment.

There is an urgent need to be able to estimate the environmental risks associated with existing waste deposits. The first step towards assessing this risk would be an ability to predict leachate generation within a specific deposit. Such an ability could be used to identify which of the existing deposits produce significant leachate and thus pose a potential hazard to the environment. Equally, if leachate generation from new deposits could be estimated as a function of waste material and characteristics of the waste deposit, this information could be used to improve the engineering design of waste deposits.

The work presented in this thesis involved identifying suitable modelling strategies which could be used to determine leachate generation within waste deposits which contain waste material typical of that produced by the minerals processing industry. Two modelling strategies have been investigated. The first modelling strategy involved a macroscopic model in which all effects such as intrinsic chemical kinetics, intra-particle diffusion, external mass transfer and hydrodynamic considerations are lumped into a single parameter. The result of this approach is an effective reaction rate for the release of hazardous constituents from a volume element of the waste deposit. The effective reaction rate is determined by fitting the model to experimental data based on lysimeter tests. The main advantage of this model is that it eliminates the need for a detailed understanding of the individual factors which contribute to leachate generation. This model was investigated both for its inherent simplicity and for use in cases where insufficient information with respect to the intrinsic chemical reaction rates, intra-particle diffusion, external mass transfer or hydrodynamic aspects exist. The main disadvantage of this model is that it has a limited predictive ability in that the individual significance of any one factor which contributes to leachate generation cannot be determined. For this reason a second, more detailed model, termed the heterogenous columnar model, has also been investigated.

The heterogenous columnar model describes the release of hazardous constituents at the single particle level and relates this information to the overall leachate generation within the deposit. This is achieved by calculating the release of hazardous constituents from the size distribution of particles to the bulk fluid between these particles. The release of hazardous constituents from individual particles is determined by making use of a particle-scale chemical reaction model. This particle-scale model is sufficiently detailed to be able to determine the relative contribution to the overall release of hazardous constituents from the particles of intrinsic chemical kinetics of the reactions to the effects of diffusion of the fluid reagent into each particle. The heterogenous columnar model can also be used to determine whether the effective rate of release of hazardous constituents from the particles (which include intrinsic kinetic and diffusional contributions) or the flow of fluid reagent through the deposit limits the release of hazardous constituents from the deposit. This information can be used to determine the main factors which affect the release of hazardous constituents from waste deposits and can thus be used to improve the design of waste deposits.

Probably the most important attribute of the heterogenous columnar model is that methods have been investigated to determine the model parameters from a simple continuously stirred tank reactor (CSTR) type experiment.

The ability of the heterogenous columnar model to predict leaching behaviour has been investigated using data on precious metal leaching found in the literature. The results are encouraging in that the model can accurately predict the leaching behaviour of precious metals. A preliminary investigation into determining suitable particle-scale model parameters for a sample of waste from a CSTR experiment has been conducted. This too has yielded encouraging results. However, the application of using the heterogenous columnar model using these parameters to describe leachate generation within waste deposits or lysimeter experiments still needs to be demonstrated. Once the heterogenous columnar model has been verified against data pertaining to leachate generation from a waste deposit it may start to provide the minerals processing industry with the information which it so desperately requires in order to dispose of wastes in a manner which minimises the impact on the environment.

Acknowledgements.

I would like to thank Dr J.G. Petrie for his guidance and assistance in this work. I believe that no graduate student can learn to do research effectively on their own and I am grateful for what I have learnt from Dr Petrie. In particular Dr Petrie has taught me the level of rigour required in research and the ability to constructively question any results obtained to determine their significance and accuracy. In addition to this, working under Dr Petrie I have become aware of the pressing environmental issues facing the chemical and minerals processing industries in South Africa. I believe that such an awareness will be invaluable to me as a chemical engineer.

I am also grateful for having had the opportunity to work in the 'Greenhouse' in the Department of Chemical Engineering at the University of Cape Town. Working with such a diverse and talented group of people has been most stimulating and has kept me on my toes. The weekend away at Wilderness, the mini investigation into possible methods to clean up oil on the beaches and the commissioning of the 'fish tank' are only a few of the many things which made working in the group so interesting.

Last, but not least, I wish to gratefully acknowledge the financial assistance of MINTEK. Without this assistance this project would not have been possible.

Table of Contents.

List of Tables.

List of Figures.

Nomenclature.

Roman Letters

b_i	stoichiometry number, kg solid reactant i/kg fluid reagent
C_A	concentration of reagent A, kg A/m_f^3
C_{ab}	concentration of reagent A external to particle, kg A/m_f^3
$C_{E;O}$	initial extractable grade of solid reactant i, kg i/kg ore
C_i	concentration of dissolved species i, kg i/m_f^3
C_{ib}	bulk concentration of dissolved species i, kg i/m_f^3
C_{pi}	grade of solid reactant i within particle, kg i/kg ore
$C_{p;O}$	initial grade of solid reactant i within particle, kg i/kg ore
C_{si}	grade of solid reactant i on particle surface, kg i/kg ore
C_{siO}	initial grade of solid reactant i on particle surface, kg i/kg ore
$Col_{\%\ Sat}$	fraction of the void space within the deposit filled with fluid, dimensionless
D_{ae}	effective pore diffusivity of reagent A, m_f^3/m_p s
D_{ie}	effective pore diffusivity of dissolved species i, m_f^3/m_p s
DG1	ratio of the actual fluid percolation velocity to a reference percolation velocity, dimensionless
$DG2_i$	ratio of the chemical reaction rate i at z=0 to the rate of fluid reagent replenishment, dimensionless
$DG3_i$	dimensionless stoichiometric ratio which indicates the fluid reagent strength relative to the solid reactant concentration i within the deposit
E_i	extraction of dissolved species i, dimensionless
k_{pi}	rate constant of solid reactant i within particle, kg i/kg ore s$[C_p]^{\sigma}$ $[C_A]$
k_{si}	rate constant of solid reactant i on particle surface, kg i/m_p^2 s $[C_s]^{\sigma}$ $[C_A]$
L	deposit depth, m
M	number of size classes in the size distribution of particles, dimensionless
n	number of solid reactants
N_{re}	pellet flow Reynolds number, dimensionless
q	volumetric flow rate of fluid into the deposit, m^3/s
r	radius, m
r_{Ai}	rate of 'production' of fluid reagent A by reaction with solid species i, kg/m^2s
R	particle radius, m
t	time, s
T	reference space time for the deposit,s
u_s	superficial bulk flow velocity, m_f^3/m_h^2 s
u^*	reference fluid velocity (percolation velocity), m/s
X_i	fractional conversion of solid reactant i, dimensionless
z	depth, cm

Greek Letters

α	dimensionless concentration of reagent A
α_b	dimensionless concentration of reagent A external to particle
β_i	reagent strength parameter relative to solid reactant i, dimensionless
γ	modulus in steady-state model
δi	ratio of diffusivity of dissolved species i to reagent A, dimensionless
ϵ_b	bulk solution volume fraction
ϵ_{Col}	heap void fraction
ϵ_o	ore porosity, cm_f^3/cm_p^3
ζ	dimensionless depth
Υ_k	ratio of the volume of the particles in size class k to the volume of the particles in the reference size class, dimensionless
η	effectiveness factor for solid reactant i, dimensionless
θ	dimensionless flow time
κ_{pi}	Damkohler II number for solid reactant i within particle, dimensionless
κ_{si}	Damkohler II number for solid reactant i on particle surface, dimensionless
λ_i	surface fraction of solid reactant i, dimensionless
ν	ratio of volume bulk fluid to fluid in particle pores, dimensionless
ξ	dimensionless radius
Ξ	dimensionless particle radius
ρ_o	ore density, g ore/cm^3 ore
σ_{pi}	dimensionless grade of solid reactant i within particle
σ_{si}	dimensionless grade of solid reactant i on particle surface
τ	dimensionless diffusion time
τ'	dimensionless flow time for the deposit
ϕ_{pi}	reaction order for solid reactant i within particle, dimensionless
ϕ_{si}	reaction oredr for solid reactant i on particle surface, dimensionless
χ_i	dimensionless concentration of dissolved species i
χ_{ib}	dimensionless bulk concentration of dissolved species i
ζ_i	ratio of the surface grade of hazardous constituents to the bulk grade of hazardous constituents within the particle, dimensionless

Chapter 1. Introduction.

Increasing environmental awareness is challenging both the chemical and minerals processing industries to examine the type and quantity of the waste which they produce, and to critically assess the methods used to dispose of this waste. Presently new technologies are being developed to minimise the overall waste produced. Since the total eradication of wastes is not feasible from both thermodynamic and economic perspectives, industry is still faced with the problem of safely disposing of chemical and mineral wastes in such a manner as to ensure their minimal impact on the environment.

One industry in particular which is being forced to assess the impacts of its wastes on the environment is the minerals processing industry. This industry produces high tonnages of waste streams which include slimes, slags, bag house dusts and leached ores. Slimes consist of solid particles suspended in fluid from the extraction process, while slags, dusts and the leached ores are all granular solid waste particles. All these solid particles contain leachable components in addition to significant quantities of inert material.

These wastes are presently disposed of in various ways. In each case attempts are made to minimise the impact on the environment by containing the waste and any leachate generated within the site of disposal. Slimes are usually disposed of in slimes dams or tailings impoundments. These structures are designed to allow the process fluid to drain from the solid particles and to be returned to the process. The solid particles accumulate in the dams or tailings impoundments until the useful capacity of these structures have been exhausted. Once this stage has been reached, the remaining process fluid is allowed to drain and the dams and tailings impoundments resemble deposits of very fine granular material on closure. There is a possibility for leachate to be generated even after closure when rain water periodically percolates through the system. For this reason these structures not only need to be maintained during their useful lifespan but also for extended periods of time after their closure.

Granular solid wastes in turn are usually disposed of in dedicated waste deposits. These deposits are engineered entities which may or may not have clay or polymer liners to prevent any leachate which may be generated from penetrating the environment. In granular deposits leachate may be generated from rain or ground water which penetrates and percolates through the system. In cases where significant leachate is generated, leachate collection sumps and treatment processes are provided. As in the case with slimes dams and tailings impoundments, these deposits need to be maintained well after they have been filled to capacity with waste. Both slimes dams and dedicated deposits have hazards associated with them. The most obvious hazard is their potential to pollute the environment if they are not sufficiently maintained, especially for the extended periods of time after they have been filled with waste. Further, the liners used in these systems are not perfect and may deteriorate with time, allowing potentially harmful leachate to enter the environment.

A more attractive option which eliminates the need for liners, leachate collection sumps

and treatment systems is where the waste is physically or chemically pretreated in such a manner to either totally eliminate leachate generation, or to reduce it to release rates which the natural environment can assimilate. Although deposits containing stabilised wastes have the advantages of not requiring maintenance and eliminate the possibility of liner failure, their long term stability needs to be assessed. In this respect, the stability of deposits over at least a few decades needs to determined as well as the stability of deposits over much greater time periods. These time periods are consistent with the time scales being considered in Life Cycle Assessment of waste disposal strategies [Clift 1995].

Since leachate generation and subsequent transport through granular waste deposits represents a significant environmental hazard, an ability to predict these phenomena would be most useful. In the case of slimes dams and dedicated waste deposits, such an ability would enable process engineers to calculate the leachate which would require treating and which would penetrate the environment in the case of a liner failure. This information could then be used to assess the potential environmental hazards of such waste disposal strategies and would form part of a rigorous risk assessment of landfill practice. Since upstream processes directly affect the properties of the waste which needs to be disposed of, an ability to predict the leachate generation as a function of the waste properties would enable process engineers to investigate upstream processing strategies which would produce wastes which limit the production of leachate. This would enable the processing system to be optimised to limit the impact of its waste on the environment. Further, an ability to predict the leachate generation from deposits which contain stabilised wastes could be used to determine the effectiveness of the stabilisation procedures. In summary, an ability to predict leachate generation and subsequent transport through granular waste deposits would start to provide the minerals processing industry with the tools which it requires in order to determine the most appropriate disposal strategy for its wastes.

In order to predict leachate generation and transport through granular waste deposits, the major physical and chemical processes involved need to be identified. In this work it has been assumed that the leachate is generated by the reaction of a fluid reagent with the granular solid waste particles. These particles usually contain several potentially hazardous and leachable constituents in addition to other components which exhibit buffering capacity within an inert matrix. Here buffering capacity refers to any component which will provide a neutralising capacity to acid which flows through the system. As an example, many wastes contain an alkali silicate matrix which behaves as a buffering component. Particle characteristics which affect the release of hazardous constituents include particle size, shape and hazardous constituent location. Because of the nature of waste streams, these characteristics vary significantly both between the types of wastes as well as between individual waste particles.

Fluid reagent flow characteristics also play an important role in both the release and subsequent transport of hazardous constituents. The flow patterns are important because they determine the extent to which the fluid is in contact with the individual waste particles. In this work it has been assumed that once hazardous constituents have been released they are transported by the bulk convection of the fluid reagent. Thus the fluid flow patterns significantly affect the mobility of the hazardous constituents. Other physical processes which affect this mobility include adsorption and desorption reactions of the released hazardous constituents onto the surfaces of the solid waste particles.

Closely associated with fluid flow characteristics is the degree of saturation within the deposit. This is an important consideration because it determines the amount of fluid reagent available for reaction as well as the available wetted surface area of the particles. The influence of the degree of fluid saturation is particularly important in the analysis of South African waste deposits which are rarely saturated with fluid. This is in stark contrast to most North American and European deposits which are usually saturated.

The work presented in this thesis is a start towards determining suitable strategies for the prediction of leachate generation and its subsequent transport through waste deposits which contain granular wastes typical of those produced by the minerals processing industry.

1.1 Statement of the Objectives.

The objectives of the thesis can be summarised as follows:

- To derive physical models which describe the generation and subsequent transportation of leachate within waste bodies which contain solid granular material with several leachable constituents.

- To represent these models mathematically and to propose suitable strategies for their solution.

- To code these models into suitable computer routines using the appropriate solution strategies.

- To identify suitable experimental techniques to quantify any model parameters which are required.

- To explore methods to verify the models against experimental data which is typically in the form of laboratory column experiments.

1.2 Research Methodology.

The literature has been reviewed critically to determine the state of the art with respect to leachate generation and mobility within granular waste deposits. As will be shown in the literature review, none of the existing modelling strategies include a sufficient level of complexity required to either predict the leachate generation or to yield sufficient information about the system which could be used to engineer improved disposal practices. Noting similarities between the leaching of hazardous constituents from waste particles and precious metal heap leaching operations, the modelling strategies used for precious metal heap leaching have also been investigated. As will be shown in the literature review, these too do not contain sufficient detail to adequately model leachate

generation when dealing with waste particles as opposed to mineral ores. Despite this, it seemed feasible that these modelling strategies could be extended to include the required complexities and this forms the major part of this work.

A second area of existing modelling strategies which has value in attempting to address the complexities associated with leachate generation and mobility in waste bodies is the design of trickle bed reactors. These models too have been reviewed and included in the present development.

The research methodology adopted in this work was to identify or develop models which are sufficiently detailed to predict leachate generation and mobility within granular waste deposits and yet not too complex to preclude implementation. Factors which often limit the usefulness of models include large computational loads, a large number of parameters which need to be determined or when required inputs to the model cannot be determined experimentally. An example of a model input which would be difficult to determine experimentally would be if the model required a full speciation of the components within the waste particles.

The first model investigated was a macroscopic model in which the effects of chemical kinetics, fluid flow characteristics, wetting efficiency and fluid saturation were 'lumped' together into a single parameter. The macroscopic, lumped parameter model describes the leachate concentrations within a granular waste deposit as well as the leachate concentration emanating from the base of a deposit as a function of time. The main reason for investigating this model was its inherent simplicity and its ability to characterise granular deposits as reacting entities. The main disadvantages of this model stem from the lumped parameter used in the model. Because this parameter lumps together the effects of chemical kinetics, fluid flow characteristics, wetting efficiency and fluid saturation on the release of hazardous constituents, it is not possible to isolate the individual contributions of any of these factors. This limits the use of this model for waste deposit design because the main characteristics of a deposit which can be altered include the chemical kinetics and hydrodynamic aspects. Without a knowledge of which particular aspect is dominant in causing contaminant release it is impossible to engineer better deposits other than on a trial and error basis. The lumped parameter is specific to a particular situation and must be determined from an appropriate laboratory column experiment. This is a severe limitation in that a column experiment, usually referred to as a lysimeter experiment, is required for each different waste and fluid flow scenario envisaged. The disadvantages of lysimeter experiments are the costs involved with such experiments and the long duration of the experiments which typically last for at least a few months and can continue up to two or three years. Because of these limitations it was decided to investigate a model which would be capable of isolating the chemical and hydrodynamic effects on the release of hazardous constituents and which, if possible, obviated the need for lysimeter experiments to determine the model parameters.

This leads to the second model investigated. This model, termed the heterogenous columnar model, also describes the leachate concentrations within a waste deposit as well as the concentration of the leachate emanating from the base of the deposit as a function of time. The main difference is that this model is neither a macroscopic nor a lumped parameter model. Instead, this model calculates the leachate generated as a function of

the individual particles present in the waste deposit as well as the fluid flow characteristics, wetting efficiency and fluid saturation. The other main advantage of this model is that the required model parameters can be determined from appropriate *CSTR* type experiments on small samples of the waste material. The time savings incurred by using CSTR type experiments over lysimeter experiments are significant.

The limitations of the heterogenous columnar model as it is presented in this thesis include the fact that only perfect plug flow of fluid through the deposit has been considered, interphase mass transfer limitations or resistances between the bulk fluid reagent and particle surfaces have been neglected, and, adsorption and desorption of released hazardous components has not been considered. It is important to note that more complex flow scenarios, interphase mass transfer and adsorption and desorption reactions can be included into the present model. The model has been specifically coded in a modular manner to allow the incorporation of these aspects.

1.3 Thesis Layout.

The thesis commences with a review of relevant considerations when modelling leachate generation and mobility in granular waste deposits. In this section, more detailed information on particulate and hydrodynamic characteristics of the waste and waste deposit is presented. This is followed by a critical review of modelling strategies with respect to the release of hazardous constituents from granular waste deposits. This review covers existing leachate generation modelling strategies, heap leaching modelling strategies and trickle bed reactor design strategies.

Chapter 3 presents the model development for the macroscopic, lumped parameter model. This chapter includes the derivation of the equations required for this model, an investigation into an appropriate solution strategy and details with respect to computer routines which have been written to implement the solution strategy. Also included in this chapter are details of how to fit the model to lysimeter data. The limitations of the macroscopic, lumped parameter model indicated that a more complete, particle scale model would be required to determine sufficient detail about the system in order to facilitate active engineering of the deposit to limit the release of hazardous constituents from granular waste deposits.

The more complete model, which determines the release of hazardous constituents from individual waste particles and relates this information to the overall waste deposit performance, is developed in Chapters 4 to 6. Chapter 4 is a summary of Dixon's [1992] chemical reaction model which describes the progression of reactions within an individual waste particle. As will be shown, this model has limited applicability on its own. Waste deposits consist of a size distribution of particles and a finite amount of fluid reagent. The performance of this system, rather than that of a single particle, is the desired end product. For this reason the chemical reaction model of Dixon [1992] was extended to form a suitable CSTR model in Chapter 5 and a heterogenous columnar type model in Chapter 6. A CSTR type model has been included due to the fact that the model

parameters required in the heterogenous columnar model can be determined from a CSTR type experiment. The heterogenous, columnar model has been verified against experimental data presented in a paper by Roman *et al.* [1974]. The details of this verification have also been included in Chapter 6.

The final chapter presents a summary of the conclusions and recommendations which can be made from this work.

Chapter 2. Relevant Considerations when Modelling Leachate Generation and Mobility.

This chapter discusses important aspects of leachate generation and mobility within granular waste deposits. As discussed briefly in the introduction, both the physical composition of the waste particles and the fluid flow characteristics within the waste deposit affect the manner in which hazardous constituents are released from the waste. The first section of this chapter considers these particulate and hydrodynamic considerations which are important in modelling leachate generation. In this section, information from the literature on trickle bed reactor design has been used to help understand the complex flow patterns of fluid within granular waste deposits.

The existing leachate generation and mobility models which can be found in the literature have been reviewed. This section critically assesses which of these models have any potential to predict leachate generation within deposits which contain granular wastes typical of those produced by the minerals processing industry.

As alluded to in the introduction, heap leaching processes, designed to extract precious metals from ores, share similarities to leachate generation within waste deposits. In both cases the leachate is generated from the reaction of a fluid reagent with solid particles. Both processes have a similar dependence on the solid reactant location within the ore or waste particles and the fluid flow characteristics. The major difference between the two processes is that the generation of leachate is desired in the case of heap leaching and undesired in the leaching of hazardous constituents. This leads to significant differences between the systems. The fluid reagent used in heap leaching for example is usually chosen in such a manner as to selectively extract one desired component from the ore matrix. This is in contrast to the leaching of hazardous constituents where several contaminants are leached simultaneously by fluid percolating through the deposit. Further, heap leaching processes are designed to optimise the wetted area of the particles. When waste deposits are designed, if any consideration is given to particle wetting, it would be to limit the surface area of the particles in contact with the fluid percolating through the deposit. Despite these differences between the two systems, the models for heap leaching of precious metals are a good starting point for developing suitable models to predict leachate generation within waste deposits. For this reason the general strategy behind these models is presented.

The final section of this chapter deals with modelling leachate generation and mobility within unsaturated deposits.

2.1 Particulate and Hydrodynamic Considerations.

2.1.1 Particulate Considerations.

Granular solid wastes produced by the minerals processing industry vary considerably from one waste stream to the next in terms of the number and type of reactive

components present within the particles. Even within a single waste stream, the spatial distribution of the components may vary between the different sized particles. These differences need to be investigated to determine their significance in the modelling of leachate generation.

Number and type of reactive components present within the waste particles.

The most desirable form of waste is a waste in which the constituent particles are totally inert to fluid reagent percolating through the deposit. In this case there would be no reactive components and leachate could not be generated. Although wastes of this nature are the goal of pretreatment and stabilisation processes, they are seldom achieved in reality. The simplest waste type which is capable of producing leachate is the waste which contains a single reactive component. Such wastes are very uncommon. Far more probable however are wastes which contain two or more reactive components which display significant leaching potential.

The reactive components in waste particles can either be hazardous constituents or other components, termed buffering components, which merely consume fluid reagent but do not release any hazardous constituents. Typical hazardous constituents in minerals processing wastes are heavy metals while the buffering capacity is usually due to fluxing agents.

It is difficult to provide a general description of the relative amounts of leachable hazardous constituents, buffering and inert material within waste particles. Leached ores or tailings residues, for example, contain a relatively high fraction of inert material with trace amounts of leachable hazardous constituents and buffering material. This is in contrast to a slag waste stream which has a high buffering material content in addition to leachable hazardous constituents and inert material.

Although there may be several potentially leachable hazardous constituents within waste particles, usually only a few occur in sufficient quantities or are sufficiently reactive to be of concern. For example, although wastes from the minerals processing industry often contain several heavy metals, usually only one or two of these are in significant concentrations to pose a hazard if leached out of the waste.

Hazardous Constituent location within waste particles.

Hazardous constituents present in waste particles may either be distributed homogenously throughout the particle or be concentrated onto its external surface. These differences in hazardous constituent location are both waste specific and particle size specific and are usually a result of the conditions under which the waste was produced. As an example of hazardous constituent location as a function of waste type, compare the hazardous constituent location in a slag waste stream and an electric arc furnace dust. The hazardous constituents present in a slag waste stream would be expected to be distributed reasonably homogenously throughout the particle. This differs

markedly from an electric arc furnace dust which usually exhibits surface concentrations of heavy metals much higher than the bulk concentrations within the particles [Driesinger *et al.* 1990].

Turning to the case of hazardous constituent location as a function of particle size, work conducted at the University of Cape Town on specific ferro-alloy waste products indicated that different sized particles exhibit different surface hazardous constituent concentrations [von Blottnitz 1994].

2.1.2 Hydrodynamic Considerations.

Hydrodynamic aspects are concerned with the transport of fluid through waste deposits and the manner in which the fluid and solid waste are contacted. Hydrodynamic aspects, for granular waste deposits as well as for solidified monolithic structures, have been investigated in order to determine their significance on the modelling of leachate generation. Monolithic structures are obtained when granular wastes are mixed with suitable binding agents, cement being one example, in order to physically bind the particles together. Even although the particles are physically bound together, monolithic structure exhibit a continuous pore structure through which fluid can enter the structure. Although monolithic structures are not directly within the brief of this work, the fluid flow patterns associated with these structures can be considered as a limiting case for granular waste deposits which contain very fine, densely packed material.

Fluid flow patterns within granular waste deposits.

Fluid flow patterns through granular waste deposits range from near perfect plug flow to highly irregular flow patterns involving a few preferential flow paths for the fluid reagent. Knowledge of the flow patterns is important because it is related directly to the solid-liquid contacting efficiency.

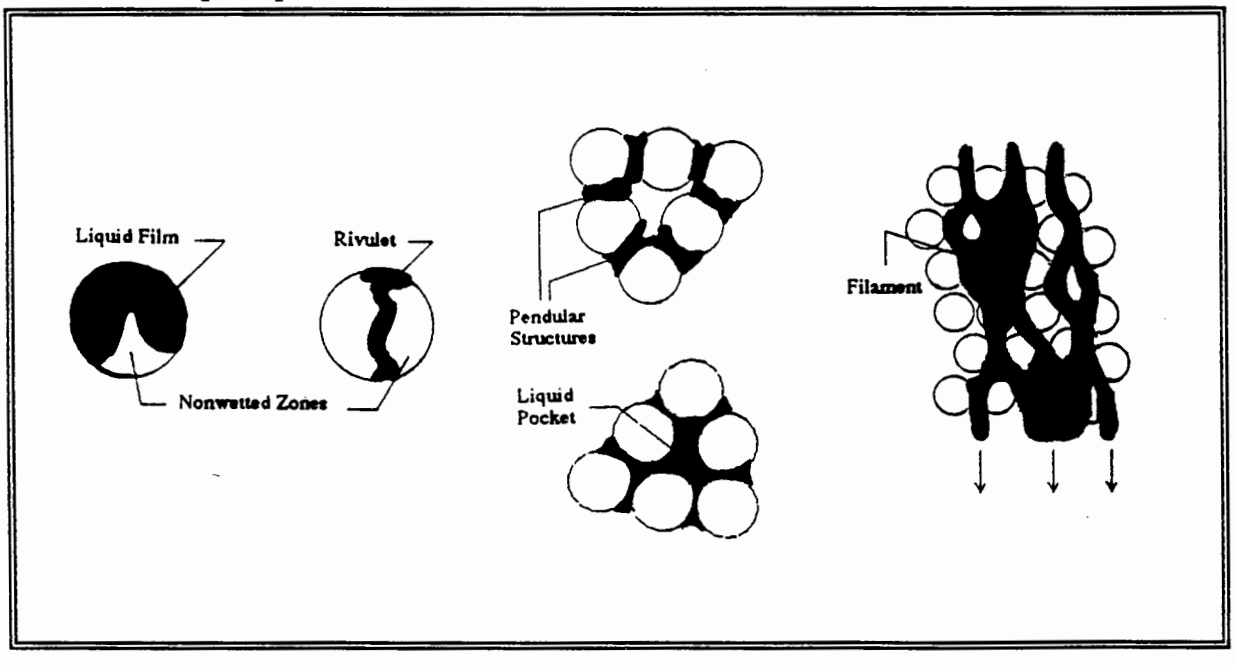
Trickle bed reactors, common within the process engineering community, are packed bed reactors through which one or more fluid is allowed to flow. These reactors exhibit a wide variety of fluid flow regimes which range from strictly trickle flow, in which a single liquid flows downward through the reactor under the influence of gravity, to pulse and foaming flow, caused by high flowrates of gases and liquids through these reactors. Since the physical situation in a trickle bed reactor operating in trickling flow regime is comparable to fluid percolation through granular waste deposits, a knowledge of the flow patterns in these reactors can be used to understand the fluid flow patterns in granular waste deposits.

A review of the literature on trickle bed reactors shows that, traditionally, with respect to hydrodynamic considerations, only two quantities, the liquid holdup and wetting efficiency, have been measured and correlated [Columbo *et al.* 1976; Schwartz *et al.* 1976; Mills and Dudukovic 1981; Ramachandran *et al.* 1986; Gianetto and Specchia 1992]. Although these two quantities are useful to describe the overall bulk effects of the

hydrodynamic interactions, they offer little insight into the flow patterns.

Until fairly recently, relatively little was understood about the flow patterns in trickling flow trickle bed reactors due to the difficulties involved with determining these patterns. A novel use for computer assisted tomography (CAT scans) has eliminated most of these difficulties and enabled Lutran *et al.* [1991] to investigate the flow patterns in a packed bed of equi-sized glass spheres. At the particle level, they identified two distinct types of fluid flow which they termed 'film' and 'rivulet' flows. Film flow represents the case where the fluid tends to cover most of the particle surface area, while rivulet flow describes the case where the flow of fluid over a particle is restricted to a narrow band. They also identified pendular structures and liquid pockets at the microscopic level. Pendular structures reside at the contact points between spheres while liquid pockets fill the pore space between spheres. At the macroscopic level Lutran *et al.* [1991] identified fluid filaments. A filament was defined to be a stream of fluid flowing down the packed bed. In effect, a filament could be considered as a string of connected liquid pockets. These definitions are graphically depicted in Figure 2-1 (taken from Lutran *et al.* [1991]).

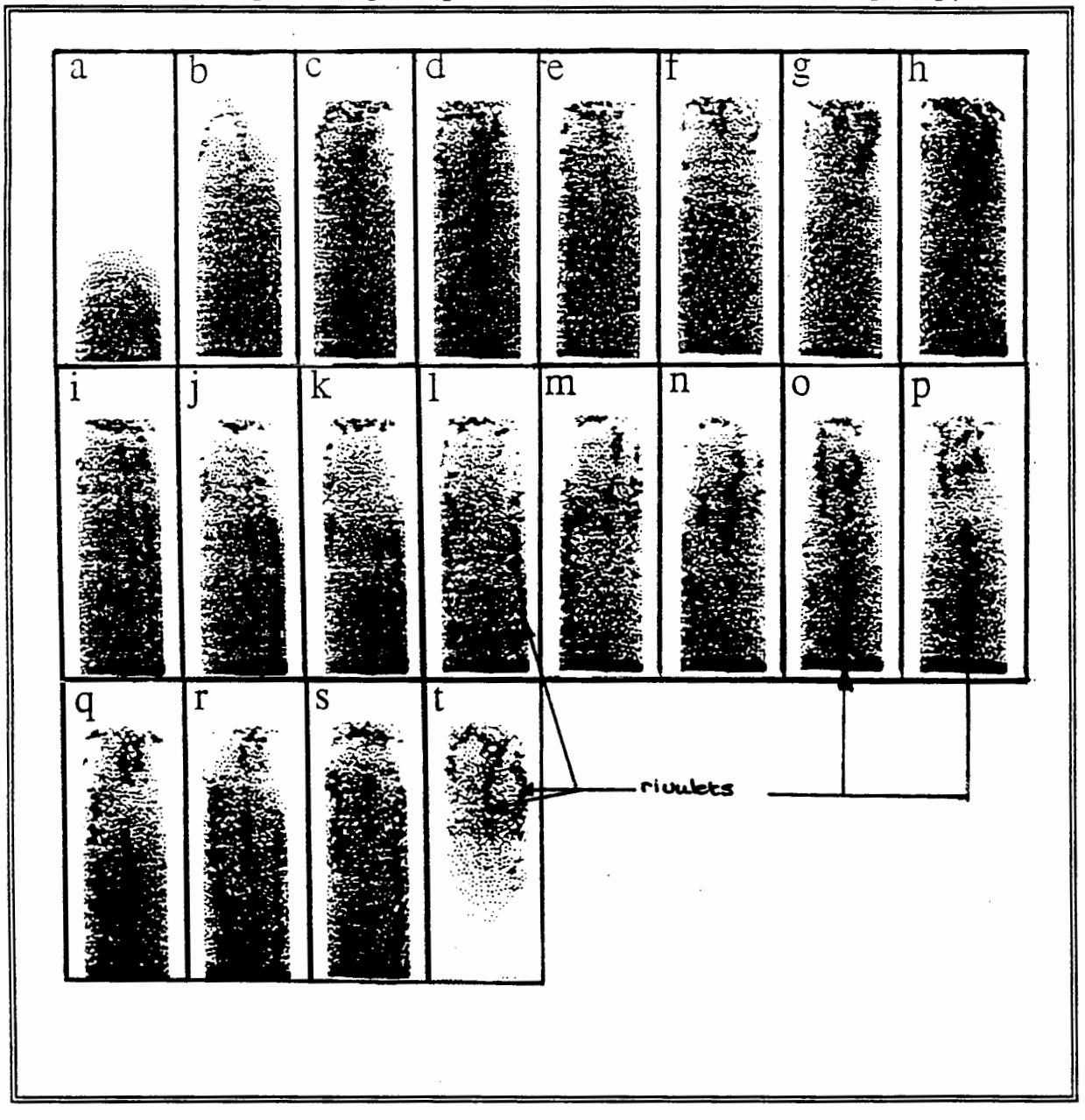
Figure 2-1. Film, rivulet and filament flow patterns as described by Lutran *et al.* [1991].



Lutran *et al.* [1991] investigated the effect of liquid flow rate, the influence of surface conditioning, the influence of particle size, the influence of inlet configurations and the effect of flow history on the flow patterns. A typical result from their work is shown in Figure 2-2 (taken from Lutran *et al.* [1991]). CAT scans (a) through (t) represent successive vertical planes from the front to the back of a square column packed with equi-sized glass spheres. Filament flow can clearly be seen in these figures as the darker regions moving from the top to the bottom of the column. The experimental conditions under which these CAT scans were taken can be summarised as follows: 3mm glass spheres were used, distilled water was evenly supplied through a uniform inlet distributor

at a rate of $3.163 l/m^2$ s and the particles were initially dry.

Figure 2-2. CAT scans used to investigate the flow patterns in a square column packed with equi-sized glass spheres. (Taken from Lutran *et al.* [1991].)



Although Lutran *et al.* [1991] do not discuss correlating their findings, it is very unlikely that any correlation would employ the particle Reynolds number as a parameter. The main reason for this is due to the fact that pore size and differences in local porosity seem to play an important role in the flow patterns which are established. As an example, Lutran *et al.* [1991] have shown that for a constant flowrate, film flow is more prevalent in packed beds containing larger particles (6mm spheres) compared to packed beds containing small particles (3mm spheres) in which filament flow is observed. They attribute this to the fact that packed beds which contain larger particles will have larger

pores between the particles. They suggest that at the fluid flow rate investigated ($3.163 l/m^2$ s), there was insufficient fluid to fill any of the pores between the 6mm spheres and thus establish filament flow. As the fluid flowrate was increased, more filaments were established which supports this hypothesis.

By a similar argument, any regions which exhibit a local decrease in porosity would favour the formation of filaments. Filament formation would also be aided in these areas by the increased number of solid-solid contact points which allows fluid to more easily distribute to these areas due to the lower surface tension forces on the fluid surface [Zimmerman and Ng 1986]. Wastes which contain a size distribution of particles or non-uniformly shaped particles usually exhibit localised regions of decreased porosity. This implies that these deposits would favour filament flow. It is important to note that channelling of fluid, where the fluid flows through deposits in preferential flow paths, is also probably a result of regions of decreased porosity. Channelling results in fluid short circuiting sections of the deposits. Although this is a highly irregular flow pattern which is extremely difficult to model, it is a very desirable flow pattern for fluid in waste deposits because it reduces the number of particles with which the fluid comes into contact as it percolates through the system.

Fluid redistribution on the other hand is accomplished by capillary and viscous forces. Capillary forces are inversely proportional to the pore size [Ng and Chu 1987]. Thus the capillary pressure is higher in deposits which contain small particles because the pores are relatively small in these deposits. Capillary forces also result in higher void space liquid holdup [Ng and Chu 1987]. This implies that deposits containing smaller particles will contain proportionately more fluid in their pore spaces compared to deposits containing larger particles.

Using the information discussed above, the following heuristics of fluid flow patterns in granular waste deposits are suggested. These are intended to give an idea of the most likely fluid flow patterns under different conditions.

The most likely fluid flow patterns in deposits which contain relatively large particles, such as leached ore particles, would be that of film flow. In this case the fluid would most likely tend to cover the surfaces of the particles as it progressed from one particle to the next. As the standard deviation for the size distribution of the particles increases, so too will the likelihood for filament formation and eventual channelling increase. Deposits of this nature are also most likely to exhibit a very low degree of fluid saturation. This implies that the external fluid holdup, that is the holdup of fluid in the pore spaces, would be very low.

As the average particle size decreases, the degree of saturation will increase due to capillary effects. Likely flow patterns include film flow with associated filament flows.

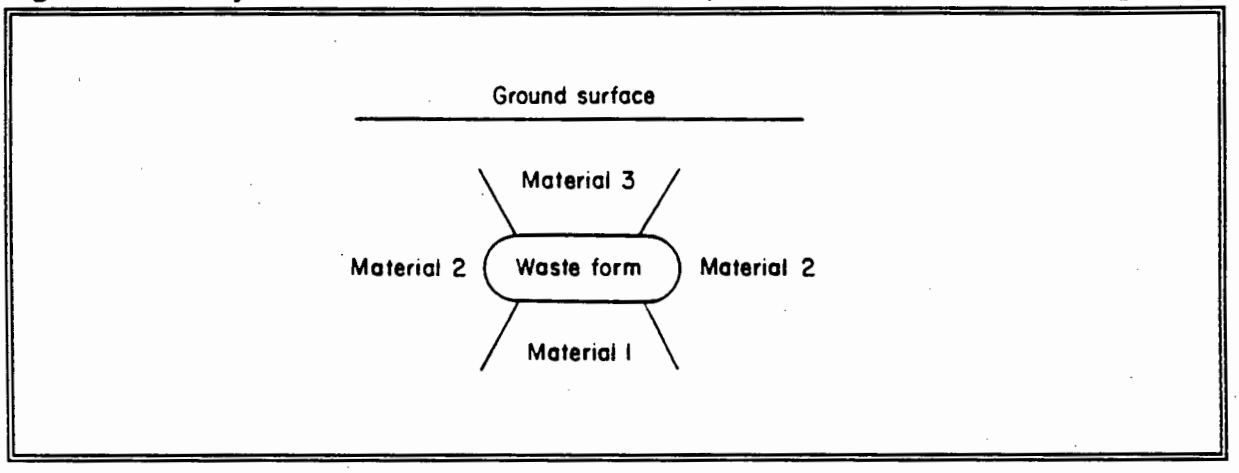
In the case where the deposit contains very fine particles, such as are encountered in slimes dam and tailings impoundment residues, the most likely flow patterns would be filament flow, and the deposit would most probably be almost totally saturated. The size distribution of particles is unlikely to affect the flow characteristics due to the high capillary forces which would be present. In all probability, the flow patterns in such

deposits could be described by plug flow models.

Fluid flow patterns within solidified structures.

Cote and Birdle [1987] have discussed the likely flow scenarios which can be associated with solidified monolithic structures. In summary they considered the situation shown in Figure 2-3 where the monolithic structure is surrounded by materials of different permeabilities.

Figure 2-3. Layout of a waste form in a landfill. (Taken from Cote and Birdle [1987].)



The first case they investigated is when the monolithic structure is in contact with a finite volume of static groundwater. This corresponds to the case when the hydraulic conductivity of the materials (1) and (2) are much smaller than the monolithic structure or material (3). Physically this situation represents the case where a pool of water exists on top of the waste deposit. In this situation, the water penetrates the solidified structure by diffusion.

The second situation addressed was when groundwater flows around the monolithic structure. This is the most common scenario because monolithic structures typically have hydraulic conductivities which are several orders of magnitude lower than that of the surrounding ground. Because groundwater will follow the path of least resistance, very little, if any, will flow through the structure. Even although there may be no convective flux through the structure, there is still the possibility of a lateral diffusive flux into the structure.

The last case considered is when the groundwater flows through the monolithic structure. This typically occurs when the monolithic structure fails and the hydraulic conductivity increases to a point that it is comparable to that of the surrounding materials. In many respects this situation can be compared to a granular deposit where the fragments of the original structure represent the 'particles' in the deposit.

2.2 Review of Existing Leachate Generation and Mobility Models.

Literature models which address leachate generation and mobility can be divided into four classes. The first two classes deal with predicting the breakthrough curves from granular waste deposits. The breakthrough curves are the curves which are formed by plotting the dissolved hazardous constituent concentrations which emanate from the base of the deposit as a function of time. The third class applies to modelling hazardous constituent release from solidified monolithic structures. The last group is concerned with hazardous constituent migration away from waste deposits.

All of the literature models are summarised in Table 2-1 which highlights their respective areas of application, advantages and disadvantages.

2.2.1 Empirical Models.

Purely empirical models fit experimental results obtained from lysimeter studies to an exponentially decaying function of time. These models [Demetracopoulos *et al.* 1986] were the only models up until the mid 1980's to describe hazardous constituent release from waste deposits. The mathematical form of the model can be summarised as:

$$C = C_0 \, e^{(-\beta t)} \qquad (2\text{-}1)$$

where C_0 hazardous constituent concentration in the leachate at the time of initial breakthrough (t=0),

t time, and,

ß an empirical constant.

Work carried out at the University of Cape Town [Petersen 1994] has shown that the fitting of lysimeter data to such a function often leads to a poor correlation. Even if the correlation was excellent, such a model could not be used to scale up to full scale deposit proportions because no size dimensions appear in the model. Thus in order to determine the breakthrough curve for leachate from granular deposits, where the fluid flow can often be approximated by one dimensional flow, a lysimeter of identical height to the envisaged deposit would be required. The only use of such a model would be the extrapolation of data to predict the release of hazardous constituents in the future. Because of the poor correlation and limited predictive ability of this model it is concluded that is has little applicability to modelling the release of hazardous constituents from hazardous waste deposits.

Table 1. Summary of Models which Describe Contaminant Release from Hazardous Waste Deposits.

Model:	Notes:	Application:	Advantages:	Disadvantages:
Lu *et al.* [1981].	First model to describe breakthrough curve.	None.		
Demetracopoulos [1986].	Contaminant release based on the continuity equation applied to a lysimeter as a single entity.	Direct extrapolation and initial analysis of lysimeter data.	Relatively simple to use. Uniquely characterises the deposit as a reacting entity.	Lumped parameter analysis. Limited ability to predict performance under different hydrodynamic conditions.
Batchelor [1990]. Cheng and Bishop [1990].	Model based on semi-infinite-media diffusion theory.	Modelling contaminant release from solidified structures.	Relatively simple to use.	Limited ability to predict performance under different contaminant and buffer concentrations.
Rowe and Booker [1990,1985a,1985b]. Dance and Reardon [1983]. Maslia *et al.* [1992]. Sudicky *et al.* [1983]. Vogt [1991].	Models describe contaminant migration *away* from hazardous waste deposits.	Determination of overall environmental impact of hazardous waste deposits.		
Bruan *et al.* [1974]. Roman [1974]. Shafer [1979].		Modelling contaminant release from granular hazardous waste deposits.	These models begin to address the underlying release mechanisms.	Assumes that diffusion of the fluid into the individual particles is the rate limiting step. Only considers a single chemical reaction taking place. Only apply to situations in which all of the particles are wetted and in which no preferential flow paths, stagnant zones or dry spots occur. They also do not address cyclic wetting and drying cycles.
Dixon [1992].	Contaminant release based on the continuity equation applied to the individual particles in the deposit.	Modelling contaminant release from granular hazardous waste deposits. Modelling contaminant release from monolithic hazardous waste deposits.	Does not assume that diffusion is the rate controlling mechanism. Includes the possibility of more than one chemical reaction taking place.	Only apply to situations in which all of the particles are wetted and in which no preferential flow paths, stagnant zones or dry spots occur. They also do not address cyclic wetting and drying cycles.

2.2.2 Models which make use of the Continuity Equation.

The second group of models makes use of the fluid continuity equation to describe the contaminant migration through the system. Demetracopoulos *et al.* [1986] have used such an approach to model leachate generation from domestic waste landfills. They considered the refuse material as a homogenous, partially saturated porous medium. Although domestic waste landfills often contain laminates, which preclude any attempts to approximate the fluid flow as one dimensional, Demetracopoulos *et al.* [1986] limited their study to cases in which the fluid flow can be approximated as one dimensional. Because of these assumptions, the refuse and flow characteristics are comparable to those found in granular waste deposits.

Demetracopoulos *et al.* [1986] make use of a hydraulic flow equation to predict the flow of fluid through the system. Demetracopoulos *et al* [1984] describe the derivation of this equation which was then solved numerically by Korfiatis *et al* [1984]. These equations can be summarised as:

$$\frac{\partial \theta}{\partial t} + \frac{\partial K(\theta)}{\partial z} - \frac{\partial}{\partial z}\left[D(\theta)\frac{\partial \theta}{\partial z}\right] = 0 \qquad (2\text{-}2)$$

$$q = K(\theta) - D(\theta)\frac{\partial \theta}{\partial z} \qquad (2\text{-}3)$$

where θ moisture content, or the fraction of the control volume occupied by liquid (m^3/m^3);
$K(\theta)$ hydraulic conductivity of the medium, (m/s);
q volumetric flux per unit bulk area (superficial velocity), (m/s);
$D(\theta)$ capillary diffusivity coefficient, defined by equation (2-4) below, (m^2/s);
t time, (s); and;
z space coordinate, measured vertically downward (m).

$$D(\theta) = -K(\theta)\frac{d\psi}{d\theta} \qquad (2\text{-}4)$$

where ψ tension suction head which is defined as the negative capillary pressure potential [Shaw 1994].

The tension suction head versus moisture content relationship which was used can be summarised as:

$$\frac{d\psi}{d\theta} = -b\psi_s\theta_s^b\theta^{(-b-1)} \qquad (2-5)$$

where s indicates saturation conditions; and;
b fitted empirical constant.

The release of hazardous waste constituents from a granular waste deposit was modelled using a continuity equation. The resulting equation can be summarised as:

$$\frac{\partial(\theta C)}{\partial t} + \frac{\partial(qC)}{\partial z} = \frac{\partial}{\partial z}[\theta\ E(\theta)\ \frac{\partial C}{\partial z}] + \theta\ R \qquad (2-6)$$

where C fluid phase concentration of the hazardous constituents (kg/m^3);
$E(\theta)$ longitudinal dispersion coefficient (m^2/s);
q volumetric flux per unit bulk area (superficial velocity) (m/s); and;
R source or sink term (kg/m^3s).

By assuming that the source or sink term R, which corresponds to the rate of release of hazardous constituents, is controlled by the mass transfer between the solid and liquid phases, an equation which describes the generation and transport of non-biodegradable hazardous constituents was obtained and can be summarised as:

$$\frac{\partial(\theta C)}{\partial t} + \frac{\partial(qC)}{\partial z} = \frac{\partial}{\partial z}[\theta\ E(\theta)\ \frac{\partial C}{\partial z}] + \theta\ k'\frac{S}{S_0}(C_{st}-C) \qquad (2-7)$$

where k' rate coefficient for mass transfer (1/s);
S local solid mass fraction of hazardous constituent available for transfer (kg/m^3);
S_0 local solid mass fraction of hazardous constituent available for transfer at time t=0 (kg/m^3); and;
C_{st} maximum possible hazardous constituent concentration in the fluid phase (kg/m^3).

Models of this nature are not limited to the case where mass transfer effects control the rate of release of hazardous constituents. By using appropriate expressions for the source or sink term, R, the model can be adapted to describe the situation where chemical kinetic or intra-particle diffusion resistances are rate limiting. In each case, a parameter is being used to quantify the rate limiting mechanism in the release of hazardous constituents.

Demetracopoulos *et al.* [1986] extended the model to incorporate biological activity. This resulted in two coupled partial differential equations, one which describes the growth and transport of the microorganism population and one which describes the release and transport of hazardous constituents.

The merit of such a model is that it offers a relatively simple method by which to study the release of hazardous constituents from waste deposits.

The disadvantage of this model is that there is no way of predicting the parameters used to describe the rates of release of hazardous constituents. These parameters need to be determined from appropriate lysimeter experiments.

2.2.3 Models Suitable for Predicting Leachate Generation from Solidified Monolithic Structures.

Batchelor [1990], Bishop [1990] and Cheng and Bishop [1990] all have developed models which are suitable to predict leachate generation from solidified monolithic structures. From CSTR leach tests on fragments of solidified structures, these investigators found the release of hazardous constituents to be dominated by diffusion internal to the solidified structure. The model considers the monolith as a single entity and determines the rate of release of hazardous constituents using semi-infinite-media diffusion theory. The following form of equation is almost always used in this approach:

$$\frac{\Sigma a_n}{A_0}\ \frac{V}{S}=2\ \left(\frac{D_e}{\pi}\right)^{0.5}\ t_n^{0.5} \qquad (2\text{-}8)$$

where
- a_n hazardous constituent loss during leaching period n (kg);
- A_0 initial amount of the hazardous constituent present in the specimen (kg);
- V volume of the specimen (m^3);
- S surface area of the specimen (m^2);
- t_n cumulative time to the end of leaching period n (s); and;
- D_e effective diffusivity (m^2/s).

Equation (2-8) is based on the following semi-infinite-media equation:

$$C(X,t)=C_0\ erf\left(\frac{X}{2\sqrt{(D_e t)}}\right) \qquad (2\text{-}9)$$

where
- C hazardous constituent concentration at position X,
- X distance from the outer surface of the monolith, and,

t time.

A limitation with respect to the use of equation (2-9) which is not mentioned by any of the investigators is that the following condition must hold before this equation can be applied to *finite* systems [Welty, Wicks and Wilson, 1984]:

$$\frac{V}{2S\sqrt{D_e t}} > 2 \qquad (2\text{-}10)$$

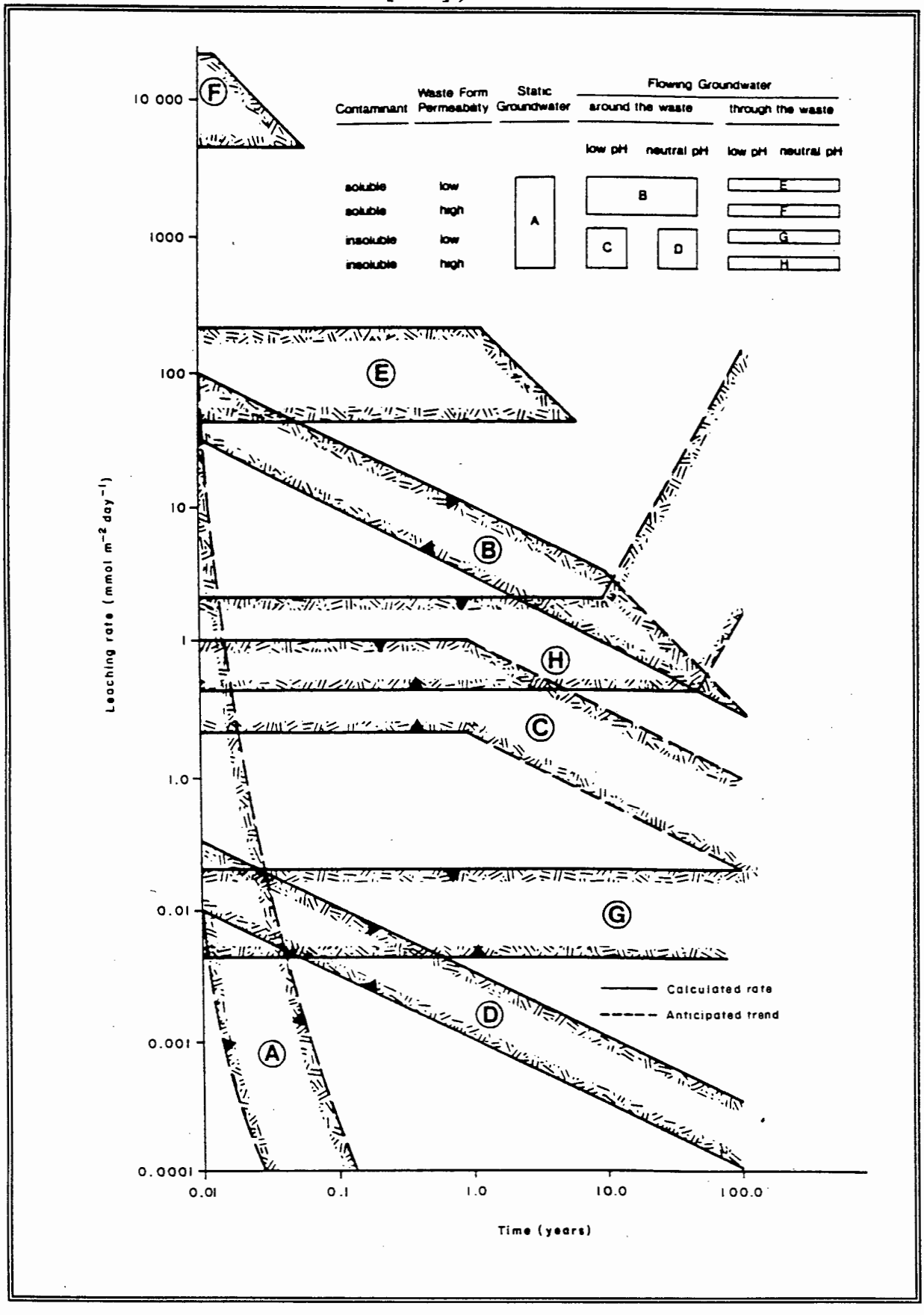
One reason for this omission may be because the assumption of a constant mid-plane hazardous constituent concentration can be considered to be a worst case scenario. This implies that the models would tend to predict a greater loss of hazardous constituents than would occur in finite systems. Another reason for this omission could be because the authors did not consider evaluating their models at times sufficiently large to violate this condition.

In general, the leach type models are suitable to extrapolate data in time as well as to predict the release of hazardous constituents from various sized monolithic structures.

The limitation of these models is that they cannot be used to extrapolate the results to different hazardous constituent and buffer concentrations within the solidified matrix. The reason for this is that the effective diffusivity is a fitted parameter which is affected by the concentrations of the hazardous constituents, buffer and inert species within the solidified waste [Cheng and Bishop 1990].

Cote and Birdle [1987] have made use of this class of models to investigate several long term leaching scenarios for solidified waste forms. In particular they investigated the effects on the release and mobilisation of hazardous constituents of the hydraulic regime of the groundwater, chemical characteristics of the groundwater, hydraulic conductivity of the solidified structure and chemical speciation of the hazardous constituents within the waste matrix. Figure 2-4 is a summary of the results of their work. This figure can be used to estimate the release rates of hazardous constituents from monolithic structures. As an example, consider the release of hazardous constituents from a fractured monolithic structure. Such a situation would resemble a granular waste deposit and fluid would most likely percolate through the fractured structured as discussed in section 2.1.2. Typical release rates of soluble hazardous constituents in such a situation are shown in region (F) in Figure 2-4. This region indicates that very high leaching rates would be observed ($\simeq$10 000 mmol m^{-2} day^{-1}) and that the leaching processes would cease within the first few months due to depletion of the hazardous constituents. Insoluble hazardous constituents could be released by active leaching and typical release rates are shown by region (H) in Figure 2-4. Notice that the leaching rates for insoluble constituents ($\simeq$1 mmol m^{-2} day^{-1}) is much lower than for soluble components as expected.

Figure 2-4. Inference of leaching rates following different leaching scenarios. (Taken from Cote and Birdle [1987].)



2.2.4 Models Describing Contaminant Migration away from Deposits.

The last group of models includes work on contaminant migration away from hazardous waste deposits. These models uncouple the hydrodynamic analysis from the chemical reaction aspects, assuming that no interaction occurs. The hydrodynamic part, which is a groundwater flow problem, is modelled using a Darcy law approach which incorporates a bulk hydraulic conductivity. A bulk hydraulic conductivity is an average measure of the distance which fluid would move through a porous medium per unit time. This quantity is a function of both the fluid type and fluid saturation within the porous medium [Shaw 1994]. Saturated hydraulic conductivities are usually determined experimentally. Hydraulic conductivities under unsaturated conditions can also be determined experimentally but are much more difficult to determine than saturated hydraulic conductivities [Fourie 1995]. An alternative approach to determine unsaturated hydraulic conductivities is to use appropriate models, presented in groundwater flow texts, which use the saturated hydraulic conductivity as a parameter [Freeze and Cherry 1979]. It is important to note that bulk hydraulic conductivities are quantities which are used to describe the macroscopic flow rates of fluids in porous media. As such, no information with respect to localised flows within the medium can be determined. In cases where this level of detail is required or in cases where local variations in hydraulic conductivities preclude the use of a bulk hydraulic conductivity, the full tensorial form of the hydraulic conductivity would need to be evaluated. Knox *et al.* [1993] present information with respect to the full tensorial definition of hydraulic conductivities.

The chemical reactions are modelled either by adsorption/desorption isotherms, which are based on experimental results [Rowe and Booker 1990, 1985a,1985b; Dance and Reardon 1983; Maslia *et al.* 1992; Sudicky *et al.* 1983] or by totally predictive thermodynamic analyses [Vogt 1991].

One of the assumptions used in these models is the value assigned to the leachate concentration at the base of the deposit. This is estimated by determining the volume of fluid within the granular deposit and assuming that all of the hazardous components are released into this fluid subject to adsorption and desorption equilibria. Once this leachate concentration has been estimated, the subsequent transport of the leachate through the underlying ground is modelled. Since these models do not address leachate generation aspects they could only be used as suitable mobility models within granular waste deposits. If this strategy could be used as a mobility model it would have the advantage of including the effects of adsorption and desorption on leachate mobility.

Before these models could be used to describe leachate mobility in granular waste deposits, the adsorption/desorption isotherms of the hazardous constituents onto the granular particles would need to be determined. The determination of these isotherms for granular waste particles will be significantly more complex than for inert ground samples. This is due to the fact that it is difficult to determine an adsorption/desorption isotherm for a material which is itself releasing the same components. Methods to deconvolute the effects of leachate generation from adsorption and desorption processes would need to be determined.

It is unlikely that thermodynamic analyses could assist to predict the chemical reactions taking place because of the complex chemical compositions of waste particles. In most cases it is impossible, due to physical and financial constraints, to determine the initial hazardous constituent and buffering material speciation required for thermodynamic analyses.

Although this group of models has limited applicability for the objectives as defined, it will be useful when a complete environmental impact assessment of a hazardous waste deposit is required. Once the release of hazardous constituents from a deposit has been calculated, this information can be used as the input data for the migration models. The subsequent migration through the environment, and thus the health risks and environmental impact, could then be determined.

2.3 Summary of the Strategy Adopted in Precious Metal Heap Leaching Models.

The general strategy used [Roman *et al.* 1974] in the analysis of heap leaching is to divide the heap conceptually into columnar sections. Each column is then considered as a simple one dimensional, non-catalytic reactor. The progression of the reactions in the individual particles is followed using a suitable chemical reaction model. These reaction models require the concentration of the fluid in contact with the particles. Thus by calculating the fluid concentration in contact with the particles and the progression of the reactions as a function of time, the dissolved hazardous component concentration can be determined. These concentrations can then be used to calculate the breakthrough curve. In order to calculate the fluid concentration in contact with the particles, the columnar sections are further divided into a set of discs which are stacked on top of each other as shown in Figure 2-5. The fluid concentration within each disc is assumed to be constant and is calculated by a simple mass balance. This strategy is summarised in Figure 2-6.

Figure 2-5. An ore heap conceptually divided into columnar sections and the columnar sections divided into discs. (Figure taken from Roman *et al.* [1974].)

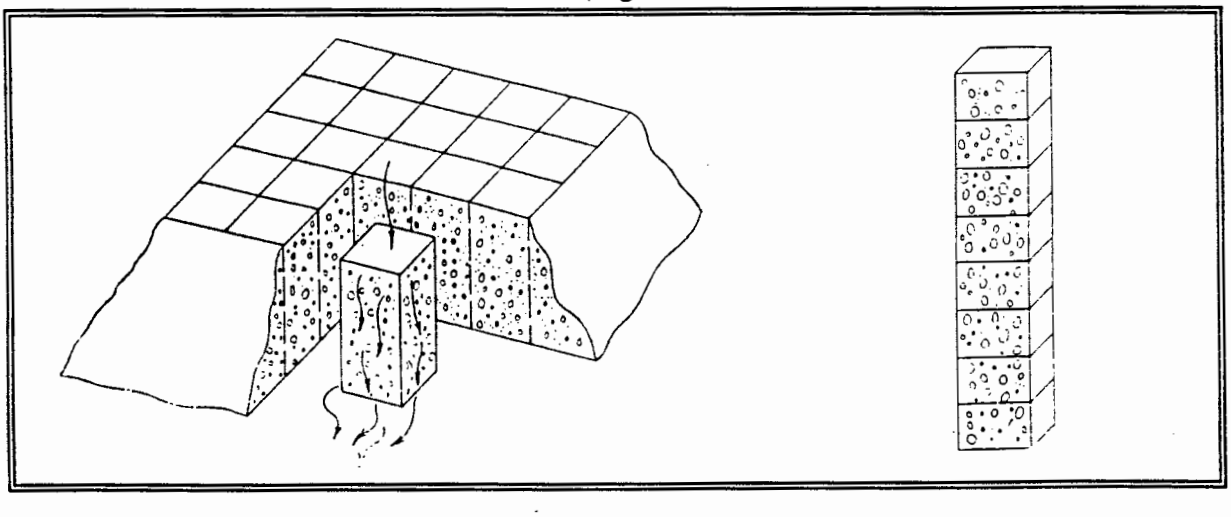
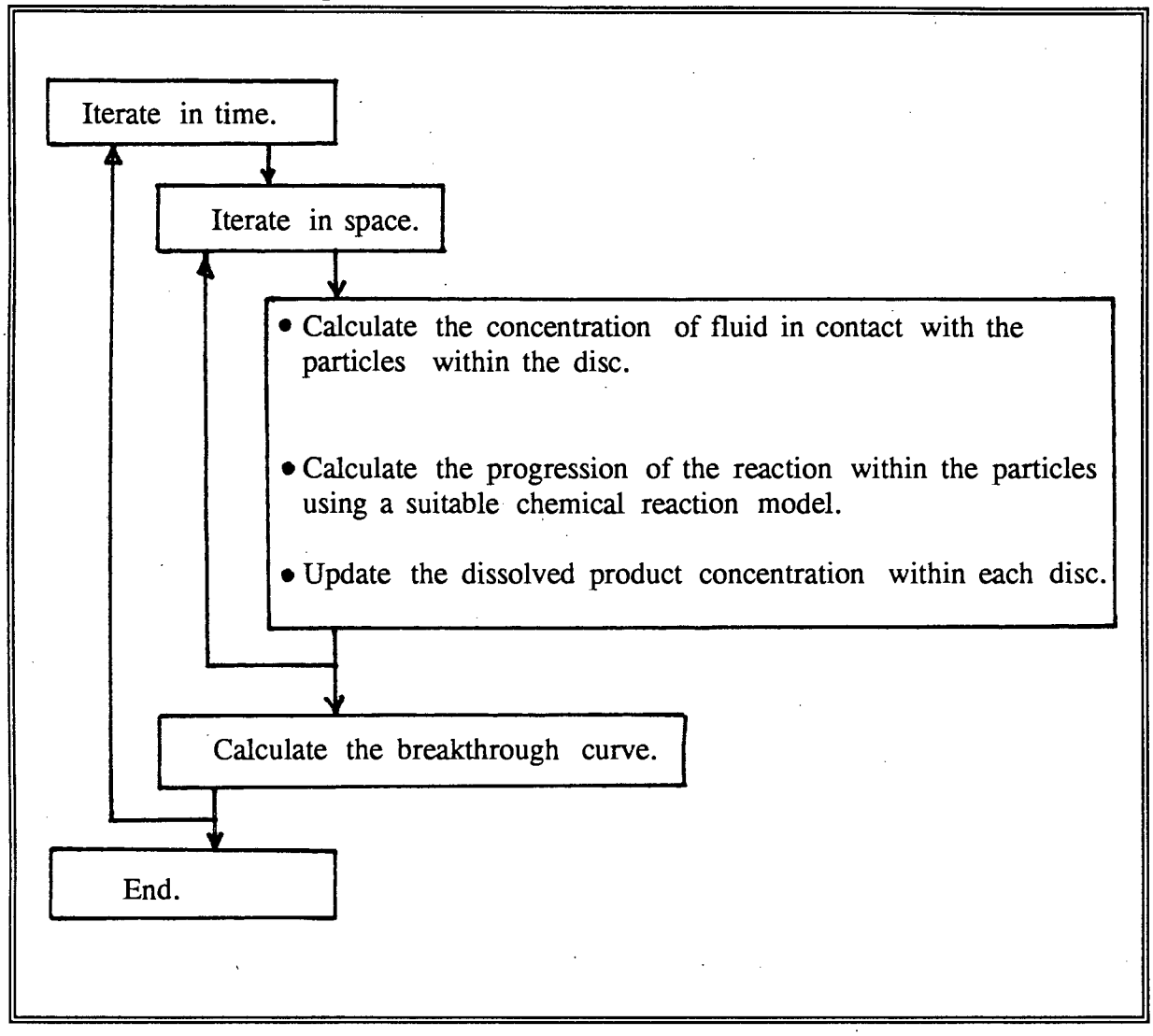


Figure 2-6. Summary of the strategy used to determine the breakthrough curve in heap leaching operations.



Braun *et al* [1974], Roman *et al.* [1974] and Shafer [1979] have used this approach to model heap leaching operations. In all of these cases very simple hydrodynamic behaviour has been considered. This is evident in that all of the particles within each disk were assumed to be totally wetted. For the application of heap leaching this is not a bad assumption because totally wetted particles is the objective of heap design and lixiviant spray patterns. Although it was not explicitly stated, these investigators have also assumed that no significant preferential flowpaths or stagnant zones are present.

The chemical reaction model almost always used in this approach is a shrinking core model. This implies that diffusion of fluid reagent into the individual particles is the elementary rate controlling step. This can be a severe limitation because in some cases the chemical reactions are sufficiently slow to result in a homogenous reaction. In these cases, the intrinsic kinetics of the chemical reactions become very important. The simple shrinking core model as used by Roman *et al.* [1974] has a further limitation in that it

only considers a single reaction taking place.

Dixon [1992] has developed a chemical reaction model which eliminates these shortcomings. Dixon's approach is similar to that of Demetracopoulos *et al.* [1986] in that it makes use of the continuity equation to model chemical release. The main difference is that Dixon applies the analysis to an individual particle while Demetracopoulos applied it to a lysimeter as a single entity.

Dixon's model includes information on the diffusivity of fluid reactant into the particles and the intrinsic kinetics of the multiple reactions which take place. Although the intrinsic kinetics are not explicitly obtained in Dixon's model, the ratio of the diffusivity over the intrinsic kinetics is determined. In many ways this corresponds to determining the effective reaction kinetics, which include the effects of intra-particle diffusion, of the waste particles. The model is developed in such a manner that once the effective reaction kinetics of a particular sized particle have been determined, the technique can be extended to other sized particles. Dixon's model also makes provision for surface solid reactant concentrations which differ from bulk solid concentrations within the particles.

In summary Dixon's chemical reaction model makes provision for the following:

- the diffusivity of the fluid reagent into the particles as well as the intrinsic kinetics of the solid reactant within the particles,
- multiple competing reactions, and,
- surface solid reactant concentrations which differ from the bulk solid concentrations within the particles.

Dixon made use of this model to describe copper release from a heap leaching operation. He made similar assumptions concerning the hydrodynamic interactions to those of Roman *et al.* [1974]. Not withstanding the complications of preferential flow paths, Dixon's model has an excellent capacity to predict the release of hazardous constituents from hazardous waste deposits which contain granular material.

Dixon's model can also be used to model the release of hazardous constituents from a monolithic structure. The ratio of the effective diffusivity of the fluid into the monolith over the rates of the hazardous constituent release can be determined by conducting a CSTR test on small fragments of the monolith. This information can then be used to predict the release of hazardous constituents from the monolithic structure. This is a more sophisticated method to the one adopted by Batchelor [1990], Bishop [1986] and Cheng and Bishop [1990], because it does not assume that diffusion is the controlling resistance or make use of the semi-infinite-media assumption. This method also has a greater predictive power in that it can also be used to investigate the effect of hazardous constituent and buffer concentration on the rate of release of hazardous constituents which the prior methods could not.

2.4 Modelling the Release of Hazardous Constituents from Partially Saturated Granular Waste Deposits.

No models exist in the literature which describe the release and subsequent transport of hazardous constituents under unsaturated flow conditions.

Probably the most comprehensive attempt to include the effects of unsaturated conditions on leachate generation are those of Demetracopoulos *et al.* [1986]. They include these effects by using a hydrodynamic equation for unsaturated flow. These equations, described previously in equations (2-2) to (2-5), are often referred to as groundwater flow equations. Hydrodynamic equations or groundwater flow equations yield information about the local levels of saturation as well as local fluid flows. In the equations presented, which are already in one dimensional format, the saturation of fluid and fluid flow within a waste deposit can be determined as a function of position within the waste deposit and time. Although Demetracopoulos *et al.* [1986] determine the effect of unsaturated conditions on fluid flow, they do not consider the effect of these conditions on the mass transfer of hazardous constituents. Since they made use of mass transfer to describe the rate of release of hazardous constituents in their model, the influence of this factor should have been investigated. To circumvent the problem, Demetracopoulos *et al.* define a distribution coefficient, which relates the hazardous constituent concentration in the fluid to the hazardous constituent concentration in the waste particles, specific to the situation being investigated. This distribution coefficient needs to be determined experimentally for each situation. In effect, this coefficient includes the average effects of the degree of saturation on hazardous constituent release.

The only other attempts to include the effects of the level of saturation on leachate generation can be found in the heap leaching models which usually include a saturation parameter [Roman *et al.* 1974]. The saturation parameter represents the average level of saturation within the lysimeter. All of these models further assume that all of the particles within the heap are totally wetted with fluid. Although this assumption is probably valid for heap leaching, it will almost certainly not hold in unsaturated waste deposits. The reason for this is that fluid which percolates through unsaturated waste deposits usually does so in a fairly random manner.

The level of particle wetting is important in the effective operation of trickle bed reactors and several researchers have investigated means to determine a suitable wetting factor [Columbo *et al.* 1976; Schwartz *et al.* 1976; Sicardi *et al.* 1980; Mills and Dudukovic 1981; Ramachandran 1986]. A wetting factor is a simple factor which describes the average fraction of the particle surface area covered by fluid. It is felt that the inclusion of similar wetting factors into the heap leaching models when they are applied to hazardous constituent leaching would be most beneficial.

Unsaturated flow through deposits drastically affects the flow patterns in deposits. As previously discussed, the flow patterns in waste deposits are extremely complex. All the hydrodynamic equations considered in the literature models are relatively simple one dimensional, *plug flow* models. More suitable hydrodynamic models need to be

developed for fluid flow in unsaturated deposits.

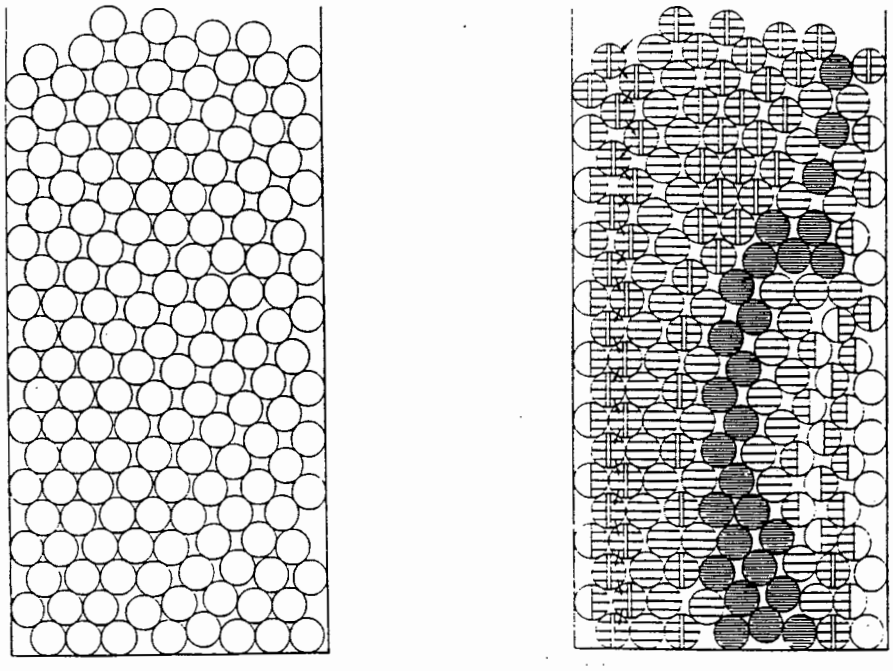
The first extension to plug flow models which would begin to incorporate irregular flow behaviour is to include axial dispersion. Levenspiel [1972] describes the use of the 'tanks in series' approach to include axial dispersion. In summary, in a 'tanks in series' approach to a plug flow reactor, the plug flow reactor is modelled as a series of continuously stirred tank reactors. A mathematical analysis of the technique yields the number of tanks required in order to approximate plug flow through the system. Dispersion is included into the model by using fewer tanks than required because fewer tanks tend to erode the plug flow nature by allowing greater degrees of mixing within the system.

The 'tanks in series' model discussed above is a one parameter hydrodynamic model. This means that one parameter is required to account for the non-uniform flow characteristics. More complex, multi-parameter models exist which account for highly irregular flow characteristics. These strategies consider the system to consist of several regions which can be described by plug flow, dispersed plug flow and mixed flow models. Zones which contain stagnant fluid can also be incorporated into these strategies. Levenspiel [1972] discusses the implementation of these techniques.

An alternative approach to dealing with the effect of the complex flow patterns on chemical reactions within trickle bed reactors has been investigated by Funk *et al.* [1990]; Zimmerman *et al.* [1987], and, Ng and Chu [1987]. These investigators model the fluid flow at the particle level. To accomplish this they make use of a porous medium model and a fluid distribution model. The porous medium model is used to describe the geometric locations of the particles in the trickle bed reactor. To date, these investigators have only investigated porous media which consist of equi-sized spheres. Using Monte-Carlo techniques, a suitable representation of the geometric locations of the spheres within the trickle bed reactor is obtained. The result of one such simulation is shown in Figure 2-7 (taken from Zimmerman *et al.* [1987]).

The fluid distribution model is used to determine the fate of fluid which impinges on a single particle in the assembly of particles. Ng [1986] has developed a wetting criterion for particles. For a given flow of fluid onto a particle, this criterion determines whether the fluid will totally cover the particle, which is comparable to film flow defined by Lutran *et al.* [1991], or whether it will be confined to a specific part of the particle - which is comparable to rivulet flow. This information is then used to determine the flow paths of the fluid as it moves from one particle to the next. In this manner the flow patterns of fluid through the trickle bed can be determined. A typical result of such a calculation strategy is also shown in Figure 2-7 (taken from Zimmerman *et al.* [1987]).

Figure 2-7. An example of a two dimensional, porous medium model used to simulate the geometrical characteristics of particles in a trickle bed reactor and a typical flow pattern predicted by using the strategy of Zimmerman *et al.* [1987].



The densely shaded spheres represent complete wetting of the entire particle surface. The spheres with lighter shading over the entire circle are partially wetted. It indicates a liquid film covering more than 50% of the sphere surface. All unshaded regions on the spheres represent dry areas. Note that the arrows indicate the flow of an isolated liquid rivulet.

Funk *et al.* [1990] use this strategy to determine the fluid flow patterns and couple this information to a chemical reaction model for reaction in a *catalyst* pellet. Catalytic reactions are usually steady state reactions and as such are described by suitable ordinary differential equations. The calculations pertaining to the chemical reactions are simplified by making use of an appropriate steady state effectiveness factor. Using this combined model Funk *et al.* predict overall performance of the trickle bed reactor.

It is very unlikely that such an approach will be able to be applied to leachate generation in granular waste deposits in the near future because of the following limitations. As pointed out earlier, the porous medium model in its present form can only accommodate equi-sized spheres. Although there is no reason why the model cannot be extended to include a size distribution of spheres, the calculation strategy to merely generate the porous medium model will become far more complex. Further, the reactions which take place in waste particles are *non-catalytic* in nature. Such reactions need to be described

using partial differential equations. Since these reactions are not steady state in nature, the calculation strategy cannot be simplified by the use of steady state effectiveness factors. This means that the full partial differential equations describing the reactions need to be solved. The computation involved in solving these equations is significant. The computers which would be required to solve the porous media model, the fluid flow path problem and a partial differential equation for each particle in the system makes this approach unattractive. Before such a method is adopted, it is felt that simpler models need to be investigated to determine whether they can be used to solve the problem. Two such simpler models are investigated in the remainder of this thesis. The first model is a macroscopic, lumped parameter model and the second model is a heterogenous columnar model.

Chapter 3. A Macroscopic, Lumped Parameter Model to Describe Leachate Generation and Mobility in Granular Waste Deposits.

In this work, the first model investigated, which is capable of predicting leachate generation and mobility within granular waste deposits, is a macroscopic, lumped parameter model. There were two main reasons for investigating a model of this nature. The first reason is due to its inherent simplicity. The macroscopic model considers the deposit as a single reacting entity. As such, it does not attempt to analyze the release of hazardous constituents at the particle level. Instead, it makes use of an overall expression which describes the rate of hazardous constituent release from a small volume element of the deposit. This expression includes the effects of chemical kinetics, hydrodynamic aspects, diffusion and hazardous constituent location on the release of contaminants. This is why the model is termed a lumped parameter model. In effect it assumes that not enough is known about the intrinsic kinetics of the system or about the complex hydrodynamics or even the hazardous constituent location to allow a more rigorous analysis. In many respects, it is this simplicity which makes the model very attractive. In any situation where these aspects cannot be determined or are uncertain, the lumped parameter model can still be used.

The second reason for investigating the macroscopic, lumped parameter approach deals with model accuracy. The lumped parameter model does not require any assumptions with respect to particulate or hydrodynamic aspects. Instead, these effects are included in the experimentally determined model parameter. In effect, the rate of release of hazardous constituents from a waste deposit as a whole is being determined. This is in contrast to more detailed models. More detailed models determine the rate of release of hazardous constituents from the individual particles within a granular waste deposit and then predict the deposit performance. Before the deposit performance can be predicted however, assumptions with respect to fluid flow patterns, fluid saturation and particle wetting need to be made. If any of these assumptions are in error then the accuracy of the more detailed models will be effected. When insufficient information about the deposit is available to make confident choices with respect to these aspects, the most reliable method may be to make use of the lumped parameter which does not require this information.

The macroscopic, lumped parameter model was derived by applying a one dimensional fluid continuity equation to the deposit as a whole. The approach adopted is very similar to the one followed by Dixon [1992], the only difference being that Dixon applied the fluid continuity equation to a single particle rather than to an assembly of particles in the form of a waste deposit. The model developed is also very similar to the model of Demetracopoulos *et al.* [1986] which was discussed in the previous chapter. The main differences between the model developed and that of Demetracopoulos *et al.* is that the 'rate of hazardous constituent release' term, R, used in the present model was assumed to be described by chemical kinetics rather than by mass transfer considerations. The

reason for this is that the effects of competing chemical reactions can be investigated with this approach. Realizing that the effects of competing chemical reactions can drastically alter the hazardous constituent release profiles, it was felt that it should be incorporated into the model. The other main difference between the present model and that of Demetracopoulos *et al.* is that the present model does not make use of hydrodynamic flow equations which describe unsaturated flow. Rather, much simpler 'plug flow' hydrodynamic equations were used. It was realised that the more complete hydrodynamic equations could always be included at a later stage and it seemed more effective to first investigate the potential of the model using the simpler hydrodynamic equations.

The remainder of this chapter presents the model derivation. It also discusses an appropriate solution strategy. This is followed by a section which indicates how a time dependent fluid percolation velocity can be incorporated into the solution strategy. The solution strategies have been coded into suitable computer routines and the details of these routines are discussed. Typical results obtained from the model are presented. Some discussion which indicates how the model can be fitted to experimental results is offered, and the limitations of the model are summarised.

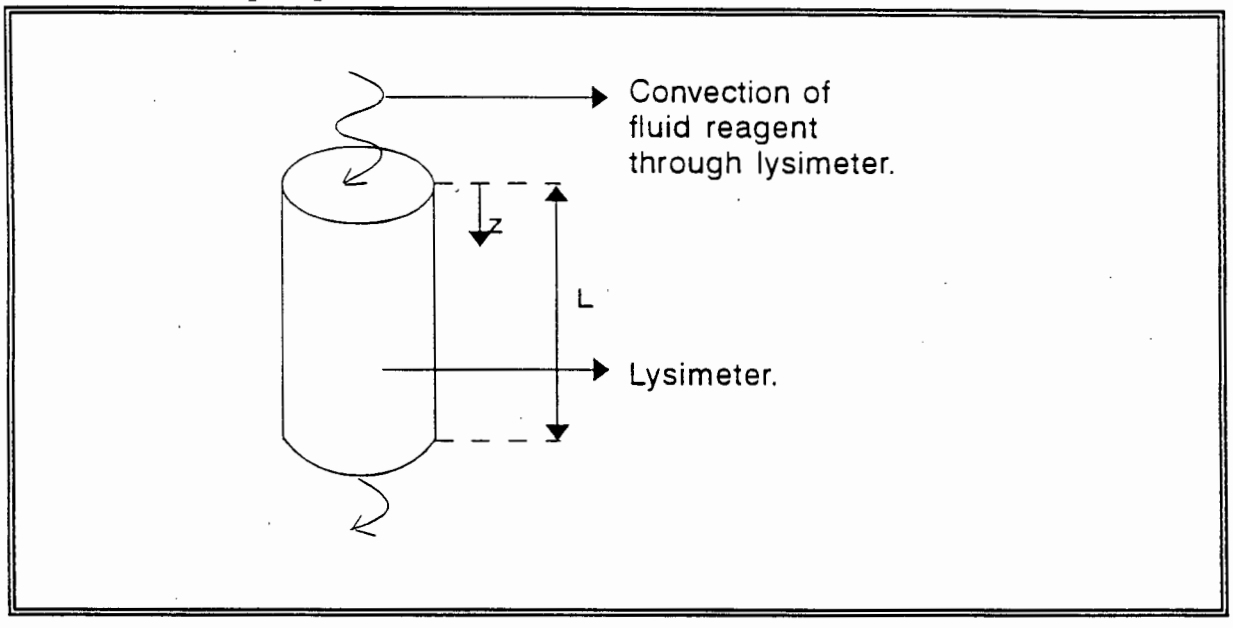
3.1 Development of the Equations.

Figure 3-1, a schematic of a lysimeter which is thought to be representative of a real waste deposit, forms the basis of the macroscopic, lumped parameter model. It is assumed that the fluid reagent, A, percolates through the lysimeter and reacts with the solid reactants, B_i, according to the following stoichiometric equation:

$$A + \sum_{i=1}^{n} b_i B_i \rightarrow dissolved\ products$$

On a mass basis the stoichiometric coefficient, b_i, would represent the mass of solid reactant, B_i, required to react with a unit mass of fluid reagent. This is very often more convenient to use than a conventional molar basis since mass concentrations are more easily determined experimentally, and the 'b_i' terms then represent an aggregated elemental behaviour rather than a species balance.

Figure 3-1. Schematic of a lysimeter which forms the basis for the macroscopic, lumped parameter model.



The continuity equation for the fluid reactant can be obtained from a statement of conservation of mass. This equation in operator format is:

$$\nabla \cdot N_A + \epsilon_{Col} Col_{\%\ Sat} \frac{\partial C_A}{\partial t} - \Sigma\ r_{A_i} = 0 \qquad (3\text{-}1)$$

where N_A fluid flux through the deposit, including a bulk convective contribution and a diffusive contribution;

ϵ_{Col} deposit porosity;

$Col_{\%\ Sat}$ deposit saturation; and;

r_{Ai} defined as the rate of *production* of fluid reagent A by reaction with species i.

The summation is required to account for the production of fluid reagent A by all the participating reactions.

The equation derived by allowing the rate of *production* of a solid reactant, B_i, to be described by a kinetic expression which is of variable order with respect to the solid reactant and first order with respect to the fluid reagent is:

$$r_{B_i} = b_i r_{A_i} = \frac{dC_{B_i}}{dt} = -k_{Bi} C_{Bi}^{\phi_{Bi}} C_A \qquad (3\text{-}2)$$

where C_{Bi} mass or moles of the solid reactant *per unit volume of solid*;
C_A mass or moles of the fluid reagent *per unit volume of fluid*;
Φ_{Bi} variable reaction order; and;
k_{Bi} reaction rate constant.

The reaction rate constant has units, depending on the reaction order term Φ_{Bi}, such that the units on the right hand side of the equation are rendered to be mass or moles of solid reactant per unit time *per unit volume of solid.*

In this analysis, the fluid flux through the column has been assumed to dominated by a convective flux due to the fact that the convective flux usually masks the effects of a diffusive flux. An expression for the convective flux is:

$$N_A = uC_A \qquad (3-3)$$

where u represents the superficial fluid velocity.

Substitution of equations (3-2) and (3-3) into equation (3-1) yields (expressed in one dimensional cylindrical coordinates with axial dependence only):

$$\epsilon_{Col} Col_{\%\ sat} \frac{\partial C_A}{\partial t} = -u\frac{\partial C_A}{\partial z} - C_A\frac{\partial u}{\partial z} - \sum_{i=1}^{n} (1-\epsilon_{Col}) \frac{k_{Bi} C_{Bi}^{\Phi_{Bi}} C_A}{b_i} \qquad (3-4)$$

where t time; and;
z axial position within the deposit.

By assuming that the fluid is incompressible and that the relative void space saturation in the deposit remains constant, which implies that saturation is not a function of fluid velocity through the deposit, the second term on the right hand side of the above equation can be shown to be zero due to the fact that the divergence of the fluid velocity is zero. Thus the equation simplifies to:

$$Col_{\%sat}\epsilon_{Col} \frac{\partial C_A}{\partial t} = -u\frac{\partial C_A}{\partial z} - \sum_{i=1}^{n} (1-\epsilon_{Col}) \frac{k_{Bi} C_{Bi}^{\Phi_{Bi}} C_A}{b_i} \qquad (3-5)$$

The initial and boundary conditions which apply are:

$$C_A(z,0)=0 \qquad (3-6)$$

$$C_B(z,0)=C_{B_0} \qquad (3-7)$$

$$C_A(0,t)=C_{A_{InletConc.}} \qquad (3-8)$$

Note that in equation (3-8) the boundary condition imposed is that the inlet concentration of the fluid reagent to the deposit is constant. It is simple to incorporate the boundary condition which considers the inlet concentration of fluid reagent as a function of time into the model - as long as this function is prescribed. Only the use of a constant boundary condition has been demonstrated in the model development since it is very unlikely for the concentration of fluid reactant entering the column to change dramatically with time.

3.2 Expressing the Equations in Dimensionless Form.

It is desirable to express model equations in a dimensionless format. When expressing an equation in dimensionless format, the original variables are grouped into dimensionless parameters which are less numerous than the original number of variables and which tend to have real physical significance. Reducing the number of variables is advantageous in that it provides results of greater generality, thereby enabling the effects of changing conditions within the deposit to be studied more easily. This is also helpful when attempting to plan experimental work or correlate experimental results [Welty, Wicks, Wilson 1976].

Equations (3-2) and (3-5) to (3-8) can be expressed in dimensionless form by defining the following dimensionless parameters and dimensionless groups:

$$\alpha=\frac{C_A}{C_{A_{InletConc.}}} \qquad (3-9) \qquad \sigma_{Bi}=\frac{C_{Bi}}{C_{Bi_0}} \qquad (3-10)$$

$$\xi=\frac{z}{L} \qquad (3-11) \qquad \tau'=\frac{t}{T} \qquad (3-12)$$

where

$$T=\frac{L}{u^*} \qquad (3\text{-}13)$$

where L deposit length scale.

$$DG1=\frac{u}{u^*\epsilon_{Col}Col_{\%\,Sat}} \qquad (3\text{-}14) \qquad DG2_i=\frac{(1-\epsilon_{Col})}{\epsilon_{Col}Col_{\%\,Sat}}\frac{k_{Bi}C_{Bi_0}^{\phi_{Bi}}L}{u^*b_i} \qquad (3\text{-}15)$$

$$DG3_i=\frac{\epsilon_{Col}Col_{\%\,Sat}\,b_i}{(1-\epsilon_{Col})}\,\frac{C_{A_{InletConc.}}}{C_{Bi_0}} \qquad (3\text{-}16)$$

u^* is a reference fluid velocity (percolation velocity) which has arbitrarily been set at 1m per 24 hours. It is important to note that the definition of the reference fluid velocity is totally arbitrary and only serves as a convenient manner to non-dimensionalise the results. Also note that:

$$u_{Percolation}=\frac{u}{\epsilon_{Col}Col_{\%Sat}} \qquad (3\text{-}17)$$

Following on, T is the equivalent of a reference space time for the column; DG1 is the ratio of the fluid *percolation* velocity to the reference fluid *percolation* velocity; DG2 is the ratio of the chemical reaction rate at z=0 to the rate of fluid reactant replenishment and DG3 is a dimensionless stoichiometric ratio which indicates fluid reagent strength relative to the solid reactant within the deposit. The rate of fluid reactant replenishment is defined to be the rate at which the fluid in a given volume of the deposit is totally replaced by new fluid due to the convective flux of fluid reactant through the lysimeter.

Equations (3-4) and (3-5) in dimensionless form and in cylindrical co-ordinates are summarised as:

$$\frac{\partial\alpha}{\partial\tau'}=-DG1\frac{\partial\alpha}{\partial\xi}-\sum_{i=1}^{n}DG2_i\alpha\sigma_{Bi}^{\phi_{Bi}} \qquad (3\text{-}18)$$

with

$$\alpha(\xi,0)=0 \qquad (3\text{-}19)$$

$$\alpha(0,\tau') = 1 \qquad (3\text{-}20)$$

$$\frac{d\sigma_{Bi}}{d\tau'} = -DG2_i \; DG3_i \alpha \sigma_{Bi}^{\phi_{Bi}} \qquad (3\text{-}21)$$

with

$$\sigma_{Bi}(\xi,0) = 1 \qquad (3\text{-}22)$$

Equations (3-18) and (3-21) represent the progression of reaction within the deposit.

3.3 Solution Strategy.

Equation (3-18) is a first order hyperbolic partial differential equation. Simple finite difference methods and finite element numerical techniques must be used with care when solving hyperbolic problems since the discontinuous nature of the solution gives rise to difficulties when these techniques are used. The discontinuous nature of the solution arises due to the fact that a sharp reaction front, corresponding to the fluid reagent front, moves through the deposit. Although it may be possible in some cases to obtain a stable method, it will invariably be very inaccurate. This implies that the solution strategy may calculate a solution but that this solution is incorrect. Note that solution strategies can also be accurate but not stable. This implies that although the correct solution is being calculated, the strategy breaks down before the entire solution is generated. Suitable solution strategies need to be both stable and accurate.

The method of characteristics is a suitable method of solution for first and second order hyperbolic partial differential equations. This method converts the partial differential equation into a set of simultaneous ordinary differential equations. The ordinary differential equations can then be solved using the standard numerical techniques of finite differences or finite elements.

Recall equation (3-18):

$$\frac{\partial \alpha}{\partial \tau'} = -DG1 \frac{\partial \alpha}{\partial \xi} - \sum_{i=1}^{n} DG2_i \alpha \sigma_{Bi}^{\phi_{Bi}} \qquad (3\text{-}23)$$

Using the method of characteristics, which is summarised in Appendix I, the following set of ordinary differential equations is obtained:

Only two of these equations are independent. The two equations used in the solution strategy are:

$$\frac{d\tau'}{1} = \frac{d\xi}{DG1} = -\frac{d\alpha}{\sum_{i=1}^{n} DG2_i \alpha \sigma^{\phi_{Bi}}} \qquad (3-24)$$

$$\frac{d\tau'}{d\xi} = \frac{1}{DG1} \qquad (3-25)$$

and

$$\frac{d\alpha}{d\tau'} = -\sum_{i=1}^{n} DG2_i \ \alpha \sigma^{\phi_{Bi}} \qquad (3-26)$$

Equation (3-25) represents the relationship between the time increment and the spatial increment. This highlights the essence of the method of characteristics. By imposing this restriction (of interdependence between two of the parameters) the partial differential equation can be reduced to a simple ordinary differential equation.

A simple explicit finite difference numerical technique was found to be an adequate method to solve equations (3-26) and (3-21). Equation (3-26) in numerical format is:

$$\alpha_{j+1} = \alpha_j - \Delta\tau' (DG2 \alpha_j \sigma_j^{\phi_{Bi}}) \qquad (3-27)$$

Equation (3-21) in numerical format is:

$$\sigma_{j+1} = \sigma_j - \Delta\tau' (DG2 \ DG3 \ \alpha_j \sigma_j^{\phi_{Bi}}) \qquad (3-28)$$

where j represents a time index.

3.4 Introduction of a Variable Fluid Velocity into the Solution Strategy.

At this point the velocity of the fluid percolating through the column has been assumed to be constant with respect to time. However, fluid velocity through granular waste deposits does vary as a function of time due to various reasons of which periodic rainfall is one example. This has been incorporated into the model in the following manner.

The dimensionless group, *DG1*, has been redefined as a function of dimensionless time:

$$DG1 = fn(\tau') \qquad (3\text{-}29)$$

Thus equation (3-25) becomes:

$$\frac{d\tau'}{d\xi} = \frac{1}{DG1(\tau')} \qquad (3\text{-}30)$$

Where before the time increment was constant, it now needs to be evaluated using equation (3-30) before each iteration in time. In the case of a functional relationship for the dimensionless group *DG1* being prescribed, equation (3-30) can be used to determine the dimensionless time increment. However, it is more likely that the functional relationship for the velocity will be prescribed in dimensional terms. (Either in terms of a mathematical function or discretely.) In this case, the following equation, derived from equation (3-30), can used to determine the *dimensionless* time increment:

$$\Delta\tau' = \frac{u^*(t_{n+1} - t_n)}{L} \qquad (3\text{-}31)$$

where $(t_{n+1}-t_n)$ is obtained from:

$$\int_{t_n}^{t_{n+1}} u(t)\, dt = \epsilon_{Col}\, \Delta z$$
$$= \epsilon_{Col}\, \Delta\xi\, L \qquad (3\text{-}32)$$

where Δz is the length of a spatial increment within the deposit; and;
L is the total height of the deposit.

3.5 Suitable Computer Routines for the Model.

Program Model4D1.PAS and Model4D2.PAS are PASCAL codes which solve these equations as a function of position and time. These codes make provision for the fluid velocity to vary as a function of time. The output of the codes include graphs of concentration versus position and time, with breakthrough curves as a function of time. The breakthrough concentration of any dissolved species in Model4D1.PAS has been calculated as the amount of that species exiting the column in the time increment over the total original leachable amount of that species in the column. Model4D2.PAS

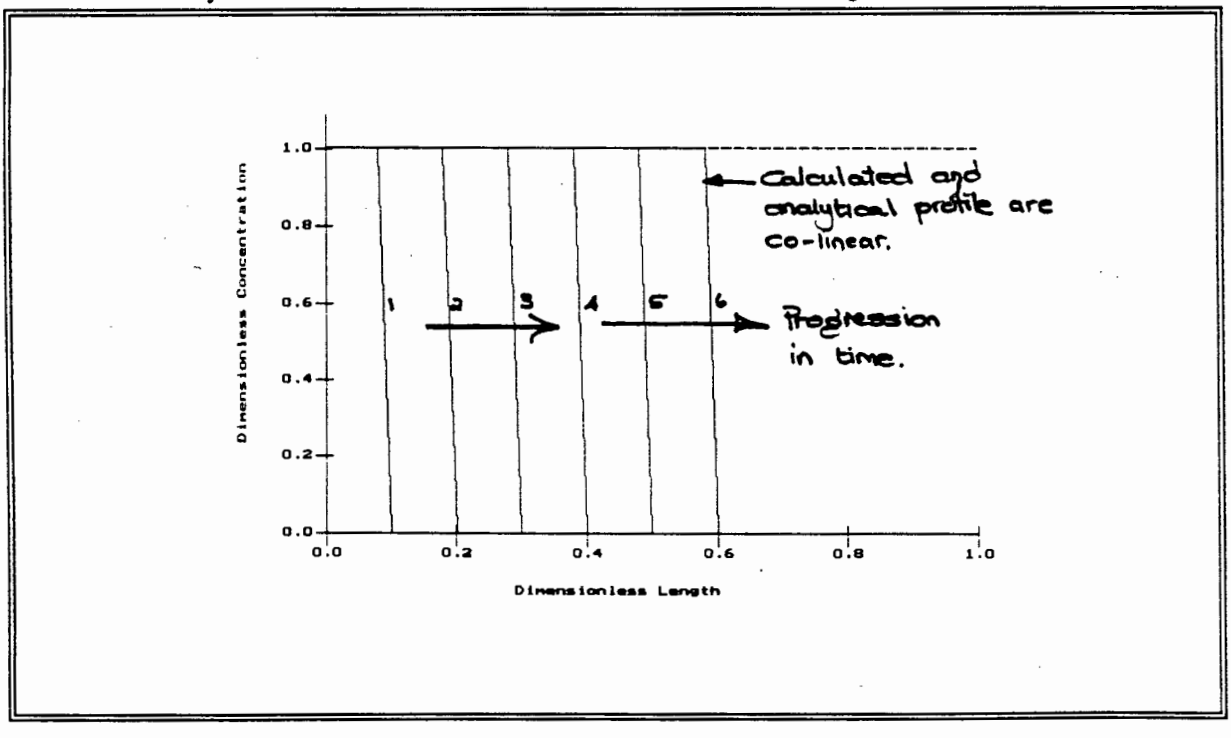
normalises all the breakthrough concentrations of the dissolved species relative to the total original amount of the first listed solid species in the program codes. This allows comparison between the relative amounts of the different species which appear in the leachate at the bottom of the column. As an example where such a comparison is useful, consider a waste which contains one leachable hazardous constituent and a large excess of buffering material. Although the fraction of the total buffering material released at any one time may be low compared to the hazardous constituent, the dissolved concentration of the buffer could be equal to, or much higher than that of the dissolved hazardous constituent concentration due to the large initial excess of the buffer material.

A copy of the codes as well as a solution algorithm can be found in Appendix II.

3.6 Verification of the Computer Routines.

The code was verified by subjecting it to a set of tests. In the first test no reaction was assumed to take place. This represents the problem of fluid reactant progressing through a deposit in plug flow. Excellent agreement was obtained between the calculated and analytical fluid profile as a function of time and is shown in Figure 3-2. Figure 3-2 represents the fluid reagent profiles within the deposit at successive time steps, beginning at the left hand side and progressing towards the right hand side of the graph.

Figure 3-2. Comparison of the predicted flow profile with the analytical profile for a lysimeter in which no chemical reactions take place.



The next test involved introducing a single reaction and determining its effect on the

concentration profiles as the rate of reaction increased. The deposit would be expected to react in a homogenous manner for a very slow reaction, proceed through a transitional phase at intermediate reaction rates and finally react in a zone-wise manner for fast reaction rates. A zone-wise reaction refers to the case where the reaction is restricted to a narrow band within the deposit. This can be clearly seen in the horizontal rows of graphs in Figure 3-5 where the reaction rate increases from left to right.

The code was also checked to ensure that it could calculate the profiles when more than one reaction was occurring. Several other self-checking strategies were used to ensure that the code was operating correctly. These included the following:

- The model was checked to ensure that chemical species were interchangeable. This ensures that the order in which the chemical species are defined in the computer routines is not important. This is demonstrated in Figure 3-3. In Figure 3-3, the profiles of the first solid reactant are depicted by solid lines while the profiles of the second solid reactant are depicted by dotted lines. (In Figure 3-3 these profiles never touch the X-Axis. Unfortunately the fluid reactant profiles are also depicted by solid lines but they can be identified as the curves which touch the X-Axis.)

- The model was checked to ensure that the concentration profiles of two solid species reacting at the same rate were co-incident. This is demonstrated in Figure 3-4. In this figure the dotted lines cannot be identified because they are co-incident with the solid lines.

- Kinetic aspects of the model were also checked by ensuring that the model predicted the same profiles for a deposit with a single solid reactant compared to a deposit which contained two solid reactants in equal quantities to the solid reactant in the first lysimeter but which each reacted at half the rate.

Figure 3-3. Chemical species in the model are interchangeable.

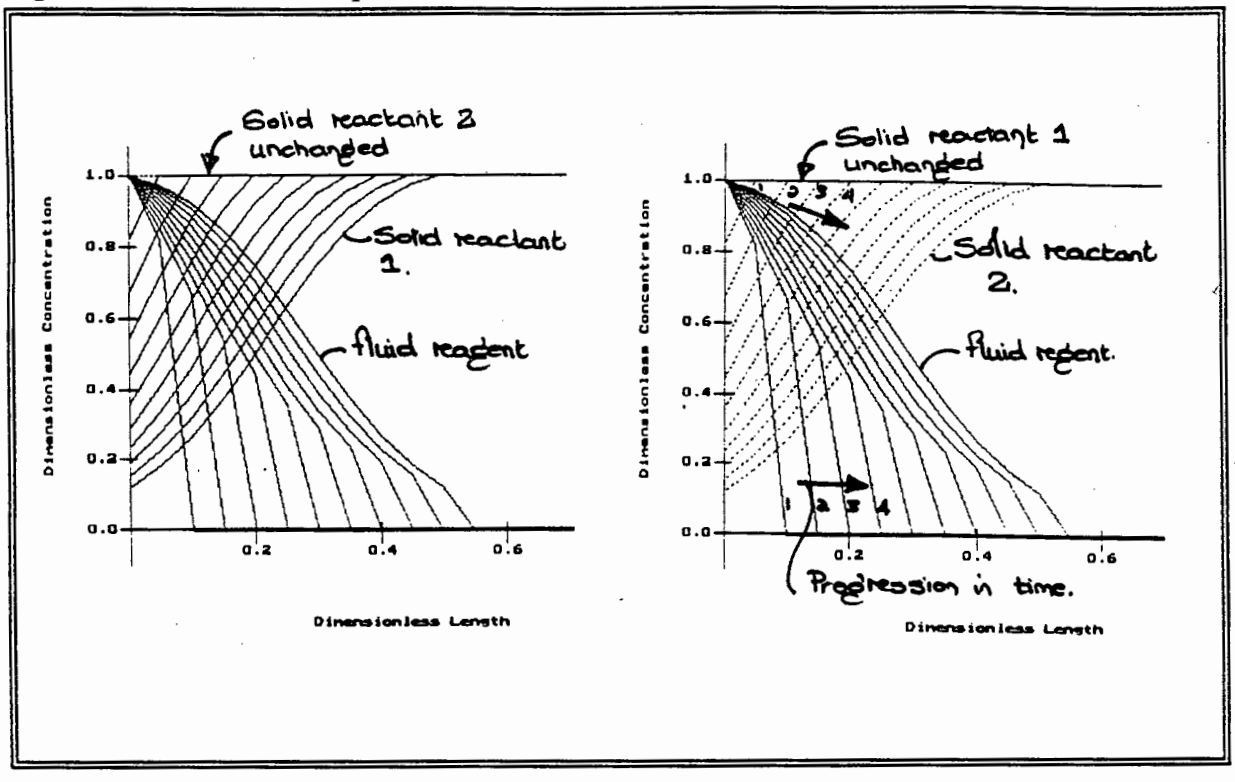
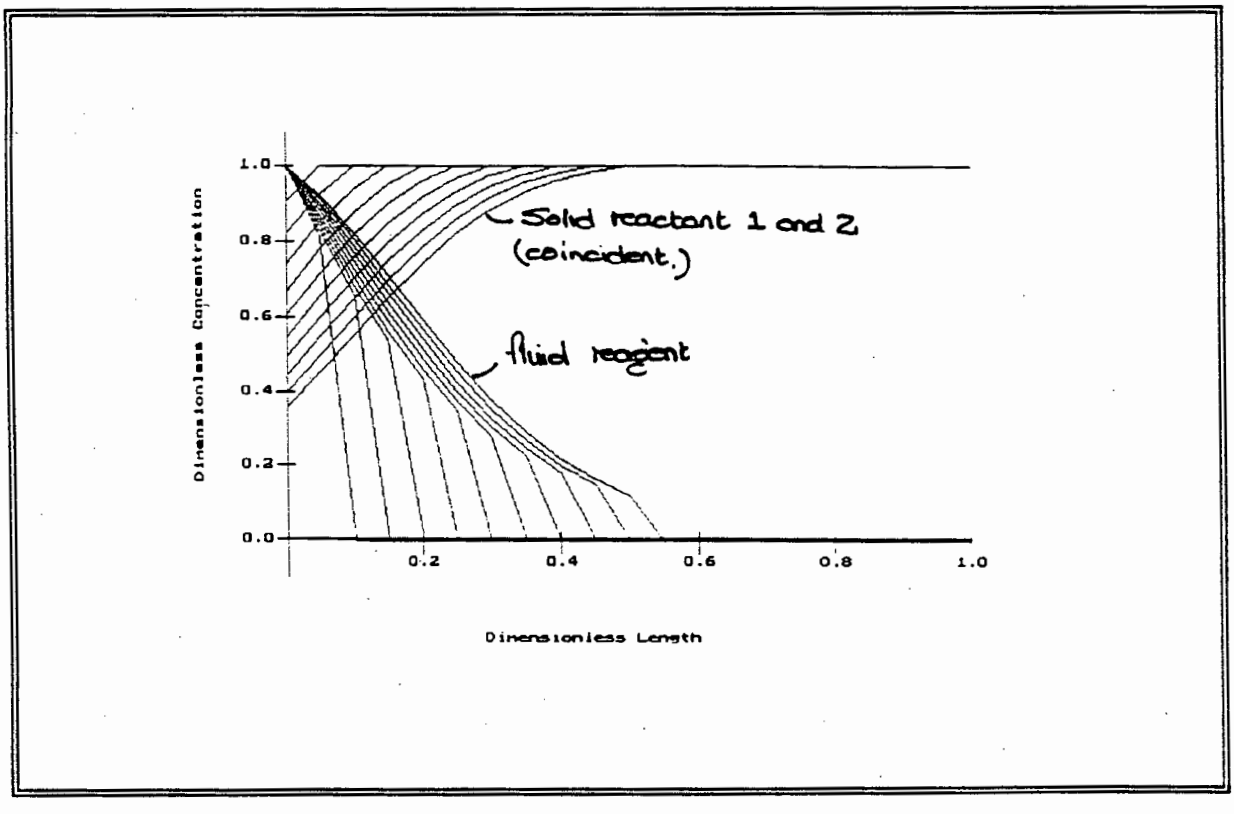


Figure 3-4. Profiles of hazardous constituents which react at the same rate are co-incident.



3.7 Results and Discussion.

3.7.1 General Behaviour.

In order to determine the general behaviour of the model and the sensitivity to the model parameters, the following scenario was investigated:

- Deposit of 0.5m in length;
- Only one solid reactant present;
- First order rate dependence in the solid reactant concentration;
- Constant fluid velocity set at 1m in 24 hours.

These conditions are typical of those used in laboratory lysimeter experiments. In a similar manner, full scale deposit behaviour could have been investigated by merely using appropriate model parameters.

The computer code calculated the dimensionless group *DG1* which under the conditions defined is 0.0084.

Figure 3-5 presents the profiles of fluid reagent A (solid curves) and one solid reactant (dashed curves) for all combinations of *DG2* = 0.1,1,10 and 100, and, *DG3* =0.1,1,10 and 100. Figure 3-6 presents the breakthrough curve of the solid reactant for the same combinations of parameters. Note that in Figure 3-6 each row of graphs have the same Y axis scaling, however the Y axis scaling changes between the different rows.

For very low values of *DG2* the chemical reaction rate is much slower than the rate of fluid reactant replenishment. Under these conditions, most columns would be expected to react in a homogenous manner. This behaviour can clearly be seen in Figure 3-5 by examining the first column of graphs. Only at very high relative fluid reagent concentrations are any solid reactant concentration gradients established. The release of hazardous constituents in columns with low values for *DG2* is controlled only by kinetic factors.

At moderate values for *DG2*, the type of kinetics depends on the relative fluid reactant concentration. At *DG2*=10, non-transient fluid and solid reactant gradients are established for all values of *DG3*. When *DG3* values are greater than 10, the column reacts in a fluid reactant limiting, zone-wise manner.

At high values of *DG3* the column reacts only in a fluid reactant limiting, zone-wise manner.

Figure 3-5. Fluid Reagent Profiles (solid lines) and Solid Reactant Profiles (dashed lines) within a Lysimeter Predicted by Model4D1 for Various Parameters.

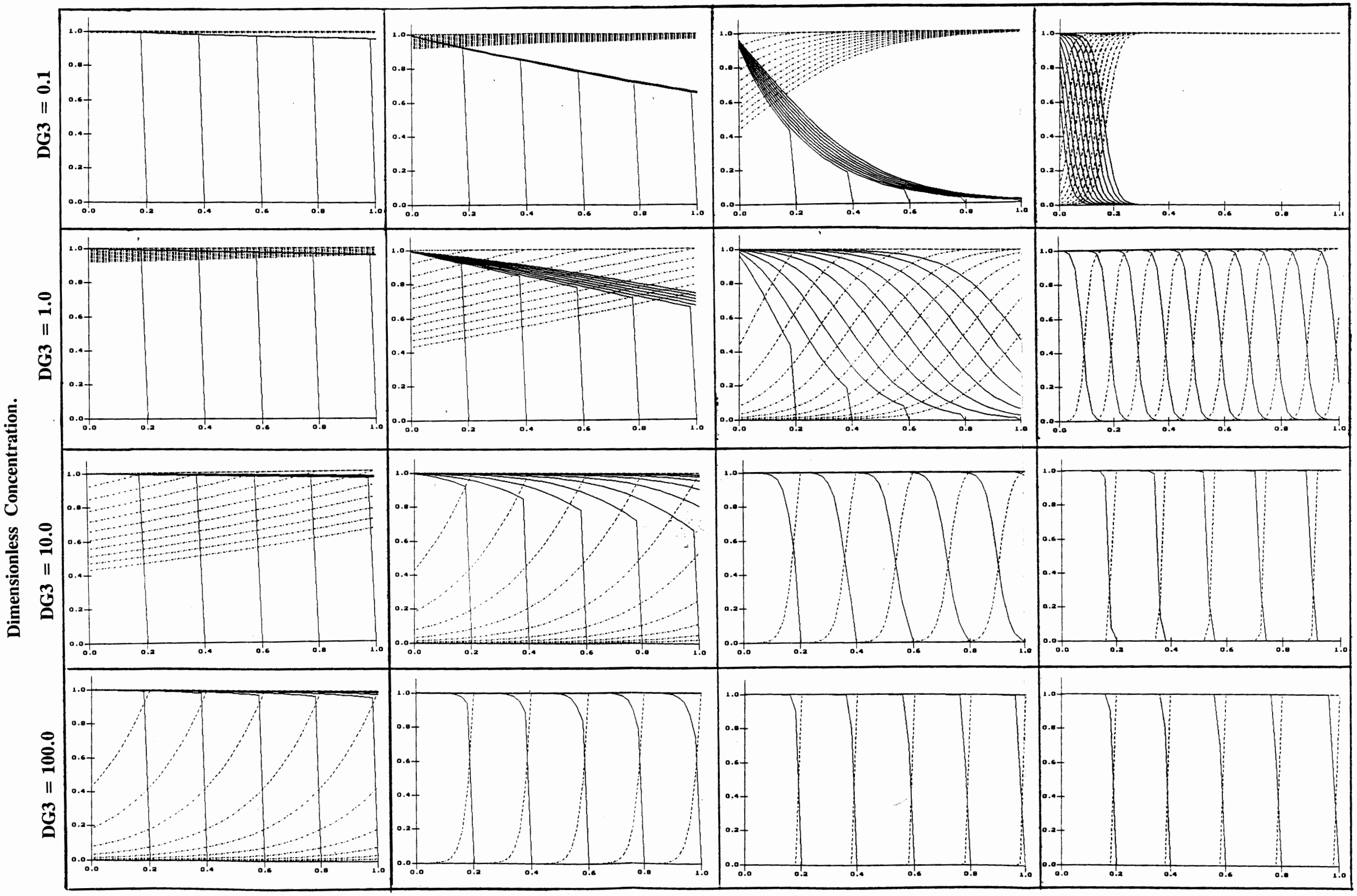
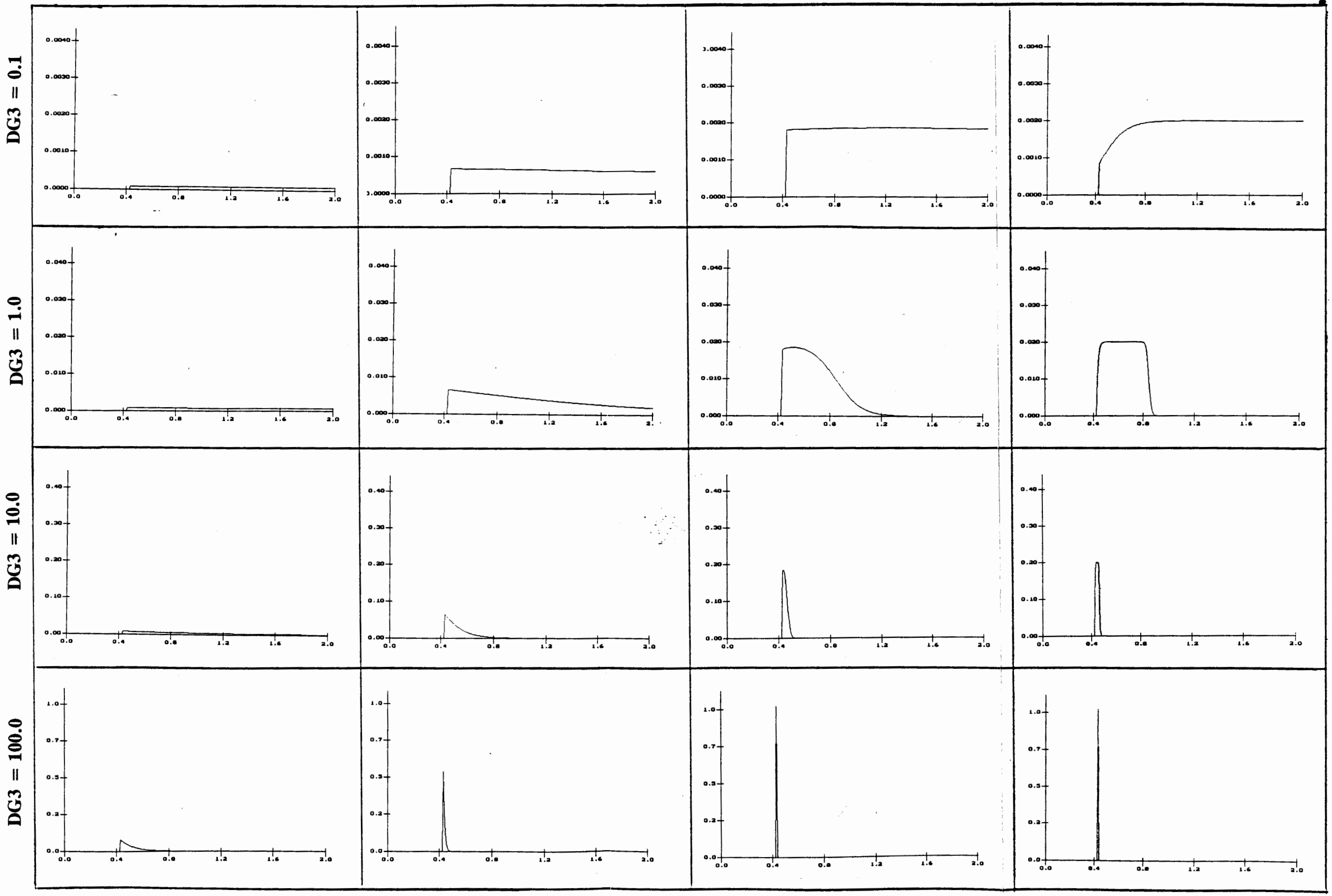


Figure 3-6. Conversion Curves for a Lysimeter Predicted by Model4D1 for Various Parameters.



Some interesting features of hazardous constituent release can be observed in Figure 3-6. Notice that the breakthrough curves for all columns which react in a homogenous manner resemble an exponential decay function. This is as expected, since a column which reacts in a homogenous manner could be considered as a continuously stirred tank reactor. The concentration profiles for first order reactions which take place in continuously stirred tank reactors are exponential decay functions [Levenspiel 1972]. In contrast the breakthrough curves for columns which react in a zone-wise manner resemble impulse and Heaviside functions (step up followed by a step down function) which are characteristic of perfect plug flow reactors. When the kinetics of the chemical reactions are sufficiently fast for the column always to react in a zone-wise manner, the width of the Heaviside function is determined by the relative fluid reactant concentration. As the fluid reactant concentration increases, the width of the heaviside function decreases. In the limit, the heaviside function approaches a Dirac delta as can be seen in the lower right hand corner of Figure 3-6.

3.7.2 Effect of Competing Reactions.

As previously discussed in Chapter 2, granular waste deposits very often contain more than one reactive component in the matrix. This has a marked effect on the release of the individual hazardous constituents as the following test scenarios indicate.

The effect of a buffering material on the release of a single solid hazardous constituent.

The scenario investigated can be summarised as:

- Deposit of 0.5m in length;
- One primary solid hazardous constituent and one buffer material present;
- First order rate dependence with respect to the solid concentrations;
- Constant fluid velocity set at 1m in 24 hours.

The parameters for the solid hazardous constituent were set at $DG2_{Contam.}=10$ and $DG3_{Contam.}=1$.

The combinations investigated are summarised in Table 3-1.

Table 3-1. Summary of the parameter combinations investigated to determine the effect a buffer material on hazardous constituent release.

	Buffer rate constant 100 times slower than contaminant rate const.	Buffer rate constant 10 times slower than contaminant rate const.
10 times as much buffer as contaminant.	$DG2_{Buffer} = 1.00$ $DG3_{Buffer} = 0.10$	$DG2_{Buffer} = 10.00$ $DG3_{Buffer} = 0.10$
100 times as much buffer as contaminant.	$DG2_{Buffer} = 10.00$ $DG3_{Buffer} = 0.01$	$DG2_{Buffer} = 100.00$ $DG3_{Buffer} = 0.01$

Figure 3-7 presents the concentration profiles for the above combinations while Figure 3-8 summarises the breakthrough curves.

The concentration profiles in the top left hand corner of Figure 3-7 closely resemble the concentration profiles when no buffer material was present. The breakthrough curves for the same parameter specification show that in this case the buffer material is not sufficiently reactive to affect the release of the solid hazardous constituents. By merely increasing the relative concentration of the buffer material it begins to play a significant role. Alternatively a more reactive buffer material can be used to limit the release of hazardous constituents.

The buffering material effectively forces the column to react in a more fluid reactant limited, zone-wise manner. This is because the total rate of acid consumption has effectively increased. Further, due to the competition between the solid hazardous constituent and the buffer material for fluid reactant, the hazardous constituent reacts at a slower rate than before and the hazardous constituent breakthrough curves start to resemble exponential decay functions.

Notice in the lower row of breakthrough graphs in Figure 3-8 that the solid hazardous constituent persists to breakthrough at large times. This is typical behaviour observed when competing reactions occur.

Figure 3-7. Concentration Profiles for Two Competing Reactions.

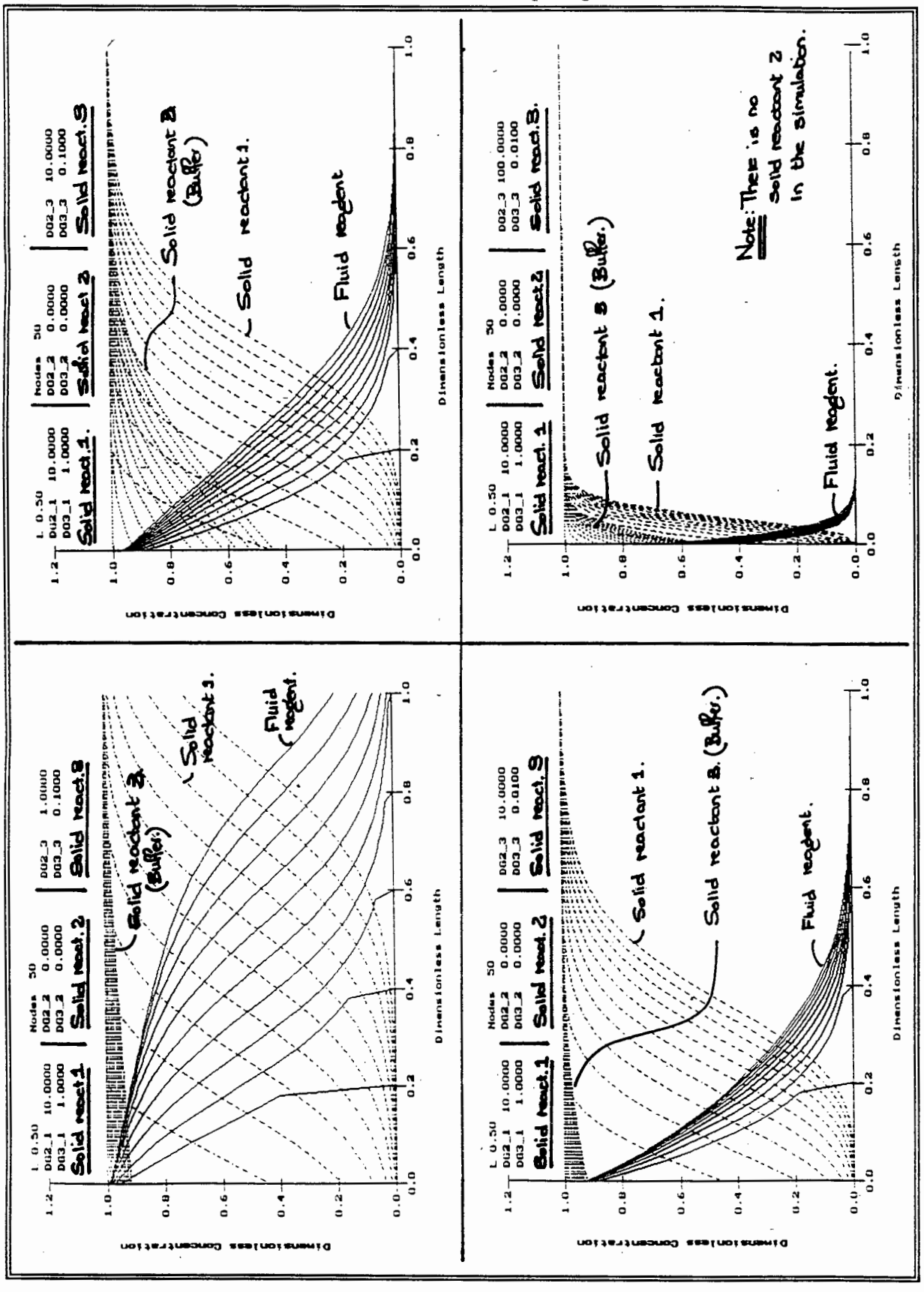
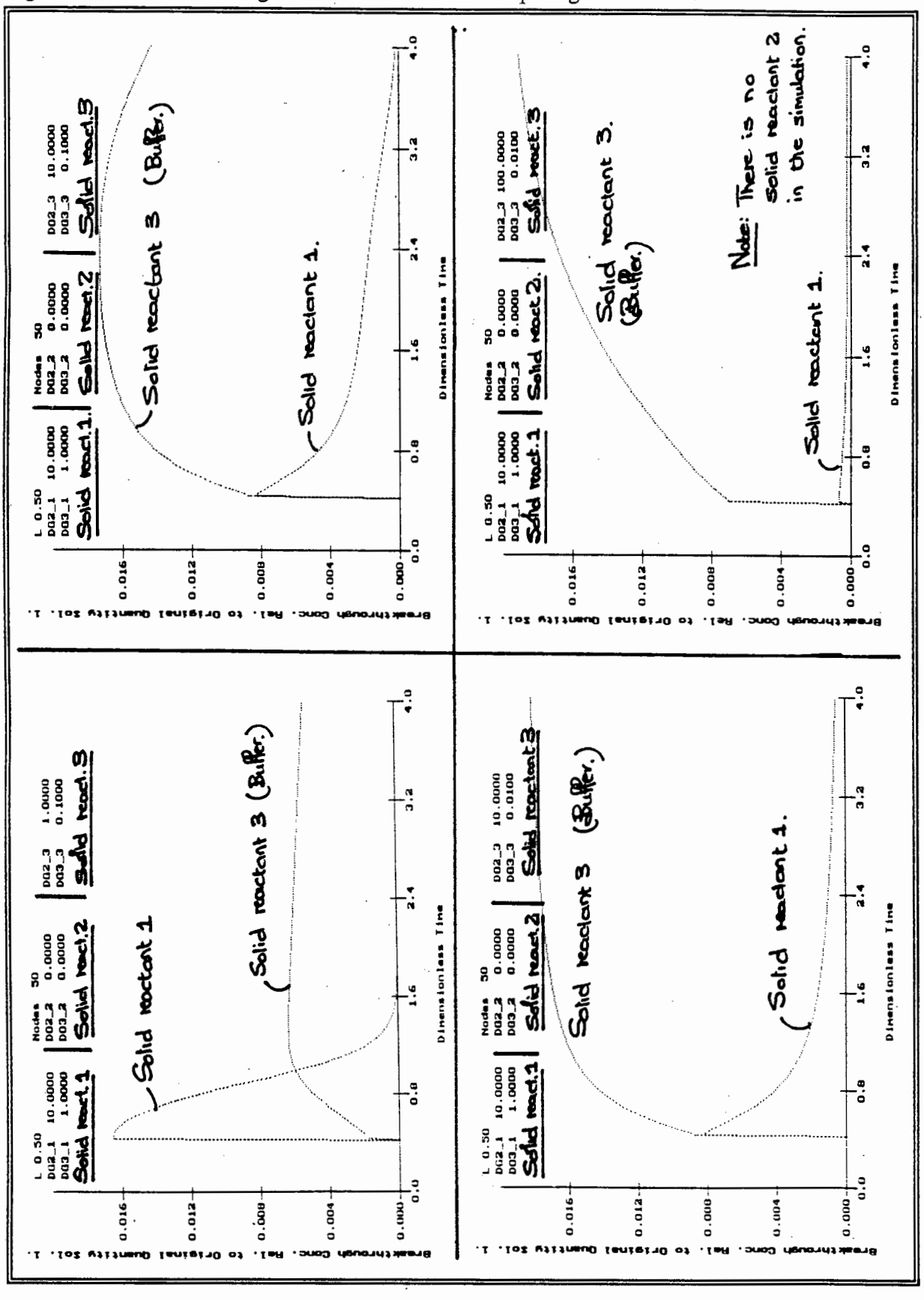


Figure 3-8. Breakthrough Curves for Two Competing Reactions.



Extension to more than one reactive hazardous constituent.

Figures 3-9 and 3-10 are sample printouts of typical concentration profiles and breakthrough curves when two hazardous constituents and a buffer material are present. The scenario investigated was a second hazardous constituent of equal amount to the first, but which reacted 10 times slower than the first reactant. These parameters are summarised in Table 3-2:

Table 3-2. Summary of parameters used to demonstrate extension of the model to more than one reactive hazardous constituent.

Contaminant 1.	*DG2_1* = 10.00 *DG3_1* = 1.00
Contaminant 2.	*DG2_2* = 1.00 *DG3_2* = 1.00
Buffer Material.	*DG2_3* = 10.00 *DG3_3* = 0.01

Figure 3-9. Sample printout of concentration profiles for more than one hazardous constituent.

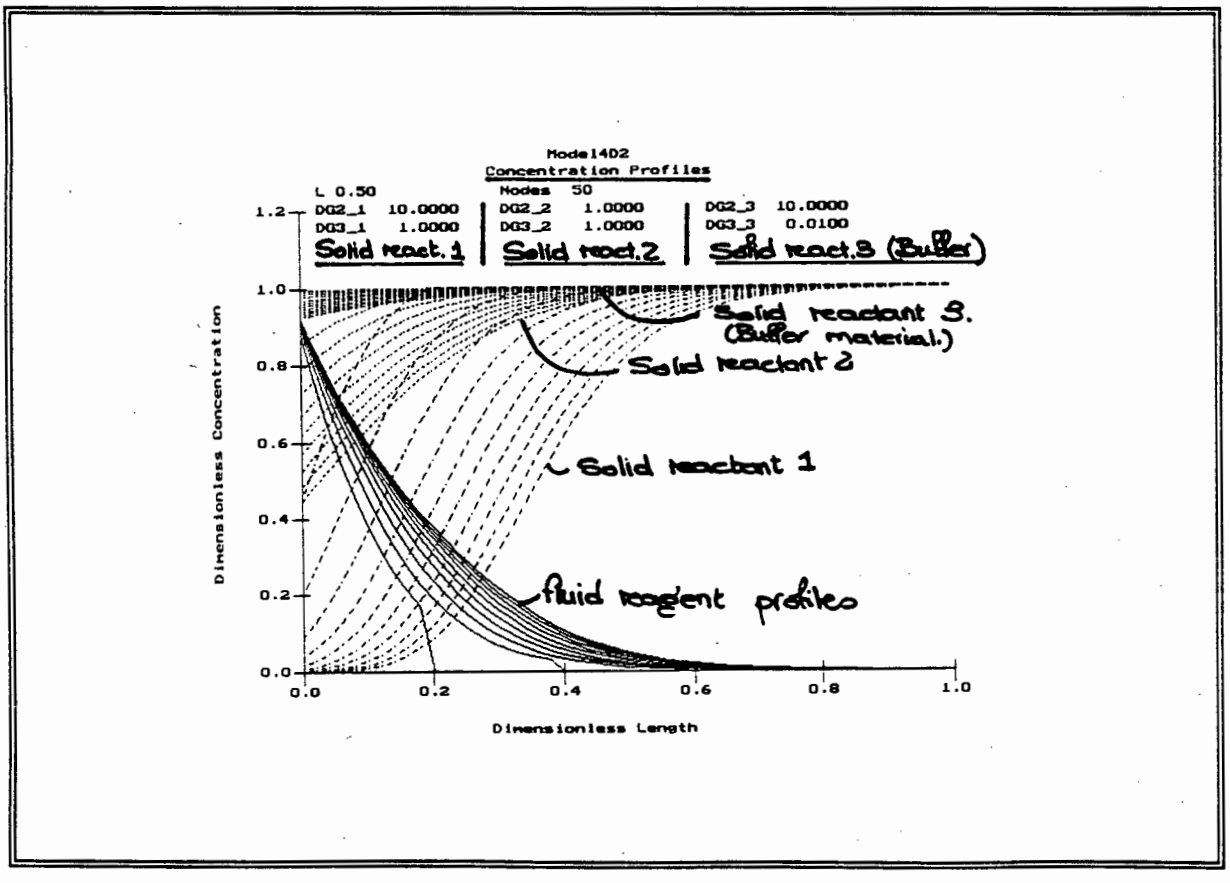
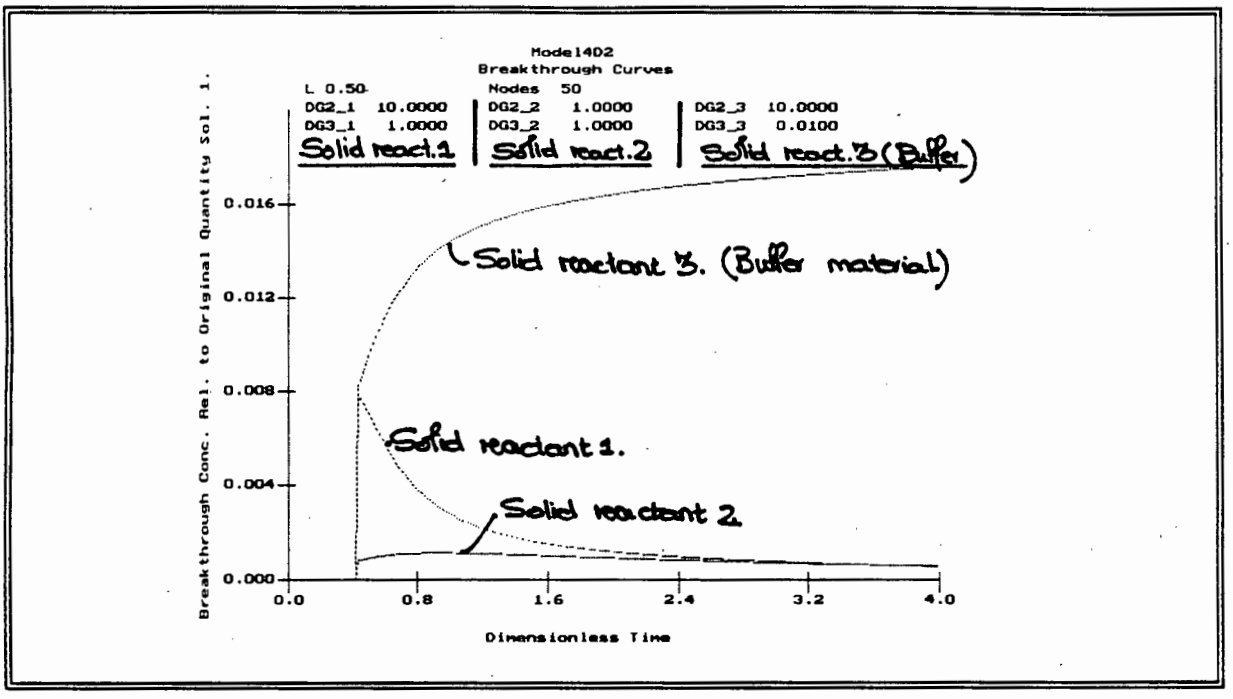


Figure 3-10. Sample printout of breakthrough curves for more than one hazardous constituent.



3.8 Fitting the Model to Lysimeter Experiments.

3.8.1 Model Requirements.

The following information needs to be known in order to fit the model to lysimeter results:

- The basic chemical reactions which actually take place within the lysimeter;
- the effective concentrations of the participating species;
- the reaction orders with respect to the solid reactants;
- the percolation velocity of the fluid as a function of time; and;
- the voidage of the column and the void space saturation.

The level of detail required in the understanding of the basic chemical reactions is to have identified the fluid reagent and the main solid reactants in the system. For example the fluid reagent is usually a dilute acid stream while the solid reactants are leachable heavy metals and buffering components.

The effective concentration refers to the total leachable concentration of any species, which in many instances is *not* equal to the total concentration of that species. This is caused by a certain portion of the species being unavailable to the leaching process. One method to determine this concentration is to conduct a CSTR leach test until equilibrium is reached. The initial effective solid concentrations of the dissolved species can then be determined by back calculation.

Where the solid reactant is not in excess, a solid reactant order of unity is suggested for the following reasons. The solid reactants are known to be located in a size distribution of waste particles within the deposit. If the deposit contained equi-sized spherical waste particles, comparison of the variable order kinetic expression, equation (3-2), to a 'grain' reaction model indicates that the appropriate solid reactant order in this case would be 2/3. Dixon [1992] has intimated that this order increases as the standard deviation of the size distribution of spherical 'grains' increases and that a reaction order of unity is reasonable. The appropriate reaction order for excess solid reactant is usually taken to be zero, which implies that the solid reactant concentration does not have an effect on the reaction kinetics.

Usually the percolation velocity is set at a constant value. As previously discussed the model can accommodate a variable fluid velocity as long as the functional relationship for the velocity is prescribed.

3.8.2 Fitted parameters.

Each dimensionless group, $DG2_i$, contains a lumped reaction rate constant for the chemical reaction corresponding to that group. These reaction rate constants cannot be determined *a priori* and so the fitted parameters in the model are the $DG2_i$ groups. The reason why the reaction rate constant cannot be determined from bench scale tests is because they include hydrodynamic effects. The hydrodynamic situation in bench scale tests is not comparable to the hydrodynamic situation in a waste deposit or its pilot-scale lysimeter equivalent.

The $DG2_i$ groups are fitted by comparing the breakthrough curves from the model to experimentally determined breakthrough curves from lysimeters. A good method to optimise the agreement between the model and experimental results is to use a simplex search technique to determine the best values for the fitted parameters.

3.9 Limitations of the Model.

The macroscopic, lumped parameter model does have several limitations due to its inherent simplicity. The first limitation of the model is that it can only be used to extrapolate to deposit proportions and in time for deposits which contain identical wastes and which exhibit identical hydrodynamic characteristics to the ones observed in the

lysimeter. This implies that a separate lysimeter experiment needs to be conducted for each waste stream and hydrodynamic situation investigated. This is a severe limitation considering the expense and time involved with lysimeter experiments.

The second limitation is that the model can be used only in situations where each solid reactant is released at a single overall rate. Often, particles which have most of the solid reactant concentrated onto the surface of the particle, can also have some of the solid reactant distributed in the bulk of the particle. The solid reactants in the two regions very often exhibit different rates of release. The first rate of release is due to the release of the solid reactants on the surface. Usually this rate of release is much faster than that of the release from within the bulk of the particle due the added diffusional resistances of the fluid reactant into the particle.

The model cannot be used to determine the individual contribution of the intrinsic chemical kinetics, the hydrodynamic aspects or the hazardous constituent location on the release of hazardous constituents. This is a direct consequence of using the lumped parameter approach. Since one aim of this work is to eventually be able to engineer better waste deposits, it is critical to be able to determine these individual contributions.

For these reasons it was deemed necessary to investigate more complex models which would begin to address the limitations of the macroscopic, lumped parameter model. In effect this implied investigating models which describe the release of hazardous constituents at the particle level. This is the focus of the remainder of this thesis.

Chapter 4. A Summary of the Chemical Reaction Model Applicable to Single Particles as Developed by Dixon [1992].

The investigation into the macroscopic, lumped parameter model revealed a need to describe the release of hazardous constituents at the particle level. As discussed in Chapter 2, Dixon [1992] has developed a chemical reaction model for investigating the leaching behaviour of precious metals from ore particles. His model is sufficiently detailed to include the effects of diffusive and chemical reaction kinetic resistances, competing reactions and precious metal location within the particle on the leaching behaviour. Since these considerations are similar to the ones encountered in leaching of hazardous constituents from waste deposits, it was felt that Dixon's model should be investigated to determine its applicability to modelling contaminant leaching from waste particles.

This chapter presents the details of the investigation into the applicability of Dixon's model to a single waste particle. This involved determining whether or not the model developed by Dixon could be used without modification. For this reason, this chapter summarises the model development followed by Dixon [1992]. Once it was determined that Dixon's particle scale model could be used, an appropriate solution strategy for the model was investigated and implemented. The computer routines which implement the solution strategy have been rigorously checked against the results obtained by Dixon. These details form the reminder of this chapter.

4.1 Development of the Equations.

Figure 4-1 depicts a porous, spherical particle of radius R which is submerged in fluid reactant and which contains small amounts of solid reactant deposits. Dixon [1992] assumed that these solid reactants, B_i, are dissolved by a single fluid reagent A. This is represented by:

$$A + \sum_{i=1}^{n} b_i B_i \rightarrow dissolved\ products$$

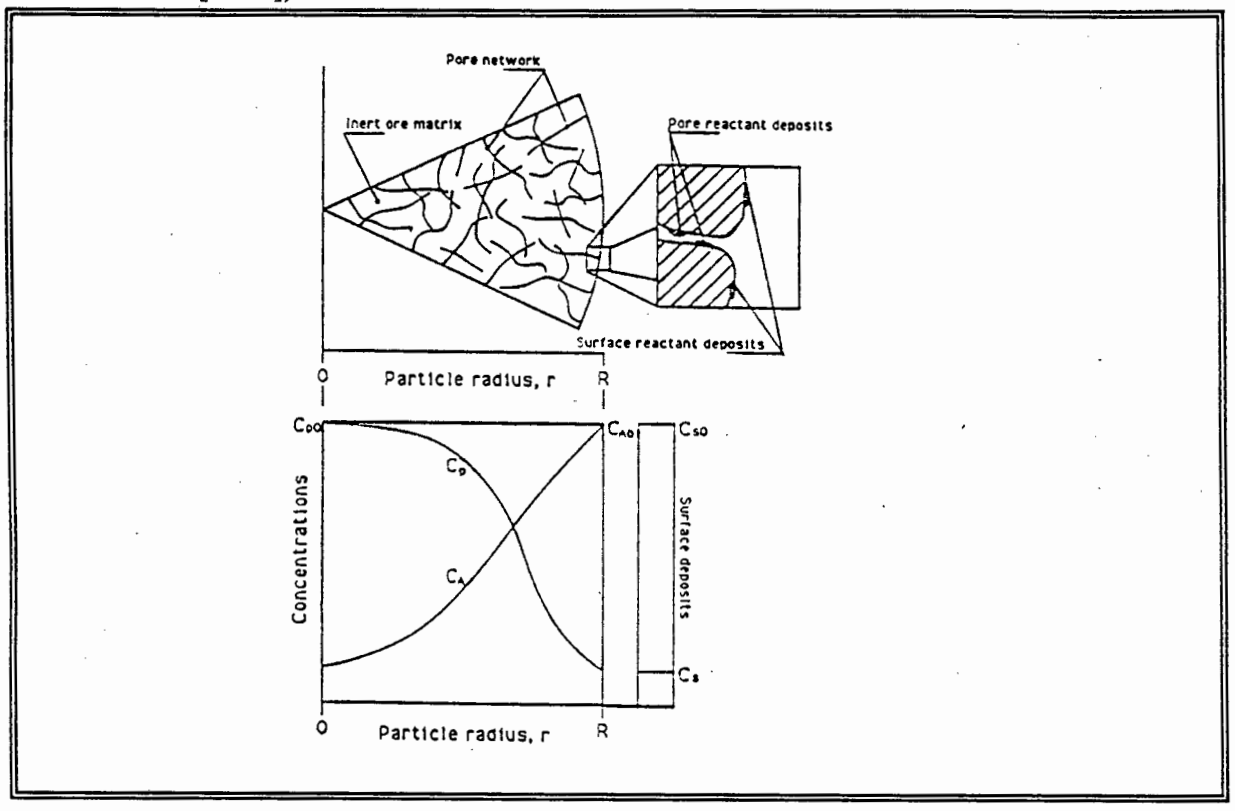
As in Chapter 3, the continuity equation for the fluid reactant can be obtained from a statement of conservation of mass. The continuity equation applicable to a single particle is:

$$\nabla \cdot N_A + \epsilon_0 \frac{\partial C_A}{\partial t} - \Sigma\, r_{A_i} = 0 \qquad (4-1)$$

where N_A fluid flux into the particle;
ϵ_0 particle porosity; and;
r_{Ai} defined as the rate of *production* of fluid reagent A by reaction i.

As before, the summation sign is required to account for the production of fluid reagent A by all the participating reactions.

Figure 4-1. Schematic diagram of a porous, spherical particle of radius R and a graph showing the concentration gradients within the particle. (taken from Dixon [1992]).



Dixon made explicit provision for different kinetics for those reactions which occur in the bulk of the particle compared to those which occur on its surface. Considering first the reactions which occur within the pores of the particle, Dixon assumed that the reaction kinetics can be described by an expression which is of variable order with respect to the solid reactant and first order with respect to the fluid reagent. This expression can be summarised as:

$$\frac{dC_{pi}}{dt} = -k_{pi}\, C_{pi}^{\Phi_{pi}}\, C_A \qquad (4-2)$$

where C_{pi} mass or moles of the solid reactant in the pores of the particle *per unit mass of solid*;

C_A mass or moles of the fluid reagent *per unit volume of fluid*; and;
k_{pi} reaction rate constant.

Here the reaction rate constant has units, depending on the reaction order term Φ_{Bi}, such that the units on the right hand side of the equation are rendered to be mass or moles of solid reactant per unit time *per unit **mass** of solid.*

In situations where the kinetics of any species on the surface of the particle is different to the kinetics of that species in the bulk of the particle, another kinetic expression will be required. The kinetic expression adopted by Dixon can be summarised as:

$$\frac{dC_{si}}{dt} = -\frac{3k_{si}C_{si}^{\phi_{si}}C_A}{R\,\rho_0(1-\epsilon_0)} \qquad (4-3)$$

where C_{si} mass or moles of the solid reactant on the surface of the particle *per unit mass of solid*;
C_A mass or moles of the fluid reagent *per unit volume of fluid*;
R radius of the particle;
ρ_0 ore density; and;
k_{si} reaction rate constant.

The following equation makes it a bit easier to see how this expression was obtained:

$$\frac{Mass\ of\ particle}{Area\ of\ particle}\ \frac{dC_{si}}{dt} = -k_{si}C_{si}^{\phi_{si}}C_A \qquad (4-4)$$

Thus the units on the left hand side of the equation are mass or moles of solid reactant per unit time *per unit **area** of particle*. Therefore, the units of the reaction constant, k_{si}, depending on the reaction order term Φ_{si}, are such which render the units on the right hand side of the equation to be mass or moles of solid reactant per unit time *per unit **area** of the particle.*

It is important to note that if the kinetics of the species on the surface of the particle are not significantly different from the same species within the bulk of the particle, that the requirement for equation (4-3) is obviated.

Substituting equation (4-2) into equation (4-1) gives:

$$\nabla \cdot N_A - \rho_0(1-\epsilon_0)\sum_{i=1}^{n}\frac{k_{pi}C_AC_{pi}^{\phi_{pi}}}{b_i} = \epsilon_0\frac{\partial C_A}{\partial t} \qquad (4-5)$$

Dixon assumed that only a diffusive flux of fluid reactant entered the particles and made use of Fick's law to describe this flux. Substituting Fick's law into equation (4-5) and expressing the equation in spherical co-ordinates with radial dependence only gives:

$$D_e \left[\frac{\partial^2 C_A}{\partial r^2} + \frac{2}{r}\frac{\partial C_A}{\partial r}\right] - \rho_0(1-\epsilon_0)\sum_{i=1}^{n}\frac{k_{pi}C_{pi}^{\phi_{pi}}C_A}{b_i} = \epsilon_0\frac{\partial C_A}{\partial t} \qquad (4-6)$$

The initial and boundary conditions which apply are:

$$C_A(r,0) = 0 \qquad (4-7)$$

$$C_A(R,t) = C_{A_b} \qquad (4-8)$$

$$\frac{\partial C_A}{\partial t}(0,t) = 0 \qquad (4-9)$$

It is important to note that equation (4-8) effectively implies that no film mass transfer resistances are being considered. This assumption is only valid where the film mass transfer is fast compared to the diffusion of the fluid reactant into the particle and the reaction of the fluid reactant within the particle.

4.2 Expressing the Equations in Dimensionless Form.

Equations (4-3) and (4-6) to (4-9) can be expressed in dimensionless form by defining the following dimensionless parameters and dimensionless groups:

$$\alpha = \frac{C_A}{C_{A_0}} \qquad (4-10) \qquad \alpha_b = \frac{C_{A_b}}{C_{A_0}} \qquad (4-11)$$

$$\sigma_{pi} = \frac{C_{pi}}{C_{pi_0}} \qquad (4-12) \qquad \sigma_{si} = \frac{C_{si}}{C_{si_0}} \qquad (4-13)$$

$$\xi = \frac{r}{R} \qquad (4-14) \qquad \tau = \frac{D_{e_A} t}{\epsilon_0 R^2} \qquad (4-15)$$

$$\beta_i = \frac{\epsilon\, b_i C_{A_0}}{\rho\,(1-\epsilon)\,(C_{pi_0}+C_{si_0})} \qquad (4\text{-}16) \qquad \lambda_i = \frac{C_{si_0}}{C_{pi_0}+C_{si_0}} \qquad (4\text{-}17)$$

$$\kappa_{pi} = \frac{\rho\,(1-\epsilon)\,k_{pi} C_{pi_0}^{\phi_{pi}} R^2}{b_i D_{e_A}} \qquad (4\text{-}18) \qquad \kappa_{si} = \frac{3 k_{si} C_{si_0}^{\phi_{si}} R}{b_i D_{e_A}} \qquad (4\text{-}19)$$

β_i represents a dimensionless stoichiometric ratio which indicates the reagent strength relative to the grade of solid reactant i. In a particle of given porosity which is in contact with a fluid reactant of concentration $\alpha = 1$, a value of $\beta_i = 1$ would imply that there is sufficient fluid reactant within the pores of the particle to completely react all of the solid reactant.

λ_i represents the fraction of solid reactant residing on the surface of the particle.

κ_{pi} and κ_{si} are ratios of the reaction rate of solid reactant i within the particle pores and on the particle surface, respectively, to the porous diffusion rate of fluid reactant A. In an investigation into the effects of flow, diffusion and heat conduction on reactor performance, Damköhler recognised the importance of four dimensionless groups. The second of these is the ratio of the chemical reaction rate to the rate of diffusion [Aris 1975]. Thus κ_{pi} and κ_{si} correspond to Damköhler numbers of the second type. These ratios can be identified a bit more easily if the equations are written in the following format:

$$\kappa_{pi} = \frac{\dfrac{\rho\,(1-\epsilon)\,k_{pi} C_{pi_0}^{\phi_{pi}} C_A}{b_i}}{\dfrac{D_{e_A} C_A}{R^2}} \qquad (4\text{-}20) \qquad \kappa_{si} = \frac{\rho\,(1-\epsilon)\,\dfrac{3 k_{si} C_{si_0}^{\phi_{si}} C_A}{b_i R \rho\,(1-\epsilon)}}{\dfrac{D_{e_A} C_A}{R^2}} \qquad (4\text{-}21)$$

The equations in dimensionless form and in spherical co-ordinates are summarised as:

$$\frac{\partial^2 \alpha}{\partial \xi^2} + \frac{2}{\xi}\frac{\partial \alpha}{\partial \xi} - \sum_{i=1}^{n} \kappa_{pi} \sigma_{pi}^{\phi_{pi}} \alpha = \frac{\partial \alpha}{\partial \tau} \qquad (4\text{-}22)$$

with

$$\alpha(\xi, 0) = 0 \qquad (4\text{-}23)$$

$$\alpha(1,\tau)=\alpha_b \qquad (4\text{-}24)$$

$$\frac{\partial\alpha}{\partial\xi}(0,\tau)=0 \qquad (4\text{-}25)$$

$$\frac{d\sigma_{pi}}{d\tau}=-\frac{\kappa_{pi}\beta_i\sigma_{pi}^{\phi_{pi}}\alpha}{1-\lambda_i} \qquad (4\text{-}26)$$

with

$$\sigma_{pi}(\xi,0)=1 \qquad (4\text{-}27)$$

$$\frac{d\sigma_{si}}{d\tau}=-\frac{\kappa_{si}\beta_i\sigma_{si}^{\phi_{si}}\alpha_b}{\lambda i} \qquad (4\text{-}28)$$

with

$$\sigma_{si}(0)=1 \qquad (4\text{-}29)$$

Equations (4-22) to (4-29) represent the progression of the reaction within a single particle.

4.3 Suitability of Dixon's Model to Hazardous Constituent Leaching From Waste Particles.

The development of the equations used in Dixon's particle scale model do not include any aspects which are unique to precious metal leaching from ore particles. The same equations would have been determined if the leaching of hazardous constituents from a waste particle had been considered. For this reason, Dixon's equations can be used without modification to model hazardous constituent release from a waste particle.

4.4 Solution Strategy.

Equation (4-22) is a second order parabolic partial differential equation. A suitable solution strategy, which would be both stable and accurate, was desired to solve the set of equations. The Crank-Nicolson formula, which is an implicit finite difference method, is a suitable solution strategy since it is both unconditionally stable and sufficiently accurate [Crank 1975].

The Crank-Nicolson approach uses a Taylor series expansion of the concentration function to obtain expressions for the partial derivatives. These are:

$$\frac{\partial \alpha}{\partial \xi}\Big|_{i,j} \simeq \frac{\alpha_{i+1,j}-\alpha_{i-1,j}}{2\Delta\xi} \qquad (4-30)$$

$$\frac{\partial^2 \alpha}{\partial \xi^2}\Big|_{i,j} \simeq \frac{\alpha_{i+1,j}-2\alpha_{i,j}+\alpha_{i-1,j}}{(\Delta\xi)^2} \qquad (4-31)$$

where i is a spatial index and
j is a time index.

Crank and Nicolson [1947] suggested that a better approximation for the above quantities would be the average of the quantities evaluated at times j and j+1. This yields the Crank-Nicolson formulas for the partial derivatives:

$$\frac{\partial \alpha}{\partial \xi}\Big|_{i,j} \simeq \frac{1}{2}\left(\frac{\alpha_{i+1,j}-\alpha_{i-1,j}}{2\Delta\xi}+\frac{\alpha_{i+1,j+1}-\alpha_{i-1,j+1}}{2\Delta\xi}\right) \qquad (4-32)$$

and

$$\frac{\partial^2 \alpha}{\partial \xi^2}\Big|_{i,j} \simeq \frac{1}{2}\left(\frac{\alpha_{i+1,j}-2\alpha_{i,j}+\alpha_{i-1,j}}{(\Delta\xi)^2}+\frac{\alpha_{i+1,j+1}-2\alpha_{i,j+1}+\alpha_{i-1,j+1}}{(\Delta\xi)^2}\right) \qquad (4-33)$$

Using these formulas, equations (4-22), (4-26) and (4-28) were converted to numerical equations.

Equation (4-22) in numerical format is:

for i=0

$$\alpha_{0,j+1}\left(-6-\sum_{i=1}^{N}\kappa_p(\Delta\xi)^2\sigma^{\phi}_{0,j+1}-2\frac{(\Delta\xi)^2}{\Delta\tau}\right)+6\alpha_{1,j+1} =$$

$$\alpha_{0,j}\left(6+\sum_{i=1}^{N}\kappa_p(\Delta\xi)^2\sigma^{\phi}_{0,j+1}-2\frac{(\Delta\xi)^2}{\Delta\tau}\right)-6\alpha_{1,j} \qquad (4-34)$$

and for $i <> 0$

$$\alpha_{i-1,j+1}(i-1)+\alpha_{i,j+1}(-2i-\sum_{i=1}^{N}\kappa_p i(\Delta\xi)^2\sigma^{\phi}_{i,j+1}-2i\frac{(\Delta\xi)^2}{\Delta\tau})+\alpha_{i+1,j+1}(i+1)=$$

$$\alpha_{i-1,j}(-i+1)+\alpha_{i,j}(2i+\sum_{i=1}^{N}\kappa_p i(\Delta\xi)^2\sigma^{\phi}_{i,j}-2i\frac{(\Delta\xi)^2}{\Delta\tau})+\alpha_{i+1,j}(-i-1) \qquad (4-35$$

The reason for two equations being required is due to a singularity which exists at the origin in equation (4-22). This problem was overcome by noting that total symmetry exists at the origin and thus equation (4-22) can be expressed in cartesian co-ordinates at this point. *Note that the cartesian co-ordinates are only valid at the origin.* The equation used to derive equation (4-34) is:

$$\frac{\partial^2\alpha}{\partial\xi^2}-\sum_{i=1}^{n}\kappa_{pi}\sigma^{\phi_{pi}}_{pi}\alpha=\frac{\partial\alpha}{\partial\tau} \qquad (4-36)$$

The numerical format of equation (4-26) is:

$$\frac{\sigma_{i,j+1}-\sigma_{i,j}}{\Delta\tau}=-\frac{\kappa_p\beta}{2(1-\lambda)}(\sigma^{\phi}_{i,j}\alpha_{i,j}+\sigma^{\phi}_{i,j+1}\alpha_{i,j+1}) \qquad (4-37)$$

The numerical format of equation (4-28) is similar to equation (4-37).

4.5 Suitable Computer Routines for the Model.

Program Model2D2.PAS is a PASCAL code which solves these numerical equations as a function of position and time. The output of this code includes a graph of the solid and fluid reagent profiles within the particle as a function of time and a graph of the fractional conversions as a function of time. The fractional conversion is defined as the fraction of a particular solid reactant species which has been released.

A copy of the code can be found in Appendix III.

4.6 Verification of the Computer Routines.

The computer routines written needed to be checked to determine whether they operate correctly. One method to check the routines would be to reproduce the results presented in Dixon's thesis. To this end, all of the results presented in Chapter 1 of Dixon's thesis which involve concentration profiles and fractional conversions as a function of time have been reproduced. To demonstrate this, Figure 4-2 is a series of graphs produced by Model2D2. These graphs can be compared to those found in Figure 4-3 which are the graphs presented by Dixon for the same parameter specification.

Further comparisons between the results predicted by Model2D2 and Dixon's work can be seen in Figure 4-4 and 4-5; and; Figure 4-6 and Figure 4-7.

Figure

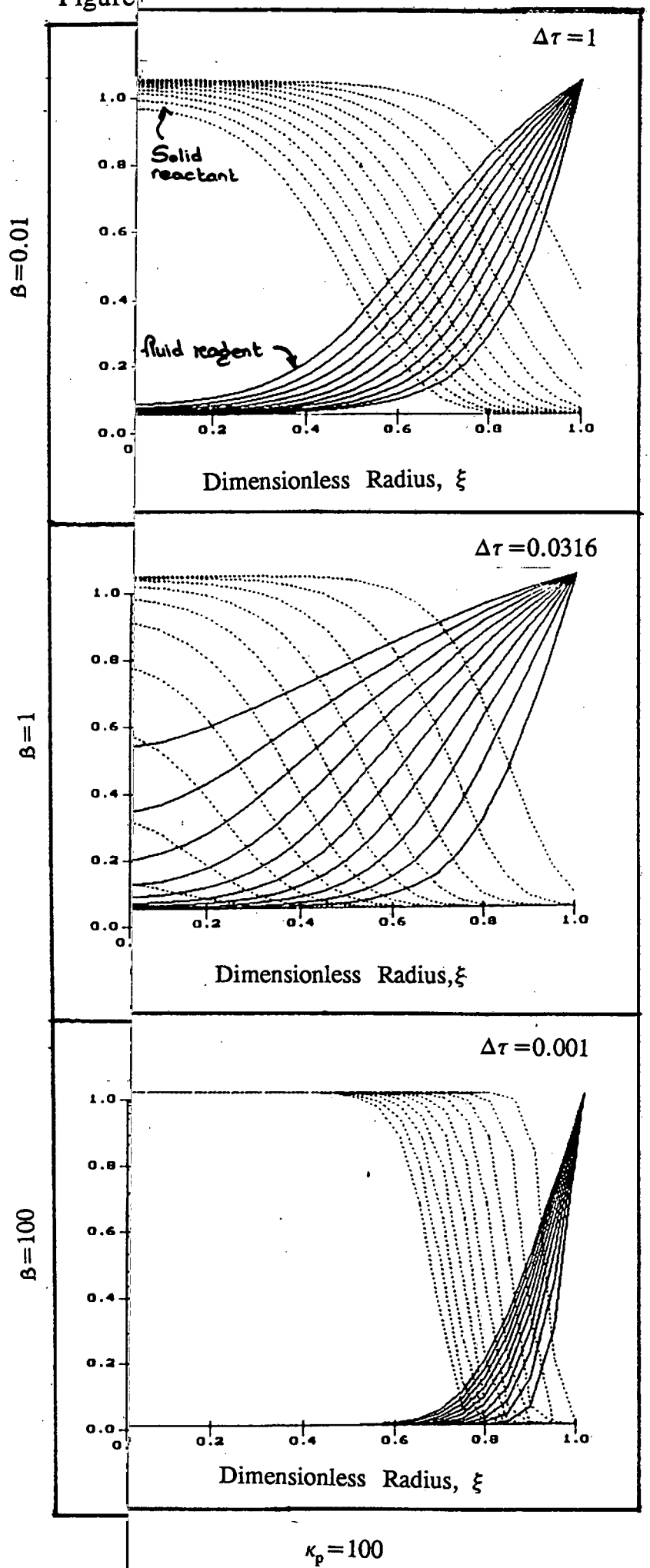

Δτ =1
Solid reactant
fluid reagent
1.0
0.8
0.6
0.4
0.2
0.0
0
0.2
0.4
0.6
0.8
1.0
Dimensionless Radius, ξ
β=0.01
Δτ =0.0316
Dimensionless Radius,ξ
β=1
Δτ =0.001
Dimensionless Radius, ξ
β=100
κp=100

Figure 4-3. The corresponding concentration profiles to Figure 4-2 presented by Dixon. (Figure 1-3 taken from Dixon's Thesis.)

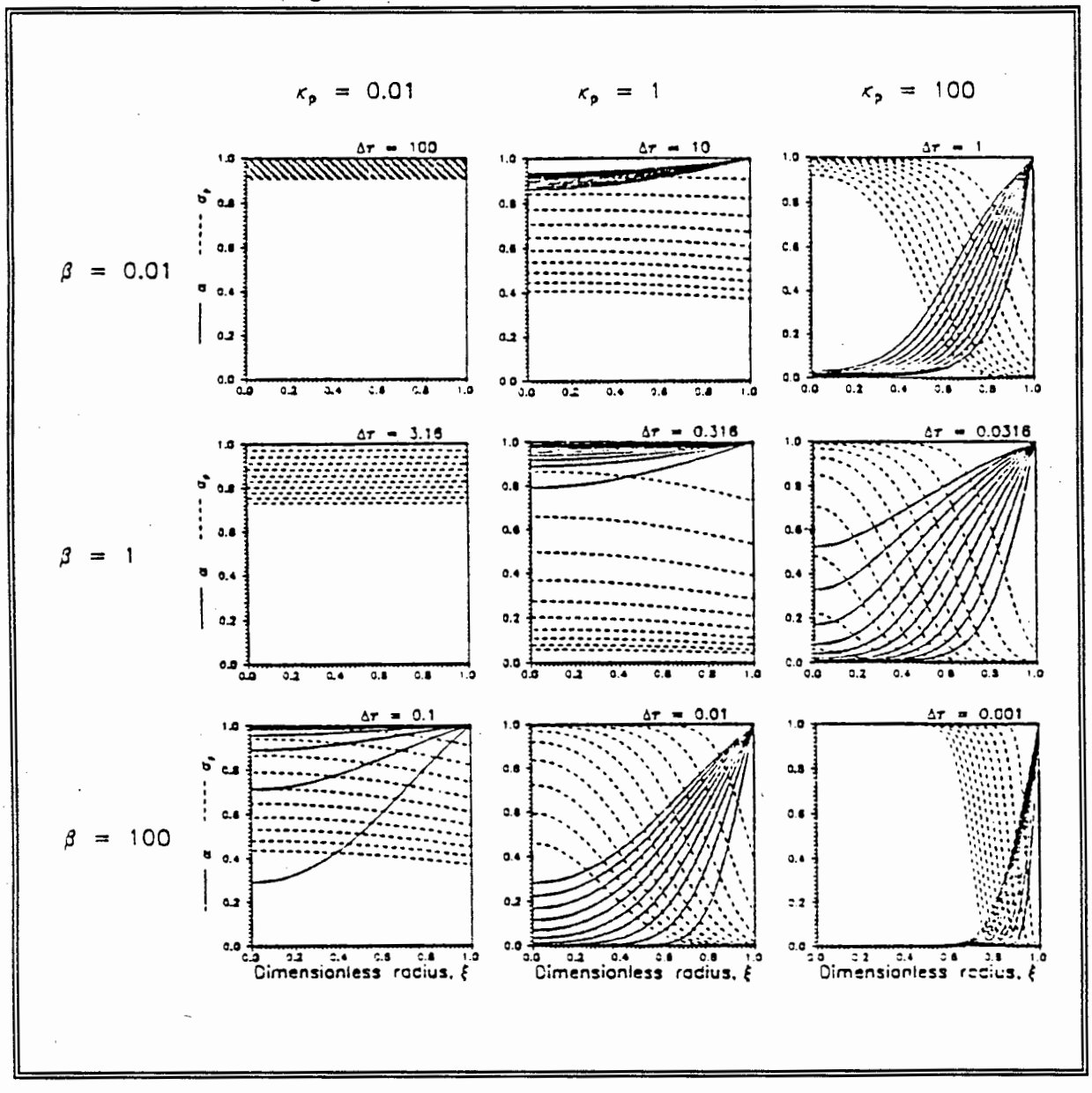


Figure 4-4. Fraction conversion profiles predicted by Model2D2 for the parameters as Indicated in the Figure.

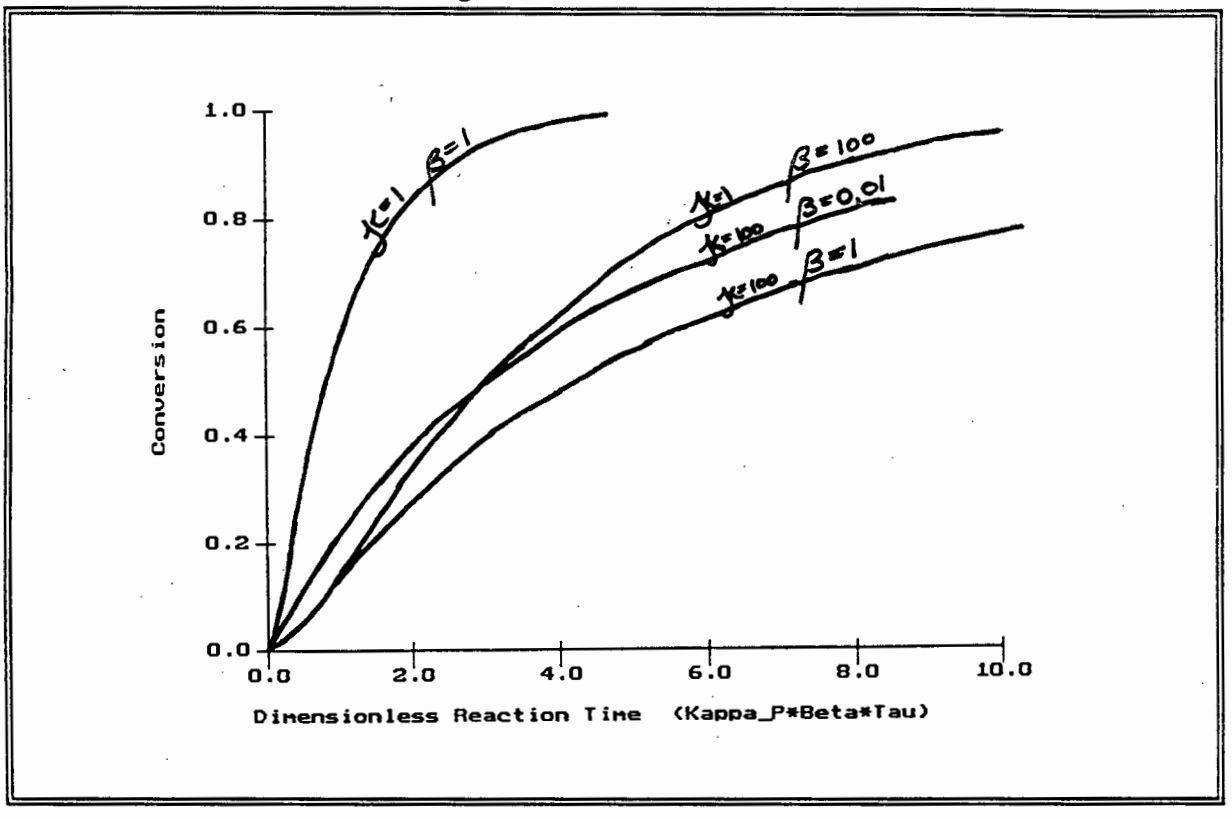


Figure 4-5. The corresponding fractional conversion profiles to Figure 4-4 presented by Dixon. (Figure 1-4 taken from Dixon's Thesis.)

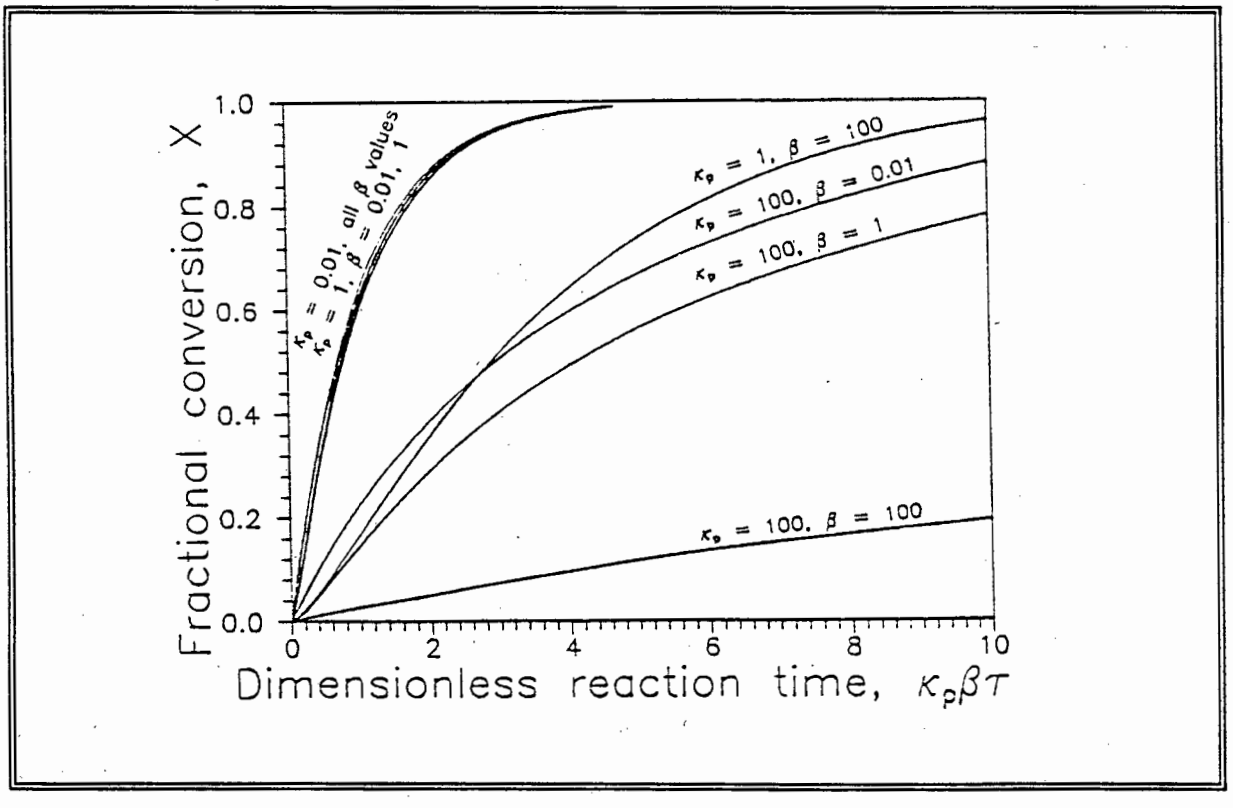


Figure 4-6. Fraction conversion profiles as a function of the variable order power predicted by Model2D2.

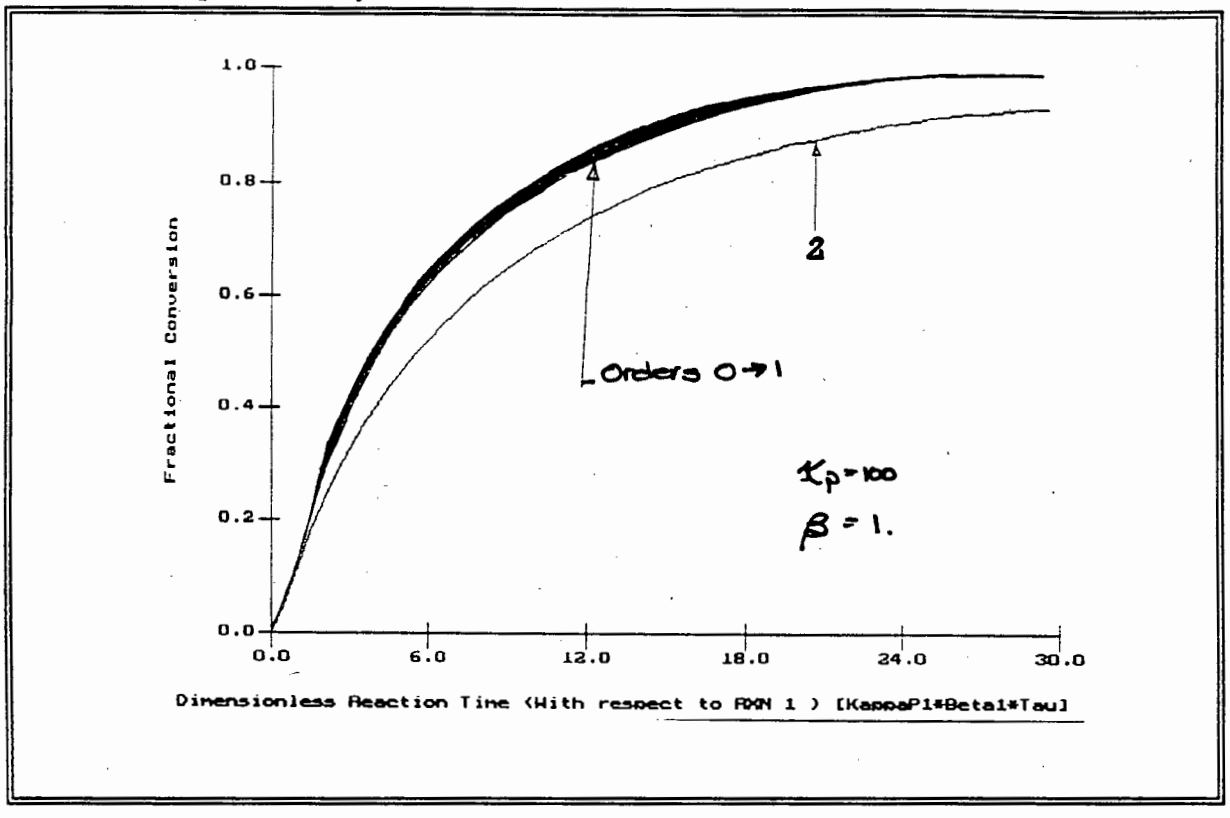
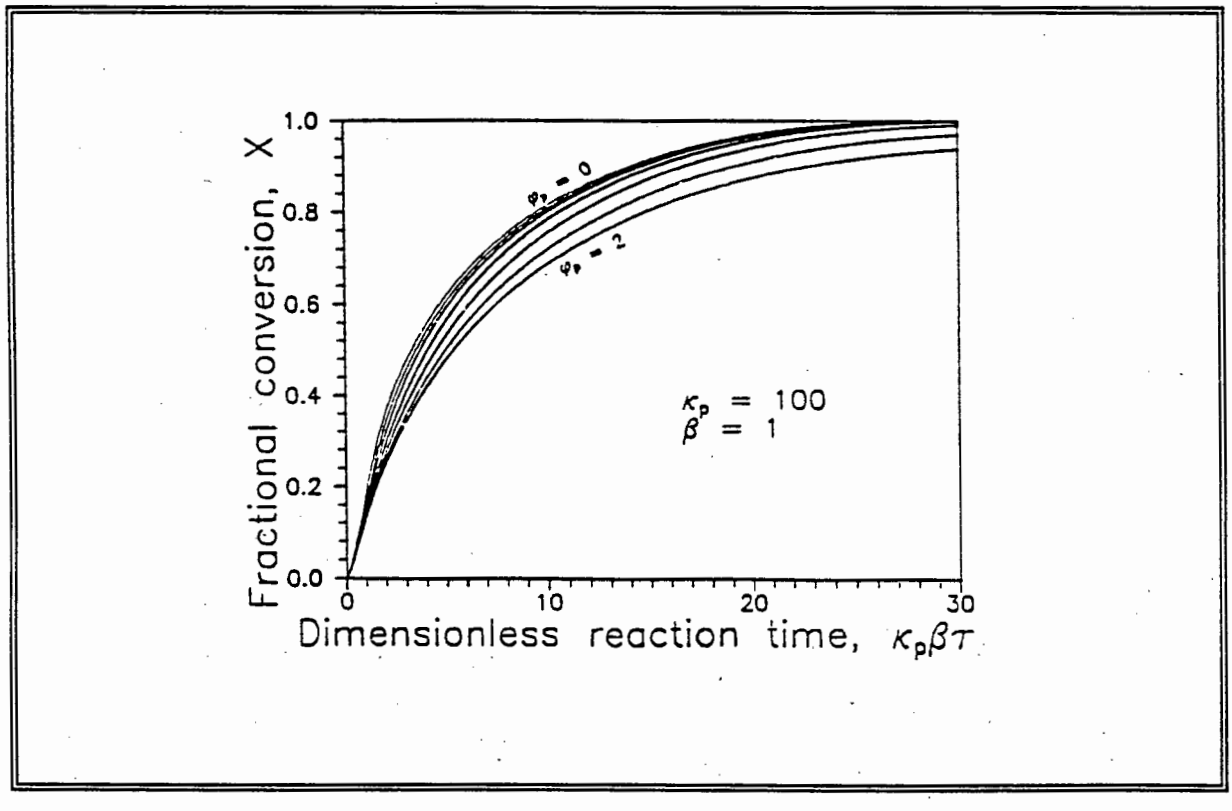


Figure 4-7. The corresponding fractional conversion profiles as a function of variable order power to Figure 4-6 presented by Dixon. (Figure 1-9 taken from Dixon's Thesis.)



Due to the excellent agreement between the results of Model2D2 and those presented by Dixon, the routines of Model2D2 can be used with confidence.

4.7 Application of the Model.

The chemical reaction model at the particle level is not that useful in itself. The model as it stands only describes the progression of the reactions in a *single particle*. It is very unlikely that the progression of the reactions within a single solitary particle will be required. Further, the boundary condition used, equation (4.8) or (4-24), implies that the fluid concentration in contact with the surface of the particle is constant. The only situation where this would arise is where a particle is submersed in an infinite amount of fluid and where mass transfer resistances are negligible.

More realistic situations are when many particles are associated with a finite volume of fluid. Such situations correspond to a CSTR experiment in which the leaching behaviour of a single size class or a size distribution of particles is investigated. Equally, a volume element of a waste deposit could be considered as an appropriate size distribution of particles associated with a finite volume of fluid. In this case the fluid would be the fluid in the void spaces *between* the particles.

In these situations, as the chemical reactions within each of the particles progress, fluid reagent will be consumed which will cause the bulk fluid concentration, that is the concentration of the fluid which surrounds the particles, to decrease. This dropping bulk fluid concentration is the appropriate boundary condition which should be used in particle scale model. In effect, this implies that a suitable boundary condition, which would replace equation (4-8) or (4-24), needs to be developed which will account for the decrease in the bulk fluid reagent as a function of time.

It is important to note that a partial differential equation, corresponding to equation (4-8) or (4-24), will be required for each particle size considered when the leaching behaviour of a size distribution of particles is investigated. The reason for this is that the rates of conversion of different sized particles will not be the same. Suitable dimensionless groups, which are defined in terms of a single reference particle size, need to be investigated. Lastly, appropriate solution strategies for the combined solution of the set of partial differential equations, which is comprised of a partial differential equation for each particle size, and the coupled boundary condition need to be determined. These aspects are considered in the following chapter.

Chapter 5. A Model to Describe Leachate Generation from Granular Wastes in a Continuously Stirred Tank Reactor Experiment.

The continuously stirred tank reactor (CSTR) model describes the increase in the hazardous constituent concentration and the decrease in the fluid reagent concentration in the bulk fluid of a CSTR type experiment. This is achieved by linking the particle scale, chemical reaction model, presented in the previous chapter, to an appropriate mass balance equation for the bulk fluid reagent concentration.

The appropriate mass balance equation is developed initially for equi-sized particles submerged in a finite volume of fluid reactant. In a CSTR experiment which contains a size distribution of particles, the release of hazardous constituents from each size class of particles needs to be determined before the overall release of hazardous constituents for the system can be calculated. This implies that a partial differential equation, corresponding to equation (4-22), will be required for each size class of particles. Before these differential equations can be used, the appropriate parameters for each size class of particles need to be determined. For this reason, a section which discusses model parameters as a function of particle size, has been included. This is followed by the development of a suitable mass balance equation for the consumption of fluid reagent in a CSTR experiment which involves a size distribution of particles.

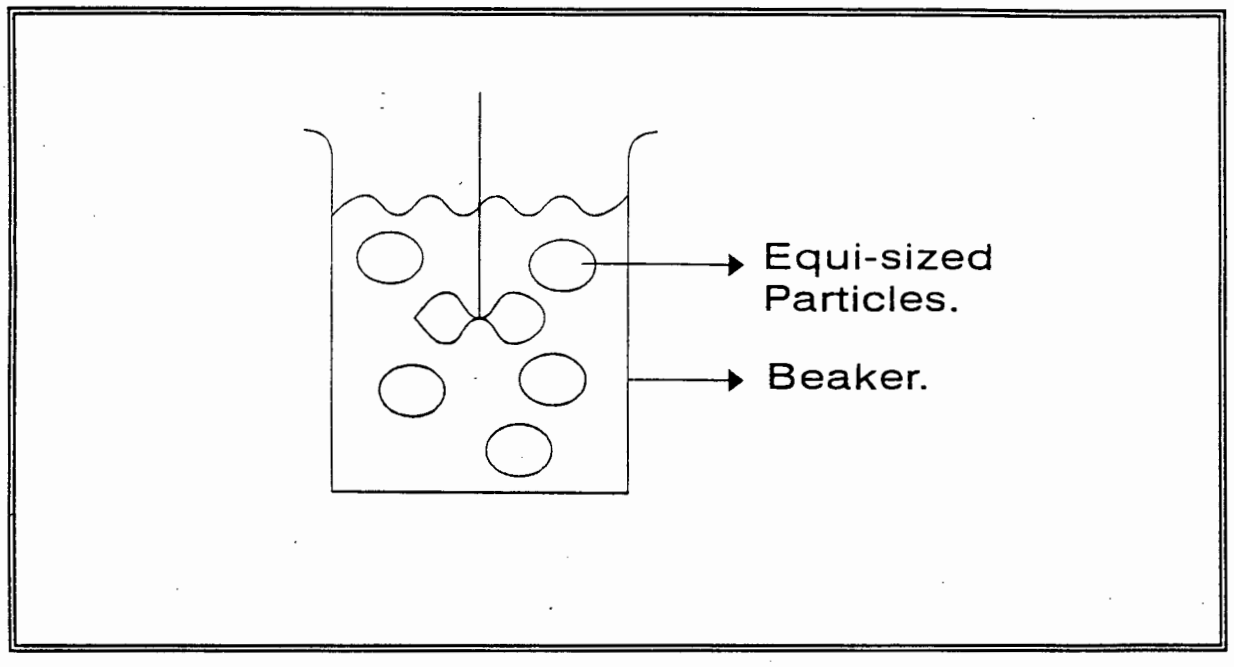
An appropriate solution strategy for the model has been investigated and implemented. A case study has been used to illustrate the general behaviour of the model and the sensitivity of the model to particle size distributions as well as to hazardous constituent distribution within each solid particle. The following section discusses how the model can be fitted to typical CSTR data. Lastly the applications and limitations of the model are summarised.

5.1 Development of the Mass Balance Equation for the Bulk Fluid Reagent in a CSTR which Contains Equi-Sized Particles.

Figure 5-1 depicts a few equi-sized spherical particles in a beaker of fluid reagent. It is assumed that the fluid reactant diffuses into the particles and reacts with the hazardous constituents which then enter the fluid phase.

The fluid reagent mass balance replaces the boundary conditions used in the model of Dixon [1992]. Referring to equations (4.22) to (4.29), note that equation (4-24) is a boundary condition which sets the bulk fluid concentration in contact with the particle at a constant value. This boundary condition needs to be replaced due to the fact that the bulk fluid reactant concentration drops as the fluid reactant diffuses into the particle.

Figure 5-1. Schematic of a few equi-sized spherical particles submerged in a well stirred beaker of fluid.



A mass balance equation which relates the consumption of the fluid phase reactant to the bulk fluid reactant concentration is:

$$\frac{V_{Part.}\rho_0(1-\epsilon_0)}{b_i}\sum_{i=1}^{N}\frac{dC_{s_i}}{dt}-D_e\nabla C_A|_R\frac{3V_{Part.}}{R}=V_{Liq.}\frac{dC_{A_{Bulk}}}{dt} \qquad (5-1)$$

where $V_{Part.}$ is the total volume of the particles; and;
$V_{Liq.}$ is the total volume of fluid reactant.

Note that the first term in the above equation represents the consumption of fluid reagent due to chemical reactions which take place on the surface of the particles. The second term represents the fluid reagent diffusing into the particles. This quantity is calculated as the product of the diffusive flux into a single particle (the diffusive flux being defined by Fick's Law), the surface area of a single particle and the number of particles in the system. The number of particles present in the system is determined by dividing the total volume of the particles by the volume of a single particle. The fluid reagent which diffuses into a particle is continually being consumed by the chemical reactions taking place within the pore volume of the particle.

The same equation expressed in dimensionless terms is:

$$-\sum_{i=1}^{N} \kappa_{s_i} \sigma_{s_i}^{\phi_{si}} \alpha_{bulk} - 3 \frac{\partial \alpha}{\partial \xi} \Big|_{\xi=1} = \nu \frac{\partial \alpha_{bulk}}{\partial \tau} \qquad (5-2)$$

where

$$\nu = \frac{V_{Liq.}}{\epsilon_0 V_{Part.}} \qquad (5-3)$$

As previously discussed, the chemical reaction model derived in Chapter 4, applies to a single particle. Equation (5-2) is a suitable boundary condition for equi-sized particles submerged in a finite volume of fluid reagent. This boundary condition, in conjunction with the chemical reaction model for the single particle, can be used to simulate both the concentration profiles of the solid reactants and fluid reagent within the particles as well as the bulk fluid phase concentration as a function of time.

5.2 Model Parameters as a Function of Particle Size.

As previously discussed, in order to simulate the progression of reactions in a size distribution of particles, the chemical reaction model needs to be applied to each individual size class. The overall progression of the reaction for the system is then obtained by integrating the results of the individual size classes over the distribution of particles in the system.

Before the chemical reaction model can be solved for each size class of particles, the model parameters applicable to each size class need to be determined. Two different approaches to defining the model parameters as a function of particle size can be used. In the first approach, model parameters are fitted to only one size class of particles and the parameters of all the other size classes are related to it. The assumptions behind this approach and the resulting relationships for the model parameters as a function of particle size are discussed in the next section. In some cases, particles in different size classes exhibit sufficiently different properties to preclude any simple relationships between them. In these cases the model parameters need to be determined for each size class individually.

5.2.1 Determination of the Model Parameters Applicable to Precious Metal Leaching with Respect to a Reference Size Class of Particles.

Dixon [1992] defined a set of relationships for the model parameters in terms of a reference particle size. The parameters of the reference particle size, which are denoted as barred quantities and which need to be specified, can be summarised as:

$$\overline{\beta}_i \qquad \overline{\lambda}_i \qquad \overline{\kappa}_{pi} \qquad \overline{\kappa}_{si}$$

Dixon's work [1992] was specific towards the extraction of precious metals from ores and so made use of information with regard to ore preparation to determine appropriate relationships for the model parameters. In summary, Dixon made the following assumptions with respect to ore preparation:

- Only the surface fraction, λ_i, is affected by crushing, and not the total extractable grade, C_{Ei0}, or any other parameter; and;
- λ_i is proportional to the ratio of the particle area to the particle volume.

Using these assumptions, the following relationship was obtained:

$$\lambda_{i_k} = \frac{\overline{\lambda}_i}{\theta_k} \qquad (5-4)$$

where

$$\theta_k = \frac{R_k}{\overline{R}} \qquad (5-5)$$

where R_k radius of particle in size class k ($k \epsilon \{1..M\}$); and;
$\overline{R}$ radius of the reference particle.

θ_k is a dimensionless total particle radius. It is important to note that the dimensionless total particle radius can be greater than one. This occurs when the radius of the reference particle is smaller than the radius of the particles in size class k.

Dixon proceeds to determine relationships for κ_{pi}, κ_{si} and τ in terms of the corresponding reference class parameters combined with λ_i and θ_k.

5.2.2 Determination of the Model Parameters Applicable to Leaching of Hazardous Constituents from Waste Particles with Respect to a Reference Size Class of Particles.

The assumptions which Dixon made with respect to ore preparation to determine the model parameters as a function of particle size do not hold for waste particles. The reason for this is that Dixon's assumptions are based on his previous assumption of the solid reactants being present in the form of discrete inclusions within the porous particle. Further, the only reason that the surface concentration of the solid reactants will increase with decreasing particle size is due to the fact that more inclusions stand a chance of

falling on the external surface area of smaller particles.

In contrast, the hazardous constituents in waste particles are usually not restricted to discrete inclusions. Also, the concentration of the hazardous constituents on the surface of particles in known to be enhanced in smaller particles in some cases [von Blottnitz 1994; Van Craen *et al.*, 1983]. (These aspects have been discussed previously in section 2.1.1.) For these reasons, appropriate relationships for the model parameters with respect to a reference particle size which would be applicable to waste particles need to be determined.

It has been assumed that the initial hazardous constituent concentration within the particles, C_{pi0}, is the same for all particles. In contrast, the initial surface hazardous constituent concentration, C_{si0}, is known to be a function of particle size [Van Craen *et al.* 1983]. In summary, it has been assumed that all properties of the particles, except the surface concentration and thus the total extractable concentration, remain fairly constant over the range of particle sizes. In a manner similar to Dixon, the reference size class's parameters have been denoted as barred quantities. The reference size class parameters which need to be defined are identical to those of Dixon.

It is not yet possible to predict the surface hazardous constituent concentration as a function of particle size from purely theoretical arguments. Instead this information needs to be determined from hazardous constituent location analyses for the particle sizes of interest or estimated from existing hazardous constituent location data. This information must be specified in a parameter which is defined as the ratio of the hazardous constituent concentration on the surface of the particles to the hazardous constituent concentration within the particles:

$$\zeta_{i_k} = \frac{C_{si,0_k}}{C_{pi,0_k}} \qquad (5\text{-}6)$$

where $C_{si,0,k}$ hazardous constituent concentration of species i on the surface of the particle in size class k; and;

$C_{pi,0,k}$ hazardous constituent concentration of species i in the pores of the particle in size class k.

Once the $\zeta_i(\Theta_k)$ values have been specified, sufficient information is known about the system to formulate suitable functional relationships for the model parameters. This is demonstrated for the λ_{ik} parameter, which represents the fraction of the solid reactant on the surface of the particles:

$$\lambda_{i_k} = \frac{C_{si,0_k}}{C_{si,0_k} + C_{pi,0_k}}$$

$$= \frac{C_{pi,0_k}\ \varsigma_{i_k}}{C_{pi,0_k}\ \varsigma_{i_k} + C_{pi,0_k}}$$

$$= \frac{\varsigma_{i_k}}{1+\varsigma_{i_k}} \qquad (5\text{-}7)$$

In a similar manner, relationships for the other model parameters can be determined. These relationships can be summarised as:

$$\lambda_{i_k} = \frac{\varsigma_{i_k}}{1+\varsigma_{i_k}} \qquad (5\text{-}8) \qquad\qquad \beta_{i_k} = \overline{\beta}_i \ \frac{1+\overline{\varsigma}_i}{1+\varsigma_{i_k}} \qquad (5\text{-}9)$$

$$\kappa_{pi_k} = \overline{\kappa}_{pi}\ \Theta_k^2 \qquad (5\text{-}10) \qquad\qquad \kappa_{si_k} = \overline{\kappa}_{si} \left(\frac{\varsigma_{i_k}}{\overline{\varsigma}_i}\right)^{\phi_{si}} \Theta_k \qquad (5\text{-}11)$$

$$\tau_k = \frac{\overline{\tau}}{\Theta_k^2} \qquad (5\text{-}12)$$

Dixon [1992] has shown that if it can be assumed that the chemical species within the particle would react to the same extent as the species on the surface of the particle, if both were exposed to the same fluid reactant concentration for the same time, then the surface parameters take on the form:

$$\kappa_{si} = \frac{\lambda_i}{1-\lambda_i}\kappa_{pi} \qquad (5\text{-}13) \qquad\qquad \phi_{si} = \phi_{pi} \qquad (5\text{-}14)$$

This information can be used to eliminate the need to define κ_{si}. Thus equation (5-9) could be written as:

$$\kappa_{si_k} = \overline{\kappa}_{pi} \frac{\overline{\lambda}_i}{1-\overline{\lambda}_i} \left(\frac{\varsigma_{i_k}}{\overline{\varsigma}_i}\right)^{\phi_{pi}} \Theta_k \qquad (5\text{-}15)$$

5.3 Extension of the Bulk Fluid Mass Balance Equation to Incorporate Fluid Reactant Consumption from a Size Distribution of Particles.

Equation (5-1) is a bulk fluid reactant mass balance equation which accounts for the fluid reactant being consumed by equi-sized particles. This equation can be extended to apply to a size distribution of particles by summing the fluid reactant consumed by the different size classes. This can be summarised as:

$$\sum_{k=1}^{M} \left[\frac{V_{Part,k}\rho_0(1-\epsilon_0)}{b_i} \sum_{i=1}^{N} \frac{dC_{s_i}}{dt} - D_e \nabla C_A|_{R_k} \frac{3V_{Part,k}}{R_k} \right] = V_{Liq.} \frac{dC_A}{dt} \qquad (5\text{-}16)$$

where $V_{Part,k}$ is the total volume of particles in size class k; and;
M number of sizes classes.

This equation re-expressed in dimensionless format is:

$$\sum_{k=1}^{M} \left[-\sum_{i=1}^{N} \frac{\Upsilon_{Part,k}}{\Theta_k^2} \kappa_{si,k} \sigma_{si,k}^{\phi_{si}} \alpha_{bulk} - \frac{3\Upsilon_{Part,k}}{\Theta_k^2} \left(\frac{\partial \alpha}{\partial \xi}\right)_k |_{\xi=1} \right] = \frac{V_{Liq}}{\epsilon_0 \overline{V}_{Part.}} \frac{d\alpha_{bulk}}{d\overline{\tau}} \qquad (5\text{-}17)$$

where

$$\Upsilon_{Part,k} = \frac{V_{Part,k}}{\overline{V}_{Part.}} \qquad (5\text{-}18)$$

where $V_{Part,k}$ total volume of particles in size class k; and;
V_{Part} total volume of particles in the reference size class.

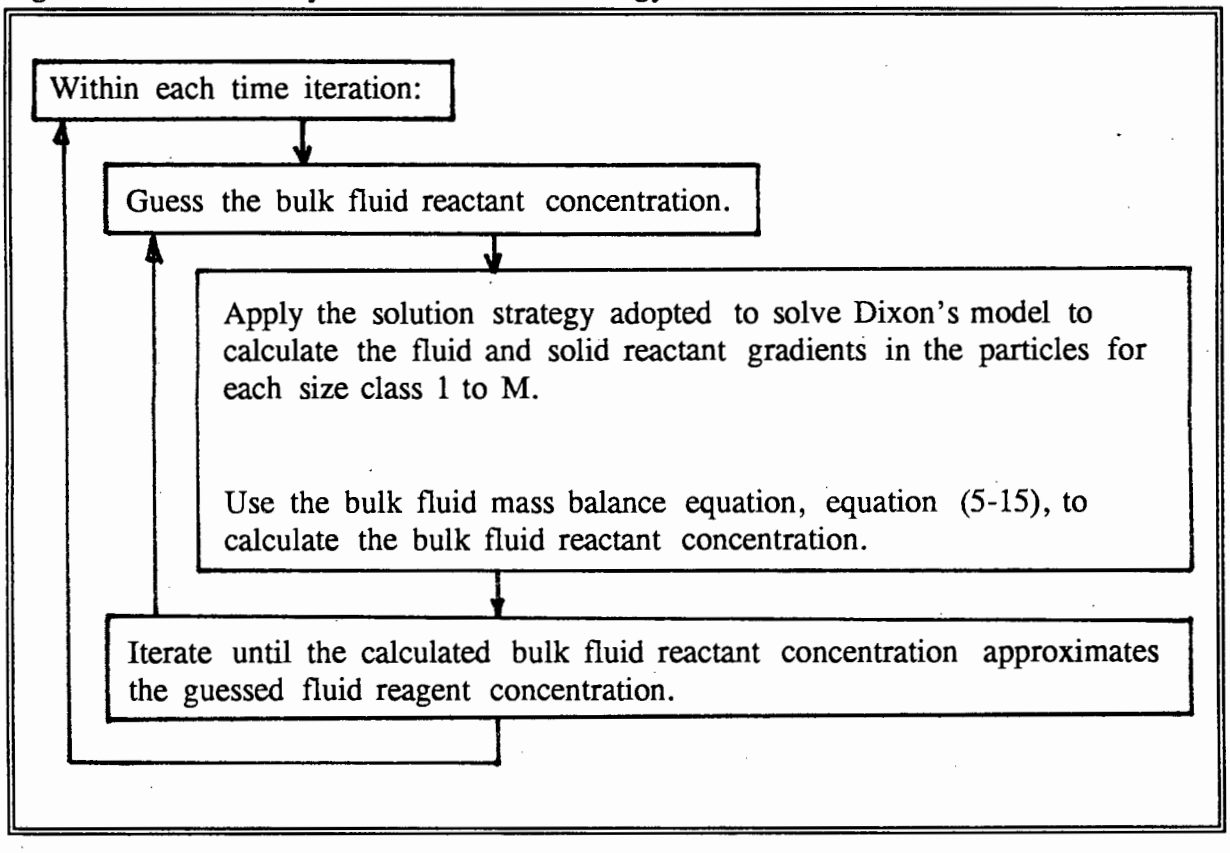
Equation (5-17) is a suitable boundary condition which applies to a size distribution of particles. This boundary condition, used in conjunction with a suitable partial differential equation, equation (4-22), for each size class of particles can be used to simulate the concentration profiles within all of the particles in the system and the bulk fluid reagent concentration as a function of time.

5.4 Solution Strategy:

The system of equations which define this model are identical to equations (4-22) through to (4-29) except that the one boundary condition, equation (4-24), is replaced by equation (5-17). This equation was incorporated into the solution strategy using an implicit finite difference technique.

As it has already been pointed out, in order to simulate chemical release from a size distribution of particles, the chemical reaction model needs to be applied to each and every size class of particles. The solution strategy adopted for the CSTR model makes use of the solution strategy used to solve Dixon's model and is summarised in Figure 5-2.

Figure 5-2. Summary of the Solution Strategy used in the CSTR model.



5.5 Suitable Computer Routines for the CSTR Model.

Programs Model5E1.PAS and Model5E2.PAS are suitable computer codes for the CSTR model. Model5E1.PAS is a code which assumes a reaction order of unity with respect to the solid reactant while Model5E2.PAS can accommodate a variable reaction order.

Copies of the code as well as solution algorithms can be found in Appendix IV.

5.6 Verification of the Computer Routines.

As before, the computer routines were rigorously checked to ensure that they were operating correctly.

The first test applied was to use Model5E1 to predict the concentration profiles within a single size class of particles with a large excess of fluid reagent. Because of the large excess of fluid reagent, its concentration would not be expected to drop significantly during the CSTR experiment. Such a constant bulk fluid concentration boundary condition corresponds to the boundary condition used in the development of the chemical reaction model described in the previous chapter. Thus, Model2D2, using the same parameters as used in Model5E1, should predict the same profiles as Model5E1 for an experiment of this nature. This is demonstrated in Figure 5-3 and Figure 5-4.

Several other self-checking strategies were employed to ensure that the routines were operating correctly. The most important strategies used included the following:

- The model was checked to ensure that it would predict the same concentration profiles for a given size class of particles irrespective of the order in which the size class of particles was entered into the program.

- The program was checked further to ensure that if the average particle size in two 'different' size classes were identical that the code would predict identical concentration profiles for both size classes.

- The fluid mass balance equation was checked to ensure that it was operating correctly by defining several 'different' size classes of particles all to contain the same sized particles. As long as the sum of the volume fractions of these size classes remain constant, the overall fluid reagent consumption should remain constant. The computer routines predicted this expected behaviour.

  The fluid mass balance equation was further checked to ensure that as bulk volume of the fluid was decreased in a series of CSTR experiments that the resulting bulk fluid phase concentration of the reagent would decrease.

Figure 5-3. Concentration profiles predicted by Model5E1 for a single size class of particles with a large excess of bulk fluid reagent.

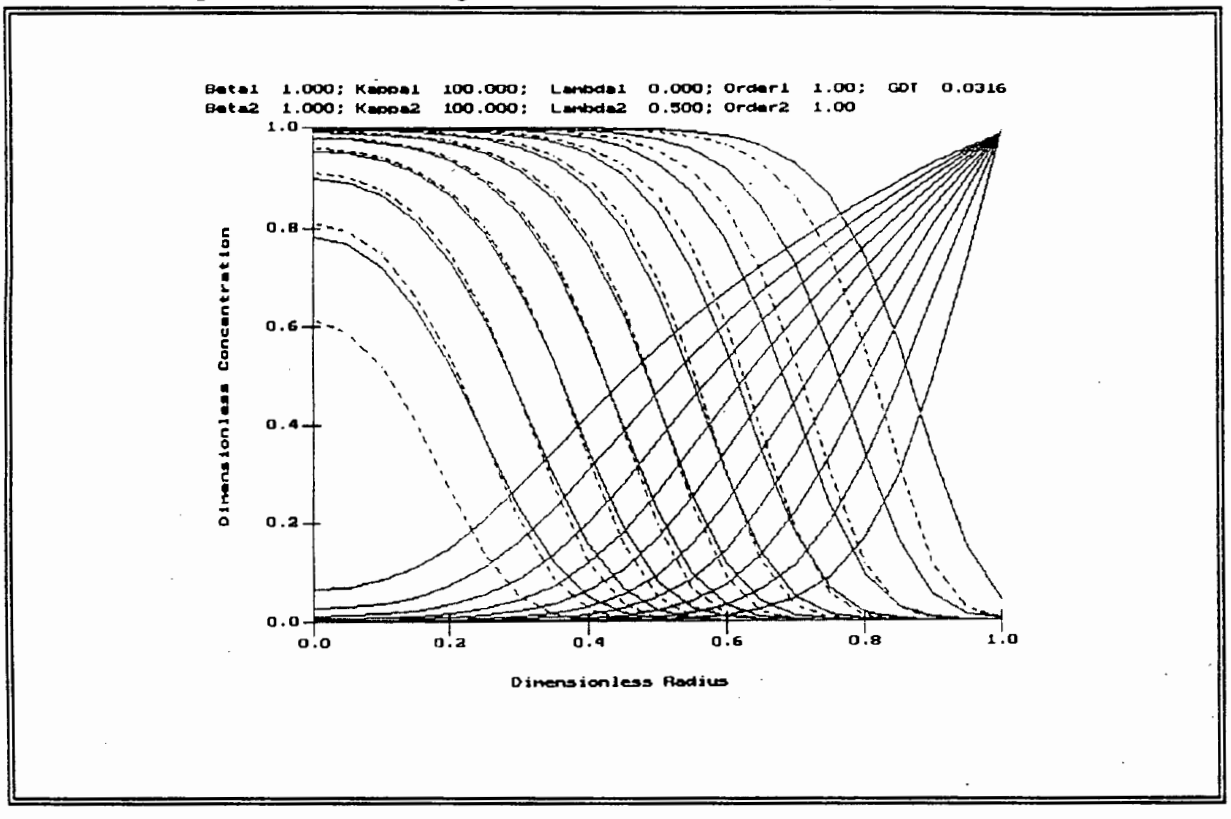
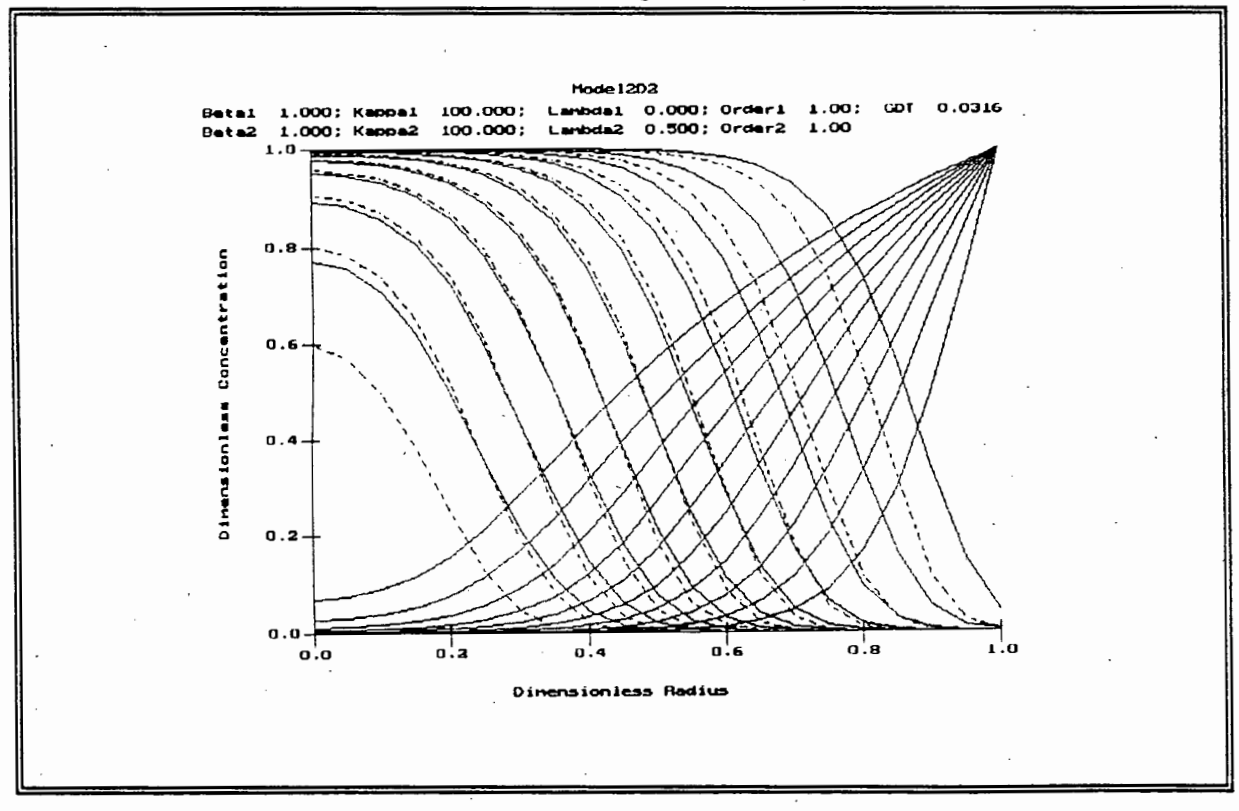


Figure 5-4. Concentration profiles predicted by Model2D2 using the same parameters used in the simulation used to generate Figure 5-3.



5.7 Results and Discussion.

5.7.1 General Behaviour.

In order to determine the general behaviour of the CSTR model, the following scenario, representative of a typical bench scale CSTR experiment, was investigated:

- Volume of fluid reactant $= 1l$;
- Total volume of solid particles $= 0.1l$;
- Size distribution of the particles used is summarised in Table 5-1 and shown in Figure 5-5;
- Only one solid reactant present;
- First order rate dependence in the solid reactant concentration;
- $\zeta_{1,k}$ used is summarised in Table 5-1 and shown in Figure 5-6;

Table 5-1. Size Distribution and $\zeta_{1,k}$ used in the Analysis.

Size Class Number.	Average Particle Size in Size Class. (mm)	% Occurrence.	$\zeta_{1,k}$
1	9.5	0.252	1.2
2	8.5	0.428	1.2
3	7.5	1.045	1.3
4	6.5	2.564	1.3
5	5.5	6.191	1.4
6	4.5	14.061	1.4
7	3.5	27.105	1.6
8	2.5	34.247	1.8
9	1.5	13.846	2.3
10	0.5	0.260	5.0

Note that the size distribution of particles included in Table 5-1 is representative of a log-normal size distribution. Also, the $\zeta_{1,k}$ values were obtained by assuming them to be inversely proportional to the radius of the particle. The equation used to generate the $\zeta_{1,k}$ values used in Table 5-1 can be summarised as:

$$\zeta_{1_k} = \frac{2}{R_k} + 1 \qquad (5\text{-}19)$$

where R_k is the average radius of the particles in size class k.

This corresponds to the situation in which infinitely large particles exhibit identical surface and bulk 'grades' of hazardous constituents. Also, the ratio of the surface to bulk grade for the particles in size class 10, which have an average radius of 0.5mm, has arbitrarily, for demonstration purposes, been chosen to be 5. In retrospect, a better relationship would have been:

$$\varsigma_{1_k} = \frac{2}{R_k} \qquad (5\text{-}20)$$

The reason why this relationship is superior to equation (5-19) is because it corresponds to the case where infinitely large particles exhibit a negligible surface grade of hazardous constituents compared to the bulk grade of hazardous constituents. This situation is far closer to what would be expected in reality. The reason for this is because the external surface area of a particle increases in proportion to the square of the radius while the volume of a particle increases in proportion to the cube of the radius. Thus infinitely large particles will have a negligible surface area compared to the volume of the particle. Hence the surface 'grade' of the hazardous constituents will be negligible. The ratio of the surface grade of hazardous constituents to the bulk grade of hazardous constituents for the reference size class of particles in equation (5-20) has once again arbitrarily been chosen for demonstration purposes to be 5.

Figure 5-5. Size Distribution of Particles used in Analysis

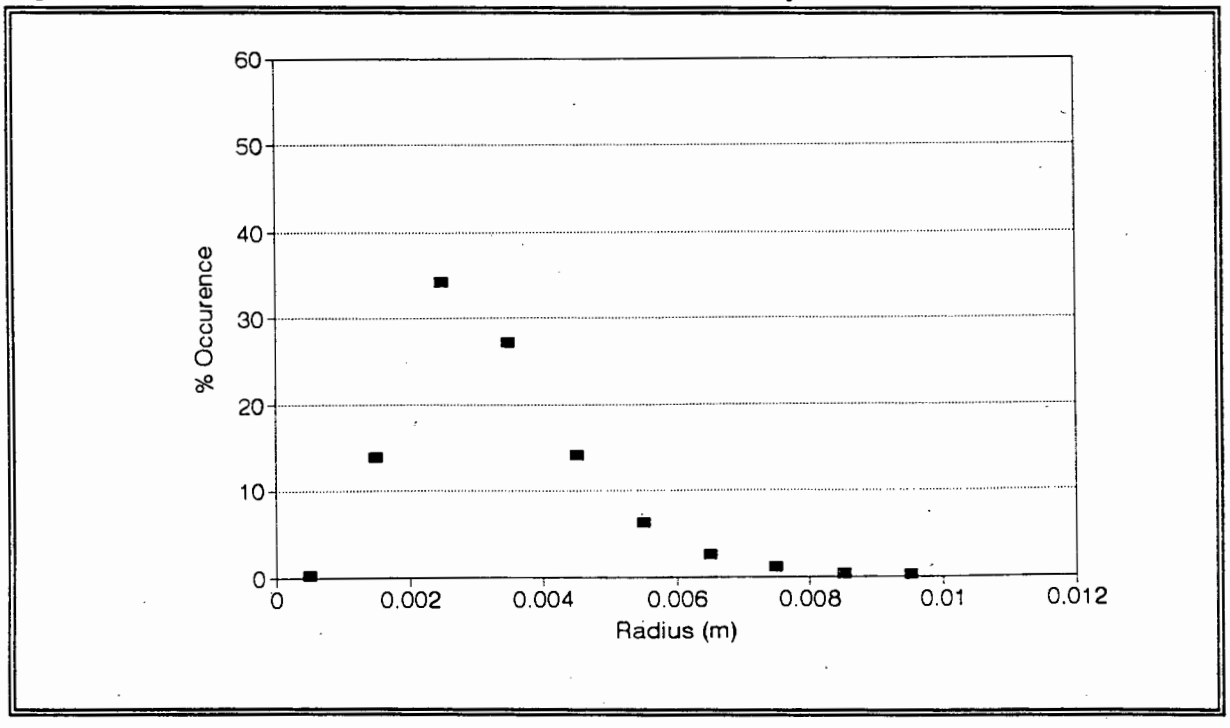
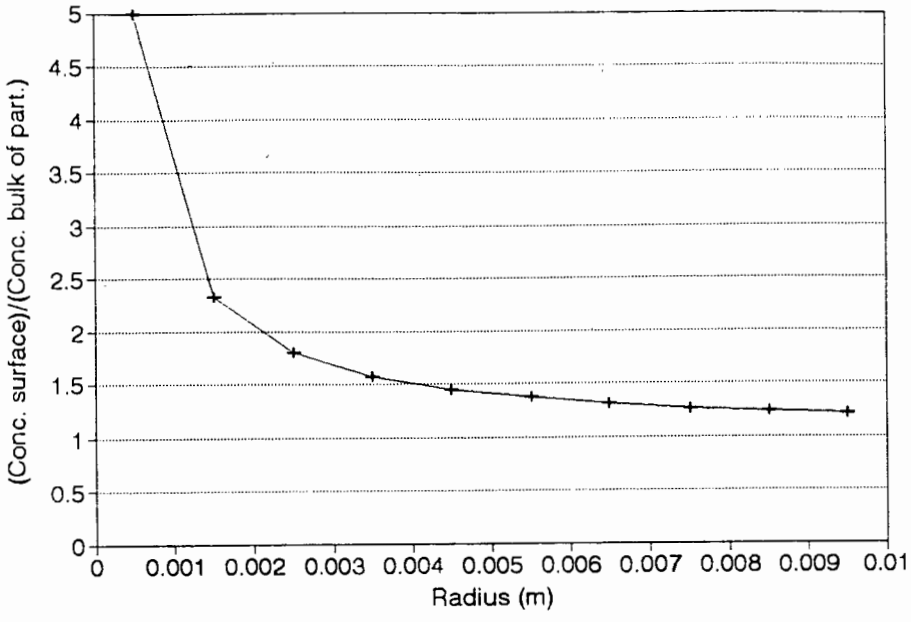


Figure 5-6. Hazardous Constituent Location Data used in the Analysis.



From Table 5-1 size class 8 can be seen to be dominant. For this reason it was used as the reference size class in the simulation. The parameters used in the simulation for this size class are summarised in Table 5-2. The κ_{p1} parameter corresponds to the chemical reaction rate being 10 times faster than the rate of fluid reagent diffusing into the reference particle.

Table 5-2. Reference size class parameters. (Reference Size Class = Size Class 8.)

β_1	1.0
$\kappa_{P,1}$	10.0
ΔT	0.001

Figure 5-7 presents the profiles of fluid reagent A (solid curves) and one solid reactant (dashed curves) for each size class of particle used in the simulation. Figure 5-8 shows the overall conversion of the system.

As expected, the smallest particles tend to react in a homogenous manner. These particles are sufficiently small for diffusion not to be rate limiting in any way. Instead, the release of hazardous constituents from these particles is dictated by kinetic considerations. As the particles get larger, they are seen to react in a more zone-wise manner.

Figure 5-7. Fluid Reagent and Solid Reactant Profiles as a Function of Particle Size.

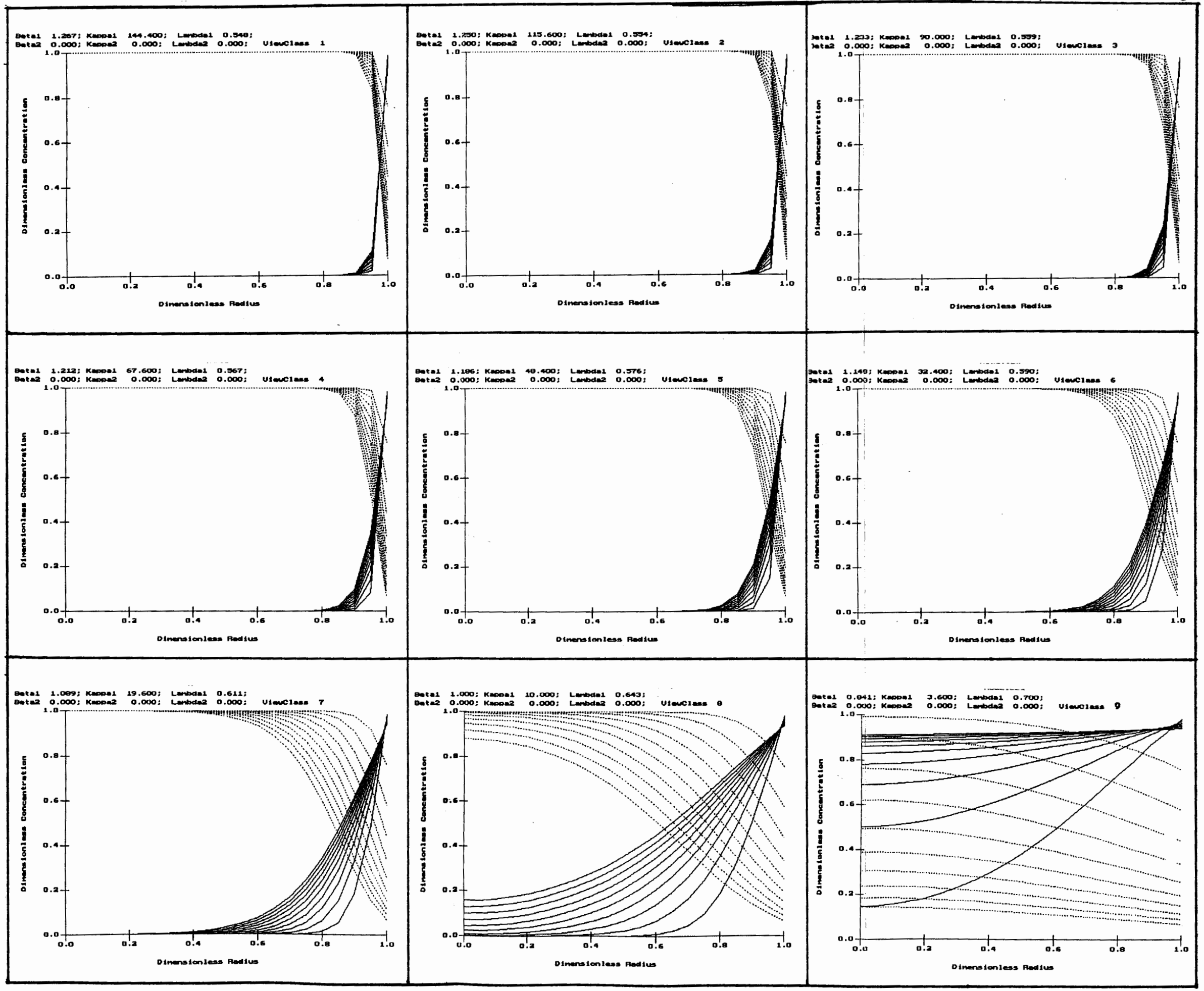


Figure 5-7 Con't. Fluid Reagent and Solid Reactant Profiles for the Smallest Size Class in the Simulation.

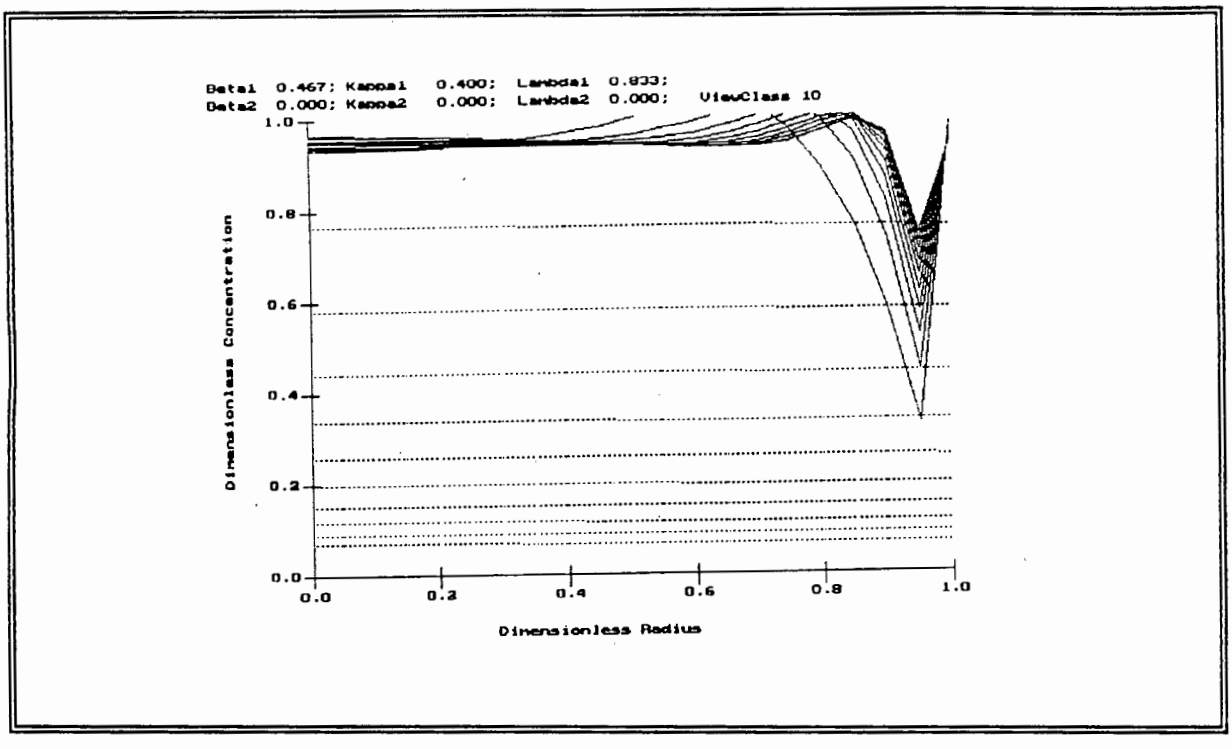
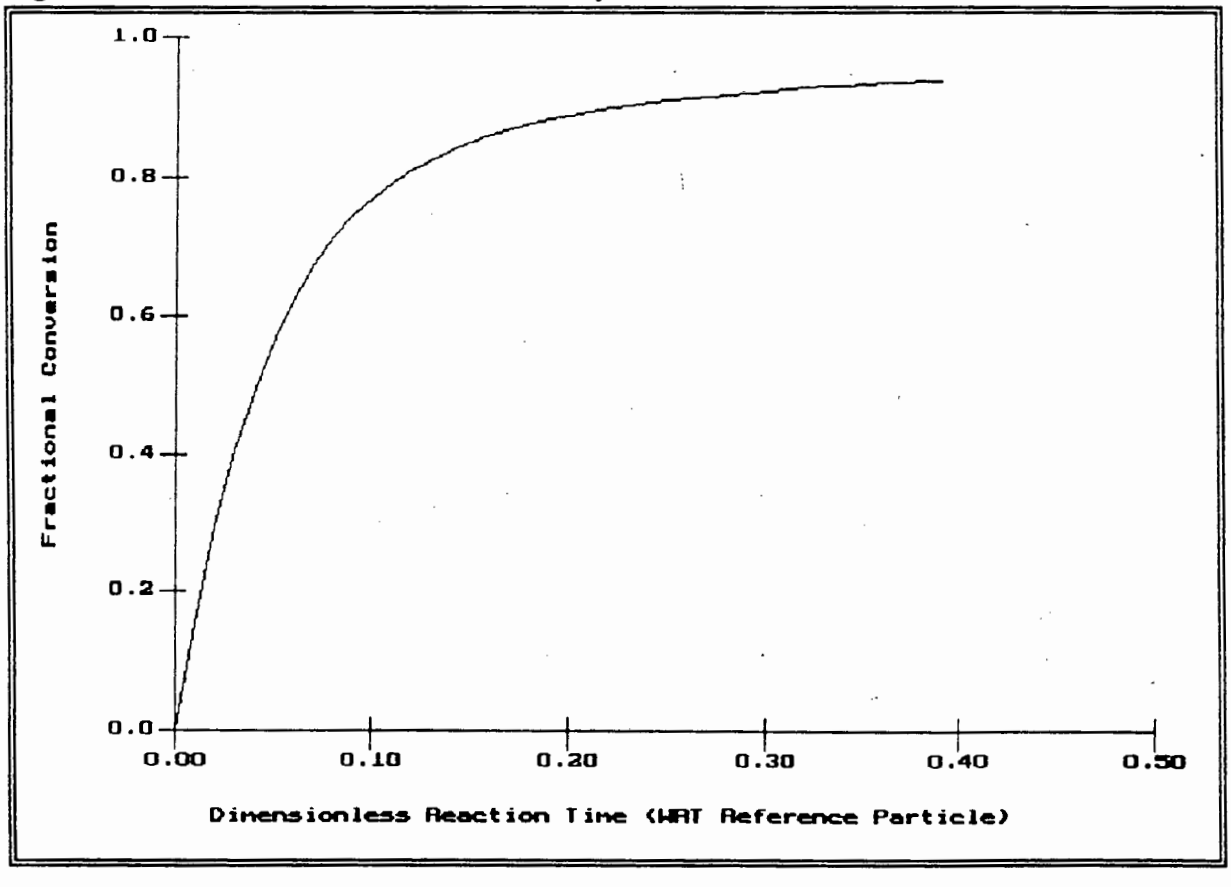


Figure 5-8. Overall conversion for the System.



Note that there is some numerical inaccuracy in the concentration profiles of the smallest size class shown in Figure 5-7. Although this inaccuracy can be eliminated by reducing the time step between iterations there is a trade off between the time for computation and numerical accuracy. It is also important to note that the smallest particles will display the largest numerical inaccuracy. This can be seen from the fact that the smallest size classes have the largest relative time steps in the simulation (equation 5-12). Further, although the profiles in the smallest size class are inaccurate, they are unconditionally stable. This unconditional stability is a property of solving second order parabolic partial differential equations using the Crank-Nicolson method [Crank 1975]. Since the smallest size class in Table 5-1 contains such a low occurrence of particles, the numerical inaccuracy in this size class was deemed to be acceptable.

Figure 5-8 exhibits some interesting characteristics. The graph consists of two distinct sections: a straight line section accounting for the fractional conversion at the beginning of the experiment, and a concave section later on in the experiment. The straight line section is characteristic of a kinetic controlled situation with an excess of fluid reactant. Effectively it represents the release of solid reactant from the small particles which react in a kinetic controlled manner. In contrast, the concave section is characteristic of a diffusion controlled reaction. Roman [1974] also observed these trends in conversion or recovery calculations. (Although Roman's calculations were for a column, a CSTR experiment can be considered as a very short column in which no mass transfer limitations are present.)

5.7.2 Effect of Particle Size Distribution on the Fractional Conversion.

The following scenarios, summarised in Table 5-3 and Figure 5-9, were used to investigate the effect of size distribution on the release of hazardous constituents. Note that all $\zeta_{1,k}$ values were set to unity to eliminate their effect on the results. The parameters for the reference particle size were as previously defined in Table 5-2.

Table 5-3. Summary of the conditions used to investigate the effect of Size Distribution on Fractional Conversion.

Particle Size. (mm)	Base Case Size Distribution	Size Distrib. with Predominantly Small Particles.	Size Distrib. with Predominantly Large Particles.	$\zeta_i(\Theta_k)$
9.5	0.252	1.0	1.0	1.0
8.5	0.428	1.0	9.0	1.0
7.8	1.045	1.0	55.0	1.0
6.5	2.564	2.0	9.0	1.0
5.8	6.191	2.0	5.0	1.0
4.5	14.061	5.0	5.0	1.0
3.5	27.105	16.0	5.0	1.0
2.5	34.247	55.0	5.0	1.0
1.5	13.846	16.0	5.0	1.0
0.5	0.260	1.0	1.0	1.0

Figure 5-9. Summary of the Size Distributions used in the Simulations.

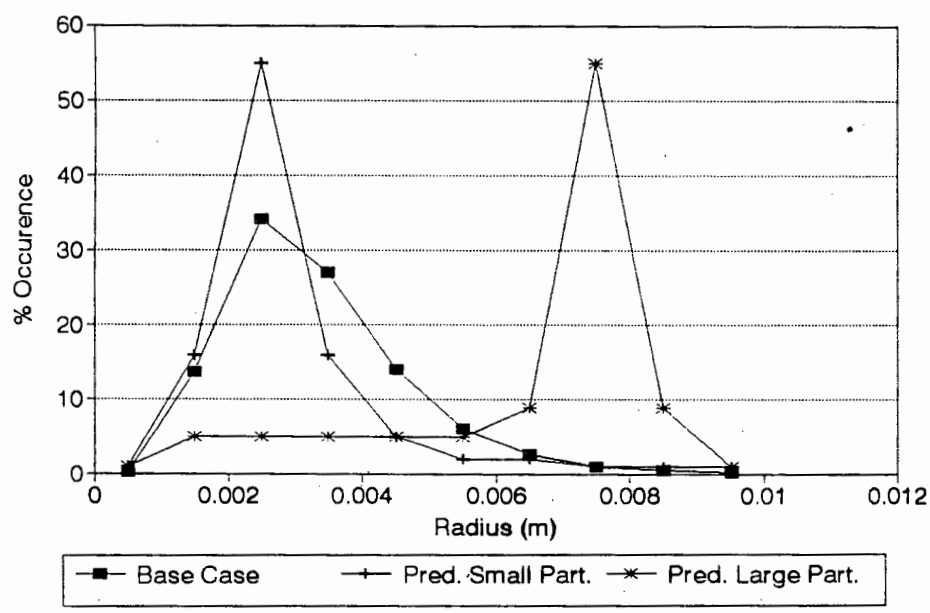
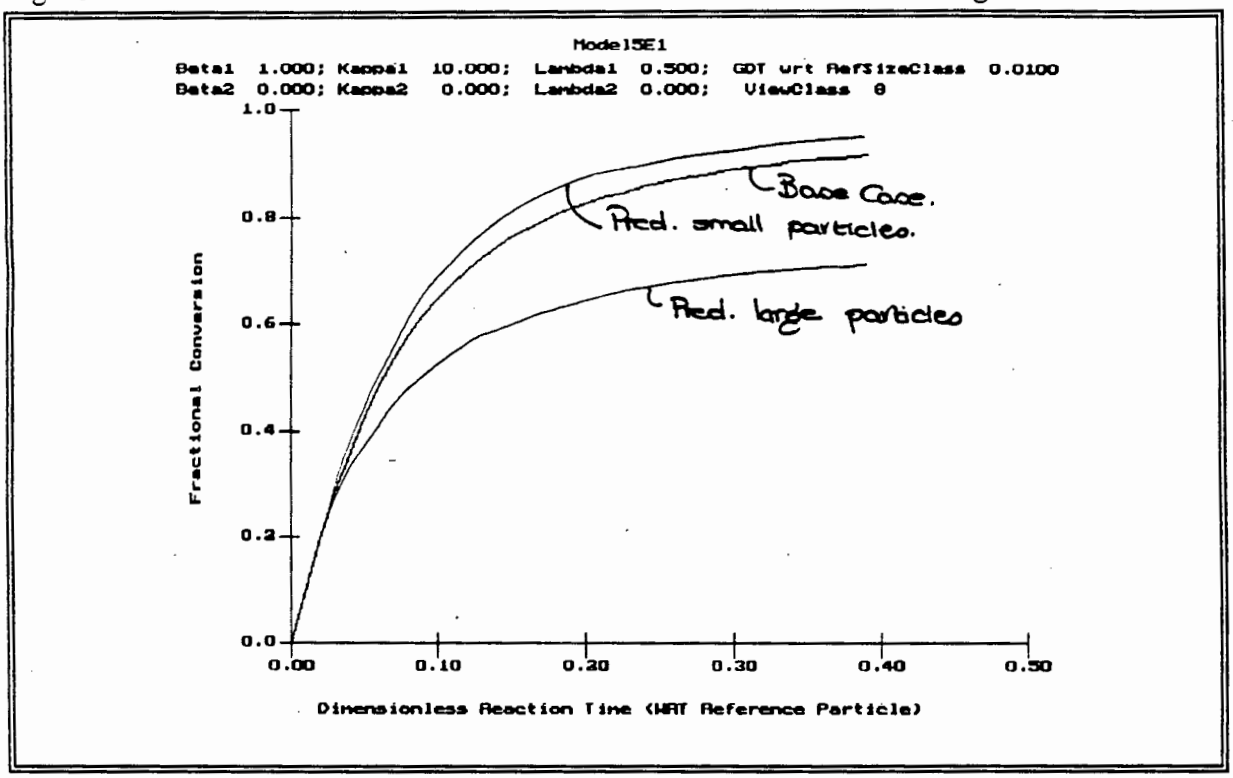


Figure 5-10 presents the fractional conversion curves for these cases. The size distribution with predominantly smaller particles is similar to the base case size distribution. As such, the fractional conversion for the two cases is very comparable. As expected, the simulation using the size distribution with predominantly smaller particles predicts a higher conversion at all times compared to the base case simulation. Further, the linear, kinetic controlled region is larger for the simulation consisting of smaller particles. The fractional conversion is significantly delayed for the size distribution consisting of larger particles. In effect, the conversion is being delayed by diffusional

resistances.

Figure 5-10. Fractional Conversion for the Size Distributions Investigated.



5.7.3 Effect of the Location of Hazardous Constituents on the Fractional Conversion.

The following scenarios, summarised in Table 5-4 and Figure 5-11, were used to investigate the effect of contaminant location on contaminant release. The first case represents the physical situation in which no hazardous constituents have been concentrated onto the surface of the particles. In other words these particles have a surface hazardous constituent concentration equal to their bulk concentration. The second and third cases represent the cases where the surface concentration of the smallest particle is 5 times and 10 times the bulk hazardous constituent concentration respectively. The $\zeta_{1,k}$ values were determined as before, using equations similar to equation (5-19), and the parameters for the reference particle size were defined previously in Table 5-2.

Table 5-4. Summary of the conditions used to investigate the effect of Hazardous Constituent Location on Fractional Conversion.

Particle Size. (mm)	Size Distribution.	Case 1-$\zeta_{1,k}$	Case 2-$\zeta_{1,k}$	Case 3-$\zeta_{1,k}$
9.5	0.252	1.0	1.21	1.47
8.5	0.428	1.0	1.24	1.53
7.5	1.045	1.0	1.27	1.60
6.5	2.564	1.0	1.31	1.69
5.5	6.191	1.0	1.36	1.82
4.5	14.061	1.0	1.44	2.00
3.5	27.105	1.0	1.57	2.29
2.5	34.247	1.0	1.80	2.80
1.5	13.846	1.0	2.33	4.00
0.5	0.260	1.0	5.00	10.00

Figure 5-11. Summary of the Hazardous Constituent Location Data used in the Simulations.

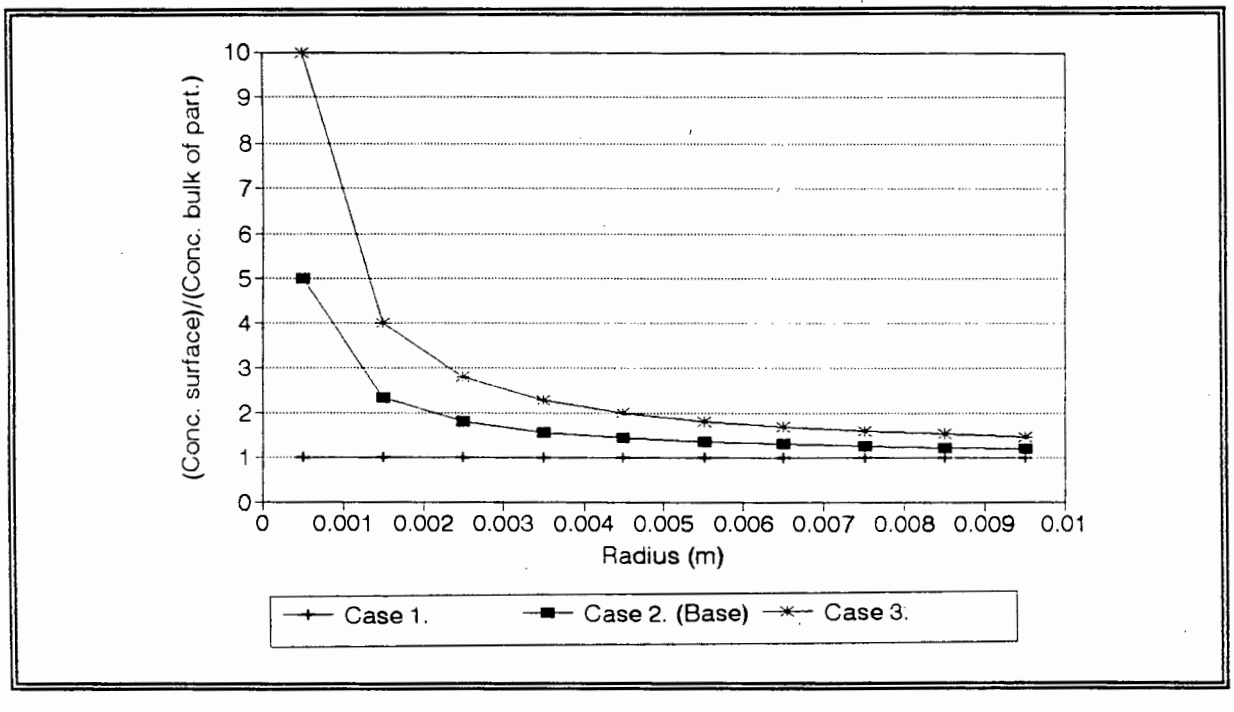
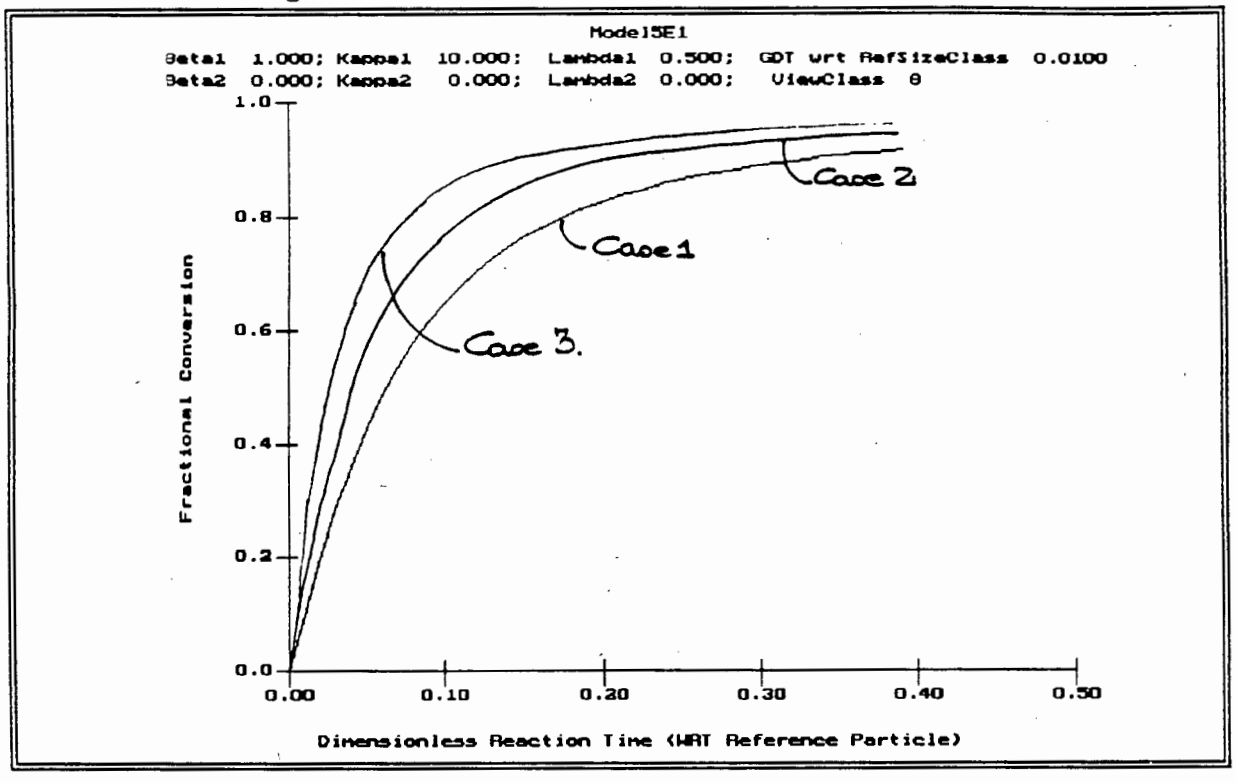


Figure 5-12 shows the conversion curves for the different scenarios. As expected, the cases where hazardous constituents are concentrated onto the surface obtain higher fractional conversions for all times. This is caused by the fact that hazardous constituent deposits on the surface of the particles cannot be retarded by diffusional resistances (although they may be retarded by mass transfer resistances which have not been accounted for in this work). Also note that as the surface concentration increases, so too, the straight line kinetic controlled portion of the conversion graph increases.

Figure 5-12. Fractional Conversion for Surface Hazardous Constituents Concentrations Investigated.



5.8 Fitting the Model to CSTR Results.

The complete CSTR model which predicts the fractional conversion as a function of dimensionless time for a size distribution of particles has many parameters. This is due to the fact that each size class of particles has the following parameters associated with it:

$$\beta_{i_k} \qquad \lambda_{i_k} \qquad \kappa_{pi_k} \qquad \kappa_{si_k}$$

It is not possible to determine all these parameters simultaneously from a single CSTR experiment. Instead, a CSTR experiment which contains only a single size class of particles, termed the reference size class, must be used to determine the parameters for that size class. If the properties of the particles in the remaining size classes are sufficiently similar to the reference size class then the remaining parameters can be determined through the relationships defined in equations (5-8) to (5-12). Alternatively when the particles in the other size classes exhibit sufficiently different properties to the reference size class, a CSTR experiment for each size class needs to be conducted to determine all the parameters.

5.8.1 Model Requirements.

The following information needs to be known in order to fit the model parameters for a single size class of particles to CSTR results:

- The relative stoichiometric consumption of the fluid reagent for each participating solid reactant;
- the effective concentrations of the participating species;
- the reaction orders with respect to the solid reactants;
- the ratio C_{si}/C_{pi}, $(\zeta_{i,k})$, for each species in the reference size class; and;
- the voidage of the particles.

Note that the discussion on model requirements presented in section 3.8.1 with respect to the chemical reactions, effective concentrations of the species and reaction orders is equally applicable to the CSTR model.

5.8.2 Fitted Parameters.

The fitted parameters in this model are the κ_{pi} and κ_{si} groups. Notice that there is one κ_{pi} and one κ_{si} group for every chemical reaction taking place. These groups are fitted by comparing the fractional conversion curves determined by the model to experimentally determined conversion curves.

Before the comparison between the model predictions and the experimental results can be made, the experimental results need to be converted into appropriate dimensionless form. Typical experimental results will be in the form of curves which represent the dissolved concentration of the hazardous constituents in the bulk fluid as a function of time. These curves can be converted into fractional conversion versus time curves. This is a straight forward procedure because the total leachable concentration of each species is known. (The total leachable concentrations of each species can be obtained by conducting a leach test until no further hazardous constituents are released. The bulk fluid concentrations can then be used to back calculate the leachable concentrations within the waste particles.) Non-dimensionalising the time variable is a more complicated procedure. It is usually not feasible to use the dimensionless diffusion time, τ, defined in equation (4-15), to non-dimensionalise the time. The reason for this is that the dimensionless diffusion time includes the effective diffusivity for the particle which is unknown. The dimensionless group defined as:

$$\kappa_{pi}\beta_i\tau = \frac{k_{pi}C_{pi,0}^{\phi_{pi}}C_{A,0}}{C_{Ei,0}}t \qquad (5\text{-}21)$$

and which is termed the dimensionless reaction time can be used to non-dimensionalise the time variable. The reason for this is that the total extractable grade, $C_{ei,0}$, and the group $k_{pi}C_{pi}^{\phi si}$ $C_{A,0}$, which represents the initial reaction rate, are known. (The initial reaction rate can be determined from the initial slope on the experimental concentration versus time graph). Thus the dimensionless reaction time, equation (5-21) can be used to non-dimensionalise the time variable to yield a totally dimensionless experimental curve. By coding the model to predict the conversion versus dimensionless reaction time curves, the model predictions can be compared to the experimental results.

Note that if the hazardous constituents on the surface of the particle can be assumed to react in the same manner as the hazardous constituents within the particle then only the κ_{pi} groups are fitted to the experimental results.

5.9 Applications and Limitations of the Model.

5.9.1 Applications of the model.

As the name implies, the CSTR model is particularly suited to the analysis of CSTR experimental data. Once parameters for a reference size class have been determined, the effects of the following factors on the release of hazardous constituents can be investigated:

- particle size distribution;
- hazardous constituent concentration;
- hazardous constituent location; and;
- competing reactions.

The CSTR model is also used to determine the model parameters for the Columnar Model presented in the next chapter.

5.9.2 Limitations of the model.

The CSTR model has two main limitations. Firstly it cannot be used in cases where there are significant mass transfer resistances because these have not been accounted for in the model. Should the functional relationship between the bulk fluid reactant and the surface fluid reactant concentration be known, it would be easy to incorporate the mass transfer resistances into the model. The second limitation is that the model cannot be applied to a case where the diffusion of the dissolved solid species is rate limiting. Dixon [1993] has included this aspect into his previous model [1992]. The inclusion of dissolved species transport limitations into the model are discussed in section 7.

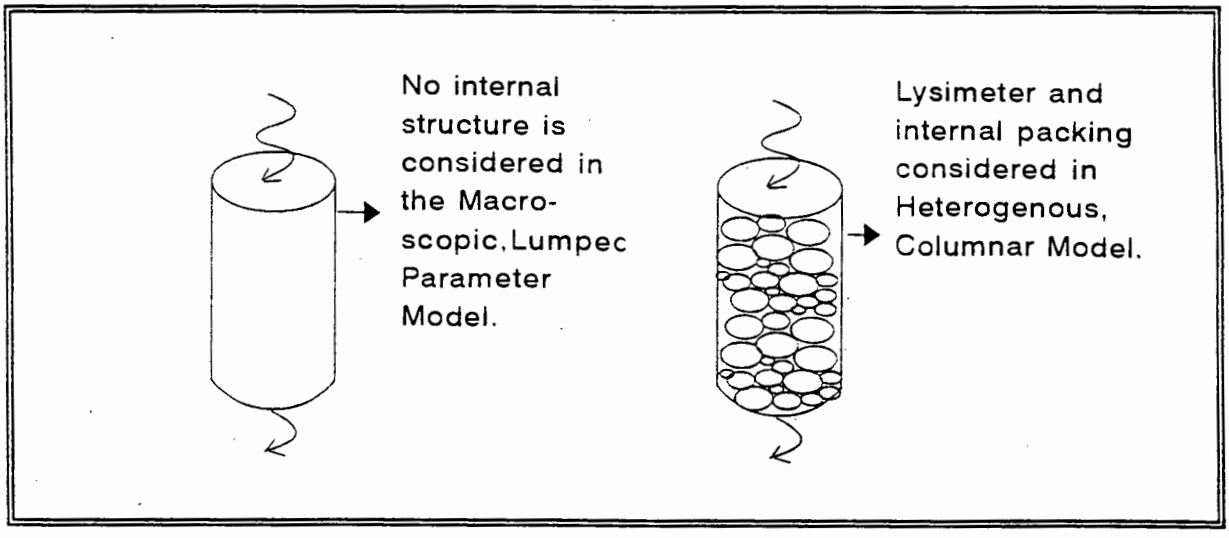
Chapter 6. A Microscopic, Columnar Model to Describe Leachate Generation and Mobility in Granular Waste Deposits.

The macroscopic, lumped parameter model, which was described in Chapter 3, essentially modelled a waste deposit as a single columnar entity. No particulate features within the column were identified in this model. Instead, the contributions of the effective chemical reaction rates, hazardous constituent location and hydrodynamic aspects where lumped together in a parameter which described the effective rate of release of hazardous constituents from the column.

As previously discussed, the main limitations of this model result from using this lumped parameter approach. The most notable limitation is that the individual contributions of the effective chemical reaction rates, hazardous constituent location or hydrodynamic aspects on contaminant release cannot be determined. The overall effective chemical reaction rate for the column is a function of the size distribution of the particles within the column since different sized particles react at different rates. Hazardous constituent location within the individual particles also plays an important role in the release of these constituents. This too in known to be a function of particle size and has been discussed in section 2.1.1. The hydrodynamic aspects, previously discussed in section 2.1.2,are affected by the superficial velocity of the fluid entering the column and by the size distribution and packing of the particles within the column. Since these factors are all dependent on particulate features, a more detailed model which includes particulate information has been investigated.

The heterogenous, columnar model is essentially a columnar, non-catalytic, packed bed reactor type model. The main difference between it and the macroscopic, lumped parameter model is that the particles within the column are included in the model description. Figure 6-1 graphically shows the differences between the two models.

Figure 6-1 Graphical comparison between the macroscopic, lumped parameter model and the heterogenous, columnar model.



Two methods to determine an appropriate solution strategy for the heterogenous, columnar model are presented. The first and more simple method is based on the heap leaching modelling strategy of Roman *et al.* [1974].. The second method, which is a more rigorous mathematical approach, results in a solution strategy identical to Roman's strategy and has been included merely as a validation of the simpler approach.

This presentation is followed by a section which indicates how a global wetting factor has been incorporated into the solution strategy. Next, the details of suitable computer routines which have been written to implement the solution strategy are discussed. The model has been verified against experimental results which have been presented in a paper by Roman *et al.* [1974]. The last section in this chapter summarises the experimental data which is required in order to verify the applicability of the heterogenous columnar model to describe the leaching of hazardous constituents from waste deposits.

The following chapter summarises the application of the heterogenous, columnar model to predict the release of hazardous constituents from waste deposits. This chapter also includes a discussion on the advantages of the heterogenous, columnar model over existing models as well as giving details with respect to possible extensions to the model.

6.1 A Modelling Strategy based on Heap Leaching Models.

The strategy adopted in heap leaching models has already been discussed in section 2.4. In summary, the general strategy is to conceptually divide the heap into columnar sections. Each column is then further sub-divided into a series of disks within which the fluid concentration is assumed to be spatially uniform. It is important to note that if the column is divided into too few disks, the assumption that the fluid reagent concentration within each disk being spatially uniform will no longer be valid. The accuracy of the assumption with respect to uniform spatial fluid concentration will increase as the number of disks within the column increase. These sub-divisions of the deposit and columnar sections were shown in Figure 2-5.

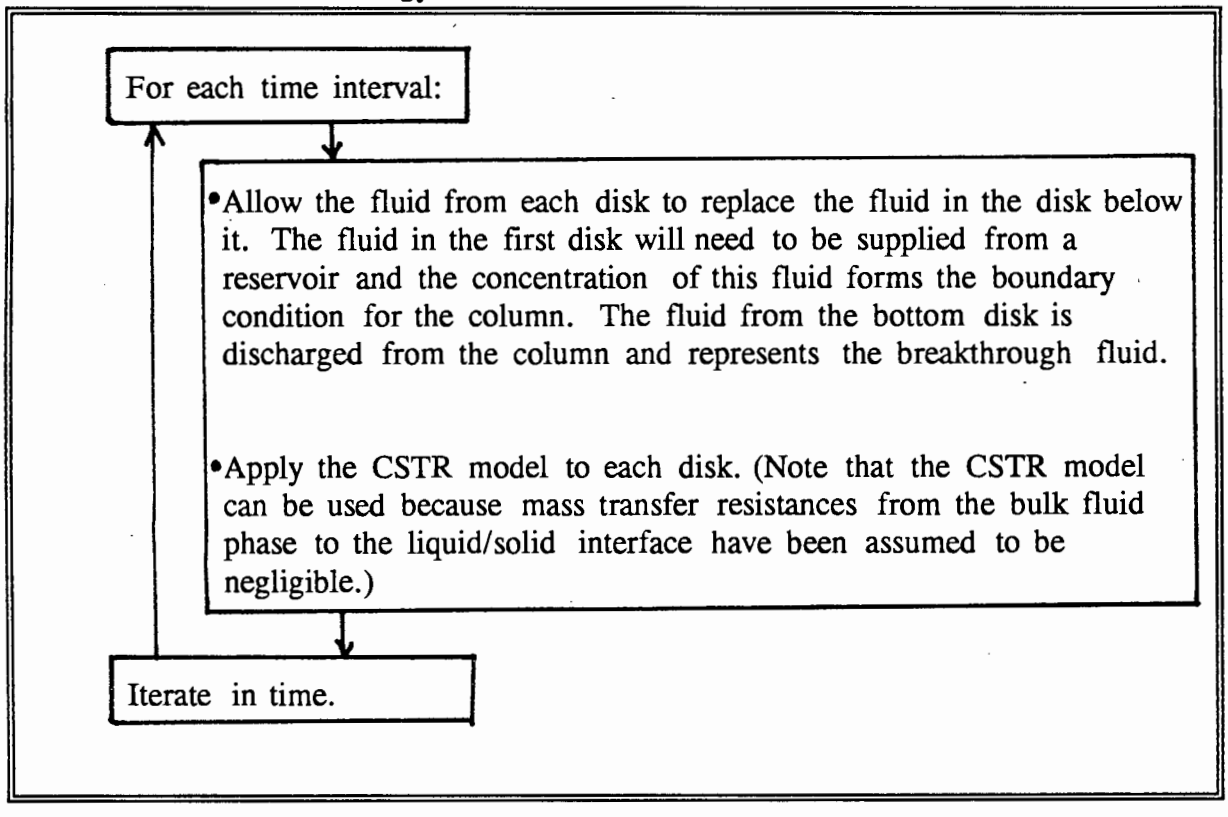
The flow of fluid through the column is simulated by allowing the fluid from one disc to replace the fluid in the disk below it at specified time intervals. During each time interval, the fluid reactant is allowed to react with the solid particles resulting in the precious metals, or hazardous constituents, being released.

It is important to note that, within each time interval, each disk in the column consists of an assembly of particles associated with a finite volume of fluid reagent. Since film mass transfer effects have been assumed to be negligible, the decrease in the fluid reagent concentration within each disk and time interval can be described by the *CSTR* model which was developed in the previous chapter *even although the physical characteristics of a CSTR are very different from that of a 'reactor slice' or disk.* Since each disk is being described by a CSTR model, the behaviour of the column is being approximated by a number of tanks in series. (The number of tanks corresponds to the

number of disks in the column.)

The modelling strategy has been summarised in Figure 6-2.

Figure 6-2. Summary of the Solution Strategy used which was based on Roman's Solution Strategy.



The equations which describe the release of hazardous constituents within each disk during each time interval, which constitute the CSTR model, have been presented in the preceding chapter.

Effectively the time intervals which elapse between successive fluid replacements represent the time which the fluid would have taken to flow through the disk. Thus for a constant volumetric flowrate, q, the time intervals can be determined from:

$$\Delta t = \frac{Fluid\ Volume\ of\ Disc}{q} = \frac{\epsilon_{col} Col_{\%\ Sat}\ \Delta\xi_{Col}\ L\ A}{q} \qquad (6-1)$$

where ϵ_{Col} column voidage;
$Col_{\%\ Sat}$ saturation of the void space within the column;
$\Delta\xi_{Col}$ dimensionless length of a disk;

L length of the column; and;
A area of column.

For a volumetric flowrate which is a function of time, Δt will need to be determined from $(t_{n+1}-t_1)$ where $(t_{n+1}-t_n)$ is obtained from:

$$\int_{t_n}^{t_{n+1}} q(t)\,dt=\epsilon_{Col}\ Col_{\%\ sat}\ \Delta\xi_{Col}\ L\ A \qquad (6-2)$$

Since the computer routines which have been written for the CSTR model are in dimensionless format, it would be convenient to express the time intervals between successive fluid replacements in dimensionless form. These time intervals can be converted to non-dimensional quantities by dividing them by a reference time period. One appropriate reference time period is the space time of the column defined by equation (3-13):

$$T=\frac{L}{u^*} \qquad (6-3)$$

where u^* is a reference fluid velocity (percolation velocity) which has arbitrarily been set at 1m per 24 hours.

6.2 A Modelling Strategy based on a Rigorous Mathematical Approach.

The starting point for this analysis is a mass balance equation which describes the bulk fluid reactant concentration within the column. To obtain this equation an approach very similar to the one adopted in the macroscopic model development can be used. The only difference is that the rate term, r_{ai}, which represents the *'production'* of fluid reagent due to chemical reaction, is retained in equation (3-5) and not replaced by an overall lumped rate expression such as equation (3-2). The modified form of equation (3-5) is:

$$\epsilon_{Col}\ Col_{\%\ sat}\ \frac{\partial C_{A_{Col}}}{\partial t}=-u\frac{\partial C_{A_{Col}}}{\partial z}+\sum_{i=1}^{n} r_{a_i} \qquad (6-4)$$

The same equation expressed in dimensionless terms is:

$$\frac{\partial \alpha_{Col}}{\partial \tau'} = -DG1 \ \frac{\partial \alpha_{Col}}{\partial \xi_{Col}} + \frac{L}{u^* C_{A_0} \epsilon_{Col} \ Col_{\% \ Sat}} \sum_{i=1}^{n} r_{A_i} \qquad (6-5)$$

where DG1 is as previously defined in equation (3-14), and has the physical significance of being the ratio of the fluid percolation velocity to the reference fluid percolation velocity.

The appropriate initial and boundary conditions for this equation are:

$$\alpha_{Col}(\xi_{Col}, 0) = 0 \qquad (6-6)$$

$$\alpha_{Col}(0, \tau') = 1 \qquad (6-8)$$

Equation (6-5) is a first order hyperbolic partial differential equation. As discussed in Chapter 3, the method of characteristics can be used to convert the partial differential equation into two ordinary differential equations. These equations are:

$$\frac{d\tau'}{d\xi_{Col}} = \frac{1}{DG1} \qquad (6-9)$$

and

$$\frac{d\alpha_{bulk}}{d\tau'} = \frac{L \ \Sigma r_i}{C_{A \ bulk_0} \epsilon_{Col} \ Col_{\% \ Sat} \ u^*} \qquad (6-10)$$

It is interesting to note that the term on the right hand side of equation (6-10) represents a normalised rate of consumption of fluid reactant over the rate of fluid reactant replenishment. In this regard, this term represents the same ratio as the previously defined dimensionless group *DG2*.

Before these equations can be solved, suitable expressions for the rate of 'production' of fluid reactant A by the i reactions, r_{Ai}, need to be determined.

Rather than redevelop these expressions, recall equation (5-16):

$$\sum_{k=1}^{M} \left[\frac{V_{Part,k} \rho_0 (1-\epsilon_0)}{b_i} \sum_{i=1}^{N} \frac{dC_{s_i}}{dt} - D_e \nabla C_A|_{R_k} \frac{3 V_{Part,k}}{R_k} \right] = V_{Liq.} \frac{dC_A}{dt} \qquad (5-16)$$

where $V_{Part,k}$ is the total volume of particles in size class k;
M number of sizes classes; and;
R_k Radius of the particles in size class k.

This equation expressed in dimensionless format is:

$$\sum_{k=1}^{M} [- \sum_{i=1}^{N} \frac{\Upsilon_{Part,k}}{\Theta_k^2} \kappa_{si,k} \sigma_{si,k}^{\phi_{si}} \alpha_{bulk} - \frac{3\Upsilon_{Part,k}}{\Theta_k^2} (\frac{\partial \alpha}{\partial \xi})_k |_{\xi=1}] = \frac{V_{Liq}}{\epsilon_0 \overline{V}_{Part.}} \frac{d\alpha_{bulk}}{d\overline{\tau}}$$

(5-17 or 6-11)

This equation describes the change in the bulk fluid reactant concentration as a result of the fluid reactant being consumed by chemical reactions within a range of different sized particles. Note that the time variable used in equation (6-11) corresponds to a dimensionless diffusion time corresponding to the reference size class of particles. In contrast, the time variable used in equation (6-10) corresponds to a dimensionless space time of the column. If the column time increment is set equal to the reference size class time increment then equation (6-11) can be used in place of equation (6-10). This implies:

$$\frac{d\alpha_{bulk}}{d\overline{\tau}} = \frac{d\alpha_{bulk}}{d\tau'}$$

$$= \frac{\epsilon_0 \overline{V}_{Part.}}{V_{Liq}} \sum_{k=1}^{M} [- \sum_{i=1}^{N} \frac{\Upsilon_{Part,k}}{\Theta_k^2} \kappa_{si,k} \sigma_{si,k}^{\phi_{si}} \alpha_{bulk} - \frac{3\Upsilon_{Part,k}}{\Theta_k^2} (\frac{\partial \alpha}{\partial \xi})_k |_{\xi=1}]$$

(6-12)

where ϵ_0 *particle* porosity;
$V_{Part.}$ in this case represents the total volume of the reference size class particles within the spatial increment in the column;
$V_{Liq.}$ in this case represents the volume of fluid within the spatial increment in the column;
$\Upsilon_{Part,k}$ defined in equation (5-18) and which represents the ratio of the volume of the particles in size class k to the volume of the particles in the reference size class; and;
Θ_k defined in equation (5-5) and which represents the ratio of the average radius of the particles in size class k to the average radius of the particles in the reference size class.

The overall mathematical solution strategy can be summarised as follows. Firstly set a suitable spatial increment, $\Delta\xi_{Col}$, for the column. This increment corresponds to the length of a disk in the previous heap leaching analysis. Equation (6-9) is then used to

determine the time increment for the column. As shown in Chapter 3, an alternative method to determine the time increment for the column is:

$$\Delta\tau' = \frac{u^*(t_{n+1} - t_n)}{L} \qquad (3\text{-}31 \text{ or } 6\text{-}13)$$

where $(t_{n+1}-t_n)$ is obtained from:

$$\int_{t_n}^{t_{n+1}} u(t)\, dt = \epsilon_{Col}\, Col_{\%\, Sat}\, \Delta z$$

$$= \epsilon_{Col}\, Col_{\%\, Sat}\, \Delta\xi_{Col}\, L \qquad (3\text{-}32 \text{ or } 6\text{-}14)$$

Notice the similarity between equation (6-14) and equation (6-2). In effect these two equations are identical.

Within each time increment and within each spatial increment the change in the bulk fluid reactant concentration can be solved using equation (6-12). Before this equation can be solved however, the fluid reagent profiles within each particle size class need to be known. These can be determined by applying the particle scale model of Dixon to each size class of particles. This approach is summarised in section 5.4.

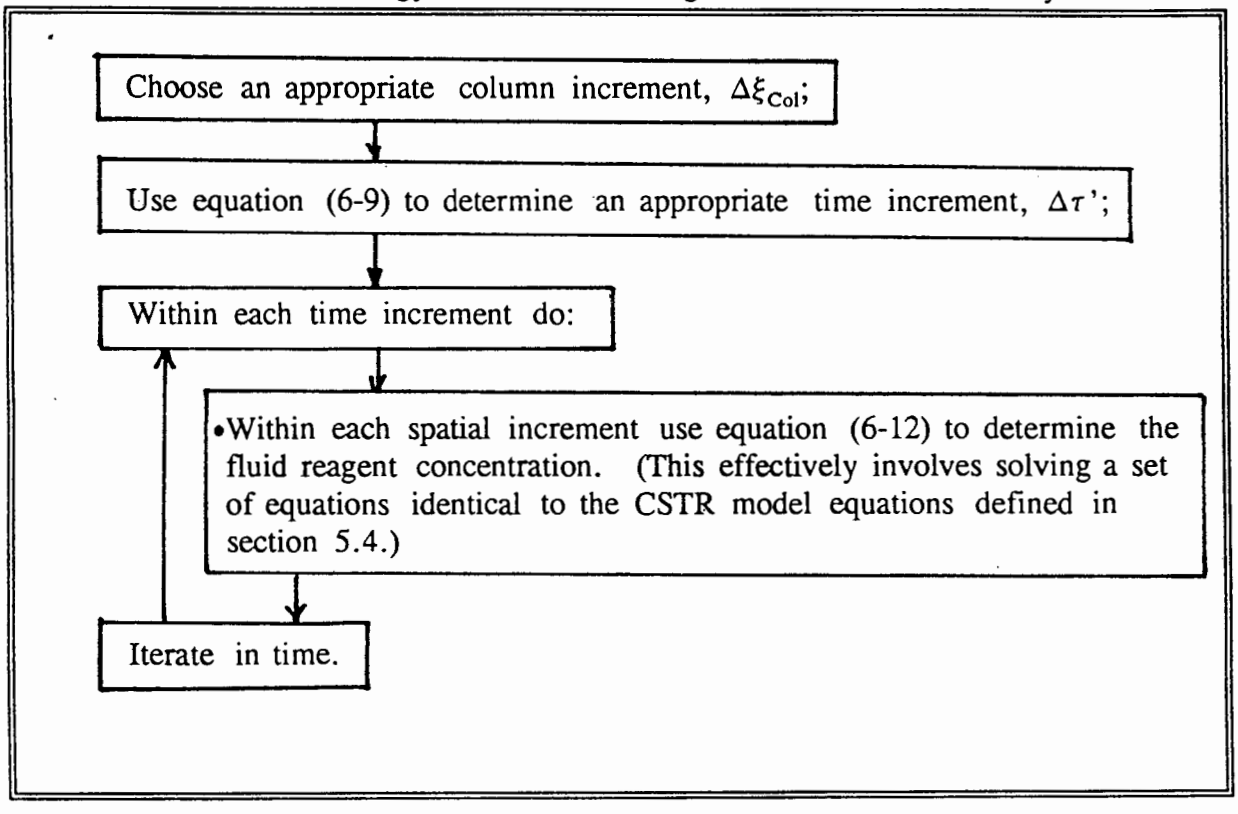
It is important to note that in reality it is not always feasible to set the column scale time increment equal to the particle scale time increment. The reason for this is that the column time increment is usually sufficiently large to result in the Crank-Nicolson method, used to solve the particle scale concentration profiles, becoming totally inaccurate. This problem is easily overcome by using a much smaller time increment for the particle scale calculations and repeating these calculations until the cumulative time increment for these calculations is equal to the column scale time increment. This can be summarised as:

$$n\, \Delta\tau_{Particle\ Scale} = \Delta\tau_{Column\ Scale} \qquad (6\text{-}15)$$

where n is the number of times that the particle scale calculations need to be repeated.

The solution strategy has been summarised in Figure 6-3.

Figure 6-3. Solution strategy derived from a rigorous mathematical analysis.



Comparison of Figures (6-2) and (6-3) shows that the solution strategy derived from the rigorous mathematical analysis is identical to the simpler heap leaching modelling strategy.

6.3 Inclusion of a Global Wetting Factor into the Solution Strategy.

A global wetting factor has been incorporated into the solution strategy to account for the partial wetting of particles. The global wetting factor represents an average fraction of particle surface area wetted by fluid reagent. As an example, a global wetting factor of 0.5 implies that all of the particles are only half covered by fluid reagent. It has further been assumed that sectors within particles which are bordered by dry external surfaces will not release hazardous constituents. In effect, this implies that even with dry zones within a particle, the diffusion will remain one dimensional - in the radial direction only.

From the above discussion, wetting can be seen to influence the release of hazardous constituents in the following manner. Firstly, the degree of wetting determines the solid which is available to react with the fluid reagent. As before, a global wetting factor of 0.5 implies that only half of the particle volume is available to react with the fluid reagent. This in effect also determines the maximum release of hazardous constituents achievable. A column with a wetting factor of 0.5 will have a maximum contaminant release (conversion) of 50%.

Using this information the effect of partial wetting was incorporated into the solution strategy as follows. Equation (6-12) has previously been used to determine the bulk fluid reagent concentration within the particles as a function of time for totally wetted particles. This equation can still be used in the case of partially wetted particles if $V_{Part.,k}$ is defined as the product of the volume of the particles in size class k within a spatial increment within the column and the global wetting factor. This can be summarised as:

$$V_{Part,k} = \gamma \; V_{Part,k_{Actual}} \qquad (6-16)$$

where $V_{Part,k,Actual}$ in this case represents the actual volume of the particles in size class k within the spatial increment in the column;

$V_{Part,k}$ in this case represents the volume of the particles in size class k which is available to react with the fluid reagent within the spatial increment in the column; and;

γ is a global wetting factor.

By using a global wetting factor, all particles have been assumed to have an equal portion of their external surface area covered by fluid reagent. If more detailed information with respect to particle wetting as a function of particle size is available, it could be included into the model as follows:

$$V_{Part,k} = \gamma_k \; V_{Part,k_{Actual}} \qquad (6-17)$$

where γ_k is a size dependent wetting factor.

6.4 Suitable Computer Routines for the Heterogenous Columnar Model.

Program Model6C1.PAS and Model6C2.PAS are suitable computer codes for the heterogenous, columnar model. As before, Model6C1.PAS is a code which assumes a solid reaction order of unity while Model6C2.PAS can accommodate a variable reaction order.

These codes are sufficiently large to justify some explanation as to their organisation. From the discussion on the solution strategy in the previous section it became evident that the codes need to calculate, as a function of time, the reactant profiles within each particle size class within each disk of the column. This information is then related to the fluid reactant and dissolved hazardous constituent profiles within the column itself.

The particle scale calculations, which are essentially identical to the CSTR model calculations are incorporated in Unit6C1 and Unit6C2. These units are almost identical to previous PASCAL codes Model5E1 and Model5E2 respectively. All the particle scale parameters are defined and set in these units. The column scale calculations are performed in the main program Model6C1 or Model6C2. All the column parameters are defined and set in these programs. Figure 6-4 summarises the overall organisation of program Model6C1.PAS.

The output of these codes is a display of the solid and fluid reactant concentration profiles for the smallest, largest and reference size class of particles as a function of time and position within the column. This information helps to visualise how the different size classes are reacting in different positions of the column at different times. This information is most useful because it helps to identify which size classes of particles react in a chemical kinetic controlled manner and which react in a diffusion controlled manner. In effect, this information can be used to determine the relative importance of intrinsic chemical kinetics and diffusive resistances on the overall rate of release of hazardous constituents. A typical display of the solid and fluid reactant concentration profiles is shown in Figure 6-5.

Once the required iterations in time have been performed and the calculations are completed the programs display the conversion and breakthrough curves as a function of dimensionless time. Although the computer routines do not presently display the concentration profiles within the column as a function of time, the data required for these curves is already calculated. Thus this feature can easily be incorporated into the computer routines if required.

Figure 6-4. Summary of the overall organisation of Program Model6C1.PAS.

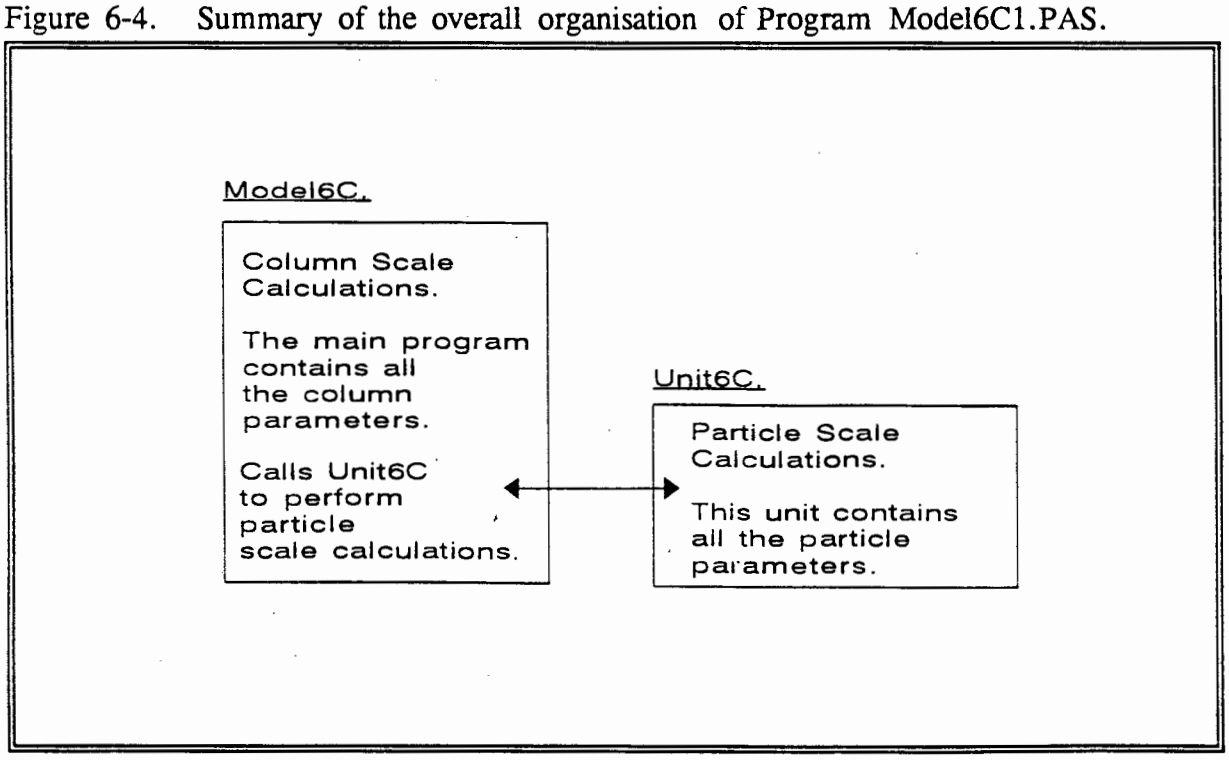
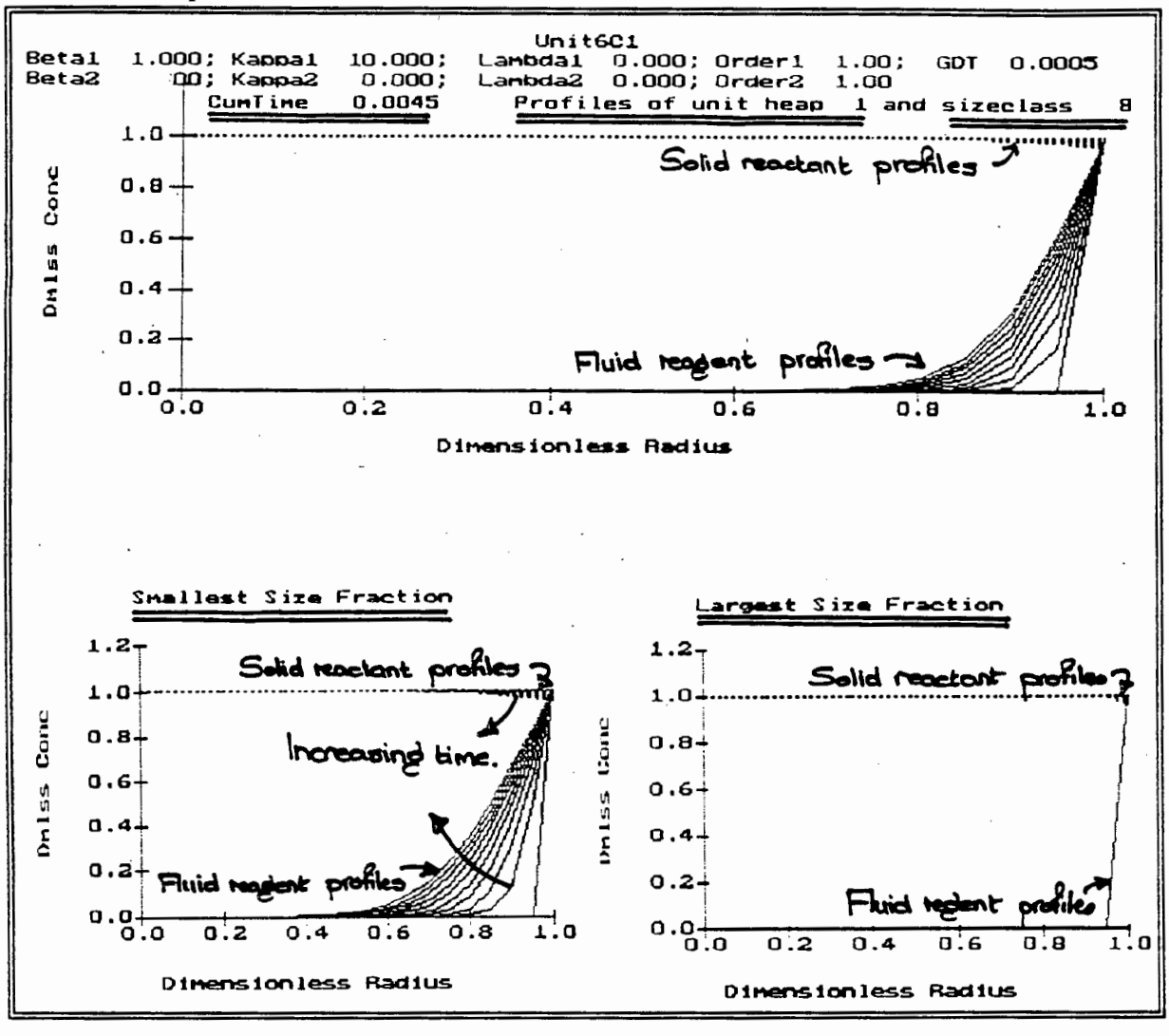


Figure 6-5. A typical display produced by Model6C1.PAS showing the solid and fluid reagent profiles for the smallest, largest and reference size class of particles.



It is worth pointing out that these programs include numerical step size scaling procedures. As such, the programs have been written to detect numerical stability and to increase the step size used in the calculation strategy whenever appropriate. The main advantage of using such procedures is that the overall calculation time can be greatly reduced.

Copies of the code as well as solution algorithms can be found in Appendix V.

6.5 Verification of the Computer Routines.

Roman *et al.* [1974] conducted two lysimeter tests which were significantly different in terms of lysimeter dimensions, fluid reagent concentrations and flow rates, and the size distributions of particles, in order to determine the predictive capability of their model.

In summary, they used one set of data to determine the single unknown parameter in their model and then used the model to predict the second set of data. Although the heterogenous, columnar model presented in this chapter is capable of incorporating far more complexities than the model presented by Roman *et al.* [1974], its predictive capacity can be tested in a similar manner.

The lysimeter experiments conducted by Roman were concerned with copper extraction. The physical properties, operating conditions and size distributions of the particles used in the two experiments have been reproduced from Roman [1974] and are summarised in Table 6-1 and 6-2, and in Figure 6-6 and 6-7 .

Roman *et al.* did not include the voidage of the ore particles in their data and this value has been assumed to be 1%. It is important to note that the κ_{pi} parameter used in the heterogenous columnar model is the ratio of the chemical reaction rate within a particle to the *effective* rate of diffusion of fluid reagent into the particle. The effective rate of diffusion of fluid reagent into the particle is a function of the particle voidage. The effect of particle voidage on the effective diffusivity has not been considered in the heterogenous columnar model. The particle voidage only affects the heterogenous columnar model through the continuity equation (4-1). Equation (4-6) is equation (4-1) in expanded form. The heterogenous columnar model was determined to be relatively insensitive to small changes in particle voidages in the region of 1%. For this reason a particle voidage of 1% has been assumed to be an adequate assumption.

The acid consumption which was determined experimentally has been assumed to include the effects of fluid reagent holdup within the particles. This implies:

$$Acid\ Consumption = \frac{\epsilon}{1-\epsilon}\, b_{copper} \qquad (6\text{-}18)$$

where b_{copper} is the mass of copper released per mass acid consumed.

Table 6-1. Physical Properties and Operating Conditions.

	Lysimeter 1.	Lysimeter 2.
Weight of ore. (kg)	121	74.5
Column height. (cm)	176	305
Column diameter. (cm)	25.4	14.3
Solution flowrate. (*l* per min per m^2)	0.155	0.652
Acid concentration. (gp*l*)	48.8	69.7
% Voidage.	49.8	42.3
% Saturation.	36.1	37.1
Acid consumption. (g Acid per g Cu.)	3.6	3.6
Copper grade. (%)	1.9	1.9
Ore specific gravity.	2.7	2.7

Table 6-2. Size Distributions. (% Occurrence.)

Particle Size. (mm)	Lysimeter 1.	Lysimeter 2.
37.5	58.1	-
37.5 - 25.0	15.8	-
25.0 - 19.0	10.3	-
19.0 - 13.2	4.1	29.8
13.2 - 9.5	3.2	22.9
9.5 - 6.7	1.8	15.0
6.7 - 4.75	1.3	6.4
4.75 - 3.35	1.0	5.4
3.35 - 2.36	0.6	3.3
2.36 - 1.70	0.5	1.9
1.70 - 1.00	0.5	1.9
1.00 - 0.85	0.3	1.6
0.85 - 0.60	0.3	1.6
0.60 - 0.425	0.3	1.4
- 0.425	1.9	8.8

The size class used as the reference size class in both simulations was arbitrarily chosen as the 9.5mm to 13.2mm size class.

Since no appreciable amount of copper is concentrated on the surface of the particles the 'surface grade', $C_{si\,0}$, was assumed to be zero for all particles.

Using this data, the only unknown parameter in the simulation is the κ_{Copper} parameter which represents the ratio of the intrinsic chemical reaction rate within the reference size class of particles to the rate of fluid reagent diffusing into these particles. Using the experimental data for the second lysimeter, this parameter was determined to be 4.5 for the reference size class (9.5mm to 13.2mm). It is worth noting that the model is fairly sensitive to this parameter. For example, running the model with $\kappa_{Copper}=4$ and $\kappa_{Copper}=5$ resulted in significantly different curves to the experimentally determined curve. Further, the model converges to a κ_{Copper} value of 4.5 irrespective of the whether the initial guess for the κ_{Copper} is larger or smaller than 4.5.

Using a value of 4.5 for the κ_{Copper} parameter, the model was used to predict the performance of the first lysimeter. The results of both simulations compared to the experimental points are shown in Figure 6-8.

Figure 6-6. Size distribution of the particles in Lysimeter 1.

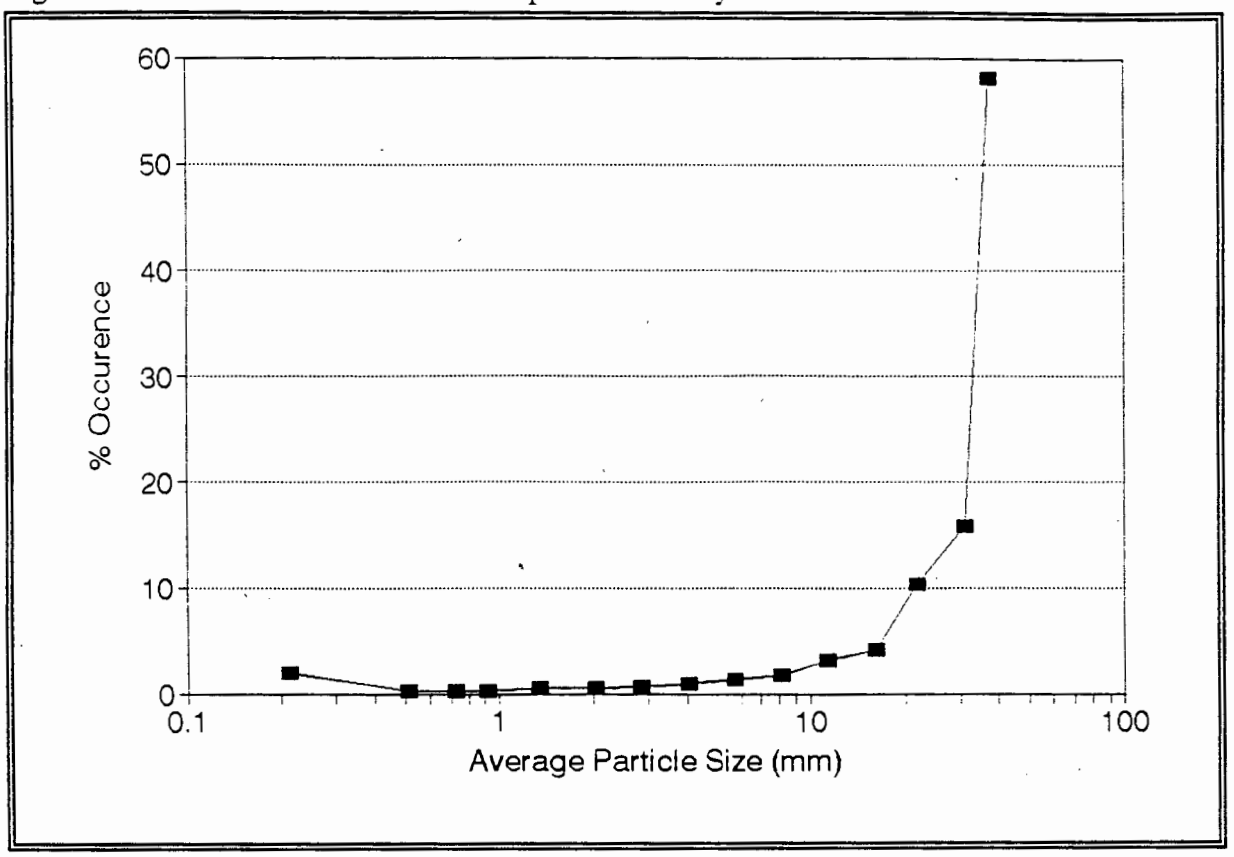


Figure 6-7. Size distribution of the particles in Lysimeter 2.

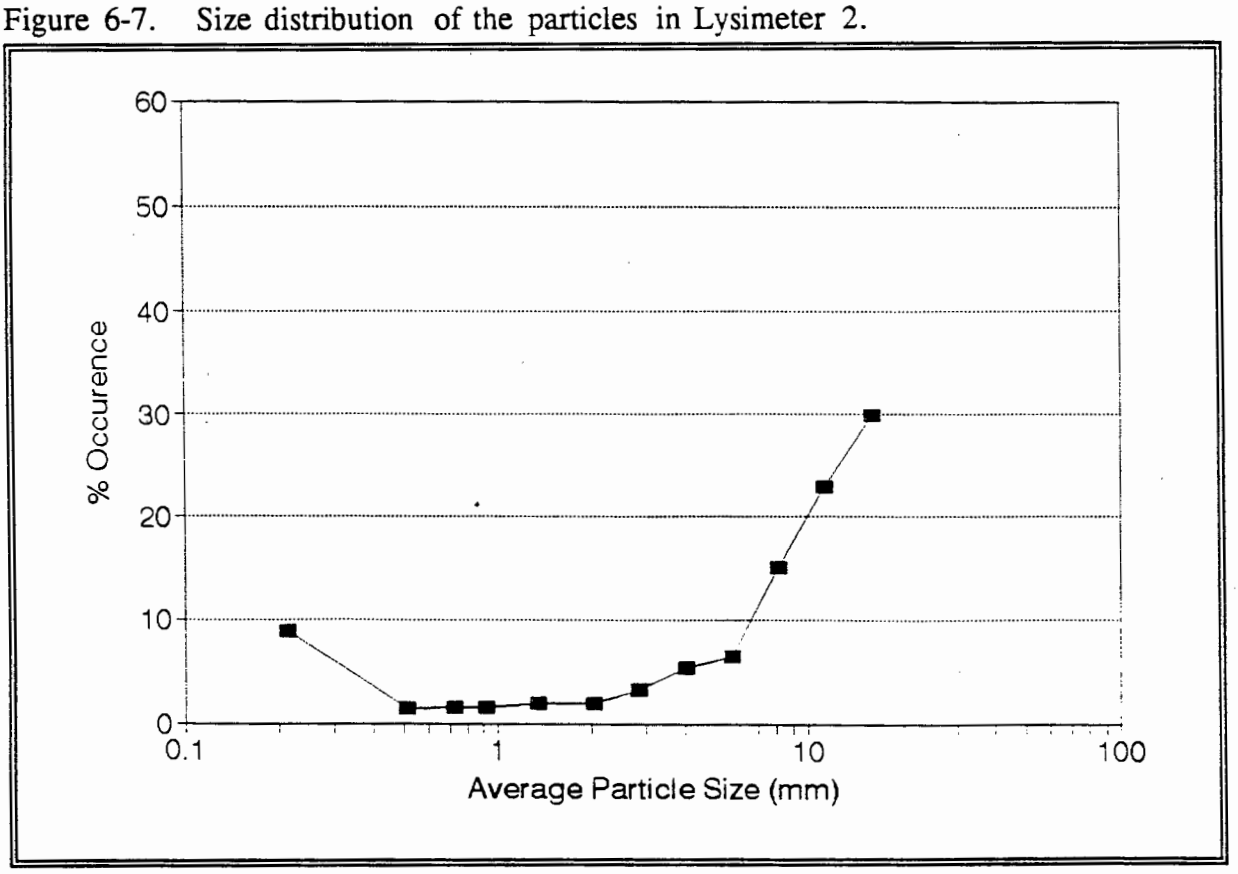


Figure 6-8. Fitted curve and predicted curve for Model6C1 compared to the experimental points of Roman [1974].

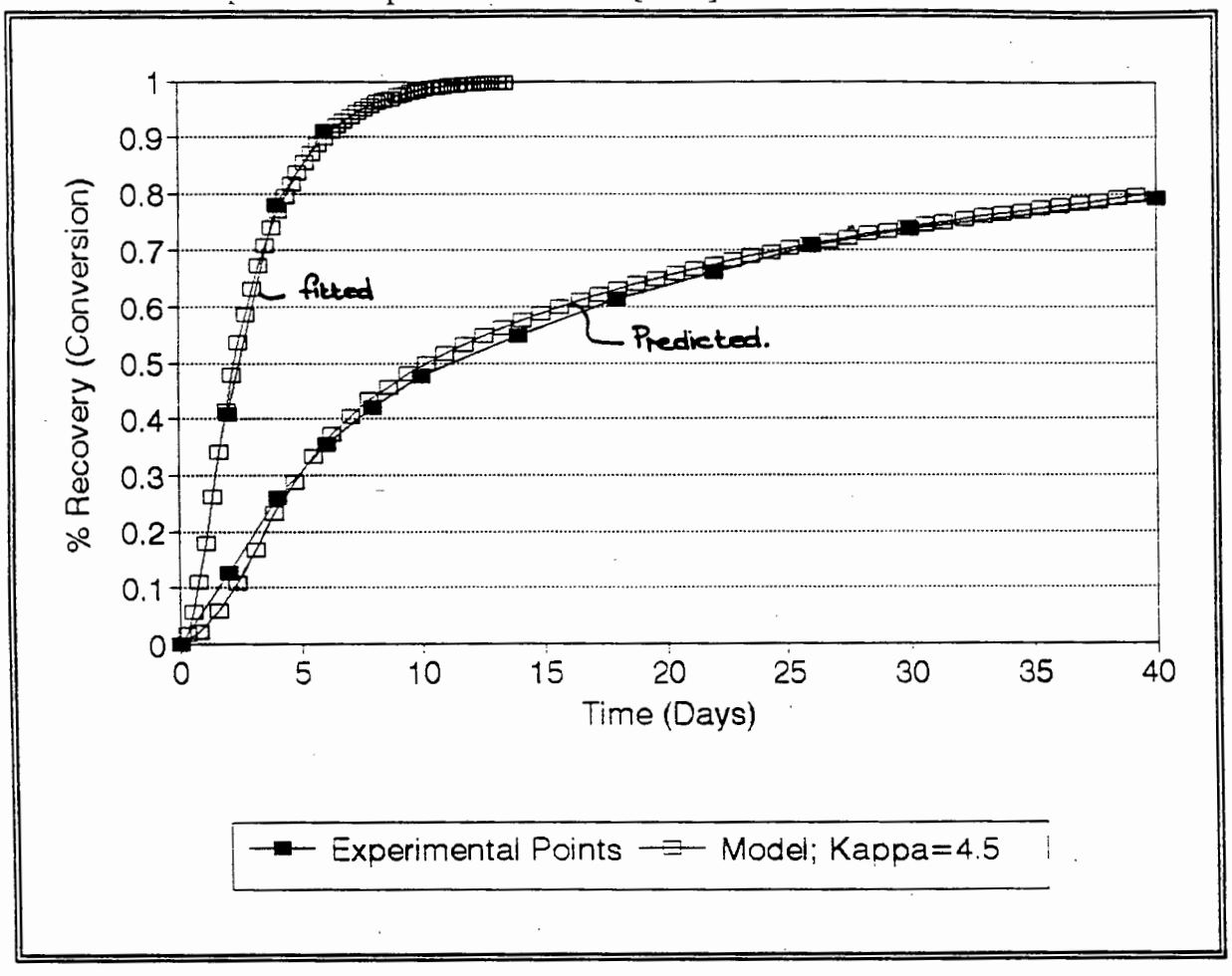


Figure 6-8 shows that Model6C1 is capable of accurately predicting the performance of heap leaching experiments.

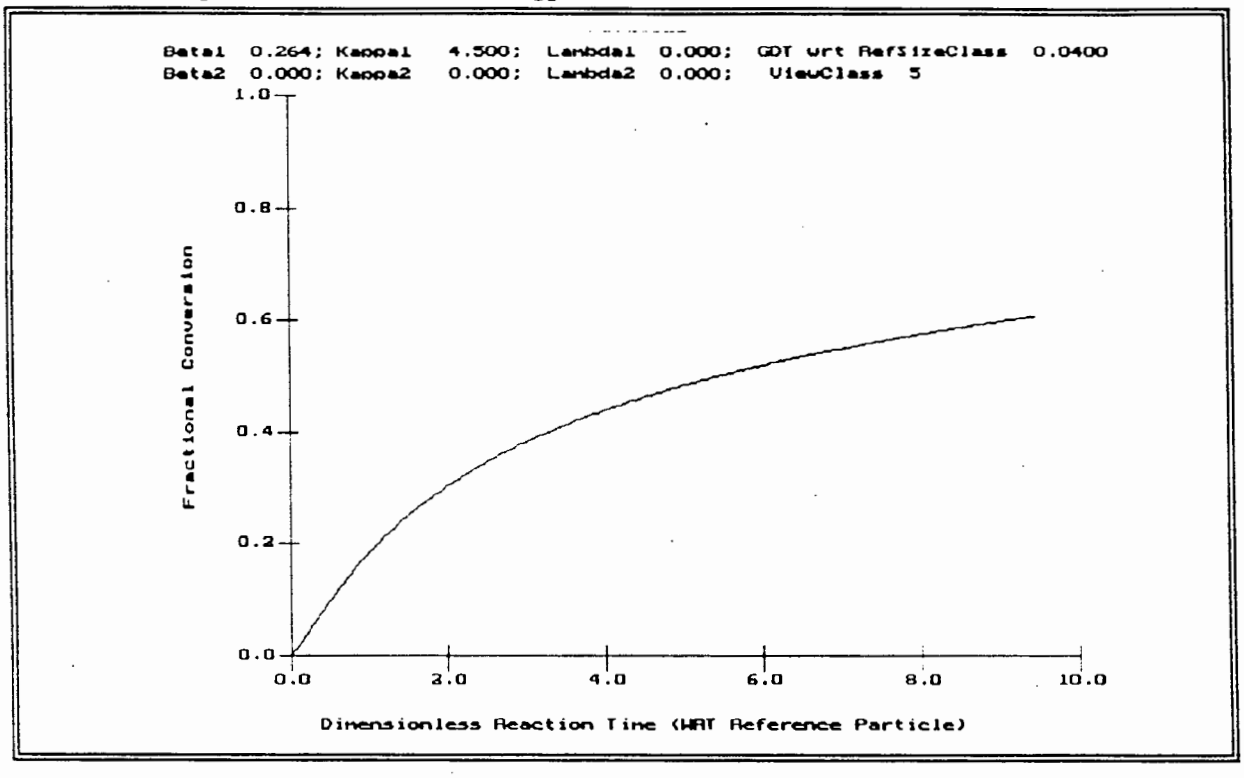
6.6 Determination of the Heterogenous Columnar Model Parameters from Appropriate CSTR type Experiments.

In the previous section, the parameters for the heterogenous, columnar model were determined from a lysimeter experiment. Where external mass transfer resistances are negligible, the model parameters can be determined from an appropriate CSTR type experiment.

The model parameters would be determined as follows. A CSTR experiment would be conducted on a single size class of particles which would then represent the reference size class of particles. The CSTR model would then be fitted to the experimental curve as discussed in section 5.8. These parameters could then be used as the reference size class parameters in the heterogenous columnar model.

This strategy will be demonstrated in reverse. The model parameters in section 6.5 were determined from fitting the heterogenous columnar model to a lysimeter result. Using the parameters determined, the predicted CSTR conversion versus time curve corresponding to 0.1*l* of 9.5mm to 13.2mm particles and 1*l* of 48.8 g*p*/*l* of acid is shown in Figure 6-9. This curve was predicted using the CSTR model presented in the previous chapter. The significance of Figure 6-9 is that it is the conversion versus dimensionless concentration curve which would have been obtained had a CSTR type leach test been used instead of a lysimeter test. This demonstrates that parameters determined from one type of experiment can be used to predict the performance of other CSTR *or* lysimeter scenarios.

Figure 6-9. Predicted CSTR conversion versus time curve for 0.1*l* of 9.5mm to 13.2mm particles and 1*l* of 48.8 g*p*/*l* of acid.



There are distinct advantages in using a CSTR type experiment to determine suitable model parameters for the heterogenous columnar model. Probably the most important advantage is the short duration of a CSTR experiment. Most heap leaching type models make use of lysimeter data in order to determine the model parameters. (This is exactly the procedure used in the previous section to verify the model.) As previously stated, lysimeter tests typically last for at least a few months and can last up to 3 years. In contrast, a CSTR type leach test takes at most a few days to complete. This means that the potential release of hazardous constituents from a proposed deposit can be determined much more quickly than before. This is a significant advantage in that the mineral processing industry needs to know the likely impacts of certain disposal strategies so that it can determine wether or not the present disposal strategies are adequate. Before, the industry would merely dispose of the wastes and conduct a lysimeter experiment to determine the likely impacts of the disposal strategies. This information would then be used to improve future disposal strategies. In cases where a disposal option was determined to pose a significant risk to the environment, the industry was faced with the expensive rehabilitation or reprocessing of the waste material. The ability

to predict the likely impacts of the disposal strategies *a priori* will help to eliminate the costly rehabilitation or reprocessing procedures.

The financial savings incurred by conducting CSTR type experiments over lysimeter experiments are also significant. CSTR experiments are much cheaper to conduct than lysimeter experiments. This has significance in that existing deposits can be investigated to determine their environmental risks. Previously, suitable lysimeter experiments for all waste deposits were not feasible due to the costs associated with these tests. By making use of the cheaper CSTR tests, samples from these deposits can now be tested to determine their potential to pollute the environment.

6.7 A Summary of the Experimental Data which is Required to Verify the Applicability of the Heterogenous Columnar Model to describe the Leaching of Hazardous Constituents from Waste Deposits.

In order to demonstrate the applicability of the heterogenous columnar model to describe the leaching of hazardous constituents from waste deposits, it needs to be verified against suitable experimental data. Section 6.5 verified that the computer routines were operating correctly by testing them against heap leaching data for copper. Although this shows that the model can be used to accurately predict the leaching of copper from a heap leaching operation, the ability to predict contaminant leaching from a waste deposit still needs to be demonstrated.

This project has not involved any experimental work. Where experimental data has been required, data which has been reported in the open literature has been used. Unfortunately, suitable data is not reported in the literature which can be used to verify the applicability of the heterogenous columnar model to contaminant leaching from waste deposits. This section summarises the type of data which is required and suitable methods to verify the model using this data.

It may be worth pointing out some of the short comings of data reported in the open literature so that these can be avoided when suitable experiments are planned and conducted. Firstly, although there is data on leach experiments for waste particles, these leach experiments have always involved a size distribution of particles. The reason for this practice is probably due to the fact that leach tests have been typically used to determine the 'leaching potential' of wastes. To determine this 'leaching potential', a sample of the waste, which inherently involves a size distribution of particles, is leached and the final leachate concentration is taken as a measure of the 'leaching potential'. These tests are usually termed Toxicity Characteristics Leaching Procedure tests (TCLP tests) [US Government Printing Office; 1988, 1980]. These experiments have not been used to model the effective release of hazardous constituents from waste particles. It has been continuously demonstrated in this thesis that the leaching of hazardous constituents from waste particles is dependant on the particle size. Thus to determine the effective

release of hazardous constituents from waste particles a leach test on o single size class of particles is required.

The second factor is that the heterogenous columnar model describes the *leaching* of hazardous components from wastes. Some hazardous components of wastes are released due to dissolution reactions. It is unlikely that the chemical reaction model used in the heterogenous columnar model can be used model dissolution reactions. The reason for this is due to the fact that dissolution reactions are controlled by solubility constraints which have not been included into the model. Instead the model describes the rate of hazardous constituent release by making use of a kinetic expression corresponding to equation (4-2). Thus care should be taken not to apply the model to instances in which hazardous constituent dissolution is significant.

It goes without saying that a model can only be applied to situations for which it was designed. With this in mind, the heterogenous columnar model has been designed to describe the active *leaching* of hazardous components from granular waste deposits in which there is no external mass transport resistances. (The external mass transport resistances refer to the transport of components from the bulk fluid phase to the fluid/solid interface and vice versa.) Additionally, the particle characteristics as a function of size have been assumed to be fairly uniform. The only exception to this is that the concentration of hazardous constituents on the surfaces of the particles, and thus the total leachable concentration of hazardous constituents, has been allowed to vary as a function of particle size.

There are two stages to verifying the ability of the heterogenous columnar model to describe the leaching of hazardous constituents from waste deposits. The first stage involves determining whether the model can be fitted to lysimeter data and then used to predict other lysimeter operating conditions. This approach is very similar to the strategy adopted by Roman *et al.* [1974] and used in section 6.5. This approach would involve setting up two different lysimeter experiments. These experiments could differ in terms of the column heights used, the fluid flow rates through the columns (which in turn would affect column saturation and the wetting efficiency), the fluid reagent concentrations and the particle size distributions. The results of one of the columns could be used to determine the parameters in the heterogenous columnar model. The model could then be used to predict the results of the second column and the predictions compared to the actual results.

The second stage involves determining whether the heterogenous columnar model parameters could be determined from a suitable CSTR experiment. This would involve conducting a CSTR experiment on a single size class of the waste particles and determining the model parameters as described in section 5.8. These parameters would then be used in the heterogenous columnar model to predict the two column experiments discussed above. Once the model has been verified in this manner it could be used with confidence to predict the leaching behaviour of hazardous waste constituents from waste.

An initial investigation into determining the particle-scale model parameters from a CSTR type experiment on a single size class of waste particles has been conducted. The data used in this analysis is from an experimental program currently being conducted at the University of Cape Town. The experimental conditions are summarised in Table 6-3.

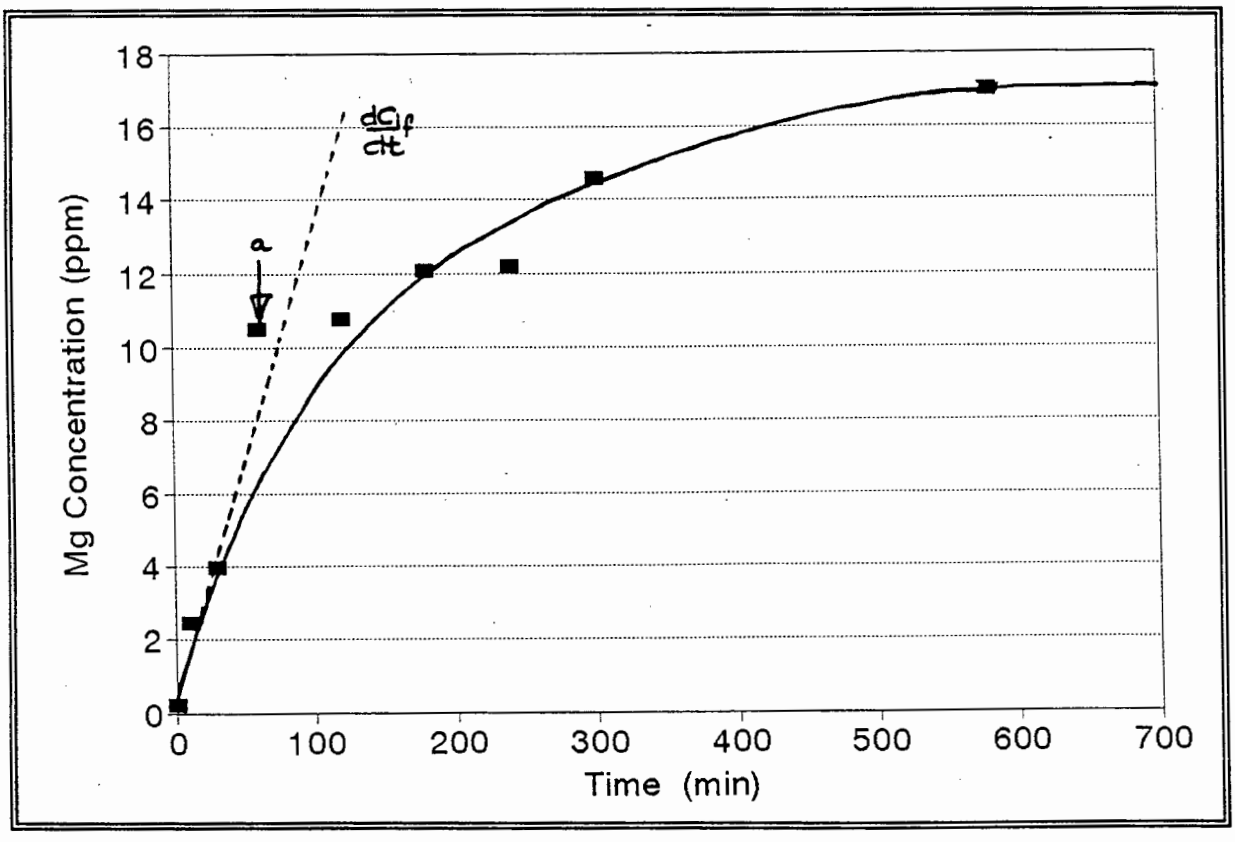
Table 6-3. Summary of the Conditions in the CSTR Test on a Waste Sample.

Waste type.	Stainless steel electric arc furnace dust.
Volume of fluid reagent.	1.6 l
Mass of waste particles.	0.04 kg
Particle Voidage.	1 %
Solid density.	2900 kg/m^3
Average particle size.	3e-6 m

The pH remained fairly constant at a value of 10 during the experiment.

The concentration of dissolved magnesium in the bulk fluid was used as a measure of the leach potential. This is shown as a function of time from one CSTR experiment in Figure 6-10.

Figure 6-10. Concentration of Dissolved Magnesium in the Bulk Fluid as a Function of Time.



The point labelled (a) in the diagram has been taken to be an outlier. The curve drawn through the experimental points is the expected increase in the magnesium concentration in the bulk fluid with time based on the data points. This curve was used to back calculate the total extractable grade, C_{Ei0}, of the magnesium.

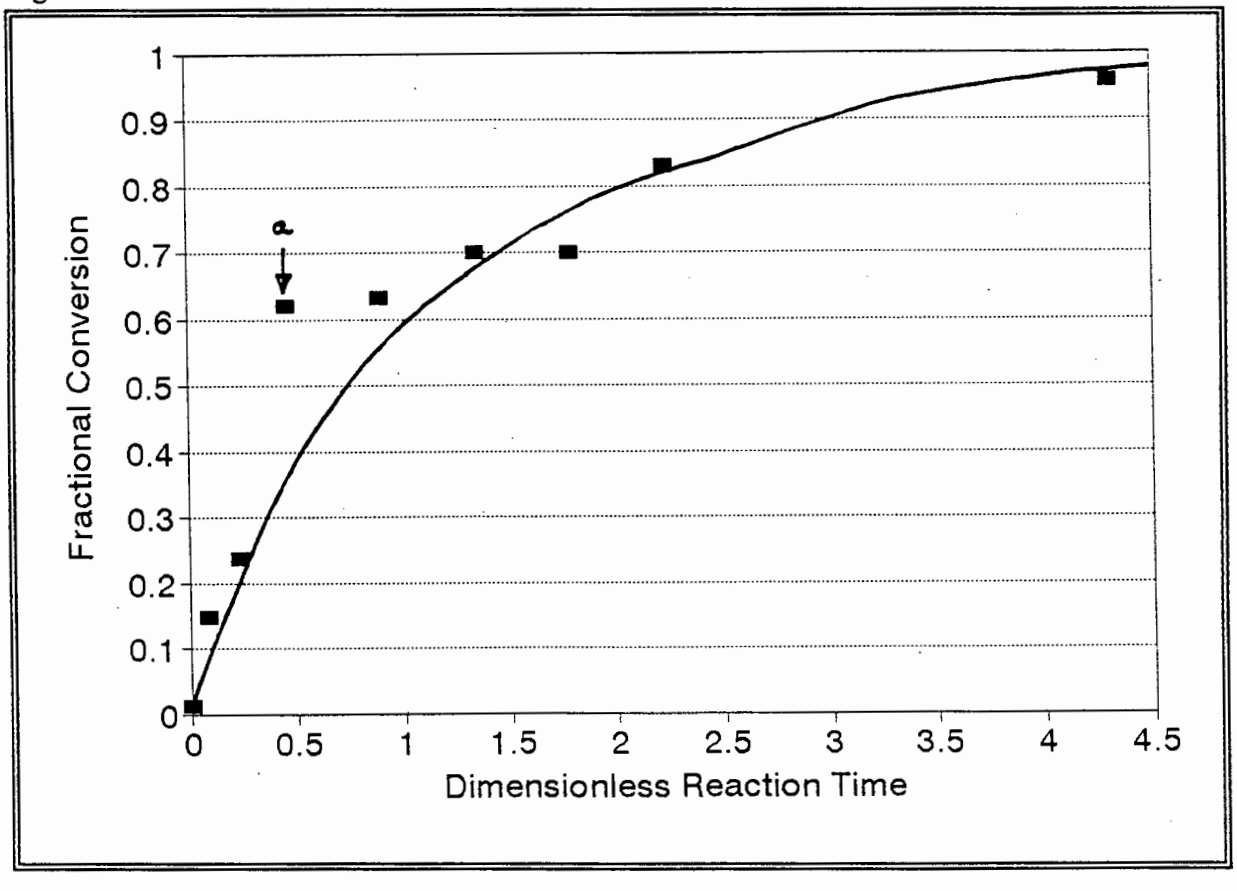
These initial reaction rate for the release of the magnesium has been determined using the initial increase in the bulk fluid concentration of magnesium as follows:

$$k_{pi} C_{pi_0}^{\phi_{pi}} C_{A_0} = -\frac{dC_i}{dt}\Big|_{t=0} = \frac{dC_{i_f}}{dt}\Big|_{t=0} \frac{V_f}{M_s} \qquad (6\text{-}19)$$

where C_{if} is the dissolved fluid phase concentration of solid reactant i;
V_f is the volume of the fluid in the batch test; and;
M_s is the mass of the waste particles in the batch test.

The total extractable grade and the initial reaction rate were used to prepare the fractional conversion versus dimensionless reaction curve as described in section (5-8). This curve is shown in Figure 6-11.

Figure 6-11. Fractional Conversion versus Dimensionless Reaction Time.



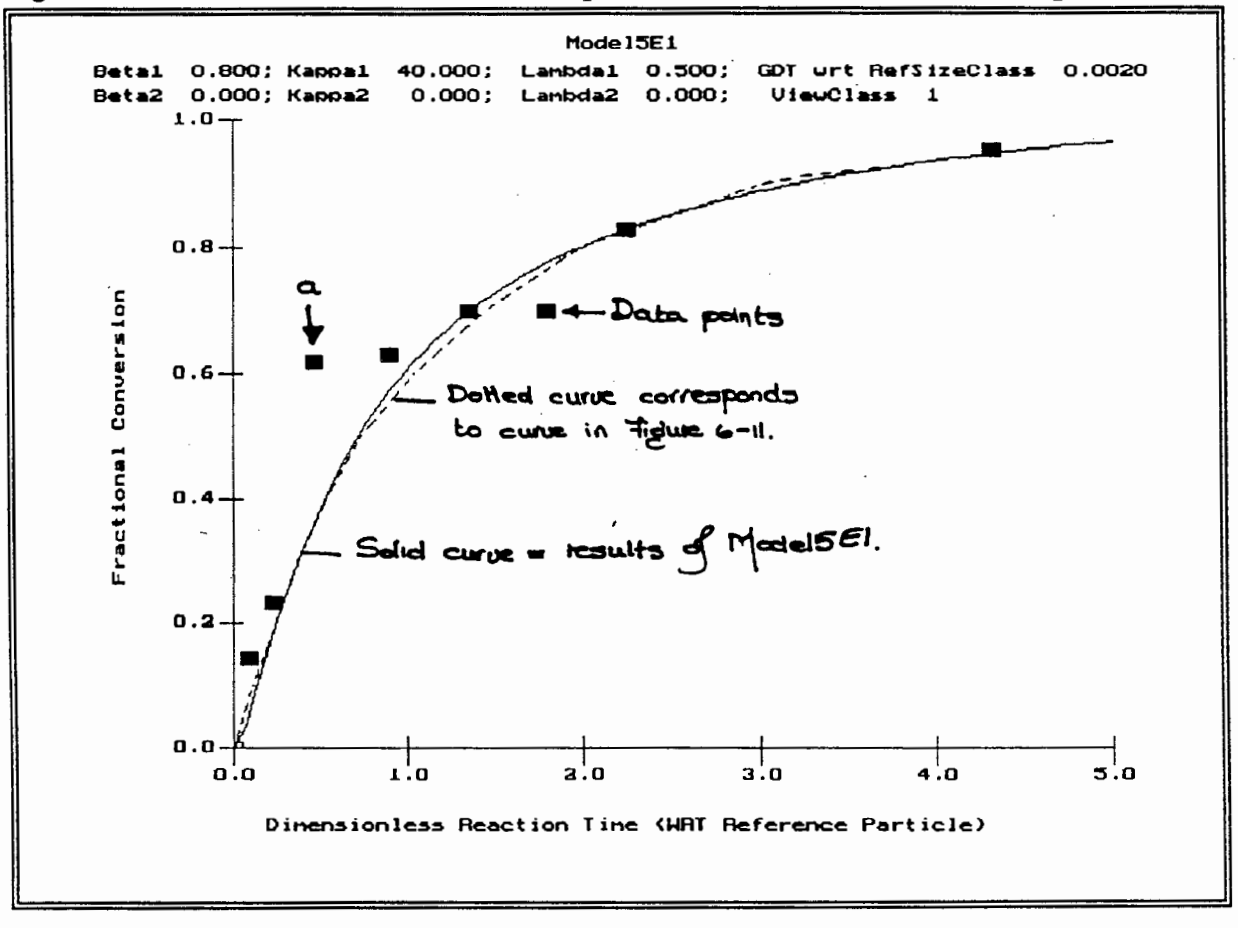
Model5E1 was fitted to Figure 6-11 and the results are shown in Figure 6-12. The dashed curve in this figure corresponds to the curve in Figure 6-11. The solid curve is the fractional conversion calculated using Model5E1. The parameters fitted in the model

were β, which represents the dimensionless reagent strength relative to the concentration of the magnesium in the waste; κ_{pi} - Damköhler II number for the magnesium within the waste particles; and; ς which is the ratio of the surface grade of magnesium to the bulk grade of magnesium. The magnesium on the surface of the particles has also been assumed to react in a similar manner to the magnesium in the bulk of the particle. This is the reason why there is no explicit dependence on the surface kinetics in equation (6-19). The values determined by an initial fit are summarised is Table 6-4. There is excellent agreement between Model5E1 and the CSTR data from a sample of metallurgical waste.

Table 6-4. Values of the Fitted Parameters.

β_1	0.8
κ_{pi}	30
ς	1.0

Figure 6-12. Model5E1 versus CSTR Experimental Data on a Waste Sample.



Chapter 7. Summary of the Applications, Limitations and Extensions of the Heterogenous, Columnar Model.

The previous chapter presented the development of the heterogenous columnar model. Conceptually, the modelling strategy involved the division of the waste deposit into columnar sections and then determining the release of hazardous constituents from a single column. The release of hazardous constituents from a single column is determined by making use of a one-dimensional, non-catalytic, packed-bed type reactor model. The reactor model is derived by applying a fluid reagent mass balance to the column. Within the column, fluid reagent is consumed due to chemical reactions which occur both on the surfaces and within the waste particles. This consumption is modelled using a particle scale, chemical reaction model which also determines the release of hazardous constituents into the bulk fluid surrounding the particles. Once the hazardous constituents are released from the particles, they are assumed to be transported out of the column by the bulk fluid flow through the column.

This chapter summarises the applications, limitations and possible extensions of the heterogenous columnar model. The first section compares this model to the model of Roman *et al.* [1974] and the columnar model of Dixon [1992, 1993]. As previously discussed, although these last two models were developed as precious metal leaching models, several similarities exist between precious metal leaching from ores and the leaching of hazardous constituents from waste particles. Most importantly, this section highlights the advantages of the heterogenous columnar model, which was specifically designed to model the leaching of hazardous constituents from waste deposits, over the heap leaching models.

The next section discusses potential engineering applications of the heterogenous columnar model. In particular, this section summarises the type of information which can be determined using the model and demonstrates how this information can be used to improve waste deposit design and to choose upstream processing options which result in more stable wastes.

The limitations of the heterogenous columnar model, which are largely due to the simplifying assumptions which have been made, are then reviewed and methods to remove these limitations are presented.

Finally, a statement of the significance of this work is made.

7.1 Comparison of the Heterogenous Columnar Model to the Model of Roman *et al.* [1974] and the Columnar Model of Dixon [1992, 1993].

The heterogenous columnar model is comparable to the model presented by Roman *et al.* [1974] and the columnar model presented by Dixon [1992, 1993]. All of these models

approximate deposit or heap performance by determining the performance of a single column packed with waste or ore particles. Further, all three models approximate the fluid flow through the deposit or heap as perfect plug flow and include a saturation index. This parameter is included to account for unsaturated conditions in the waste deposit or ore heap. The heterogenous columnar model includes a global wetting parameter, presented in section 6.3, to accommodate the effects of incomplete particle wetting on the release of hazardous constituents. Since the fluid reagent distribution systems in ore heaps are specifically designed to maximise particle wetting, incomplete particle wetting is not considered by Roman *et al* [1974] or Dixon [1992, 1993].

Table 7.1 summarises the properties of the heterogenous columnar model and the heap leaching models of Roman *et al.* [1974] and Dixon [1992, 1993].

Table 7.1. Summary of the properties of the heterogenous columnar model and the heap leaching models of Roman [1974] and Dixon [1992,1993].

	Heterogenous Columnar Model.	**Heap Leaching Model of Roman *et al.***	**Heap Leaching Model of Dixon.**
Fluid flow model:	Plug flow model. Constant or variable fluid flow into deposit. Includes a saturation index and a global wetting factor.	Plug flow model. Constant fluid flow into heap. Only includes a saturation factor.	Plug flow model. Constant fluid flow into heap. Only includes a saturation factor.
Chemical reaction model:	Particle scale model which includes the effects of intrinsic chemical kinetics, diffusion and hazardous constituent location on contaminant release. Parameter spec. as a function of particle size is particularly suited to waste particles.	Particle scale shrinking core model which assumes that diffusion of the fluid reagent is rate limiting.	Particle scale model which includes the effects of intrinsic chemical kinetics, diffusion and precious metal location on metal release. Parameter spec. as a function of particle size is particularly suited to ore particles.

7.1.1 Comparison to the Model of Roman *et al.* [1974].

The main difference between the heterogenous columnar model and the model of Roman *et al.* [1974] is in the method used to determine the rate of hazardous constituent or precious metal release from the individual particles. Roman *et al.* assumed that the release of the precious metals was dominated by diffusion of the fluid reagent into the ore particles and thus made use of a diffusion controlled, shrinking core reaction model. As previously discussed in section 2.4, this model cannot include the effects of intrinsic chemical kinetics, precious metal location within the ore particle or competing chemical reactions on the rate of precious metal release.

The heterogenous columnar model developed here employs a chemical reaction model which is based on a fluid continuity equation at the particle scale. This model, which was used by Dixon [1992] and summarised in Chapter 4, is used to determine the release of hazardous constituents from waste particles. (The particle scale model of Dixon is distinct from his columnar model.) This approach is capable of including the effects of intrinsic chemical kinetics, intra-particle diffusion, hazardous constituent location within the particle and competing chemical reactions on the rate of release of the hazardous constituents.

The ability of the particle scale chemical reaction model, which is incorporated into the heterogenous columnar model, to include the effects of competing chemical reactions is particularly important when describing release of hazardous constituents from waste particles. The reason for this is that most waste particles consist of two or more solid reactive species present in significant concentrations which compete for fluid reagent. As long as the voidage of the waste particles remains fairly constant, this competition has the effect of reducing the available fluid reagent for each reactive species. Thus the release of an individual species is slowed down for each additional competing reaction which occurs. In contrast, competing reactions which result in an increase in the voidage of the particles may result in increased leaching rates. The reason for this is due to the fact that an increase in the particle voidage will result in an increased effective diffusivity of the fluid reagent into the waste particles. Only the case of constant particle voidage has been considered in this work. The extension of the model to include a dynamic particle voidage is discussed in the next section.

It is important to note that the particle scale chemical reaction model will reflect diffusion control, chemical kinetic control or an intermediate condition depending on the κ_{pi} parameter specification. The Kappa parameter, κ_{pi}, defined in section 4.2, represents the ratio of the chemical reaction rate within a particle to the rate of diffusion of fluid reagent into the particle. As such, the κ_{pi} parameter corresponds to a Damköhler number of the second type [Aris 1975]. A high value for κ_{pi} implies that the reaction is controlled by diffusion of the fluid reagent into the particle. Similarly, a very low value for κ_{pi} implies that the reaction is controlled by the intrinsic chemical kinetics of the reactions. Figure 4-2 showed reagent profiles for several parameter combinations. Note that diffusion controlled reactions result in steep fluid reagent and solid reactant gradients within the particles. In contrast, reactions which are controlled by the intrinsic chemical kinetics exhibit very flat fluid reagent and solid reactant gradients.

7.1.2 Comparison to the Columnar Model of Dixon [1992, 1993].

Dixon [1992, 1993] has also incorporated the particle scale, chemical reaction model into a column type model to predict the performance of heap leaching operations. Probably the most important difference between his model and the heterogenous columnar model developed here is the manner in which the particle scale model parameters have been determined as a function of particle size. In summary, it is desirable to determine or specify parameters for a single size class of particles, termed a reference size class, within the column. The parameters of all the other size classes of particles are then determined from the reference size class parameters using suitable mathematical relationships. The reason why this approach is desirable is that it eliminates the need to define suitable particle scale model parameters for each size class of particles. This greatly reduces the number of parameters required.

Dixon [1992] presented mathematical relationships for the particle scale model parameters as a function of particle size for ore particles. These relationships, discussed in section 5.2.1,were found not to be applicable to waste particles because they required information about precious metal location in the ore particles as a function of particle size. This information was determined from a knowledge of ore preparation. Hazardous constituent location in waste particles is usually highly size specific and a result of the processing operations which generate the waste. For this reason, additional mathematical relationships, which are specifically applicable to the determination of the model parameters for waste particles as a function of particle size, have been proposed in section 5.2.2. It is worth noting that these relationships are more complex than those defined by Dixon, but, under appropriate conditions, will simply to those of Dixon. This implies that the heterogenous columnar model developed in this work can be applied to both the leaching of hazardous constituents from waste deposits as well as to precious metal leaching from ore heaps. These equations have been included in a bulk fluid reagent mass balance. These aspects and an appropriate solution strategy have been discussed in section 5.3 and 5.4 respectively.

The next main difference between the columnar model of Dixon and the heterogenous columnar model is that the new model includes the possibility for a variable fluid reagent flow into the deposit. The inclusion of a variable fluid velocity into the model is important in order to model the effects of periodic rainfall on the release of hazardous constituents from waste deposits. Further, work at the University of Cape Town [Petersen 1994, 1995] has revealed that fluid flow through lysimeter columns can vary significantly during the experiment. Thus, in order to fit the heterogenous columnar model to lysimeter data, it must be capable of including variable fluid reagent flow into the deposit. The strategy adopted to include a variable fluid reagent flow into the heterogenous columnar model is identical to the strategy adopted to include this effect into the macroscopic, lumped parameter model which was discussed in section 3.4. In summary equation (6-9) is redefined in a manner similar to equation (3-29). This equation is then used to determine the appropriate time interval for each successive iteration in time.

Another difference between the models is that Dixon [1993] has included a mass balance equation to include the effects of dissolved species transport within the particles. Due to time limitations, this was not included in the heterogenous columnar model. The inclusion of this mass balance equation into the heterogenous columnar model is discussed in a later section in this chapter. The importance of including this aspect into the model is that it will extend the applicability of the model to cases where dissolved species transport within the waste particles is significant.

7.2 A Summary of the Potential Engineering Applications of the Heterogenous Columnar Model.

The heterogenous, columnar model, described in the preceding sections, has two particular attributes which make it attractive in the engineering design of better waste deposits. The first of these attributes is that it is capable of determining the relative contributions of the intrinsic chemical kinetics and the diffusion of fluid reagent into the particles to the release of hazardous constituents. These rate terms are in turn a function of the hazardous constituent location within the particles, the size distribution of the particles, competing chemical reactions and the bulk fluid reagent flow. The second attribute is that the parameters required for this model can be determined from a simple CSTR type experiment. One particular advantage of being able to determine the model parameters from CSTR type experiments is the short duration of these experiments. This implies that the potential release of hazardous constituents can be determined much more quickly than existing methods which are fitted to lysimeter data. This ability will enable the minerals processing industry to determine *a priori* the suitability of various wastes for disposal in waste deposits. Another distinct advantage of using CSTR tests to determine the model parameters is that CSTR tests are much cheaper to conduct than lysimeter experiments. This implies that more wastes can be tested prior to disposal and that samples from existing waste deposits, which could not be tested previously due to the expense of lysimeter experiments, can now be tested.

7.2.1 Improved Deposit Design based on Results from the Heterogenous Columnar Model.

Once the particle scale model parameters for a given waste stream have been determined, the heterogenous columnar model can be used to investigate the behaviour of a typical deposit which would contain this material. The results of the heterogenous columnar model will not only give an indication of the release of hazardous constituents from the waste deposit but can also be used to improve the disposal strategy to minimise the release of hazardous constituents.

Deposits Which React in a Zone-Wise, Fluid Reagent Limited Manner.

As an example, consider the case where the results of the heterogenous columnar model indicate that the *deposit* is reacting in a zone-wise,fluid reagent limited manner. The rate of the chemical reactions occurring in the particles of such deposits is relatively fast compared to the rate of fluid flow through the deposit. Even if the breakthrough curves determined by the heterogenous columnar model are acceptable in terms of the concentrations of the hazardous constituents, deposits of this nature present an environmental risk. The reason for this is that the release of hazardous constituents is only limited by the amount of fluid flowing through the deposit. Should this increase, so too would the rate of release of hazardous constituents. It is a high priority therefore to keep deposits which react in a fluid reagent limited manner dry. This could be achieved by using suitable liners to prevent rain or ground water from percolating through the deposit.

As an alternative to keeping the deposit dry to limit the release of hazardous constituents, pretreatment options for the waste could be investigated which would slow down the rate of release of hazardous constituents at the particle level. One likely factor which would result in relatively fast chemical reactions would be if a significant fraction of the hazardous constituents are concentrated onto the surface of the particles. (These reactions would be relatively fast because they do not experience any fluid reagent diffusional resistances.) If significant concentrations of hazardous constituents are known to exist on the surface of the particles, their release could be reduced by removing these components in a pretreatment process. Surface hazardous components could be removed from the particles by making use of an active leach procedure. It may be possible to purify the leachate generated by such a process and return the 'contaminants', which are often heavy metals, to the process which generated the waste. If it is not feasible or economical to pretreat the particles using a leach procedure then alternative methods to stabilise the waste need to be investigated. One alternative method would be to agglomerate the particles into larger particles using suitable binding agents. In this process, much of the external surface area of the original particles would be incorporated into the bulk of the larger agglomerated particle. Before the fluid reagent could react with the hazardous constituents it would need to diffuse through the agglomerated particle to the location of the hazardous constituents. Binding agents usually exhibit a buffering capacity with respect to the fluid reagent. This too, would slow down the release of the hazardous constituents from the particles due to the competition between the binding agent and the hazardous constituents for the fluid reagent. Suitable particle scale model parameters can be determined for such agglomerated particles using a CSTR experiment. These parameters could then be used to determine whether the agglomeration would improve the overall deposit performance.

In the case where the release of hazardous constituents from particles is fast and where there is not a significant concentration of hazardous components on the surfaces of the particles, methods to retard the effective chemical reaction rates need to be investigated. The first way in which this can be achieved would be to enhance the diffusional resistances which limit the fluid transport into the particle. As before, this can be achieved by agglomerating the particles using suitable binding agents. Alternatively, attempts can be made to slow down the intrinsic kinetics within the particle by

chemically altering the speciation of the hazardous constituents. Usually little is known about the complex chemistry of waste particles and any attempts to alter the intrinsic chemical kinetics will most probably require extensive experimentation. In contrast, note that the only instance in which agglomeration will not result in reduced effective chemical reaction rates will be when the intrinsic chemical kinetics are many orders of magnitude slower than the rate of fluid reagent diffusion into the particle. Such a situation will be easy to identify because the particles will react in a totally homogenous manner. (This situation corresponds to a very low Kappa, κ_{pi}, parameter.)

Deposits Which React in a Homogenous Manner.

In contrast to deposits which react in a zone-wise, fluid reagent limited manner, some deposits react in a homogenous manner. The release of hazardous constituents from these deposits is not limited by the flow of fluid through the deposit but rather by the release of hazardous constituents from the individual waste particles. These deposits present little or no environmental risk if the results of the heterogenous, columnar model indicate that the release of hazardous constituents is acceptable. The reason for this is that the individual particles within the deposit, release hazardous constituents at a sufficiently slow rate which the natural environment can assimilate. An increase in the fluid reagent flow through the deposit will not increase the rate of hazardous constituent release.

Should the results of the heterogenous columnar model indicate that the release of hazardous constituents from the deposit is unacceptable, then methods to slow down the effective chemical reaction rates similar to those discussed in the previous section need to be investigated.

7.2.2 Using the Heterogenous Columnar Model to Choose Upstream Processes which would Result in More Stable Wastes.

The properties of a waste stream are dictated by the upstream processes from which it arose. These properties include the chemical composition of the waste stream, the location of the hazardous constituents within individual particles and the size distribution of the particles within the waste stream.

Using the heterogenous columnar model, the particle characteristics which contribute most to the release of hazardous constituents from waste particles can be identified. This information can then be used to identify the upstream processes which result in the waste particles exhibiting the particular characteristic. Alternative processing options can then be investigated which would result in more stable wastes.

As an example, should the results of the heterogenous columnar model indicate that it is mainly the smaller particles within the size distribution of waste particles which are responsible for the release of hazardous constituents (due to smaller diffusional resistances in small particles), then the upstream processes which produce these particles should be identified and investigated. In this case, alternative processes which still

achieve the desired process goal but which do not produce the small particles would be preferred.

Equally, the heterogenous columnar model could indicate that significant concentrations of hazardous constituents on the surfaces of particles are responsible for the release of contaminants. In this case, process options which aim at recovering the 'contaminants' from the wastes should be investigated.

7.2.3 Using the Heterogenous Columnar Model to Asses the Risks and Liabilities Associated with Existing Waste Deposits.

South Africa is not unique in the fact that it has many waste deposits which could pose a significant environmental hazard due to the generation and release of leachate. For various reasons, which include financial constraints of conducting lysimeter experiments and ineffective legislation with respect to the disposal of industrial wastes, the risks associated with many of these deposits is unknown. The heterogenous columnar model can be used to estimate the potential leachate generation of these deposits and thus determine the risks associated with them. This would be achieved by conducting a CSTR test on a single size class of the material in the deposit being examined. As before, this would be used to determine appropriate model parameters for the heterogenous columnar model which would then be used to predict the deposit performance. This information could then be used to estimate the risks associated with these deposits and to identify which of the existing deposits require remediation.

7.3 Limitations and Possible Extensions of the Heterogenous Columnar Model.

The heterogenous columnar model has been presented as a suitable model to describe the release of hazardous constituents from waste deposits. The applicability of this model can be extended by beginning to remove the simplifying assumptions which were made during the model development. This section summarises the limitations of the model and suggests methods to address these limitations.

7.3.1 Incorporation of External Mass Transfer Resistances into the Model.

The heterogenous columnar model assumes that the mass transfer of the fluid reagent between the bulk fluid, which is the fluid between the particles, and the surfaces of the particles is negligible. Such an assumption is only valid when the effective reaction rate of hazardous constituent release is slow compared to the supply of fluid reagent to the particle surface. In cases where this assumption does not hold, external mass transfer resistances need to be accounted for explicitly.

Equation (5-1) is a mass balance equation which relates the consumption of the fluid

reagent within the particles to the overall bulk fluid reagent concentration between the particles. This equation is valid only when external mass transfer resistances are negligible and needs to be modified when external mass transfer resistances are important. The modification involves relating the consumption of the fluid reagent within the particles to the rate of fluid reagent transfer to the particles. Equation (5-1) as well as the new equation are summarised below:

$$\frac{V_{Part.}\rho_0(1-\epsilon_0)}{b_i}\sum_{i=1}^{N}\frac{dC_{s_i}}{dt}-D_e\nabla C_A|_R\frac{3V_{Part.}}{R}=V_{Liq.}\frac{dC_{A_{Bulk}}}{dt} \qquad (5\text{-}1 \text{ or } 7\text{-}1)$$

where $V_{Part.}$ is the total volume of the particles; and;
$V_{Liq.}$ is the total volume of fluid reactant.

This equation is equivalent to:

$$\frac{V_{Part.}\rho_0(1-\epsilon_0)}{b_i}\sum_{i=1}^{N}\frac{3k_{si}C_{si}^{\phi_{si}}C_A}{R\rho_0(1-\epsilon_0)}-D_e\nabla C_A|_R\frac{3V_{Part.}}{R}=V_{Liq.}\frac{dC_{A_{Bulk}}}{dt} \qquad (7\text{-}2)$$

The new equation is:

$$\frac{V_{Part.}\rho_0(1-\epsilon_0)}{b_i}\sum_{i=1}^{N}\frac{3k_{si}C_{si}^{\phi_{si}}C_A}{R\rho_0(1-\epsilon_0)}-D_e\nabla C_A|_R\frac{3V_{Part.}}{R}=k_m a_m(C_{A_{Bulk}}-C_{A_{Surface}}) \qquad (7\text{-}3)$$

where k_m is the mass transfer coefficient, determined using correlations for Sherwood numbers or j_D factors;
a_m is the total external area of the particles;
$C_{A,Bulk}$ is the bulk fluid reagent concentration; and;
$C_{A,Surface}$ is the concentration of the fluid reagent on the particle surface.

Suitable correlations are required to determine the external mass transfer rate, k_m, before the heterogenous columnar model can be applied to cases where external mass transfer resistances are important. Several correlations are presented in the literature [Agarwal 1988; Kawase and Ulbrecht 1985; Wakoa and Funazkri 1978; Nelson and Galloway 1975; Mochizuki and Matsui 1973; Calderbank 1967; Wilson and Geankoplis 1966; Rowe and Claxton 1965; Pfeffer 1964; Ranz 1952; Smith 1981] for Sherwood numbers and j_D factors as a function of Reynolds and Schmidt numbers. The Sherwood number is a ratio of the rate of convective mass transport to the rate of molecular mass transport at the particle surface and can be used to determine the external mass transfer

coefficient. Similarly j_D factors can be used to determine external mass transfer coefficients.

It is important to note that the external mass transfer rates are normally defined in terms of average mass transfer coefficients [Smith 1981]. Thus a single value of the mass transfer coefficient can be used to describe the rates of transfer between the bulk fluid phase and the particle surface. The error introduced in using an average coefficient is not as serious as might be expected, since the correlations for the mass transfer coefficient, k_m, are based on experimental data for packed beds of particles [Smith 1981].

7.3.2 Inclusion of Intra-Particle Dissolved Species Transport Resistances into the Model.

The intra-particle dissolved species transport resistances have been assumed to be negligible compared to the diffusion resistances of the fluid reactant into the particles and the chemical reaction rates within the particles. As discussed in section 7.1.2,Dixon [1993] has included these effects into his columnar model. The strategy adopted was to apply a mass balance equation for each dissolved species at the particle level. The equations derived are very similar to equation (4-6) which represented a mass balance for the fluid reagent at the particle level. Equation (4-6), and the mass balances for the dissolved species are summarised as:

$$D_e \left[\frac{\partial^2 C_A}{\partial r^2} + \frac{2}{r}\frac{\partial C_A}{\partial r}\right] - \rho_0 (1-\epsilon_0) \sum_{i=1}^{n} \frac{k_{pi} C_{pi}^{\phi_{pi}} C_A}{b_i} = \epsilon_0 \frac{\partial C_A}{\partial t} \qquad (4\text{-}6 \text{ or } 7\text{-}4)$$

$$D_i \left[\frac{\partial^2 C_i}{\partial r^2} + \frac{2}{r}\frac{\partial C_i}{\partial r}\right] + \rho_0 (1-\epsilon_0) k_{pi} C_{pi}^{\phi_{pi}} C_A = \epsilon_0 \frac{\partial C_i}{\partial t} \qquad (7\text{-}5)$$

where C_i dissolved hazardous constituent concentration of species i.

Equation (7-5) can be used to determine the dissolved concentration of hazardous constituents within the particle and at the particle surface. Should external mass transport of the dissolved hazardous constituents from the particle to the bulk fluid be significant, a suitable mass transfer correlation, similar to the correlations described in section 7.3.1, would be required to determine the bulk concentration of dissolved hazardous constituents. If external mass transport resistances are negligible, then a simple mass balance can be used to determine the bulk concentration of dissolved hazardous constituents.

7.3.3 Inclusion of Matrix Dissolution and Hazardous Constituent Re-Precipitation in the Heterogenous Columnar Model.

Neither matrix dissolution or the re-precipitation of dissolved hazardous constituents have been incorporated into the heterogenous columnar model. Matrix dissolution refers to the process of the waste particles becoming more porous due to significant quantities of the particle being dissolved during the life span of a deposit. It is unlikely that the effects of matrix dissolution will be easily incorporated into the particle scale chemical reaction model. The reason for this is that the particle scale model makes use of an effective diffusivity D_e to determine the diffusion of fluid reagent into the particle. As matrix dissolution takes place, the voidage of the particles will increase. A larger voidage within the particles will substantially effect the rate of diffusion of fluid reagent into the particles. Thus before the effects of matrix dissolution can be incorporated in the particle scale model, a suitable model for the effective diffusivity, D_e, would need to be determined. It is worth noting that simple models for the effective diffusivity such as the parallel pore more model, described by Smith [1981] and summarised in equation (7-6), are inappropriate. The reason for this is that the tortuosity factor used in the models is also a function of matrix dissolution and would be an unknown in the model.

$$D_e = \frac{\epsilon D}{\delta} \qquad (7-6)$$

where D_e effective diffusivity;
D actual diffusivity;
ϵ particle voidage; and;
δ tortuosity factor.

Once the hazardous constituents have been dissolved there is a good possibility for them to reprecipitate either within the particle or onto the surfaces of surrounding the particles. These aspects need to eventually be incorporated into the heterogenous columnar model.

7.3.4 Inclusion of More Realistic Hydrodynamic Flow Models into the Heterogenous Columnar Model.

The only fluid flow pattern through the deposit which was considered in the heterogenous columnar model was perfect plug flow. As discussed in section 2.1.2,the hydrodynamic flow patterns within waste deposits are often far more complex than a simple plug flow pattern. These irregular flow patterns need to be incorporated into the heterogenous columnar model.

The simplest manner in which 'non-ideal' flow patterns can be incorporated into the heterogenous columnar model would be to make use of a one parameter fluid dispersion

model. In this approach any back mixing and short circuiting of fluid reagent is considered as fluid dispersion. A suitable one parameter fluid dispersion model which is applicable to columnar reactors is the 'tanks-in-series' approach described by Levenspiel [1972]. In this approach, the reaction in the column is approximated by a number a discrete, equi-sized continuously stirred tank reactors in series. The single parameter in the approach is the number of tanks-in-series which are required to approximate the fluid flow behaviour within the columnar reactor. A tracer study which yields the residence time distribution of the column can be used to determine this parameter. This technique is described by Levenspiel [1972].

This approach would be easy to implement in the heterogenous columnar model. In section 6.1 a modelling strategy for the heterogenous columnar model which was based on heap leaching models was described. In summary the column was assumed to consist of a number of discrete disks. Within each disk the fluid reagent concentration was assumed to be uniform. It was noted in section 6.1 that the column would need to be divided into a sufficient number of disks in order to ensure that this assumption would be valid. In effect, this corresponds to the case of '$N \to \infty$' in the tanks-in-series approach where N represents the number of tanks in the chain. In reality, the number of 'tanks' or disks required to model perfect plug flow is finite and is determined as the number of disks beyond which model accuracy does not improve. This number of disks represents the minimum number of disks which are required within the column to adequately simulate perfect plug flow in the column. In effect, the 'inaccuracies' introduced when fewer disks are used can be interpreted as fluid dispersion and can be taken as an indication of the performance of columns which exhibit irregular flow patterns. The performance of a particular column which exhibits irregular flow patterns can be approximated by conducting a tracer experiment on the column and using the residence time distribution obtained to determine a suitable number of tanks, N, which will approximate the flow patterns within that column. By using the heterogenous columnar model with N disks, the performance of the column can then be estimated.

More complex multi-parameter dispersion models will be required as the flow patterns within a deposit become more irregular. These models consider the column to consist of several hydrodynamically distinct regions which can each be described by plug flow, dispersed plug flow and mixed flow models. These models were discussed briefly in section 2.5.

To improve the ability of the heterogenous columnar model to predict the release of hazardous constituents under truly unsaturated conditions, the ground water flow equations used by Demetracopoulos *et al.* [1986] can be incorporated into the model. These equations, presented in section 2.3.2,determine the saturation and fluid velocity as a function of position within the column and time. These factors are included in the heterogenous columnar model by updating equation (6-4). The updated equation can be summarised as:

$$\epsilon_{Col}\ Col_{\%\ Sat}\ \frac{\partial C_{A_{Col}}}{\partial t} = -u\frac{\partial C_{A_{Col}}}{\partial z} - C_{A_{Col}}\frac{\partial u}{\partial z} + \sum_{i=1}^{n} r_{a_i} \qquad (7-7)$$

where the $Col_{\%\ Sat}$ is now a function of position within the column and time. ($Col_{\%\ Sat}$ corresponds to θ/ϵ_{Col}, where θ is the moisture content used in equation (2-2) and (2-3).)

Note that the second term on the right hand side of the equation is a direct result of the divergence of the velocity field within a volume element in the column no longer being equal to zero. (Compare equation 3-1, 3-4 and 3-5.)

7.4 Statement of the Significance of the Work Presented in this Thesis.

Waste deposits which contain wastes from the minerals processing industry pose environmental hazards due to the possibility of leachate generation and the subsequent release of this leachate into the environment. The impacts of waste deposits have been traditionally investigated using lysimeter experiments. These experiments are both very time consuming and costly. This and other factors, such as ineffective legislation with respect to waste disposal strategies, have resulted in a limited study of the environmental impacts of disposing of mineral wastes in waste deposits.

Due to increased environmental awareness, the minerals processing industry is being encouraged to investigate the environmental impacts of its disposal strategies. As such, suitable modelling strategies which could predict deposit performance would be most useful. The reason for this is that these models could be used to investigate the potential for waste deposits to pollute the environment. If sufficiently detailed, these models could also be used to identify the major contributing factors to the release of hazardous constituents from waste deposits. This information could then be used to engineer better waste deposits in future. Models could also be used to investigate modifications to the upstream processes with the aim of producing more stable wastes. Equally as important, the risks associated with the many existing waste deposits could be determined and remedial action taken before a catastrophic situation arises.

The work presented in this thesis is an investigation into identifying suitable modelling strategies to describe the release of hazardous constituents from waste deposits. The first model investigated was a macroscopic, lumped parameter model. This model was investigated due to its inherent simplicity and its ability to characterise a deposit as a reacting entity. The investigation into this model revealed that it had limited applicability to identify the main factors which result in the release of hazardous constituents from waste deposits. This implies that this model has limited applicability in the design of new deposits. Further, the parameters required for the macroscopic lumped parameter model are determined from fitting the model to lysimeter data. This too is a limitation due to the fact that lysimeter experiments take a minimum of a few months and can take up to three years to complete. For these reasons a second model, termed the heterogenous columnar model was investigated.

The heterogenous columnar model describes the release of hazardous constituents from the individual particles within a waste deposit and relates this release to the overall

deposit performance. This model has been developed to apply specifically to the release of hazardous constituents from waste deposits. The heterogenous columnar model has the capability of identifying the main factors which can be associated with the release of hazardous constituents. As an example, the heterogenous columnar model can be used to determine the relative contributions of the intrinsic chemical kinetics of the waste particle and the diffusion of fluid reagent into the particle to the release of hazardous constituents. Equally, the significance of the bulk fluid flow through the deposit compared to the effective rate of hazardous constituent release can be determined. This information could be used in the engineering design of waste deposits and was discussed in section 7.2.1. Further, this information could be used to investigate upstream processing options which would result in more stable wastes. This was discussed in section 7.2.2.

The heterogenous columnar model has the distinct advantage that the parameters which it requires can be determined from CSTR type experiments. Probably the most important advantage of being able to determine the heterogenous columnar model parameters from CSTR experiments is that CSTR type experiments are considerably quicker than lysimeter experiments. This means that the large delay times associated with the traditional lysimeter experiments can be eliminated. Using the heterogenous columnar model, the minerals processing industry would be able to determine the suitability of a particular waste for disposal in a waste deposit. The second significant advantage of being able to determine the model parameters from a CSTR type experiment is the relatively low costs associated with these experiments when compared to lysimeter experiments.

The heterogenous columnar model can also be used to investigate the risks associated with existing waste deposits. This would be achieved by collecting samples from these deposits and conducting a CSTR type experiment on a single size class of the particles. The data from this experiment could be used to determine the heterogenous columnar parameters as discussed in section 5.8 and 6.6. The heterogenous columnar model could then be used to estimate the release of hazardous constituents from these deposits. This information could be used to asses the risks with these deposits and to identify which deposits require remedial attention in order to prevent pollution of the environment.

This work has been a first attempt to identify or develop suitable modelling strategies to determine the release of hazardous constituents from waste deposits. The heterogenous columnar model has been identified as a possible candidate for this purpose. This model has been applied to the leaching of precious metals from ore particles and the capability of the model to describe and predict the leaching behaviour is encouraging. The verification of the model against data from waste deposits or lysimeter experiments still needs to be completed. The type of data required for this purpose has been summarised in section 6.7. Once the heterogenous columnar model has been verified against waste leaching data it may start to provide the minerals processing industry with the information which it so desperately requires in order to dispose of their wastes in a responsible manner which will not pose a threat to the environment.

References.

Agarwal, P.K. "*Transport Phenomena in Multi-particle Systems-II. Particle Fluid Heat and Mass Transfer.*", Chemical Engineering Science 43(9):2501-2510, 1988.

Aris, R. The Mathematical Theory of diffusion and reaction in permeable catalysts - The theory of the steady state., Oxford, Clarendon Press, 1975.

Batchelor, B. "*Leach Models:Theory and Application.*", Journal of Hazardous Materials. 24:255-266, 1990.

Bishop, P.L. "*Prediction of Heavy Metal Leaching Rates from Stabilised/Solidified Hazardous Wastes.*" In: Proceedings of 18th Mid-Atlantic Industrial Waste Conference. edited by Boardman, G.D.Lancaster:Technomic,p.236-252, 1986.

Braun, R.L., Lewis,A.E. and Wadsworth, M.E. "*In-Place Leaching of Primary Sulphide Ores:Laboratory LEaching Data and Kinetics Model.*", Metallurigal Transactions, 5: 1717-1726, 1974.

Cheng, K. and Bishop, P. "*Developing a Kinetic Leaching Model for Solidified/Stabilized Hazardous Wastes.*", Journal of Hazardous Materials 24:213-224, 1990.

Clift R. and Doig A. "A Life-Cycle View of Waste Management", Centre for Environmental Strategy, University of Surrey, United Kingdom.

Columbo A.J., Baldi G., and Sicardi S. "*Solid-Liquid Contacting Effectiveness in Trickle Bed Reactors*", Chemical Engineering Science, 31:1101-1108, 1976.

Cornish, A.R.H. "*Note on minimum possible rate of heat transfer from a sphere when other spheres are adjacent to it.*", Transactions of the Institution of Chemical Engineers 43:332-333, 1965.

Cote, P.H. and Bridle, T.R. "*Long-Term Leaching Scenarios for Cement-Based Waste Forms.*", Waste Management and Research. 5:55-66, 1987.

Cote, P.L. and Constable, T.W. "*An evaluation of cement-based waste forms using the results of approximately two years of dynamic leaching.*", Nuclear and Chemical Waste Management. 7:129-139, 1987.

Crank, J. The Mathematics of Diffusion. Oxford, Clarendon Press, 1975.

Demetracopoulos, A.C., Sehayek, L., and Erdogan, H. "*Modelling Leachate Production From Municipal Landfills.*", Journal of Environmental Engineering 112(5):849-866, 1986.

Dixon, D.G. Predicting the Kinetics of Heap Leaching with Unsteady-State Models, PhD Thesis, University of Nevada - Reno, 1992

Dixon, D.G. and Hendrix, J.L. "*Theoretical Basis for Variable Order Assumption in the Kinetics of Leaching of Discrete Grains.*" AICHE 39(5), 1993.

Dixon, D.G. and Hendrix, J.L. "*A General Model for Leaching of One or More Solid Reactants from Porous Ore Particles*", Metallurgical Transactions B, 24B:157-169, 1993.

Dixon, D.G and Hendrix, J.L. "*A Mathematical Model for Heap Leaching of One or More Solid Reactants from Porous Ore Pellets.*", Metallurgical Transactions B, 24B:1087-1102, 1987.

Dreisinger, D.B., Peters, E., and Morgan, G. "*The Hydrometallurgical treatment of carbon steel electric arc furnace dusts by the UBC-Chaparral process.*" Hydrometallurgy 25:137-152, 1990.

Fourie, A. "*Predicting Contaminant Migration in Unsaturated Soil: Some recent Advances.*", 1995. (University of the Witwatersrand.)

Freeze R.A. and Cherry J.A. Groundwater. Englewood Cliffs, New York, Prentice Hall, 1979.

Funk, G.A., Harold, M.P., and Ng, K.M. "*A Novel Model for Reaction in Trickle Beds with Flow Maldistribution.*", Industrial and Engineering Chemistry Research 29:738-748, 1990.
Gianetto, A. and Specchia, V. "*Trickle-Bed Reactors: State of Art and Perspectives.*", Chemical Engineering Science 47(13/14):3197-3213, 1992.

Godbee, H.W., Compere, E.L., Joy, D.S., Kibbey, A.H., Moore, J.G., and Nestor, C.W. "*Application of Mass Transport Theory to the Leaching of Radionuclides from Waste Solids.*", Nuclear and Chemical Waste Management 1:29-35, 1980.

Hanratty, P.J. and Dudokovic, M.P. "*Detection of Flow Maldistribution in Packed Beds via Tracers.*", AICHE 36(1):127-131, 1990.

Kawase, Y. and Ulbrecht, J.J. "*A New Approach for Heat and Mass Transfer in Granular Beds Based on the Capillary Model.*", Industrial Engineering Chemistry Fundamentals 24:115-116, 1985.

Knox R.C., Sabatini D.A. and Canter L.W Subsurface Transport and Fate Processes. USA, Lewis Publishers, 1993.

Korfiatis, G.P. and Dermetracopoulos A.C. "*Moisture Transport in a Solid Waste Column*", Journal of Environmental Engineering, 110(4):780-798, 1984.

Levenspiel, O., Chemical Reaction Engineering, John Wiley and Sons, 1972

Lutran, P.G.,Ng, K.M.,and Delikat, E.P. "*Liquid Distribution in Trickle-Beds. An Experimental Study Using Computer Aided Tomography.*" Industrial and Engineering Chemistry Research 30:1270-1280, 1991.

Mills, P.L., Dudukovic, M.P. "*Evaluation of Liquid-Solid Contacting in Trickle-Bed Reactors by Tracer Methods.*", AICHE 27(6):893-904, 1981.

Nelson, P.A. and Gallaway, T.R. "*Particle-to-Fluid Heat and Mass Transfer in Dense Systems of Fine Particles.*", Chemical Engineering Science 30:1-6, 1975.

Ng, K.M. "*A model for Flow Regime Transitions in Cocurrent Down-Flow Trickle-Bed Reactors.*", AICHE 32(1):115-117, 1986.

Ng, K.M. and Chu, C.F. "*Trickle Bed Reactors.*", Chemical Engineering Progress. 55-63,November 1987.

Petersen J., Personal Communication, University of Cape Town, 1994, 1995.

Pfeffer, R. "*Heat and Mass Transport in Multiparticle Systems.*", Industrial and Engineering Chemistry: Fundamentals 3(4):380-383, 1964.

Ramachandran, P.A., Dudukovic, M.P., and Mills, P.L. "*A New Model for Assessment of External Liquid-Solid Contacting in Trickle-Bed Reactors from Tracer Response Measurements.*", Chemical Engineering Science 41(4):855-860, 1986.

Roman, R.J. "*The Limitations of Laboratory Testing and Evaluation of Dump and In Situ Leaching.*", In Situ 1(4):305-324, 1977.

Roman, R.J., Benner, B.R., and Becker, G.W. "*Diffusion Model for Heap Leaching and its Applications to Scale-up.*",Society of Mining Engineers, AIME, Transactions 256:247-249, 1974.

Rowe, P.N. and Claxton, K.T. Transactions of the Institution of Chemical Engineers. 43, 1965.

Rowe, R.K. and Booker J.R. "*Contaminant Migration through Fractured Till into an Underlying Aquifer*", Canadian Geotechnical Journal, 27:484-495, 1990.

Rowe, R.K. and Booker J.R. "*Pollutant Migration Through Liner Underlain by Fractured Soil*", Journal of Geotechnical Engineering, 117(12):1902-1919, 1991.

Rowe, R.K. and Booker J.R. "*1-D Pollutant Migration in Soils of Finite Depth*", Journal of Geotechnical Engineering, 111(4):479-499, 1985.

Rowe, R.K. and Booker J.R. "*Two-dimensional Pollutant Migration in Soils of Finite Depth*", Canadian Geothechnical Journal, 22:429-436, 1985.

Schwartz, J.G., Weger, E., and Dudukovic, M.P. "*A New Tracer Method for Dtermination of Liquid-Solid Contacting Efficiency in Trickle-Bed Reactors.*", AICHE 22(5):894-904, 1976.

Sicardi, S., Baldi, G., Gianetto, A., and Specchia, V. "*Catalyst Areas Wetted By Flowing and Semistagnant Liquid in Trickle-Bed Reactors.*", Chemical Engineering Science 35:67-73, 1980.

Shafer J.L., White M.L. and Caenepeel C.L. "*Application of the Shrinking Core Model for Copper Oxide Leaching*", Society of Mining Mining Engineers, 165-171, February 1979.

Shaw E.M., Hydrology in Practice, Chapman and Hall, London, 1994.

Sicardi, S., Baldi G., Gianetto A. and Specchia V. "*Catalyst Areas Wetted By Flowing and Semistagnant Liquid in Trickle-Bed Reactors*", Chemical Engineering Science, 35:67-73, 1980.

Smith, G.D. Numerical Solution of Partial Differential Equations: Finite Difference Methods, Clarendon Press U.S.A. 1985.

Smith J.M., Chemical Engineering Kinetics, McGraw-Hill, Singapore, 1981.

Sudicky E.A., Cherry J.A. and Frind E.O. "*Migration of Contaminants in Groundwater at a Landfill: A Case Study*", Journal of Hydrology, 63:81-108, 1983.

Van Craen M.J., Denoyer E.A., Natush D.F.S. and Adams F. "*Surface Enrichment of Trace Elements in Electric Steel Furnace Dust*", Environmental Science and Technology, 17(7):435-439, 1983.

Vogt, M. "*A Vectorized Multicomponent Transport Reaction Model: Theory and Application.*", Computer Methods in Water Resources II. Proceedings of the Second International Coference, held in Marrakesh, Morocco, 20-22 February 1991. Computational Mechanics Publications. Vol 1:173-184, 1991.

Von Blottnitz H.B., Personal Communication, University of Cape Town, 1994.

Wakao, N. and Funaztri, T. "*Effect of Fluid Dispersion Coefficients on Particle-to-Fluid Mass Transfer Coefficients in Packed Beds.*", Chemical Engineering Science 33:1375-1384, 1978.

Welty J.R., Wicks C.E. amd Wilson R.E., Fundamentals of Momentum, Heat and Mass Transfer, John Wiley and Sons, Singapore, 1984.

Wilson E.J. and Geankoplis C.J. "*Liquid Mass Transfer at Very Low Reynolds Numbers in Packed Beds*", Industrial and Engineering Chemistry Fundamentals, 5(1):9-14, 1966.

Zimmerman, S.P., Chu, C.F., and Ng, K.M. "*Axial and Radial Dispersion in Trickle Bed Reactors with Trickling Gas-Liquid Down-Flow.*", Chemical Engineering Communications 50:213-240, 1987.

Appendix I. Summary of the Method of Characteristics.

This appendix presents a summary of the method of characteristics which is used to solve first and second order hyperbolic partial differential equations. Much of the matter presented in this section has been summarised from Chapter 9 of the book *Analysis and Solution of Partial Differential Equations* by Robert L. Street [1978].

The solution of first-order or second-order partial differential equations with a single dependent variable and two independent variables can be visualised as a surface in (x,y,z) space. Analytical geometry can be used to enhance an understanding of the solution technique for such equations. This is particularly valid for initial-value problems in which we expect the solution to propagate from the region on which the initial data was specified.

In the case of first-order and hyperbolic second-order equations, information from the initial data propagates over well-defined paths in the surface representing the solution. These propagation paths are called *characteristics*. A knowledge of the existence of characteristics gives considerable insight into the expected behaviour of a problem's solution, even before the solution is known.

This summary is restricted to first-order equations which involve a single dependant variable and two independent variables.

The general first-order partial differential equation in two independent variables (x,y) is:

$$F(x,y,z,p,q)=0 \qquad (A1\text{-}1)$$

where

$$p=\frac{\partial z}{\partial x} \qquad (A1\text{-}2) \qquad q=\frac{\partial z}{\partial y} \qquad (A1\text{-}3)$$

If equation (A1-1) is a quasi-linear equation, it can be written as:

$$Pp+Qq=R \qquad (A1\text{-}4)$$

where P,Q and R are functions of x,y,and z but not of p or q. It is assumed that P,Q and R, together with their first derivatives, are continuous in the region of the problem under consideration. Further, P and Q are assumed not to vanish simultaneously. This implies:

$$P^2+Q^2\neq 0 \qquad (A1\text{-}5)$$

First-order equations arise in many physical systems. Equations (3-5) and (3-23) were both first-order hyperbolic partial differential equations. These equations model the bulk flow and chemical reaction of a fluid reagent through a column. These equations, which are so-called Convection Equations, are identical to the quasi-linear equation (A1-4). To demonstrate this, equation (3-23) has been written below in exactly the same form as equation (A1-4):

$$\frac{\partial \alpha}{\partial \tau'} + DG1\frac{\partial \alpha}{\partial \xi} = -DG2\,\alpha\,\sigma_{Bi}^{\phi_{Bi}} \qquad (A1\text{-}6)$$

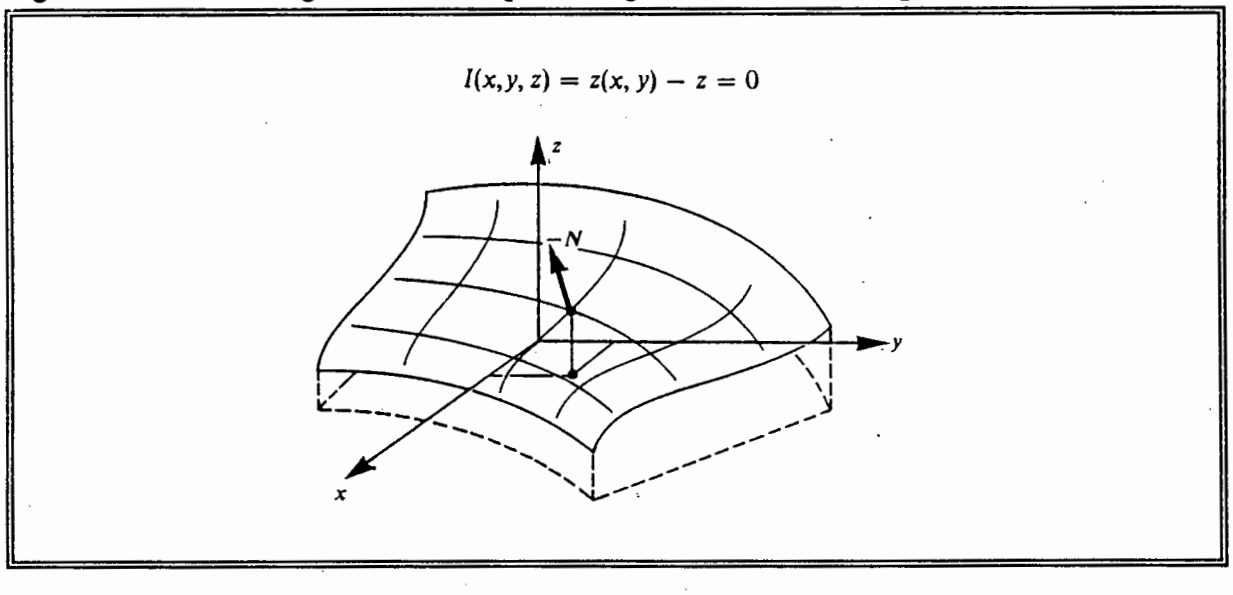
where

$$1=P \quad (A1\text{-}7) \qquad DG2=Q \quad (A1\text{-}8) \qquad -DG2\,\alpha\,\sigma_{Bi}^{\phi_{Bi}}=R \quad (A1\text{-}9)$$

The solution z(x,y) to the quasi-linear equation (A1-4) can be represented as a surface in (x,y,z)space shown in Figure A1.1. This surface is called an integral surface because it represents the solution or 'integral' of the partial differential equation. The integral surface for (A1-4) is represented by the formal identity:

$$I(x,y,z)=z(x,y)-z \qquad (A1\text{-}10)$$

Figure A1.1 An integral surface representing the solution to equation (A1-4).



Taking the differential of equation (A1-10) will, from analytical geometry considerations,

result in the equation for the tangent plane to the integral surface at any point. The equation for the equation plane is thus:

$$\frac{\partial I}{\partial x}dx+\frac{\partial I}{\partial y}dy+\frac{\partial I}{\partial z}=0 \qquad (A1\text{-}11)$$

or

$$\frac{\partial z}{\partial x}dx+\frac{\partial z}{\partial y}dy-1dz=0 \qquad (A1\text{-}12)$$

Equation (A1-12) is nothing more than the total differential and can be written as:

$$dz=p\ dx+q\ dy \qquad (A1\text{-}13)$$

Equation (A1-13) is a tangent plane to the solution plane at any point. Accordingly, the vector $\mathbf{N}=\mathbf{pi}+\mathbf{qj}-1\mathbf{k}$; that is, the vector (p,q,-1); is normal to the integral surface, depicted in Figure A1.1, at any point.

The scalar product of **N** and the vector (P,Q,R) is zero because:

$$\begin{aligned}(P,Q,R)\cdot(p,q,-1) &= (Pi+Qj+Rk)\cdot(pi+qj-1k)\\ &= Pp+Qq-R = 0\end{aligned} \qquad (A1\text{-}14)$$

from equation (A1-4). Thus (P,Q,R) is perpendicular to **N**. Consequently, (P,Q,R) is tangent to the integral solution surface at every point and lies in the plane of the tangent equation (A1-13). This means that the first-order equation (A1-4) can be regarded as a geometric requirement that any integral or solution z(x,y)through the point (x,y,z)must be tangent to the vector (P,Q,R). In fact, by beginning at some point on the integral surface, one line which lies in the integral surface can be determined from the *known* tangent vector (P,Q,R). This line is termed a *characteristic*. By determining sets of characteristics, the solution surface can be generated.

Stated in another way, the vector (P,Q,R) has been determined to be tangent to the solution surface at all points. A single line in the integral surface can be determined by starting on a point on the integral surface, moving in the direction of (P,Q,R) and determining the curve which is tangent to (P,Q,R).

Suppose that the *position vector* **r** of a point on a characteristic curve is:

$$r=xi+yj+zk \qquad (A1\text{-}15)$$

This vector can be expressed parametrically in terms of distance s along a particular curve:

$$r(s) = x(s)\,i + y(s)\,j + z(s)\,k \qquad (A1\text{-}16)$$

and the tangent vector to the curve at a particular point is:

$$\frac{\partial r}{\partial s} = \frac{\partial x}{\partial s}\,i + \frac{\partial y}{\partial s}\,j + \frac{\partial z}{\partial s}\,k \qquad (A1\text{-}17)$$

The vector (*dx/ds;dy/ds;dz/ds*) is tangent to the characteristic curve and the solution or integral surface. (P,Q,R) is also tangent to the solution or integral surface thus the two vectors are co-incident and their components must thus be proportional. This implies:

$$\frac{dx/ds}{P} = \frac{dy/ds}{Q} = \frac{dz/ds}{R} \qquad (A1\text{-}18)$$

or

$$\frac{dx}{P} = \frac{dy}{Q} = \frac{dz}{R} \qquad (A1\text{-}19)$$

The system of equations represented in (A1-19) must be satisfied by any characteristic in the solution surface. Thus the partial differential equation (A1-4) can be converted into a system of ordinary differential equations represented by (A1-19). Equivalent systems of equations (A1-19) are:

$$\frac{dy}{dx} = \frac{Q}{P} \qquad \frac{dz}{dx} = \frac{R}{P} \qquad (A1\text{-}20)$$

or

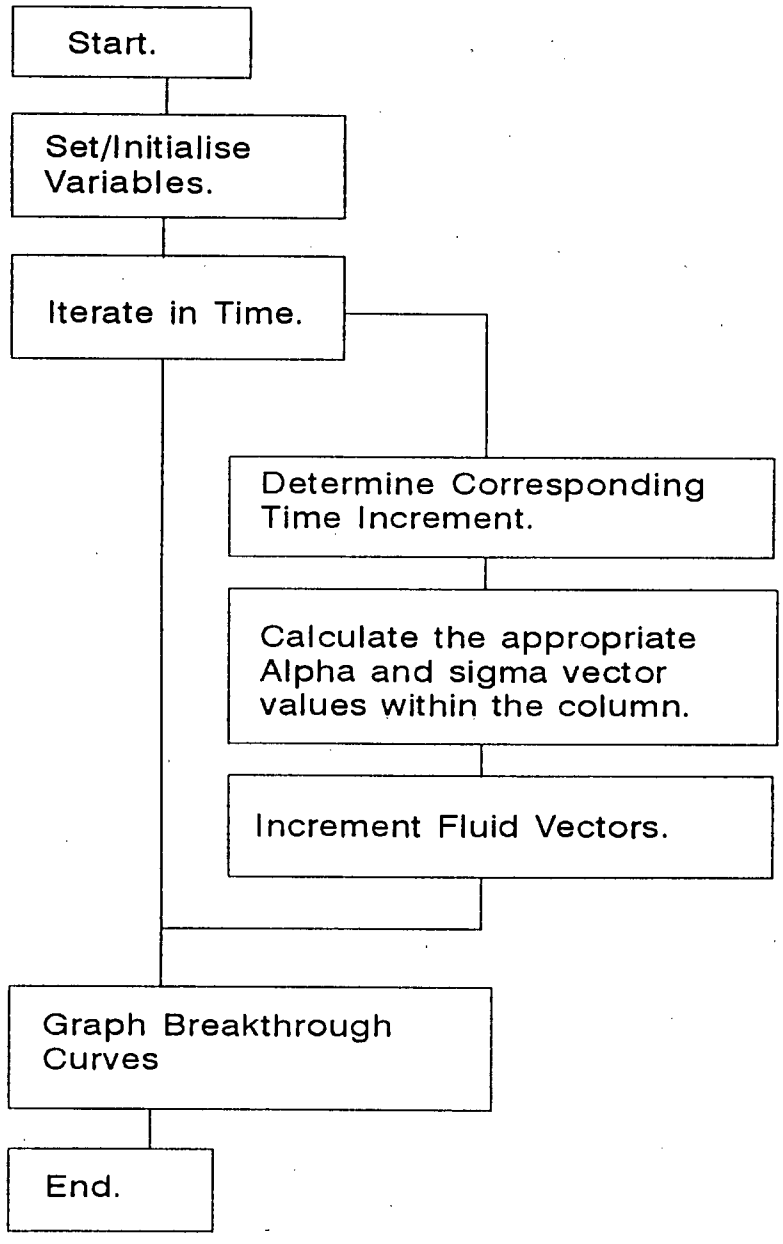
$$\frac{dx}{dy} = \frac{P}{Q} \qquad \frac{dz}{dy} = \frac{R}{Q} \qquad (A1\text{-}21)$$

Equations (A1-20) or (A1-21) are ordinary differential equations which can be solved by conventional means.

Appendix II. Solution Algorithm and Code for Model4D1.PAS and Model4D2.PAS.

Model4 represents the computer routines for the Macroscopic, Lumped Parameter Model Presented in Chapter 3.

Solution Algorithm for Model4D.



```
Program Model4D1;

{Model4D1 .   This program is a code for the convective flux model        }
{             which incorporates unsteady state flow.                     }
{             Deals with variable solid reaction orders.                  }
{             Note that this code uses the NEW DIMENSIONLESS GROUPS       }
{             defined on 18 January 1995.                                 }

{Coded:       Graham Davies.                                              }
{             Department of Chemical Engineering.                         }
{             University of Cape Town.                                    }
{             19 January 1995.                                            }
{             10 May     1995 - Updated.                                  }
{=========================================================================}
{Declarations:                                                            }

uses  crt,stdhdr,graph,worlddr,segraph,integrat;


var

   SupVelFile,ExpData        :text;

   SupVelData                :array[1..2,1..2]   of double;
   Data                      :array[0..256,1..3] of double;


   ART,ARTP1,SRT_1,SRT_2,SRT_3,SolSRT_1,SolSRT_2,SolSRT_3  :ShortVector;
   HoldSRT_1,HoldSRT_2,HoldSRT_3                           :ShortVector;
   DataSetX,DataSetY,InterG_1,InterG_2,InterG_3            :VeryLongVector;
   Conv_1,Conv_2,Conv_3                                    :VeryLongVector;
   BTCART,BTCSRT_1,BTCSRT_2,BTCSRT_3,TimeVector            :VeryLongVector;

   IntLength,IntLengthCrit,Delta_t,DeltaT,ArbValue         :Double;
   CumTime,ErrorSum                                        :Double;
   DG2_1,DG2_2,DG2_3,DG3_1,DG3_2,DG3_3,Ord_1,Ord_2,Ord_3   :Double;
   Voidage,Length,uStar,DeltaE                             :Double;

   GfMax_X,GfMax_Y                                         :Real;

   ArbIndex,SubIntervalIndex,Continue,PrintFilterIndex     :Integer;
   VectorIndex,HardCopy,error,No_of_DataPoints,Position    :Integer;
   N,NumSubInt,Iterations,PrintFilter                      :Integer;
   No_of_Points                                            :Integer;

   TimeCounter                                             :LongInt;

   NS,LengthS                                              :String;
   DG2_1S,DG2_2S,DG2_3S,DG3_1S,DG3_2S,DG3_3S               :String;
   GLabel2,GLabel4,GLabel5                                 :String;

{-------------------------------------------------------------------------}
Procedure Define_Variables;

begin

   DeltaE     := 0.02;       {Dimensionless space increment.             }
   N          := 50;         {Number of nodes.  (1/DeltaE)               }

   uStar      := 1/86400;    {Reference fluid velocity.  Equivalent to   }
```



```
                             {1m in 24 hours.                            }

   Length     := 0.5;        {Length of the column. (m)                  }
   Voidage    := 0.42;       {Voidage of the column.                     }

   Ord_1      := 1.0;        {Reaction order of first reaction.          }
   Ord_2      := 1.0;        {Reaction order of second reaction.         }
   Ord_3      := 1.0;        {Reaction order of third reaction.          }

   DG2_1      := 10.0;       {Defined dimensionless group.               }
   DG2_2      := 0.0;        {Defined dimensionless group.               }
   DG2_3      := 0.0;        {Defined dimensionless group.               }

   DG3_1      := 1.0;        {Defined dimensionless group.               }
   DG3_2      := 0.0;        {Defined dimensionless group.               }
   DG3_3      := 0.0;        {Defined dimensionless group.               }


   GfMax_X    := 2.0;        {Maximum x-value for conversion graph.      }
                             {(Dimensionless Time.)                      }
   GfMax_Y    := 1.0;        {Maximum y-value for breakthrough graph.    }
                             {(Concentration - kg/m3 fluid.)             }

   Iterations := 100;        {Iterations in time.                        }
   PrintFilter:= 1;          {Graph printing filter.                     }
   NumSubInt  := 10;         {Number of subintervals within each spatial }
                             {increment.                                 }

end;


{-------------------------------------------------------------------------}
Procedure Initialise_Superficial_Velocity_File;

begin
   assign(SupVelFile,'SupVelFi.Dat');
   reset(SupVelFile);
   SupVelData[1,1]:=0;
   SupVelData[1,2]:=0;
   read(SupVelFile,SupVelData[2,1]);
   read(SupVelFile,SupVelData[2,2]);
   IntLengthCrit:=Voidage*DeltaE*Length;
end;

{-------------------------------------------------------------------------}
Procedure Time0_ART_and_SRTs;
{This procedure initialises the alpha and sigma vectors.                  }

begin
   ART[0]     :=1;
   SRT_1[0]   :=1;
   SRT_2[0]   :=1;
   SRT_3[0]   :=1;
   SolSRT_1[0]:=0;
   SolSRT_2[0]:=0;
   SolSRT_3[0]:=0;
   HoldSRT_1[0]:=1;
   HoldSRT_2[0]:=1;
   HoldSRT_3[0]:=1;
   for ArbIndex:=1 to N do begin
```

Code Listing for Model4D1.PAS.

```
      ART[ArbIndex]:=0;
      SRT_1[ArbIndex]:=1;
      SRT_2[ArbIndex]:=1;
      SRT_3[ArbIndex]:=1;
      SolSRT_1[ArbIndex]:=0;
      SolSRT_2[ArbIndex]:=0;
      SolSRT_3[ArbIndex]:=0;
      HoldSRT_1[ArbIndex]:=1;
      HoldSRT_2[ArbIndex]:=1;
      HoldSRT_3[ArbIndex]:=1;
   end;
end;

{------------------------------------------------------------------------}
Procedure Determine_Corresponding_DeltaT;
{This procedure uses the method of characteristics to determine the      }
{progression in time for a given CharDeltaE.  It allows for unsteady     }
{state flow.                                                             }

begin
   IntLength:=0;
   Delta_t:=0;
   while IntLength<>IntLengthCrit do begin
      if EOF(SupVelFile) then begin
         closesegraphics;
         writeln('Insufficient Data in the Superficial Velocity Data File to');
         writeln('execute the required number of iterations.');
         writeln('Present number of iterations:=  ',TimeCounter);
         Continue:=0;
         IntLength:=IntLengthCrit;
         readln;
      end;
      IntLength:=IntLength+(SupVelData[2,1]-SupVelData[1,1])*SupVelData[2,2];
      Delta_t:=Delta_t+(SupVelData[2,1]-SupVelData[1,1]);
      If IntLength>IntLengthCrit then begin
         ArbValue:=Intlength-IntLengthCrit;
         Delta_t:=Delta_t-ArbValue/SupVelData[2,2];
         IntLength:=IntLengthCrit;
         SupVelData[1,1]:=SupVelData[2,1]-ArbValue/SupVelData[2,2];
         SupVelData[1,2]:=SupVelData[2,2];
      end else begin
         SupVelData[1,1]:=SupVelData[2,1];
         SupVelData[1,2]:=SupVelData[2,2];
         read(SupVelFile, SupVelData[2,1]);
         read(SupVelFile, SupVelData[2,2])
      end;
   end;
   DeltaT:=Delta_t*uStar/Length;
end;

{------------------------------------------------------------------------}
Procedure Iterate_within_DeltaEs;
{This procedure updates the alpha and sigma vectors during the time it   }
{takes for the fluid to move through a spatial increment.                }

begin
   for ArbIndex:=0 to N do begin
      for SubIntervalIndex:=1 to NumSubInt do begin
         ARTP1[ArbIndex]  :=ART[ArbIndex]+
                            DeltaT/NumSubInt*(-DG2_1*ART[ArbIndex]
```



```
                            *exp(Ord_1*ln(SRT_1[ArbIndex]))+
                            DeltaT/NumSubInt*(-DG2_2*ART[ArbIndex]
                            *exp(Ord_2*ln(SRT_2[ArbIndex]))+
                            DeltaT/NumSubInt*(-DG2_3*ART[ArbIndex]
                            *exp(Ord_3*ln(SRT_3[ArbIndex])));
         SRT_1[ArbIndex]  :=SRT_1[ArbIndex]+DeltaT/NumSubInt*
                            (-DG2_1*DG3_1*ART[ArbIndex]
                            *exp(Ord_1*ln(SRT_1[ArbIndex])));
         SRT_2[ArbIndex]  :=SRT_2[ArbIndex]+DeltaT/NumSubInt*
                            (-DG2_2*DG3_2*ART[ArbIndex]
                            *exp(Ord_2*ln(SRT_2[ArbIndex])));
         SRT_3[ArbIndex]  :=SRT_3[ArbIndex]+DeltaT/NumSubInt*
                            (-DG2_3*DG3_3*ART[ArbIndex]
                            *exp(Ord_3*ln(SRT_3[ArbIndex])));
         ART[ArbIndex]    :=ARTP1[ArbIndex];
      end;
   end;
end;

{------------------------------------------------------------------------}
Procedure Increment_Fluid_Vectors;
{This procedure updates the alpha vector once the fluid has progressed   }
{through a spatial increment.                                            }

begin
   for ArbIndex:=1 to N do begin
      SolSRT_1[ArbIndex] :=SolSRT_1[ArbIndex]+(HoldSRT_1[ArbIndex]-
                           SRT_1[ArbIndex])/N;
      SolSRT_2[ArbIndex] :=SolSRT_2[ArbIndex]+(HoldSRT_2[ArbIndex]-
                           SRT_2[ArbIndex])/N;
      SolSRT_3[ArbIndex] :=SolSRT_3[ArbIndex]+(HoldSRT_3[ArbIndex]-
                           SRT_3[ArbIndex])/N;
      HoldSRT_1[ArbIndex]:=SRT_1[ArbIndex];
      HoldSRT_2[ArbIndex]:=SRT_2[ArbIndex];
      HoldSRT_3[ArbIndex]:=SRT_3[ArbIndex];
   end;

   for ArbIndex:=N downto 1 do begin
      ART[ArbIndex]     :=ART[ArbIndex-1];
      SolSRT_1[ArbIndex]:=SolSRT_1[ArbIndex-1];
      SolSRT_2[ArbIndex]:=SolSRT_2[ArbIndex-1];
      SolSRT_3[ArbIndex]:=SolSRT_3[ArbIndex-1];
   end;

   ART[0]     :=1;                {*****  Imposed boundary condition.    }
   SolSRT_1[0]:=0;                {*****  Imposed boundary condition.    }
   SolSRT_2[0]:=0;                {*****  Imposed boundary condition.    }
   SolSRT_3[0]:=0;                {*****  Imposed boundary condition.    }
end;

{------------------------------------------------------------------------}
Procedure Update_BreakThroughVectors;

begin
   TimeVector[VectorIndex]:=CumTime;
   BTCART[VectorIndex]    :=ART[N];
   BTCSRT_1[VectorIndex]  :=SolSRT_1[N];
   BTCSRT_2[VectorIndex]  :=SolSRT_2[N];
   BTCSRT_3[VectorIndex]  :=SolSRT_3[N];
end;
```

```
{-------------------------------------------------------------------------}
Procedure Label_Graph;

begin
   Str(N:3,NS);                        Str(Length:4:2,LengthS);
   Str(DG2_1:8:4,DG2_1S);              Str(DG2_2:8:4,DG2_2S);
   Str(DG2_3:8:4,DG2_3S);              Str(DG3_1:8:4,DG3_1S);
   Str(DG3_2:8:4,DG3_2S);              Str(DG3_3:8:4,DG3_3S);
   GLabel2:=Concat(' L ',LengthS,'           Nodes ',NS);
   GLabel4:=Concat(' DG2_1 ',DG2_1S,'    DG2_2 ',DG2_2S,'      DG2_3 ',DG2_3S);
   GLabel5:=Concat(' DG3_1 ',DG3_1S,'    DG3_2 ',DG3_2S,'      DG3_3 ',DG3_3S);
   LabelGraphWindow(150,905,GLabel2,0,0);
   LabelGraphWindow(150,875,GLabel4,0,0);
   LabelGraphWindow(150,845,GLabel5,0,0);
end;

{-------------------------------------------------------------------------}
Procedure Graph1_Initialise;

begin;
   initSEGraphics('f:\tp6');
   SetCurrentWindow(2);
   BorderCurrentWindow(1);
   SetAxesType(0,0);
   ScalePlotArea(0.0,0.0,1,1.2);
   SetXYIntercepts(0.0,0.0);
   SetColor(2);
   DrawXAxis(0.2,1);
   DrawYAxis(0.2,1);
   LabelXAxis(1,0);
   LabelYAxis(1,0);
   TitleXAxis('Dimensionless Length');
   TitleYAxis('Dimensionless Concentration');
   TitleWindow('Model4D1');
   LabelGraphWindow(375,935,'Concentration Profiles',0,0);
   Label_Graph;
   for ArbIndex:=0 to N do DataSetX[ArbIndex]:=ArbIndex*DeltaE;
end;

{-------------------------------------------------------------------------}
Procedure Graph1_Results;

begin
   for ArbIndex:=0 to N do DataSetY[ArbIndex]:=ART[ArbIndex];
   LinePlotData(DataSetX,DataSetY,N+1,3,0);
   for ArbIndex:=0 to N do DataSetY[ArbIndex]:=SRT_1[ArbIndex];
   LinePlotData(DataSetX,DataSetY,N+1,5,2);
   for ArbIndex:=0 to N do DataSetY[ArbIndex]:=SRT_2[ArbIndex];
   LinePlotData(DataSetX,DataSetY,N+1,4,1);
   for ArbIndex:=0 to N do DataSetY[ArbIndex]:=SRT_3[ArbIndex];
   LinePlotData(DataSetX,DataSetY,N+1,7,1);
end;

{-------------------------------------------------------------------------}
Procedure Graph_BreakThroughCurves;

begin
   closesegraphics;
   initsegraphics('f:\tp6');
```



```
   SetCurrentWindow(2);
   BorderCurrentWindow(1);
   SetAxesType(0,0);
   ScalePlotArea(0.0,0.0,GfMax_X,GfMax_Y);
   SetXYIntercepts(0.0,0.0);
   SetColor(2);
   DrawXAxis((GfMax_X/5),1);
   DrawYAxis((GfMax_Y/5),1);
   LabelXAxis(1,0);
   LabelYAxis(1,0);
   TitleXAxis('Dimensionless Time');
   TitleYAxis('Fraction of Total Original Contaminant Removed');
   TitleWindow('Model4D1 ');
   LabelGraphWindow(380,935,'Breakthrough Curves',0,0);
   Label_Graph;
   for ArbIndex:=0 to VectorIndex do DataSetX[ArbIndex]:=TimeVector[ArbIndex];
(*   for ArbIndex:=0 to VectorIndex do DataSetY[ArbIndex]:=BTCART[ArbIndex];
   LinePlotData(DataSetX,DataSetY,(VectorIndex+1),3,0);
*)   for ArbIndex:=0 to VectorIndex do DataSetY[ArbIndex]:=BTCSRT_1[ArbIndex];
   LinePlotData(DataSetX,DataSetY,(VectorIndex+1),5,1);
   for ArbIndex:=0 to VectorIndex do DataSetY[ArbIndex]:=BTCSRT_2[ArbIndex];
   LinePlotData(DataSetX,DataSetY,(VectorIndex+1),4,1);
   for ArbIndex:=0 to VectorIndex do DataSetY[ArbIndex]:=BTCSRT_3[ArbIndex];
   LinePlotData(DataSetX,DataSetY,(VectorIndex+1),7,1);

end;

{-------------------------------------------------------------------------}
{MAIN PROGRAM                                                             }

begin

Continue:=1;
CumTime :=0;
PrintFilterIndex:=0;
TimeCounter:=0;
VectorIndex:=0;
TimeVector[VectorIndex]:=CumTime;

BTCART[VectorIndex]    :=0;
BTCSRT_1[VectorIndex]  :=0;
BTCSRT_2[VectorIndex]  :=0;
BTCSRT_3[VectorIndex]  :=0;

Conv_1[VectorIndex]    :=0;
Conv_2[VectorIndex]    :=0;
Conv_3[VectorIndex]    :=0;

Define_Variables;
Time0_ART_and_SRTs;
Initialise_Superficial_Velocity_File;
Graph1_Initialise;


while (Continue=1) and (TimeCounter<Iterations) do
begin
   TimeCounter:=TimeCounter+1;
   PrintFilterIndex:=PrintFilterIndex+1;
   Determine_Corresponding_DeltaT;
```

```
   CumTime:=CumTime+DeltaT;
   Iterate_within_DeltaEs;
   if PrintFilterIndex=PrintFilter then
   begin
      VectorIndex:=VectorIndex+1;
      PrintFilterIndex:=0;
      Update_BreakThroughVectors;
      Graph1_Results;
   end;
   Increment_Fluid_Vectors;
end;


readln(HardCopy);
if hardcopy=1 then ScreenDump(3,0,2,1.0,1.0,0,1,0,error);


Graph_BreakThroughCurves;
readln(Hardcopy);
if hardcopy=1 then ScreenDump(3,0,2,1.0,1.0,0,1,0,error);

closesegraphics;

end.
{=============================================================================}
```

```
Program Model4D1;

{Model4D2 .   This program is a code for the convective flux model        }
{             which incorporates unsteady state flow.                     }
{             Deals with variable solid reaction orders.                  }
{             Note that this code uses the NEW DIMENSIONLESS GROUPS       }
{             defined on 18 January 1995.                                 }
{             This version presents breakthrough curves relative to solid }
{             reactatn 1.                                                 }

{Coded:       Graham Davies.                                              }
{             Department of Chemical Engineering.                         }
{             University of Cape Town.                                    }
{             19 January 1995.                                            }
{             10 May     1995 - Updated.                                  }
{=========================================================================}
{Declarations:                                                            }

uses  crt,stdhdr,graph,worlddr,segraph,integrat;


var

   SupVelFile,ExpData        :text;

   SupVelData                :array[1..2,1..2]   of double;
   Data                      :array[0..256,1..3] of double;


   ART,ARTP1,SRT_1,SRT_2,SRT_3,SolSRT_1,SolSRT_2,SolSRT_3  :ShortVector;
   HoldSRT_1,HoldSRT_2,HoldSRT_3                           :ShortVector;
   DataSetX,DataSetY,InterG_1,InterG_2,InterG_3            :VeryLongVector;
   Conv_1,Conv_2,Conv_3                                    :VeryLongVector;
   BTCART,BTCSRT_1,BTCSRT_2,BTCSRT_3,TimeVector            :VeryLongVector;

   IntLength,IntLengthCrit,Delta_t,DeltaT,ArbValue         :Double;
   CumTime,ErrorSum                                        :Double;
   DG2_1,DG2_2,DG2_3,DG3_1,DG3_2,DG3_3,Ord_1,Ord_2,Ord_3   :Double;
   Voidage,Length,uStar,DeltaE                             :Double;

   GfMax_X,GfMax_Y                                         :Real;

   ArbIndex,SubIntervalIndex,Continue,PrintFilterIndex     :Integer;
   VectorIndex,HardCopy,error,No_of_DataPoints,Position    :Integer;
   N,NumSubInt,Iterations,PrintFilter                      :Integer;
   No_of_Points                                            :Integer;

   TimeCounter                                             :LongInt;

   NS,LengthS                                              :String;
   DG2_1S,DG2_2S,DG2_3S,DG3_1S,DG3_2S,DG3_3S               :String;
   GLabel2,GLabel4,GLabel5                                 :String;

{-------------------------------------------------------------------------}
Procedure Define_Variables;

begin

   DeltaE     := 0.02;       {Dimensionless space increment.             }
   N          := 50;         {Number of nodes.  (1/DeltaE)               }
```



```
   uStar      := 1/86400;    {Reference fluid velocity.  Equivalent to   }
                             {1m in 24 hours.                            }

   Length     := 0.5;        {Length of the column. (m)                  }
   Voidage    := 0.42;       {Voidage of the column.                     }

   Ord_1      := 1.0;        {Reaction order of first reaction.          }
   Ord_2      := 1.0;        {Reaction order of second reaction.         }
   Ord_3      := 1.0;        {Reaction order of third reaction.          }

   DG2_1      := 10.0;       {Defined dimensionless group.               }
   DG2_2      := 1.0;        {Defined dimensionless group.               }
   DG2_3      := 10.0;       {Defined dimensionless group.               }

   DG3_1      := 1.0;        {Defined dimensionless group.               }
   DG3_2      := 1.0;        {Defined dimensionless group.               }
   DG3_3      := 0.01;       {Defined dimensionless group.               }


   GfMax_X    := 4.0;        {Maximum x-value for conversion graph.      }
                             {(Dimensionless Time.)                      }
   GfMax_Y    := 0.02;       {Maximum y-value for breakthrough graph.    }
                             {(Concentration - kg/m3 fluid.)             }

   Iterations := 100;        {Iterations in time.                        }
   PrintFilter:= 10;         {Graph printing filter.                     }
   NumSubInt  := 10;         {Number of subintervals within each spatial }
                             {increment.                                 }

end;


{-------------------------------------------------------------------------}
Procedure Initialise_Superficial_Velocity_File;

begin
   assign(SupVelFile,'SupVelFi.Dat');
   reset(SupVelFile);
   SupVelData[1,1]:=0;
   SupVelData[1,2]:=0;
   read(SupVelFile,SupVelData[2,1]);
   read(SupVelFile,SupVelData[2,2]);
   IntLengthCrit:=Voidage*DeltaE*Length;
end;

{-------------------------------------------------------------------------}
Procedure Time0_ART_and_SRTs;
{This procedure initialises the alpha and sigma vectors.                  }

begin
   ART[0]:=1;
   SRT_1[0]:=1;
   SRT_2[0]:=1;
   SRT_3[0]:=1;
   SolSRT_1[0]:=0;
   SolSRT_2[0]:=0;
   SolSRT_3[0]:=0;
   HoldSRT_1[0]:=1;
   HoldSRT_2[0]:=1;
```

Code Listing for Model4D2.PAS.

```
   HoldSRT_3[0]:=1;
   for ArbIndex:=1 to N do begin
      ART[ArbIndex]:=0;
      SRT_1[ArbIndex]:=1;
      SRT_2[ArbIndex]:=1;
      SRT_3[ArbIndex]:=1;
      SolSRT_1[ArbIndex]:=0;
      SolSRT_2[ArbIndex]:=0;
      SolSRT_3[ArbIndex]:=0;
      HoldSRT_1[ArbIndex]:=1;
      HoldSRT_2[ArbIndex]:=1;
      HoldSRT_3[ArbIndex]:=1;
   end;
   for ArbIndex:=0 to maxv do begin
      BTCART[ArbIndex]    :=0;
      BTCSRT_1[ArbIndex]  :=0;
      BTCSRT_2[ArbIndex]  :=0;
      BTCSRT_3[ArbIndex]  :=0;
   end;
end;

{-------------------------------------------------------------------------}
Procedure Determine_Corresponding_DeltaT;
{This procedure uses the method of characteristics to determine the      }
{progression in time for a given CharDeltaE.  It allows for unsteady     }
{state flow.                                                             }

begin
   IntLength:=0;
   Delta_t:=0;
   while IntLength<>IntLengthCrit do begin
      if EOF(SupVelFile) then begin
         closesegraphics;
         writeln('Insufficient Data in the Superficial Velocity Data File to');
         writeln('execute the required number of iterations.');
         writeln('Present number of iterations:=  ',TimeCounter);
         Continue:=0;
         IntLength:=IntLengthCrit;
         readln;
      end;
      IntLength:=IntLength+(SupVelData[2,1]-SupVelData[1,1])*SupVelData[2,2];
      Delta_t:=Delta_t+(SupVelData[2,1]-SupVelData[1,1]);
      If IntLength>IntLengthCrit then begin
         ArbValue:=Intlength-IntLengthCrit;
         Delta_t:=Delta_t-ArbValue/SupVelData[2,2];
         IntLength:=IntLengthCrit;
         SupVelData[1,1]:=SupVelData[2,1]-ArbValue/SupVelData[2,2];
         SupVelData[1,2]:=SupVelData[2,2];
      end else begin
         SupVelData[1,1]:=SupVelData[2,1];
         SupVelData[1,2]:=SupVelData[2,2];
         read(SupVelFile, SupVelData[2,1]);
         read(SupVelFile, SupVelData[2,2])
      end;
   end;
   DeltaT:=Delta_t*uStar/Length;
end;

{-------------------------------------------------------------------------}
Procedure Iterate_within_DeltaEs;
```



```
{This procedure updates the alpha and sigma vectors during the time it   }
{takes for the fluid to move through a spatial increment.                }

begin
   for ArbIndex:=0 to N do begin
      for SubIntervalIndex:=1 to NumSubInt do begin
         ARTP1[ArbIndex]  :=ART[ArbIndex]+
                            DeltaT/NumSubInt*(-DG2_1*ART[ArbIndex]
                            *exp(Ord_1*ln(SRT_1[ArbIndex])))+
                            DeltaT/NumSubInt*(-DG2_2*ART[ArbIndex]
                            *exp(Ord_2*ln(SRT_2[ArbIndex])))+
                            DeltaT/NumSubInt*(-DG2_3*ART[ArbIndex]
                            *exp(Ord_3*ln(SRT_3[ArbIndex])));
         SRT_1[ArbIndex]  :=SRT_1[ArbIndex]+DeltaT/NumSubInt*
                            (-DG2_1*DG3_1*ART[ArbIndex]
                            *exp(Ord_1*ln(SRT_1[ArbIndex])));
         SRT_2[ArbIndex]  :=SRT_2[ArbIndex]+DeltaT/NumSubInt*
                            (-DG2_2*DG3_2*ART[ArbIndex]
                            *exp(Ord_2*ln(SRT_2[ArbIndex])));
         SRT_3[ArbIndex]  :=SRT_3[ArbIndex]+DeltaT/NumSubInt*
                            (-DG2_3*DG3_3*ART[ArbIndex]
                            *exp(Ord_3*ln(SRT_3[ArbIndex])));
         ART[ArbIndex]    :=ARTP1[ArbIndex];
      end;
   end;
end;

{-------------------------------------------------------------------------}
Procedure Increment_Fluid_Vectors;
{This procedure updates the alpha vector once the fluid has progressed   }
{through a spatial increment.                                            }

begin
   for ArbIndex:=1 to N do begin
      SolSRT_1[ArbIndex] :=SolSRT_1[ArbIndex]+(HoldSRT_1[ArbIndex]-
                           SRT_1[ArbIndex])/N;
      SolSRT_2[ArbIndex] :=SolSRT_2[ArbIndex]+(HoldSRT_2[ArbIndex]-
                           SRT_2[ArbIndex])/N;
      SolSRT_3[ArbIndex] :=SolSRT_3[ArbIndex]+(HoldSRT_3[ArbIndex]-
                           SRT_3[ArbIndex])/N;
      HoldSRT_1[ArbIndex]:=SRT_1[ArbIndex];
      HoldSRT_2[ArbIndex]:=SRT_2[ArbIndex];
      HoldSRT_3[ArbIndex]:=SRT_3[ArbIndex];
   end;

   for ArbIndex:=N downto 1 do begin
      ART[ArbIndex]     :=ART[ArbIndex-1];
      SolSRT_1[ArbIndex]:=SolSRT_1[ArbIndex-1];
      SolSRT_2[ArbIndex]:=SolSRT_2[ArbIndex-1];
      SolSRT_3[ArbIndex]:=SolSRT_3[ArbIndex-1];
   end;

   ART[0]     :=1;              {*****  Imposed boundary condition.     }
   SolSRT_1[0]:=0;              {*****  Imposed boundary condition.     }
   SolSRT_2[0]:=0;              {*****  Imposed boundary condition.     }
   SolSRT_3[0]:=0;              {*****  Imposed boundary condition.     }
end;

{-------------------------------------------------------------------------}
Procedure Update_BreakThroughVectors;
```

```
begin
   TimeVector[VectorIndex]:=CumTime;
   BTCART[VectorIndex]    :=ART[N];
   BTCSRT_1[VectorIndex]  :=SolSRT_1[N];
   if DG3_2<>0 then BTCSRT_2[VectorIndex]  :=SolSRT_2[N]*DG3_1/DG3_2;
   if DG3_3<>0 then BTCSRT_3[VectorIndex]  :=SolSRT_3[N]*DG3_1/DG3_3;
end;

{-------------------------------------------------------------------------}
Procedure Label_Graph;

begin
   Str(N:3,NS);                        Str(Length:4:2,LengthS);
   Str(DG2_1:8:4,DG2_1S);              Str(DG2_2:8:4,DG2_2S);
   Str(DG2_3:8:4,DG2_3S);              Str(DG3_1:8:4,DG3_1S);
   Str(DG3_2:8:4,DG3_2S);              Str(DG3_3:8:4,DG3_3S);
   GLabel2:=Concat(' L ',LengthS,'          Nodes ',NS);
   GLabel4:=Concat(' DG2_1 ',DG2_1S,'    DG2_2 ',DG2_2S,'      DG2_3 ',DG2_3S);
   GLabel5:=Concat(' DG3_1 ',DG3_1S,'    DG3_2 ',DG3_2S,'      DG3_3 ',DG3_3S);
   LabelGraphWindow(150,905,GLabel2,0,0);
   LabelGraphWindow(150,875,GLabel4,0,0);
   LabelGraphWindow(150,845,GLabel5,0,0);
end;

{-------------------------------------------------------------------------}
Procedure Graph1_Initialise;

begin;
   initSEGraphics('f:\tp6');
   SetCurrentWindow(2);
   BorderCurrentWindow(1);
   SetAxesType(0,0);
   ScalePlotArea(0.0,0.0,1,1.2);
   SetXYIntercepts(0.0,0.0);
   SetColor(2);
   DrawXAxis(0.2,1);
   DrawYAxis(0.2,1);
   LabelXAxis(1,0);
   LabelYAxis(1,0);
   TitleXAxis('Dimensionless Length');
   TitleYAxis('Dimensionless Concentration');
   TitleWindow('Model4D2');
   LabelGraphWindow(375,935,'Concentration Profiles',0,0);
   Label_Graph;
   for ArbIndex:=0 to N do DataSetX[ArbIndex]:=ArbIndex*DeltaE;
end;

{-------------------------------------------------------------------------}
Procedure Graph1_Results;

begin
   for ArbIndex:=0 to N do DataSetY[ArbIndex]:=ART[ArbIndex];
   LinePlotData(DataSetX,DataSetY,N+1,3,0);
   for ArbIndex:=0 to N do DataSetY[ArbIndex]:=SRT_1[ArbIndex];
   LinePlotData(DataSetX,DataSetY,N+1,5,2);
   for ArbIndex:=0 to N do DataSetY[ArbIndex]:=SRT_2[ArbIndex];
   LinePlotData(DataSetX,DataSetY,N+1,4,1);
   for ArbIndex:=0 to N do DataSetY[ArbIndex]:=SRT_3[ArbIndex];
   LinePlotData(DataSetX,DataSetY,N+1,7,1);
```



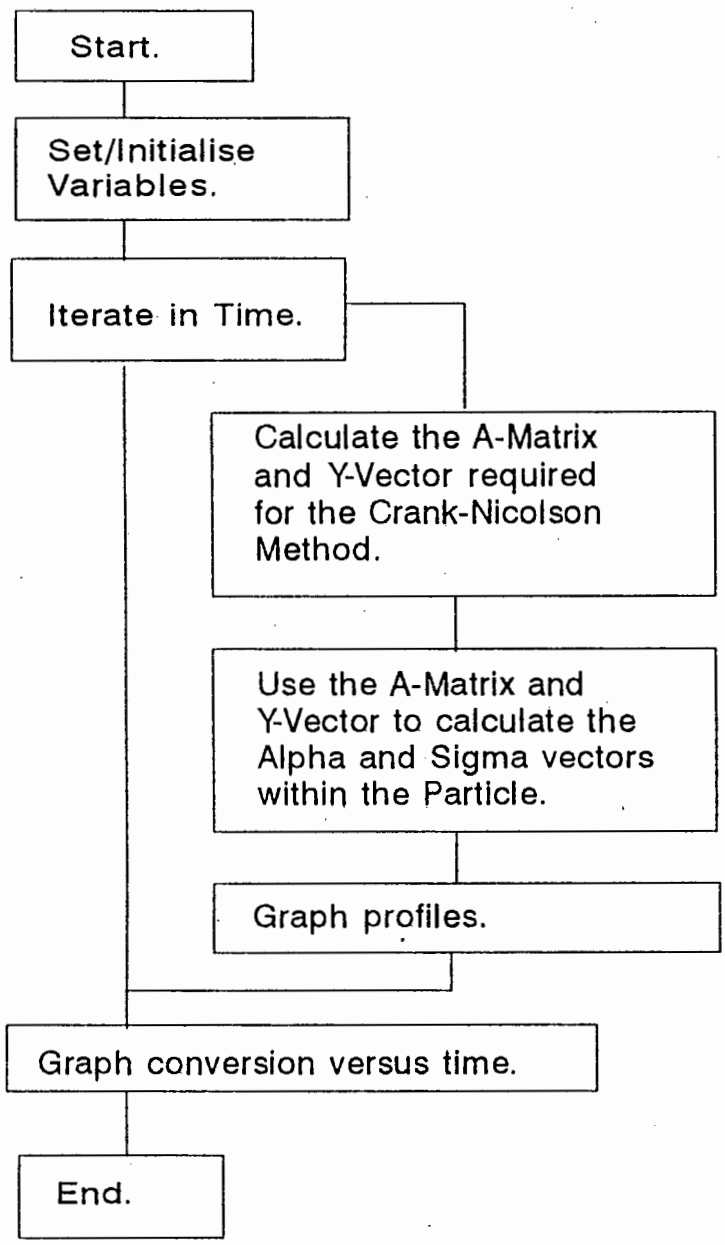
```
end;

{-------------------------------------------------------------------------}
Procedure Graph_BreakThroughCurves;

begin
   closesegraphics;
   initsegraphics('f:\tp6');
   SetCurrentWindow(2);
   BorderCurrentWindow(1);
   SetAxesType(0,0);
   ScalePlotArea(0.0,0.0,GfMax_X,GfMax_Y);
   SetXYIntercepts(0.0,0.0);
   SetColor(2);
   DrawXAxis((GfMax_X/5),1);
   DrawYAxis((GfMax_Y/5),1);
   LabelXAxis(1,0);
   LabelYAxis(1,0);
   TitleXAxis('Dimensionless Time');
   TitleYAxis('Breakthrough Conc. Rel. to Original Quantity Sol. 1.');
   TitleWindow('Model4D2 ');
   LabelGraphWindow(380,935,'Breakthrough Curves',0,0);
   Label_Graph;
   for ArbIndex:=0 to VectorIndex do DataSetX[ArbIndex]:=TimeVector[ArbIndex];
(*   for ArbIndex:=0 to VectorIndex do DataSetY[ArbIndex]:=BTCART[ArbIndex];
   LinePlotData(DataSetX,DataSetY,(VectorIndex+1),3,0);
*)   for ArbIndex:=0 to VectorIndex do DataSetY[ArbIndex]:=BTCSRT_1[ArbIndex];
   LinePlotData(DataSetX,DataSetY,(VectorIndex+1),5,1);
   for ArbIndex:=0 to VectorIndex do DataSetY[ArbIndex]:=BTCSRT_2[ArbIndex];
   LinePlotData(DataSetX,DataSetY,(VectorIndex+1),4,1);
   for ArbIndex:=0 to VectorIndex do DataSetY[ArbIndex]:=BTCSRT_3[ArbIndex];
   LinePlotData(DataSetX,DataSetY,(VectorIndex+1),7,1);

end;

{-------------------------------------------------------------------------}
{MAIN PROGRAM                                                             }

begin

Continue:=1;
CumTime :=0;
PrintFilterIndex:=0;
TimeCounter:=0;
VectorIndex:=0;
TimeVector[VectorIndex]:=CumTime;

BTCART[VectorIndex]    :=0;
BTCSRT_1[VectorIndex]  :=0;
BTCSRT_2[VectorIndex]  :=0;
BTCSRT_3[VectorIndex]  :=0;

Conv_1[VectorIndex]    :=0;
Conv_2[VectorIndex]    :=0;
Conv_3[VectorIndex]    :=0;

Define_Variables;
Time0_ART_and_SRTs;
Initialise_Superficial_Velocity_File;
Graph1_Initialise;
```

```
while (Continue=1) and (TimeCounter<Iterations) do
begin
   TimeCounter:=TimeCounter+1;
   PrintFilterIndex:=PrintFilterIndex+1;
   Determine_Corresponding_DeltaT;
   CumTime:=CumTime+DeltaT;
   Iterate_within_DeltaEs;
   if PrintFilterIndex=PrintFilter then
   begin
      VectorIndex:=VectorIndex+1;
      PrintFilterIndex:=0;
      Update_BreakThroughVectors;
      Graph1_Results;
   end;
   Increment_Fluid_Vectors;
end;


readln(HardCopy);
if hardcopy=1 then ScreenDump(3,0,2,1.0,1.0,0,1,0,error);


Graph_BreakThroughCurves;
readln(Hardcopy);
if hardcopy=1 then ScreenDump(3,0,2,1.0,1.0,0,1,0,error);

closesegraphics;

end.
{=============================================================================}
```

Appendix III. Solution Algorithm and Code for Model2D.PAS.

Model2 essentially represents suitable computer routines for Dixon's [1992] particle scale model discussed in Chaper 4.

Solution Algorithm for Model2D.



```
Program Model2D2;

{Model2D2.   This program calculates the concentration profile of fluid   }
{            and solid reactants within a particle using the equations as }
{            developed by Dixon.                                          }
{            Includes the possibility of a second solid reactant.         }
{            Includes the possibility of a surface concentration          }
{            different to the bulk concentration.                         }
{            Allows for a reaction order in terms of the solid reactant   }
{            other than 1.                                                }
{            This program also calculates the fractional conversion.      }
{            Numerical strategy used is a form of Crank-Nicolson Implicit }
{            finite difference.                                           }

{            Assumptions in this model include:                           }
{                    The solid reactant deposits within the particle      }

{                    resemble those on the surface.                       }
{                    The deposits would both react to the same extent     }
{                    if each were exposed to the same acid concentration  }
{                    for the same time.                                   }
{                    Surface acid concentration is still assumed to be    }
{                    constant - ie always 1.                              }

{Coded:      Graham Davies.                                               }
{            Department of Chemical Engineering.                          }
{            University of Cape Town.                                     }
{            5  August 1994.                                              }
{            28 July   1995 - Updated.                                    }
{-------------------------------------------------------------------------}
{Declarations:                                                            }

uses crt,stdhdr,gj,graph,worlddr,segraph,integrat,M2C1BrentRoots;

const

   Beta1     =1.0;     {Dimensionless Stoichiometric ratio defined pg 11  }
   KappaP1   =10.0;    {Ratio of reation rate of solid reactant residing  }
                       {within the particle to porous diffusion of fluid  }
                       {into the particle. Defined pg 11 Dixon.           }
   Lambda1   =0.0;     {Fraction of the leachable material on the surface.}
                       {Lambda1=1 implies all the material is on the      }
                       {surface.                                          }
   OrdSP1    =1.0;     {Reaction order of the solid in the pores.         }

   Beta2     =0.0;
   KappaP2   =0.0;
   Lambda2   =0.0;
   OrdSP2    =0.0;

   N         =24;      {Half the number of interior points.  r=0 and r=R  }
                       {not included.                                     }
   DeltaT    =0.0005;          {Time Increment                           }
   DeltaE    =1/(N+1); {Space Increment.  Calculated from 1/(N+1). (Since }
                       {                  R=i*dE and R is at point N+1.)  }
   Norm_Crit =1e-10;   {Convergence criteria based on the norm of vector  }
                       {***                                               }
   Conv_Crit =1e-8;    {Convergence criteria for the Brent Routine.       }
   SConc_Tol =1e-5;    {Dimensionless concentration of solid below which  }
                       {it is assumed to be negligible.                   }
```



```
   MaxIter   =100;     {Maximum iterations for the Brent Routine.         }

   Iterations=600;     {Iterations in time.                               }
   Print_Crit=10;      {Determined from desired Print_Delta_T/Delta_T      }
   Gf2_MaxX  =30;      {Graph 2 maximum dimensionless reaction time.      }

var

   ART,ARTP1,SRT1,SRT1P1,SRT1P1C          :^ShortVector;
   SRT2,SRT2P1,SRT2P1C,YVector            :^ShortVector;
   DataSetX,DataSetY,InterG1,InterG2      :^VeryLongVector;
   Conv1,Conv2                            :^VeryLongVector;
   AMatrix,AInverse                       :^SqrMat;
   AMatDet,IntegVal,XAxisMax              :real;
   Norm1,Norm2,A,AP1,S,Root,ValueAtRoot   :real;


   Rows,Cols,ArbIndex,DataPoints,Repeats  :integer;
   error,Plot_Var,Conv_Var,NewtIters      :integer;
   HardCopy1,HardCopy2                    :integer;

   Beta1S,KappaP1S,Lambda1S,GDTS,GLabel1  :string;
   Beta2S,KappaP2S,Lambda2S,GLabel2       :string;
   OrdSP1S,OrdSP2S                        :string;


   {ART     : Alpha fn(r) at time T                          range 0..N     }
   {ARTP1   : Alpha fn(r) at time T+1                        range 0..N     }
   {SRT1    : Sigma fn(r) at time T                          range 0..N     }
   {SRTsP1  : Sigma fn(r) at time T+1      (guessed)         range 0..N     }
   {SRTsP1C : Sigma fn(r) at time T+1      (calculated)      range 0..N     }
   {YVect   : 'Const' vector in Crank-Nicolson method        range 0..N     }
   {AMatrix : Matrix of Crank-Nicolson coefficients          range N* N     }
   {DataSetX: X vector used in the graphing routine          range 0..N+1   }

{-------------------------------------------------------------------------}
Function Power(Base,Pow:real):Extended;
begin
   if Pow=0  then Power:=1
   else
   if Base=0 then Power:=0
   else
   Power:=exp(Pow*ln(base));
end;


{-------------------------------------------------------------------------}
Procedure Time0_ART_and_SRTs;
{This procedure uses the initial conditions to set the alpha and sigma    }
{vectors.                                                                 }
{Note that ART[N+1], ARTP1[N+1], SRT1[N+1] and SRT1P1[N+1] are the surface}
{concentrations of the liquid and solid reactants.                        }
begin
   for ArbIndex:=0 to N do begin
      ART^[ArbIndex]:=0;
      SRT1^[ArbIndex]:=1;
      SRT2^[ArbIndex]:=1;
   end;
   ART^[N+1]:=1;
   SRT1^[N+1]:=1;
```

Code Listing for Model2D2.PAS.

```
  SRT2^[N+1]:=1;
  ARTP1^[N+1]:=1;  {Imposed boundary condition of the concentration of  }
                   {the bulk fluid remaining constant to the original  }
                   {value.                                             }

   if KappaP2=0 then for ArbIndex:=0 to maxc do
   begin
      SRT2^[ArbIndex]   :=0;
      SRT2P1^[ArbIndex] :=0;
      SRT2P1C^[ArbIndex]:=0;
   end;
end;



{-------------------------------------------------------------------------}
Procedure Guess_SRTsP1;
{This procedure provides the initial "guess" for the iteration.  It uses  }
{the previous time interval's values as the guess.                        }
begin
   for ArbIndex:=0 to (N+1) do begin
      SRT1P1^[ArbIndex]:=SRT1^[ArbIndex];
      SRT2P1^[ArbIndex]:=SRT2^[ArbIndex];
   end;
end;



{-------------------------------------------------------------------------}
Procedure ReGuess_SRTsP1;
{This procedure provides an updated "guess" for the next iteration. It    }
{uses the SRT1P1C vector as the updated guess.                            }
begin
   for ArbIndex:=0 to (N+1) do begin
      SRT1P1^[ArbIndex]:=SRT1P1C^[ArbIndex];
      SRT2P1^[ArbIndex]:=SRT2P1C^[ArbIndex];
   end;
end;



{-------------------------------------------------------------------------}
Procedure Calc_AMatrix_and_Y_vector;
{This procedure calculates the Crank-Nicolson coefficient matrix.         }
begin
   for Rows:=0 to (N+1) do
   begin
      for Cols:=0 to (N+1) do
         AMatrix^[Rows,Cols]:=0;
   end;

   AMatrix^[0,0]:=-6-DeltaE*DeltaE*(KappaP1*Power(SRT1P1^[0],OrdSP1)
                  +KappaP2*Power(SRT2P1^[0],OrdSP2))
                  -2*DeltaE*DeltaE/DeltaT;
   AMatrix^[0,1]:=6;
   YVector^[0]  :=ART^[0]*(6+DeltaE*DeltaE*(KappaP1*Power(SRT1^[0],OrdSP1)
                  +KappaP2*Power(SRT2^[0],OrdSP2))
                  -2*DeltaE*DeltaE/DeltaT)-ART^[1]*6;
```



```
   for Rows:=1 to N-1 do
   begin
      AMatrix^[Rows,Rows-1]:=Rows-1;
      AMatrix^[Rows,Rows]  :=-2*Rows-Rows*DeltaE*DeltaE
                             *(KappaP1*Power(SRT1P1^[Rows],OrdSP1)+
                             KappaP2*Power(SRT2P1^[Rows],OrdSP2))
                             -2*Rows*DeltaE*DeltaE/DeltaT;
      AMatrix^[Rows,Rows+1]:=Rows+1;
      YVector^[Rows]       :=ART^[Rows-1]*(-Rows+1)+ART^[Rows]*(2*Rows+Rows*
                             DeltaE*DeltaE
                             *(KappaP1*Power(SRT1^[Rows],OrdSP1)
                             +KappaP2*Power(SRT2^[Rows],OrdSP2))
                             -2*Rows*DeltaE*DeltaE/DeltaT)
                             +ART^[Rows+1]*(-Rows-1);

   end;

   AMatrix^[N,N-1]:=N-1;
   AMatrix^[N,N]  :=-2*N-N*DeltaE*DeltaE*(KappaP1*Power(SRT1P1^[N],OrdSP1)
                    +KappaP2*Power(SRT2P1^[N],OrdSP2))
                    -2*N*DeltaE*DeltaE/DeltaT;
   YVector^[N]    :=ART^[N-1]*(-N+1)+ART^[N]*(2*N+N*DeltaE*DeltaE*
                    (KappaP1*Power(SRT1^[N],OrdSP1)
                    +KappaP2*Power(SRT2^[N],OrdSP2))
                    -2*N*DeltaE*DeltaE/DeltaT)+
                    ART^[N+1]*(-N-1)-ARTP1^[N+1]*(N+1);

end;



{-------------------------------------------------------------------------}
Procedure CalcSRTsP1C;
{This procedure is a non-linear root finding procedure.  It uses Brent's  }
{Method to solve for the root.                                            }
begin
   if (KappaP1<>0) then begin
      if OrdSP1=1 then begin
         SRT1P1C^[ArbIndex]:=SRT1^[ArbIndex]*(1-DeltaT*KappaP1*Beta1*ART^[ArbInde
x]
                             /(2*(1-Lambda1)))/(1+DeltaT*KappaP1*Beta1*
                             ARTP1^[ArbIndex]/(2*(1-Lambda1)));
      end
      else if SRT1P1^[ArbIndex]<SConc_Tol then SRT1P1C^[ArbIndex]:=SRT1P1^[ArbInd
ex]
      else begin
         A  :=ART^[ArbIndex];
         AP1:=ARTP1^[ArbIndex];
         S  :=SRT1^[ArbIndex];
         SRT1P1C^[ArbIndex]:=BrentRoots(0.0,1.0,DeltaT,KappaP1,Beta1,Lambda1,OrdS
P1,A,AP1,
                  S,1e-8,100,ValueAtRoot,error);
      end;
   end;
   if (KappaP2<>0) then begin
      if OrdSP2=1 then begin
         SRT2P1C^[ArbIndex]:=SRT2^[ArbIndex]*(1-DeltaT*KappaP2*Beta2*ART^[ArbInde
x]
                             /(2*(1-Lambda2)))/(1+DeltaT*KappaP2*Beta2*
                             ARTP1^[ArbIndex]/(2*(1-Lambda2)));
```

```
      end
      else if SRT2P1^[ArbIndex]<SConc_Tol then SRT2P1C^[ArbIndex]:=SRT2P1^[ArbInd
ex]
      else begin
         A  :=ART^[ArbIndex];
         AP1:=ARTP1^[ArbIndex];
         S  :=SRT2^[ArbIndex];
         SRT2P1C^[ArbIndex]:=BrentRoots(0.0,1.0,DeltaT,KappaP2,Beta2,Lambda2,OrdS
P2,A,AP1,
                     S,1e-8,100,ValueAtRoot,error);
      end;
   end;
end;



{-------------------------------------------------------------------------}
Procedure Check_Convergence;
{This procedure checks whether or not the solution has converged by       }
{comparing the guessed value of SRT1P1 with a calculated value of SRT1P1. }
begin
   Norm1:=0;         {Using a norm ***                                     }
   Norm2:=0;
   for ArbIndex:=0 to (N+1) do begin
      CalcSRTsP1C;
      if (Beta1<>0) and (KappaP1<>0) then begin
         Norm1:=(SRT1P1C^[ArbIndex]-SRT1P1^[ArbIndex])*(SRT1P1C^[ArbIndex]-
                SRT1P1^[ArbIndex]);
      end
      else Norm1:=0;
      if (Beta2<>0) and (KappaP2<>0) then begin
         Norm2:=(SRT2P1C^[ArbIndex]-SRT2P1^[ArbIndex])*(SRT2P1C^[ArbIndex]-
                SRT2P1^[ArbIndex]);
      end
      else Norm2:=0;
   end;
end;



{-------------------------------------------------------------------------}
Procedure Update_ART_and_SRTs;
{This procedure updates the alpha and sigma vectors for the next iteration}
{by replacing their components with the alphaT+1 and sigmaT+1 vectors.    }
begin
   for ArbIndex:=0 to (N+1) do begin
      ART^[ArbIndex]:=ARTP1^[Arbindex];
      SRT1^[ArbIndex]:=SRT1P1^[ArbIndex];
      SRT2^[ArbIndex]:=SRT2P1^[ArbIndex];
   end;
end;



{-------------------------------------------------------------------------}
Procedure Calc_Conversion;
{This procedure calculates the fractional conversion of the particle for  }
{the time interval.  It uses the formula on pg. 13 Dixon.  The integrator }
{is the Quinn-Curtis vector integrator.                                   }
begin
```



```
   if KappaP1<>0 then begin
      for ArbIndex:=0 to (N+1) do
         InterG1^[ArbIndex]:=(1-SRT1^[ArbIndex])*ArbIndex*DeltaE*ArbIndex*DeltaE;

      IntegrateVector(InterG1^,DeltaE,0,(N+1),IntegVal);
      Conv1^[Conv_Var]:=3*(1-Lambda1)*IntegVal+Lambda1*(1-SRT1^[N+1]);
   end;
   if KappaP2<>0 then begin
      for ArbIndex:=0 to (N+1) do
         InterG2^[ArbIndex]:=(1-SRT2^[ArbIndex])*ArbIndex*DeltaE*ArbIndex*DeltaE;

      IntegrateVector(InterG2^,DeltaE,0,(N+1),IntegVal);
      Conv2^[Conv_Var]:=3*(1-Lambda2)*IntegVal+Lambda2*(1-SRT2^[N+1]);
   end;

end;



{-------------------------------------------------------------------------}
Procedure Graph1_Initialise;
begin
   initSEGraphics('F:\tp6');
   SetCurrentWindow(2);
   BorderCurrentWindow(1);
   SetAxesType(0,0);
   ScalePlotArea(0.0,0.0,1.0,1.0);
   SetXYIntercepts(0.0,0.0);
   SetColor(2);
   DrawXAxis(0.2,1);
   DrawYAxis(0.2,1);
   LabelXAxis(1,0);
   LabelYAxis(1,0);
   TitleXAxis('Dimensionless Radius');
   TitleYAxis('Dimensionless Concentration');
   TitleWindow('Model2D2');
   Str(Beta1:6:3,Beta1S);
   Str(KappaP1:6:3,KappaP1S);
   Str((DeltaT*Print_Crit):6:4,GDTS);
   Str(Lambda1:5:3,Lambda1S);
   Str(OrdSP1:5:2,OrdSP1S);
   Str(Beta2:6:3,Beta2S);
   Str(KappaP2:6:3,KappaP2S);
   Str(Lambda2:5:3,Lambda2S);
   Str(OrdSP2:5:2,OrdSP2S);
   GLabel1:=Concat(' Beta1 ',Beta1S,'; Kappa1  ',KappaP1S,';  Lambda1  ',Lambda1S
,
                   '; Order1 ',OrdSP1S,';  GDT  ',GDTS);
   GLabel2:=Concat(' Beta2 ',Beta2S,'; Kappa2  ',KappaP2S,';  Lambda2  ',Lambda2S
,
                   '; Order2 ',OrdSP2S);
   LabelGraphWindow(1,930,GLabel1,0,0);
   LabelGraphWindow(1,900,Glabel2,0,0);
   for ArbIndex:=0 to N+1 do DataSetX^[ArbIndex]:=ArbIndex*DeltaE;
   DataPoints:=N+2;
end;



{-------------------------------------------------------------------------}
```

```
Procedure Graph1_Results;
begin
   if KappaP1<>0 then begin
      for ArbIndex:=0 to (N+1) do DataSetY^[ArbIndex]:=ARTP1^[ArbIndex];
      LinePlotData(DataSetX^,DataSetY^,DataPoints,3,0);
      for ArbIndex:=0 to (N+1) do DataSetY^[ArbIndex]:=SRT1P1^[ArbIndex];
      LinePlotData(DataSetX^,DataSetY^,DataPoints,5,0);
   end;
   if KappaP2<>0 then begin
      for ArbIndex:=0 to (N+1) do DataSetY^[ArbIndex]:=ARTP1^[ArbIndex];
      LinePlotData(DataSetX^,DataSetY^,DataPoints,3,0);
      for ArbIndex:=0 to (N+1) do DataSetY^[ArbIndex]:=SRT2P1^[ArbIndex];
      LinePlotData(DataSetX^,DataSetY^,DataPoints,4,2);
   end;

end;



{-------------------------------------------------------------------------}
Procedure Graph2_Initialise_and_Draw;
begin
   SetCurrentWindow(2);
   BorderCurrentWindow(1);
   SetAxesType(0,0);
   XAxisMax:=KappaP1*Beta1*DeltaT*Repeats;
   ScalePlotArea(0.0,0.0,Gf2_MaxX,1.0);
   SetXYIntercepts(0.0,0.0);
   SetColor(2);
   DrawXAxis((Gf2_MaxX/5),1);
   DrawYAxis(0.2,1);
   LabelXAxis(1,0);
   LabelYAxis(1,0);
   TitleXAxis('Dimensionless Reaction Time (With respect to RXN 1 ) [KappaP1*Beta
1*Tau]');
   TitleYAxis('Fractional Conversion');
   TitleWindow('Model2D2');
   LabelGraphWindow(1,930,GLabel1,0,0);
   LabelGraphWindow(1,900,Glabel2,0,0);
   if KappaP1<>0 then begin
      for ArbIndex:=0 to Conv_Var do begin
         DataSetX^[ArbIndex]:=ArbIndex*Print_Crit*DeltaT*KappaP1*Beta1;
         DataSetY^[ArbIndex]:=Conv1^[ArbIndex];
      end;
      DataSetY^[0]:=0;
      DataPoints:=Conv_Var;
      LinePlotData(DataSetX^,DataSetY^,DataPoints,5,0);
   end;
   if KappaP2<>0 then begin
      for ArbIndex:=0 to Conv_Var do begin
         DataSetX^[ArbIndex]:=ArbIndex*Print_Crit*DeltaT*KappaP1*Beta1;
         DataSetY^[ArbIndex]:=Conv2^[ArbIndex];
      end;
      DataSetY^[0]:=0;
      DataPoints:=Conv_Var;
      LinePlotData(DataSetX^,DataSetY^,DataPoints,4,2);
   end;

end;
```



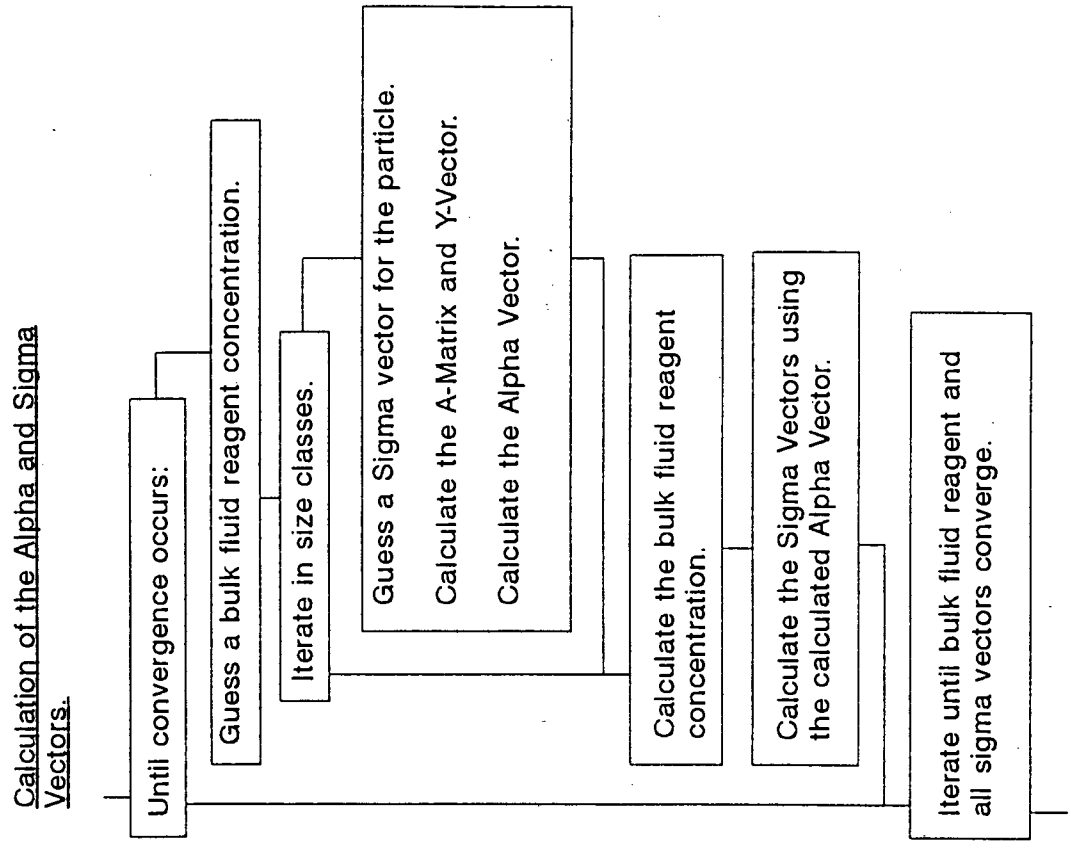
```
{-------------------------------------------------------------------------}
{MAIN PROGRAM:                                                            }

begin

new(ART);       new(ARTP1);
new(SRT1);      new(SRT1P1);      new(SRT1P1C);
new(SRT2);      new(SRT2P1);      new(SRT2P1C);
new(YVEctor);   new(DataSetX);    new(DataSetY);
new(InterG1);   new(InterG2);
new(Conv1);     new(Conv2);
new(AMatrix);   new(AInverse);



clrscr;
Graph1_Initialise;
repeats:=0;
Norm1:=1;
Norm2:=1;
Plot_Var:=0;
Conv_Var:=0;
Time0_ART_and_SRTs;
Guess_SRTsP1;
while repeats<Iterations do begin
   while (Norm1>Norm_Crit) or (Norm2>Norm_Crit) do begin
      Calc_AMatrix_and_Y_Vector;
      GaussJordan(AMatrix^,YVector^,(N+1),ARTP1^,AInverse^,AMatDet);
      Check_Convergence;
      ReGuess_SRTsP1;
   end;

   Plot_Var:=Plot_Var+1;

   if Plot_Var=Print_Crit then begin
      Graph1_Results;
      Plot_Var:=0;
      Conv_Var:=Conv_Var+1;
      Calc_Conversion;
   end;

   Update_ART_and_SRTs;
   Guess_SRTsP1;
   Norm1:=1;
   Norm2:=1;
   repeats:=repeats+1;
end;

readln(HardCopy1);

if HardCopy1=1 then begin
   ScreenDump(3,0,2,1.5,1.5,0,1,0,error);
end;

ClearWindow;

Graph2_Initialise_and_Draw;

readln(HardCopy2);
```

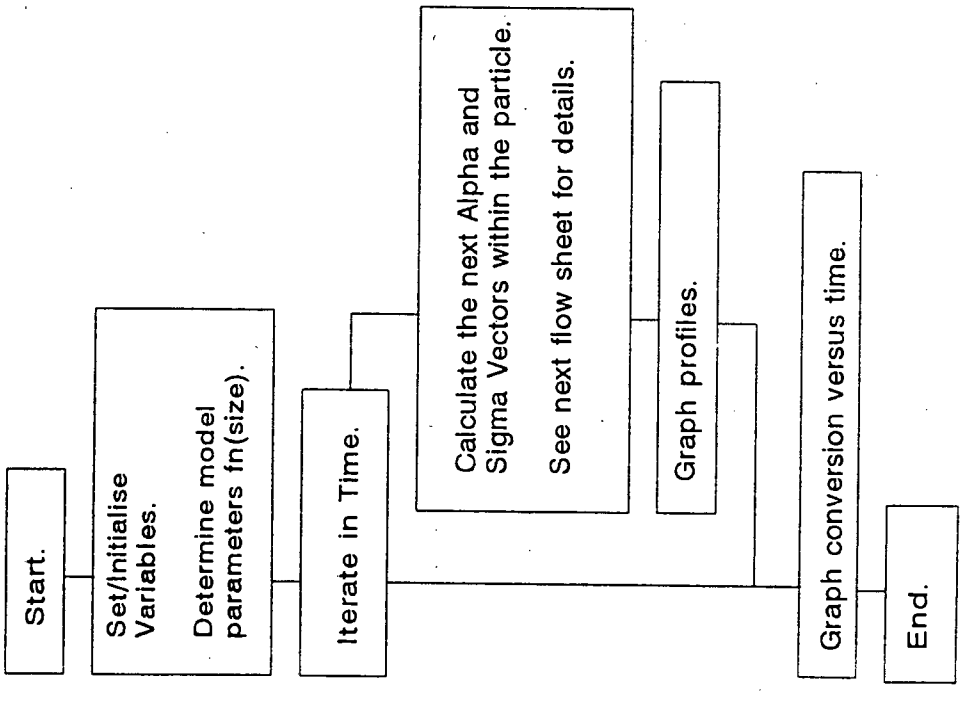
```
if HardCopy2=1 then begin
   ScreenDump(3,0,2,1.5,1.5,0,1,0,error);
end;


end.


(-------------------------------------------------------------------------)
```

Appendix IV. Solution Algorithm and Code for Model5E1.PAS and Model5E2.PAS.

Model5 represents the computer routines for the CSTR model discussed in Chapter 5.

Solution Algorithm for Model5E.





```
Program Model5E1;

{Model5E1. * This program is a tool for analyzing TCLP data.            }
{          * The program calculates the concentration profile of fluid  }
{            and solid reactants within a particle using the equations as }
{            developed by Dixon.                                        }
{          * This program only provides for solid reactant orders of 1. }

{          * Assumptions in this model include:                        }
{                - The solid reactant deposits within the particle      }
{                  resemble those on the surface.                       }
{                - The deposits would both react to the same extent     }
{                  if each were exposed to the same acid concentration  }
{                  for the same time.                                   }

{Coded:      Graham Davies.                                             }
{            Department of Chemical Engineering.                        }
{            University of Cape Town.                                   }
{            28 February 1995.                                          }
{            29 May      1995 Updated (GMD).                            }
{-------------------------------------------------------------------------}
{Declarations:                                                          }

uses crt,stdhdr,gj,graph,worlddr,segraph,integrat,M2C1BrentRoots;

const

   {***These are the parameters applicable to the reference size class.***}

   Beta1     =10.0;    {Dimensionless Stoichiometric ratio defined pg 11  }
   KappaP1   =0.12;    {Ratio of reation rate of solid reactant residing  }
                       {within the particle to porous diffusion of fluid  }
                       {into the particle. Defined pg 11 Dixon.           }

   Beta2     =0.0;
   KappaP2   =0.0;

   Voidage   =0.01;    {Voidage of the solid partilces.(Porosity)         }


   {***These are parameters with repect to the CSTR experiment.***********}

   VolLiq    =0.159;   {Volume of liquid lixiviant. (m3)                  }

   TotVolPart=0.577;  {Total volume of the solid partilces.              }

   M         =2;       {Number of size classes.                          }

   RefSizeCl =1;       {Defines the reference size class.                }


   {***These are numerical method parameters.*****************************}

   N         =19;      {Half the number of interior points.  r=0 and r=R  }
                       {not included.                                     }
   DeltaE    =1/(N+1); {Space Increment.  Calculated from 1/(N+1). (Since }
                       {                  R=i*dE and R is at point N+1.)  }

   DeltaT    =0.001;   {Time Increment.                                   }
   Norm_Crit =1e-6;    {Convergence criteria based on the norm of vector  }
```



```
                       {***                                              }
   Conv_Crit =1e-8;    {Convergence criteria for the Brent Routine.      }
   SConc_Tol =1e-4;    {Dimensionless concentration of solid below which }
                       {it is assumed to be negligible.                  }
   MaxIter   =100;     {Maximum iterations for the Brent Routine.        }

   Iterations=400;     {Iterations in time.                              }
   Print_Crit=10;      {Determined from desired Print_Delta_T/Delta_T     }
   ViewSizeCl=2;       {The concentration profiles of this sizeclass are }
                       {graphed in graph 1.  (NB  ViewSizeCl <= M.)      }


   {***Other information required.*****************************************}


   ValidDir  ='F:\TP6';{Valid directory for graphics drivers.            }

   OrdSP1    =1.0;     {Reaction order of the solid in the pores.      }
   OrdSP2    =1.0;     {For this program these need to be set at unity.}
                       {See Model5D2 for variable order reaction orders.}

type
   array1= Array[1..50,1..2] of Double;

var

   ArbIndex,SizeIndex,error                                  :Integer;
   Repeats,Plot_Var,Conv_Var,Flag1,HardCopy                  :Integer;
   Conv1,Conv2,DataSetX,DataSetY                             :^VeryLongVector;
   Beta1_k,Beta2_k,KappaP1_k,KappaP2_k,KappaS1_k,KappaS2_k   :^ShortVector;
   Lambda1_k,Lambda2_k,ContRatioV1,ContRatioV2               :^ShortVector;
   DeltaT_k                                                  :^Array1;
   ART,ARTP1,SRT1,SRT2,SRT1P1,SRT2P1,SRT1P1C,SRT2P1C         :^SqrMat;
   AMatrix,AInverse                                          :^SqrMat;
   ARTP1CVal,SumSizeData_2,NuStar                            :Extended;
   Lambda1,Lambda2,KappaS1,KappaS2                           :Extended;
   GLabel1,Glabel2                                           :String;

   SizeData : Array[1..50,1..2] of Real;

   {ART     : Alpha fn(r) at time T                    range 0..M  0..N+1}
   {ARTP1   : Alpha fn(r) at time T+1                  range 0..M  0..N+1}
   {SRT1    : Sigma fn(r) at time T                    range 0..M  0..N+1}
   {SRTsP1  : Sigma fn(r) at time T+1      (guessed)   range 0..M  0..N+1}
   {SRTsP1C : Sigma fn(r) at time T+1      (calculated) range 0..M  0..N+1}
   {YVect   : 'Const' vector in Crank-Nicolson method  range 0..N+1      }
   {AMatrix : Matrix of Crank-Nicolson coefficients    range N* N        }
   {DataSetX: X vector used in the graphing routine    range 0..N+1      }

{-------------------------------------------------------------------------}
Procedure Size_Distribution_Initialisation;

{This procedure sets up the size distribution data array. Initially the   }
{SizeData array contains radius information and fractional volume         }
{information (ie R and Vp/Vtot).  On output it contains  relative radius  }
{information and relative volume information (ie R/R_ref and Vp/Vp_ref).  }
{Ideally this information would be read in from a data file.              }

var
   RefRadius,VolRefPart :Double;
```

Code Listing for Model5E1.PAS.

```
begin
   SizeData[1,1] :=11.35e-3;          SizeData[1,2] := 1/100;
   SizeData[2,1] :=11.35e-3;          SizeData[2,2] :=99/100;
   SizeData[3,1] := 0.0e-3;           SizeData[3,2] := 0/100;
   SizeData[4,1] := 0.0e-3;           SizeData[4,2] := 0/100;
   SizeData[5,1] := 0.0e-3;           SizeData[5,2] := 0/100;
   SizeData[6,1] := 0.0e-3;           SizeData[6,2] := 0/100;
   SizeData[7,1] := 0.0e-3;           SizeData[7,2] := 0/100;
   SizeData[8,1] := 0.0e-3;           SizeData[8,2] := 0/100;
   SizeData[9,1] := 0.0e-3;           SizeData[9,2] := 0/100;
   SizeData[10,1]:= 0.0e-3;           SizeData[10,2]:= 0/100;

   VolRefPart:=TotVolPart*SizeData[RefSizeCl,2];
   RefRadius :=SizeData[RefSizeCl,1];

   SumSizeData_2:=0;

   for SizeIndex:=1 to M do
   begin
      SizeData[SizeIndex,1]:=SizeData[SizeIndex,1]/RefRadius;
      SizeData[SizeIndex,2]:=SizeData[SizeIndex,2]*TotVolPart/VolRefPart;
      SumSizeData_2:=SumSizeData_2+SizeData[SizeIndex,2];
   end;

   NuStar:=VolLiq/(Voidage*VolRefPart);

   {Ratio of volume of bulk fluid to fluid in particle pores.          }

end;


{-------------------------------------------------------------------------}
Procedure Contaminant_Location_Initialisation;

{This procedure is used to define the contaminant ratio vector (ratio of  }
{the surface contaminant concentration to bulk contaminant concentration).}
{NOTE: If any size class of particles have a surface concentration of     }
{      contaminant, then so too must the reference size class.            }


begin
   ContRatioV1^[1] :=0;          ContRatiov2^[1] :=0;
   ContRatioV1^[2] :=0;          ContRatiov2^[2] :=0;
   ContRatioV1^[3] :=0;          ContRatiov2^[3] :=0;
   ContRatioV1^[4] :=0;          ContRatiov2^[4] :=0;
   ContRatioV1^[5] :=0;          ContRatiov2^[5] :=0;
   ContRatioV1^[6] :=0;          ContRatiov2^[6] :=0;
   ContRatioV1^[7] :=0;          ContRatiov2^[7] :=0;
   ContRatioV1^[8] :=0;          ContRatiov2^[8] :=0;
   ContRatioV1^[9] :=0;          ContRatiov2^[9] :=0;
   ContRatioV1^[10]:=0;          ContRatiov2^[10]:=0;
end;


{-------------------------------------------------------------------------}
Procedure Determine_Model_Parameters_fn_Size;

{This procedure determines the model parameters each size class of        }
{particles.                                                               }
```



```
var
   Flag1,Flag2 :Integer;

begin
  Flag1:=0;
  Flag2:=0;
  for SizeIndex:=1 to M do
  begin
     Lambda1_k^[SizeIndex]:=ContRatioV1^[SizeIndex]
                           /(1+ContRatioV1^[SizeIndex]);
     Lambda2_k^[SizeIndex]:=ContRatioV2^[SizeIndex]
                           /(1+ContRatioV2^[SizeIndex]);
     if ContRatioV1^[SizeIndex]<>0 then Flag1:=Flag1+1;
     if ContRatioV2^[SizeIndex]<>0 then Flag2:=Flag2+1;
   end;

   if (((Flag1=1) and (ContRatioV1^[RefSizeCl]=0)) or
       ((Flag2=1) and (ContRatiov2^[RefSizeCl]=0))) then begin
      closesegraphics;
      writeln('Contaminant Location Violoation.');
      readln;
   end;
   Lambda1              :=Lambda1_k^[RefSizeCl];
   Lambda2              :=Lambda2_k^[RefSizeCl];
   KappaS1              :=Lambda1*KappaP1/(1-Lambda1);
   KappaS2              :=Lambda2*KappaP2/(1-Lambda2);

   for SizeIndex:=1 to M do
   begin
      DeltaT_k^[SizeIndex,1] :=DeltaT/SizeData[SizeIndex,1]
                             /SizeData[SizeIndex,1];
      Beta1_k^[SizeIndex]  :=Beta1*(1+ContRatioV1^[RefSizeCl])
                             /(1+ContRatioV1^[SizeIndex]);
      Beta2_k^[SizeIndex]  :=Beta2*(1+ContRatioV2^[RefSizeCl])
                             /(1+ContRatioV2^[SizeIndex]);
      KappaP1_k^[SizeIndex]:=KappaP1*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1];
      KappaP2_k^[SizeIndex]:=KappaP2*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1];
      {*****Note:Unity power assumption involved in next few lines*****   }
      if ContRatioV1^[SizeIndex]=0 then KappaS1_k^[SizeIndex]:=0 else
         KappaS1_k^[SizeIndex]:=KappaS1*ContRatioV1^[SizeIndex]
                             /ContRatioV1^[RefSizeCl]*SizeData[SizeIndex,1];
      if ContRatioV2^[SizeIndex]=0 then KappaS2_k^[SizeIndex]:=0 else
         KappaS2_k^[SizeIndex]:=KappaS2*ContRatioV2^[SizeIndex]
                             /ContRatioV2^[RefSizeCl]*SizeData[SizeIndex,1];

   end;
end;
{-------------------------------------------------------------------------}
Procedure Time0_ART_and_SRTs;

{This procedure uses the initial conditions to set the alpha and sigma    }
{vectors.                                                                 }
{Note that ART[N+1], ARTP1[N+1], SRT1[N+1] and SRT1P1[N+1] are the surface}
{concentrations of the liquid and solid reactants.                        }

begin
   for SizeIndex:=1 to M do
```

```
   begin
      for ArbIndex:=0 to N do
      begin
         ART^[SizeIndex,ArbIndex] :=0;
         SRT1^[SizeIndex,ArbIndex]:=1;
         SRT2^[SizeIndex,ArbIndex]:=1;
      end;
      ART^[SizeIndex,N+1] :=1;
      SRT1^[SizeIndex,N+1]:=1;
      SRT2^[SizeIndex,N+1]:=1;
   end;
   for arbIndex:=0 to maxv do begin
      Conv1^[ArbIndex]:=0;
      Conv2^[ArbIndex]:=0;
   end;
end;

{-------------------------------------------------------------------------}
Procedure Guess_SRTsP1;

{This procedure provides the initial "guess" for the iteration.  It uses  }
{the previous time interval's values as the guess.                        }

begin
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
      begin
         SRT1P1^[SizeIndex,ArbIndex]:=SRT1^[SizeIndex,ArbIndex];
         SRT2P1^[SizeIndex,ArbIndex]:=SRT2^[SizeIndex,ArbIndex];
      end;
      ARTP1^[SizeIndex,N+1]:=ART^[SizeIndex,N+1];
   end;
end;

{-------------------------------------------------------------------------}
Procedure ReGuess_SRTsP1;

{This procedure provides an updated "guess" for the next iteration. It    }
{uses the SRT1P1C vector as the updated guess.                            }

begin
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
      begin
         SRT1P1^[SizeIndex,ArbIndex]:=SRT1P1C^[SizeIndex,ArbIndex];
         SRT2P1^[SizeIndex,ArbIndex]:=SRT2P1C^[SizeIndex,ArbIndex];
      end;
      ARTP1^[SizeIndex,N+1]:=ARTP1CVal;
   end;
end;
{-------------------------------------------------------------------------}
Procedure Calc_Crank_Nicolson_Matrix(var YVector :ShortVector);

{*****Note:Unity power assumption involved in this procedure*****         }

var
   Rows,Cols                                    :Integer;
```



```
begin
   for Rows:=0 to (N+1) do
   begin
      for Cols:=0 to (N+1) do AMatrix^[Rows,Cols]:=0;
   end;

   AMatrix^[0,0]:=(-6-DeltaE*DeltaE*SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                  *(KappaP1_k^[SizeIndex]*SRT1P1^[SizeIndex,0]
                  +KappaP2_k^[SizeIndex]*SRT2P1^[SizeIndex,0])-2*DeltaE*DeltaE
                  *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                  /DeltaT_k^[SizeIndex,1]);
   AMatrix^[0,1]:=6;
   YVector[0]    :=ART^[SizeIndex,0]*(6+DeltaE*DeltaE*SizeData[SizeIndex,1]
                  *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]
                  *SRT1^[SizeIndex,0]+KappaP2_k^[SizeIndex]
                  *SRT2^[SizeIndex,0])-2*DeltaE*DeltaE*SizeData[SizeIndex,1]
                  *SizeData[SizeIndex,1]/DeltaT_k^[SizeIndex,1])
                  -ART^[SizeIndex,1]*6;

   for Rows:=1 to N-1 do
   begin
      AMatrix^[Rows,Rows-1]:=Rows-1;
      AMatrix^[Rows,Rows]  :=-2*Rows-Rows*DeltaE*DeltaE*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]*
                             SRT1P1^[SizeIndex,Rows]+KappaP2_k^[SizeIndex]*
                             SRT2P1^[SizeIndex,Rows])-2*Rows*DeltaE
                             *DeltaE*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1]/DeltaT_k^[SizeIndex,1];
      AMatrix^[Rows,Rows+1]:=Rows+1;
      YVector[Rows]        :=ART^[SizeIndex,Rows-1]*(-Rows+1)+
                             ART^[SizeIndex,Rows]*(2*Rows+Rows*DeltaE*DeltaE
                             *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                             *(KappaP1_k^[SizeIndex]*SRT1^[SizeIndex,Rows]
                             +KappaP2_k^[SizeIndex]*SRT2^[SizeIndex,Rows])
                             -2*Rows*DeltaE*DeltaE*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1]/DeltaT_k^[SizeIndex,1])
                             +ART^[SizeIndex,Rows+1]*(-Rows-1);
   end;

   AMatrix^[N,N-1]:=N-1;
   AMatrix^[N,N]  :=-2*N-N*DeltaE*DeltaE*SizeData[SizeIndex,1]
                    *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]
                    *SRT1P1^[SizeIndex,N]+KappaP2_k^[SizeIndex]
                    *SRT2P1^[SizeIndex,N])-2*N*DeltaE*DeltaE
                    *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                    /DeltaT_k^[SizeIndex,1];
   YVector[N]     :=ART^[SizeIndex,N-1]*(-N+1)+ART^[SizeIndex,N]*(2*N+
                    N*DeltaE*DeltaE*SizeData[SizeIndex,1]
                    *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]
                    *SRT1^[SizeIndex,N]+KappaP2_k^[SizeIndex]
                    *SRT2^[SizeIndex,N])-2*N*DeltaE*DeltaE
                    *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                    /DeltaT_k^[SizeIndex,1])+ART^[SizeIndex,N+1]*(-N-1)
                    -ARTP1^[SizeIndex,N+1]*(N+1);

end;

{-------------------------------------------------------------------------}
Procedure Calc_ARTP1_and_SRTsP1C;
```

```
{As it stands this procedure can cope with a variable reaction order. This}
{is due to the inclusion of the Brent Routine.                           }

var
   MassBalVal,AMatDet                              :Double;
   ValueAtRoot                                     :Real;
   OutputVector,YVector                            :ShortVector;

begin

   {First calculate the ARTP1 vectors.                                    }

   for SizeIndex:=1 to M do
   begin
      Calc_Crank_Nicolson_Matrix(YVector);
      GaussJordan(AMatrix^,YVector,(N+1),OutputVector,AInverse^,AMatDet);
      for ArbIndex:=0 to N do ARTP1^[SizeIndex,ArbIndex]:=OutputVector[ArbIndex];

   end;

   {The GaussJordan procedure calculates values for the ART vector from   }
   {point 0 to point N (although it makes use of the N+1_th point).  To   }
   {determine the N+1_th point value, make use of the mass balance of     }
   {fluid reactant in the CSTR.                                           }


   MassBalVal:=0;

   for SizeIndex:=1 to M do
   begin
      MassBalVal:=MassBalVal-SizeData[SizeIndex,2]/2*(KappaS1_k^[SizeIndex]
                  *SRT1^[SizeIndex,(N+1)]*ART^[SizeIndex,(N+1)]
                  +KappaS1_k^[SizeIndex]*SRT1P1^[SizeIndex,(N+1)]
                  *ARTP1^[SizeIndex,(N+1)]+KappaS2_k^[SizeIndex]
                  *SRT2^[SizeIndex,(N+1)]*ART^[SizeIndex,(N+1)]
                  +KappaS2_k^[SizeIndex]*SRT2P1^[SizeIndex,(N+1)]
                  *ARTP1^[SizeIndex,(N+1)])-3*SizeData[SizeIndex,2]
                  /(SizeData[SizeIndex,1]*SizeData[SizeIndex,1])/2
                  *((ART^[SizeIndex,(N+1)]+ARTP1^[SizeIndex,(N+1)])
                  -(ART^[SizeIndex,N]+ARTP1^[SizeIndex,N]))/DeltaE;

   end;

   ARTP1CVal:=ART^[1,(N+1)]+DeltaT/NuStar*MassBalVal;

   {Calculate the SRTsP1C vectors.  Code makes use of Brent's Method      }
   {(A non-linear root finding procedure) to solve for the root.          }

   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do begin
         if (Beta1_k^[SizeIndex]<>0) and (KappaP1_k^[SizeIndex]<>0) then
         begin
            if OrdSP1=1 then
            begin
               SRT1P1C^[SizeIndex,ArbIndex]:=SRT1^[SizeIndex,ArbIndex]
                       *(1-DeltaT_k^[SizeIndex,1]*KappaP1_k^[SizeIndex]
                       *Beta1_k^[SizeIndex]*ART^[SizeIndex,ArbIndex]
                       /(2*(1-Lambda1_k^[SizeIndex])))/(1+DeltaT_k^[SizeIndex,1]
                       *KappaP1_k^[SizeIndex]*Beta1_k^[SizeIndex]
```



```
                       *ARTP1^[SizeIndex,ArbIndex]/(2*(1-Lambda1_k^[SizeIndex])))
;
            end
            else if SRT1P1^[SizeIndex,ArbIndex]<SConc_Tol then
               SRT1P1C^[SizeIndex,ArbIndex]:=SRT1P1^[SizeIndex,ArbIndex]
            else
            begin
               SRT1P1C^[SizeIndex,ArbIndex]:=BrentRoots(0.0,1.0,DeltaT_k^[SizeInd
ex,1],
                       KappaP1_k^[SizeIndex],Beta1_k^[SizeIndex],
                       Lambda1_k^[SizeIndex],OrdSP1,ART^[SizeIndex,ArbIndex],
                       ARTP1^[SizeIndex,ArbIndex],SRT1^[SizeIndex,ArbIndex],
                       1e-8,100,ValueAtRoot,error);
            end;
         end;
         if (Beta2_k^[SizeIndex]<>0) and (KappaP2_k^[SizeIndex]<>0) then
         begin
            if OrdSP2=1 then
            begin
               SRT2P1C^[SizeIndex,ArbIndex]:=SRT2^[SizeIndex,ArbIndex]
                       *(1-DeltaT_k^[SizeIndex,1]*KappaP2_k^[SizeIndex]
                       *Beta2_k^[SizeIndex]*ART^[SizeIndex,ArbIndex]
                       /(2*(1-Lambda2_k^[SizeIndex])))/(1+DeltaT_k^[SizeIndex,1]
                       *KappaP2_k^[SizeIndex]*Beta2_k^[SizeIndex]
                       *ARTP1^[SizeIndex,ArbIndex]/(2*(1-Lambda2_k^[SizeIndex])))
;
            end
            else if SRT2P1^[SizeIndex,ArbIndex]<SConc_Tol then
               SRT1P1C^[SizeIndex,ArbIndex]:=SRT2P1^[SizeIndex,ArbIndex]
            else
            begin
               SRT2P1C^[SizeIndex,ArbIndex]:=BrentRoots(0.0,1.0,
                       DeltaT_k^[SizeIndex,1],KappaP2_k^[SizeIndex],
                       Beta2_k^[SizeIndex],Lambda2_k^[SizeIndex],OrdSP2,
                       ART^[SizeIndex,ArbIndex],ARTP1^[SizeIndex,ArbIndex],
                       SRT2^[SizeIndex,ArbIndex],1e-8,100,ValueAtRoot,error);
            end;
         end;
      end;
   end;
end;


{-------------------------------------------------------------------------}
Procedure Check_Convergence;

{This procedure checks whether or not the solution has converged by       }
{comparing the guessed value of SRT1P1 with a calculated value of SRT1P1. }
{Also compares the calculated value of ARTP1C with guessed value of       }
{ARTP1.                                                                   }

var
   Norm1,Norm2,Norm3                                        :Double;

begin
   Norm1:=0;          {Using a norm ***                                   }
   Norm2:=0;
   Norm3:=0;
   for SizeIndex:=1 to M do
   begin
```

```
      for ArbIndex:=0 to (N+1) do
      begin
         if (Beta1_k^[SizeIndex]<>0) and (KappaP1_k^[SizeIndex]<>0) then begin
            Norm1:=Norm1+(SRT1P1C^[SizeIndex,ArbIndex]
                  -SRT1P1^[SizeIndex,ArbIndex])*(SRT1P1C^[SizeIndex,ArbIndex]
                  -SRT1P1^[SizeIndex,ArbIndex]);
         end
         else Norm1:=0;
         if (Beta2_k^[SizeIndex]<>0) and (KappaP2_k^[SizeIndex]<>0) then begin
            Norm2:=Norm2+(SRT2P1C^[SizeIndex,ArbIndex]
                  -SRT2P1^[SizeIndex,ArbIndex])*(SRT1P1C^[SizeIndex,ArbIndex]
                  -SRT2P1^[SizeIndex,ArbIndex]);
         end
         else Norm2:=0;
      end;
   end;
   Norm3:=(ARTP1CVal-ARTP1^[1,N+1])*(ARTP1CVal-ARTP1^[1,N+1]);
   if (Norm1<Norm_Crit) and (Norm2<Norm_Crit) and (Norm3<Norm_Crit)
      then Flag1:=1;
end;

{-------------------------------------------------------------------------}
Procedure Update_ART_and_SRTs;

{This procedure updates the alpha and sigma vectors for the next iteration}
{by replacing their components with the alphaT+1 and sigmaT+1 vectors.    }

begin
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
      begin
         ART^[SizeIndex,ArbIndex] :=ARTP1^[SizeIndex,Arbindex];
         SRT1^[SizeIndex,ArbIndex]:=SRT1P1^[SizeIndex,ArbIndex];
         SRT2^[SizeIndex,ArbIndex]:=SRT2P1^[SizeIndex,ArbIndex];
      end;
   end;
end;


{-------------------------------------------------------------------------}
Procedure Calc_Conversion;

{This procedure calculates the fractional conversion of the particle for  }
{the time interval.  It uses the formula on pg. 13 Dixon.  The integrator }
{is the Quinn-Curtis vector integrator.                                   }

var
   InterG1,InterG2                                    :VeryLongVector;
   IntegVal,Conversion                                :Real;

begin
   for SizeIndex:=1 to M do
   begin
      if KappaP1_k^[SizeIndex]<>0 then
      begin
         for ArbIndex:=0 to (N+1) do
            InterG1[ArbIndex]:=(1-SRT1^[SizeIndex,ArbIndex])*ArbIndex*DeltaE
                              *ArbIndex*DeltaE;
         IntegrateVector(InterG1,DeltaE,0,(N+1),IntegVal);
```



```
         Conversion      :=3*(1-Lambda1_k^[SizeIndex])*IntegVal
                           +Lambda1_k^[SizeIndex]*(1-SRT1^[SizeIndex,N+1]);
         Conv1^[Conv_Var]:=Conv1^[Conv_Var]+Conversion*SizeData[SizeIndex,2]
                           /SumSizeData_2;
      end;

      if KappaP2_k^[SizeIndex]<>0 then
      begin
         for ArbIndex:=0 to (N+1) do
         InterG2[ArbIndex]:=(1-SRT2^[SizeIndex,ArbIndex])*ArbIndex*DeltaE
                            *ArbIndex*DeltaE;
         IntegrateVector(InterG2,DeltaE,0,(N+1),IntegVal);
         Conversion      :=3*(1-Lambda2_k^[SizeIndex])*IntegVal
                           +Lambda2_k^[SizeIndex]*(1-SRT2^[SizeIndex,N+1]);
         Conv2^[Conv_Var]:=Conv2^[Conv_Var]+Conversion*SizeData[SizeIndex,2]
                           /SumSizeData_2;
      end;
   end;
end;

{-------------------------------------------------------------------------}
Procedure Graph1_Initialise;

var

   Beta1S,KappaP1S,Lambda1S,GDTS                          :string;
   Beta2S,KappaP2S,Lambda2S                               :string;
   ViewSizeClS                                            :string;


begin
   initSEGraphics(ValidDir);
   SetCurrentWindow(2);
   BorderCurrentWindow(1);
   SetAxesType(0,0);
   ScalePlotArea(0.0,0.0,1.0,1.0);
   SetXYIntercepts(0.0,0.0);
   SetColor(2);
   DrawXAxis(0.2,1);
   DrawYAxis(0.2,1);
   LabelXAxis(1,0);
   LabelYAxis(1,0);
   TitleXAxis('Dimensionless Radius');
   TitleYAxis('Dimensionless Concentration');
   TitleWindow('Model5E1');
   Str(Beta1_k^[ViewSizeCl]:6:3,Beta1S);
   Str(KappaP1_k^[ViewSizeCl]:6:3,KappaP1S);
   Str((DeltaT*Print_Crit):6:4,GDTS);
   Str(Lambda1_k^[ViewSizeCl]:5:3,Lambda1S);
   Str(Beta2_k^[ViewSizeCl]:6:3,Beta2S);
   Str(KappaP2_k^[ViewSizeCl]:6:3,KappaP2S);
   Str(Lambda2_k^[ViewSizeCl]:5:3,Lambda2S);
   Str(ViewSizeCl:2,ViewSizeClS);
   GLabel1:=Concat(' Beta1 ',Beta1S,'; Kappa1  ',KappaP1S,';  Lambda1  ',
                   Lambda1S,';  GDT wrt RefSizeClass  ',GDTS);
   GLabel2:=Concat(' Beta2 ',Beta2S,'; Kappa2  ',KappaP2S,';  Lambda2  ',
                   Lambda2S,';   ViewClass ',ViewSizeClS);
   LabelGraphWindow(1,930,GLabel1,0,0);
   LabelGraphWindow(1,900,Glabel2,0,0);
end;
```

```
{-------------------------------------------------------------------------}
Procedure Graph1_Results;

begin
   if ViewSizeCl>M then
   begin
      Closesegraphics;
      writeln('Graph 1 Draw ERROR');
      writeln('You have instucted the Graphing Routine to graph the conversion');

      writeln('curves of a size class which does not exist.');
      writeln('Respecify = ViewSizeClass in Declarations section');
      readln;
   end;
   SizeIndex:=ViewSizeCl;
   for ArbIndex:=0 to (N+1) do DataSetX^[ArbIndex]:=ArbIndex*DeltaE;
   if KappaP1_k^[SizeIndex]<>0 then begin
      for ArbIndex:=0 to (N+1) do
         DataSetY^[ArbIndex]:=ARTP1^[SizeIndex,ArbIndex];
      LinePlotData(DataSetX^,DataSetY^,(N+2),3,0);
      for ArbIndex:=0 to (N+1) do
         DataSetY^[ArbIndex]:=SRT1P1^[SizeIndex,ArbIndex];
      LinePlotData(DataSetX^,DataSetY^,(N+2),5,1);
   end;
   if KappaP2_k^[SizeIndex]<>0 then begin
      for ArbIndex:=0 to (N+1) do
         DataSetY^[ArbIndex]:=ARTP1^[SizeIndex,ArbIndex];
      LinePlotData(DataSetX^,DataSetY^,(N+2),3,1);
      for ArbIndex:=0 to (N+1) do
         DataSetY^[ArbIndex]:=SRT2P1^[SizeIndex,ArbIndex];
      LinePlotData(DataSetX^,DataSetY^,(N+2),4,2);
   end;

end;



{-------------------------------------------------------------------------}
Procedure Graph2_Initialise_and_Draw;

var
   XAxisMax                                                :Real;


begin
   SetCurrentWindow(2);
   BorderCurrentWindow(1);
   SetAxesType(0,0);
   XAxisMax:=5;                                (*****DeltaT*Repeats;****)
   ScalePlotArea(0.0,0.0,XAxisMax,1.0);
   SetXYIntercepts(0.0,0.0);
   SetColor(2);
   DrawXAxis((XAxisMax/5),1);
   DrawYAxis(0.2,1);
   LabelXAxis(1,0);
   LabelYAxis(1,0);
   TitleXAxis('Dimensionless Reaction Time (WRT Reference Particle)');
   TitleYAxis('Fractional Conversion');
   TitleWindow('Model5E1');
```



```
   LabelGraphWindow(1,930,GLabel1,0,0);
   LabelGraphWindow(1,900,Glabel2,0,0);
   if KappaP1_k^[SizeIndex]<>0 then begin
      for ArbIndex:=0 to Conv_Var do begin
         DataSetX^[ArbIndex]:=ArbIndex*Print_Crit*DeltaT;
         DataSetY^[ArbIndex]:=Conv1^[ArbIndex];
      end;
      DataSetY^[0]:=0;
      LinePlotData(DataSetX^,DataSetY^,Conv_Var,5,0);
   end;
   if KappaP2_k^[SizeIndex]<>0 then begin
      for ArbIndex:=0 to Conv_Var do begin
         DataSetX^[ArbIndex]:=ArbIndex*Print_Crit*DeltaT;
         DataSetY^[ArbIndex]:=Conv2^[ArbIndex];
      end;
      DataSetY^[0]:=0;
      LinePlotData(DataSetX^,DataSetY^,Conv_Var,4,0);
   end;

end;

{-------------------------------------------------------------------------}
begin
   new(Beta1_k);          new(Beta2_k);
   new(KappaP1_k);        new(KappaP2_k);
   new(KappaS1_k);        new(KappaS2_k);          new(DeltaT_k);
   new(Lambda1_k);        new(Lambda2_k);
   new(ContRatioV1);      new(ContRatioV2);
   new(ART);              new(ARTP1);
   new(SRT1);             new(SRT1P1);             new(SRT1P1C);
   new(SRT2);             new(SRT2P1);             new(SRT2P1C);
   new(Conv1);            new(Conv2);
   new(DataSetX);         new(DataSetY);
   new(AMatrix);          new(AInverse);

   clrscr;

   Size_Distribution_Initialisation;
   Contaminant_Location_Initialisation;
   Determine_Model_Parameters_fn_Size;
   Graph1_Initialise;

   Repeats:=0;
   Flag1  :=0;
   Plot_Var:=0;
   Conv_Var:=0;

   Time0_ART_and_SRTs;
   Guess_SRTsP1;

   while Repeats<Iterations do
   begin
      while (Flag1=0) do
      begin
         Calc_ARTP1_and_SRTsP1C;
         Check_Convergence;
         ReGuess_SRTsP1;
      end;

      Plot_Var:=Plot_Var+1;
```

```
      if Plot_Var=Print_Crit then
      begin
         Plot_Var:=0;
         Conv_Var:=Conv_Var+1;
         Graph1_Results;
         Calc_Conversion;
      end;

      Update_ART_and_SRTs;
      Guess_SRTsP1;
      Flag1:=0;
      Repeats:=Repeats+1;

  end;

readln(HardCopy);
if HardCopy=1 then ScreenDump(3,0,2,1.5,1.5,0,1,0,error);

ClearWindow;

Graph2_Initialise_and_Draw;
readln(HardCopy);
if HardCopy=1 then ScreenDump(3,0,2,1.5,1.5,0,1,0,error);


closesegraphics;


end.
```

```
Program Model5E2;

{Model5E2. * This program is a tool for analyzing TCLP data.              }
{          * The program calculates the concentration profile of fluid    }
{            and solid reactants within a particle using the equations as }
{            developed by Dixon.                                          }
{          * This program only provides for variable solid reactant       }
{            orders.                                                      }

{          * Assumptions in this model include:                           }
{                - The solid reactant deposits within the particle        }
{                  resemble those on the surface.                         }
{                - The deposits would both react to the same extent       }
{                  if each were exposed to the same acid concentration    }
{                  for the same time.                                     }

{Coded:      Graham Davies.                                               }
{            Department of Chemical Engineering.                          }
{            University of Cape Town.                                     }
{            28 February 1995.                                            }
{            29 May      1995 Updated (GMD).                              }
{-------------------------------------------------------------------------}
{Declarations:                                                            }

uses crt,stdhdr,gj,graph,worlddr,segraph,integrat,M2C1BrentRoots;

const

   {***These are the parameters applicable to the reference size class.***}

   Beta1     =10.0;    {Dimensionless Stoichiometric ratio defined pg 11  }
   KappaP1   =0.12;    {Ratio of reation rate of solid reactant residing  }
                       {within the particle to porous diffusion of fluid  }
                       {into the particle. Defined pg 11 Dixon.           }
   OrdSP1    =1.0;     {Reaction order of the solid reactants within the  }
                       {pores.                                            }

   Beta2     =0.0;
   KappaP2   =0.0;
   OrdSP2    =1.0;

   Voidage   =0.01;    {Voidage of the solid partilces.(Porosity)         }


   {***These are parameters with repect to the CSTR experiment.***********}

   VolLiq    =0.159;   {Volume of liquid lixiviant. (m3)                  }

   TotVolPart=0.577;  {Total volume of the solid partilces.              }

   M         =2;       {Number of size classes.                          }

   RefSizeCl =1;       {Defines the reference size class.                }


   {***These are numerical method parameters.*****************************}

   N         =19;      {Half the number of interior points.  r=0 and r=R  }
                       {not included.                                     }
   DeltaE    =1/(N+1); {Space Increment.  Calculated from 1/(N+1). (Since }
```



```
                       {                R=i*dE and R is at point N+1.)    }

   DeltaT    =0.001;   {Time Increment.                                   }
   Norm_Crit =1e-6;    {Convergence criteria based on the norm of vector  }
                       {***                                               }
   Conv_Crit =1e-8;    {Convergence criteria for the Brent Routine.       }
   SConc_Tol =1e-4;    {Dimensionless concentration of solid below which  }
                       {it is assumed to be negligible.                   }
   MaxIter   =100;     {Maximum iterations for the Brent Routine.         }


   Iterations=100;     {Iterations in time.                               }
   Print_Crit=10;      {Determined from desired Print_Delta_T/Delta_T      }
   ViewSizeCl=2;       {The concentration profiles of this sizeclass are  }
                       {graphed in graph 1.  (NB  ViewSizeCl <= M.)       }


   {***Other information required.*****************************************}


   ValidDir  ='F:\TP6';{Valid directory for graphics drivers.             }


type
   array1= Array[1..50,1..2] of Double;

var

   ArbIndex,SizeIndex,error                                  :Integer;
   Repeats,Plot_Var,Conv_Var,Flag1,HardCopy                  :Integer;
   Conv1,Conv2,DataSetX,DataSetY                             :^VeryLongVector;
   Beta1_k,Beta2_k,KappaP1_k,KappaP2_k,KappaS1_k,KappaS2_k   :^ShortVector;
   Lambda1_k,Lambda2_k,ContRatioV1,ContRatioV2               :^ShortVector;
   DeltaT_k                                                  :^Array1;
   ART,ARTP1,SRT1,SRT2,SRT1P1,SRT2P1,SRT1P1C,SRT2P1C         :^SqrMat;
   AMatrix,AInverse                                          :^SqrMat;
   ARTP1CVal,SumSizeData_2,NuStar                            :Extended;
   Lambda1,Lambda2,KappaS1,KappaS2                           :Extended;
   GLabel1,Glabel2                                           :String;

   SizeData : Array[1..50,1..2] of Real;

   {ART     : Alpha fn(r) at time T                          range 0..M  0..N+1}
   {ARTP1   : Alpha fn(r) at time T+1                        range 0..M  0..N+1}
   {SRT1    : Sigma fn(r) at time T                          range 0..M  0..N+1}
   {SRTsP1  : Sigma fn(r) at time T+1      (guessed)         range 0..M  0..N+1}
   {SRTsP1C : Sigma fn(r) at time T+1      (calculated)      range 0..M  0..N+1}
   {YVect   : 'Const' vector in Crank-Nicolson method        range 0..N+1      }
   {AMatrix : Matrix of Crank-Nicolson coefficients          range N* N        }
   {DataSetX: X vector used in the graphing routine          range 0..N+1      }

{-------------------------------------------------------------------------}
Function Power(Base,Pow:real):Extended;
begin
   if Pow=0 then Power:=1
   else
   if Base=0 then Power:=0
   else
   Power:=exp(Pow*ln(base));
```

Code Listing for Model5E2.PAS.

```
end;


{-------------------------------------------------------------------------}
Procedure Size_Distribution_Initialisation;

{This procedure sets up the size distribution data array. Initially the   }
{SizeData array contains radius information and fractional volume          }
{information (ie R and Vp/Vtot).  On output it contains  relative radius   }
{information and relative volume information (ie R/R_ref and Vp/Vp_ref).   }
{Ideally this information would be read in from a data file.               }

var
   RefRadius,VolRefPart :Double;

begin
   SizeData[1,1] :=11.35e-3;          SizeData[1,2] := 1/100;
   SizeData[2,1] :=11.35e-3;          SizeData[2,2] :=99/100;
   SizeData[3,1] := 0.0e-3;           SizeData[3,2] := 0/100;
   SizeData[4,1] := 0.0e-3;           SizeData[4,2] := 0/100;
   SizeData[5,1] := 0.0e-3;           SizeData[5,2] := 0/100;
   SizeData[6,1] := 0.0e-3;           SizeData[6,2] := 0/100;
   SizeData[7,1] := 0.0e-3;           SizeData[7,2] := 0/100;
   SizeData[8,1] := 0.0e-3;           SizeData[8,2] := 0/100;
   SizeData[9,1] := 0.0e-3;           SizeData[9,2] := 0/100;
   SizeData[10,1]:= 0.0e-3;           SizeData[10,2]:= 0/100;

   VolRefPart:=TotVolPart*SizeData[RefSizeCl,2];
   RefRadius :=SizeData[RefSizeCl,1];

   SumSizeData_2:=0;

   for SizeIndex:=1 to M do
   begin
      SizeData[SizeIndex,1]:=SizeData[SizeIndex,1]/RefRadius;
      SizeData[SizeIndex,2]:=SizeData[SizeIndex,2]*TotVolPart/VolRefPart;
      SumSizeData_2:=SumSizeData_2+SizeData[SizeIndex,2];
   end;

   NuStar:=VolLiq/(Voidage*VolRefPart);

   {Ratio of volume of bulk fluid to fluid in particle pores.             }

end;


{-------------------------------------------------------------------------}
Procedure Contaminant_Location_Initialisation;

{This procedure is used to define the contaminant ratio vector (ratio of  }
{the surface contaminant concentration to bulk contaminant concentration).}
{NOTE: If any size class of particles have a surface concentration of     }
{      contaminant, then so too must the reference size class.            }


begin
   ContRatioV1^[1] :=0;          ContRatiov2^[1] :=0;
   ContRatioV1^[2] :=0;          ContRatiov2^[2] :=0;
   ContRatioV1^[3] :=0;          ContRatiov2^[3] :=0;
   ContRatioV1^[4] :=0;          ContRatiov2^[4] :=0;
```



```
   ContRatioV1^[5] :=0;          ContRatiov2^[5] :=0;
   ContRatioV1^[6] :=0;          ContRatiov2^[6] :=0;
   ContRatioV1^[7] :=0;          ContRatiov2^[7] :=0;
   ContRatioV1^[8] :=0;          ContRatiov2^[8] :=0;
   ContRatioV1^[9] :=0;          ContRatiov2^[9] :=0;
   ContRatioV1^[10]:=0;          ContRatiov2^[10]:=0;
end;


{-------------------------------------------------------------------------}
Procedure Determine_Model_Parameters_fn_Size;

{This procedure determines the model parameters each size class of        }
{particles.                                                               }

var
   Flag1,Flag2 :Integer;

begin
  Flag1:=0;
  Flag2:=0;
  for SizeIndex:=1 to M do
  begin
     Lambda1_k^[SizeIndex]:=ContRatioV1^[SizeIndex]
                           /(1+ContRatioV1^[SizeIndex]);
     Lambda2_k^[SizeIndex]:=ContRatioV2^[SizeIndex]
                           /(1+ContRatioV2^[SizeIndex]);
     if ContRatioV1^[SizeIndex]<>0 then Flag1:=Flag1+1;
     if ContRatioV2^[SizeIndex]<>0 then Flag2:=Flag2+1;
   end;

   if (((Flag1=1) and (ContRatioV1^[RefSizeCl]=0)) or
       ((Flag2=1) and (ContRatiov2^[RefSizeCl]=0))) then begin
      closesegraphics;
      writeln('Contaminant Location Violoation.');
      readln;
   end;
   Lambda1              :=Lambda1_k^[RefSizeCl];
   Lambda2              :=Lambda2_k^[RefSizeCl];
   KappaS1              :=Lambda1*KappaP1/(1-Lambda1);
   KappaS2              :=Lambda2*KappaP2/(1-Lambda2);

   for SizeIndex:=1 to M do
   begin
      DeltaT_k^[SizeIndex,1] :=DeltaT/SizeData[SizeIndex,1]
                             /SizeData[SizeIndex,1];
      Beta1_k^[SizeIndex]  :=Beta1*(1+ContRatioV1^[RefSizeCl])
                             /(1+ContRatioV1^[SizeIndex]);
      Beta2_k^[SizeIndex]  :=Beta2*(1+ContRatioV2^[RefSizeCl])
                             /(1+ContRatioV2^[SizeIndex]);
      KappaP1_k^[SizeIndex]:=KappaP1*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1];
      KappaP2_k^[SizeIndex]:=KappaP2*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1];
      if ContRatioV1^[SizeIndex]=0 then KappaS1_k^[SizeIndex]:=0 else
         KappaS1_k^[SizeIndex]:=KappaS1*Power((ContRatioV1^[SizeIndex]
                             /ContRatioV1^[RefSizeCl]),OrdSP1)
                             *SizeData[SizeIndex,1];
      if ContRatioV2^[SizeIndex]=0 then KappaS2_k^[SizeIndex]:=0 else
         KappaS2_k^[SizeIndex]:=KappaS2*Power((ContRatioV2^[SizeIndex]
```

```
                              /ContRatioV2^[RefSizeCl]),OrdSP2)
                              *SizeData[SizeIndex,1];

   end;
end;
{-------------------------------------------------------------------------}
Procedure Time0_ART_and_SRTs;

{This procedure uses the initial conditions to set the alpha and sigma    }
{vectors.                                                                 }
{Note that ART[N+1], ARTP1[N+1], SRT1[N+1] and SRT1P1[N+1] are the surface}
{concentrations of the liquid and solid reactants.                        }

begin
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to N do
      begin
         ART^[SizeIndex,ArbIndex] :=0;
         SRT1^[SizeIndex,ArbIndex]:=1;
         SRT2^[SizeIndex,ArbIndex]:=1;
      end;
      ART^[SizeIndex,N+1] :=1;
      SRT1^[SizeIndex,N+1]:=1;
      SRT2^[SizeIndex,N+1]:=1;
   end;
   for arbIndex:=0 to maxv do begin
      Conv1^[ArbIndex]:=0;
      Conv2^[ArbIndex]:=0;
   end;
end;

{-------------------------------------------------------------------------}
Procedure Guess_SRTsP1;

{This procedure provides the initial "guess" for the iteration.  It uses  }
{the previous time interval's values as the guess.                        }

begin
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
      begin
         SRT1P1^[SizeIndex,ArbIndex]:=SRT1^[SizeIndex,ArbIndex];
         SRT2P1^[SizeIndex,ArbIndex]:=SRT2^[SizeIndex,ArbIndex];
      end;
      ARTP1^[SizeIndex,N+1]:=ART^[SizeIndex,N+1];
   end;
end;

{-------------------------------------------------------------------------}
Procedure ReGuess_SRTsP1;

{This procedure provides an updated "guess" for the next iteration. It    }
{uses the SRT1P1C vector as the updated guess.                            }

begin
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
```



```
      begin
         SRT1P1^[SizeIndex,ArbIndex]:=SRT1P1C^[SizeIndex,ArbIndex];
         SRT2P1^[SizeIndex,ArbIndex]:=SRT2P1C^[SizeIndex,ArbIndex];
      end;
      ARTP1^[SizeIndex,N+1]:=ARTP1CVal;
   end;
end;
{-------------------------------------------------------------------------}
Procedure Calc_Crank_Nicolson_Matrix(var YVector :ShortVector);

var
   Rows,Cols                                        :Integer;

begin
   for Rows:=0 to (N+1) do
   begin
      for Cols:=0 to (N+1) do AMatrix^[Rows,Cols]:=0;
   end;

   AMatrix^[0,0]:=(-6-DeltaE*DeltaE*SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                  *(KappaP1_k^[SizeIndex]*Power(SRT1P1^[SizeIndex,0],OrdSP1)
                  +KappaP2_k^[SizeIndex]*Power(SRT2P1^[SizeIndex,0],OrdSP2))
                  -2*DeltaE*DeltaE*SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                  /DeltaT_k^[SizeIndex,1]);
   AMatrix^[0,1]:=6;
   YVector[0]   :=ART^[SizeIndex,0]*(6+DeltaE*DeltaE*SizeData[SizeIndex,1]
                  *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]
                  *Power(SRT1^[SizeIndex,0],OrdSP1)+KappaP2_k^[SizeIndex]
                  *Power(SRT2^[SizeIndex,0],OrdSP2))
                  -2*DeltaE*DeltaE*SizeData[SizeIndex,1]
                  *SizeData[SizeIndex,1]/DeltaT_k^[SizeIndex,1])
                  -ART^[SizeIndex,1]*6;

   for Rows:=1 to N-1 do
   begin
      AMatrix^[Rows,Rows-1]:=Rows-1;
      AMatrix^[Rows,Rows]  :=-2*Rows-Rows*DeltaE*DeltaE*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]
                             *Power(SRT1P1^[SizeIndex,Rows],OrdSP1)
                             +KappaP2_k^[SizeIndex]
                             *Power(SRT2P1^[SizeIndex,Rows],OrdSP2))
                             -2*Rows*DeltaE*DeltaE*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1]/DeltaT_k^[SizeIndex,1];
      AMatrix^[Rows,Rows+1]:=Rows+1;
      YVector[Rows]        :=ART^[SizeIndex,Rows-1]*(-Rows+1)+
                             ART^[SizeIndex,Rows]*(2*Rows+Rows*DeltaE*DeltaE
                             *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                             *(KappaP1_k^[SizeIndex]
                             *Power(SRT1^[SizeIndex,Rows],OrdSP1)
                             +KappaP2_k^[SizeIndex]
                             *Power(SRT2^[SizeIndex,Rows],OrdSP2))
                             -2*Rows*DeltaE*DeltaE*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1]/DeltaT_k^[SizeIndex,1])
                             +ART^[SizeIndex,Rows+1]*(-Rows-1);
   end;

   AMatrix^[N,N-1]:=N-1;
   AMatrix^[N,N]  :=-2*N-N*DeltaE*DeltaE*SizeData[SizeIndex,1]
                    *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]
                    *Power(SRT1P1^[SizeIndex,N],OrdSP1)+KappaP2_k^[SizeIndex]
```

```
                    *Power(SRT2P1^[SizeIndex,N],OrdSP2))-2*N*DeltaE*DeltaE
                    *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                    /DeltaT_k^[SizeIndex,1];
   YVector[N]      :=ART^[SizeIndex,N-1]*(-N+1)+ART^[SizeIndex,N]*(2*N+
                    N*DeltaE*DeltaE*SizeData[SizeIndex,1]
                    *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]
                    *Power(SRT1^[SizeIndex,N],OrdSP1)+KappaP2_k^[SizeIndex]
                    *Power(SRT2^[SizeIndex,N],OrdSP2))-2*N*DeltaE*DeltaE
                    *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                    /DeltaT_k^[SizeIndex,1])+ART^[SizeIndex,N+1]*(-N-1)
                    -ARTP1^[SizeIndex,N+1]*(N+1);

end;

{-------------------------------------------------------------------------}
Procedure Calc_ARTP1_and_SRTsP1C;

var
   MassBalVal,AMatDet                          :Double;
   ValueAtRoot                                 :Real;
   OutputVector,YVector                        :ShortVector;

begin
   {First calculate the ARTP1 vectors.                                    }

   for SizeIndex:=1 to M do
   begin
      Calc_Crank_Nicolson_Matrix(YVector);
      GaussJordan(AMatrix^,YVector,(N+1),OutputVector,AInverse^,AMatDet);
      for ArbIndex:=0 to N do ARTP1^[SizeIndex,ArbIndex]:=OutputVector[ArbIndex];

   end;

   {The GaussJordan procedure calculates values for the ART vector from   }
   {point 0 to point N (although it makes use of the N+1_th point).  To   }
   {determine the N+1_th point value, make use of the mass balance of     }
   {fluid reactant in the CSTR.                                           }


   MassBalVal:=0;

   for SizeIndex:=1 to M do
   begin
      MassBalVal:=MassBalVal-SizeData[SizeIndex,2]/2*(KappaS1_k^[SizeIndex]
                  *SRT1^[SizeIndex,(N+1)]*ART^[SizeIndex,(N+1)]
                  +KappaS1_k^[SizeIndex]*SRT1P1^[SizeIndex,(N+1)]
                  *ARTP1^[SizeIndex,(N+1)]+KappaS2_k^[SizeIndex]
                  *SRT2^[SizeIndex,(N+1)]*ART^[SizeIndex,(N+1)]
                  +KappaS2_k^[SizeIndex]*SRT2P1^[SizeIndex,(N+1)]
                  *ARTP1^[SizeIndex,(N+1)])-3*SizeData[SizeIndex,2]
                  /(SizeData[SizeIndex,1]*SizeData[SizeIndex,1])/2
                  *((ART^[SizeIndex,(N+1)]+ARTP1^[SizeIndex,(N+1)])
                  -(ART^[SizeIndex,N]+ARTP1^[SizeIndex,N]))/DeltaE;

   end;

   ARTP1CVal:=ART^[1,(N+1)]+DeltaT/NuStar*MassBalVal;

   {Calculate the SRTsP1C vectors.  Code makes use of Brent's Method      }
   {(A non-linear root finding procedure) to solve for the root.          }
```



```
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do begin
         if (KappaP1_k^[SizeIndex]=0) then SRT1P1C^[SizeIndex,ArbIndex]:=0 else
         begin
            if OrdSP1=1 then
            begin
               SRT1P1C^[SizeIndex,ArbIndex]:=SRT1^[SizeIndex,ArbIndex]
                       *(1-DeltaT_k^[SizeIndex,1]*KappaP1_k^[SizeIndex]
                       *Beta1_k^[SizeIndex]*ART^[SizeIndex,ArbIndex]
                       /(2*(1-Lambda1_k^[SizeIndex])))/(1+DeltaT_k^[SizeIndex,1]
                       *KappaP1_k^[SizeIndex]*Beta1_k^[SizeIndex]
                       *ARTP1^[SizeIndex,ArbIndex]/(2*(1-Lambda1_k^[SizeIndex])))
;
            end
            else if SRT1P1^[SizeIndex,ArbIndex]<SConc_Tol then
               SRT1P1C^[SizeIndex,ArbIndex]:=SRT1P1^[SizeIndex,ArbIndex]
            else
            begin
               SRT1P1C^[SizeIndex,ArbIndex]:=BrentRoots(0.0,1.0,DeltaT_k^[SizeInd
ex,1],
                       KappaP1_k^[SizeIndex],Beta1_k^[SizeIndex],
                       Lambda1_k^[SizeIndex],OrdSP1,ART^[SizeIndex,ArbIndex],
                       ARTP1^[SizeIndex,ArbIndex],SRT1^[SizeIndex,ArbIndex],
                       1e-8,100,ValueAtRoot,error);
            end;
         end;
         if (KappaP2_k^[SizeIndex]=0) then SRT2P1C^[SizeIndex,ArbIndex]:=0 else
         begin
            if OrdSP2=1 then
            begin
               SRT2P1C^[SizeIndex,ArbIndex]:=SRT2^[SizeIndex,ArbIndex]
                       *(1-DeltaT_k^[SizeIndex,1]*KappaP2_k^[SizeIndex]
                       *Beta2_k^[SizeIndex]*ART^[SizeIndex,ArbIndex]
                       /(2*(1-Lambda2_k^[SizeIndex])))/(1+DeltaT_k^[SizeIndex,1]
                       *KappaP2_k^[SizeIndex]*Beta2_k^[SizeIndex]
                       *ARTP1^[SizeIndex,ArbIndex]/(2*(1-Lambda2_k^[SizeIndex])))
;
            end
            else if SRT2P1^[SizeIndex,ArbIndex]<SConc_Tol then
               SRT1P1C^[SizeIndex,ArbIndex]:=SRT2P1^[SizeIndex,ArbIndex]
            else
            begin
               SRT2P1C^[SizeIndex,ArbIndex]:=BrentRoots(0.0,1.0,
                       DeltaT_k^[SizeIndex,1],KappaP2_k^[SizeIndex],
                       Beta2_k^[SizeIndex],Lambda2_k^[SizeIndex],OrdSP2,
                       ART^[SizeIndex,ArbIndex],ARTP1^[SizeIndex,ArbIndex],
                       SRT2^[SizeIndex,ArbIndex],1e-8,100,ValueAtRoot,error);
            end;
         end;
      end;
   end;
end;


{-------------------------------------------------------------------------}
Procedure Check_Convergence;

{This procedure checks whether or not the solution has converged by       }
```

```
{comparing the guessed value of SRT1P1 with a calculated value of SRT1P1. }
{Also compares the calculated value of ARTP1C with guessed value of       }
{ARTP1.                                                                   }

var
   Norm1,Norm2,Norm3                                          :Double;

begin
   Norm1:=0;          {Using a norm ***                                   }
   Norm2:=0;
   Norm3:=0;
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
      begin
         if (Beta1_k^[SizeIndex]<>0) and (KappaP1_k^[SizeIndex]<>0) then begin
            Norm1:=Norm1+(SRT1P1C^[SizeIndex,ArbIndex]
                  -SRT1P1^[SizeIndex,ArbIndex])*(SRT1P1C^[SizeIndex,ArbIndex]
                  -SRT1P1^[SizeIndex,ArbIndex]);
         end
         else Norm1:=0;
         if (Beta2_k^[SizeIndex]<>0) and (KappaP2_k^[SizeIndex]<>0) then begin
            Norm2:=Norm2+(SRT2P1C^[SizeIndex,ArbIndex]
                  -SRT2P1^[SizeIndex,ArbIndex])*(SRT1P1C^[SizeIndex,ArbIndex]
                  -SRT2P1^[SizeIndex,ArbIndex]);
         end
         else Norm2:=0;
      end;
   end;
   Norm3:=(ARTP1CVal-ARTP1^[1,N+1])*(ARTP1CVal-ARTP1^[1,N+1]);
   if (Norm1<Norm_Crit) and (Norm2<Norm_Crit) and (Norm3<Norm_Crit)
      then Flag1:=1;
end;

{-------------------------------------------------------------------------}
Procedure Update_ART_and_SRTs;

{This procedure updates the alpha and sigma vectors for the next iteration}
{by replacing their components with the alphaT+1 and sigmaT+1 vectors.    }

begin
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
      begin
         ART^[SizeIndex,ArbIndex] :=ARTP1^[SizeIndex,Arbindex];
         SRT1^[SizeIndex,ArbIndex]:=SRT1P1^[SizeIndex,ArbIndex];
         SRT2^[SizeIndex,ArbIndex]:=SRT2P1^[SizeIndex,ArbIndex];
      end;
   end;
end;


{-------------------------------------------------------------------------}
Procedure Calc_Conversion;

{This procedure calculates the fractional conversion of the particle for  }
{the time interval.  It uses the formula on pg. 13 Dixon.  The integrator }
{is the Quinn-Curtis vector integrator.                                   }
```



```
var
   InterG1,InterG2                                            :VeryLongVector;
   IntegVal,Conversion                                        :Real;

begin
   for SizeIndex:=1 to M do
   begin
      if KappaP1_k^[SizeIndex]<>0 then
      begin
         for ArbIndex:=0 to (N+1) do
            InterG1[ArbIndex]:=(1-SRT1^[SizeIndex,ArbIndex])*ArbIndex*DeltaE
                               *ArbIndex*DeltaE;
         IntegrateVector(InterG1,DeltaE,0,(N+1),IntegVal);
         Conversion      :=3*(1-Lambda1_k^[SizeIndex])*IntegVal
                           +Lambda1_k^[SizeIndex]*(1-SRT1^[SizeIndex,N+1]);
         Conv1^[Conv_Var]:=Conv1^[Conv_Var]+Conversion*SizeData[SizeIndex,2]
                           /SumSizeData_2;
      end;

      if KappaP2_k^[SizeIndex]<>0 then
      begin
         for ArbIndex:=0 to (N+1) do
         InterG2[ArbIndex]:=(1-SRT2^[SizeIndex,ArbIndex])*ArbIndex*DeltaE
                            *ArbIndex*DeltaE;
         IntegrateVector(InterG2,DeltaE,0,(N+1),IntegVal);
         Conversion      :=3*(1-Lambda2_k^[SizeIndex])*IntegVal
                           +Lambda2_k^[SizeIndex]*(1-SRT2^[SizeIndex,N+1]);
         Conv2^[Conv_Var]:=Conv2^[Conv_Var]+Conversion*SizeData[SizeIndex,2]
                           /SumSizeData_2;
      end;
   end;
end;

{-------------------------------------------------------------------------}
Procedure Graph1_Initialise;

var

   Beta1S,KappaP1S,Lambda1S,GDTS                              :string;
   Beta2S,KappaP2S,Lambda2S                                   :string;
   ViewSizeCls                                                :string;


begin
   initSEGraphics(ValidDir);
   SetCurrentWindow(2);
   BorderCurrentWindow(1);
   SetAxesType(0,0);
   ScalePlotArea(0.0,0.0,1.0,1.0);
   SetXYIntercepts(0.0,0.0);
   SetColor(2);
   DrawXAxis(0.2,1);
   DrawYAxis(0.2,1);
   LabelXAxis(1,0);
   LabelYAxis(1,0);
   TitleXAxis('Dimensionless Radius');
   TitleYAxis('Dimensionless Concentration');
   TitleWindow('Model5E2');
   Str(Beta1_k^[ViewSizeCl]:6:3,Beta1S);
   Str(KappaP1_k^[ViewSizeCl]:6:3,KappaP1S);
```

```
   Str((DeltaT*Print_Crit):6:4,GDTS);
   Str(Lambda1_k^[ViewSizeCl]:5:3,Lambda1S);
   Str(Beta2_k^[ViewSizeCl]:6:3,Beta2S);
   Str(KappaP2_k^[ViewSizeCl]:6:3,KappaP2S);
   Str(Lambda2_k^[ViewSizeCl]:5:3,Lambda2S);
   Str(ViewSizeCl:2,ViewSizeClS);
   GLabel1:=Concat(' Beta1 ',Beta1S,'; Kappa1  ',KappaP1S,';  Lambda1  ',
                   Lambda1S,';  GDT wrt RefSizeClass  ',GDTS);
   GLabel2:=Concat(' Beta2 ',Beta2S,'; Kappa2  ',KappaP2S,';  Lambda2  ',
                   Lambda2S,';   ViewClass ',ViewSizeClS);
   LabelGraphWindow(1,930,GLabel1,0,0);
   LabelGraphWindow(1,900,Glabel2,0,0);
end;

{-------------------------------------------------------------------------}
Procedure Graph1_Results;

begin
   if ViewSizeCl>M then
   begin
      Closesegraphics;
      writeln('Graph 1 Draw ERROR');
      writeln('You have instucted the Graphing Routine to graph the conversion');

      writeln('curves of a size class which does not exist.');
      writeln('Respecify = ViewSizeClass in Declarations section');
      readln;
   end;
   SizeIndex:=ViewSizeCl;
   for ArbIndex:=0 to (N+1) do DataSetX^[ArbIndex]:=ArbIndex*DeltaE;
   if KappaP1_k^[SizeIndex]<>0 then begin
      for ArbIndex:=0 to (N+1) do
         DataSetY^[ArbIndex]:=ARTP1^[SizeIndex,ArbIndex];
      LinePlotData(DataSetX^,DataSetY^,(N+2),3,0);
      for ArbIndex:=0 to (N+1) do
         DataSetY^[ArbIndex]:=SRT1P1^[SizeIndex,ArbIndex];
      LinePlotData(DataSetX^,DataSetY^,(N+2),5,1);
   end;
   if KappaP2_k^[SizeIndex]<>0 then begin
      for ArbIndex:=0 to (N+1) do
         DataSetY^[ArbIndex]:=ARTP1^[SizeIndex,ArbIndex];
      LinePlotData(DataSetX^,DataSetY^,(N+2),3,1);
      for ArbIndex:=0 to (N+1) do
         DataSetY^[ArbIndex]:=SRT2P1^[SizeIndex,ArbIndex];
      LinePlotData(DataSetX^,DataSetY^,(N+2),4,2);
   end;

end;



{-------------------------------------------------------------------------}
Procedure Graph2_Initialise_and_Draw;

var
   XAxisMax                                                :Real;


begin
   SetCurrentWindow(2);
```



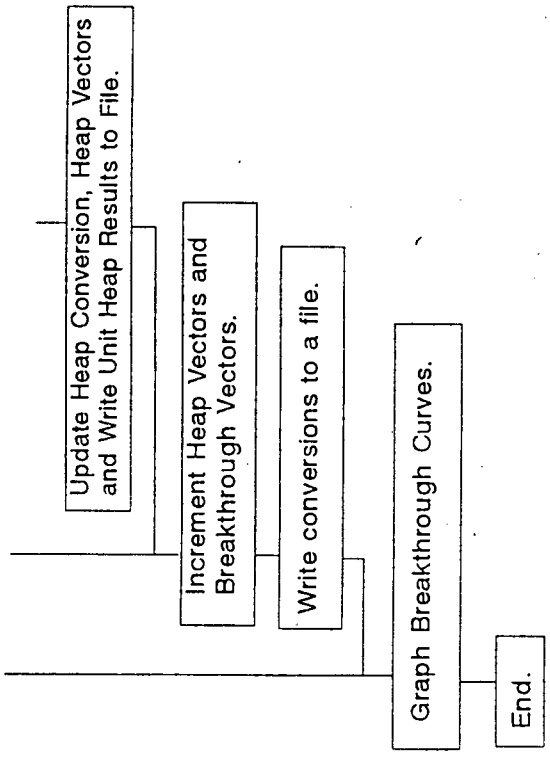
```
   BorderCurrentWindow(1);
   SetAxesType(0,0);
   XAxisMax:=5;                              (***** or  DeltaT*Repeats;****)
   ScalePlotArea(0.0,0.0,XAxisMax,1.0);
   SetXYIntercepts(0.0,0.0);
   SetColor(2);
   DrawXAxis((XAxisMax/5),1);
   DrawYAxis(0.2,1);
   LabelXAxis(1,0);
   LabelYAxis(1,0);
   TitleXAxis('Dimensionless Reaction Time (WRT Reference Particle)');
   TitleYAxis('Fractional Conversion');
   TitleWindow('Model5E2');
   LabelGraphWindow(1,930,GLabel1,0,0);
   LabelGraphWindow(1,900,Glabel2,0,0);
   if KappaP1_k^[SizeIndex]<>0 then begin
      for ArbIndex:=0 to Conv_Var do begin
         DataSetX^[ArbIndex]:=ArbIndex*Print_Crit*DeltaT;
         DataSetY^[ArbIndex]:=Conv1^[ArbIndex];
      end;
      DataSetY^[0]:=0;
      LinePlotData(DataSetX^,DataSetY^,Conv_Var,5,0);
   end;
   if KappaP2_k^[SizeIndex]<>0 then begin
      for ArbIndex:=0 to Conv_Var do begin
         DataSetX^[ArbIndex]:=ArbIndex*Print_Crit*DeltaT;
         DataSetY^[ArbIndex]:=Conv2^[ArbIndex];
      end;
      DataSetY^[0]:=0;
      LinePlotData(DataSetX^,DataSetY^,Conv_Var,4,0);
   end;

end;

{-------------------------------------------------------------------------}
begin

   new(Beta1_k);          new(Beta2_k);
   new(KappaP1_k);        new(KappaP2_k);
   new(KappaS1_k);        new(KappaS2_k);          new(DeltaT_k);
   new(Lambda1_k);        new(Lambda2_k);
   new(ContRatioV1);      new(ContRatioV2);
   new(ART);              new(ARTP1);
   new(SRT1);             new(SRT1P1);             new(SRT1P1C);
   new(SRT2);             new(SRT2P1);             new(SRT2P1C);
   new(Conv1);            new(Conv2);
   new(DataSetX);         new(DataSetY);
   new(AMatrix);          new(AInverse);

   clrscr;

   Size_Distribution_Initialisation;
   Contaminant_Location_Initialisation;
   Determine_Model_Parameters_fn_Size;
   Graph1_Initialise;

   Repeats:=0;
   Flag1  :=0;
   Plot_Var:=0;
   Conv_Var:=0;
```

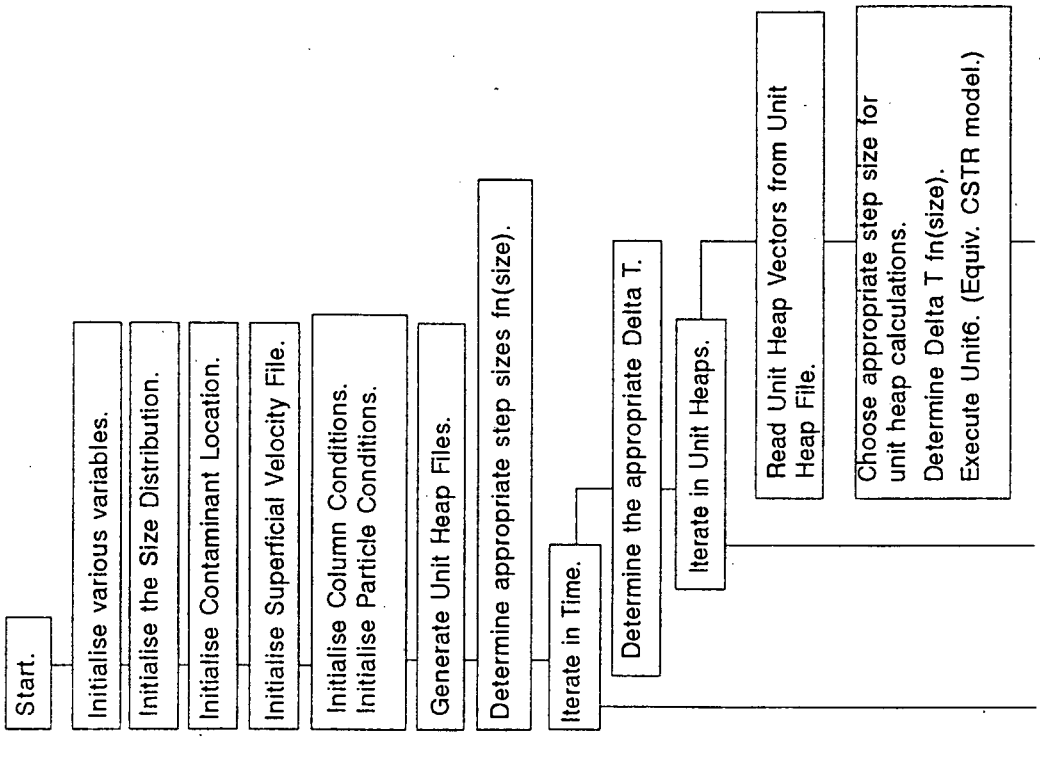
```
    Time0_ART_and_SRTs;
    Guess_SRTsP1;

    while Repeats<Iterations do
    begin
       while (Flag1=0) do
       begin
          Calc_ARTP1_and_SRTsP1C;
          Check_Convergence;
          ReGuess_SRTsP1;
       end;

       Plot_Var:=Plot_Var+1;

       if Plot_Var=Print_Crit then
       begin
          Plot_Var:=0;
          Conv_Var:=Conv_Var+1;
          Graph1_Results;
          Calc_Conversion;
       end;

       Update_ART_and_SRTs;
       Guess_SRTsP1;
       Flag1:=0;
       Repeats:=Repeats+1;

  end;

readln(HardCopy);
if HardCopy=1 then ScreenDump(3,0,2,1.5,1.5,0,1,0,error);

ClearWindow;

Graph2_Initialise_and_Draw;
readln(HardCopy);
if HardCopy=1 then ScreenDump(3,0,2,1.5,1.5,0,1,0,error);


closesegraphics;


end.
```

Appendix V. Solution Algorithm and Code for Model6C1.PAS and Model6C2.PAS.

Model6 represents the computer routines for the Heterogenous Columnar Model discussed in Chapter 6.

Solution Algorithm for Model6C.





```
Program Model6C1;

{Model6C1.  *This program is a code for heap leaching analysis.          }
{           *Incorporates unsteady state flow.                           }
{           *Uses Unit6C1 which is a modified form of the chemical       }
{            CSTR model.                                                 }
{           *Model6C1 valid for first order kinetics, use Model6C2 for   }
{            variable order kinetics.                                    }

{Coded:     Graham Davies.                                               }
{           Department of Chemical Engineering.                          }
{           University of Cape Town.                                     }
{           6 March 1995.                                                }
{           6 June  1995.  Updated. (GMD)                                }
{============================================================================}
{Declarations:                                                           }

uses crt,stdhdr,gj,graph,worlddr,segraph,integrat,M2C1BrentRoots,Unit6C1;

const

   {***These are the parameters appliable to the column.*****************}

   ColLength  =1.76;   {Column length. (m)                                }
   ColVoidage =0.498;  {Voidage of the column.                            }
   SatFrac    =0.361;  {Fraction of the void space filled with fluid.     }
                       {Assumed to be constant in time. ie Does not change}
                       {with a change in the flowrate.                   }
   GlobWetFac =1.0;    {Global wetting factors of the particles.          }


   {***These are column numerical method parameters.*********************}

   NoUnitHeaps=5;      {Number of unit heaps.                             }

   ColDeltaE =1/NoUnitHeaps;

   uStar     =1/86400;{Reference superficial velocity. Equivalent to 1m  }
                      {in 24 hours.                                       }

   SpecColIters=1;

   GfMax_X   =1.0;     {Maximum x-value for conversion graphs.           }
   GfMax_Y   =1.0;     {Maximum y-value for conversion graphs.           }


   {***Other information required.****************************************}

   ValidDir  ='F:\TP6\MODEL6';{Valid directory for graphics drivers.     }

   {*********************************************************************}

var
   SupVelFile      :Text;
   SupVelData      :array[1..2,1..2] of Double;
   WatchSizeDT     :array[1..maxc] of Integer;

   IntLengthCrit,IntLength,ColDelta_t,ColDeltaT,ArbValue   :Double;
   HoldConv1,HoldConv2                                     :Double;
   NumSubInt,TimeCounter                                   :Integer;
```



```
   Continue,ColIters                                       :Integer;

   ARTBTC,SRT1BTC,SRT2BTC,TotHeapConv1,TotHeapConv2        :^VeryLongVector;
   TimeVector                                              :^VeryLongVector;
   TotHeapARTVec,TotHeapSolSRT1Vec,TotHeapSolSRT2Vec       :^ShortVector;
   DT_for_WatchSize                                        :^SqrMat;
   ConvResults                                             :Text;

{============================================================================}
Procedure Initialise_Various_Variables;

begin

   VolLiq        :=ColDeltaE*ColVoidage*ColLength*SatFrac;
   TotVolPart    :=ColDeltaE*(1-ColVoidage)*ColLength*GlobWetFac;

   CumTime       :=0;
   UnitHeapConv1 :=0;
   UnitHeapConv2 :=0;
   ColIters      :=0;
   TimeVector^[0]:=0;
end;


{----------------------------------------------------------------------------}
Procedure Initialise_Superficial_Velocity_File;

begin
   assign(SupVelFile,'SupVelFi.Dat');
   reset(SupVelFile);
   SupVelData[1,1]:=0;
   SupVelData[1,2]:=0;
   read(SupVelFile,SupVelData[2,1]);
   read(SupVelFile,SupVelData[2,2]);
   IntLengthCrit:=ColVoidage*ColDeltaE*ColLength;
end;

{----------------------------------------------------------------------------}
Procedure Col_Initial_Conditions;

begin
   for ArbIndex:=0 to maxc do
   begin
      TotHeapARTVec^[ArbIndex]    :=0;
      TotHeapSolSRT1Vec^[ArbIndex]:=0;
      TotHeapSolSRT2Vec^[ArbIndex]:=0;
   end;

   TotHeapARTVec^[1]:=1;                          {***Imposed BC for heap. }

   for ArbIndex:=0 to maxv do
   begin
      ARTBTC^[ArbIndex] :=0;
      SRT1BTC^[ArbIndex]:=0;
      SRT2BTC^[ArbIndex]:=0;
      TotHeapConv1^[ArbIndex]:=0;
      TotHeapConv2^[ArbIndex]:=0;
   end;
end;
```

Code Listing for Model6C1.PAS.

```
{-------------------------------------------------------------------------}
Procedure Generate_UnitHeapDataFiles;

(This procedure is used to generate the UnitHeapDataFiles.  The files are }
(named M6C1H1 through to M6C1HX.                                           }

var
   UnitHeapNoS,FileName                                    :String;
   HeapFile                                                :Text;

begin
   for UnitHeapIndex:=1 to NoUnitHeaps do
   begin
      Str(UnitHeapIndex,UnitHeapNoS);
      FileName:=Concat('M6C1H',UnitHeapNoS,'.Dat');
      assign(HeapFile,FileName);
      rewrite(HeapFile);
      for SizeIndex:=1 to M do
      begin
         for ArbIndex:=0 to N+1 do
         begin
            write(HeapFile,ART^[SizeIndex,ArbIndex],'    ');
            write(HeapFile,SRT1^[SizeIndex,ArbIndex],'    ');
            writeln(HeapFile,SRT2^[SizeIndex,ArbIndex]);
         end;
         writeln(HeapFile);
      end;

      writeln(HeapFile,UnitHeapConv1,'      ',UnitHeapConv2);
      writeln(HeapFile);
      writeln(HeapFile,'CumTime   ',CumTime);
      close(HeapFile);

   end;
end;

{-------------------------------------------------------------------------}
Procedure Read_ART_and_SRTs;

var
   UnitHeapNoS,FileName                                    :String;
   HeapFile                                                :Text;

begin
   Str(UnitHeapIndex,UnitHeapNoS);
   FileName:=Concat('M6C1H',UnitHeapNoS,'.Dat');
   assign(HeapFile,FileName);
   reset(HeapFile);
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to N do
      begin
         read(HeapFile,ART^[SizeIndex,ArbIndex]);
         read(HeapFile,SRT1^[SizeIndex,ArbIndex]);
         readln(HeapFile,SRT2^[SizeIndex,ArbIndex]);
      end;
      read(HeapFile,ArbValue);
      ART^[SizeIndex,N+1]:=TotHeapARTVec^[UnitHeapIndex];
      read(HeapFile,SRT1^[SizeIndex,N+1]);
      readln(HeapFile,SRT2^[SizeIndex,N+1]);
```



```
   end;
   readln(HeapFile,HoldConv1,HoldConv2);
   close(HeapFile);

end;

{-------------------------------------------------------------------------}
Procedure Write_ART_and_SRTs;

var
   UnitHeapNoS,FileName                                    :String;
   HeapFile                                                :Text;

begin
   Str(UnitHeapIndex,UnitHeapNoS);
   FileName:=Concat('M6C1H',UnitHeapNoS,'.Dat');
   assign(HeapFile,FileName);
   rewrite(HeapFile);
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to N+1 do
      begin
         if ART^[SizeIndex,ArbIndex]>FConc_Tol then
            write(HeapFile,ART^[SizeIndex,ArbIndex],'    ')
         else
            write(HeapFile,0.000,'    ');
         if SRT1^[SizeIndex,ArbIndex]>SConc_Tol then
            write(HeapFile,SRT1^[SizeIndex,ArbIndex],'    ')
         else
            write(HeapFile,0.000,'    ');
         if SRT2^[SizeIndex,ArbIndex]>SConc_Tol then
            writeln(HeapFile,SRT2^[SizeIndex,ArbIndex])
         else
            writeln(HeapFile,0.000);
      end;
      writeln(HeapFile);
   end;
   writeln(HeapFile,UnitHeapConv1,'      ',UnitHeapConv2);
   writeln(HeapFile);
   writeln(HeapFile,'CumTime    ',Cumtime);
   close(HeapFile);
   UnitHeapConv1:=0;
   UnitHeapConv2:=0;
end;

{-------------------------------------------------------------------------}
Procedure Determine_Maximum_DeltaT_fn_Size;

begin
   for UnitHeapIndex:=1 to NoUnitHeaps do
   begin
      for SizeIndex:=1 to M do
         DT_for_WatchSize^[UnitHeapIndex,SizeIndex]:=MaxDeltaT
              *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
              /SizeData[M,1]/SizeData[M,1];
   end;

   for UnitHeapIndex:=1 to NoUnitHeaps do WatchSizeDT[UnitHeapIndex]:=M;

end;
```

```
{-------------------------------------------------------------------------}
Procedure Determine_Corresponding_ColDeltaT;

{This procedure uses the method of characteristics to determine the       }
{progression in time for a given CharColDeltaE.  It allows for unsteady   }
{state flow.                                                              }

begin
   IntLength :=0;
   ColDelta_t:=0;
   while IntLength<>IntLengthCrit do begin
      if EOF(SupVelFile) then begin
         closesegraphics;
         writeln('Insufficient Data in the Superficial Velocity Data File to');
         writeln('execute the required number of iterations.');
         writeln('Present number of iterations:=  ',TimeCounter);
         Continue:=0;
         IntLength:=IntLengthCrit;
         readln;
      end;
      IntLength:=IntLength+(SupVelData[2,1]-SupVelData[1,1])*SupVelData[2,2];
      ColDelta_t:=ColDelta_t+(SupVelData[2,1]-SupVelData[1,1]);
      If IntLength>IntLengthCrit then begin
         ArbValue:=Intlength-IntLengthCrit;
         ColDelta_t:=ColDelta_t-ArbValue/SupVelData[2,2];
         IntLength:=IntLengthCrit;
         SupVelData[1,1]:=SupVelData[2,1]-ArbValue/SupVelData[2,2];
         SupVelData[1,2]:=SupVelData[2,2];
      end else begin
         SupVelData[1,1]:=SupVelData[2,1];
         SupVelData[1,2]:=SupVelData[2,2];
         read(SupVelFile, SupVelData[2,1]);
         read(SupVelFile, SupVelData[2,2])
      end;
   end;
   ColDeltaT:=ColDelta_t*uStar/ColLength;
   CumTime:=CumTime+ColDeltaT;
end;

{-------------------------------------------------------------------------}
Procedure Determine_Appropriate_Ref_Size_Class_DeltaT;

begin
   SizeIndex:=WatchSizeDT[UnitHeapIndex];
   Iterations:=INT(ColDeltaT/(DT_for_WatchSize^[UnitHeapIndex,SizeIndex]))
               +1;
   DeltaT:=ColDeltaT/Iterations;
   writeln(SizeIndex);
end;


{-------------------------------------------------------------------------}
Procedure Check_Whether_WatchSize_Still_Valid;

var
   Flag           :Integer;
   InterG1        :VeryLongVector;
   IntegVal       :Real;
```



```
begin
   SizeIndex:=WatchSizeDT[UnitHeapIndex];
   Flag:=0;
   if ART^[SizeIndex,N]<0.5*ART^[SizeIndex,(N+1)] then
   begin
      for ArbIndex:=0 to N do
      begin
        if (ART^[SizeIndex,ArbIndex]<0.8*ART^[SizeIndex,(ArbIndex+1)])
           or (ART^[SizeIndex,ArbIndex]>1.2*ART^[SizeIndex,(ArbIndex+1)])
           then Flag:=Flag+1;
      end;
   end;
   if Flag=0 then
   begin
      for ArbIndex:=0 to N+1 do InterG1[ArbIndex]:=ART^[SizeIndex,ArbIndex];
      IntegrateVector(InterG1,DeltaE,0,(N+1),IntegVal);
      for ArbIndex:=0 to N+1 do if (ART^[SizeIndex,ArbIndex]<(0.9*Integval))
         or (ART^[SizeIndex,ArbIndex]>(1.1*Integval)) then Flag:=1;
      if (Flag=0) and (WatchSizeDT[UnitHeapIndex]>1) then
         WatchSizeDT[UnitHeapIndex]:=WatchSizeDT[UnitHeapIndex]-1;
   end else if ART^[SizeIndex,N+1]>FConc_Tol then DT_for_WatchSize^[UnitHeapIndex
,SizeIndex]
      :=2*DT_for_WatchSize^[UnitHeapIndex,SizeIndex];

end;


{-------------------------------------------------------------------------}
Procedure UpDate_TotHeapConv;

begin
   TotHeapConv1^[CollIters]:=TotHeapConv1^[CollIters]
                            +GlobWetFac*UnitHeapConv1/NoUnitHeaps;
   TotHeapConv2^[CollIters]:=TotHeapConv2^[CollIters]
                            +GlobWetFac*UnitHeapConv2/NoUnitHeaps;
end;

{-------------------------------------------------------------------------}
Procedure UpDate_TotHeapVector;

begin
   TotHeapARTVec^[UnitHeapIndex]    :=ART^[1,N+1];
   TotHeapSolSRT1Vec^[UnitHeapIndex]:=TotHeapSolSRT1Vec^[UnitHeapIndex]+
                                      (UnitHeapConv1-HoldConv1);
   TotHeapSolSRT2Vec^[UnitHeapIndex]:=TotHeapSolSRT2Vec^[UnitHeapIndex]+
                                      (UnitHeapConv2-HoldConv2);
end;


{-------------------------------------------------------------------------}
Procedure Increment_TotHeapVectors_and_BTCVectors;

begin
   ARTBTC^[CollIters +1] :=TotHeapARTVec^[NoUnitHeaps];
   SRT1BTC^[CollIters +1]:=GlobWetFac*(TotHeapSolSRT1Vec^[NoUnitHeaps]);
   SRT2BTC^[CollIters +1]:=GlobWetFac*(TotHeapSolSRT2Vec^[NoUnitHeaps]);
   for ArbIndex:=NoUnitHeaps downto 2 do
   begin
      TotHeapArtVec^[ArbIndex]:=TotHeapArtVec^[ArbIndex-1];
      TotHeapSolSRT1Vec^[ArbIndex]:=TotHeapSolSRT1Vec^[ArbIndex-1];
```

```
      TotHeapSolSRT2Vec^[ArbIndex]:=TotHeapSolSRT2Vec^[ArbIndex-1];
   end;
   TotHeapArtVec^[1]:=1;
   TotHeapSolSRT1Vec^[1]:=0;
   TotHeapSolSRT2Vec^[1]:=0;

end;

{-------------------------------------------------------------------------}
Procedure Graph_Conversion_and_BTC_Curves;

begin
   initSEGraphics(ValidDir);
   SetCurrentWindow(2);
   BorderCurrentWindow(1);
   SetAxesType(0,0);
   ScalePlotArea(0.0,0.0,(TimeVector^[ColIters]),GfMax_Y);  {***}
   SetXYIntercepts(0.0,0.0);
   SetColor(2);
   DrawXAxis((TimeVector^[ColIters]/5),1);    {***}
   DrawYAxis((GfMax_Y/5),1);
   LabelXAxis(1,0);
   LabelYAxis(1,0);
   TitleXAxis('Dmlss Time (Dmlss Time=(time * uStar)/ColLength');
   TitleYAxis('Frac. Conv. and BTC Curves');
   TitleWindow('Model6C1');
   LabelGraphWindow(1,930,GLabel1,0,0);
   LabelGraphWindow(1,900,Glabel2,0,0);
   for ArbIndex:=0 to ColIters  do DataSetX^[ArbIndex]:=TimeVector^[ArbIndex];
   for ArbIndex:=0 to ColIters  do DataSetY^[ArbIndex]:=TotHeapConv1^[ArbIndex];
   LinePlotData(DataSetX^,DataSetY^,(ColIters +1),5,0);
   for ArbIndex:=0 to ColIters  do DataSetY^[ArbIndex]:=TotHeapConv2^[ArbIndex];
   LinePlotData(DataSetx^,DataSetY^,(ColIters +1),4,0);
   for ArbIndex:=0 to ColIters  do DataSetY^[ArbIndex]:=ARTBTC^[ArbIndex];
   LinePlotData(DataSetx^,DataSetY^,(ColIters +1),3,1);
   for ArbIndex:=0 to ColIters  do DataSetY^[ArbIndex]:=SRT1BTC^[ArbIndex];
   LinePlotData(DataSetx^,DataSetY^,(ColIters +1),5,1);
   for ArbIndex:=0 to ColIters  do DataSetY^[ArbIndex]:=SRT2BTC^[ArbIndex];
   LinePlotData(DataSetx^,DataSetY^,(ColIters +1),4,1);



end;


{=========================================================================}
{MAIN PROGRAM:                                                            }
{=========================================================================}
begin

   new(Beta1_k);          new(Beta2_k);
   new(KappaP1_k);        new(KappaP2_k);
   new(KappaS1_k);        new(KappaS2_k);          new(DeltaT_k);
   new(Lambda1_k);        new(Lambda2_k);
   new(ContRatioV1);      new(ContRatioV2);
   new(ART);              new(ARTP1);
   new(SRT1);             new(SRT1P1);             new(SRT1P1C);
   new(SRT2);             new(SRT2P1);             new(SRT2P1C);
```



```
   new(DataSetX);         new(DataSetY);
   new(AMatrix);          new(AInverse);
   new(ARTBTC);           new(SRT1BTC);            new(SRT2BTC);
   new(TotHeapARTVec);    new(TotHeapSolSRT1Vec);  new(TotHeapSolSRT2Vec);
   new(TotHeapConv1);     new(TotHeapConv2);       new(TimeVector);
   new(DT_for_WatchSize);

   assign(ConvResults,'ConvResults.Dat');
   rewrite(ConvResults);
   close(ConvResults);

   Initialise_Various_Variables;

   Size_Distribution_Initialisation;
   Contaminant_Location_Initialisation;
   Determine_Model_Parameters_fn_Size;

   Initialise_Superficial_Velocity_File;

   Col_Initial_Conditions;
   Particle_Initial_Conditions;
   Generate_UnitHeapDataFiles;

   Determine_Maximum_DeltaT_fn_Size;

   while ColIters<SpecColIters do
   begin
      ColIters:=ColIters+1;
      Determine_Corresponding_ColDeltaT;
      TimeVector^[ColIters]:=CumTime;

      for UnitHeapIndex:=1 to NoUnitHeaps do
      begin
         Read_ART_and_SRTs;
         if ART^[T,N+T]>FConc_Tol then
         begin
            Closesegraphics;
            Determine_Appropriate_Ref_Size_Class_DeltaT;
            Determine_DeltaT_fn_Size;
            Execute_Unit6C1;
            Check_Whether_WatchSize_Still_Valid;
         end;
         UpDate_TotHeapConv;
         UpDate_TotHeapVector;
         Write_ART_and_SRTs;
      end;

      Increment_TotHeapVectors_and_BTCVectors;

      Assign(ConvResults,'ConvResults.Dat');
      Append(ConvResults);
      Writeln(ConvResults,TimeVector^[ColIters]:8:4,'    '
              ,TotHeapConv1^[ColIters]:8:4);
      Close(ConvResults);

   end;

   Graph_Conversion_and_BTC_Curves;

   readln(HardCopy);
```

```
   if HardCopy=1 then ScreenDump(3,0,2,1.5,1.5,0,1,0,error);
   closesegraphics;
end.
{=============================================================================}
```

```
Unit Unit6C1;

{Unit6C1.  * This program is similar to Model5E1.PAS.                    }
{          * The program calculates the concentration profile of fluid   }
{            and solid reactants within a particle using the equations as }
{            developed by Dixon.                                         }
{          * This program only provides for solid reactant orders of 1.  }

{          * Assumptions in this model include:                          }
{                - The solid reactant deposits within the particle       }
{                  resemble those on the surface.                        }
{                - The deposits would both react to the same extent      }
{                  if each were exposed to the same acid concentration   }
{                  for the same time.                                    }

{Coded:      Graham Davies.                                              }
{            Department of Chemical Engineering.                         }
{            University of Cape Town.                                    }
{            28 February 1995.                                           }
{            06 June     1995 Updated (GMD).                             }
{============================================================================}
Interface

{============================================================================}
{Declarations:                                                             }

uses crt,stdhdr,gj,graph,worlddr,segraph,integrat,M2C1BrentRoots;

const

   {***These are the parameters applicable to the reference size class.***}

   Beta1     = 0.264;  {Dimensionless Stoichiometric ratio defined pg 11  }
   KappaP1   = 4.5;    {Ratio of reation rate of solid reactant residing  }
                       {within the particle to porous diffusion of fluid  }
                       {into the particle. Defined pg 11 Dixon.           }

   Beta2     = 0.0;
   KappaP2   = 0.0;

   Voidage   = 0.01;   {Voidage of the solid partilces.(Porosity)         }


   M         =10;      {Number of size classes.                           }
   RefSizeCl =5;       {Defines the reference size class.                 }
   ViewSizeCl=5;       {The concentration profiles of this sizeclass are  }
                       {graphed in graph 1.  (NB  ViewSizeCl <= M.)       }


   {***These are numerical method parameters.*****************************}

   N         =19;      {Half the number of interior points.  r=0 and r=R  }
                       {not included.                                     }
   DeltaE    =1/(N+1); {Space Increment.  Calculated from 1/(N+1). (Since }
                       {                  R=i*dE and R is at point N+1.)  }

   MaxDeltaT =0.001;   {Maximum permitted time increment for particles.   }

   Norm_Crit =1e-6;    {Convergence criteria based on the norm of vector  }
                       {***                                               }
```



```
   Conv_Crit =1e-8;    {Convergence criteria for the Brent Routine.       }
   SConc_Tol =1e-4;    {Dimensionless concentration of solid below which  }
   FConc_Tol =1e-4;    {Dimensionless concentration of fluid reactant     }
                       {below which it is assumed to be negigible.        }
                       {it is assumed to be negligible.                   }
   MaxIter   =100;     {Maximum iterations for the Brent Routine.         }

   Print_Crit=1;

   {***Other information required.*****************************************}

   ValidDir  ='F:\TP6\';{Valid directory for graphics drivers.      }

   OrdSP1    =1.0;     {Reaction order of the solid in the pores.         }
   OrdSP2    =1.0;     {For this program these need to be set at unity.   }
                       {See Unit6C2 for variable order reaction orders.   }

   {**************************************************************************}

type
   array1= Array[1..50,1..2] of Double;

var

   TotVolPart,VolLiq,DeltaT,Iterations,CumTime             :Double;
   UnitHeapConv1,UnitHeapConv2                             :Double;
   ArbIndex,SizeIndex,error,UnitHeapIndex                  :Integer;
   Repeats,Plot_Var,Flag1,HardCopy                         :Integer;
   DataSetX,DataSetY                                       :^VeryLongVector;
   Beta1_k,Beta2_k,KappaP1_k,KappaP2_k,KappaS1_k,KappaS2_k :^ShortVector;
   Lambda1_k,Lambda2_k,ContRatioV1,ContRatioV2             :^ShortVector;
   DeltaT_k                                                :^Array1;
   ART,ARTP1,SRT1,SRT2,SRT1P1,SRT2P1,SRT1P1C,SRT2P1C       :^SqrMat;
   AMatrix,AInverse                                        :^SqrMat;
   ARTP1CVal,SumSizeData_2,NuStar                          :Extended;
   Lambda1,Lambda2,KappaS1,KappaS2                         :Extended;
   GLabel1,Glabel2,GLabel3                                 :String;

   SizeData : Array[1..50,1..2] of Real;

   {ART     : Alpha fn(r) at time T                    range 0..M  0..N+1}
   {ARTP1   : Alpha fn(r) at time T+1                  range 0..M  0..N+1}
   {SRT1    : Sigma fn(r) at time T                    range 0..M  0..N+1}
   {SRTsP1  : Sigma fn(r) at time T+1      (guessed)   range 0..M  0..N+1}
   {SRTsP1C : Sigma fn(r) at time T+1      (calculated) range 0..M  0..N+1}
   {YVect   : 'Const' vector in Crank-Nicolson method  range 0..N+1      }
   {AMatrix : Matrix of Crank-Nicolson coefficients    range N* N        }
   {DataSetX: X vector used in the graphing routine    range 0..N+1      }

   {**************************************************************************}

   Procedure Size_Distribution_Initialisation;
   Procedure Contaminant_Location_Initialisation;
   Procedure Determine_Model_Parameters_fn_Size;
   Procedure Determine_DeltaT_fn_Size;
   Procedure Particle_Initial_Conditions;
   Procedure Guess_SRTsP1;
   Procedure ReGuess_SRTsP1;
   Procedure Calc_Crank_Nicolson_Matrix(var YVector :ShortVector);
   Procedure Calc_ARTP1_and_SRTsP1C;
```

```
   Procedure Check_Convergence;
   Procedure Update_ART_and_SRTs;
   Procedure Calc_Conversion;
   Procedure Graph1_Initialise;
   Procedure Graph1_Results;
   Procedure Execute_Unit6C1;

{=========================================================================}
Implementation

{=========================================================================}
Procedure Size_Distribution_Initialisation;

{This procedure sets up the size distribution data array. Initially the   }
{SizeData array contains radius information and fractional volume         }
{information (ie R and Vp/Vtot).  On output it contains  relative radius  }
{information and relative volume information (ie R/R_ref and Vp/Vp_ref).  }
{Ideally this information would be read in from a data file.              }

var
   RefRadius,VolRefPart :Double;

begin
   SizeData[1,1] := 37.00e-3/2;          SizeData[1,2] := 58.1/100;
   SizeData[2,1] := 31.25e-3/2;          SizeData[2,2] := 15.8/100;
   SizeData[3,1] := 22.00e-3/2;          SizeData[3,2] := 10.3/100;
   SizeData[4,1] := 16.10e-3/2;          SizeData[4,2] :=  4.1/100;
   SizeData[5,1] := 11.35e-3/2;          SizeData[5,2] :=  3.2/100;
   SizeData[6,1] :=  8.10e-3/2;          SizeData[6,2] :=  1.8/100;
   SizeData[7,1] :=  5.73e-3/2;          SizeData[7,2] :=  1.3/100;
   SizeData[8,1] :=  4.05e-3/2;          SizeData[8,2] :=  1.0/100;
   SizeData[9,1] :=  2.86e-3/2;          SizeData[9,2] :=  0.6/100;
   SizeData[10,1]:=  1.18e-3/2;          SizeData[10,2]:=  3.8/100;

   VolRefPart:=TotVolPart*SizeData[RefSizeCl,2];
   RefRadius :=SizeData[RefSizeCl,1];

   SumSizeData_2:=0;

   for SizeIndex:=1 to M do
   begin
      SizeData[SizeIndex,1]:=SizeData[SizeIndex,1]/RefRadius;
      SizeData[SizeIndex,2]:=SizeData[SizeIndex,2]*TotVolPart/VolRefPart;
      SumSizeData_2:=SumSizeData_2+SizeData[SizeIndex,2];
   end;

   NuStar:=VolLiq/(Voidage*VolRefPart);

   {Ratio of volume of bulk fluid to fluid in particle pores.             }

end;


{-------------------------------------------------------------------------}
Procedure Contaminant_Location_Initialisation;

{This procedure is used to define the contaminant ratio vector (ratio of  }
{the surface contaminant concentration to bulk contaminant concentration).}
{NOTE: If any size class of particles have a surface concentration of     }
{      contaminant, then so too must the reference size class.            }
```



```
begin
   ContRatioV1^[1] :=0.00;          ContRatiov2^[1] :=0;
   ContRatioV1^[2] :=0.00;          ContRatiov2^[2] :=0;
   ContRatioV1^[3] :=0.00;          ContRatiov2^[3] :=0;
   ContRatioV1^[4] :=0.00;          ContRatiov2^[4] :=0;
   ContRatioV1^[5] :=0.00;          ContRatiov2^[5] :=0;
   ContRatioV1^[6] :=0.00;          ContRatiov2^[6] :=0;
   ContRatioV1^[7] :=0.00;          ContRatiov2^[7] :=0;
   ContRatioV1^[8] :=0.00;          ContRatiov2^[8] :=0;
   ContRatioV1^[9] :=0.00;          ContRatiov2^[9] :=0;
   ContRatioV1^[10]:=0.00;          ContRatiov2^[10]:=0;
end;

{-------------------------------------------------------------------------}
Procedure Determine_Model_Parameters_fn_Size;

{This procedure determines the model parameters each size class of        }
{particles.  Note that the DeltaT is defined separately in another        }
{procedure.                                                               }

var
   Flag1,Flag2 :Integer;

begin
  Flag1:=0;
  Flag2:=0;
  for SizeIndex:=1 to M do
  begin
      Lambda1_k^[SizeIndex]:=ContRatioV1^[SizeIndex]
                             /(1+ContRatioV1^[SizeIndex]);
      Lambda2_k^[SizeIndex]:=ContRatioV2^[SizeIndex]
                             /(1+ContRatioV2^[SizeIndex]);
      if ContRatioV1^[SizeIndex]<>0 then Flag1:=Flag1+1;
      if ContRatioV2^[SizeIndex]<>0 then Flag2:=Flag2+1;
   end;

   if (((Flag1=1) and (ContRatioV1^[RefSizeCl]=0)) or
       ((Flag2=1) and (ContRatiov2^[RefSizeCl]=0))) then begin
      closesegraphics;
      writeln('Contaminant Location Violoation.');
      readln;
   end;
   Lambda1              :=Lambda1_k^[RefSizeCl];
   Lambda2              :=Lambda2_k^[RefSizeCl];
   KappaS1              :=Lambda1*KappaP1/(1-Lambda1);
   KappaS2              :=Lambda2*KappaP2/(1-Lambda2);

   for SizeIndex:=1 to M do
   begin
      Beta1_k^[SizeIndex]  :=Beta1*(1+ContRatioV1^[RefSizeCl])
                             /(1+ContRatioV1^[SizeIndex]);
      Beta2_k^[SizeIndex]  :=Beta2*(1+ContRatioV2^[RefSizeCl])
                             /(1+ContRatioV2^[SizeIndex]);
      KappaP1_k^[SizeIndex]:=KappaP1*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1];
      KappaP2_k^[SizeIndex]:=KappaP2*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1];
      {*****Note:Unity power assumption involved in next few lines*****   }
```

```
      if ContRatioV1^[SizeIndex]=0 then KappaS1_k^[SizeIndex]:=0 else
         KappaS1_k^[SizeIndex]:=KappaS1*ContRatioV1^[SizeIndex]
                               /ContRatioV1^[RefSizeCl]*SizeData[SizeIndex,1];
      if ContRatioV2^[SizeIndex]=0 then KappaS2_k^[SizeIndex]:=0 else
         KappaS2_k^[SizeIndex]:=KappaS2*ContRatioV2^[SizeIndex]
                               /ContRatioV2^[RefSizeCl]*SizeData[SizeIndex,1];

   end;
end;
{-------------------------------------------------------------------------}
Procedure Determine_DeltaT_fn_Size;

begin
   for SizeIndex:=1 to M do
      DeltaT_k^[SizeIndex,1] :=DeltaT/SizeData[SizeIndex,1]
                             /SizeData[SizeIndex,1];

end;

{-------------------------------------------------------------------------}
Procedure Particle_Initial_Conditions;

{This procedure uses the initial conditions to set the alpha and sigma    }
{vectors.                                                                 }
{Note that ART[N+1], ARTP1[N+1], SRT1[N+1] and SRT1P1[N+1] are the surface}
{concentrations of the liquid and solid reactants.                        }

begin
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to N do
      begin
         ART^[SizeIndex,ArbIndex] :=0;
         SRT1^[SizeIndex,ArbIndex]:=1;
         SRT2^[SizeIndex,ArbIndex]:=1;
         SRT1P1C^[SizeIndex,ArbIndex]:=0;
         SRT2P1C^[SizeIndex,ArbIndex]:=0;
      end;
      ART^[SizeIndex,N+1] :=1;
      SRT1^[SizeIndex,N+1]:=1;
      SRT2^[SizeIndex,N+1]:=1;
      SRT1P1C^[SizeIndex,ArbIndex]:=0;
      SRT2P1C^[SizeIndex,ArbIndex]:=0;
   end;
end;

{-------------------------------------------------------------------------}
Procedure Guess_SRTsP1;

{This procedure provides the initial "guess" for the iteration.  It uses  }
{the previous time interval's values as the guess.                        }

begin
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
      begin
         SRT1P1^[SizeIndex,ArbIndex]:=SRT1^[SizeIndex,ArbIndex];
         SRT2P1^[SizeIndex,ArbIndex]:=SRT2^[SizeIndex,ArbIndex];
      end;
```



```
      ARTP1^[SizeIndex,N+1]:=ART^[SizeIndex,N+1];
   end;
end;

{-------------------------------------------------------------------------}
Procedure ReGuess_SRTsP1;

{This procedure provides an updated "guess" for the next iteration. It    }
{uses the SRT1P1C vector as the updated guess.                            }

begin
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
      begin
         SRT1P1^[SizeIndex,ArbIndex]:=SRT1P1C^[SizeIndex,ArbIndex];
         SRT2P1^[SizeIndex,ArbIndex]:=SRT2P1C^[SizeIndex,ArbIndex];
      end;
      ARTP1^[SizeIndex,N+1]:=ARTP1CVal;
   end;
end;
{-------------------------------------------------------------------------}
Procedure Calc_Crank_Nicolson_Matrix(var YVector :ShortVector);

{*****Note:Unity power assumption involved in this procedure*****         }

var
   Rows,Cols                                          :Integer;

begin
   for Rows:=0 to (N+1) do
   begin
      for Cols:=0 to (N+1) do AMatrix^[Rows,Cols]:=0;
   end;

   AMatrix^[0,0]:=(-6-DeltaE*DeltaE*SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                  *(KappaP1_k^[SizeIndex]*SRT1P1^[SizeIndex,0]
                  +KappaP2_k^[SizeIndex]*SRT2P1^[SizeIndex,0])-2*DeltaE*DeltaE
                  *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                  /DeltaT_k^[SizeIndex,1]);
   AMatrix^[0,1]:=6;
   YVector[0]   :=ART^[SizeIndex,0]*(6+DeltaE*DeltaE*SizeData[SizeIndex,1]
                  *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]
                  *SRT1^[SizeIndex,0]+KappaP2_k^[SizeIndex]
                  *SRT2^[SizeIndex,0])-2*DeltaE*DeltaE*SizeData[SizeIndex,1]
                  *SizeData[SizeIndex,1]/DeltaT_k^[SizeIndex,1])
                  -ART^[SizeIndex,1]*6;

   for Rows:=1 to N-1 do
   begin
      AMatrix^[Rows,Rows-1]:=Rows-1;
      AMatrix^[Rows,Rows]  :=-2*Rows-Rows*DeltaE*DeltaE*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]*
                             SRT1P1^[SizeIndex,Rows]+KappaP2_k^[SizeIndex]*
                             SRT2P1^[SizeIndex,Rows])-2*Rows*DeltaE
                             *DeltaE*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1]/DeltaT_k^[SizeIndex,1];
      AMatrix^[Rows,Rows+1]:=Rows+1;
      YVector[Rows]        :=ART^[SizeIndex,Rows-1]*(-Rows+1)+
                             ART^[SizeIndex,Rows]*(2*Rows+Rows*DeltaE*DeltaE
```

```
                        *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                        *(KappaP1_k^[SizeIndex]*SRT1^[SizeIndex,Rows]
                        +KappaP2_k^[SizeIndex]*SRT2^[SizeIndex,Rows])
                        -2*Rows*DeltaE*DeltaE*SizeData[SizeIndex,1]
                        *SizeData[SizeIndex,1]/DeltaT_k^[SizeIndex,1])
                        +ART^[SizeIndex,Rows+1]*(-Rows-1);
   end;

   AMatrix^[N,N-1]:=N-1;
   AMatrix^[N,N]  :=-2*N-N*DeltaE*DeltaE*SizeData[SizeIndex,1]
                    *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]
                    *SRT1P1^[SizeIndex,N]+KappaP2_k^[SizeIndex]
                    *SRT2P1^[SizeIndex,N])-2*N*DeltaE*DeltaE
                    *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                    /DeltaT_k^[SizeIndex,1];
   YVector[N]     :=ART^[SizeIndex,N-1]*(-N+1)+ART^[SizeIndex,N]*(2*N+
                    N*DeltaE*DeltaE*SizeData[SizeIndex,1]
                    *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]
                    *SRT1^[SizeIndex,N]+KappaP2_k^[SizeIndex]
                    *SRT2^[SizeIndex,N])-2*N*DeltaE*DeltaE
                    *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                    /DeltaT_k^[SizeIndex,1])+ART^[SizeIndex,N+1]*(-N-1)
                    -ARTP1^[SizeIndex,N+1]*(N+1);

end;

{-------------------------------------------------------------------------}
Procedure Calc_ARTP1_and_SRTsP1C;

{As it stands this procedure can cope with a variable reaction order. This}
{is due to the inclusion of the Brent Routine.                            }

var
   MassBalVal,AMatDet                          :Double;
   ValueAtRoot                                 :Real;
   OutputVector,YVector                        :ShortVector;

begin

   {First calculate the ARTP1 vectors.                                    }

   for SizeIndex:=1 to M do
   begin
      Calc_Crank_Nicolson_Matrix(YVector);
      GaussJordan(AMatrix^,YVector,(N+1),OutputVector,AInverse^,AMatDet);
      for ArbIndex:=0 to N do
         if OutPutVector[ArbIndex]>FConc_Tol then
            ARTP1^[SizeIndex,ArbIndex]:=OutputVector[ArbIndex]
         else
            ARTP1^[SizeIndex,ArbIndex]:=0;
   end;

   {The GaussJordan procedure calculates values for the ART vector from   }
   {point 0 to point N (although it makes use of the N+1_th point). To    }
   {determine the N+1_th point value, make use of the mass balance of     }
   {fluid reactant in the CSTR.                                           }


   MassBalVal:=0;
```



```
   for SizeIndex:=1 to M do
   begin
      MassBalVal:=MassBalVal-SizeData[SizeIndex,2]/2*(KappaS1_k^[SizeIndex]
                  *SRT1^[SizeIndex,(N+1)]*ART^[SizeIndex,(N+1)]
                  +KappaS1_k^[SizeIndex]*SRT1P1^[SizeIndex,(N+1)]
                  *ARTP1^[SizeIndex,(N+1)]+KappaS2_k^[SizeIndex]
                  *SRT2^[SizeIndex,(N+1)]*ART^[SizeIndex,(N+1)]
                  +KappaS2_k^[SizeIndex]*SRT2P1^[SizeIndex,(N+1)]
                  *ARTP1^[SizeIndex,(N+1)])-3*SizeData[SizeIndex,2]
                  /(SizeData[SizeIndex,1]*SizeData[SizeIndex,1])/2
                  *((ART^[SizeIndex,(N+1)]+ARTP1^[SizeIndex,(N+1)])
                  -(ART^[SizeIndex,N]+ARTP1^[SizeIndex,N]))/DeltaE;

   end;

   ARTP1CVal:=ART^[1,(N+1)]+DeltaT/NuStar*MassBalVal;

   {Calculate the SRTsP1C vectors.  Code makes use of Brent's Method      }
   {(A non-linear root finding procedure) to solve for the root.         }

   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do begin
         if (Beta1_k^[SizeIndex]<>0) and (KappaP1_k^[SizeIndex]<>0) then
         begin
            if OrdSP1=1 then
            begin
               SRT1P1C^[SizeIndex,ArbIndex]:=SRT1^[SizeIndex,ArbIndex]
                        *(1-DeltaT_k^[SizeIndex,1]*KappaP1_k^[SizeIndex]
                        *Beta1_k^[SizeIndex]*ART^[SizeIndex,ArbIndex]
                        /(2*(1-Lambda1_k^[SizeIndex])))/(1+DeltaT_k^[SizeIndex,1]
                        *KappaP1_k^[SizeIndex]*Beta1_k^[SizeIndex]
                        *ARTP1^[SizeIndex,ArbIndex]/(2*(1-Lambda1_k^[SizeIndex])))
;
            end
            else if SRT1P1^[SizeIndex,ArbIndex]<SConc_Tol then
               SRT1P1C^[SizeIndex,ArbIndex]:=SRT1P1^[SizeIndex,ArbIndex]
            else
            begin
               SRT1P1C^[SizeIndex,ArbIndex]:=BrentRoots(0.0,1.0,DeltaT_k^[SizeInd
ex,1],
                        KappaP1_k^[SizeIndex],Beta1_k^[SizeIndex],
                        Lambda1_k^[SizeIndex],OrdSP1,ART^[SizeIndex,ArbIndex],
                        ARTP1^[SizeIndex,ArbIndex],SRT1^[SizeIndex,ArbIndex],
                        1e-8,100,ValueAtRoot,error);
            end;
         end;
         if (Beta2_k^[SizeIndex]<>0) and (KappaP2_k^[SizeIndex]<>0) then
         begin
            if OrdSP2=1 then
            begin
               SRT2P1C^[SizeIndex,ArbIndex]:=SRT2^[SizeIndex,ArbIndex]
                        *(1-DeltaT_k^[SizeIndex,1]*KappaP2_k^[SizeIndex]
                        *Beta2_k^[SizeIndex]*ART^[SizeIndex,ArbIndex]
                        /(2*(1-Lambda2_k^[SizeIndex])))/(1+DeltaT_k^[SizeIndex,1]
                        *KappaP2_k^[SizeIndex]*Beta2_k^[SizeIndex]
                        *ARTP1^[SizeIndex,ArbIndex]/(2*(1-Lambda2_k^[SizeIndex])))
;
            end
            else if SRT2P1^[SizeIndex,ArbIndex]<SConc_Tol then
```

```
            SRT1P1C^[SizeIndex,ArbIndex]:=SRT2P1^[SizeIndex,ArbIndex]
         else
         begin
            SRT2P1C^[SizeIndex,ArbIndex]:=BrentRoots(0.0,1.0,
                  DeltaT_k^[SizeIndex,1],KappaP2_k^[SizeIndex],
                  Beta2_k^[SizeIndex],Lambda2_k^[SizeIndex],OrdSP2,
                  ART^[SizeIndex,ArbIndex],ARTP1^[SizeIndex,ArbIndex],
                  SRT2^[SizeIndex,ArbIndex],1e-8,100,ValueAtRoot,error);
         end;
      end;
   end;
end;


{-------------------------------------------------------------------------}
Procedure Check_Convergence;

{This procedure checks whether or not the solution has converged by       }
{comparing the guessed value of SRT1P1 with a calculated value of SRT1P1. }
{Also compares the calculated value of ARTP1C with guessed value of       }
{ARTP1.                                                                   }

var
   Norm1,Norm2,Norm3                                        :Double;

begin
   Norm1:=0;         {Using a norm ***                                    }
   Norm2:=0;
   Norm3:=0;
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
      begin
         if (Beta1_k^[SizeIndex]<>0) and (KappaP1_k^[SizeIndex]<>0) then begin
            Norm1:=Norm1+(SRT1P1C^[SizeIndex,ArbIndex]
                  -SRT1P1^[SizeIndex,ArbIndex])*(SRT1P1C^[SizeIndex,ArbIndex]
                  -SRT1P1^[SizeIndex,ArbIndex]);
         end
         else Norm1:=0;
         if (Beta2_k^[SizeIndex]<>0) and (KappaP2_k^[SizeIndex]<>0) then begin
            Norm2:=Norm2+(SRT2P1C^[SizeIndex,ArbIndex]
                  -SRT2P1^[SizeIndex,ArbIndex])*(SRT1P1C^[SizeIndex,ArbIndex]
                  -SRT2P1^[SizeIndex,ArbIndex]);
         end
         else Norm2:=0;
      end;
   end;
   Norm3:=(ARTP1CVal-ARTP1^[1,N+1])*(ARTP1CVal-ARTP1^[1,N+1]);
   if (Norm1<Norm_Crit) and (Norm2<Norm_Crit) and (Norm3<Norm_Crit)
      then Flag1:=1;
end;

{-------------------------------------------------------------------------}
Procedure Update_ART_and_SRTs;

{This procedure updates the alpha and sigma vectors for the next iteration}
{by replacing their components with the alphaT+1 and sigmaT+1 vectors.    }

begin
```



```
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
      begin
         ART^[SizeIndex,ArbIndex] :=ARTP1^[SizeIndex,Arbindex];
         SRT1^[SizeIndex,ArbIndex]:=SRT1P1^[SizeIndex,ArbIndex];
         SRT2^[SizeIndex,ArbIndex]:=SRT2P1^[SizeIndex,ArbIndex];
      end;
   end;
end;


{-------------------------------------------------------------------------}
Procedure Calc_Conversion;

{This procedure calculates the fractional conversion of the particle for  }
{the time interval.  It uses the formula on pg. 13 Dixon.  The integrator }
{is the Quinn-Curtis vector integrator.                                   }

var
   InterG1,InterG2                                          :VeryLongVector;
   IntegVal,Conversion                                      :Real;

begin
   for SizeIndex:=1 to M do
   begin
      if KappaP1_k^[SizeIndex]<>0 then
      begin
         for ArbIndex:=0 to (N+1) do
            InterG1[ArbIndex]:=(1-SRT1^[SizeIndex,ArbIndex])*ArbIndex*DeltaE
                  *ArbIndex*DeltaE;
         IntegrateVector(InterG1,DeltaE,0,(N+1),IntegVal);
         Conversion      :=3*(1-Lambda1_k^[SizeIndex])*IntegVal
                  +Lambda1_k^[SizeIndex]*(1-SRT1^[SizeIndex,N+1]);
         UnitHeapConv1:=UnitHeapConv1+Conversion*SizeData[SizeIndex,2]
                  /SumSizeData_2;
      end;

      if KappaP2_k^[SizeIndex]<>0 then
      begin
         for ArbIndex:=0 to (N+1) do
         InterG2[ArbIndex]:=(1-SRT2^[SizeIndex,ArbIndex])*ArbIndex*DeltaE
                  *ArbIndex*DeltaE;
         IntegrateVector(InterG2,DeltaE,0,(N+1),IntegVal);
         Conversion      :=3*(1-Lambda2_k^[SizeIndex])*IntegVal
                  +Lambda2_k^[SizeIndex]*(1-SRT2^[SizeIndex,N+1]);
         UnitHeapConv2:=UnitHeapConv2+Conversion*SizeData[SizeIndex,2]
                  /SumSizeData_2;
      end;
   end;
end;

{-------------------------------------------------------------------------}
Procedure Graph1_Initialise;

var
   GraphType                                                :Integer;
   Beta1S,KappaP1S,Lambda1S,GDTS                            :String;
   Beta2S,KappaP2S,Lambda2S                                 :String;
   OrdSP1S,OrdSP2S                                          :String;
```

```
   CumTimeS,UnitHeapNoS,ViewSizeClS                    :String;

begin
   initSEGraphics(ValidDir);
   SetCurrentWindow(3);
   BorderCurrentWindow(1);
   SetAxesType(0,0);
   ScalePlotArea(0.0,0.0,1.0,1.2);
   SetXYIntercepts(0.0,0.0);
   SetColor(2);
   DrawXAxis(0.2,1);
   DrawYAxis(0.2,1);
   LabelXAxis(1,0);
   LabelYAxis(1,0);
   TitleXAxis('Dimensionless Radius');
   TitleYAxis('Dmlss Conc');
   TitleWindow('Unit6C1');
   Str(Beta1:6:3,Beta1S);
   Str(KappaP1:6:3,KappaP1S);
   Str((DeltaT*Print_Crit):6:4,GDTS);
   Str(Lambda1:5:3,Lambda1S);
   Str(OrdSP1:5:2,OrdSP1S);
   Str(Beta2:6:3,Beta2S);
   Str(KappaP2:6:3,KappaP2S);
   Str(Lambda2:5:3,Lambda2S);
   Str(OrdSP2:5:2,OrdSP2S);
   Str(CumTime:8:4,CumTimeS);
   Str(UnitHeapIndex:2,UnitHeapNoS);
   Str(ViewSizeCl:2,ViewSizeClS);
   GLabel1:=Concat(' Beta1 ',Beta1S,'; Kappa1  ',KappaP1S,';  Lambda1  ',
                   Lambda1S,'; Order1 ',OrdSP1S,';  GDT  ',GDTS);
   GLabel2:=Concat(' Beta2 ',Beta2S,'; Kappa2  ',KappaP2S,';  Lambda2  ',
                   Lambda2S,'; Order2 ',OrdSP2S);
   GLabel3:=Concat(' CumTime ',CumTimeS,'     Profiles of unit heap ',
                   UnitHeapNoS,' and sizeclass  ',ViewSizeClS);
   LabelGraphWindow(1,900,GLabel1,0,0);
   LabelGraphWindow(1,850,GLabel2,0,0);
   LabelGraphWindow(160,800,GLabel3,0,0);
   for ArbIndex:=1 to 2 do
   begin
      if ArbIndex=1 then GraphType:=9 else GraphType:=10;
      SetCurrentWindow(GraphType);
      BorderCurrentWindow(1);
      SetAxesType(0,0);
      ScalePlotArea(0.0,0.0,1.0,1.2);
      SetXYIntercepts(0.0,0.0);
      SetColor(2);
      DrawXAxis(0.2,1);
      DrawYAxis(0.2,1);
      LabelXAxis(1,0);
      LabelYAxis(1,0);
      TitleXAxis('Dimensionless Radius');
      TitleYAxis('Dmlss Conc');
      TitleWindow('Unit6C1');
      if ArbIndex=1 then LabelGraphWindow(200,900,'Smallest Size Fraction',0,0)
      else LabelGraphWindow(200,900,'Largest Size Fraction',0,0);
   end;

end;
```



```
{-------------------------------------------------------------------------}
Procedure Graph1_Results;

var
   GraphIndex      :Integer;

begin
   if ViewSizeCl>M then
   begin
      Closesegraphics;
      writeln('Graph 1 Draw ERROR');
      writeln('You have instucted the Graphing Routine to graph the conversion');

      writeln('curves of a size class which does not exist.');
      writeln('Respecify = ViewSizeClass in Declarations section');
      readln;
   end;
   for GraphIndex:=1 to 3 do
   begin
      if GraphIndex=1 then
      begin
         SizeIndex:=ViewSizeCl;
         SetCurrentWindow(3);
      end else if GraphIndex=2 then
      begin
         SizeIndex:=M;
         SetCurrentWindow(9);
      end else
      begin
         SizeIndex:=1;
         SetCurrentWindow(10);
      end;
      for ArbIndex:=0 to (N+1) do DataSetX^[ArbIndex]:=ArbIndex*DeltaE;
      if KappaP1<>0 then begin
         for ArbIndex:=0 to (N+1) do
            DataSetY^[ArbIndex]:=ART^[SizeIndex,ArbIndex];
         LinePlotData(DataSetX^,DataSetY^,(N+2),3,0);
         for ArbIndex:=0 to (N+1) do
            DataSetY^[ArbIndex]:=SRT1^[SizeIndex,ArbIndex];
         LinePlotData(DataSetX^,DataSetY^,(N+2),5,0);
      end;
      if KappaP2<>0 then begin
         for ArbIndex:=0 to (N+1) do
            DataSetY^[ArbIndex]:=ART^[SizeIndex,ArbIndex];
         LinePlotData(DataSetX^,DataSetY^,(N+2),3,0);
         for ArbIndex:=0 to (N+1) do
            DataSetY^[ArbIndex]:=SRT2^[SizeIndex,ArbIndex];
         LinePlotData(DataSetX^,DataSetY^,(N+2),4,2);
      end;
   end;
end;


{=========================================================================}
Procedure Execute_Unit6C1;

begin

   clrscr;
```

```
   Graph1_Initialise;

   Repeats:=0;
   Flag1  :=0;
   Plot_Var:=0;

   Guess_SRTsP1;

   Graph1_Results;


   while (Repeats<Iterations) and (ART^[1,N+1]>FConc_Tol) do
   begin
      while (Flag1=0) do
      begin
         Calc_ARTP1_and_SRTsP1C;
         Check_Convergence;
         ReGuess_SRTsP1;
      end;

      Plot_Var:=Plot_Var+1;

      if Plot_Var=Print_Crit then
      begin
         Plot_Var:=0;
         Graph1_Results;
      end;

      Update_ART_and_SRTs;
      Guess_SRTsP1;
      Flag1:=0;
      Repeats:=Repeats+1;

   end;

   Graph1_Results;
   Calc_Conversion;

end;

end.

{=============================================================================}
```

```
Program Model6C2;

{Model6C2.  *This program is a code for heap leaching analysis.          }
{           *Incorporates unsteady state flow.                           }
{           *Uses Unit6C2 which is a modified form of the chemical       }
{            CSTR model.                                                 }
{           *Model6C2 valid for variable order kinetics.                 }

{Coded:      Graham Davies.                                              }
{            Department of Chemical Engineering.                         }
{            University of Cape Town.                                    }
{            6 March 1995.                                               }
{            6 June  1995.  Updated. (GMD)                               }
{============================================================================}
{Declarations:                                                           }

uses crt,stdhdr,gj,graph,worlddr,segraph,integrat,M2C1BrentRoots,Unit6C2;

const

   {***These are the parameters appliable to the column.*****************}

   ColLength  =1.76;   {Column length. (m)                                }
   ColVoidage =0.498;  {Voidage of the column.                            }
   SatFrac    =0.361;  {Fraction of the void space filled with fluid.     }
                       {Assumed to be constant in time. ie Does not change}
                       {with a change in the flowrate.                   }
   GlobWetFac =1.0;    {Global wetting factors of the particles.          }


   {***These are column numerical method parameters.*********************}

   NoUnitHeaps=5;      {Number of unit heaps.                             }

   ColDeltaE =1/NoUnitHeaps;

   uStar     =1/86400;{Reference superficial velocity. Equivalent to 1m  }
                      {in 24 hours.                                       }

   SpecColIters=1;

   GfMax_X   =1.0;     {Maximum x-value for conversion graphs.           }
   GfMax_Y   =1.0;     {Maximum y-value for conversion graphs.           }


   {***Other information required.***************************************}

   ValidDir  ='F:\TP6\MODEL6';{Valid directory for graphics drivers.     }

   {********************************************************************}

var
   SupVelFile      :Text;
   SupVelData      :array[1..2,1..2] of Double;
   WatchSizeDT     :array[1..maxc] of Integer;

   IntLengthCrit,IntLength,ColDelta_t,ColDeltaT,ArbValue   :Double;
   HoldConv1,HoldConv2                                     :Double;
   NumSubInt,TimeCounter                                   :Integer;
   Continue,ColIters                                       :Integer;
```



```
   ARTBTC,SRT1BTC,SRT2BTC,TotHeapConv1,TotHeapConv2        :^VeryLongVector;
   TimeVector                                              :^VeryLongVector;
   TotHeapARTVec,TotHeapSolSRT1Vec,TotHeapSolSRT2Vec       :^ShortVector;
   DT_for_WatchSize                                        :^SqrMat;
   ConvResults                                             :Text;

{============================================================================}
Procedure Initialise_Various_Variables;

begin

   VolLiq        :=ColDeltaE*ColVoidage*ColLength*SatFrac;
   TotVolPart    :=ColDeltaE*(1-ColVoidage)*ColLength*GlobWetFac;

   CumTime       :=0;
   UnitHeapConv1 :=0;
   UnitHeapConv2 :=0;
   ColIters      :=0;
   TimeVector^[0]:=0;
end;


{----------------------------------------------------------------------------}
Procedure Initialise_Superficial_Velocity_File;

begin
   assign(SupVelFile,'SupVelFi.Dat');
   reset(SupVelFile);
   SupVelData[1,1]:=0;
   SupVelData[1,2]:=0;
   read(SupVelFile,SupVelData[2,1]);
   read(SupVelFile,SupVelData[2,2]);
   IntLengthCrit:=ColVoidage*ColDeltaE*ColLength;
end;

{----------------------------------------------------------------------------}
Procedure Col_Initial_Conditions;

begin
   for ArbIndex:=0 to maxc do
   begin
      TotHeapARTVec^[ArbIndex]    :=0;
      TotHeapSolSRT1Vec^[ArbIndex]:=0;
      TotHeapSolSRT2Vec^[ArbIndex]:=0;
   end;

   TotHeapARTVec^[1]:=1;                          {***Imposed BC for heap. }

   for ArbIndex:=0 to maxv do
   begin
      ARTBTC^[ArbIndex] :=0;
      SRT1BTC^[ArbIndex]:=0;
      SRT2BTC^[ArbIndex]:=0;
      TotHeapConv1^[ArbIndex]:=0;
      TotHeapConv2^[ArbIndex]:=0;
   end;
end;

{----------------------------------------------------------------------------}
```

Code Listing for Model6C2.PAS.

```
Procedure Generate_UnitHeapDataFiles;

{This procedure is used to generate the UnitHeapDataFiles.  The files are }
{named M6C1H1 through to M6C1HX.                                           }

var
   UnitHeapNoS,FileName                                    :String;
   HeapFile                                                :Text;

begin
   for UnitHeapIndex:=1 to NoUnitHeaps do
   begin
      Str(UnitHeapIndex,UnitHeapNoS);
      FileName:=Concat('M6C1H',UnitHeapNoS,'.Dat');
      assign(HeapFile,FileName);
      rewrite(HeapFile);
      for SizeIndex:=1 to M do
      begin
         for ArbIndex:=0 to N+1 do
         begin
            write(HeapFile,ART^[SizeIndex,ArbIndex],'    ');
            write(HeapFile,SRT1^[SizeIndex,ArbIndex],'    ');
            writeln(HeapFile,SRT2^[SizeIndex,ArbIndex]);
         end;
         writeln(HeapFile);
      end;

      writeln(HeapFile,UnitHeapConv1,'      ',UnitHeapConv2);
      writeln(HeapFile);
      writeln(HeapFile,'CumTime   ',CumTime);
      close(HeapFile);

   end;
end;

{-------------------------------------------------------------------------}
Procedure Read_ART_and_SRTs;

var
   UnitHeapNoS,FileName                                    :String;
   HeapFile                                                :Text;

begin
   Str(UnitHeapIndex,UnitHeapNoS);
   FileName:=Concat('M6C1H',UnitHeapNoS,'.Dat');
   assign(HeapFile,FileName);
   reset(HeapFile);
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to N do
      begin
         read(HeapFile,ART^[SizeIndex,ArbIndex]);
         read(HeapFile,SRT1^[SizeIndex,ArbIndex]);
         readln(HeapFile,SRT2^[SizeIndex,ArbIndex]);
      end;
      read(HeapFile,ArbValue);
      ART^[SizeIndex,N+1]:=TotHeapARTVec^[UnitHeapIndex];
      read(HeapFile,SRT1^[SizeIndex,N+1]);
      readln(HeapFile,SRT2^[SizeIndex,N+1]);
   end;
```



```
   readln(HeapFile,HoldConv1,HoldConv2);
   close(HeapFile);

end;

{-------------------------------------------------------------------------}
Procedure Write_ART_and_SRTs;

var
   UnitHeapNoS,FileName                                    :String;
   HeapFile                                                :Text;

begin
   Str(UnitHeapIndex,UnitHeapNoS);
   FileName:=Concat('M6C1H',UnitHeapNoS,'.Dat');
   assign(HeapFile,FileName);
   rewrite(HeapFile);
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to N+1 do
      begin
         if ART^[SizeIndex,ArbIndex]>FConc_Tol then
            write(HeapFile,ART^[SizeIndex,ArbIndex],'    ')
         else
            write(HeapFile,0.000,'    ');
         if SRT1^[SizeIndex,ArbIndex]>SConc_Tol then
            write(HeapFile,SRT1^[SizeIndex,ArbIndex],'    ')
         else
            write(HeapFile,0.000,'    ');
         if SRT2^[SizeIndex,ArbIndex]>SConc_Tol then
            writeln(HeapFile,SRT2^[SizeIndex,ArbIndex])
         else
            writeln(HeapFile,0.000);
      end;
      writeln(HeapFile);
   end;
   writeln(HeapFile,UnitHeapConv1,'      ',UnitHeapConv2);
   writeln(HeapFile);
   writeln(HeapFile,'CumTime    ',Cumtime);
   close(HeapFile);
   UnitHeapConv1:=0;
   UnitHeapConv2:=0;
end;

{-------------------------------------------------------------------------}
Procedure Determine_Maximum_DeltaT_fn_Size;

begin
   for UnitHeapIndex:=1 to NoUnitHeaps do
   begin
      for SizeIndex:=1 to M do
         DT_for_WatchSize^[UnitHeapIndex,SizeIndex]:=MaxDeltaT
              *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
              /SizeData[M,1]/SizeData[M,1];
   end;

   for UnitHeapIndex:=1 to NoUnitHeaps do WatchSizeDT[UnitHeapIndex]:=M;

end;
```

```
{-------------------------------------------------------------------------}
Procedure Determine_Corresponding_ColDeltaT;

{This procedure uses the method of characteristics to determine the      }
{progression in time for a given CharColDeltaE.  It allows for unsteady  }
{state flow.                                                             }

begin
   IntLength :=0;
   ColDelta_t:=0;
   while IntLength<>IntLengthCrit do begin
      if EOF(SupVelFile) then begin
         closesegraphics;
         writeln('Insufficient Data in the Superficial Velocity Data File to');
         writeln('execute the required number of iterations.');
         writeln('Present number of iterations:=  ',TimeCounter);
         Continue:=0;
         IntLength:=IntLengthCrit;
         readln;
      end;
      IntLength:=IntLength+(SupVelData[2,1]-SupVelData[1,1])*SupVelData[2,2];
      ColDelta_t:=ColDelta_t+(SupVelData[2,1]-SupVelData[1,1]);
      If IntLength>IntLengthCrit then begin
         ArbValue:=Intlength-IntLengthCrit;
         ColDelta_t:=ColDelta_t-ArbValue/SupVelData[2,2];
         IntLength:=IntLengthCrit;
         SupVelData[1,1]:=SupVelData[2,1]-ArbValue/SupVelData[2,2];
         SupVelData[1,2]:=SupVelData[2,2];
      end else begin
         SupVelData[1,1]:=SupVelData[2,1];
         SupVelData[1,2]:=SupVelData[2,2];
         read(SupVelFile, SupVelData[2,1]);
         read(SupVelFile, SupVelData[2,2])
      end;
   end;
   ColDeltaT:=ColDelta_t*uStar/ColLength;
   CumTime:=CumTime+ColDeltaT;
end;

{-------------------------------------------------------------------------}
Procedure Determine_Appropriate_Ref_Size_Class_DeltaT;

begin
   SizeIndex:=WatchSizeDT[UnitHeapIndex];
   Iterations:=INT(ColDeltaT/(DT_for_WatchSize^[UnitHeapIndex,SizeIndex]))
               +1;
   DeltaT:=ColDeltaT/Iterations;
   writeln(SizeIndex);
end;


{-------------------------------------------------------------------------}
Procedure Check_Whether_WatchSize_Still_Valid;

var
   Flag          :Integer;
   InterG1       :VeryLongVector;
   IntegVal      :Real;

begin
```



```
   SizeIndex:=WatchSizeDT[UnitHeapIndex];
   Flag:=0;
   if ART^[SizeIndex,N]<0.5*ART^[SizeIndex,(N+1)] then
   begin
      for ArbIndex:=0 to N do
      begin
         if (ART^[SizeIndex,ArbIndex]<0.8*ART^[SizeIndex,(ArbIndex+1)])
            or (ART^[SizeIndex,ArbIndex]>1.2*ART^[SizeIndex,(ArbIndex+1)])
            then Flag:=Flag+1;
      end;
   end;
   if Flag=0 then
   begin
      for ArbIndex:=0 to N+1 do InterG1[ArbIndex]:=ART^[SizeIndex,ArbIndex];
      IntegrateVector(InterG1,DeltaE,0,(N+1),IntegVal);
      for ArbIndex:=0 to N+1 do if (ART^[SizeIndex,ArbIndex]<(0.9*Integval))
          or (ART^[SizeIndex,ArbIndex]>(1.1*Integval)) then Flag:=1;
      if (Flag=0) and (WatchSizeDT[UnitHeapIndex]>1) then
          WatchSizeDT[UnitHeapIndex]:=WatchSizeDT[UnitHeapIndex]-1;
   end else if ART^[SizeIndex,N+1]>FConc_Tol then DT_for_WatchSize^[UnitHeapIndex
,SizeIndex]
      :=2*DT_for_WatchSize^[UnitHeapIndex,SizeIndex];

end;


{-------------------------------------------------------------------------}
Procedure UpDate_TotHeapConv;

begin
   TotHeapConv1^[ColIters]:=TotHeapConv1^[ColIters]
                            +GlobWetFac*UnitHeapConv1/NoUnitHeaps;
   TotHeapConv2^[ColIters]:=TotHeapConv2^[ColIters]
                            +GlobWetFac*UnitHeapConv2/NoUnitHeaps;
end;

{-------------------------------------------------------------------------}
Procedure UpDate_TotHeapVector;

begin
   TotHeapARTVec^[UnitHeapIndex]    :=ART^[1,N+1];
   TotHeapSolSRT1Vec^[UnitHeapIndex]:=TotHeapSolSRT1Vec^[UnitHeapIndex]+
                                      (UnitHeapConv1-HoldConv1);
   TotHeapSolSRT2Vec^[UnitHeapIndex]:=TotHeapSolSRT2Vec^[UnitHeapIndex]+
                                      (UnitHeapConv2-HoldConv2);
end;


{-------------------------------------------------------------------------}
Procedure Increment_TotHeapVectors_and_BTCVectors;

begin
   ARTBTC^[ColIters +1] :=TotHeapARTVec^[NoUnitHeaps];
   SRT1BTC^[ColIters +1]:=GlobWetFac*(TotHeapSolSRT1Vec^[NoUnitHeaps]);
   SRT2BTC^[ColIters +1]:=GlobWetFac*(TotHeapSolSRT2Vec^[NoUnitHeaps]);
   for ArbIndex:=NoUnitHeaps downto 2 do
   begin
      TotHeapArtVec^[ArbIndex]:=TotHeapArtVec^[ArbIndex-1];
      TotHeapSolSRT1Vec^[ArbIndex]:=TotHeapSolSRT1Vec^[ArbIndex-1];
      TotHeapSolSRT2Vec^[ArbIndex]:=TotHeapSolSRT2Vec^[ArbIndex-1];
```

```
   end;
   TotHeapArtVec^[1]:=1;
   TotHeapSolSRT1Vec^[1]:=0;
   TotHeapSolSRT2Vec^[1]:=0;

end;

{-------------------------------------------------------------------------}
Procedure Graph_Conversion_and_BTC_Curves;

begin
   initSEGraphics(ValidDir);
   SetCurrentWindow(2);
   BorderCurrentWindow(1);
   SetAxesType(0,0);
   ScalePlotArea(0.0,0.0,(TimeVector^[ColIters]),GfMax_Y);  (***)
   SetXYIntercepts(0.0,0.0);
   SetColor(2);
   DrawXAxis((TimeVector^[ColIters]/5),1);    (***)
   DrawYAxis((GfMax_Y/5),1);
   LabelXAxis(1,0);
   LabelYAxis(1,0);
   TitleXAxis('Dmlss Time (Dmlss Time=(time * uStar)/ColLength');
   TitleYAxis('Frac. Conv. and BTC Curves');
   TitleWindow('Model6C2');
   LabelGraphWindow(1,930,Glabel1,0,0);
   LabelGraphWindow(1,900,Glabel2,0,0);
   for ArbIndex:=0 to ColIters  do DataSetX^[ArbIndex]:=TimeVector^[ArbIndex];
   for ArbIndex:=0 to ColIters  do DataSetY^[ArbIndex]:=TotHeapConv1^[ArbIndex];
   LinePlotData(DataSetX^,DataSetY^,(ColIters +1),5,0);
   for ArbIndex:=0 to ColIters  do DataSetY^[ArbIndex]:=TotHeapConv2^[ArbIndex];
   LinePlotData(DataSetx^,DataSetY^,(ColIters +1),4,0);
   for ArbIndex:=0 to ColIters  do DataSetY^[ArbIndex]:=ARTBTC^[ArbIndex];
   LinePlotData(DataSetx^,DataSetY^,(ColIters +1),3,1);
   for ArbIndex:=0 to ColIters  do DataSetY^[ArbIndex]:=SRT1BTC^[ArbIndex];
   LinePlotData(DataSetx^,DataSetY^,(ColIters +1),5,1);
   for ArbIndex:=0 to ColIters  do DataSetY^[ArbIndex]:=SRT2BTC^[ArbIndex];
   LinePlotData(DataSetx^,DataSetY^,(ColIters +1),4,1);



end;


{=========================================================================}
{MAIN PROGRAM:                                                            }
{=========================================================================}
begin

   new(Beta1_k);          new(Beta2_k);
   new(KappaP1_k);        new(KappaP2_k);
   new(KappaS1_k);        new(KappaS2_k);        new(DeltaT_k);
   new(Lambda1_k);        new(Lambda2_k);
   new(ContRatioV1);      new(ContRatioV2);
   new(ART);              new(ARTP1);
   new(SRT1);             new(SRT1P1);           new(SRT1P1C);
   new(SRT2);             new(SRT2P1);           new(SRT2P1C);
   new(DataSetX);         new(DataSetY);
```



```
   new(AMatrix);          new(AInverse);
   new(ARTBTC);           new(SRT1BTC);          new(SRT2BTC);
   new(TotHeapARTVec);    new(TotHeapSolSRT1Vec); new(TotHeapSolSRT2Vec);
   new(TotHeapConv1);     new(TotHeapConv2);     new(TimeVector);
   new(DT_for_WatchSize);

   assign(ConvResults,'ConvResults.Dat');
   rewrite(ConvResults);
   close(ConvResults);

   Initialise_Various_Variables;

   Size_Distribution_Initialisation;
   Contaminant_Location_Initialisation;
   Determine_Model_Parameters_fn_Size;

   Initialise_Superficial_Velocity_File;

   Col_Initial_Conditions;
   Particle_Initial_Conditions;
   Generate_UnitHeapDataFiles;

   Determine_Maximum_DeltaT_fn_Size;

   while ColIters<SpecColIters do
   begin
      ColIters:=ColIters+1;
      Determine_Corresponding_ColDeltaT;
      TimeVector^[ColIters]:=CumTime;

      for UnitHeapIndex:=1 to NoUnitHeaps do
      begin
         Read_ART_and_SRTs;
         if ART^[1,N+1]>FConc_Tol then
         begin
            Closesegraphics;
            Determine_Appropriate_Ref_Size_Class_DeltaT;
            Determine_DeltaT_fn_Size;
            Execute_Unit6C2;
            Check_Whether_WatchSize_Still_Valid;
         end;
         UpDate_TotHeapConv;
         UpDate_TotHeapVector;
         Write_ART_and_SRTs;
      end;

      Increment_TotHeapVectors_and_BTCVectors;

      Assign(ConvResults,'ConvResults.Dat');
      Append(ConvResults);
      Writeln(ConvResults,TimeVector^[ColIters]:8:4,'   '
              ,TotHeapConv1^[ColIters]:8:4);
      Close(ConvResults);

   end;

   Graph_Conversion_and_BTC_Curves;

   readln(HardCopy);
   if HardCopy=1 then ScreenDump(3,0,2,1.5,1.5,0,1,0,error);
```

```
   closesegraphics;
end.
{=========================================================================}
```

```
Unit Unit6C2;

{Unit6C2.  * This program is similar to Model5E2.PAS.                    }
{          * The program calculates the concentration profile of fluid   }
{            and solid reactants within a particle using the equations as }
{            developed by Dixon.                                         }
{          * This program provides for variable solid reactant orders.   }

{          * Assumptions in this model include:                          }
{                - The solid reactant deposits within the particle       }
{                  resemble those on the surface.                        }
{                - The deposits would both react to the same extent      }
{                  if each were exposed to the same acid concentration   }
{                  for the same time.                                    }

{Coded:    Graham Davies.                                                }
{          Department of Chemical Engineering.                           }
{          University of Cape Town.                                      }
{          28 February 1995.                                             }
{          06 June     1995 Updated (GMD).                               }
{=======================================================================}
Interface

{=======================================================================}
{Declarations:                                                           }

uses crt,stdhdr,gj,graph,worlddr,segraph,integrat,M2C1BrentRoots;

const

   {***These are the parameters applicable to the reference size class.***}

   Beta1     = 0.264;  {Dimensionless Stoichiometric ratio defined pg 11  }
   KappaP1   = 4.5;    {Ratio of reation rate of solid reactant residing  }
                       {within the particle to porous diffusion of fluid  }
                       {into the particle. Defined pg 11 Dixon.           }
   OrdSP1    = 1.0;    {Reaction order of the solid in the pores.         }

   Beta2     = 0.0;
   KappaP2   = 0.0;
   OrdSP2    = 0.0;

   Voidage   = 0.01;   {Voidage of the solid partilces.(Porosity)         }


   M         =10;      {Number of size classes.                           }
   RefSizeCl =5;       {Defines the reference size class.                 }
   ViewSizeCl=5;       {The concentration profiles of this sizeclass are  }
                       {graphed in graph 1.  (NB  ViewSizeCl <= M.)       }


   {***These are numerical method parameters.*************************}

   N         =19;      {Half the number of interior points.  r=0 and r=R  }
                       {not included.                                     }
   DeltaE    =1/(N+1); {Space Increment.  Calculated from 1/(N+1). (Since }
                       {                  R=i*dE and R is at point N+1.)  }

   MaxDeltaT =0.001;   {Maximum permitted time increment for particles.   }
```



```
   Norm_Crit =1e-6;    {Convergence criteria based on the norm of vector  }
                       {***                                               }
   Conv_Crit =1e-8;    {Convergence criteria for the Brent Routine.       }
   SConc_Tol =1e-4;    {Dimensionless concentration of solid below which  }
   FConc_Tol =1e-4;    {Dimensionless concentration of fluid reactant     }
                       {below which it is assumed to be negigible.        }
                       {it is assumed to be negligible.                   }
   MaxIter   =100;     {Maximum iterations for the Brent Routine.         }

   Print_Crit=1;

   {***Other information required.*************************************}

   ValidDir  ='F:\TP6\';{Valid directory for graphics drivers.      }


   {*********************************************************************}

type
   array1= Array[1..50,1..2] of Double;

var

   TotVolPart,VolLiq,DeltaT,Iterations,CumTime                :Double;
   UnitHeapConv1,UnitHeapConv2                                :Double;
   ArbIndex,SizeIndex,error,UnitHeapIndex                     :Integer;
   Repeats,Plot_Var,Flag1,HardCopy                            :Integer;
   DataSetX,DataSetY                                          :^VeryLongVector;
   Beta1_k,Beta2_k,KappaP1_k,KappaP2_k,KappaS1_k,KappaS2_k    :^ShortVector;
   Lambda1_k,Lambda2_k,ContRatioV1,ContRatioV2                :^ShortVector;
   DeltaT_k                                                   :^Array1;
   ART,ARTP1,SRT1,SRT2,SRT1P1,SRT2P1,SRT1P1C,SRT2P1C          :^SqrMat;
   AMatrix,AInverse                                           :^SqrMat;
   ARTP1CVal,SumSizeData_2,NuStar                             :Extended;
   Lambda1,Lambda2,KappaS1,KappaS2                            :Extended;
   GLabel1,Glabel2,GLabel3                                    :String;

   SizeData : Array[1..50,1..2] of Real;

   {ART     : Alpha fn(r) at time T                    range 0..M  0..N+1}
   {ARTP1   : Alpha fn(r) at time T+1                  range 0..M  0..N+1}
   {SRT1    : Sigma fn(r) at time T                    range 0..M  0..N+1}
   {SRTsP1  : Sigma fn(r) at time T+1      (guessed)   range 0..M  0..N+1}
   {SRTsP1C : Sigma fn(r) at time T+1      (calculated) range 0..M  0..N+1}
   {YVect   : 'Const' vector in Crank-Nicolson method  range 0..N+1      }
   {AMatrix : Matrix of Crank-Nicolson coefficients    range N* N        }
   {DataSetX: X vector used in the graphing routine    range 0..N+1      }

   {*********************************************************************}

   Procedure Size_Distribution_Initialisation;
   Procedure Contaminant_Location_Initialisation;
   Procedure Determine_Model_Parameters_fn_Size;
   Procedure Determine_DeltaT_fn_Size;
   Procedure Particle_Initial_Conditions;
   Procedure Guess_SRTsP1;
   Procedure ReGuess_SRTsP1;
   Procedure Calc_Crank_Nicolson_Matrix(var YVector :ShortVector);
   Procedure Calc_ARTP1_and_SRTsP1C;
   Procedure Check_Convergence;
```

```
   Procedure Update_ART_and_SRTs;
   Procedure Calc_Conversion;
   Procedure Graph1_Initialise;
   Procedure Graph1_Results;
   Procedure Execute_Unit6C2;

{=====================================================================}
Implementation

{=====================================================================}
Function Power(Base,Pow:real):Extended;
begin
   if Pow=0 then Power:=1
   else
   if Base=0 then Power:=0
   else
   Power:=exp(Pow*ln(base));

end;


{---------------------------------------------------------------------}
Procedure Size_Distribution_Initialisation;

{This procedure sets up the size distribution data array. Initially the   }
{SizeData array contains radius information and fractional volume          }
{information (ie R and Vp/Vtot).  On output it contains  relative radius   }
{information and relative volume information (ie R/R_ref and Vp/Vp_ref).   }
{Ideally this information would be read in from a data file.               }

var
   RefRadius,VolRefPart :Double;

begin
   SizeData[1,1] := 37.00e-3/2;          SizeData[1,2] := 58.1/100;
   SizeData[2,1] := 31.25e-3/2;          SizeData[2,2] := 15.8/100;
   SizeData[3,1] := 22.00e-3/2;          SizeData[3,2] := 10.3/100;
   SizeData[4,1] := 16.10e-3/2;          SizeData[4,2] :=  4.1/100;
   SizeData[5,1] := 11.35e-3/2;          SizeData[5,2] :=  3.2/100;
   SizeData[6,1] :=  8.10e-3/2;          SizeData[6,2] :=  1.8/100;
   SizeData[7,1] :=  5.73e-3/2;          SizeData[7,2] :=  1.3/100;
   SizeData[8,1] :=  4.05e-3/2;          SizeData[8,2] :=  1.0/100;
   SizeData[9,1] :=  2.86e-3/2;          SizeData[9,2] :=  0.6/100;
   SizeData[10,1]:=  1.18e-3/2;          SizeData[10,2]:=  3.8/100;

   VolRefPart:=TotVolPart*SizeData[RefSizeCl,2];
   RefRadius :=SizeData[RefSizeCl,1];

   SumSizeData_2:=0;

   for SizeIndex:=1 to M do
   begin
      SizeData[SizeIndex,1]:=SizeData[SizeIndex,1]/RefRadius;
      SizeData[SizeIndex,2]:=SizeData[SizeIndex,2]*TotVolPart/VolRefPart;
      SumSizeData_2:=SumSizeData_2+SizeData[SizeIndex,2];
   end;

   NuStar:=VolLiq/(Voidage*VolRefPart);

   {Ratio of volume of bulk fluid to fluid in particle pores.             }
```



```
end;


{---------------------------------------------------------------------}
Procedure Contaminant_Location_Initialisation;

{This procedure is used to define the contaminant ratio vector (ratio of  }
{the surface contaminant concentration to bulk contaminant concentration).}
{NOTE: If any size class of particles have a surface concentration of      }
{      contaminant, then so too must the reference size class.             }


begin
   ContRatioV1^[1] :=0.00;          ContRatiov2^[1] :=0;
   ContRatioV1^[2] :=0.00;          ContRatiov2^[2] :=0;
   ContRatioV1^[3] :=0.00;          ContRatiov2^[3] :=0;
   ContRatioV1^[4] :=0.00;          ContRatiov2^[4] :=0;
   ContRatioV1^[5] :=0.00;          ContRatiov2^[5] :=0;
   ContRatioV1^[6] :=0.00;          ContRatiov2^[6] :=0;
   ContRatioV1^[7] :=0.00;          ContRatiov2^[7] :=0;
   ContRatioV1^[8] :=0.00;          ContRatiov2^[8] :=0;
   ContRatioV1^[9] :=0.00;          ContRatiov2^[9] :=0;
   ContRatioV1^[10]:=0.00;          ContRatiov2^[10]:=0;
end;

{---------------------------------------------------------------------}
Procedure Determine_Model_Parameters_fn_Size;

{This procedure determines the model parameters each size class of        }
{particles.  Note that the DeltaT is defined separately in another        }
{procedure.                                                               }

var
   Flag1,Flag2 :Integer;

begin
  Flag1:=0;
  Flag2:=0;
  for SizeIndex:=1 to M do
  begin
      Lambda1_k^[SizeIndex]:=ContRatioV1^[SizeIndex]
                             /(1+ContRatioV1^[SizeIndex]);
      Lambda2_k^[SizeIndex]:=ContRatioV2^[SizeIndex]
                             /(1+ContRatioV2^[SizeIndex]);
      if ContRatioV1^[SizeIndex]<>0 then Flag1:=Flag1+1;
      if ContRatioV2^[SizeIndex]<>0 then Flag2:=Flag2+1;
   end;

   if (((Flag1=1) and (ContRatioV1^[RefSizeCl]=0)) or
       ((Flag2=1) and (ContRatiov2^[RefSizeCl]=0))) then begin
      closesegraphics;
      writeln('Contaminant Location Violoation.');
      readln;
   end;
   Lambda1              :=Lambda1_k^[RefSizeCl];
   Lambda2              :=Lambda2_k^[RefSizeCl];
   KappaS1              :=Lambda1*KappaP1/(1-Lambda1);
   KappaS2              :=Lambda2*KappaP2/(1-Lambda2);
```

```
   for SizeIndex:=1 to M do
   begin
      Beta1_k^[SizeIndex]  :=Beta1*(1+ContRatioV1^[RefSizeCl])
                            /(1+ContRatioV1^[SizeIndex]);
      Beta2_k^[SizeIndex]  :=Beta2*(1+ContRatioV2^[RefSizeCl])
                            /(1+ContRatioV2^[SizeIndex]);
      KappaP1_k^[SizeIndex]:=KappaP1*SizeData[SizeIndex,1]
                            *SizeData[SizeIndex,1];
      KappaP2_k^[SizeIndex]:=KappaP2*SizeData[SizeIndex,1]
                            *SizeData[SizeIndex,1];
      if ContRatioV1^[SizeIndex]=0 then KappaS1_k^[SizeIndex]:=0 else
         KappaS1_k^[SizeIndex]:=KappaS1*Power((ContRatioV1^[SizeIndex]
                            /ContRatioV1^[RefSizeCl]),OrdSP1)
                            *SizeData[SizeIndex,1];
      if ContRatioV2^[SizeIndex]=0 then KappaS2_k^[SizeIndex]:=0 else
         KappaS2_k^[SizeIndex]:=KappaS2*Power((ContRatioV2^[SizeIndex]
                            /ContRatioV2^[RefSizeCl]),OrdSP2)
                            *SizeData[SizeIndex,1];

   end;
end;
{-------------------------------------------------------------------------}
Procedure Determine_DeltaT_fn_Size;

begin
   for SizeIndex:=1 to M do
      DeltaT_k^[SizeIndex,1] :=DeltaT/SizeData[SizeIndex,1]
                            /SizeData[SizeIndex,1];

end;

{-------------------------------------------------------------------------}
Procedure Particle_Initial_Conditions;

{This procedure uses the initial conditions to set the alpha and sigma    }
{vectors.                                                                 }
{Note that ART[N+1], ARTP1[N+1], SRT1[N+1] and SRT1P1[N+1] are the surface}
{concentrations of the liquid and solid reactants.                        }

begin
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to N do
      begin
         ART^[SizeIndex,ArbIndex] :=0;
         SRT1^[SizeIndex,ArbIndex]:=1;
         SRT2^[SizeIndex,ArbIndex]:=1;
         SRT1P1C^[SizeIndex,ArbIndex]:=0;
         SRT2P1C^[SizeIndex,ArbIndex]:=0;
      end;
      ART^[SizeIndex,N+1] :=1;
      SRT1^[SizeIndex,N+1]:=1;
      SRT2^[SizeIndex,N+1]:=1;
      SRT1P1C^[SizeIndex,ArbIndex]:=0;
      SRT2P1C^[SizeIndex,ArbIndex]:=0;
   end;
end;

{-------------------------------------------------------------------------}
Procedure Guess_SRTsP1;
```



```
{This procedure provides the initial "guess" for the iteration.  It uses  }
{the previous time interval's values as the guess.                        }

begin
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
      begin
         SRT1P1^[SizeIndex,ArbIndex]:=SRT1^[SizeIndex,ArbIndex];
         SRT2P1^[SizeIndex,ArbIndex]:=SRT2^[SizeIndex,ArbIndex];
      end;
      ARTP1^[SizeIndex,N+1]:=ART^[SizeIndex,N+1];
   end;
end;

{-------------------------------------------------------------------------}
Procedure ReGuess_SRTsP1;

{This procedure provides an updated "guess" for the next iteration. It    }
{uses the SRT1P1C vector as the updated guess.                            }

begin
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
      begin
         SRT1P1^[SizeIndex,ArbIndex]:=SRT1P1C^[SizeIndex,ArbIndex];
         SRT2P1^[SizeIndex,ArbIndex]:=SRT2P1C^[SizeIndex,ArbIndex];
      end;
      ARTP1^[SizeIndex,N+1]:=ARTP1CVal;
   end;
end;
{-------------------------------------------------------------------------}
Procedure Calc_Crank_Nicolson_Matrix(var YVector :ShortVector);

var
   Rows,Cols                                        :Integer;

begin
   for Rows:=0 to (N+1) do
   begin
      for Cols:=0 to (N+1) do AMatrix^[Rows,Cols]:=0;
   end;

   AMatrix^[0,0]:=(-6-DeltaE*DeltaE*SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                  *(KappaP1_k^[SizeIndex]*Power(SRT1P1^[SizeIndex,0],OrdSP1)
                  +KappaP2_k^[SizeIndex]*Power(SRT2P1^[SizeIndex,0],OrdSP2))
                  -2*DeltaE*DeltaE*SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                  /DeltaT_k^[SizeIndex,1]);
   AMatrix^[0,1]:=6;
   YVector[0]    :=ART^[SizeIndex,0]*(6+DeltaE*DeltaE*SizeData[SizeIndex,1]
                  *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]
                  *Power(SRT1^[SizeIndex,0],OrdSP1)+KappaP2_k^[SizeIndex]
                  *Power(SRT2^[SizeIndex,0],OrdSP2))
                  -2*DeltaE*DeltaE*SizeData[SizeIndex,1]
                  *SizeData[SizeIndex,1]/DeltaT_k^[SizeIndex,1])
                  -ART^[SizeIndex,1]*6;

   for Rows:=1 to N-1 do
```

```
   begin
      AMatrix^[Rows,Rows-1]:=Rows-1;
      AMatrix^[Rows,Rows]  :=-2*Rows-Rows*DeltaE*DeltaE*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]
                             *Power(SRT1P1^[SizeIndex,Rows],OrdSP1)
                             +KappaP2_k^[SizeIndex]
                             *Power(SRT2P1^[SizeIndex,Rows],OrdSP2))
                             -2*Rows*DeltaE*DeltaE*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1]/DeltaT_k^[SizeIndex,1];
      AMatrix^[Rows,Rows+1]:=Rows+1;
      YVector[Rows]        :=ART^[SizeIndex,Rows-1]*(-Rows+1)+
                             ART^[SizeIndex,Rows]*(2*Rows+Rows*DeltaE*DeltaE
                             *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                             *(KappaP1_k^[SizeIndex]
                             *Power(SRT1^[SizeIndex,Rows],OrdSP1)
                             +KappaP2_k^[SizeIndex]
                             *Power(SRT2^[SizeIndex,Rows],OrdSP2))
                             -2*Rows*DeltaE*DeltaE*SizeData[SizeIndex,1]
                             *SizeData[SizeIndex,1]/DeltaT_k^[SizeIndex,1])
                             +ART^[SizeIndex,Rows+1]*(-Rows-1);
   end;

   AMatrix^[N,N-1]:=N-1;
   AMatrix^[N,N]  :=-2*N-N*DeltaE*DeltaE*SizeData[SizeIndex,1]
                    *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]
                    *Power(SRT1P1^[SizeIndex,N],OrdSP1)+KappaP2_k^[SizeIndex]
                    *Power(SRT2P1^[SizeIndex,N],OrdSP2))-2*N*DeltaE*DeltaE
                    *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                    /DeltaT_k^[SizeIndex,1];
   YVector[N]     :=ART^[SizeIndex,N-1]*(-N+1)+ART^[SizeIndex,N]*(2*N+
                    N*DeltaE*DeltaE*SizeData[SizeIndex,1]
                    *SizeData[SizeIndex,1]*(KappaP1_k^[SizeIndex]
                    *Power(SRT1^[SizeIndex,N],OrdSP1)+KappaP2_k^[SizeIndex]
                    *Power(SRT2^[SizeIndex,N],OrdSP2))-2*N*DeltaE*DeltaE
                    *SizeData[SizeIndex,1]*SizeData[SizeIndex,1]
                    /DeltaT_k^[SizeIndex,1])+ART^[SizeIndex,N+1]*(-N-1)
                    -ARTP1^[SizeIndex,N+1]*(N+1);

end;

{-------------------------------------------------------------------------}
Procedure Calc_ARTP1_and_SRTsP1C;

{As it stands this procedure can cope with a variable reaction order. This}
{is due to the inclusion of the Brent Routine.                            }

var
   MassBalVal,AMatDet                          :Double;
   ValueAtRoot                                 :Real;
   OutputVector,YVector                        :ShortVector;

begin

   {First calculate the ARTP1 vectors.                                    }

   for SizeIndex:=1 to M do
   begin
      Calc_Crank_Nicolson_Matrix(YVector);
      GaussJordan(AMatrix^,YVector,(N+1),OutputVector,AInverse^,AMatDet);
      for ArbIndex:=0 to N do
```



```
         if OutPutVector[ArbIndex]>FConc_Tol then
            ARTP1^[SizeIndex,ArbIndex]:=OutputVector[ArbIndex]
         else
            ARTP1^[SizeIndex,ArbIndex]:=0;
   end;

   {The GaussJordan procedure calculates values for the ART vector from   }
   {point 0 to point N (although it makes use of the N+1_th point). To    }
   {determine the N+1_th point value, make use of the mass balance of     }
   {fluid reactant in the CSTR.                                           }


   MassBalVal:=0;

   for SizeIndex:=1 to M do
   begin
      MassBalVal:=MassBalVal-SizeData[SizeIndex,2]/2*(KappaS1_k^[SizeIndex]
                  *SRT1^[SizeIndex,(N+1)]*ART^[SizeIndex,(N+1)]
                  +KappaS1_k^[SizeIndex]*SRT1P1^[SizeIndex,(N+1)]
                  *ARTP1^[SizeIndex,(N+1)]+KappaS2_k^[SizeIndex]
                  *SRT2^[SizeIndex,(N+1)]*ART^[SizeIndex,(N+1)]
                  +KappaS2_k^[SizeIndex]*SRT2P1^[SizeIndex,(N+1)]
                  *ARTP1^[SizeIndex,(N+1)])-3*SizeData[SizeIndex,2]
                  /(SizeData[SizeIndex,1]*SizeData[SizeIndex,1])/2
                  *((ART^[SizeIndex,(N+1)]+ARTP1^[SizeIndex,(N+1)])
                  -(ART^[SizeIndex,N]+ARTP1^[SizeIndex,N]))/DeltaE;

   end;

   ARTP1CVal:=ART^[1,(N+1)]+DeltaT/NuStar*MassBalVal;

   {Calculate the SRTsP1C vectors.  Code makes use of Brent's Method      }
   {(A non-linear root finding procedure) to solve for the root.         }

   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do begin
         if (Beta1_k^[SizeIndex]<>0) and (KappaP1_k^[SizeIndex]<>0) then
         begin
            if OrdSP1=1 then
            begin
               SRT1P1C^[SizeIndex,ArbIndex]:=SRT1^[SizeIndex,ArbIndex]
                       *(1-DeltaT_k^[SizeIndex,1]*KappaP1_k^[SizeIndex]
                       *Beta1_k^[SizeIndex]*ART^[SizeIndex,ArbIndex]
                       /(2*(1-Lambda1_k^[SizeIndex])))/(1+DeltaT_k^[SizeIndex,1]
                       *KappaP1_k^[SizeIndex]*Beta1_k^[SizeIndex]
                       *ARTP1^[SizeIndex,ArbIndex]/(2*(1-Lambda1_k^[SizeIndex])))
;
            end
            else if SRT1P1^[SizeIndex,ArbIndex]<SConc_Tol then
               SRT1P1C^[SizeIndex,ArbIndex]:=SRT1P1^[SizeIndex,ArbIndex]
            else
            begin
               SRT1P1C^[SizeIndex,ArbIndex]:=BrentRoots(0.0,1.0,DeltaT_k^[SizeInd
ex,1],
                       KappaP1_k^[SizeIndex],Beta1_k^[SizeIndex],
                       Lambda1_k^[SizeIndex],OrdSP1,ART^[SizeIndex,ArbIndex],
                       ARTP1^[SizeIndex,ArbIndex],SRT1^[SizeIndex,ArbIndex],
                       1e-8,100,ValueAtRoot,error);
            end;
```

```
        end;
        if (Beta2_k^[SizeIndex]<>0) and (KappaP2_k^[SizeIndex]<>0) then
        begin
           if OrdSP2=1 then
           begin
              SRT2P1C^[SizeIndex,ArbIndex]:=SRT2^[SizeIndex,ArbIndex]
                     *(1-DeltaT_k^[SizeIndex,1]*KappaP2_k^[SizeIndex]
                     *Beta2_k^[SizeIndex]*ART^[SizeIndex,ArbIndex]
                     /(2*(1-Lambda2_k^[SizeIndex])))/(1+DeltaT_k^[SizeIndex,1]
                     *KappaP2_k^[SizeIndex]*Beta2_k^[SizeIndex]
                     *ARTP1^[SizeIndex,ArbIndex]/(2*(1-Lambda2_k^[SizeIndex])))
;
           end
           else if SRT2P1^[SizeIndex,ArbIndex]<SConc_Tol then
              SRT1P1C^[SizeIndex,ArbIndex]:=SRT2P1^[SizeIndex,ArbIndex]
           else
           begin
              SRT2P1C^[SizeIndex,ArbIndex]:=BrentRoots(0.0,1.0,
                     DeltaT_k^[SizeIndex,1],KappaP2_k^[SizeIndex],
                     Beta2_k^[SizeIndex],Lambda2_k^[SizeIndex],OrdSP2,
                     ART^[SizeIndex,ArbIndex],ARTP1^[SizeIndex,ArbIndex],
                     SRT2^[SizeIndex,ArbIndex],1e-8,100,ValueAtRoot,error);
           end;
        end;
     end;
  end;
end;


{-------------------------------------------------------------------------}
Procedure Check_Convergence;

{This procedure checks whether or not the solution has converged by       }
{comparing the guessed value of SRT1P1 with a calculated value of SRT1P1. }
{Also compares the calculated value of ARTP1C with guessed value of       }
{ARTP1.                                                                   }

var
   Norm1,Norm2,Norm3                                          :Double;

begin
   Norm1:=0;         {Using a norm ***                                    }
   Norm2:=0;
   Norm3:=0;
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
      begin
         if (Beta1_k^[SizeIndex]<>0) and (KappaP1_k^[SizeIndex]<>0) then begin
            Norm1:=Norm1+(SRT1P1C^[SizeIndex,ArbIndex]
                   -SRT1P1^[SizeIndex,ArbIndex])*(SRT1P1C^[SizeIndex,ArbIndex]
                   -SRT1P1^[SizeIndex,ArbIndex]);
         end
         else Norm1:=0;
         if (Beta2_k^[SizeIndex]<>0) and (KappaP2_k^[SizeIndex]<>0) then begin
            Norm2:=Norm2+(SRT2P1C^[SizeIndex,ArbIndex]
                   -SRT2P1^[SizeIndex,ArbIndex])*(SRT1P1C^[SizeIndex,ArbIndex]
                   -SRT2P1^[SizeIndex,ArbIndex]);
         end
         else Norm2:=0;
```



```
      end;
   end;
   Norm3:=(ARTP1CVal-ARTP1^[1,N+1])*(ARTP1CVal-ARTP1^[1,N+1]);
   if (Norm1<Norm_Crit) and (Norm2<Norm_Crit) and (Norm3<Norm_Crit)
      then Flag1:=1;
end;

{-------------------------------------------------------------------------}
Procedure Update_ART_and_SRTs;

{This procedure updates the alpha and sigma vectors for the next iteration}
{by replacing their components with the alphaT+1 and sigmaT+1 vectors.    }

begin
   for SizeIndex:=1 to M do
   begin
      for ArbIndex:=0 to (N+1) do
      begin
         ART^[SizeIndex,ArbIndex] :=ARTP1^[SizeIndex,Arbindex];
         SRT1^[SizeIndex,ArbIndex]:=SRT1P1^[SizeIndex,ArbIndex];
         SRT2^[SizeIndex,ArbIndex]:=SRT2P1^[SizeIndex,ArbIndex];
      end;
   end;
end;


{-------------------------------------------------------------------------}
Procedure Calc_Conversion;

{This procedure calculates the fractional conversion of the particle for  }
{the time interval.  It uses the formula on pg. 13 Dixon.  The integrator }
{is the Quinn-Curtis vector integrator.                                   }

var
   InterG1,InterG2                                            :VeryLongVector;
   IntegVal,Conversion                                        :Real;

begin
   for SizeIndex:=1 to M do
   begin
      if KappaP1_k^[SizeIndex]<>0 then
      begin
         for ArbIndex:=0 to (N+1) do
            InterG1[ArbIndex]:=(1-SRT1^[SizeIndex,ArbIndex])*ArbIndex*DeltaE
                               *ArbIndex*DeltaE;
         IntegrateVector(InterG1,DeltaE,0,(N+1),IntegVal);
         Conversion      :=3*(1-Lambda1_k^[SizeIndex])*IntegVal
                           +Lambda1_k^[SizeIndex]*(1-SRT1^[SizeIndex,N+1]);
         UnitHeapConv1:=UnitHeapConv1+Conversion*SizeData[SizeIndex,2]
                           /SumSizeData_2;
      end;

      if KappaP2_k^[SizeIndex]<>0 then
      begin
         for ArbIndex:=0 to (N+1) do
         InterG2[ArbIndex]:=(1-SRT2^[SizeIndex,ArbIndex])*ArbIndex*DeltaE
                            *ArbIndex*DeltaE;
         IntegrateVector(InterG2,DeltaE,0,(N+1),IntegVal);
         Conversion      :=3*(1-Lambda2_k^[SizeIndex])*IntegVal
                           +Lambda2_k^[SizeIndex]*(1-SRT2^[SizeIndex,N+1]);
```

```
        UnitHeapConv2:=UnitHeapConv2+Conversion*SizeData[SizeIndex,2]
                       /SumSizeData_2;
      end;
   end;
end;

{-------------------------------------------------------------------------}
Procedure Graph1_Initialise;

var
   GraphType                                          :Integer;
   Beta1S,KappaP1S,Lambda1S,GDTS                      :String;
   Beta2S,KappaP2S,Lambda2S                           :String;
   OrdSP1S,OrdSP2S                                    :String;
   CumTimeS,UnitHeapNoS,ViewSizeClS                   :String;

begin
   initSEGraphics(ValidDir);
   SetCurrentWindow(3);
   BorderCurrentWindow(1);
   SetAxesType(0,0);
   ScalePlotArea(0.0,0.0,1.0,1.2);
   SetXYIntercepts(0.0,0.0);
   SetColor(2);
   DrawXAxis(0.2,1);
   DrawYAxis(0.2,1);
   LabelXAxis(1,0);
   LabelYAxis(1,0);
   TitleXAxis('Dimensionless Radius');
   TitleYAxis('Dmlss Conc');
   TitleWindow('Unit6C2');
   Str(Beta1:6:3,Beta1S);
   Str(KappaP1:6:3,KappaP1S);
   Str((DeltaT*Print_Crit):6:4,GDTS);
   Str(Lambda1:5:3,Lambda1S);
   Str(OrdSP1:5:2,OrdSP1S);
   Str(Beta2:6:3,Beta2S);
   Str(KappaP2:6:3,KappaP2S);
   Str(Lambda2:5:3,Lambda2S);
   Str(OrdSP2:5:2,OrdSP2S);
   Str(CumTime:8:4,CumTimeS);
   Str(UnitHeapIndex:2,UnitHeapNoS);
   Str(ViewSizeCl:2,ViewSizeClS);
   GLabel1:=Concat(' Beta1 ',Beta1S,'; Kappa1  ',KappaP1S,';  Lambda1  ',
                   Lambda1S,'; Order1 ',OrdSP1S,';  GDT  ',GDTS);
   GLabel2:=Concat(' Beta2 ',Beta2S,'; Kappa2  ',KappaP2S,';  Lambda2  ',
                   Lambda2S,'; Order2 ',OrdSP2S);
   GLabel3:=Concat(' CumTime ',CumTimeS,'     Profiles of unit heap ',
                   UnitHeapNoS,' and sizeclass  ',ViewSizeClS);
   LabelGraphWindow(1,900,GLabel1,0,0);
   LabelGraphWindow(1,850,GLabel2,0,0);
   LabelGraphWindow(160,800,GLabel3,0,0);
   for ArbIndex:=1 to 2 do
   begin
      if ArbIndex=1 then GraphType:=9 else GraphType:=10;
      SetCurrentWindow(GraphType);
      BorderCurrentWindow(1);
      SetAxesType(0,0);
      ScalePlotArea(0.0,0.0,1.0,1.2);
      SetXYIntercepts(0.0,0.0);
```



```
      SetColor(2);
      DrawXAxis(0.2,1);
      DrawYAxis(0.2,1);
      LabelXAxis(1,0);
      LabelYAxis(1,0);
      TitleXAxis('Dimensionless Radius');
      TitleYAxis('Dmlss Conc');
      TitleWindow('Unit6C2');
      if ArbIndex=1 then LabelGraphWindow(200,900,'Smallest Size Fraction',0,0)
      else LabelGraphWindow(200,900,'Largest Size Fraction',0,0);
   end;

end;

{-------------------------------------------------------------------------}
Procedure Graph1_Results;

var
   GraphIndex      :Integer;

begin
   if ViewSizeCl>M then
   begin
      Closesegraphics;
      writeln('Graph 1 Draw ERROR');
      writeln('You have instucted the Graphing Routine to graph the conversion');

      writeln('curves of a size class which does not exist.');
      writeln('Respecify = ViewSizeClass in Declarations section');
      readln;
   end;
   for GraphIndex:=1 to 3 do
   begin
      if GraphIndex=1 then
      begin
         SizeIndex:=ViewSizeCl;
         SetCurrentWindow(3);
      end else if GraphIndex=2 then
      begin
         SizeIndex:=M;
         SetCurrentWindow(9);
      end else
      begin
         SizeIndex:=1;
         SetCurrentWindow(10);
      end;
      for ArbIndex:=0 to (N+1) do DataSetX^[ArbIndex]:=ArbIndex*DeltaE;
      if KappaP1<>0 then begin
         for ArbIndex:=0 to (N+1) do
            DataSetY^[ArbIndex]:=ART^[SizeIndex,ArbIndex];
         LinePlotData(DataSetX^,DataSetY^,(N+2),3,0);
         for ArbIndex:=0 to (N+1) do
            DataSetY^[ArbIndex]:=SRT1^[SizeIndex,ArbIndex];
         LinePlotData(DataSetX^,DataSetY^,(N+2),5,0);
      end;
      if KappaP2<>0 then begin
         for ArbIndex:=0 to (N+1) do
            DataSetY^[ArbIndex]:=ART^[SizeIndex,ArbIndex];
         LinePlotData(DataSetX^,DataSetY^,(N+2),3,0);
         for ArbIndex:=0 to (N+1) do
```

```
          DataSetY^[ArbIndex]:=SRT2^[SizeIndex,ArbIndex];
        LinePlotData(DataSetX^,DataSetY^,(N+2),4,2);
      end;
   end;
end;


{=============================================================================}
Procedure Execute_Unit6C2;

begin

   clrscr;

   Graph1_Initialise;

   Repeats:=0;
   Flag1  :=0;
   Plot_Var:=0;

   Guess_SRTsP1;

   Graph1_Results;


   while (Repeats<Iterations) and (ART^[1,N+1]>FConc_Tol) do
   begin
      while (Flag1=0) do
      begin
         Calc_ARTP1_and_SRTsP1C;
         Check_Convergence;
         ReGuess_SRTsP1;
      end;

      Plot_Var:=Plot_Var+1;

      if Plot_Var=Print_Crit then
      begin
         Plot_Var:=0;
         Graph1_Results;
      end;

      Update_ART_and_SRTs;
      Guess_SRTsP1;
      Flag1:=0;
      Repeats:=Repeats+1;

   end;

   Graph1_Results;
   Calc_Conversion;

end;

end.

{=============================================================================}
```